COMMODORE 64®

KAIKKI KUUSNELOSESTA

TIM ONOSKO

AMERSOFT

Tim Onosko

KAIKKI KUUSNELOSESTA

COMMODORE 64®

AMERSOFT

Englanninkielisestä alkuteoksesta
Commodore 64, getting the most from it

Suomentaneet
Veikko Rekunen
Tuomo Hakala

Kansi
Matti Savimaa

Original English language edition published by
Prentice-Hall International

Copyright © 1983 by Robert J. Brady Company
All Rights Reserved

© Amersoft/WEILIN + GÖÖS

Amer-yhtymä Oy Weilin + Göösin kirjapaino
Espoo 1984

ISBN 951-35-3230-5

Sisällys

JOHDANTO 7
1 ALKUTOIMET 11
Tietokoneen asentaminen 12
Toimintatarkistus 14
Näppäimistö 21
QWERTY-näppäimistö 28
2 PERUSTAIDOT 31
RETURN-näppäin 31
Ohjelman lataus 32
Lataaminen nauhalta 33
Ohjelman lataus ei onnistu 36
Tallentaminen nauhalle 38
Tallennuksen tarkistus 39
Ohjelman suorittaminen 40
Commodoren levykeasema 41
Levykeaseman käyttö 43
Levykkeen hakemisto 46
Levykkeen valmistelu 48
Lataaminen levykkeeltä 50
Tallentaminen ja varmistaminen 52
Levyketiedostojen poistaminen 53
Levykkeen järjestäminen 54
Levykevirheet 54

Huomautuksia levykeaseman käytöstä 55
Commodoren ohjelmayhteensopivuus 57
Tietokoneen käyttö laskimena 58
Kuinka paljon muistia on jäljellä? 60
Varo lainausmerkkejä 61
Paljonko kello on? 62
3 JOHDATUS OHJELMOINTIIN 65
BASIC-kieli 66
Mikä on ohjelma? 68
Numerointi ja listaaminen 69
The END 71
Muuttujat: kuinka tietokone pitää tiedot järjestyksessä? 72
Välimerkkien käyttö BASICissä 78
PRINT-käskyn käyttö 80
4 OHJELMOINNIN KYMMENEN TÄRKEINTÄ SÄÄNTÖÄ 93
INPUT-käsky pyytää tietoja 94
GOTO eli mene jonnekin 97
IF, THEN ja OR-käskyt eli ehtojen asettaminen 98
Käskyllä GET annat lisää tietoa 102
FOR ja NEXT tekevät silmukoita 105
GOSUB JA RETURN ovat aliohjelmakäskyjä 110
REM-käskyä käytetään huomautuksiin 113
5 OHJELMOINTI — KUINKA TIETOKONE VARASTOI TIETOA 117
Uudenlaisia muuttujia 118
DIM-käsky antaa uusia ulottuvuuksia 121
READ-käskyllä luetaan DATAa 125
Taulukoiden käyttö levyke- ja nauhatiedostoina 129
Postituslistaohjelman tekeminen 144
Ohjelman suunnittelu 145
Ohjelman laadinnassa huomattavia seikkoja 146
Ohjelmoinnin aloittaminen 147
Ohjelman käyttö 149
Tietokantaohjelman valitseminen 151
6 OHJELMOINTI — LISÄÄ BASIC-KÄSKYJÄ 163
PEEK ja POKE 163
SYS ja USR 165
CMD 166
SPC ja TAB 167
STOP ja WAIT 168
ON 169

Matemaattiset funktiot 171
Merkkijonojen käsittely 173
LEN, VAL ja STRS 173
ASC ja CHRS 176
LEFTS, MIDS ja RIGHTS 179
7 VÄRIT, GRAFIIKKA, ÄÄNI JA PELIT 183
VIC ja SID 184
Satunnaisia RND-huomautuksia 187
Ruudun ja rajauksen värit 192
Merkkien väri 193
Grafiikan piirtäminen 195
Digitaalinen noppa 199
Sprite-grafiikka 204
Vanha pähkinänkuoripeli 205
SID-ääni 211
Hupia joystickistä 218
Mitä on jäljellä? 220
LIITE 1: TUTKIMASSA KUUSNELOSTA Jim Butterfield .. 221
Miksi peukaloida koneen sisuksia? 221
Tietokoneen muisti 222
Muistin tyypit 222
Muistikartat 224
Tutkimusmatkalla RAMissa 225
Suuri muistinsiirto 227
Yhteenveto 228
LIITE 2: KUUSNELOSEN GRAFIIKKAA TUTKIMASSA
Paul F. Schatz 245
Grafiikkaa pikseleillä 246
Miten kuvat luodaan? 249
Kuinka VICille puhutaan? 250
Sprite-grafiikka 252
Yhteenveto sprite-rekisteristä 262
Ohjelmoivat merkit 267
Tarkkuusgrafiikka 277
Loppuhuomioita 287
LIITE 3: ÄÄNTÄ JA MUSIIKKIA TUTKIMASSA Frank H.
Covitz 291
Ääniteorian perusteet 292
SID-piirin osoitteet 296
Äänen dynamiikasta 299
Hyvin viritetty tietokone 309

Ääniä hyperavaruudesta 320
LIITE 4: VIRHEILMOITUKSET 325
LIITE 5: ASCII/CHRS-KOODIT JA MUUNNOSTAULUKKO 333
SANASTO 339
HAKEMISTO 346

Johdanto

Jos luet juuri tätä kirjaa, olet jo ostanut tietokoneen tai aiot ostaa sen taikka ainakin harkitset asiaa. Tämän kirjan tarkoituksena on antaa sinulle riittävästi tietoa Commodore 64 -tietokoneesta, jotta saisit siitä kaiken mahdollisen hyödyn.

Ensiksi on huomautettava, että tässä kirjassa käsitellään ainoastaan Commodore 64:ää ja sen kannettavaa versiota. Lukuun ottamatta eräitä useimmille tietokoneille yhteisiä perusasioita ja viittauksia muihin Commodoren valmistamiin tietokoneisiin — etenkin VIC-20:een — et opi kovinkaan paljon siitä, kuinka käytetään Applea, TRS-80:aa tai Ataria — tai muitakaan kotitietokoneita.

Jos olet jo ollut tekemisissä henkilökohtaisten tietokoneiden kanssa alusta alkaen on tässä sinullekin jotakin. Commodore 64:ään on suunniteltu monta mielenkiintoista, tehokasta ja *ainutlaatuista* piirrettä.

Luultavimmin olet kuitenkin joko juuri ostanut Commodore 64:n tai voit käyttää sitä työpaikallasi tai koulussa. Et ehkä ole oikein perillä tietokoneesta. Arvelet ehkä, ettet osaa tarpeeksi matematiikkaa pystyäksesi käyttämään sitä tai että sinulta puuttuu tarvittavaa ohjelmointitaitoa. Ohjelmointi saattaa tuntua monimutkaiselta ja ehkä pelottavaltakin.

Huolimatta kymmenen viime vuoden aikana tapahtuneesta edistymisestä tietokoneet tuntuvat usein ”kylmiltä” koneilta (vaikka valmistajat kuinka kutsuisivat niitä ”käyttäjäystävällisiksi”). Saatat ajatella, että tietokone ei pidä sinusta sen kummemmin kuin muistakaan sitä lähestyvistä.

Unohda epäluulosi. Tietokoneet ovat hämmästyttäviä ja kiehtovia koneita, mutta eivät oikeastaan juuri sen monimutkaisempia kuin autotkaan; niiden käyttämiseen tarvitset suurin piirtein yhtä paljon taitoa — ehkä vähemmänkin. Ohjelmointiin ei välttämättä tarvita monimutkaista matematiikkaa. Aluksi täytyy vain miettiä miten itse ratkaisisit ongelman. Kun se on selvillä, tulee ratkaisu jakaa osiin, jotka sitten muutetaan tietokoneen ymmärtämään muotoon. Ja matematiikasta puheen ollen, monet kotitietokoneiden käyttäjät, jotka olivat koulussa surkeita matematiikassa, huomaavat pian, että he ovat aika hyviä ratkomaan käytännön ongelmia. Laskemisen harjoittelu luokkahuoneessa vain matematiikan itsensä vuoksi ei aina ole kaikkein hauskinta puuhaa. Mutta tietokonetta ohjelmoitaessa on mahdollista nähdä, miten numerot tekevät työtä puolestasi.

Vaikka ohjelmointi onkin omalla tavallaan kiehtovaa, eräät tietokoneiden käyttäjät ohjelmoivat itse hyvin harvoin. Sen sijaan he käyttävät erilaisia myytäviä ohjelmia. Ohjelman käytön oppimiseen ei yleensä tarvita niin paljoa taitoa kuin sen kirjoittamiseen. Ohjelma vain ladataan tietokoneeseen ja sitten seurataan ohjeita. Mutta ole varovainen. Monissa tehokkaissa ohjelmissa vaaditaan, että opit myös käyttämään niiden ominaisuuksia täydellisesti, ja se vie melkein yhtä paljon työtä ja keskittymiskykyä kuin ohjelmoimisen oppiminen.

Vaikka et tarvitsisi tietokonetta ratkomaan ongelmia tai helpottamaan päivittäistä elämääsi, voit nauttia siitä leluna; se luo musiikkia, kuvia ja pystyy huvittamaan sinua tuntikaupalla. Sen kanssa on hauska leikkiä silloin, kun kenestäkään muusta ei ole seuraa. Tietokoneita voi käyttää niin leikkiin kuin työhönkin.

Hyvä syy hankkia ja käyttää tietokonetta on sekin, että haluaa vain oppia käyttämään sitä. Voit käydä esimerkiksi jonkin

kurssin tai ehkä saat tietoja koulussa tai työssäsi. Oman koneen hankkiminen ja käyttäminen saattaa olla elämäsi paras oppimiskokemus.

Monet ovat ymmällään tai pelkäävät tietokoneita siksi, että näiden ”ajattelevien” koneiden ympärille on kehittynyt kokonaan oma kieli. Useimmiten tietokonekieli on vain ”teknistä höpötystä”. Mutta tietyt tietokonesanat eivät ole pelkkää ammattislangia, sillä ne kuvaavat sitä, mistä tietokoneet on tehty ja kuinka ne toimivat. Nämä sanat löytyvät tästä kirjasta (ja sen lopussa olevasta hakemistosta). Sen sijaan mukana ei ole ammattislangia ammattislangin vuoksi. Tämä kirja ei opeta sinua puhumaan ”bittinikkarin” lailla, mutta se saattaa selvittää sinulle, mistä nämä ihmiset puhuvat.

Tässä kirjassa on hiukan tietoa tavallisimmista tietokoneen sovellutuksista. Niiden perusteella voit päätellä, helpottaako tietokone työtäsi.

Teos jakautuu kahteen pääosaan. Ensimmäisessä käsitellään tietokonetta ja sen käyttöä. Siihen sisältyy oppitunteja BASIC-kielisestä ohjelmoinnista ja vähän tietoa sovellutuksistakin. Kirjan lopussa on kokeneiden Commodoren käyttäjien kirjoittama osuus edistyneimmille, joka selvittelee tietokoneen toimintaperiaatteita sekä tietokonegrafiikkaa, -musiikkia ja -ääntä.

Tulee kuitenkin muistaa, että tämä kirja sen paremmin kuin mikään muukaan ei kerro *kaikkea* tietokoneista. Jos sellainen kirja olisi, ei muita enää tarvittaisi. Sen sijaan tämä kirja on — hiukan voileipäpöydän tapaan — tarkoitettu moneen makuun.

Kun luet tätä kirjaa ja ryhdyt käyttämään tietokonettasi, pidä muutama perusasia mielessä.

* Ole kärsivällinen. Älä turhaudu. *Jokainen* voi oppia käyttämään tietokonetta. Älä anna periksi liian pian.

* Älä odota, että tietokone ryhtyy heti auttamaan sinua. Sinun on ensiksi opittava pyytämään siltä apua.

* Seuraa ohjeita tarkasti. Toimit koneen kanssa, joka on looginen ja ottaa kaiken kirjaimellisesti.

* Mikäli mahdollista, helpota opiskeluasi tutkimalla tietokonetta jonkun toisen kanssa. Näin autatte toisianne ja ymmärrätte uudet asiat nopeammin.

* Ja muista, että saat tietokoneesta ulos sen minkä kirjoitat sisään.

Ennen kaikkea yritä pitää tietokoneinnostuksesi tasapainossa muun elämäsi kanssa. Ehkäpä kaikkein ''taianomaisinta'' kotitietokoneessa on se, kuinka kone saattaa viedä aikaasi. On helppoa uppoutua työskentelemään koneen kanssa tai jäädä miettimään ongelman ratkaisua. Mutta oppiminen ei suju silloin, kun on viettänyt tuntikausia jonkin ongelman parissa. Jos et tunnu edistyvän, katkaise virta ja palaa asiaan myöhemmin. Mahdollisesti uusi aloitus ja uusi lähestymistapa riittävät.

Ihannetapauksessa tietokoneen tulisi *säästää*, ei tuhlata aikaasi. Sen tarkoitus on helpottaa elämääsi, ei hankaloittaa sitä.

1

Alkutoimet

Tietokoneiden historia alkaa vuodesta 1953, jolloin niitä kutsuttiin ”elektronisiksi aivoiksi”. Mutta tietokoneella ja aivoilla on varsin vähän yhtäläisyyksiä. Tietokoneet ovat koneita, jotka eivät yksin tee yhtään mitään. Ne tarvitsevat ihmistä jolta ottaa vastaan käskyjä. Ne liikuttelevat numeroita muistissaan ja tekevät alkeellisia päätöksiä — siinä kaikki.

Niin sanotut henkilökohtaiset tietokoneet eli kotitietokoneet tulivat lähes jokaisen käyttäjän ulottuville 1970-luvun puolivälissä, jolloin Apple-niminen yhtiö rakensi tietokoneen yhdelle ainoalle piirilevylle. Samoihin aikoihin syntyi Commodore PET. Kumpikin näistä käytti samaa 6502-mikroprosessoria, jonka eräs versio saavutti suosiota myös Atarin videopelin prosessorina.

Commodore PET -tietokoneeseen perustuu myös Commodore 64. Sen mikroprosessorina on uusi versio 6502-mikroprosessorista, joka kykenee hallitsemaan laitteen sisäistä maailmaa. Commodore 64 pystyy ”pudottamaan” BASIC-kielen pois, jolloin esimerkiksi moduuleille jää enemmän tilaa keskusmuistista. Tätä järjestelmää ovat muut valmistajat sittemmin ryhtyneet matkimaan.

Muistitilan ja rakenteen uudistumisen lisäksi Commodore 64:ssä on erittäin hyvät grafiikka- ja ääniominaisuudet. Sen liitännät ovat samoin erinomaiset.

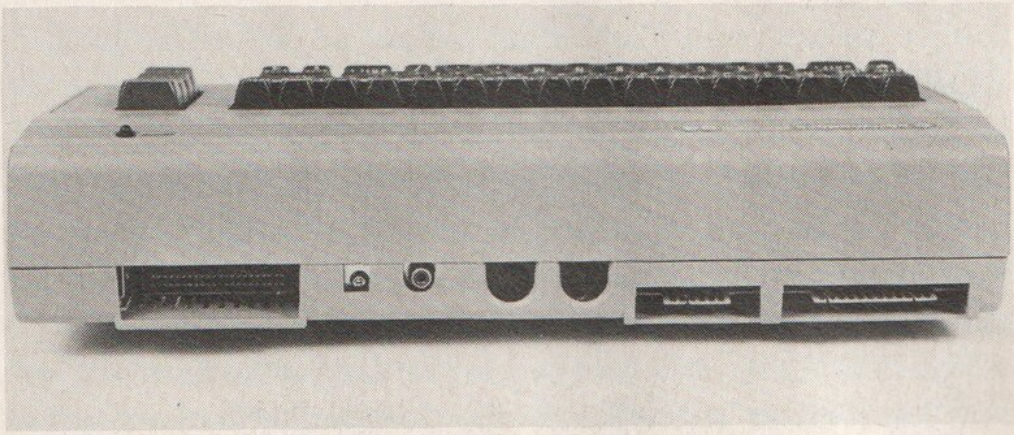
Tietokoneen asentaminen

Olet varmasti jo malttamattomana ottanut tietokoneen kuljetuslaatikosta. Vaikka käsikirjassa lukeekin, että "ohjeet on luettava huolella", monet ihmiset eivät innostuksissaan siitä välitä. Älä kuitenkaan anna intosi estää sinua tutkimasta, onko kaikki kunnossa. Tarkista, että kaikki käsikirjassa mainitut osat ovat mukana. Tarkasta myös huolellisesti, että mikään osa ei ole rikki. Säröt ja naarmut voivat merkitä sitä, että laite on vahingossa pudonnut, mistä on saattanut seurata näkymättömiä sisäisiä vaurioita. Jos jotain epäilyttävää ilmenee, ota yhteys laitteen myyjään.

Käsikirjassa annetaan tietokoneen asentamisohjeet, joten lue käsikirja huolellisesti. Varaa kylliksi tilaa muille tarvittaville laitteille, kuten televisiolle tai monitorille, käsikirjoille ja tarvikkeille. Älä myöskään unohda valaistusta; ylhäältä tuleva valo on yleensä paras. Televisio/monitori tulisi sijoittaa siten, ettei heijastumia näy ruudussa.

Jos käytät monitorina televisiota, on paras vaihtoehto uusi ja huokea televisio. Käytetyn ostaminen on onnenkauppaa, ja tavalliseen katseluun käytettävää tv-vastaanotinta on useimmiten vaikea käyttää tietokoneen monitorina. Ongelmia saattavat aiheuttaa tietokoneen itsensä synnyttämä videosignaali sekä vanhalle väritelevisiolle ominainen värien sekoittuminen. Mustavalkoinen televisio toimii yleensä luotettavasti myös tietokoneen näyttöpäätteenä jos värien käyttö ei ole tärkeää. Muussa tapauksessa kelvollinen nykyaikainen väritelevisio on hyvä ratkaisu, joskin nimenomaan tietokonekäyttöön rakennettu monitori antaa paremman kuvan. Niin sanottua RGB-monitoria, jonka kuvan laatu on erinomainen, ei Commodoressa (kuten ei useimmissa muissakaan saman hintaluokan laitteissa) voi käyttää.

Käsikirjassa esitellään koneen osat joten lue se tarkoin. Commodore 64:n asentaminen käyttökuntoon ei ole juuri sen vaikeampaa kuin stereo-laitteistonkaan, eikä teknistä kykyään epäilevälläkään tulisi olla mitään vaikeuksia. Muista kuitenkin käyttää asianmukaista virtalähdettä, muuten tietokoneesi saattaa vahingoittua.



Kasettiportti on laitteen takaa katsoen toinen aukko oikealta. Siihen liitetään Commodoren kasettinauhuri. **RF-modulaattori** eli TV-liitäntä on takaa katsoen viides reikä oikealta. **Sarjaportti**, joka on takaa katsoen kolmas aukko oikealta, on tärkeä, koska siihen liitetään Commodoren levyasema ja/tai kirjoitin. **Laajennusportti** on äärimmäisenä vasemmalla, ja sillä on lukuisia käyttömahdollisuuksia. Muun muassa moduuleina myytävät pelit liitetään tähän porttiin. **Käyttäjän portti** on äärimmäisenä oikealla, ja sen käyttömahdollisuudet ovat myös moninaiset. Paitsi että siihen voidaan sovittaa puhelinyhteyteen tarvittava modeemi, myös eräät kirjoittimet ja lisälaitteet yhdistetään Commodoreen tämän portin kautta. Kokeilijat käyttävät mielellään tätä porttia omiin lisälaitteisiinsa. Sitä kutsutaan toisinaan myös rinnakkaisportiksi, koska siinä on kahdeksan *rinnakkaista* tietoja välittävää pistoketta.

Kontrolliportteja käytetään pääasiassa pelisäätimien liittämiseen, mutta eräät ohjelmat vaativat ns. ”mustan laatikon” liittämisen jompaankumpaan porttiin. (Tämä on turvatoimi ohjelmien luvattoman kopioinnin estämiseksi.) Kontrolliportit sijaitsevat laitteen oikeassa laidassa virtakytkimen vieressä.

A-V-porttia käytetään monitorien liittämiseen. Siihen voi myös kytkeä stereolaitteet tai sen kautta voi syöttää tietokoneeseen ääntä ulkoisesta lähteestä. Kätevä rakentaja voi itsekin juottamalla tehdä omia liitäntöjään, mutta muiden kannattaa kääntyä laitteen myyjän puoleen. Jos käytät monitorina tavallista televisiota, ei tästä liitännästä tarvitse välittää.

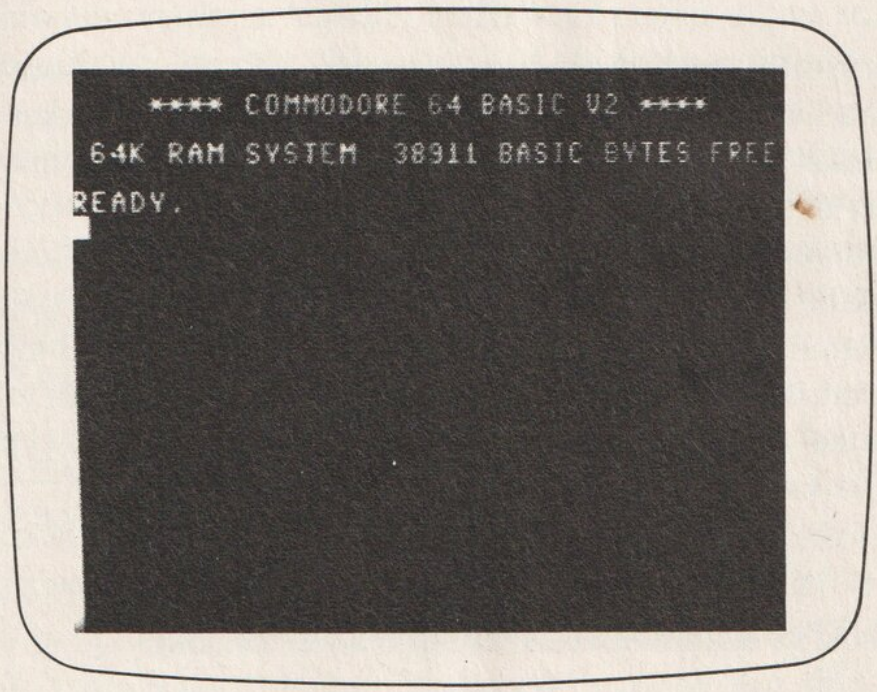
Toimintatarkistus

Jos olet jo kytkenyt koneesi ja tiedät sen toimivan hyvin, voit hypätä tämän osan yli. Mutta jos olet juuri ostanut koneesi ja haluat tietää, toimiiko kaikki kunnolla, tässä on yksinkertainen tarkistusohjelma.

Varmista ensin, että tietokoneesi toimii. Kytke laitteisto yhteen käsikirjassa esitetyllä tavalla. Jos sinulla on kannettava

versio, suurin osa liitännöistä on jo valmiina. Kytke virta ja säädä television/monitorin värit siten, että kuvaruutu on sininen ja kirjaimet vaaleansiniset kuten rajauskin. Kuvaruudussa tulisi nyt lukea:

****COMMODORE 64 BASIC V2****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.



(Kannettavassa laitteessa viestin tulisi olla täysin sama.)

Sanan READY alapuolella pitäisi olla pieni vaaleansininen vilkkuva neliö. Tätä neliötä kutsutaan kohdistimeksi eli kursoriksi, joka on peräisin ranskalaisesta, juoksijaa merkitsevästä sanasta. Tämä kohdistin ”juoksee” kuvaruudussa ja näyttää aina, missä kohdassa olet.

Viestin ylin rivi kertoo, että tämä on BASIC V2 eli tietokoneen ohjelmointikielen versio 2. Merkintä 64K RAM tarkoittaa 64 ”kiloa” eli että ns. luku- ja kirjoitusmuistia (random access memory) on *koko järjestelmän käytettävissä 64 000 merkkiä.*

Ja 38911 BASIC BYTES FREE tarkoittaa, että koko muistista on 38911 *tavua* käytettävissä BASIC-ohjelmien kirjoittamiseen. (Jokainen tavu vastaa yhtä merkkiä.) Koko 64 kilon muisti ei ole käytettävissä, koska tietokone tarvitsee siitä osan BASIC-kieleen ja omiin toimintoihinsa.

Jos tämä maaginen luku — 38911 BASIC BYTES FREE — ei ilmesty kuvaruutuun, palauta laite myyjälle. Jotain on vialla ja tietokone täytyy korjata tai vaihtaa. Jotakin on vialla myös silloin, jos kohdistin ei ole sanan READY lähellä tai jos se ei vilku.

Jos koneesi on tätä myöten kunnossa, voit nyt tarkistaa toiminnan kirjoittamalla lyhyen ohjelman. Ei haittaa, vaikket tiedä mitään ohjelmoinnista, koska tämän lyhyen testin voit toteuttaa tietämättä mitään sen yksityiskohdista.

Commodore 64 -laitteissa näppäimistö ei toimi aivan samalla lailla kuin tavallisessa kirjoituskoneessa. Ensiksikin näppäimiä on enemmän, ja kun koneeseen kytketään virta, kaikki kirjaimet ovat isoja kirjaimia. Jos painat näppäintä SHIFT ja yrität kirjoittaa kirjaimen, näet merkillisen kuvion, mutta siitä ei tällä hetkellä kannata välittää.

Kirjoita tämä lyhyt ohjelma *aivan* samalla lailla kuin se on esitetty jäljempänä. Jokaisen numeroidun rivin lopussa paina RETURN-näppäintä, jolla tieto siirtyy koneeseen.

```
10 PRINT "TAMA ON COMMODORE-TIETOKONE"
20 GOTO 10
```

(Jos Commodoressasi on skandinaavinen näppäimistö, voit tietysti kirjoittaa "tämä", mutta koska useissa tietokoneissa on amerikkalainen näppäimistö, ei ole aina mahdollista käyttää kirjaimia ä, ö ja å.)

Rivillä 10 tämä pieni ohjelma käskee tietokoneen kirjoittaa kuvaruutuun lauseen TAMA ON COMMODORE-TIETOKONE. Rivi 20 käskee koneen toistaa sitä loputtomiin. Tämä lienee sentyyppinen ohjelma, jonka lähes jokainen on ensimmäiseksi BASIC-kielellä kirjoittanut.

Kirjoita nyt sana RUN ja paina näppäintä RETURN, jotta tietokone ymmärtää käskyn.

Kuvaruudun tulisi pikaisesti täyttyä lauseesta TAMA ON COMMODORE-TIETOKONE. Vaikka alin rivi vilkkuukin nopeasti (koska kuvaruutuun tulee useita uusia lauseita sekunnissa), lauseen tulisi olla selvä ja helppolukuinen. Jos kuvaruutuun ilmaantuu ''lumisadetta'' tai muita häiriöitä, testi on epäonnistunut. Käy vaihtamassa tietokoneesi.

Jotkin Commodoret tulevat kokoomahihnalta viallisina. Yleisimpiä ongelmia (etenkin varhaisissa malleissa) on se, että kuvaruutuun ilmestyy ''lumisadetta'' ohjelman suorituksen aikana.

Jos testiohjelma voidaan suorittaa ja tietokone toimii, pysäytä ohjelma painamalla näppäimistön vasemmassa laidassa olevaa RUN/STOP-näppäintä. Kuvaruudussa tulisi nyt lukea:

BREAK IN LINE 10
READY.

tai

BREAK IN LINE 20
READY.

Kohdistimen tulisi taas näkyä vilkkuvana.

Yritä nyt tallentaa tämä ohjelma kasettinauhalle ja ladata se sieltä. Laita kasetti nauhuriin kun olet varmistunut siitä, että nauha on kelattu alkuun. Tallentaaksesi ohjelman kirjoita näin:

SAVE ''OHJELMA YKSI''
Paina näppäintä RETURN.
Tietokoneen pitäisi vastata ilmoituksella:
PRESS RECORD & PLAY ON TAPE

Paina Commodoren kasettinauhurin näppäimiä RECORD ja PLAY. Kuvaruudun pitäisi tyhjentyä ja muuttua rajauksen väriseksi noin kahdenkymmenen sekunnin ajaksi. Kun kuvaruutuun tulee taas tekstiä, siinä pitäisi lukea READY. Kelaa nyt nauha takaisin alkuun.

Ilman sopivaa adapteria et voi käyttää muita kuin Commodoren omaa nauhuria. Commodore Datasette -nauhurissa on erityinen piiri, joka havaitsee, onko näppäimiä painettu.

Kun nauha on kelattu alkuun, katkaise virta tietokoneesta ja kytke se jälleen. Tietokoneen muisti on nyt täysin tyhjä. (Tämä on vain yksi tapa asettaa tietokone alkutilaan. Myöhemmin opit parempia keinoja.) Kirjoita sana LOAD ja paina näppäintä RETURN. Tietokoneen tulisi ilmoittaa:

PRESS PLAY ON TAPE

Paina kasettinauhurin PLAY-näppäintä. Kuvaruutu tyhjenee jälleen muutamaksi sekunniksi. Kun nauhuri pysähtyy, kuvaruutuun tulevat sanat:

FOUND OHJELMA YKSI

Kone odottaa nyt sinulta käskyä jatkaa ja ladata ohjelma. Voit tehdä sen painamalla näppäimistön vasemmassa alalaidassa olevaa näppäintä, jossa on Commodoren liikemerkki. Kuvaruutu tyhjenee jälleen muutamaksi sekunniksi, jonka jälkeen siinä pitäisi lukea:

LOADING
READY.

Kirjoita RUN ja paina näppäintä RETURN. Ohjelman tulisi toimia kuten ennenkin. Vaikka tietokoneen muisti tyhjeni katkaistessasi virran (vaikka vain pieneksi hetkeksi), tämä koe todistaa, että kone pystyy tallentamaan ja lataamaan ohjelman kasettinauhalta. Jos tämä testi onnistuu, tietokone on hyväkuntoinen.

Jos sinulla on levykeasema, yritä tallentaa ja ladata ohjelma levykkeeltä. Kytke levykeasemaan virta ja varmistu sitä ennen, ettei levykeasemassa ole levykettä.

Huomaa, että Commodoren levykeasema vahingoittaa hyvin

helposti levykkeellä olevia tietoja, jos levykeasemassa on levyke virtaa katkaistaessa tai kytkettäessä.

Laita uusi, tyhjä levyke levykeasemaan siten, että etiketti on ylöspäin ja levykkeen pitkänomainen aukko takana. Työnnä levyke sisään pohjaan asti. Sulje levykeaseman luukku vetämällä metalliliuskaa alas itseäsi kohti, kunnes se lukkiutuu.

Sinun on valmisteltava levyke käyttöä varten; näkymättömiä uria kirjoittuu levykkeen magneettipinnalle. Valmistelua tarvitsevat *vain* sellaiset levykkeet, jotka ovat uusia tai joita pyyhitään puhtaiksi uutta käyttöä varten. Joka kerta kun valmistelet levykkeen, tuhoat tiedot ja ohjelmat, jotka olet siihen aikaisemmin tallentanut.

Valmistellaksesi levykkeen kirjoita:

```
OPEN 1,8,15 [ja paina RETURN]
PRINT#1, "N:OHJELMALEVY,OL" [ja paina RETURN]
```

Levykeasemassa olevan punaisen valon tulisi palaa minuutin tai pari, ja sinun pitäisi kuulla, miten levyke pyörii levykeasemassa.

Commodoren levykeasemassa on kaksi pientä valoa. Vihreä valo osoittaa, että virta on kytketty. Toisella, punaisella valolla on kaksi merkitystä. Ensiksikin se osoittaa, että tietoa kirjoitetaan levykkeelle tai levykkeeltä. Punaisen valon vilkkuminen osoittaa, että on tapahtunut virhe. Se voi olla sinun tai koneen tekemä.

Jos punainen valo vilkkuu, yritä uudelleen ja varmistu, että kirjoitat edellä olevan käskyn aivan oikein. Jos valo ei vilku, olet onnistunut valmistelemaan levykkeen. Kirjoita nyt:

```
SAVE "OHJELMA YKSI",8
```

Paina näppäintä RETURN. Levykeaseman tulisi toimia muutama sekunti, jolloin tiedät, että se tallentaa ohjelmaasi levykkeelle.

Kun levykeasema lakkaa toimimasta (ääni vaimenee ja punai-

nen valo sammuu), sammuta taas tietokone hetkeksi. Kytke virta ja kirjoita:

```
LOAD "$",8 [ja paina RETURN]
```

Kun kone ilmoittaa READY, kirjoita LIST ja paina RETURN. Kuvaruudussa tulisi lukea jotensakin näin:

```
0 "OHJELMALEVY  " OL 2A
1  "OHJELMA YKSI"  PRG
663 BLOCKS FREE
READY.
```

Tämä on levykkeen sisällysluettelo. Se kertoo levykkeen nimen ("OHJELMALEVY"), siihen tallennetut ohjelmat (vain yksi, nimeltään "OHJELMA YKSI") ja sen, kuinka paljon tilaa on jäljellä (663 ns. sivua, joka on suunnilleen 168 000 tavua muistitilaa). Toistaiseksi saat vain uskoa, että se on totta.

Kirjoita lopuksi:

```
LOAD "OHJELMA YKSI",8 [ja paina RETURN]
```

Tämän pitäisi ladata ohjelmamme koneeseen siten, että voit taas suorittaa sen.

Commodoren levykeaseman alistaminen tähän testiin on hyvin tärkeää. Testaamattomiakin levykeasemia on markkinoilla. Tiedät ostaneesi sellaisen, jos levykeasemasi ei valmistele levykettä oikein tai jos ohjelman tallennus ja lataus eivät onnistu moitteettomasti.

Useimmiten uuden levykeaseman ongelmana on nopeus, jolla se pyörittää sisällä olevaa levykettä. Jos olet ostanut levykeasemasi elektroniikkaliikkeestä tai tietokonemyymälästä, voit todennäköisesti palauttaa sen sinne yksinkertaista säätöä varten. Jos taas olet ostanut sen tavaratalosta, jolla ei ole omaa tietokonehuoltoa, pyydä kaupasta toinen sen sijaan että jättäisit ostamasi korjattavaksi. Tai voit valita valtuutetun Commodoren

huollon ja pyytää heitä tarkastamaan se.

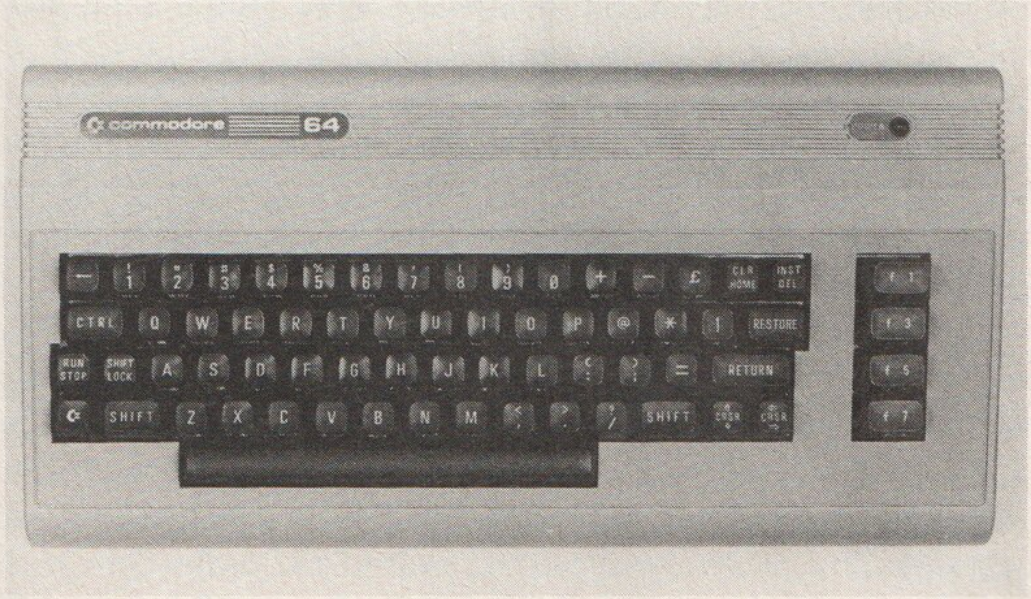
Toistaiseksi saatat olla ymmällä siitä, mitä olet tehnyt, mutta opit kyllä ''miksi'' ja ''kuinka'' muutaman sivun kuluttua. Olet nyt kuitenkin oppinut perusmenetelmän ohjelmien tallentamiseksi ja lataamiseksi kasetilta ja levykkeeltä.

Päätehtävämme oli tarkistaa pikaisesti laitteen toiminta. Ei liene tarpeen sanoa, ettei tämä ollut mikään perusteellinen testaus. Jos tietokonelaitteisto selviää testistä, tiedät vain sen, että se näyttää toimivan.

Näppäimistö

Commodore 64 -laitteiden näppäimistö on suunnittelultaan hieman epätavallinen. Lähes jokainen näppäin toimii kolmella eri tavalla. Kaikkia näitä toimintoja ei ole esitetty selkeästi näppäimistössä. Kaikkea ei toki tarvitse yrittää oppia kerralla. Näppäimistössä on paljon toimintoja, ja siihen tottuminen vie aikansa.

RETURN-näppäin vie kuvaruutuun kirjoitetun sanan tai rivin tietokoneen keskusmuistiin. Se on ilmeisesti kaikkein tärkein yksittäinen näppäin. Joissakin malleissa ja tietokoneissa se



on merkitty sanalla ENTER. Tämän näppäimen nimi on peräisin kirjoituskoneiden ja kaukokirjoittimien sanastosta, ja niissä sitä sanotaan "telan-" tai "vaunupalautukseksi", mikä olikin sen alkuperäinen tehtävä.

SHIFT-näppäimellä valitaan kirjoituskoneissa joko isot tai pienet kirjaimet. Tietokoneen näppäimistössä se toimii suunnilleen samalla tavalla. Vaikka sillä onkin useampia tehtäviä, yleissääntö pätee. Jos painat näppäintä SHIFT LOCK kerran (ja kuulet naksahduksen), näppäimistö toimii kuin SHIFT-näppäintä painettaisiin koko ajan. Tästä voi aiheutua ongelmia, joten vältä SHIFT LOCK -näppäimen käyttöä ja tarkista sen asento aika ajoin, jos epäilet ettei jokin näppäin toimi haluamallasi tavalla. Voit vapauttaa SHIFT LOCK -näppäimen painamalla sitä uudelleen.

RUN/STOP-näppäintä käytetään ohjelman suorituksen pysäyttämiseen. Joissain toisissa tietokoneissa sen nimi saattaa olla "Break". RUN/STOP-näppäintä voidaan myös käyttää SHIFT-näppäimen kanssa (painamalla kumpaakin yhtä aikaa) ohjelmien lataamiseen ja suorittamiseen kasettinauhurilta.

Grafiikkatilasta ja kirjoituskonetilasta on muistettava, että Commodoren näppäimistö toimii eri tavalla kuin kirjoituskoneen näppäimistö. Tavallisesti pienten kirjaimien näppäimet antavatkin tulokseksi isoja kirjaimia. Tämä johtuu siitä, että 64-sarjan kotitietokoneissa on ns. grafiikkatila. Se tarkoittaa, että kun painetaan useita näppäimiä yhtä aikaa SHIFT-näppäimen kanssa, saadaan erityisiä graafisia symboleja, joita käytetään yksinkertaisten kuvien, laatikoiden, rajausten jne. tekemiseen. On toinenkin toimintatila — kirjoituskonetila — jossa tietokone käyttäytyy kirjoituskoneen tavoin.

COMMODORE-näppäintä painetaan yhtä aikaa SHIFT-näppäimen kanssa toimintatilan vaihtamiseksi kirjoituskonetilaan. COMMODORE-näppäin on vasemmassa alakulmassa ja siinä on Commodore-liikemerkki. Tulosten tulisi näkyä heti ruudussa siten, että aiemmat isot kirjaimet ovat muuttuneet pieniksi.

Kun muutetaan toimintatilaa grafiikkatilasta kirjoituskoneti-

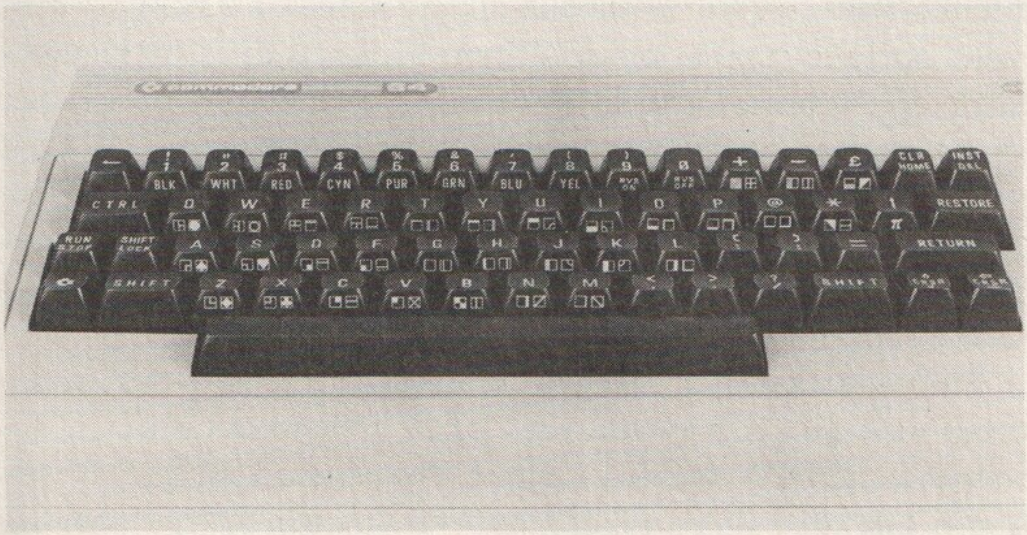
laan, on tärkeää muistaa, että tietokone ei erottele isoja ja pieniä kirjaimia samalla tavalla kuin sinä. Grafiikkatilassa ollessaan kone odottaa, että BASIC-käskyrivit ja muut käskyt on kirjoitettu isoilla kirjaimilla. Kirjoituskonetilassa tietokone odottaa näitä samoja sanoja pienikirjaimisina. BASIC- ja muut käskyt kirjoitetaan siis aina tarvitsematta painaa SHIFT-näppäintä, onpa koneen toimintatila mikä tahansa. Jos käytät isoja kirjaimia kirjoituskonetilassa, kone ei tiedä, mitä yrität tehdä.

Palataan nyt toistaiseksi grafiikkatilaan, jotta voimme tarkastella eräitä muita näppäimistön piirteitä. Käytä COMMODORE- ja SHIFT-näppäimiä yhtä aikaa vaihtaaksesi toimintatilaa.

Muistanet, että COMMODORE-näppäintä käytettiin myös käskettäessä konetta lataamaan ohjelma kasetilta. Ja sille löytyy vielä muitakin käyttömahdollisuuksia.

Grafiikkasymbolit ovat lähes jokaisen näppäimen etuosassa kahtena pienenä neliönä. Jokainen näistä sisältää tietyn graafisen symbolin.

Kun kone on grafiikkatilassa, paina näppäintä W. Kuvaruu-



dussa näkyy W, eikö totta? Paina nyt SHIFT-näppäintä ja W:tä yhtä aikaa. Kuvaruutuun ilmestyy jotakin, joka näyttää kirjaimelta O, mutta ei kuitenkaan ole O. Se on graafinen

merkki. Paina nyt COMMODORE-näppäintä ja W:tä. Näin saat merkin, joka näyttää oikealle kallistuneelta T:ltä. Olet juuri saanut samalla näppäimellä aikaan kolme erilaista merkkiä.

Jos painat SHIFT- ja COMMODORE-näppäimiä yhtä aikaa, näet, mitä kirjoituskonetilassa tapahtuu. Kuvaruutuun ilmestyy pieni w, iso kirjain ja sama graafinen merkki jonka näit viimeksikin, kyljellään oleva T. Yleensä käy näin: näkyy isoja ja pieniä kirjaimia, ja vasemmalla oleva graafinen merkki pysyy samana vaihdettaessa grafiikkatilasta kirjoituskonetilaan. Tästä on tosin muutamia poikkeuksia. Yritä samaa näppäimellä, jossa on ylöspäin suuntautuva nuoli, ja toisella, jossa on punnan merkki. Kirjoita nämä kolmella eri tavalla (siis ilman SHIFT-näppäintä, SHIFT-näppäimen kanssa ja COMMODORE-näppäimen kanssa). Siirry sitten grafiikkatilasta kirjoituskonetilaan, jolloin näet erilaiset graafiset merkit. Tämä on eräs poikkeus säännöstä, mutta se tuskin haitannee työtäsi.

CTRL-näppäin on kuten COMMODORE-näppäinkin toiminnoiltaan SHIFT-näppäimen tapainen. CTRL on ”control” eli hallinta.

Näytön käänteistila osoittaa erään tavan käyttää hallintanäppäintä. Paina CTRL ja samaan aikaan numeroa 9 ylimmältä riviltä. Kirjoita jotakin kuvaruutuun. Kirjoitus tulostuu käänteisenä, eli jokainen kirjoittamasi merkki on kuvaruudun värinen omaa vaaleaa taustaansa vasten. (Tämä pätee vain, jos värit ovat edelleen samat kuin heti virran kytkemisen jälkeen. Käänteisen merkin taustan väri on aina sama kuin merkin alkuperäinen väri.) Tämä käänteisväritys on hyödyllinen korostettaessa tiettyjä sanoja tai lauseita kuvaruudussa.

Pääset palaamaan takaisin tavanomaiseen (siis ei-käänteiseen) näyttöön painamalla yhtä aikaa näppäintä CTRL ja numeronäppäintä 0. Näiden kahden näppäimen toiminta on selkeästi merkitty niiden etuosaan. RVS ON merkitsee ”käänteisyys päälle” ja RVS OFF ”käänteisyys pois päältä”.

Värien käyttöä opit painamalla yhtä aikaa CTRL-näppäintä ja mitä tahansa numeronäppäintä 1—8, sekä katsomalla samaan aikaan kohdistinta. Olet juuri keksinyt uuden tavan

muuttaa kuvaruudulle kirjoittamiesi merkkien väriä. Tässä on luettelo väreistä, jotka saat aikaan painamalla yhtä aikaa CTRL-näppäintä ja jotakin numeronäppäintä.

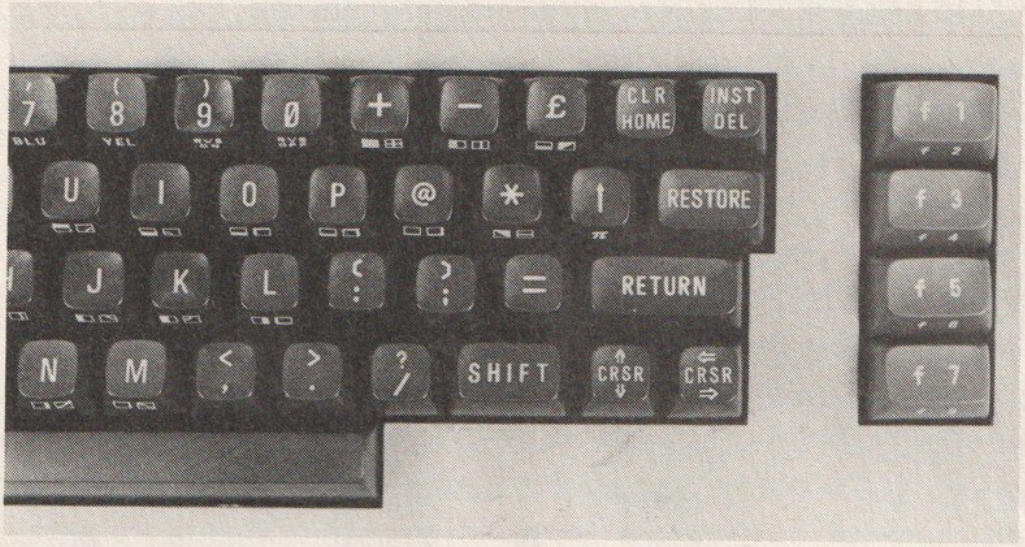
Näppäimet	Väri	Näppäinmerkintä
CTRL—1	Musta	BLK
CTRL—2	Valkoinen	WHT
CTRL—3	Punainen	RED
CTRL—4	Puuterinsininen	CYN
CTRL—5	Sinipunainen	PUR
CTRL—6	Vihreä	GRN
CTRL—7	Sininen	BLU
CTRL—8	Keltainen	YEL

Vaikka näppäimistöstä ei ilmenekään, on näiden lisäksi vielä käytettävissä kahdeksan väriä kun painat toista näppäinyhdistelmää. Käytä vasemmassa alakulmassa olevaa COMMODORE-näppäintä:

Näppäimet	Väri	Näppäinmerkintä
COMM—1	Oranssi	BLK
COMM—2	Ruskea	WHT
COMM—3	Vaaleanpunainen	RED
COMM—4	Harmaa 1 (tumma)	CYN
COMM—5	Harmaa 2 (keski)	PUR
COMM—6	Vaaleanvihreä	GRN
COMM—7	Vaaleansininen	BLU
COMM—8	Harmaa 3 (vaalea)	YEL

Kaikki kirjoittamasi ei välttämättä näy sinistä taustaa vasten. Monissa Commodore 64 -tietokoneissa (etenkin varhaisimmissa malleissa) ei voi käyttää kaikkia mahdollisia tausta- ja rajausvärejä. Ei ole mitään sääntöä siitä, mitkä värit näkyvät hyvin milläkin taustalla.

Taustan ja rajauksen värejä ei valitettavasti voida muuttaa yhdellä ainoalla näppäimellä kuten merkkien värejä.



Toiminäppäimet ovat neljä suurta näppäintä näppäimistön oikeassa laidassa. Näiden näppäinten yläosassa on merkit f1, f3, f5 ja f7 ja etuosassa f2, f4, f6 ja f8. Jälkimmäisiä voidaan ajatella SHIFT-näppäimen avulla toimiviksi. Ironista on tosin, että näillä toiminäppäimiksi kutsutuilla ei tehdä mitään ilman niitä hyödyntäviä ohjelmia. Monissa ohjelmissa näitä näppäimiä kuitenkin käytetään eri tavoin.

RESTORE-näppäin, joka on näppäimistön oikeassa laidassa, painettuna yhtä aikaa RUN/STOP-näppäimen kanssa, keskeyttää ohjelman suorituksen (kuten pelkkä RUN/STOP-näppäinkin) ja palauttaa tietokoneen lähtötilaan. Kuvaruutuun tulevat siis alkuperäiset värit (vaaleansininen tummansinisellä taustalla). Muut sisäiset toiminnot palautuvat siihen tilaan, jossa ne olivat konetta käynnistettäessä.

RESTORE-näppäimen käytön etuna on, että mahdolliset muistissa olevat ohjelmat säilyvät, vaikka tietokone onkin palautettu lähtötilaan. Koneen sammuttamista ja virran kytkemistä kutsutaan usein ”kylmäkäynnistykseksi”, ja se tuhoaa muistissa olevan ohjelman. RESTORE-näppäimen painamista voidaan pitää ”lämpimänä käynnistyksenä”, koska muisti ei tyhjene.

Korjausnäppäimet kuuluvat Commodoren kotitietokoneiden tehokkaimpiin näppäimiin. Ne tekevät ohjelmien kirjoittamisen mahdollisimman helpoksi. Ohjelmarivien ja muun kuvaruututekstin korjailun lisäksi korjausnäppäimiä voidaan käyttää ohjelmoinnissa. Seuraavassa selostetaan lyhyesti korjausnäppäinten toimintaa.

CRSR-näppäimet, jotka ovat näppäimistön oikeassa alalaidassa liikuttavat kohdistinta. Niihin on merkitty CRSR ylös ja alas ja CRSR vasemmalle ja oikealle. Kun painat CRSR ylös ja alas -näppäintä, kohdistimen tulisi liikkua alaspäin. Kun painat SHIFT-näppäintä ja samaa CRSR-näppäintä, kohdistin liikkuu ylöspäin. Paina CRSR vasemmalle ja oikealle -näppäintä. Nyt kohdistin liikkuu oikealle. Jos painat yhtä aikaa SHIFT-näppäintä ja tätä näppäintä, kohdistin liikkuu takaisin vasemmalle. (Jos näin ei tapahdu, tarkista, onko SHIFT LOCK -näppäin jäänyt alas.)

CLR/HOME-näppäin on oikeassa ylälaidassa INST/DEL -näppäimen vieressä. HOME tarkoittaa kohdistimen alkuasentoa kuvaruudun vasemmassa ylälaidassa, joten CRL/HOME -näppäimen painaminen palauttaa kohdistimen aina sinne. Jos painat yhtäaikaisesti näppäimiä SHIFT ja CRL/HOME, palaa kohdistin alkuasentoonsa ja tyhjentää lisäksi ruudun kaikista merkeistä. Jos painat yhtä aikaa näppäimiä SHIFT ja CLR/HOME ennen kuin olet painanut RETURN-näppäintä, hävität rivin jota olet ollut kirjoittamassa.

INST/DEL-näppäimessä DEL merkitsee poistamista ja INST lisäämistä. Kirjoita rivi tekstiä ja paina INST/DEL-näppäintä. Joka kerran kun painat sitä, kohdistin liikkuu merkin verran takaisinpäin ja poistaa merkin jonka yli se kulkee. Jos painat yhtä aikaa näppäimiä SHIFT ja INST/DEL, tulee riviin tyhjiä aukkoja.

Kirjoita jotakin. Aseta sitten kohdistin mihin tahansa kohtaan riville käyttämällä näppäintä CRSR vasemmalle ja oikealle. Paina SHIFT-näppäintä ja napauta muutaman kerran INST/DEL-näppäintä. Jokainen kohdistimesta oikealle oleva merkki liikkuu kauemmaksi kuvaruudussa jättäen aukkoja nii-

hin kohtiin, joissa aikaisemmin oli kirjaimia. Nämä aukot voidaan täyttää millä tahansa kirjaimilla tai jopa välilyöntinäppäimellä.

Nämä neljän näppäimen kahdeksan eri toimintoa muodostavat mukavan tavan kirjoittaa tietoja tietokoneeseen. Mikään ei kuitenkaan onnistu suoralta kädeltä, ja korjausnäppäinten käyttö saattaa aluksi tuntua hankalalta. Mutta kun tottuu näiden näppäinten toimintaan ja sijaintiin, voi hallita kuvaruutua aivan mielensä mukaan.

Näitä korjausnäppäimiä kannattaa opetella käyttämään erityisesti siinä tapauksessa, että aiot käyttää tietokonettasi tekstinkäsittelyyn.

Toistonäppäimiä Commodoressa on neljä. Ne toistavat toimintaansa, jos niitä painaa jatkuvasti lyhyenkin aikaa. Nämä toistonäppäimet ovat CRSR-näppäimet, INST/DEL-näppäin ja välilyöntinäppäin.

Paras tapa tutustua näppäimistöön on harjoitella. Kirjoita kuin kirjoituskoneella ja käytä korjausnäppäimiä muutoksiin.

(Ei muuten kannata painaa RETURN-näppäintä kuten kirjoituskoneessa. Tietokone ei ehkä ymmärrä, mitä yrität sanoa, ja se saattaa ilmoittaa, että olet tehnyt virheen.)

QWERTY-näppäimistö

Monilla on vaikeuksia tottua kirjoituskonemaiseen näppäimistöön. Commodoressa (kuten lähes kaikissa muissakin kotitietokoneissa) on niin sanottu QWERTY-näppäimistö, jonka nimi tulee ylärivin kirjainten järjestyksestä. Ylärivi on kokonaisuudessaan QWERTYUIOP. Tarina kertoo, että se suunniteltiin sellaiseksi siitä syystä, että kirjoituskoneiden myynnin alkaessa vuonna 1873 myyntimiehet pystyivät kirjoittamaan sanan ”typewriter” eli kirjoituskone englanniksi pelkästään ylärivin kirjaimia käyttämällä.

Todellinen syy tähän kömpelöön järjestykseen on kuitenkin se, että se suunniteltiin hidastamaan konekirjoittajien vauhtia.

Erottamalla usein tarvittavat näppäimet kauas toisistaan suunnittelijat halusivat varmistaa sen, että konekirjoittajat joutuisivat hakemaan oikeita kirjaimia ja kirjoittaisivat niin hitaasti, että ensimmäiset hennot mekanismit eivät juuttuisi kiinni. Ja niin me vielä yli 100 vuotta myöhemmin käytämme tätä vanhaa ja epäkäytännöllistä QWERTY-näppäimistöä, ja nyt, kotitietokoneiden tultua markkinoille, on QWERTYn keksijöillä entistä enemmän syytä hykerrellä haudoissaan. QWERTYä näkee nykyään enemmän kuin koskaan aikaisemmin.

Miksei kukaan ole sitten tehnyt mitään asian hyväksi? On kyllä yritetty. Joissakin viime vuosina myydyissä taskutietokoneissa oli aakkosellinen näppäimistö. Se osoittautui vielä hankalammaksi kuin QWERTY. Toinen vaihtoehto oli DSK (sanoista Dvorak Simplified Keyboard eli Dvorakin yksinkertaistettu näppäimistö), joka saatiin aikaan vuonna 1932 kahden vuosikymmenen tutkimuksen pohjalta. DSK sijoitti käytetyimmät näppäimet yläriville (ASDF tuli QWERTYn tilalle). Väitettiin, että kolmisen tuhatta englannin yleisintä sanaa voitiin kirjoittaa pelkästään käyttämällä ylärivin näppäimiä. Joitakin vuosia sitten yritettiin ottaa käyttöön yhdessä kädessä pidettävää näppäimistöä, jossa merkit muodostettiin sormien avulla. (Englannissa — ja myös Suomessa — on myynnissä pieni tekstinkäsittelylaite joka käyttää tätä näppäimistöä.)

QWERTY-näppäimistön ylivalta kuitenkin jatkuu entisellään ja muut näppäimistöt on lähes unohdettu. Vain muutamissa kirjoituskoneissa on toinen järjestelmä, eikä sellaista ole edes odotettavissa mihinkään kotitietokoneeseen.

Voidaksesi viestiä tietokoneesi kanssa sinun on siis kamppailtava perinteisen näppäimistön heikkouksien parissa. Toisaalta tietokoneen käytön etuja on se, että opit kirjoittamaan koneella lähes vaivattomasti — totut yhä paremmin näppäimistöön. Jos harrastaa tekstinkäsittelyä tietokoneella, voi pian alkaa ihmetellä, miksi kukaan enää haluaa kirjoittaa kirjoituskoneella.

2

Perustaidot

Aikoopa sitten ohjelmoida tietokoneella tai vain käyttää ostettuja ohjelmia, on joka tapauksessa opittava tietyt perustaidot. On osattava ladata ja tallentaa kasetilla tai levykkeellä olevia ohjelmia ja kyettävä suorittamaan ne. On myös tiedettävä jotakin BASIC-ohjelmien korjaamisesta eli ”editoinnista”, tietokoneen käyttämisestä laskimena ja siihen valmiiksi rakennetusta ajastimesta.

RETURN-näppäin

Tässä kirjassa sinua kehotetaan jatkuvasti kirjoittamaan ohjelmaesimerkkejä ja näppäimistön toimintaa selventäviä esimerkkejä. Voidaksesi välittää nämä tiedot tietokoneelle sinun on aina painettava näppäimistön oikeassa laidassa sijaitsevaa pitkänomaista näppäintä, jossa lukee RETURN. Jos et paina sitä, tietokone jää odottamaan vielä lisäohjeita. Toisinaan sinua muistutetaan tämän näppäimen painamisesta, toisinaan taas oletetaan, että olet jo oppinut tämän ensiarvoisen tärkeän seikan. RETURN-näppäintä tulisi painaa aina, kun olet saanut

valmiiksi BASIC-kielisen ohjelmarivin, kun olet muuttanut jo kirjoittamaasi riviä tai kun haluat antaa koneelle käskyn suoraan näppäimistöltä.

Ohjelman lataus

Tietokoneohjelmia tehdään kirjoittamalla käskyt koneen muistiin näppäimistöltä. Veisi kuitenkin tavattomasti aikaa kirjoittaa jokainen ohjelma joka kerran kun sitä haluaa käyttää. Ohjelmat voidaan kuitenkin tallentaa kasettinauhalle tai magneettiselle levykkeelle (jota tietokonekielessä usein kutsutaan ''lerpuksi'', jotta se erottuisi toisenlaisesta levymuistista, jäykästä levystä, johon mahtuu valtaisia tietomääriä).

Kun ohjelma on tallennettu käskyllä SAVE joko levykkeelle tai nauhalle, se voidaan myöhemmin kutsua takaisin koneen muistiin käskyllä LOAD. Nämä käskyt ovat siinä mielessä samanlaiset, että ne kopioivat ohjelmaa muistista johonkin ulkopuoliseen tallennusvälineeseen ja päinvastoin. Kun tallennat ohjelman käskyllä SAVE, siitä syntyy kopio nauhalle tai levykkeelle, mutta se jää edelleen myös koneen muistiin. Ja kun lataat ohjelman koneen muistiin käskyllä LOAD, se jää edelleen myös nauhalle tai levykkeelle.

Tietokoneen pääasiallinen luku- ja kirjoitusmuisti, jota kutsutaan RAM-muistiksi, on luonteeltaan tilapäinen. Kaikki mitä siihen kirjoitetaan — ohjelmia, tietoja, kuvia — katoaa heti kun kone sammutetaan edes sekunnin murto-osaksi. LOAD-käsky vastaanottaa digitaalisia signaaleja levykkeeltä tai äänisignaaleja nauhalta ja varastoi ne RAM-muistiin. SAVE-käsky muuntaa muistin sisällön vastaaviksi signaaleiksi ja lähettää ne nauhalle tai levykkeelle.

LOAD tyhjentää aina koneen muistin ja täyttää sen sitten uudella ohjelmalla. (Et voi käyttää käskyä LOAD liittämään ohjelmaa toiseen.)

Lataaminen nauhalta

Voit aloittaa lataamisen käyttämällä nauhaa, jonka loit edellisessä luvussa esitettyjen testien yhteydessä. Voit tietenkin käyttää myös jotain toista nauhaa, jolla on jokin ennalta tallennettu ohjelma.

Käskyä LOAD voidaan käyttää ohjelmanimen kera tai ilman sitä. Kirjoita:

LOAD [ja paina näppäintä RETURN]
tai
LOAD ''OHJELMA YKSI'' [ja paina RETURN]

Kun kirjoitat LOAD ja painat RETURN-näppäintä, tietokone vastaa aina seuraavasti:

PRESS PLAY ON TAPE

Kun painat nauhurin PLAY-näppäintä, kuvaruutu tyhjenee. Itse asiassa koko kuvaruutu muuttuu rajauksen väriseksi, jolloin

kaikki aikaisemmin kirjoitettu katoaa tilapäisesti. Kun näyttö palaa normaalitilaan, näet taas uuden viestin:

FOUND

Tämä tarkoittaa sitä, että nauhuri on löytänyt ohjelman. Jos ohjelmalla on nimi, sekin näkyy kuvaruudussa. Seuraavaksi on sinun vuorosi. Sinun on painettava COMMODORE-näppäintä, joka on näppäimistön vasemmassa alakulmassa. Jos et ole kirjoittanut ohjelman nimeä LOAD-käskyn jälkeen, latautuu ensimmäinen nauhalla oleva ohjelma.

Jos olet käyttänyt ohjelman nimeä, tietokone jättää huomiotta kaikki muut ohjelmat, kunnes löytää nimeämäsi ohjelman. Se pysähtyy ja näyttää jokaisen nimen, ja sinun on painettava COMMODORE-näppäintä jatkaaksesi oikean ohjelman hakemista.

Kun ohjelma on ladattu, näyttö palautuu normaaliksi ja näet viestin:

READY.

Nyt voit kirjoittaa RUN (ja painaa näppäintä RETURN), jolloin ohjelman suoritus alkaa.

Varhemmissa Commodore-tietokoneissa, kuten PET, CBM (joka oli paranneltu PET) ja VIC, ei ohjelman lataamiseksi tarvinnut painaa mitään näppäintä. Mutta Commodore 64 -mallissa oleva videopiiri asettaa koko järjestelmälle uusia vaatimuksia, jolloin on käynyt välttämättömäksi tyhjentää kuvaruutu ohjelman lataamisen ja tallentamisen ajaksi. Jos olet käyttänyt noita varhaisempia Commodoren tietokoneita, pidä mielessäsi tämä ylimääräinen vaihe, koska muuten tietokone ei lataa ohjelmaasi.

Toinen tapa ladata ensimmäinen nauhalla oleva ohjelma on painaa kumpaa tahansa SHIFT-näppäintä ja samanaikaisesti RUN/STOP-näppäintä. Kun teet tämän, ilmestyvät kuvaruu-

tuun heti sanat LOAD ja PRESS PLAY ON TAPE. Sinun on siitä huolimatta painettava COMMODORE-näppäintä nähdessäsi sanan FOUND. Tällä kertaa ohjelman suoritus kuitenkin alkaa automaattisesti — tämä on kätevä oikotie.

Kun käytät käskyä LOAD yhdessä ohjelman nimen kanssa, voit olla varma siitä, että ainoastaan sen niminen ohjelma latautuu. Nimen edessä on aina oltava lainausmerkki, mutta aina ei suinkaan ole tarpeen kirjoittaa koko nimeä.

Sanotaanpa vaikka, että samalle puolelle nauhaa on tallennettu kolme ohjelmaa, OHJELMA YKSI, OHJELMA KAKSI ja OHJELMA KOLME. Käsky LOAD ''OHJELMA KOLME'' lataa vain tämän yhden ohjelman.

Tunnistaakseen ohjelmat, jotka on ladattava, Commodoren tietokoneet käyttävät tekniikkaa, jota kutsutaan ''tyyppihauksi''. Jos käytät vain nimen ensimmäistä kirjainta — LOAD ''O — tietokone lataa minkä tahansa O:lla alkavan ohjelman — ''OHJELMA YKSI'', ''OHJELMA KAKSI'' tai vaikkapa ''OMPPO'' (jos sellainen on nauhalla). Jos käytät ensimmäistä sanaa — LOAD ''OHJELMA — tietokone lataa tällä sanalla alkavat ohjelmat jne.

Jotta tyyppihaku onnistuisi, on viimeinen lainausmerkki jätettävä pois. Jos sitä käytetään, tietokone luulee lainausmerkeissä olevaa nimeä ainutlaatuiseksi ja lataa ainoastaan sellaisen ohjelman, joka vastaa nimen kirjoitusasua *täsmälleen*.

```
LOAD ''OHJE
LOAD ''OHJELMA''
```

Ensimmäinen käsky lataa minkä tahansa OHJE-alkuisen ohjelman. Toinen lataa *ainoastaan* ohjelman, jonka nimi on OHJELMA.

Toisinaan tulee eteen ohjelmia, jotka on ladattava tietyllä tavalla. Latausohjeet ovat tällaiset:

```
LOAD ''OHJELMANIMI'',1,1
```

Kaksi numeroa, 1 ja 1, antavat tietokoneelle tietynlaiset ohjeet. Ensimmäinen ykkönen tarkoittaa kasettinauhurin *laitenumeroa,* joka on aina 1. (Laitenumeroista tarkemmin kohdassa Kuinka tietokone varastoi tietoa.) Tällä lailla tietokone tietää, että käytät kasettinauhuria. Mutta jos kirjoitat käskyn LOAD ilman *mitään* numeroita, tietokone *olettaa* sinun käyttävän nauhuria.

Toinen numero 1 käskee tietokoneen ladata ohjelman tiettyyn paikkaan. VICistä alkaen Commodoren tietokoneet ovat käyttäneet ''uudelleen sijoittavaa latausta''. Se on koneen käyttöjärjestelmässä oleva ohjelma, jonka ansiosta on mahdollista käyttää VIC- ja 64-perheiden tietokoneissa monia PET- ja CMB-koneille tehtyjä BASIC-ohjelmia. Sitä tarvitaan siksi, että koneet varastoivat ohjelmia eri tavoin.

Ellei toista ykköstä käytetä, tietokone aloittaa aina ohjelmien lataamisen kohdasta, johon BASIC-ohjelmat varastoidaan. Jotkin ohjelmat on kuitenkin kirjoitettu BASICin asemesta konekieleksi kutsutulla koodilla, joka toimii paljon nopeammin. Usein sinua pyydetään lataamaan ohjelmia käyttäen muotoa ''1,1'', jotta voisit varmistaa niiden latautumisen.

Ohjelman lataus ei onnistu

On kymmeniä syitä siihen, miksi ohjelma ei lataudu hyvin. Tässä on neljä yleisintä syytä.

Yksi: Kasettinauhurissa on jotakin vikaa. Syynä voi olla mikä tahansa, vaikkapa viallinen kosketin tai rikkoutunut elektroninen virtapiiri. Jos tämä tuntuu olevan syynä, ota yhteyttä laitteen myyjään. Valitettavasti se voi joskus olla turhauttava kokemus.

Kaksi: Nauhurisi toimii hyvin, mutta sen säädöt eivät ole kohdallaan. Tämä on useimmiten syynä silloin, jos pystyt lataamaan *itse* tallentamiasi ohjelmia, mutta toisten tallentamien ohjelmien lataus ei onnistu. Ratkaisu on äänitys- ja toistopään

säädössä. Se ei kuitenkaan ole helppo tehtävä, koska et kuule, mitä nauhalta tulee. Voit tehdä pikkusäädön ja katsoa, korjautuuko ongelma.

Commodoren kasettinauhurin yläpinnalla on pieni reikä. Kun painat PLAY-näppäintä, voit työntää pienen ruuvimeisselin tähän reikään ja kääntää säätöruuvia. Kun saat ruuvimeisselin sisään, muista, missä asennossa se on. Käännä ruuvia neljänneskierroksen verran ja yritä sitten ladata kasetti. Jos se ei auta, käännä ruuvia toiseen suuntaan.

Yritä säätämistä kuitenkin vain siinä tapauksessa, että tiedät tarkoin mitä teet. Jos et muista ruuvin alkuasentoa tai käännät sitä liikaa, voit säätää nauhurin sellaiseksi, ettet pysty enää lataamaan omiakaan nauhojasi.

Jos olet onnistunut lataamaan muita, eri lähteistä tulleita kasetteja, ja vain muutaman lataaminen ei onnistu, ei nauhurissasi todennäköisesti ole mitään vikaa. Vika saattaa olla siinä nauhurissa, jolla ne on tehty.

Jos sinulla on edelleen vaikeuksia ja sinusta tuntuu, että ne johtuvat väärästä säädöstä, vie nauhuri myyjälle tai Commodoren huoltoon, jotta säädöt saadaan kohdalleen.

Kolme: Jos sinulla on lataamisvaikeuksia silloin, kun koneessa on toinen, jo suoritettu ohjelma, vika ei ehkä ole tietokoneessa eikä nauhurissakaan. Eräät ohjelmat voivat muuttaa tietokoneen toimintaa niin, ettei kopiointi olisi mahdollista. Eräät ohjelmat taas saattavat käyttää tiettyjä toimintoja, jotka voivat vaikuttaa ohjelmien latautumisjärjestelmään.

Eräs nauhaa käytettäessä syntynyt vaikeus esiintyy silloin kun ohjelmassa, etenkin jossain pelissä, on käytetty spritegrafiikkaa. (Spritet ovat värikkäitä kuvaruudussa liikkuvia kohteita.) Toisinaan ohjelmoijat käyttävät ”kasettipuskuriksi” nimitettyä aluetta näiden kuvien varastoimiseen. Jos näitä spritejä näkyy kuvaruudussa, ei ohjelman lataaminen ehkä onnistu normaalisti. Helpoin tapa päästä eroon spriteistä on painaa yhtä aikaa RUN/STOP- ja RESTORE-näppäimiä.

Hyvä perussääntö silloin, kun ohjelmien lataaminen ei onnistu, on sammuttaa tietokone hetkeksi ja kytkeä virta jälleen.

Neljä: Jos ohjelma ei lataudu omalta eikä miltään muultakaan kokeilemaltasi kasettinauhurilta, on syynä todennäköisesti huono tai vahingoittunut nauha. Nauhan laatua on vaikea arvioida. Yleensä tulisi käyttää sellaista kasettinauhaa, joka sopii hyvin äänen nauhoittamiseen. On kuitenkin korostettu liikaa sitä, että tulisi ostaa kalliita nauhoja. Jotkin halvat nauhat ovat aivan riittävän hyviä ohjelmien nauhoittamiseen. Kun löydät hyvän merkin, pysy siinä.

Nauhan vahingoittuminen kasetin sisällä mitä todennäköisimmin myös tuhoaa ohjelman. Koska ohjelma varastoidaan äänisignaalina, pieninkin äänisignaalin katkeaminen aiheuttaa latauksen epäonnistumisen. Jos kasetti vahingoittuu, heitä se pois tai älä ainakaan käytä sitä enää tietokoneessa. Älä edes yritä korjata sitä tai silittää kurttuun mennyttä osaa.

Tallentaminen nauhalle

Käskyä SAVE käytetään miltei aina ohjelmanimen kanssa.

```
SAVE "OHJELMANIMI"
```

Tämä käsky tallentaa BASIC-ohjelman kasettinauhalle. Jos et anna ohjelmalle nimeä, se tallentuu joka tapauksessa, mutta et kykene tunnistamaan sitä ladatessasi. Kuten LOAD myöskään SAVE ei toimi tiettyjen ohjelmien yhteydessä — niiden jotka on suojattu luvattoman kopioinnin estämiseksi, ja eräiden sellaisten, jotka on kirjoitettu jollain muulla kielellä kuin BASIC.

Kun kirjoitat SAVE ja ohjelman nimen (ja muista painaa RETURN-näppäintä), tietokone vastaa:

```
PRESS RECORD AND PLAY ON TAPE
```

Kun painat näitä kahta kasettinauhurin näppäintä, kuvaruutu tyhjenee, kunnes ohjelma on tallennettu.

Sinun on muistettava painaa sekä PLAY- että RECORD

-näppäintä tallentaessasi ohjelmaa. Tietokone ei itse asiassa tiedä, oletko painanut molempia, ja se toimii samalla lailla, vaikka olisit painanut pelkästään näppäintä PLAY. (Mutta siinä tapauksessa et tietenkään voi tallentaa ohjelmaa.)

Jos haluat suojata tämän ohjelman vahingossa tapahtuvalta päällekkäistaltioinnilta, poista vasemmanpuoleinen pieni muovinen liuska kasetin yläreunasta. Nyt et voi painaa näppäimiä PLAY ja RECORD yhtäaikaisesti. Jos sitten muutat mielesi, riittää pieni pala teippiä peittämään syntyneen kolon, jotta voit jälleen nauhoittaa kasetille.

Tallennuksen tarkistus

Kannattaa tarkistaa, onko kirjoitettu ohjelma turvallisesti nauhalla. Tätä varten on olemassa käsky VERIFY. Se vertaa nauhalla olevaa ohjelmaa siihen ohjelmaan, joka on koneen keskusmuistissa.

Jos liität VERIFY-käskyyn ohjelman nimen, kasettinauhuri käy läpi kaikki nauhalla olevat ohjelmat ja etsii samannimisen ohjelman vertailtavakseen. Jos kirjoitat pelkästään käskyn VERIFY, kone vertaa ensimmäistä nauhalla olevaa ohjelmaa.

Jos ohjelma on tallentunut oikein, näkyy kuvaruudussa viesti ”OK”. Jos ohjelma ei ole samanlainen, näkyy seuraava viesti:

? VERIFYING ERROR

Tämä kertoo, että nauhalla oleva ohjelma ei ole samanlainen kuin koneen muistissa oleva. Nauha on ehkä vahingoittunut tai olet saattanut erehdyksessä muuttaa jonkin ohjelmarivin.

VERIFY-käskyä voidaan käyttää myös selvittämään, millä kohtaa nauhaa olet. Sinulla on esimerkiksi kaksi ohjelmaa, ”OHJELMA YKSI” ja ”OHJELMA KAKSI” samalla puolella nauhaa. Olet tehnyt muutoksia ”OHJELMA KAHTEEN”, joka on nyt tietokoneen muistissa, ja haluat tallentaa sen nauhalle uudelleen. Jos nyt lataat ”OHJELMA YHDEN”, muis-

tissa oleva ohjelma tuhoutuu. Voit kuitenkin antaa käskyn VERIFY "OHJELMA YKSI", jolloin nauha siirtyy "OHJELMA KAHDEN" alkuun vaikuttamatta muistissa olevaan ohjelmaan. Näet tosin yhä edellä olevan virheilmoituksen, mutta sinun ei tarvitse välittää siitä. Tämän jälkeen voit tallentaa "OHJELMA KAHDEN" uudelleen entisen version päälle, kunhan muutokset eivät ole kovin paljon pidentäneet ohjelmaa. (Jos ohjelma on liian pitkä, uusi nauhoitus tuhoaa nauhalta seuraavan ohjelman alun.)

Ohjelman suorittaminen

Käsky RUN aloittaa ohjelman suorittamisen. Useimpien ohjelmien suoritus voidaan keskeyttää painamalla näppäintä RUN/STOP.

RUN nollaa aina kaiken ohjelmassa olevan tiedon. (Tietoja muuttujista on seuraavassa luvussa.) Tämä tarkoittaa sitä, että numerot, nimet ja muut ohjelman käyttötiedot tuhoutuvat, kun annat seuraavan kerran käskyn RUN. On kaksi tapaa jatkaa ohjelman suoritusta keskeytyksen jälkeen ja kuitenkin säilyttää sen tarvitsemat tiedot.

Painettuasi näppäintä RUN/STOP voit kirjoittaa käskyn CONT, joka merkitsee "jatka". (Kirjoita se suoraan näppäimistöltä ja paina RETURN.) Ohjelman suoritus jatkuu siitä kohdasta johon se jäi.

Toinen tapa päästä ohjelmaan on käsky GOTO, jota on seurattava se BASIC-ohjelman rivinumero, josta haluat aloittaa ohjelman suorituksen. (Voit taas kirjoittaa käskyn GOTO suoraan näppäimistöltä, mutta muista painaa RETURN.) Jotta voit käyttää käskyä GOTO, sinun on ymmärrettävä ohjelman toiminta joko lukemalla REM-rivit (joilla on selventäviä huomautuksia) tai pystymällä lukemaan BASICiä. Älä koskaan koeta käyttää GOTO-käskyä sattumanvaraisen rivinumeron kanssa.

Voit liittää numeron myös käskyyn RUN. Se aloittaa ohjelman suorituksen määrätyltä riviltä, mutta pelkän RUN-käskyn tavoin sekin nollaa kaikki muuttujatiedot.

Commodoren levykeasema

Commodore 1541 -levykeasema lataa ja tallentaa tietokoneen antamaa tietoa aivan kuten kasettinauhurikin. Tieto tallentuu mustaan muovikuoreen suljetun joustavan levykkeen eli 'lerpun' magneettiselle pinnalle. Päinvastoin kuin kasettinauhuri levykeasema voi hakea ohjelmia sieltä täältä ja paljon nauhuria nopeammin. (Ohjelmat voidaan ladata levykkeeltä noin seitsemän kertaa nopeammin kuin nauhalta.) Levykkeeltä on myös mahdollista valita ohjelmia ja tietoja, vaikka ne eivät olisikaan peräkkäin. Ohjelmia ei siis ole tarpeen tallentaa tiettyyn järjestykseen.

Mustan muovikuoren oikeassa yläkulmassa on kolo. Se kertoo tietokoneelle, voiko se tallentaa tietoa levykkeelle. Jos kolo on auki ja näkyvissä, voit tallentaa ja ladata ohjelmia tälle levykkeelle. Jos kolo on peitetty (yleensä levykkeiden kotelossa olevalla pienellä tarralla), se on ''tallennussuojattu''. Se merkitsee sitä, että voit kyllä ladata levykkeellä olevia ohjelmia, mutta et voi tallentaa sille ohjelmia.

Tyhjät levyt täytyy valmistella levykeasemaa varten. Voit käyttää melkein minkälaista levykettä tahansa, kunhan varmistut siitä, että niissä on merkintä ''single-sided double density'' tai ''one-sided double density''. ''Single-sided'' tai ''one-sided'' merkitsee yksipuolista, toisin sanoen voit kirjoittaa tietoa vain levykkeen toiselle puolelle. (Eräät levyasemat kirjoittavat levykkeen kummallekin puolelle, mutta 1541 ei tee niin.) ''Double density'' eli kaksoistiheys on vanha ilmaisu niiltä ajoilta, jolloin levykkeille tallennettava tietomäärä oli kasvamassa. 1541-levyasema ei oikeastaan ole kaksoistiheysasema (jos ollaan tarkkoja), mutta sinun tulisi käyttää tällaisia levykkeitä välttyäksesi ongelmilta.

Muita sanoja, joita käytetään tyhjien levyjen kuvaamiseen, ovat ''soft sector'' ja ''hard sector''. Kumpikin niistä toimivat yhtä hyvin 1541-levyasemassa. Voit käyttää minkä tahansa valmistajan levykkeitä. Kun löydät hyvän, kohtuuhintaisen levykemerkin, pysy sitten siinä.

Levykkeen valmistelua kutsutaan *formatoinniksi.* Kun levyke formatoidaan, tapahtuu useita asioita. Se saa nimen ja kaksimerkkisen tunnusnumeron. Osa levykkeestä muodostaa hakemiston, johon tallentuvat ohjelmien ja muiden tiedostojen nimet. Ja viimeksi levykkeen pinnalle muodostuu 35 uraa. Jokaista näistä urista käytetään tiedon tallentamiseen; esimerkiksi ura 18 käsittää hakemiston. Jokainen ura jakautuu puolestaan *sektoreihin,* joista jokainen sisältää tietoa 256 tavua eli merkkiä. Näitä sektoreita kutsutaan myös *sivuiksi.* Levykkeen kokonaistila on 664 sivua eli lähes 170 000 tavua eli merkkiä.

Sinun tulisi ymmärtää heti aloittaessasi, ettet voi käyttää millään toisella tietokoneella — vaikkapa Applella tai Atarilla —

valmisteltua levykettä omassa Commodore 64:ssäsi, vaikka tällaisella levykkeellä olisikin omassa koneessasi toimiva BASIC-ohjelma. Jokaisessa tietokoneessa on erilainen menetelmä BASIC-ohjelmien käyttämiseen ja varastoimiseen, ja jokaisessa on myös oma menetelmänsä varastoida tietoa levykkeille. (Samoin on kasettien laita. Ei kannata edes yrittää ladata jollain toisella tietokoneella tallennettua kasettia.)

Levykkeitä tulisi kohdella lempeästi. Vaikka usein liioitellaankin lerppujen haurautta (sillä ne sietävät aivan hyvin jokapäiväistä käsittelyä), on mahdollista vahingoittaa levykkeellä olevaa tietoa. Useimpien kaupassa myytävien levykkeiden koteloissa on käsittely- ja säilytysohjeita, mutta eräitä asioita kannattaa aina toistaa (varsinkin suomen kielellä).

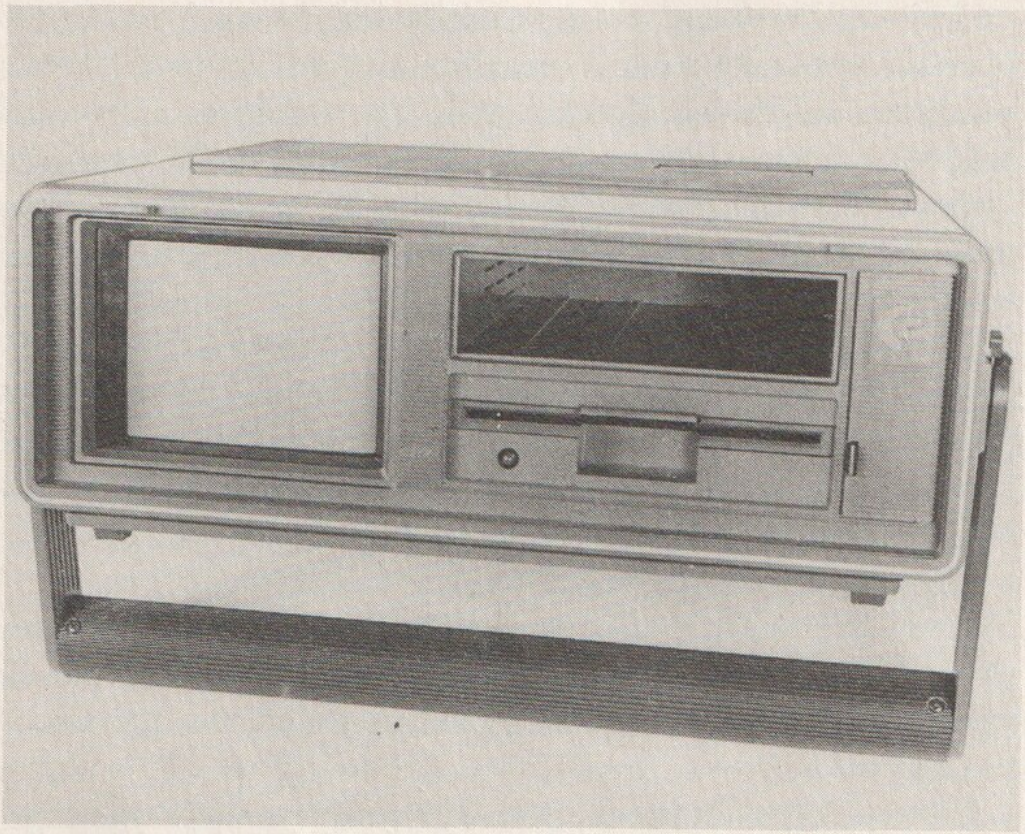
* Älä milloinkaan kosketa levykkeen suojakuoren aukosta näkyvää pintaa. Varo erityisesti levykkeen takaosassa olevaa aukkoa, sillä sille puolelle 1541 kirjoittaa. Älä myöskään taivuttele levykettä.

* Kun et käytä levykettä, pidä se paperisessa suojakotelossa. Säilytä levykkeet laatikossa tai muussa kotelossa kaukana tupakansavusta, pölystä ja liasta, joka voi vahingoittaa levykkeellä olevaa tietoa. Muista säilyttää levykkeet kuivassa ja viileässä paikassa.

* Älä säilytä levykkeitä magneettien tai magneettisten kenttien läheisyydessä. Magneettikenttiä voivat aiheuttaa mm. stereokaiuttimien tai televisioiden muuntajat.

Levykeaseman käyttö

Levykeaseman etupaneelissa on kaksi LED-valoa. Vihreä valo osoittaa, että virta on kytketty, ja punainen valo merkitsee kahta seikkaa: se palaa ja toisinaan vilkkuu levykeaseman ladatessa tai tallentaessa tietoa. Kun punainen valo vilkkuu ja levykeasema on lakannut toimimasta (et kuule tavanomaista hurinaa), on valo merkkinä virheestä. Tarkkaile aina tätä valoa, kun käytät levykeasemaa.



Jokaisen levykeaseman mukana tulisi olla levyke, jonka nimi on VIC-1541 TEST/DEMO. Muutaman testin lisäksi siihen sisältyy ohjelma nimeltä ''Wedge'' eli DOS Manager (DOS-ohjain). Tämä ohjelma on niin hyödyllinen, että ilman sitä et tule toimeen. (Testit ovat myös ehdottoman välttämättömiä. Suorita ne käskyllä RUN todetaksesi levykeasemasi toimintakelpoisuuden ja palauta asema myyjälle, jos se ei läpäise testejä.)

Wedgen avulla voit helposti ladata ohjelmia, tarkastaa levykkeen hakemiston, poistaa ohjelmia ja tiedostoja, formatoida levykkeitä ja lukea levykevirheitä. Sen nimi on Wedge eli ''kiila'' kahdestakin syystä. Ensiksikin sen käskyt on tavallaan kiilattu tietokoneen toimintajärjestelmään, ja toiseksi se käyttää jokaisen levykekäskyn edellä merkkiä ''>''. Nimi DOS-ohjain tulee siitä, että tämä pieni ohjelma ohjaa koneen keskeistä levynkäyttöjärjestelmää (Disk Operating System).

Wedgen avulla voit tehdä sellaista, joka on vaikeaa tavanomaisilla levykekäskyillä. Jäljempänä näet sitten näiden käskyjen oikeat BASIC-muodot.

Ladataksesi Wedgen avaa levykeaseman luukku, laita levyke sisään siten, että sen nimilappu on ylöspäin ja avoin aukko takana, ja sulje sitten ovi varovaisesti vetämällä luukun liuskaa alas ja itseesi päin. Kirjoita sitten

LOAD "C-64 WEDGE",8 [ja paina näppäintä RETURN.]
Tietokoneen tulisi vastata
SEARCHING FOR C-64 WEDGE
LOADING
READY.

Kuulet levykeaseman hurisevan, ja punainen valo syttyy. Kuten huomaat, ohjelman lataaminen levykkeeltä on miltei samanlaista kuin kasetiltakin. Erona on tuo pilkku ja numero 8.

Kaikilla laitteilla, kuten kasettinauhureilla, levykeasemilla, kirjoittimilla ja modeemeilla eli puhelinliitännöillä, on Commodoren järjestelmässä tunnusnumerot. Nämä numerot on asetettu jokaisen laitteen sisään. Kasettinauhuri on aina laite 1. Kirjoitin on melkein aina numero 4. Levykeasema tulee Commodoren tehtaalta numerona 8. Kun siis käsket tietokoneen ladata ohjelman "C-64 WEDGE",8, käsket sen ladata sen laitteesta 8, joka on siis levyasema.

Kun kuvaruudussa lukee READY, kirjoita RUN. Levykeaseman punaisen valon tulisi jälleen syttyä, ja siitä tiedät, että uutta ohjelmaa ladataan. (Ensimmäisen Wedge-ohjelman osan tarkoituksena on ladata seuraava.) Sinun ei tarvitse kirjoittaa RUN enää uudelleen, vaan näet viestin, josta tiedät Wedgen latautuneen koneeseen:

DOS MANAGER (ja sitten version numero)
BY BOB FAIRBAIRN

Siihen asti kun sammutat koneen, Wedge on koneen muistissa sellaisessa osassa, että useimmat BASIC-ohjelmat eivät sitä häiritse. Se pysyy koneen muistissa, vaikka suorittaisitkin ”lämpimän käynnistyksen” RUN/STOP- ja RESTORE-näppäimillä.

Levykkeen hakemisto

Jos menet johonkin rakennukseen ja tahdot tietää, missä jokin toimisto sijaitsee, katsot rakennuksen huoneluetteloa. Levykkeessä sitä vastaa hakemisto. Se kertoo levykkeen nimen sekä ilmoittaa kaksimerkkisen tunnusnumeron, ohjelmien nimet ja tyhjien sivujen eli sektoreiden määrän. Tässä on näyte levykkeen hakemistosta.

```
0 ”1541 TEST/DEMO  ” ZX 2A
13  ”HOW TO USE”          PRG
5   ”HOW PART TWO”        PRG
4   ”VIC-20 WEDGE”        PRG
1   ”C-64 WEDGE”          PRG
4   ”DOS 5.1”             PRG
11  ”MAIL”                SEQ
47  ”DATA BASE”           PRG
579 BLOCKS FREE
```

Ensimmäinen numero ylhäällä on *levykeaseman numero*. Se on jäännös aiemmasta Commodoren levykeasemajärjestelmästä, jossa oli kaksi levykeasemaa, 0 ja 1. Koska Commodore 64 muistuttaa paljon edeltäjiään, se pitää edelleen 1541-levykeasemaa levykeasemana 0. Tulet huomaamaan, että tämä levykeasemanumero on hyödyllinen muiden levykekäskyjen yhteydessä.

