Pearce Bland

KAIKKIEN MSX

AMERSOFT

Kaikkien MSX

Graham Bland ja Jonathan Pearce

AMERSOFT

Englanninkielisestä alkuteoksesta
MSX an introduction

Suomentanut
Jukka Jääskeläinen

Kansi
Adam Korpak

Tampereen Kirjapaino Oy Tamprint
Tampere 1985

ISBN 951-35-3340-9

Sisällys

Esipuhe

MSX-standardin julkistaminen on epäilemättä kotitietokoneiden historian tärkein tapahtuma. MSX:llä on todella paljon kauaskantoisemmat vaikutukset kuin yksittäisillä laitteilla, esimerkiksi Sinclairin ZX Spectrumiin verrattavilla erittäin suosituilla laitteilla.

Tähän asti kotitietokoneiden markkinat ovat olleet erittäin hajanaiset, ja monet laitteet ovat olleet joko täysin tai lähes täysin yhteensopimattomia. Pelit ja ohjelmatarjonta ylipäänsä on laadittu vain yhden laitteen käyttöön. Sauvaohjaimet ja muut lisälaitteet, jotka on suunniteltu vain tietylle koneelle, eivät nekään ole olleet kytkettävissä muihin koneisiin. Tuosta kaikesta on aiheutunut käyttäjälle turhia hankaluuksia ja kustannuksia kautta linjan. Vaikka kotitietokoneille löytyykin runsaasti ohjelmia, on tarjonta jakaantunut melko laajalle, joten vain murto-osa ohjelmistoista on toiminut kullakin koneella. Edes suosituimpien laitteiden käyttäjä ei ole välttämättä onnistunut hankkimaan omaan tarpeeseensa parhaiten soveltuvaa pakettia.

MSX osoittaa, että tuollaiset vaikeudet ovat ohi. Se asettaa yleiset laitteistomääritykset, jotka koskevat kaikkia MSX-laitteita, ja käyttää niille yhteistä MSX-BASIC-kieltä sovellusten ja pelipakettien kirjoittamiseen. Niinpä minkä hyvänsä MSX-laitteen ostaja voi luottaa siihen, että kaikki MSX-ohjelmistot sopivat ehdottomasti ja täydellisesti hänen valitsemaansa koneeseen. Tämä uudistus helpottaa huomattavasti ohjelmistoja laativien työtä, sillä heidän ei tarvitse enää sovittaa ohjelmia erikseen kymmeniä samankaltaisia mutta yhteensopimattomia laitteita varten. Hyöty on sitäkin suurempi käyttäjälle, jolle on tarjolla ennennäkemätön valikoima ohjelmistopaketteja. Ja

luulisi ohjelmien osoittautuvan melko edullisiksi jo myynnin mittavuuden takia.

MSX-idea perustuu siis varsin järkevälle pohjalle ja hyödyttää suuresti tietokoneen käyttäjää. Ei olekaan liioiteltua sanoa, että MSX-standardia tullaan pitämään merkkipaaluna matkalla yhtenäiseen tietokonekulttuuriin.

Myös MSX:n taustalla olleet vaikuttimet ja järjestelmää tukevat yhtiöt ovat mielenkiintoisia. Toisin kuin Isossa-Britanniassa ja Yhdysvalloissa, Japanin kotitietokonemarkkinat ovat olleet melko kehittymättömät. Keväällä 1983 NEC ja Matsushita ottivat yhteyttä Microsoftiin, joka on maailman suurimpia itsenäisiä mikrotietokoneohjelmistojen valmistajia. Tarkoituksena oli laatia määritykset yhteisesti kehitettävälle henkilökohtaiselle tietokoneelle. Yhtiöt toivoivat, että Microsoft laatisi teollisuudelle standardiksi muodostuneesta BASICistaan uuden muunnelman tuota konetta varten, jotta myynti Japanin kotimarkkinoilla kehittyisi. Heinäkuuhun 1983 mennessä 12 muuta valmistajaa, mukaan lukien yksi yhdysvaltalaisyhtiö, oli pyytänyt Microsoftilta samaa asiaa.

Jotta ei olisi tarvinnut laatia 14 BASICin versiota ja jotta kotimikrojen yhteensopivuus ei olisi entisestään huonontunut, Microsoft päätyi MSX-standardin luomiseen. Siihen tarvittiin yleisesti saatavilla olevia mikroprosessoreita ja tietenkin optimoitu, laajennettu versio Microsoftin BASIC-tulkista. 16. kesäkuuta 1983 Microsoft julkisti MSX-standardin, jota tukivat Canon, Fujitsu, General, Hitachi, JVC, Kyocera, Matsushita, Mitsubishi, NEC, Pioneer, Sanyo, Sony, Spectravideo (Yhdysvallat), Toshiba ja Yamaha – kaikki suuria, kotielektroniikassa asemansa vakiinnuttaneita yhtiöitä.

Julkistaminen osoitti sen, että Japanin tietokoneita ja kotielektroniikkaa valmistava teollisuus oli täysin sitoutunut tuottamaan yhtenäisen määrityksen mukaisia kotitietokoneita. Ensimmäiset tulokset nähtiin loka-

kuussa 1983 Japanin elektroniikkanäyttelyssä, kun esiteltävien MSX-koneiden – Hitachi, JVC, Matsushita, Mitsubishi, National, NEC, Sanyo, Sony ja Toshiba – pelikasetteja saattoi yleisön hienoiseksi hämmästykseksi vaihdella keskenään.

Jo pelkästään japanilaisten markkinointiteho ja tuotantovalmius (aluksi 53 000 MSX-konetta kuukaudessa) sanelevat tietokoneteollisuuden tyvipään tuotteille standardin suunnilleen yhtä mullistavasti kuin seitsemänkymmenluvun alussa tapahtui kasettinauhoille.

On tietenkin hyvin vaikea laatia ennusteita, sillä MSX kasvaa valtavaa vauhtia. Hankkeella on kuitenkin takanaan muutamia virstanpylväitä, jotka tarjoavat jonkin verran arviointimahdollisuuksia.

Ensiksi, Microsoft julkisti 5. lokakuuta 1983 MSX-DOSin, 8 bitin käyttöjärjestelmän, joka on suunniteltu juuri MSX-koneita varten. Voidaan lyhyesti todeta, että MSX-DOSin avulla MSX-koneet eivät pelkästään kykene hyödyntämään levyketallennusta – jolloin tietoja voidaan tallentaa valtavasti lisää – vaan mahdollistavat myös niiden sovittamisen MS-DOSiin ja XENIXiin, mikro- ja minitietokoneiden yleisiin käyttöjärjestelmiin.

Yksinkertaistaen voidaan todeta, että käyttäjä voi hyödyntää sovellusohjelmia, kuten taulukkolaskentaa tai tekstinkäsittelyä, toimistonsa mikrolla ja ottaa päivän päättyessä datalevykkeen kotiinsa MSX-koneella viimeisteltäväksi. Ei enää kaksinkertaista työtä eikä pitkiä iltoja toimistossa!

Toiseksi, MSX:n BASICia voidaan laajentaa valvomaan lisälaitteita miltei rajattomasti. Tavanomaisten kasettisoittimien, kirjoittimien, sauva- ja peliohjaimien ja kosketuslevyjen lisäksi voidaan ohjata kehittyneitä äänisyntetisaattoreita, radiovirittimiä ja videolevyjä. Jo nyt on ilmestynyt laitteita, joilla voi yhdistää videotekniikkaa grafiikkaan, videokameroita nauhureihin ja valokyniä robottien käsiin. Varhainen Yamahan kone on

suunniteltu opettamaan syntetisaattorinsoittoa, joten mukana seuraa täysimittainen syntetisaattorin kosketti-misto. Ensimmäinen Sanyon kone kykenee "sieppaa-maan ruudun" eli tallentamaan TV- tai videokuvan ja muuttamaan sitä käyttäjän grafiikkaohjeiden mukaan. Huhuillaan jopa MSX:n super-hifistä, eikä ole periaat-teessa mahdotonta sekään, että kaikki kotielektro-niikka olisi MSX-sovitteista ja yhden MSX-keskusko-neen valvonnassa. Niinpä MSX avaa kotitietokoneille sovellusmahdollisuuksia, joita tietokoneilla ei ole kos-kaan ennen yritetty luoda. Nuo seikat kannattaa pitää mielessä kirjaa lukiessa, sillä siinä esiteltävät ohjel-mointikeinot auttavat pian tekemään paljon muutakin kuin vain laskemaan, kuinka paljon korkoa kasvaa korolle tai kuinka edullinen oma auto on!

Kaikkien MSX

Kaikkien MSX toimii MSX:n käyttäjän alkuoppaana. Se opettaa kaiken tarpeellisen, jotta lukija kykenee laatimaan MSX-BASICilla ohjelmia sekä huviksi että hyödyksi. Kirja ei toki pyri kertomaan MSX:stä kaikkea; erityistoimintoihin käyttäjät hankkivat ohjelmointilistauksia ja taitojen kehittyessä haluavat ehkä oppia jotakin MSX:n konekielestä, levykekäyttöjärjestelmästä ja muista ominaisuuksista.

Kirjassa edetään aivan ohjelmoinnin alkeista – MSX-BASICin käskyjen nimistä, merkityksistä ja kyvyistä – ohjelmistorakenteiden perustana oleviin näkemyksiin, jopa edistyneen BASIC-ohjelmoinnin hienouksiin. Mukana on myös oma erillinen lukunsa sekä grafiikalle että musiikille. Kumpaakin käytetään erikseen omalla erityiskielellään MSX-BASICin kautta. Sen avulla voit itse säveltää ja laatia pelejä.

Kirjan esittelyosassa annetaan yleiskuva tietokoneista ja varsinkin MSX-koneista. Esipuheesta huomaa, että MSX on joltinenkin poikkeus henkilökohtaisten ja kotikäyttöisten tietokoneiden joukossa. Ensimmäisessä luvussa esitellään terminologiaa sekä koneiden, niiden lisälaitteiden ja ohjelmien perusteet.

Toivottavasti kirjan lukeminen on yhtä hauskaa kuin sen kirjoittaminen oli. Toivottavasti lukeminen myös osoittautuu vaivan arvoiseksi.

Tammikuussa 1984

Jonathan Pearce
Graham Bland
Mark Adams
Tom Lewis
Reflex Communications Ltd

MSX:n esittely

Onpa kiinnostuksesi tietokoneisiin minkälaista tahansa, tiedät varmaan jo jotain niiden toiminnasta tai olet ainakin kuullut alan ammattislangia. Tässä luvussa esitelläänkin yleisimmät ilmaukset ja kuvataan tarkkaan, miten hankkimasi MSX-tietokone eroaa edeltäjäpolvien kotitietokoneista. Aloitamme tarkastelemalla erityyppisiä koneita ja niiden yleisimpiä sovelluksia. Sitten käsitellään tietokonejärjestelmän rakenneosia: tietokonetta **laitteena,** siihen kytkettäviä **oheislaitteita** ja käytettäviä **ohjelmistoja.**

Mikrot, minit ja keskuslaitteet

Juuri **mikrotietokoneen** ilmestyminen on epäilemättä eniten edistänyt tietokoneiden hyväksyntää. Mikrotietokoneiden vaikutus on todella ollut niin suuri, että tietokoneella kirjoittamisen taitoa pidetään usein jopa modernin elämän perusvaatimuksena – joidenkin mielestä perusasioihin kykenemättömät voisivat saman tien luovuttaa!

Kotitietokoneet olivat aluksi lähinnä paranneltuja versioita taskulaskimista. Nykyisin ne ovat erittäin kehitty-

neitä laitteita ja miltei kaikkien kukkarolle sopivia. Miljoonat ihmiset eri puolilla maailmaa ovatkin laittaneet ne hyötykäyttöön leikkien sijasta. Esimerkiksi ensimmäiset MSX-koneet eivät ainoastaan sovi pelivälineiksi ja peruslaskutehtäviin, vaan ne sisältävät myös erityistoimintoja, joiden avulla voi opetella syntetisaattorin koskettimiston käytön tai "siepata" kuvan televisioruudusta ja syöttää kuvaan grafiikanmuokkauksen avulla itse tehtyjä kuvia. Vielä muutama vuosi sitten tämänkaltaisia keinoja olisi pidetty uskomattomina, joten MSX:ää voidaan hyvinkin pitää kvanttihyppynä.

Ei sovi kuitenkaan unohtaa, että kotitietokoneet ovat vain pieni osa mikrotietokoneteollisuutta. Mikrotietokoneet ovat alkaneet vaikuttaa myös liikemaailmassa, jossa niitä käytetään yleisesti tekstinkäsittelyyn, taloussuunnitteluun, tilinpitoon ja varastovalvontaan.