Seuraavana samalla rivillä on levykkeen nimi, tässä tapauksessa ”1541TEST/DEMO”. Seuraavat kaksi kirjainta ZX ovat levykkeen tunnus. Levykeasema vaatii kultakin levykkeeltä oman *ainutlaatuisen* kaksimerkkisen tunnuksen. 2A viittaa

Commodoren levykäyttöjärjestelmän siihen versioon, joka kirjoitti levykkeen. Toistaiseksi sillä ei ole merkitystä.

Vasemmassa laidassa olevat numerot osoittavat, kuinka monta levykkeen sivua eli sektoria kuhunkin ohjelmaan on käytetty. Keskellä on ohjelman nimi, jota seuraa kolmikirjaiminen lyhenne. Tämä lyhenne osoittaa, onko kyseessä ohjelma vai jokin muu tiedosto. Vain sellaiset ohjelmat, joiden perässä on PRG, voidaan ladata suoraan levykkeeltä käskyllä LOAD. Jos nimeä seuraava lyhenne on SEQ tai REL (jotka tarkoittavat ''sekventiaalista'' ja ''relatiivista'' tiedostoa), se on tietoa sisältävä tiedosto eikä ohjelmatiedosto ja sitä voidaan käyttää vain sellaisesta ohjelmasta käsin, joka on erityisesti suunniteltu käyttämään näitä tietoja. (Katso lukua 5 Kuinka tietokone varastoi tietoa.)

Lopussa on sitten vielä käytettävissä olevien sivujen eli sektorien määrä.

Voit nähdä levykkeen hakemiston joko käyttämällä Wedgeä tai BASIC-käskyä. Olet jo oppinut, kuinka ohjelma ladataan levykkeeltä. Annat käskyn LOAD ja ohjelmanimen, jota seuraa pilkku ja laitenumero 8 (''8''). Muistanet, että käytit merkkiä '' >'' Wedge-käskyn kanssa. (Mukavuussyistä voit käyttää myös merkkiä @ Wedgen kanssa.) Kirjoita kumpi tahansa näistä käskyistä:

```
  LOAD "$",8
tai
  > $ (tai @ $)
```

Kumpikin käsky toimii samalla tavalla — niiden avulla näet levykkeen hakemiston. Jos käytät BASIC-muotoa — LOAD ''$'',8 — latautuu hakemisto tietokoneen muistiin. Nähdäksesi sen sinun on kirjoitettava LIST (ja painettava RETURN-näppäintä). Hakemiston lataaminen tällä tavalla tuhoaa aina kaiken, mitä tietokoneen muistissa on, myös siellä mahdollisesti olevat BASIC-ohjelmat.

On siis parempi käyttää Wedgen käskyjä > $ tai @ $, koska

Wedge siirtää hakemiston suoraan levyltä kuvaruutuun tuhoamatta muistissa olevaa ohjelmaa.

Jos levyke on tyhjä ja formatoimaton, et näe mitään hakemistoa, vaan levyke jää pyörimään ja saatat jopa kuulla muutaman kirskahtavan äänen. Ja sitten alkaa vilkkua punainen virhettä osoittava valo.

Levykkeen valmistelu

Uusi levyke voidaan valmistella käyttämällä joko BASICiä tai Wedgeä. Koska 1541-levykeasema on ''älykäs'' (siinä on oma mikroprosessori, muisti ja ohjausohjelma), se osaa käskyn saatuaan päättää, mitä sen tulee tehdä. Käskyt ovat ohjelman tai levykkeen nimen sisältävien lainausmerkkien välissä olevia kirjaimia, välimerkkejä ja numeroita. Voit valmistella levyn esimerkiksi seuraavalla tavalla käyttämällä näppäimistöltä kirjoittamiasi käskyjä:

```
OPEN 1,8,15
PRINT#1,''N0:LEVYN NIMI,TN''
```

Ensimmäinen kirjoittamasi käsky — OPEN 1,8,15 — on tapa kertoa levykeasemalle, että sen tulee valmistautua tiettyyn toimintaan, esimerkiksi levykkeen formatointiin ja nimeämiseen. Ensimmäinen OPEN-käskyn jälkeisistä numeroista (1) on *tiedoston numero*. Se voi olla mikä numero tahansa väliltä 0—255, mutta ei ole mitään syytä käyttää muita numeroita, jos ainoastaan valmistelet levykettä. Toinen numero (8) on laitenumero, mikä tarkoittaa, että kyseessä on levykeasema. Viimeinen numero on *toinen osoite*. Toisista osoitteista on lisää luvussa 5 Kuinka tietokone varastoi tietoa. Toistaiseksi sinun tarvitsee vain tietää, että numerolla 15 järjestelmä aina kertoo levykkeelle, että sen tulee valmistautua tiettyyn käskyyn.

Sinun kannattaa aina sulkea OPEN-käskyllä avatut tiedostot

koneen suoritettua halutut toimenpiteet, etenkin jos tulet käyttämään tiedoston sisältämiä tietoja ohjelmissasi. Se tapahtuu kirjoittamalla

```
CLOSE 1
```

Lisää OPEN- ja CLOSE-käskyistä on luvussa Kuinka tietokone varastoi tietoa.

Lainausmerkeissä olevat numerot ja kirjaimet muodostavat levykeaseman ymmärtämän koodin. "N" kertoo levylle, että tuleva erityistoimi on *uuden* levykkeen valmistelu eli formatointi. Seuraava merkki on *numero* 0. Se kertoo levykeasemalle, että käytät levykeasemaa 0. (Muistat kai, että levykeasemasi on aina numero 0, koska 1541 on yksöislevykeasema.) Kaksoispiste ":" on tarpeen erottamaan tämän käskyn levyn nimestä (joka tässä on LEVYN NIMI.)

Seuraavaa välimerkkiä, pilkkua, seuraa Commodoren levykkeiden vaatima kaksimerkkinen levykkeen tunnus. Sinun tulisi pyrkiä tekemään jokaisesta levyketunnuksesta ainutlaatuinen. Tässä esimerkissä levykkeentunnus on tietysti TN.

Wedgen edut levykkeenkäsittelykäskyissä ilmenevät siitä, kuinka yksinkertaisella tavalla se valmistelee uuden levykkeen.

```
>N0:LEVYN NIMI,TN
```

Wedgeä käyttäessäsi et tarvitse lainkaan käskyjä OPEN, PRINT# tai CLOSE uuden levykkeen formatoimiseen. Pane myös merkille, ettet tarvitse lainausmerkkejä käskyn kirjainten, numeroiden ja nimen ympärillä.

Levykkeen valmistelu vie muutaman minuutin, vaikka sinusta saattaa tuntua, että levykeasema toimii ikuisuuden ja punainen valo palaa jatkuvasti. Kaikki sujuu kyllä niin kuin pitääkin. Kun levykeasema on lopettanut valmistelunsa, punainen valo sammuu ja kuvaruudussa lukee READY. Tarkista levykkeesi katsomalla hakemistoa. Siinä pitäisi lukea näin:

0 ”LEVYN NIMI ” TN 2A
664 BLOCKS FREE

Jos olet valinnut jonkin toisen nimen ja tunnuksen, niiden tulisi tietysti olla näkyvillä.

Lataaminen levykkeeltä

Ohjelman lataaminen levykkeeltä on suunnilleen yhtä helppoa kuin kasetilta lataaminen. On tosin pari poikkeusta. Muista, että voit ladata ohjelman levykkeeltä vain jos se todella on ohjelma, jota osoittaa lyhenne PRG. Voit aina katsoa levykkeen hakemistoa ja käyttää sitten käskyä:

LOAD ”OHJELMANNIMI”,8

Jos haluat kirjoittaa ohjelman koko nimen, sinun on kirjoitettava se täsmälleen samanlaisena kuin se näkyy hakemistossa. Jos et tee niin, tietokone vastaa FILE NOT FOUND (tiedostoa ei löydy) ja levykeaseman etupaneelin punainen valo vilkkuu säännöllisesti. Jos olet laiska tai haluat vain säästää aikaa, voit käyttää oikotietä.

Voit ladata kasetilta ohjelman käyttämällä vain nimen alkukirjaimia, ja sama pätee levykeohjelmiin. Älä kuitenkaan jätä pois viimeistä lainausmerkkiä, kirjoita vain tähden eli ”*” ennen sitä.

Commodoren suunnittelijat ovat osanneet käyttää tähteä helpottaakseen levykeasemien käyttöä. Tässä on ohjeita sen käyttämiseen:

* Kun aloitat levykeaseman käytön, LOAD ”*”,8 lataa ensimmäisen hakemistossa olevan ohjelman. (Tämä on erityisen kätevä, jos esimerkiksi Wedge on aina ensimmäisenä.)

* Minkä tahansa ohjelman latauksen jälkeen LOAD ”*”,8 lataa aina saman ohjelman uudelleen, vaikka se ei olisikaan hakemistossa ensimmäisenä.

* Tähden käyttäminen missä tahansa ohjelmannimessä lataa ensimmäisen sellaisen hakemistossa olevan ohjelman, jonka nimi sopii tähden edellä oleviin kirjaimiin. Esimerkiksi LOAD ''OHJE*'',8 lataa sellaisia ohjelmia kuin OHJELMA, OHJELMA YKSI, OHJELMA KAKSI, OHJEKIRJA jne.

Wedge tekee mahdolliseksi ladata ohjelmia kahdella tavalla. (Myös Wedgen käskyissä voi käyttää tähteä.)

Voit ladata ohjelman käskyllä

/OHJELMANNIMI

Ensimmäinen merkki on siis vinoviiva, ja se löytyy samasta näppäimestä kuin kysymysmerkki. Se lataa ohjelman ja aloittaa sen suorittamisen, kun annat käskyn RUN.

Toisella Wedge-käskyllä voit automaattisesti ladata ja suorittaa ohjelman:

[Nuoli ylös] OHJELMANNIMI

Et tietenkään kirjoita sanoja ''Nuoli ylös'', vaan painat toisella ylärivillä tähden ja RESTORE-näppäimen välissä olevaa näppäintä, jossa on ylös suunnatun nuolen kuva. Kun BASIC-ohjelma on ladattu, tietokone aloittaa automaattisesti ohjelman suorituksen.

Eräät ohjelmat vaativat tietynlaisen lataamisen, jotta ne latautuisivat tiettyyn paikkaan koneen muistissa. Näitä ohjelmia ei yleensä ole kirjoitettu BASICillä. Tämä LOAD-käsky on tällainen:

LOAD ''OHJELMANNIMI'',8,1

Viimeinen numero 1 käskee tietokoneen ladata ohjelman muistin sellaiseen paikkaan, jossa se voi toimia kunnolla, eikä tavanomaiseen BASIC-ohjelmien paikkaan. Jos olet käyttänyt Commodore PET tai CBM-tietokonetta, tiedät, että tämä on erilainen käsky kuin mihin olet tottunut. (Noissa koneissa ei tarvita

erillistä LOAD-käskyä, vaan ohjelmat latautuvat aina oikeaan paikkaan.)

Vastaava Wedge-käsky olisi tällainen:

```
% ''OHJELMANNIMI''
```

Miltei aina sinua kehotetaan erikseen käyttämään tätä LOAD-käskyä sellaisissa ohjelmissa, jotka sitä vaativat. Jos et saa tällaista kehotusta, käytä edellä esitettyjä tavanomaisia tai Wedge-käskyjä.

Tallentaminen ja varmistaminen

Ohjelma tallennetaan levykkeelle aivan samalla tavalla kuin kasettinauhallekin. Ainoa ero on se, että on lisättävä pilkku ja laitenumero 8.

```
SAVE ''OHJELMANNIMI'',8
```

Jos käytät Wedgeä, tallenna ohjelma käyttämällä ''nuoli vasemmalle'' -näppäintä, joka on näppäimistön vasemmassa yläkulmassa.

```
[Paina nuoli vasemmalle] ''OHJELMANNIMI''
```

Voit käyttää myös levykeaseman numeroa (0) tallentaessasi ohjelmaa.

```
SAVE ''0:OHJELMANNIMI'',8
```

Kasettinauhaa käyttäessäsi voit aina kelata nauhan alkuun ja nauhoittaa uuden version vanhan päälle. Koska levykettä ei voida kelata takaisin alkuun, käytetään tiettyä merkkiä silloin, kun halutaan tallentaa uusi ohjelma vanhan samannimisen päälle. Tämä merkki on ''@''. Sitä käytetään ennen ohjelmannimeä ja se erotetaan ohjelmannimestä kaksoispisteellä.

```
SAVE ''@:OHJELMANNIMI'',8
```

tai

```
SAVE ''@0:OHJELMANNIMI'',8
```

Muista kirjoittaa nimi aivan samanlaisena kuin se esiintyy levykkeen hakemistossa, muutoin ohjelma ei tallennu entisen ohjelman päälle.

Voit varmistaa, onko ohjelma tallentunut levykkeelle oikein. Käytä käskyä VERIFY ohjelmannimen ja laitenumero 8:n kanssa.

```
VERIFY "OHJELMANNIMI",8
```

Koska tähteä voidaan käyttää korvaamaan levykeaseman viimeksi käyttämää nimeä, voit tallentaa ensin ohjelman sen nimellä ja sitten välittömästi varmistaa asian tähden avulla.

```
VERIFY "*",8
```

Jos ohjelma ei ole tallentunut oikein, näet kuvaruudussa ilmoituksen VERIFY ERROR.

Levyketiedostojen poistaminen

Vaikka yhteen levykkeeseen mahtuukin lähes 170 000 merkkiä, levykkeet täyttyvät nopeasti ohjelmien ensimmäisistä versioista, sekalaisista tiedostoista ja ohjelmista, joista et enää välitä. Voit poistaa nämä ohjelmat ja tiedostot levykkeestä tehdäksesi tilaa muulle.

Kuten formatointi eli levykkeen valmistelu myös ohjelmien ja tiedostojen poistaminen voidaan toteuttaa joko BASICin tai Wedgen avulla.

```
OPEN 1,8,15
PRINT#1,"S0:OHJELMANNIMI",8
```

tai

```
> S0:OHJELMANNIMI
```

(S-kirjaimen jälkeen tulee siis numero 0.)

Levykkeen järjestäminen

Toisin kuin kasetti, levyke on monimutkainen järjestelmä, jossa ohjelmia ja niiden osia on siellä täällä raidoilla ja sektoreilla, ja koko järjestelmää pitää yllä ns. Block Availability Map (BAM) eli vapaiden sektorien kartta. Toisinaan käy niin, että ohjelmiin ja tiedostoihin käytettyjen sivujen ja vapaiden sivujen yhteismäärä ei olekaan tuo maaginen 664 sivua.

Tässä tapauksessa levyke on kovin sekaisin, vaikka se saattaakin edelleen toimia kelvollisesti. On kuitenkin olemassa vaara, että epäjärjestys pahenee ja saatat kadottaa arvokkaan ohjelman tai tiedoston. Voit järjestää levykkeen uudelleen joko BASIC- tai Wedge-ohjelman käskyllä. Tällaista uudelleenjärjestelyä kannattaa tehdä aika ajoin, ja aina silloin kun huomaa, että ongelmia saattaa syntyä.

```
  OPEN 1,8,15
  PRINT#1,''V''
tai
  PRINT#1,''V0''
  Wedge-käsky on:
  > V
tai
  > V0
  (Jälleen V:n jälkeen tulee numero 0.)
```

Levykevirheet

Levykeasemassa on jotakin vialla aina, jos punainen valo vilkkuu säännöllisesti. On tapahtunut levyvirhe, vaikka et voikaan tietää millainen se virhe on, koska kuvaruutuun ei ilmaannu mitään selitystä. Valo kuitenkin jatkaa vilkkumistaan, kunnes olet keksinyt, millaisesta virheestä on kyse. Mitä on tehtävissä?

Ratkaisu ei ole helppo, jos et käytä Wedgeä.

Yritä ensiksi aiheuttaa tahallinen virhe yrittämällä ladata ohjelma, jota levykkeellä ei ole.

```
LOAD "MMFPLTSK",8
```

Todennäköisesti sinulla ei ole tämännimistä ohjelmaa, joten punainen valo alkaa vilkkua. Kirjoita nyt seuraava pieni ohjelma aivan mallin mukaan ja suorita se käskyllä RUN.

```
10 OPEN 1,8,15
20 INPUT# 1,A$,B$,C$,D$
30 PRINT A$,B$,C$,D$
40 CLOSE 1
```

Kuvaruudussa pitäisi nyt lukea:

```
62 FILE NOT FOUND 00 00
```

Ensimmäinen numero on virheen tunnus, ja sen oikealla puolella on virhe kirjaimin, FILE NOT FOUND (tiedostoa ei löydy). Seuraavat kaksi lukua ovat sen uran ja sektorin numerot, joissa virhe tapahtui. (Niitä ei käytetä FILE NOT FOUND -virheen yhteydessä.)

Tämän ohjelman kirjoittaminen on kuitenkin turhan hankalaa, eikä sitä voi aina käyttää, jos koneen muistissa on jo ohjelma. Paras tapa huomata levykevirheet on siis käyttää Wedgeä. Kirjoita vain merkki " >" aina kun punainen valo vilkkuu. Näet samat tiedot, virheen tunnuksen, virheilmoituksen ja sen uran ja sektorin numeron, jossa virhe on tapahtunut.

Huomautuksia levykeaseman käytöstä

Commodoren levykeasemaa on paljon kätevämpi käyttää kuin kasettinauhuria. Ja koska se nopeuden lisäksi tarjoaa muitakin etuja, se tekee tietokonejärjestelmästäsi varsin tehokkaan.

Mikään järjestelmä ei kuitenkaan ole täydellinen, eikä 1541-levykeasema ole poikkeus säännöstä. Kannattaa siis ottaa huo-

mioon eräitä seikkoja.

* Jos unohdat sulkea levykeaseman luukun, kuvaruutuun ilmestyy virheilmoitus FILE NOT FOUND. Vaikka suljetkin luukun, saman ohjelman lataaminen ei ehkä onnistu, vaan näet taas virheilmoituksen. Siinä tapauksessa avaa luukku, ota levyke pois ja laita se sitten takaisin. Yleensä ongelma poistuu.

* Jos haluat käyttää kahta levykeasemaa, täytyy toiselle niistä antaa jokin muu laitenumero kuin 8. Ellei sinulla ole kokemusta elektroniikasta, pyydä myyjää muuttamaan laitenumero, koska se vaatii elektronisen kytkennän katkaisemista laitteen sisältä. Kun olet kytkenyt molemmat levykeasemat koneeseesi, muista aina kytkeä virta *viimeiseksi* tietokoneeseen. Tällä tavalla se tunnistaa kummankin levykeaseman oikein. Jos unohdat tämän, paina yhtäaikaisesti näppäimiä RUN/STOP ja RESTORE ”herättääksesi” koneen.

* Ole varovainen, jos käytät samaa levykettä sekä 1541:ssä että alkuperäisessä Commodore 4040 -kaksoislevykeasemassa. Niiden tulisi sopia yhteen, mutta asianlaita ei aina ole niin. Ongelma piilee eräiden 1541-levykeasemien levykkeenkäyttöjärjestelmässä. Asiantunteva tietokonemyyjä osaa paikallistaa tämän vian. Jos et ole varma siitä, että oma 1541:si ei kärsi tästä viasta, älä käytä näitä levykeasemia sekaisin. Toisin sanoen, jos levyke on valmisteltu 1541-levyasemalla, älä kirjoita sille tietoja 4040-levyasemilta. Tulos voi olla tuhoisa.

* Jotkin 1541:t tulevat Commodoren tehtaalta epätäydellisesti säädettyinä. Ongelmana on yleensä nopeus. Tarkista aina levykeasemasi 1541:n kanssa tulevalla Performance Test -ohjelmalla. Kokeile sitten levykettäsi jossain toisessa 1541:ssä. Jos lataus sujuu hyvin, on kaikki luultavasti kunnossa. Muussa tapauksessa ongelmana voi olla huono säätö. Tavallisesti kyseessä on levykkeen pyörimisnopeus. Vaikka korjaus onkin mahdollista tehdä itse (levykeaseman pohjassa on impulssikaavio ja pieni säädin), sinun tulisi viedä levykeasema myyjän säädettäväksi.

* Jos sinun täytyy jostakin syystä viedä levykeasemasi myyjälle korjausta tai säätöä varten, käske hänen samalla tarkistaa

metallinen hihnapyörä, joka yhdistää vetoakselin hihnalla moottoriin. Joissakin levykeasemissa tämä hihnapyörä saattaa irrota ja pudota paikaltaan. Jos satut tallentamaan ohjelmaa tai tietoa juuri silloin, kun se putoaa, voi levykkeesi tuhoutua ja tärkeitä tietoja kadota. Pyydä myyjää varmistamaan, että tämä hihnapyörä on tukevasti kiinni, ja päinvastaisessa tapauksessa korjaamaan asia.

* Älä koskaan tuki 1541:n muovikuoressa olevia tuuletusaukkoja. Älä siis kasaa paperia ja kirjoja levykeaseman päälle siten, että ne peittävät tuuletusaukot, äläkä aseta levykeasemaa huovalle tai muulle sellaiselle pinnalle, joka voi tukkia tuuletusaukot alhaaltapäin. Levykeasema kuumenee helposti liikaa, jolloin herkkiä elektronisia osia laitteen sisältä voi tuhoutua.

Commodoren ohjelmayhteensopivuus

Commodore 64 -laitteiston mukavimpia ominaisuuksia on se, että siinä voi käyttää muillekin Commodoren tietokoneille, VIC:lle, PET:lle ja CBM:lle, kirjoitettuja ohjelmia. Sellaisia ohjelmia, joita Commodore 64:ssä *ei* voi suorittaa, ovat mm.:

* useimmat jollakin muulla kielellä kuin BASIC:llä kirjoitetut ohjelmat
* BASIC-käskyt, jotka viittaavat itse koneeseen, tavallisesti PEEK- ja POKE-käskyt
* ohjelmat, jotka on kirjoitettu VIC:in 22-merkkistä kuvaruutua varten
* ohjelmat, jotka on kirjoitettu Commodore 4.0 BASIC:in käskyillä.

Sellaisia Commodore 64 -koneella tehtyjä ohjelmia, jotka eivät toimi kunnolla PET:issä, VIC:issä tai CMB:ssä, ovat mm.:

* useimmat jollain muulla kielellä kuin BASIC:illä kirjoitetut ohjelmat
* monet sellaiset ohjelmat, joissa käytetään värikäskyjä, ja miltei kaikki sellaiset ohjelmat, joissa on käytetty 64:n äänitekniikkaa.

On huomattava myös seuraavaa:

* Ohjelmat, joissa tarvitaan funktionäppäimiä, eivät sovi PETiin eivätkä CBM:ään, mutta saattavat toimiva VICissä.

* Ohjelmat, joissa tarvitaan yli 22 merkkiä rivillä kuvaruudussa, eivät toimi VICissä.

* Ohjelmat, jotka vaativat yli 32 kiloa muistia, eivät toimi VICissä (edes muistilaajennuksen kera), PETissä tai CBM:ssä.

Ladataksesi eräitä PETin ja CBM:n ohjelmia Commodore 64:ään tarvitset ohjelman nimeltä "PET Emulator". Eräs sellainen ohjelma on saatavissa Commodorelta. Bob Fairbairn on suunnitellut ohjelman siten, että sen avulla voidaan suorittaa PETin ohjelmia, jotka on kirjoitettu Commodore 2.0 BASICillä, joka on lähimpänä 64-laitteiston käyttämää BASIC-versiota. PET Emulatorin avulla 64 voi jopa käyttää PETin tyyppisiä ääniefektejä, siihen sisältyy Wedge DOS Manager -ohjelma ja se toimitetaan levyversiona.

PET Emulatorin avulla ei voida suorittaa kaikkia PETin ohjelmia (esimerkiksi suosittua VisiCalc-ohjelmaa). Sinun on kokeiltava nähdäksesi, mitkä ohjelmat voidaan suorittaa.

Kolme nopeaa käskyä riittää lataamaan Commodore 64:llä kirjoitetut ohjelmat PETiin tai CBM:ään. Näille koneille on kirjoitettava käskyt:

```
POKE 41,8
POKE 2048,0
NEW
```

Tämän jälkeen lataat 64:n ohjelman. Mutta muista, että kaikkia 64:n ohjelmia ei voida kunnolla suorittaa PETissä ja CBM:ssä.

Tietokoneen käyttö laskimena

On mahdollista käyttää Commodore 64:n tehokkaita matemaattisia ominaisuuksia tarvitsematta kirjoittaa ohjelmaa sitä

varten. Tavanomaista taskulaskinta käyttäessäsi kirjoittaisit yhtälön alusta alkaen ja päättäisit sen yhtäläisyysmerkillä (=), esimerkiksi

```
12+484.25=
```

Ainoa ero tietokonetta käyttäessäsi on se, että voit jättää yhtäläisyysmerkin pois ja pyytää sen sijaan vastausta PRINT-käskyllä.

```
PRINT 12+484.25 [ja paina RETURN]
```

Vastaus ilmaantuu kuvaruutuun.

Voit myös käyttää *muuttujia* suoraan näppäimistöltä. Jos et tiedä, mitä muuttujat ovat, niin BASIC-ohjelmointia käsittelevässä luvussa saat sen tietää. Kirjoita tämä:

```
A=484.25
PRINT 12+A
```

Käytä tavanomaisia merkkejä plus (+) ja miinus (—) yhteen- ja vähennyslaskuissa. Kertolaskuissa käytetään tähteä (*) ja jakolaskuissa vinoviivaa (/). Negatiivisia lukuja edeltää miinusmerkki ja eksponenttilukuja ylöspäin suunnattu nuoli. Neliöjuuret saat käskyllä SQR. Tietokone ymmärtää myös merkit ”pienempi kuin” ja ”suurempi kuin” (<>), sekä niin sanotut loogiset operaattorit AND, OR ja NOT. On myös trigonometrisiä funktioita: SIN (sini), COS (kosini), TAN (tangentti), ATN (arcus tangentti) ja LOG (logaritmi). Näitä käytetään siten, että niiden kohdeluvut on pantu sulkeisiin. Esimerkiksi neliöjuuri kaksi saadaan käskyllä:

```
PRINT SQR(2)
```

Sulkeita voidaan käyttää myös samalla tavalla kuin laskettaessa

paperilla eli merkitsemään tiettyä laskutoimitusta yhtälön sisällä.

```
PRINT (12+484.25)*53
```

Tämä yhtälö laskee ensin yhteen 12 ja 484.25 sekä kertoo sitten tuloksen 53:lla.

Kaksi varoitusta: käytä aina desimaalipilkun paikalla *pistettä* ja esimerkiksi luku miljoona kirjoitetaan 1000000 eikä 1.000.000. Älä käytä suoraan näppäimistöltä yhtälöitä, joissa on yli 80 merkkiä eli kaksi kuvaruuturiviä.

Vaikka et aikoisikaan käyttää konettasi laskimena tai ohjelmoida sitä ”numeromurskaimeksi”, älä silti pelästy näitä yhtälöitä. Pidä vain mielessäsi, että ne ovat siellä, jos sattuisit joskus tarvitsemaan.

On nopeampikin tapa antaa käsky PRINT. Voit käyttää kysymysmerkkiä (?) sen sijaan että kirjoittaisit joka kerta PRINT. Kokeile. Saat tietää siitä lisää BASIC-ohjelmointia käsittelevässä osuudessa, ja tarkista sitä koskeva varoitus tietojen käsittelyä koskevassa luvussa.

Kuinka paljon muistia on jäljellä?

Tiedät jo, että Commodore 64:ssä on muistitilaa yli 64 000 tavua eli merkkiä. Kytkiessäsi virran olet todennäköisesti huomannut ilmoituksen, jonka mukaan niistä on 38 911 tavua käytettävissäsi BASIC-kieliseen ohjelmointiin.

Kun ohjelmoit tai olet ladannut ohjelman muistiin nauhalta tai levykkeeltä, voit kysyä tietokoneelta, kuinka paljon muistitilaa on sillä hetkellä käytettävissäsi.

```
PRINT FRE(0)
```

Sana FRE tarkoittaa vapaita tavuja. Sulkujen sisällä voi olla mikä tahansa numero tai kirjain. (Koska sillä ei ole väliä, sitä

kutsutaan ”valemerkiksi” englanniksi ”dummy”.) Kuvaruutuun ilmaantuva luku on vapaiden tavujen määrä vähennettynä kolmella, sillä käsky PRINT FRE(0) vie itse kolme tavua.

Jos ruudussa näkemäsi luku on negatiivinen (eli sitä edeltää miinusmerkki), on sinun tehtävä pieni laskutoimitus ja vähennettävä se luvusta 65 536. Tämä luku on oikeastaan se 64K eli kiloa, josta koko ajan puhutaan, koska yksi kilo (eli tuhat tavua tietokoneslangissa) on itse asiassa 1 024 tavua.

(Tässä on ilmeisesti erehdys Commodore 64:n suunnittelussa. Ohjelma joka ilmoittaa vapaiden tavujen määrän, on sama kuin PETissä, mutta 64:ssä on kaksi kertaa niin paljon muistitilaa.)

Varo lainausmerkkejä

Hyvä tapa tutustua Commodore 64:n korjailutoimintoihin (CRSR oikealle ja vasemmalle sekä INST/DEL -näppäimiin) on harjoitella kirjoittamaan näppäimistöllä. Vielä parempi tapa on kirjoittaa koneeseen kaikki tämän kirjan BASIC-ohjelmaesimerkit. Harjoittelu on ainoa tapa tottua koneen erinomaiseen kuvaruutueditoriin (kuvaruudussa tapahtuvaan korjailuun). Jos saat tilaisuuden verrata sitä muihin koneisiin, huomaat pian, kuinka hyvä 64 on.

On helppoa joutua tilanteeseen, jossa luulet juuttuneesi ja vielä rikkoneesi koneenkin, vaikka niin ei ole käynyt. Näin tapahtuu, kun olet kirjoittanut vain yhden lainausmerkin tai parittoman määrän lainausmerkkejä. Kirjoittaessasi riviä kirjoitat vahingossa tai tahallaan myös lainausmerkin. Muutat sitten mielesi ja käytät CRSR-näppäintä suorittaaksesi korjauksen, mutta näetkin hassunnäköisen merkin.

Olet niin kutsutussa ”lainausmerkkitilassa”. Se merkitsee sitä, että seitsemän kahdeksasta korjaustoiminnosta toimii eri lailla kuin normaalisti. CLR (kuvaruudun tyhjennys), HOME (kursorin siirto alkuasemaan), CRSR oikealle, vasemmalle, ylös

ja alas sekä INST (kirjoita väliin) tuottavat kaikki tulokseksi näitä hassuja merkkejä sen sijaan, että toimisivat normaalisti. Vain DEL (merkin poisto) toimii edelleen totutusti, niin että voit poistaa kysymysmerkin tai muun kirjoittamasi. Tämän aiheuttajasta puhutaan BASIC-ohjelmointia käsittelevässä luvussa PRINT-käskyn yhteydessä. Lyhyesti sanottuna, kun näitä korjailutoimintoja käytetään lainausmerkeissä ja PRINT-käskyn kanssa BASIC-ohjelmissa, ne toimivat ohjelmaa suoritettaessa.

Vaikka DEL-näppäin poistaakin erehdyksessä kirjoittamasi, olet edelleen lainausmerkkitilassa. Ja pysyt siinä, kunnes joko kirjoitat toisen lainausmerkin tai painat RETURN-näppäintä. Tai jos et halua kirjoittaa sitä koneen muistiin, voit painaa näppäintä SHIFT ja RETURN yhtä aikaa. (Tietokone ei pidä SHIFT ja RETURN -näppäimiä merkkinä rivin kirjoittamiseksi koneen muistiin.)

Tuntuu kuitenkin ilmaantuvan toinen ongelma, jos palaat takaisin riville ja *haluat* päästä lainausmerkkitilaan ohjelman niin vaatiessa. Ratkaisu on sijoittaa kohdistin siihen paikkaan, jossa ensimmäisen lainausmerkin tulisi olla, ja kirjoittaa siihen lainausmerkki.

Kun saat hiukan kokemusta, lainausmerkkitila ei tunnu lainkaan niin oudolta kuin nyt siitä lukiessasi. Ratkaisu löytyy, kunhan muistat, että olet aina lainausmerkkitilassa kirjoitettuasi yhdelle riville *parittoman määrän* lainausmerkkejä.

Paljonko kello on?

Jokaisessa Commodore 64:ssä on kello. Se lähtee käyntiin kytkiessäsi virran ja pysähtyy sammuttaessasi koneen.

Kelloa merkitsevät kirjaimet TI ja TI$. Näitä kutsutaan ”varatuiksi muuttujiksi”, koska ne merkitsevät muuttuvaa tietoa ja kun toisaalta ilmaisuja TI ja TI$ ei voida käyttää millään muulla tavoin. Tietokone pitää ilmaisua TI numerona ja TI$

merkkijonona. (Lisää tavallisista ja merkkimuuttujista jäljempänä.)

Nähdäksesi tietokoneen kellonajan, kirjoita:

```
PRINT TI$
```

Sinun tulisi nähdä kuusi numeroa, esimerkiksi 012236. Ajattele näitä kuutta numeroa kolmena kahden numeron ryhmänä — 01 22 36. Ensimmäiset kaksi numeroa ilmaisevat, kuinka monta tuntia sitten tietokoneeseen kytkettiin virta. Seuraavat numerot tarkoittavat minuutteja ja viimeiset kaksi sekunteja. Tietokoneessa on siis ollut virta kytkettynä tunnin, 22 minuuttia ja 36 sekuntia.

Tietokoneen sammuttaminen ja virran takaisin kytkeminen ei ole kovin kummallinen keino kellonajan asettamiseen. Sen voi tehdä näppäimistöltäkin. Kirjoita:

```
TI$ = ''000000''
```

Anna uudelleen käsky PRINT TI$. Numeroiden tulisi olla lähellä 000000:aa. Kokeile tätä:

```
TI$ = ''020510''
```

Käsky PRINT TIS antaa nyt tulokseksi vähän yli 2 tuntia, 5 minuuttia ja 10 sekuntia, koska asetit kellon siihen aikaan.

Voit käyttää muuttujaa TI$ tällä tavoin tai sitten BASIC-ohjelmassa voidaksesi lukea kellonajan ja asettaaksesi sen toiseen aikaan ohjelmassa. Sinun on muistettava käyttää lainausmerkkejä kellonajan ympärillä asettaessasi aikaa, muuten näet virheilmoituksen.

Tätä kelloa mainitaan usein tosiaikakelloksi, koska se käy *todellista aikaa.* Se on 24 tunnin kello, joten jos asetat sen aikaan 00 00 00 keskiyöllä, se näyttää 12 00 00 iltapäivän alussa ja 23 00 00 yöllä.

Kannattaa muistaa, että tosiaikakello *pysähtyy* silloin, kun

lataat tai tallennat ohjelmia tai tietoa kasetilta. Itse asiassa kello pysähtyy aina kun kuvaruutu tyhjenee, onpa syy mikä tahansa. Sinun ei siis tule käyttää kelloa kasettitoiminnan aikana.

Tietokoneessa on toinenkin kello, vaikka se ei ole niin hyödyllinen kellonajan kannalta. Sitä kutsutaan ''kotvakelloksi'', ja yksi kotva (jiffy) on 1/60 sekuntia. Kello alkaa käydä nollasta ja jatkaa, kunnes pääsee 51839999 kotvaan, jolloin se taas aloittaa nollasta. Jos käytät kotvakelloa, älä huolehdi siitä, että se saattaisi aloittaa nollasta kesken sinun ajanmittauksesi. Viisikymmentäyksi miljoonaa kotvaa on yli 230 tuntia.

Voit nähdä ajan kotvina kirjoittamalla näin:

```
PRINT TI
```

Kuvaruutuun ilmaantuva luku on kotvien määrä siitä hetkestä, kun koneeseen kytkettiin virta. Jos jaat sen kuudellakymmenellä, saat koneen käynnissäoloajan sekunteina. Jos jaat sen 3600:lla (60 kertaa 60), saat ajan minuutteina jne.

Kotvakelloa ei voida nollata asettamalla TI nollaksi vaan asettamalla TI$:n arvoksi ''000000''.

3

Johdatus ohjelmointiin

Ei ole kovinkaan kauan siitä, kun henkilöä, joka osasi ohjelmoida tietokoneita, pidettiin nerona tai pahimmassa tapauksessa jonkinlaisena poppamiehenä. Todellisuudessa nämä henkilöt vain tunsivat tietokoneen mahdollisuudet ja osasivat jakaa ongelmat pienemmiksi osaongelmiksi. He tunsivat myös tietokoneiden kielen kuten oman äidinkielensä.

Nykyisin ei ohjelmointitaitoisia henkilöitä enää pidetä mitenkään erikoisina, ainoastaan taitavina. Tietokoneita osaavat ohjelmoida monet muutkin kuin ne, jotka ovat ottaneet sen ammatikseen. Ohjelmointi on taito, joka voi rikastuttaa elämääsi ja jota voidaan käyttää niin yksityiselämässä kuin ammatissakin. Ja pelkästään älyllisenä harjoituksenakin harva taito on yhtä palkitseva kuin ohjelmoinnin opiskelu.

Jo oppimalla muutamia yksinkertaisia sanoja alat kehittyä tässä taidossa ja huomaat, kuinka saat tietokoneen tekemään hyvinkin vaikuttavia asioita.

Luettuasi tämän osan tiedät jo jonkin verran siitä, mitä tietokoneohjelma oikein on, kuinka tietokone varastoi sanoja ja numeroita ja kuinka käsky PRINT toimii.

Jos olet aloitteleva ohjelmoija, tulet yllättymään siitä, kuinka

paljon saat aikaan vain muutaman sanan opittuasi.

BASIC-ohjelmarivit ovat tässä kirjassa samalla esimerkkejä. Aina kun haluat nähdä, kuinka jokin niistä toimii, sinun on tyhjennettävä tietokoneen muisti ja siis valmistettava se uudelle ohjelmalle. Ellet saa muita ohjeita, kirjoita NEW ja paina RETURN-näppäintä aina ennen uuden esimerkin kirjoittamista.

Kun opiskelet BASICiä, teet myös melko varmasti virheitä. Se on aivan tavallista, eikä sinun tulisi masentua kohdatessasi ongelmia. Aina kun teet virheen, kirjoita ohjelmaesimerkki uudelleen ja tee uusi yritys.

Seuraavissa luvuissa on esimerkkejä virheilmoituksista (joka englanniksi on ERROR). Nämä ERROR- eli virheilmoitukset ovat tietokoneen tapa kertoa sinulle, että olet tehnyt virheen. Tekstissä kuvaillaan jokainen virhe ja sen mahdolliset syyt sekä annetaan vihjeitä sen korjaamiseksi. Virheilmoituksia saavat toisinaan aikaan hyvinkin pienet seikat, ja selvität parhaiten syyt, kun muistat, että tietokone ottaa kaiken hyvin täsmällisesti ja kirjaimellisesti.

BASIC-kieli

Commodoren tietokoneiden yleisimmin käyttämä kieli on nimeltään BASIC. Sen kehittivät John Kemeny ja Thomas Kurtz 1960-luvulla inhimillisemmäksi vaihtoehdoksi tietokoneohjelmointiin. Sen sanat ovat yleensä selkeää englantia tai ainakin englannin sanojen johdannaisia.

Nimi BASIC tulee sanoista Beginner's All-purpose Symbolic Instruction Code eli aloittelijan kaikkiin tarkoituksiin sopiva symbolinen ohjekoodi. Se on kokoelma yksinkertaisia ja hyödyllisiä käskyjä mutta se ei ole kaikkein tehokkain tietokonekieli. Tiettyihin tarkoituksiin on olemassa parempiakin. Eräs niistä, FORTRAN, sopii erityisesti tietojen tallentamiseen ja lasku-

toimituksiin. COBOL-kieli taas soveltuu hyvin tilastojen laatimiseen. LISP-nimellä tunnettu kieli sopii puolestaan jonojen laatimiseen ja korjailemiseen sekä tekoälykokeisiin.

Kaikkien kielten tapaan BASICissakin on oma *sanasto* ja *kielioppi,* jonka mukaan sanoja tulee käyttää.

BASIC soveltuu hyvin tietokoneisiin totutteluun ja käy yleiskieleksi ohjelmointiin. Miltei mikä ohjelma tahansa on mahdollista kirjoittaa BASICin avulla, mutta jokin toinen kieli saattaa joskus vaatia vähemmän muistitilaa tai olla nopeampi.

Commodoren tietokoneissa käytetty BASIC-versio on ns. BASIC-tulkki. Se tarkoittaa sitä, että tietokone tarkastelee kutakin ohjelmakohtaa askel askeleelta ja tekee niin kuin siinä käsketään. On olemassa myös BASIC-kääntäjä. Se muuntaa ohjelman numerosarjaksi, jonka kone kykenee suorittamaan paljon nopeammin. BASIC-kääntäjällä kirjoitetut ohjelmat ovat suoritettavissa paljon nopeammin kuin Commodoressasi olevalla versiolla.

Lisätietoja Commodore 64 -laitteistolle sopivasta BASIC-kääntäjästä saat kirjan lopussa olevasta luvusta.

Commodoren BASIC on alunperin Microsoft-yhtiön kehittämä versio, joka kirjoitettiin nyt jo valmistuksesta pois jääneelle MITS Altair -tietokoneelle. Microsoft BASICista on ajan mittaan tullut lähes standardimainen pienten tietokoneiden BASIC-kielen versio.

On kuitenkin eroja niissä BASIC-versioissa, joita Microsoft on suunnitellut TRS—80:lle, Apple II:lle ja muille koneille. Useimmat BASIC-käskyt ovat samat kaikissa koneissa, mutta on merkittäviä erojakin. Commodoren Microsoft BASICillä kirjoitettu ohjelma ei siis välttämättä toimi kunnolla Applessa ja päinvastoin.

Commodore on myös muokannut Microsoft BASICiään vuosien mittaan, ja jokainen tietokonesukupolvi eroaa hiukkasen aiemmasta. Esimerkiksi varhaisten Commodore PET -tietokoneiden BASICissä oli harmittavia virheitä. Kun Commodore sitten toi markkinoille parannetun version PETistä, se paranteli myös BASICiä korjatakseen joitakin näistä virheistä. Commo-

dore 64 -koneiden BASIC on lähimpänä tätä paranneltua versiota. Se on nimeltään V2 eli versio numero 2.

Mikä on ohjelma?

Tietokoneohjelma on joukko toimintoja, jotka tietokoneen tulisi tehdä. Ohjelma siis ottaa suuren tehtävän ja pilkkoo sen pienemmiksi osiksi.

Esimerkkinä on ohje — eli ohjelma — pakastetun aterian valmistamiseksi.

1. Lämmitä uuni 200-asteiseksi.
2. Ota pakastettu ateria pakkauksesta.
3. Laita se uuniin.
4. Odota 45 minuuttia.
5. Avaa kelmu ja ryhdy syömään.

Näin saattaa lukea pakkauksessa. Todellisuudessa ehkä kuitenkin haluat tehdä asiat eri tavalla. Saatat vaikka tarkistaa ensiksi, että uuni lämpiää kunnolla. Jos se ei lämpiä, otat selville, missä on vikaa. Tämä pakasteateriaohjelma voidaan siis kirjoittaa uudelleen vastaamaan sellaisiin kysymyksiin kuin ”onko uuni toimintakunnossa?”

Myös tietokoneella voidaan suorittaa tällaisia kokeita, esittää kysymyksiä ja tehdä päätöksiä.

Numerointi ja listaaminen

Aivan kuten edellä oleva pakasteateriaesimerkki myös BASIC-ohjelma on numeroitu ohjelista. Koska BASIC on kieli, voi-

daan jokaista ohjelmariviä pitää ikään kuin lauseena. Jokainen Commodoren BASICin rivi saa numeron väliltä 0-63999. (Tätä suurempia rivinumeroita tietokone ei hyväksy.)

Näet tietokoneessa olevien ohjelmarivien täydellisen listan, kun kirjoitat käskyn LIST ja painat RETURN-näppäintä. Rivit ilmaantuvat yksitellen kuvaruutuun niin nopeasti, ettet yleensä pysty niitä lukemaan. Hidastaaksesi listausta paina CTRL-näppäintä. Listaus jatkuu, mutta nyt huomattavasti verkkaisemmin.

(Painaessasi CTRL-näppäintä voit painaa välilyöntinäppäintä nopeuttaaksesi listausta. CTRL-näppäimen painaminen hidastaa myös useimpien BASIC-ohjelmien suoritusta.)

Jos painat RUN/STOP-näppäintä listauksen aikana, koko listaus pysähtyy ja kuvaruutuun tulee sana READY. (Jos haluat aloittaa listauksen uudelleen, kirjoita LIST.) Käskyä LIST voidaan käyttää myös vain tiettyjen rivien seuraamiseen. Tässä esimerkkejä:

Kirjoita	Tietokone listaa
LIST	Kaikki muistissa olevat rivit
LIST 100	Ainoastaan ohjelmarivin 100
LIST 100—150	Kaikki rivit väliltä 100—150
LIST —150	Kaikki rivit alusta riviin 150 asti se mukaan luettuna
LIST 150—	Kaikki rivit alkaen rivistä 150

Vaikka et tarkasti muistakaan tarkasti etsimiesi rivien numeroita, ei ole syytä huoleen. Kolmas, neljäs ja viides esimerkki käyttävät numeroita vain alueen määrittämiseen. Vaikka ei olisikaan rivejä 100 tai 150, tietokone näyttää siitä huolimatta ne rivit, joiden numero on 100:n ja 150:n väliltä, esimerkiksi rivit 101 ja 149.

Yleensä kannattaa aloittaa ohjelmien kirjoittaminen suuremmasta rivinumerosta kuin 0, jotta on mahdollista lisätä uusia rivejä ennen ensimmäiseksi kirjoitettua. Hyvät BASIC-ohjelmoijat jättävät aukkoja myös numeroiden väliin, jotta olisi mahdollista lisätä uusia rivejä entisten väliin. Kannattaa numeroida rivit kymmenen välein, jotta lisäyksille olisi kylliksi tilaa.

Jokainen Commodoren BASICin rivi voi olla pitempi kuin yksi kuvaruuturivi. Rivinumero mukaan luettuna yksi rivi voi olla 80 merkkiä pitkä. (Koska kuvaruudussa näkyy 40 merkkiä rivillä, tämä vastaa kahta kuvaruuturiviä.)

Aloittaessasi ohjelmoinnin sinun tulee varmistua siitä, että tietokoneen muisti on tyhjä. Voit joko kirjoittaa NEW ja painaa RETURN-näppäintä tai sammuttaa koneen ja kytkeä taas virran. (Pyri välttämään koneen sammuttamista ja virran kytkemistä liian usein, koska se rasittaa elektronisia komponentteja.) Tämä valmistaa tietokoneen vastaanottamaan uutta ohjelmaa ja poistaa mahdolliset aikaisemmat ohjelmat muistista. Aloita ohjelmarivi rivinumerolla, jota seuraa asianmukainen BASIC-käsky. RETURN-näppäimen painaminen vie rivin koneen muistiin. Jos et paina tätä näppäintä, ohjelmarivi on ainoastaan kuvaruudussa.

Kirjoittaessasi voit siirtyä mihin tahansa kohtaan ohjelmarivillä (joka siis voi olla kaksi kuvaruuturiviä pitkä) ja korjailla sitä INST/DEL- ja CRSR(kohdistin)-näppäimillä. Tietokoneeseen rakennettu kuvaruutueditori on suunniteltu seuraamaan aina, mitä teet.

Kun olet painanut RETURN-näppäintä ja rivi on koneen muistissa, kohdistin siirtyy kuvaruudussa automaattisesti seuraavan rivin alkuun. Jos olet tehnyt virheen tai muuttanut mielesi etkä haluakaan siirtää työstämääsi riviä koneen muistiin, siirrä kohdistin joko seuraavalle riville tai paina yhtäaikaisesti näppäimiä SHIFT ja RETURN. (Jos painat näitä kahta näppäintä yhtä aikaa, kohdistin siirtyy seuraavalle riville, vaikka riviä ei ole siirretty koneen muistiin.)

Muista, että 80 merkkiä on pisin BASIC-rivi jonka voit siirtää koneen muistiin. Jos kirjoitat pitemmän rivin, kone ei hy-

väksy ylimääräisiä merkkejä.

Poistaaksesi rivejä voit kirjoittaa pelkän rivinumeron ja painaa sitten RETURN-näppäintä. Voit esimerkiksi poistaa rivin 100 kirjoittamalla tyhjälle riville 100 ja painamalla RETURN. Jos yrität listata tämän rivin, sitä ei enää näy.

Voit käyttää hyväksesi Commodoren tietokoneiden tehokkaita korjailutoimintoja, jos haluat kopioida rivin tai sen osan. Kirjoita BASIC-rivi:

```
10 PRINT ''RIVI 10''
```

Paina näppäintä RETURN siirtääksesi rivin koneen muistiin. Siirrä nyt kursori takaisin riville ja muuta rivinumeroksi 20. Muuta 20:ksi myös lainausmerkeissä oleva 10. Kun nyt painat näppäintä RETURN, rivi siirtyy koneen muistiin aivan kuin se olisi kirjoitettu uudelleen. Anna käsky LIST, ja näet molemmat rivit. Tämä kätevä piirre tekee helpoksi kirjoittaa monta samanlaista riviä ja saa myös BASIC-ohjelmoinnin hauskemmaksi.

The END

Monissa tietokoneissa vaaditaan, että BASIC-ohjelma pitää päättää käskyllä END, joka aiheuttaa ohjelman suorituksen lopettamisen. Vaikka onkin epäilemättä hyvä tapa käyttää käskyä END, ei Commodoren BASIC välttämättä vaadi sitä.

On kuitenkin selviä syitä käyttää käskyä END muussa tarkoituksessa kuin ohjelman lopussa. Näitä tapauksia käsitellään myöhemmin.

Muuttujat: kuinka tietokone pitää tiedot järjestyksessä

Kuinka tietokone tietää, mitä me tarkoitamme kysyessämme siltä kaupan laskun loppusummaa tai pyytäessämme sitä vastaamaan johonkin kysymykseen? Kuinka se tietää, mitkä tiedot ovat mitäkin?

BASIC-ohjelman on annettava nimi jokaiselle tiedolle jota tietokoneen on käsiteltävä, ovatpa nämä tiedot sitten numeroita, sanoja, lauseita tai vaikkapa kuvia. Näitä nimiä kutsutaan *muuttujiksi,* koska niiden sisältämää tietoa voidaan jatkuvasti muuttaa. Muuttujia on kahta laatua: numeroita merkitseviä muuttujanimiä sekä kirjaimia ja muita merkkejä merkitseviä muuttujanimiä. Tässä eräitä esimerkkejä muuttujanimistä:

```
A = 10.20
A$ = "TAMA ON COMMODORE-TIETOKONE"
AB = 4096
AB$ = "TAMA ON TOINEN TIETOKONE"
A2 = 45.50
A2$ = "45.50"
```

Tietokone tunnistaa yhtäläisyysmerkin oikealla puolella olevan tiedon vasemmalla puolella annetun nimen perusteella. Kysy, mikä A on, ja tietokone vastaa 10.20. Kysy, mitä AB$ tarkoittaa, ja saat vastaukseksi seuraavan lauseen "TAMA ON TOINEN TIETOKONE".

Kirjaimia ja numeroita tarkoittavat muuttujanimet Commodoren BASIC tunnistaa yksi- tai kaksikirjaimisina. Ensimmäisen merkin on aina oltava kirjain, mutta toinen voi olla joko numero tai kirjain. Muuttujanimi voi siis olla yksittäinen kirjain (A), kaksi kirjainta (AB) tai kirjain ja numero (A2). Jos muuttujanimenä olisi kaksi numeroa (esim. 22 = 25), se sekoittaisi tietokoneen.

Numeroita merkitseviä nimiä kutsutaan *numeerisiksi* muuttujiksi. Esimerkiksi A = 10.20, AB = 4096 ja A2 = 45.40.

Sanoja, lauseita tai muita merkkejä tarkoittavia nimiä kutsutaan *merkki*muuttujiksi. Niiden lopussa on aina oltava dollarin merkki ($). (Edellisessä listassa niitä olivat A$ = "TAMA ON COMMODORE TIETOKONE", AB$ = "TAMA ON TOINEN TIETOKONE" ja A2$ = "45.50".)

Muuttujia katsomalla pitäisi käydä selväksi, että merkkijonot — joita kutsutaan sillä nimellä siksi, että tietokone pitää niitä yksinkertaisesti yhteenliitettyinä merkkeinä — ovat aina lainausmerkeissä. Numerotkin voivat olla osa merkkijonoa, mutta siinä tapauksessa tietokone käsittelee niitä aivan samalla tavalla kuin muitakin merkkejä, kirjaimia ja symboleita.

Lainausmerkkejä käytetään tavallisessa tekstissä usein erottamaan jonkun henkilön sanoja pitemmästä tekstijonosta. BASIC-rivillä lainausmerkit toimivat suunnilleen samalla tavoin. Ne ovat ainoa tapa kertoa tietokoneelle, mistä merkkijono alkaa ja mihin se päättyy. Ja tietokone muistaa aina tuon merkkijonon sanatarkasti, kunhan sen ympärillä vain on lainausmerkit.

Eräs sekaannus saattaa syntyä viimeisen kaltaisten merkkimuuttujien käytössä — A2$ = "45.50". Miksi se on merkkimuuttuja, jos se kerran on numero? Vastaus piilee lainausmerkeissä. Laittamalla lainausmerkit sen ympärille me käskemme tietokoneen pitää sitä merkkiryhmänä, samanlaisena kuin jotakin sanaa tai lausetta, lainkaan välittämättä sen lukuarvosta.

Muista siis, että A2$ = "45.50" *ei ole sama* kuin A2 = 45.50. Ensiksimainittu on merkkijono, toinen on luku.

Numeeristen muuttujien sisältöä voidaan käsitellä — laskea yhteen, vähentää, kertoa ja jakaa. Jos AB on 10 ja CD on 250, niin AB plus CD on 260. Ja CD miinus AB on tietenkin 240.

Samalla tavalla on mahdollista liittää yhteen merkkimuuttujia jokseenkin samalla lailla kuin numeroita lasketaan yhteen. Jos AB$ merkitsee "TAMA ON MINUN" ja CD$ taas "COMMODORE-TIETOKONEENI", niin AB$ plus CD$ on lause "TAMA ON MINUN COMMODORE-TIETOKONEENI". Tätä kahden merkkimuuttujan liittämistä yhteen kutsutaan toisinaan *ketjuttamiseksi,* ja BASICillä se kirjoitetaan ohjelmaksi näin:

```
10 AB$ = "TAMA ON MINUN"
20 CD$ = "COMMODORE-TIETOKONEENI"
30 EF$ = AB$ + CD$
```

Uusi merkkimuuttuja EF$ on AB$ plus CD$ eli koko lause.

Vaikka voitkin liittää yhteen merkkimuuttujia sanoiksi ja lauseiksi, niin *et voi* vähentää niitä toisistaan tai suorittaa niillä muita laskutoimituksia. Et myöskään voi lisätä numeerista muuttujaa merkkimuuttujaan. Jos siis A2 on 45.50 ja A2$ on "45.50", niitä ei voi laskea yhteen ja saada tulokseksi "91.00". Numeroita ja merkkijonoja ei siis voida sekoittaa keskenään. Tietokoneen mielestä A2 on numero ja A2$ merkkijono.

(On olemassa keino saada merkkimuuttujasta tietty luku erilleen, mutta siitä tuonnempana.)

****

TYPE MISMATCH ERROR

Tällainen virheilmoitus näkyy aina, jos yrittää kirjoittaa merkin numeeriseen muuttujaan tai varsinaisen luvun merkkimuuttujaan. Merkkijonoja ja numeroita ei voida sekoittaa keskenään. Vaikeuksia aiheuttavat käskyt ovat esimerkiksi tällaisia:

```
10 A$ = 9 (tai) A$ = A
```

tai

```
10 A = "ESIMERKKI" (tai) A = A$
```

****

Kuten dollarin merkki myös prosenttimerkki antaa muuttujalle erityisen merkityksen. Kirjain A sellaisenaan voi olla miltei mikä luku tahansa, jopa desimaaliluku. Luku 10.5 on numero 10 plus viisi kymmenesosaa (eli puoli). Numeeriset muuttujat, joiden perässä on prosenttimerkki, esimerkiksi A%, merkitsevät tietokoneelle vain kokonaislukuja —32 768:n ja 32 767:n väliltä. Jos on käytettävä pienempiä tai suurempia lukuja, ne

voidaan aina merkitä tavallisina numeerisina muuttujina, esimerkiksi A = 365000.

Miksi sitten pitäisi käyttää kokonaislukumuuttujia? Ensiksikin siksi, että ohjelma toimisi hivenen nopeammin. Ja toiseksi, kokonaislukumuuttujien käyttö vie vähemmän tilaa koneen muistista. Kokonaislukumuuttujia voidaan myös käyttää muuntamaan desimaalilukuja kokonaisluvuiksi. Kokeile tätä:

```
10 A% = 103.987
20 PRINT A%
```

Kun suoritat tämän esimerkkiohjelman, A% tulostuu muotoon 103. Huomaat, että ei ole kyse mistään todellisesta ”pyöristämisestä”. Tavallisesti 103.987 pyöristyisi 104:ksi, ei 103:ksi. Joidenkin ohjelmoijien mielestä on aiheellista aina käyttää kokonaislukumuuttujia merkitsemään kokonaislukuja, toiset taas käyttävät niitä hyvin harvoin.

****

ILLEGAL QUANTITY ERROR

Kokonaislukumuuttujat tunnistetaan prosenttimerkin (%) perusteella. Jos tämä virheilmoitus näkyy rivillä, jossa on kokonaislukumuuttuja, tarkista, etteivät muuttujat merkitse lukuja, jotka ovat suurempia kuin 32767 tai pienempiä kuin – 32768. Jos virhe ei ole tapahtunut sellaisella rivillä, jossa on kokonaislukumuuttuja, tarkista, onko POKE-käskyt annettu oikein.

****

Entäpä jos haluat käyttää kahta kirjainta pitempiä muuttujanimiä? Mikäs siinä. Commodoren tietokoneet muistavat koko nimen, mutta tunnistavat vain *kaksi ensimmäistä kirjainta*. Pidä siis huoli siitä, ettet sekoita konetta. Jos käytät ohjelmassa muuttujia PENSA = 25.87 ja PELIT = 35.55, tietokone ei erota toisistaan sitä kuinka paljon maksoit polttoaineesta ja kuinka paljon olet kuluttanut pelaamiseen. Joidenkin kotitietokonei-

den BASIC-versio sallii tällaisten pitkien muuttujanimien käytön (etenkin Atarin koneet), mutta Commodoren BASICissä niitä ei voida käyttää.

Lopuksi tulisi myös muistaa, että Commodoren tietokoneissa ei voida käyttää tiettyjä muuttujanimiä. Näihin ''kiellettyihin'' nimiin kuuluvat IF, OR, TO ja GO, koska ne ovat BASIC-käskyjä. Myöskään muuttujia TI ja TI$ ei voida käyttää, koska ne on varattu kaikkiin Commodoren tietokoneisiin rakennettuja kelloja varten. Nimeä ST$ voidaan käyttää merkkimuuttujanimenä, mutta ST ei voi merkitä lukua.

Seuraavassa on lyhyt luettelo muuttujanimistä muistamisen helpottamiseksi.

Muuttujanimi	Käykö?	Miksi?
A$,A tai A%	Kyllä	Mitä tahansa yksittäistä kirjainta voidaan käyttää.
AA$,AA tai AA%	Kyllä	Useimmat kahden kirjaimen yhdistelmät ovat mahdollisia.
22$,2B tai 22%	Ei	Nimi aloitettava kirjaimella.
A1$,A1 tai A1%	Kyllä	Kirjainten ja numeroiden yhdistelmät mahdollisia.
TI$,TI tai TI%	Ei	Varattu koneen kellolle.
TO$,TO tai TO%	Ei	TO on BASIC-käsky.
ST	Ei	BASIC-varattu sana.
ST$	Kyllä	Kaksikirjaiminen nimi.
HOTEL$ tai HOTEL	Kyllä	Voi käyttää kahta kirjainta pitempiä nimiä, mutta vain kaksi ensimmäistä kirjainta tunnistetaan.
HOME$ tai HOME	Ei	Sekaantuvat edellisten kanssa.

Tietokone pystyy varastoimaan vain tietyn kokoisia lukuja numeerisiin muuttujiin ja tietyn pituisia merkkijonoja merkkimuuttujiin. Numeeriset muuttujat eivät voi olla yhdeksää nu-

meroa pitempiä. Muuttujanimi AB voi siis merkitä lukua 999999999 tai lukua .999999999, joka on tietysti desimaaliluku.

(Päinvastoin kuin ihmiset tietokone ei halua lukuja kolmen numeron sarjoina. Älä siis koskaan käytä pistettä tai pilkkua erottamaan 999:ää suurempia lukuja. Tietokone vaatii, että ne kirjoitetaan esimerkiksi 1000 tai 12520.)

On mahdollista varastoida suurempiakin lukuja kuin 999 999 999 tai pienempiä kuin .000000001, mutta tietokone ei käsittele niitä tavanomaisina lukuina. Jos annat AB:lle arvon 200 000 000 000 (eli kaksisataa miljardia), niin tietokone antaa kysyttäessä vastaukseksi 2E+11. Kyseessä on tietokoneen tapa merkitä luvut eksponenttimuodossa.

Eksponenttimuoto tarkoittaa sitä, että luku 10 itsellään kerrottuna eli 10 *korotettuna toiseen potenssiin* (10^2) on 100. 10 *korotettuna kolmanteen potenssiin* (eli 10^3) on 10 kertaa 10 kertaa 10, siis 1000. Kummassakin tapauksessa nollien määrä ykkösen jälkeen on sama luku kuin 10:n potenssi (10^n). Kaksisataa miljardia (200 000 000 000) on kakkonen, jota seuraa 11 nollaa, joten se on *2 kertaa 10* (10 potenssiin 11). Potenssit voivat olla myös negatiivisia. Desimaaliluku .000000000002 on 2 kertaa 10^{-11} (10 potenssiin miinus 11).

Koska Commodoren tietokoneet eivät pysty näyttämään kuvaruudussa sellaisia lukuja kuin $2 \cdot 10^{11}$, ne näyttävät kantaluvun (tässä esimerkissä 2), jota seuraa eksponenttia eli potenssiin korottamista merkitsevä kirjain E ja sitten itse eksponentti —35:n ja +35:n väliltä. Lopullinen eksponenttiluku on siis muotoa 2 kertaa 10 potenssiin 11 (2E+11).

(Lisätietoja eksponenttiluvuista ja niiden merkitsemistavoista löytyy yläasteen matematiikan oppikirjoista.)

Jos et aio käyttää niin suuria tai pieniä lukuja, että ne täytyy kirjoittaa eksponenttimuodossa, sinun ei kannata välittää koko eksponentiaalisesta merkitsemistavasta. On kirjoitettu tuhansia kelvollisia ohjelmia vailla mitään tietoja näistä luvuista.

****

OVERFLOW ERROR

Tämä virheilmoitus esiintyy silloin kun yrität käsitellä lukua 1.7141884E + 38 suurempia lukuja tai jotain lukua, joka kerrotaan ykkösellä jota seuraa 38 nollaa. Ratkaisu (jos se on mahdollinen), on hajottaa laskutoimituksesi osiin siten, että tätä lukua ei koskaan saavuteta.

****

Merkkimuuttujien pituuden määräävä sääntö on paljon yksinkertaisempi. Mikään merkkijono ei voi olla pitempi kuin 255 merkkiä. Tähän lukuun sisältyvät myös yhteenliitetyt eli ketjutetut merkkijonot.

****

STRING TOO LONG ERROR

Merkkijonot saavat olla vain 255 merkkiä pitkiä. Merkkijonovirheitä esiintyy yleensä silloin, kun muuttujia liitetään yhteen eli lisätään jokin merkkimuuttuja toiseen. Tarkista jokaisen merkkimuuttujan pituus yrittäessäsi selvittää syytä tähän virheilmoitukseen.

****

Joskus, tosin hyvin harvoin, saatat nähdä sanan LET jossakin BASIC-ohjelmassa. Sitä käytetään muuttujan arvon asettamiseen. Commodoren BASICissä se kuitenkin voidaan jättää pois, mutta toisaalta se selkeyttää ohjelmaa. Kumpikin seuraavista riveistä on oikein:

```
10 LET AB = 4096
```

tai

```
10 AB = 4096
```

Välimerkkien käyttö BASICissä

Koska BASIC on sanoista ja niiden käyttöä koskevista säännöistä kokoonpantu kieli, siinä on muiden kielten lailla oltava

välimerkkejäkin. BASICin välimerkkisäännöt ovat kuitenkin paljon yksinkertaisemmat kuin esimerkiksi suomen kielen.

Aivan kuten useat suomen lauseet, voidaan useita BASIC-lauseitakin kirjoittaa samalle riville. Suomessa lauseita erottavat toisistaan pisteet, esimerkiksi

Mene kotiin. Syö lounas.

BASIC-rivillä ei käytetä kahta erillistä lausetta erottamassa pistettä vaan kaksoispistettä (:). BASIC-rivi, jolla on useita lauseita, näyttää siis tältä:

```
10 PRINT ''TAMA ON TIETOKONEENI'':GOTO 10
```

Tärkeintä tässä rivissä on huomata, kuinka kaksoispistettä käytetään, tässä tapauksessa siis erottamaan toisistaan kahta erillistä BASIC-lausetta. BASIC-ohjelmarivi, mukaan luettuna numerot ja tyhjät välit, voi olla enintään 80 merkkiä eli kaksi kuvaruuturiviä pitkä.

Voidaan myös käyttää kaksoispistettä, jotta BASIC-riveistä tulee luettavampia. Tämä saadaan aikaan siten, että yksi tai useampia kaksoispisteitä voidaan kirjoittaa riville ensimmäisiksi merkeiksi rivinumeron jälkeen, jotta rivi saadaan erotetuksi muusta ohjelmalistauksesta. Esimerkiksi

```
10 PRINT ''TAMA ON TIETOKONEENI''
20 ::GOTO 10
```

Nämä kaksi kaksoispistettä muodostavat tavallaan sisennyksen, joka auttaa tunnistamaan ohjelmarivin helpommin.

****

SYNTAX ERROR

Tällaisten ''kielioppivirheiden'' syitä on monia, mutta ne kaikki liittyvät siis BASICin sääntöihin. Yleisimpiä virheitä ovat kirjoitusvirheet ja välimerkkien, erityisesti kaksoispisteiden ja pilkkujen, puuttuminen. Tarkkaile myös, oletko vahingossa kirjoittanut puolipisteen kaksoispisteen asemesta ja

oletko muistanut sulkea sulut. Tarkista, oletko käyttänyt varattuja sanoja.

****

BASIC-ohjelmoinnissa käytetään muitakin välimerkkejä, kuten pilkkua (,), puolipistettä (;) ja lainausmerkkejä. Niitä käytetään pääasiassa PRINT-käskyn kanssa.

PRINT-käskyn käyttö

PRINT on BASIC-kielen käytetyimpiä sanoja. Sen käytön periaate on helppo ymmärtää. Välimerkkien kanssa sitä voidaan kuitenkin käyttää monin tavoin. PRINT-käsky saattaa alkuun tuntua hankalalta. Koska PRINT-käskyyn liittyy enemmän muuntelumahdollisuuksia kuin mihinkään muuhun BASICin sanaan, kestää myös kauemmin oppia sen kaikki mahdollisuudet.

PRINT käskee tietokoneen kirjoittaa jotakin kuvaruutuun, sanoja ja lauseita, laskutoimituksia ja jopa kuvia.

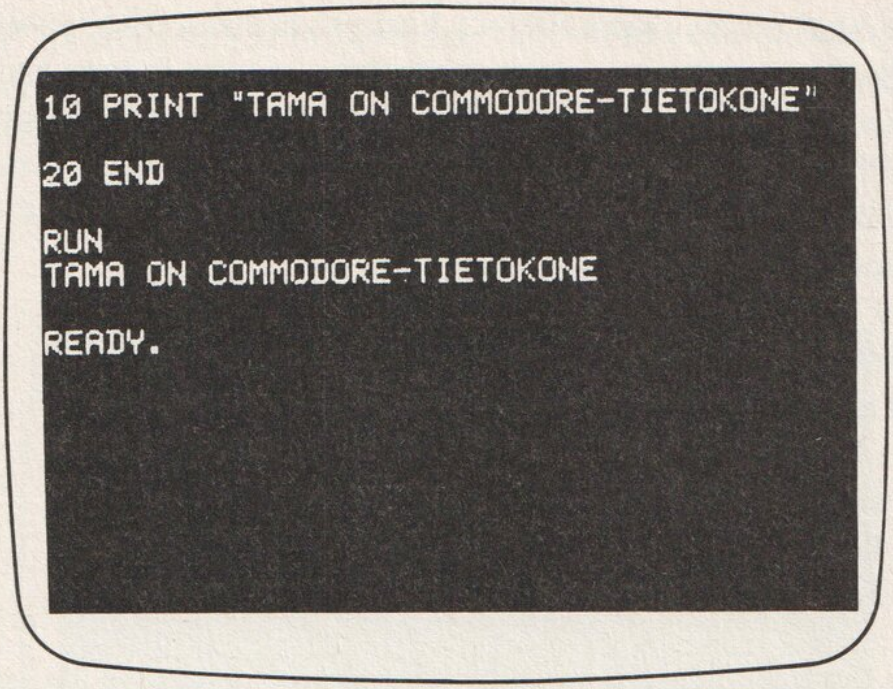
Näit pienen ohjelman jo kirjan alussa, kun puhuimme koneesi toimintatarkistuksesta. Sen ensimmäinen rivi oli

```
10 PRINT "TAMA ON COMMODORE-TIETOKONE"
```

Tämä rivi tekee juuri sitä mitä se näyttääkin tekevän.

PRINT-käskyä seuraa yleensä se, mitä haluat tietokoneen kirjoittavan. Tässä tapauksessa on kyse lauseesta "TAMA ON COMMODORE-TIETOKONE".

Kuten näet, lause on ohjelmassa lainausmerkeissä, mutta lainausmerkit katoavat, kun lause ilmestyy kuvaruutuun. Lainausmerkit pelkästään osoittavat tietokoneelle, että sen tulisi kirjoittaa niiden *välissä* oleva lause.



Yritä kirjoittaa edelläoleva BASIC-rivi siten, että sijoitat lainausmerkkien väliin jotakin muuta. Anna sitten käsky RUN. Sinun pitäisi nähdä, mitä olet kirjoittanut lainausmerkkien väliin.

PRINT-käskyllä saadaan kone kirjoittamaan numeeristen ja merkkimuuttujien merkitsemiä lukuja ja merkkejä.

```
10 A = 4096
20 B$ = ''TAMA ON COMMODORE-TIETOKONE''
30 PRINT A
40 PRINT B$
```

Oikopolku

Kokeile jo nyt oikopolkua, jotta voit käyttää sitä vastedes. Kirjoita ohjelmarivi uudelleen tällä tavalla:

```
10 ? ''TAMA ON COMMODORE-TIETOKONE''
```

Kirjoitettuasi rivin näin (siis kysymysmerkki sanan PRINT sijaan) listaa rivi käskyllä LIST. Vaikket kirjoittanutkaan sanaa PRINT, tietokone ymmärsi mitä tarkoitit. Se tietää, että kysy-

mysmerkki on tavallaan PRINT-käskyä vastaava pikakirjoitusmerkki. Mutta älä oleta sen ymmärtävän mitään muuta. Se tietää ainoastaan sen, että sana PRINT voidaan korvata kysymysmerkillä. Ellei tämän kirjan ohjeissa erityisesti muuta sanota, voit lähes aina käyttää koko PRINT-sanan asemesta tätä nopeampaa tapaa.

Tyhjän rivin kirjoittaminen
Yritä antaa koneelle käsky PRINT, jota ei seuraa mitään.

```
10 PRINT ''TAMA ON TIETOKONEENI''
20 PRINT
30 PRINT ''TAMA ON TIETOKONEENI''
```

Sinun pitäisi nähdä kuvaruudussa kaksi samanlaista lausetta, joiden välissä on tyhjä rivi. Sanan PRINT käyttäminen sellaisenaan on yksi tapa hypätä rivin yli kirjoitettaessa kuvaruutuun.

Kokeile nyt toista tapaa sanoa sama asia vain yhdellä ohjelmarivillä BASICin kaksoispistettä käyttäen.

```
10 PRINT ''TAMA ON TIETOKONEENI'':PRINT:PRINT
   ''TAMA ON TIETOKONEENI''
```

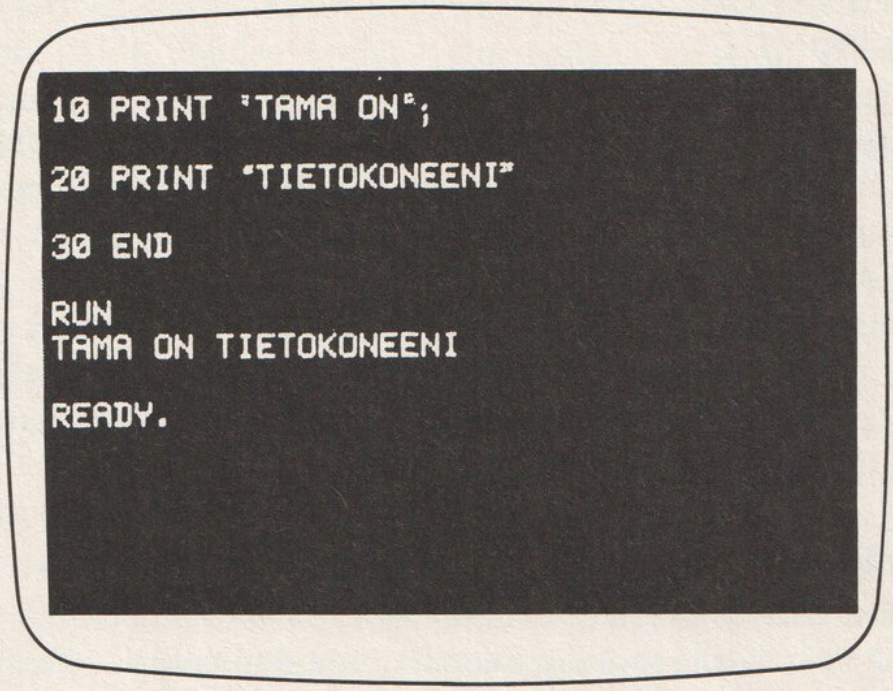
Kahta muuta BASICin välimerkkiä, pilkkua ja puolipistettä, käytetään usein PRINT-lauseessa. Kumpikin niistä vaikuttaa PRINT-käskyn toimintaan.

PRINT-käskyn käyttö puolipisteen kanssa
Kirjoita nämä ohjelmarivit *täsmälleen* sellaisina kuin ne ovat tässä, tyhjät välit mukaan luettuina. (Voit korvata PRINT-käskyn kysymysmerkillä.)

```
10 PRINT ''TAMA ON  '';
20 PRINT ''TIETOKONEENI''
```

Suorita ohjelma (käskyllä RUN) ja katso mitä tapahtuu. Jos kirjoitit esimerkin oikein ja jätit rivillä 10 tyhjän välin sanan

''on'' ja sitä seuraavan lainausmerkin väliin sekä muistit puolipisteen, ohjelman tulisi kirjoittaa lause ''TAMA ON TIETOKONEENI''.



Puolipiste kertoo tietokoneelle, mistä kohdasta sen tulisi alkaa ruutuun kirjoittaminen. Kun PRINT-käskyä käytetään ohjelmassa, tietokone yleensä aloittaa kirjoittamisen seuraavalta kuvaruuturiviltä. Puolipiste käskee tietokoneen jäädä *samalle* riville ja ryhtyä kirjoittamaan sen rivin seuraavasta kohdasta. Tällä tavalla voidaan liittää yhteen yhden lauseen eri osia tai merkkiryhmiä. Näin voidaan myös liittää lauseita ja muuta tekstiä yhteen kuvaruudussa. (Tämä tapahtuu *ainoastaan* kuvaruudussa. Eri muuttujanimillä tallennetut merkkimuuttujat pysyvät erillään koneen muistissa.)

Tyhjä väli sanan ''on'' ja lainausmerkin välillä rivillä 10 on tärkeä. Ilman sitä kuvaruudussa lukisi: ''TAMA ONTIETOKONEENI''.

Sanojen tavoin voidaan numeroita kirjoittaa kuvaruutuun PRINT-käskyn ja puolipisteen avulla.

```
10 A = 100
20 B = 1
30 C = A + B
40 PRINT C
```

Rivillä 10 merkitään lukua 100 muuttujalla A ja rivillä 20 lukua 1 muuttujalla B. Tämän ohjelman suorittaminen antaa ensiksi muuttujille nämä arvot ja sitten kirjoittaa kuvaruutuun summan A(100) plus B(1) eli 101.

Katsomatta ohjelmaa ei A:n ja B:n summa ehkä kerro kovinkaan paljon. Jotta tietäisimme mistä on kyse, voimme käskeä tietokoneen ilmoittamaan, mistä luvuista kuvaruutuun kirjoittuva summa koostuu. Muuta rivi 40:

```
40 PRINT "A + B = ";C
```

Tämä voi näyttää turhalta toistolta, mutta niin ei ole. Rivillä 40 tietokoneen käsketään kirjoittaa ruutuun ilmaisu A + B = . Sitten puolipiste ilmoittaa, että kirjoittaminen jatkuu samalta riviltä, ja lopuksi tietokone kirjoittaa C:n *arvon,* A:n ja B:n summan eli 101. Jos tällä rivillä kerran kirjoitetaan ruutuun kaksi eri asiaa, niin miksi BASIC-käskyä PRINT käytetään vain kerran? Tietokone tekee erään tärkeän oletuksen kohdatessaan jollain rivillä puolipisteen. Jos rivillä ei esiinny kaksoispistettä, tietokone olettaa, että se on jatkettava perässä seuraavan asian kirjoittamista ruutuun.

Huomaat varmasti, että lukua 101 ei ole kirjoitettu välittömästi yhtäläisyysmerkin viereen. Se johtuu siitä, että puolipiste vaikuttaa numeroihin hiukan eri lailla kuin merkkimuuttujiin. Ne kirjoitetaan kahden tyhjän välin päähän edellisestä kirjoitetusta merkistä. Tämä ei ole BASICissä oleva virhe, vaan se on suunniteltu pitämään numerot aina erillään, ettei tapahtuisi sekaannuksia. Tässä on edellisen ohjelman muunnelma:

```
10 A=100
20 B=1
30 C=A+B
40 PRINT A;B;C
```

Nyt kirjoittuvat kaikkien ohjelmassa olevien muuttujien lukuarvot, kukin kahden tyhjän välin päähän toisesta. Tällainen looginen erottelu on voimassa myös silloin, kun merkkimuuttuja kirjoitetaan numeron jälkeen puolipistettä käyttäen. Kokeilepa tätä:

```
10 A=100
20 B=1
30 C=A+B
40 PRINT A;B;C;"OVAT A:N, B:N JA C:N ARVOT"
```

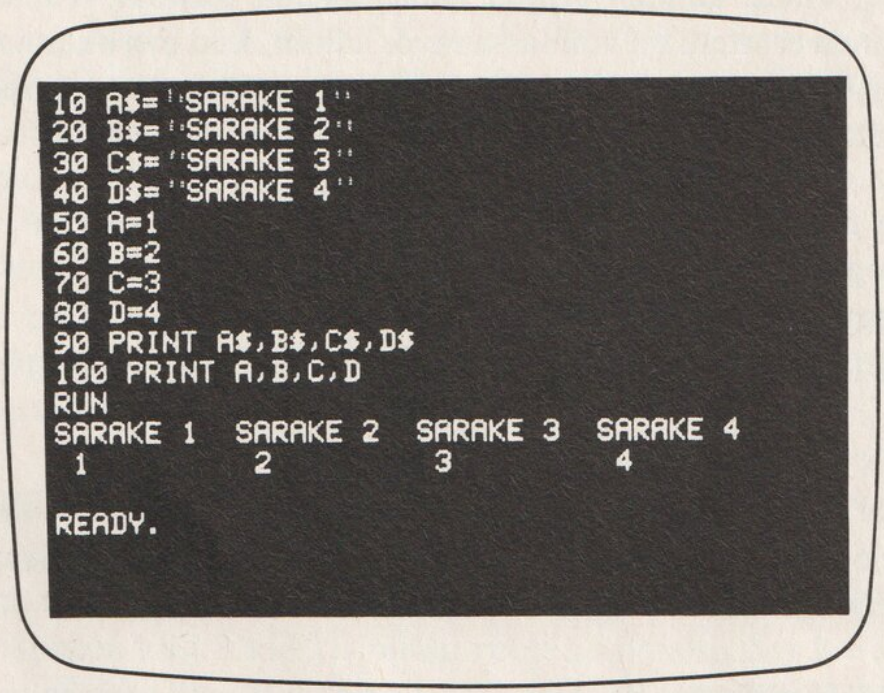
PRINT-käskyn käyttö pilkun kanssa

PRINT-käskyä voidaan käyttää myös pilkun kanssa. Commodore 64 -koneissa se järjestää kuvaruudun sarakkeiksi. Seuraava ohjelma esittää selkeästi, kuinka pilkkua käytetään:

```
10 A$="SARAKE 1"
20 B$="SARAKE 2"
30 C$="SARAKE 3"
40 D$="SARAKE 4"
50 A=1
60 B=2
70 C=3
80 D=4
90 PRINT A$, B$, C$, D$
100 PRINT A,B,C,D
```

Kun suoritat tämän ohjelman, huomaat, että 40 merkin kuvaruuturivi on jakautunut neljään sarakkeeseen, joista kukin on 10 merkkiä leveä. Sinun tulisi myös huomata merkkimuuttujien ja lukujen tulostuksen erot. Merkkimuuttujat kirjoitetaan aina

sarakkeen ensimmäisestä kohdasta alkaen. (Kuvaruudussa merkkimuuttujat alkavat asemista 1, 11, 21 ja 31, kun annat ensimmäiselle kohdalle tunnuksen 1.) Luvut hyppäävät ensimmäisen kohdan yli ja kirjoittuvat toisesta kohdasta alkaen. (Lukusarakkeiden aloituskohdat ovat siis 2, 12, 22 ja 32 kuvaruudussa.)



Muuta rivi 40 seuraavalla tavalla:

40 DS = "OLEN SARAKKEESSA NUMERO NELJA"

Suorita ohjelma jälleen. Näet mitä tapahtuu, kun sarakkeessa oleva tieto vaatii yli kymmenen merkkiä (tai yhdeksän merkkiä, jos on kyse pelkästään numeroista). Nyt tietokone kirjoittaa myös seuraavaaseen sarakkeeseen ja siirtyy sitten sitä seuraavaan. Tällaiset pitkät sarakkeet voivat pilata huolellisesti suunnitellun kuvaruutukuvan, joten karta niitä.

Sinun tulisi seurata tarkoin mitä teet, aina kun käytät puolipistettä tai pilkkua. Jos käytät jompaakumpaa rivin lopussa, kirjoittaa seuraava PRINT-käsky kuvaruutuun välimerkin

määräämään paikkaan — seuraavaan tai sitä seuraavaan kohtaan, jos olet käyttänyt puolipistettä, ja seuraavaan sarakkeeseen, jos olet käyttänyt pilkkua. Jos siis PRINT-lauseen viimeinen merkki ei ole puolipiste tai pilkku, kohdistin siirtyy automaattisesti seuraavalle riville jatkaessaan kirjoittamista seuraavan PRINT-käskyn kohdatessaan.

PRINT-esimerkki	Tulos
10 PRINT A	Kirjoittaa A:lla merkityn luvun kuvaruutuun ja kohdistin siirtyy seuraavalle riville.
10 PRINT A;B;C;D	Kirjoittaa A:lla, B:llä, C:llä ja D:llä merkityt luvut kuvaruutuun samalle riville, kunkin kahden tyhjän välin päähän toisesta.
10 PRINT A,B,C,D	Kirjoittaa A:lla, B:llä, C:llä ja D:llä merkityt luvut neljään sarakkeeseen yhdeksän välin päähän toisistaan.
10 PRINT A;B 20 PRINT C;D	Kirjoittaa A:lla ja B:llä merkityt luvut yhdelle riville kahden välin päähän toisistaan ja kirjoittaa sitten C:llä ja D:llä merkityt luvut seuraavalle riville samalla tavalla.
10 PRINT A,B 20 PRINT C,D	Kirjoittaa A:lla, B:llä, C:llä ja D:llä merkityt luvut kahdelle riville kummankin rivin kahteen ensimmäiseen sarakkeeseen.
10 PRINT A$	Kirjoittaa A$:lla nimetyn merkkimuuttujan, ja kohdistin siirtyy seuraavalle riville.
10 PRINT AS;B$;C$;D$	Kirjoittaa A$:lla, B$:lla, C$:lla ja D$:lla nimetyt merkkimuuttujat toistensa viereen samalle riville ilman välejä.

10 PRINT AS,B$,C$,D$	Kirjoittaa A$:lla, B$:lla, C$:lla ja D$:lla nimetyt merkkimuuttujat neljään kuvaruutusarakkeeseen, elleivät ne ole liian pitkiä.
10 PRINT A$;B$ 20 PRINT C$;D$	Kirjoittaa A$:lla ja B$:lla nimetyt merkkimuuttujat yhdelle riville toistensa viereen ilman välejä ja kirjoittaa sitten C$:lla ja D$:lla nimetyt merkkimuuttujat seuraavalle riville samalla tavalla.
10 PRINT A$,B$ 20 PRINT C$,D$	Kirjoittaa A$:lla, B$:lla, C$:lla ja D$:lla nimetyt merkkimuuttujat kahdelle ensimmäiselle riville kunkin rivin kahteen ensimmäiseen sarakkeeseen, elleivät ne ole liian pitkiä.

PRINT-käskyn käyttö suoraan näppäimistöltä

Jos haluat välittömän tiedon siitä, mitkä luvut tai merkkijonot vastaavat kutakin muuttujanimeä, ei ole tarpeen kirjoittaa ohjelmariviä sitä varten. Käsky PRINT (tai ?) on käytettävissä suoraan näppäimistöltä.

PRINT [tai ?] AB [ja paina RETURN]

Edellinen rivi kirjoittaa kuvaruutuun, mikä arvo on AB:llä.

PRINT [tai ?] AB$ [ja paina RETURN]

Tämä rivi puolestaan kirjoittaa sen merkkijonon, jota AB$ vastaa.

PRINT-käskyn kuvaruututoiminnot

Käskyä PRINT käytetään myös useisiin kuvaruututoimintoihin, mm. kohdistimen siirtelyyn kuvaruudussa, sen siirtämiseen

”home”-asemaan kuvaruudun vasempaan yläkulmaan ja kuvaruudun tyhjentämiseen.

Paitsi että kohdistinta voidaan liikutella kuvaruudussa näppäimistöltä (käyttämällä kohdistin ylös ja alas tai oikealle ja vasemmalle -näppäimiä), voi BASIC-ohjelmakin siirrellä kohdistinta tiettyihin paikkoihin. Se onnistuu yhdistämällä PRINT-käskyyn kohdistimen siirtokäskyjä. Voit kokeilla seuraavalla ohjelmalla:

```
10 PRINT ”RIVI 1”
20 PRINT ”[paina CRSR ALAS -näppäintä viisi kertaa]”
```

Kun olet kirjoittanut ensimmäisen lainausmerkin, älä kirjoita hakasulkeissa olevaa lauseketta, vaan paina CRSR ALAS -näppäintä viisi kertaa. Kohdistin ei liiku, mutta näet kuvaruudussa viisi merkkiä. (Ne näyttävät Q-kirjaimilta joilla on käänteinen taustaväri.)

Kirjoita toinen lainausmerkki lopettaaksesi rivin. Kirjoita sitten uusi rivi:

```
30 PRINT ”RIVI 2”
```

Suorita ohjelma. Se kirjoittaa ruutuun sanat RIVI 1, sen jälkeen viisi tyhjää riviä ja sitten sanat RIVI 2.

Voit käyttää kaikkia kohdistinkäskyjä (ylös, alas, vasemmalle, oikealle) tällä tavalla. Joskus kohdistimen liikkeitä on kuitenkin hankala tietää etukäteen. Tee silloin muistiinpanoja paperille tietääksesi, mihinpäin kohdistin on liikkunut. Kun ohjelmaa suoritetaan, kohdistin on näkymätön ja sen liikkeet liian nopeita huomata.

Käytä samaa ohjelmaa (älä kirjoita käskyä NEW) ja lisää siihen seuraava rivi:

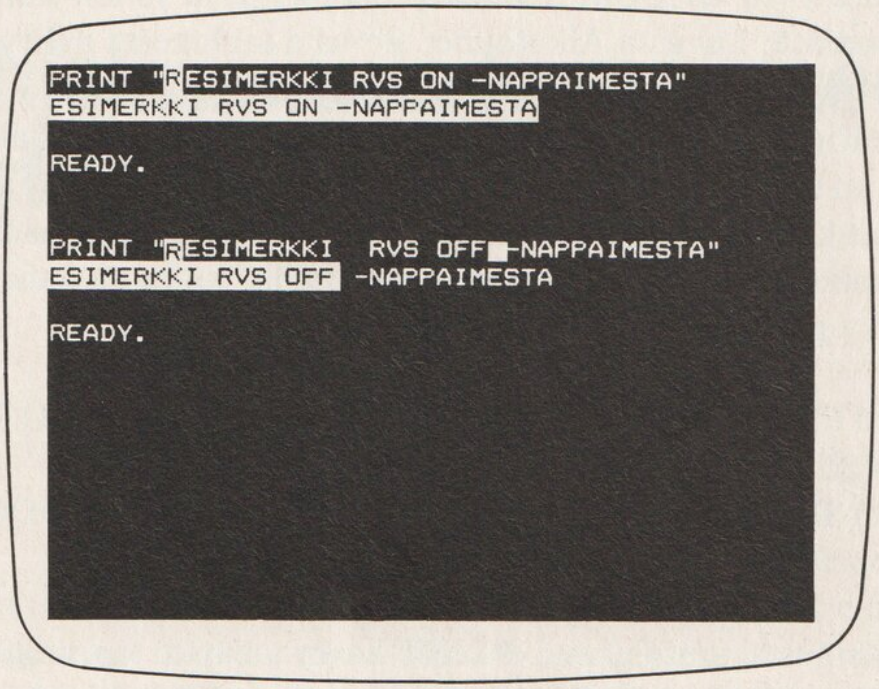
5 PRINT "[paina SHIFT-näppäintä ja samanaikaisesti näppäintä CLR/HOME]"

Näet nyt toisen symbolin, käänteisvärisellä taustalla olevan sydämen. Suorita ohjelma. Tämä rivi käskee tietokoneen tyhjentää kuvaruudun ennen kuin se suorittaa aiemmin tähän ohjelmaan kirjoitetut rivit. Se myös sijoittaa kohdistimen ruudun vasempaan yläkulmaan.

Jos olisit painanut näppäintä CLR/HOME *ilman* SHIFT-näppäintä, olisit nähnyt lainausmerkkien välissä erilaisen symbolin, käänteisvärisellä taustalla olevan kirjaimen S. Se olisi sijoittanut kohdistimen ruudun vasempaan yläkulmaan tyhjentämättä kuvaruutua. Tämän "alkuasemaa" osoittavan symbolin käyttö on kätevää silloin, kun tehdään mutkikkaita kohdistimen siirtoja. Sillä voidaan varmistaa aloitus aina samasta paikasta, vasemmasta yläkulmasta.

Huomasit ehkä, että tämä ohjelma aloittaa kirjoittamisen kuvaruudun toiselta riviltä ensimmäisen asemesta. Jos PRINT "tyhjennä kuvaruutu" tai "siirrä kohdistin alkuasemaan" -käskyä ei seuraa puolipiste, kohdistin siirtyy aina tuolle toiselle riville. (Tämä on loogista ja noudattaa puolipisteen ja pilkun käytöstä annettuja sääntöjä.)

Kuvaruutuun kirjoitettua tekstiä voidaan korostaa käyttämällä tähän tarkoitukseen suunniteltuja näppäimiä RSV ON ja RSV OFF. Jos kirjoitetaan lainausmerkki ja painetaan sitten samanaikaisesti näppäimiä CTRL ja RVS ON, tietokone kirjoittaa seuraavat merkit käänteisväriselle taustalle. Grafiikkatilassa näkyy tietty merkki, käänteisellä pohjalla oleva kirjain R, kun painat lainausmerkin jälkeen RVS ON -näppäintä.



Kirjoitus palaa automaattisesti normaaliksi sellaisen rivin lopussa, jolla on RVS ON. Voit palauttaa näytön normaaliksi myös rivin keskellä käyttämällä RVS OFF -näppäintä. Käänteistaustan muuttamiseksi takaisin normaaliksi on painettava yhtä aikaa näppäimiä CTRL ja RVS OFF. Grafiikkatilassa näet neliön, jonka alalaidassa on viiva ja joka näkyy käänteistä taustaa vasten. Kun tämä RVS OFF -merkki esiintyy keskellä riviä, lainausmerkkien sisällä ja RVS ON -merkin jälkeen, kaikki sitä edeltävä kirjoittuu käänteisenä ja kaikki sen jälkeen tuleva normaalinäyttöisenä.

```
10 PRINT "NORMAALINAYTTO [paina CTRL ja RVS
ON] KAANTEISNAYTTO [paina CTRL ja RVS OFF]
NORMAALINAYTTO"
```

Värinäppäimiä käytetään samalla tavoin. Niistä jokainen kirjoittaa eri värillä kuvaruutuun silloin kun niitä käytetään PRINT-lauseessa lainausmerkkien välissä. Voit kirjoittaa väreillä painamalla CTRL-näppäintä ja haluamaasi numeronäppäintä 1—8 tai COMMODORE-näppäintä ja jotain numeronäppäintä. Luvussa Alkutoimet olevasta taulukosta näkyy, mikä näppäinyhdistelmä antaa tulokseksi minkin värin.

Värinäppäinten ja käänteistaustanäppäinten pääasiallinen ero on se, että tietokone kirjoittaa määrätyllä värillä, kunnes sitä käsketään muuttamaan se. Jos siis olet kirjoittanut sinisellä ja haluat korostaa jotain sanaa punaisella, sinun on annettava käsky palata siniseen.

```
10 PRINT "[paina CTRL ja WHT] TAMA JATKUU VAL-
KOISELLA KIRJOITETTUNA"
20 PRINT "KUNNES KÄSKET [paina COMMODORE-
näppäintä ja BLU] TOISIN"
```

Jos sinusta tuntuu, että PRINT-käskyn käyttö on monimutkaista, olet oikeassa. Se tarjoaa niin paljon mahdollisuuksia, että sen käyttämiseenkin liittyy paljon sääntöjä. Useimmilla BASICin sanoilla ei kuitenkaan ole niin monia käyttötarkoituksia, ja niitä onkin huomattavasti helpompi kuvata ja ymmärtää.

4

Ohjelmoinnin kymmenen tärkeintä sääntöä

Kuten missä tahansa kielessä on myös BASICissä tietyt eniten käytetyt sanat. Ne ovat myös kaikkein hyödyllisimpiä. Tässä luvussa esitellyt käskyt — INPUT, GOTO, IF ja THEN, GET, FOR ja NEXT, GOSUB ja RETURN sekä REM — tarjoavat runsaasti mahdollisuuksia. Näiden (ja edellisessä luvussa esitellyn PRINT-käskyn) avulla voit aloittaa kunnon ohjelmoinnin. Nämä kymmenen sanaa tutustuttavat sinut myös eräisiin uusiin käsitteisiin. Esimerkiksi GOTO, GOSUB ja RETURN liittyvät siihen, missä järjestyksessä ohjelma tekee eri asioita. INPUT ja GET opettavat millä lailla tietokoneelle annetaan tietoa. Käskyjen IF ja THEN avulla kone voi koestaa tietoa ja tehdä yksinkertaisia päätöksiä. REM-käskyn avulla voit lisätä huomautuksia BASIC-ohjelmiisi tehdäksesi niistä ymmärrettävämpiä. Vaikket oppisi kuin nämä muutamat sanat, pystyt saamaan tietokoneestasi paljon irti.

INPUT-käsky pyytää tietoja

BASICin sana INPUT on monella tapaa PRINT-sanan täydellinen vastakohta. Kuitenkin tämä sana noudattaa joitakin samoja sääntöjä kuin PRINT.

Käskyä INPUT käytetään tiedon hankkimiseen näppäimistöltä ja sen sijoittamiseen tiettyyn muuttujaan. Kun INPUT-käskyä käytetään, tietokone kirjoittaa ruutuun kysymysmerkin ja jää odottamaan, kunnes joku on kirjoittanut vastauksen (merkkimuuttujan tai numeron) ja painanut näppäintä RETURN. Jos sitten on vielä ohjelmarivejä jäljellä, INPUT jatkaa ohjelman suorittamista.

Yksinkertaisimmassa muodossaan käskyä INPUT käytetään vain muuttujan kanssa.

```
10 INPUT A
```

Edellinen rivi kirjoittaa ruutuun kysymysmerkin ja odottaa, että joku kirjoittaisi sen luvun, joka pitäisi varastoida muuttujanimellä A.

```
10 INPUT A$
```

Samalla tavalla toimii tämä rivi, paitsi että se tarvitsee vastaukseksi merkkijonon, joka varastoidaan muuttujaan A$. INPUT-käskyn omituisuuksia on se, että se ei hyväksy vastaukseksi merkkijonoja, joissa on pilkkuja. Jos INPUT-lause kysyy esimerkiksi osoitettasi, et voi vastata ”Kehtopolku 6, Helsinki” tai ”Sampsantie 14, Tornio”. INPUT hyväksyy sen, mitä on kirjoitettu ennen pilkkua, ja jättää muun huomiotta.

****

EXTRA IGNORED

Tätä virhettä et oikeastaan voi korjata, sillä se aiheutuu siitä, että INPUT-käskyn vastauksessa on pilkku paikassa jossa sen ei tulisi olla. Commodore-tietokone käyttää pilkkuja eri-

tyismerkkeinä. Ne sallitaan vastaukseksi INPUT-käskyyn vain silloin, kun ohjelma pyytää vastauksia yhtä aikaa, esimerkiksi:

10 INPUT A$,B$,C$

Merkkimuuttujissa olevat pilkut aiheuttavat ongelmia myös silloin, kun niitä tallennetaan ja luetaan levyketiedostoista.

****

On hankalaa käyttää INPUT-käskyä tällä tavalla, siis tiedon pyytämiseen ilman kuvaruutuun PRINT-käskyllä kirjoitettua kysymystä. Ohjelman tietopyynnöt saa selkeämmiksi kirjoittamalla jotakin juuri ennen kysymysmerkkiä, jonka se aina kirjoittaa ruutuun. Kirjoita kysymyksesi lainausmerkkien sisään aivan samalla tavalla kuin käskyssä PRINT. Esimerkiksi:

```
10 INPUT ''ANNA A:N ARVO'';A
```

Tämä rivi kirjoittaa kuvaruutuun ANNA A:N ARVO? (INPUT-käsky kirjoitti tuon kysymysmerkin.) Sinun on käytettävä puolipistettä, kuten esimerkissä, erottamaan muuttujanimi kysymyksestäsi.

Voit pyytää myös useita lukuja yhtä aikaa erottamalla numeeriset muuttujat toisistaan pilkulla.

```
10 INPUT ''ANNA A:N, B:N JA C:N ARVOT'';A.B,C
```

Tämä rivi kirjoittaa ruutuun kysymyksen ANNA A:N, B:N JA C:N ARVOT? ja jää sitten odottamaan vastauksia. Kun INPUT-käsky pyytää useampia vastauksia kuin yhden, on kirjoitettava kaikki vastaukset ennen RETURN-näppäimen painallusta. Jotta tietokone ymmärtäisi kunkin luvun erilliseksi, ne on kirjoitettava pilkuilla erotettuina, kuten tässä:

```
100,10,25 [ja paina RETURN]
```

Koska ohjelman käyttäjälle on usein epäselvää, vaatiiko tieto-

kone nämä pilkut, on yleensä turvallisinta käyttää erillisiä INPUT-lauseita, kuten tässä:

```
10 INPUT "ANNA A:N ARVO";A
20 INPUT "ANNA B:N ARVO",B
30 INPUT "ANNA C:N ARVO";C
```

Jos edellä olevan kaltainen ohjelma pyytää lukuja, se ei hyväksy vastaukseksi mitään muuta. Jos kirjoitat esimerkiksi kysymykseen ANNA A:N ARVO? vastaukseksi KAKSI, tietokone ilmoittaa sinun tehneen virheen ja pyytää kirjoittamaan vastauksen numeroina, ei kirjaimina.

****

REDO FROM START

Tämä virheilmoitus näkyy silloin, kun vastaat INPUT-kysymykseen merkkijonolla, vaikka se odottaa numeroa. Korjaa tilanne kirjoittamalla numero.

****

Jos INPUT-kysymykseen ei anneta vastausta — ei kirjoiteta lukua tai merkkijonoa — ja painetaan ainoastaan näppäintä RETURN, INPUT olettaa, että vastaus on joko 0 (jos kyseessä on luku) tai *ei yhtään mitään* (jos kyseessä on merkkimuuttuja). Tällainen "tyhjä" merkkijono voidaan kirjoittaa esimerkiksi A$="" (siten, ettei lainausmerkkien väliin tule tyhjää väliä). Tulet myöhemmin huomaamaan, että tyhjät merkkijonot voivat olla hyödyksi.

Jotta oppisit paremmin käyttämään käskyä INPUT, kirjoita seuraava lyhyt ohjelma:

```
10 INPUT "ANNA NIMESI";NI$
20 INPUT "KUINKA VANHA OLET";IK
30 PRINT "NIMESI ON ";NI$
40 PRINT "OLET";IK;"-VUOTIAS"
```

Tämä ohjelma kirjoittaa ensiksi kysymyksen ANNA NIMESI?

Sitten se odottaa vastaustasi. Kun olet kirjoittanut nimesi, se kirjoittaa seuraavan kysymyksen, KUINKA VANHA OLET? Taas se odottaa vastaustasi, ja heti kun olet sen antanut, se kirjoittaa välittömästi kaksi lausetta: NIMESI ON (nimi jonka kirjoitit) ja OLET (luku jonka iäksesi kirjoitit)-VUOTIAS.

Kuten olet jo ehkä arvannutkin, INPUT on melkoisen yksinkertainen ja selkeä käsky. Sitä on vaikea käyttää väärin muussa tapauksessa kuin ohjelman ulkopuolella (siis suoraan näppäimistöltä).

****

ILLEGAL DIRECT ERROR

Useita BASIC-käskyjä voidaan käyttää ohjelman ulkopuolella kirjoittamalla ne ruutuun suoraan näppäimistöltä. INPUT -käskyä ei kuitenkaan voida käyttää siten. Tämä onkin selvää, jos muistaa, kuinka sitä käytetään.

****

GOTO eli mene jonnekin

Tärkeä ja keskeinen ajatus tietokoneen ohjelmoinnissa on se, ettei ohjelmaa välttämättä suoriteta siinä järjestyksessä kuin se on kirjoitettu. Käskyä GOTO käytetään hyppäämiseen. Se kertoo tietokoneelle, mikä rivi sen on seuraavaksi suoritettava — vaikka se ei olisikaan järjestyksessä seuraava. Sitä käytetään myös ns. silmukan tekemiseen, jolloin samaa ohjelmakohtaa toistetaan jatkuvasti.
Ensimmäinen tässä kirjassa ollut ohjelma oli esimerkki silmukasta.

```
10 PRINT ''TAMA ON COMMODORE-TIETOKONE''
20 GOTO 10
```

Lause GOTO 10 tarkoittaa ''mene riville 10''. Tämä on täydellinen esimerkki päättymättömästä silmukasta: kun se on alka-

nut, ohjelma jatkuu loputtomiin.

GOTO-käskyä ei aina suinkaan käytetä silmukoiden tekemiseen.

```
10 PRINT ''TAMA ON TIETOKONEENI''
20 GOTO 40
30 PRINT ''TAMA RIVI EI TOIMI''
40 PRINT ''TAMA ON TIETOKONEENI''
```

Vaikka ohjelmassa on kolme ruutuun kirjoitettavaa riviä, rivillä 20 (GOTO 40) käsketään jättämään pois se rivi, jolla lukisi TAMA RIVI EI TOIMI.

Ei tietenkään kannata kirjoittaa ohjelmaa, jossa on tarpeettomia rivejä. Mutta jos voimme käskeä tietokonetta tekemään päätöksiä muun tiedon perusteella, tällainen hyppykäsky tulee tarpeelliseksi.

****

UNDEFINED STATEMENT ERROR

Sinun on käytettävä käskyjen GOTO ja GOSUB yhteydessä sellaisia rivinumeroita, jotka ovat todella käytössä ohjelmassa. Tämä virheilmoitus esiintyy usein silloin, kun editoit tai muutat rivinumerointia, mutta unohdat muuttaa rivinumeroihin viittaavat GOTO- ja GOSUB-käskyt. Tarkista ne rivinumerot, joilla virhe esiintyy, ja yritä sitten listata ne rivit. Koska käskyä RUN voi käyttää myös rivinumeroiden kanssa, saattaa tällainen virheilmoitus johtua siitäkin.

****

IF, THEN ja OR-käskyt eli ehtojen asettaminen

Suurin osa tietokoneen tehokkuudesta johtuu sen kyvystä tehdä yksinkertaisia päätöksiä, jotka perustuvat yksinkertaisiin kysymyksiin tai testeihin. Sana IF tarkistaa näitä ehtoja ja mahdol-

listaa päätösten teon. THEN-käskyä käytetään aina IF-käskyn kanssa ilmoittamaan koneelle, mitä sen tulee seuraavaksi tehdä. Voit kirjoittaa seuraavan ohjelman käyttäen sinulle jo tuttuja sanoja:

```
10 INPUT "VASTAA KYLLA TAI EI. PIDATKO
   TIETOKONEISTA";A$
20 IF A$="KYLLA" THEN PRINT "HYVA. OPI
   VAIN ENEMMAN NIISTA."
30 IF A$="EI" THEN PRINT "HYVASTI"
```

Tämä on tässä kirjassa ensimmäinen ohjelma, joka edellyttää päätöksen tekemistä. Se kysyy PIDATKO TIETOKONEISTA? käyttämällä käskyä INPUT rivillä 10. Vastaus kysymykseen — KYLLA tai EI — varastoidaan muuttujaan A$.

Tämän jälkeen ohjelma tekee päätöksensä, joka itse asiassa on yksi kolmesta vaihtoehdosta. Rivillä 20 se kysyy, onko vastaus (A$) KYLLA. Jos on, se kirjoittaa ruutuun vastaukseksi HYVA. OPI VAIN ENEMMAN NIISTA. Jos vastaus ei ole KYLLA, se siirtyy seuraavalle riville ja kysyy, onko vastaus EI. Jos on, se vastaa lyhyesti HYVASTI. Jos vastaus ei ole kumpikaan — KYLLA tai EI — koska sinähän olisit voinut vastata millä tahansa sanalla, se ei kirjoita vastaukseksi yhtään mitään.

Voit käyttää (ja usein se onkin tarpeen) käskyä IF paitsi merkkimuuttujien myös lukujen kanssa. Sen lisäksi että voit sovittaa kysymyksiin numeroituja vastauksia, voit myös verrata *lukualueita* käyttämällä jotain muuta merkkiä kuin yhtäläisyysmerkkiä. Tätä tarkoitusta varten koneessa on merkit "pienempi kuin" (<) ja "suurempi kuin" (>). Jos sinun on vaikea muistaa, kumpi on kumpi, kuvittele niitä suppiloiksi. Avoimesta päästä menee aina enemmän sisään, suljetusta päästä tulee vähemmän ulos.

```
10 INPUT "KUINKA VANHA OLET";A
20 IF A<18 THEN PRINT "OLET ALLE 18-VUOTIAS."
30 IF A>18 THEN PRINT "OLET YLI 18-VUOTIAS."
```

Kuten edellisessä ohjelmassa tässäkin koneen vastaukset perustuvat kysymykseen annettuun vastaukseen. Se on kuitenkin erilainen kuin KYLLÄ/EI-esimerkki, koska se tarkistaa, kuuluuko vastaus johonkin laajempaan ikäryhmään, alle 18-vuotiaisiin vai yli 18-vuotiaisiin. Rivi 20 sanoo, että jos A (vastauksen lukuarvo) on *pienempi kuin* 18, kone kirjoittaa ruutuun tietyn vastauksen, ja jos A on *suurempi kuin* 18, se kirjoittaa toisen vastauksen.

Tässä ohjelmassa on ongelmana se, että se ei ota huomioon sitä mahdollisuutta, että vastaaja olisi nimenomaan 18-vuotias. Luvun 18 kirjoittaminen ei antaisi tulokseksi mitään vastausta, koska ohjelma testaa vain 18:aa suurempia tai pienempiä lukuja.

Eräs ratkaisu on lisätä uusi rivi:

```
40 IF A = 18 THEN PRINT "OLET TASAN 18-VUOTIAS."
```

Nyt jokainen vastauksesi kirjoitettu luku tuottaa koneelta jonkinlaisen vastauksen.

On toinenkin ratkaisu. Merkkejä > ja < voidaan käyttää yhtäläisyysmerkin kanssa merkitsemään jotakin uutta asiaa.

```
20 IF A < = 18 THEN PRINT "OLET TASAN TAI ALLE
   18-VUOTIAS."
30 IF A > = 18 THEN PRINT "OLET TASAN TAI YLI
   18-VUOTIAS."
```

Rivi 20 merkitsee: jos A on *pienempi tai yhtä suuri kuin* 18. Rivi 30 merkitsee: jos A on *suurempi tai yhtä suuri kuin* 18.

Ja lopuksi: voit käyttää kahta IF-käskyä yhdessä ainoassa lauseessa verrataksesi lukualueita.

```
10 INPUT "KUINKA VANHA OLET";A
20 IF A > = 13 AND IF A < = 19 THEN PRINT
   "OLET TEINI-IKAINEN."
30 IF A < 13 THEN PRINT "LIIAN NUORI."
40 IF A >19 THEN PRINT "LIIAN VANHA."
```

Rivit 30 ja 40 ovat ilmeisen selkeitä. Ne vain testaavat, onko vastaus A pienempi kuin 13 tai suurempi kuin 19, jotka ovat teini-iän rajat. Rivi 20 merkitsee: jos A on *suurempi tai yhtä suuri kuin* 13 *JA* jos A on *pienempi tai yhtä suuri kuin* 19, niin kirjoita ruutuun. Vastausta OLET TEINI-IKAINEN ei ilmaannu kuvaruutuun, elleivät nämä molemmat ehdot täyty. Sana AND tarkistaa aina, ovatko kaikki IF-ehdot totta. Samalla IF-rivillä voidaan käyttää lukemattomia AND-sanoja.

OR-operaattori tarkistaa, onko kumpikaan (tai mikään) kahdesta (tai useammasta) IF-testistä tosi.

```
10 INPUT "KUINKA VANHA OLET";A
20 IF A< =12 OR A> =20 THEN PRINT
   "ET OLE TEINI-IKAINEN."
```

Viimeinen tapa käyttää merkkejä < ja > on käyttää niitä yhdessä, siis <> . Tämä on yhtäläisyysmerkin vastakkaismerkki ja tarkoittaa "eri suuri kuin". Pieni arvausleikki:

```
10 PRINT"AJATTELEN NUMEROA VALILTA 1—100"
20 INPUT"ARVAA MIKA SE ON";A
30 IF A <> 56 THEN PRINT "EI OLE":GOTO 10
40 IF A=56 THEN PRINT "OIKEIN!"
```

Huomaat varmasti, että peli on ennalta järjestetty. (Vastaushan on aina 56.) Ensin kone ilmoittaa AJATTELEN NUMEROA VALILTA 1—100 ja sitten kysyy INPUT-lauseen avulla ARVAA MIKA SE ON? Sitten se odottaa, kunnes arvaus on kirjoitettu. Rivi 30 käskee tietokoneen kirjoittaa EI OLE, jos valitsemasi luku (A) on *eri suuri kuin* 56, ja kehottaa sitten palaamaan ohjelman alkuun. (Jos luku on juuri 56, saat kiitosta.)

Käsky IF toimii järkähtämättömän loogisesti. Paitsi että sitä voidaan käyttää käskyjen PRINT ja GOTO yhteydessä, kuten näissä esimerkeissä, sitä voidaan käyttää muuttujien uudelleenmäärittämiseen ja melkein minkä tahansa muunkin BASIC-käskyn kanssa. Aloittelevia ohjelmoijia hämää kuitenkin se, et-

tä jos IF-ehto on *väärä,* kaikki sen jälkeen samalla ohjelmarivillä oleva jää huomiotta. On siis hyvin tärkeää muistaa, ettei sanalla IF alkavalle riville saa kirjoittaa *mitään* sellaista, joka ei riipu kyseisen testin tai kysymyksen tuloksesta tai vastauksesta.

```
10 INPUT "KYLLA VAI EI";A$
20 IF A$="KYLLA" THEN PRINT "VASTASIT
   KYLLA":A=100
```

Kun vastaus on EI, A:n arvoksi jää nolla. Kun vastaus on KYLLA, A:n arvoksi tulee 100. Toisinaan ohjelmoijat unohtavat, että rivillä on sana IF, ja lisäävät sinne tärkeitä käskyjä. Ne jäävät aina huomiotta, ellei IF-ehto ole tosi. Ole tässä erityisen huolellinen, koska näitä virheitä on toisinaan hyvin vaikea löytää ohjelmasta, eikä tästä virheestä ole omaa virheilmoitusta.

Käskyllä GET annat lisää tietoa

INPUT-käskyn lisäksi voit käyttää toistakin sanaa varastoimaan näppäimistöltä annettua tietoa. Tämä sana on GET.

Päinvastoin kuin INPUT-käsky GET-käsky ei jää odottamaan käyttäjän vastausta. INPUT hyväksyy useampiakin merkkejä tai numeroita sisältävät tiedot, GET taas hyväksyy korkeintaan yhden merkin kerrallaan. Ja INPUT odottaa, kunnes painetaan näppäintä RETURN, joka ilmaisee kaikki tiedot annetuiksi, mutta GET siirtyy automaattisesti takaisin ohjelmaan sen jälkeen kun tietokone on tarkistanut onko jotain näppäintä painettu. Jollei mitään näppäintä ole painettu, palauttaa GET tyhjän merkkijonon muuttujaan.

Käskyä GET on aina seurattava merkkimuuttuja ja sitä käytetään lähes poikkeuksetta IF-käskyn kanssa.

```
10 PRINT "PAINA JOTAIN NAPPAINTA LOPETTAAK-
   SESI"
```

```
20 GET C$:IF C$ = " " THEN GOTO 20
30 PRINT "LOPPUI"
```

Edellistä esimerkkiä käytetään usein Commodoren tietokoneille kirjoitetuissa BASIC-ohjelmissa. Sen tarkoituksena on pysäyttää ohjelman suoritus, kunnes käyttäjä haluaa jatkaa. Tämä on kätevää esimerkiksi ruutuun kirjoitetun tiedon pitämiseksi näkyvillä niin kauan että sen ehtii lukea tai että pelaajalla on aikaa ajatella ennen kuin aloittaa pelaamisen.

Tässä esimerkissä ohjelma kirjoittaa ensin ruutuun viestin PAINA JOTAIN NAPPAINTA LOPETTAAKSESI. Sen jälkeen käsky GET tarkistaa, onko jotain näppäintä painettu. IF-lause hoitaa varsinaisen tarkistamisen. Se kysyy, onko C$:n arvo "". Näitä kahta lainausmerkkiä, joiden välissä ei ole mitään, ei edes tyhjää väliä, kutsutaan *tyhjäksi* merkkijonoksi. Jos C$:n arvo on tyhjä, se tarkoittaa, ettei mitään näppäintä ollut painettu ja ohjelma palaa GET-käskyllä saman rivin alkuun. Kun sitten jotain näppäintä painetaan ja C$ saa jonkin muun arvon kuin "", ohjelma jatkuu GET-riviä seuraavalta riviltä.

On kuitenkin eräitä näppäimiä, joiden painamisella ei ole mitään vaikutusta GET-käskyyn. Nämä ovat molemmat SHIFT-näppäimet sekä näppäimet SHIFT LOCK, RESTORE, CTRL ja COMMODORE. Ei ole myöskään mahdollista painaa näppäintä RUN/STOP, koska se pysäyttäisi ohjelman suorituksen kokonaan.

GET-käskyn pääasiallinen tehtävä ei tietenkään ole tämä ohjelman suorituksen keskeytys. Sen tarkoitus on saada tietty arvo — luku tai merkkijono — siihen liitetylle muuttujalle.

```
10 PRINT "KIRJOITA K=KYLLA, E=EI"
20 GET A$:IF A$ = "" THEN GOTO 20
30 IF A$ = "K" THEN PRINT "VASTASIT KYLLA"
40 IF A$ = "E" THEN PRINT "VASTASIT EI"
50 IF A$ <> "K" OR IF A$ <> "E" THEN GOTO 10
```

Tässä tapauksessa merkkimuuttujaan A$ varastoituu kirjain. Ohjelma etsii joko kirjainta K tai E ja tekee sitten päätöksensä

niiden perusteella. Rivillä 50 hylätään *kaikki* muut mahdollisuudet. Se merkitsee: jos A$ *ei ole sama kuin* ''K'' tai jos A$ *ei ole sama kuin* ''E'' niin mene riville 10. (Eli riville joka pyytää kirjoittamaan K tai E.)

Voit käyttää käskyä GET valitaksesi tietyn vaihtoehdon jostain kuvaruudussa olevasta listasta. Tietokoneslangissa tällaista valintataulua eli valikkoa kutsutaan nimellä 'menu'. Kokeillaanpa valikkoa, jossa on neljä eri valintamahdollisuutta. Numeronäppäimien sijaan me käytämme kuitenkin koneen oikeassa laidassa olevia merkillisiä ''toimintanäppäimiä'' eli ''funktionäppäimiä'', joita emme ole vielä kokeilleet. Näppäinten yläosassa on merkit ''f1'', ''f3'', ''f5'' ja ''f7''. Tietokoneelle ne ovat aivan samanlaisia kuin muutkin näppäimet, vaikka ne eivät painettaessa aiheutakaan mitään kuvaruutumerkintöjä. Kun kirjoitat tämän esimerkin, tee, mitä sinun käsketään tehdä hakasulkeiden ''[]'' välissä. *Älä kirjoita* niiden välissä olevia sanoja.

```
10 PRINT ''[Paina yhtäaikaa näppäimiä SHIFT ja
   CLR/HOME]''
20 PRINT ''VALITSE JOKIN AIHE LISTALTA''
30 PRINT ''PAINA OIKEAA FUNKTIONAPPAINTA''
40 PRINT ''F1--ENSIMMAISEN KUUHUN LASKEUTU-
   MISEN VUOSILUKU''
50 PRINT ''F3--EIFFEL-TORNIN KORKEUS''
60 PRINT ''F5--VALON NOPEUS''
70 PRINT ''F7--VEDEN KIEHUMISPISTE''
80 GET C$:IF C$=" " THEN 80
90 IF C$= ''[Paina näppäintä f1]'' THEN PRINT ''1969''
100 IF C$= ''[f3]'' THEN PRINT ''316 METRIA''
110 IF C$= ''[f5]'' THEN PRINT ''299 792 KILOMETRIA
    SEKUNNISSA''
120 IF C$= ''[f7]'' THEN PRINT ''100 ASTETTA''
130 PRINT ''JATKUU PAINAMALLA JOTAIN NAP-
    PAINTA''
140 GET C$:IF C$=" " THEN 140
150 GOTO 10
```

Tässä esimerkissä rivi 10 tyhjentää kuvaruudun. Riveillä 20 ja 30 annetaan ohjeet valita jokin aihe listalta ja painaa asianmukaista toiminäppäintä. Rivit 40—70 kirjoittavat valikon kuvaruutuun. Rivillä 80 oleva GET-käsky aiheuttaa pysähdyksen samalla tavalla kuin aikaisemmassakin esimerkissä.

(Huomasit varmaankin, että riveiltä 80—140 puuttuu jotakin. Sen sijaan että sanottaisiin GOTO 80, rivi 80 ilmoittaa vain IF C$ = "" THEN 80. Tietokone on kyllin taitava tietääkseen, että THEN-sanaa seuraava numero on aina se rivinumero, jolle sen pitää hypätä.)

Rivillä 90 näet, kuinka toiminäppäimiä käytetään. Yleensä kuvaruutuun ei ilmaannu mitään merkkejä silloin kun toiminäppäimiä painetaan. Mutta kun kirjoitat lainausmerkin ensin, näet kutakin kirjoittamaasi näppäintä kohti erityisen merkin. Näppäimiä f1 ja f3 painettuasi näet käänteistaustalla olevan laatikon, jonka läpi kulkee vaakasuora viiva. Jos painat näppäimiä f5 ja f7, näet samanlaisen laatikon, jossa on pystysuora viiva. Kun tietokone näkee nämä merkit esimerkiksi sellaisissa lauseissa kuin riveillä 90—120, se pitää toiminäppäimiä aivan samanlaisina kuin muitakin näppäimiä.

Kun toiminäppäintä painetaan, ohjelma kirjoittaa asianmukaiset tiedot — ensimmäisen kuuhun laskeutumisen vuosiluvun, Eiffel-tornin korkeuden jne. — ja jää odottamaan. Sitten kuvaruutuun ilmaantuu ohje JATKUU PAINAMALLA JOTAIN NAPPAINTA. Minkä tahansa näppäimen painaminen aloittaa ohjelman uudelleen viimeisellä rivillä olevan GOTO-käskyn mukaisesti.

FOR ja NEXT tekevät silmukoita

Olet jo nähnyt, kuinka käskyä GOTO voidaan käyttää silmukan tekemiseen siten, että ohjelma kiertää samaa kohtaa jatkuvasti. Olet myös nähnyt, että tietokone jatkaa silmukkaa loputtomiin, ellei se saa muuta käskyä.

On useita tapoja käskeä ohjelmaa toistamaan samaa silmuk-

kaa tietyn ajan. Yksinkertaisinta on käyttää FOR/NEXT-silmukkaa.

```
10 FOR I=1 TO 20
20 PRINT ''TAMA ON COMMODORE-TIETOKONE''
30 NEXT I
```

Tämä lyhyt ohjelma on miltei samanlainen kuin tämän kirjan ensimmäinen ohjelma. Ensimmäisessä ohjelmassa tietokoneen käskettiin kirjoittaa lausetta TAMA ON COMMODORE-TIETOKONE niin kauan kunnes painetaan RUN/STOP-näppäintä. Tässä uudessa versiossa sitä käsketään kirjoittamaan lausetta vain 20 kertaa.

Tavallaan FOR on kuin LET (mutta paljon tärkeämpi). Se määrittelee numeerisen muuttujan, joka tässä tapauksessa on I. Sitten se käskee tietokoneen ensiksi antaa I:lle arvon 1. Kun tietokone kohtaa käskyn NEXT, se palaa riville, jossa FOR on, ja muuttaa I:n arvoksi seuraavan luvun eli 2, sitten 3 jne. Tämä silmukan tekeminen jatkuu, kunnes FOR-käskyssä oleva muuttuja saavuttaa sanan TO toisella puolella olevan arvon.

Todetaksesi, että I:llä merkitty muuttuja todella kasvaa, kirjoita rivi 20 uudelleen:

```
20 PRINT I;''TAMA ON COMMODORE-TIETOKONE''
```

Nyt ohjelma numeroi jokaisen rivin aina kun se kirjoittuu kuvaruutuun käyttäen I:n arvoa.

Samalla lailla voit käyttää hyödyksesi muuttujan arvoa. Seuraava ohjelma kirjoittaa ruutuun kertotaulun:

```
10 PRINT ''[paina näppäimiä SHIFT ja CLR/HOME]
   18:N KERTOTAULU''
20 FOR I=1 TO 20
30 PRINT I;''KERTAA 18 ON'';I*18
40 NEXT I
```

Näet tästä esimerkistä, että muuttujaa I käytetään sekä lukuna, jolla 18 kerrotaan, että tarvittavan tuloksen laskemiseen kaavalla I*18.

Tällaisia FOR/NEXT-silmukoita voidaan kirjoittaa toistensa sisäänkin. Sisäkkäisiä silmukoita käytetään yleensä silloin, kun jotakin muuttujaa on kasvatettava toisesta riippumatta. Voimme muuntaa edellä olevan ohjelman laskemaan kaikki kertotaulut väliltä 1—100 lisäämällä uuden FOR/NEXT silmukan.

```
10 FOR I=2 TO 100
20 PRINT ''[paina näppäimiä SHIFT ja CLR/HOME]
   LUVUN'';I;''KERTOTAULU''
30 FOR J=1 TO 20
40 PRINT J;''KERTAA'';I;''ON'';J*I
50 NEXT J
60 PRINT ''JATKUU PAINAMALLA JOTAIN NAP-
   PAINTA''
70 GET C$:IF C$='' '' THEN 70
80 NEXT I
```

Nämä sisennetyt silmukat saattavat näyttää jossain määrin sekavilta, mutta ne eivät ole sellaisia. Rivillä 10 valitset, kuinka monta kertotaulua haluat koneen kirjoittavan — 99. Se asettaa ensimmäisen (ulomman) silmukan, FOR I=2 TO 100. Rivi 20 kirjoittaa kunkin kertotaulun ''otsikon''. Kun I on 2, tietokone kirjoittaa LUVUN 2 KERTOTAULU.

Rivi 30 asettaa toisen (sisemmän) silmukan, FOR J=1 TO 20. Tämä taas on kunkin kertotaulun koko. Rivi 40 kirjoittaa jokaiselle riville luvun (J) KERTAA taulukon numero (I) ON ja lopulta tuloksen J kertaa I. Rivi 50 lopettaa toisen silmukan. Riveillä 60 ja 70 käytetään pikku temppua, jotta voimme pysäyttää ohjelman hetkeksi GET-käskyn avulla. Tämä pitää kertotaulut ruudussa niin, että voimme lukea ne. Jos tätä GET-riviä ei olisi, kertotaulut kirjoittuisivat ruutuun niin nopeasti, että tuskin pystyisit erottamaan niitä toisistaan.

Viimeinen rivi, rivi 80, lopettaa ensimmäisen silmukan.

NEXT I käskee koneen kirjoittaa uuden, tällä kertaa seuraavan luvun kertotaulun.

Voimme lisätä kertotauluohjelmaamme kolmannenkin silmukan. Poista rivit 60 ja 70 ohjelmasta (kirjoittamalla kumpikin rivinumero ja painamalla niiden jälkeen RETURN-näppäintä). Lisää sitten seuraava rivi:

```
60 FOR K=1 TO 1000:NEXT K
```

Nyt ohjelmassa on kolme erillistä silmukkaa, jotka toimivat eli ”pyörivät” toistensa sisällä. Kuten huomaat, tämä viimeinen on tyhjä silmukka, joka pelkästään lisää K:n arvoa joka kerran kohdatessaan sanan NEXT K. Sen tarkoitus (joka on varsin yleinen FOR/NEXT silmukoiden tarkoitus) on vain lisätä ohjelmaan viivettä. Nyt ohjelma seisahtuu pariksi sekunniksi jokaisen kirjoitetun kertotaulun jälkeen. Aikaa ei oikeastaan ole kylliksi, jotta ehtisit lukea jokaisen kertotaulun, mutta silmukka jatkaa ohjelman suoritusta hiukan hitaammin, vaikka et painelisikaan mitään näppäimiä.

Mitä pienempi on K:n lukualue (1 TO mitä vain), sen vähemmän aikaa siltä menee silmukan suorittamiseen. Kokeile jotain muutakin lukua.

Tulet huomaamaan, että muuttujia I, J ja K käytetään varsin usein BASIC-ohjelmissa nimenomaan FOR/NEXT silmukoiden yhteydessä. Näillä kirjaimilla ei sinänsä ole mitään merkitystä. (Itse asiassa voit käyttää mitä tahansa sallittua muuttujanimeä.) Nämä ovat kuitenkin osa tiettyä ohjelmointikäytäntöä, joka on asettanut ne muista muuttujanimistä erilleen juuri tätä tarkoitusta varten.

On tärkeää muistaa eräitä seikkoja FOR/NEXT silmukoiden yhteydessä. Ensiksikin muista mitä olet tehnyt. Menet helposti sekaisin, etenkin jos käytät useampaa kuin kahta sisäkkäistä silmukkaa. Toiseksi koeta pitää sellaiset hyppykäskyt kuin GOTO poissa silmukoista, koska niiden käyttö johtaa helposti sekaisin meneviin ohjelmiin.

Käsky, jota käytetään yksinomaan FOR/NEXT-käskyn yh-

teydessä, on STEP. Se kertoo tietokoneelle, kuinka pitkiä askeleita sen on otettava aina kun FOR-lauseen muuttujaa kasvatetaan.

```
10 FOR I=1 TO 20 STEP 2
20 PRINT I;"TAMA ON COMMODORE-TIETOKONE"
30 NEXT I
```

Tässä STEP 2 merkitsee sitä, että tietokoneen on lisättävä I:n lukuarvoa 2:lla 1:n sijasta. Kun suoritat tämän ohjelman, huomaat, että se numeroi rivit 1, 3, 5, 7, 9 jne. Kokeile samaa esimerkkiä vaihtamalla STEP-käskyn jälkeinen numero.

Voit myös askeltaa STEP-käskyllä takaperin silmukassa. Muuta edellisen esimerkin rivi 10:

```
10 FOR I=20 TO 1 STEP -1
```

Tämä negatiivinen luku (-1) numeroi jokaisen rivin laskevassa järjestyksessä 20:sta 1:een.

****

NEXT WITHOUT FOR ERROR

Tämä virheilmoitus johtuu siitä, että olet käyttänyt sanaa NEXT aukaisematta ensin silmukkaa käskyllä FOR. Tämä tapahtuu silloin, jos sanaa NEXT on käytetty väärän muuttujan kanssa.

```
10 FOR I=1 TO 10
20 NEXT K
```

Jos tapahtuu päinvastoin eli käskyä FOR käytetään ilman sanaa NEXT, ei mitään virheilmoitusta tule. Silmukka ei siinä tapauksessa vain toimi niin kuin sen pitäisi.

Tämä virheilmoitus tulostuu myös jos GOTO- tai GOSUB-käskyllä hypätään silmukan ulkopuolelta sen sisään.

****

GOSUB ja RETURN ovat aliohjelmakäskyjä

GOTO on hyvin suoraviivainen BASICin sana. Se merkitsee, että ohjelman tulisi mennä jonnekin. GOSUB on lähes yhtä helppo ymmärtää. GOTO-käskyn lailla sitäkin käytetään rivinumeron kanssa ohjaamaan ohjelman suoritus tietylle riville. Toisin kuin GOTO GOSUB myös muistaa, missä kohtaa ohjelmaa se oli menossa, ja se voi palata käskyllä RETURN siihen kohtaan lopetettuaan omat puuhansa. Itse sana GOSUB on lyhennys englannin sanoista "GO to the SUBroutine", suomeksi mene aliohjelmaan.

Aliohjelmat ovat olennainen osa tietokoneen ohjelmointia. Kuten nimikin kertoo, aliohjelmat ovat lyhyitä toisiin ohjelmiin sisältyviä ohjelmia. Kukin aliohjelma hoitaa tavallisesti pienen tehtävän, joka on suoritettava monta kertaa. Jottei tarvitsisi kirjoittaa samoja BASIC-rivejä yhä uudelleen eri paikkoihin, ne kirjoitetaan vain kerran aliohjelmaksi. Toisaalta aliohjelma voi olla hyvinkin suuri ohjelman osa, joka on kirjoitettu aliohjelmaksi järjestyksen vuoksi.

Aina kun tietokone kohtaa käskyn GOSUB, se siirtyy aliohjelman ensimmäiselle riville ja suorittaa ohjelmaa siitä lähtien. Kun se kohtaa sanan RETURN, se palaa takaisin siihen mistä lähti aliohjelmaan, vaikka vain keskelle edellistä riviä. (BASIC-käskyllä RETURN ei ole yhtään mitään tekemistä RETURN-näppäimen kanssa — siinä toinen syy, miksi näppäintä voisi kutsua nimellä ENTER.)

Olet ilmeisesti huomannut, että GOSUB voi olla hyvin tehokas käsky, paljon hyödyllisempi kuin GOTO voisi koskaan olla.

Palautetaan mieleen GET-käskyn yhteydessä laadittu menuohjelma. Yritetään kirjoittaa se aliohjelmina.

Tässä ensiksi aliohjelma, joka tyhjentää kuvaruudun ja esittää menun:

```
100 PRINT ''[Paina yhtä aikaa näppäimiä SHIFT ja
    CLR/HOME]''
110 PRINT ''VALITSE JOKIN AIHE LISTALTA''
120 PRINT ''PAINA OIKEAA FUNKTIONAPPAINTA''
130 PRINT ''F1--ENSIMMAISEN KUUHUN LASKEUTU-
    MISEN VUOSILUKU''
140 PRINT ''F3--EIFFEL-TORNIN KORKEUS''
150 PRINT ''F5--VALON NOPEUS''
160 PRINT ''F7--VEDEN KIEHUMISPISTE''
170 RETURN
```

Jo nämä rivit tyhjentävät kuvaruudun ja kirjoittavat siihen menun. Ne toimisivat jo sinänsä ohjelmana, ellei viimeisellä rivillä olisi RETURN-käsky. (Jos yrität suorittaa aliohjelman joka loppuu RETURN-käskyyn, saat virheilmoituksen.)

****

RETURN WITHOUT GOSUB ERROR

GOSUB ja RETURN toimivat yhdessä. Tämä virheilmoitus ilmestyy silloin, jos ohjelma kohtaa käskyn RETURN eikä sille ole annettu käskyä GOSUB. Se tavataan useimmiten silloin, kun kyseistä ohjelmariviä kutsutaan käskyllä GOTO.

****

Huomasitko, että edelläoleva aliohjelma alkaa rivinumerosta 100, ei 10. Tällä tavalla erotamme sen ohjelman perusrungosta.

Hyödyllisempi aliohjelma olisi sellainen, joka yksinkertaistaa ohjelmaa vähentämällä ohjelmarivien uudelleenkirjoittamisen tarvetta. Alkuperäinen menuohjelma käytti GET-käskyä kahdesti. Tämäkin voidaan kirjoittaa vain yhdeksi aliohjelmaksi:

```
200 GET C$:IF C$ = '' '' THEN 200
210 RETURN
```

Nyt meillä on kaksi aliohjelmaa: aliohjelma 100, joka kirjoittaa menun, ja aliohjelma 200, joka suorittaa GET-käskyn. Kun nyt

tiedämme näiden kahden sijainnin, voimme kirjoittaa ohjelman perusrungon. Kun osat on yhdistetty, ohjelma näyttää seuraavalta:

```
10 GOSUB 100
20 GOSUB 200
30 IF C$= ''[Paina näppäintä f1]'' THEN PRINT ''1969''
40 IF C$= ''[f3]'' THEN PRINT ''316 METRIA''
50 IF C$= ''[f5]'' THEN PRINT ''299 792 KILOMETRIA
   SEKUNNISSA''
60 IF C$= ''[f7]'' THEN PRINT ''100 ASTETTA''
70 PRINT ''JATKUU PAINAMALLA JOTAIN NAP-
   PAINTA''
80 GOSUB 200
90 GOTO 10
100 PRINT ''[Paina yhtä aikaa näppäimiä SHIFT ja
    CLR/HOME]''
110 PRINT ''VALITSE JOKIN AIHE LISTALTA''
120 PRINT ''PAINA OIKEAA FUNKTIONAPPAINTA''
130 PRINT ''F1--ENSIMMAISEN KUUHUN LASKEUTU-
    MISEN VUOSILUKU''
140 PRINT ''F3--EIFFEL-TORNIN KORKEUS''
150 PRINT ''F5--VALON NOPEUS''
160 PRINT ''F7--VEDEN KIEHUMISPISTE''
170 RETURN
200 GET C$:IF C$='' '' THEN 200
210 RETURN
```

Tämä ohjelma suoritetaan aivan samalla tavalla kuin aikaisempikin menuohjelma, mutta se on järjestetty eri tavalla. Kaikkein hyödyllisin muutos on se, että ohjelma siirtyy riville 200 joka kerta kun tarvitaan GET-käskyn avulla aikaansaatu keskeytys.

Aliohjelmien käyttöön liittyy eräs vaara. Aliohjelmat on selkeästi erotettava itse ohjelman perusrungosta. Yritä suorittaa menuohjelma poistettuasi ensin rivin 90 (kirjoittamalla rivinumeron ja painamalla näppäintä RETURN). Näet virheilmoituk-

sen, eikä ohjelmaa voi suorittaa oikein.

Tämä johtuu siitä, että tietokone ei osaa lopettaa. Se ei ymmärrä, että rivillä 100 alkaa aliohjelma, se vain jatkaa ja yrittää suorittaa siitä eteenpäin olevia rivejä aivan kuin ne kuuluisivat pääohjelmaan.

Paras tapa erottaa pääohjelma aliohjelmista on käyttää käskyä END. Voit pitää sitä tavallaan pääohjelman ja aliohjelman välisenä aitana.

```
90 END
```

Nyt menuohjelma voidaan suorittaa loogisesti, vaikka se ei toimikaan jatkuvasti kuten aikaisemmin. Muistat varmaan, että käskyä END ei Commodoren BASICissä välttämättä tarvittaisi. Tämä tapaus on poikkeus, mutta samalla myös hyvin käytännöllinen tapa käyttää tätä käskyä.

Ovatko aliohjelmat sitten ainoastaan uusi tapa järjestää ohjelmia? Eivät. Paitsi että GOSUB auttaa pitämään ohjelmien rivien määrän minimissä, se myös tekee niistä entistä luettavampia. Kukin aliohjelma on helppo havaita, koska ne alkavat GOSUB-käskyä seuraavasta rivinumerosta ja päättyvät käskyyn RETURN.

BASIC-ohjelmia arvostellaan usein siitä, että kaikkien muiden kuin ohjelman alunperin kirjoittaneen ohjelmoijan on hankala niitä lukea. Voi käydä niinkin, että kirjoitat ohjelman, ja puoli vuotta tai vuosi myöhemmin et itsekään pysty lukemaan sitä. Aliohjelmista on siinä apua. Ohjelmaa, jossa käytetään aliohjelmia aina kun se on mahdollista ja käytännöllistä, kutsutaan joskus *strukturoiduksi ohjelmaksi,* ja monet ohjelmoijat pitävät juuri tätä hyvänä BASIC-*tyylinä.*

REM-käskyä käytetään huomautuksiin

Ohjelmien tekeminen luettaviksi on hyvä ohjenuora ohjelmoijille, etenkin aloitteleville. Kelvollisen kirjoitustyylin luominen

on yhtä tärkeää BASICissä kuin äidinkielessäkin. On itse asiassa olemassa BASIC-käsky, joka sallii kirjoittaa ohjelmaan mitä tahansa kieltä, jota näppäimistöltä voi kirjoittaa. Tämä käsky on REM, joka tarkoittaa huomautusta, ja jotkut pitävät sitä muistin apukeinona. Ohjelmaan voidaan kirjoittaa melkein millainen huomautus tahansa, kunhan sen edessä on käsky REM.

```
10 REM KOKO TAMA RIVI ON HUOMAUTUS
```

Sanaa REM voidaan käyttää myös ohjelmarivin lopussa, mikäli sen edessä on kaksoispiste. Kaikki ennen kaksoispistettä olevat käskyt toimivat normaalisti, ja kun tietokone huomaa sanan REM, se jättää huomiotta sitä seuraavat huomautukset.

```
10 PRINT "REM EROTETAAN MUUSTA KAKSOISPIS-
   TEELLA" :REM — KUTEN MUUTKIN SANAT
```

Commodoren BASICille ominainen piirre on se, että se sallii REM-käskyn jälkeen vain ilman näppäintä SHIFT painetut kirjaimet. Voit tarkastella tätä ongelmaa kirjoittamalla näin:

```
10 REM [paina SHIFT ja kirjoita ABCDEFGHI]
```

Kirjoita LIST ja katso tulosta.

```
10 REM ATNPEEKLENSTR$VALASCCHR$LEFT$
   RIGHT$
```

Tulos ei lainkaan näytä alunperin kirjoittamaltasi huomautukselta. (Tietokone nimittäin tulkitsee SHIFT-näppäimen kanssa yhtä aikaa kirjoitetut kirjaimet BASIC-käskyiksi.) Siitä ei kuitenkaan tule ongelmaa, kunhan muistat aina kirjoittaa huomautukset käyttämättä näppäintä SHIFT. Grafiikkatilassa kirjoitat siis ainoastaan isoilla kirjaimilla ja kirjoituskonetilassa ainoastaan pienillä kirjaimilla.

Vielä muutama vuosi sitten tietokoneen muistisirut olivat kalliita ja niitä oli vähän. Kotitietokoneissa oli suhteellisen vaatimaton muisti ja ohjelmoijien oli usein pakko käyttää jokainen tavu huolellisesti hyväksi. Koska REM-käskyn jälkeen tulevat huomautukset saattavat viedä runsaasti muistitilaa (yksi tavu jokaista merkkiä, kirjainta tai numeroa kohden), niitä ei käytetty niin usein kuin olisi ehkä pitänyt.

Vaikka muistitilan säästäminen on edelleen hyväksi pitkissä ohjelmissa, se ei nykyisin ole tavattoman tärkeää. Commodore 64 -laitteisto antaa käytettäväksesi miltei viisinkertaisesti muistia verrattuna alkuperäisen Commodore PETin muistiin. Huomautuksia voidaan siis käyttää useammin. Ja jos teet niin, ohjelmasi on ehkä selkeämpi muille ja itsellesikin.

Muista kuitenkin, että tietokoneiden tulisi tehdä elämämme helpommaksi, ei hankaloittaa sitä. Mitä meistä olisi tullut, jos tärkeät oppikirjat, lähdeteokset ja tietosanakirjat olisivat liian vaikeita ymmärtää? Pidä se mielessäsi joka kerta kirjoittaessasi tietokoneohjelmaa.

5

Ohjelmointi — kuinka tietokone varastoi tietoa

Tiedon varastoiminen tietokoneeseen on helppoa. Tiedon järjestäminen on jo toinen juttu, samoin sen muuttaminen ja sen saaminen ulos koneesta. Tämä luku kertoo siitä.

Tiedon varastoiminen on tietokoneen hyödyllisimpiä käyttömahdollisuuksia. Kotona elektroninen arkistointijärjestelmä yksinkertaistaa talonpitoa. Koulussa voivat sekä opettaja että oppilaat käyttää samanlaista järjestelmää koearvostelujen ja kirjalistojen tuottamiseen. Tarkkojen muistiinpanojen tekeminen työssä on välttämätöntä, ja tietokone voi helpottaa jatkuvasti vaihtuvista asioista ja luvuista aiheutuvia ongelmia.

Commodore 64 -tietokoneiden suuri muistitila riittää yleensä lähes kaikkiin henkilökohtaisiin tarpeisiin. Laajaan ammattikäyttöön tarvitaan kuitenkin useimmiten levykkeitä muistitilan kasvattamiseksi. (Levykkeen muistitila on kolminkertainen tietokoneen muistiin verrattuna.)

Tässä luvussa tarkastelemme, kuinka tietokone, levykeasema ja kasettinauhuri järjestävät ja varastoivat tietoa tiedostoihin. Opit myös uusia muuttujatyyppejä. Useita BASICin sanoja

käytetään nimenomaan tiedon käsittelemiseen. Tällaisia sanoja ovat DATA ja READ, DIM, PRINT#, INPUT#, GET#, OPEN, CLOSE, RESTORE ja CLR. Kuinka tällaisia tiedonkäsittelyohjelmia eli *tietokantaohjelmia* sitten kirjoitetaan? Saat tietää, kuinka sellaiset ohjelmat toimivat ja kuinka ne suunnitellaan. Voit kirjoittaa esimerkkiohjelman omalla tietokoneellasi omaan käyttöösi.

Ja lopuksi saat sitten tietää jotakin niistä eduista, joita kaupoista ostettavilla tietokantaohjelmilla on, ja millaisia asioita tulisi ottaa huomioon ostoa harkitessa.

Uudenlaisia muuttujia

Muistat varmasti, että muuttujat ovat nimiä, joiden avulla tunnistetaan tietokoneeseen varastoitua tietoa, esimerkiksi:

```
A$ = "AUTOT"
B$ = "LINNUT"
C$ = "KAUPUNGIT"
D$ = "KOIRAT"
```

Tiedämme, että tämä on merkkimuuttujalista, koska muuttujanimessä on kirjain, jota seuraa dollarin merkki. Jokainen muuttuja merkitsee tiettyä asiakokonaisuutta — A$ autoja, B$ lintuja jne. Jokaisen tällaisen muuttujanimen alle voisimme kirjoittaa lisää tietoja: automerkkejä, lintu- ja koiralajeja, kaupunkien nimiä. Mutta näihin muuttujanimiin itseensä emme voi enää varastoida tietoa, koska olemme jo käyttäneet nimet A$, B$, C$ ja D$.

Ratkaisu on käyttää uudenlaista muuttujaa, jota kutsutaan nimellä *indeksoitu muuttuja.* Vaikealta kuulostavasta nimestä huolimatta asia ei ole kovin vaikea.

Indeksi on numero, jonka avulla tunnistetaan yksi ryhmän jäsen. Matematiikassa indeksoidut muuttujat ovat tällaisia:

$$A_6 B_{25} C_{1000}$$

Nämä esimerkit tarkoittavat: ”kuudes A:ksi nimetty asia”, ”kahdeskymmenesviides B:ksi nimetty asia” ja ”tuhannes C:ksi nimetty asia”. (Näiden ryhmän merkin alapuolella olevien *alaindeksien* lisäksi on olemassa myös *yläindeksit,* jotka tavallisesti ilmoittavat potenssiin korottamista, esim. 3^2, joka tarkoittaa kolme korotettuna toiseen potenssiin, kolme kertaa kolme, kolme kerrottuna itsellään.) Tietokone pystyy järjestämään tietoa näiden indeksien mukaan, mutta se ei pysty tunnistamaan muuttujanimen alapuolelle kirjoitettuja alaindeksejä, joten ne on kirjoitettava tällä tavalla:

```
A(6) B(25) C(1000)
```

Sulkumerkit eli sulkeet ilmaisevat, mitkä muuttujat ovat indeksoituja eli voidaan tunnistaa numeron perusteella. Tavallisesti alaindeksejä ei osoiteta puheessa, A(6) on yleensä vain ”A kuusi”.

```
 1 Triumph Herald
 2 Morris Minor
 3 Sunbeam Alpine
 4 Citroen 2 CV
 5 Honda Civic
 6 Buick Century
 7 Nash Rambler
 8 Nissan Micra
 9 Jeep
10 Chevette
```

Edellä on lueteltu joukko autoja numeroituina 1:stä 10:een. Voimme antaa jokaiselle oman muuttujanimen A$, jos käytämme indeksejä. Listasta näkee suoraan alaindeksin numeron. Voimme siis sanoa, että A$(7) on Nash Rambler ja A$ (2) on Morris Minor.

Indeksoidut muuttuvat voivat olla merkkimuuttujia, kuten esimerkissä, tai lukuja, ja ne noudattavat aikaisemmin kerrot-

tuja muuttujasääntöjä.

Kirjoita tämä esimerkki selvittääksesi, mitä indeksoidut muuttujat todella ovat:

```
10 A$ = "AUTOT"
20 A$(1) = "TRIUMPH HERALD"
30 A$(2) = "MORRIS MINOR"
40 A$(3) = "SUNBEAM ALPINE"
50 A$(4) = "CITROEN 2 CV"
60 A$(5) = "HONDA CIVIC"
70 A$(6) = "BUICK CENTURY"
80 A$(7) = "NASH RAMBLER"
90 A$(8) = "NISSAN MICRA"
100 A$(9) = "JEEP"
110 A$(10) = "CHEVETTE"
120 PRINT " [CLR] "
130 PRINT A$
140 FOR I = 1 TO 10
150 PRINT A$(I)
160 NEXT I
```

Ohjelma sijoittaa muuttujaan A$ nimen "AUTOT", ja sitten muuttujiin A$(1) — A$(10) yhden jokaisesta listan numeroidusta muuttujasta A$( ). Pane merkille, että *A$ ei ole sama kuin A$(1).* Ilman indeksiä oleva A$ on yksi muuttuja, ja se on sukua A$(1):lle tai mille tahansa A$( )-muuttujalle vain nimen osalta.

Sulkeissa oleva indeksinumero voi olla *mikä tahansa positiivinen luku* väliltä 0—32767. Sulkeissa voi olla myös *numeerinen muuttuja.*

Rivi 140 asettaa yksinkertaisen silmukan, FOR I = 1 TO 10 ja kirjoittaa sitten ruutuun jokaisen automerkin sen alaindeksin mukaan.

Voit todeta, että numeroita voidaan varastoida samalla lailla kuin merkkimuuttujia, kirjoittamalla nämä kolme riviä:

```
10 FOR I=0 TO 10
20 A(I)=I+25
30 NEXT I
```

Silmukka FOR I=0 TO sijoittaa indeksoituun muuttujaan A( ) järjestyksessä luvut 25:stä 35:een. Voit kokeilla tätä kysymällä lukuja suoraan näppäimistöltä.

```
PRINT (tai ?) A(5)
```

Vastauksen tulisi olla 30. Siis mikä tahansa sulkeissa oleva luku lisättynä 25:llä.

DIM-käsky antaa uusia ulottuvuuksia

Jos haluat käyttää indeksoituja muuttujia, sinun on ensiksi määriteltävä kuinka monta muuttujaa haluat käyttää.

Mutta malta hetki. Vastahan me onnistuimme käyttämään indeksoituja muuttujia kertomatta tietokoneelle, kuinka monta muuttujaa halusimme käyttää.

Lisää tämä rivi aikaisempaan automerkkiohjelmaan:

```
115 AS(11)="TOYOTA TERCEL"
```

Ja muuta rivi 140:

```
140 FOR I=1 TO 11
```

Suorita ohjelma. Mitä tapahtuu? Näet seuraavan virheilmoituksen:

```
?BAD SUBSCRIPT ERROR IN LINE 115
```

Tietokone muistuttaa tässä meitä juuri siitä, että on aina ilmoitettava, kuinka monta muuttujaa haluamme käyttää. Miksi vasta nyt? Siksi, että Commodoren BASIC sallii meidän käyttää 11:tä indeksoitua muuttujaa numeroiltaan 0—10, ennen

kuin koneelle on kerrottava, kuinka monta muuttujaa halutaan käyttää.

****

BAD SUBSCRIPT ERROR

Tämä virheilmoitus on suomeksi: sopimaton indeksi. Ohjelmassa tuli esiin indeksi, joka on liian suuri DIM-käskyllä määriteltyyn taulukkokokoon nähden. Jos DIM-käsky on esimerkiksi DIM A$(100), ei voi olla muuttujaa nimeltä A$(101). Tämä virheilmoitus näkyy myös silloin, jos DIM-käskyä ei ole lainkaan annettu ja käytetään muuttujia, joiden indeksi on suurempi kuin 10. Kun näet tämän virheilmoituksen, tarkista indeksoidut muuttujat.

****

Indeksoitujen muuttujien käyttämisestä kerrotaan tietokoneelle käskyllä DIM. Se määrittelee, kuinka laajoja muuttujia tarvitaan. Kun käytämme käskyä DIM, se asettaa *taulukon* eli määrittelee tarvittavien muuttujien määrän.

```
10 DIM A$(100)
```

Tällä käskyllä pyydämme BASICiä varaamaan 101 numeroitua paikkaa indeksoidulle muuttujalle A$( ). (Muista, että indeksinä voi käyttää nollaakin.) Voimme käyttää muuttujia vähemmänkin, mutta emme lainkaan sellaisia, joiden indeksi olisi yli 100. Tätä kutsutaan *yksiulotteisen taulukon* asettamiseksi. Se ulottuu vain yhteen suuntaan, koska se on yksittäisten muuttujien lista:

```
A$(1), A$(2), A$(3) ..... A$(100)
```

Taulukko voidaan asettaa myös moniulotteiseksi.

```
10 DIM A$(100,5)
```

Jokaisella muuttujalla A$( ) on nyt kaksi toisistaan riippumatonta tunnuslukua. Voi myös olla 101 erillistä muuttujaa

A$( ), joille suurin indeksi on 100, mutta jokaista tällaista muuttujaa kohden on vielä kuusi muuttujaa *lisää*. Muuttujien indeksointi etenee nyt tällä tavalla:
A$(1,0), A$(1,1), A$(1,2), A$(1,3), A$(1,4) ja A$(1,5). Taulukosta on tullut kaksiulotteinen ja sillä on siis tavallaan sekä korkeus että leveys. Lista voisi näyttää tällaiselta:

A$(1,5), A$(2,5), A$(3,5) ..... A$(100,5)
A$(1,4), A$(2,4), A$(3,4) ..... A$(100,4)
A$(1,3), A$(2,3), A$(3,3) ..... A$(100,3)
A$(1,2), A$(2,2), A$(3,2) ..... A$(100,2)
A$(1,1), A$(2,1), A$(3,1) ..... A$(100,1)
A$(1,0), A$(2,0), A$(3,0) ..... A$(100,0)

Moniulotteiset taulukot ovat tuntuneet monista aloittelevista ohjelmoijista täysin käsittämättömiltä. Tällaista taulukkoa käytetään tavallisesti tiedon hankkimiseen jostakin lukuja sisältävästä taulukosta käyttämällä kahta tai useampaa muuttujaa. Jos siis X:n arvoksi saadaan 3 ja Y:n arvoksi 2, niin A$(X,Y) on edellisessä esimerkissä A$(3,2).

Pannaanpa kaksiulotteinen taulukko töihin. Tehdään taulukko, jossa on kuukaudet tammikuusta huhtikuuhun, numeroiltaan 1—4, ja kunkin kuukauden viikot, myös numeroiltaan 1—4. Jokaisen kuukauden jokaisen viikon kohdalla lasketaan, kuinka monta kertaa olemme käyttäneet omaa tietokonettamme.

Viikko/Tammi (1)	Helmi (2)	Maalis (3)	Huhti (4)
1/25	15	30	17
2/19	6	11	22
3/13	17	5	3
4/15	9	8	21

Viikkojen numerot kirjoitetaan pystysuoraan ja kuukaudet vaakasuoraan. Nämä luvut voidaan varastoida kaksiulotteiseen taulukkoon.

```
10 DIM A(4,4)
20 A(1,1)=25
30 A(1,2)=15
40 A(1,3)=30
50 A(1,4)=17
60 A(2,1)=19
70 A(2,2)=6
80 A(2,3)=11
90 A(2,4)=22
100 A(3,1)=13
110 A(3,2)=17
120 A(3,3)=5
130 A(3,4)=3
140 A(4,1)=15
150 A(4,2)=9
160 A(4,3)=8
170 A(4,4)=21
```

Suorita tämä esimerkkiohjelma. Kysy nyt lukuja taulukosta, jonka olemme kirjoittaneet koneen muistiin.

PRINT (tai ?) A(Viikon numero, Kuukauden numero)

Jos kysyt helmikuun ensimmäisen viikon lukua, niin kirjoitat PRINT A(1,2). Saamasi vastauksen tulisi olla sama kuin edellä olevassa taulukossa.

Mahdollisesti et tule koskaan käyttämään omissa ohjelmissasi kuin yksiulotteisia taulukoita, mutta moniulotteisten taulukoiden perusidea on kuitenkin esitetty edellä pohdittavaksesi.

Samalla rivillä voidaan antaa DIM-käsky usealle muuttujalle tällä tavalla:

```
10 DIM A$(100),B$(100),C$(100),D$(100)
```

Kukin näistä muuttujista, A$( ), B$( ), C$( ) ja D$( ), voi nyt varastoida tietoa, joka on indeksoitu numeroin 0—100. Kun varaat tilaa näille taulukoille, pidä huoli siitä, että tiedät, kuinka monta paikkaa käsket BASICin varata. Et voi enää

muuttaa mieltäsi ja käyttää käskyä DIM uudelleen ohjelmassa saman taulukon tilanvarauksen muuttamiseen.

****

REDIM'D ARRAY ERROR

Käskyä DIM voidaan käyttää vain kerran saman taulukkomuuttujan määrittelyyn. Kun taulukolle on varattu tila, sitä ei voi enää muuttaa uudella DIM-käskyllä. Tämä virheilmoitus esiintyy pääasiassa silloin, kun editoit ohjelmia tai yrität liittää yhteen ohjelmia, joissa kummassakin on DIM-käskyjä.

****

Voit kuitenkin käyttää muuttujaa taulukon tilanvaraukseen.

```
10 INPUT''KUINKA MONTA AUTOMERKKIA'';A
20 DIM A$(A)
```

Tässä esimerkissä ohjelma kysyy käyttäjältä, kuinka monta automerkkiä tullaan käsittelemään, ja varastoi sen tiedon muuttujaan A. Kun se varaa DIM-käskyllä tilaa taulukolle A$( ) rivillä 20, se käyttää tätä arvoa määrätäkseen taulukon koon.

READ-käskyllä luetaan DATAa

Aikaisemmassa ohjelmointiesimerkissä esitettiin tapa varastoida automerkkejä indeksoituihin merkkimuuttujiin. Tapa jolla ne varastoitiin — A$(1) = ''TRIUMPH HERALD'', A$(9) = ''JEEP'' jne. — on yksinkertaisin tapa määritellä näitä muuttujia. Se ei kuitenkaan ole kaikkein käyttökelpoisin keino. Jos muuttaisimme listalla olevat nimet, meidän täytyisi muuttaa monen monta ohjelmariviä. Voit kuvitella, kuinka epäkäytännöllistä tämä olisi siinä tapauksessa, että meillä olisi paljon varastoitavaa tietoa.

BASICin suunnittelijat ovat keksineet ratkaisun tähänkin ongelmaan. Entäpä jos voisimme ryhmittää kaiken ohjelmassa tarvittavan tiedon tiettyihin paikkoihin. Kaikki muuttujat voitaisiin siinä tapauksessa etsiä näistä paikoista.

Tätä tarkoitusta varten BASICissä on käskyt DATA ja READ. Käskyllä DATA alkavat rivit sisältävät tietoa. Käsky READ ottaa nämä tiedot yhden kerrallaan listasta ja varastoi sen mihin tahansa käskettyyn muuttujaan, onpa se numeerinen tai merkkimuuttuja, indeksoitu muuttuja tai ei.

Kutakin yksittäistä tietoa, lukua tai merkkijonoa, joka seuraa DATA-sanaa, erottaa muista listan osista pilkku. (Tämä merkitsee tietysti sitä, että merkkijonoihin ei voi sisältyä pilkkuja.) Kullakin BASIC-tietorivillä voi olla kaksi kuvaruuturiviä eli 80 merkkiä, mukaan luettuina rivinumero ja käsky DATA. DATA-rivit näyttävät tällaisilta:

```
10 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13
```

tai

```
10 DATA YLOS, ALAS, VASEN, OIKEA
```

Sanaa READ käytetään aina joko numeerisen tai merkkimuuttujan kanssa sen mukaan, minkälaista tietoa DATA-lauseisiin sisältyy. Muuttujanimi voi olla lopullinen tai tilapäinen, joka sitten puolestaan siirtää tiedon lopulliseen muuttujaan.

```
10 FOR I=1 TO 5
20 READ A$(I)
30 NEXT I
```

tai

```
10 FOR I=1 TO 5
20 READ N$
30 A$(I)=N$
40 NEXT I
```

Tämän avulla voimme kirjoittaa uudelleen esimerkkiohjelmamme automerkeistä.

```
10 DIM A$(11)
20 FOR I=1 TO 11
30 READ N$
40 A$(I)=N$
50 NEXT I
60 FOR I=1 TO 11
70 PRINT A$(I)
80 NEXT I
100 DATA TRIUMPH HERALD,MORRIS MINOR,SUN-
    BEAM ALPINE
110 DATA CITROEN 2CV,HONDA CIVIC,BUICK CEN-
    TURY
120 DATA NASH RAMBLER,NISSAN MICRA,JEEP
130 DATA CHEVETTE,TOYOTA TERCEL
```

Vaikka DATA-listassa voi olla tietoa enemmänkin kuin on tarpeen, siinä ei voi olla tietoa *vähemmän*. Edellisessä esimerkissä voisi olla useampiakin automerkkejä, mutta ohjelma käyttäisi vain ensimmäiset 11. Mutta jos DATA-listassa olisi ainoastaan 10 nimeä, ohjelman suoritus keskeytyisi ja näkisit virheilmoituksen.

****

OUT OF DATA ERROR

Laske, kuinka monta tietoa DATA-lauseissa on, ja tarkista sitten, kuinka monta kertaa ohjelma käy läpi käskyn READ. (Eräs keino on etsiä FOR/NEXT-silmukoita, joissa on käsky READ.) Tarkista DATA-lauseet esimerkiksi lisäämällä READ-käskyjen jälkeen PRINT-käsky, jolla ohjelma kirjoittaa luetut tiedot. Tarkista sitten olivatko tiedot halutunlaisia.

****

DATA-listaan voi kirjoittaa sekä numeroita että merkkijonoja, kunhan muistaa tarkasti, millaisia tietoja ohjelmassa kysytään. (Merkkijonoa ei voi antaa vastaukseksi silloin, jos muuttujaan odotetaan lukua.) Seuraavassa on toinen versio automerkkioh-

jelmasta. Nyt teemme taulukon sekä automerkeistä että niiden hinnoista jossakin käytettyjen autojen myymälässä. Siihen vaaditaan merkkijonojen ja numerotietojen keskinäistä vaihtelua.

```
10 DIM A$(11),A(11)
20 FOR I=1 TO 11
30 READ A$(I),A(I)
40 NEXT I
50 PRINT "[CLR] AUTO    HINTA"
60 FOR I=1 TO 11
70 PRINT A$(I),"MK";A(I)
80 NEXT I
100 DATA TRIUMPH HERALD,5000,MORRIS MINOR,
    6200
110 DATA SUNBEAM ALPINE,9300,CITROEN 2 CV,3700
120 DATA HONDA CIVIC,8200,BUICK CENTURY,18900
130 DATA NASH RAMBLER,11000,NISSAN MICRA,6600
140 DATA JEEP,4200,CHEVETTE,6400
150 TOYOTA TERCEL,13000
```

Rivillä 10 asetetaan kaksi taulukkoa, A$(11), johon mahtuu 11 automerkkiä merkkimuuttujina, ja A(11), numeerinen muuttuja, johon mahtuu 11 lukua eli kunkin auton hinta. Rivit 20—40 lukevat automerkit ja luvut DATA-listasta. Rivi 50 tyhjentää kuvaruudun ja kirjoittaa sitten sanat AUTO ja HINTA. Rivit 60—80 kirjoittavat ruutuun automerkin — A$( ) — sitten MK ja sen viereen kunkin auton hinnan.

Sinun tulisi myös tietää, että tietokone huolehtii aina siitä, että se on oikealla kohdalla DATA-listassa ja tunnistaa tiedot, vaikka DATA-listan rivit olisivatkin hajallaan eri osissa ohjelmaa. Kaksi muuta BASIC-käskyä, RESTORE ja CLR, vaikuttavat READ-käskyn toimintaan ja muuttujina varastoituihin tietoihin.

Käskyllä RESTORE ei ole mitään tekemistä Commodoren näppäimistössä olevan samannimisen näppäimen kanssa. Se palauttaa DATA-toiminnon takaisin alkuun siten, että käskyllä

READ aletaan lukea tietoja jälleen DATA-listan alusta.

Käskyä CLR käytetään poistamaan kaikki muuttujiin varastoitu tieto, onpa se numeerista tai merkkijonotietoa. On kuitenkin käytettävä käskyä CLR varoen, koska se ei valikoi. Se tyhjentää kaikki muuttujat, nekin jotka saattaisivat olla tarpeen, jotta ohjelma edelleen toimisi hyvin. Jos haluat tyhjentää vain tietyt muuttujat, voit aina tehdä jotensakin näin:

```
10 A$=" "
```
tai
```
10 A=0
```
tai
```
10 FOR I=1 TO 10
20 A$(I)=" "
30 NEXT I
```

Älä missään nimessä sekoita käskyä CLR tämänkin kirjan ohjelmissa esiintyvään näppäimeen CLR/HOME. Jos sanan [CLR] ympärillä on hakasulut, se merkitsee: paina *näppäintä* CLR/HOME *eikä* kirjoita BASICin käsky CLR.

Taulukoiden käyttö levyke- ja nauhatiedostoina

Ennen kuin ryhdyt opiskelemaan tätä asiaa, kertaa, mitä puhuimme luvussa 2 Commodoren levy- ja nauha-asemista.

Kun tieto on kirjoitettu tietokoneeseen, on oltava jokin keino varastoida se levykkeelle tai kasettinauhalle. Muutoin se häviää samalla hetkellä kun virta katkaistaan. Eräs tapa varastoida tietoa nauhalle tai levykkeelle on tallentaa se jonkin BASICillä kirjoitetun ohjelman yhteyteen. Olet jo DATA-lauseista puhuttaessa nähnyt, kuinka se tehdään. Mutta samalla tavalla kuin DATA-lauseet helpottivat muuttujiin sijoitettavan tiedon kirjoittamista, on olemassa menetelmä helpottaa tiedon käsittelyä.

Kuten aikaisempiakin Commodoren tietokoneita myös Commodore 64 -tietokonetta voidaan kutsua *tiedostoihin suuntautuneiksi* tietokoneiksi. Se tarkoittaa, että tietokone lähettää kaiken ulkopuolisen tiedon näihin tiedostoihin ja hakee sen myöhemmin niistä takaisin. Levy- tai nauhatiedostoja voidaan avata OPEN-käskyllä, niitä voidaan sulkea käskyllä CLOSE, niihin voidaan kirjoittaa ja niistä voidaan lukea tietoja aivan kuten paperikansioistakin. BASIC pitää myös 64-laitteistoon liittyvää kirjoitinta näiden tiedostojen käsittelylaitteena, ja monet tiedostojen käsittelykäskyt soveltuvat siihenkin. Uudet tiedostokäskyt ovat oikeastaan jo tuttujen sanojen muunnelmia: PRINT#, INPUT# ja GET#.

Commodore 64 kykenee muodostamaan nauhalle tai levykkeelle monenlaisia tiedostoja. Yleensä tietoa varastoidaan *peräkkäistiedostoihin*. Niitä kutsutaan peräkkäistiedostoiksi, koska tieto varastoidaan niihin jonona, yksi tieto aina toisen perään. Tällaiset levy- tai nauhatiedostot ovat erilaisia kuin ohjelmat siinä mielessä, että niitä ei voida sellaisinaan ladata koneen muistiin.

Tiedostojen käsittelyn aloittava BASIC-käsky on OPEN. Tätä sanaa voidaan käyttää monella tavalla, mutta tässä osassa puhumme vain siitä, kuinka luodaan peräkkäistiedostoja.

Käskyn OPEN yhteydessä käytetään *tiedoston numeroa, laitenumeroa* ja *toisen osoitteen numeroa.*

```
10 OPEN 1,8,3
```

Tässä esimerkissä 1 on tiedoston numero, 8 laitenumero ja 3 toisen osoitteen numero.

Tiedoston numero on aina *sinun* antamasi numero, ja samalla kertaa voi enintään viisi peräkkäistiedostoa olla auki. Laitenumero on numero, joka annetaan erikseen nauha-asemalle, levykeasemalle ja kirjoittimelle. (Tietokone ei ymmärrä sellaisia sanoja kuin nauhuri, levyke ja kirjoitin, joten se ”ajattelee” niitä numeroina.) Nämä laitenumerot on asetettu valmiiksi. *Kasettinauhuri* on aina laite numero 1. *Kirjoitin* on lähes aina laite

numero 4. Jos käytät kahta kirjoitinta (esimerkiksi matriisikirjoitinta ja laatukirjoitinta), on *toinen kirjoitin* yleensä laite numero 5. *Levykeasemat* toimitetaan Commodoren tehtaalta asetettuna laitteeksi numero 8. Tätä numeroa voidaan kuitenkin muuttaa, ja toinen levykeasema onkin tavallisesti laite numero 9.

Kun tehdään nauhatiedostoja, on kolme tärkeää toista osoitetta. Jos et käytä toista osoitetta, tietokone olettaa numeroksi nolla, joka tarkoittaa sitä, että tiedosto on auki tiedon *hakemiseksi* nauhalta. Toinen osoite 1 merkitsee, että tiedosto on auki *tiedon lähettämistä* varten. Toinen osoite 2, jos sitä käytetään nauhan yhteydessä, kirjoittaa erikoismerkin, joka osoittaa *nauhan loppumista*. Tämän ansiosta sinun ei tarvitse kelata koko nauhaa loppuun hakeaksesi tietoa, jota nauhalla ei edes ole. On mahdollista tehdä ''nauhan loppumistiedostoja'', joissa ei ole mitään muuta tietoa kuin tämä merkki.

****

NOT INPUT FILE ERROR
NOT OUTPUT FILE ERROR

Kasettitiedostot avataan käskyllä OPEN joko INPUT- tai OUTPUT-tiedostoina, siis joko tietojen lukemiseksi tiedostosta tai niiden kirjoittamiseksi tiedostoon. Osoite 1 tarkoittaa, että nauhatiedosto on auki siihen kirjoittamista varten. Jos käytät käskyjä INPUT# tai GET#, näet ensimmäisen näistä virheilmoituksista. Jos kasettinauhurin yhteydessä ei ole käytetty mitään toista osoitetta, tietokone olettaa, että tiedostosta haetaan tietoa. Käskyn PRINT# käyttäminen antaa tulokseksi toisen näistä virheilmoituksista.

****

Levyketiedostoja tehdessäsi voit itse valita toisen osoitteen numerot. Jokaisella numeroidulla levyketiedostolla tulisi olla oma toinen osoite. Näihin levyketiedostoihin voidaan kirjoittaa tietoja tai niistä voidaan lukea tietoja ilman nauhatiedostojen vaatimia toisia osoitteita.

Commodoren levykeasemaa 1541 käytettäessä on suositeltavaa käyttää toisina osoitteina vain numeroita 2 ja 3 ja olla avaamatta OPEN-käskyllä useampaa kuin kahta tiedostoa kerrallaan. Tämä ei ole mikään varsinainen rajoitus, koska pystyt varmasti hoitamaan lähes kaiken tiedonkäsittelyn yhdellä ainoalla avatulla levyketiedostolla.

Toista osoitetta numero 15 käytetään erikoistarkoituksiin, lähettämään käskyjä Commodoren levykeasemalle ja etsimään syytä levykeaseman punaisen valon vilkkumiseen.

Tiedoston numeron, laitenumeron ja toisen osoitteen jälkeen voimme *nimetä* avatun tiedoston pystyäksemme jälleen tunnistamaan sen kutsuessamme sitä nauhalta tai levykkeeltä.

****

FILE NOT OPEN ERROR
FILE OPEN ERROR

Kun käytämme tiedonkäsittelysanoja PRINT#, INPUT# ja GET# on ensiksi avattava tiedosto käskyllä OPEN. Muuten näkyy virheilmoitus FILE NOT OPEN ERROR. Se esiintyy myös silloin, kun käskyä CMD, jonka tarkoitus on siirtää kuvaruutuun kirjoitettavaksi tarkoitettua tietoa kirjoittimelle tai levykkeelle, käytetään avaamatta oikeaa tiedostoa tai käskyn CLOSE kanssa. FILE OPEN ERROR esiintyy silloin, kun yrität avata tiedoston, joka on aikaisemmin avattu käskyllä OPEN mutta jota ei ole suljettu käskyllä CLOSE. Tarkista aina tiedoston käsittelykäskyt varmistaaksesi, että tiedosto suljetaan käytön jälkeen käskyllä CLOSE.

****

Seuraava tiivistelmä auttaa sinua ymmärtämään kuinka BASIC:in käsky OPEN toimii. Muista, että nämä ovat ainoastaan esimerkkejä ja että valittavissasi ovat tiedoston numerot ja toisinaan myös toiset osoitteet.

OPEN 2,1,1,"KOE"

Avaa tiedoston numero 2 KOE-nimisen peräkkäistiedoston

kirjoittamiseksi kasettinauhalle (laitenumero 1). Tiedoston numero on käyttäjän valitsema.

OPEN 3,1,0,"KOE"

Avaa tiedoston numero 3 KOE-nimisen peräkkäistiedoston lukemiseksi kasettinauhalta (laitenumero 1). Tiedoston numero on käyttäjän valitsema.

OPEN 1,1

Avaa tiedoston numero 1 minkä tahansa peräkkäistiedoston lukemiseksi kasettinauhalta (laitenumero 1). Tiedoston numero on käyttäjän valitsema.

OPEN 2,1,2,"NAUHAN LOPPU"

Avaa tiedoston numero 2 NAUHAN LOPPU -nimisen tiedoston kirjoittamiseksi kasettinauhalle (laitenumero 1) siten, että siihen sisältyy erityinen nauhan loppumista osoittava merkki. Tiedoston numero on käyttäjän valitsema.

OPEN 1,8,2,"KOE,SEQ,WRITE"

Avaa tiedoston numero 1 KOE-nimisen peräkkäistiedoston kirjoittamiseksi levykkeelle (laitenumero 8). Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 3,8,3,"KOE"

Avaa tiedoston numero 3 KOE-nimisen peräkkäistiedoston lukemiseksi levykkeeltä (laitenumero 8). Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 2,8,2,A$+"SEQ,WRITE"

Avaa tiedoston numero 2 merkkimuuttujalla A$ nimetyn peräkkäistiedoston kirjoittamiseksi levykkeelle (laitenumero 8). Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 1,9,2,"KOE,SEQ,WRITE"

Avaa tiedoston numero 1 KOE-nimisen peräkkäistiedoston kirjoittamiseksi toisessa levykeasemassa olevalle levykkeelle (laitenumero 9). Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 4,8,1,"0:"+A$+"SEQ,WRITE"

Avaa tiedoston numero 4 merkkimuuttujalla A$ nimetyn peräkkäistiedoston kirjoittamiseksi kaksoislevykeaseman 0:lla numeroidussa asemassa olevalle levykkeelle (laitenumero 8). Com-

modore 64 -laitteissa tähän tarvitaan IEEE-488 liitäntä. Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 1,8,1,"KOE"

Avaa tiedoston numero 1 KOE-nimisen peräkkäistiedoston lukemiseksi levykkeeltä (laitenumero 8). Toinen osoite on käyttäjän valitsema.

OPEN 1,8,15

Avaa tiedoston numero 1 levykeasemassa (laitenumero 8) lähettääkseen viestin tai lukeakseen virheilmoituksen Commodoren levykkeenkäyttöjärjestelmästä. Tiedoston numero on käyttäjän valitsema, mutta toinen osoite on tarkoitettu nimenomaan tähän tarkoitukseen.

OPEN 1,4,3

Avaa tiedoston numero 1 kirjoittaakseen tekstiä kirjoittimella (laitenumero 4). Tiedoston numero ja toinen osoite ovat käyttäjän valitsemia.

OPEN 4,4,7

Avaa tiedoston numero 4 kirjoittimelle (laitenumero 4) lähettääkseen erityiskäskyn (toinen osoite) Commodoren kirjoittimelle tai älykkääseen kirjoitinliitäntään. Tiedoston numero on käyttäjän valitsema.

Se BASICin käsky, joka lähettää tietoa kasettinauhurille, levykeasemaan tai kirjoittimelle, on PRINT#. Käskyt PRINT ja PRINT# *eivät siis ole samat*, vaikka ne toimivat hyvin samanlaisesti ja noudattavat samoja käyttösääntöjä. PRINT kirjoittaa tietoa kuvaruutuun. PRINT# lähettää tietoa — merkkijonoja tai numeroita — sillä numerolla tunnettuun tiedostoon, joka sitä seuraa.

Jos olet jo tottunut lyhentämään käskyn PRINT muotoon ?, muista, että sitä voidaan käyttää *vain* käskyn PRINT lyhenteenä. *Et voi* käyttää muotoa ?# lyhentämään käskyä PRINT#. Jos vahingossa kirjoitat ?#, tietokone antaa virheilmoituksen SYNTAX ERROR. Kun listaat virheilmoituksen synnyttäneen rivin, näyttää käsky PRINT# kuvaruudussa aivan oikealta. Tietokoneen mielestä olet kuitenkin tehnyt virheen. Toisinaan tällä tavalla syntyneet SYNTAX ERROR -virheilmoitukset ovat erit-

täin vaikeita huomata ja korjata.

Aivan samalla lailla kuin INPUT on käskyn PRINT vastakohta, myös INPUT# on käskyn PRINT# vastakohta. INPUT# lukee kasettinauhurilta tai levykeasemasta peräkkäistiedostona tallennettua tietoa. Se lukee tallennettuja merkkijonoja tai numeroita, ja sitäkin käytetään tiedoston numeron kanssa.

Kuten käskyn INPUT kanssa ole varovainen, ettet käytä pilkkuja sellaisissa merkkimuuttujissa, joita luet käskyllä INPUT# nauha- tai levyketiedostoista. Tietokone ja levykeasema käyttävät kumpikin pilkkuja tiettyihin tarkoituksiin, ja pilkun jälkeen tuleva tieto jätetään huomiotta.

GET# on myös käskyn GET sukulainen. GET hyväksyy enintään yhden näppäimistöltä kirjoitetun merkin. GET# lukee yhden tiedon, yksittäisen nauhalle tai levykkeelle tallennetun numeron tai merkkijonon yhden merkin ja varastoi sen muuttujaan. Jos sinun täytyy käyttää pilkkuja peräkkäistiedostoissasi, käytä käskyä GET# . Mutta pane merkille, että sen käyttäminen hidastaa tiedostojen käsittelyä.

Peräkkäistiedostojen käsittely aloitetaan BASICin käskyllä OPEN eli avaa. On siis loogista, että kun olemme lopettaneet tiedoston käsittelyn, me suljemme sen käskyllä CLOSE. Käskyyn CLOSE liitetään sama tiedoston numero kuin vastaavaan OPEN-käskyyn.

Kasettinauhatiedostot

Peräkkäistiedostot tallentuvat kasettinauhalle samalla tavalla kuin ohjelmatiedostot, mutta kuten aikaisemmin on sanottu, niitä ei voi ladata sellaisinaan. Tiedot tallennetaan nauhalle ja tietokone lukee ne sitten *täsmälleen samassa* järjestyksessä. Näin siis BASIC-ohjelmarivit, jotka lukevat tietoja nauhalta, näyttävät miltei aina samanlaisilta kuin ne rivit, jotka tallentavat nauhalle tietoa.

Oletetaan, että meillä on tietty asiakokonaisuus, jota seuraa tuon asiakokonaisuuden nimen alle tallennettua tietoa. Voimme käyttää vaikka aikaisempaa esimerkkiä A$, joka merkitsi

automerkkejä, ja A$( ), joka taas sisälsi varsinaisen nimilistan. Voidaksemme tehdä tällaisen jonotiedoston meidän on ensiksi käskyllä OPEN avattava tiedosto ja annettava sille nimi. Sitten tallennamme luokan nimen A$, joka merkitsee ''AUTOT''. Seuraavaksi tallennetaan indeksoidut merkkimuuttujat, jotka sisältävät kaikki 11 automerkkiä.

Kokeillaanpa. Seuraava esimerkki tallentaa tiedon kasettinauhalle peräkkäistiedostoksi.

```
10 DIM A$(11):A$="AUTOT"
20 FOR I=1 TO 11
30 READ N$
40 A$(I)=N$
50 NEXT I
60 OPEN 2,1,1,"LISTA"
70 PRINT#2,A$
80 FOR I=1 TO 11
90 PRINT#2,A$(I)
100 NEXT I
110 CLOSE 2
200 DATA TRIUMPH HERALD,MORRIS MINOR
210 DATA SUNBEAM ALPINE,CITROEN 2 CV
220 DATA HONDA CIVIC,BUICK CENTURY,NASH
    RAMBLER
230 DATA NISSAN MICRA,JEEP,CHEVETTE
240 DATA TOYOTA TERCEL
```

Ensimmäiseksi tässä ohjelmassa asetetaan taulukko A$( ), johon varataan tilaa 12 tiedolle. Sitten määritellään muuttujan A$ sisällöksi ''AUTOT''. Rivit 20—60 lukevat DATA-riveillä 200—240 olevat nimet.

Sitten alkaa tiedon varastointi. Rivillä 60 avataan tiedosto numero 2 kasettinauhurilla (laitenumero 1). Numero 1 kertoo, että tiedosto avataan tiedon lähettämistä varten! Tiedoston nimi on LISTA. Tässä vaiheessa tietokone kirjoittaa kuvaruutuun ohjeen:

Kun aloitat tallentamisen, rivit 70—100 kirjoittavat käskyllä PRINT# tiedot A$(1):stä A$(11):een nauhalle. Ja lopuksi tiedosto 2 suljetaan CLOSE-käskyllä rivillä 110.

Olemme nyt luoneet nauhalle jonotiedoston. Mutta emme voi ladata tiedostoa emmekä voi mitenkään tarkistaa, onko kaikki tieto siellä. Ainoa tapa suorittaa tarkistus on kirjoittaa vielä toinen pieni ohjelma, joka lukee tiedot takaisin tietokoneeseen. Teemme samaan tapaan kuin yllä. (Muista kirjoittaa NEW, ennen kuin kirjoitat tämän ohjelman, ja muista myös *kelata nauha takaisin alkuun.*)

```
10 DIM J$(11)
20 OPEN 1,1
30 INPUT#1,J$
40 FOR I=1 TO 11
50 INPUT#1,J$(I)
60 NEXT I
70 CLOSE 1
80 PRINT ''[CLR]''
90 PRINT J$
100 FOR I=1 TO 11
110 PRINT J$(I)
120 NEXT I
```

Tämä ohjelma käy heti töihin. Se varaa taulukon 12:lle J$( ):lla merkitylle muuttujalle ja avaa sitten tiedoston numero 1 kasettinauhurilla. Toista osoitetta ei tarvita, koska tiedosto avataan tiedon lukemista varten. (Toinen osoite olisi itse asiassa 0). Rivi 50 ottaa ensimmäisen nauhalle tallentamamme tiedon, sanan AUTOT, ja nimeää sen J$:ksi. Seuraavaksi rivit 40—60 kirjoittavat asianmukaiset tiedot muuttujiin J$(1) — J$(11). Sitten tiedosto suljetaan. Ja viimeiseksi ohjelma tyhjentää kuvaruudun, kirjoittaa ruutuun muuttujan J$, jota seuraa nimilista, jotta tietäisimme kaiken olevan kunnossa.

Kuten huomaat, avatun tiedoston numero ei ollut sama kuin kirjoittaessamme tiedot nauhalle. Tiedostojen numeroiden ei tarvitse olla samat, koska niitä ei merkitä muistiin tiedostoon. Numeron on oltava sama kuin *käyttämämme ohjelman sisällä.* Viime esimerkissä tiedoston numero oli koko ajan 1. Huomaa myös, että käytämme eri muuttujanimiä. Ensimmäisessä ohjelmassa ne olivat A$ ja A$( ), toisessa taas J$ ja J$( ). Olisimme voineet antaa niille minkä nimen tahansa, kunhan vain toinen on merkkimuuttuja ja toinen indeksoitu merkkimuuttuja, jolle on varattu taulukosta tilaa 11 nimeä varten. Peräkkäistiedostoon ei siis myöskään kirjoiteta muuttujien nimiä.