Mikroista seuraava askel ylöspäin on **mini-** eli **pientietokone.** Pientietokoneet olivat hyvin suosittuja 1970-luvun lopulla ja 1980-luvun alussa, jolloin niitä pidettiin käytännöllisenä vaihtoehtona **keskuslaitteille.** Keskuslaitteet ovat valtavia tietokoneita, joita käytetään pääasiassa suurimittaisiin tehtäviin, kuten verohallinnon valtavien tietomäärien varastointiin. Pientietokoneet ovat tavallaan mikrojen ja keskuslaitteiden välimuoto. Niiden suosio kasvoi ripeästi, kun käyttäjät saattoivat tallentaa tietoa useihin laitteistoihin, jotka olivat puhelinyhteydessä toisiinsa, eikä heidän tarvinnutkaan jäädä yhden valtavan keskuslaitteen varaan. Pientietokoneiden teho alkoi vauhdikkaasti lähestyä keskuslaitteiden tehoa, mikä mahdollisti tiedon todella hajautetun hallinnan.

Aivan viime aikoina suurimpien mikrotietokoneiden tehon kasvu on kuitenkin tehnyt pientietokoneille saman kuin ne itse tekivät muutama vuosi varhemmin keskuslaitteille. Toisin sanoen mikrotietokoneista on tullut entistä tehokkaampia samalla kun niiden hinta on alentunut ja koko pienentynyt.

Esimerkiksi Hewlett-Packard on jopa maininnut erään mallinsa olevan "pöydälle sopiva keskuslaitteisto". Kun tietokoneiden perinteiset erot alkavat hämärtyä, mikroista tulee jopa niin merkittävä osa teollisuutta, että pelkästään mikron käyttöä varten omaksuttu tieto osoittautuu aivan riittäväksi useisiin muihinkin tehtäviin.

On aiheellista mainita lopuksi **supertietokoneet.** Supertietokoneita esiintyi muun muassa sellaisissa elokuvissa kuin *Avaruusseikkailu 2001* ja *War Games.* Ennen kuin kotimikrot oli keksitty, supertietokoneet vastasivat yleistä mielikuvaa tietokoneista: ne olivat valtavia, pelottavia ja aivan käsittämättömiä. Tämän kirjan myötä toivomme lukijan omaksuvan saman kannan kuin me: tietokoneet ovat yleensä pieniä, ystävällisiä ja itse asiassa hyvinkin helppokäyttöisiä!

Olemme nyt tutustuneet lyhyesti erilaisiin tietokoneisiin ja niiden käyttötarkoituksiin. Sitten otamme hieman selvää osista, joista ne muodostuvat. Yleisrakenteeltaan ja osiensa keskinäisen toiminnan kannalta hankkimasi MSX-tietokone muistuttaa sitä monta kertaa suurempia koneita. Vaikka muut tietokoneet saattavat toimia hieman eri tavoin, sisältö ja perustoimintamalli ovat kaikilla samat. Niinpä käytämme MSX:ää apuna näkemyksiemme esittämisessä. Aloitamme siitä laatikosta, joka varsinaisesti kannetaan kaupasta – laitteistosta.

Laitteisto

Tuskin kukaan tämän kirjan lukija on voinut olla kuulematta **mikrosiruista.** Niitä näyttää putkahtelevan kaikenlaisten laitteiden osiksi, leivänpaahtimista ja mikroaaltouuneista televisioihin ja videoihin. Niitä on asennettu jopa autoihin, jotta ne neuvoisivat kuljettajiaan! On todellakin vaikea kuvitella yhtään sähkömekaanista laitetta, johon mikrosirua ei kohta lisättäisi.

MSX-koneessa on kolme keskeistä sirua eli **mikrosuoritinta,** joilla kaikilla on oma erityistehtävänsä. Tärkein suoritin on **CPU** eli **keskussuoritin,** joka MSX-koneissa on Zilog Z80A (epäilemättä kaikkein yleisin siru nykyisissä kotitietokoneissa). Keskussuoritin on kuin koneen moottori. Se suorittaa kaikki koneen toiminnalle välttämättömät matemaattiset tehtävät ja sillä on ohjeiden vastaanottamiseksi oma kielikin: **konekieli.** Käytätpä tietokoneen kanssa mitä kieltä hyvänsä –BASICia, LOGOa, Assembleria tai muita – se käännetään kuitenkin lopuksi keskussuorittimen konekielelle. BASICin kaltaisten kielten kääntäminen on raskasta ja aikaavievää puuhaa. Toiset kielet, kuten Assembler, eivät tarvitse paljoa kääntämistä ja toimivat siis nopeammin kuin BASIC. Ne ovat kuitenkin hyvin hankalia välineitä ohjelmoinnissa. Kaikki tietokoneeseen hankkimasi pelit ja pitkälle kehittyneet ohjelmat ovat todennäköisesti konekielellä laadittuja.

Toinen tärkeä siru on Texas Instrumentsin grafiikkasuoritin TMS 9918A. Sen avulla tietokone hoitaa kaiken kuvaruutuun ilmestyvän. Monissa kotitietokoneissa on ainoastaan yksi suoritin, jonka täytyy hoitaa kaikki tehtävät. MSX-koneissa Z80-keskussuoritinta käytetään ainoastaan näppäimistön seurantaan ja sen ohjeiden suorittamiseen, jotta laite tietäisi mitä käyttäjä haluaa. Grafiikkasuoritin ja General Instrumentsin äänisiru AY-3-8910 valvovat muita toimintoja. Näiden keinojen avulla luotiin MSX-koneiden erinomaiset grafiikka- ja musiikkiominaisuudet, joita kokeilemme ohjelmoinnin yhteydessä. Grafiikan ja musiikin makrokieltä käsittelevissä luvuissa on monia ohjelmia, joista näkyy selvästi, kuinka noita ominaisuuksia voi hyödyntää.

Muisti

Kun pääsemme pitemmälle tietokoneen suoritus- eli prosessointikyvyn käsittelyssä, huomaat varmaankin käsikirjasta, että laitteessa on vähintään 32K **RAM**ia ja

32K **ROM**ia. Mutta mitä nuo 32K, RAM ja ROM oikein tarkoittavat?

Jotta MSX-koneen suorittimet toimisivat, niiden täytyy saada tietoa jostain. Ne eivät voi varastoida tietoa itseensä vaan ainoastaan prosessoida sitä. Jotta tietoa voitaisiin varastoida tietokoneeseen, siinä täytyy olla muistiväline, ja silloin tulevat mukaan RAMit ja ROMit.

Tietokoneen täytyy "muistaa" kahdenlaista informaatiota. Ensimmäinen informaatiotyyppi ovat ohjeet, joita keskussuorittimen eli CPU:n täytyy noudattaa tehtävää suorittaessaan. Nuo ohjeet on ryhmitelty ohjelmiksi, joilla kone toteuttaa tiettyjä tehtäviä, kuten pelejä tai tekstinkäsittelyä. Toinen informaatiotyyppi on varsinainen käsiteltävä tietoaines eli **data.** Dataa ovat sanat ja luvut joita ohjelma muokkaa, kuten pelitulokset tai käsiteltävän asiakirjan teksti.

Ennen kuin CPU voi toteuttaa mitään, sille täytyy ilmoittaa, mitä aineksia sen on käsiteltävä (vaikka kertoakseen kaksi lukua keskenään) ja mitä niille (eli mainituille luvuille) on tehtävä. Ohjelman toteuttamiseen CPU saa ohjeet ja datan RAMista (Random-Access Memory) eli käyttömuistista. Jokaista ostamaasi tai laatimaasi ohjelmaa suorittaessaan CPU hakee tarpeelliset ohjeet ja tiedot RAMista eli **lukee** RAMista. Jos ohjelma on levykkeellä tai kasettinauhalla, täytyy tiedot ensin siirtää RAMiin, jotta CPU voisi käyttää ohjelmaa.

Kun tietokone suorittaa laskutehtävän, se sijoittaa vastauksen RAMiin (eli **kirjoittaa**). RAMista vastaus voidaan esittää ruutuun tahi kirjoittaa levykkeelle tai nauhalle.

ROM (Read-Only Memory) on erityismuisti, josta CPU kykenee lukemaan, mutta johon se ei kykene kirjoittamaan. RAMin ja ROMin ero on se, että kun tietokoneesta katkaistaan virta, RAMiin tallennettu data ja ohjelmistot katoavat kuten taskulaskimista. Jos ohjel-

maa halutaan käyttää toiste, se täytyy antaa RAMiin uudestaan näppäimistön, levykkeen tai nauhan kautta. Sitä vastoin ROM säilyttää aina sisältönsä, vaikka virta katkaistaankin koneesta.

ROMin kyky säilyttää tietonsa on erittäin hyödyllinen. Esimerkiksi kun tietokoneeseen kytketään virta, se alkaa heti noudattaa ROMiin tallennettuja ohjeita. Ensimmäinen ohje käskee tarkistamaan koneen osat. Koska ohjeet ovat ROMissa, ei niitä tarvitse antaa joka kerta erikseen virran kytkemisen jälkeen.

ROMin toinen tärkeä käyttö on MSX-BASICin toiminta-ohjeiden varastointi. Niinpä aina kun virta kytketään, kone tarkistaa itsensä ja tervehtii käyttäjäänsä ruutuun ilmestyvällä ilmoituksella "OK", joka osoittaa, että MSX-BASIC on käyttövalmis.

Se RAMista ja ROMista, mutta mitä tarkoittaa 32K?

Bitit ja bytet

32K on sen muistin määrä, jota tietokone voi käyttää. Asian ymmärtääkseen täytyy hieman ottaa selvää tavasta, jolla koneet tallentavat ja käyttävät tietoa.

Tietokone siirtää ja varastoi vain koodattua informaatiota. Koodaus on välttämätöntä, koska vain siten käsiteltyä tietoa voidaan siirtää sähköisesti, ja olisi epäkäytännöllistä lähettää kukin tiedonmurunen omalla erillisellä jännitteellään. Niinpä tietokone siirtää informaatiota vain kahden jännitemäärän avulla: 0 voltin ja noin 5 voltin. Koska käytössä on vain nuo kaksi jännitettä, niitä täytyy yhdistellä, jotta voitaisiin kuvata merkkejä, esimerkiksi kirjaimia. Jännitesykäyksiä kutsutaan **biteiksi.** Bitti on joko "1", joka tarkoittaa 5 volttia, eli että virta on kytketty, tai "0", joka tarkoittaa 0 volttia, eli ettei virtaa ole kytketty.

Esimerkiksi jos kolme bittiä (joko 5 V:n tai 0 V:n

merkkejä) lähetetään kerralla, niillä voidaan esittää kahdeksaa eri merkkiä seuraavan mallin mukaisesti:

Merkki Luku	Ensimmäinen Bitti	Toinen Bitti	Kolmas Bitti
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Mallista voi päätellä, että jos kaksi tietokonetta sopisi lähettävänsä toisilleen informaatiota kolmen sykäyksen mittaisissa jaksoissa, ne voisivat viestiä kahdeksan merkkiä. Vain kahdeksasta merkistä koostuva viestintä olisi toki jokseenkin riittämätöntä. Tietokoneiden bittijonoa nimitetään sanaksi. Esimerkkikoodistomme sanat ovat sen mukaisesti kolmebittisiä. On tärkeää muistaa, että tietokoneen "sana" vastaa tavallisessa kielessä esiintyvää yksittäistä merkkiä.

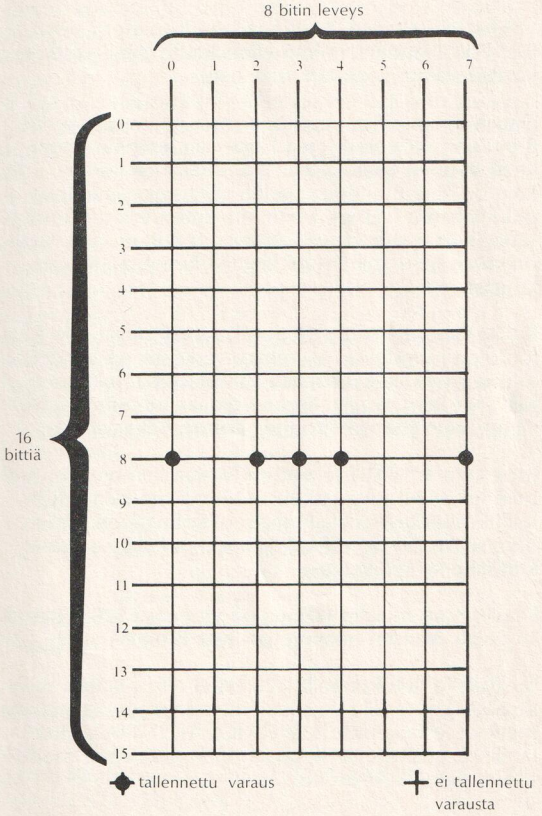
Jos koodi pidennetään nelibittiseksi, tietokone voi lähettää jo 16 merkkiä. Asian voi haluttaessa tarkistaa listaamalla kaikki mahdolliset merkit. Kuvitellaanpa vaikka, että luettelon jokaisen luvun eteen lisätään "0" (jolloin saadaan kahdeksan nelibittistä lukua, jotka kaikki alkavat 0:lla) ja sitten vaihdetaan "0" "1":ksi (jolloin saadaan taas kahdeksan nelibittistä lukua, jotka kaikki alkavat 1:llä). Laskemalla molemmat ryhmät yhteen saadaan lopputulokseksi 16 nelibittistä merkkiä.