Aikaisemmin mainittiin nauhan loppumista osoittava erityismerkki. Se saadaan aikaan helposti, eikä sitä varten tarvitse kirjoittaa edes ohjelmaa. Kirjoita vain tämä suoraan näppäimistöltä:

```
OPEN 1,1,2:CLOSE 1
```

Kelaa nauha takaisin alkuun, kirjoita LOAD ja paina RETURN-näppäintä. Kun tietokone löytää nauhoituksen, ilmestyy kuvaruutuun sana FOUND. Paina COMMODORE-näppäintä. Kuvaruutuun ilmaantuu lause DEVICE NOT PRESENT ERROR. Virheilmoituksesta (laitetta ei ole kytketty) huolimatta kasettinauhuri on kuitenkin käynnissä ja oikein kytketty. Tämä on Commodore 64:n sinänsä merkillinen tapa kertoa, että olet kohdannut nauhan loppumista osoittavan merkin.

Peräkkäistiedostoa tehtäessä kaikkein tärkeintä on pitää aina mielessä, missä järjestyksessä tieto on tallennettu, ja toistaa sitten tämä tieto kirjoitettaessa tietoa lukevia ohjelmarivejä. Muista myös sulkea tiedosto lopetettuasi. Äläkä kirjoita pilkkuja peräkkäistiedostoon kirjoitettuihin merkkimuuttujiin.

Kuten huomaat, nauhalle kirjoitetuissa peräkkäistiedostoissa ei ole mitään salaperäistä. Tulet pian näkemään, että levyketiedostot eivät ole kovin paljon monimutkaisempia.

Levyketiedostot

Jos olet käsittänyt nauhalle tehdyt peräkkäistiedostot, sinun pi-

täisi kyllä oppia ymmärtämään myös levykkeelle tehdyt tiedostot.

Jos olet katsonut levykkeen hakemistoa olet todennäköisesti huomannut kuvaruudun oikeassa reunassa kolmikirjaimisia lyhenteitä. Sellaisiin ohjelmiin, jotka voidaan ladata ja suorittaa, liittyy lyhenne PRG. Kun lisäät levykkeelle peräkkäistiedoston, sen tunnukseksi tulevat kirjaimet SEQ.

Kuten kasettinauhurin myös Commodoren levykeaseman on saatava tietää, milloin sen tulisi luoda peräkkäistiedosto. Levykeasemaa käytettäessä tiedosto on avattava määrätyllä tavalla. Käskyä OPEN seuraa tiedoston numero (jonka saat itse valita), laitenumero (joka levykeasemalle on miltei aina 8) ja toinen osoite (valitse joko 2 tai 3). Koko peräkkäistiedoston luomiskäsky levykeasemaa käytettäessä on tällainen:

```
10 OPEN 1,8,3,"KOE,SEQ,WRITE"
```

Tämä käskysarja panee tietokoneen ja levykkeen avaamaan laitteen numero 8 tiedoston numero 1. Tiedoston nimi on KOE ja se on peräkkäistiedosto (SEQ). Sana WRITE kertoo tietokoneelle, että sen on avattava tiedosto nimenomaan siihen kirjoittamista varten. Toisinaan saatat nähdä tällaisen käskyn kirjoitettuna näinkin:

```
10 OPEN 1,8,3,"0:KOE,SEQ,WRITE"
```

Lainausmerkeissä oleva nolla merkitsee sitä, että tiedosto tulisi kirjoittaa levykeasemaan 0. Tämähän oli jäänne entisestä Commodoren levykeyksiköstä, jossa, päinvastoin kuin 1541:ssä, oli yhden sijasta kaksi levykeasemaa. Tästä huolimatta tietokone piti *niitä kumpaakin* laitteena numero 8. Levykeasemien ero saatiin aikaan siten, että toinen niistä tunnettiin numerolla 0 ja toinen numerolla 1. Kaikki tällaiset levykekomennot, joissa levykeaseman numerona esiintyy nolla, toimivat 1541:ssä. Ne komennot, joissa käytetään levykeaseman numerona 1:tä, eivät sen sijaan toimi, ellei sinulla ole käytössäsi Commodoren kaksoislevykeasemaa. (Ja siihen tarvitaan IEEE—488-liitäntä.)

Jos sinulla on kaksi 1541-levyasemaa, niin ensimmäinen on laite numero 8 ja toiselle on sitten annettava eri laitenumero. (Tavallisesti siitä tehdään laite numero 9.) *Et* kuitenkaan kykene käsittelemään niitä levyasemina 0 ja 1.

On tosin muotoseikka, mutta sellainen joka estää ennalta arvaamattomia vaikeuksia, käyttää aina valinnaista etuliitettä ”0:”, kun avaa levytiedostoja.

Muuntaaksemme ”automerkkiohjelmamme” peräkkäistiedostoksi levykkeelle meidän on muutettava vain yksi rivi. Vaihda aikaisemman ohjelman rivi 60 seuraavanlaiseksi:

```
60 OPEN 2,8,2,”0:LISTA,SEQ,WRITE”
```

Jotta voisit muuntaa ohjelman, joka luki tietoa nauhatiedostosta, sinun on myös muutettava yksi rivi. Tämä rivi korvaa aikaisemman ohjelman rivin 20:

```
20 1,8,3,”0:”LISTA,SEQ”
```

Entäpä jos tiedoston nimi ei aina olekaan sama? Voit käyttää INPUT-käskyä, joka korvaisi nimen merkkimuuttujalla, ja käyttää sitten tuota muuttujaa. Tämän tehdäksesi sinun on liitettävä muuttuja tiedostokäskyyn.

```
10 INPUT ”ANNA TIEDOSTON NIMI”;N$
20 OPEN 2,8,2,N$+”,SEQ,WRITE”
```

Huomaat varmasti, että ohjelman levykeversio *vaatii* tiedoston nimen, vaikka kasettinauhuria käytettäessä nimen voikin jättää pois. Jos nimeä ei käytetä, kasettinauhurista luetaan ensimmäinen tiedosto. Koska levykkeeseen voi sisältyä monen monta tiedostoa, levykeasemalle on aina annettava tämä tiedoston nimi.

Sinun on oltava *erityisen huolellinen,* kun suljet levykkeellä olevia peräkkäistiedostoja. Saat selville, onko levyketiedosto auki, katsomalla levykkeen hakemistoa. Jos lyhenteen SEQ vieressä on tähti (*), tiedosto on vielä auki. Sellainen tiedosto on

hyödytön (koska siihen ei voida lisätä tietoja) ja tällaisen tiedoston tuhoaminen (katso lukua Perustaidot) voi vahingoittaa levykkeellä olevia tietoja. Mitään ongelmia ei kuitenkaan pitäisi syntyä, jos suljet tiedoston aina käskyllä CLOSE avattuasi sen tiedon varastoimista varten.

****

FILE NOT FOUND

Tämä virheilmoitus tulee esiin silloin, kun yrität ladata ohjelman tai tiedoston sellaiselta levykkeeltä, jossa se ei ole. Jos olet varma, että sinulla on sellainen ohjelma, tarkista levykkeen hakemisto nähdäksesi, onko se todellakin juuri tällä levykkeellä tai oletko kirjoittanut sen oikein. Tämä levykeasema vaatii ehdotonta tarkkuutta ohjelman nimeämisessä. Käytä tähteä tyyppihaussa välttyäksesi tältä virheeltä.

****

Vielä yksi varoitus sekä nauhalla että levykkeellä olevista peräkkäistiedostoista. Tiedät jo, että on sellaisia merkkimuuttujia, jotka eivät sisällä mitään tietoa, ”tyhjiä” merkkimuuttujia. Vaikka Commodoren BASICissä onkin mahdollista käyttää tällaisia tyhjiä merkkijonoja, voi yritys kirjoittaa niitä peräkkäistiedostoon sekoittaa tiedoston.

```
10 A$ = ”KYLLA”:B$ = ” ”:C$ = ”EI”
20 OPEN 1,8,2,”0:KOE,SEQ,WRITE”
30 PRINT# 1,A$,B$,C$
40 CLOSE 1
```

Kun tämä esimerkkiohjelma kirjoittaa tätä tiedostoa levykkeelle peräkkäistiedostoksi, se hyppää B$:n yli, koska se on tyhjä merkkijono. Kun sitten yrität lukea tiedostoa, tulee toisesta tietokohdasta, B$:sta, ”EI”. Eräs keino poistaa tämä pulma on kokeilla, onko merkkijono tyhjä vai ei. Jos se on, se voidaan korvata toisella merkkijonolla, joka on mahdollista kirjoittaa tiedostoon.

```
10 A$ = ''KYLLA'':B$ = '' '':C$ = ''EI''
20 OPEN 1,8,2,''0:KOE,SEQ,WRITE''
30 PRINT# 1,A$
40 IF B$ = '' '' THEN B$ = ''***''
50 PRINT# 1,B$
60 PRINT# 1,C$
70 CLOSE 1
```

Tässä esimerkissä rivi 40 kokeilee ensiksi, onko B$ tyhjä, ja jos se on, se korvaa tuon tyhjän merkkijonon kolmella tähdellä — ***. Nämä kolme tähteä on tässä valittu siksi, että ne eivät todennäköisesti sekoitu mihinkään tärkeään tietoon ja silti kirjoittuvat tiedostoon hyvin. Yhtä helposti ja loogisin perustein olisimme voineet valita jonkin toisen korvikkeen — FFFF tai ???.

Kun tiedosto luetaan, me kokeilemme, onko siinä tuota korviketta.

```
10 OPEN 1,8,2,''KOE,SEQ''
20 INPUT# 1,A$
30 INPUT# 1,B$
40 IF B$ = ''***'' THEN B$ = '' ''
50 INPUT# 1,C$
60 CLOSE 1
```

Rivi 40 kysyy, onko merkkijono ''***'', ja jos on, se merkitsee B$:n taas tyhjäksi.

Kun haet korvikkeita tyhjille merkkijonoille, muista valita niin erikoinen, ettei sitä voida sekoittaa oikeaan tietoon, joka siinä tapauksessa katoaisi.

Kirjoitintiedostot

Commodore 64 -laitteissa voidaan mikä tahansa tieto kirjoittaa paperille vain avaamalla tiedosto ja lähettämällä se kirjoittimeen. Näiden tiedostojen avaamisessa voivat toiset osoitteet olla merkityksellisiä, mutta nämä luvut merkitsevät toisinaan eri

asioita eri kirjoittimille. Nimenomaan Commodoren omat kirjoittimet käyttävät toisia osoitteita siirtyäkseen grafiikkatulostuksesta tekstintulostukseen ja asettaakseen muita toimintaehtoja. "Älykkäät" kirjoitinliitännät tulkitsevat myös toisia osoitteita toimintaehtoina. Jos käytät kirjoitinta tai älykästä liitäntää, lue huolellisesti sen ohjekirja ja opiskele näiden toisten osoitteiden käyttö. Tutki myös ohjekirjan sitä osaa, jossa puhutaan erityisistä "ohjausmerkeistä" (control characters) ja niihin tarvittavista numerosarjoista.

Useimpia kirjoittimia ohjaa mikroprosessori, joka mahdollistaa lukuisia erityistoimintoja. Mitkään kaksi kirjoitinta eivät kuitenkaan ole täysin samanlaisia, joten näiden ohjauskoodien luetteleminen olisi täysin mahdotonta.

Kirjoitintiedoston avaamiseen riittää tiedoston numero ja laitenumero. Kirjoittaminen paperille sujuu lähes samalla lailla kuin kuvaruutuunkin. Käytät tosin käskyä PRINT#, mutta välimerkit, kuten pilkut ja puolipisteet, käyttäytyvät paperilla aivan samalla tavalla.

```
10 OPEN 1,4
20 PRINT#1, "HEI! KIRJOITTIMENI TOIMII!"
30 CLOSE 1
```

Ongelman voivat aiheuttaa ylimääräiset tyhjät rivit, joita saattaa hivuttautua kirjoittamiesi rivien väliin. Tämä tavallista suurempi riviväli johtuu siitä, että *sekä* tietokone *että* kirjoitin lähettävät "rivinvaihtokäskyn". Helpoin tapa kytkeä tämä ylimääräinen rivinvaihto pois toiminnasta on estää se kirjoittimessa. (Katso ohjekirjaa.) Eräät älykkäät liitännät poistavat myös tämän ylimääräisen rivinsiirron ennen kuin se ehtii kirjoittimeen saakka.

Kirjoittimesi hallinta vaatii tarkkuutta ja sen toiminnan ymmärtämistä aina erityismerkkien toimintaa myöten. Eräiden laajojen toimintojen hyödyntäminen voi olla hyvin hankalaa. Mutta toistaiseksi riittää tämä helpoin tapa kirjoittaa paperille tietokoneen tiedostonkäsittelyjärjestelmän avulla.

Postituslistaohjelman tekeminen

Tietokantaohjelma on ohjelma, johon varastoidaan ja josta luetaan tietoa. Sellaisia ohjelmia on eritasoisia yksinkertaisista monimutkaisiin sen mukaan, mitä niiden avulla voidaan saada aikaan. Itse ohjelmat eivät itse asiassa ole tietokantaohjelmia. Tietokannat ovat oikeastaan näiden ohjelmien keräämiä tietoja. Pesäpallon lyöjätilasto on tietokanta, samoin pörssinoteeraukset sanomalehdissä. Keittokirjaakin voitaisiin pitää reseptien tietokantana. Tietokantana voidaan oikeastaan pitää mitä tahansa järjestynyttä tietoryhmää.

Tämän luvun lopussa käytämme yksinkertaista BASICillä kirjoitettua tietokantaohjelmaa. Voit käyttää sitä vaikkapa henkilökohtaisena postituslistana. Sen tarkoituksena on näyttää, kuinka yksinkertainen tietokanta luodaan. Monet myytävänä olevat tietokantaohjelmat ovat joustavampia ja monipuolisempia ja varastoivat tietoja eri tavalla. Tämä postituslistaohjelma rajoittuu siihen, että se varastoi tietoa tietokoneen muistiin ja tallentaa sen peräkkäistiedostoksi.

Jos olet aloitteleva ohjelmoija, kannattaa lukea koko ohjelma ennen sen kirjoittamista koneen muistiin. Silloin ymmärrät, kuinka se toimii, ja saat näppäimistöharjoitusta samalla kertaa.

Kun kirjoitat ohjelmaa tietokoneeseen, tee vain vähän kerrallaan ja kopioi jokainen rivi *täsmälleen* sellaisena kuin se on kirjaan merkitty. Kuten aikaisemmissakin BASIC-esimerkeissä paina sulkeissa olevia *näppäimiä,* älä kirjoita samanlaisia *käskyjä.*

Kun kirjoitat tätä tai mitä tahansa ohjelmaa, muista tallentaa se käskyllä SAVE nauhalle tai levykkeelle suhteellisen *usein.* Työsi ei katoa, jos tietokoneesta yhtäkkiä katkeaisi virta. (Voit kelata nauhan takaisin alkuun ja tallentaa aiemmin tallennetun päälle tai tallentaa levykkeelle aikaisemman version päälle käskyllä SAVE"@".) Jos et muista kuinka tämä tapahtuu, tarkista asia luvusta Perustaidot.

Teet varmasti virheitä ohjelmaa kirjoittaessasi. Jos näet virheilmoituksen, se on todennäköisimmin SYNTAX ERROR, jo-

ka johtuu kirjoitusvirheestä tai kaksoispisteen tai muun BASICin välimerkin pois jäämisestä. Jotkin kirjoitusvirheet saattavat tosin aiheuttaa muitakin virheilmoituksia. Yritä itse keksiä, mistä virhe johtuu, ennen kuin alat tutkailla kirjaa. Jos et keksi ongelmaa, katso tekstissä olevia virheilmoituksia ja sitten ohjelmalistauksessa olevaa riviä varmistuaksesi siitä, että olet kirjoittanut kaikki oikein.

Ohjelman suunnittelu

Mitä me haluamme postituslistaohjelman tekevän? Meillä pitäisi olla suunnitelma, ennen kuin ryhdymme ohjelmoimaan. Emmehän me rakentaisi taloakaan ilman piirustuksia. Suunnitelmamme hahmottuu, kun mietimme, mitä ominaisuuksia haluamme ohjelmalla olevan.

Postituslistaohjelman tulisi kyetä varastoimaan nimiä, osoitteita ja puhelinnumeroita. Niitä pitäisi pystyä lukemaan, muuttamaan tarvittaessa ja tallentamaan ne nauhalle tai levykkeelle. Postituslista pitäisi voida kirjoittaa myös paperille. Ja lopuksi meidän tulisi kyetä poistamaan yksittäinen tieto tai kaikki tiedot. Seuraavassa on yksityiskohtainen kuvaus ohjelman eri osista:

Tietojen kirjoittaminen

Tämä ohjelman osa ottaa tietoa vastaan ja varastoi sen tietokoneen muistiin oikeaan paikkaan ja oikeassa muodossa. Sen tulisi olla helppokäyttöinen ja pyytää nimeltä halutessaan asianmukaista tietoa. Tämä osa pitää myös kirjaa siitä, kuinka monta nimeä ja osoitetta tietokone pystyy kerrallaan käsittelemään ja kuinka paljon tilaa on vielä jäljellä.

Lukeminen

Tämän ohjelman tarkoitus on, että pystymme lukemaan tiedostossa olevat nimet ja osoitteet. Joskus haluamme lukea koko tiedoston, toisinaan tahdomme vain saada tietää jonkin nimen tai vain ne osoitteet, joiden suuntanumerossa tai postinumerossa on jotakin yhteistä.

Muuttaminen

Tietomme eivät aina pysy oikeina tai ajanmukaisina. (Ihmiset saattavat muuttaa tai vaihtaa puhelinnumeroaan.) Tämä ohjelman osa siis sallii meidän tehdä muutoksia.

Poistaminen

Koska tietokoneen muistitila rajoittaa mahdollisuuksiamme, haluamme kenties pitää tiedostossamme vain tärkeimmät nimet ja osoitteet. Aika ajoin haluamme siis ehkä poistaa jonkin tiedon. Tässä osassa voimme joko käydä läpi postituslistaa tai poistaa tietoja.

Paperille kirjoittaminen

Entä jos haluamme postituslistastamme kopion paperille? On suhteellisen helppoa siirtää tietoa tiedostosta paperille.

Lataaminen ja tallentaminen

Vaikka kutsummekin näitä osia näillä nimillä, emme käytä BASICin käskyjä LOAD ja SAVE, vaan luomme nauhalle tai levykkeelle peräkkäistiedostoja ja luemme sitten niitä.

Menu

Koko ohjelmaa käytetään menun eli valikon avulla. Tämä yksinkertaistaa ohjelman käyttöä ja sallii meidän käyttää näppäimistössä olevia toiminäppäimiä. Kuhunkin ohjelman pääosaan päästään ”päämenusta”.

Ohjelman laadinnassa huomattavia seikkoja

Postituslistan tekeminen on melko helppoa, koska ohjelma on suunniteltu tietyntyyppisen tiedon — nimien, osoitteiden ja puhelinnumeroiden — keräämiseen. Käytämme yksinkertaisia taulukoita nimiä, katuosoitteita, postitoimipaikkoja, postinumeroita ja puhelinnumeroita varten. Varaamme myös numeerisen muuttujan pysyäksemme selvillä siitä, kuinka monta tietoa meillä listassa on. Tiedot järjestetään tällä tavalla:

N$( ) — nimet
L$( ) — katuosoitteet
P$( ) — postitoimipaikat
Z$( ) — postinumerot
T$( ) — puhelinnumerot

Mainitunlaisten muuttujaluokkien asettaminen rajoittaa minkä tahansa tietokantaohjelman joustavuutta, mutta yleisempi ohjelma olisi paljon pitempi ja monimutkaisempi kuin tässä esitetty esimerkki.

Teemme tilaa 100 tiedolle asettamalla kunkin taulukon koon 100:aan. Numeerinen muuttuja N laskee tietojen kokonaismäärän. Tähän tarkoitukseen käytämme seuraavaa riviä:

```
15 X=101:DIM N$(X),L$(X),P$(X),Z$(X),T$(X):N=0
```

Muuttuja X merkitsee varastoitavien tietojen kokonaismäärää, joten DIM N$(X) on sama kuin DIM N$(101). Miksi pitäisi käyttää muuttujaa luvun 101 asemesta? Muuttamalla X:ää me voimme aina muuttaa mahdollisten tietojen määrää; muutamme vain yhden rivin (rivi 120) tarvitsematta etsiä, missä muualla sama luku esiintyy. (Vaikka varastoimmekin vain 100 kappaletta tietoja, X on 101, jotta voisimme helpommin poistaa tietoja.)

Ohjelmoinnin aloittaminen

Commodoren valitsemat kuvaruutuvärit tekevät ruudussa olevan tekstin vaikealukuiseksi. Me voimme valita helppolukuisemmat värit.

```
20 POKE 53280,12:REM RAJAUS HARMAA 2
30 POKE 53281,15:REM RUUTU HARMAA 3
```

Nämä luvut luetaan käskyllä POKE oikeisiin paikkoihin tietokoneen näyttöpiirissä, jotta värit muuttuisivat. Voit vaihtaa nä-

mä haluamiksesi väreiksi kirjoittamalla POKE-käskyllä muistipaikkoihin muita lukuja 0:sta 15:een. (Tarvittavan listan saat värigrafiikkaa käsittelevästä luvusta.)

Ohjelmassa olevat tiedot kirjoitetaan ruutuun mustalla. (Katso riviä 60.)

Rivit 40 ja 50 tekevät pari mukavaa temppua.

```
40 PRINT CHR$(14)
50 PRINT CHR$(8)
```

BASICissä CHR$( ) on erityinen tapa kirjoittaa merkkejä kuvaruutuun. Kullakin merkillä on oma numeronsa, joka on sulkeissa CHR$:n perässä. CHR$(8) ja CHR$(14) ovat kuitenkin sellaisia erityismerkkejä, jotka eivät näy kuvaruudussa.

PRINT CHR$(14) asettaa tietokoneen kirjoituskonetilaan tarvitsematta painaa yhtä aikaa SHIFT- ja COMMODORE-näppäimiä; se tehdään siis *ohjelmallisesti*. Entäpä jos joku sattuisi painamaan näitä näppäimiä ohjelman aikana? Sitä varten on olemassa tuo käsky PRINT CHR$(8). Se lukitsee näppäimet SHIFT ja COMMODORE siten, että et voi vaihtaa kirjoituskonetilasta grafiikkatilaan asettamatta ensin tietokonetta alkutilaan käyttämällä RESTORE-näppäintä.

Koska tämä on menuvalintainen ohjelma, sen kaikki päätoiminnot haarautuvat perussilmukasta. Tämä silmukka tekee vain kaksi asiaa: se kirjoittaa menun ruutuun ja odottaa, kunnes ilmoitat mitä haluat tehdä painamalla toiminäppäintä. Kaikki ohjelman tekemät asiat voidaan aina jäljittää tähän silmukkaan, joka alkaa riviltä 60 ja päättyy käskyyn GOTO 60 rivillä 220. Kun näppäintä painetaan, alkaa valittu aliohjelma toimia.

Päämenu näyttää tältä:
f1--Tietojen kirjoittaminen
f3--Tietojen lukeminen
f5--Kirjoittaminen paperille
f7--Tietojen poistaminen

f2--Lataaminen
f4--Tallentaminen
f6--Tietojen muuttaminen

Tässä on lista ohjelman tärkeimmistä aliohjelmista.

Rivino.	Aliohjelman tarkoitus
10000	Tietojen kirjoittaminen koneen muistiin
11000	Nimien ja osoitteiden lukeminen koneen muistista
12000	Listan kirjoittaminen paperille
13000	Nimien ja osoitteiden poistaminen tiedostosta
14000	Lataaminen — lukee tiedoston nauhalta tai levykkeeltä
15000	Tallentaminen — kirjoittaa tiedoston nauhalle tai levykkeelle
16000	Tietojen muuttaminen

Ohjelman käyttö

Postituslistaohjelman vaihtoehdot käyvät hyvin selville päämenusta. Asianmukaisen näppäimen painaminen vie sinut paikasta toiseen ohjelman sisällä. Muita vaihtoehtoja voidaan kuvata sanoilla selailu ja vertailu. Selailemalla voit katsoa kaikki nimi/osoitetiedoston tiedot ja tehdä omat päätöksesi matkan varrella. Voit myös verrata tiettyä nimeä tiedostossa olevaan nimeen.

Kun kirjoitat tietoja koneen muistiin, kone aina neuvoo, millaista tietoa se odottaa seuraavaksi saavansa. Kun haluat tehdä muutoksia, sinulta kysytään, millaisen tiedon haluat muuttaa.

Kaikkein tärkeintä on muistaa, että BASICillä kirjoitetussa ohjelmassa on tiettyjä rajoituksia. Pääasiallisin rajoitus on se, että pilkkuja ei voida käyttää kirjoitettaessa tietoja INPUT-lauseeseen. Esimerkiksi tieto

Erkki Jokinen, asianajaja
Purotie 6, 02300 ESPOO
(90) 123456

täytyy kirjoittaa koneeseen ilman pilkkuja, tähän tapaan:

Erkki Jokinen — asianajaja
Purotie 6
ESPOO
02300
(90)123456

Voit myös jättää jonkin tiedon pois kirjoittamalla koneeseen tyhjän merkkijonon (painamalla näppäintä RETURN).

Ole huolellinen siirtyessäsi toiminnasta toiseen ohjelman sisällä. Joskus on mahdollista käydä välillä päämenussa, joskus se taas tuottaa virheilmoituksen. Jos yrität lukea tietoa nauhalta tai levykkeeltä luomatta ensin tiedostoa, saat joko virheilmoituksen tai joudut odottamaan ikuisuuksia nauhurin etsiessä tiedostoa jota ei ole edes olemassa. Jos yrität kirjoittaa listan paperille, mutta et ole kytkenyt kirjoitinta paikalleen, ohjelma pysähtyy. Kummassakin näistä tapauksista menetät mahdollisesti jo kirjoittamasi tiedot, kun annat käskyn RUN saadaksesi ohjelman jälleen toimimaan.

Kaikkein selvin ohjelmasta puuttuva seikka on mahdollisuus järjestää lista jonkin ominaisuuden mukaan. Lajitteluohjelmat ovat kuitenkin usein monimutkaisia ja BASICillä kirjoitettuina hitaita.

Toinen rajoittava seikka on peräkkäistiedostojen käyttö. Koko tiedosto siirretään aina koneen muistiin nauhalta tai levykkeeltä ja päinvastoin. Ohjelma on asetettu käyttämään 600 merkkimuuttujan taulukoita. Koska merkkimuuttuja voi olla jopa 255 merkkiä pitkä, voivat pitkät merkkijonot viedä liikaa tilaa, jos kaikki 600 muuttujaa ovat käytössä. Commodoren tietokoneilla voidaan toki luoda toisenlaisiakin tiedostoja, joita kutsutaan *hajatiedostoiksi*. Jos käyttäisimme hajatiedostoja, olisi mahdollista käyttää koko levyketila yhteen ainoaan tiedostoon.

(Hajatiedostojen luominen on hyvin vaikeaa ja monimutkaista Commodore 64:n BASICillä. Hajatiedostot eivät kuulu tämän kirjan piiriin, mutta niistä saa tietoa perusteellisista BASIC-oppaista.)

Tietokantaohjelman valitseminen

Mikään myytävänä oleva tietokantaohjelma ei ole täysin samanlainen kuin muut, joten niihin kannattaa tutustua ennen ostoa. Kun on hankkinut tietyn ohjelman, ei vaihtaminen ole yleensä helppoa, koska eri ohjelmien luomat tiedostot eivät tavallisesti sovi keskenään yhteen. Toinen vaihtoehto on tietenkin käyttää useita ohjelmia, jos tarvittavat sovellutukset ovat kovin erilaiset.

Ensimmäisen huomioon otettavan seikan tietokantaohjelmaa valittaessa tulisi olla kieli, jolla se on kirjoitettu. BASIC ei yleensä ole sopiva kieli tietokantaohjelmaan, ja kokonaan BASICillä kirjoitettuja ohjelmia tulisi välttää, koska ne ovat hitaita ja joustamattomia. Useimmat kelvolliset tietokantaohjelmat ovat konekielellä kirjoitettuja.

Joustavuus on kaikkein tärkeintä. Eräät tietokantaohjelmat asettavat rajoituksia yksittäisten muistiinmerkintöjen koolle, määrälle ja muodolle. Hyvä ohjelma sallii jopa yhtä kuvaruutua laajemmat muistiinmerkinnät. Tärkeä mielessä pidettävä seikka on se, että ohjelmalla pitäisi kyetä tekemään melkein kaikki mikä on mahdollista tavallisella paperikansiomenetelmälläkin.

Jokaisessa kunnon tietokantaohjelmassa on myös nopea lajittelumahdollisuus. Paitsi että pystyt lajittelemaan tietoa yhden ominaisuuden perusteella, sinun tulisi myös kyetä valitsemaan sellaiset tiedot, joilla on enemmän kuin yksi yhteinen ominaisuus. Jos sinulla on siis pitkä postituslista, jossa on myös tiettyjen seura- tai ammattitovereiden nimet, voit kenties haluta postituslistallesi ne, jotka asuvat Helsingissä ja jotka ovat seurasi jäseniä. Joustava ja hyvin tehty tietokantaohjelma suoriutuu tästä.

Useimmat kelvolliset tietokantaohjelmat antavat myös mahdollisuuden kirjoittaa raportteja. Raportti on lista, joka sisältää koko tietokannan alueelta valittua tietoa. Tehokas raporttiohjelma antaa muitakin mahdollisuuksia. Sillä voidaan suunnitella, millä tavalla tieto kirjoitetaan paperille, ja se täyttää etukäteen suunniteltuja lomakkeita. Siihen voi jopa liittyä matemaattisia ominaisuuksia, joiden avulla voit suorittaa tietokantaan perustuvia laskutoimituksia.

Commodoren tietokoneille suunnitellun tietokannan tulisi aina käyttää vain hajatiedostoja. Vaikka jotkin tietokantaohjelmat saattavat käyttää omia erityisesti suunniteltuja tiedostojaan, pelkästään peräkkäistiedostoja käyttävät ohjelmat ovat yleensä käyttömahdollisuuksiltaan hyvin rajallisia aivan perussovellutuksia lukuun ottamatta.

Tunnet kyllä kelvollisen tietokantaohjelman, kun näet sen. Koska tällaiset ohjelmat ovat hyvin monimutkaisia, ne voivat olla vaikeita omaksua. Sinun tulisi kyetä käyttämään ohjelmaa tarvitsematta pitää koko ajan vieressäsi avattua käsikirjaa. Ohjelman tulisi olla looginen ja sen käskyjen helppoja muistaa. Luonnollisesti sen tehokkaimmat toiminnot saattavat vaatia opiskelua ja harjoittelua, mutta sinun tulisi pystyä käyttämään ohjelmaa sen yksinkertaisimmassa muodossa hyvin vähällä harjoittelulla.

Tiedon varastointi ja lukeminen tiedostosta ovat jokaisen tietokoneen parhaimpia ja hyödyllisimpiä ominaisuuksia. Saatat kuitenkin hämmästyä havaitessasi, että jo melko vähäiselläkin harjoittelulla pystyt itsekin kirjoittamaan sellaisen arkistointiohjelman, joka sopii omiin yksilöllisiin tarpeisiisi.

```
10 REM ** OSOITEREKISTERIOHJELMA **
15 X=101 :
   DIM N$(X),L$(X),P$(X),Z$(X),T$(X) :
   N=0
20 POKE 53280,12 : REM REUNA HARMAA 2
30 POKE 53281,15 : REM RUUTU HARMAA 3
40 PRINT CHR$(14) :
   REM VAIHDA PIENIIN KIRJAIMIIN
```

```
50 PRINT CHR$(8) :
   REM POISTA COMM NAPPAIN
60 PRINT "(CLR)(BLK)(RVS ON)(SHIFT O)SOIT
   EREKISTERI"
70 PRINT  :
   PRINT "(RVS ON)F1(RVS OFF) (SHIFT K)IR
   JOITA"
80 PRINT "(RVS ON)F3(RVS OFF) (SHIFT L)UE
   "
90 PRINT "(RVS ON)F5(RVS OFF) (SHIFT T)UL
   OSTA"
100 PRINT "(RVS ON)F7(RVS OFF) (SHIFT P)Y
    YHI POIS"
110 PRINT  :
    PRINT "(RVS ON)F2(RVS OFF) (SHIFT L)A
    TAA MUISTIIN"
120 PRINT "(RVS ON)F4(RVS OFF) (SHIFT T)A
    LLETA"
130 PRINT "(RVS ON)F6(RVS OFF) (SHIFT M)U
    UTA"
140 GET C$ : IF C$="" THEN 140
150 IF C$="(F1)" THEN GOSUB 10000
160 IF C$="(F3)" THEN GOSUB 11000
170 IF C$="(F5)" THEN GOSUB 12000
180 IF C$="(F7)" THEN GOSUB 13000
190 IF C$="(F2)" THEN GOSUB 14000
200 IF C$="(F4)" THEN GOSUB 15000
210 IF C$="(F6)" THEN GOSUB 16000
220 GOTO 60
10000 REM ** KIRJOITA **
10002 IF N>=X-1 THEN RETURN
10004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA"
10005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
10006 GET C$ : IF C$="" THEN 10006
10007 IF C$="(F1)" THEN 10010
10008 IF C$="(F8)" THEN RETURN
10009 GOTO 10000
10010 N=N+1
10020 PRINT "(CLR)(RVS ON)(SHIFT K)IRJOIT
      A NIMET/OSOITTEET"
10030 PRINT "(SHIFT O)SOITE #";N;SPC(3);X
      -1;" STA"
```

```
10040 INPUT "(SHIFT N)IMI:(13 VALIA)";N$(
      N)
10050 INPUT "(SHIFT K)ATUOSOITE:(7 VALIA)
      ";L$(N)
10060 INPUT "(SHIFT P)OSTITOIMIPAIKKA: ";
      P$(N)
10070 INPUT "(SHIFT P)OSTINUMERO:(6 VALIA
      )";Z$(N)
10080 INPUT "(SHIFT P)UHELIN:(10 VALIA)";
      T$(N)
10100 PRINT  :
      PRINT "(RVS ON)(SHIFT U)USI OSOITE
      ((SHIFT K)/(SHIFT E))"
10110 GET C$ : IF C$="" THEN 10110
10115 IF N>=X-1 THEN RETURN
10120 IF C$="K" THEN GOTO 10010
10130 IF C$="E" THEN RETURN
10140 GOTO 10010
11000 REM ** LUE **
11005 IF N=0 THEN RETURN
11010 PRINT "(CLR)(RVS ON)(SHIFT L)UE(2 V
      ALIA)NIMET/OSOITTEET"
11020 PRINT
11030 PRINT "(RVS ON)F1(RVS OFF) (SHIFT L
      )UE KAIKKI"
11040 PRINT "(RVS ON)F3(RVS OFF) (SHIFT V
      )ERTAA NIMEEN"
11050 GET C$ : IF C$="" THEN 11050
11060 IF C$="(F1)" THEN 11090
11070 IF C$="(F3)" THEN 11220
11080 GOTO 11000
11090 FOR I=1 TO N
11100 PRINT "(CLR)(SHIFT O)SOITE #";I;SPC
      (3);N;":STA"
11110 PRINT "(SHIFT N)IMI:(13 VALIA)";N$(
      I)
11120 PRINT "(SHIFT K)ATUOSOITE:(7 VALIA)
      ";L$(I)
11130 PRINT "(SHIFT P)OSTITOIMIPAIKKA: ";
      P$(I)
11140 PRINT "(SHIFT P)OSTINUMERO:(6 VALIA
      )";Z$(I)
11150 PRINT "(SHIFT P)UHELIN:(10 VALIA)";
      T$(I)
```

```
11170 PRINT  :
      PRINT "(RVS ON)(SHIFT P)AINA NAPPAI
      NTA JATKAAKSESI"
11180 GET C$ : IF C$="" THEN 11180
11200 NEXT I
11210 RETURN
11220 INPUT "(CLR)(SHIFT M)IHIN NIMEEN VE
      RRATAAN";T$
11230 FOR I=1 TO N
11240 IF N$(I)=T$ THEN GOTO 11290
11250 NEXT I
11260 PRINT  :
      PRINT "(RVS ON)(SHIFT N)IMEA EI OLE
       TIEDOSTOSSA"
11270 PRINT  :
      PRINT "(SHIFT P)AINA NAPPAINTA PALA
      TAKSESI VALINTAAN"
11280 GET C$ : IF C$="" THEN 11280
11285 RETURN
11290 PRINT "(CLR)(SHIFT O)SOITE #";I;SPC
      (3);I":STA"
11300 PRINT "(SHIFT N)IMI:(13 VALIA)";N$(
      I)
11310 PRINT "(SHIFT K)ATUOSOITE:(7 VALIA)
      ";L$(I)
11320 PRINT "(SHIFT P)OSTITOIMIPAIKKA: ";
      P$(I)
11330 PRINT "(SHIFT P)OSTINUMERO:(6 VALIA
      )";Z$(I)
11340 PRINT "(SHIFT P)UHELIN:(10 VALIA)";
      T$(I)
11360 PRINT  :
      PRINT "(RVS ON)(SHIFT P)AINA NAPPAI
      NTA PALATAKSESI VALINTAAN"
11370 GET C$ : IF C$="" THEN 11370
11380 RETURN
12000 REM ** TULOSTUS KIRJOITTIMELLE **
12002 IF N=0 THEN RETURN
12004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA"
12005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
12006 GET C$ : IF C$="" THEN 12006
12007 IF C$="(F1)" THEN 12010
```

```
12008 IF C$="(F8)" THEN RETURN
12009 GOTO 12000
12010 PRINT "(CLR)(RVS ON)(SHIFT T)ULOSTA
      "
12020 OPEN 1,4,4
12030 FOR I=1 TO N
12040 PRINT#1,N$(I)
12050 PRINT#1,L$(I)
12060 PRINT#1,P$(I)
12070 PRINT#1,Z$(I)
12080 PRINT#1,T$(I)
12100 PRINT#1,CHR$(13)
12110 NEXT I
12120 CLOSE 1
12130 RETURN
13000 REM ** PYYHI **
13002 IF N=0 THEN RETURN
13004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA"
13005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
13006 GET C$ : IF C$="" THEN 13006
13007 IF C$="(F1)" THEN 13010
13008 IF C$="(F8)" THEN RETURN
13009 GOTO 13000
13010 PRINT "(CLR)(RVS ON)(SHIFT P)YYHI N
      IMET/OSOITTEET"
13030 PRINT "(RVS ON)F1(RVS OFF) (SHIFT S
      )ELAA KAIKKI"
13040 PRINT "(RVS ON)F3(RVS OFF) (SHIFT V
      )ERTAA NIMEEN"
13050 GET C$ : IF C$="" THEN 13050
13060 IF C$="(F1)" THEN 13090
13070 IF C$="(F3)" THEN 13230
13080 GOTO 13000
13090 FOR I=1 TO N
13100 PRINT "(CLR)(SHIFT O)SOITE #";I;SPC
      (3);N;":STAA"
13110 PRINT "(SHIFT N)IMI:(13 VALIA)";N$(
      I)
13120 PRINT "(SHIFT K)ATUOSOITE:(7 VALIA)
      ";L$(I)
13130 PRINT "(SHIFT P)OSTITOIMIPAIKKA: ";
      P$(I)
```

```
13140 PRINT "(SHIFT P)OSTINUMERO:(6 VALIA
      )";Z$(I)
13150 PRINT "(SHIFT P)UHELIN:(6 VALIA)";T
      $(I)
13170 PRINT  :
      PRINT "(RVS ON)F1(RVS OFF) (SHIFT J
      )ATKA"
13180 PRINT  :
      PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )YYHI"
13190 GET C$ : IF C$="" THEN 13190
13200 IF C$="(F1)" THEN GOTO 13225
13210 IF C$="(F8)" THEN GOSUB 13800 :
      GOTO 13224
13220 GOTO 13190
13224 N=N-1 : I=I-1
13225 NEXT I
13226 RETURN
13230 INPUT "(CLR)(SHIFT M)IHIN NIMEEN VE
      RRATAAN";T$
13240 FOR I=1 TO N
13250 IF N$(I)=T$ THEN GOSUB 13800 :
      N=N-1 : I=I-1
13260 NEXT I
13265 RETURN
13270 PRINT  :
      PRINT "(RVS ON)(SHIFT N)IMEA EI OLE
       TIEDOSTOSSA"
13280 PRINT  :
      PRINT "(SHIFT P)AINA NAPPAINTA PALA
      TAKSESI VALINTAAN"
13290 GET C$ : IF C$="" THEN 11280
13800 FOR K=I TO N
13810 N$(K)=N$(K+1)
13820 L$(K)=L$(K+1)
13830 P$(K)=P$(K+1)
13840 Z$(K)=Z$(K+1)
13850 T$(K)=T$(K+1)
13870 NEXT K
13880 RETURN
14000 REM ** LATAA MUISTIIN **
14004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA"
14005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
```

```
14006 GET C$ : IF C$="" THEN 14006
14007 IF C$="(F1)" THEN 14010
14008 IF C$="(F8)" THEN RETURN
14009 GOTO 14000
14010 PRINT "(CLR)(RVS ON)(SHIFT L)ATAA T
      IEDOSTO MUISTIIN"
14020 PRINT :
      PRINT "(RVS ON)F1(RVS OFF) (SHIFT L
      )ATAA LEVYLTA"
14030 PRINT "(RVS ON)F4(RVS OFF) (SHIFT L
      )ATAA KASETILTA"
14040 GET C$ : IF C$="" THEN 14040
14050 IF C$="(F1)" THEN 14100
14060 IF C$="(F4)" THEN 14250
14070 GOTO 14040
14100 OPEN 1,8,3,"0:OSOITELISTA,SEQ"
14140 INPUT#1,N
14150 FOR I=1 TO N
14160 INPUT#1,N$(I)
14170 INPUT#1,L$(I)
14180 INPUT#1,P$(I)
14190 INPUT#1,Z$(I)
14200 INPUT#1,T$(I)
14220 NEXT I
14230 CLOSE 1
14240 RETURN
14250 OPEN 1,1,0,"OSOITELISTA"
14260 GOTO 14140
15000 REM ** TALLETA **
15002 IF N=0 THEN RETURN
15004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA "
15005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
15006 GET C$ : IF C$="" THEN 15006
15007 IF C$="(F1)" THEN 15010
15008 IF C$="(F8)" THEN RETURN
15009 GOTO 15000
15010 PRINT "(CLR)(RVS ON)(SHIFT T)ALLETA
       TIEDOSTO"
15020 PRINT :
      PRINT "(RVS ON)F1(RVS OFF) (SHIFT L
      )EVYLLE"
```

```
15030 PRINT "(RVS ON)F4(RVS OFF) (SHIFT K
      )ASETILLE"
15040 GET C$ : IF C$="" THEN 15040
15050 IF C$="(F1)" THEN 15100
15060 IF C$="(F4)" THEN 15250
15070 GOTO 15040
15100 OPEN 1,8,15
15110 PRINT#1,"S0:OSOITELISTA"
15120 CLOSE 1
15130 OPEN 1,8,3,"0:OSOITELISTA,SEQ,WRITE
      "
15140 PRINT#1,N
15150 FOR I=1 TO N
15160 PRINT#1,N$(I)
15170 PRINT#1,L$(I)
15180 PRINT#1,P$(I)
15190 PRINT#1,Z$(I)
15200 PRINT#1,T$(I)
15220 NEXT I
15230 CLOSE 1
15240 RETURN
15250 OPEN 1,1,1,"OSOITELISTA"
15260 GOTO 15140
16000 REM ** MUUTA **
16002 IF N=0 THEN RETURN
16004 PRINT "(CLR)(RVS ON)F1(RVS OFF) (SH
      IFT J)ATKA"
16005 PRINT "(RVS ON)F8(RVS OFF) (SHIFT P
      )ALUU VALINTAAN"
16006 GET C$ : IF C$="" THEN 16006
16007 IF C$="(F1)" THEN 16010
16008 IF C$="(F8)" THEN RETURN
16009 GOTO 16000
16010 PRINT "(CLR)(RVS ON)(SHIFT M)UUTA O
      SOITE"
16020 PRINT :
      PRINT "(RVS ON)F1(RVS OFF) (SHIFT S
      )ELAA JA MUUTA"
16030 PRINT "(RVS ON)F3(RVS OFF) (SHIFT V
      )ERTAA"
16040 GET C$ : IF C$="" THEN 16040
16050 IF C$="(F1)" THEN 16100
16060 IF C$="(F3)" THEN 16300
16070 GOTO 16040
```

```
16100 FOR I=1 TO N
16110 PRINT "(CLR)";N$(I)
16120 PRINT L$(I)
16130 PRINT P$(I)
16140 PRINT Z$(I)
16150 PRINT T$(I)
16170 PRINT
      PRINT "(RVS ON)F1(RVS OFF) (SHIFT J
      )ATKA"
16180 PRINT "(RVS ON)F3(RVS OFF) (SHIFT M
      )UUTA"
16190 GET C$ : IF C$="" THEN 16190
16200 IF C$="(F1)" THEN GOTO 16230
16210 IF C$="(F3)" THEN GOSUB 16800 :
      GOTO 16230
16220 GOTO 16190
16230 NEXT I
16240 RETURN
16300 PRINT "(CLR)"
16310 INPUT "(SHIFT M)IHIN NIMEEN VERRATA
      AN";M$
16320 FOR I=1 TO N
16330 IF N$(I)=M$ THEN GOSUB 16500 :
      RETURN
16340 NEXT I
16350 PRINT "(RVS ON)(SHIFT N)IMEA EI OLE
       TIEDOSTOSSA"
16360 PRINT "(RVS ON)(SHIFT P)AINA NAPPAI
      NTA PALATAKSESI VALINTAAN"
16370 GET C$ : IF C$="" THEN 16370
16380 RETURN
16500 PRINT "(CLR)";N$(I)
16510 PRINT L$(I)
16520 PRINT P$(I)
16530 PRINT Z$(I)
16540 PRINT T$(I)
16800 PRINT "1 -- (SHIFT N)IMI"
16810 PRINT "2 -- (SHIFT K)ATUOSOITE"
16820 PRINT "3 -- (SHIFT P)OSTITOIMIPAIKK
      A"
16830 PRINT "4 -- (SHIFT P)OSTINUMERO"
16840 PRINT "5 -- (SHIFT P)UHELIN"
16860 PRINT
      INPUT "(SHIFT M)INKA MUUTAT";C
```

```
16870 ON CGOSUB 16910,16920,16930,16940,1
      6950
16880 RETURN
16910 PRINT
      INPUT "(SHIFT K)ORVAAVA";D$
      M$(I)=D$ : I=I-1 : RETURN
16920 PRINT
      INPUT "(SHIFT K)ORVAAVA";D$
      L$(I)=D$ : I=I-1 : RETURN
16930 PRINT
      INPUT "(SHIFT K)ORVAAVA";D$
      P$(I)=D$ : I=I-1 : RETURN
16940 PRINT
      INPUT "(SHIFT K)ORVAAVA";D$
      Z$(I)=D$ : I=I-1 : RETURN
16950 PRINT
      INPUT "(SHIFT K)ORVAAVA";D$
      T$(I)=D$ : I=I-1 : RETURN
```

6

Ohjelmointi — lisää BASIC-käskyjä

Tässä luvussa saat tietoja siitä, millä tavalla voit käyttää useampia BASIC-käskyjä. Opit jotakin käskyistä PEEK ja POKE, SYS ja USR, CMD, SPC ja TAB, STOP ja WAIT sekä ON.

Jos olet kiinnostunut tietokoneen käytöstä matemaattisiin tarkoituksiin, haluat varmasti tietää jotakin matemaattisista funktiosta INT, ABS, SGN, SQR, SIN, COS, TAN, ATN, DEF ja FN.

Saat myös tietää, kuinka merkkijonoja voidaan hajottaa ja taas koota käskyillä LEN, VAL, STR$, ASC, CHR$, LEFT$, MID$ ja RIGHT$.

PEEK ja POKE

Sanat PEEK (kurkistaa) ja POKE (tökätä) ovat oikeastaan hyvin selkeitä, vaikka ne joskus herättävätkin englannintaitoisissa hykertelyä. PEEK-käskyllä voit katsoa, mitä koneen muistissa tietyssä muistipaikassa on, ja POKE taas sijoittaa arvon haluttuun muistipaikkaan. Näitä sanoja kutsutaan usein konetason käskyiksi, koska ne vaikuttavat suoraan koneen muistiin.

PEEK- ja POKE-käskyjen avulla voidaan leikkiä tietokoneella. POKE-käsky on myös tarpeen tehtäessä värigrafiikkaa ja ääntä Commodore 64:llä. Tiettyjen numeroiden sijoittaminen koneen muistipaikkoihin saattaa vaikuttaa jopa siihen, kuinka kone toimii. (Voit myös ''kaataa'' tietokoneen siten, että se ei enää reagoi käskyihisi, sijoittamalla POKE-käskyllä *väärän* numeron *väärään* muistipaikkaan. Onneksi kone ei siitä rikkoonnu. Sammuta se vain hetkeksi, ja kun kytket taas virran, kaikki on ennallaan. Käsky PEEK ei koskaan vaikuta tietokoneen toimintakuntoon millään lailla.)

Käskyä POKE seuraa aina muistipaikan osoite, pilkku ja sen jälkeen numeerinen muuttuja tai numero joka sijoitetaan tuohon muistipaikkaan. POKE-käskyllä voidaan muistipaikkoihin sijoittaa vain numeroita väliltä 0—255. (Jos haluat tietää syyn, lue kirjan lopussa olevia liitteitä.) Commodore 64:ssä on 64K eli 65 536 muistipaikkaa, joihin voidaan POKE-käskyllä sijoittaa numeroita. Käskyä POKE välittömästi seuraavan numeron on oltava kokonaisluku 0—65535 tai tällaisia lukuja merkitsevä numeerinen muuttuja.

Käskyä POKE voidaan käyttää ohjelmarivillä tai suoraan näppäimistöltä.

```
POKE 53272,23
```

tai

```
10 POKE 53272,23
```

Edellisessä esimerkissä luvun 23 kirjoittaminen muistipaikkaan 53272 vaihtaa tietokoneen grafiikkatilasta (jossa on käytettävissä isot kirjaimet ja graafiset merkit) kirjoituskonetilaan (jossa on käytettävissä isot ja pienet kirjaimet). Luvun 21 kirjoittaminen muistipaikkaan 53272 suorittaa vaihdon takaisin. Tämä muistipaikka on tietokoneen VIC-näyttöpiirissä.

Eri luvut saavat eri muistipaikoissa aikaan erilaisia asioita. Voidaksesi käyttää POKE-käskyä sinun on tutustuttava kirjan lopussa olevaan muistikarttaan. Koeta kuitenkin ensi hätään kirjoittaa käskyllä POKE luvut 0:sta 15:een muistipaikkoihin

52380 ja 52381. Katso sitten, mitä kuvaruudussa tapahtuu.

Käytä aina käskyä PEEK muistipaikan numeron kanssa (0-65535). Tämä numero on aina kirjoitettava sulkeisiin. Voit käyttää sitä ohjelmassa tai suoraan näppäimistöltä. Jälkimmäisessä tapauksessa kirjoitat yleensä tuloksen kuvaruutuun:

```
PRINT PEEK(53272)
```

Ohjelmassa voit myös varastoida PEEK-käskyn avulla saamasi numeron numeeriseen muuttujaan.

```
10 A = PEEK(53272)
```

****

ILLEGAL QUANTITY ERROR

Kun tämä virheilmoitus esiintyy ohjelmarivillä, jolla on käsky POKE, tarkista, ettei muistipaikkaan ole sijoitettu numeroa, joka olisi suurempi kuin 255 tai pienempi kuin 0. Jos olet käyttänyt muuttujia, tarkista niiden numeroarvo nähdäksesi, ettei niiden laskettu tulos ole yli 255 tai alle 0.

****

Käskyjen PEEK ja POKE käyttäminen voi tehdä ohjelmasta sekavan, koska kaikki toiminta ei ole selvää. Sitä paitsi liian monen PEEKin tai POKEn käyttäminen voi tehdä ohjelmastasi sellaisen, ettei se sovi yhteen Commodoren muiden tietokoneiden kanssa, koska kumpikin näistä sanoista käyttää hyväkseen kullekin konemallille ominaisia piirteitä. Toisinaan, etenkin grafiikkaa ja ääntä ohjelmoitaessa, on näiden sanojen välttäminen kuitenkin täysin mahdotonta.

SYS ja USR

Käskyä SYS käytetään vaihdettaessa BASICistä konekieliseen ohjelmaan tai aliohjelmaan. Konekieli on alkeellisempi — mutta myös vaikeampi — keino ohjelmoida tietokone. Sen etuna on

nopeus. Konekieliset ohjelmat voidaan usein suorittaa satoja kertoja nopeammin kuin vastaavat BASICillä kirjoitetut ohjelmat.

Käskyä SYS seuraa muistipaikka, josta konekielinen ohjelma tai aliohjelma alkaa. Sitä voidaan käyttää ohjelmallisesti tai suoraan näppäimistöltä. Esimerkki:

```
SYS(2061)
```

tai

```
10 SYS(2061)
```

Kun konekielinen aliohjelma on suoritettu, tietokone palaa BASIC-ohjelmaansa.

USR merkitsee suurin piirtein samaa kuin SYS, mutta se toimii eri tavalla. Se on monimutkaisempi eikä sitä käytetä ohjelmissa yhtä usein kuin käskyä SYS.

```
10 USR(10)
```

Tämä käsky lähettää ohjelman konekieliseen aliohjelmaan, joka alkaa muistipaikkoihin 785 ja 786 varastoidusta osoitteesta ja siirtää sulkumerkkien välisen luvun (tai numeerisen muuttujan arvon) tietokoneen sellaiseen osaan, joka tunnetaan nimellä ”liukulukuakku”.

Koska sekä SYS että USR ovat konekieleen liittyviä käskyjä, ei niitä ole tarkoitettu aloittelevien ohjelmoijien käyttöön. Toisinaan sinun on kuitenkin käytettävä käskyä SYS käskyn RUN asemesta aloittaaksesi konekielisen ohjelman suorituksen.

CMD

Käsky CMD toimii helppotajuisesti — se lähettää normaalisti kuvaruutuun PRINT-käskyllä kirjoitettavat tiedot johonkin toiseen laitteeseen, esimerkiksi kasettinauhuriin tai levykeasemaan. Tavallisesti sitä käytetään tiedon lähettämiseksi kirjoitti-

meen. Sitä seuraa laitenumero, ja sen kanssa käytetään käskyjä OPEN ja CLOSE, (katso lukua Kuinka tietokone varastoi tietoa) ja sitä seuraa tiedoston numero. Sitä voidaan käyttää ohjelmallisesti tai suoraan näppäimistöltä.

Seuraavassa on kaikkein tavallisin esimerkki käskyn CMD käytöstä. Siinä kirjoitetaan BASIC-ohjelman listaus paperille. Voit kirjoittaa sen suoraan näppäimistöltä tai käyttää sitä ohjelmallisesti.

```
OPEN 1,4,4
CMD 1
LIST
CLOSE 1
```

Ohjelma avaa laitteen 4 (kirjoittimen) tiedoston numero 1, ja toinen osoite on 4. CMD 1 viittaa tiedoston numeroon. Käsky LIST kirjoittaisi normaalisti ohjelmalistauksen kuvaruutuun, mutta koska käsky CMD siirsi toiminnan kirjoittimeen, se kirjoittuukin sinne. CLOSE 1 sulkee tiedoston 1.

SPC ja TAB

Kumpaakin näistä käskyistä käytetään PRINT-käskyn yhteydessä. SPC tarkoittaa tyhjiä välejä, TAB taas tabulaattoria (kuten kirjoituskoneessa).

```
  10 PRINT SPC(17)"KOE"
tai
  10 PRINT TAB(17)"KOE"
```

Kumpikin näistä komennoista siirtää kohdistinta 17 sanaväliä sen nykyisestä asemasta ja kirjoittaa sitten sanan KOE. TAB ja SPC ovat keskenään erilaisia siten, että käskyä TAB ei voida käyttää käskyn PRINT# yhteydessä (joka oli siis tiedostonkäsittelykäsky), mutta käskyä SPC voidaan. TAB laskee tyhjät

välit aina kuvaruudun vasemmasta laidasta (eli sarakkeesta 1) alkaen. Kun näitä sanoja käytetään käskyn PRINT kanssa, ne siirtyvät haluttuun paikkaan tuhoamatta kuitenkaan välissä olevia merkkejä. Tässä mielessä kumpikaan, SPC tai TAB, ei siis kirjoita tyhjiä välejä.

STOP ja WAIT

STOP on kätevä käsky ohjelmien testaamiseen. Se voidaan sijoittaa mihin tahansa kohtaan ohjelmassa, jotta nähtäisiin, kuinka ohjelma toimii. Jos esimerkiksi tietty ohjelman osa ei toimi kunnolla, voit sijoittaa käskyn STOP niille riveille, joilla ongelma esiintyy. Kun tietokone kohtaa käskyn STOP, se toimii samoin kuin jos olisi painettu näppäintä RUN/STOP. Näet kuvaruudussa tämäntapaisen viestin:

BREAK IN LINE 100 (tai sen rivin numero, jolla ohjelmassa on käsky STOP)

Nyt sitten tiedät, pääsikö tietokone tuolle riville, ja jos pääsi, ohjelma toimii siihen saakka hyvin. Ohjelman suorittamista voit jatkaa käskyllä CONT. (Jos käytät käskyä RUN, ohjelman suoritus alkaa alusta ja tyhjentää muuttujat.)

****

CAN'T CONTINUE ERROR

Käskyä CONT voidaan käyttää ohjelman suorituksen jatkamiseen sen jälkeen kun on painettu näppäintä RUN/STOP tai ohjelmassa on käytetty käskyä STOP. Jos kuitenkin muutat jonkin rivin ennen kuin annat käskyn CONT, näet tämän virheilmoituksen. Se näkyy myös silloin, kun ohjelmaa ei ole lainkaan suoritettu. Kummassakin tapauksessa anna uudelleen käsky RUN.

****

Käskyä WAIT käytetään muistipaikoissa olevien yksittäisten *bittien* muodostamien kokonaisuuksien havaitsemiseen. Kun

tietokone kohtaa käskyn WAIT, se pitää taukoa, kunnes tiettyyn muistipaikkaan sijoitettu luku vastaa WAIT-käskyn asettamaa ehtoa. Monet (ja ehkä useimmat) ohjelmoijat pitävät käskyä WAIT kovin mystisenä, ja Commodorekin suosittelee, ettet käyttäisi sitä. Mutta jos tunnet pystyväsi käyttämään sitä, niin etsi tietoja Commodoren BASICiä käsittelevistä teoksista.

ON

Käsky ON on BASICin älykkäimpiä ja kiintoisimpia käskyjä. Silti useat ohjelmoijat eivät koskaan käytä sitä. Se johtuu ehkä siitä, että käsky ON näyttää käytössä kummalliselta, joten ohjelmoijat eivät vaivaudu opiskelemaan sen käyttöä. Se on kuitenkin helppoa.

Käskyä ON käytetään numeerisen muuttujan ja yhden tai useamman GOTOn tai GOSUBin kanssa. Tyypillinen ON-rivi näyttää tällaiselta:

```
20 ON A GOSUB 100,150,180,220
```

Jos A:lla merkitty numero on yksi, ohjelma hyppää käskyllä GOSUB aliohjelmaan, joka alkaa rivillä 100. Jos A on 2, se hyppää rivillä 150 alkavaan aliohjelmaan. Numero 3 saa ohjelman hyppäämään rivillä 180 alkavaan aliohjelmaan ja niin edelleen.

Kokeile tätä esimerkkiä:

```
10 INPUT "KIRJOITA JOKIN LUVUISTA 1—5";A
20 ON A GOSUB 100,150,180,220
30 GOTO 10
100 PRINT "YKSI"
110 RETURN
150 PRINT "KAKSI"
160 RETURN
180 PRINT "KOLME"
```

```
190 RETURN
220 PRINT ''NELJA''
230 RETURN
```

Ohjelma kirjoittaa ensiksi ruutuun KIRJOITA JOKIN LUVUISTA 1—5 ja odottaa sitten, kunnes kirjoitat numeron ja painat näppäintä RETURN. Tästä numerosta tulee numeerinen muuttuja A. Jos rivillä 20 A = 1, ohjelma hyppää aliohjelmaan riville 100, kirjoittaa ruutuun YKSI ja palaa pääohjelmaan. Sitten koko juttu alkaa alusta ja pyytää uutta numeroa. Jos kirjoitat numeron 1—4, näet ruudussa numeron kirjaimin kirjoitettuna. Jos kirjoitat numeron 5 tai nollan, jätetään ON-lause huomiotta.

Toisinaan ON-käskyä voidaan käyttää korvaamaan lukuisia IF/THEN-lauseita. Etsi ohjelmasta sellaiset kohdat, joissa IF etsii tiettyä numeroa. Jos edellä olevassa ohjelmassa olisi käytetty ON-käskyn asemesta käskyjä IF ja THEN, se näyttäisi seuraavalta:

```
10 INPUT ''KIRJOITA JOKIN LUVUISTA 1—5'';A
20 IF A=1 THEN GOSUB 100
30 IF A=2 THEN GOSUB 150
40 IF A=3 THEN GOSUB 180
50 IF A=4 THEN GOSUB 220
60 GOTO 10
100 PRINT ''YKSI''
110 RETURN
150 PRINT ''KAKSI''
160 RETURN
180 PRINT ''KOLME''
190 RETURN
220 PRINT ''NELJA''
230 RETURN
```

Käskyn ON käyttäminen on tapa tehdä ohjelmista ''elegantimpia'', selkeämmin ja tehokkaammin kirjoitettuja. Se säästää

ohjelman kirjoittamiseen kuluvaa aikaa, ja vaikka et joutuisikaan koettelemaan tietokoneesi muistin rajoja, se joka tapauksessa säästää muistitilaakin.

Matemaattiset funktiot

Yksi tietokoneen luontaisista tehtävistä on suorittaa toistuvia ja monimutkaisia laskutoimituksia. Tietokoneveteraanit kutsuvat sitä "numeroiden murskaukseksi".

Koska tämä on tietokoneteos eikä mikään matematiikan oppikirja, ei Commodoren BASICin matemaattisia funktioita ole kuvailtu kovin yksityiskohtaisesti. Jos haluat lisää tietoa siitä, mitä ne merkitsevät ja kuinka niitä käytetään yhtälöissä, lue matematiikan oppikirja.

INT merkitsee kokonaislukua, ja se muodostaa desimaaliluvusta kokonaisluvun. Se on BASICin matemaattisista funktioista käytetyin. Katso tätä esimerkkiä:

```
10 A = 10.258
20 PRINT INT(A)
```

INT yksinkertaisesti poistaa luvun desimaaliosan ja kirjoittaa ruutuun numeron 10. Ole kuitenkin tarkkana, sillä käskyä INT ei voida käyttää lukujen pyöristämiseen. Toisin sanoen luvusta 10.568 *ei* tule 11.

Käsky ABS poistaa miinusmerkin negatiivisten lukujen edestä. Käskyä ABS käytetään näin:

```
10 A = -10
20 PRINT ABS(A)
```

SGN määrittelee, onko luku positiivinen vai negatiivinen. Jos luku on positiivinen, SGN saa arvon 1. Jos luku on negatiivinen, SGN:n arvoksi tulee -1. Kokeilepa tätä:

```
10 A = -20.258
20 PRINT SGN(A)
```

Tai tätä:

```
10 A = -20.258
20 IF SGN(A) = 1 THEN PRINT ''POSITIIVINEN LUKU''
30 IF SGN(A) = -1 THEN PRINT ''NEGATIIVINEN
   LUKU''
```

Muuta lukua A siten, että näet, kuinka SGN toimii. (Sivumennen sanoen SGN(0) = 0.)

SQR antaa luvun neliöjuuren.

```
10 PRINT SQR(2)
```

SIN, COS ja TAN antavat kulmien SINin, COSinin ja TANgentin, jotka mitataan aina radiaaneina.

ATN antaa luvun Arcus TaNgentin radiaaneina. Jos haluat muuntaa asteina ilmoitetun kulman radiaaneiksi, voit käyttää kaavaa A(radiaanit) = A(asteet) kertaa pii jaettuna 180:llä.

```
10 PRINT SIN(2)
20 PRINT COS(2)
30 PRINT TAN(2)
40 PRINT ATN(2)
```

DEF tarkoittaa määrittää ja FN tarkoittaa FuNktiota. Näiden yhdistelmää käytetään varastoimaan kahden muuttujan avulla nimetty yhtälö ja käyttämään tuota funktiota ohjelmassa. Toinen muuttuja siirtyy yhtälöön, esimerkiksi:

```
10 DEF FN B(D) = X + Y + D
20 X = 5:Y = 6
30 FOR D = 1 TO 10
40 PRINT FN B(D)
50 NEXT D
```

Tässä esimerkissä rivi 10 määrittelee funktion nimeltä B(D) tulokseksi laskusta X + Y + D. Rivillä 20 X saa arvon 5 ja Y arvon 6. Rivit 30—50 kirjoittavat ruutuun kymmenen tulosta yhtälöstä B(D) laskutoimituksen X + Y + D perusteella, siis luvut 12:sta 21:een D:n arvon muuttuessa 1:stä 10:een.

Merkkijonojen käsittely

Tietokone pystyy käsittelemään merkkijonoja yhtä hyvin kuin numeroita. Tiedät jo, että merkkijonoja voidaan liittää yhteen (mutta ei vähentää toisistaan).

Merkkijonojen käsittelykäskyt ovat Commodoren BASICin tehokkaimpia käskyjä. LEN määrittelee merkkijonon pituuden merkkeinä. VAL voi muuntaa numeroita sisältävän merkkijonon varsinaisiksi numeroiksi, joita tietokone pystyy sitten käsittelemään. STR$ puolestaan muuntaa minkä tahansa numeron merkkijonoksi. ASC on lyhennys sanasta ASCII. Se taas on numerokoodi, jonka avulla kaikki merkit määritellään. Käskyä CHR$ käytetään ASCII-numerolla kutsutun merkin kirjoittamiseen.

LEFT$, MID$ ja RIGHT$ ovat sanoja, joiden avulla voidaan ottaa erilleen tietty osa jostain merkkijonosta.

LEN, VAL ja STR$

Merkkijono voi olla 255 merkkiä pitkä. Joissakin ohjelmissa voi olla tarpeen tietää tarkalleen, kuinka monta merkkiä johonkin merkkijonoon sisältyy. Ensimmäiseksi mieleen tuleva tapa olisi kai kirjoittaa kyseinen merkkijono ruutuun ja laskea, kuinka paljon tilaa se vie. Se ei ole kuitenkaan aina mahdollista, koska eräät merkkijonot tekevät kuvia ja toisissa taas voi olla näkymättömiä tyhjiä välejä tai kohdistimen siirtoja. Merkkijonojen pituus saattaa myös muuttua joka kerta ohjelmaa suoritettaessa. Ja jos merkkijono esimerkiksi sijoitetaan ohjelmaan

INPUT-käskyn avulla, et pysty ennalta määräämään, kuinka monta merkkiä siihen sisältyy.

Aina käytännöllinen tapa on käyttää BASIC-käskyä LEN. Se on hämmästyttävän helppokäyttöinen ja käytettävissä ohjelmallisesti tai suoraan näppäimistöltä.

```
A$ = ''ABCD1234''
PRINT LEN(A$)
```

tai

```
10 A$ = ''ABCD1234''
20 X = LEN(A$)
```

Kummassakin tapauksessa vastaus kysymykseen LEN(A$) on 8. Tämän käskyn käyttö selviää paremmin, kun jatkat lukemista.

Merkkijonohan voi koostua minkälaisista merkeistä tahansa — kirjaimista, symboleista, kohdistimen siirroista, editointikäskyistä ja jopa numeroista. Esimerkiksi ''1234'' on merkkijono ja 1234 on numero. Et voi laskea yhteen merkkijonoa ja numeroa, mutta voit sen sijaan *muuntaa* merkkijonon numeroksi. Se onnistuu pyytämällä tietokoneelta lainausmerkeissä olevien numeroiden lukuarvoja käskyllä VAL.

```
10 A$ = ''1234''
20 B = 1234
30 A = VAL(A$)
40 PRINT A + B
```

Eräs VAL-käskyn käyttömahdollisuus on auttaa GET-käskyä etsimään numeroita näppäimistöltä. Muistat varmaan, että GET-käskyn käyttö siihen on aika hankalaa. Kokeile tätä esimerkkiä:

```
10 GET C$:IF C$ = '''' THEN GOTO 10
20 C = VAL(C$)
30 PRINT C + 100
```

Käskyllä GET haetaan *merkkijono* C$. Emme voi suorittaa laskutoimituksia C$:lla, joten teemme C:stä yhtä suuren kuin C$:n lukuarvo. Jos C$ on mikä tahansa muu näppäin — esimerkiksi kirjain tai kohdistinnäppäin — C:n arvoksi tulee 0. Kuten huomaat, tämä toimii hyvin silloin, kun kyseessä on numero väliltä 0—9. Entäpä jos tarvitset lukua väliltä 0—99? Tässä yksi ratkaisu:

```
10 PRINT "KIRJOITA JOKIN LUKU 0—99"
20 GET C$:IF C$ = "" THEN GOTO 20
30 PRINT C$;
40 GET D$:IF D$ = "" THEN GOTO 40
50 PRINT D$
60 E$ = C$ + D$
70 E = VAL(E$)
80 PRINT E + 100
```

Huomasit varmaan, että olemme lisänneet yhden askelen. Kirjoittamalla C$:n ja sen viereen ruutuun D$:n näemme, mitä näppäimiä olemme painaneet. Kaksi yksimerkkistä merkkijonoa, C$ ja D$, liitetään yhteen, jotta saisimme kaksimerkkisen merkkijonon, josta sitten otetaan lukuarvo VAL-käskyllä. Muista kuitenkin VAL-käskyn luonne. Jos painat näppäintä 4 ja sitten jotain muuta kuin numeronäppäintä, esimerkiksi kirjainta M, E$:n arvo on 4, ei 40. Ja jos painat ensiksi jotain muuta kuin numeronäppäintä, E$:n arvo on 0. Kun käytät jotain tällaista ohjelmaa, olisi hyvä katsoa ensiksi, onko kukin painettu näppäin juuri numeronäppäin.

Jos jostain syystä haluat muuntaa numeron merkkijonoksi, voit kääntää edellä esitetyn menettelyn päinvastaiseksi käskyllä STR$.

```
10 A = 1234
20 A$ = STR$(A)
```

Miksi sitten haluaisit muuntaa jonkin luvun merkkijonoksi?

Eräs syy saattaisi olla se, että numerot ja merkkijonot kirjoitetaan ruutuun erilaisten sääntöjen mukaan. Muistat varmaan, että kun kirjoitat numeron merkkijonon tai toisen numeron viereen puolipistettä käyttäen, tulee ruutuun ylimääräinen tyhjä väli. Muuntamalla numeron merkkijonoksi voit kirjoittaa numeroita vierekkäin ilman tyhjiä välejä.

Kun sanomme, että käskyt VAL ja STR$ muuntavat merkkijonoja numeroiksi ja päinvastoin, emme itse asiassa tee alkuperäiselle muuttujalle yhtään mitään, onpa se sitten numeerinen tai merkkimuuttuja. Jos muodostamme uuden numeerisen muuttujan nimeltä EE merkkijonosta EE$, jää tuo alkuperäinen merkkimuuttuja (EE$) edelleen voimaan. Sama sääntö pätee myös käskyyn STR$.

ASC ja CHR$

Kuten jo aiemmin sanottiin, ASC merkitsee ASCII, joka on nimi sille numerokoodille, jonka mukaan kullekin Commodoren tietokoneiden käyttämälle merkille annetaan oma numeronsa. Kun käskyä ASC käytetään sulku- ja lainausmerkeissä olevan merkin kanssa, se antaa tuolle merkille varatun numeron.

```
PRINT ASC(''5'')
```

Kuvaruudussa näkemäsi vastaus *ei* ole 5 vaan 53. Mitä on tapahtunut?

Numero 53 on ASCII-koodinumero, joka merkitsee numerolla 53 tarkoitettua merkkiä. ASCII on lyhennys sanoista American Standard Code for Information Interchange (amerikkalainen standardikoodi tiedonvaihtoon). Lähes kaikki nykyaikaiset tietokoneet käyttävät ASCIIta merkkien merkitsemiseen voidakseen välittää tietoa toisilleen puhelinlinjaa myöten.

Commodoren tietokoneet eivät käytä *todellisia* ASCII-koodeja. Niiden on käytettävä omaa ASCII-versiotaan, koska tietokoneen käyttämien merkkien määrä — mukaan luettuina

graafiset merkit, pienet kirjaimet, kohdistin- ja editointikäskyt, värikoodit jne. — on paljon suurempi kuin ne 128 merkkiä, jotka alunperin muodostivat todellisen ASCII-koodin. Aina ensimmäisestä Commodore PETistä alkaen Commodoren tietokoneet ovat käyttäneet lempinimen PET ASCII saanutta ASCII-versiota. Kuinka paljon PET ASCII sitten eroaa tavallisesta ASCIIsta? Eräät merkit ovat samoja ja toiset taas erilaisia, ja koska PET ASCIIssa merkkejä on 256, on siis 160 sellaista numeroa, jotka eivät sovi mihinkään todellisiin ASCII-koodinumeroihin.

Tämä ei tuota kovin suuria ongelmia, kunhan vain muistat, että on olemassa PET ASCII, joka täytyy ottaa huomioon silloin, kun haluat viestiä koneesi välityksellä muiden tietokoneiden tai kirjoittimen kanssa.

Toistaiseksi riittää se, että muistat ASC-käskyn antamien numeroiden olevan PET ASCIIn numeroita.

ASC antaa aina merkin numeron tai merkkijonon ensimmäisen merkin numeron. Esimerkiksi:

```
PRINT ASC(''52370 SUURLAHTI'')
```

Tämä esimerkiksi antaa siis tulokseksi 53, koska merkkijonon ensimmäinen merkki on 5.

ASC-käsky ei hyväksy tyhjiä merkkijonoja. Jos annat käskyn ASC('''') tai ASC(A$) silloin, kun A$ = '''', ohjelman suoritus pysähtyy ja näet virheilmoituksen ILLEGAL QUANTITY ERROR.

Käskyä CHR$ voidaan pitää ASC-käskyn vastakkaiskäskynä. Se kirjoittaa tiettyä PET ASCIIn numeroa vastaavan merkin.

```
PRINT CHR$(53)
```

Tämä saattaa vaikuttaa hiukan rasittavalta kirjoitustavalta. Miksemme voisi kirjoittaa yksinkertaisesti PRINT 5? Eräitä merkkejä on kuitenkin vaikea tai jopa mahdoton kirjoittaa ruu-

tuun käskyllä PRINT. Yritä kirjoittaa ruutuun lainausmerkki.

```
10 A$ = " " "
20 PRINT A$
```

Ei onnistu. Koetapa tätä:

```
10 A$ = CHR$(34)
20 PRINT A$
```

Nyt ruutuun ilmestyy lainausmerkki, koska 34 on juuri sitä merkkiä vastaava koodinumero. Käskyllä PRINT et voi aikaansaada rivinsiirtoa (jonka voit tehdä painamalla näppäintä RETURN). Sen sijaan *voit* kyllä kirjoittaa PRINT CHR$(13).

Tästä on apua silloin, kun esimerkiksi annat käskyn PRINT# kirjoittimelle. Kirjoittimet tunnistavat muitakin tällaisia koodeja ja suorittavat tiettyjä erityistoimintoja, kun lähetät asianmukaisen koodin kirjoittimelle.

Seuraavassa eräitä kiinnostavimpia Commodore 64:ään sisältyviä CHR$-koodeja.

PRINT CHR$( )	TOIMINTA
8	Estää näppäinten SHIFT ja COMMODORE painamisen yhtä aikaa, joten grafiikkatilasta ei ole mahdollista vaihtaa kirjoituskonetilaan.
9	Palauttaa em. näppäimet toimimaan.
13	Suorittaa rivinsiirron, kuten myös RETURN-näppäimen painaminen; siirtää kohdistimen seuraavalle riville.
14	Vaihtaa kirjoituskonetilasta grafiikkatilaan.
34	Kirjoittaa lainausmerkin.
142	Vaihtaa grafiikkatilasta kirjoituskonetilaan.

Muut CHR$-koodit liikuttavat kohdistinta ja muuttavat merk-

kien väriä. Kirjan lopussa on täydellinen luettelo näistä koodinumeroista. Tarkista myös kirjoittimesi käsikirjasta, mitkä CHR$-koodit ohjaavat sen toimintaa.

LEFT$, MID$ ja RIGHT$

Edellä esitettyjen hyvin monipuolisten käskyjen ymmärtäminen ei oikein onnistu muuten kuin istuutumalla koneen ääreen ja kokeilemalla. Kutakin niistä käytetään hajottamaan merkkijonoja ja ottamaan niistä erilleen haluttua tietoa. LEFT$-käskyllä luetaan merkkijonon vasemmanpuolisia merkkejä, RIGHT$ taas lukee merkkejä oikealta. MID$ hakee merkkejä jonon keskeltä.

Seuraavassa tarvitaan myös käskyä LEN:

```
10 A$ = ''1234567890''
20 L = LEN(A$)
30 FOR I = 1 TO L
40 PRINT LEFT$(A$,I)
50 NEXT I
```

Jos suoritat tämän esimerkkiohjelman, sinun pitäisi nähdä seuraavaa:

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
1234567890
```

Tämä onnistuu siten, että LEFT$ antaa tulokseksi vasemmasta laidasta lähtien sulkeissa olevan numeron ilmaiseman

määrän merkkejä. Käytimme käskyä LEN määrittämään FOR/NEXT-silmukan merkkien määrän, jottemme yrittäisi kirjoittaa ruutuun useampia merkkejä kuin jonossa on. Kirjoita nyt sama ohjelma, mutta käytä käskyn LEFT$ asemesta tällä kertaa käskyä RIGHT$. Suorita taas ohjelma. Näet äskeisen tuloksen käännettynä.

```
0
90
890
7890
67890
567890
4567890
34567890
234567890
1234567890
```

Tämä taas johtuu siitä, että RIGHT$ antaa tulokseksi sulkeissa olevan numeron ilmaiseman määrän merkkejä oikeasta laidasta.

Käskyn MID$ käyttäminen saattaa olla hankalaakin, ellet seuraa tarkasti, mitä olet tekemässä. Nyt sinun on kirjoitettava sulkeisiin kaksi lukua yhden sijasta.

```
10 A$ = ''1234567890''
20 PRINT MID$(A$,5,2)
```

Ensimmäinen numero kertoo, kuinka monenteen paikkaan merkkijonossa on mentävä. Toinen numero taas kertoo, kuinka monta merkkiä on tuosta kohdasta alkaen kirjoitettava ruutuun. Edellinen esimerkki kirjoittaa ruutuun 56. Tarkemman käsityksen tämän käskyn käyttämisestä saa kirjoittamalla seuraavan esimerkin:

```
10 A$ = ''123456789''
20 L = LEN(A$)
```

```
30 FOR I=1 TO L
40 FOR K=1 TO L
50 PRINT MID$(A$,I,K)
60 NEXT K
70 NEXT I
```

Jotta ymmärtäisit paremmin, mitä on tekeillä, paina CTRL-näppäintä hidastaaksesi ohjelman suoritusta. Näet käskyn MIDS tehon.

Käskyjä LEFT$, MID$ ja RIGHT$ käytetään Commodoren reaaliaikakellon esittämiseen. Tässä on lyhyt ohjelma, joka sijoittaa kellon kuvaruudun vasempaan yläkulmaan. Se hajottaa muuttujan TI$ ja järjestää sen uudelleen siten, että se näyttää digitaalikellon näytöltä. Muuttamalla rivinumerot ja käyttämällä tätä aliohjelmana voit kirjoittaa ohjelmia, jotka näyttävät aikaa silloin kun ohjelmia suoritetaan.

```
10 INPUT ''TUNNIT'';T$
20 INPUT ''MINUUTIT'';M$
30 TI$=T$+M$+''00''
40 PRINT ''[paina CLR]''
50 PRINT ''[paina HOME]'';
60 PRINT ''KELLO'';
70 PRINT LEFT$(TI$,2);'':'';MID$(TI$,2,2);'':'';
   RIGHT$(TI$,2)
80 GOTO 50
```

Sinun on kirjoitettava tunnit, minuutit ja sekunnit kaksinumeroisina. Jos siis kello on 1.03, kirjoita kysymykseen TUNNIT vastaukseksi 01 ja kysymykseen MINUUTIT 03. Ohjelma olettaa sekuntien alkavan 00:sta. Se ottaa nämä kaksi merkkijonoa, T$ ja M$, ja lisää niihin ''00'' sekä sijoittaa sen jälkeen tuon merkkijonon TI$:n arvoksi asettaen kellon aikaan. Ohjelman pääsilmukassa riveillä 50—80 tapahtuu seuraavaa: kohdistin lähetetään vasempaan yläkulmaan asemaan HOME, se jää sille riville ja ryhtyy kirjoittamaan kuvaruutuun. Se kirjoittaa järjes-

tyksessä sanan KELLO, tyhjän välin, kaksi vasenta merkkiä muuttujasta TI$ (eli tunnit), kaksoispisteen, muuttujan TI$ kaksi keskimmäistä merkkiä (siis minuutit), toisen kaksoispisteen ja lopuksi muuttujan TI$ kaksi oikeanpuolimmaista merkkiä (eli sekunnit).

Kaikki tämä alkaa uudelleen alusta, kun ohjelma palaa riville 50 ja kirjoittaa uuden ajan edellisen päälle. Näet siis sekuntien vaihtuvan ja ohjelma näyttää sujuvan rauhaisaa tahtia. Tietokoneen sisällä kaikki tapahtuu kuitenkin raivoisalla vauhdilla. Se kirjoittaa riviä yhä uudelleen ja uudelleen, vaikka kellonaika ei olisi muuttunut edes sekunnilla. Jos haluat tämän aliohjelmaksi johonkin omaan ohjelmaasi, se on parasta sijoittaa kauas varsinaisesta pääohjelmasta.

(Ota siinä tapauksessa pois GOTO-lause ja kirjoita sen paikalle RETURN sekä kirjoita kellonajan asettavat rivit oman ohjelmasi alkuun. Aina kun haluat nähdä kellonajan kuvaruudussa, sinun on kuitenkin mentävä näille riveille käskyllä GOSUB. Silloin ei ehkä ole kovin käytännöllistä näyttää sekunteja.)

Merkkijonojen käsitteleminen on ajatuksena helppo ymmärtää, mutta kokeneetkaan ohjelmoijat eivät aina hallitse sitä. Merkkijonojen käsittelemiseen tarvitaan harjoittelua, kärsivällisyyttä ja selkeää ajattelukykyä.

7

Värit, grafiikka, ääni ja pelit

Jotkut kotitietokoneiden omistajat eivät koskaan käytä hyväkseen koneittensa väri-, grafiikka- ja ääniominaisuuksia. Toisille ne taas ovat kaikkein tärkeimmät tietokoneen mahdollisuudet. On niitä, joiden mielestä tällaiset ”hörhelöt” eivät kuulu vakavaan ohjelmointiin — matematiikkaan, tekstinkäsittelyyn, tietokantojen käyttöön jne. He kehottavat käyttämään niitä videopeleissä.

Tietenkin kehittyneet grafiikka- ja ääniominaisuudet tekevät koneen peleille sopivaksi, ja ne antavat muitakin hienoja mahdollisuuksia. Grafiikka on muodostumassa yhä tärkeämmäksi, ja tietokone antaa uusia mahdollisuuksia tiedon välittämiseen näköaistin avulla (kuten nykyisin jo monista mainoksistakin voimme havaita). Uudet kehittyneet tietokoneet, kuten Xerox Star tai Apple Lisa, ovat myös avanneet tietä grafiikan käytölle.

Se että tietokone osaa tuottaa videopelin ääniefektejä, ei silti välttämättä tarkoita, että se ei pysty *mihinkään* muuhun. Tietokoneet voivat olla musikaalisia tai toistaa ihmisääntä hämmästyttävän tarkasti.

Commodore 64 -laitteisto on parhaimmillaan juuri grafiikka- ja ääniominaisuuksia tarvittaessa. Ikävä kyllä noiden mahdolli-

suuksien käyttö voi olla monimutkaista ja aikaavievää. Se johtuu siitä, että ainoa käyttökelpoinen yhteytemme tietokoneeseen on BASIC-kieli. Kuvien ja äänen aikaansaamista BASICillä voidaan verrata auringonlaskun tai sinfonian kuvailemiseen vain muutamalla sanalla.

Tässä luvussa tutustutaan Commodore 64 -laitteiden väri-, grafiikka- ja ääniominaisuuksiin. Niitä ei käsitellä tyhjentävästi, vaan ruokahalusi herättämiseksi. Kirjan lopussa on lisää tietoa sekä grafiikan että äänen käyttämisestä, mutta tuon lisätiedon ymmärtäminen edellyttää koko kirjan hallitsemista. Et sitä paitsi opi noista loppuluvuistakaan kaikkea mahdollista, sillä sekä grafiikka että ääni vaatisivat oman kirjansa.

Tässä luvussa opit käyttämään muutamaa grafiikan peruskeinoa. Opit myös, kuinka tietokone saadaan soittamaan helppoja sävelmiä. Jos pelit kiinnostavat sinua, saat tietää satunnaislukugeneraattorista, joka osaa heittää noppaa tai pyörittää onnenpyörää. Saat myös tietää jotakin kontrolliporteista, joihin liitetään joystick- ja paddle-peliohjaimet.

VIC ja SID

VIC ja SID kuulostavat nimiltä, mutta oikeastaan VIC tarkoittaa kuvaruutuliitännän ohjainta ja SID ääniliitäntälaitetta. Ne ovat kumpikin hyvin tehokkaita mikropiirejä, jotka ohjaavat kaikkea minkä näet ruudusta tai kuulet kovaäänisestä.

VIC antaa mahdollisuuden käyttää monenlaista grafiikkaa. Ensiksikin on *merkkitila,* joka on toiminnassa aina kun koneeseen kytketään virta. Se on helppokäyttöisin. Käskyllä PRINT kirjoitetaan merkkejä ruutuun. Vaikka tämä tila on mahdollisuuksiltaan kaikkein rajallisin, nokkelat ohjelmoijat keksivät jatkuvasti uusia mahdollisuuksia muodostaa kauniita videokuvia tässäkin tilassa.

Eräs tapa parantaa kuvaruudun esitysmuotoa merkkitilassa on tietokoneen merkkien *uudelleen määrittäminen.* VIC-piirin avulla voidaan muuttaa vakioaakkoset, numerot ja muut mer-

kit itse suunnitelluiksi merkeiksi. Uusina merkkeinä voidaan tietokoneeseen ohjelmoida matemaattisia ja tieteellisiä erityismerkkejä, vieraiden kielten kirjaimia tai jopa nuotteja.

Merkkitilassa kuvaruudussa on 1000 yksittäistä paikkaa (40 merkkiä 25 rivillä), joista jokaista edustaa 1 *tavu* muistissa. Kuvaruutua varten tarvitaan siis yhteensä 1000 tavua. Koska kuvaruudussa voidaan esittää jopa 16 väriä yhtä aikaa, tämä lisäväritieto on varastoitu lisämuistiin. Jokaisen merkin värimääritystä varten on varattu puolikas tavua eli neljä *bittiä*. Kuvaruudun värejä varten tarvitaan siis vielä 500 tavua.

Toista grafiikkatilaa, ns. tarkkuusgrafiikkaa, tarvitaan, kun merkkitilan rajat täytyy ylittää. Tarkkuusgrafiikkatilassa koko kuvaruutu koostuu pisteistä eli *pikseleistä*. (Nyt ei voida käyttää käskyä PRINT.) Kutakin pistettä varten on varattu yksi bitti muistitilaa. Tarkkuusgrafiikkatilassa kuvaruudun koko on 320 pistettä vaakasuoraan ja 200 pistettä pystysuoraan. Se vastaa 8000 bittiä muistitilaa.

Tässä tilassa voi käyttää kahta väriä, toista taustassa ja toista pisteissä. Tämän tilan avulla on mahdollista määritellä eri *alueita* kuvaruutuun siten, että kukin alue koostuu merkkien kokoisista nelikulmioista. Kullekin alueelle voidaan asettaa eriväri-nen tausta ja pisteväri tarkkuusgrafiikkatilassa, mutta kullakin alueella voidaan käyttää ainoastaan kahta väriä.

Voit kaksinkertaistaa kuvaruudussa olevien värien määrän käyttämällä *moniväritilaa*. Siinä voidaan samanaikaisesti käyttää neljää väriä, mutta kuvaruudun erotuskyky pienenee. Moniväritilassa on kuvaruudussa 160 pistettä vaakasuoraan ja 200 pystysuoraan. (Tässäkään tilassa ei voida käyttää käskyä PRINT.)

Missä tahansa grafiikkatilassa voidaan käyttää *spritejä*. Ne ovat pieniä tarkkuusgrafiikalla tehtyjä kuvia, jotka on varastoitu muistissa moniin paikkoihin. VIC-piiri pystyy tunnistamaan nämä kuvat ja piirtämään ne ruutuun, jossa niitä voidaan liikutella BASICin avulla. Sprite koostuu peräti 540 pisteestä — 24 vaakasuoraan ja 21 pystysuoraan — eli se vie 63 tavua muistia. Se voi olla minkävärinen tahansa ja sitä voidaan suurentaa joko

korkeus- tai leveyssuunnassa. Yksivärispriten lisäksi on olemassa myös monivärisprite. Moniväritarkkuusgrafiikan tavoin monivärispritekin voi olla nelivärinen, mukaan luettuna taustan väri. Monivärispritet ovat vaikeampia käyttää ja ohjelmoida kuin yksiväriset spritet.

Spriteistä, merkkien uudelleen määrittämisestä ja tarkkuusgrafiikasta on lisätietoja kirjan liitteissä. Tässä luvussa puhutaan merkkigrafiikasta ja siitä, kuinka spritejä voidaan tehdä ja liikutella ruudussa.

VIC ei ole ainoa piiri, joka tarjoaa luovia mahdollisuuksia. VICin avulla voidaan luoda visuaalisia nautintoja, SID taas tarjoaa kuulonautintoja.

SID voi tuottaa samanaikaisesti kolme erilaista *ääntä.* Pääasiallisin ero 64 -laitteiston ja sen edeltäjän VIC-20:n ääniominaisuuksissa onkin nimenomaan äänen laatu. (VIC-20:ssä ei ole omaa SID-piiriä.)

Tuotettaessa musiikkia elektronisesti ovat äänen osatekijät englanniksi *attack, decay, sustain* ja *release,* joista muodostuu lyhenne ADSR. Juuri nämä ominaisuudet tekevät pianon äänestä erilaisen kuin banjon ääni ja urkujen äänestä erilaisen kuin rumpujen ääni. Ne mitataan millisekunteina — pieninä sekunnin murto-osina. Kullakin SIDin äänellä on oma ADSR-asemansa.

Attack (nousuaika) tarkoittaa sitä nopeutta, jolla ääni alkaa. Esimerkiksi banjolla tai jollain muulla näppäilysoittimella on lyhyt ja terävä attack.

Decay (tasaantumisaika) on voimakkuuden lasku. Piano tuottaa aluksi voimakkaan äänen, mutta se hiljenee asteittain ''soivalle'' tasolle.

Sustain (jälkisointi) on se aika, jonka ääni kuuluu. Aika jonka ääni tarvitsee hiljetäkseen kokonaan, on release (laskuaika).

Nämä ovat yksinkertaisia käsitteitä, mutta niitä on usein vaikea ymmärtää, ja niistä on enemmän kirjan lopussa olevassa liitteessä. Tässä luvussa saat kuitenkin tietää, millä tavalla SID tuottaa äänen ja kuinka tietokone pystyy soittamaan nuotteja ja helppoja sävelmiä.

ADSR:n asettamisen tulosta kutsutaan *verhoksi*. Eri ADSR-arvojen lisäksi SID voi antaa kullekin äänelle erilaisen peruslähtökohdan eli *aaltomuodon*. Aaltomuotoa voidaan tavallaan pitää kankaana, josta paita on leikattu, ja verhoa taas itse paidan lopullisena muotona. Ääntä voidaan sitten muokata enemmän käyttämällä erilaisia *suotimia*. Jos tunnet elektronisia musiikkisyntetisaattoreita, sinua kiinnostanee tieto, että SID-piirissä on *sointimodulaattori* ja voit jopa ''miksata'' todellista ääntä nauhurista, mikrofonista tai jostain soittimesta SIDin ääneen. (Tietokoneen takana olevassa Audio/Video-liitännässä on tätä varten liittimet, piikit 5 ja 2. Älä kuitenkaan yritä liittää niihin mitään, ellet tiedä mitä olet tekemässä.)

SIDissä ja VICissä on myös muita ominaisuuksia. VICin sisällä on valokynän vaatimat muistipaikat ja SIDissä taas pelisäätimien (nimenomaan paddlepeliohjainten) asemaa lukevat muistipaikat.

BASICissä ei ole sellaisia sanoja, jotka vaikuttaisivat suoraan VICiin ja SIDiin. Voimme kuitenkin viestiä suoraan näiden piirien kanssa käyttämällä käskyä POKE sijoittaaksemme numeroita niiden sisällä oleviin muistipaikkoihin. VICissä on 24 muistipaikkaa numeroarvoiltaan 53248—53271. SIDissä on 25 musiikin tekemiseen tarvittavaa muistipaikkaa numeroarvoiltaan 53272—53296.

Satunnaisia RND-huomautuksia

Ennen kuin jatkamme, tarkastelemme BASICin sanaa, joka esiintyy kahdessa tässä luvussa esitetyssä ohjelmassa. Tuo sana on RND, ja se tuottaa satunnaislukuja kaikissa Commodoren tietokoneissa VICistä 64-laitteistoon.

Satunnaisluvut ovat pelien olennainen osa. Aina kun heität noppaa, pyörität onnenpyörää tai sekoitat kortteja, tarvitset satunnaislukuja. Tietokone tuottaa niitä *satunnaislukugeneraattoriksi* kutsutun matemaattisen kaavan avulla. Nämä luvut eivät oikeastaan ole täysin satunnaisia, mutta kaava on niin ove-

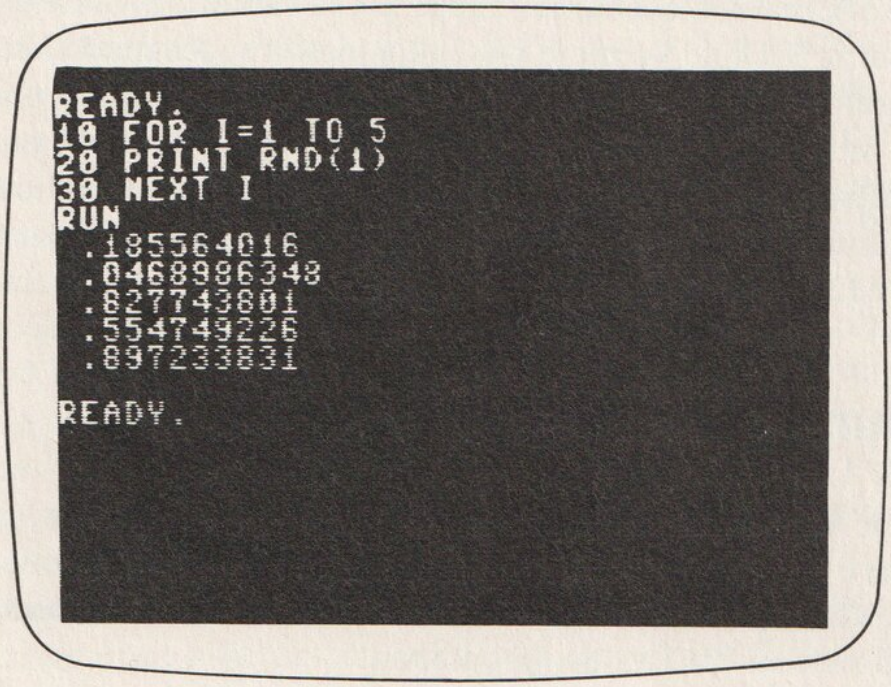
la, että et huomaa mitään järjestelmällisyyttä.

Käskyä RND käytetään ''siemenluvun'' kanssa. Tämä luku määrittää, millä tavalla satunnaisluvut valitaan. Voit käyttää erilaisia siemenlukuja, jotka kirjoitetaan sulkeisiin RND:n jälkeen. Katkaise ja kytke virta ja kirjoita tämä esimerkki:

```
10 FOR I=1 TO 5
20 PRINT RND(1)
30 NEXT I
```

Näet ruudussa viisi lukua.