Koska on tarpeellista lähettää lukuja, kirjaimia, välimerkkejä ja joitakin erityismerkkejä, kuten *, £ ja >, tietokone tarvitsee merkkiä kohden huomattavasti enemmän kuin vain neljä bittiä. Tietokoneiden valmistajat ja käyttäjät ovatkin sopineet seitsemän bitin käy-

ASCII Koodi	Merkki	ASCII Koodi	Merkki	ASCII Koodi	Merkki
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	\
007	BEL	050	2	093	]
008	BS	051	3	094	Λ
009	HT	052	4	095	<
010	LF	053	5	096	'
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	>	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035		078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

1 ASCII-koodit
(ASCII-koodit ovat desimaalijärjestelmän mukaisia.
LF = rivinvaihto, CR = telanpalautus ja DEL = poisto ovat esimerkkejä ohjauskoodeista 1–32.)

töstä merkkiä kohden. Miltei kaikki mikrot käyttävät tuota American Standard Code for Information Interchangen (ASCII) koodia. Kaaviossa 1 on esitetty koko ASCII-koodisto.



2 Tämä kaavio kuvaa muistikennostoa.

Mustat kiekot johdosristeyksissä osoittavat kohdat, joihin on syötetty sähkövaraus eli "1"-bitti. Jokaisella tavulla on oma muistipaikkansa; esimerkkimme binaaritavu 10111001 on tallennettu verkolla muistipaikkaan 8.

Vaikka ASCII onkin 7-bittinen koodisto, tieto siirtyy 8-bittisinä sanoina, jolloin ylimääräistä bittiä voidaan käyttää datansiirron tarkistukseen.

Datansiirron lisäksi tietokone myös tallentaa merkit 8-bittisinä. Tallennuksessa käytetään kuitenkin nimikettä **byte** eli **tavu.** "Sanan" ja "tavun" ero on se, että tavu on aina 8-bittinen, mutta sana voi olla kuinka pitkä tahansa (vaikka 3-bittinen, kuten esimerkkimme sana). Kun kyseessä on kahdeksanbittinen sana, sana on myös tavu, jolloin voi käyttää kumpaa ilmausta tahansa.

Kuvitellaanpa sitten, että tietokoneen muisti olisi kuin johtojen muodostama verkko. Verkossa on kahdeksan johtoa pysty- ja kuusitoista vaakatasossa. Jokaiseen johtojen leikkauspisteeseen voidaan aiheuttaa sähkövaraus kuin pieneen kondensaattoriin (kaavio 2).

Jotta tavu 10110011 tallentuisi (vastaa kymmenjärjestelmän lukua 51, joka tarkoittaa tiettyä merkkiä ASCII-taulukossa), täytyy ensimmäisen, kolmannen, neljännen, seitsemännen ja kahdeksannen leikkauspisteen kohdalle syntyä varaus.

Kaaviomme muistin laajuus on yhteensä 128 bittiä eli 16 tavua. Muistin rakenne on 16 x 8 bittiä.

Tavalliselle tietokoneelle 128 bittiä olisi mitätön muistimäärä, koska sillä voisi tallentaa enintään kuusitoista merkkiä (esimerkiksi "OHJELMOITAVUUTTA"). Muutama vuosi sitten 4096 tavua oli kohtuullinen muistimäärä pienelle mikrolle.

Tietokonealalla ei muistia kuitenkaan mainita 4096 tavun suuruiseksi, vaan käytetään kirjainta "K", joka tarkoittaa 1024 tavua (kuten K on 1000 tutussa ilmauksessa 1 km), joten 4096 tavua on 4K. Luku 1024 saattaa tuntua hieman oudolta kantaluvulta, mutta se valittiin, koska kahdeksan bittiä on sangen yleinen sananpituus. Jos sanan pituuteen lisää bitin, käytettävissä oleva sanasto kaksinkertaistuu – kuten osoitimme 3-bittisten sanojen muuttamisesta 4-bittisiksi. Jos kaksinkertaistamista jatketaan kahdeksanbittiseen, vaihtoehtoja saadaan 1024, ja koska jokaisen muistin laajuus on joko luvun 1024 kerrannainen tai murto-osa, on mielekästä käyttää pikemminkin sitä kuin k:n varsinaista vastinetta eli 1000:tta.

Ennen 4K:n muisti oli siis mikroissa yleinen. Nykyään ohjelmat laaditaan siten, että niiden tallentaminen ja käyttö edellyttää usein vähintään 8K:n tai 16K:n muistia. Liike-elämässä käytettävissä suurissa mikroissa muisti on yleensä joko 64K tai 128K (eli 64 kerrottuna kahdella). Niiden muistia voi usein laajentaa 256K:oon, 512K:oon tai jopa 1024K:oon. Viimeksi mainitussa tapauksessa K:n tilalle tulee kirjain M, joka tarkoittaa megabyteä eli -tavua; 1M tai 1MB tarkoittaa siis 1024 kilotavua (ks. taulukko 3).

1 bitti	=	O tai 5 voltin sähkömerkki
1 sana	=	tietty määrä bittejä (konekohtainen, yleensä kuitenkin 8 tai 16)
1 tavu	=	8 bittiä
1 kilotavu (K)	=	1024 tavua
1 megatavu (M)	=	1024 kilotavua

3 Tiivistelmä nimistöstä

MSX-tietokone rakentuu 8-bittisiä sanoja käyttävään mikrosuorittimeen ja siinä on ROM-muistia 32K ja RAM-muistia 32K – 64K.

On tärkeää muistaa, että mitä suurempi ja mutkikkaampi koneelle annettu ohjelma on, sitä enemmän muistia koneella täytyy olla. Kaikkien MSX-koneiden RAMia ja ROMia voidaan laajentaa, mikä kannattaa muistaa ohjelmia laadittaessa. Vaikka koneen vähimmäismuisti onkin 32K, siitä tarvitaan 16K näyttöruudun tietojen tallentamiseen, joten alkuperäinen 32K onkin supistunut 16K:oon. Joka tapauksessa tämän kirjan kaikki ohjelmat mahtuvat helposti tuohonkin tilaan. Seuraapa, kuinka vähäisellä muistimäärällä kyetään hyvinkin paljoon.

ROM-laajennukset

Kotitietokoneiden lyhyen taipaleen aikana on tullut tavaksi hankkia ja tallentaa informaatiota kasettinauhuriin. Tarkastelemme kohta tuota hieman ongelmalliseksi osoittautunutta menetelmää. ROM-laajennukset sitä vastoin ovat osoittautuneet perin mutkattomiksi.

Kun ROM-laajennus kytketään tietokoneeseen, laitteen sisään asennettu ROM syrjäytyy heti. Niinpä ruutuun ilmestyykin MSX-BASICin sijaan peli, kotikirjanpito-ohjelma tai jokin muu tietokoneeseen hankittu ROM-laajennus. Kyse ei ole pelkästään hyvin helposta keinosta hankkia ohjelmia, vaan siten ohjelman koko ei rajoitu pelkästään koneen RAMien määrään. Mikään ei estä lataamasta 64K ROM-laajennusta, vaikka tietokoneessa onkin vain 32K RAMia.

Näppäimistö

Viimeisimpänä laitteiston osana käsitellään vihdoin näppäimistö. On useitakin perusteita käsitellä näppäimistö oheislaitteiden yhteydessä, kuten varsin pian käy ilmi. Näppäimistö käsitellään tässä yhteydessä, koska useissa kotitietokoneissa siihen liittyy yksi tärkeä ominaisuus: ohjelmoitavat toimintonäppäimet. Konetta käynnistettäessä huomaa, että viidellä näppäi-

mellä kyetään SHIFT-näppäimen avulla kymmeneen erityistoimintoon. Toimintonäppäimien käytön vaihtamisesta kiinnostuneet voivat vilkaista seuraavassa luvussa mainittuja KEY-käskyjä.

Tietokoneesi taitojen ja keinojen käsittelyn jälkeen selvitämme, kuinka se viestii ympäristön kanssa oheislaitteiden avulla.

Oheislaitteet

Oheislaite on pelkistetysti mikä tahansa laite, jonka voi liittää tietokoneeseen. Tuo määritys kattaa valtavan valikoiman laitteita. Mukaan voidaan lukea kaikki laitteet, jotka mahdollistavat informaation siirtämisen tietokoneeseen tai tietokoneesta, tallentamisen tai erityistehtävät. Kukin tyyppi käsitellään vuorollaan.

Tiedon syöttö ja tulostus

Kuten luonnollista, tietokoneeseen syötetään informaatiota näppäimistön avulla, ja juuri sen vuoksi näppäimistöä ei haluttu käsitellä peruslaitteiston yhteydessä. Koska sen avulla voidaan syöttää informaatiota, sitä kutsutaan **syöttövälineeksi.** MSX ei sisällä mitään standardia näppäimistön asettelulle, mutta kaikissa MSX-tietokoneissa on joka tapauksessa toimintonäppäimiä juuri se määrä, joka mainittiin edellisessä jaksossa. Ja vaikka ne ensi silmäykseltä näyttävätkin vain vähäisiltä kirjoituksen nopeuttimilta, niitä voi käyttää sangen monin tavoin, kuten koeohjelmasta huomaa.

Ennen kuin tarkastelemme muita syöttölaitteita, vilkaistaan tyypillisintä **tulostinta,** televisiota. Miltei kaikki saatavilla olevat kotitietokoneet käyttävät informaation näyttämiseen ensisijaisesti televisiota, koska se on entuudestaan miltei jokaisen tietokoneen ostajan kodissa.

Kodin televisio on ehkä jatkuvasti kuormitettu videoiden, teletex-palvelun ja muiden tehtävien vuoksi, joten värimonitori saattaa osoittautua edulliseksi valinnaksi käyttäjälle, joka on karkotettu olohuoneestaan. Monitorin näyttö- ja väriominaisuudet ovat paremmat kuin television, minkä ansiosta MSX:n grafiikka pääsee silloin parhaimpaansa.

Jos MSX:n väriominaisuuksille on vähän käyttöä, on edullisin vaihtoehto mustavalkoinen matkatelevisio tai monitori. Mustavalkoisenakin MSX:n grafiikka ällistyttää – sen huomasi kirjaa tehdessä.

Siinä tulivat lyhykäisesti kuvatuiksi tiedonsyötössä ja -tulostuksessa tarvittavat vakiolaitteet. Muut oheislaitteet ovat pelkästään niiden muunnelmia. Näppäimistön sijasta voidaan esimerkiksi käyttää sauvaohjaimia, jotka toimivat tietokonepeleissä nopeammin kuin näppäimistö. Valokynät pääsevät oikeuksiinsa piirustuspaketin yhteydessä, ja kosketuslevyjen edut tulevat parhaiten esiin, kun ruutuun tulee näkyviin kosketuslevyn näppäimiä vastaavia vaihtoehtoja. Kosketuslevyt ovat lähinnä toimintonäppäimien laajennusyksikkö. Kirjan ilmestymisen aikoihin saatavilla on varmaankin monia sellaisia erityisoheislaitteita, joista ei kirjoittamisen aikoihin tiedetty mitään ja joiden ominaisuuksia ylistetään harrastajalehtien mainoksissa.

Kirjoittimet

Muista tulostimista tulevat ensimmäisinä mieleen kirjoittimet, sillä niiden avulla saa työstään **kirjoitteen** eli jäljennöksen paperille. Seuraavassa käsitellään kolmea keskeisintä kirjoitintyyppiä.

Matriisikirjoittimet ovat nykyään halvin vaihtoehto. Merkit muodostuvat yleensä seitsemän vaaka- ja viiden pystytasoisen matriisipisteen avulla, tosin myös 9 x 7 pisteen matriisia käyttävät kirjoittimet ovat yleisiä. Pisteet ovat jälkiä, joita pienet neulat ovat paina-

neet joko tavallisen värinauhan tai lämpöpaperin avulla. Niillä voidaan kirjoittaa vakiomerkkejä, joillakin laitteilla jonkin verran grafiikkamerkkejäkin.