```
.185564016
.0468986348
.827743801
.554749226
.897233831
```



Et ehkä näe *juuri näitä* lukuja, mutta aina kun kytket koneeseesi virran ja suoritat tämän ohjelman, näet *samat* luvut. Saatat ihmetellä syytä, jos ne kerran ovat *satunnaislukuja?* Hyvä kysymys.

Satunnaislukugeneraattori toimii siten, että aina kun koneeseen kytketään virta, siemenluku valiutuu automaattisesti. Tämä siemenluku puolestaan saa aikaan numerolistan. Aina kun RND-funktiota käytetään siten, että sulkeissa on *mikä tahansa* positiivinen luku, kone antaa tulokseksi listasta seuraavan luvun. Aina kun RND-funktiota käytetään eri negatiivisen luvun kanssa, valiutuu uusi lukulista. Jos funktiota RND käytetään 0:n (nollan) kanssa, saadaan TI-kelloon perustuva satunnaisluku. Esimerkki:

```
10 X = RND(-1)
20 FOR I = 1 TO 10
30 PRINT RND(1)
40 NEXT I
```

Aina kun suoritat tämän ohjelman, näet saman RND(1):n aikaansaaman lukusarjan. Jos muutat ensimmäisen *negatiivisen* siemenluvun, näet erilaisen listan, joka kuitenkin on aina sama. Kokeile ja muuta RND(-1):n paikalle RND(-2).

Mikä sitten on paras tapa kehittää satunnaislukuja? Yksi tapa on valita negatiivinen siemenluku satunnaisesti käyttämällä TI-kelloa.

```
10 N = TI
20 X = RND(-N)
30 FOR I = 1 TO 10
40 PRINT RND(1)
```

Kun käskyä RND käytetään tällä tavalla, tulee siemenluvuksi eri numero (negatiivinen TI). Aina ohjelmaa suoritettaessa se tuottaa erilaisen listan.

Voit myös käyttää funktiota RND(0), joka sekin kehittää satunnaislukuja TI-kellon pohjalta. Kummankin tavan etu on siinä, ettet voi koskaan ennalta tietää, mikä arvo TI:llä on 0:n ja 51839999:n väliltä. Ja vaikka voisitkin, olisi lähes mahdotonta

odottaa tiettyä siemenlukua, koska TI:n arvo muuttuu joka 1/60-sekunti.

Olet todennäköisesti jo huomannut, että RND:n tuottamia lukuja ei voida oikein hyvin käyttää peleissä. Saadaksemme desimaaliluvuista kokonaislukuja meidän on käytettävä matemaattista funktiota INT. Saadaksemme tietyn numeroalueen käyttöön kerromme RND-luvun. Seuraava esimerkki toimii mallina lähes kaikkiin satunnaislukugeneraattoreihin.

Oletetaan, että haluamme satunnaislukuja 1:n ja 10:n väliltä. Se on helppoa.

```
10 N = INT(10*RND(0) + 1)
20 PRINT N
30 GOTO 10
```

Kun suoritat tämän esimerkkiohjelman, näet tasaisen lukujoukon väliltä 1—10. Luku jolla kerrot RND:n, asettaa aina lukualueen. Joudumme lisäämään ykkösen siksi, että RND antaa satunnaislukuja nollasta ykköseen siten, että nolla on mahdollinen mutta ykkönen ei ole. INT käskee tietokoneen käyttää vain kokonaislukuja ja poistaa kaikki desimaalit.

Kun käytät RND-funktiota, varmistu siitä, että olet kirjoittanut oikean määrän sulkumerkkejä. Jos niitä on yksikin liikaa tai liian vähän, tietokone antaa virheilmoituksen SYNTAX ERROR. Muista, että sulkumerkkejä on oltava parillinen määrä, ja tarkista, että ne osoittavat oikeaan suuntaan.

Tässä on lyhyt esimerkki funktion RND toiminnasta korttipelissä. Asetamme taulukon, johon sopii korttipakan 52 korttia. Ohjelmassa voit ”nostaa” kortteja.

```
10 PRINT ''[CLR]''
20 DIM KO$(52)
30 FOR I=1 TO 13:READ KO$(I):KO$(I)=''HERTTA''+
   KO$(I)
40 NEXT I
50 RESTORE
60 FOR I=14 TO 26:READ KO$(I):KO$(I)=''PATA''+
   KO$(I)
70 NEXT I
80 RESTORE
90 FOR I=27 TO 39:READ KO$(I):KO$(I)=''RISTI''+
   KO$(I)
100 NEXT I
110 RESTORE
120 FOR I=40 TO 52:READ KO$(I):KO$(I)=''RUUTU''
    +KO$(I)
130 NEXT I
150 PRINT ''NOSTA KORTTI PAINAMALLA JOTAIN
    NAPPAINTA''
160 KO=INT(52*RND(0)+1)
170 GET C$:IF C$='' '' THEN 170
180 PRINT KO$(KO)
190 GOTO 150
200 DATA ASSA, KUNINGAS,ROUVA,SOTAMIES,10,9,
    8,7,6,5,4,3,2
```

Tämä ohjelma ei sekoita kortteja. Pakka on samanlainen kuin avatessasi uuden pakan, jossa kaikki kortit ovat järjestyksessä. RND poimii satunnaisluvun väliltä 1—52 ja näyttää, mitä korttia tuo satunnaisluku edustaa. Huomaat myös, kuinka käsky RESTORE toimii. Tarvitaan vain yksi DATA-rivi, joka luetaan neljä kertaa. Joka kerta lisätään kortin arvoon maa — hertta, pata, risti ja ruutu.

Ruudun ja rajauksen värit

Kaksi VIC-piirin muistipaikkaa ohjaa kuvaruudun ja sen rajauksen värejä. Ne asetetaan sijoittamalla POKE-käskyä käyttäen jokin luvuista 0—15 näihin muistipaikkoihin. POKE 53280 asettaa *rajauksen* värin ja POKE 53281 asettaa *kuvaruudun* värin.

```
10 POKE 53280,11:POKE 53281,15
```

Tämä rivi tekee kuvaruudusta harmaan numero 3, joka on kaikkein vaalein harmaan sävy (lähes valkoinen), ja rajauksesta harmaan 1, kaikkein tummimman harmaan. Kullakin Commodore 64:n värillä on oma numeronsa.

POKE 53280 53281	VÄRI
0	Musta
1	Valkoinen
2	Punainen
3	Puuterinsininen
4	Sinipunainen
5	Vihreä
6	Sininen
7	Keltainen
8	Oranssi
9	Ruskea
10	Vaaleanpunainen
11	Harmaa 1 (tummanharmaa)
12	Harmaa 2 (keskiharmaa)
13	Vaaleanvihreä
14	Vaaleansininen
15	Harmaa 3 (vaaleanharmaa)

Eräs tapa muistaa nämä värit ja muistipaikat on määritellä muuttujia ja käyttää niitä POKE-numeroiden asemesta.

```
10 RAJAUS = 53280:RUUTU = 53281
20 REM SEURAAVAKSI VARIT
30 MU = 0:VA = 1:PU = 2:PS = 3:SP = 4:VI = 5:SI = 6:KE = 7
40 OS = 8:RU = 9:VP = 10:H1 = 11:H2 = 12:VV = 13:VS = 14:
   H3 = 15
50 POKE RUUTU,PU:REM RUUTU PUNAINEN
60 POKE RAJAUS,VI:REM RAJAUS VIHREA
```

Tämä on hyvä tapa muistaa VICin ja SIDin muistipaikat. Kannattaa kuitenkin olla huolellinen etteivät lyhenteet ole samoja tai etteivät ne ole BASICin varattuja sanoja (esimerkiksi oranssia merkitään lyhenteellä OS, ei OR.)

Merkkien väri

Tiedät jo, että kun painat CTRL-näppäintä ja jotakin numeronäppäimistä 0—9, kuvaruutuun kirjoittamiesi merkkien väri muuttuu. Painamalla COMMODORE-näppäintä ja jotakin numeronäppäintä saat valittavaksesi vielä kahdeksan väriä lisää. CTRL-näppäimen avulla saatavien värien englanninkieliset lyhenteet on painettu kunkin näppäimen etuosaan, mutta COMMODORE-näppäimen avulla saatavia värejä ei ole näkyvissä.

Merkkien värin muuttaminen on oikeastaan välttämätöntä, jos muutat kuvaruudun väriä, koska jotkin värit ovat käytännössä liian himmeitä näkyäkseen tietynväristä taustaa vasten. Mutta itse kirjoittamiesi merkkien värin muuttaminen ei muuta ohjelmassa PRINT-käskyllä kirjoitettujen merkkien väriä.

Editointinäppäinten — CSRS ylös, alas, oikealle ja vasemmalle, INST/DEL ja CLR/HOME — tavoin värinäppäimiäkin voidaan käyttää PRINT-käskyn yhteydessä. Käskyn PRINT, lainausmerkkien ja värinäppäimen kirjoittaminen ei muuta sillä hetkellä kirjoittamiesi merkkien väriä, vaan siitä jää käskymerkki, joka muuttaa värejä, kun ohjelmaa suoritetaan. Käskymerkki on käänteisenä taustaa vasten näkyvä symboli.

Kokeilepa tätä: (Paina hakasulkeissa olevia näppäimiä, älä kirjoita kyseisiä sanoja.)

```
05 POKE 53281, 11:POKE 53280, 15
10 PRINT ''[CTRL ja BLK]MUSTA [CTRL ja WHT] VALKOINEN''
20 PRINT ''[CTRL ja RED]PUNAINEN [CTRL ja CYN] PUUTERINSININEN''
30 PRINT ''[CTRL ja PUR]SINIPUNAINEN [CTRL ja GRN] VIHREA''
40 PRINT ''[CTRL ja BLU]SININEN [CTRL ja YEL] KELTAINEN''
50 PRINT ''[COMM ja BLK]ORANSSI [COMM ja WHT] RUSKEA''
60 PRINT ''[COMM ja RED]VAALPUNAINEN [COMM ja CYN] HARMAA 1''
70 PRINT ''[COMM ja PUR]HARMAA 2 [COMM ja GRN] VAALVIHREA''
80 PRINT ''[COMM ja BLU]VAALSININEN [COMM ja YEL] HARMAA 3''
```

Kun olet kirjoittanut ensimmäisen lainausmerkin, mikä tahansa CTRL- tai COMMODORE-näppäimen ja värinäppäimen yhdistelmä tuottaa tulokseksi käskymerkin. Mutta jos teet virheen ja siirryt pois tästä ''lainausmerkkitilasta'' (katso lukua Perustaidot), alkaa itse kirjoittamiesi merkkien väri muuttua. Voidaksesi jättää värikäskymerkin sinun on päästävä takaisin lainausmerkkitilaan; sijoita kohdistin takaisin paikalleen ja kirjoita yksi lainausmerkki.

Kun suoritat edellisen esimerkkiohjelman, näet jokaisen värin nimen omalla värillään ruutuun kirjoitettuna. Yksi väri, harmaa 3, ei näy ruudussa, koska harmaa 3 on taustan väri. Kokeile muuttamalla POKE 53281:n numeroarvoja nähdäksesi, mikä merkkien väri sopii hyvin kuvaruudun väreihin.

Ruutuun kirjoittaminen jatkuu aina valitulla värillä, kunnes ohjelma käskee toisin. Jos haluat korostaa jotakin sanaa teks-

tissä, sinun on muutettava väriä kahdesti — ensiksi alkuperäisestä väristä korostettuun väriin ja sitten takaisin. Esimerkki:

```
10 PRINT ''[CTRL BLK] SEURAAVA [CTRL RED] PU-
   NAINEN [CTRL BLK] JA TAKAISIN MUSTAAN''
```

Grafiikan piirtäminen

Lähes jokaisen näppäimen etuosassa on graafiset merkit. Näitä merkkejä yhdistelemällä saadaan rajauksia, laatikoita ja, jos mielikuvitusta riitää, jopa kuvia. Näiden merkkien avulla tehtävällä grafiikalla on lukuisia nimiä, mutta me puhumme tässä merkkigrafiikasta.

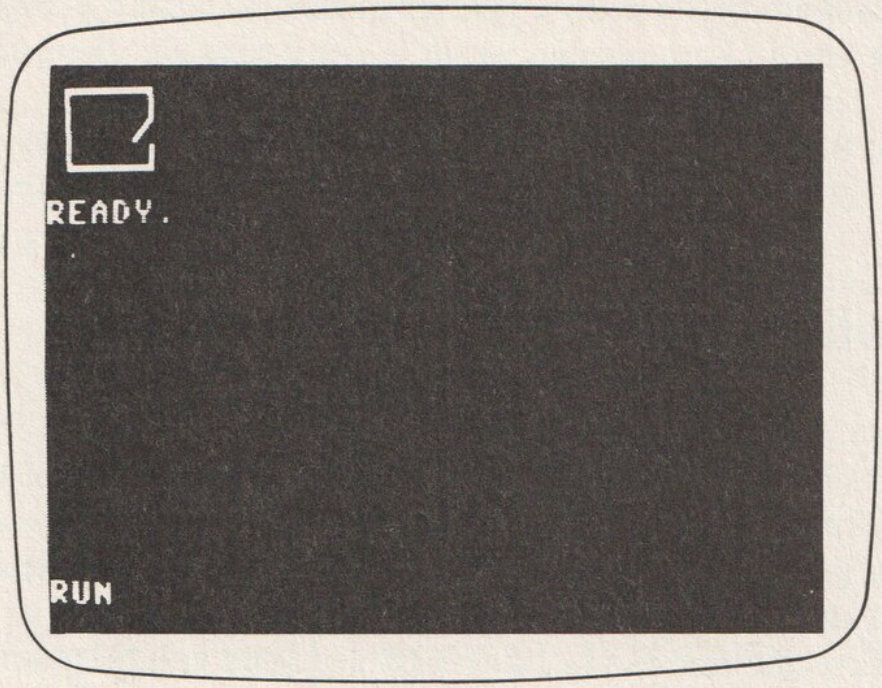
Piirretään ruutuun laatikko käyttäen näitä graafisia merkkejä. Muista taas painaa hakasulkeissa olevia näppäimiä.

```
10 PRINT ''[CLR] [CRSR ALAS]'';
20 PRINT ''[VÄLILYÖNTINÄPPÄIN] [SHIFT O]
   [COMM Y] [COMM Y] [SHIFT P]''
30 PRINT ''[VÄLILYÖNTINÄPPÄIN] [COMM H] [VÄLI-
   LYÖNTINÄPPÄIN] [VÄLILYÖNTINÄPPÄIN] [SHIFT
   N]''
40 PRINT ''[VÄLILYÖNTINÄPPÄIN] [SHIFT L] [COMM
   P] [COMM P] [SHIFT@]''
```

Kun suoritat tämän ohjelman, pitäisi ruudun vasempaan yläkulmaan ilmestyä nelikulmainen laatikko. Jos rivillä 20 tai 40 ei olisi ensin painettu välilyöntinäppäintä tai jos CRSR alas puuttuisi riviltä 10, laatikon reunat sekoittuisivat taustan väriin. Käytä CRSR-näppäintä sijoittaaksesi laatikon toiseen paikkaan kuvaruudussa. Muuta rivi 10:

```
10 PRINT ''[CLR] [5 CRSR ALAS]'';
```

Sen sijaan että painaisit CRSR alas -näppäintä vain kerran, pai-



na sitä viisi kertaa. Laatikko piirtyy nyt ruudussa alemmaksi. Yritä lisätä tyhjiä välejä riveille 20 ja 40 ensimmäisen lainausmerkin jälkeen. Jos olet lisännyt kumpaankin kohtaan yhtä monta tyhjää väliä, koko laatikko siirtyy oikealle.

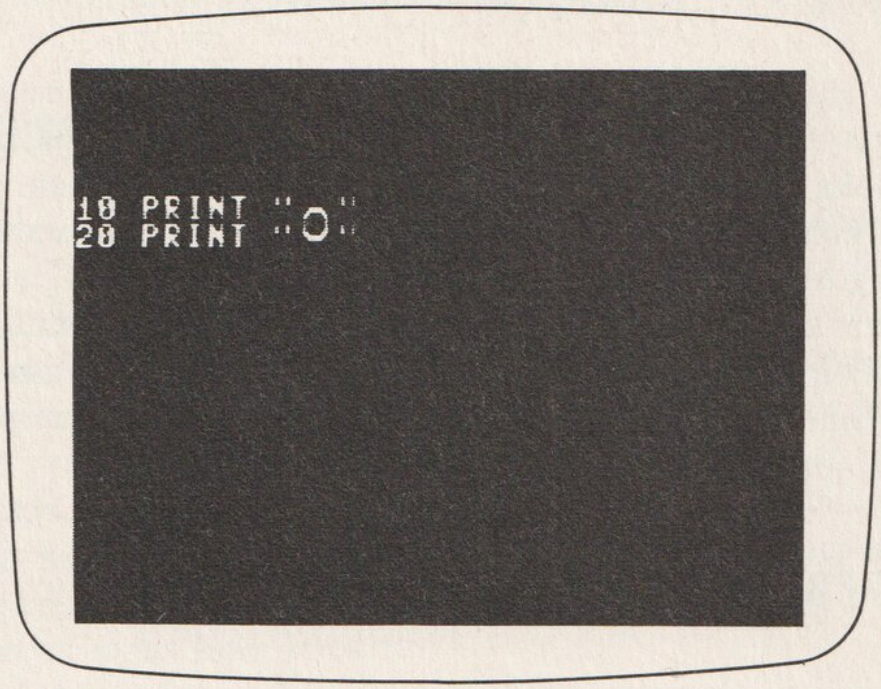
Kuten huomaat, ruutuun piirrettävä laatikko on aivan selvästi nähtävissä, kun listaat ohjelmarivit. Ole kuitenkin varovainen, sillä kun rivinumerot kasvavat 99:stä 100:aan tai 100:sta 1000:een, niiden aiheuttama rivin aloituskohdan muutos saattaa harhauttaa sinut. Muuta rivin 40 numeroksi 10000 ja listaa ohjelmarivit uudelleen. Vaikka laatikko näyttääkin listauksessa vääristyneeltä, se piirtyy kuitenkin ruutuun täysin oikein ohjelmaa suoritettaessa.

Joitakin pieniä graafisia kuvioita voidaan varastoida merkkimuuttujinakin. Kuhunkin merkkimuuttujaan voi sisältyä graafisten merkkien lisäksi kohdistimen siirtoja, jotka sijoittavat kyseisen kuvion oikeaan paikkaan ruudussa.

Näppäin SHIFT yhdessä kirjainnäppäinten U, I, J ja K kanssa kirjoittaa ruutuun graafisia merkkejä, joiden avulla syntyy pyöreitä kulmia. Jos yhdistämme ne kaikki, saamme aikaan

ympyrän (joka tosin on hiukan kulmikas). Eräs tapa piirtää ympyrä ruutuun on käyttää kahta erillistä PRINT-lausetta.

```
10 PRINT "[SHIFT U] [SHIFT I]"
20 PRINT "[SHIFT J] [SHIFT K]"
```



Koska kohdistin siirtyy uuden rivin alkuun piirrettyään ympyrän yläosan rivillä 10, asettuu ympyrän alaosa oikealle paikalleen rivillä 20. Saman voimme saada aikaan kirjoittamalla ympyrää tarkoittavan merkkimuuttujan, johon kuuluvat tarvittavat kohdistimen siirrot.

```
10 A$="[SHIFT U] [SHIFT I] [CRSR VASEMPAAN]
   [CRSR VASEMPAAN] [CRSR ALAS] [SHIFT J]
   [SHIFT K]"
20 PRINT "[CLR]";A$
```

Kuvaruutugrafiikka pyyhitään pois kirjoittamalla ruutuun PRINT-käskyn avulla tyhjiä välejä. Jos esimerkiksi haluamme pyyhkiä pois edellä esitetyn ympyrän, voimme määritellä uuden

merkkimuuttujan, johon sisältyy neljä tyhjää väliä ja CRSR-siirtoja. Lisää nämä rivit:

```
30 FOR I=1 TO 1000:NEXT I
40 B$="[2 VÄLILYÖNTIÄ] [2 CRSR VASEMPAAN]
   [CRSR ALAS] [2 VÄLILYÖNTIÄ]"
50 PRINT "[HOME]";B$
```

Kun ohjelma ehtii riville 30, se pysähtyy, jotta näkisit ruutuun piirretyn ympyrän. Rivi 40 määrittelee B$:n merkkimuuttujaksi, joka kirjoittaa neljä ympyrän osien kokoista tyhjää väliä. Kun kohdistin asetetaan täsmälleen samaan kohtaan kuin mihin ympyrä on piirretty, B$ pyyhkii sen pois.

Eräs etu graafisen kuvion määrittelemisestä merkkimuuttujaksi on se, että voimme piirtää sen mihin tahansa kuvaruudun kohtaan tarvitsematta huolehtia siitä, ovatko rivinumerot oikein, jotta kuvio muodostuisi. Muuta rivi 20:

```
20 PRINT "[CLR] [CRSR ALAS] [CRSR OIKEAAN]";A$
```

Kokeile painamalla useita kertoja näppäimiä CRSR oikeaan ja CRSR alas. Pidä huoli siitä, että pysyttelet ruudun määräämissä rajoissa, 25 CRSR alas ja 40 CRSR oikeaan.

Paras sääntö grafiikkaa tehtäessä on laskea tyhjien välien määrä ja ruutuun kirjoittumattomat merkit (kuten CLR, CRSR-siirrot ja värikäskyt), jotta pysytään ajan tasalla ja varmistutaan siitä, että saadaan aikaan se mitä halutaan. Jos tarvitset apua, piirrä kaavio paperille ja tutki kohdistimen siirtoja sen avulla. Hyvä ajatus on myös käyttää aina näppäintä HOME. Sen avulla voit siirtää kohdistimen takaisin vasempaan yläkulmaan ennen kuin ryhdyt tekemään mitään monimutkaisia siirtoja. Siitä tulee silloin pysyvä lähtökohta.

Pidä erityinen huoli siitä, että kirjoitat puolipisteet oikeille paikoilleen. Jos et ole aivan varma puolipisteiden vaikutuksesta, kertaa, miten niitä käytetään PRINT-käskyn yhteydessä.

Puolipisteet ovat tärkeitä haluttaessa sijoittaa merkkigrafiikka oikeaan paikkaan.

Tottuneetkin ohjelmoijat tekevät silloin tällöin virheitä, ja graafisten kuvioiden piirtäminen ruutuun vaatii keskittymistä. Ohjelmat, joihin sisältyy grafiikkaa, toimivat harvoin hyvin heti ensimmäisellä yrityksellä. Kirjoita grafiikkaohjelma pieninä pätkinä ja testaa jokainen osa sitä kirjoittaessasi.

Digitaalinen noppa

Voimme käyttää käskyä RND ja graafisia merkkejä matkiaksemme nopanheittoa. Tämän yksinkertaisen ohjelman listaus on jäljempänä. Kirjoita se koneeseesi muistaen painaa sulkeisiin merkittyjä näppäimiä. Jos sulkeissa on numero ja näppäin — esim. 8 CRSR alas — paina näppäintä niin monta kertaa kuin numero ilmoittaa, tässä esimerkkitapauksessa siis näppäintä CRSR alas 8 kertaa. Kun painat grafiikkanäppäintä (jotakin näppäintä joko SHIFT- tai COMMODORE-näppäimen kanssa), sinun tulisi nähdä ne graafiset merkit, joista kunkin heitetyn nopan kuva koostuu.

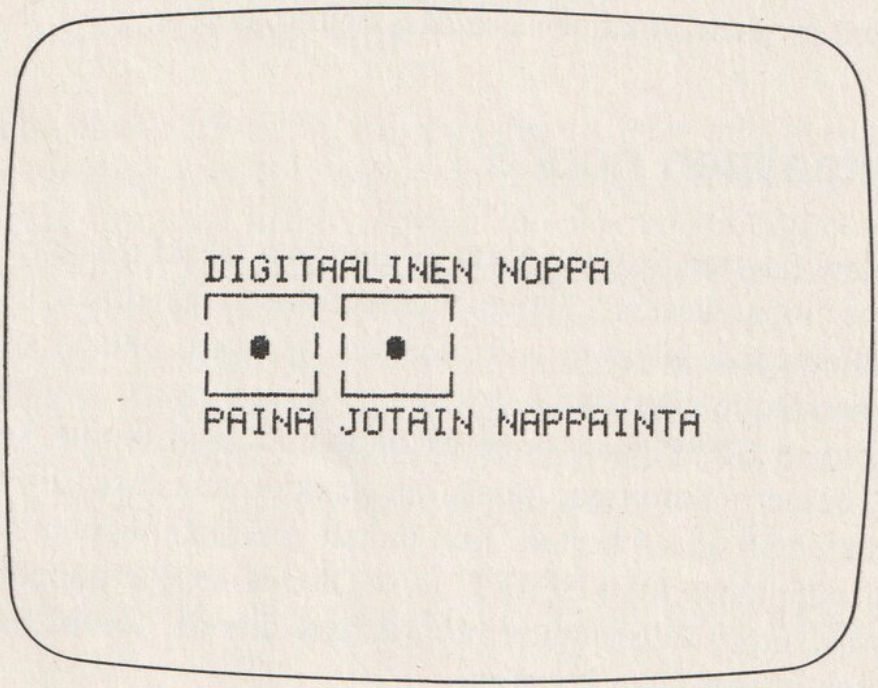
Rivi 10 sijoittaa POKE-käskyn avulla numeroita muistipaikkoihin 53280 ja 53281 muuttaakseen ruudun harmaa 3:ksi (vaaleanharmaa) ja rajauksen harmaa 1:ksi (tummanharmaa). Riviltä 30 alkaa FOR/NEXT-silmukka.

```
30 FOR I=1 TO 25
```

Silmukassa tapahtuu seuraavaa: kehitetään kaksi satunnaislukua, numeeriset muuttujat D1 ja D2, jotka kuvaavat noppaa 1 ja noppaa 2. Kukin luku voi olla väliltä 1—6, koska nopassa ovat juuri ne mahdollisuudet. Kunkin mahdollisen nopan kuva on varastoitu aliohjelmaan, joka alkaa riviltä 1100.

Rivi 60 sijoittaa näppäimellä HOME kohdistimen vakituiseen lähtökohtaansa, josta se sitten siirretään 10 riviä alas ja 15 tyhjää väliä oikealle. Käskyä ON käytetään rivillä 70 piirtämään

oikean nopan kuva. Jos D1 on 1, piirretään noppa, jossa on vain yksi silmä. Jos D1 = 2, piirtyy ruutuun noppa, jossa on kaksi silmää jne. Kohdistin sijoitetaan taas rivillä 80 lähtöasemaan ja sitten piirretään toisen nopan kuva sen saaman arvon mukaan.



Huomaat varmasti, että kukin noppa on aivan samanlainen lukuun ottamatta sen silmälukua. Koska nopat piirretään nopeasti toinen toisensa perään, saadaan aikaan vaikutelma, että ne pyörivät. Tästä syystä ohjelmassa on silmukka FOR I = 1 TO 25. Viimeinen satunnaislukusarja 25. muodostaa sen numeroparin, joka jää noppien arvoksi. Rivi 120 odottaa, että painetaan jotakin näppäintä, ja sitten noppia ”heitetään” uudelleen.

Tämän ohjelman kirjoittaminen koneeseen on paljon helpompaa kuin miltä se näyttää. Säästääksesi aikaa ja vaivaa kirjoita aliohjelma, joka piirtää riveille 1100—1170 nopan kuvan, jossa on yksi silmä. Käytä näppäimiä SHIFT ja Q silmien tekemiseen. Kirjoitettuasi aliohjelman kertaalleen muuta rivinumerot kirjoittamalla niiden päälle ja painamalla näppäintä RETURN. Siirrä sitten kohdistinta muuttaaksesi nopan arvoksi

kaksi. Paina RETURN-näppäintä aina tehtyäsi muutoksen. Muuta rivinumerot — 1100, 1200, 1300, 1400, 1500 ja 1600 — kunnes jokaista kuutta nopan silmälukua vastaava aliohjelma on valmis.

Muista pitää huolta siitä, että kunkin nelikulmion (eli nopan) sisällä olevat merkit ovat tyhjiä välejä, eivät kohdistimen siirtoja. Kuva vaihtuu oikein ainoastaan siinä tapauksessa, että tyhjät välit pyyhkivät pois aikaisemman arvon ja piirtävät uuden.

```
10 POKE 53280,11 : POKE 53281,15
20 PRINT "(CLR)(BLK)(8 CRSR ALAS)(12 CRSR
    OIK)DIGITAALINEN NOPPA";
30 FOR I=1 TO 25
40 D1=INT(6*RND(0)+1)
50 D2=INT(6*RND(0)+1)
60 PRINT "(HOME)(10 CRSR ALAS)(15 CRSR OI
   K)";
70 ON D1GOSUB 1100,1200,1300,1400,1500,16
   00
80 PRINT "(HOME)(10 CRSR ALAS)(21 CRSR OI
   K)";
90 ON D2GOSUB 1100,1200,1300,1400,1500,16
   00
100 NEXT I
110 PRINT "(HOME)(16 CRSR ALAS)(9 CRSR OI
    K)PAINA JOTAKIN NAPPAINTA"
120 GET C$ : IF C$="" THEN 120
130 GOTO 30
1100 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1110 PRINT "(COMM H)(3 VALIA)(COMM N)(5 C
     RSR VAS)(CRSR ALAS)";
1120 PRINT "(COMM H) (SHIFT Q) (COMM N)(5
      CRSR VAS)(CRSR ALAS)";
1150 PRINT "(COMM H)(3 VALIA)(COMM N)(5 C
     RSR VAS)(CRSR ALAS)";
1160 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1170 RETURN
```

```
1200 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1210 PRINT "(COMM H)(SHIFT Q)(2 VALIA)(CO
     MM N)(5 CRSR VAS)(CRSR ALAS)";
1220 PRINT "(COMM H)(3 VALIA)(COMM N)(5 C
     RSR VAS)(CRSR ALAS)";
1230 PRINT "(COMM H)(2 VALIA)(SHIFT Q)(CO
     MM N)(5 CRSR VAS)(CRSR ALAS)";
1260 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1270 RETURN
1300 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1310 PRINT "(COMM H)(SHIFT Q)(2 VALIA)(CO
     MM N)(5 CRSR VAS)(CRSR ALAS)";
1320 PRINT "(COMM H) (SHIFT Q) (COMM N)(5
      CRSR VAS)(CRSR ALAS)";
1330 PRINT "(COMM H)(2 VALIA)(SHIFT Q)(CO
     MM N)(5 CRSR VAS)(CRSR ALAS)";
1360 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1370 RETURN
1400 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1410 PRINT "(COMM H)(SHIFT Q) (SHIFT Q)(C
     OMM N)(5 CRSR VAS)(CRSR ALAS)";
1420 PRINT "(COMM H)(3 VALIA)(COMM N)(5 C
     RSR VAS)(CRSR ALAS)";
1450 PRINT "(COMM H)(SHIFT Q) (SHIFT Q)(C
     OMM N)(5 CRSR VAS)(CRSR ALAS)";
1460 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1470 RETURN
1500 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1510 PRINT "(COMM H)(SHIFT Q) (SHIFT Q)(C
     OMM N)(5 CRSR VAS)(CRSR ALAS)";
```

```
1520 PRINT "(COMM H) (SHIFT Q) (COMM N)(5
      CRSR VAS)(CRSR ALAS)";
1550 PRINT "(COMM H)(SHIFT Q) (SHIFT Q)(C
     OMM N)(5 CRSR VAS)(CRSR ALAS)";
1560 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1570 RETURN
1600 PRINT "(SHIFT O)(COMM Y)(COMM Y)(COM
     M Y)(SHIFT P)(5 CRSR VAS)(CRSR ALAS)
     ";
1610 PRINT "(COMM H)(SHIFT Q)(SHIFT Q)(SH
     IFT Q)(COMM N)(5 CRSR VAS)(CRSR ALAS
     )";
1620 PRINT "(COMM H)(3 VALIA)(COMM N)(5 C
     RSR VAS)(CRSR ALAS)";
1650 PRINT "(COMM H)(SHIFT Q)(SHIFT Q)(SH
     IFT Q)(COMM N)(5 CRSR VAS)(CRSR ALAS
     )";
1660 PRINT "(SHIFT L)(COMM P)(COMM P)(COM
     M P)(SHIFT @)(5 CRSR VAS)(CRSR ALAS)
     ";
1670 RETURN
```

Sprite-grafiikka

Sprite-grafiikkaa on BASICillä vaikeampi ohjelmoida kuin merkkigrafiikkaa. Spriteä ei piirretä kuvaruutuun käskyllä PRINT, vaan ohjelmoinnissa käytetään käskyjä POKE (ja toisinaan PEEK). Liitteessä 2, jossa puhutaan grafiikasta lisää, on paljon tietoa spriteistä, joten tässä luvussa selvitetään vain, kuinka ne toimivat.

Spritet ovat kohteita, joita kuvaillaan sijoittamalla käskyllä POKE arvoja tietokoneen muistissa oleviin lohkoihin. Kukin sprite vie 63 tavua muistia ja on kooltaan 24 pistettä vaakasuoraan ja 21 pistettä pystysuoraan. Kukin spritelohko on numeroitu 0:sta 255:een. VIC-piiri pystyy tulkitsemaan nämä muistilohkot, värittämään ne, piirtämään ne kuvaruutuun ja liikuttelemaan niitä. VIC pystyy myös laajentamaan spritejä vaaka- tai pystysuorassa suunnassa tai kummassakin ja sijoittamaan ne siten, että toinen näyttää peittävän toisen.

Kullakin spritellä on numero 0:sta 7:ään, ja kaikki kahdeksan spriteä voivat olla ruudussa yhtä aikaa. Spritet liikkuvat ruudussa toistensa ylitse ja antavat näin vaikutelman kolmiulotteisesta liikkeestä. Kunkin spriten numero määrittelee sen aseman muihin nähden, esimerkiksi sprite 0 kulkee aina sprite 1:n *editse* ruudussa. Sprite 5 voidaan nähdä numero 4:n *takaa,* jos niiden tiet kohtaavat. Edistyneemmät spritekäskyt määrittävät, onko jokin sprite törmännyt toiseen, johonkin merkkiin tai taustaan. Ns. monivärispritetilassa voit muodostaa jopa neljästä väristä koostuvia spritejä, mutta silloin niiden vaakasuorassa olevien pisteiden määrä on puolet pienempi.

Commodore 64 -koneita varten suunnitellussa sprite-grafiikassa on neljä selvää heikkoutta.

Ensiksikin BASICin tapa käyttää tietokoneen muistitilaa rajoittaa yhtä aikaa käytettävien *erilaisten* spritejen määrän kolmeen. Tästä ongelmasta pääsee muuttamalla BASICin tapaa käyttää tätä muistitilaa. (Se ei ole mitenkään vaikeaa, mutta aloittelijoiden voi olla hankala ymmärtää sitä.) Voit kuitenkin käyttää kaikkia kahdeksaa spriteä yhtä aikaa, kunhan ne vain

ovat samanlaisia.

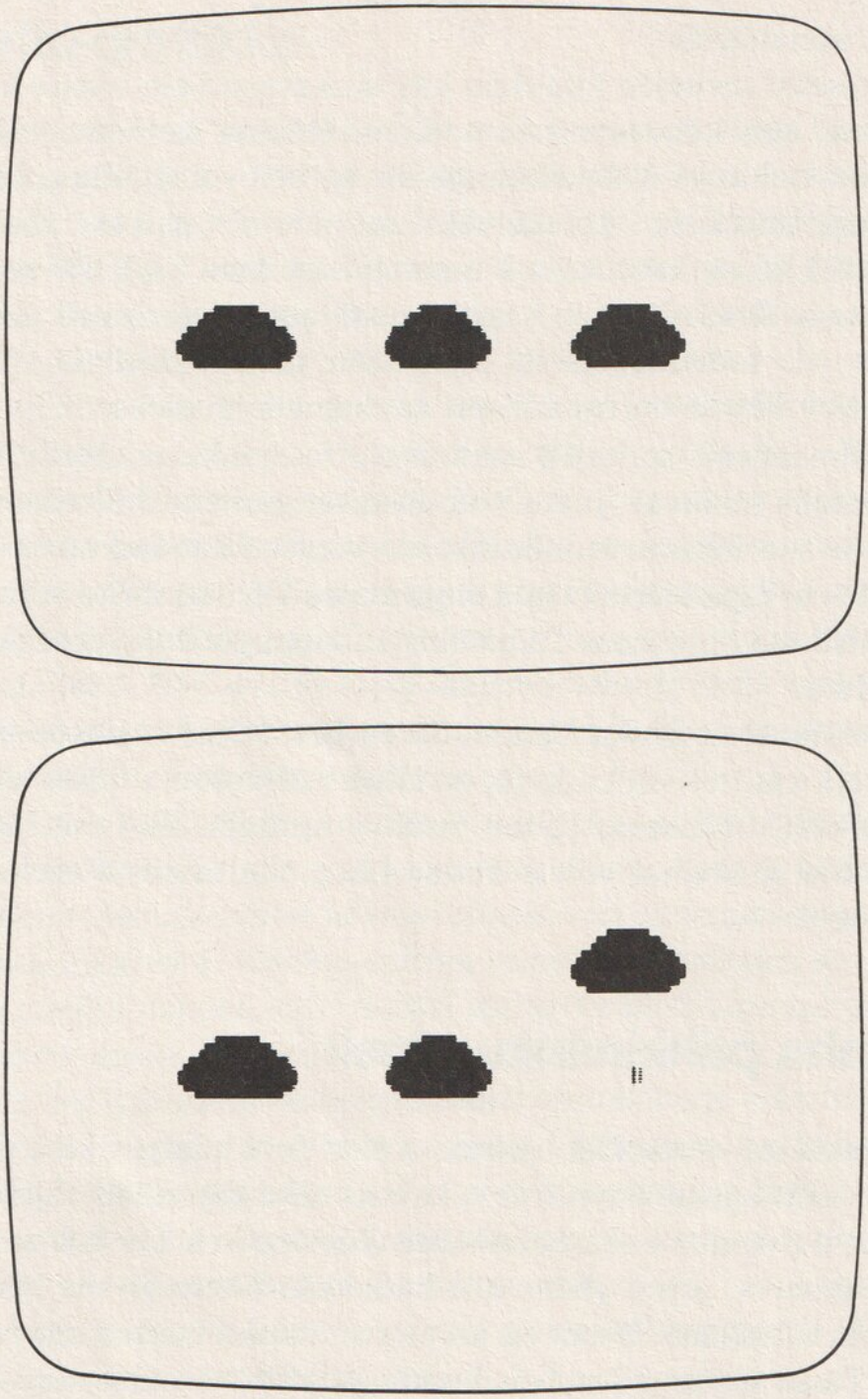
Toiseksi spritejen liikuttelu käy mukavasti vain *osassa* kuvaruutua, sen kahdessa vasemmanpuoleisessa kolmanneksessa. Vaikka onkin yli 300 paikkaa joihin spriten voi sijoittaa, kaikki oikeanpuoleisessa kolmanneksessa olevat paikat vaativat POKE-käskyn käyttämistä useammassa kuin yhdessä muistipaikassa. BASICissä se saattaa osoittautua monimutkaiseksi. Tulet siis todennäköisesti näkemään useita ohjelmia, joissa spritejen käyttö on rajoittunut vasemmalle puolelle.

Kolmanneksi spritejen suunnittelu on hankalaa. Jotta voisit määritellä ne luvut, jotka muodostavat spritemuistilohkon, sinun on suoritettava monimutkaisia ja pitkällisiä laskutoimituksia. Paras tapa kiertää tämä ongelma on käyttää sprite-editoriksi kutsuttua ohjelmaa. Tämän kirjan loppupuolella on sellainen ohjelma.

Neljänneksi, koska Commodoren BASICissä ei ole *lainkaan* spritejä käsitteleviä käskyjä, on lähes mahdotonta *lukea* sellaista BASIC-ohjelmaa, johon sisältyy spritejä. Erityiset ohjelmoinnin apukeinot voivat auttaa tässä, mutta niistä enemmän myöhemmin.

Vanha pähkinänkuoripeli

Seuraavasta esimerkkiohjelmasta saat jonkinlaisen käsityksen siitä, miltä spritet näyttävät ja kuinka niitä käytetään ohjelmassa. Se on versio vanhasta pähkinänkuoripelistä. On kolme pähkinänkuorta, joista yhden alle kätketään herne, ja sen jälkeen kuoria siirrellään. Sinun on arvattava, minkä kuoren alla herne on. Tässä spritet edustavat kuoria ja peittävät aina minkä tahansa niiden alle kirjoitetun merkin. Kuoret kätkevät ”herneen”, joka itse asiassa on vihreä merkki, joka saadaan painamalla yhtä aikaa näppäimiä SHIFT ja Q.



Herne ei liiku lainkaan, vaan valitaan satunnaisluku 1:stä 3:een ja herne piirretään yhden kuoren alle juuri ennen kuin se 'nostetaan'. Tämä ei oikeastaan ole peli siinä mielessä, että se toimii valiten jatkuvasti satunnaislukuja ja pyytäen sinua nostamaan kuoren. Mutta jos näet vähän vaivaa — esim. kirjoittamalla oh-

jelmaan käskyjä GET tai INPUT — voit muuttaa sen varsinaiseksi arvauspeliksi.

Koko ohjelmassa on paljon REM-huomautuksia, jotta pysyisit kärryillä siitä mitä tapahtuu. Ehkä kuitenkin pari huomautusta jo tässä.

Rivi 30 ”herättää” spritet. Se kertoo VIC-piirille, millaisia ja kuinka monta spriteä tarvitaan. Ohjelmassa käytetään vain kolmea spriteä ja ne ovat kaikki aivan samanlaisia. Niiden kuvaus luetaan POKE-käskyllä yhteen spritelohkoon, jonka numero on 13. Muistipaikat 2040, 2041 ja 2042 eivät ole VIC-piirissä, vaan VIC etsii näistä paikoista tiedon siitä, missä spritet on kuvattu.

Rivit 60, 70 ja 80 asettavat spritet oikeille paikoilleen. Ajattele, että kuvaruutu on ikkuna alueella, jolla spritet liikkuvat. Sprite voidaan sijoittaa suunnilleen 344 vaakasuoraan ja 250 pystysuoraan paikkaan. (Muuttujaa X käytetään ilmoittamaan vaakasuora asema ja muuttujaa Y pystysuora asema.) Joissain paikoissa sprite on näkymätön, koska kuvaruudun rajaus peittää sen. Oikea asema merkitään sijoittamalla POKE-käskyllä vastaavat numerot muistipaikkapareihin, jotka alkavat paikasta 53248 (vaakasuora asema, sprite numero 0) ja 53249 (pystysuora asema, sprite numero 0). Muista kuitenkin, että 255:tä suurempia numeroita ei POKE-käskyn avulla voida muistipaikkoihin sijoittaa. (Tämän vuoksi on vaikea liikuttaa spriteä koko kuvaruudun leveydellä.)

Spritejen liikuttaminen — kuorien nostaminen, laskeminen ja ”siirteleminen” — saadaan aikaan antamalla POKE-käskyllä kullekin spritelle uusi arvo sekä pystysuorassa että vaakasuorassa. Näet, kuinka nämä toimivat ”nostoa” esittävissä aliohjelmissa riveillä 13000, 13100 ja 13200 sekä ”siirtelyä” esittävässä aliohjelmassa, joka alkaa riviltä 14000.

```
10 PRINT "(CLR)"; : POKE 53280,12 :
   POKE 53281,15 :
   REM RUUTU VAALEANHARMAA
20 V=53248 : REM VIDEOPIIRIN 1. OSOITE
```

```
30 POKE V+21,7 : REM ALUSTA SPRITET 0,1,2
35 FOR I=0 TO 62 : READ D :
   POKE 832+I,D : NEXT  :
   REM LUE DATA SPRITEILLE KASETTIPUSKURI
   IN
40 POKE 2040,13 : POKE 2041,13 :
   POKE 2042,13 :
   REM KAIKKI SPRITET SAMALLA DATALLA
50 POKE V+39,9 : POKE V+40,9 :
   POKE V+41,9 : REM SPRITET RUSKEITA
55 POKE V+23,7 : POKE V+29,7 :
   REM LAAJENNA KAIKKI SPRITET X JA Y SUU
   NTAAN
60 POKE V+0,85 : POKE V+1,120 :
   REM SPRITE 0 : N PAIKKA (85,120)
70 POKE V+2,160 : POKE V+3,120 :
   REM SPRITE 1 : N PAIKKA (160,120)
80 POKE V+4,235 : POKE V+5,120 :
   REM SPRITE 2 : N PAIKKA (235,120)
90 PRINT "(BLU)(HOME)(15 CRSR ALAS) TERVE
   TULOA VANHAAN PAHKINANKUORIPELIIN"
100 GOSUB 30000 : GOSUB 30000 : REM VIIVE
110 GOSUB 15000 : REM PYYHI TEKSTI
120 PRINT "(BLU)(HOME)(15 CRSR ALAS)(8 VÄ
    LIÄ)PELIN TARKOITUKSENA ON"
130 PRINT "(BLU)(HOME)(16 CRSR ALAS)(12 V
    ÄLIÄ)LOYTAA HERNE"
135 GOSUB 30000 : REM VIIVE
140 PRINT "(BLU)(HOME)(19 CRSR ALAS)(11 V
    ÄLIÄ)HERNE > (GRN)(SHIFT Q)(BLU) < HE
    RNE"
145 GOSUB 30000
150 GOSUB 15000 : REM PYYHI TEKSTI
165 GOSUB 30000 : REM VIIVE
170 PRINT "(BLU)(HOME)(14 CRSR ALAS)(15 V
    ÄLIÄ)ESIMERKKI:"
175 GOSUB 30000 : REM VIIVE
190 GOSUB 12000 :
    REM LAITA HERNE KESKELLE
200 GOSUB 13000 :
    REM NOSTA KESKIMMAINEN KUORI
255 GOSUB 30000 : REM VIIVE
270 GOSUB 14000 : REM SEKOITA KUORET
365 GOSUB 30000 : REM VIIVE
```

```
370 PRINT "(BLU)(HOME)(15 CRSR ALAS)(14 V
    ÄLIÄ)MISSÄ SE ON?"
375 GOSUB 30000 : REM VIIVE
380 A=INT(3*(RND(0))+1)
382 IF A=1 THEN GOSUB 12000 :
    GOSUB 13200 : GOSUB 13100 :
    GOSUB 13000
384 IF A=2 THEN GOSUB 12050 :
    GOSUB 13000 : GOSUB 13100 :
    GOSUB 13200
386 IF A=3 THEN GOSUB 12100 :
    GOSUB 13200 : GOSUB 13000 :
    GOSUB 13100
410 GOSUB 15000 : REM PYYHI TEKSTI
420 GOSUB 30000 : GOSUB 30000 :
    GOSUB 30000 :
    REM KOLMINKERTAINEN VIIVE
999 RUN
9999 REM SEURAAVANA SPRITE DATA
10000 DATA  0,0,0,0,0,0,0,0,0
10010 DATA  0,0,0,0,0,0,0,0,0
10020 DATA  0,0,0,0,0,0,0,0,0
10030 DATA  1,254,0,7,255,128,15,255,192
10040 DATA  63,255,240,127,255,248,255,25
      5,252
10050 DATA  255,255,252,127,255,248,31,25
      5,224
10060 DATA  0,0,0,0,0,0,0,0,0
12000 REM LAITA HERNE KESKIMMAISEN KUOREN
       ALLE
12010 PRINT "(HOME)(12 CRSR ALAS)(19 VÄLI
      Ä)(GRN)(SHIFT Q)(BLU)(11 VÄLIÄ)"
12020 RETURN
12050 REM LAITA HERNE VASEMMANPUOLEISEN K
      UOREN ALLE
12060 PRINT "(HOME)(12 CRSR ALAS)(10 CRSR
       OIK)(GRN)(SHIFT Q)(BLU)(30 VÄLIÄ)"
12070 RETURN
12100 REM LAITA HERNE OIKEANPUOLEISEN KUO
      REN ALLE
12110 PRINT "(HOME)(12 CRSR ALAS)(29 VÄLI
      Ä)(GRN)(SHIFT Q)(BLU)"
12120 RETURN
13000 REM PÄLJÄSTÄ KESKUSTA
```

```
13010 FOR I=120 TO 90 STEP -1
13020 POKE V+2,160 : POKE V+3,I
13030 NEXT I
13040 FOR I=1 TO 500 : NEXT I
13050 FOR I=90 TO 120
13060 POKE V+2,160 : POKE V+3,I
13070 NEXT I
13080 RETURN
13100 REM PALJASTA OIKEA LAITA
13110 FOR I=120 TO 90 STEP -1
13120 POKE V+0,235 : POKE V+1,I
13130 NEXT I
13140 FOR I=1 TO 500 : NEXT I
13150 FOR I=90 TO 120
13160 POKE V+0,235 : POKE V+1,I
13170 NEXT I
13180 RETURN
13200 REM PALJASTA VASEN LAITA
13210 FOR I=120 TO 90 STEP -1
13220 POKE V+4,85 : POKE V+5,I
13230 NEXT I
13240 FOR I=1 TO 500 : NEXT I
13250 FOR I=90 TO 120
13260 POKE V+4,85 : POKE V+5,I
13270 NEXT I
13280 RETURN
14000 REM SEKOITA KUORET
14040 FOR I=85 TO 235
14050 POKE V+0,I : POKE V+1,120
14060 NEXT I
14070 FOR I=235 TO 85 STEP -1
14080 POKE V+4,I : POKE V+5,120
14090 NEXT I
14100 RETURN
15000 REM PYYHI TEKSTI RUUDULTA
15010 PRINT "(BLU)(HOME)(14 CRSR ALAS)(39
      VÄLIÄ)"
15020 PRINT "(BLU)(HOME)(15 CRSR ALAS)(39
      VÄLIÄ)"
15030 PRINT "(BLU)(HOME)(16 CRSR ALAS)(39
      VÄLIÄ)"
15040 PRINT "(BLU)(HOME)(17 CRSR ALAS)(39
      VÄLIÄ)"
```

```
15050 PRINT "(BLU)(HOME)(18 CRSR ALAS)(39
      VALIA)"
15060 PRINT "(BLU)(HOME)(19 CRSR ALAS)(39
      VALIA)"
15070 RETURN
30000 REM VIIVESILMUKKA
30010 FOR I=1 TO 1000 : NEXT I
30020 RETURN
```

SID-ääni

Grafiikan lailla äänikin on ohjelmoijalle todellinen haaste. Kuten VIC-piirin ei myöskään SID-piirin kanssa ole mahdollista viestiä muulla lailla kuin sijoittamalla POKE-käskyllä numeroita sen tärkeisiin muistipaikkoihin. Sitä paitsi et voi käyttää käskyä PEEK tarkistaaksesi, mitkä arvot on asetettu näihin SIDin muistipaikkoihin — tässä tapauksessa PEEK antaa aina tulokseksi nollan.

Tässä yleiset suuntaviivat siitä, millä tavalla SID muodostaa äänen.

Ensin on asetettava SID-piirin äänitaso, jotta ääntä voitaisiin tuottaa. Äänitasoja on 16, numeroiltaan 0—15, joista 0 on ”ei ääntä” ja 15 voimakkain ääni. Tämä taso asetetaan sijoittamalla POKE-käskyllä numero muistipaikkaan 54296.

Tuotettavaa ääntä määrää neljä arvoa: attack/decay, sustain/release, aallonpituus ja aallon muoto.

Attack ja decay asetetaan laskemalla kaksi lukua yhteen ja sijoittamalla niiden summa POKE-käskyn avulla oikeaan muistipaikkaan. Attackin numeron tulisi olla joko 128, 64, 32 tai 16 taikka minkä tahansa näiden summa. Lisää attackin numeroon mikä tahansa decayn numero 8, 4, 2 tai 1 tahi minkä tahansa näiden summa. Mitä suurempi on attackin numero, sitä kauemmin kestää ennen kuin ääni on voimistunut. Jos attackin numero on 0, ääni kuuluu heti. Mitä suurempaa decayn numeroa käytetään, sitä kauemmin kestää äänen tasaantuminen. Jos numero on 16, ääni vaimenee hitaasti. Jos taas numero on 0, mitään tasaantumisaikaa ei ole.

Asettaaksesi äänen 1 attack- ja decay-arvot, sinun on sijoitettava käskyllä POKE näiden numeroiden summa muistipaikkaan 54277. Äänet 2 ja 3 asetetaan sijoittamalla numerot muistipaikkoihin 54284 ja 54291.

Sustain ja release asetetaan samalla tavalla kuin attack ja decay. Äänelle 1 sustain ja release -numeroiden summa sijoitetaan POKE-käskyllä muistipaikkaan 54278. Äänen 2 muistipaikka on 54285 ja äänen 3 muistipaikka on 54292.

Äänen korkeus eli aallonpituus sijoitetaan POKE-käskyllä kahteen paikkaan, ''korkeaan'' ja ''matalaan''. Lue nuottien arvot luvun lopussa olevasta taulukosta. Äänelle 1 on ''matala'' muistipaikka POKE 54272 ja ''korkea'' POKE 54273. (Matala ja korkea eivät viittaa suurempiin tai pienempiin lukuihin, vaan ne ovat kaksi tiettyä numeroa jotka näet nuottitaulukosta.) Äänelle 2 tämä muistipaikkapari on 54279 ja 54280, äänelle 3 muistipaikat ovat 54286 ja 54287.

Sinun on myös valittava aaltomuoto eli äänen väri. Voit valita minkä tahansa neljästä aaltomuodosta, joista jokaisella on erilainen sointi — kolmio, sahalaita, pulssi ja kohina. Kolme ensimmäistä tuottavat ääniä, ja kohinaa käytetään yleensä lyöntiääniin, kuten rumpuihin ja kivääriinlaukauksiin, tahi sihiseviin ääniin tai tyrskyjen kohinaan.

Äänen 1 aaltomuodon muistipaikka on 54276.

POKE 54276,33 saa aikaan sahalaita-aallon,
POKE 54276,17 kolmioaallon,
POKE 54276,65 pulssiaallon,
POKE 54276,129 kohinan.

Äänet 2 ja 3 asetetaan sijoittamalla samat numerot muistipaikkoihin 54283 (äänelle 2) ja 54290 (äänelle 3).

Kun kaikki tämä tehdään ohjelmassa, ääni kuuluu jatkuvasti. Jos haluat lopettaa äänen kuulumisen, sijoita POKE-käskyn avulla 0 kunkin äänen muistipaikkaan. Ja kun asianmukaiset tiedot sijoitetaan uudelleen tarvittaviin muistipaikkoihin käskyllä POKE, kuuluu toisenlainen ääni.

Seuraavassa on ohjelma, joka saa aikaan yhden ainoan äänen. Säädä televisiosi ääni kuulumaan, tai jos käytät monitoria, varmistu siitä, että äänilaite on kytketty koneeseen.

```
10 VL = 54296:REM AANEN VOIMAKKUUS
20 MA = 54272:KO = 54273:REM AANEN KORKEUS
30 AM = 54276:REM AALTOMUOTO
40 AD = 54277:REM ATTACK/DECAY
50 SR = 54278:REM SUSTAIN/RELEASE
60 POKE VL,15
70 POKE AD,190:POKE SR,0
80 POKE AM,33
90 POKE MA,28:POKE KO,214
100 FOR I = 1 TO 500:NEXT I:REM AANEN KESTO
110 POKE AD,0:POKE SR,0:POKE AM,0:POKE VL,0
```

Kuten huomaat, yhden ainoan äänen tuottaminen vaatii paljon työtä. Tee kokeita vaihtamalla attack/decay, sustain/release ja aaltomuotonumeroita.

Yksinkertaisenkin sävelmän soittaminen tällä lailla vaatisi paljon ohjelmoimista. Voimme kuitenkin kirjoittaa pienen ohjelman, joka soittaa nuotteja peräkkäin. Ensiksi muutama sana ohjelman toiminnasta ja sitten sen listaus. Se tuottaa taulukoita, joissa on äänen korkeus, sävelmän nuotit ja kunkin nuotin soittoaika. Tämä musiikillinen tieto varastoidaan DATA-lauseisiin.

Rivinumerosta 10000 alkavat DATA-lauseet sisältävät kukin kolmeen ryhmään jaettua tietoa. Ensimmäinen, C4, on nuotti C ja tarvittava oktaavi, joka on neljäs kahdeksasta mahdollisesta. Seuraavat kaksi numeroa ovat korkean ja matalan äänen numerot. Aina kun näet merkin #, on kyseinen nuotti ylennetty. Toistaiseksi ohjelmassa on vain kaksi oktaavia, mutta loput nuotit voidaan helposti lisätä ohjelmaan tekemällä uusia DATA-lauseita mallin mukaan ja käyttämällä nuottitaulukosta saatavia numeroita.

DATA-lauseiden lopussa rivillä 10040 on epätavallisia kirjai-

mia ja numeroita. ”R,0,0” merkitsee musiikillista ”taukoa”, jolloin ei tuoteta mitään ääntä. Ohjelman READ-käsky taas vaatii viimeiseksi merkiksi nuottiDATA-lauseissa X:n tietääkseen, että kaikki tiedot on luettu.

Rivillä 20000 alkavat DATA-lauseet ovat tietoja sävelmästä kahdessa ryhmässä. Ensimmäisessä on nuotti ja toisessa sen kesto. Kestonumero 16 edustaa kokonuottia, 8 puolinuottia, 4 neljäsosanuottia jne.

Ohjelmalistauksessa on sävelmästä oikeastaan vain kaksi kahdeksan nuotin muodostamaa oktaavia (do-re-mi jne). Muuttamalla nuotteja ja niiden kestoa voit kirjoittaa helppoja yksiäänisiä sävelmiä. Tempoa ohjaa muuttujan T arvo rivillä 300.

```
10 SID=54272 : REM SID-PIIRIN OSOITE
20 DIM N$(94),NL(94),NH(94),S$(200),D(200
   ),SH(200),SL(200)
30 X=1 : REM X-1 :
   STA TULEE NUOTTIEN LUKU MUISTISSA
40 READ N$(X) : IF N$(X)="X" THEN 60 :
   REM TRKISTA NUOTTIDATAN LOPPUMERKKI
50 READ NH(X) : READ NL(X) : X=X+1 :
   GOTO 40
60 N=1 : REM N-1 :
   STA TULEE NUOTTIEN LUKUMAARA LAULUSSA
70 READ S$(N) : IF S$(N)="X" THEN GOTO 90
80 READ D(N) : N=N+1 : GOTO 70
90 FOR I=1 TO N-1
100 FOR K=1 TO X-1 :
    IF S$(I)=N$(K) THEN SL(I)=NL(K) :
    SH(I)=NH(K) : GOTO 120
110 NEXT K
120 NEXT I
300 VL=54272 : VH=54273 : AD=54277 :
    SR=54278 : CR=54276 : VO=54296 :
    T=10 : PRINT "(CLR)"
310 POKE VO,15 :
    REM AANENVOIMAKKUUS MAKSIMI
320 FOR I=1 TO N-1 :
    REM SOITTOSILMUKAN ALKU
330 POKE VL,SL(I) : POKE VH,SH(I)
```

```
340 PRINT "POKE SH,";SH(I);"POKE SL,";SL(
    I);"DURATION";D(I)
350 POKE AD,90 : POKE SR,128 : POKE CR,17
360 FOR K=1 TO T*D(I) :
    REM NUOTIN KESTO-SILMUKKA
370 NEXT K
380 POKE AD,0 : POKE SR,0 : POKE CR,0 :
    REM SAMMUTA
390 NEXT I
400 POKE VO,0 :
    REM ÄÄNENVOIMAKKUUS NOLLAKSI
9999 REM NUOTTIDATA C4 - C6
10000 DATA  C4,16,195,C#4,17,195,D4,18,20
      9,D#4,19,239
10010 DATA  E4,21,31,F4,22,96,F#4,23,181
10020 DATA  G4,25,30,G#4,26,156,A4,28,49,
      A#4,29,233,B4,31,165,C5,33,135
10030 DATA  C#5,35,134,D5,37,162,D#5,39,2
      33,E5,42,62,F5,45,198,F#5,47,107
10040 DATA  G5,50,60,G#5,51,197,A5,56,99,
      A#5,59,190,B5,64,188,C6,68,149,R,0,
      0,X
19999 REM LAULUN DATA, KAKSI OKTAAVIA
20000 DATA  C4,16,D4,16,E4,16,F4,16,G4,16
      ,A4,16,B4,16,R,16,R,16
20010 DATA  C5,16,D5,16,E5,16,F5,16,G5,16
      ,A5,16,B5,16,C6,16,X
```

Nuotteja vastaavat taajuudet

Nuotti	Oskillaattorin taajuus		
Oktaavi	Desimaali	HI (Ylätavu)	LOW (Alatavu)
C – 0	268	1	12
C# – 0	284	1	28
D – 0	301	1	45
D# – 0	318	1	62
E – 0	337	1	81
F – 0	358	1	102
F# – 0	379	1	123
G – 0	401	1	145
G# – 0	425	1	169
A – 0	451	1	195
A# – 0	477	1	221
B – 0	506	1	250
C – 1	536	2	24
C# – 1	568	2	56

Nuotteja vastaavat taajuudet

Nuotti	Oskillaattorin taajuus		
Oktaavi	Desimaali	HI (Ylätavu)	LOW (Alatavu)
D – 1	602	2	90
D# – 1	637	2	125
E – 1	675	2	163
F – 1	716	2	204
F# – 1	758	2	246
G – 1	803	3	35
G# – 1	851	3	83
A – 1	902	3	134
A# – 1	955	3	187
B – 1	1012	3	244
C – 2	1072	4	48
C# – 2	1136	4	112
D – 2	1204	4	180
D# – 2	1275	4	251
E – 2	1351	5	71
F – 2	1432	5	152
F# – 2	1517	5	237
G – 2	1607	6	71
G# – 2	1703	6	167
A – 2	1804	7	12
A# – 2	1911	7	119
B – 2	2025	7	233
C – 3	2145	8	97
C# – 3	2273	8	225
D – 3	2408	9	104
D# – 3	2551	9	247
E – 3	2703	10	143
F – 3	2864	11	48
F# – 3	3034	11	218
G – 3	3215	12	143
G# – 3	3406	13	78
A – 3	3608	14	24
A# – 3	3823	14	239
B – 3	4050	15	210
C – 4	4291	16	195
C# – 4	4547	17	195
D – 4	4817	18	209
D# – 4	5103	19	239
E – 4	5407	21	31
F – 4	5728	22	96
F# – 4	6069	23	181
G – 4	6430	25	30
G# – 4	6812	26	156
A – 4	7217	28	49
A# – 4	7647	29	223
B – 4	8101	31	165
C – 5	8583	33	135
C# – 5	9094	35	134
D – 5	9634	37	162
D# – 5	10207	39	223
E – 5	10814	42	62
F – 5	11457	44	193
F# – 5	12139	47	107

Nuotteja vastaavat taajuudet

Nuotti	Oskillaattorin taajuus		
Oktaavi	Desimaali	HI (Ylätavu)	LOW (Alatavu)
G – 5	12860	50	60
G# – 5	13625	53	57
A – 5	14435	56	99
A# – 5	15294	59	190
B – 5	16203	63	75
C – 6	17167	67	15
C# – 6	18188	71	12
D – 6	19269	75	69
D# – 6	20415	79	191
E – 6	21629	84	125
F – 6	22915	89	131
F# – 6	24278	94	214
G – 6	25721	100	121
G# – 6	27251	106	115
A – 6	28871	112	199
A# – 6	30588	119	124
B – 6	32407	126	151
C – 7	34334	134	30
C# – 7	36376	142	24
D – 7	38539	150	139
D# – 7	40830	159	126
E – 7	43258	168	250
F – 7	45830	179	6
F# – 7	48556	189	172
G – 7	51443	200	243
G# – 7	54502	212	230
A – 7	57743	225	143
A# – 7	61176	238	248
B – 7	64814	253	46

Hupia joystickistä

Kun totut grafiikkaan ja ääneen, haluat todennäköisesti kirjoitta itse omia pelejä. Kaikkein suosituin peliohjain on joystick. Joystickin käyttäminen Commodore 64:ssä on hyvin helppoa.

Joystick ei ole mikään uusi keksintö. Niitä on käytetty säätiminä lentokoneissa, ja tuodessaan markkinoille ensimmäiset videopelinsä Atari-yhtiö otti joystick-peliohjaimet käyttöön. Atarin hupitikusta tuli pian videopelistandardi, ja melkein kaikki nykyajan joystickit toimivat samalla tavalla.

Atarin tyyppinen joystick on katkaisinperiaatteella toimiva ohjain. (On myös ''analoginen'' hupitikku, mutta sitä ei voida käyttää katkaisinperiaatteella toimivien kanssa.) Sisällä on viisi katkaisinta, joista neljän avulla määritellään suunta — pohjoiseen (ylös), etelään (alas), itään (oikealle) ja länteen (vasemmalle) — ja yhden avulla ''ammutaan''. Muut neljä suuntaa — koillinen, kaakko, lounas ja luode — saadaan sulkemalla kaksi katkaisinta yhtä aikaa.

Jos omistat joystickin, liitä se tietokoneen oikeassa laidassa olevaan kontrolliporttiin 2. Miksi tuo numero? Voidaan toki käyttää porttia 1, mutta sen ohjelmoiminen on hivenen hankalampaa. Toistaiseksi siis pysyttelemme portissa 2.

Kurkistamalla käskyllä PEEK tiettyyn muistipaikkaan tiedämme, onko joystickiä työnnetty mihinkään suuntaan.

```
10 PRINT PEEK(56320):GOTO 10
```

Kun suoritat tämän ohjelman, pitäisi kuvaruutuun ilmaantua tasainen numerosarja. Numeron tulisi olla 127. Työnnä nyt peliohjainta ylöspäin. Koska suljet laitteessa olevan katkaisimen, numero vaihtuu. (Luku 0:sta 255:een muodostuu kahdeksasta bitistä, joista viittä käytetään edustamaan joystickin sisällä olevia katkaisimia.) Voimme käyttää näitä lukuja tulkitsemaan suunnanmuutoksia.

```
10 A=PEEK (56320)
20 IF A=127 THEN PRINT ''KESKIPISTE'';
30 IF A=126 THEN PRINT ''POHJOINEN'';
40 IF A=125 THEN PRINT ''ETELÄ'';
50 IF A=123 THEN PRINT ''LANSI'';
60 IF A=119 THEN PRINT ''ITA'';
70 IF A=122 THEN PRINT ''LUODE'';
80 IF A=118 THEN PRINT ''KOILLINEN'';
90 IF A=117 THEN PRINT ''KAAKKO'';
100 IF A=121 THEN PRINT ''LOUNAS'';
105 FOR V=1 TO 500: NEXT:REM VIIVE
110 B=PEEK(56320) AND 16
120 IF B=0 THEN PRINT ''——AMMU'':GOTO 10
130 PRINT '' '':GOTO 10
```

Voit käyttää kontrolliportin avulla saamiasi tuloksia ohjelmassa päättämään ja ohjaamaan kuvaruutuun kirjoitettavia asioita. Voit pitää joystickiä tavallaan mininäppäimistönä, jossa on vain viisi näppäintä. Voit esimerkiksi GET-käskyn ja pyynnön PAINA JOTAKIN NAPPAINTA asemesta käyttää käskyä PEEK(56320) tarkistamaan, onko painettu näppäintä FIRE.

Joystickien serkuissa, paddleissa ei ole katkaisimia (tai on yksi, firenäppäin.) Niissä käytetään *potentiometrejä,* samanlaisia kuin television äänenvoimakkuuden säätönappulat. Tietokone osaa tulkita niitäkin, vaikka se ei olekaan yhtä helppoa kuin joystickin tulkitseminen. Paddlet antavat tulokseksi numeron 0:sta 255:een. Jos käännät paddlen säätimen kokonaan oikealle, on tulos 0, jos taas kokonaan vasemmalle, on tulos 255.

Jos sinulla on paddlet, liitä ne toiseen liitäntään, kontrolliporttiin 1. Kurkista käskyllä PEEK muistipaikkaan 54297.

```
10 PRINT PEEK(54297):GOTO 10
```

Suorita tämä ohjelmarivi. Käännä toista paddlea. Jos numerot eivät muutu 0:sta 255:een, kokeile toista paddlea. (Paddlen ampumisnappulaa luetaan muistipaikassa 56321.)

Käännä nyt paddle keskelle ja laske se pöydälle. On hyvinkin mahdollista, että kuvaruudussa oleva numero muuttuu hiukan (pari numeroa pienemmäksi tai suuremmaksi). Tämän epätarkkuuden vuoksi sinun on suoritettava laskutoimituksia saadaksesi keskiarvon, ja juuri tästä syystä paddlen käyttö on hankalampaa kuin joystickin. Itse asiassa Commodore suosittelee, että pelipaddleja luetaan konekielisellä aliohjelmalla, ei BASICillä.

Mitä on jäljellä?

Kun PEEK- ja POKE-käskyjen käyttö alkaa tuntua rasittavalta, voit alkaa kysellä itseltäsi, mihin oikein olet joutunut. Mutta ei hätää, sillä apu on lähellä.

Näiden hankalien PEEK- ja POKE-käskyjen käytöstä voi päästä eroon käyttämällä nimenomaan grafiikan ja musiikin tekemiseen suunniteltuja ohjelmoinnin apuvälineitä. Commodoren tietokoneisiin on saatavissa lukuisia tällaisia apukeinoja, joista yksi on Simons' BASIC. Simons' BASIC on nuoren englantilaisen ohjelmoijan kehittämä moduuli, josta löytyy 114 uutta BASIC-käskyä. Suuri osa näistä käskyistä helpottaa nimenomaan grafiikan ja musiikin ohjelmoimista.

Tämän kirjan liitteissä on mm. grafiikkaohjelmia, jotka voit kirjoittaa koneeseesi ja käyttää välittömästi hyväksesi. Eräiden mielenkiintoisten esimerkkiohjelmien lisäksi grafiikkaa käsittelevässä liitteessä on helppokäyttöinen sprite-editori ja joitakin ohjelmia, joiden avulla voit piirtää tarkkuusgrafiikkaa. Myös musiikin, äänen ja SID-piirin ohjelmoimisesta on perinpohjainen selitys, josta saat mm. tietää, kuinka soitetaan kolmiäänistä musiikkia.

Liite 1

Tutkimassa Kuusnelosta

Jim Butterfield

Commodore 64:ssä on tutkittavana kokonainen uusi maailma. Voit seurata tietokoneen sisäisiä toimintoja ja jopa yrittää muuttaa asioita nähdäksesi, mitä tapahtuu.

Tietokonetta ei tarvitse avata tätä varten. Voit kurkistaa tietokoneen sisään ”loogisesti” käyttämällä tutkimusvälineenä PEEK-funktiota. Eikä sinun tarvitse näperrellä johtojen kanssa voidaksesi penkoa koneen sisuksia ja muutellaksesi toimintoja — POKE-käskyllä voit tehdä haluamasi muutokset.

Tehdäänpä pieni matka Commodore 64:n sisäisiin toimintoihin ja tutkitaan muutamia kiinnostavia kohtia. Tämän tutustumismatkan jälkeen pystyt tutkimaan asioita jo ominkin päin.

Miksi peukaloida koneen sisuksia?

Miksi ihmeessä haluaisit penkoa tietokonetta? Siihen on moniakin syitä.

Esimerkiksi Commodore 64:n käyttöopas neuvoo sinua muuttamaan näytön reunuksen väriä käskyllä POKE 53280,X, missä X on värikoodi väliltä 0—15. Jotain siis kehotetaan

muuttamaan tietokoneen sisuksissa. Saatuasi enemmän kokemusta tulet löytämään koko joukon osoitteita, jotka ovat hyödyllisiä erilaisissa töissä.

Toiseksi penkominen on kätevä tapa oppia koneen sisäistä toimintaa. Voit tutkia tietoja, joita on tallennettu tietokoneeseen. Voit yrittää peukaloida niitä nähdäksesi, mitä tapahtuu.

Muistin tutkimisella et voi vahingoittaa mitään. Enintään voit saada aikaan sen, että tietokone lakkaa toimimasta. Mutta jos niin käy, sinun tarvitsee vain katkaista virta ja kytkeä se uudelleen ... ja kone toimii taas.

Tietokoneen muisti

Tietokone käyttää muistiaan kaikkeen mitä se tekee. Miltei kaikki sen tärkeimmistä toiminnoista tapahtuvat muistin avulla. Muisti sisältää esim. käskykoodit, tiedon (datan), omat ohjelmasi ja näytön kopion. Siellä on paljon asioita.

Muisti koostuu soluista, joihin tietoa voidaan tallentaa. Kullakin solulla on ''osoite''. Commodore 64:ssä on 65 536 osoitettavaa solua; osoitteet alkavat 0:sta ja päättyvät 65 535:een. Kukin solu voi sisältää lukuarvon väliltä 0—255. Monet muistialueet on varattu erityistehtäviin.

Tietokoneessa käytetään kolmenlaisia muisteja; yksi niistä ei kuitenkaan ole varsinainen muisti. Puhutaan nyt niistä ja katsotaan, missä ne sijaitsevat.

Muistin tyypit

RAM-muistiin voimme kirjoittaa (POKElla) ja sitä voimme myös lukea (PEEKillä). Se on kuin varastoalue: voimme viedä sinne tietoa ja myöhemmin palata hakemaan tallennettua tietoamme. Kun kirjoitat ohjelman Commodore 64:ään, se tallentuu RAM-muistiin. Kun ohjelmasi laskee arvoja, ne tallennetaan RAMiin (ja myöhemmin luetaan sieltä). RAM tulee sa-

noista Random Access Memory (suorasaantimuisti). Se ei ole kovin kuvaava nimi.

Normaalitilassa Commodore 64:ssä on RAM-muistia käytettävissä osoitteissa 2—40959 ja 49152—59151. Alempaa aluetta käytetään BASIC-ohjelmiin ja työn aikaisiin arvoihin, ylempää aluetta ei käytetä BASIC-ohjelmien yhteydessä lainkaan. Myöhemmin ilmenee, että sinulla on käytettävissäsi enemmänkin RAMia.

ROM on kiinteätä muistia. Voit lukea sitä (PEEK-käskyllä), mutta et voi kirjoittaa siihen. Tiedot ROMiin on asetettu jo tehtaassa, eikä niitä voi muuttaa. ROM tulee sanoista Read Only Memory (lukumuisti), joka onkin hyvä nimitys. Saattaa tuntua rajoitukselta, että on ROMia, jota ei voi muuttaa. Mutta ROMin voit kytkeä halutessasi pois ja ottaa RAMia tilalle. ROM sisältää tietokoneen tärkeimmän logiikan: esimerkiksi miten neliöjuuri lasketaan tai kuinka tietoa tallennetaan kasettinauhalle. Useimmat ''sisäänrakennetut'' piirteet missä tahansa tietokoneessa on tavallisesti tallennettu ROM-muistiin.

Commodore 64:ssä on ROM-muisti tavallisesti käytettävissä osoitteissa 40960—49151 ja 57644—65535. Ensimmäinen alue sisältää BASIC-tulkin, jonka avulla tietokone ymmärtää BASIC-kieltä. Toinen alue sisältää ohjeet siitä, miten hoidetaan syöttö- ja tulostustoiminnot näppäimistöllä, ruudussa, levykkeellä tai nauhalla. On kolmaskin, syrjään työnnetty ROM-alue, jonne emme normaalisti voi kurkistaa PEEKillä. Se on merkkigeneraattori, joka näyttää, miten kukin merkki piirretään ruutuun. Ja jos kytket pelimoduulin koneeseen, todennäköisesti kytket lisäROMin, joka sisältää peliohjelman.

Kolmatta muistivälinetyyppiä kutsutaan liitännäksi (interface). Se ei oikeastaan ole muistia lainkaan, sillä emme voi olla varmoja siitä, että siihen tallennettuja tietoja voidaan myöhemmin palauttaa. Liitännän tärkein tehtävä on liittää asioita keskusyksikön ulkopuolelle. Näyttöpiiri (Video Interface Chip) luo kuvan, jonka näemme ruudussa. Äänipiiri (Sound Interface Device) tekee äänet kuultaviksemme. Ja kaksi monitoimimuunninta (Complex Interface Adaptor) hoitavat yhteyksiä näppäi-

mistöön, levy-yksikköön, peliohjaimiin ... ja tekevät ohessa monia muitakin tehtäviä, kuten ajoituksen.

Näyttöpiiri (eli videopiiri) sijaitsee osoitteissa 53262—53286 ja äänipiiri osoitteissa 54272—54300. Kaksi monitoimimuunninta sijaitsevat osoitteissa 56230—56335 ja 56576—56591. Osoitteilla 0 ja 1 on aivan omat tehtävänsä: niiden tärkein työ on säädellä, miten muistin toiset osat ovat mukana muistin kokoonpanossa.

Muistikartat

Tämän liitteen lopussa on joitakin pitkähköjä ''muistikarttoja''. Niiden tarkoituksena on antaa sinulle vihjeitä muistipaikkojen käytöstä.

RAM on mitä kiintoisinta muistia, sillä siellä voimme kokeilla ja muuttaa asioita nähdäksemme, mitä tapahtuu. Tutkimme RAMia hieman tarkemmin tuonnempana.

ROMia emme voi muuttaa. Niinpä sen sisältö on kiinnostavaa lähinnä vain tutkimisen vuoksi. Jos haluamme analysoida, kuinka kone toimii, voimme tutkia tarkasti ROMia. Mutta silloin meidän pitää ymmärtää uudenlaista kieltä: 6502-konekieltä.

6502-prosessori on hyvin läheistä sukua Commodore 64:ssä käytettävälle 6510-prosessorille. ROMiin tallennetut käskyt ovat 6502-tyyppisiä käskyjä. On kuitenkin aivan eri asia oppia lukemaan niitä.

Liitäntäpiirien käyttö on hyvin käytännöllistä; itse asiassa monet liitäntäosoitteet on mainittu 64:n käyttöoppaassa. Sinun pitää tuntea joitakin niistä jos haluat saada aikaan ääniä tai muuttaa asioita näytöllä.

Olet ehkä huomannut, että liitäntäpiirejä kuvataan karttoina. Tähän on syynä se, että monet paikat säätelevät useampaa kuin yhtä toimintaa. Muistipaikka koostuu yksittäisistä kontrollielementeistä, joita kutsutaan biteiksi. Bitit taas on koodattu käyttäen seuraavia lukuja:

128 64 32 16 8 4 2 1

Jos esimerkiksi näet jonkin muistipaikan sisältävän luvun 20, pieni laskutoimitus paljastaa, että 16:lla ja 4:llä koodatut bitit ovat toiminnassa ja muut eivät. Mikään muu yhdistelmä ei tuota arvoksi 20:tä.

Kun tutkit liitäntäpiirien karttoja, näet yksittäiset bitit ja sen kuinka niitä pitää yhdistellä, jotta saat halutun kontrolliarvon.

Tutkimusmatkalla RAMissa

RAM-muisti on niin mielenkiintoinen, että siitä kannattaa puhua pitempään. Katsellaanpa osoitteita ja poimitaan esiin muutamia kiintoisimpia.

Osoitteet 2—1023 ovat tietokoneen työalue; siellä on paljon tärkeitä paikkoja. Aluetta 2—255 kutsutaan ''nollasivuksi''. Se on täynnä koneen työskentelyarvoja. Jos annat esimerkiksi käskyn PRINT PEEK(203), saat selville, onko jokin näppäin sillä hetkellä alas painettuna vai ei: arvo 64 merkitsee, että mitään näppäintä ei paineta. Tällä käskyllä ei kuitenkaan ole juuri käyttöä suorassa käyttömoodissa: sinun pitäisi olla tosi nopea nostaaksesi sormesi RETURNilta ennen kuin tulos tulee ruutuun. Mutta ohjelman sisällä PEEK(203) voi kertoa, painaako joku näppäintä vai ei. Ja mitä näppäintä, sen kertoo GET-lause.

Muistipaikat 256—511 ovat melko hiljaisia. Tätä aluetta käytetään tietokoneen sisäisenä työalueena, jota kutsutaan ''pinoksi''. Et saa paljoakaan irti tästä alueesta.

Muistipaikat 512—677 ovat aika vilkkaita. Eräs suosikeistani täällä on paikka 650, joka säätelee näppäinten automaattista toistoa. Olet ehkä huomannut, että kursorin ohjaus- ja välilyöntinäppäimet toistavat automaattisesti, kun niitä pidetään koko ajan painettuina. Yritäpä sijoittaa POKElla paikkaan arvot 255, 127 sekä 0 ja katso mitä tapahtuu.

0300 0333

Muistipaikoilla 768—819 on erityinen tehtävä. Ne sisältävät joukon liittymisosoitteita, joiden avulla konekieliohjelmoijat voivat muuttaa tietokoneen käyttäytymistä. He tekevät sen muuttamalla liittymisosoitteita uusiksi omiksi osoitteikseen. Tällä tavalla he voivat saada tietokoneen tekemään hämmästyttäviä asioita. Ellet osaa konekieltä, näitä muistipaikkoja muuttamalla saat aikaan vain sen, että kone käyttäytyy virheellisesti ... tai jumiutuu kokonaan.

Muistipaikoissa 828—1019 on kasettiaseman puskurialue. Kasetilta lukiessasi tai sinne tallentaessasi tällä alueella on toimintaa, kun nauhan tietoja käsitellään. Ellet satu käyttämään nauhaa, tämä alue on vapaa ja käytettävissäsi. Sitä käytetäänkin usein ''spritehahmojen'' sijoittamiseen.

Muistialueella 1024—2047 sijaitsee ''kuvaruutumuisti''. Se on hauska alue: POKE sinne aiheuttaa jonkin merkin ilmaantumisen kuvaruutuun. Yritäpä millä tahansa arvolla 0—255, kukin luo ruutuun erilaisen merkin. Tällä alueella voi toimia molempiin suuntiin: jos ohjelma PEEKkaa jotain näistä osoitteista, se ''näkee'', mitä ruudun siinä kohdassa on. Monet pelit käyttävät POKE- ja PEEK-käskyjä kuvaruutumuistiin luomaan animaatioita.

Alueella 2048—40959 on muisti, jossa tietokone säilyttää sisään näppäilemämme BASIC-ohjelman. Kun ohjelma suoritetaan, sen muuttujat ja taulukot tallennetaan myös samalle alueelle. Tätä aluetta on hieman vaikea käyttää, koska BASIC käyttää sitä, mutta voit ehkä yrittää tutkia sitä, miten ohjelma tallennetaan muistiin ... se on monimutkaisempaa kuin saattaisi luullakaan.

2048—40969 on laaja alue, etkä ehkä tarvitse sitä kaikkea BASIC-ohjelmassa. Voit pienentää sitä tehdäksesi tilaa esim. sprite-koodille tai konekielisille aliohjelmille. Se edellyttää näpertelyä osoitteiden 43—44 (BASICin alkuosoite) tai 55—56 (BASICin loppuosoite) kanssa. Kuinka voit lukea osoitteita, jotka on tallennettu kahteen tavuun? Lue (PEEK) ensimmäinen tavu ja lisää siihen 256 kertaa toisen tavun arvo. Asiat menevät kuitenkin konstikkaiksi ... BASIC-ohjelma ei voi tietenkään

muuttaa omia alku- ja loppuosoitteitaan silloin kun se on käynnissä.

Osoitteissa 49152—53247 on vielä ylimääräistä RAMia, mutta normaalisti kukaan ei käytä sitä. Se on mainio paikka tehdä kokeita ... erityisesti yrittää tehdä ja testata konekielisiä ohjelmia.

Suuri muistinsiirto

Näyttäisi siltä, että olemme jo käyneet läpi kaikki paikat. Mutta Commodore 64:ssa onkin RAM-muistia kaikkialla ... itse asiassa 64K tarkoittaa 65 536 muistipaikkaa. Jopa niissä osoitteissa, joissa ROM-muistia, on RAM piilossa ROMin takana.

Jos esimerkiksi annamme käskyn PRINT PEEK(45000), saamme arvon ROM-muistista; se on osa BASIC-tulkkia. Mutta jos annamme käskyn POKE 45000,100, tietokone tietää, ettemme voi tallentaa ROMiin ... joten se tallentaa arvon 100 RAMiin, joka on ROMin takana. Entä voitko lukea RAMia? Et silloin kun ROM on tiellä: uusi PEEK(45000) antaa vain ROMin sisällön. Mutta me voimme kytkeä ROMin pois tieltä käyttäen osoitteen 1 kontrollielementtejä.

Odotapa hetki. Jos kytkemme ROMin pois, menetämme BASICin. Se on paha juttu, koska tarvitsemme sitä antaaksemme käskyjä — jopa POKE on BASIC-käsky. Onko mitään keinoa, jolla saamme ROMin pois ja silti voimme pitää BASICin?

Se on helppoa. Kopioidaan ROM RAMiin alapuolelle ja kytketään sitten ROM elektronisesti pois, ja BASIC on käytettävissä RAMissa. Tämä tuo käyttöömme jotain uutta: mahdollisuuden muuttaa BASICia.

Kopioidaanpa BASIC ROMista RAMiin alapuolelle. Näppäile seuraava rivi:

```
FOR J=40960 TO 49151: POKE J,PEEK(J): NEXT J
```

Ensi silmäyksellä tämä näyttää älyttömältä. Asetamme PO-

KElla jokaiseen muistipaikkaan saman tiedon, jonka olemme juuri lukeneet siitä. Mutta kun ymmärrämme, että se tulee ROMista ja menee RAMiin, koko jutussa on sittenkin mieltä. Tietokoneellakin vie hetken aikaa kopioida kaikki kahdeksantuhatta tavua. Kun se on valmis, näppäile:

```
POKE 1,54
```

BASIC toimii nyt RAMista. Yritäpä tätä:

```
POKE 41848,66
```

Koneesi pitäisi nyt ilmoittaa READYn sijasta BEADY. Olet muuttanut BASICia ... pienellä tavalla, mutta se on silti todellinen muutos.

Voit jälleen ottaa käyttöön BASICin ROMista näppäilemällä POKE 1,55. BEADYstä tulee taas READY.

Yhteenveto

Voin antaa vain vihjeitä niistä kaikista aarteista, jotka voit löytää seikkailemalla koneessasi. Tietenkään sinun ei tarvitse, jos et halua; Kuusnelosta voi toki käyttää järkevänä, suoraviivaisena tietokoneena, jollaisena olet sen ennenkin tuntenut.

Commodore 64:n muistikartta, laatinut Jim Butterfield

Hex	Desim.	Kuvaus
0000	0	Prosessorin datan suuntarekisteri
0001	1	Prosessorin s/t-rekisteri; muistin ja nauhan kontrolli
0002	2	Ei käytössä

Hex	Desim.	Kuvaus
0003—0004	3—4	Hyppyosoite: muunnos liukuluvuista kokonaislukuihin
0005—0006	5—6	Hyppyosoite: muunnos kokonaisluvuista liukulukuihin
0007	7	Etsittävä merkki
0008	8	Lippu: etsitään merkkijonon loppulainausmerkkiä
0009	9	Viimeisen TAB-sarakkeen tallennus
000A	10	Lippu: 0 = LOAD, 1 = VERIFY
000B	11	Syöttöpuskurin osoite/#-indeksit
000C	12	Lippu: oletus DIM-arvo
000D	13	Tiedon tyyppi: $FF = merkkijono, $00 = numeerinen
000E	14	Tiedon tyyppi: $80 = kokonaisluku, $00 = liukuluku
000F	15	Lippu: DATAn selaus/LIST-viittaus/roskien keruu
0010	16	Lippu: indeksiviittaus/käyttäjän funktio
0011	17	Lippu: $00 = INPUT, $40 = GET, $98 = READ
0012	18	Lippu: ATN merkki/vertailun tulos
0013	19	Lippu: nykyinen s/t-kehote

Hex	Desim.	Kuvaus
0014—0015	20—21	Kokonaisluvun väliaikainen säilytys
0016	22	Osoitin: väliaikainen merkki-jonopino
0017—0018	23—24	Viimeisen väliaikaisen merkkijonon osoite
0019—0021	25—33	Pino väliaikaisille merkki-jonoille
0022—0025	34—37	Apuohjelmien osoitinalue
0026—002A	38—42	Kertolaskun liukulukutulos
002B—002C	43—44	Osoitin: BASICin alku
002D—002E	45—46	Osoitin: muuttujien alku
002F—0030	47—48	Osoitin: taulukoiden alku
0031—0032	49—50	Osoitin: taulukoiden loppu
0033—0034	51—52	Osoitin: merkkijonojen talletusalueen loppu
0035—0036	53—54	Apuohjelmien merkkijono-osoitin
0037—0038	55—56	Osoitin: BASICin käyttämän muistin yläraja
0039—003A	57—58	Nykyinen BASIC-ohjelman rivinumero
003B—003C	59—60	Edellinen BASIC-ohjelman rivinumero
003D—003E	61—62	Osoitin: BASIC-lause CONT-käskyä varten
003F—0040	63—64	Nykyinen DATA-lauseen rivinumero
0041—0042	65—66	Osoitin: nykyisen DATA-tiedon osoite
0043—0044	67—68	Hyppyosoite: INPUT-rutiini

X) BASIC MUISTIN YLÄRAJAN ASETUS
POKE (51-52) : POKE (55-56) : CLR

XX) BASIC MUISTINALUEIDEN ASETUS
POKE (43-44) : POKE (PEEK(43) + PEEK(44)*256), 0
CLR : NEW

Hex	Desim.	Kuvaus
0045—0046	69—70	Nykyisen BASIC-muuttujan nimi
0047—0048	71—72	Osoitin: nykyisen BASIC-muuttujan sisältö
0049—004A	73—74	Osoitin: FOR/NEXT-silmukan indeksimuuttuja
004B-004C	75—76	Y-talletus; op-talletus; BASIC-osoitin talletus
004D	77	Vertailusymboliakku
004E—0053	78—83	Sekalainen työalue, osoittimia jne.
0054—0056	84—86	Hyppyosoite funktioille
0057—0060	87—96	Sekal. numeerinen työalue
0061	97	Liukulukuakku n:o 1: eksponentti
0062—0065	98—101	Liukulukuakku n:o 1: mantissa
0066	102	Liukulukuakku n:o 1: merkki
0067	103	Osoitin: sarjojen arviointivakio
0068	104	Liukulukuakku n:o 1: ylivuoto
0069—006E	105—110	Liukulukuakku n:o 2: eksponentti jne. kuten akku n:o 1
006F	111	Esimerkkivertailun tulos: akku n:o 1 vs. akku n:o 2
0070	112	Liukulukuakku n:o 1: pyöristys
0071—0072	113—114	Osoitin: kasettipuskuri

Hex	Desim.	Kuvaus
0073—008A	115—138	CHRGET-aliohjelma; ota seuraava merkki BASIC-ohjelmassa
007A—007B	122—123	Osoitin: nykyinen BASIC-merkki
008B—008F	139—143	RND-funktion siemenluku
0090	144	Tilamuuttuja ST
0091	145	Lippu: STOP-näppäin/RVS-näppäin
0092	146	Nauhan ajoitusvakio
0093	147	Lippu: 0 = Load, 1 = Verify
0094	148	Lippu: sarjaväylän ulosmenevä merkki
0095	149	Puskuroitu merkki sarjaväylään
0096	150	Nauhan loppu (EOT) vastaanotettu
0097	151	Tilapäinen rekisteri
0098	152	Avoimien tiedostojen lukumäärä
0099	153	Syöttölaite, normaalisti 0
009A	154	Tulostuslaite, normaalisti 3
009B	155	Nauhan pariteetti
009C	156	Lippu: tavu saatu nauhalta
009D	157	Lippu: $80 = suora käyttö, $00 = ohjelma
009E	158	Nauhaluku 1: virheloki
009F	159	Nauhaluku 2: virheloki
00A0—00A2	160—162	Kotvakello (1(60 sek)
00A3	163	Sarjaväylän bittilaskuri
00A4	164	Syklilaskuri
00A5	165	Kasetin sykronointilaskuri

Hex	Desim.	Kuvaus
00A6	166	Osoitin: nauhan s/t-puskuri
00A7	167	RS232 syöttöbitit/kasetin väliaikaisia tietoja
00A8	168	RS232 syöttöbittien laskuri/ nauhan väliaikaisia tietoja
00A9	169	RS232 lippu: aloitusbitin tarkistus
00AA	170	RS232 syöttötavupuskuri/ nauhan väliaikaisia tietoja
00AB	171	RS232 syöttöpariteetti/ nauhan laskuri
00AC—00AD	172—173	Osoitin: nauhapuskuri/ ruudun vieritys
00AE—00AF	174—175	Nauhan lopun osoitin/ ohjelman loppu
00B0—00B1	176—177	Nauhan ajoitusvakioita
00B2—00B3	178—179	Osoitin: nauhapuskurin alku
00B4	180	RS232 tulostusbittilaskuri/ 1 = nauhan ajoitus päällä
00B5	181	RS232 seuraava lähetysbitti/ nauhan loppulippu (EOT)
00B6	182	RS232 tulostustavupuskuri
00B7	183	Tiedostonimen pituus
00B8	184	Nykyisen loogisen tiedoston numero
00B9	185	Nykyinen sekundaarinen osoite
00BA	186	Nykyinen laitenumero
00BB—00BC	187—188	Osoitin: nykyinen tiedostonimi

Hex	Desim.	Kuvaus
00BD	189	RS232 tulostuspariteetti/kasetin väliaikaisia tietoja
00BE	190	Kasetin luku/kirjoitus-blokkien laskuri
00BF	191	Sarjaväylän sanapuskuri
00C0	192	Nauhan moottorin kytkentä
00C1—00C2	193—194	S/T aloitusosoite
00C3—00C4	195—196	Käyttöjärjestelmän pohjustusosoite
00C5	197	Viimeksi painettu näppäin
00C6	198	Näppäimistöpuskurissa olevien merkkien määrä
00C7	199	Lippu: käänteiset merkit 1 = kyllä, 0 = ei
00C8	200	Osoitin: loogisen rivin loppu INPUT-lauseessa
00C9—00CA	201—202	Kursorin sijainti (rivi, sarake)
00CB	203	Mikä näppäin: 64 jos ei mikään
00CC	204	Kursorin välkyntäkytkin: 0 = välkytä
00CD	205	Kursorin ajoitin
00CE	206	Kursorin alla oleva merkki
00CF	207	Lippu: kursorin välkyntä-vaihe
00D0	208	Lippu: syöttö ruudulta tai näppäimistöltä
00D1—00D2	209—210	Osoitin: nykyinen rivi näytöllä
00D3	211	Kursorin sarake yo. rivillä

Hex	Desim.	Kuvaus
00D4	212	Lippu: 0 = suora kursori, muuten ohjelmoitu
00D5	213	Nykyisen rivin pituus
00D6	214	Kursorin rivinumero
00D7	215	Väliaikaisia tietoja
00D8	216	Lippu: Insert-moodi/ lisäysten lukumäärä
00D9—00F2	217—242	Näyttörivin linkkitaulu
00F3—00F4	243—244	Osoitin: näytön värimuisti
00F5—00F6	245—246	Hyppyosoite: näppäimistön dekoodaustaulukko
00F7—00F8	247—248	Osoitin: RS232 syöttö puskuri
00F9—00FA	249—250	Osoitin: RS232 tulostus-
FB ? FE	251-255	puskuri Varaa
0100—010A	256—266	Työalue: liukuluvut/ASCII-muunnos
0100—103E	256—318	Nauhan virheiden lokialue
0100—01FF	256—511	Prosessorin pinoalue
0200—0258	512—600	BASICin syöttöpuskuri
0259—0262	601—610	Taulukko: aktiivisten loogisten tiedostojen numerot
0263—026C	611—620	Taulukko: kunkin tiedoston laitenumerot
026D—0276	621—630	Taulukko: kunkin tiedoston sekundaariset osoitteet
0277—0280	631—640	Näppäimistöpuskuri (jonoperiaate)
0281—0282	641—642	Osoitin: BASIC-muistin alku
0283—0284	643—644	Osoitin: BASIC-muistin loppu

Hex	Desim.	Kuvaus
0285	645	Lippu: sarjaväylän ajoitus
0286	646	Nykyisen merkin värikoodi
0287	647	Taustaväri kursorin alla
0288	648	Kuvaruutumuistin sivu
0289	649	Näppäimistöpuskurin maksimikoko
028A	650	Näppäinten toiston kontrolli
028B	651	Toiston nopeuslaskuri
028C	652	Toiston viivelaskuri
028D	653	Lippu: SHIFT-näppäin/ CTRL-näppäin/Commodore näppäin
028E	654	Viimeinen shift-tyyppi
028F—0290	655—656	Hyppyosoite: näppäimistötaulukon pohjustus
0291	657	Lippu: näppäimistön shift-moodi ($00 = SHIFT-näppäimet pois, $80 = SHIFT-näppäimet päälle)
0292	658	Lippu: 0 = vieritys päällä
0293	659	RS232 kontrollirekisteri
0294	660	RS232 käskyrekisteri
0295—0296	661—662	RS232 bittiajoitus
0297	663	RS232 tilarekisteri
0298	664	RS232 bittejä jäljellä lähetykseen
0299—029A	665—666	RS232 baudinopeuskoodi
029B	667	RS232 syöttöpuskurin loppuindeksi

Hex	Desim.	Kuvaus
029C	668	RS232 syöttöpuskurin alku (sivu)
029D	669	RS232 tulostuspuskurin alku (sivu)
029E	670	RS232 tulostuspuskurin loppuindeksi
029F—02A0	671—672	Keskeytyshyppyosoite nauhan s/t-toiminnan aikana
02A1	673	CIA 2 (NMI) keskeytys-kontrolli
02A2	674	CIA 1 ajastin A:n kontrolli-loki
02A3	675	CIA 1 keskeytysloki
02A4	676	Lippu: CIA 1 ajastin A päällä
02A5	677	Näytön rivin indeksi
02C0—02FE	704—766	(Sprite 11)
0300—0301	768—769	Hyppyosoite: BASICin virheilmoitus
0302—0303	770—771	Hyppyosoite: BASICin lämmin käynnistys
0304—0305	772—773	Hyppyosoite: tiivistä BASIC-koodi
0306—0307	774—775	Hyppyosoite: BASIC koodin listaus
0308—0309	776—777	Hyppyosoite: aloita uusi BASIC-koodi
030A-030B	778—779	Hyppyosoite: aritmeettisen elementin otto
030C	780	Prosessorin A-rekisterin talletus

Hex	Desim.	Kuvaus
030D	781	Prosessorin X-rekisterin talletus
030E	782	Prosessorin Y-rekisterin talletus
030F	783	Prosessorin status-rekisterin talletus
0310-0312	784—786	USR-funktion hyppykoodi ja osoite (B248)
0313	787	Ei käytössä
0314—0315	788—789	Hyppyosoite: laitteiston (hardware) keskeytykset (EA31)
0316—0317	790—791	Hyppyosoite: BREAK-keskeytykset (FE66)
0318—0319	792—793	Hyppyosoite: NMI-keskeytykset (FE47)
031A—031B	794—795	Hyppyosoite: OPEN-rutiini
031C—031D	796—797	Hyppyosoite: CLOSE-rutiini
031E—031F	798—799	Hyppyosoite: CHKIN-rutiini
0320—0321	800—801	Hyppyosoite: CHKOUT-rutiini (F250)
0322—0323	802—803	Hyppyosoite: CLRCHN-rutiini (F333)
0324—0325	804—805	Hyppyosoite: CHRIN-rutiini
0326—0327	806—807	Hyppyosoite: CHROUT-rutiini
0328—0329	808—809	Hyppyosoite: STOP-rutiini
032A—032B	810—811	Hyppyosoite: GETIN-rutiini
032C—032D	812—813	Hyppyosoite: CLALL-rutiini
032E—032F	814—815	Hyppyosoite: lämmin käynnistys (FE66)
0330—0331	816—817	Hyppyosoite: LOAD-rutiini

Hex	Desim.	Kuvaus
0332—0333	818—819	Hyppyosoite: SAVE-rutiini
033C—03FB	828—1019	Kasetin s/t-puskuri
0340—037E	832—894	(Sprite 13)
0380—03BE	896—958	(Sprite 14)
03CO—03FE	960—1022	(Sprite 15)
0400—07FF	1024—2047	Kuvaruutumuisti
0800—9FFF	2048—40959	BASIC RAM-muisti
8000—9FFF	32768—40959	Vaihtoehtoisesti: ROM-moduulialue
A000—BFFF	40960—49151	BASIC-ROM
A000—BFFF	40960—59151	Vaihtoehtoisesti: RAM
C000—CFFF	49152—53247	RAM-muisti
D000—D02E	53248—53294	Videopiiri (6566)
D400—D41C	54272—54300	Äänipiiri (6581 SID)
D800—DBFF	55296—56319	Värimuisti
DC00—DC0F	56320—56335	Liityntäpiiri 1, IRQ(6526 CIA)
DD00—DD0F	56576—56591	Liityntäpiiri 2, NMI(6526 CIA)
D000—DFFF	53248—53294	Vaihtoehtoisesti: merkki-generaattori
E000—FFFF	57344—65535	ROM: käyttöjärjestelmä
E000—FFFF	57344—65535	Vaihtoehtoisesti: RAM
FF81—FFF5	65409—65525	Hyppytaulukko, sisältäen mm.:
FFC6		Avaa syöttökanava
FFC9		Avaa tulostuskanava
FFCC		Sulje s/t-kanavat
FFCF		INPUT-rutiini
FFD2		PRINT-rutiini
FFE1		Testaa STOP-näppäintä
FFE4		GET-rutiini

Prosessorin s/t-portti (6510)

$0000	IN	IN	OUT	IN	OUT	OUT	OUT	OUT	DDR 0
$0001			Nauhurin moottori	Nauha-tunniste	Kirjoitus nauhalle	D-ROM kytkin	EF RAM kytkin	AB RAM kytkin	PR 1

SID (6581)

Ääni 1	Ääni 2	Ääni 3		Ääni 1	Ääni 2	Ääni 3
$D400	$D407	$D40E	Taajuus L	54272	54279	54286
$D401	$D408	$D40F	Taajuus H	54273	54280	54287
$D402	$D409	$D410	Pulssin leveys L	54274	54281	54288
$D403	$D40A	$D411	0 0 0 0 Pulssin leveys H	54275	54282	54289
$D404	$D40B	$D412	Aaltomuoto: NSE PUL SAW TRI Key	54276	54283	54290
$D405	$D40C	$D413	Nousuaika 2ms - 8ms; Tasaantumisaika 6ms - 24 sec	54277	54284	54291
$D406	$D40D	$D414	Jälkisointitaso; Laskuaika 6ms 24 sec	54278	54285	54292

Äänet (vain kirj.)

$D415	0 0 0 0 0	L	54293
$D416	Suodatustaajuus	H	54294
$D417	Resonanssi; Suodatinäänet: Ext V3 V2 V1		54295
$D418	Kaista: V3 off HI BP LO; Yleisvoimakkuus		54296

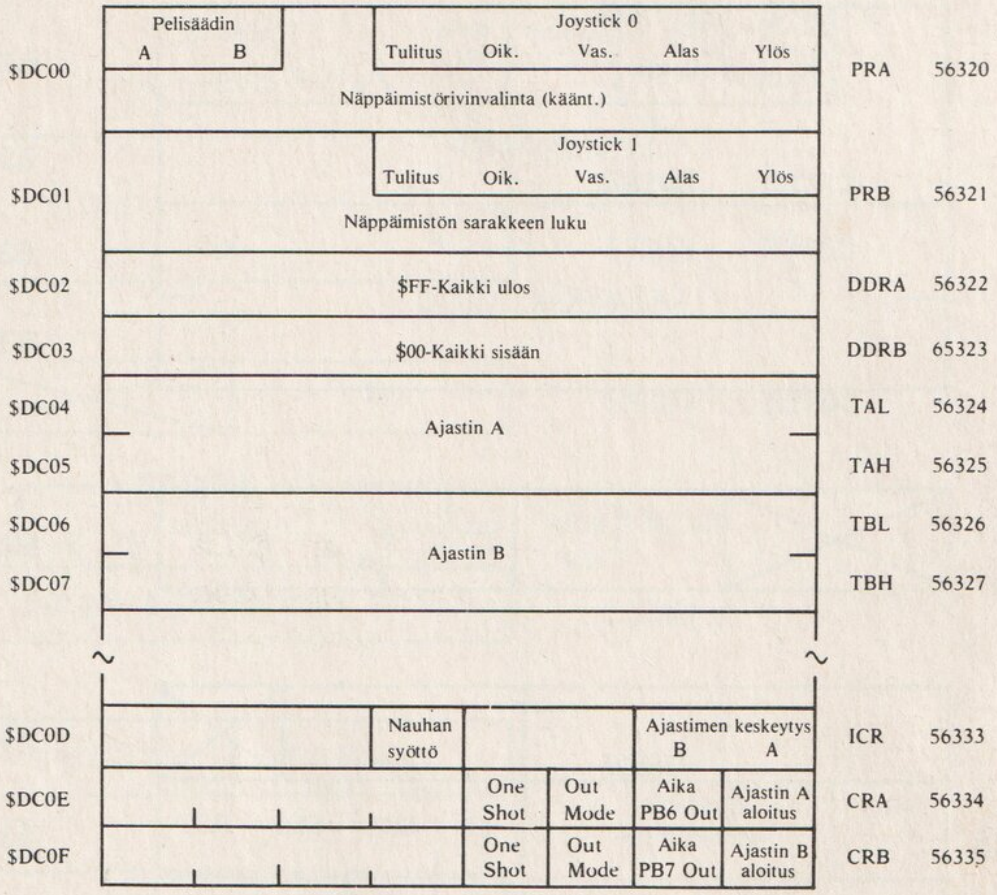
Suodatus ja voimakkuus (vain kirj.)

Pelisäädin X (A/D# 1)
Pelisäädin Y (A/D# 2)
Kohina 3 (satunn.)
Muotoilu 3

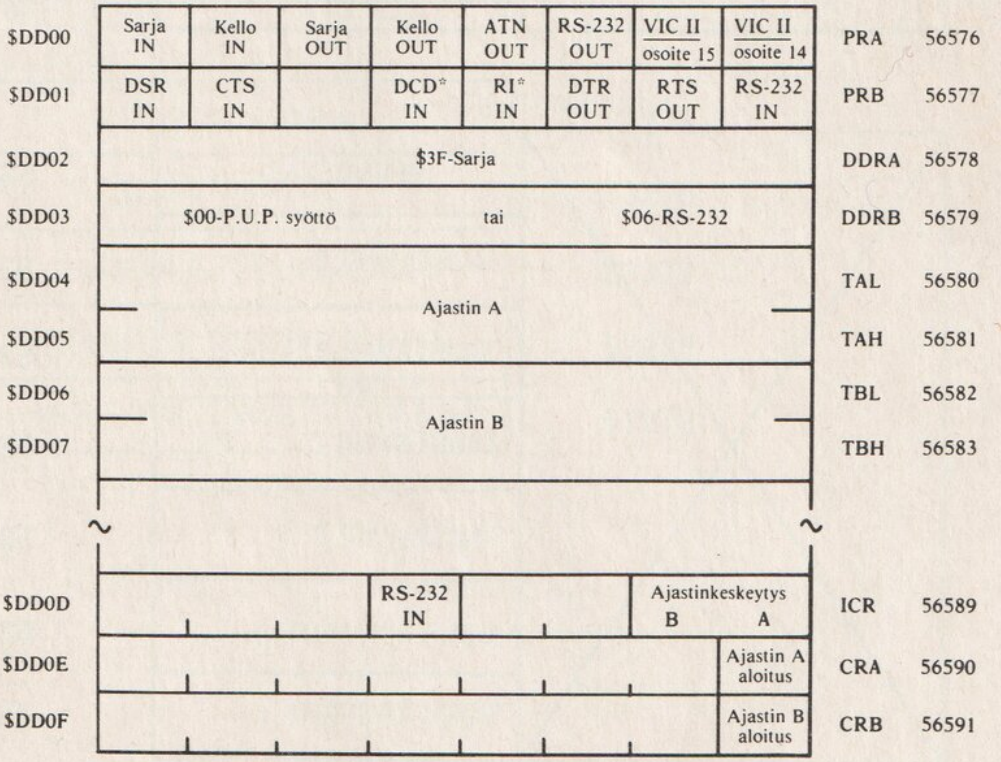
Tunnistus (vain luku)

Huom! Äänen muodostuksen erikoispiirteet
(TEST, RING, MOD, SYNC)
on jätetty pois yllä olevasta kaaviosta.

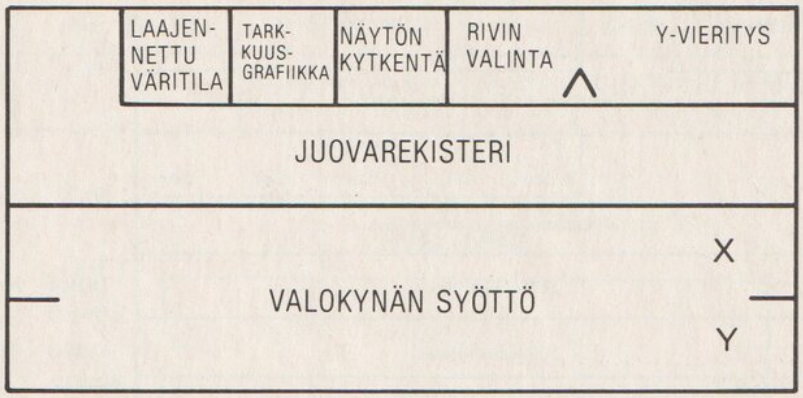
CIA (IRQ) (6526)



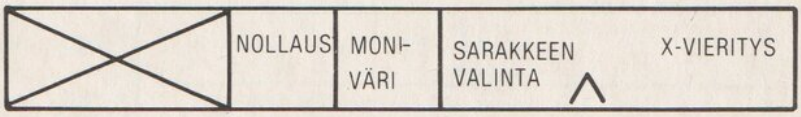
CIA 2 (NMI) (6526)



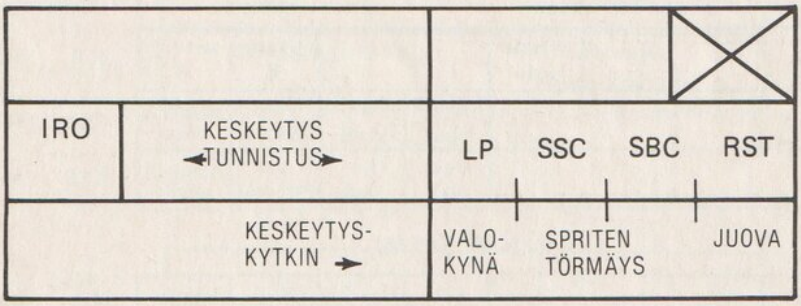
VIDEO (6566)



53265
53266
53267
53268

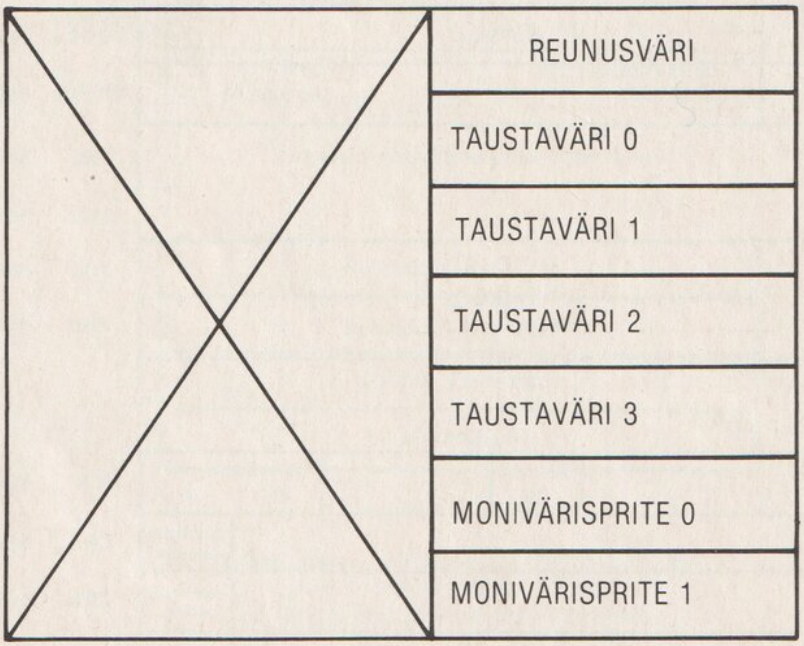


53270



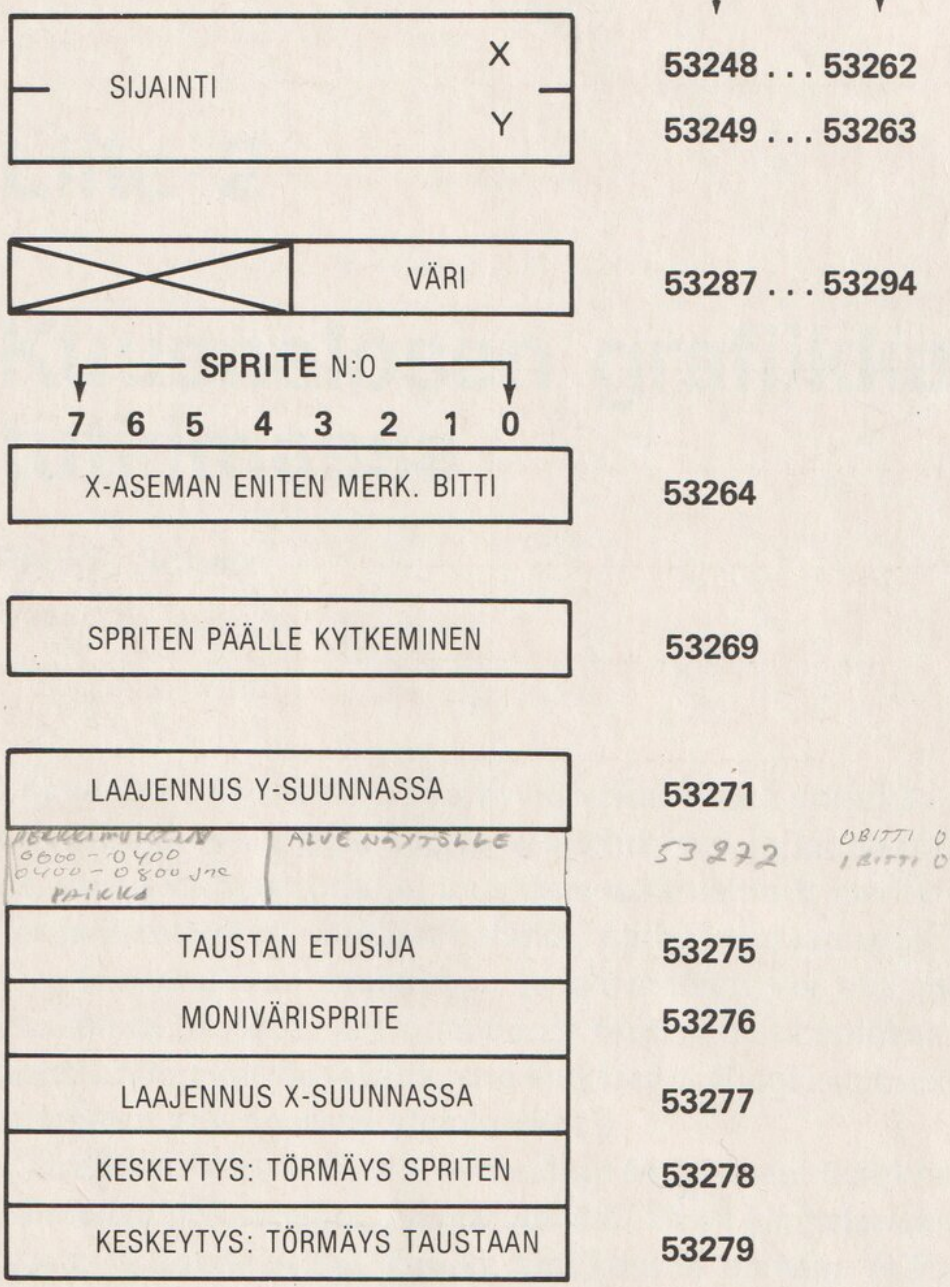
53272
53273
53274

VÄRIREKISTERIT



53280
53281
53282
53283
53284
53285
53286

SPRITE-REKISTERIT



Liite 2

Kuusnelosen grafiikkaa tutkimassa

Paul F. Schatz

Commodore 64:ssä on eräitä hyvin voimakkaita grafiikkaominaisuuksia, esimerkiksi suuren erottelukyvyn tarkkuusgrafiikka, useita värinäyttötiloja, uudelleen määriteltävät merkkijoukot ja liikutettavat värilliset hahmot, joita kutsutaan spriteiksi. Tietokonegrafiikan suunnittelu ja ohjelmointi voi olla monimutkainen prosessi, ja Commodore 64:n piirteitä voidaan yhdistellä niin monella tavalla, että kaikkien mahdollisuuksien selittäminen käy nopeasti ylivoimaiseksi.

Grafiikan käyttäminen Commodore 64:llä vaatii tietokoneen toiminnan perusteellista ymmärtämistä. Tässä kappaleessa selitetään yksinkertaisesti, kuinka tietokoneen muistiin tallennetuista luvuista saadaan kuva näyttöruutuun. Tulet oppimaan jotain kolmesta erityisestä grafiikkaominaisuudesta — spriteistä, ohjelmoitavista merkeistä ja tarkkuusgrafiikkatekniikasta. Kukin selitetään ja esitellään yksinkertaisten ohjelmien avulla. Näiden tietojen perusteella pystyt ohjelmoimaan jo varsin näyttäviä kuvia.

Grafiikkaa pikseleillä

Avain grafiikan ymmärtämiseen on tajuta, kuinka tietokoneen muistissa käytetään numeroita kuvan määrittelemiseen. Sinun on ymmärrettävä, mitä tarkoitetaan käsitteellä tavu. Kun kytket virran tietokoneeseesi, ruutuun ilmestyy seuraava yhden rivin ilmoitus:

64K RAM SYSTEM 38911 BYTES FREE

38911 BYTES FREE ilmaisee niiden muistipaikkojen (rekisterien) määrän, jotka tietokoneessa on käytettävissä BASIC-ohjelmaa varten. Mikä tahansa 256 erilaisesta merkistä voidaan tallentaa yhteen muistipaikkaan. Näitä merkkejä esitetään luvuilla 0—255. Miksi juuri 256 merkkiä? Vastaus tähän kysymykseen piilee siinä, kuinka tietokone tallentaa lukuja. Kutakin muistipaikkaa voidaan pitää joukkona kytkimiä, numeroituina 0—7, jotka voidaan kytkeä ”päälle” tai ”pois”. Jos kutakin kytkintä kuvaa jokin sopiva luku, kuten seuraavassa on osoitettu, mikä tahansa luku 0—255 voidaan esittää laskemalla yhteen ”päällä” oleviin kytkimiin liitetyt luvut.

KYTKIN	7	6	5	4	3	2	1	0
LUKU	128	64	32	16	8	4	2	1

Kuva 1

Kytkinten tila voidaan esittää ykkösillä ja nollilla, joista 1 esittää tilaa ”päällä” ja 0 tilaa ”poissa”. Ykkösten ja nollien sarjaa kutsutaan binaariluvuksi. Se vastaa tavanomaisempaa desimaalilukua. (Lukuja, joita käytämme jokapäiväisessä elämässämme, kutsutaan desimaaliluvuiksi, koska ne käyttävät kymmentä erilaista numeroa 0—9). Tietokonekielessä näitä kytkimiä kutsutaan biteiksi, ja tavu koostuu kahdeksasta bitistä. Esimerkiksi luku 16 esitetään kytkin (bitti) 4 käännettynä päälle (1) ja muut pois (0).

KYTKIN	7	6	5	4	3	2	1	0
PÄÄLLÄ/POIS	0	0	0	1	0	0	0	0
LUKU	128	64	32	16	8	4	2	1

Luku 129 esitetään kytkimet (bitit) 0 ja 7 ''päällä'' ja kytkimet (bitit) 6, 5, 4, 3, 2 ja 1 ''poissa''.

KYTKIN	7	6	5	4	3	2	1	0
PÄÄLLÄ/POIS	1	0	0	0	0	0	0	1
LUKU	128	64	32	16	8	4	2	1

Suurin desimaaliluku, joka voidaan esittää yhdellä tavulla, on 255 — kaikki kahdeksan bittiä ovat ykkösiä eli ''päällä''. Taulukko 1 kuvaa kaikki kahdeksan bitin binaariluvut, jotka vastaavat desimaalilukuja 0—255. Sitä voidaan käyttää apuvälineenä grafiikkaa ohjelmoitaessa.

Tietokonegrafiikassa tavua käytetään määrittelemään pisteriviä tai kuvaelementtejä näyttöruutuun. Näitä kuvaelementtejä kutsutaan nimellä pikseli (pixel). Vasemmanpuoleisin pikseli kahdeksan pikselin joukosta on sama kuin bitti 7 tavussa. Mikäli bitti tavussa on ''päällä'', vastaava pikseli on myös ''päällä''. Jos bitti on ''poissa päältä'', on myös pikseli ''poissa päältä''. Eräs tapa ajatella tavua on kuvitella, että kytkimet (bitit) säätelevät kahdeksan lampun riviä. Asettamalla useita tällaisia lamppurivejä lähekkäin voidaan luoda kuva, aivan kuin urheilustadionin tulostaululla.

Spritet, ohjelmoitavat merkit ja tarkkuusgrafiikka ovat kaikki samanlaisia siinä, että kuvat muodostuvat kahdeksan pikselin riveistä, joiden sisällön määräävät tiettyjen muistipaikkojen arvot. Nämä grafiikan piirteet eroavat kuitenkin niiden muistipaikkojen määrissä, joita käytetään kuvan määrittelemiseen, sekä siinä, kuinka tietokone järjestää pikselirivit, peräkkäin vai vierekkäin.

TAULUKKO 1: Muunnos binaarijärjestelmästä desimaalijärjestelmään

Bitit 0—3 ovat sarakkeiden päällä ja bitit 4—7 rivien alussa.
Esimerkki: Luvun 18 binaariesitys on 0001 0010 (toinen rivi - kolmas sarake).

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0001	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0010	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0011	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
0100	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
0101	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
0110	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
1000	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
1001	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
1010	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
1011	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
1100	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
1101	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
1110	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
1111	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Miten kuvat luodaan?

Kuvan luomiseen käytetään muistipaikkoja. Siitä miten tieto esitetään ruudussa, huolehtii videopiiri (VIC-piiri). Tämä kehittynyt mikropiiri Commodore 64:n sisällä muodostaa videosignaalit. VIC-piiri kasvattaa tietokoneen tehokkuutta sekä muistin käytössä että ohjelman suoritusajassa. Monet VICin suorittamista tehtävistä — esimerkiksi spritet — voidaan tehdä tietokoneella monimutkaisten ohjelmien avulla. Mutta ohjelma joka tekee VICin tehtävät spriteillä, tarvitsee muistitilaa, ja sen suoritusnopeus on hitaampi kuin mikropiirillä hoidetun spritegrafiikan.

Vastauksena mahdollisesti heräävään kysymykseen: tämä VIC-niminen piiri ei ole sama, joka on Commodoren toisessa suositussa tietokoneessa VIC-20:ssä. Se on sukua tälle, mutta tämä VIC on kuitenkin paljon monipuolisempi kuin pikkuserkkunsa.

Se miten VIC-piiri luo kuvansa, määräytyy sen mukaan, mitä on sen sisältämissä 47 muistipaikassa, rekistereissä 53248—53294. Kuvaan voidaan saada dramaattisia muutoksia muuttamalla arvoja missä tahansa näistä muistipaikoista.

Jotkin muistipaikat ovat päällä/pois-kytkimien ryhmiä. Esimerkiksi muistipaikkaa 53269 voidaan ajatella kahdeksana kytkimenä, jotka kytkevät tai vapauttavat yksittäisiä sprite-hahmoja. Piirissä on myös kytkimiä grafiikkatilojen vaihtamiseen, esim. merkkigrafiikan, tarkkuusgrafiikan tai laajennetun taustaväritilan valintaan. Eräät rekisterit kertovat VIC-piirille, minne videoinformaatio on tietokoneen muistissa tallennettu.

Ennen kuin jatkamme, on hyvä tietää, että VIC-piirillä on myös rajoituksia. Tärkeimpiä niistä on se, että se voi ''nähdä'' vain 16 384 (16K) muistipaikkaa. Kaiken videoinformaation on oltava 16K muistialueen sisällä. Kun Commodore 64 kytketään toimintaan, VIC-piiri asettuu automaattisesti niin, että sen näkemä 16K:n lohko ulottuu muistipaikasta 0 muistipaikkaan 16383. Tämä asettaa joitakin rajoituksia siihen, miten videoin-

3FFF

formaatiota voidaan tallentaa, sillä tiettyjä osia tästä alueesta käytetään muihin tehtäviin. Merkittävin alue on 0—827, jota BASIC ja käyttöjärjestelmä käyttävät osoittimien ja lukujen tallentamiseen. Toinen ongelma-alue on se osa muistia, johon BASIC-ohjelma tallennetaan. Tietokone alkaa tallentaa BASIC-ohjelmaa muistipaikasta 2048 eteenpäin. Monet ohjelmat ovat niin pieniä, että niiltä jää tilaa 16K:n lohkon loppupäähän videoinformaation tallentamiseen. Kuitenkin pitkä ohjelma kuluttaa kaiken muistitilan alueella 2048—16383. Siinä tapauksessa on mahdollista kertoa VIC-piirille, että se ottaa videoinformaationsa joltakin toiselta 16K:n muistialueelta. Koska tässä johdannossa yritetään pitää asiat yksinkertaisina, vältetään videoinformaation tallentamista muille alueille kuin 0—16383.

Kuinka VICille puhutaan?

BASIC-sanat, joita tarvitaan muuttamaan VIC-piirin sisältöä, ovat PEEK, POKE, AND, OR ja NOT. Ellet ole ohjelmoinut vähään aikaan, nämä saattavat olla hieman outoja.

PEEK-käskyä käytetään tietyn muistipaikan sisällön katsomiseen. Yritäpä tätä:

```
PRINT PEEK(53272)
```

Tietokone vastaa tulostamalla luvun 21. Muistipaikan 53272 sisältö säätelee sitä, missä merkkijoukko sijaitsee. Kytke toinen sisäänrakennettu merkkijoukko mukaan painamalla Commodore- ja SHIFT-näppäimiä yhtä aikaa. Suorita sitten sama PEEK-käsky uudelleen. Nyt saatkin eri luvun. Tietokone ottaa merkit eri muistialueelta.

POKE-käskyllä asetetaan lukuja tiettyihin muistipaikkoihin. Toinen tapa muuttaa merkkijoukkoa on antaa POKE-käsky.

```
POKE 53272,21
```

Kumpaa merkkijoukkoa tietokone käyttää? Muuta merkkijoukko taas kirjoittamalla:

```
POKE 53272,23
```

Vain välillä 0—255 olevia lukuja voidaan asettaa POKElla muistipaikkoihin. Muut arvot tuottavat virheilmoituksen ILLEGAL QUANTITY ERROR.

Valitettavasti sanojen AND, OR ja NOT selittäminen ei ole yhtä suoraviivaista kuin PEEKin ja POKEn. Näitä kutsutaan loogisiksi operaattoreiksi, joita grafiikassa käytetään tiettyjen bittien (kytkinten) päälle ja pois kytkemiseen muuttamatta toisten bittien arvoja. Katsotaanpa, kuinka niitä käytetään.

ORia käytetään bittien asettamiseen. Kun esimerkiksi halutaan kytkeä muistipaikassa 53272 bitti 1 vaikuttamatta muihin bitteihin, käytetään käskyä:

```
POKE 53272, PEEK(53272) OR 2
```

PEEK ottaa luvun, joka on muistipaikassa 53272. ORia seuraava luku määrää, mitkä bitit asetetaan. OR 2 asettaa bitin 1 binaariarvoksi ykkösen. POKE asettaa uuden luvun takaisin muistipaikkaan 53272.

KYTKIN	7	6	5	4	3	2	1	0
PÄÄLLÄ/POIS	0	0	0	0	0	0	1	0
LUKU	128	64	32	16	8	4	2	1

Edellä oleva asetelma esittää lukua 2. Koska ainoa ”päällä” oleva kytkin on numero 2, se asettaa tämän bitin mihin tahansa lukuun, jonka kanssa sitä käytetään. Useampiakin kuin yksi bitti voidaan asettaa kerralla. Esimerkiksi OR 129 asettaa sekä bitin 7 että bitin 0 ykköseksi.

KYTKIN	7	6	5	4	3	2	1	0
PÄÄLLÄ/POIS	1	0	0	0	0	0	0	1
LUKU	128	64	32	16	8	4	2	1

Taulukkoa 1 voidaan käyttää sopivan desimaaliluvun määrittelyssä asetettaessa ORilla bittejä.

Bitit kytketään pois käyttämällä AND- ja NOT-käskyjen yhdistelmää. Esimerkiksi bitti 1 käännetään pois päältä muistipaikassa 53272 seuraavasti:

```
POKE 53272, PEEK(53272) AND NOT 2
```

PEEK-käsky ottaa luvun. Bitti 1 tyhjennetään AND NOT 2 -käskyllä ja uusi luku tallennetaan POKElla takaisin muistipaikkaan 53272.

Sprite-grafiikka

Commodore 64:n mielenkiintoisimpia piirteitä on sen sprite-grafiikka. Spritet ovat tekemiäsi kuvia, joita tietokone käsittelee erillään muusta grafiikasta ja tekstistä. Nämä pikku kuvat määritellään 63 tavulla muistia, ja niitä voidaan liikuttaa ympäri näyttöruutua BASICin käskyillä. Kun asetetaan lukuja tiettyihin VIC-piirin muistipaikkoihin, voidaan saada jopa kahdeksan hahmoa ilmestymään ja häviämään, vaihtamaan väriä ja kokoa, sekä vaihtamaan muotoa ja sijaintia. Tämä tekee mahdolliseksi luoda BASIC-ohjelmassa pehmeästi liikkuvia ja vaikuttavia animaatioita.

Sprite-kuvan määrittelee 63 muistipaikan muodostaman lohkon sisältö. Spriten ensimmäisen rivin määrittelevät kolme ensimmäistä muistipaikkaa, toisen rivin seuraavien kolmen muistipaikan sisältö jne. Spritessä on 21 riviä. Koska kukin muistipaikka kontrolloi 8 pikseliä ja sprite-rivi sisältää 24 pikseliä, spriten erotuskyky on 24×21 pistettä. Kuva 2 näyttää tavujen järjestymisen muistipaikkoihin.

TAVU 1	TAVU 2	TAVU 3
TAVU 2	TAVU 5	TAVU 6
TAVU 3	TAVU 8	TAVU 9
•	•	•
•	•	•
•	•	•
TAVU 55	TAVU 56	TAVU 57
TAVU 58	TAVU 59	TAVU 60
TAVU 61	TAVU 62	TAVU 63

Kuva 2. Spriten muistikartta

Tapa, jolla sprite-hahmoja käytetään ohjelmassa, voidaan pilkkoa useiksi osatoiminnoiksi.

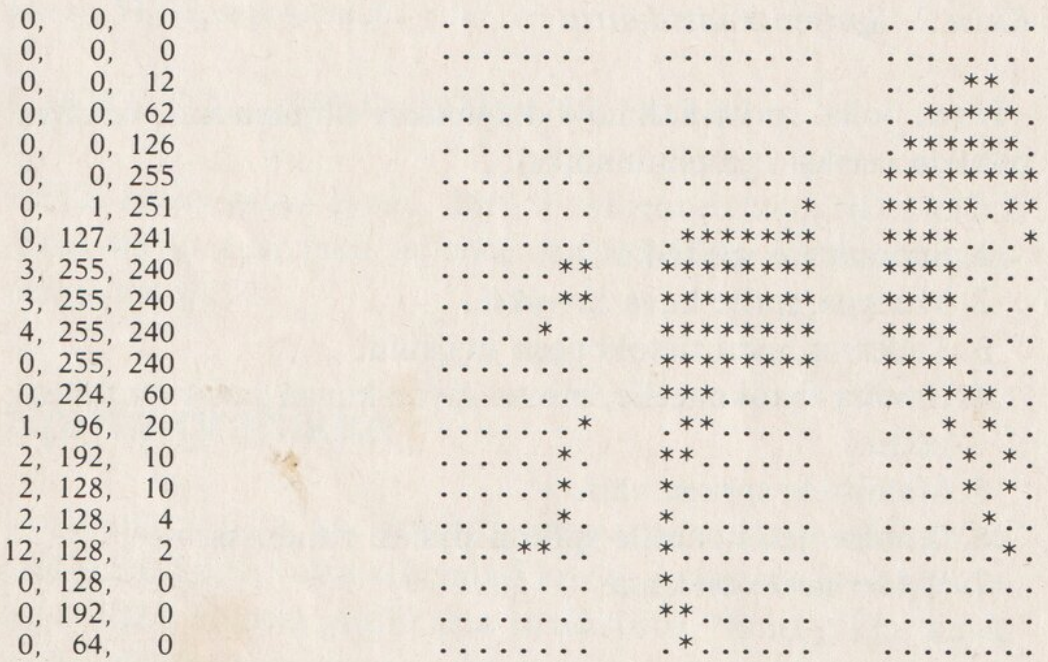
1. Suunnittele sprite-kuva.
2. Muunna sprite-kuva luvuiksi.
3. Tallenna luvut tietokoneen muistiin.
4. Ilmoita tietokoneelle, minne sprite-kuvan luvut on tallennettu.
5. Määrittele spriten väri.
6. Ilmoita tietokoneelle spriten sijainti ruudussa.
7. Kytke sprite mukaan.

Kuinka voit määritellä, mitä lukuja tarvitaan tuottamaan haluttu sprite-kuva? Ensinnäkin tarvitset sopivat välineet: arkin ruutupaperia, kynän ja kumin. Ruutupaperi, jossa on melko pienet ruudut, on hyvä, koska siinä on tilaa myös luvuille. Seuraavaksi rajaa 24 ruutua vaakasuoraan ja 21 ruutua pystysuoraan käsittävä alue arkilta. Nämä 504 ruutua vastaavat 504 pikseliä (63 tavua), jotka muodostavat spriten kuvan.

Luo sprite-kuva täyttämällä ruutuja. Tavallisesti joudut piirtämään moneen kertaan, joten paina kynää aluksi kevyesti. Kun lopulta olet tyytyväinen kuvaan, voit ruveta laskemaan lukuja, jotka tietokone tarvitsee voidakseen näyttää spriten. Ruudukko jaetaan kolmeen pystysuoraan osaan, joiden kunkin leveys on kahdeksan ruutua. Nämä vastaavat kuvassa esitettyä tavujakoa. Kutakin kahdeksan pikselin muodostamaa ryhmää

vastaavat binaariluvut saadaan korvaamalla täytetyt ruudut ykkösillä ja tyhjät ruudut nollilla. Voit muuntaa ne desimaaliluvuiksi käyttämällä taulukkoa 1.

Seuraavassa on esimerkki sprite-kuvan ja lukujen välisestä yhteydestä havainnollistettuna hevosen sprite-hahmolla. Tähti kuvaa mukaan kytkettyä pikseliä ja piste pois kytkettyä pikseliä.



Kuva 3. Hevonen sprite-kuvana.

Jos katsot kuvaa tarkasti ja tutkit esimerkkiä, huomaat, että luvut hevosen sprite-kuvassa todella määrittelevät kuvan. Esimerkiksi viimeinen muistipaikka kolmannella rivillä näyttää, että bitit 2 ja 3 ovat kytkettyinä ja muut bitit poissa. Nämä vastaavat desimaalilukuja kahdeksan ja neljä, jotka yhteensä tekevät kaksitoista.

Nyt on löydettävä tietokoneen muistista 63 muistipaikkaa, joihin spriten tiedot voidaan tallentaa. VIC-piiri jakaa 16K:n grafiikan muistialueen 256 sprite-muistilohkoon. Lohkot sisältävät oikeastaan 64 tavua, koska tietokoneen on helpompi käsitellä niitä sillä tavoin. Viimeistä tavua sprite-muistilohkosta ei

koskaan käytetä. Nämä lohkot numeroidaan 0:sta 255:een. On kuitenkin varottava, ettei mikään sprite-lohko mene päällekkäin minkään muun grafiikan tietoalueen, kuten kuvaruutumuistin, merkkigeneraattorin tai tarkkuusgrafiikan kanssa.

Minkä tahansa sprite-lohkon ensimmäisen muistipaikan osoite on aina 64:n kerrannainen (64, 128, 192 jne.). Ellet käytä enempää kuin kolmea sprite-kuvaa, kätevä paikka tallentaa spritet on kasettinauhurin käyttämä kasettipuskurialue. Se sijaitsee alueella 828—1023. Sprite-lohkoja tässä puskurissa ovat numerot 13 (832—895), 14 (896—959) ja 15 (960—1023).

Tämä rajoitus kolmeen sprite-lohkoon ei tarkoita, että myös sprite-hahmojen määrä olisi rajoitettu kolmeen. Ruudussa voi yhä olla kahdeksan sprite-hahmoa, mutta enintään kolmea tyyppiä. Jos ohjelma tarvitsee enemmän kuin kolme erilaista sprite-hahmoa, on välttämätöntä muuttaa erityisiä osoittimia ja varata RAM-muistia grafiikkalohkon lopusta tai muuttaa BASIC-ohjelman tallennusaluetta. Palataan tähän siinä yhteydessä, kun puhutaan ohjelmoitavista merkeistä ja tarkkuusgrafiikasta.

Sen jälkeen kun sprite on suunniteltu ja sen sijoittamisesta muistiin on päätetty, on aika kirjoittaa muutamia ohjelmarivejä, jotka lukevat ja tallentavat tiedot valittuihin muistialueisiin. Seuraavat rivit tallentavat hevosen kuvan määrittelevät luvut DATA-lauseista sprite-lohkoon 13.

```
100 READ SB: IF SB<0 THEN 1000
110 SS=SB*64
120 FOR I=0TO62: READ SD
130 POKE SS+I,SD: NEXT I
140 GOTO 100
150 DATA 13
160 DATA 0,0,0,0,0,0,0,0,12
170 DATA 0,0,62,0,0,126,0,0,255
180 DATA 0,1,251,0,127,241,3,255,240
190 DATA 3,255,240,4,255,240,0,255,240
200 DATA 0,224,60,1,96,20,2,192,10
210 DATA 2,128,10,2,128,4,12,128,2
220 DATA 0,128,0,0,192,0,0,64,0
999 DATA -1
1000 REM OHJELMAN ALKU
```

Lause 100 lukee ensimmäisen DATA-lauseen. Siinä on sprite-lohkon numero, ja sitä käytetään sen ensimmäisen muistipai-

kan osoitteen laskemisessa (lauseessa 110). Lauseet 120 ja 130 lukevat seuraavat 63 lukua DATA-lauseista ja tallentavat ne 63 muistipaikkaan alkaen lauseessa 110 lasketusta aloitusosoitteesta 832.

Kun 63 lukua on luettu, lause 140 määrää koneen hyppäämään takaisin lauseeseen 100, jossa se lukee luvun sprite-tietoja seuraavasta DATA-lauseesta. Jos tässä on toinen sprite-lohkon numero, ohjelma laskee aloitusosoitteen sille. Sitten se lukee ja tallentaa seuraavat 63 lukua siihen lohkoon. Jos luku on negatiivinen, ohjelma haarautuu seuraavaan osaan ohjelmaa. Negatiivinen luku on merkkinä siitä että sprite-tiedot loppuivat.

VIC-piirin pitää tietää, mitä sprite-lohkoa käytetään spriten piirtämiseen. Commodore 64 käyttää kuvaruutumuistin kahdeksaa viimeistä tavua osoittimina sprite-lohkoihin. Sprite-lohkojen osoittimet spriteille 0—7 ovat 2040—2047. Jatketaan hevosohjelmaa ilmoittamalla koneelle, että spritet 0—3 saavat hahmonsa hevosen kuvina lohkosta 13:

```
1010 FOR I=0TO3
1020 POKE 2040+I,13: NEXT I
```

Kuten huomaat, meillä voi olla enemmän kuin yksi sprite, vaikka vain yksi sprite-kuva on tallennettu muistiin.

Hevosten värit määritellään antamalla POKE-käskyillä värikoodit muistipaikkoihin 53287—53294. Nämä kontrolloivat kahdeksan spriten (0—7) värejä.

```
1030 POKE 53287,0: REM SPRITE 0 MUSTAKSI
1040 POKE 53288,1: REM SPRITE 1 VALKOISEKSI
1050 POKE 53289,9: REM SPRITE 2 RUSKEAKSI
1060 POKE 53290,12: REM SPRITE 3 KESKIHARMAAKSI
```

Kahdeksan spriteä numeroituna 0—7 kytketään mukaan asettamalla (ykköset) tai tyhjentämällä (nollaamalla) vastaavat bitit muistipaikassa 53269. Bitti 0 kytkee sprite 0:n, bitti 1 sprite 1:n jne. Kytkeäksemme esimerkiksi sprite 1:n, luku 2 (00000010) tallennetaan muistipaikkaan 53269. Taulukko 1 on jälleen hyödyllinen määrättäessä desimaalilukua, joka pitää POKElla sijoittaa rekisteriin 53269. Enintään kahdeksan spri-

teä voi olla kytkettynä samanaikaisesti. Tällä tavoin kytket mukaan neljä (0—3) spriteä:

```
1070 POKE 53269,15: REM SPRITET 0-3 PÄÄLLE
```

Monimutkaisin asia sprite-hahmojen käytössä on niiden paikkojen määrittely ruudussa. Kukin sprite käyttää sijainnin määräytymiseen ruudussa kahta muistipaikkaa ja osaa kolmannesta muistipaikasta. Ja jotta kaikki olisi vielä monimutkaisempaa, ”videokartta”, jolla spritet esitetään, on suurempi kuin näyttöruutu! (Tällä tavalla spritet voivat liikkua pehmeästi ruudun alueelle ruudun reunojen takaa.)

Sprite-kuvan vasenta yläkulmaa käytetään sen sijainnin määrittelyyn. Jotta koko sprite näkyisi ruudussa, pystysuoran sijainnin pitää olla 50 tai sitä suurempi, mutta ei yli 229. Vaakasuoran sijainnin pitää olla 24 tai suurempi, mutta ei yli 320. Sijainnin määräävät VIC-rekisterit on esitetty taulukossa 2.

TAULUKKO 2: Spriten sijaintirekisterit

Sprite	Vaakas. asema	Pystysuora asema
0	53249	53248 + bitti 0 53264:stä
1	53251	53250 + bitti 1 53264:stä
2	53253	53252 + bitti 2 53264:stä
3	53255	53254 + bitti 3 53264:stä
4	53257	53256 + bitti 4 53264:stä
5	53259	53258 + bitti 5 53264:stä
6	53261	53260 + bitti 6 53264:stä
7	53263	53262 + bitti 7 53264:stä

Koska pystysuora sijainti ei voi olla koskaan suurempi kuin 255, tarvitaan sitä varten vain yksi rekisteri spriteä kohti. Vaakasuora asema voi kuitenkin ylittää 255:n ja tarvitaan toinen muistipaikka, jotta voidaan esittää lukuja, jotka ovat suurempia kuin 255. Yksi ylimääräinen bitti riittää, joten muistipaikan

53264 bittejä käytetään ilmaisemaan, että vaakasuoraa asemaa kuvaavan rekisterin sisältöön on lisättävä 256. Tämä käy jo hieman konstikkaaksi. Käyttämällä ylimääräistä bittiä toisesta muistipaikasta saadaan käyttöön 9-bittinen binaariluku, jolla voidaan esittää lukuja aina 511:een asti. (Et kuitenkaan voi nähdä spriteä ruudussa, jos sen vaakasuora sijainti on suurempi kuin 344.) Tehdäksemme asiat kuitenkin alussa helpommiksi emme käytä ensimmäisessä esimerkissämme suurempia sijaintilukuja kuin 255. Lisää seuraavat käskyt esimerkkiohjelmaamme. Niillä sijoitat hevoset ympäri ruutua.

```
1080 POKE 53248,15: POKE 53249,60: REM SPRITE 0 SIJAINTI
1090 POKE 53250,160: POKE 53251,150: REM SPRITE 1 SIJAINTI
1100 POKE 53252,255: POKE 53253,229: REM SPRITE 2 SIJAINTI
1110 POKE 53254,100: POKE 53255,235: REM SPRITE 3 SIJAINTI
1120 POKE 53281,13: REM TAUSTA VAALEAN VIHREÄKSI
1130 PRINT"□": REM RUUDUN TYHJENNYS
```

Oljelma on nyt valmis suoritettavaksi. Osa mustasta hevosesta näkyy ruudun vasemmassa reunassa lähellä yläreunaa, valkoinen hevonen on lähellä ruudun keskustaa, ruskea hevonen on lähellä alareunaa oikealla ja osa harmaasta hevosesta näkyy alakulmassa. Yritä muutella ohjelmaa vaihtamalla värejä ja hevosten sijainteja. Koska ohjelma käyttää vaakasuoran sijainnin määritykseen vain yhtä rekisteriä, mikään hevosista ei ole ruudun oikean kolmanneksen alueella. Spriten sijoittamista sille alueelle käsitellään myöhemmin.

On mielenkiintoista kokeilla sprite-kuvien sijoittamista siten, että ne menevät osittain päällekkäin. Huomaa, että sprite 0 näkyy päällimmäisenä, sprite 1:llä on etuasema numeroihin 2—7 nähden jne. Näin ollen sprite 7 on aina kaikkien muiden alla.

Yritä listata ohjelma, kun spritet ovat vielä ruudussa. Huomaat, että teksti jää sprite-kuvien alle. Sprite-kuvien etuaseman tekstiin (tai tarkkuusgrafiikassa etualaan) nähden määrää muistipaikan 53275 sisältö. Se on yksi niistä rekistereistä, joissa kukin bitti valitsee vastaavan spriten tilan. Jos bitti 1 on asetettu ykköseksi, vastaava sprite näkyy tekstin takana. Tyhjennetty bitti määrää, että sprite on ennen tekstiä. Kokeile näppäilemällä

POKE 53275,15. Nyt kaikki spritet ovatkin tekstin takana. Jos POKElla asetetaankin arvo 3, spritet 0 ja 1 ovat tekstin takana ja 2 ja 3 tekstin edessä.

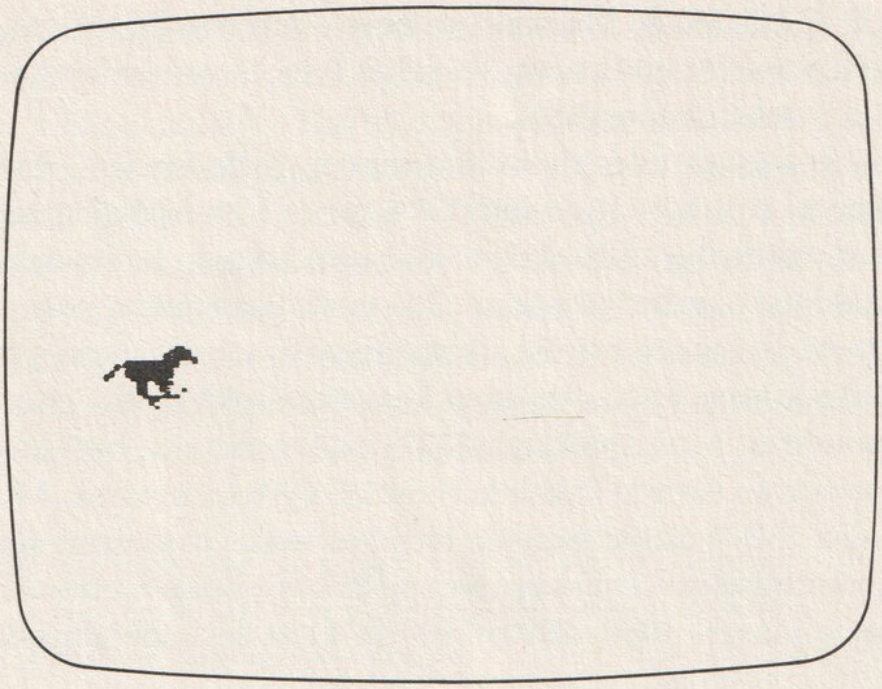
Jos etusijoista tulee ristiriitaa, saavat sprite-kuvien keskinäiset asemat etusijan. Jos esimerkiksi sprite 1 on määrätty tekstin eteen ja sprite 0 tekstin taakse, kaikkien kolmen kohtaamisessa teksti esiintyy sprite 0:n edessä. Jos sprite 0 siirretään pois, teksti häipyy. Jos sprite 1 sitten siirretään pois, teksti ilmestyy taas.

Sprite-kuvien kokoa voidaan kasvattaa sekä pysty- että vaakasuunnassa. Muistipaikassa 53271 päälle asetettu bitti kasvattaa vastaavan spriten kaksinkertaiseksi pystysuunnassa. Muistipaikassa 53277 päälle asetettu bitti kasvattaa vastaavan spriten kaksinkertaiseksi vaakasuunnassa. Yritä seuraava esimerkkiä. Muunna lauseet 1080, 1090, 1100 ja 1110 hevosohjelmassa.

```
1080 POKE 53248,30: POKE 53249,70: REM SPRITE 0 SIJAINTI
1090 POKE 53250,200: POKE 53251,70: REM SPRITE 1 SIJAINTI
1100 POKE 53252,30: POKE 53253,170: REM SPRITE 2 SIJAINTI
1110 POKE 53254,200: POKE 53255,170: REM SPRITE 3 SIJAINTI
```

Aja ohjelma, ja ruutuun tulee neljä hevosta. Anna käskyt POKE 53271,10 ja POKE 53277,12. Hevosten ulkonäkö muuttuu. Sprite 0 ei muutu, sprite 1 kaksinkertaistuu korkeussuunnassa, sprite 2 kaksinkertaistuu leveyssuunnassa ja sprite 3 kaksinkertaistuu molemmissa suunnissa.

Tähän asti olemme pitäneet asiat yksinkertaisina käyttämällä pienempiä vaakasuoria sijainteja kuin 255. Kun nyt olet kuitenkin päässyt alkuun sprite-tekniikassa, et varmaankaan halua rajoittua tähän alueeseen. Seuraava ohjelma havainnollistaa, kuinka spriteä voidaan liikuttaa yli 255:n rajan vaakasuoraan. Se näyttää myös, kuinka voidaan tehdä animaatioita muuttamalla spriten muotoa samalla kun se liikkuu yli ruudun. Ohjelmassa määritellään kahdeksan sprite-hahmoa kuvaten hevosen laukan eri vaiheet oikeassa järjestyksessä. Joka kerran kun spriteä siirretään, muutetaan sprite 0:n tietolohkon osoitinta, muistipaikkaa 2040. Ohjelmassa testataan, onko viimeinen lohko jo käytössä. Jos se on, tietolohkon osoitin muutetaan taas ensimmäiseen lohkoon.



Spriten liikuttaminen aseman 255 ohi oikealle vaatii muistipaikan 53264 käyttöä. Tämä paikka pitää kirjaa siitä, mitkä spritet ovat näytön oikealla puoliskolla. Jos näytettävän spriten numeroa vastaava bitti on kytkettynä (ykkönen), tietokone lisää 256 vastaavan spriten vaakasuoran aseman kertovan rekisterin sisältöön ja käyttää tulosta spriten sijoittamisessa näyttöön. Jos aiotaan käyttää koko vaakasuoraa aluetta, ohjelman pitää huolehtia siitä, että bitti muistipaikassa 53264 on nolla, kun sijainti on alle 256, ja ykkönen, kun sijainti on 256 tai sen yli. Lauseet 1090—1100 huolehtivat sprite 0:n sijainnista.

Tutkitaanpa tarkasti, kuinka sprite 0 sijoitetaan näyttöön. Pystysuora asema (muistipaikka 53249) pidetään vakiona. Muuttuja X vastaa vaakasuoraa asemaa. Lauseessa 1090 vaakasuoran aseman ja luvun 255 välille suoritetaan AND-operaatio ja tulos sijoitetaan muistipaikkaan 53248. AND 255 takaa vaakasuoran aseman arvon sellaiseksi, että muistipaikkaan asetettava luku on välillä 0—255. Jos yritetään asettaa luku, joka on tämän välin ulkopuolella, tietokone vastaa virheilmoituksella. Jos sijainti on pienempi kuin 256, tulos operaatiosta X AND 255 on sama kuin X:n sisältö. Jos taas asema on suurempi kuin

256, tulos on 255-X. Sitten ohjelma tarkistaa, onko sijainti suurempi kuin 255. Jos se on, bitti 0 rekisterissä 53264 asetetaan ykköseksi ja ohjelma hyppää lauseeseen 1110. Seuraava lause (1100) suoritetaan vain siinä tapauksessa, että spriten vaakasuora asema on pienempi kuin 256. Tämä lause tyhjentää muistipaikan 53264 bitin 0 nollaksi.

"GALLOP"-ohjelma

```
10 POKE 52,62:POKE 56,62: CLR: REM SIIRRETÄÄN BASICIN YLÄRAJAA
20 READ SB: IF SB<0 THEN 1000: REM LUETAAN SPRITE-TIETOJEN TALLETUSALUE
30 SS=SB*64 :REM LASKETAAN ALUEEN OSOITE
40 FOR I=0 TO 62
50 :READ SD :L=L+1
60 :POKE SS+I,SD
70 NEXT I
80 GOTO 20
90 REM SPRITE-TIEDOT
100 DATA 248,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0
110 DATA 30,0,0,62,0,124,255,1,255,243,3,255,240,6,255,240,12
120 DATA 255,224,0,239,224,1,231,192,1,193,192,3,129,64,2,130,64
130 DATA 4,140,64,0,0,64,0,0,64,0,0,32,249,0,0,0,0
140 DATA 0,0,6,0,96,0,0,240,0,1,240,0,1,248,0,255,200
150 DATA 3,255,192,2,255,128,4,255,128,12,255,128,8,127,0,0,55
160 DATA 128,0,51,64,0,51,32,0,49,64,0,31,192,0,12,0,0
170 DATA 8,0,0,4,0,0,0,0,250,0,0,0,0,0,0,0,0
180 DATA 96,0,0,240,0,1,240,0,1,248,0,199,200,1,255,128,7
190 DATA 255,128,15,255,128,25,255,128,49,255,128,1,223,192,0,194,64
200 DATA 0,193,64,0,229,64,0,147,64,0,136,64,0,68,0,0,64
210 DATA 0,0,32,0,251,0,0,0,0,0,0,0,0,0,0,30
220 DATA 0,0,62,0,0,127,0,0,251,0,63,249,1,255,240,3,255
230 DATA 240,7,127,240,4,127,240,0,115,248,0,240,8,0,176,8,1
240 DATA 160,12,1,32,10,1,32,1,1,16,0,1,16,0,0,136,0
250 DATA 252,0,0,0,0,0,0,0,0,12,0,0,62,0,0,126,0
260 DATA 0,254,0,1,251,0,127,241,3,255,240,3,255,240,4,255,240
270 DATA 0,255,240,0,224,60,1,96,20,2,192,18,2,128,10,2,128
280 DATA 10,12,128,4,0,128,2,0,192,0,0,64,0,253,0,0,0
290 DATA 0,0,0,0,0,0,0,0,0,0,0,28,0,0,126,0,0
300 DATA 254,0,115,251,3,255,243,6,255,240,12,255,240,0,255,224,1
310 DATA 255,224,3,193,240,7,128,72,31,0,68,2,0,66,2,0,33
320 DATA 2,0,32,2,0,32,2,0,16,254,0,0,0,0,0,0,0
330 DATA 0,0,0,0,0,0,0,28,0,0,62,0,0,126,0,255,251
340 DATA 3,255,241,6,255,240,12,255,240,0,255,224,1,199,224,3,193
350 DATA 240,3,0,144,14,0,136,2,0,132,4,0,130,12,1,1,0
360 DATA 1,0,0,0,128,255,0,0,0,0,0,0,0,0,0,0,0
370 DATA 12,0,0,28,0,0,62,0,120,254,3,255,247,6,255,241,12
380 DATA 255,224,0,255,224,1,255,224,1,195,224,3,1,96,2,1,32
390 DATA 4,2,32,8,2,16,0,4,16,0,4,16,0,4,8,0,4
400 DATA 8,-1
1000 PRINT"□"
1010 POKE 53287,9: REM SPRITE 0 RUSKEAKSI
1020 POKE 53269,1: REM SPRITE 0 PÄÄLLE
1030 POKE 53249,150: REM SPRITE 0 - PYSTYSUORA SIJAINTI
1040 POKE 53277,1: REM LAAJENNUS VAAKASUUNNASSA
1050 POKE 53281,13: REM TAUSTA VAALEAN VIHREÄKSI
1060 SD=248: POKE 2040,SD: REM SPRITE-LOHKON VALINTA
1070 FOR X=0 TO 345 STEP 6: REM LIIKUTETAAN SPRITEÄ NÄYTÖN YLI
1080 :FOR T=0 TO 50: NEXT T: REM VIIVYTYSSILMUKKA
1090 :POKE 53248,X AND 255: IF X>255 THEN POKE 53264,1: GOTO 1110
1100 :POKE 53264,0
1110 :SD=SD+1: IF SD=256 THEN SD=248: REM VAIHDETAAN MUOTOA
1120 :POKE 2040,SD
1130 NEXT X
1140 GOTO 1070
```

Yhteenveto sprite-rekistereistä

Sprite-rekisterit voidaan jakaa kahteen tyyppiin. Ensimmäinen rekisterityyppi on sellainen, johon tallennetaan tietoa vain yhdestä spritestä (yksi sprite/rekisteri). Ne on lueteltu taulukossa 3.

TAULUKKO 3: Kahdeksan bitin sprite-rekisterit

Sprite	Vaakasuora sijainti (Bitit 0—7)	Pystysuora sijainti	Väri
0	53248	53249	53287
1	53250	53251	53288
2	53252	53253	53289
3	53254	53255	53290
4	53256	53257	53291
5	53258	53259	53292
6	53260	53261	53293
7	53262	53263	53294

Toinen tyyppi käyttää yhtä rekisteriä kaikkien kahdeksan spriten tietoja varten (yksi sprite/bitti). Kukin bitti säätelee vastaavan spriten tilaa. Nämä rekisterit on lueteltu taulukossa 4.

TAULUKKO 4: Yhden bitin sprite-rekisterit

Rekisteri	Toiminta
53264	Vaakasuora sijainti suurempi kuin 255
53269	Spritet päälle ja pois
53271	Pystysuora suurentaminen
53275	Etusija ennen merkkejä
53277	Vaakasuora suurentaminen

Haluttu bitti asetetaan ykköseksi suorittamalla OR-operaatio rekisterin vanhan sisällön ja sopivan luvun välillä. Bitti tyhjennetään (asetetaan nollaksi) suorittamalla AND-operaatio rekisterin vanhan sisällön ja bitin päälle asettamisessa käytetyn luvun komplementtiluvun kanssa. Bitin asettamisen yleinen muoto on

```
POKE rekisteri, PEEK(rekisteri) OR B2
```

Bitin tyhjennyksen yleinen muoto on

POKE rekisteri, PEEK(rekisteri) AND NOT B2

Näissä lauseissa rekisteri tarkoittaa jotakin yhden bitin sprite-rekisteriä ja B2-muuttujan sisältö riippuu spriten numerosta (katso kuvaa 1).

Sprite-hahmojen suunnittelu ja kirjoittaminen tietokoneeseen saattaa olla aika työlästä. Spriten luontiohjelma voi helpottaa paljon työtä ja samalla se antaa tekijälle suoran visuaalisen palautteen jo spriteä luotaessa. Seuraava ohjelma on spriten luontiohjelma (Sprite-editori), jonka avulla voit luoda spriten, nähdä miltä se näyttää ruudussa, ja sitten listata spriten luomiseksi ohjelmassa tarvittavat käskyt. Ohjelma jakaa näytön kolmeen alueeseen. Suuri 24 × 21 ruudukko on spriten suunnittelua varten. Kun sprite on suunniteltu, se ilmestyy rajatulle alueelle näytön oikeaan yläkulmaan. Laajennetut spritet näkyvät rajatulla alueella alempana oikealla. Kursorin moodin ilmaisee kirjain. Piirtämismoodissa D (= Draw) pikselit kääntyvät päälle kursorin liikkuessa ruudukossa. Pyyhkimismoodissa E (= Erase) pikselit kääntyvät pois päältä, kun kursori liikkuu ruudukossa. Normaalimoodissa N kursorin liikkeillä ei ole vaikutusta pikseleihin. Kun sprite on luotu, käskyt ja DATA-lauseet, jotka ohjelmassa tarvitaan spriten luomiseen, voidaan listata näyttöön tai kirjoittimelle. Spriten luontiohjelman käskyt on lueteltu seuraavassa taulukossa.

TAULUKKO 5: Spriten luontiohjelman käskyt

Käsky	Toiminta
*	Pikseli päälle kursorin kohdalla
välilyönti	Pikseli pois kursorin kohdalla
kursori vas.	Siirrä kursoria vasemmalle
kursori oik.	Siirrä kursoria oikealle
kursori ylös	Siirrä kursoria ylöspäin

kursori alas	Siirrä kursoria alaspäin
return	Siirrä kursori vasempaan reunaan
home	Siirrä kursori vasempaan yläkulmaan
clear	Tyhjennä ruudukko
X	Kasvata spriteä vaakasuoraan
Y	Kasvata spriteä pystysuoraan
B	Kasvata spriteä molempiin suuntiin
L	Listaa komennot
F1	Valitse taustan väri
F3	Valitse spriten väri
F7	Valitse kursorin moodi
Q	Lopeta ohjelman suoritus

"SPRITE"-ohjelma

```
5 CS=2:BO=1:PL=0: REM VALKOINEN TAUSTA, PUNAINEN SPRITE
7 PM(0)=14:PM(1)=4:PM(2)=5
10 POKE 53280,6: POKE 53281,1: REM SININEN REUNUS, VALKOINEN TAUSTA NÄYTÖLLÄ
20 SC=1024:SP=832:CR=55296: REM KUVARUUTU- JA VÄRI-RAM
30 REM VÄRI-RAM SYAANIKSI
40 PRINT"[illegible]": FOR I=0 TO 999: POKE CR+I,3: NEXT I
50 PRINT"[illegible]      SPRITE-EDITORI / PAUL SCHATZ      [illegible]"
60 REM RAJATAAN TYÖSKENTELYALUE RUUDULLA
70 POKE SC+81,112: POKE SC+106,110: POKE SC+961,109: POKE SC+986,125
80 FOR I=0 TO 23: POKE SC+82+I,114: POKE SC+962+I,113: NEXT I
90 FOR I=0 TO 20: POKE SC+121+(40*I),107: POKE SC+146+(40*I),115: NEXT I
100 REM TYÖALUE PUNAISEKSI
110 FOR I=0 TO 20: FOR J=0 TO 23: POKE CR+122+(40*I)+J,2: NEXT J,I
115 GOSUB 1700: REM VÄRITETÄÄN SPRITE-ALUEET
120 REM RAJATAAN SPRITE-ALUE
130 POKE SC+110,112: POKE SC+115,110: POKE SC+270,109: POKE SC+275,125
140 FOR I=0 TO 3: POKE SC+111+I,64: POKE SC+271+I,64: NEXT I
150 FOR I=0 TO 2: POKE SC+150+(40*I),93: POKE SC+155+(40*I),93: NEXT I
160 FOR I=0 TO 2: FOR J=0 TO 3: POKE SC+151+(40*I)+J,160: NEXT J,I
170 PRINT"[illegible]S[illegible]P[illegible]R[illegible]I[illegible]T[illegible]E"
200 REM ALUE LAAJENNETULLE SPRITELLE
210 POKE SC+710,112: POKE SC+718,110: POKE SC+990,109: POKE SC+998,125
220 FOR I=0 TO 6: POKE SC+711+I 64: POKE SC+991+I,64: NEXT I
230 FOR I=0 TO 5: POKE SC+750+(40*I),93: POKE SC+758+(40*I),93: NEXT I
240 FOR I=0 TO 5: FOR J=0 TO 6: POKE SC+751+(40*I)+J,160: NEXT J,I
250 PRINT"[illegible]";
260 PRINT"[illegible]S[illegible]U[illegible]L[illegible]R[illegible]E[illegible]N[illegible]"
300 REM KÄSKYTAULUKKO
305 PRINT"[illegible]KURSORI:":PRINT
310 PRINT"[illegible]KASKYT:"
320 PRINT"[illegible]L[illegible]=LISTAA"
330 PRINT"[illegible]Q[illegible]=LOPPU"
340 PRINT"[illegible]X[illegible]=SUUR. X"
350 PRINT"[illegible]Y[illegible]=SUUR. Y"
360 PRINT"[illegible]B[illegible]=SUUR. XY"
370 PRINT"[illegible]F1 F3 F7[illegible]"
380 POKE SC+318,PM(PL): POKE CR+318,10
400 REM SIIRRETÄÄN SPRITE-TIEDOT NAYTTÖÖN
410 GOSUB 1000
420 REM LAITETAAN SPRITE LAATIKKOON
430 GOSUB 1200
440 REM KURSORI KOTIIN JA PÄÄLLE
```

```
450 PRINT"[CLR]": POKE 211,2: POKE 204,0
500 REM KASKYJEN LUKEMINEN
510 GET A$: IF A$="" THEN 510
520 IF A$="Q" THEN POKE 53269,0: PRINT"[CLR]": END: REM LOPETUS
530 IF A$="L" THEN GOSUB 2000: GOTO 40
540 IF A$="X" THEN GOSUB 1400
550 IF A$="Y" THEN GOSUB 1500
560 IF A$="B" THEN GOSUB 1600
570 IF A$="[F1]" THEN GOSUB 1300: GOTO 450
580 IF A$="[F3]" THEN GOSUB 1100: GOSUB 900: GOTO 450
590 IF A$="[F5]" THEN GOSUB 3000
600 IF A$="[F7]" THEN GOSUB 3100
610 IF A$="[F2]" THEN GOSUB 3200
620 IF A$="[F4]" THEN GOSUB 3300
630 IF A$=CHR$(13) THEN GOSUB 3400
640 IF A$=" " THEN GOSUB 3500: GOSUB 3300
650 IF A$="*" THEN GOSUB 3600: GOSUB 3300
660 IF A$="[F6]" THEN BG=BG+1: GOSUB 1700
670 IF A$="[F8]" THEN CS=CS+1: GOSUB 1800
680 IF A$="[HOME]" THEN PL=PL+1: IF PL>2 THEN PL=0
690 POKE SC+318,PM(PL)
700 GOTO 510
800 REM ---------- ALIOHJELMAT ----------
900 REM NAYTON TYHJENNYS
905 POKE 204,1: REM KURSORI POIS PAALTA
910 FOR I=0 TO 20: FOR J=0 TO 23: POKE SC+122+(40*I)+J,32: NEXT J,I
920 RETURN
1000 REM SPRITEN SIIRTO NAYTOLLE
1005 POKE 204,1: REM KURSORI POIS PAALTA
1010 FOR I=0 TO 20: FOR J=0 TO 2: FOR K= 7 TO 0 STEP-1
1020 :CH=32: IF (PEEK(SP+(3*I)+J)AND(2^K)) THEN CH=204
1030 :POKE (SC+122+(40*I)+(8*J)+(7-K)),CH
1040 NEXT K,J,I: RETURN
1100 REM SPRITEN TYHJENNYS
1110 FOR I=0 TO 62: POKE 832+I,0: NEXT I: RETURN
1200 REM SPRITE RUUDULLE
1210 POKE 2040,13: REM SPRITE-TIEDOT MUISTIPAIKOISSA 832-894 (64*13=832)
1220 POKE 53248,20: POKE 53264,1: REM SPRITE 0 X-KOORDINAATTI
1230 POKE 53249,75: REM SPRITE 0 Y-KOORDINAATTI
1240 POKE 53287,CS: REM SPRITE 0 VARI
1250 POKE 53269,1: REM SPRITE 0 PAALLE
1260 RETURN
1300 REM KURSORIN SIIRTO
1310 LO=SC+(40*PEEK(214))+PEEK(211): REM SELVITA KURSORIN SIJAINTI
1320 CH=32: POKE 204,1: REM KURSORI POIS PAALTA
1330 IF (PEEK(LO)AND127)<>32 THEN CH=204
1340 POKE LO,CH: RETURN
1400 REM SUURENNUS X-SUUNNASSA
1410 POKE 2041,13: REM SPRITE-TIETO 832-894
1420 POKE 53277,2: REM SPRITE 1 X-SUURENNUS
1430 POKE 53271,0: REM EI Y-SUURENNUSTA
1440 POKE 53250,20: POKE 53264,PEEK(53264) OR 2: REM SPRITE 1 X-KOORDINAATTI
1450 POKE 53251,197: REM SPRITE 1 Y-KOORDINAATTI
1460 POKE 53288,CS: REM SPRITE 1 VARI
1470 POKE 53269,3: REM SPRITET 0 JA 1 PAALLE (2+1=3)
1480 RETURN
1500 REM SUURENNUS Y-SUUNNASSA
1510 POKE 2041,13: REM SPRITE-TIETO 832-894
1520 POKE 53277,0: REM EI X-SUURENNUSTA
1530 POKE 53271,2: REM Y-SUURENNUS
1540 POKE 53250,20: POKE 53264,PEEK(53264) OR 2: REM SPRITE 1 X-KOORDINAATTI
1550 POKE 53251,197: REM SPRITE 1 Y-KOORDINAATTI
1560 POKE 53288,CS: REM SPRITE 1 VARI
1570 POKE 53269,3: REM SPRITET 0 JA 1 PAALLE (2+1=3)
1580 RETURN
1600 REM SUURENNUS MOLEMPIIN SUUNTIIN
1610 POKE 2041,13: REM SPRITE-TIETO 832-894
1620 POKE 53277,2: REM X-SUURENNUS
1630 POKE 53271,2: REM Y-SUURENNUS
1640 POKE 53250,20: POKE 53264,PEEK(53264) OR 2: REM SPRITE 1 X-KOORDINAATTI
1650 POKE 53251,197: REM SPRITE 1 Y-KOORDINAATTI
1660 POKE 53288,CS: REM SPRITE 1 VARI
```

```
1670 POKE 53269,3: REM SPRITET 0 JA 1 PÄÄLLE (2+1=3)
1680 RETURN
1700 REM VÄRITETÄÄN SPRITE-ALUEITTEN TAUSTAT
1710 IF BG>15 THEN BG=0
1720 FOR I=0 TO 2: FOR J=0 TO 3: POKE CR+151+(40*I)+J,BG: NEXT J,I
1730 FOR I=0 TO 5: FOR J=0 TO 6: POKE CR+751+(40*I)+J,BG: NEXT J,I: RETURN
1800 REM VÄRI SPRITEILLE
1810 IF CS>15 THEN CS=0
1820 POKE 53287,CS: POKE 53288,CS: RETURN
2000 REM TULOSTUSRUTIINI
2010 POKE 204,1: POKE 53269,0: REM SPRITET POIS PÄÄLTÄ
2020 INPUT"■■TULOSTETTAVAN SPRITEN NRO";S: IF S>7 OR S<0 THEN 2020
2030 INPUT"TIETOLOHKON NRO";B
2040 INPUT"■PRINTTERILLE VAI ■NÄYTTÖÖN";B$:PRINT"■"
2050 OD=3: IF B$="P" THEN OD=4
2060 OPEN 1,OD: IF OD=4 THEN GOSUB 2500: REM PRINTTERITULOSTUKSEEN
2070 VC=53248: BP=2040
2080 PRINT#1," 1000 POKE 53269,PEEK(53269)OR";2↑S;
2085 PRINT#1,":REM SPRITE";S;" PÄÄLLE";CHR$(13);
2090 PRINT#1," 1010 POKE";BP+S;",";B;
2095 PRINT#1,":REM SPRITE";S;"ALUEELLA";B*64;"-";B*64+62;CHR$(13);
2100 PRINT#1," 1020 FOR I=0 TO 62: READ DT: POKE";B*64;"+I,DT: NEXT I";CHR$(13);
2105 PRINT#1," 1030 IF ABS(X/256) THEN POKE 53264,PEEK(53264)OR";
2107 PRINT#1,2↑S;":X=X-256";CHR$(13);
2110 PRINT#1," 1040 POKE";VC+2*S;",X: REM SPRITE";S;" X-KOORDINAATTI";CHR$(13);
2120 PRINT#1," 1050 POKE";VC+1+2*S;
2125 PRINT#1,",Y: REM SPRITE";S;" Y-KOORDINAATTI";CHR$(13);
2130 PRINT#1," 1060 POKE";53287+S;",";CS;": REM SPRITE";S;" VÄRI";CHR$(13);
2140 PRINT#1," 1070 POKE 53277,PEEK(53277)OR";2↑S;
2145 PRINT#1,": REM SPRITE";S;" X-SUURENNUS";CHR$(13);
2150 PRINT#1," 1080 POKE 53271,PEEK(53271)OR";2↑S;
2155 PRINT#1,": REM SPRITE";S;" Y-SUURENNUS";CHR$(13);
2160 IF OD=3 THEN PRINT#1,CHR$(13);"■ PAINA JOTAIN NAPPIA ■": GOSUB 2300
2170 FOR I=0 TO 6
2180 PRINT#1,1090+I*10;"DATA";
2190 FOR J=0 TO 7: PRINT#1,PEEK(SP+I*9+J);",";:NEXT J
2200 PRINT#1,PEEK(SP+I*9+8);CHR$(13): NEXT I
2210 PRINT#1: CLOSE 1: PRINT: PRINT"■ PAINA JOTAIN NAPPIA ■"
2300 GET C$: IF C$="" THEN 2300
2310 RETURN
2500 REM SPRITE-KUVAN PIIRTÄMISRUTIINI
2505 OPEN 4,4,4
2510 PRINT#4,"SPRITE-KUVA";CHR$(13);CHR$(13);
2520 PRINT#4,".........................";CHR$(13);
2530 FOR I=0 TO 20: LN$=""
2540 :FOR J=0 TO 2
2545 ::FOR K=7 TO 0 STEP-1
2550 :::CA$=" ":IF (PEEK(SP+(3*I)+J)AND(2↑K)) THEN CA$="*"
2555 :::LN$=LN$+CA$
2560 ::NEXT K
2565 :NEXT J
2570 :PRINT#4,".";LN$;".";CHR$(13);
2575 NEXT I
2580 PRINT#4,CHR$(13);CHR$(13);
2590 RETURN
3000 REM KURSORI YLÖSPÄIN
3010 IF PEEK(214)<4 THEN RETURN
3012 IF PL=2 THEN GOSUB 3500
3014 IF PL=1 THEN GOSUB 3600
3020 GOSUB 1300: CM=PEEK(211): RO=PEEK(209)-40
3025 IF RO<0 THEN RO=256+RO: POKE 210,PEEK(210)-1
3030 POKE 209,RO: POKE 211,CM: POKE 204,0: REM KURSORIN SIIRTO JA VÄLKE PÄÄLLE
3040 POKE 214,PEEK(214)-1: POKE 201,PEEK(201)-1: RETURN
3100 REM KURSORI ALASPÄIN
3110 IF PEEK(214)>22 THEN RETURN
3112 IF PL=2 THEN GOSUB 3500
3114 IF PL=1 THEN GOSUB 3600
3120 GOSUB 1300: CM=PEEK(211): RO=PEEK(209)+40
3125 IF RO>255 THEN RO=RO-256: POKE 210,PEEK(210)+1
3130 POKE 209,RO: POKE 211,CM: POKE 204,0: REM KURSORIN SIIRTO JA VÄLKE PÄÄLLE
3140 POKE 214,PEEK(214)+1: POKE 201,PEEK(201)+1: RETURN
3200 REM KURSORI VASEMMALLE
```

```
3210 IF PEEK(211)<3 THEN RETURN
3212 IF PL=2 THEN GOSUB 3500
3214 IF PL=1 THEN GOSUB 3600
3220 GOSUB 1300
3230 POKE 211,PEEK(211)-1:POKE 204,0: REM KURSORIN SIIRTO JA VÄLKE PÄÄLLE
3240 RETURN
3300 REM KURSORI OIKEALLE
3310 IF PEEK(211)>24 THEN RETURN
3312 IF PL=2 THEN GOSUB 3500
3314 IF PL=1 THEN GOSUB 3600
3320 GOSUB 1300
3330 POKE 211,PEEK(211)+1:POKE 204,0: REM KURSORIN SIIRTO JA VÄLKE PÄÄLLE
3340 RETURN
3400 GOSUB 1300
3410 POKE 211,2: POKE 204,0
3420 RETURN
3500 REM BITIN TYHJENNYS
3510 LO=SC+(40*PEEK(214))+PEEK(211)
3520 POKE LO,32
3530 J=INT((PEEK(211)-2)/8): K=9-PEEK(211)+(J*8)
3540 I=PEEK(214)-3
3550 POKE (SP+(3*I)+J),PEEK(SP+(3*I)+J)AND(255-(2↑K))
3570 RETURN
3600 REM BITTI PÄÄLLE
3610 LO=SC+(40*PEEK(214))+PEEK(211)
3620 POKE LO,204
3630 J=INT((PEEK(211)-2)/8): K=9-PEEK(211)+(J*8)
3640 I=PEEK(214)-3
3650 POKE (SP+(3*I)+J),PEEK(SP+(3*I)+J)OR(2↑K)
3670 RETURN
```

Ohjelmoitavat merkit

Commodore 64:ssä on kaksi sisäänrakennettua merkkijoukkoa, jotka kumpikin sisältävät 256 merkkiä. On mahdollista vaihtaa näiden merkkijoukkojen välillä painamalla samanaikaisesti Commodore- ja SHIFT-näppäintä. Kun Commodore 64 kytketään toimintaan, isot kirjaimet ja grafiikkamerkkejä sisältävä merkkijoukko ovat käytössä. Toisessa merkkijoukossa on pienet ja isot kirjaimet. Käytössä olevan merkkijoukon määrää bittien 1—3 sisältö rekisterissä 53272. Yritäpä seuraavaa koetta. Kytke kone päälle ja näppäile

```
PRINT PEEK (53272)
```

Saatava arvo on 21. Paina nyt SHIFT- ja Commodore-näppäimiä vaihtaaksesi toiseen merkkijoukkoon. PEEKkaa rekisteriä 53272 uudelleen. Nyt arvo onkin 23.

Nähdäksesi, mitkä merkit sisältyvät kuhunkin merkkijoukkoon, kirjoita seuraava ohjelma.