Kiekkokirjoittimien mekanismi sitä vastoin on sangen monimutkainen ja verrattavissa pallokirjoittimeen. Kukin kirjain ja symboli on kiekkokirjoittimen kehällä oman pinteensä varassa. Kiekko pyörii hurjaa vauhtia ja vasara napauttaa tietyn pinteen kärjessä olevaa merkkiä, joka painaa värinauhan paperia vasten. Koska kiekot ovat joko muovia tai metallia, on mahdollista tulostaa hyvin selkeää jälkeä (laatujälkeä). Kiekkokirjoittimet ovat myös nopeasti ja helposti muunneltavissa, esimerkiksi kirjasinmalli voidaan vaihtaa. Kiekkokirjoittimet eivät yleensä ole käytössä niin joustavia kuin matriisikirjoittimet; niillä ei voi tulostaa grafiikkaa tai vedostaa ruudulla esitettyä materiaalia. Ne ovat sitä paitsi (rakenteensa takia) hitaita ja kalliita. Niiden varsinainen etu matriisikirjoittimiin verrattuna on erinomaisessa painoasussa.

Viimeisinä käsitellään **piirturit,** joilla voidaan valmistaa kiinteä tuloste mistä tahansa järjestelmän tuottamasta kuvasta. Piirtureissa käytetään yleensä joko yhtä kynää tai useita erivärisiä kyniä (väripiirturit), joilla saadaan paperille tarkalleen sama kuvio kuin näyttöruutuun on hahmoteltu. Piirturit ovat paljon mutkikkaampia laitteita kuin kirjoittimet, eikä niitä voida käyttää MSX-mikron perusmallin yhteydessä. Siihen tarvitaan erityisohjelma, **laiteohjain,** kääntämään ruudussa näkyvä materiaali käskyiksi, joiden avulla piirturi piirtää. Laiteohjain edellyttää **käyttöjärjestelmää (MSX-DOS) ja levykkeitä,** joista tulee vielä puhe. Kerrottakoon aluksi, että sen vuoksi piirturit ovatkin huomattavasti kalliimpia kuin muut tulostimet ja sopivat vain sellaisille henkilöille, joiden täytyy tuottaa paljon mutkikkaita graafisia esityksiä, kuten rakennepiirustuksia ja kytkentäkaavioita.

Kaiken kaikkiaan, MSX-standardin ansiosta ilmestyy varmaankin aivan uusia MSX-sovelteisia kirjoittimia ja

piirtureita, jotka toivottavasti saavat hinnat alenemaan huomattavasti ja tuovat siten laitteet laajan käyttäjäkunnan ulottuville.

Muistilaitteet

Tiedonsyötön ja -tulostuksen oheislaitteiden jälkeen ovat vuorossa laitteet, joiden avulla tietoa tallennetaan. Tietokoneen ROMeista ja RAMeista olikin jo puhe. RAMien ongelmahan oli siinä, että tunteja vieneen ohjelmanlaadinnan jälkeen menettää luomuksensa, kun lopettaa ja katkaisee virran. Tarvitaan siis muisti eli laite, jonka avulla data säilyy vaikka virta katkaistaan.

Kotimikroissa voi käyttää lähinnä kahden tyyppisiä muistilaitteita: kasettinauhureita ja levykkeitä. Kumpaisellakin on haittansa ja hyötynsä. Datan tallentamisesta kasetille aiheutuu toisinaan ongelmia, joita käsiteltiinkin jo hieman, mutta kasetit ovat hyvin halpa, joskin hidas tiedontallennuskeino. Toisaalta levykkeille voi tallentaa melkoisen määrän informaatiota (vajaat 100K tai 400K tai jopa 800K yhdelle levykkeelle). Levykkeet ovat nopeita käyttää mutta myös melko kalliita. Kaikesta huolimatta, jos tarkoituksena on käyttää tietokonetta ammatillisessa mielessä eikä vain harrastusvälineenä, levykkeet ovat välttämättömiä.

Robotit ja puhesyntetisaattorit

Viimeisenä oheislaitteiden ryhmästä tarkastellaan erityissovelluksia, joiden määrän oletetaan alati kasvavan. Peliohjainten, valokynien ja kosketuslevyjen kaltaiset laitteet kuuluvat tuohon joukkoon. Niiden lisäksi ovat äänisyntetisaattorit, puheentunnistimet, robotit ja miltei kaikki muutkin välineet, jotka saattavat nykyään kuulostaa hieman tieteiskuvitelmilta mutta joista hyvinkin pian tulee aivan arkipäiväisiä. Kuten kirjan esipuheessa mainittiin, MSX-tietokone

kykenee keskustelemaan miltei minkä tahansa laitteen kanssa, kunhan vain joku järjestää keinon, jolla tavalliset kodinkoneet voidaan liittää MSX-tietokoneeseen.

Toiveikkain mielin voimmekin sitten valmistautua seuraavaan lukuun tarkastelemalla ensin tietokoneiden ohjelmistoja.

Ohjelmistot

MSX-tietokoneeseen ja sen tiedonsyöttö- ja tulostuslaitteisiin alustavasti perehdyttyämme on aiheellista kysyä: kuinka tuon kaiken tekniikan saa tekemään mitä haluan? Vastaus: sille annetaan käskyjä ohjelmien tai ohjelmiston avulla.

Kuten laitteistoa käsittelevässä osassa mainittiin, keskussuoritin ymmärtää konekieltä. On erittäin epätodennäköistä, että mikronkäyttäjä joutuisi koskaan kirjoittamaan mitään konekielellä, sillä se on uskomattoman vaikeaselkoista ja sen oppiminen vie paljon aikaa. Olisikin sangen helppoa käskeä tietokonetta omalla äidinkielellä. Ikävä kyllä, kielten suuret erot ja ilmausten epätäsmällisyys estävät sen. BASICin ilmaukset ovat jokseenkin englannin kielen mukaisia, ja tuohon kansainväliseen käytäntöön täytynee siis tyytyä.

Microsoftin BASIC on käytännössä teollisuuden hyväksymä standardikieli sekä kehittyneisyytensä että suosionsa perusteella. Se on asennettu jo yli kahteen miljoonaan mikrotietokoneeseen. Siitä kehitetty MSX-BASIC sisältää monia erityiskäskyjä, joiden avulla pääsee täysin hyödyntämään MSX-koneen ominaisuuksia. Pohjimmiltaan MSX-BASICissa on kyse siitä, että se vastaanottaa ihmisen ymmärrettävissä olevaa kieltä ja kääntää viestin kielelle jonka kone ymmärtää.

Seuraavissa kappaleissa MSX-BASICin ohjeet ja niiden vaikutukset käydään askel askeleelta tarkemmin läpi. Kaikki tietokonekielet koostuvat **käskyistä, lauseista** ja

funktioista eli **toiminnoista,** joita voidaan käyttää joko erikseen tai yhdessä, jolloin niistä muodostuu ohjelma.

MSX-BASICin ohjeiden yhteensommittelu ohjelmaksi edellyttää jäsentelyä, ja sitä kutsutaan ongelmanratkomiseksi. Jäsentelynäkemyksen tukena tietokoneohjelmoijat käyttävät lohkokaaviota, jossa ongelma paloitellaan alkutekijöihinsä.

Kun ongelma sitten on määritelty, sen osat on helppo laatia MSX-BASICin mukaisiksi ohjeiksi. Kun kaikki osat lopulta ovat yhdessä, ne muodostavat ohjelman. Tarvitaan vain rahtunen mielikuvitusta, jolla muuttaa ratkottava ongelma **lohkokaavioksi.**

Käskyt

Käskyt kertovat tietokoneelle sille tarkoitetun tehtävän, kuten RUN (= aja ohjelma) tai LIST (= luetteloi). Käskyt toteutuvat kirjoitushetkellä, eikä niitä tarvitse sijoittaa mukaan ohjelmaan.

Lauseet

Lauseet ovat ohjelmaan numeroituja ohjeita. Esimerkiksi lause LET X = 5 antaa X:lle arvon 5.

Funktiot

MSX-BASIC tarjoaa käyttöön monia matemaattisia funktioita, kuten sinin, cosinin ja tangentin. Funktioita seuraa aina jokin luku, muuttuja tai suluissa oleva tieto. Esimerkiksi SIN (sinifunktio), PRINT SIN (PI/2) ilmoittaa 90 asteen kulman sinifunktion arvon radiaaneina; jotkut muistanevat tämän kouluajoiltaan.

Siinä kielen keinot olivatkin pääpiirteissään. Tämä jakso sisälsikin melkoisesti terminologiaa, joten lienee

soveliasta lopuksi katsoa "tyypillinen" esimerkki ohjelmoinnista.

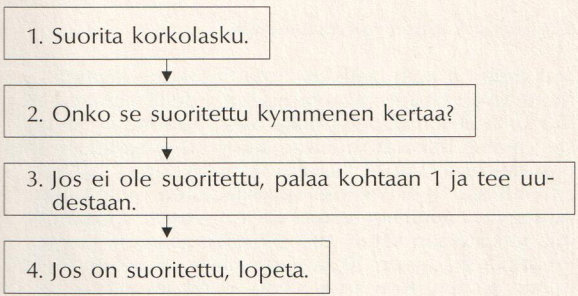
Ohjelmoinnin periaatteet

Kuten alussa mainittiin, pyrimme välttämään pankki- ja korkolaskutehtäviä, niin kiinnostavia kuin nuo asiat ovatkin. Ne tarjoavat kuitenkin helpon tavan osoittaa, mistä oikein on kyse. Ei kannata kuitenkaan huolestua, sillä muut esimerkkitehtävät ovat sitäkin kiinnostavampia.

Tässä ongelmamme. Olen sijoittanut 100 markkaa tililleni. Jos vuosittainen korko on pysyvästi 10 %, paljonko tililläni on 10 vuoden kuluttua?

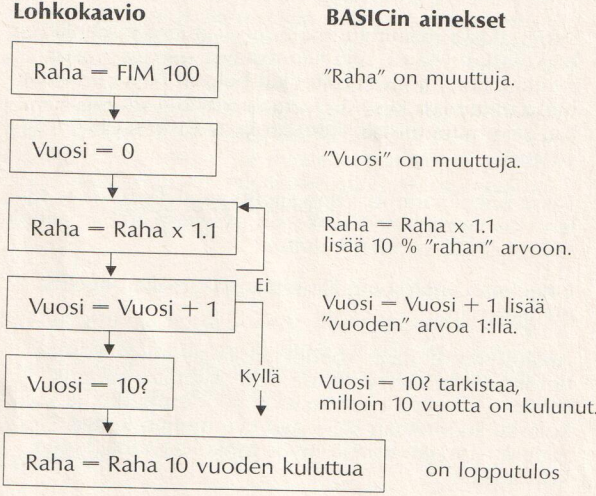
Ongelman analysointi aloitetaan jakamalla ongelma pieniin osiin.

Analysointi tehdään kolmessa päävaiheessa. Ensiksi tietokoneelle täytyy kertoa tilille sijoitettu summa. Sitten täytyy laskea ensimmäisen vuoden korko, ja korkolasku suoritetaan kaikkiaan kymmenen kertaa. Nuo vaiheet voidaan esittää myös lohkokaaviona, kuten kaaviossa 4.



4 Yksinkertainen lohkokaavio

Samat kohdat näkyvät myös kaaviosta 5, jossa näytetään myös, kuinka BASIC-muuttujia käytetään ja kuinka tietokone suorittaa sellaisia toimintoja kuten "toista kymmenen kertaa".



5 Lohkokaavio koron muodostumisesta

Muuttuja on kuin laatikko , jota tietokone käyttää muuntuvan tiedon tallentamiseen. Edellä olevassa esimerkissä tiesimme, että tilillä oleva summa ja tilivuosien määrä tulisivat muuttumaan, jotta ongelmaan löytyisi ratkaisu. "Vuosi" kasvaisi 0:sta 10:een ja "raha" FIM 100:sta 10 %:n vuosivauhdilla, joten molemmat muuttujat mainitaan lohkokaavion alussa. Vuosimuuttuja tarkistetaan sitten joka silmukan jälkeen, kunnes muuttuja saa arvon 10, ja vasta silloin silmukan toistuminen päättyy. Kun arvo on 10 eli rahamuuttuja on saanut arvon, joka osoittaa tilillä 10 vuoden kuluttua olevan summan, tämä ohjelma pysähtyy.

Jos tuon ymmärsi, eivät kirjan seuraavat kappaleet tuota lainkaan vaikeuksia. Ensiksi tutustutaan ohjeisiin ja niillä toteutuviin seikkoihin ja sitten selvitetään, kuinka ohjeita yhdistelemällä muodostetaan yhä monimutkaisempia ohjelmia.

Toivottavasti tämä tietokoneista ylipäänsä ja erityisesti MSX:stä kertonut esittely on ollut hyödyksi. Tähän päättyy kirjan johdanto-osa, ja on aika päästää yleisö mukaan. Kytkekääpä siis virta tietokoneeseen ja hankkikaa sormituntumaa. On aika kokeilla käskyjä!