```
10 POKE 53281,1: REM TAUSTA VALKOISEKSI
20 SC=1024: CR=55296: REM KUVARUUTU- JA VARIMUISTIN OSOITTEET
30 PRINT"■": REM RUUDUN TYHJENNYS JA KIRJOITUSVÄRI MUSTAKSI
40 FOR I = 0 TO 255
50 POKE SC+80+2*I,I: POKE CR+80+2*I,0: NEXT I
60 POKE 53272,21: REM GRAFIIKKAMERKKIJOUKKO PÄÄLLE
70 PRINT"■GRAFIIKKAMERKKIJOUKKO"
80 GET A$: IF A$="" THEN 80
90 POKE 53272,23: REM PIENET/ISOT KIRJAIMET SIS. MERKKIJOUKKO PÄÄLLE
100 PRINT"■ISOT JA PIENET KIRJAIMET"
110 GET A$: IF A$="" THEN 110
120 GOTO 60
```

Kaikki 256 merkkiä tulostuvat näyttöön. Mitä tahansa näppäintä painamalla voit vaihtaa merkkijoukkoa.

Näiden kahden merkkijoukon lisäksi voit määritellä täysin uuden merkkijoukon. Täten voidaan luoda persoonallisia merkkijoukkoja esimerkiksi vieraita kieliä, animaatioita tai grafiikkaa varten. Muunneltujen merkkien kuvaukset tallennetaan RAMiin. 256 merkkiä tarvitsevat 2048 muistipaikkaa. Bittien 1—3 sisältö muistipaikassa 53272 toimii osoittimena merkkijoukon sisältämän 2K RAM -lohkon alkuun. Muistipaikassa 53272 olevat bitit 4—7 osoittavat kuvaruutumuistin alkuun. Siksi on tärkeätä voida muuttaa vain bittejä 1—3 kajoamatta bitteihin 4—7. Tämä voidaan tehdä lauseella

```
POKE 53272, (PEEK(53272) AND 240) OR CS
```

Seuraavassa taulukossa on annettu sopivia arvoja CS-muuttujalle. Arvoja, jotka aiheuttavat ristiriitoja BASICin tai käyttöjärjestelmän kanssa, ei ole otettu mukaan.

TAULUKKO 6: Merkkijoukon osoitin

Merkkijoukon alku		CS
4096	(Isot kirj./grafiikkamerkit)	4
6144	(Pienet/isot kirjaimet)	6
8192		8
10240		10
12288		12
14336		14

Huomaa, että sisäänrakennettujen merkkijoukkojen osoitteet ovat 4096 ja 6144. Näitä alueita ei kuitenkaan voida käyttää määriteltyjen merkkien tallennukseen, mikä johtuu tavasta, jolla Commodore 64 lukee merkkimuistia. Itse asiassa näillä alueilla ei ole lainkaan merkkijoukkojen kuvauksia. Juuri ennen kun tietokone on valmis näyttämään merkin ruudussa, kytkeytyy ROM-muisti, josta videopiiri luulee löytävänsä merkkijoukon osoitteesta 4096. Todellisuudessa ROM, jossa merkkien kuvaukset ovat, sijaitsee kuitenkin osoitteissa 53248—57343.

Jotta oppisi määrittelemään uusia merkkijoukkoja, on hyödyllistä ensin tutkia, kuinka merkki luodaan. Commodore 64:ssä kukin merkki muodostuu pistejoukosta, joka on järjestetty 8×8-matriisiin. Esimerkiksi merkki 0 eli @ on kuvassa 4 esitetty pikselien yhdistelmä.

```
60   . . * * * * . .
102  . * * . . * * .
110  . * * . * * * .
110  . * * . * * * .
96   . * * . . . . .
98   . * * . . . * .
60   . . * * * * . .
0    . . . . . . . .
```

Kuva 4. CHR$(0):n (" @ ") pikselikoostumus

Tähti (*) esittää mukaan kytkettyä pikseliä ja piste (.) poissa olevaa pikseliä. Tietokone tallentaa tämän kuvan kahdeksaan peräkkäiseen muistipaikkaan. Kukin luku kuvaa yhden rivin pistekoostumuksen, ja luvut on järjestetty ylimmästä rivistä alaspäin. Kutakin riviä kuvaavat luvut on johdettu luvun 8-bittisen binaariesityksen pohjalta. Kahdeksan lukua, jotka tarvitaan kuvaamaan @ merkkiä ovat 60, 102, 110, 110, 96, 98, 60 ja 0. Nämä luvut tallennetaan kahdeksaan peräkkäiseen muistipaikkaan. Kun tietokonetta käsketään näyttämään merkki numero nolla, näyttöpiiri ottaa kahdeksan lukua niistä muistipaikoista, jotka on varattu merkin 0 säilyttämiseen, ja muodostaa niiden perusteella ruutuun @ -merkin (=CHR(0)). Kun

konetta käsketään näyttämään kirjain A, käytetään toista kahdeksan luvun ryhmää.

Kun määritellään uusi merkki, on ensimmäinen askel muodostaa siitä pistekuvaus 8 × 8 ruudukkoon. Hanki ruutupaperia ja rajaa siitä 8 × 8 ruudun alue. Tee merkki mustaamalla osa ruuduista. Kukin rivi voidaan sitten muuttaa binaariluvuksi korvaamalla mustattu ruutu ykkösellä ja tyhjä ruutu nollalla. Binaarilukuja vastaavat desimaaliluvut löytyvät taas taulukon 1 avulla.

Kun olet tyytyväinen suunnittelemaasi merkkijoukkoon, olet valmis aloittamaan ohjelmoinnin. Ensinnäkin sinun täytyy varata muistitila merkkijoukolle, jotta voit suojata sen BASIC-järjestelmältä. Tämä tehdään siirtämällä muistin loppuosoitetta ja merkkijonojen alun osoitetta. Kirjoita käsky

```
10 POKE 52,56:POKE 56,56:CLR
```

Edellä oleva käsky ei salli BASICin käyttää muistipaikkoja osoitteen 14336 yläpuolelta. Osoitteissa 14336—16383 sijaitseva merkkijoukko ei pääse sotkeentumaan.

On hyvä idea siirtää merkit merkkigeneraattorin ROM-alueelta varatulle merkkialueelle. Se varmistaa, että jokaisesta merkistä on kuvaus. On helpointa siirtää 256 peräkkäistä merkkiä. Jotta tietokone voisi päästä lukemaan merkki-ROMia muuta kuin merkkien näyttämistä varten, on välttämätöntä kytkeä näppäimistö pois ja merkki-ROM päälle. Seuraava koodi siirtää koko grafiikkamerkkijoukon uudelle alueelle.

```
10 POKE 52,56: POKE 56,56: CLR
20 POKE 56334,PEEK(56334)AND254
30 POKE 1,PEEK(1)AND251
40 FOR I=0 TO 2047
50 POKE I+14336,PEEK(53248+I)
60 NEXT I
70 POKE 1,PEEK(1)OR4
80 POKE 56334,PEEK(56334)OR1
```

Lauseet 20 ja 30 avaavat pääsyn merkki-ROMin alueelle. Lauseet 70 ja 80 palauttavat tietokoneen entiseen tilaansa.

PEEK-funktion arvo määrittelee, mitkä merkit siirretään. Taulukossa 7 esitetään merkkijoukkojen alkuosoitteet.

On mahdollista siirtää myös valikoiden merkkiryhmiä tai yksittäisiä merkkejä. Esimerkiksi seuraava koodi rakentaa uuden merkkijoukon siten, että merkit 0—127 ovat isoja/pieniä kirjaimia, merkit 128—191 grafiikkamerkkejä ja merkit 192—255 käänteisiä grafiikkamerkkejä.

```
10 POKE 52,56: POKE 56,56: CLR
20 POKE 56334,PEEK(56334)AND254
30 POKE 1,PEEK(1)AND251
40 FOR I=0 TO 1023: POKE I+14336,PEEK(55296+I):NEXT I
50 FOR I=0 TO 511: POKE I+15360,PEEK(53760+I):NEXT I
60 FOR I=0 TO 511: POKE I+15872,PEEK(54784+I):NEXT I
70 POKE 1,PEEK(1)OR4
80 POKE 56334,PEEK(56334)OR1
```

TAULUKKO 7: Merkki-ROMin osoitteet

PEEK	Merkkijoukko	Näyttökoodit
53248 + I	Isot kirjaimet	0— 63
53760 + I	Grafiikkamerkit	64—127
54272 + I	Käänteiset isot kirjaimet	128—191
54784 + I	Käänteiset grafiikkamerkit	192—255
55296 + I	Pienet kirjaimet	0— 63
55808 + I	Isot kirjaimet	64—127
56320 + I	Käänteiset pienet kirjaimet	128—191
56832 + I	Käänteiset isot kirjaimet	192—255

Kätevä tapa siirtää suunnitellut omat merkit varatulle RAM-alueelle on käyttää FOR/NEXT-silmukkaa, jossa luetaan DATA-lauseita ja siirretään arvot POKE-käskyllä sopiviin muistipaikkoihin. Seuraava BASIC-koodi havainnollistaa tätä.

```
90 READ CN: IF CN<0 THEN 1000
100 FOR I=0 TO 7: READ CD
110 POKE 14336+8*CN+I,CD: NEXT I
120 GOTO 90
130 DATA 0,40,16,146,214,254,238,198,0
140 DATA ....
999 DATA -1
1000 POKE 53272,(PEEK(53272)AND240)OR14: REM OHJELMAN ALKU
```

Lause 90 lukee DATA-lauseiden ensimmäisen luvun. Siinä annetaan määriteltävän merkin numero. Se toimii myös lippuna ilmoittaen ohjelmalle, koska DATA loppuu. Viimeinen DATA-lause sisältää negatiivisen luvun, ja luettuaan sen ohjelma haarautuu siihen osaan, joka käyttää uusia merkkejä. Lausessa 10 luetaan seuraavat kahdeksan arvoa DATA-lauseista. Lauseessa 110 käytetään merkin numeroa CN ja rivi-indeksiä I laskemaan muistipaikka, johon rivin kuvaus tallennetaan. Lause 120 lähettää ohjelman takaisin lauseeseen 90, jossa uusi merkkinumero luetaan. Lause 1000 aloittaa ohjelman, joka käyttää uutta merkkijoukkoa. Ohjelman alussa asetetaan merkkijoukon osoitin osoitteeseen 53272.

On toinenkin tapa määritellä merkkijoukkoja. Se voi olla helpompi käyttää, koska sen syöttötiedot vastaavat selvästi merkkien visuaalista hahmoa. Se on kuitenkin hitaampi menetelmä ja kuluttaa muistia enemmän kuin ensimmäinen menetelmä. Se on hyödyllisin silloin, kun määriteltävänä on vain muutamia merkkejä.

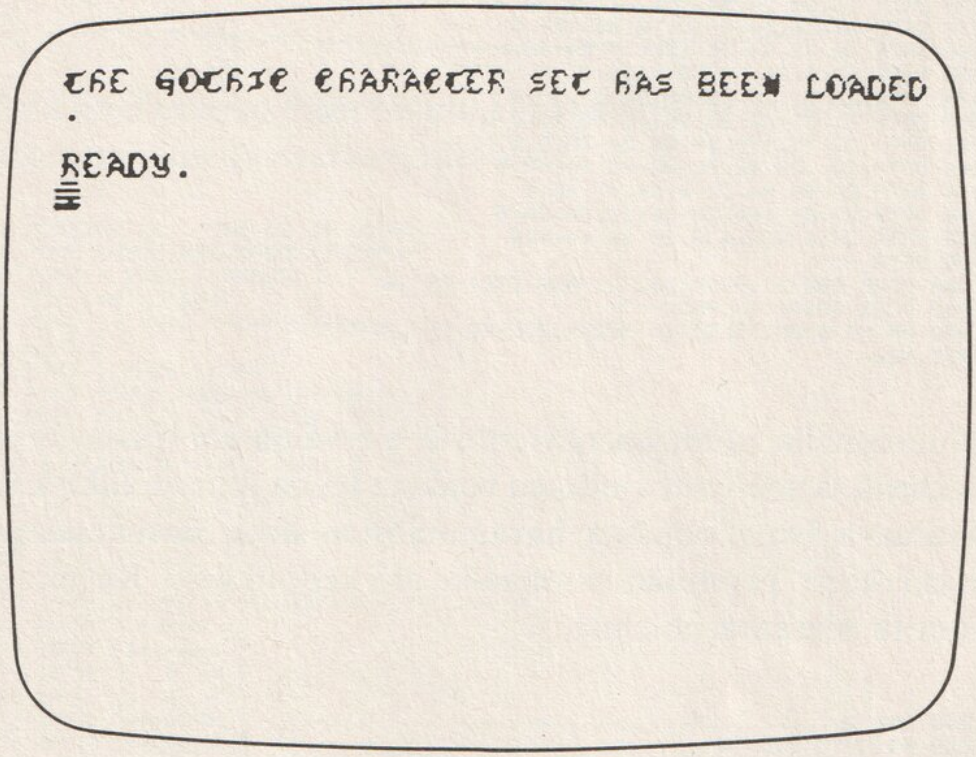
```
90 READ CN: IF CN<0 THEN 1000
100 FOR I=0 TO 7: READ CD$
110 BY=14336+8*CN+I: POKE BY,0
120 FOR J=0 TO 7
130 IF MID$(CD$,J+1,1)=" " GOTO 150
140 POKE BY,PEEK(BY)OR2↑(7-J)
150 NEXT J: NEXT I
160 GOTO 90
170 DATA 0
171 DATA "  * *   "
172 DATA "   *    "
173 DATA "*  *  * "
174 DATA "** * ** "
175 DATA "******** "
176 DATA "***  *** "
177 DATA "**    ** "
178 DATA "         "
999 DATA -1
1000 POKE 53272,(PEEK(53272)AND240)OR14
```

Edelläkuvatuissa merkkijoukkojen laatimismenetelmissä on joitakin rajoituksia. Vakavin rajoitus on se, että laatimisohjelma ja siihen liittyvät DATA-lauseet kuluttavat huomattavasti muistia. Jos pitää määritellä vain muutamia merkkejä ja loppu ohjelmasta on melko lyhyt, tämä ei ehkä ole suuri rajoitus. Mutta jos pitää määritellä paljon merkkejä tai jos merkkejä

käyttävä ohjelma on suuri, on käytettävä toista menetelmää merkkijoukon lukemiseksi sisään. Paremmallakin menetelmällä joudutaan silti uhraamaan 2048 tavua merkkien tallennukseen.

Suoraviivaisin ratkaisu on jakaa ohjelma kahteen osaan. Ensimmäinen ohjelma varaa tilan ja muodostaa uuden merkkijoukon. Toinen ohjelma, joka käyttää merkkijoukkoa, ladataan sitten edellisen päälle.

Seuraava ohjelma lataa merkkijoukon, joka korvaa kirjaimet A—Z grafiikkamerkkijoukosta vastaavilla goottilaistyylisillä kirjaimilla. Huomaa, että merkkien 1—26 korvaamisen ohella korvataan myös merkit 129—154. Ne ovat kirjainten A—Z käänteiskuvaukset.



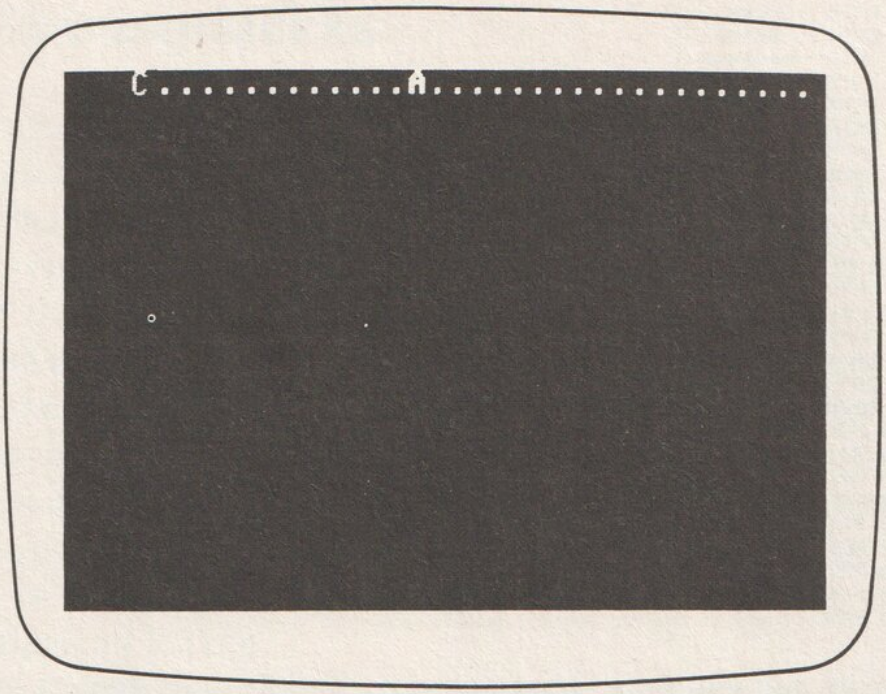
Ohjelma "GOOTTILAISET MERKIT"

```
10 POKE 52,56: POKE 56,56: CLR: REM SIIRRETÄÄN BASICIN LOPPUOSOITETTA
20 POKE 56334,PEEK(56334) AND 254
30 POKE 1,PEEK(1) AND 251
40 FOR I=0 TO 511
50 POKE I+14336,PEEK(53248+I)
60 NEXT I: REM SIIRRETÄÄN MERKKIJOUKKO
```

```
70 POKE 1,PEEK(1) OR 4
80 POKE 56334,PEEK(56334) OR 1
90 READ CN: IF CN<0 THEN 1000
100 FOR I=0 TO 7: READ CD
110 POKE 14336+8*CN+I,CD
120 POKE 14336+8*(128+CN)+I,255-CD: NEXT I
130 GOTO 90: REM MÄÄRITELLÄÄN UUDET MERKIT
140 POKE 56334,PEEK(56334) OR 1
150 READ CN: IF CN<0 THEN 1000
160 FOR I=0 TO 7: READ CD
170 POKE 14336+8*CN+I,CD: POKE 14336+8*(128+CN)+I,255-CD: NEXT I
180 GOTO 150
200 DATA 1,48,72,20,34,62,34,65,0
202 DATA 2,92,34,66,124,66,34,92,0
204 DATA 3,28,34,64,80,80,34,28,0
206 DATA 4,88,100,66,66,66,100,88,0
208 DATA 5,92,34,64,112,64,34,92,0
210 DATA 6,92,34,32,120,32,32,64,0
212 DATA 7,28,34,64,94,98,62,2,6
214 DATA 8,28,34,32,60,34,34,36,0
216 DATA 9,2,60,72,8,10,60,64,0
218 DATA 10,1,2,2,2,34,68,56,0
220 DATA 11,66,36,40,112,40,36,66,0
222 DATA 12,24,36,32,32,32,33,94,0
224 DATA 13,84,42,42,106,42,42,64,0
226 DATA 14,66,50,42,106,42,42,68,0
228 DATA 15,28,34,81,81,81,34,28,0
230 DATA 16,92,34,34,124,32,32,64,0
232 DATA 17,56,84,34,2,12,26,124,0
234 DATA 18,92,34,34,120,36,34,66,0
236 DATA 19,2,60,64,60,2,60,64,0
238 DATA 20,1,126,48,80,80,33,30,0
240 DATA 21,33,82,18,18,18,18,12,0
242 DATA 22,76,178,34,34,34,20,8,0
244 DATA 23,128,92,82,82,82,84,40,0
246 DATA 24,34,84,12,8,24,37,66,0
248 DATA 25,66,164,36,36,26,66,60,0
250 DATA 26,126,2,4,8,16,32,64,126
990 DATA -1
1000 POKE 53272,(PEEK(53272) AND 240) OR 14
1010 POKE 53281,1: PRINT"□"
1020 PRINT"■OOTTILAINEN MERKKIJOUKKO ON LADATTU"
1030 NEW
```

Muuttamalla ruudussa näytettäviä merkkejä sopivassa järjestyksessä ja sopivasti ajoittaen voidaan luoda illuusio liikkeestä. Seuraava lyhyt ohjelma havainnollistaa tätä. Ensimmäisessä esimerkissä käytetään normaalia merkkijoukkoa. Kirjoita ja suorita seuraava ohjelma.

```
1010 S$="■.........A.........................."
1020 PRINT"□";S$;"■";
1030 FOR I=0 TO 38
1040 PRINT"O■■";
1050 FOR T=0 TO 100: NEXT T
1060 PRINT" C■■";
1070 FOR T=0 TO 100: NEXT T
1080 NEXT I
```



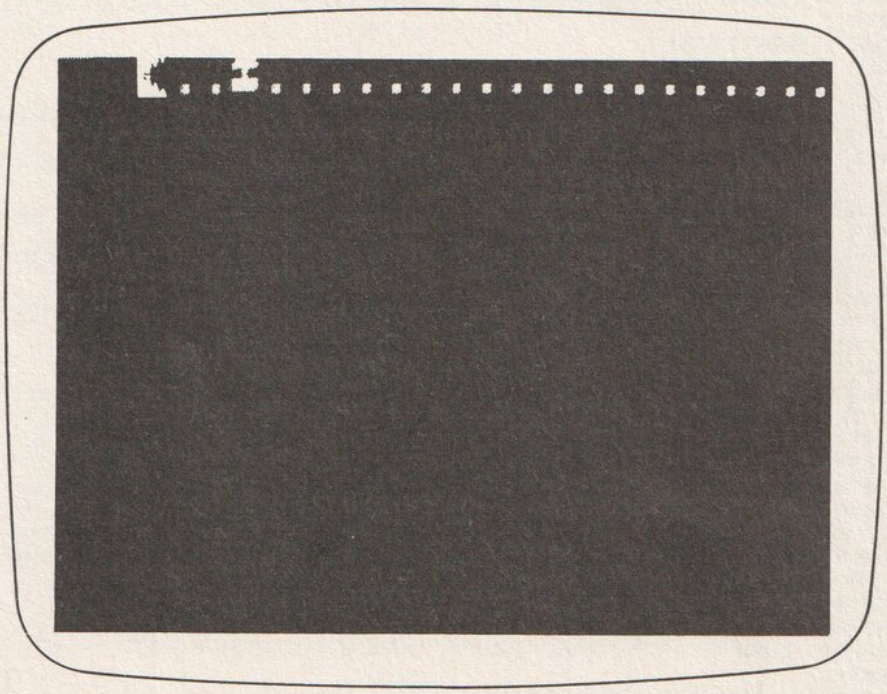
Tätä ohjelmaa voidaan muuttaa muunnetuilla merkeillä. Lisää ohjelman alkuun seuraavat lauseet:

```
10 POKE 52,56: POKE 56,56: CLR
20 POKE 56334,PEEK(56334)AND254
30 POKE 1,PEEK(1)AND251
40 FOR I=0 TO 2047
50 POKE I+14336,PEEK(53248+I)
60 NEXT I
70 POKE 1,PEEK(1)OR4
80 POKE 56334,PEEK(56334)OR1
90 READ CN: IF CN<0 THEN 1000
100 FOR I=0 TO 7: READ CD$
110 BY=14336+8*CN+I: POKE BY,0
120 FOR J=0 TO 7
130 IF MID$(CD$,J+1,1)=" " GOTO 150
140 POKE BY,PEEK(BY)OR2↑(7-J)
150 NEXT J: NEXT I
160 GOTO 90
170 DATA 1: REM UUDELLEENMÄÄRITELLÄÄN A
171 DATA "  *****  "
172 DATA "******* "
173 DATA "*  *  * "
174 DATA "*  *  * "
175 DATA "******* "
176 DATA "******* "
177 DATA "* * * * "
178 DATA "        "
180 DATA 3: REM UUDELLEENMÄÄRITELLÄÄN C
181 DATA "  ***   "
182 DATA " *****  "
183 DATA "****    "
184 DATA "**      "
185 DATA "****    "
186 DATA " *****  "
187 DATA "  ***   "
188 DATA "        "
190 DATA 15: REM UUDELLEENMÄÄRITELLÄÄN O
```

```
191 DATA "   ***    "
192 DATA "  *****   "
193 DATA " ******* "
194 DATA " ******* "
195 DATA " ******* "
196 DATA "  *****   "
197 DATA "   ***    "
198 DATA "          "
200 DATA 46: REM UUDELLEENMÄÄRITELLÄÄN .
201 DATA "          "
202 DATA "          "
203 DATA "          "
204 DATA "    **    "
205 DATA "    **    "
206 DATA "          "
207 DATA "          "
208 DATA "          "
999 DATA -1: REM DATAT LOPPU
1000 POKE 53272,30
1010 S$="■.........A.............................."
1020 PRINT"□";S$;"■";
1030 FOR I=0 TO 38
1040 PRINT"O■";
1050 FOR T=0 TO 100: NEXT T
1060 PRINT" C■";
1070 FOR T=0 TO 100: NEXT T
1080 NEXT I
```

Suorita nyt ohjelma. Huomaa ero, jonka uudet merkit saavat aikaan ohjelman visuaalisessa vaikutelmassa. Päästäksesi takaisin normaaliin merkkijoukkoon anna käsky

POKE 53272,21



Tarkkuusgrafiikka

Kuusnelonen pystyy käsittelemään 64 000 erillistä pistettä kuvaruudussa. Se on helppo todeta laskemalla. Kuten ohjelmoitavien merkkien yhteydessä mainittiin, kukin merkki muodostuu 8×8 pisteestä. Koska rivi on 40 merkkiä pitkä ja käytettävissä on 25 riviä, tietokone näyttää 320 pistettä vaakasuoraan ja 200 pistettä pystysuoraan. Kunkin ruudussa havaittavan pisteen tila (toiminnassa tai ei) vastaa yhden bitin tilaa (0 tai 1) tietokoneen muistissa. Tarkkuusgrafiikkamoodissa kukin piste voidaan yksilöllisesti sytyttää tai sammuttaa. Ruudussa on kuva tietystä osasta tietokoneen RAM-muistia. Koska näytöllä on 64 000 pistettä ja tavussa on 8 bittiä, bittikartta vaatii muistista 8000 muistipaikan lohkon.

Muistipaikkojen sisältö kuvautuu näyttöön samaan tapaan kuin merkitkin, kahdeksan tavun ryhminä pystysuoraan järjestettynä. Seuraava kuva selittää järjestelmän.

Merkki-sarake	*0*	*1*	*2*	.....	*38*	*39*	Pystysuora asema
Rivi	0	8	16	.....	304	312	0
	1	9	17	.....	305	313	1
	2	10	18	.....	306	314	2
0	3	11	19	.....	307	315	3
	4	12	20	.....	308	316	4
	5	13	21	.....	309	317	5
	6	14	22	.....	310	318	6
	7	15	23	.....	311	319	7
	320	328	336	.....	624	632	8
	321	329	337	.....	625	633	9
	322	330	338	.....	626	634	10
1	323	331	339	.....	627	635	11
	324	332	340	.....	628	636	12
	325	333	341	.....	629	637	13
	326	334	342	.....	630	638	14
	327	335	343	.....	631	639	15
Merkkirivit 2—23							
	7680	7688	7696	.....	7984	7992	192
	7681	7689	7697	.....	7985	7993	193
	7682	7690	7698	.....	7986	7994	194
24	7683	7691	7699	.....	7987	7995	195
	7684	7692	7700	.....	7988	7996	196
	7685	7693	7701	.....	7989	7997	197
	7686	7694	7702	.....	7990	7998	198
	7687	7695	7703	.....	7991	7999	199
Vaakasuor.	0–7	7–15	16–23		304–311	312–319	

Kuva 5: Tarkkuusgrafiikan järjestys

Jos yhden tarkkuusgrafiikan pikselin (kuvapisteen) vaaka- ja pystykoordinaatteja kuvataan muuttujilla PX ja PY, voidaan johtaa yhtälö, joka ilmaisee vastaavan bitin sisältävän muistipaikan ML. Muistipaikan merkkirivi RW ja merkkisarake CL saadaan jakamalla 8:lla vaaka- ja pystykoordinaatit. Tuloksesi saadaan merkkirivi ja merkkisarake.

```
CL = INT(PX/8)
RW = INT(PY/8)
```

Tietyn rivin ja sarakkeen merkkilohkon ensimmäinen muistipaikka FML saadaan seuraavasti.

```
FML = 8*CL + 320*RW
FML = 8*(CL + 40*RW)
```

Jotta saadaan selville halutun pikselin sisältämä muistipaikka merkkilohkon sisällä, lohkon ensimmäiseen muistipaikkaan pitää vielä lisätä jotain. Lisäys OY saadaan seuraavasti.

```
OY = PY—8*INT(PY/8)
OY = PY—8*RW
```

Muistipaikka, joka sisältää pikselin pisteessä (PX, PY) löydetään siis lisäämällä merkkilohkon ensimmäiseen muistipaikkaan FML lisäys OY. Korvaamalla OY kaavasta muilla ja keräämällä tekijöitä yhteen saadaan kaavaksi

```
ML = 8*(CL + 40*RW) + OY
ML = 8*(CL + 40*RW) + PY—8*RW
ML = 8*(CL + 39*RW) + PY
```

Korvaamalla CL ja RW saadaan

```
ML = 8*INT(PX/8) + 39*(INT(PY/8)) + PY
```

Tässä oletetaan tarkkuusgrafiikan muistialueen alkavan osoitteesta 0. Tarkkuusgrafiikan muistialueen sijainti voi kuitenkin vaihdella sen mukaan, miten se on määritelty. Täten kaavaan on vielä lisättävä tarkkuusgrafiikan muistialueen alkuosoite SA.

ML = SA + 8*INT(PX/8) + 39*(INT(PY/8)) + PY

Haluttua pikseliä kontrolloivan bitin sijainti tavussa saadaan jakojäännöksenä, kun PX jaetaan 8:lla. Bitti lasketaan vasemmalta oikealle.

bitin järjestysnumero vasemmalta = PX—8*INT(PX/8)

Jos esimerkiksi PX = 19, jakojäännös kahdeksalla jaon jälkeen on 3. Haluttu bitti sijaitsee 3 bittiä oikealle vasemmanpuolisesta bitistä. Bittien järjestyksessä näytöllä muistipaikan suurin bitti, bitti 7, on vasemmanpuoleisin. Esimerkissämme kolmas bitti oikealle on bitti 4. Bitti, joka halutaan asettaa tai nollata PX:n arvolla, saadaan siis seuraavasti

BT = 7—(PX—8*INT(PX/8))

Pikseli sytytetään korottamalla luku 2 bitin järjestysnumeron osoittamaan potenssiin ja tekemällä OR-operaatio muistipaikan entisen sisällön ja saadun luvun kanssa. Pikseli sammutetaan vastaavalla tavalla mutta OR-operaation tilalla käytetään AND NOT-operaatiota. Pisteiden sytyttäminen ja sammuttaminen voidaan tiivistää seuraaviin vaiheisiin:

1. Etsi pikselin koordinaatteja PX ja PY vastaava muistipaikka ja bitti

ML = SA + 8*INT(PX/8) + 39*(INT(PY/8)) + PY
BT = 7—(PX—8*INT(PX/8))

2. Sytytä piste

POKE ML, PEEK(ML) OR (2↑BT)

tai sammuta piste

POKE ML, PEEK(ML) AND NOT (2↑BT)

Edustan (piste sytytetty) ja taustan (piste sammutettu) värit määräytyvät värikartan mukaan. Tätä värikarttaa ei pidä sekoittaa värimuistiin, jota käytetään merkkimoodissa. Värikartta käsittää 1K RAM-muistia. Värikartan kukin tavu määrää edustan ja taustan värit 8×8 pisteen alueella näytöllä. Kunkin värialueen sijainti vastaa tekstinäytöllä yhden merkin varaamaa aluetta. Väritavu on jaettu kahteen osaan. Bitit 0—3 määräävät taustan värin ja bitit 4—7 edustan värin. Kaikkia Commodore 64:ssä käytössä olevia 16 väriä voidaan käyttää kaikkina yhdistelminä. Ainoa rajoitus on, että kahta väriä enempää ei voi esiintyä kerrallaan yhdellä 8×8 pisteen alueella. Taulukossa 8 luetellaan arvot kaikille väriyhdistelmille.

Tarkkuusgrafiikkamoodi kytkeytyy toimintaan muistipaikan 53265 bitin 5 tilan perusteella.

POKE 53265,PEEK(53265) OR 32

Tarkkuusgrafiikkamoodi kytketään pois käskyllä

POKE 53265,PEEK(53265) AND NOT 32

Seuraava ohjelma havainnollistaa tarkkuusgrafiikan käyttöä. Ensiksi tarkkuusgrafiikan käyttämä alue on suojattava BASICiltä. Se tehdään muuttamalla BASICin alueen loppuosoitetta ja merkkijonojen alkuosoitetta.

```
10 POKE 52,32: POKE 56,32: CLR
```

Valitaan tarkkuusgrafiikan ja värikartan alueet. Tarkkuusgrafiikka sijoitetaan muistialueelle 8192—16383 ja värikartta

TAULUKKO 8: Tarkkuusgrafiikkatilan väritavut
Piirtovärit ovat sarakkeilla ja taustavärit riveillä.

	Musta	Valk.	Pun.	Sinivihr.	Sinipun.	Vihr.	Sin.	Kelt.	Oranssi	Rusk.	Vaal. pun.	Tumman harmaa	Keski-harmaa	Vaal. vihreä	Vaal. sininen	Vaal. harmaa
Musta	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valkoinen	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Punainen	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Sinivihreä	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Sinipunainen	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Vihreä	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Sininen	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Keltainen	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Oranssi	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Ruskea	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Vaal.pun.	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
Tumman harm.	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Keskiharmaa	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Vaal.vihr.	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Vaal.sin.	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
Vaal.harm.	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

alueelle 1024—2023. Sitten tarkkuusgrafiikkamoodi kytketään päälle.

```
20 SA=8192: SC=1024
30 POKE 53272,24
40 POKE 53265,PEEK(53265)OR32: REM BITTIKARTTA PÄÄLLE
```

Seuraavaksi värikartta täytetään ykkösillä. Valkoiselle pohjalle ilmestyy mustia pisteitä. Tarkkuusgrafiikan muistialue tyhjennetään täyttämällä sen muistipaikat nollilla. Ohjelman nämä osat on kirjoitettu aliohjelmiksi.

```
50 GOSUB 230: REM VARITETÄÄN KUVARUUTU
60 GOSUB 250: REM TYHJENNETÄÄN KUVARUUTU
230 FOR I=SC TO SC+999: POKE I,1: NEXT I
240 RETURN
250 FOR I=SA TO SA+7999: POKE I,0: NEXT I
260 RETURN
```

Ensimmäinen piste asetetaan keskelle muistialuetta. Pisteiden asetus on kirjoitettu aliohjelmaksi.

```
70 PX=159: PY=99: GOSUB 270: REM SYTYTETÄÄN PISTE
270 ML=SA+8*(INT(PX/8)+39*INT(PY/8))+PY
280 BT=7-(PX-8*INT(PX/8))
290 POKE ML,PEEK(ML)OR(2↑BT)
300 RETURN
```

Kytkettyään muistialueen ensimmäisen pisteen päälle ohjelma lukee pelisäädinporttia 2. Se on taempi kahdesta oikealla olevasta liittimestä. Jos joystickiltä tulee syöttöä, ohjelma kytkee mukaan uuden pisteen ja palaa uudestaan lukemaan peliporttia.

```
80 GOSUB 110: IF J=15 THEN 80: REM LUETAAN JOYSTICKIÄ
90 GOSUB 270: REM SYTYTETÄÄN UUSI PISTE
100 GOTO 80
110 J=PEEK(56320)AND15
120 IF (J AND 8)=0 THEN PX=PX+1: REM OIKEALLE
130 IF (J AND 4)=0 THEN PX=PX-1: REM VASEMMALLE
140 IF (J AND 2)=0 THEN PY=PY+1: REM ALAS
150 IF (J AND 1)=0 THEN PY=PY-1: REM YLÖS
160 IF PY>199 THEN PY=199
170 IF PY<0 THEN PY=0
180 IF PX>319 THEN PY=319
190 IF PX<0 THEN PX=0
200 GET A$: IF A$="□" THEN GOSUB 250: GOSUB 270: REM RUUDUN TYHJENNYS
210 IF A$="■" THEN 310: REM LOPETUS F7:LLÄ
220 RETURN
```

Jos CLR/HOME-näppäintä painetaan yhdessä SHIFT-näppäimen kanssa, muistialue tyhjenee. F7-näppäin kytkee tarkkuusgrafiikkamoodin pois ja lopettaa ohjelman suorituksen.

```
310 POKE 53272,21: REM NORMAALI KUVARUUTU JA MERKIT
320 POKE 53265,PEEK(53265) AND NOT 32: REM BITTIKARTTA POIS PÄÄLTÄ
330 END
```

Tämä ohjelma tekee Commodore 64:stä joystickillä ohjattavan piirustuslaudan.

Tarkkuusgrafiikan kuvaus on tähän asti ollut varsin rajallista. Esimerkiksi kaikki BASIC-ohjelmat rajoittuvat edellä kuvatulla tekniikalla kooltaan enintään 6 kiloon. Se johtuu muistin käytöstä grafiikkaa varten. Käytettävät käskyt tarkkuusgrafiikkamoodin kytkemiseksi ja pisteiden asettamiseksi ovat yhä konstikkaita. Näistä puutteista päästään kuitenkin käyttämällä seuraavaa ohjelmaa, MINIGRAPHia. Ohjelma ladataan joukkona konekielisiä rutiineja muistialueelle 36883—37681. Tarkkuusgrafiikkatilan kytkeminen ja siihen piirtäminen yksinkertaistuvat yksittäisiksi käskyiksi. Tarkkuusgrafiikan muistialue sijaitsee BASICin alueen ulkopuolella jättäen 33 K RAMia ohjelmia varten. Ja rutiinit ovat hyvin nopeita, koska ne on kirjoitettu konekielellä. Kun syötät lukuja DATA-lauseisiin, on hyvin tärkeää olla tarkkana. Väärä luku saattaa sekottaa tietokoneesi. Jos niin käy, on ainoa keino katkaista virta hetkeksi. Tällöin menetetään muistissa ollut ohjelma.

Kirjoita ohjelma ja tallenna se heti nauhalle tai levykkeelle. Tämä on apuohjelma, joka ladataan tietokoneeseen ja suoritetaan ennen muiden tätä ohjelmaa käyttävien ohjelmien lataamista. Rutiineja kutsutaan SYS-käskyillä. SYS-käsky siirtää kontrollin määrättyyn muistipaikkaan ja aiheuttaa siellä olevan konekoodin suorituksen. Rutiinisuorituksen jälkeen kontrolli palaa takaisin BASIC-ohjelmalle. Tarkkuusgrafiikkamoodi kytketään käskyllä

```
SYS 50195
```

Tarkkuusgrafiikkamoodi kytketään pois käskyllä

SYS 50198

Tarkkuusgrafiikan muistialue tyhjennetään eli kaikki bitit nollataan käskyllä

SYS 50207

Tarkkuusgrafiikan värit määrätään käskyllä

SYS 50210,F,B

F ja B ovat lukuja tai muuttujia, jotka saavat arvoja välillä 0—15. Ensimmäinen luku F asettaa edustavärin ja B asettaa taustavärin. Piste asetetaan päälle käskyllä

SYS 50201,X,Y,M

X ja Y ovat pisteen vaaka- ja pystykoordinaatit. Vaakakoordinaatti voi vaihdella välillä 0—319 ja pystykoordinaatti välillä 0—199. Vasen yläkulma on piste 0,0. Viimeinen muuttuja M määrää piirtomoodin. Jos M = 1, edustanvärinen piste asetetaan päälle. Jos M = 2, taustanvärinen piste asetetaan päälle. Jos M = 0, pisteen väri vaihdetaan. Jos piste on ollut edustanvärinen, siitä tuleekin taustanvärinen tai päinvastoin. Viiva vedetään käskyllä

SYS 50204,X,Y,M

X ja Y ovat viivan loppukohdan vaaka- ja pystykoordinaatit. Alkupisteen määrää joko edellinen pisteenasetuskäsky tai edellisen viivanpiirron loppupiste. Piirtomoodi M on sama kuin pisteen asetusrutiinissakin. Sallittujen rajojen ulkopuoliset arvot parametreillä X, Y, M, F tai B aiheuttavat virheilmoituksen ILLEGAL QUANTITY ERROR.

HUOM: Viivan piirtokäskyn pitää aina olla joko viimeinen käsky BASIC-ohjelmarivillä tai sen on oltava yksin omalla rivillään. Muuten saat virheilmoituksen. Jos SYNTAX- tai ILLEGAL-virheilmoituksia alkaa ilmestyä MINIGRAPHia käyttävissä ohjelmissa, tarkista tämä seikka.

MINIGRAPHia seuraava ohjelma esittelee, kuinka rutiineja voidaan käyttää piirrettäessä tarkkuusgrafiikalla erilaisia geometrisia muotoja. Ohjelma päättyy, kun painat jotain näppäintä.

Ohjelma "MINIGRAPH.DAT"

```
10 PRINT"■MINIGRAPH-LATAUSOHJELMA / PAUL SCHATZ"
20 PRINT"ODOTA HETKEN AIKAA..."
30 A=0: FOR I = 50195 TO 50975: READ BY
40 POKE I,BY: A=A+BY: NEXT I
50 IF A<>95619 THEN PRINT"VIRHE DATA-LAUSEISSA": STOP
60 PRINT"■MINIGRAPH-KÄSKYT:"
70 PRINT"■BITTIKARTTA PÄÄLLE        SYS 50195"
80 PRINT"BITTIKARTTA POIS PÄÄLTÄ  SYS 50198"
90 PRINT"BITTIKARTAN TYHJENNYS    SYS 50207"
100 PRINT"BITTIKARTAN VÄRIASETUS   SYS 50210,F,B"
110 PRINT"PISTEEN (X,Y) ASETUS     SYS 50201,X,Y,M";
120 PRINT"VIIVAN PIIRTO (X,Y):HYN  SYS 50204,X,Y,M"
130 PRINT"■F = PIIRTOVÄRI     B = TAUSTAVÄRI"
140 PRINT"X = 0 - 319         Y = 0 - 199"
150 PRINT"M = 0, 1, 2 (VAIHDA, PIIRRÄ TAI PYYHI)"
160 NEW
200 DATA 76,244,198,76,7,199,76,37,196,76,114,197,76,30,197,76,66
210 DATA 197,169,0,141,4,196,32,253,174,32,235,183,224,200,144,3,76
220 DATA 26,199,142,3,196,166,20,165,21,240,8,201,1,208,240,224,64
230 DATA 176,236,141,2,196,142,1,196,32,253,174,32,158,183,224,3,176
240 DATA 220,142,0,196,173,0,196,240,27,74,144,12,32,174,196,32,135
250 DATA 196,29,158,196,145,251,96,32,174,196,32,135,196,61,166,196,145
260 DATA 251,96,32,174,196,32,135,196,93,158,196,145,251,96,173,1,196
270 DATA 41,7,170,120,160,52,132,1,160,0,177,251,160,55,132,1,88
280 DATA 160,0,96,128,64,32,16,8,4,2,1,127,191,223,239,247,251
290 DATA 253,254,169,0,133,251,169,224,133,252,173,1,196,41,248,24,101
300 DATA 251,133,251,173,2,196,101,252,133,252,173,3,196,72,41,7,24
310 DATA 101,251,133,251,144,2,230,252,104,74,74,74,10,170,189,236,196
320 DATA 24,101,251,133,251,189,237,196,101,252,133,252,96,0,0,64,1
330 DATA 128,2,192,3,0,5,64,6,128,7,192,8,0,10,64,11,128
340 DATA 12,192,13,0,15,64,16,128,17,192,18,0,20,64,21,128,22
350 DATA 192,23,0,25,64,26,128,27,192,28,0,30,162,32,169,224,133
360 DATA 252,169,0,133,251,168,145,251,200,208,251,230,252,202,208,246,96
370 DATA 32,253,174,32,158,183,224,16,144,3,76,26,199,96,169,192,133
380 DATA 252,169,0,133,251,32,52,197,138,10,10,10,10,141,9,196,32
390 DATA 52,197,138,13,9,196,162,2,160,0,145,251,200,208,251,230,252
400 DATA 202,16,246,145,251,200,192,232,144,249,96,169,0,141,8,196,141
410 DATA 10,196,32,253,174,32,235,183,224,200,144,3,76,26,199,142,7
420 DATA 196,166,20,165,21,240,8,201,1,208,240,224,64,176,236,141,6
430 DATA 196,142,5,196,32,253,174,32,158,183,201,3,176,220,142,0,196
440 DATA 173,5,196,56,237,1,196,141,12,196,173,6,196,237,2,196,141
450 DATA 13,196,16,20,206,10,196,56,169,0,237,12,196,141,12,196,169
460 DATA 0,237,13,196,141,13,196,169,0,141,11,196,173,7,196,56,237
470 DATA 3,196,141,14,196,173,8,196,237,4,196,141,15,196,16,20,206
480 DATA 11,196,56,169,0,237,14,196,141,14,196,169,0,237,15,196,141
490 DATA 15,196,169,0,141,18,196,173,14,196,56,237,12,196,173,15,196
500 DATA 237,13,196,144,27,174,14,196,173,12,196,141,14,196,142,12,196
510 DATA 174,15,196,173,13,196,141,15,196,142,13,196,206,18,196,173,12
520 DATA 196,141,16,196,173,13,196,141,17,196,32,91,196,173,18,196,208
```

```
530 DATA 18,173,1,196,205,5,196,208,27,173,2,196,205,6,196,208,19
540 DATA 240,16,173,3,196,205,7,196,208,9,173,4,196,205,8,196,208
550 DATA 1,96,173,18,196,208,6,32,192,198,76,118,198,32,218,198,32
560 DATA 152,198,32,152,198,16,20,173,18,196,208,6,32,218,198,76,140
570 DATA 198,32,192,198,32,172,198,32,172,198,32,91,196,76,64,198,173
580 DATA 16,196,56,237,14,196,141,16,196,173,17,196,237,15,196,141,17
590 DATA 196,96,173,16,196,24,109,12,196,141,16,196,173,17,196,109,13
600 DATA 196,141,17,196,96,173,10,196,208,9,238,1,196,208,3,238,2
610 DATA 196,96,173,1,196,208,3,206,2,196,206,1,196,96,173,11,196
620 DATA 208,9,238,3,196,208,3,238,4,196,96,173,3,196,208,3,206
630 DATA 4,196,206,3,196,96,173,0,221,41,252,141,0,221,169,59,141
640 DATA 17,208,169,8,141,24,208,96,173,0,221,9,3,141,0,221,169
650 DATA 27,141,17,208,169,21,141,24,208,96,32,7,199,76,72,178
```

```
MINIGRAPH INSTRUCTIONS

TURN ON BITMAP          -SYS 50195
TURN OFF BITMAP         -SYS 50198
CLEAR BITMAP            -SYS 50207
COLOR BITMAP            -SYS 50210,F,B
PLOT POINT @ X,Y        -SYS 50201,X,Y,M
DRAW LINE TO X,Y        -SYS 50204,X,Y,M

F = FOREGROUND          B = BACKGROUND
X = 0 TO 319            Y = 0 TO 199
M = 0, 1, OR 2 (FLIP, DRAW, OR ERASE)

READY.
```

Ohjelma ”MINIGRAPH.DEMO”

```
10 PRINT"■MINIGRAPH.DEMO"
20 PRINT"PAUL SCHATZ"
30 GOSUB 1000: REM TAUKO
40 SYS 50207: REM BITTIKARTAN TYHJENNYS
50 SYS 50210,1,2: REM VARIEN VALINTA
60 SYS 50195: REM BITTIKARTTA PÄÄLLE
70 N1=30: N2=33: REM PIIRRETÄÄN NELIDITÄ
80 FOR I=0 TO 12: X1=N1-2*I: X2=N2+I*2
90 SYS 50201,X1,X1,1: REM PISTEEN ASETUS
100 SYS 50204,X2,X1,1
110 SYS 50204,X2,X2,1
120 SYS 50204,X1,X2,1
130 SYS 50204,X1,X1,1
140 NEXT I
150 GOSUB 1000
160 SYS 50210,5,1
170 SYS 50207
180 FP=1: A1=77: B1=23: DL=9: C=0
190 FOR I=0 TO 40: C=C+DL: A=A1*C*1/180: B=B1*C*1/180
200 X=INT(100*SIN(A)+160.5)
210 Y=INT(80*COS(B)+100.5)
220 IF FP=1 THEN FP=0: SYS 50201,X,Y,0: GOTO 240
230 SYS 50204,X,Y,0
240 NEXT I
250 GOSUB 1000
260 SYS 50210,7,2
270 SYS 50207
280 X1=0: X2=319
290 FOR I=0 TO 199
300 Y1=I: Y2=199-I
310 SYS 50201,X1,Y1,0
320 SYS 50204,X2,Y2,0
```

```
330 NEXT I
340 Y1=199: Y2=0
350 FOR I=0 TO 319
360 X1=I: X2=319-I
370 SYS 50201,X1,Y1,0
380 SYS 50204,X2,Y2,0
390 NEXT I
400 GOSUB 1000
410 SYS 50210,0,10
420 SYS 50207
430 FP=1: A1=13: B1=26: DL=6: C=0
440 FOR I=0 TO 80: C=C+DL: A=A1*C*1/180: B=B1*C*1/180
450 X=INT(100*COS(A)+160.5)
460 Y=INT(80*SIN(B)+100.5)
470 IF FP=1 THEN FP=0: SYS 50201,X,Y,1: GOTO 240
480 SYS 50204,X,Y,1
490 NEXT I
500 GOSUB 1000
510 SYS 50210,15,12
520 SYS 50207
530 FOR X=0 TO 319
540 Y=INT(100+70*SIN(X/10))
550 SYS 50201,160,10,1
560 SYS 50204,X,Y,1
570 NEXT X
900 GET A$: IF A$="" THEN 30
910 SYS 50198
920 END
1000 FOR I=0 TO 2000: NEXT I: RETURN
```

Loppuhuomioita

Tämä kappale on hyvin lyhyt johdanto Commodore 64 -grafiikkaan. Toivottavasti se antaa sinulle sysäyksen grafiikkapiirteiden sisällyttämiseen omiin ohjelmiisi. Seuraavassa taulukossa luetellaan kaikki 47 muistipaikkaa, jotka säätelevät VIC-piiriä. Viimeinen sarake ilmaisee grafiikan piirteen, johon rekisteri vaikuttaa, S = spritegrafiikka, B = tarkkuusgrafiikka ja C = merkkigrafiikka. Taulukko on kätevä apuneuvo, kun kokeilet grafiikan uusia piirteitä.

TAULUKKO 9: VIC-piirin rekisterit

Muistipaikka	Toiminta	
53248	Sprite 0 vaakasijainti	S
53249	Sprite 0 pystysijainti	S
53250	Sprite 1 vaakasijainti	S
53251	Sprite 1 pystysijainti	S
53252	Sprite 2 vaakasijainti	S
53253	Sprite 2 pystysijainti	S

53254		Sprite 3 vaakasijainti	S
53255		Sprite 3 pystysijainti	S
53256		Sprite 4 vaakasijainti	S
53257		Sprite 4 pystysijainti	S
53258		Sprite 5 vaakasijainti	S
53259		Sprite 5 pystysijainti	S
53260		Sprite 6 vaakasijainti	S
53261		Sprite 6 pystysijainti	S
53262		Sprite 7 vaakasijainti	S
53263		Sprite 7 pystysijainti	S
53264	Bitit 0—7	Eniten merkitsevä bitti spritejen 0—7 vaakasijaintiin	S
53265	Bitit 0—2	Pystysuora asema vierityksessä	BC
	Bitti 3	Rivimäärän valinta (24 tai 25)	BC
	Bitti 4	Näytön kytkeminen pois päältä	
	Bitti 5	Bittikarttamoodin valinta	B
	Bitti 6	Laajennetun taustavärimoodin valinta	C
	Bitti 7	Juova-aseman eniten merkitsevä bitti	
53266		Kuvajuova-asema	
53267		Valokynän vaaka-asema	
53268		Valokynän pystyasema	
53269	Bitit 0—7	Spritejen 0—7 kytkentä	S
53270	Bitit 0—2	Vaakasuora asema vierityksessä	BC
	Bitti 3	Sarakemäärän valinta (38 tai 40)	BC
	Bitti 4	Monivärimoodin valinta	BC
53271	Bitit 0—7	Pystysuora laajennus (spritet 0—7)	S
53272	Bitit 1—3	Merkkijoukon osoitin	C
	Bitti 3	Bittikartan osoitin	B
	Bitit 4—7	Merkkinäytön tai värikartan osoitin	BC
53273		Keskeytysrekisteri	
53274		Keskeytysrekisteri	
53275	Bitit 0—7	Etusijarekisteri spriteille 0—7	S
53276	Bitit 0—7	Monivärispritejen (0—7) valinta	S
53277	Bitit 0—7	Vaakasuora laajennus (spritet 0—7)	S

53278	Bitit 0—7	Sprite-spriten törmäysrekisteri	S
53279	Bitit 0—7	Sprite-merkin törmäysrekisteri	S
53280	Bitit 0—3	Reunuksen väri näytöllä	BC
53281	Bitit 0—3	Taustan väri näytöllä	C
53282	Bitit 0—3	Monivärimerkin väri 1	C
53283	Bitit 0—3	Monivärimerkin väri 2	C
53284	Bitit 0—3	Monivärispriten väri 1	S
53285	Bitit 0—3	Monivärispriten väri 2	S
53286	Bitit 0—3	Sprite 0 väri	S
53287	Bitit 0—3	Sprite 1 väri	S
53288	Bitit 0—3	Sprite 2 väri	S
53289	Bitit 0—3	Sprite 3 väri	S
53290	Bitit 0—3	Sprite 4 väri	S
53291	Bitit 0—3	Sprite 5 väri	S
53292	Bitit 0—3	Sprite 6 väri	S
53293	Bitit 0—3	Sprite 7 väri	S

Liite 3

Ääntä ja musiikkia tutkimassa

Frank H. Covitz

Viidestä ihmisen aistista kuulo on oletettavasti toiseksi tärkein näön jälkeen. (''Kuva kertoo enemmän kuin tuhat sanaa.'') Vastaanottamamme kuuloinformaation määrä on suunnaton. Ajattelepa sitä hetki: kaikkea puhetta kaikilla kielillä, kaikkea musiikkia ja miltei määrättömästi vaihtelevia luonnon ääniä aaltojen kohinasta lintujen lauluun.

Aivan kuten tarvitaan paljon tekstiä kuvaa kuvailemaan, tarvitaan paljon sanoja kuvaamaan ääniä, erityisesti jos päämääränä on ymmärtää niitä tarpeeksi voidaksemme luoda niitä. SID-äänipiiri Commodore 64:ssä pystyy tuottamaan runsaasti korkeatasoisia ääniä. Jotta voisimme hyödyntää tätä ominaisuutta tehokkaasti, meidän on ensin sovittava joistakin periaatteista, joita tarvitaan äänen kuvaamisen ymmärtämiseksi.

On pidettävä mielessä, että puhumme koko ajan värähtelyistä. Normaalissa kuulemisessa korvan mekanismit pyydystävät ilman värähtelyt ja muuntavat ne hermoimpulsseiksi, jotka aivot sitten tulkitsevat. Jos värähtely on hyvin matalataajuista, voimme oikeastaan tuntea sen yhtä hyvin kuin kuulla — laitapa kätesi ison jyskyttävää rokkia toistavan kaiuttimen eteen!

Värähtelyt voidaan esittää muussakin muodossa kuin varsi-

naisina ilmanpaineen vaihteluina. Käytämme tässä yksinkertaisia tapoja. Vaikka esimerkiksi magneettinauhoille (magneettisuuden muutoksina) tai äänilevylle (pinnan kaiverruksena) tallennettu informaatio ei sinänsä ole ääntä, se voidaan helposti muuntaa ääneksi vahvistimella ja kaiuttimilla. Se informaatio, jota tarvitaan äänen muodostamiseksi tietokoneella, on numeromuodossa tietokoneen muistissa. Mitä nämä luvut merkitsevät, mihin ne vaikuttavat ja miten niistä tulee ääntä? Näihin kysymyksiin vastataan tässä luvussa.

Peruskysymyksiä on kaksi: 1) Mitkä ovat äänen muodostavan värähtelyn peruspiirteet? 2) Miten voimme käyttää näitä piirteitä luodaksemme halutun äänen?

Ääniteorian perusteet

Tarkastellaan ensin suhteellisen yksinkertaisia ääniä, joita kutsutaan nuoteiksi. Meidän tarkoitukseemme nuotti on vakaa ääni, jolla on tietty sävelkorkeus. Vakaalla tarkoitan sitä, että sen laatu ei muutu (paitsi että sillä tietenkin on alku ja loppu). Käsite sävelkorkeus on hieman subjektiivinen, mutta se voi tarkoittaa, että äänellä on tietty taajuus, jolla voimme esimerkiksi viheltää mukana ”samassa nuotissa”.

Taajuus on luku, joka määrittelee kuinka monta värähdystä tapahtuu sekunnissa. Taajuuden yksikköä kutsuttiin joskus värähtelyiksi sekunnissa, mutta elektromagneettisten aaltojen ja siihen liittyvien ilmiöiden tutkimuksen uranuurtajan Heinrich Hertzin kunniaksi on yleisesti otettu käyttöön termi hertsi (lyh. Hz).

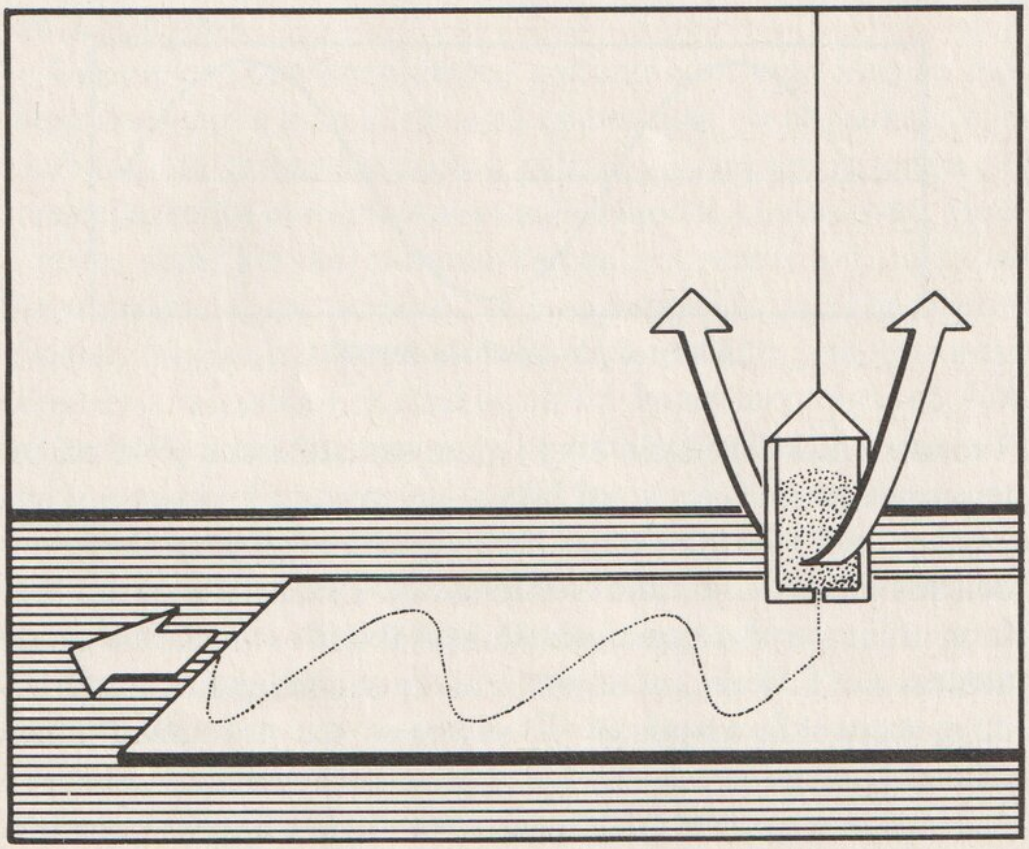
Koska on tärkeää ymmärtää taajuuden merkitys hyvin, katsotaan paria esimerkkiä. Ajattele, miten heiluri liikkuu vanhanaikaisessa kellossa. Sen liikkeen taajuus on 1/2 Hz (kukin puoliheilahdus kestää sekunnin). Tässä tapauksessa voit nähdä ja laskea heilahdukset. Mutta äänessä taajuudet ovat liian suuria, jotta voisit nähdä yksittäiset syklit ilman oskilloskoopin tapaisen laitteen apua. Otetaanpa tavallinen keittiöveitsi, joka

asetetaan puoliksi pöydän reunan yli. Kun pidetään toista päätä tiukasti pöydän pintaan painettuna ja näpäytetään vapaata päätä, veitsi saadaan värähtelemään noin 10 Hz:n taajuudella.

Ihmisen korvan erottamien taajuuksien rajat ovat n. 20 Hz — 20 000 Hz, vaikkakin keskimäärin ihminen erottaa ääniä yleensä vain väliltä 30 Hz — 15 000 Hz. (Silti ihminen voi ''tuntea'' myös näitä alhaisempia taajuuksia). Jos emme voi kuulla alle 20 Hz:n ääniä, niin miten voimme helposti kuulla pöytäveitsen värähtelyn, jonka juuri esitettiin olevan n. 10 Hz? Se mitä todella kuulet, ovat terän iskut pöydän reunaa vasten — kuulet jokaisen iskun erillisenä.

Veitsen värähtelyn yksityiskohdat ovat hieman monimutkaisempia kuin heilurin liike ja johdattavat meidät hyvin lähelle niitä asioita, joita aiomme tarkastella, nimittäin ''puhtaaseen'' nuottiin.

Heilurin liike on puhdas siinä mielessä, että sen kulkureitillä

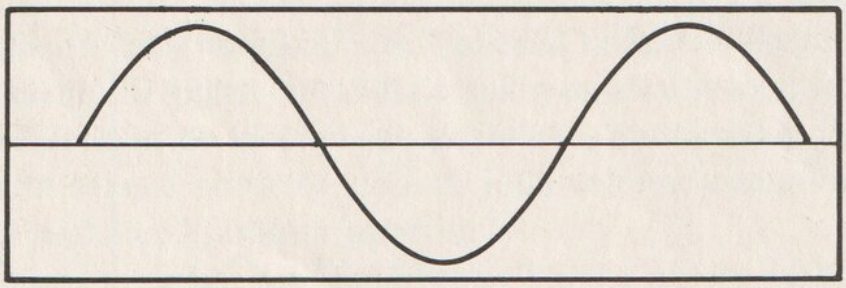


Kuva 1. Hiekkaheiluri

ei ole kohtaa, jossa joko sen nopeus tai suunta muuttuisi äkillisesti. Ääniraudan ääni on lähellä puhdasta ääntä. Puhtaat äänet ovat luonnossa hyvin harvinaisia, mutta mikä tahansa ääni voidaan muodostaa sekoittamalla sopivan taajuisista ja voimakkuisista puhtaista äänistä. Koska puhtaita ääniä on helppo tuottaa elektronisesti, ne muodostavat äänten keinotekoisen syntetisoinnin ja analysoinnin perustan.

Ymmärtääksemme tämän asian selvästi otetaan jälleen esiin hitaasti liikkuva heiluri, nyt hieman uudella tavalla. Kuvittele heilurin nupin ja sitä kannattavan tukevan varren sijasta, että meidän heilurimme muodostuu hiekka-astiasta, jota kannattaa naru. Hiekka pääsee valumaan pienestä reiästä astian pohjasta. Jos laitamme tämän heilurin liikkeeseen ja samalla vedämme pitkää paperiarkkia alapuolella, huomaamme hiekan muodostavan paperille kuvion.

Sama asia esitettynä ideaalisemmassa muodossa:

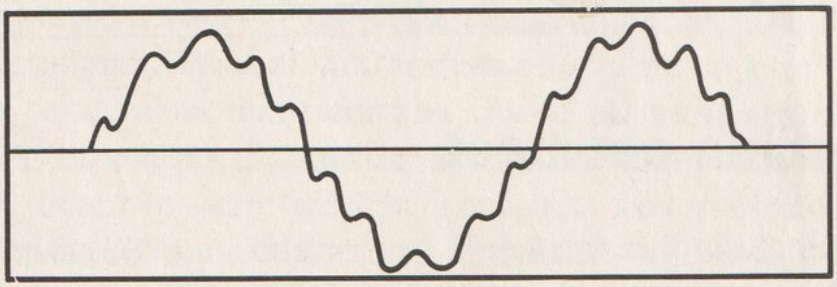


Kuva 2. Siniaallon muoto

Teoriassa hiekkaheilurin liike jatkuu samanlaisena pitkään, joten tarvitsemme vain kaksi lukua kuvaamaan liikkeen tarkasti — amplitudin (kuinka kauas keskiviivasta se ulottuu) ja taajuuden (heilahdusten määrä sekunnissa). Tämän ideaalisen liikkeen muotokäyrää kutsutaan sinikäyräksi. Jos voisimme nähdä ääniraudan liikkeen, se näyttäisi aivan samanlaiselta. Ääniraudassa kuitenkin amplitudi on suhteessa sen tuottaman äänen voimakkuuteen ja taajuus äänen sävelkorkeuteen.

Ajatellaan nyt hiekkaheiluria, joka jättää jälkensä paperille, ja kuvitellaan mitä tapahtuisi, jos iskisit sitä kannattavaan na-

ruun pienen, terävän iskun. Nyt ilmaantuisi hitaan heilumisen lisäksi ylimääräinen, paljon nopeampi värinä ja hiekka piirtäisi syntyvän liikkeen jotenkin tähän tapaan:



Kuva 3. Siniaalto, jossa on korkeataajuinen komponentti

Jos tämä olisi äänen aaltomuoto, miltä se kuulostaisi? Sävelkorkeus tuntuisi samanlaiselta kuin puhtaassa äänessä, mutta äänen laatu olisi havaittavasti erilainen — sillä olisi ''kirkkaampi'' tai ''elävämpi'' sointi. Tämän ominaisuuden tekninen termi on äänen väri.

Useimpien todellisten äänien aaltomuodot ovat hieman monimutkaisempia kuin edellisessä esimerkissä, ja on vaikeaa ellei mahdotonta tarkasti ennustaa pelkän visuaalisen muodon perusteella, miltä monimutkaiset aaltomuodot kuulostavat. Huolimatta siitä, kuinka monimutkainen sen muoto on, mikä tahansa toistuva aaltomuoto voidaan kuitenkin pilkkoa osiinsa (monimutkaisella matemaattisella menetelmällä, jota kutsutaan Fourier-analyysiksi). Kullakin näistä komponenteista on siniaalto (joilla kullakin oma amplitudinsa) ja kullakin on taajuus, joka saadaan alimman taajuuskomponentin kerrannaisena.

Alin taajuuskomponentti määrää suurimmaksi osaksi nuotin sävelkorkeuden, ja korkeampien komponenttien suhteelliset määrät tuottavat äänen värin. On helpompaa (mutta silti varsin vaikeata) ymmärtää, ennustaa ja muuntaa ääniä, jotka perustuvat tietoon sen harmonisesta sisällöstä, kuin tutkia äänen aaltomuotoa.

Katsotaan vielä seuraavaa taulukkoa ennen kuin laitamme SID-piirin töihin ja teemme joitakin ääniä.

Tekninen termi	Kuultava vastine
Taajuus	Sävelkorkeus
Amplitudi	Voimakkuus
Harmoninen sisältö	Äänen väri

SID-piirin osoitteet

Kuvittele SID-piiri taloksi ja sen tutkimista seikkailupeliksi. Päästäksemme tiettyyn huoneeseen meidän täytyy tietää talon osoite ja huoneen nimi. Tässä vaiheessa aiomme tutkia vain niitä huoneita, joilla on tekemistä taajuuden ja amplitudin (tai sävelkorkeuden ja voimakkuuden) kanssa. Kun ajatellaan taas SIDiä tietokoneen piirinä, sinne päästään samalla tavalla kuin mihin tahansa muistin osaan. Niinpä syötämme tietoa SIDille POKE-käskyllä. SID-piirissä on 29 rekisteriä (rekisteri on sama kuin muistipaikka) alkaen muistipaikasta 54272. Koska ei ole helppoa jatkuvasti pitää mielessä tällaista lukua, annetaan muistaminen tietokoneelle näppäilemällä seuraava ohjelmarivi:

```
10 SID=54272
```

(Muista, että Commodore BASIC käyttää vain muuttujanimen kahta ensimmäistä merkkiä, joten emme voi käyttää muita muuttujanimiä, jotka alkavat kirjaimilla SI). SID pystyy käsittelemään kolmea erillistä ääntä. (Osoitteita voit kuvitella talon kolmeksi kerrokseksi). Kukin ääni tarvitsee taajuuden, joten meidän on tiedettävä, mihin tallennetaan tieto sävelkorkeudesta:

```
20 F1=SID: F2=SID+7: F3=SID+14
```

Aloittelijoiden on kuitenkin viisainta toimia vain yhdellä äänellä kerrallaan, joten vaimennetaan äänet kaksi ja kolme nollaamalla niiden taajuusrekisterit:

```
30 FOR I=0 TO 1: POKE F2+I,0: POKE F3+I,0: NEXT I
```

Ääntä kohti vaaditaan kaksi POKEa kun taajuustieto asetetaan rekistereihin. Se johtuu siitä, että taajuutta kuvaava luku voi saada arvoja väliltä 0—65535, ja näin suuret luvut eivät mahdu yhteen muistipaikkaan. SID ratkaisee ongelman käyttämällä kahta peräkkäistä muistipaikkaa, joista ensimmäinen sisältää luvun alemman osan (ajattele sitä vaikka desimaaliosaksi) ja toinen ylemmän osan (ajattele sitä vaikka kokonaisosaksi).

Oletetaan, että meillä on luku F väliltä 0—65535 ja haluamme laittaa sen SID-rekisteriin. Käytämme seuraavia kaavoja: FH = INT(F/256) ja FL = F—256*FH. FH merkitsee taajuuden ylempää osaa (Frequency High) ja FL taajuuden alempaa osaa (Frequency Low). Kuusnelosen mikroprosessori vaatii, että nämä kaksi lukua on tallennettu peräkkäisiin rekistereihin niin, että alempi (vähemmän merkitsevä) osa on ensin ja ylempi (eniten merkitsevä) osa on toisena. Niinpä laittaisimme tämän luvun taajuusrekisteriin 1 seuraavilla käskyillä: POKE F1,FL:POKE F1 + 1,FH (muista, että F1 ja F1 + 1 ovat rekisterien osoitteita ja FL ja FH ovat lukuja, jotka laitetaan rekistereihin.)

Onko juuri taajuusrekisteriin asettamamme luku se taajuus, jolla ääni alkaa kuulua? Ei. Taajuus hertseinä (Hz) on suhteessa taajuusrekisterissä olevaan lukuun seuraavan kaavan mukaan:

F = HZ * 16.40439

Annetaan nyt äänelle 1 taajuus 1000 Hz. BASICillä se käy näin:

```
40 HZ=1000: C=16.40439
50 F=HZ*C
60 FH=INT(F/256)
70 FL=F-256*FH
80 POKE F1,FL: POKE F1+1,FH
```

Vaikka syöttäisit tämän esimerkkiohjelman, et vielä saisi mitään ääntä aikaan. Se johtuu siitä, että SID ei vielä tiedä tar-

peeksi hyvin mitä haluat. SID-piirin pitää tietää ainakin kolme muuta asiaa ennen kuin se voi tehdä ääniä. Ne ovat yleinen voimakkuus (kaikkien 3 äänen voimakkuustaso), verhokäyrän muoto (2 rekisteriä kullekin äänelle) ja aaltomuodon tyyppi (1 rekisteri kullekin äänelle). Nämä selostetaan myöhemmin tarkemmin, mutta lisätään tässä vaiheessa näiden rekisterien osoitteet ohjelmaamme:

```
90 W1=F1+4: W2=F2+4: W3=F3+4: REM AALTOMUOTOREKISTERIEN OSOITTEET
100 A1=F1+5: A2=F2+5: A3=F3+5: REM NOUSU/TASAANTUMISREKISTERIEN OSOITTEET
110 S1=F1+6: S2=F2+6: S3=F3+6: REM JÄLKISOINTI/LASKUREKISTERIEN OSOITTEET
120 VOL=SID+24: REM YLEISVOIMAKKUUSREKISTERIN OSOITE
```

Välittämättä sen enempää luvuista, jotka asetetaan näihin rekistereihin (niistä myöhemmin), lisää seuraavat ohjelmarivit:

```
130 POKE VOL,15
140 POKE A1,85: POKE S1,85
150 WV=32: POKE W1,WV+1
160 FOR I=0 TO 1000: NEXT I
170 POKE W1,WV
```

Jos olet näppäillyt nämä rivit tietokoneeseen ja ajanut ohjelman, olet lopultakin tuottanut ääntä SID-piiristä. Nyt kannattaa tallentaa ohjelmarivit 10—120, koska käytämme niitä myöhemmin. Tallennettuasi ohjelman voit vapaasti kokeilla erilaisia lukuja rekistereihin A1 ja S1 (rivi 140), lukuja 128, 64, 32 tai 16 muuttujaan WV rivillä 150 ja silmukan loppuarvoja rivillä 160.

Yhteenveto tähän mennessä käytetyistä osoitteista ja niiden merkityksestä:

Nimi	Osoite	Merkitys
Ääni 1:		
SID	54272	SID-piirin perusosoite
F1	SID	Tajuus, alempi osa
	F1 + 1	Tajuus, ylempi osa
W1	F1 + 4	Aaltomuodon kontrollirekisteri
A1	F1 + 5	attack/decay-rekisteri
S1	F1 + 6	sustain/release-rekisteri

Ääni 2:

F2	SID + 7	Taajuus, alempi osa
	F2 + 1	Tajuus, ylempi osa
W2	F2 + 4	Aaltomuodon kontrollirekisteri
A2	F2 + 5	attack/decay-rekisteri
S2	F2 + 6	sustain/release-rekisteri

Ääni 3:

F3	SID + 14	Taajuus, alempi osa
	F3 + 1	Taajuus, ylempi osa
W3	F3 + 4	Aaltomuodon kontrollirekisteri
A3	F3 + 5	attack/decay-rekisteri
S3	F3 + 6	sustain/release-rekisteri

Kaikki Äänet:

VOL	SID + 24	Voimakkuusrekisteri (0—15)

BASIC-ohjelma, jolla määritellään SID-osoitteet

```
10 SID=54272: REM PERUSOSOITE
20 C=16.40439: VAKIO HERTSIEN MUUNTAMISEKSI REKISTERIARVOIKSI
30 F1=SID: F2=SID+7: F3=SID+14: REM TAAJUUSREKISTERIEN OSOITTEET
40 W1=F1+4: W2=F2+4: W3=F3+4: REM AALTOMUODON KONTROLLIREKISTERIEN OSOITTEET
50 A1=F1+5: A2=F2+5: A3=F3+5: REM NOUSU/TASAANTUMISREKISTERIEN OSOITTEET
60 S1=F1+6: S2=F2+6: S3=F3+6: REM JÄLKISOINTI/LASKUREKISTERIEN OSOITTEET
70 VOL=SID+24: REM YLEISVOIMAKKUUSREKISTERIN OSOITE (REK. SAA ARVOJA 0-15)
```

Kirjoita ja tallenna tämä ohjelman osa, ellet ole sitä jo tehnyt, sillä sitä tullaan jatkossa käyttämään aina avauksena SID-ohjelmien alkuasetuksiin.

Äänen dynamiikasta

Tässä vaiheessa sinulla pitäisi olla jonkinlainen käsitys äänen luonteesta, taajuuden, sävelkorkeuden ja äänen värin yleisestä merkityksestä. Nyt lähdemme tarkastelemaan niitä SID-rekistereitä, joilla on tekemistä aaltomuodon tyypin ja kontrol-

lin kanssa. Nuo rekisterit käsittelevät nousuaikaa (attack), tasaantumisaikaa (decay), jälkisointitasoa (sustain) ja laskuaikaa (release) ja eräillä rekistereillä säädellään äänen voimakkuutta ja suodatusta.

Aaltomuodon kontrolli-rekisteri

Rekistereillä, joita olemme kutsuneet nimillä W1, W2 ja W3, on kaksi tarkoitusta, kuten useillakin SID-rekistereillä. Kukin tietokoneesi muistipaikka voi sisältää luvun välillä 0—255. Tämä luku koostuu kahdeksasta yksittäisestä bitistä — joista kukin voidaan asettaa ykköseksi tai nollaksi, kytkeä mukaan tai pois. Bitit numeroidaan nollasta (oikeanpuoleisin) seitsemään (vasemmanpuoleisin). Aaltomuodon kontrollirekisterin yksittäisillä biteillä on seuraava erityinen merkitys:

Bitti n:o	Paino	Merkitys
7	128	Kohina-aaltomuodon valinta
6	64	Pulssiaaltomuodon valinta
5	32	Saha-aaltomuodon valinta
4	16	Kolmioaaltomuodon valinta
3	8	Äänen ulostulon katkaisu
2	4	Rengasmodulaation kytkentä
1	2	Synkronoinnin kytkentä
0	1	ADSR-aloitus

Painoluvut lisätään yhteen jokaisesta bitistä, joka halutaan kytkeä päälle, ja näin saadaan luku väliltä 0—255. Tällä tavalla voimme säädellä yksittäisiä bittejä. Käytetään esimerkiksi muuttujaa WC kuvaamaan aaltomuodon kontrollirekisterin sisältöä äänelle 1 (tämä oli asetettu WV:ksi edellisessä luvussa). Olkoon alkutilanteessa WC:n sisältö mikä tahansa, ja me haluamme kääntää vain yhden bitin, esim. bitin 7, niin voimme kirjoittaa:

```
WC=WC OR 128:POKE W1, WC
```

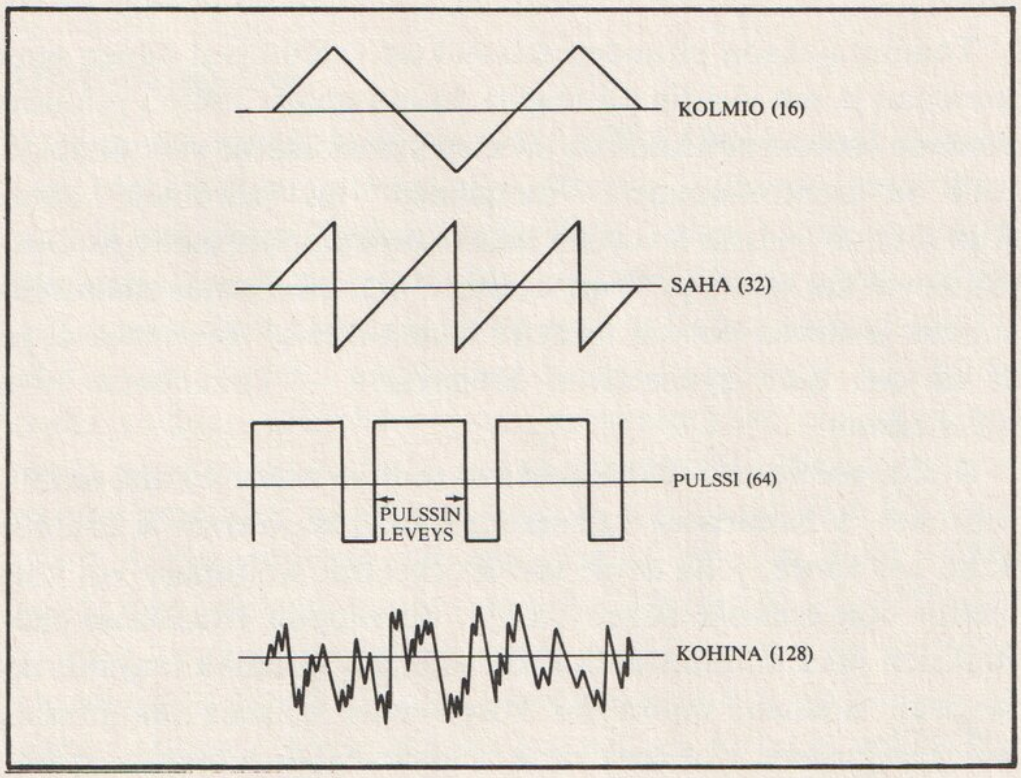
Käytämme muuttujaa WC sen takia, että PEEK ei toimi SID-rekisterien yhteydessä. WC on muuttuja, joka sisältää saman tiedon kuin muistipaikka W1. Voimme kutsua SID-rekistereitä ''kirjoitusrekistereiksi'', sillä vaikka voimme POKElla tallentaa niihin tietoa, emme voi sitä lukea.

Jos haluamme kytkeä pois vain bitin 7, voimme näppäillä:

```
WC=WC AND 254:POKE W1, WC
```

Jos tämä tuntuu oudolta, sinun kannattaa lukea uudelleen liitteestä 2 AND- ja OR-operaatioista. Useat SID-rekisterit ovat bittiorientoituneita, joten sinun tulee osata kontrolloida yksittäisiä bittejä, jos aiot todella ymmärtää SID-piirin ohjelmointia.

Kuhunkin SID-ääneen on liityttävä aaltomuodon tyyppi. Niitä ovat kolmioaalto, saha-aalto, pulssiaalto tai kohina, ja niitä kontrolloivat yksittäiset bitit.



Kuva 4. SID-aaltomuodot

Käsitellään lyhyesti näiden aaltomuotojen laatua, äänen väriä. Yksinkertaisin ja puhtain on kolmioaalto (bitti n:o 4), jonka yksi aalto koostuu suoraviivaisesti huippuunsa kasvavasta arvosta ja sitä seuraavasta suoraviivaisesta laskusta minimiin, aivan kuten jo nimestä voi päätellä. Vaikka käyrä muuttuu jyrkästi huipulla ja pohjalla, varsinaisissa arvoissa pitkin aaltoa ei ole jyrkkää muutosta.

Seuraava SIDin tuottama aaltomuoto on saha-aalto (bitti n:o 5). Kuten nimi sanoo, se koostuu suoraviivaisesta kasvusta minimistä maksimiin ja jyrkästä putouksesta maksimista minimiin. Tämä jyrkkä amplitudin muutos saa aikaa rikkaan äänen ja ”kirkkaan” värin.

Viimeinen jaksollinen aaltomuoto, jonka SID pystyy tuottamaan, on pulssiaalto (bitti n:o 6). Se koostuu osajaksosta, jonka aikana amplitudilla on negatiivinen vakioarvo ja sitten toisesta, jolla on positiivinen vakioarvo lopun jakson ajan. Pulssiaallon kuvaamiseen tarvitaan myös toinen luku — pulssin leveys.

Toimintajakson pituuden muutokset vaikuttavat äänen harmoniaan ja sen yleisiin piirteisiin. Muuttamalla jakson pituutta voidaan tuottaa neliöaaltoa, joka on kolmioaaltoa rikkaampi ja jolle on luonteenomaista ”metallinen” tai ”säröinen” ääni. Kun muutat pulssin leveyden lähelle nollaa, pulssiaalto kuulostaa kovin karkealta ja ”surisevalta”. SID-piiri antaa mahdollisuuden muuttaa pulssin leveyttä äänen ollessa toiminnassa, ja se tarjoaa näin dynaamisen äänenvärin — kokeilemme sitä myöhemmin.

Kukin kolmesta SID-äänestä voi tuottaa myös kohina-aaltoa (bitti n:o 7). Kohinassa on mukana monia taajuuksia ja kohina-aalto luo ääntä, jolla ei ole sävelkorkeutta. Kohinassa voi kuitenkin olla erilaisia äänen värejä. Se riippuu pääasiassa taajuuksien jakautumisesta. Jos korostetaan korkeita taajuuksia, saadaan suhinaa, mutta jos korostetaan matalia taajuuksia, saadaan jyrinää. Kohinan väriä voidaan SIDissä muuttaa vaihtelemalla sopivien rekisterien taajuusarvoja äänen kuuluessa.

Tehdäänpä yhteenvetoa: ennen kuin SID-ääntä voidaan

tuottaa, sille on määrättävä aaltomuoto — kohina, pulssi, saha tai kolmio sen mukaan, miten aaltomuodon kontrollirekisterin bitit n:o 7, 6, 5 tai 4 on asetettu. Kullakin on ominainen äänenvärinsä ja pulssiaallon ja kohinan värejä voidaan säädellä dynaamisesti. Ennen kuin mennään eteenpäin, on syytä muistuttaa eräästä tärkeästä seikasta. Kerrallaan voidaan valita vain yksi aaltomuoto. Kullakin äänellä voi kuitenkin olla oma aaltomuotonsa.

Aaltomuodon kontrollirekisterien (W1, W2 ja W3) bitit n:o 3, 2 ja 1 eivät koske meitä vielä; bitti 0 on kirjaimellisesti ''avain'' SID-piirin kytkemiseksi. Kun bitti on ''päällä'', ääni alkaa kuulua. Kun bitti asetetaan ''pois päältä'', ääni alkaa vaimeta. Oletetaan, että olet asettanut WC:hen sopivan aaltomuotoyypin (128, 64, 32 tai 16). Voit nyt aktivoida äänen n:o 1 käskyllä POKE W1,WC + 1 ja vapauttaa äänen n:o 1 käskyllä POKE W1,WC. Voit myös hiljentää äänen n:o 1 koska tahansa asettamalla sen taajuusrekisteriin 0 käskyllä POKE F1,0:POKE F1 + 1,0 jne.

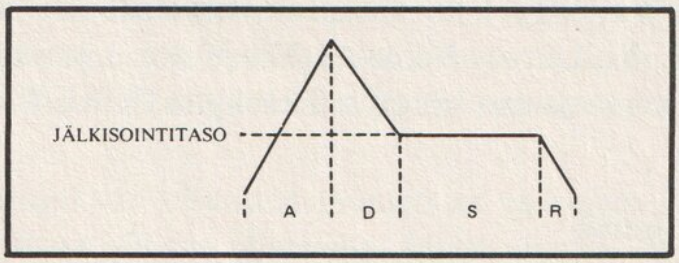
ADSR-järjestelmä

Olemme nyt valmiit tutkimaan ns. ADSR-järjestelmää. Nousuaika (Attack), tasaantumisaika (Decay), jälkisointitaso (Sustain) ja laskuaika (Release) määräävät, miten äänen amplitudi (tai voimakkuus) muuttuu ajan mukana. Monia ääniä, erityisesti musiikillisiä ääniä, voidaan matkia ''amplitudipaketilla'', joka koostuu edellä mainituista neljästä vaiheesta. SID-piirin avulla voidaan säädellä nousun, tasaantumisen ja laskun nopeutta eli sen ajan pituutta, jonka ne kestävät. Jälkisointitason arvo on amplitudi tai voimakkuus, ei nopeus tai aika. Kullakin näistä ADSR-vaiheista voi olla yksi 16 arvosta, luku väliltä 0—15, joka voidaan muodostaa neljällä bitillä. Kukin SID-tavu koostuu kahdeksasta bitistä, joten ADSR-vaiheet on paritettu seuraavasti: attack/decay (nousuaika/tasaantumisaika) ja sustain/release (jälkisointitaso/laskuaika). Kutakin neljää bittiä kutsutaan puolitavuksi.

Nousuajan ja tasaantumisajan puolitavut ovat rekistereissä

A1, A2 ja A3 (kullakin äänellä voi olla oma ADSR-pakettinsa) ja jälkisointitason ja laskuajan puolitavut ovat rekistereissä S1, S2 ja S3. Äänen alkaminen on eräänlainen portin aukaiseminen. Niinpä sitä kutsutaan äänen aukaisuksi. Seuraavassa kuvataan tapahtumien järjestys, kun ääni avataan kytkemällä toimintaan bitti 0 aaltomuodon kontrollirekisterissä.

Äänen amplitudi eli voimakkuus kasvaa nollasta maksimiin nopeudella, jonka määrää nousuajan puolitavu. Sitten se vaimenee nopeudella, jonka määrää tasaantumisajan puolitavu, kunnes voimakkuus saavuttaa jälkisointitason puolitavussa määritellyn arvon. Ääni pysyy samalla tasolla, kunnes se suljetaan kytkemällä pois aaltomuodon kontrollirekisterin bitti 0. Sitten voimakkuus vähenee nopeudella, joka on määritelty laskuajan puolitavussa.



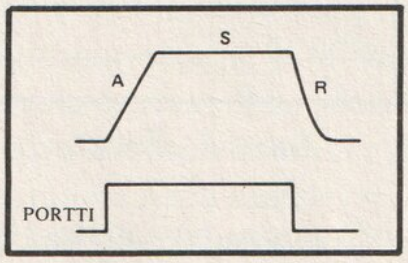
Kuva 5. ADSR-verhokäyrä

Tässä taulukossa annetaan 16 puolitavun arvoja vastaavat todelliset nopeudet (ms = millisekunteja, s = sekunteja):

Puolitavu	Nousuaika (A)	Tasaantumisaika (D)/Laskuaika (R)
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

Nousu- ja tasaantumisnopeudet asetetaan rekistereihin A1, A2 ja A3 (kullakin äänellä on oma ADSR:nsä) ja jälkisointitaso sekä laskunopeus asetetaan rekistereihin S1, S2 ja S3. Nämä asetetaan vain kerran eikä joka kerta kun ääni toistetaan. Kokeillaan esimerkillä, että ymmärrämme kaiken välttämättömän määritelläksemme tietyn ADSR-paketin. Puhaltimen äänessä, esimerkiksi trumpetin äänessä, on suuri nousu- ja tasaantumisnopeus, suhteellisen korkea jälkisointitaso ja varsin suuri laskunopeus. Käytetään 16 ms:n nousuaikaa, 72 ms:n tasaantumisaikaa, jälkisointitasoa 12 ja laskuaikaa 114 ms. Edellisen taulukon avulla saadaan:

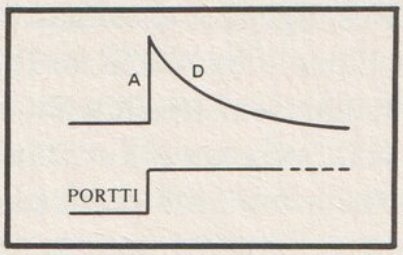
AD = (nousuaika/tasaantumisaika) = 16*2 + 3
SR = (jälkisointitaso/laskuaika) = 16*12 + 4



Kuva 6. Puhallettu ääni

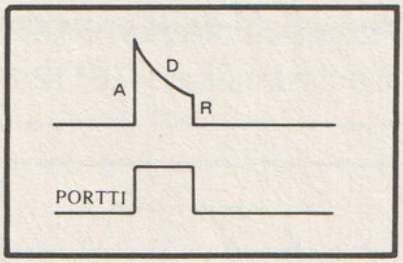
Annetaan äänelle 1 käskyt POKE A1,35:POKE S1,196. Kuuluuko mitään? Muista, että on määriteltävä myös aaltomuoto ja taajuus sekä kytkettävä ääni mukaan. Muista myös, että lasku ei tapahdu ennen kuin suljet äänen pois. Seuraavassa esitetään muutamia ADSR-paketteja, jotta saisit selvemmän kuvan siitä mitä kaikkea tarvitaan. Kokeile aaltomuodoilla saadaksesi haluamasi äänenvärin. Ensimmäinen luku on nousuaika/tasaantumisaika-rekisteriä varten, toinen jälkisointitaso/laskuaika-rekisteriä varten.

Lyömäsoitin — 9, 8 = hyvin nopea nousuaika (16*0), kohtalainen tasaantumisaika (9), jälkisointitaso nolla (16*0) ja keskinkertainen laskuaika (8). (Yritä kohinalla.)



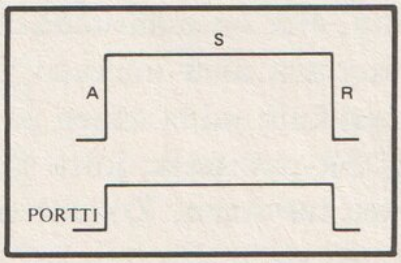
Kuva 7. Lyömäsoitin

Piano — 10, 48 = hyvin nopea nousuaika (16*0), keskitasoinen tasaantumisaika (10), matala jälkisointitaso (16*4) ja hyvin nopea laskuaika (0). (Yritä saha-aallolla.)



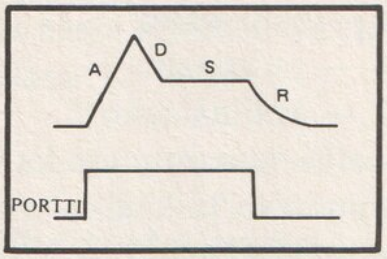
Kuva 8. Piano

Urut — 0, 240 = hyvin nopea nousuaika (16*0), ei tasaantumisaikaa (0), korkea jälkisointitaso (15*16) ja hyvin nopea laskuaika (0). (Yritä pulssiaallolla.)



Kuva 9. Urut

Viulu — 133, 165 = kohtalaisen hidas nousuaika (8*16), kohtalainen tasaantumisaika (5), keskinkertainen jälkisointitaso (10*160) ja keskinkertainen laskuaika (5). (Yritä pulssiaalloilla.)



Kuva 10. Viulu

Pidä mielessäsi, että SID-piirin pääavu on luoda ääniä, joita ei saada helposti aikaan todellisilla soittimilla. Voidaan esimerkiksi helposti kääntää normaalityyppinen ADSR päinvastaiseksi ja saadaan hyvin outo ääni.

Kuten olemme nähneet, SIDillä on monta rekisteriä ja toimintaa, ja on hyödyllistä, jopa välttämätöntä pitää ne järjestyksessä, mikäli aiomme käyttää sen täyttä kapasiteettia. SIDin toiminnot voidaan kätevästi jakaa kahteen ryhmään: niihin jotka vaikuttavat yksittäisiin ääniin, ja niihin jotka vaikuttavat kaikkiin ääniin samalla kertaa. Kun aloitetaan SIDin perusosoitteesta, ensimmäiset kolme seitsemän ryhmää ovat yksittäisiä ääniä varten. Seuraavat neljä säätelevät suodatinta ja äänen voimakkuutta ja jäljelle jäävät neljä ovat erityistarkoituksiin. (Emme ole päässeet niihin asti vielä.) Luetellaanpa rekisterit ja niiden sisältö:

Rekisteri	Merkitys
0, 7,14	Taajuus (alempi tavu)
1, 8,15	Taajuus (ylempi tavu)
2, 9,16	Pulssin leveys (alempi tavu)
3,10,17	Pulssin leveys (bitit 0—3 muodostavat ylemmän puolitavun)
4,11,18	Kontrollirekisteri
	Bitti 0 — portti (1 = auki, 0 = kiinni)
	Bitti 1 — synkronointibitti
	Bitti 2 — rengasmodulaatiobitti
	Bitti 3 — testibitti
	Bitti 4 — kolmioaalto

	Bitti 5 — saha-aalto
	Bitti 6 — pulssiaalto
	Bitti 7 — kohina-aalto
5,12,19	Nousuaika/tasaantumisaika-rekisteri
6,13,20	Jälkisointitaso/laskuaika-rekisteri
21	Suodatustaajuus (bitit 0,1,2 muodostavat alemman osan)
22	Suodatustaajuus (ylempi tavu)
23	Resonanssi/suodatin-rekisteri
	Bitti 0 — kytkee äänen 1 suodattimeen
	Bitti 1 — kytkee äänen 2 suodattimeen
	Bitti 2 — kytkee äänen 3 suodattimeen
	Bitti 3 — kytkee ulkoisen signaalin suodattimeen
	Bitit 4—7 — suodattimen resonanssi
24	Tila/voimakkuus
	Bitit 0—3 — yleisvoimakkuus
	Bitti 4 — alipäästösuodattimen valinta
	Bitti 5 — kaistapäästösuodattimen valinta
	Bitti 6 — ylipäästösuodattimen valinta
	Bitti 7 — kytketään ääni 3 pois audio ulostulosta

Seuraavat ovat vain luettavissa:

25	POTX
26	POTY
27	Ulostulo oskillaattorista 3
28	Äänen 3 verhokäyrä

Suhteellisen yksinkertaisille äänille (kuten musiikissa käytettäville, vastakohtana ääniefekteissä käytettäville) ovat tärkeimmät käsittelemämme rekisterit SID + 0:sta SID + 20:een. Niillä asetetaan taajuudet, aaltomuodot, ADSR-paketit ja äänen aloitus (porttibitti = 1) ja lopetus (porttibitti = 0). Alempi muistipaikassa SID + 24 oleva puolitavu säätelee yleistä äänen voimak-

kuutta. Musiikillisia ääniä ajatellen on järkevää käsitellä kaikkia näitä tiedon palasia.

Hyvin viritetty tietokone

Tähän mennessä emme ole kuulleet kovinkaan paljoa ääniä SIDistä; olemme pääasiassa tarkastelleet ääniteoriaa ja SIDin järjestelmää. Katsomme hetken, millaisia taajuuksia pitää käyttää, ja sitten olemme valmiit soittamaan hieman musiikkia SIDin avulla. Länsimainen musiikki on aika selväpiirteistä. Sävelasteikko koostuu 12 nuotista (do, re, mi, fa, sol, la, ti ja 5 ylennettyä tai alennettua nuottia välissä) oktaavia kohden. Kussakin oktaavissa taajuudet perustuvat edelliseen oktaaviin. Ns. kromaattisessa (yleisimmin käytössä olevassa) asteikossa kukin sävelaskel saadaan edellisen kiinteänä kerrannaisena. Tämä kiinteä suhde on itse asiassa kompromissi, joka mahdollistaa sen, että viritetyt soittimet voivat soida harmoniassa keskenään.

Tämä kiinteä kerroin on (ja sen on oltava) 12. juuri 2:sta (joka on 1.0594630943593...). Tämä luku kerrottuna itsellään 12 kertaa antaa tulokseksi 2; toisin sanoen silloin päästään seuraavaan oktaaviin ja kaikki voi alkaa uudelleen kaksinkertaisella sävelkorkeudella. Periaatteessa meidän on tällöin sovittava vain yhden nuotin tarkasta taajuudesta, ja KAIKKI loput voidaan johtaa tästä kertolaskun (ylemmät nuotit) tai jakolaskun (alemmat nuotit) avulla.

Muusikot ovat kansainvälisesti sopineet, että nuotin A4 (''la'' neljännessä oktaavissa) tarkka taajuus on 440 Hz. Tästä tiedosta ja muunnostekijästä, joka on annettu taajuusrekisteriä koskevan kappaleen yhteydessä, voimme laskea sopivan kahden tavun luvun kullekin nuotille, jonka haluamme soittaa. Kaikkea tätä työtä ei toki tarvitse tehdä joka kerta kun haluamme soittaa nuotin. Sen sijaan käytämme hyväksemme oktaavin suhteita ja lyhennämme työtä huomattavasti laskemalla taulukon vain 12 taajuudelle (13. aloittaa seuraavan korkeamman

oktaavin). Tästä saamme nopeasti sopivat arvot toisten oktaavien nuoteille jakamalla kahdella riittävän monta kertaa.

Seuraavassa on BASIC-ohjelma, joka laskee (kerran ja kaikille) taajuusarvot ”ylimmälle” oktaaville. (Liitä tämä aikaisemmin annettuun perusosoitteet määrittelevään ohjelmaan.)