MSX-BASICilla ohjelmointi

Käskyjen tulkitseminen

Kun MSX-koneeseen kytketään virta, ruutuun ilmestyy valmiusilmoitus "OK". Se ilmoittaa koneen olevan valmis vastaanottamaan ja suorittamaan BASIC-ohjeita. Ohjeet voidaan toteuttaa kahdella tavalla. Ne voidaan antaa suoraan, jolloin tietokone **tulkitsee** ja noudattaa niitä heti. Tätä kutsutaan **suoraksi käyttötavaksi.** Toinen tapa on ilmoittaa ohjeet rivinumeroina, jolloin niitä käsitellään kuin ohjelmaa ja ne suoritetaan ohjelmoidussa järjestyksessään. Ohjelman suorittaminen **epäsuoralla käyttötavalla** alkaa vasta silloin, kun käyttäjä on kirjoittanut suoraan RUN-käskyn. Helpoimmin käskyn ja sen aikaansaannoksen oppii ymmärtämään kirjoittamalla käskyn ja seuraamalla tietokoneen toimintaa. Kirjoitapa esimerkiksi:

```
PLAY "04CDEFGAB05C"
```

ja paina RETURN-näppäintä. PLAY kertoo tietokoneelle, että sen tulee soittaa jotain. "04CDEFGA05" kertoo koneelle sitten, mitä sen todella tulee soittaa. Ja konehan soitti sitten koko asteikon. Aikoinaan osoitetaan aivan tarkkaan, kuinka kone suoritti tehtävän.

Useimmat MSX-BASICin ja muidenkin BASICien ohjeista koostuvat ohjeen nimestä ja käyttötarkoituksen osoittavasta lausekkeesta. Ohjeen käytössä on erittäin tärkeää, että sitä seuraavan lauseen rakenne on virheetön. Jo senkin vuoksi kannattaa ensin tutustua suoraan käyttötapaan. Eräät esiteltävistä ohjeista ovat kuitenkin sellaisia, että ne näyttävät mielekkäiltä vasta ohjelmaan sijoitettuina. Ohjelmat on toki pidetty aina mahdollisimman lyhyinä, vaikka ne osoittavatkin täydellisesti ohjeen käyttömahdollisuudet. Aina kun on tarpeen kokeilla ohjeen käyttöä ohjelman avulla, tee seuraavasti:

1. Kirjoita ohjelma tarkalleen kirjan antamassa muodossa.

2. Kirjoita joko RUN ilman edeltävää rivinumeroa (eli suoran käyttötavan mukaan) tai paina näppäimistön toimintonäppäintä 5.

3. Useimmat itse kirjoittamasi ohjelmat päättyvät siten, että ruutuun ilmestyy jälleen BASICin valmiusilmoitus "OK". Kirjan ohjelmat pysähtyvät ja "OK" palautuu, kun painetaan samanaikaisesti näppäimiä CONTROL ja STOP. Muunnettavan ohjelman sisältö saadaan näppäimellä LIST, ja kun haluaa päästä eroon vanhasta ohjelmasta ja aloittaa uuden, täytyy kirjoittaa NEW.

MSX-BASICin avulla voi myös **editoida** ruudussa näkyviä lauseita. PLAY-esimerkissämme voi siirtää kohdistinta ja muuttaa kaikkia kirjaimia. Muutettu rivi syötetään ENTER- tai telanpalautusnäppäintä painamalla, aivan kuin tyystin uusi rivi.

Edellä kerrottu pätee kaikkiin ruudussa esitettyihin riveihin. Jos kirjoittamasi rivi on yhä ruudussa, voi riville palata, muokata sitä ja painaa ENTERiä, jolloin rivi tallentuu. Jos ENTERin painaminen unohtuu, rivi säilyy entisellään.

Tämä ominaisuus on sangen hyödyllinen MSX-BASICin ohjelmissa, jos pitkiin ja mutkikkaisiin riveihin tulee virheitä. Kun listataan rivit ruutuun, niitä pääsee korjaamaan, ilman että täytyisi kirjoittaa riviä alusta loppuun. Monissa kirjan esimerkkeinä käytetyissä ohjelmissa painotetaan sitä, että täytyy itse kokeilla ohjelmien käyttöä. Kun olet kirjoittanut esimerkkiohjelmia koneella, käytä editointia kokeillaksesi eri muunnelmia.

MSX-BASICin editorin ominaisuudet luetellaan liitteessä C. Kannattaa tutustua liitteeseen ennen kuin jatkaa kirjan tekstin parissa.

Tämän ja seuraavan, grafiikkaa käsittelevän luvun tarkoituksena on opettaa kaksi seikkaa. Tärkeintä on oppia ymmärtämään yleisimmät BASICin ohjeet – mitä ne tekevät ja miten ne sen tekevät. Tähän osaan sisällytetyt helpot ohjelmat antavat jonkinlaisen käsityksen BASIC- ohjelmista ja niiden toiminnasta sekä valmistavat luvun lopun kehittyneempien ja siten myös hyödyllisempien ohjelmien laadintaan.

On varmaankin paikallaan esitellä tässä yhteydessä merkinnät, joita käytetään ohjeiden määrittelyssä. Kuten seuraavasta AUTO-käskystä huomaa, merkkijärjestelmässä käytetään merkkejä [ ] ja < >, ja myöhemmissä lauseissa käytetään myös merkkejä ( ). Niiden käyttötarkoitukset ovat seuraavat:

<> Kaiken näiden sulkujen välissä esiintyvän täytyy kuulua ohjeeseen, muuten tietokone ei ymmärrä sitä.

[ ] Hakasuluissa esiintyvät seikat ovat ehdollisia – käyttäjä päättää itse niiden sisällyttämisen tarpeellisuudesta. Jos päättää jättää ne käyttämättä, tietokone olettaa, että käyttäjä tyytyy oletusehdotukseen (esimerkiksi AUTO: jollei käyttäjä määritä aloitusriviä tai riviväliä, rivinumerointi alkaa 10:stä ja riviväli on 10).

( ) Vain nämä sulkumerkit on kirjoitettava mukaan, sillä ne ovat osa varsinaista lauserakennetta. Niiden käyttö tulee hyvin selkeästi esiin, kun sopivien ohjeiden vuoro tulee.

Aluksi tarkastelemme yksityiskohtaisesti muutamia käskyjä. Onnea opinnoille!

AUTO [<**aloitusrivi**>] [,<**riviväli**>]

Kaikki ohjelman rivit täytyy numeroida. Tietokone toteuttaa riveillä olevat ohjeet numerojärjestyksessä. AUTO-käsky numeroi rivit automaattisesti syöttöjärjestyksessä, minkä ansiosta välttyy runsaasti aikaa vievältä tehtävältä. AUTOa käytetään ainoastaan ohjelman kirjoittamisvaiheessa. Esimerkiksi AUTO 100,10 tarkoittaa, että rivinumerointi alkaa riviltä 100 ja jatkuu 10 rivin välein; AUTO 100,20 tarkoittaa, että riviväli on 20, jne. Jos AUTOa käytetään vanhassa ohjelmassa ja jos AUTO tuottaa rivinumeron, joka on jo käytössä, rivin loppuun ilmestyy asteriski, joka varoittaa siitä, että uusi rivi syrjäyttää vanhan rivin. AUTOn voi kytkeä pois painamalla näppäimiä CONTROL ja STOP samanaikaisesti.

BEEP

Kun kirjoitetaan BEEP, syntyy äänimerkki. Tämä on yksinkertaisin ääntä tuottava käsky. Sitä jännittävämpiä löytyy SOUND-käskyn kuvauksesta ja musiikkia käsittelevästä luvusta.

CLEAR [<**jonon tila**>] [,<**ylin asema**>]

CLEAR-käsky tyhjentää tietokoneen muistista kaikki muuttujat, mutta se ei poista RAMiin kulloinkin tallennettua ohjelmaa. Kun ohjelma on suoritettu, sen muuttujat säilyvät RAMissa. CLEAR ainoastaan siivoaa muistista kaiken tarpeettoman.

CLS

CLS poistaa ruudusta kaiken sille ilmestyneen.

COLOR [<**edustan väri**>] [,<**taustaväri**>] [,<**rajausväri**>]

COLOR-käsky asettaa ruudun värit. Kuten rakenteesta huomaa, käsky on hyvin joustava ja sen avulla voi määrätä edustan, taustan ja rajauksen värin yhdellä ainoalla rivillä. Heti kun virta on kytketty, tietokone käyttää COLOR-käskyä asettaakseen näyttöruudun värit. Tietokoneella on COLOR-**oletusarvot** 15,5,4, jotka vastaavat valkoista sekä tumman- ja vaaleansinistä. Oletus on ohje, jota noudatetaan, jos muita huomautuksia tai tekstiä ei ole annettu.

Huom.! Laitekohtaisten erojen vuoksi rajausväri ei ehkä näy kaiken aikaa television ruudussa. Ohjelman suorittamisen aikana rajausväri on kuitenkin näkyvissä (tietenkin vain silloin kun sen on ensin määritellyt). Tällä kohtaa on helppotajuisuuden vuoksi vältetty ohjelmien kokeilua, mutta kolmen ruutualueen osoittaminen ei ole mahdollista ilman pientä ohjelmaa. Kirjoitapa:

```
10 SCREEN 2
20 COLOR 14,2,9:CLS
30 CIRCLE (125,100),50
40 GOTO 40
```

Nyt pitäisi vihreätaustaiseen ruutuun ilmestyä harmaa ympyrä, jota ympäröi vaaleanpunainen rajaus.

COLORilla voi säätää värit oman silmän mukaan. Esimerkiksi kirjoittamalla COLOR 12,15 saadaan vihreitä merkkejä valkoiselle taustalle. Kannattaa kokeilla, mikä yhdistelmä tuntuu mukavimmalta. Värit ja vastaavat numerot ovat seuraavan taulukon mukaiset:

Numero	Väri
0	Läpinäkyvä, näkymätön
1	Musta
2	Vihreä
3	Vaaleanvihreä
4	Tummansininen
5	Vaaleansininen
6	Tummanpunainen
7	Sinivihreä
8	Punainen
9	Vaaleanpunainen
10	Tummankeltainen
11	Vaaleankeltainen
12	Tummanvihreä
13	Sinipunainen
14	Harmaa
15	Valkoinen

CONT

CONTinue- eli jatka-käskyä käytetään BASIC-ohjelmassa, jotta tietokone jatkaisi ohjelman suorittamista pysäytyksen jälkeen, esimerkiksi silloin kun ohjelmassa on suoritettu STOP- tai BREAK-käsky.

DELETE [<**alkurivinumero**>] [-<**loppurivinumero**>]

Nimensä mukaisesti DELETE-käsky poistaa ohjelmasta rivejä käyttäjän haluamalla tavalla. Tarvitsee vain määrittää ensimmäinen ja viimeinen poistettava rivi. Esimerkiksi DELETE 20 – 40 poistaa ohjelmasta rivit 20:stä 40:een, DELETE 20 poistaa vain rivin 20 ja niin edelleen. Yhden rivin poistaa yksinkertaisimmin kirjoittamalla pelkän rivinumeron ja painamalla sitten ENTERiä. Silloin tietokone tallentaa ennestään rivillä olevan tekstin tilalle tyhjän rivin.

KEY <**näppäimen numero**>,"<**käsky**>"

MSX-tietokoneissa on viisi ohjelmoitavaa toimintonäp-

päintä, jotka SHIFT-näppäimen avulla mahdollistavat 10 ohjelmoitavan toiminnon tallennuksen. Oletustoiminnot saa selville näppäilemällä KEY LIST. Kaikki ohjelmoidut toiminnot ovat muunnettavissa KEY-käskyn avulla. Kirjoita esimerkiksi:

```
KEY 3, "COLOR 12,15"
```

Jos sen jälkeen painat näppäintä 3, ruudun näyttö vaihtuu valkopohjaiseksi ja teksti vihreäksi. Jokainen toimintonäppäin on ohjelmoitavissa minkä tahansa ohjeen mukaan. Se säästää aikaa ja vaivaa melkoisesti, koska usein toistuvia käskyjä ei tarvitsekaan kirjoittaa yhtenään. Näppäinten nimikkeet näkyvät ruudusta, ja ne voidaan poistaa siitä KEY OFFilla ja palauttaa KEY ONilla. Jotta käsky suoritettaisiin heti, on yleensä tarpeen liittää siihen telanpalautus (tai ENTER). Sen voi tehdä lisäämällä näppäimen määritykseen "CHR$(13)", esimerkiksi:

```
KEY 1, "TRON"+CHR$(13)
```

muuttaa jäljityskäskyn (TRON) niin, että se toimii pelkällä näppäin 1:n painalluksella.

LIST [<**rivinumero**>] [<**rivinumero**>]

LIST-käskyä voidaan käyttää monin tavoin ohjelman listaukseen. Pelkän LISTin kirjoittaminen tuo näyttöruutuun kulloinkin tietokoneen muistiin tallennetun ohjelman koko sisällön. LIST-käskyyn voi kuitenkin lisätä rivinumeroita, esimerkiksi:

```
LIST 20 - 40   listaa kulloisenkin ohjelman rivit 20:stä
               40:een.
LIST 20        listaa vain rivin 20.
LIST 20 -      listaa kaikki rivit rivistä 20 alkaen.
LIST - 40      listaa kaikki riviin 40 asti.
```

LLIST [<**rivinumero**>] [<**rivinumero**>]

LLIST toimii muutoin aivan samoin kuin LIST, paitsi

että se ei lähetäkään listaa ruutuun vaan kirjoittimeen – mikäli tietokoneeseen on liitetty kirjoitin.

MAXFILES = <lauseke>

Kuten sanottu, aina kun BASIC-ohjelmaa suoritetaan, tietokone asettaa muistiinsa joukon tiedostoja, jotka vuorotellen hoitavat koneelle asetettuja tehtäviä. MAXFILES-käskyn avulla voi rajoittaa koneen avattaviksi haluttavien tiedostojen määrän. Esimerkiksi MAXFILES = 10 antaa koneen avata enintään 10 tiedostoa. Tiedostojen määrä on rajattavissa 0:sta 15:een. Kun käytetään MAXFILES = 0:aa, voidaan toteuttaa vain SAVE- ja LOAD-käskyjä.

MOTOR [ON] [OFF]

MOTOR-käskyllä käynnistetään ja pysäytetään MSX-tietokoneen käyttämä kasettinauhuri. Käsky on varsin hyödyllinen esimerkiksi silloin, kun on kirjoitettu pitkä ohjelma, josta halutaan tallentaa nauhalle vain tiettyjä osia. MOTOR ON/OFF -käskyt voidaan sijoittaa ohjelmaan sopiviin kohtiin.

NEW

Aina kun ohjelman käyttö lopetetaan, siitä voi poistua kahdella tavalla. Tietokoneesta katkaistaan virta ja kytketään sitten uudelleen, mikä on vähintäänkin kömpelö menettely. Voi myös kirjoittaa NEW, mikä heti poistaa tietokoneesta sekä ohjelman että sen tiedostot ja muuttujat, ja uusi ohjelma on heti ladattavissa tai kirjoitettavissa.

RENUM [<uusi numero>] [,<vanha numero>] [,<riviväli>]

RENUM-käskyllä numeroidaan BASIC-ohjelman rivit uudelleen esimerkiksi silloin, kun uusien rivien lisääminen on tehnyt ohjelmasta liian ahtaan. Pelkkä RENUM aloittaa ohjelman rivien numeroinnin 10:stä ja

jatkaa siitä 10 rivin välein. Esimerkiksi ohjelma, jonka rivit on numeroitu 10,13,15,20,30, numeroituisi seuraavasti: 10,20,30,40,50. Vanhoja ja uusia numeroilmauksia voi yhdistellä. Esimerkiksi RENUM 15,10 muuttaa ohjelman, jonka numerointi on 10,20,30,40,50, uuteen rivitykseen: 15,25,35,45,55 – vanha rivi 10 muutettiin 15:ksi ja kaikkien sitä seuraavien rivien väli on sitten 10. Jos tietokone havaitsee, että riville on jo numeroitu ennestään jokin lauseke, luodaan uusi numero. Riviväliä voi sitten muuttaa, kuten RENUM,,20, joka muuttaa ohjelman rivit 10,20,30 uuteen muotoon 10,30,50. Aina kun rivinumerot liittyvät käskyriviin, esimerkiksi GOTO-lauseeseen, muuttuu niidenkin numerointi, joten ohjelman rakenteeseen ei tule muutoksia.

RUN [<**rivinumero**>]

RUN-käskyllä tietokonetta ohjataan suorittamaan heti jokin ohjelma. Pelkkä RUN saa tietokoneen suorittamaan kulloisenkin ohjelman alusta loppuun, mikäli kaikki rivit on kirjoitettu. Mikäli käskyn yhteydessä annetaan rivinumero, tietokone suorittaa ohjelman siltä riviltä eteenpäin. Esimerkiksi kun ohjelmassa on rivit 10,20,30,40,50, RUN saisi koneen suorittamaan kaikki rivit, RUN 30 taas saisi suoritetuksi vain rivit 30,40 ja 50.

SOUND

SOUND-käskyn avulla tietokoneen äänisiru pystyy todella ihmeellisiin suorituksiin. Tietokoneen valtava sointivalikoima tulee hämmästyttämään sinua, kunhan pääsemme musiikkia käsittelevään lukuun, mutta ehkä haluaisit kokeilla aluksi tätä ohjelmaa:

```
10 SOUND 8,5
20 FOR I=1 TO 255 STEP 2
30 SOUND 0,I
40 NEXT I
50 GOTO 20
```

(Tästä saattaa olla varsin paljon hyötyä, jos talo syttyy tuleen!)

TRON

On jokseenkin harvinaista, että joku onnistuu ensiyrityksellä kirjoittamaan haluamansa ohjelman aivan oikeaan muotoon. Aivan pienimpiä ohjelmia lukuun ottamatta kuluu melkoisesti aikaa ohjelman vikojen etsimiseen ja poistamiseen. Tuon työn helpottamiseksi MSX-BASICissa on TRACE-käsky, joka käynnistyy, kun kirjoitetaan TRON (TRACE ON). TRACE seuraa ohjelmaa kirjoittamalla ruutuun ohjelman rivit samalla kun niitä suoritetaan. Tuolloin näkee, kuinka ohjelma toimii käytön aikana. TROFF on TRONin vastakohta, joka kytkee TRACEn pois. Näitä käskyjä käytetään harvoin ohjelmien osana. Niitä hyödynnetäänkin lähinnä ennen tai jälkeen ohjelmaa.

TROFF

Katso edellisestä kohdasta.

WIDTH <näytön leveys tekstitilassa>

WIDTH-käsky määrää rivin pituuden eli merkkien määrän, joka on nähtävissä ruudusta tekstinkirjoitusvaiheessa. Kun on valittu 40 x 24 merkin näyttöruutu 0, voi näytön leveys olla enintään 40. 32 x 24 merkin näyttöruudussa voi näytön leveys olla enintään 32.

WIDTH-käsky onkin viimeinen tässä käsiteltävistä perusBASIC-käskyistä. Nyt vilkaistaan erityisiä grafiikkakäskyjä, jotka on sisällytetty MSX-BASICiin (käskyt kuvataan aikanaan yksityiskohtaisesti asiaa käsittelevässä luvussa). Vasta sitten tarkastellaan niiden yhdistelyä hyödyllisiksi ohjelmiksi.

Grafiikkakäskyt

Tähän mennessä olemme esitelleet vasta osan tietokoneen taidoista – sen kyvyn näyttää 23 tekstiriviä, joilla kullakin on 40 merkkiä, ja toimintonäppäinrivin.

Tietokoneellasi on 920 (23 x 40) merkin **resoluutio** eli erotuskyky. Pelkillä ASCII-merkeilläkin voi piirtää kuvia. Vaikka tuo ilmaisukeino on rajallinen, sillä voi toki piirtää esimerkiksi yksinkertaisia kuvia tai vaakatasoisia taulukoita.

MSX-BASICiin on sisällytetty erityisiä **grafiikkamerkkejä,** joiden avulla voi piirtää paljon vaativampia kuvia kuin pelkillä kirjaimilla tai välimerkeillä. Nuo merkit saattavat olla näkyvissä MSX-tietokoneesi näppäimistössä, mutta joka tapauksessa ne löytyvät seuraavan ohjelman avulla, joka tulostaa koko merkkijoukon ruutuun:

```
10 FOR I=1 TO 255
20 PRINT I,CHR$(I)
30 A$=INKEY$:IF A$="" THEN 30
40 NEXT I
```

Ohjelma tulostaa merkit ja niitä vastaavat numerot vieretysten. Paina jotain näppäintä, niin saat näkyviin uuden numeron ja merkin. Tutustu eri numeroihin ja merkkeihin. Jos haluat käyttää jotain merkkiä ohjelmissa, käsky "PRINT CHR$(170)" (tai muu haluamasi numero) tulostaa merkin ruutuun. MSX-BASICin **merkkigrafiikka** käyttää **ruutua 0.**

Taidokkaita kuvia varten tarvitaan ruutuun hyvin suuri resoluutio, jolloin saa käyttöön merkkejäkin pienempiä aineksia. Silloin voimme piirtää kuvia, jotka ovat paljon yksityiskohtaisempia kuin ASCII-merkeillä aikaansaadut.

MSX-BASIC tarjoaa käyttäjälleen ruutu 0:n lisäksi kolme muuta ruutua: ruudut 1, 2 ja 3. Ruutua 1 käytetään tekstiruudun vaihtoehtona, ruutuja 2 ja 3 suur- ja pienresoluutiografiikkaan.

Grafiikkakäskyjen käyttöä varten täytyy tietokoneelle ensin kertoa, kumpaa ruutua halutaan. Jos grafiikkakäskyä yrittää käyttää tavallisella merkkiruudulla, kone antaa virheilmoituksen.

Ruutujen eroavuus tulee selväksi, kun olet kirjoittanut seuraavan ohjelman:

```
10 SCREEN 2
20 CIRCLE (125,100),50
30 GOTO 30
```

Ruudussa tulisi nyt näkyä tummansinisellä taustalla oleva melko selkeä valkoinen ympyrä, jolla on vaaleansininen reunus. Kokeile ohjelmaa vielä kerran, mutta vaihda SCREEN 2:n tilalle SCREEN 3. Tällä kertaa ympyrä ei ainoastaan ole paljon paksumpi vaan myös epämääräisempi kuin edellisellä kerralla. Syynä on ruutujen resoluution erilaisuus.

Ruutu 2 on kuin millimetripaperi, jossa on 256 neliötä vaakatasossa ja 192 pystytasossa; niinpä resoluutio onkin 49152 neliötä. Koska television näyttö on melko tiuha, neliöt eli **pikselit** ovat sangen pieniä. Jokaisella ruudussa esiintyvällä pikselillä on oma **osoitteensa,** joten niiden väriä voi vaihtaa yksitellen, ja sen vuoksi on mahdollista piirtää viivoja, ympyröitä ja muita kuvioita, joista tulee aikanaan puhe. Minkä tahansa pikselin väriä voi muuttaa PSET-käskyllä. Tarvitsee vain määrittää x- ja y-koordinaateilla pikseli, jonka haluaa muuttaa. (Pikseli, jonka koordinaatit ovat 100,50, on 100:s ruudun vasemmasta reunasta ja 50:s yläreunasta.) Toiminta selviää seuraavan ohjelman avulla:

```
10 SCREEN 2
20 PSET(120,20)
30 PSET(121,30)
40 PSET(122,40)
50 PSET(123,50)
60 PSET(124,60)
70 PSET(125,70)
80 PSET(126,80)
90 PSET(127,90)
100 PSET(128,100)
110 GOTO 110
```

Ohjelma siis muutti värin pikseleiltä, joista ensimmäinen oli kohdassa 120,20 ja viimeinen kohdassa 128,100. Kun vaihdat SCREEN 2:n tilalle SCREEN 3:n ja käytät ohjelman uudelleen, huomaat eron.

Kuten ympyrää piirrettäessä huomasi, ilmeisin ero on tavassa, jolla pikselit piirtyvät ruutuihin. Sen sijaan että PSET olisi vaihtanut yksittäisten pikselien väriä, muutos tapahtuikin 4 x 4 pikselin ryhminä. Vaakatason koordinaateilla 120, 121, 122 ja 123 ei ole toisistaan poikkeavaa vaikutusta ryhmän piirtymiseen vaakatasolla. Kun vuorossa on kohta 124, ryhmä piirtyy uuteen vaakatason asemaan, joka säilyy samana kohtaan 128 asti.

Kuten ruutu 2:lla, ruutu 3:lla on myös 256 x 192 pikseliosoitetta, mutta pikselit piirtyvät neljän ryhminä, joita on kaikkiaan 64 x 48, ja siten resoluutio on 3072 ryhmää. Tuo seikka on hyvä muistaa ruutua 3 käytettäessä, koska muuten tulokset saattavat osoittautua hyvinkin sekaviksi.