```
100 DIM F(12): REM F(1) ON SID-TAAJUUS NUOTILLE A6, F(12) NUOTILLE G#7
110 A7=6*440: REM TAAJUUS HERTSEISSÄ KORKEIMMALLE A-NUOTILLE 7. OKTAAVISSA
120 K=2↑(1/12): REM TASAVÄLINEN ASTEIKKO
130 FRQ=A7*C: REM MUUNNETAAN TAAJUUS SID-ARVOKSI (VAKIO C PERUSOHJELMASSA)
140 FOR I=12 TO 1 STEP -1
150 FRQ=FRQ/K: REM SEURAAVA PUOLIASKEL ALASPÄIN
160 F(I)=FRQ
170 NEXT I
```

Tyhjennetään tässä vaiheessa SID-rekisterit estääksemme vanhojen ohjelmien perua olevien arvojen vaikutukset ja asetetaan joitakin alkuarvoja:

```
200 FOR I=0 TO 28: POKE SID+I,0: NEXT I
210 POKE VOL,15: REM ÄÄNEN VOIMAKKUUS MAKSIMIIN
220 TI$="000000": REM AJASTIN NOLLATAAN
230 DIM N(12): REM PARAMETREJÄ VARTEN
240 FOR I=1 TO 9: READ N(I): REM LUETAAN ÄÄNIPARAMETRIT KULLEKIN ÄÄNELLE
250 NEXT I
260 REM SEURAAVA DATA ON VAIN ESIMERKIKSI, KOKEILE VAPAASTI OMIA DATOJASI
270 DATA 16,16,16: REM AALTOMUOTO KULLEKIN ÄÄNELLE
280 DATA 55,55,55: REM NOUSUAIKA/TASAANTUMISAIKA KULLEKIN ÄÄNELLE
290 DATA 55,55,55: REM JÄLKISOINTITASO/LASKUAIKA KULLEKIN ÄÄNELLE
300 POKE W1,N(1): POKE W2,N(2): POKE W3,N(3): REM ASETETAAN ADSR
310 POKE A1,N(4): POKE A2,N(5): POKE A3,N(6)
320 POKE S1,N(7): POKE S2,N(8): POKE S3,N(9)
330 FOR I=1 TO 3: REM HUOLEHDITAAN PULSSISTA
340 READ W: N(I+9)=W*4096
360 NEXT I
370 DATA .5,.5,.5: REM TULEE OLLA VÄLILLÄ 0-.5
380 POKE F1+3,INT(N(10)/256): POKE F1+2,N(10)-256*INT(N(10)/256)
390 POKE F2+3,INT(N(11)/256): POKE F2+2,N(11)-256*INT(N(11)/256)
400 POKE F3+3,INT(N(12)/256): POKE F3+2,N(12)-256*INT(N(12)/256)
```

(Muista, että toimintajakson leveys vaikuttaa ääneen vain jos olet valinnut pulssiaallon arvolla 64 ko. rekisteriin.)

Jos aiot myöhemmin harrastaa musiikin ohjelmointia, tallenna koko ohjelma myöhempää käyttöä varten: rivit 10—70 (määrittelevät SID-osoitteet), rivit 100—160 (laskevat ylimmän oktaavin taajuudet) ja rivit 200:sta eteenpäin. (Voimme muuttaa tätä osaa myöhemmin). SID-piiri on nyt valmis soittamaan musiikkia, kunhan vain annamme sille DATA-lauseissa syöttötietoja. Aloitetaan yksinkertaisesti käyttämällä vain yhtä ääntä mutta muutellen aaltomuotoa ja ADSR-tietoja.

```
490 S=1: REM TEMPOVAKIO
500 READ TM: REM LUETAAN NUOTIN KESTO
510 IF TM=0 THEN 9999: REM LOPPU JOS TM=0
520 IF TM<0 THEN 9999: REM LOPPU JOS TM<0
530 T=TI+S*TM: REM KERTOO KOSKA NUOTIN SOITTAMINEN PITÄÄ LOPETTAA
540 READ N,O: REM LUETAAN NUOTTI JA OKTAAVI
550 FR=F(N)/(2*O):REM LASKETAAN TAAJUUS SIDILLE
560 FH=INT(FR/256): FL=FR-256*FH
570 POKE F1,FL: POKE F1+1,FH: REM ASETETAAN TAAJUUS
580 POKE W1,N(1)+1: REM NOUSUAIKA ALKAA
590 IF TI<T THEN 590: REM ODOTETAAN NUOTIN PÄÄTTYMISTÄ
600 POKE W1,N(1): REM LASKUAIKA ALKAA
610 GOTO 500: REM LISÄÄ DATAA
700 DATA 40,4,2
710 DATA 40,6,2
720 DATA 40,8,2
730 DATA 40,9,2
740 DATA 40,11,2
750 DATA 40,1,1
760 DATA 40,3,1
770 DATA 40,4,1
780 DATA 0
9999  FOR I=0 TO 28: POKE SID+I,0: NEXT I:END:REM HILJENNETÄÄN SID JA
LOPETETAAN
```

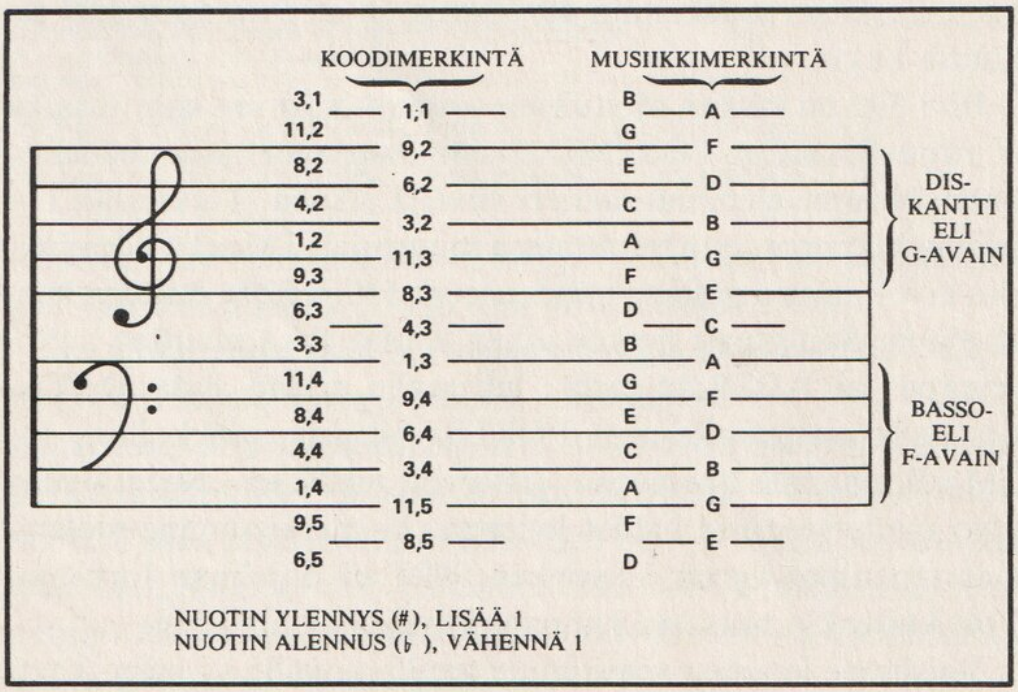
Suorita nyt ohjelma. Tätä on hyvä kokeilla erilaisilla aaltomuodoilla ja ADSR:illä. Yritä pulssiaaltoa (aseta 64 ensimmäiseksi luvuksi rivillä 270) ja käytä erilaisia pulssin leveyksiä (ensimmäinen luku rivillä 370). Rivillä 590 käytetyllä muuttujalla TI on erityinen merkitys. Se on tietokoneen sisäinen kellomuuttuja, aivan kuten TI$, paitsi että se on numeerinen muuttuja (eikä merkkijonomuuttuja kuten TI$) ja se mittaa aikaa 1/60-sekunneissa. Niinpä 40:t DATA-lauseissa riviltä 700 alkaen merkitsevät sekunnin 40/60-osia eli ne ovat noin 0.67 sekuntia kukin.

Rivi 590 on tarkan ajoituksen avain, koska sen suoritusaika ei riipu BASICin FOR/NEXT-silmukan suoritusnopeudesta, kuten olemme aiemmin saaneet viiveitä aikaan. Tutkiessasi ohjelmaa huomaat, että testattaessa muuttujaa TM sekä negatiivisiin arvoihin että nollaan nähden (rivit 510 ja 520), etsitään kahta ehtoa. Soitettuasi hetken aikaa haluat ehkä muuttaa aaltomuotoa tai ADSR:ää esim. lukemalla sisään lisää DATA-tietoja ja asettamalla niitä POKElla oikeisiin paikkoihin. Jos TM = 0, ohjelma haarautuu, jotta voit tehdä sen. Negatiivinen arvo merkitsee, että haluat lopettaa. Esimerkissämme molemmat tapaukset vievät lauseeseen 9999 eli ohjelman loppuun. Voit kuitenkin muuttaa hyppyosoitteita haluamiksesi.

Voisimme koodata nuottimme useilla tavoilla — esim. käyt-

täen merkkijonoarvoja, kuten A2, C#3, Bbl ... tai käyttämällä suoraan taajuuslukuja, kuten 880, 2113, 987.2 ... tai do, re, mi ... tai ehkä jopa graafisesti, kuten nuottiviivastolla. Voisimme myös tehdä jotain samansuuntaista nuottien keston suhteen. Seuraava tapa on kuitenkin luotettavuuden ja koodausnopeuden hyvä kompromissi. Se ei tietenkään ole ainoa ja oikea, joten jos myöhemmin keksit mukavamman tavan, siirry siihen ja ohjelmoi sen avulla. Mutta pitäydytään nyt tähän menetelmään ja tutustutaan seuraavaan taulukkoon ja kuvaan, jotka auttavat nuottien kääntämisessä:

Nuotti #	Musiikkisymboli
1	la tai A
2	(le) tai A# tai Bb
3	ti tai B
4	do tai C
5	(di) tai C# tai Db
6	re tai D
7	(ra) tai D# tai Eb
8	mi tai E
9	fa tai F
10	(fi) tai F# tai Gb
11	sol tai G
12	(si) tai G# tai Ab



Huomaat nyt miksi käytimme jaksoa 4,6,8,9,11,1,3,4 nuottikoodeinamme soittaessamme C-asteikon ja miksi muutimme oktaavista 2 oktaaviin 1 11:n ja 1:n välillä. Normaalissa musiikin kirjoituksessa nuottien kestot ilmaistaan joko suoraan kokonuotin osina, kuten 1/4-nuotti, 1/2-nuotti jne., tai graafisten symbolien avulla. Jos keinotekoisesti kutsumme 160:tä 1/60 sekuntia kokonuotin kestoajaksi, koodataan kokonuotti arvolla 160, puolinuotti arvolla 80, neljäsosanuotti arvolla 40 jne. Tällä tekniikalla voidaan käyttää myös ''epäsymmetrisiä'' kestoja, kuten 1/3-nuotteja = 53 (160/3) tai pisteellisiä neljäsosanuotteja = 60 (40 + puolet 40:stä). Titenkään sinun ei tarvitse noudattaa juuri tätä käytäntöä, vaan voit ottaa käyttöösi minkä tahansa järkevän ajoitusvakion. Tässä taulukossa on muunnettu tavanomaiset nuottimerkinnät kestokoodeiksi.

NAME	SYMBOL	VALUE	NAME	SYMBOL	VALUE
PISTEELLINEN KOKONUOTTI		240	PISTEELL. 1/8-OSAN.		30
KOKONUOTTI		160	TRIOLI NELJÄSOSIN		27
PISTEELLINEN PUOLINUOTTI		120	KAHDEKSASOSAN.		20
PUOLINUOTTI		80	TRIOLI KAHDEKSASOSIN		13
PISTEELLINEN NELJÄSOSAN.		60	1/16-OSANUOTTI		10
NELJÄSOSANUOTTI		40	1/32-OSANUOTTI		5

Kaikkihan on suhteellista, niin myös S:n arvo (rivillä 490) on suhdeluku, joka kertoo aika-arvon ja toimii näin keinonamme säädellä tempoa (musiikin yleisnopeutta). Jos S saisi arvon 2, tempo olisi kaksi kertaa hitaampi; jos se olisi 1/2, tempo olisi kaksi kertaa nopeampi. Yritä asettaa S = 2 ja S = .5 vakuuttuak-

sesi, että se toimii kuvatulla tavalla. Tässä vaiheessa sinun kannattaa ilman muuta yrittää omin käsin koodata erilaisia säveliä DATA-lauseisiin, jotta saat kokemusta nuottien ja niiden kestojen syöttämisestä.

Kun olet kokeillut tarpeeksi yhdellä äänellä, alamme tehdä jotain vielä hauskempaa, moniäänistä musiikkia. Arvannetkin miten aiomme menetellä. DATA-lauseemme tulevatkin nyt sisältämään kolme tietoryhmää yhden sijasta. Voisimme tehdä kolme POKE-käskyryhmää SIDin aktivoimiseksi. Tämä kävisi hyvin päinsä, jos kaikki kolme nuottia aina alkaisivat ja päättyisivät samanaikaisesti, mutta näin ei tavallisesti ole laita musiikissa.

Yleensä nuotit eivät ''järjesty riviin''. Tästä syntyy moniääninen sointi. Koodattaessa musiikkia tietokoneille on käytössä kaksi yleistä tekniikkaa. Ensimmäistä kutsutaan horisontaaliseksi. Siinä yksinkertaisesti pidetään ajoitus- ja taajuusvakioita kullekin äänelle erillisessä taulukossa (ikään kuin lukisit kutakin ääntä partituurista selaamalla sitä vaakasuoraan). Tämä olisi suhteellisen helppoa, jos meillä olisi erillinen ajoitin kullekin äänelle.

Toista menetelmää kutsutaan vertikaaliseksi, mikä tarkoittaa, että kaikkien kolmen äänen tiedot ryhmitellään yhteen ikään kuin ne olisivat musiikillisia akordeja. (Tämä on lähempänä sitä, miten ihmiset lukevat ja soittavat moniäänistä musiikkia.) Pystysuorassa koodauksessa tulee kuitenkin vastaan vaikeus, kun esimerkiksi yhtä ääntä on muutettava, mutta toiset kaksi soivat edelleen. Vaikka molemmat menetelmät voitaisiin ohjelmoida, pystysuora koodaus on kuitenkin helpompi.

Tämä tarkoittaa sitä, että aina kun jotain ''uutta'' tapahtuu, tietokoneen on saatava tieto siitä, esimerkiksi aina kun yhden tai useamman äänen vahvistusta tai päästöä pitää säätää. Oletetaan, että kolme nuottia alkaa yhtä aikaa, mutta yksi niistä on neljäsosanuotti ja toiset puolinuotteja. Joudumme jakamaan tämän pystysuorassa menetelmässä kahdeksi erilliseksi ''tapahtumaksi''. Kukin tapahtuma merkitsee tässä tapauksessa uutta DATA-lausetta. Ensimmäisessä tapahtumassa saatamme kaik-

ki nuotit vahvistusvaiheisiinsa. Ajoittimemme saisi arvokseen neljäsosanuotin keston. Seuraavassa vaiheessa poimisimme uuden neljäsosanuotin pituuden sekä aloittaisimme alkuperäisen nuotin laskuajan (release) tai aloittaisimme uuden nousuajan (attack) samalla kun kaksi muuta nuottia jäisivät edelleen soimaan. On tärkeätä ymmärtää, miksi kahden vaiheen summan pitää olla puolen nuotin pituus ja miksi ensimmäisen vaiheen pituuden pitää olla puoli nuottia. Jos ymmärrät tämän selvästi, sinulla on hyvät valmiudet jatkossa. Ellet ymmärrä, yritä ajatella uudestaan. Ehkä asia selviää lisäesimerkeillä.

Tässä on kolmiääninen ohjelmamme, joka alkaa ensimmäisellä yksinkertaisella esimerkillä ohjelmariveillä 800—880. Lataa vanha avausohjelmamme (riville 400 asti):

```
410 BYTE=256: Z=0: W=1: M=-1: REM VAKIOITA
420 DIM FH(12,6),FL(12,6),O(6): REM TAAJUUKSIA JA OKTAAVIN JAKAJIA VARTEN
430 FOR J=0 TO 6: O(J)=2*J: NEXT J
440 FOR I=1 TO 12: FOR J=1 TO 6
450 FR=F(I)/O(J): FH(I,J)=INT(FR/BY): FL(I,J)=FR-BY*FH(I,J)
460 NEXT J,I
490 S=1: REM TEMPOVAKIO
500 READ TM: REM LUETAAN NUOTIN KESTO
510 IF TM=0 THEN 9999: REM LOPPU JOS TM=0
520 IF TM<0 THEN 9999: REM LOPPU JOS TM<0
530 T=TI+S*TM: REM KERTOO KUINKA KAUAN NUOTTIA PITÄÄ SOITTAA
540 READ N1,O1,N2,O2,N3,O3: REM LUETAAN NUOTIT JA OKTAAVIT
550 REM LASKUAIKA ALKAA
560 IF N1 THEN POKE W1,N(1)
570 IF N2 THEN POKE W2,N(2)
580 IF N3 THEN POKE W3,N(3)
590 REM NOUSUAIKA ALKAA
600 IF N1>Z THEN POKE F1,FL(N1,O1): POKE F1+W,FH(N1,O1): POKE W1,N(1)+W
610 IF N2>Z THEN POKE F2,FL(N2,O2): POKE F2+W,FH(N2,O2): POKE W2,N(2)+W
620 IF N3>Z THEN POKE F3,FL(N3,O3): POKE F3+W,FH(N3,O3): POKE W3,N(3)+W
630 :
640 IF TI>T THEN 500: REM LISÄÄ DATAA
650 GOTO 640: REM ODOTA AJAN KULUMISTA
```

Tässä on käytetty hieman erilaista tyyliä kuin ensimmäisessä musiikkiohjelmassamme. Nopeuden parantamiseksi annoimme muuttujanimet eniten käytetyille vakioille — 256,1,0,-1 — sillä näin BASICin ei tarvitse koko ajan tehdä muunnoksia. Olemme myös laskeneet jo valmiiksi oktaavin jakajat vektoriin 0(I) ja koko taajuustaulukon taulukkoihin FH(I,J) ja FL(I,J). (REM-lauseet ja ylimääräiset tyhjät kannattaa poistaa lopullisesta versiostasi.) Nopeus on nyt huomattavasti ratkaisevampi seikka, koska on syötettävä SIDille kolme parametriryhmää yhden ase-

mesta, ja tietenkin korva on hyvin herkkä ajoituseroille, kun kolmen äänen pitäisi soida yhtaikaisesti. Tämä ohjelma kannattaa tallentaa (nimeä se vaikka SIDMUSIC3:ksi) ennen sen suoritusta.

Lisää nyt nämä DATA-lauseet:

```
800 DATA 40,04,2,04,1,00,0
810 DATA 40,06,2,03,1,00,0
820 DATA 40,08,2,01,2,00,0
830 DATA 40,09,2,11,2,00,0
840 DATA 40,11,2,09,2,00,0
850 DATA 40,01,1,08,2,00,0
860 DATA 40,03,1,06,2,00,0
870 DATA 40,04,1,04,2,00,0
880 DATA 0
9999 FOR I=0 TO 28: POKE SID+I,0: NEXT I:END:REM HILJENNETÄÄN SID JA
LOPETETAAN
```

Suoritettuasi tämän huomaat, että ”tapahtumat” ovat tässä hyvin yksinkertaisia, koska kumpikin kahdesta äänestä on synkronissa, toinen soittaa C-asteikkoa ylös ja toinen alas. Tehdäänpä asia hieman monimutkaisemmaksi, niin näet, miksi tarvitsemme lisäehdon (IF N1>Z ohjelmassa).

```
800 DATA 40,04,2,04,1,00,0
810 DATA 40,06,2,00,0,00,0
820 DATA 40,08,2,01,2,00,0
830 DATA 40,09,2,00,0,00,0
840 DATA 40,11,2,09,2,00,0
850 DATA 40,01,1,00,0,00,0
860 DATA 40,03,1,06,2,00,0
870 DATA 40,04,1,00,0,00,0
```

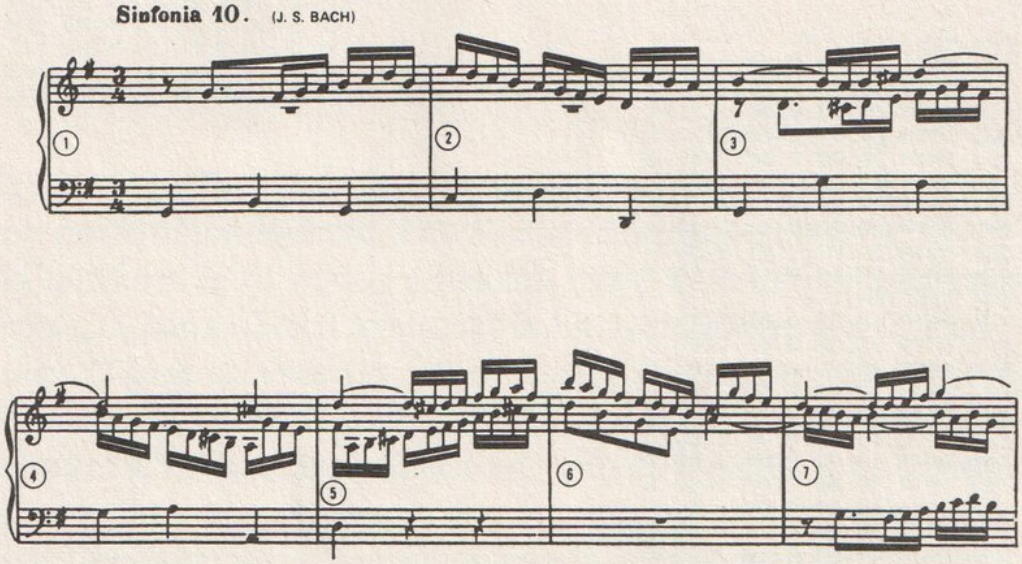
Nollat toisen äänen nuottimuuttujalle merkitsevät ”älä tee mitään”; toisin sanoen jatka jälkisointia (sustain). Niinpä meillä on nyt äänessä 1 joukko 1/4 nuotteja, jotka soivat yhdessä äänen 2 1/2-nuottien kanssa. Huomaatko, että 1/2-nuotin soittaminen ei ole sama asia kuin kahden 1/4-nuotin soittaminen? Muista, että joka kerran kun ”soitamme” nuotin, se joutuu nousuvaiheeseen. Yritä lisätä seuraavat rivit, jotta voit varmistua siitä, että ymmärrät tämän eron. (Tuhoa ensin rivi 880.)

```
900 DATA 40,04,2,04,1,00,0
910 DATA 40,06,2,04,1,00,0
920 DATA 40,08,2,01,2,00,0
930 DATA 40,09,2,01,2,00,0
940 DATA 40,11,2,09,2,00,0
950 DATA 40,01,1,09,2,00,0
960 DATA 40,03,1,06,2,00,0
970 DATA 40,04,1,06,2,00,0
980 DATA 0
```

Oleteteenpa, että nyt haluammekin muuntaa ohjelmaa niin, että saamme toisenkin äänen soittamaan 1/4-nuotteja, mutta pitämään 1/4:n tauon aina välillä. Yhden äänen ohjelmassammehan ei ollut taukomahdollisuutta. Musiikissa tauot ovat kuitenkin hyvin tärkeitä. Jotta saisimme taukoja, käytämme -1:tä nuottikoodina ilmaisemaan, että haluamme sen äänen loppuvan ilman välitöntä aluketta. Siksi tarvitsemme ohjelmassa erityisiä ehtorakenteita.

```
900 DATA 40,04,2,04,1,00,0
910 DATA 40,06,2,-1,0,00,0
920 DATA 40,08,2,01,2,00,0
930 DATA 40,09,2,-1,0,00,0
940 DATA 40,11,2,09,2,00,0
950 DATA 40,01,1,-1,0,00,0
960 DATA 40,03,1,06,2,00,0
970 DATA 40,04,1,-1,0,00,0
980 DATA 0
```

Yhdistetään nyt kaikki ja tehdään jotain oikeaa musiikkia. Mikäpä sopisi paremmin kolmiääniselle SIDille kuin Bachin kolmiosainen sinfonia? Ensimmäiset tahdit sinfoniasta n:o 10 on esitetty kuvassa 13, joten voit seurata nuotteja.



Kuva 13

Tässä tulee ohjelmakoodi ensimmäiseen tahtiin:

```
795 REM TAHTI 1
800 DATA 20,00,0,00,0,11,5
802 DATA 20,11,3,00,0,00,0
804 DATA 10,00,0,00,0,03,4
806 DATA 10,10,3,00,0,00,0
808 DATA 10,11,3,00,0,00,0
810 DATA 10,01,2,00,0,00,0
812 DATA 10,03,2,00,0,11,5
814 DATA 10,04,2,00,0,00,0
816 DATA 10,06,2,00,0,00,0
818 DATA 10,03,2,00,0,00,0
9997 DATA 0
```

Suorita tämä katkelma ennen kuin jatkat ohjelman syöttämistä. Ehdotan, että kuuntelet ohjelmaa jokaisen tahdin jälkeen.

```
819 REM TAHTI 2
820 DATA 20,08,2,00,0,04,4
822 DATA 20,06,2,00,0,00,0
824 DATA 20,04,2,00,0,00,0
826 DATA 20,03,2,00,0,00,0
828 DATA 20,01,1,00,0,06,4
830 DATA 20,11,3,00,0,00,0
832 DATA 20,10,3,00,0,00,0
834 DATA 20,08,3,00,0,00,0
836 DATA 20,06,3,00,0,06,5
838 DATA 20,04,2,00,0,00,0
840 DATA 20,03,2,00,0,00,0
842 DATA 20,01,2,00,0,00,0
```

Ääni 2 tulee mukaan seuraavassa tahdissa:

```
844 REM TAHTI 3
846 DATA 40,03,2,00,0,11,5
848 DATA 40,00,0,06,3,00,0
850 DATA 20,00,0,00,0,11,4
852 DATA 20,01,2,05,3,00,0
854 DATA 20,03,2,06,3,00,0
856 DATA 20,05,2,08,3,00,0
858 DATA 20,06,2,10,3,10,4
860 DATA 20,00,0,11,3,00,0
862 DATA 20,00,0,01,2,00,0
864 DATA 20,00,0,10,3,00,0
866 REM TAHTI 4
868 DATA 10,00,0,03,2,11,4
870 DATA 10,00,0,01,2,00,0
872 DATA 10,00,0,11,3,00,0
874 DATA 10,00,0,10,3,00,0
876 DATA 10,00,0,08,3,01,3
878 DATA 10,00,0,06,3,00,0
880 DATA 10,00,0,05,3,00,0
882 DATA 10,00,0,03,3,00,0
884 DATA 10,05,2,01,3,01,4
886 DATA 10,00,0,11,3,00,0
888 DATA 10,00,0,10,3,00,0
890 DATA 10,00,0,08,3,00,0
```

Huomaa tauko (-1,0) äänelle 3 seuraavan tahdin viidennessä vaiheessa.

```
892 REM TAHTI 5
894 DATA 10,06,2,10,3,06,4
896 DATA 10,00,0,01,3,00,0
898 DATA 10,00,0,03,3,00,0
900 DATA 10,00,0,05,3,00,0
902 DATA 10,00,0,06,3,-1,0
904 DATA 10,05,2,09,3,00,0
906 DATA 10,06,2,10,3,00,0
908 DATA 10,08,2,11,3,00,0
910 DATA 10,10,2,01,2,00,0
912 DATA 10,11,2,03,2,00,0
914 DATA 10,01,1,04,2,00,0
916 DATA 10,10,2,01,2,00,0
918 REM TAHTI 6
920 DATA 10,03,1,06,2,00,0
922 DATA 10,01,1,00,0,00,0
924 DATA 10,11,2,03,2,00,0
926 DATA 10,10,2,00,0,00,0
928 DATA 10,08,2,11,3,00,0
930 DATA 10,06,2,00,0,00,0
932 DATA 10,04,2,08,3,00,0
934 DATA 10,03,2,00,0,00,0
936 DATA 10,04,2,01,2,00,0
938 DATA 10,11,2,00,0,00,0
940 DATA 10,10,2,00,0,00,0
942 DATA 10,08,2,00,0,00,0
```

Ääni 3 palaa seuraavassa tahdissa ja toistaa teeman.

```
944 REM TAHTI 7
946 DATA 10,06,2,00,0,00,0
948 DATA 10,00,0,04,2,00,0
950 DATA 10,00,0,03,2,11,4
952 DATA 10,00,0,01,2,00,0
954 DATA 10,00,0,03,2,00,0
956 DATA 10,06,2,00,0,10,4
958 DATA 10,08,2,00,0,11,4
960 DATA 10,09,2,00,0,01,3
962 DATA 10,11,2,00,0,03,3
964 DATA 10,00,0,01,2,04,3
966 DATA 10,00,0,03,2,06,3
968 DATA 10,00,0,11,3,03,3
```

Yritä muutella äänten luonnetta ja sovittaa tempoa, kunnes se on miellyttävimmillään. Elävässä musiikissa kestot eivät voi olla tarkkoja (eikä niiden pidäkään olla). Saat inhimillisemmän soinnin, jos vaihtelet kestoaikoja hitaasti kiihdyttäen ja hidastaen. Tämä on eräs syy siihen, miksi valitsimme juuri tämän koodaustavan. Se ei kuitenkaan rajoita 1/4, 1/8 jne. kestoihin. (Jotkut säveltäjät antavat erityisiä ohjeita tempon muutoksista: rit. = hidastuu, acc. = kiihtyy, a tempo = takaisin alkuperäiseen tempoon jne.) Voit myös yrittää kehittää parempia musiikin koodaustapoja ja tiedon tallennustapoja. Voit esimerkiksi lukea (READ) dataa ja sitten tallentaa sen PRINT #-käskyillä nauhalle tai levylle ja jälleen myöhemmin lukea sitä INPUT #-

käskyillä. Ja ehdotus pitemmälle ehtineille ohjelmoijille ja muusikoille: kokeilkaa myös muilla kuin 12 sävelen asteikoilla.

Ääniä hyperavaruudesta

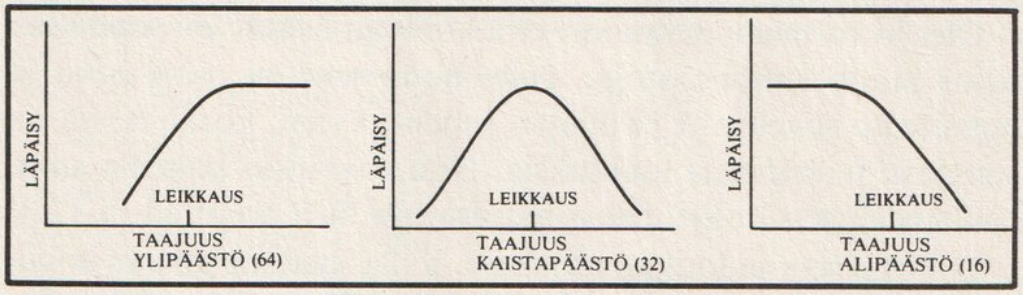
Et voi menetellä väärin ääniefektien kanssa, koska päätät itse säännöistä. SIDin käyttökelpoisimpia välineitä ääniefektejä varten on sen kohina-aaltomuoto. Sitä ei olekaan vielä käytetty. Vaikka on totta, että kohinalla ei ole sävelkorkeutta, sillä on ''väritys'', jota voidaan säädellä. SID tuottaa kohinaa tulostamalla satunnaislukuja ''aaltomuotonaan'', mutta nopeus jolla se tämän tekee, riippuu siitä, mitä on tallennettu taajuusrekisteriin. Alhaisilla nopeuksilla ääni kuulostaa jyrisevältä, taas suurilla nopeuksilla sihisevältä. Äänen laatua voidaan säädellä myös suodattimen avulla, ja tietenkin kohinaakin voidaan kontrolloida ADSR-järjestelmän avulla. Laitetaanpa joitakin näistä töihin. Lataa ensin vanha avausohjelma (rivit 10—70), jolla määritellään joitakin SIDin osoitteita.

```
100 POKE VOL,15: REM VOIMAKKUUS MAKSIMIIN
110 WV=128: REM KOHINA-AALTOMUOTO
120 AT=1: REM NOUSUNOPEUS
130 DE=8: REM TASAANTUMISNOPEUS
140 SU=8: REM JÄLKISOINTITASO
150 RE=10: REM LASKUNOPEUS
160 FRQ=1000: REM TAAJUUS (HZ)
170 TM=6: REM AIKA SEKUNNEISSA
180:
190 FR=FR*C: FH=INT(FR/256): FL=FR-256*FH: REM TAAJUUDEN ALEMPI JA YLEMPI
TAVU
200 TI$="000000": TM=TM*60: REM AIKA KOTVINA (1/60 SEK.)
210 :
220 POKE A1,16*AT+DE: REM NOUSU/TASAANTUMISNOPEUDET
230 POKE S1,16*SU+RE: REM JÄLKISOINTITASO JA LASKUAIKANOPEUS
240 POKE F1,FL: POKE F1+1,FH: REM ASETETAAN TAAJUUS
250 :
260 T=TI+TM: POKE W1,WV+1: REM AVATAAN ÄÄNI
270 IF TI<T THEN 270: REM JATKETAAN ÄÄNTÄ
280 POKE W1,WV: REM LASKUAIKA ALKAA
```

Suorita tämä useaan kertaan, ensin kokeillen erilaisilla FRQ:n arvoilla (ei yli 3900 Hz), sitten muuttujien AT, DE, SU ja RE eri luvuilla. (Jos TM on liian pieni, lasku (release) tapahtuu ennen kuin jälkisointitaso (sustain) on saavutettu, mikä voi antaa vaikutelman, että säätelet voimakkuutta.) Kokeile sekä hyvin

matalilla (n. 20 Hz) että korkeilla (n. 3000 Hz) taajuuksilla. Nopealla nousuajalla (AT = 0) ilman jälkisointia (SU = 0) saat aikaan pyssyn laukauksia, oven paisketta, askelia ja muita iskutyyppisiä ääniä. Keskinkertaisella nousuajalla, tasaantumisajalla ja jälkisointitasolla sekä pitkällä laskuajalla saat tuulen, aaltojen, kaukaisten laukausten, ukkosen ja muiden sen tyyppisten ilmiöiden ääniä.

SIDissä on myös suodattimia, joiden avulla voit ''värittää'' tuotettuja ääniä. Suodattimien käyttö voi olla vaikeata, erityisesti BASICillä ohjelmoitaessa. Voit kuitenkin tutkia seuraavia ääniefektiesimerkkejä ja muutella lukuja, joita asetetaan POKE-lauseilla suodattimen rekistereihin. Kuten nimikin kertoo, yläpäästösuodatin päästää läpi vain tietyn resonanssitaajuuden ylittävät taajuudet. Alapäästösuodatin päästää vastaavasti läpi vain tietyn resonanssitaajuuden alittavat taajuudet ja kaistapäästösuodatin vain tietyllä välillä olevat taajuudet.



Kuva 14. Suodatustavat

Jotta voit kokeilla suodatinta, lisää seuraavat rivit edelliseen ääniefektiohjelmaan:

```
172 MODE=32: REM SUODATUSTAPA
174 RS=15: REM TAYSI VOIMAKKUUS
176 RFRQ= 10000: REM SUODATUSTAAJUUS
178 RF=(RFRQ-30)/5.8: RH=INT(RF/8): RL=RF-8*RH: REM SUODATUSTAAJUUDEN MUUNN
242 POKE SID+23,16*RS+1: REM SUODATTIMEN KYTKEMINEN PAALLE
244 POKE SID+21,RH: POKE SID+22,RL: REM SUODATUSTAAJUUDEN ASETUS SIDIIN
246 POKE VOL,MO+15: REM SUODATUSTAVAN JA VOIMAKKUUDEN ASETUS SIDIIN
```

Aja ohjelma ja kokeile muuttujaan MODE (rivillä 172) arvoja 64 (ylipäästä), 32 (kaistapäästö) ja 16 (alipäästö). Kokeile myös muuttujien FRQ (rivi 160) ja RFRQ (rivi 176) muuttami-

sen vaikutukset. Huomaat, että yleensä äänen voimakkuus laskee, koska suodatin monessa tapauksessa poistaa paljon ''energiaa'' äänestä. Kokeile myös erilaisilla nousuaika/tasaantumisaika- ja jälkisointitaso/laskuaika-arvoilla.

Suodattimen täysi vaikutus voidaan paremmin kuulla muuttamalla sen resonanssitaajuutta, kun ääni on jälkisointivaiheessa. Voit tehdä sen muuttamalla tai lisäämällä seuraavat rivit:

```
270 ML=1.1: REM TEKIJÄ JOLLA MUUTETAAN UODATUSTAAJUUTTA
280 E=8: W=1: HI=2000: LO=500: REM VAKIOITA
290 IF TI>T THEN 360: REM LOPETUS KUN AIKA ON TÄYSI
300 RF=RF*ML
310 IF RF>HI AND ML>W THEN ML=W/ML: RF=RF*ML: REM MUUTETAAN RF
320 IF RF<LO AND ML<W THEN ML=W/ML: RF=RF*ML
330 RH=INT(RF/E): RL=RF-E*RH: REM LASKETAAN ALEMPI JA YLEMPI TAVU TAAJUUDESTA
340 POKE SID+21,RL: POKE SID+22,RH: REM ASETETAAN TAAJUUS SIDIIN
350 GOTO 290
360 POKE W1,WV: REM LASKUAIKA ALKAA
```

Nyt voit leikkiä suodatindynamiikalla. Muuta muuttujaa ML (rivi 270), joka säätelee suodattimen resonanssitaajuutta. Tallenna tämä ohjelma nimellä SIDNOISE.

Olet jo varmaan arvannut, että kohinan lisäksi voi suodattaa myös muita aaltomuotoja. Tulet huomaamaan, että saha- ja pulssiaalto soveltuvat parhaiten suodatukseen, koska niissä on runsaasti harmonisia taajuuksia. Niinpä voitkin kokeilla antamalla edellisessä ohjelmassa muuttujalle WV arvot 64 ja 32.

Lopuksi seuraa joukko ohjelmia, joilla saadaan aikaan erilaisia ääniefektejä. Oletan taas, että sinulla on avausohjelmarivit (10—70) jo ladattuina. Otsikko ja REM-lauseet kertovat kunkin ohjelman tarkoituksen, ja voit kokeilla niillä vapaasti.

''Ammunta''

```
100 POKE F1,200: POKE F1+1,40: REM TAAJUUS ÄÄNELLE 1
110 POKE A1,0+15: REM NOPEA NOUSUAIKA - HIDAS TASAANTUMISAIKA
120 POKE S1,0: REM EI JÄLKISOINTIA - NOPEA LASKUAIKA
130 FOR I=15 TO 0 STEP -1: REM 15 ASKELTA
140 POKE W1,128+1: REM KOHINA-AALTO ÄÄNELLE 1 - ALOITETAAN NOUSU
150 POKE VOL,I: REM DYNAAMISTA VOIMAKKUUTTA VOI KÄYTTÄÄ KOHINA-AALLOLLA
160 NEXT I
170 POKE W1,0: REM LASKUAIKA ALKAA
```

”Nuken itku”

```
100 POKE VOL,15: REM VOIMAKKUUS MAKSIMIIN
110 POKE A1,15: REM NOPEA NOUSU - HIDAS TASAANTUMISAIKA
120 POKE S1,0: REM EI JÄLKISOINTIA - NOPEA LASKUAIKA
130 POKE F1+1,40: REM YLÄTAAJUUS EI MUUTU
140 POKE W1,32+1: REM SAHA-AALTO - ALOITETAAN NOUSUAIKA
150 FOR I=200 TO 5 STEP -2: REM TAAJUUDEN ALENNUSSILMUKKA
160 POKE F1,I: REM MUUTETAAN ALATAAJUUTTA
170 NEXT I
180 FOR I=150 TO 5 STEP -2: REM TOINEN TAAJUUDEN ALENNUSSILMUKKA
190 POKE F1,I
200 NEXT I
210 POKE W1,0: REM LASKUAIKA
```

”Vibrato”

```
100 POKE SID+3,4: REM 25% PULSSIN LEVEYS
110 POKE A1,2*16+9: POKE S1,5*16+9: REM ADSR ÄÄNELLE 1
120 POKE F3,120: REM TAAJUUS ÄÄNELLE 3 (7.5 HZ)
130 POKE W3,16: REM KOLMIOAALTO ÄÄNELLE 3
140 POKE VOL,128+15: REM ÄÄNI 3 POIS PÄÄLTÄ - MAKSIMI YLEISVOIMAKKUUS
150 READ FR,DR: REM LUETAAN TAAJUUS JA KESTO
160 IF FR=0 THEN END
170 POKE W1,64+1: REM ÄÄNI 1 NOUSUAIKA ALKAA
180 FOR T=1 TO DR*3: REM SYKLIEN MÄÄRÄ
190 FQ=FR+PEEK(SID+27)/2:REM LUETAAN ÄÄNEN 3 ULOSTULOA JA SOVITETAAN TAAJUUTTA
200 FH=INT(FQ/256): FL=FQ AND 255: REM YLÄ- JA ALATAAJUUDET
210 POKE F1,FL: POKE F1+1,FH
220 NEXT T
230 POKE W1,64: REM ÄÄNEN 1 LASKUAIKA
240 GOTO 150: REM LISÄÄ DATAA
250 :
260 DATA 4817,2,5103,2,5407,2
270 DATA 8583,4,5407,2,8583,4
280 DATA 5407,4,8583,12,9634,2
290 DATA 10207,2,10814,2,8583,2
300 DATA 9634,4,10814,2,8583,2
310 DATA 9634,4,8583,12
320 DATA 0,0
```

”Sireeni”

```
100 R3=SID+27: REM ÄÄNEN 3 LUKUOSOITE
110 BYTE=256: K=3.5: REM VAKIOITA
111 POKE F3,5: REM ÄÄNEN 3 TAAJUUS (1/3 HZ)
112 POKE W3,16: REM KOLMIOAALTO ÄÄNI 3:LLE
120 POKE F1+3,2: REM ÄÄNELLE 1 12% SYKLI
130 POKE VOL,128+15: REM ÄÄNI 3 POIS - VOIMAKKUUS MAKSIMIIN
140 POKE S1,15*16+0: REM ÄÄNI 1 MAX JÄLISOINTI - NOPEA LASKU
150 POKE A1,0: REM ÄÄNI 1 NOPEA NOUSU - NOPEA TASAANTUMINEN
160 POKE W1,64+1: REM ÄÄNI 1 NOUSUAIKA ALKAA PULSSIAALLOLLA
170 FR=7000: REM ÄÄNEN 1 TAAJUUS = 440 HZ
180 FOR T=1 TO 400: REM TAAJUUDEN MUUTOSSILMUKKA
190 FQ=FR+K*PEEK(R3): REM SOVITETAAN TAAJUUS
200 FH=INT(FQ/BY): FL=FQ-BY*FH
210 POKE F1,FL: POKE F1+1,FH: REM TAAJUUDEN ASETUS SIDIIN
220 NEXT T
230 POKE W1,0: REM LASKUAIKA ALKAA (ÄÄNI 1)
```

”Vasarointi”

```
100 RF=SID+23: REM SUODATTIMEN RESONANSSIREKISTERI
110 FC=SID+21: REM SUODATTIMEN TAAJUUSREKISTERI
120 POKE F1+1,30: REM TAAJUUS = 480 HZ
130 POKE A1,0+6: REM NOPEA NOUSU - KESKINKERTAINEN TASAANTUMISAIKA
140 POKE S1,0: REM EI JÄLKISOINTIA - NOPEA LASKUAIKA
150 POKE FC+1,150: REM SUODATUSTAAJUUS
160 POKE RF,0+1: REM EI RESONANSSIA - ÄÄNI 1 KYTKETÄÄN SUODATTIMEEN
170 POKE VOL,64+15: REM YLIPÄÄSTÖMOODI - MAKSIMI VOIMAKKUUS
180 FOR I=1 TO 15: REM ISKUJEN LUKUMÄÄRÄ
190 POKE W1,128+1: REM ÄÄNEN 1 NOUSU ALKAA KOHINA-AALLOLLA
200 FOR J=1 TO 200: NEXT J: REM VIIVE
210 POKE W1,128: REM LASKUAIKA ALKAA
220 FOR J=1 TO 100: NEXT J: REM VIIVE
230 NEXT I
240 END
```

”Hyttynen”

```
100 POKE F1+1,100: REM ÄÄNEN 1 TAAJUUS 1600 HZ
110 POKE A1,13*16+13: REM ÄÄNI 1 HIDAS NOUSUAIKA JA TASAANTUMISAIKA
120 POKE S1,0: REM ÄÄNI 1 EI JÄLKISOINTIA - NOPEA LASKUAIKA
130 POKE F3+1,28: REM ÄÄNEN 3 TAAJUUS 450 HZ
140 POKE VOL,15: REM VOIMAKKUUS MAKSIMIIN
150 POKE W1,16+2+1: REM ÄÄNI 1 KOLMIOAAL. SYNKRONIASSA 3:N KANSSA - NOUSU ALKAA
160 FOR I=1 TO 5000: NEXT I: REM VIIVE
170 POKE W1,16+2: REM LASKU ALKAA
180 FOR I=1 TO 1000: NEXT I: REM VIIVE
190 POKE W1,0: REM LASKU ALKAA
200 END
```

”Kellon lyönti”

```
100 POKE F1+1,130: REM ÄÄNEN 1 TAAJUUS 2000 HZ
110 POKE A1,0+9: REM ÄÄNI 1 - NOPEA NOUSU JA KESKINKERTAINEN TASAANTUMINEN
120 POKE F3+1,30: REM ÄÄNEN 3 TAAJUUS 480 HZ
130 POKE VOL,15: REM VOIMAKKUUS MAKSIMIIN
140 FOR I=1 TO 12: REM 12 LYÖNTIÄ
150 POKE W1,16+4+1: REM ÄÄNI 1 KOLMIOAAL. RENGASMODULAATIO ÄÄNEN 3 KANSSA
160 FOR J=1 TO 500: NEXT J: REM VIIVE
170 POKE W1,16+4: REM LASKUAIKA
180 FOR J=1 TO 100: NEXT J: REM VIIVE
190 NEXT I: REM SEURAAVA LYÖNTI
200 POKE W1,0: REM LASKUAIKA
210 END
```

Liite 4

Virheilmoitukset

****

BAD SUBSCRIPT ERROR

Tämä virheilmoitus on suomeksi huono indeksi. Ohjelmassa tuli esiin indeksin numero, joka on liian suuri DIM-käskyllä määriteltyyn taulukkokokoon nähden. Jos DIM-käsky on esimerkiksi DIM A$(100), ei voi olla muuttujaa nimeltä A$(101). Tämä virheilmoitus näkyy myös silloin, jos DIM-käskyä ei ole lainkaan annettu ja käytetään muuttujia, joiden indeksi on suurempi kuin 10. Tämän virheilmoituksen nähdessäsi tarkista indeksoidut muuttujat.

****

****

CAN'T CONTINUE ERROR

Käskyä CONT voidaan käyttää ohjelman suorituksen jatkamiseen sen jälkeen kun on painettu näppäintä RUN/STOP tai on käytetty ohjelmassa käskyä STOP. Jos kuitenkin muutat jonkin rivin ennen kuin annat käskyn CONT, näet tämän virheilmoituksen. Se näkyy myös silloin, kun ohjelmaa ei ole lainkaan suoritettu. Kummassakin tapauksessa anna uudelleen käsky RUN.

****

****

EXTRA IGNORED

Tätä virhettä et oikeastaan voi korjata, sillä se aiheutuu siitä, että INPUT-käskyn vastauksessa on pilkku väärässä paikassa. Commodore-tietokone käyttää pilkkua erityismerkkinä. Ne sallitaan vastaukseksi INPUT-käskyyn vain silloin, kun ohjelma pyytää useaa vastausta yhtä aikaa, esimerkiksi:
10 INPUT A$,B$,C$.

Merkkimuuttujissa olevat pilkut aiheuttavat ongelmia myös silloin, kun niitä tallennetaan ja luetaan levyketiedostoista.

****

****

FILE NOT FOUND

Tämä virheilmoitus tulee esiin silloin, kun yrität ladata ohjelman tai tiedoston sellaiselta levykkeeltä, jossa se ei ole. Jos olet varma, että sinulla on sellainen ohjelma, tarkista levykkeen hakemisto nähdäksesi, onko se todellakin juuri tällä levykkeellä tai oletko kirjoittanut sen oikein. Levykeasema vaatii ehdotonta tarkkuutta ohjelman nimeämisessä. Käytä tähteä tyyppihaussa välttyäksesi tältä virheeltä.

****

****

FILE NOT OPEN ERROR
FILE OPEN ERROR

Kun käytämme tiedonkäsittelysanoja PRINT #, INPUT # ja GET #, on ensiksi käskyllä OPEN avattava tiedosto. Muuten näet virheilmoituksen FILE NOT OPEN ERROR. Se esiintyy myös silloin, kun käskyä CMD, jonka tarkoitus on siirtää kuvaruutuun kirjoitettavaksi tarkoitettua tietoa kirjoittimelle tai levykkeelle, käytetään avaamatta oikeaa tiedostoa tai käskyn CLOSE kanssa. FILE OPEN ERROR esiintyy silloin, kun yrität avata tiedoston, joka on aikaisemmin avattu käskyllä OPEN mutta jota ei ole suljettu käskyllä CLOSE.

Tarkista aina tiedoston käsittelykäskyt varmistaaksesi, että tiedosto suljetaan käytön jälkeen käskyllä CLOSE.

****

****

ILLEGAL DIRECT

Useita BASICin käskyjä voidaan käyttää suoraan näppäimistöltä, mutta INPUT-käskyä ei voida käyttää siten. Se tuntuu loogiselta, kun muistaa, kuinka sitä käytetään.

****

****

ILLEGAL QUANTITY ERROR

Kun tämä virhe esiintyy ohjelmarivillä, jolla on POKE-käsky, tarkista, ettei muistipaikkaan ole yritetty sijoittaa numeroa joka olisi suurempi kuin 255 tai pienempi kuin 0. Jos olet käyttänyt muuttujia, tarkista, ettei niiden edustamilla luvuilla suoritettu laskutoimitus ole antanut tulokseksi 255:tä suurempaa tai 0:aa pienempää lukua.

****

****

ILLEGAL QUANTITY ERROR

Kokonaismuuttujat tunnistetaan prosenttimerkin (%) perusteella. Jos tämä virheilmoitus näkyy rivillä, jossa on kokonaislukumuuttuja, tarkista, etteivät muuttujat merkitse lukuja jotka ovat suurempia kuin 32767 tai pienenmpiä kuin —32768. Jos virhe ei ole tapahtunut sellaisella rivillä, jossa on kokonaislukumuuttuja, tarkista, onko POKE-käskyt annettu oikein.

****

****

NEXT WITHOUT FOR ERROR

Tämä virheilmoitus johtuu siitä, että olet käyttänyt sanaa NEXT aukaisematta ensin silmukkaa käskyllä FOR. Tämä

tapahtuu silloin kun sanaa NEXT on käytetty väärän muuttujan kanssa.

```
10 FOR I=1 TO 10
20 NEXT K
```

Jos tapahtuu päinvastoin eli käskyä FOR käytetään ilman sanaa NEXT, ei mitään virheilmoitusta tule. Silmukka ei siinä tapauksessa vain toimi niin kuin sen pitäisi. Tämä virheilmoitus tulostuu myös jos GOTO- tai GOSUB-käskyllä hypätään silmukan ulkopuolelta sen sisään.

****

****

NOT INPUT FILE ERROR
NOT OUTPUT FILE ERROR

Kasettitiedostot avataan käskyllä OPEN joko INPUT- tai OUTPUT -tiedostoina, siis joko tietojen hakemiseksi tiedostosta tai niiden kirjoittamiseksi tiedostoon. Osoite 1 tarkoittaa, että nauhatiedosto on auki siihen kirjoittamista varten. Jos käytät käskyjä INPUT# tai GET#, näet ensimmäisen näistä virheilmoituksista. Jos kasettinauhurin yhteydessä ei ole käytetty mitään toista osoitetta, tietokone olettaa, että tiedostosta haetaan tietoa. Käskyn PRINT# käyttäminen antaa tulokseksi toisen näistä virheilmoituksista.

****

****

OUT OF DATA

Laske, kuinka monta tietoa DATA-lauseissa on, ja tarkista sitten, kuinka monta kertaa ohjelma käy läpi käskyn READ. (Eräs keino on etsiä FOR/NEXT-silmukoita, joissa on käsky READ.) Tarkista DATA-lauseet joko ottamalla listaus kirjoittamastasi ohjelmasta (käskyllä LIST) tai vertaamalla tietoja, joiden pohjalta olet DATA-lauseet kirjoittanut.

****

****

OVERFLOW ERROR

Tämä virheilmoitus esiintyy silloin kun yrität käsitellä suurempaa lukua kuin 1.70141884E+35. Ratkaisu (jos se on mahdollinen) on hajottaa laskutoimitus osiin siten, että tätä lukua ei koskaan saavuteta.

****

****

REDIM'D ARRAY ERROR

Käskyä DIM voidaan käyttää vain kerran tietyn taulukon määrittelyyn. Kun taulukolle on varattu tila, sitä ei voida enää muuttaa uudella DIM-käskyllä. Tämä virheilmoitus esiintyy pääasiassa silloin, kun editoit ohjelmia tai yrität liittää yhteen ohjelmia, joissa kummassakin on DIM-käskyjä.

****

****

REDO FROM START

Tämä virheilmoitus näkyy silloin kun vastaat INPUT-kysymykseen merkkijonolla, vaikka se odottaa numeroa. Korjaa tilanne kirjoittamalla numero.

****

****

RETURN WITHOUT GOSUB

GOSUB ja RETURN toimivat yhdessä. Tämä virheilmoitus ilmestyy, jos ohjelma kohtaa käskyn RETURN eikä sille ole annettu käskyä GOSUB. Se tavataan useimmiten silloin, kun kyseistä ohjelmariviä kutsutaan käskyllä GOTO.

****

****

STRING TOO LONG ERROR

Merkkimuuttujat saavat olla vain 255 merkkiä pitkiä. Merkkimuuttujavirheitä esiintyy yleensä silloin, kun liitetään jokin merkkimuuttuja toiseen. Tarkista jokaisen merkkimuuttujan pituus yrittäessäsi selvittää syytä tähän virheilmoitukseen.

****

****

SYNTAX ERROR

Tällaisten ''kielioppivirheiden'' syynä voi olla vaikka mikä, mutta ne kaikki liittyvät BASICin sääntöihin. Yleisimpiä virheitä ovat kirjoitusvirheet ja välimerkkien, erityisesti kaksoispisteiden ja pilkkujen, puuttuminen. Tarkkaile myös, oletko vahingossa kirjoittanut puolipisteen kaksoispisteen asemesta ja oletko muistanut sulkea sulkumerkit. Oletko käyttänyt varattuja sanoja?

****

****

TYPE MISMATCH ERROR

Tällainen virheilmoitus näkyy aina kun yrittää kirjoittaa merkin numeeriseen muuttujaan tai varsinaisen luvun merkkimuuttujaan. Merkkimuuttujia ja numeroita ei voida sekoittaa keskenään. Vaikeuksia aiheuttavat käskyt ovat esimerkiksi tällaisia:

10 A$=9 (tai) A$=A

tai

10 A=''ESIMERKKI'' (tai) A=A$

****

****

UNDEFINED STATEMENT ERROR

Sinun on käytettävä käskyjen GOTO ja GOSUB yhteydessä sellaisia rivinumeroita, jotka ovat ohjelmassa todella käytössä. Tämä virheilmoitus esiintyy usein silloin, kun editoit tai muutat rivinumerointia, mutta unohdat muuttaa rivinumeroihin viittaavat GOTO- ja GOSUB-käskyt. Tarkista ne rivinumerot, joilla virhe esiintyy, ja yritä sitten listata ne rivit. Käskyä RUN ei myöskään voida käyttää rivinumeroiden kanssa, joten tällainen virheilmoitus saattaa johtua siitäkin.

****

Liite 5

ASCII/CHRS -koodit ja muunnostaulukko

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0	CRSR ↓	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
// 2 WHT	5		22	•	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
Päälle SHIFT C=	8		25	*	42	;	59
Pois päältä SHIFT C=	9		26	+	43	<	60
	10		27	,	44	=	61
	11	# 3 RED	28	–	45	>	62
	12	CRSR →	29	.	46	?	63
RETURN	13	& 6 GRN	30	/	47	@	64
PIENET KIRJAIMET	14	' 7 BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
D	68		99		130		161
E	69		100		131		162
F	70		101		132		163
G	71		102	f1	133		164
H	72		103	f3	134		165
I	73		104	f5	135		166
J	74		105	f7	136		167
K	75		106	f2	137		168
L	76		107	f4	138		169
M	77		108	f6	139		170
N	78		109	f8	140		171
O	79		110	SHIFT RETURN	141		172
P	80		111	ISOT KIR-JAIMET	142		173
Q	81		112		143		174
R	82		113	BLK	144		175
S	83		114	CRSR	145		176
T	84		115	RVS OFF	146		177
U	85		116	CLR HOME	147		178
V	86		117	INST DEL	148		179
W	87		118	Ruskea	149		180
X	88		119	Vaal. punainen	150		181
Y	89		120	Harmaa 1	151		182
Z	90		121	Harmaa 2	152		183
[	91		122	Vaal. vihreä	153		184
£	92		123	Vaal. sininen	154		185
]	93		124	Harmaa 3	155		186
↑	94		125	PUR	156		187
←	95		126	CRSR	157		188
	96		127	YEL	158		189
	97		128	CYN	159		190
	98	Orange	129	SPACE	160		191

192 to 223 = 96 to 127
224 to 254 = 160 to 190
255 = 126

Lukujärjestelmien muunnostaulukko

BIN	*DEC*	*HEX*	*BIN*	*DEC*	*HEX*
0	0	00	110101	53	35
1	1	01	110110	54	36
10	2	02	110111	55	37
11	3	03	111000	56	38
100	4	04	111001	57	39
101	5	05	111010	58	3A
110	6	06	111011	59	3B
111	7	07	111100	60	3C
1000	8	08	111101	61	3D
1001	9	09	111110	62	3E
1010	10	0A	111111	63	3F
1011	11	0B	1000000	64	40
1100	12	0C	1000001	65	41
1101	13	0D	1000010	66	42
1110	14	0E	1000011	67	43
1111	15	0F	1000100	68	44
10000	16	10	1000101	69	45
10001	17	11	1000110	70	46
10010	18	12	1000111	71	47
10011	19	13	1001000	72	48
10100	20	14	1001001	73	49
10101	21	15	1001010	74	4A
10110	22	16	1001011	75	4B
10111	23	17	1001100	76	4C
11000	24	18	1001101	77	4D
11001	25	19	1001110	78	4E
11010	26	1A	1001111	79	4F
11011	27	1B	1010000	80	50
11100	28	1C	1010001	81	51
11101	29	1D	1010010	82	52
11110	30	1E	1010011	83	53
11111	31	1F	1010100	84	54
100000	32	20	1010101	85	55
100001	33	21	1010110	86	56
100010	34	22	1010111	87	57
100011	35	23	1011000	88	58
100100	36	24	1011001	89	59
100101	37	25	1011010	90	5A
100110	38	26	1011011	91	5B
100111	39	27	1011100	92	5C
101000	40	28	1011101	93	5D
101001	41	29	1011110	94	5E
101010	42	2A	1011111	95	5F
101011	43	2B	1100000	96	60
101100	44	2C	1100001	97	61
101101	45	2D	1100010	98	62
101110	46	2E	1100011	99	63
101111	47	2F	1100100	100	64
110000	48	30	1100101	101	65
110001	49	31	1100110	102	66
110010	50	32	1100111	103	67
110011	51	33	1101000	104	68
110100	52	34	1101001	105	69

Lukujärjestelmien muunnostaulukko

BIN	*DEC*	*HEX*	*BIN*	*DEC*	*HEX*
1101010	106	6A	10011111	159	9F
1101011	107	6B	10100000	160	A0
1101100	108	6C	10100001	161	A1
1101101	109	6D	10100010	162	A2
1101110	110	6E	10100011	163	A3
1101111	111	6F	10100100	164	A4
1110000	112	70	10100101	165	A5
1110001	113	71	10100110	166	A6
1110010	114	72	10100111	167	A7
1110011	115	73	10101000	168	A8
1110100	116	74	10101001	169	A9
1110101	117	75	10101010	170	AA
1110110	118	76	10101011	171	AB
1110111	119	77	10101100	172	AC
1111000	120	78	10101101	173	AD
1111001	121	79	10101110	174	AE
1111010	122	7A	10101111	175	AF
1111011	123	7B	10110000	176	B0
1111100	124	7C	10110001	177	B1
1111101	125	7D	10110010	178	B2
1111110	126	7E	10110011	179	B3
1111111	127	7F	10110100	180	B4
10000000	128	80	10110101	181	B5
10000001	129	81	10110110	182	B6
10000010	130	82	10110111	183	B7
10000011	131	83	10111000	184	B8
10000100	132	84	10111001	185	B9
10000101	133	85	10111010	186	BA
10000110	134	86	10111011	187	BB
10000111	135	87	10111100	188	BC
10001000	136	88	10111101	189	BD
10001001	137	89	10111110	190	BE
10001010	138	8A	10111111	191	BF
10001011	139	8B	11000000	192	C0
10001100	140	8C	11000001	193	C1
10001101	141	8D	11000010	194	C2
10001110	142	8E	11000011	195	C3
10001111	143	8F	11000100	196	C4
10010000	144	90	11000101	197	C5
10010001	145	91	11000110	198	C6
10010010	146	92	11000111	199	C7
10010011	147	93	11001000	200	C8
10010100	148	94	11001001	201	C9
10010101	149	95	11001010	202	CA
10010110	150	96	11001011	203	CB
10010111	151	97	11001100	204	CC
10011000	152	98	11001101	205	CD
10011001	153	99	11001110	206	CE
10011010	154	9A	11001111	207	CF
10011011	155	9B	11010000	208	D0
10011100	156	9C	11010001	209	D1
10011101	157	9D	11010010	210	D2
10011110	158	9E	11010011	211	D3

Lukujärjestelmien muunnostaulukko

BIN	*DEC*	*HEX*	*BIN*	*DEC*	*HEX*
11010100	212	D4	11101010	234	EA
11010101	213	D5	11101011	235	EB
11010110	214	D6	11101100	236	EC
11010111	215	D7	11101101	237	ED
11011000	216	D8	11101110	238	EE
11011001	217	D9	11101111	239	EF
11011010	218	DA	11110000	240	F0
11011011	219	DB	11110001	241	F1
11011100	220	DC	11110010	242	F2
11011101	221	DD	11110011	243	F3
11011110	222	DE	11110100	244	F4
11011111	223	DF	11110101	245	F5
11100000	224	E0	11110110	246	F6
11100001	225	E1	11110111	247	F7
11100010	226	E2	11111000	248	F8
11100011	227	E3	11111001	249	F9
11100100	228	E4	11111010	250	FA
11100101	229	E5	11111011	251	FB
11100110	230	E6	11111100	252	FC
11100111	231	E7	11111101	253	FD
11101000	232	E8	11111110	254	FE
11101001	233	E9	11111111	255	FF

Sanasto

Aliohjelma
Ohjelman osa, jossa suoritetaan usein tarvittavia käskysarjoja ja jota pääohjelma kutsuu.

ASCII
Tietojen välitykseen tarvittavaan merkkijoukkoon liitetty koodijärjestelmä, johon kuuluu 128 koodimerkkiä. Commodore-tietokoneissa käytetään varsinaisen ASCII-koodijärjestelmän muunnosta (255 koodia).

BASIC
Lähinnä ohjelmoinnin opettamiseen tarkoitettu yksinkertainen ohjelmointikieli.

BASIC-kääntäjä
Ohjelma, joka kääntää BASIC-kielisen ohjelman konekielelle.

Bitti
Binäärinumero. Muistin pienin yksikkö, joka voi saada vain arvot 0 tai 1. Kahdeksan bittiä muodostaa tavun.

Data
Tieto

DOS
Levykeaseman käyttöjärjestelmä; ohjelma, joka huolehtii levykeaseman toiminnoista.

Eksponentti
Liittyy potenssin käsitteeseen; luku, joka ilmoittaa kuinka monta kertaa annettu luku (kantaluku) kerrotaan itsellään.

Grafiikkatila
Käyttäjällä on käytettävänään näppäimistöltä isot kirjaimet ja graafiset merkit.

Haarautuminen, hyppääminen
Yleensä jonkin päätöksen seurauksena tapahtuva poikkeaminen ohjelman normaalista suoritusjärjestyksestä.

K
Kilo, tuhat. Yksi kilo muistia 1K = 1024 tavua.

Kasettipuskuri
Muistialue, jota käytetään tietojen väliaikaiseen tallentamiseen siirrettäessä tietoja kasettiasemalta keskusmuistiin.

Kirjoituskonetila
Käyttäjällä on käytettävissään näppäimistöltä isot ja pienet kirjaimet.

Kirjoitussuoja
Levykkeen reunassa olevan loven päälle asetettu tarra, joka estää tietojen kirjoittamisen levykkeelle.

Kohdistin
''Kursori'' ; välkkyvä neliö, joka osoittaa seuraavaksi kirjoitettavan merkin paikan.

Kokonaisluku
Luvut ... —5,—4,—3,—2,—1,0,1,2,3,4,5,...

Kokonaisosa
Kokonaisluku, joka muodostuu desimaaliluvusta, kun desimaalit jätetään pois.

Konekieli
Konekäskyistä koostuva kieli, jolla tehty ohjelma voidaan kääntämättä suorittaa tietokoneella.

Kopiosuojaus
Ohjelman käsittely laittoman kopioinnin estämiseksi.

Kotva
''Jiffy''; sekunnin kuudeskymmenesosa.

Kylmäkäynnistys
''Cold start''; virran kytkeminen pois ja takaisin tietokoneeseen. Tällöin muisti tyhjenee ja muistissa olevat ohjelmat tuhoutuvat.

Käyttöjärjestelmä
Oheisohjelmat, jotka tehostavat ja helpottavat tietokoneen käyttöä.

Lajittelu
Tietojen järjestäminen jonkin säännön mukaan.

Levyke
Taipuisa levy, jonka magneettista pintaa käytetään muistina.

Liitin, liitäntä
''Interface''; sähköinen piiri, joka liittää tietokonejärjestelmän eri osat toisiinsa.

Lisäys
Muuttujan arvon kasvattaminen yhdellä tai jollain muulla vakioarvolla.

Lukumuisti
ROM; pysyvän muistin alue, jota voidaan ainoastaan lukea.

Luku- ja kirjoitusmuisti
RAM; muistin alue, johon voidaan kirjoittaa ja jota voidaan lukea.

Lämmin aloitus
''Warm start''; tietokoneen asettaminen alkutilaansa siten, että muistissa oleva ohjelma ei tuhoudu.

Merkkijono
Kirjainten, grafiikkamerkkien, numeroiden tai editointimerkkien muodostama jono.

Mikrosiru
Yleensä piistä valmistettu integroitu piiri.

Mikrosuoritin
''Mikroprosessori''; numeerista tietoa käsittelevä laite, jota ohjaa muistiin tallennettu ohjelma.

Modeemi
Muunnin; puhelinverkkoa pitkin tapahtuvaan tiedonsiirtoon tarvittava laite, joka suorittaa tiedon muuntamisen.

Muistikartta
Muistin eri osa-alueita ja niiden tehtäviä kuvaava esitys.

Muistin siivous
''Garbage collection''; toimitus, jonka aikana tietokone suorittaa muistin uudelleenjärjestelyä ja poistaa tarpeetonta tietoa.

Nollajono
Tyhjä merkkijono, merkitään ” ”.

Nollasivu
Muistipaikat 0—255 sisältävä muistialue.

Näyttömuisti
Muistin alue, joka sisältää kuvaruudulla olevat merkit.

Ohjelmointikieli
Tietokoneen ohjelmointiin käytettyjen ohjeiden, sääntöjen ja sanojen muodostama kokonaisuus.

Peräkkäistiedosto
Tiedosto, jossa tietueet ovat säilytysjärjestyksessään. Jonkin tietueen esille saamiseksi on ensin luettava kaikki edellä olevat tietueet.

Pikseli
Kuva-alkio, pienin kuvan osa.

Puolitavu, nybble
Neljä bittiä, tavun puolikas.

Rakenteinen, strukturoitu ohjelmointi
Määrättyjä rakennemalleja käyttävä ohjelmointitapa. Ohjelma muodostuu useasta erillisestä aliohjelmasta.

Rinnakkais-
Tiedon siirron standardi, jossa tavun kahdeksan bittiä siirretään samanaikaisesti kahdeksaa kanavaa pitkin.

Sarja-
Tiedon siirron standardi, jossa tavun kahdeksan bittiä siirretään bitti kerrallaan.

Satunnaisluku
Satunnaisesti valittu luku. Saadaan yleensä jostakin siemenluvusta ”satunnaisilla” laskutoimituksilla.

Silmukka
Käskysarja, jonka suorittamista toistetaan kunnes jokin annettu ehto on voimassa.

Sisäkkäiset silmukat
Silmukka on kokonaisuudessaan toisen silmukan sisäpuolella.

Sprite
Liikuteltava kuvalohko (MOB); pieni hienografiikkakuva, joka määritellään näyttömuistista riippumattomassa muistialueessa.

Syntaksi
Ohjelmointikielen muotosäännöt.

Taulukko
Läheisesti yhteenkuuluvien tietojen joukko. Kukin alkio voidaan yksilöidä indeksin perusteella.

Tavu
Muistin perusyksikkö, kahdeksan bitin muodostama bittijono. Tavun sisältämä arvo on kymmenjärjestelmässä väliltä 0—255.

Tekstinkäsittely
Tekstin tallentaminen, muuttaminen ja tulostaminen tietokoneen ja oheislaitteiden avulla.

Tietokanta
Kokoelma tietoja; usein tiedot on järjestetty jollakin tavalla.

Tietokoneohjelma
Jollakin ohjelmointikielellä kirjoitettu sarja toimenpiteitä.

Tietue
Johonkin yksikköön liittyvät tiedot esitettynä samassa yhteydessä.

Tulkittu BASIC
Menetelmä, jossa BASIC-kielinen ohjelma tulkitaan osa kerrallaan ja suoritetaan siinä olevat toimenpiteet.

Tyyppihaku
''Pattern matching''; menetelmä, jota käytetään levykkeellä olevien ohjelmien tai tiedostojen tunnistamiseen.

Valikko
''Menu''; ohjelman tarjoamien vaihtoehtojen luettelo.

Vähennys
Muuttujan arvon pienentäminen yhdellä tai jollain muulla vakioarvolla.

Yhteenliittäminen, ketjuttaminen
Merkkijonojen liittäminen toisiinsa.
A$ = ''AUTO'':B$ = ''KAUPPA'':A$ + B$ = ''AUTOKAUPPA''

Hakemisto

A—V-portti 14
Aaltomuoto 187, 211, 212, 295, 298
  kohina- 320
  kolmio- 301
  kontrollirekisteri 300
  pulssi- 301, 311
  saha- 301
ABS 171
ADSR 186, 303
Aika 62
  reaali- 181
Alaindeksi 119
Aliohjelmat 110, 149, 339
AND-operaattori 251, 260
Animaatio 259
Apple 11
ASCII 176, 339
  -koodit 333
Atari 76, 218
ATN 172
BAM 54
BASIC 23, 31, 46, 53, 57, 339
  -kääntäjä 67
  Microsoft 67
  Simons' 220
  -tulkki 67
  V2 15, 68
  välimerkkien käyttö 78
Binaariluvut 247
  muunnos desimaalijärjestelmään 248, 335
  muunnos heksadesimaalijärjestelmään 335
Bitti 168, 185, 224, 246, 260, 339
  asettaminen 251
Byte
  katso Tavu
CHR$ 148, 177
  -koodit 333
CLOSE-käsky 49, 130, 135, 141, 167
CLR-käsky 129
CLR/HOME-näppäin 27, 61, 129
  PRINT-käskyn kanssa 90
CMD 166
COBOL 67
COMMODORE-näppäin 22, 34, 92, 148
CONT-käsky 40, 168
COS 172
CRSR-näppäimet 27, 61, 70, 89
CTRL-näppäin 24, 69, 103
DATA
  lukeminen 125
  tallennus 117, 140, 152
DATA-lause 125, 136
DEF 172

Digitaalinen noppa 199
DIM 121, 136, 147
DOS 340
DOS-ohjain 44, 58
Editointi
-käskyt 174
-näppäimet 27, 61, 70
Eksponenttiluvut 77
END-käsky 71, 113
FN 172
FOR 105
FOR/NEXT-silmukka 108, 271
FORTRAN 66
GET 102, 130, 135, 174
GOSUB 98, 110, 169
GOTO 40, 98, 169
Grafiikka 183, 245
merkki- 185, 195, 267
sprite- 37, 204, 252
symbolit 22
tarkkuus- 184, 220, 249, 255, 277, 283
tila 22, 143, 164, 185
väri- 164
Haarautuminen 340
Hajatiedosto 150
Hyppääminen 97, 340
Hz 292
Häiriöt 17
IEEE-488-liitäntä 139
IF-lause 98
IF/THEN-lause 170
INPUT 94, 131, 135, 149, 170
INST/DEL-näppäin 27, 61, 70
INT 171, 190
Joystick (sauvaohjain) 218
Kaksoispiste 79
Kasettinauha 32, 130
Kasettinauhatiedostot 135
Kasettinauhuri 17, 36, 45, 130, 133, 134, 166
Kasettiportti 14
Kasettipuskuri 37, 226, 340
Kello 62
reaaliaika- 181
Ketjuttaminen 73, 140, 173, 345
Kielioppivirhe (Syntax error) 79, 134, 144
Kirjoitin 14, 45, 130, 134, 166
-tiedostot 142
Kirjoituskonetila 22, 340
Kirjoitussuojaus 42
Kohdistin 15, 27, 61, 340
PRINT-käskyn kanssa 88
spriten luontiohjelmissa 263
TAB:n kanssa 167
Kokonaislukumuuttujat 75, 341
Konekieli 36, 151, 163, 224, 283, 341
Kotvakello 64
Kuvaruutu 70, 165
-grafiikka 195
väri 185, 192
Kylmäkäynnistys 26, 341
Kysymysmerkki 81
tulostuskäskynä 134
Käyttöjärjestelmä 47, 341
Käänteistila 24
Laajennusportti 14
Lainausmerkkitila 61
Laitenumero 36, 48, 130, 131, 132, 143, 167
Lajittelu 150, 151, 341
Laskin 58
Lataaminen
levykkeeltä 50
nauhalta 33
LEFT$ 179
LEN 173, 179
LET-käsky 78
Levyke 18, 41, 43, 46, 48, 117, 129
hakemisto 41, 46, 141
käyttöjärjestelmä 44, 340
tiedostojen poistaminen 53
valmistelu 48
Levykeasema 41, 43, 166
pyörimisnopeus 56
toinen 56, 133
Levykeaseman numero 46
Levyketiedostot 131,138
Levykevirhe 54

merkkivalo 43, 48, 54
Liitäntäpiiri 224, 342
LISP 67
LIST-käsky 20, 47, 69
Liukulukuakku 166
LOAD-käsky 20, 45, 47
Loogiset operaattorit 251
Lukujärjestelmien muunnostaulukko 335
Lukumäärävirhe 75
Lämmin käynnistys 26, 342
Matemaattiset funktiot 163, 171
Merkkijoukko 267
Merkkilohko 278
Merkkimuuttuja 73, 342
käsittely 173
tyhjä 96, 103, 141, 177, 343
MID$ 179
Mikroprosessori 143, 342
Modeemi 14, 45, 342
Muisti 18, 20, 48, 60, 71, 75, 114, 117, 222, 272, 342
kuvaruutu- 226
katso myös RAM, ROM, VIC
Muistikartta 224, 229, 252
Muistipaikka 163, 169, 187, 277, 278
Musiikkisyntetisaattori 187
Muuttujat 59, 72, 118, 125
indeksoitu 118
ja alaindeksi 119
merkki- 73, 118, 125, 133, 173
nimet 75
numeeriset 72, 118, 169, 174
rajat 78
silmukoissa 108
varatut 62
Nauha 17, 33, 37, 38, 129
Nauhan loppuminen 131, 133
NEW-käsky 66, 70
NEXT-käsky 105
NOT-operaattori 251
Näppäimistö 21
Näyttöpiiri 147, 164, 223, 269
katso myös mikroprosessori, SID, VIC
Ohjelma 68, 144
LISTaus 69
osoiterekisteriohjelma 152
suunnittelu 145
Ohjelmayhteensopivuus 57
Ohjelmointivirhe 66, 150
ON-käsky 169
OPEN-käsky 48, 130, 167
OR-operaattori 251
Osoite 222, 225
toinen 48, 130, 139, 143
PEEK-funktio 57, 164, 221, 271
Peräkkäistiedosto 139, 343
PET Emulator 58
Pikseli 185, 246, 252, 263, 269, 279, 343
Pilkut 135
DATA-lauseessa 126
desimaalien esityksessä 60, 77
INPUT-lauseessa 95, 149
PRINT-käskyn kanssa 85
PLAY-näpäin 17, 33, 37
POKE-käsky 57, 147, 163, 220, 221, 271
Portit 14
kontrolli- 14, 184, 219
Prosessorin s/t- 240
PRG 47, 50, 139
PRINT-käsky 80, 131, 134, 142, 167, 177, 178
ja FRE-funktio 60
ja kuvaruututoiminnot 88
ja käänteiset merkit 91
laskimena 59
näppäimistöltä 88
pilkun kanssa 85
puolipisteen kanssa 82
Puolipiste PRINT-lauseessa 82
Puolitavu 185, 304, 343
Pyöristäminen 75, 171
QWERTY-näppäimistö 28
Radiaanit 172
RAM 15, 32, 222, 268, 277
Raporttiohjelma 152
READ-käsky 126, 136
RECORD-näppäin 17, 38

REL 47
REM 40, 113
RESTORE-käsky 128
RESTORE-näppäin 26, 37, 103, 148
RETURN-käsky 110
RETURN-näppäin 16, 28, 31, 70, 150, 170, 178
RF-modulaattori 14
RIGHT$ 179
Rivinvaihtokäsky 143
RND 187, 199
ROM 223, 227, 269, 342
RUN-käsky 34, 40, 44, 166, 168
RUN/STOP-näppäin 17, 22, 37, 40, 69, 103, 168
RVS 24, 91
Sarjaportti 14
Satunnaislukugeneraattori 184, 187
Satunnaisluvut 187, 320, 344
SAVE-käsky 19, 32, 144
  levykkeelle 52
  nauhalle 38
Sektori 42, 46, 54
SEQ 47, 133, 139, 140
SGN 171
SHIFT LOCK-näppäin 22, 27, 103
SHIFT-näppäin 16, 22, 27, 103, 114
SID 184, 193, 240, 291, 296
  -rekisterit 307
Siemenluku 188
Silmukka 97, 105, 271, 344
SIN 172
Sisäkkäiset silmukat 107, 344
Sivu 42, 46, 54
SPC 167
Sprite 37, 185, 226, 247, 249, 344
  editor-ohjelma 205, 263
  -grafiikka 204
  -lohko 204, 252
  sijaintirekisterit 257, 262
SQR 59, 172
STEP 109
STOP 168
STR$ 173
Strukturoitu ohjelmointi 113, 343
Sulkeet 119, 165
SYS-käsky 165, 283
TAB 167
Tallennussuojaus 42
TAN 172
Tarkistusvirhe (Verifying error) 39
Taulukko 122, 128, 136, 146, 344
Tavu 16, 42, 60, 185, 246, 344
Tekstinkäsittely 29, 344
THEN-käsky 98, 105
Tiedosto 117, 129
  haja- 150
  kasettinauha- 131, 135
  kirjoitin- 142
  levyke- 131, 138
  nauha- 129, 135
  numero 48, 130, 167
  ohjelma- 47
  peräkkäis- 130, 132, 135, 150, 343
  syöttö- 131
  tietoa sisältävä 47, 117
  tulostus- 131
Tietokanta 118, 144, 151, 344
Tila 22, 178
  grafiikka- 143, 148, 164, 178, 185
  kirjoituskone- 143, 148
Toimintanäppäimet 26, 58, 104
Toinen osoite 48, 130, 139, 143
Toistonäppäimet 28
Tosiaika 63
Trigonometriset funktiot 59
TV-liitäntä 14
Tyhjä merkkijono 96, 103, 177, 343
Tyyppihaku 35, 345
Ura 42
USR 165
VAL 173
Varatut
  muuttujat 62

sanat 76
VERIFY-käsky 53
VIC 184, 193, 249, 256, 287
muistipaikat 287
Videomonitori 13
Videosignaali 13
Virheilmoitukset 325
Välilyöntinäppäin 28, 69
Väri 24, 91, 147, 284
kääntеistilassa 24, 91
merkit 193
näytön 184, 264
spriten 264
Värikartta 281
Väritavut 281
Värähtelyt 291
WAIT 168
Wedge-ohjelma 44, 58
Yläindeksi 119
Ääni 165, 186, 211, 291, 292
ja suodattimet 320
taajuus 211, 292, 295, 296, 304
tempo 313

Suomenkielisiä MIKROKIRJOJA KUUSNELOSELLE

KIRJALLISUUS

Chris Callender
COMMODORE 64
OHJELMOINNIN OPAS
– HYÖTYOHJELMAT

Valikoima hyötyohjelmia kotitalouksille, yhdistyksille ja pienyrityksille. Kirjan sisältämät ohjelmat on sovitettu Suomen oloihin ja ohjelmat voidaan edelleen muokata omiin erityistarpeisiin.
Kierreselkä.

Owen Bishop
COMMODORE 64
OHJELMOINNIN OPAS
– PELIT

Valikoima testattuja, jännittäviä ja taitoa vaativia pelejä. Kirja on opas Commodore 64:n grafiikan ja äänentoistomahdollisuuksien hyödyntämiseen sekä peliohjelmointiin.
Kierreselkä.

Norbert Treitz
COMMODORE 64
OHJELMOINNIN OPAS
– GRAFIIKKA

Kirja esittelee Commodore 64:n grafiikan ominaisuuksia. Täydellisten ohjelmalistausten avulla esitetään 111 edelleen muokattavaa grafiikkaohjelmaa, jotka on tehty SIMONS' BASIC-laajennuksella.
Kierreselkä.

Norbert Treitz
COMMODORE 64
OHJELMOINNIN OPAS
– GRAFIIKKA + KIRJAN OHJELMAT SISÄLTÄVÄ LEVYKE

Vaihtoehto niille, jotka eivät halua uhrata aikaa kirjan ohjelmien sisäänkirjoittamiseen omalla 64:llä.

AMERSOFT

TIM ONOSKO

COMMODORE 64®

KAIKKI KUUSNELOSESTA

Kaikki Kuusnelosesta on asiantunteva ja kattava esitys Commodore 64 -tietokoneen eri osa-alueista ja sen monista ainutlaatuisista piirteistä. Teos jakautuu kahteen osaan, joista ensimmäinen osa sisältää selvityksen Commodore 64 -tietokoneen ominaisuuksista ja käyttömahdollisuuksista. Lisäksi tässä osassa on BASIC-kielisen ohjelmoinnin perusteet. Kirjan toinen osa on asiantuntijoiden kirjoittama ja tarkoitettu kokeneemmille Commodore 64 -käyttäjille. Se käsittelee Kuusnelosen toimintaperiaatteita, tietokonegrafiikkaa ja -musiikkia.

Kaikki Kuusnelosesta on helppotajuisesti ja selkeästi kirjoitettu teos, joka soveltuu Commodore 64 -käsikirjaksi. Kirja sisältää arvokasta tietoa Kuusnelosen sovelluksista kotona, koulussa ja työelämässä. **Kaikki Kuusnelosesta** auttaa Commodore 64 -tietokoneen täysipainoisessa hyödyntämisessä.

61

ISBN 951-35-3230-5