CIRCLE (<**x-koordinaatti**>,<**y-koordinaatti**>),<**säde**> [,<**väri**>] [,<**sektorin alku**>] [,<**sektorin loppu**>] [,<**mittasuhde**>]

CIRCLE-käskyn rakenne on tähänastisista ehdottomasti vaikein. Tarkastellaanpa siksi hetken aikaa, mitä sen osat tekevät. Kuten ohjelmasta kävi ilmi, ympyrän voi helposti piirtää vain käskyn kolmea ensimmäistä osaa

käyttäen. Niistä kaksi ensimmäistä määrittivät ympyrän keskipisteen sijainnin, etäisyyden sivu- ja yläreunasta. Kolmas osa määritti ympyrän säteen. Kun muuttaa ohjelman arvoja, huomaa pian, kuinka ympyrää voi siirtää ja sen kokoa muuttaa. Neljäntenä käsitellään piirrettävän ympyrän väriä, joka on vaihdettavissa vaikka seuraavalla tavalla:

CIRCLE (125,100),50,10

jolloin valkoinen ympyrä vaihtuu keltaiseksi. (Täydellinen väriluettelo löytyy COLOR-käskyn kohdalta.) Mikäli halutaan piirtää vain osa ympyrästä, ilmauksen kaksi viimeistä osaa ovat ehdottoman tarpeellisia. Molemmille voi antaa arvon väliltä −2 x pii. +2 x pii − pii (pii on ympyrän kehän suhde säteeseen, kuten koulussa opetettiin, eli n. 3.142). Tietokone siis hyväksyy niille arvon väliltä −6.284. +6.284. Noiden kahden osan arvojen vaihtaminen saa aikaan muutoksia, joiden avulla osien merkitys selviää paremmin kuin yksityiskohtaisilla selityksillä. Kannattaa siis itse kokeilla, mitä niillä voi tehdä. Kun niiden merkitys on selvinnyt, voi käyttää ohjelmaa, joka on muunnelma CIRCLE-käskyn yhteydessä kokeillusta:

```
10 SCREEN 2
20 LET X=0:LET Y=-.1
30 CIRCLE(125,100),90,7,Y,X
40 LET X=X+.1:LET Y=Y-.1
50 IF X<6.2 THEN GOTO 30
60 GOTO 60
```

Käytä ohjelmaa toisen kerran, mutta vaihda rivillä 20 oleva arvo x = 0 arvoon x = 0.1.

Ohjelma on otettu mukaan, koska sen avulla on helppo osoittaa, kuinka joustavasti ja käytännöllisesti käskyn voi laatia vain joko määrittelemällä siihen lisää osia tai lisäämällä siihen muutamia käskyrivejä. Ei ole vielä suinkaan välttämätöntä ymmärtää koko ohjelmaa − tärkeintä on havaita CIRCLEn joustavuus. Lopuksi käsitellään mittasuhde. Mittasuhde on ympyrän kor-

keuden ja leveyden suhde, jonka avulla on mahdollista piirtää kaikenlaisia ellipsejä. Ei taaskaan ole tarpeen selittää kaikkea – parasta on, että itse vaihtelee arvoja, jolloin näkee vaikutuksen. Kaiken kukkuraksi voidaan vielä sijoittaa kokeeksi yksi X lisää rivin 30 loppuun:

```
30 CIRCLE(125,100),90,7,Y,X,X
```

Hupaisaa eikö vain?

COLOR [<**edustan väri**>],[<**taustan väri**>], [<**rajausväri**>]

Katso sivu 37.

DRAW <merkkijono>

DRAW- ja myöhemmin selittyvä PLAY-käsky käyttävät molemmat tehokasta grafiikkamakrokieltä. DRAW-käskyn avulla voi värittää ruudulta kohtia (pikseleitä), joiden väliin sitten piirtyy viiva. Grafiikkamakrokielessä käytetään lyhennyksiä pikselien värittämiseksi: U on ylöspäin, D alaspäin, L vasempaan ja R oikeaan. Voit esimerkiksi piirtää laatikon käskyllä, joka värittää ensin 100 pikseliä oikeaan, sitten 100 pikseliä alaspäin, 100 pikseliä vasempaan ja lopuksi 100 pikseliä ylöspäin. Sen toteuttaa seuraava ohjelma:

```
10 SCREEN 2
20 DRAW "R100D100L100U100"
30 GOTO 30
```

Tämä onkin sitten helpoimpia kuvioita, joita grafiikkamakrokielellä voi laatia. Kielen näppäryys selviää, kun tutustuu lukuun "MSX-BASICin grafiikka", jossa asia käsitellään tyhjentävästi.

LINE (<**x1,y1**>)-(<**x2,y2**>[,<**väri**>] [,<**B/BF**>]

LINE-käsky piirtää viivan kahden tietyn pisteen väliin. Kirjoita vaikka seuraava esimerkki:

```
10 SCREEN 2
20 LINE(20,10)-(240,180)
30 GOTO 30
```

Ohjelman avulla sait siis piirretyksi viivan, joka alkaa kohdasta 20,10 ja päättyy kohtaan 240,180. Viivan voi värittää lisäämällä riville 20 jonkin luvun 0:n ja 15:n väliltä.

```
20 LINE(20,10)-(240,180),10
```

Nytpä viiva värittyi keltaiseksi.

LINE-käskyllä voi piirtää vaikka suorakulmion, jossa viiva on lävistäjänä (lisää riville 20: ,B) ja jonka voi myös värittää (muuta rivin 20 B muotoon BF), jolloin

```
20 LINE(20,10)-(240,180),10,BF
```

muuttaa viivan keltaiseksi suorakulmioksi.

LOCATE <x,y>

LOCATE-käskyä voidaan käyttää sekä grafiikka- että kirjoitusruuduissa paikantamaan piirtämisen aloituskohta. Kirjoita esimerkiksi seuraava ohjelma:

```
10 SCREEN 0
20 LOCATE 20,10
30 PRINT "Terve"
```

Käytäntö on aivan sama ruudussa 1. Grafiikkaruutujen vastaava toiminto käy selville, kun ohjelmaan ensin lisätään muutama rivi, jotta se näyttäisi seuraavalta:

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 LOCATE 20,10
40 PRINT #1,"Terve"
50 GOTO 50
```

Lisärivit ovat tarpeen, jotta voit tulostaa grafiikkaruutuihin. Käytä sitten ohjelmaa uudelleen.

PAINT (<x,y>)[,<**maalin väri**>][,<**rajausväri**>]

PAINT-käskyn avulla voi tietyllä värillä maalata minkä tahansa muotoiseksi piirretyn grafiikkahahmon. Käskyn tarjoamia mahdollisuuksia tarkastellaan kunnolla kirjan seuraavassa luvussa; olennaisin seikka on toki PAINTin kyky täyttää haluamallasi värillä mikä tahansa ruutuun piirtämäsi kuva, johon on määritetty tietty aloituskohta. Käskyn toimintaa kuvastamaan tarvittiin hieman pitempi ohjelma kuin muuten tähän mennessä, mutta tulos on aika värikäs ja kirjoittamisesta aiheutuvan vaivan arvoinen:

```
10 SCREEN 2
20 LET C=2
30 FOR I=100 TO 1 STEP -10
40 CIRCLE (125,100),I,C
50 CIRCLE (125,100),C
60 LET C=C+1
70 NEXT I
80 GOTO 80
```

Kuten huomaat, ohjelma piirtää ympyröitä, joista viimeisin on aina edeltäjäänsä pienempi, ja aloittaa värittämisen keskipisteestä. Ehkä osaat päätellä ohjelmasta, kuinka se suoriutui tehtävästään; mutta ei kannata murehtia, jos ei siinä onnistu – seuraavan jakson lopussa asian kyllä ymmärtää!

POINT (<x,y>)

POINT-käskyn avulla voit ruudulta määrätä haluamasi värin tietylle pikselille. POINT on todella tarpeellinen vain pitkälle edistyneiden grafiikkaohjelmoijien käyttöön, eikä käskyä selvyyden vuoksi käsitellä nyt tämän enempää.

PSET (<**x,y**>)[,<**väri**>]

Kun PAINT-käsky mahdollisti joko tietyn ryhmän tai muodon värittämisen, PSET-käskyn avulla voit vuorostaan määrittää tietyn yksittäisen pikselin. Voit kokeilla esimerkiksi seuraavalla ohjelmalla:

```
10 SCREEN 2
20 PSET (100,100)
30 GOTO 30
```

jolloin huomaat, että aseman 100,100 pikselin väri on vaihtunut samaksi kuin edustalla on. Jos muutat rivin 20 muotoon

```
20 PSET (100,100),10
```

niin tuo piste muuttuu keltaiseksi. Kuten grafiikkakäskyt ylipäätään; keltainen piste muuttuu laatikoksi, jos ruutu 2 vaihdetaan ruutuun 3.

PRESET (<**x,y,**>),<**taulukon nimi**>[,<**valinta**>]

PRESET on PSETin täydellinen vastakohta siinä mielessä, että kun värejä ei ole määritetty, PRESET värittääkin pikselin (tai pikselit) taustan eikä edustan väriseksi. Kun värit on määritelty, PRESET toimii aivan samoin kuin PSET.

PUT SPRITE <**spritetaso**>[,<,**x,y**>][,<**väri**>] [,<**n**>]

Sprite on grafiikkamerkki, jota voi siirtää nopeasti ruudussa. Spritet ovat tuttuja kaikille Space Invadersin tai muiden videopelien ystäville, sillä juuri spritejen avulla avaruuden valloittajat ja useimmat muutkin asiat piirtyvät ruutuun.

Olethan jo tottunut siihen, että grafiikkaruutu koostuu 256 x 192 pikselistä. Spritet on määritetty ohjelmaan, ja ne voivat olla minkä tahansa muotoisia ja koko voi olla yhdestä pikselistä 32 x 32 pikseliin. Niitä saattaa

olla ohjelmassa kaiken kaikkiaan 256 kappaletta, joista 32 voi olla samanaikaisesti ruudussa yhteensä 32 tasolla – – mikä riittää innokkaimmillekin pelinlaatijoille!

Spriten muoto määritetään SPRITE-muuttujan avulla, ja sen ominaisuudet määritetään PUT SPRITE -käskyllä. Käskyssä mainitaan, mille tasolle (0:sta 31:een) sprite tulee piirtää ruutuun, mihin kohtaan (yleiseen käyttöön tarkoitetut x-koordinaatit −32 – +255 ja y:t −32 – +191) ja tyyppinumero, jonka täytyy olla pienempi kuin 256, jos sprite koostuu 8 x 8 pikselistä, tai pienempi kuin 64, jos se koostuu 32 x 32 pikselistä. Esimerkiksi:

```
10 PUT SPRITE 0,(100,100),7,17
```

tarkoittaa, että tasolla 0 on asemassa 100,100 sinivihreä sprite, jonka tyyppinumero on 17. Oletetaan tietenkin, että sprite 17 on tätä ennen määritetty.

HUOM.! PUT SPRITE -käsky hyödyntää seuraavia asioita: jos x- ja y-koordinaatit on määritetty STEP(x,y)-käskynä eikä erikseen x:nä ja y:nä, niin sprite siirtyy entisestä paikastaan x:n ja y:n verran, eli STEP (100,50) siirtää spriteä 100 pikselin verran oikealle ja 50 pikselin verran alas vanhasta paikastaan eikä asemaan 100,50. Kun y:n koordinaatiksi määritetään 208, kaikki sellaiset tasot, jotka sijaitsevat spriten takana, häviävät kunnes y saa jonkin muun arvon kuin 208. Kun y:n koordinaatiksi määritetään 209, sprite häviää ruudulta.

SCREEN [<**tila**>][,<**spriten koko**>][,<**näppäinääni**>] [,<**kasetin baudinopeus**>][,<**kirjoitin**>]

SCREEN-käskyn tulisi olla jo vähintäänkin tuttu. Sillä on kuitenkin sellaisia ominaisuuksia, joita ei ole vielä käsitelty kirjan alkupuolella. Kuten varsin hyvin tiedät, tilavalintaa käytetään toivotun ruututyypin määritykseen, jonka vaihtoehdot ovat siis seuraavat:

0 40 x 24 tekstitila
1 32 x 24 tekstitila
2 256 x 192 suurresoluutiografiikkatila
3 moniväri-pienresoluutiografiikkatila

Spriten kokovaihtoehdot ovat seuraavat:
0 8 x 8 suurentamaton
1 8 x 8 suurennettu
2 16 x 16 suurentamaton
3 16 x 16 suurennettu

Jos MSX-koneesi näppäinten kilahtelu ei miellytä, vaihda näppäinvalinta 0:ksi. Valinta 1 palauttaa takaisin näppäimen äänen.

Lauserakenteen kaksi viimeistä osaa auttavat kasettinauhurin tai kirjoittimen kanssa mahdollisesti aiheutuvissa ongelmissa. Edellisen avulla voi määrittää kasetille siirtämisen nopeuden joko 1:ksi eli 1 200 baudiksi tai 2:ksi eli 2 400 baudiksi. Jälkimmäistä käytetään, jos kirjoitin ei ole MSX-standardin mukainen, eikä arvo silloin ole nolla.

SPRITE$ (<N>) = <jonolauseke>

SPRITE$ ei ole varsinaisesti käsky vaan muuttuja. Katso 3. luvusta muuttujat ja vakiot.

Spritejen määrittelyä ja niitä täydellisesti hyödyntäviä ohjelmia tarkastellaan kunnolla seuraavassa luvussa. Vielä ei tarvitse tietää muuta kuin se, että spriten määrittämiseksi sille annetaan nimi ja valitaan sen muoto. Homman voi hoitaa monin tavoin, jotka selitetään vielä tarkalleen. Jos olet jo kiinnostunut spriten kokeilusta, voit viettää tovin kirjoittamalla seuraavan ohjelman:

```
10 SCREEN 2,0,0
20 FOR I=1 TO 8
30 READ B$
40 S$=S$+CHR$(VAL("&B"+B$))
```

```
50 NEXT I
60 SPRITE$(0)=S$
70 X%=INT(RND(1)*256):Y%=INT(RND(1)*192)
80 PUT SPRITE 0,(X%,Y%),15,0:BEEP
90 FOR J=1 TO 300:NEXT J
100 GOTO 70
110 DATA 00011000
120 DATA 00111100
130 DATA 01100110
140 DATA 11011011
150 DATA 11011011
160 DATA 01100110
170 DATA 00111100
180 DATA 00011000
```

Huomaathan, kuinka spriten muoto määriytyy lausekkeen sisältämien ykkösten mukaan.

VPEEK (<**video-RAMin osoite**>)

VPEEK-käskyn avulla voit vilkaista, mitä tietokoneen videomuistin tietyssä osassa on. PEEK-kurkistelu ja seuraavaksi käsiteltävä POKE-käsittely kuuluvat vain harjaantuneelle ohjelmoijalle, eivätkä aloittelijan taidot suinkaan riitä niihin. Aiheeseen ei kuitenkaan kannata vielä puuttua.

VPOKE (<**video-RAMin osoite**>,<**tavu**>)

VPOKEn avulla voidaan lisätä tavun verran tietoa suoraan koneen muistin osoitteisiin 0 – 16383. Jos et ole aivan varma siitä mitä olet tekemässä, käskyä **EI** tule kokeilla omin päin – toisin kuin miltei mitä tahansa muita tämän kielen käskyjä! Jos kokeilet tätä ja koneesi muisti romahtaa, ei koneesta silloin ota tolkkua. Virran katkaiseminen ja uudelleen aloittaminen ratkaisee todennäköisesti ongelman; aloittelijalle tätä ei kuitenkaan suositella.

Tässä jaksossa pyrittiin esittelemään yksityiskohtaisesti MSX-BASICin käytännöllisimmät ja niin ollen myös

yleisimmät käskyt. Toivottavasti niiden kokeilu on osoittautunut kiinnostavaksi. Niin vaikuttavia kuin käskyt ovatkin, niiden ominaisuudet pääsevät täyteen mittaansa vasta kun ne yhdistetään ohjelmiksi. Toivottavasti tuo seikka on tullut selväksi kirjamme esimerkkiohjelman avulla. Kuten luvun alussa mainittiin, tarkoituksena oli kuitenkin käyttää mahdollisimman vähän ohjelmia. Toivottavasti nuo tarpeelliset esimerkkimme ovat sitten osoittaneet selvästi sen, kuinka muutaman rivin lisääminen saattaa ällistyttävästi muuttaa ruutuun syntyvää kuvaa.

Nyt ei sitten enää käytetä tarjolla olevia käskyjä sellaisinaan, vaan on aika ruveta käyttämään tietokonetta oman mielen mukaan, ohjelmointitaitojen avulla!

Ohjelmoinnin aloittaminen

MSX-BASICin lukuisten käskyjen ja lauseiden jälkeen on jo aika vilkaista, kuinka niiden avulla laaditaan hyödyllisiä ohjelmia. Kaikkiin BASIC-ohjelmointia käsitteleviin kirjoihin näkyy kuuluvan keskeisesti pyyntö, että lukija laatisi ohjelman tilinsä koron laskemista varten (sehän sivuutettiin jo kirjan alussa) tai satunnaisten lukujen arvaamista tms. varten. Tuosta poiketen olemmekin päättäneet käyttää MSX-tietokoneiden erinomaisia grafiikkaominaisuuksia osoittamaan, kuinka ohjelmia voi käyttää yhä taidokkaampien kuvioiden laadintaan, ja opastamaan ohjelmoinnin rakenteen melkoisesti mielenkiintoisemmin kuin yleensä. Sitten tarkastellaan, kuinka tästä osasta omaksutut ajatukset ovat käytännössä hyödynnettävissä.

Ohjelmien lataus ja tallennus

Ensiksi käsitellään hieman arkipäiväisiä seikkoja – on opittava vielä muutama käsky. Ne on jätetty näin myöhäiseen vaiheeseen, koska niitä tarvitaan ohjelmien lataamiseen ja tallentamiseen. Mitä pitemmiksi

ohjelmat vähitellen muodostuvat, sitä masentavammaksi käy niiden kirjoittaminen paperille jokaisen onnistuneen kokeilun jälkeen ja uudelleenkirjoittaminen kutakin käyttökertaa varten. Jos tietokoneeseen on liitetty kasettinauhuri tai levyasema, seuraavat käskyt ratkaisevat tuon ongelman:

SAVE "<**laitemääritys**>,[<**tiedoston nimi**>]"

SAVE-käskyn avulla BASIC tallentaa kulloinkin koneen muistissa olevan ohjelman tiettyyn laitteeseen, joka on mainittu laitemäärityksessä, kuten CAS tai DISC. Esimerkiksi SAVE "CAS:JON" antaa tiedostolle nimen JON ja tallentaa sen kasetille.

LOAD <"**laitemääritys**:[<**tiedoston nimi**>]" >[,**R**]

Tällä käskyllä BASIC-ohjelma ladataan kasetilta ja samalla poistetaan muistissa mahdollisesti entuudestaan oleva ohjelma. Kun R-vaihtoehto määritetään, ladattu ohjelma alkaa automaattisesti. Jos ohjelman nimeä ei mainita, latautuu ensimmäinen ohjelma joka kasetilta löytyy.

MERGE "[**laitemääritys**:<**tiedoston nimi**>]"

MERGE-käskyn avulla ohjelma kykenee yhdistämään ohjelmia. On tärkeää, että ohjelmissa ei esiinny ristiriitaista rivinumerointia (kuten jos molemmissa ohjelmissa olisi rivi, jonka numero olisi 20), koska vain muistiin lisättävä rivi tallentuu. Käsky yhdistää BASIC-ohjelman kasetilta (jolla ohjelman tulisi olla) tietokoneen muistissa entuudestaan olevaan ohjelmaan. Esimerkiksi MERGE "CAS:JON" yhdistää kasetilta JON-ohjelman tietokoneen muistissa sillä hetkellä olevaan ohjelmaan. Jos ohjelmaa ei mainita nimeltä, yhdistetään ensimmäinen kasetilta löytyvä ohjelma.

BSAVE "<**laitemääritys**>:[<**tiedoston nimi**>]" [,**lähtöosoite**] [,**määräosoite**] [,**toimiosoite**]

Mitä paremmin opit hallitsemaan ohjelmointia, sitä todennäköisemmin siirryt BASIC-ohjelmoinnista yhä mutkikkaampaan konekieliseen ohjelmointiin.

BSAVE-käskyn avulla tallentuu konekielinen ohjelma. Se eroaa SAVEsta, jolla tallennetaan BASIC-ohjelma siten, että se siirtää kasetille minkä hyvänsä muistissa olevan asian, myös datan. BSAVE-käskyn B tarkoittaa "binaaria", koska RAMista suoraan siirtyvä tieto on juuri binaarista dataa. Sen käyttö ei kuitenkaan kuulu tämän kirjan aiheisiin.

BLOAD "<**laitemääritys**>:[<**tiedostonimi**>]"[,**R**] [,<**offset**>]

BLOAD-käsky lataa konekielisen ohjelman suoraan muistiin. Tämäkään käsky ei kuulu kirjamme piiriin. Mainittakoon kuitenkin, että [,**R**] saa ohjelman toimimaan heti kun se on ladattu.

Ohjelmien laatiminen

Kun ohjelmien tallennus ja haku on nyt käsitelty, on aika vilkaista varsinaisten ohjelmien rakennetta. Aluksi perehdytään siihen, kuinka ohjelmat voi jäsentää tehokkaiksi ja säästää samalla suuresti kirjoitusvaivojaan.

FOR . . . NEXT

Lyhyet ohjelmat ovatkin jo tuttuja, esimerkiksi:

```
10 SCREEN 2
20 CIRCLE (125,100),100
30 GOTO 30
```

Tämän avulla piirretään ympyrä, jonka säde on 100 pikseliä ja keskipiste kohdassa 125,100. Entä jos haluaakin piirtää joukon sisäkkäisiä ympyröitä? Yksi keino – jota ei missään tapauksessa kannata käyttää – on kirjoittaa seuraava ohjelma:

```
10 SCREEN 2
20 CIRCLE (125,100),100
30 CIRCLE (125,100),90
40 CIRCLE (125,100),80
50 CIRCLE (125,100),70
60 CIRCLE (125,100),60
70 CIRCLE (125,100),50
80 CIRCLE (125,100),40
90 CIRCLE (125,100),30
100 CIRCLE (125,100),20
110 CIRCLE (125,100),10
120 GOTO 120
```

Tämä ohjelma käskee tietokonetta piirtämään ensin ympyrän jonka säde on 100 pikseliä, sitten toisen jonka säde on 90, ja kolmannen jonka säde on 80, kunnes rivillä 110 piirretään ympyrä jonka säde on 10 pikseliä. Ohjelman viimeisellä rivillä on GOTO-lause, joka on esiintynyt kaikissa tähänastisissa ohjelmissa ja jota käsitellään vielä perusteellisesti. Tässä tapauksessa GOTO ainoastaan käskee konetta palaamaan riville 120 eikä anna sen lopuksi poistua ohjelmasta. Jollei GOTO-lausetta olisi määritelty, ympyrät katoaisivat ruudusta ja tietokone siirtyisi merkkiruututilaan heti ympyröiden piirtämisen jälkeen.

Tuon ohjelman kirjoittaminen oli melko rasittavaa – paljon rasittavampaa kuin sen lukeminen – ja kaiken lisäksi turhaa työtä, koska saman tuloksen saa aikaan asettamalla ohjelmaan FOR . . . NEXT-silmukan. Silmukoita käsitellään vielä melkoisesti, ja oleellisinta on se, että niiden avulla tietokoneen saa tekemään saman tehtävän monta kertaa joko samanlaisesti tai hieman muunnettuna. FOR . . . NEXT-silmukan saa kokeiltua vaikkapa seuraavasti:

```
10 SCREEN 2
20 FOR N=10 TO 100
30 CIRCLE (125,100),N
40 NEXT N
50 GOTO 50
```

Tietokone on tämän avulla piirtänyt joukon ympyröitä, joista ensimmäisen säde on 10 pikseliä ja viimeisen 100, ja keskipisteenä on 125,100. Kuinka se suoriutui tehtävästään? Ensiksi se huomaa rivillä 20 olevan FOR-lausekkeen, joka käskee antamaan N-nimiselle muuttujalle arvon 10. Niinpä se sitten tietääkin riviä 30 suorittaessaan, että sen täytyy piirtää ympyrä, jonka säde on 10 pikseliä. Rivi 40 kertoo tietokoneelle, että sen tulee ottaa uusi N:n arvo. Se palaa riville 20, ja koska määrityksen mukaan N:n arvo on 10:stä 100:aan, se lisää rivillä olevaan arvoon 1:n, ja rivi 30 piirtää ympyrän, jonka säde on 11 pikseliä. Se toistaa suoritusta, kunnes N saa arvon 100, jolloin tietokone lukee rivin 50 ja pysähtyy noudattaen sillä olevaa käskyä.

Alkuperäiseen ongelmaan palataksemme, haluttiin piirtää sarja ympyröitä, joiden säteiden ero oli 10 pikseliä. Se saadaan aikaan, kun muutetaan rivi 20 seuraavanlaiseksi:

```
20 FOR N=10 TO 100 STEP 10
```

Lisäämällä käskyyn STEP tietokone on saatu lisäämään N:n arvoa 1:n sijasta määritetyllä arvolla eli tässä tapauksessa 10:llä. Niinpä tietokone piirtääkin ympyröitä, joiden säde on 10, 20, 30 . . . 100 pikseliä, ja lopputuloksena on juuri sellainen piirros kuin ohjelman alunperin haluttiin tekevän.

Ympyrät eivät kuitenkaan piirry aivan oikealla tavalla. Mallina olleessa pitkästyttävässä ohjelmassahan tietokoneen käskettiin piirtää ympyröitä, joista ensimmäisen säde oli 100 pikseliä ja viimeisen 10. Viimeinen ohjelma tekee ympyrät päinvastaisessa järjestyksessä.

Ongelman voi ratkaista muuttamalla riviä 20 seuraavasti:

```
20 FOR N=100 TO 10 STEP -10
```

Ympyrät piirtyvät tällöin kolmen eivätkä kymmenen rivin avulla, kuten alunperin oli tarpeen. Jos nyt vilkaiset PAINT-käskyn esittelyyn käytettyä ohjelmaa, ymmärrät hyvin, kuinka ympyrät täyttyvät omilla väreillään. Silloin käytettiin seuraavaa ohjelmaa:

```
10 SCREEN 2
20 LET C=2
30 FOR N=100 TO 0 STEP -10
40 CIRCLE (125,100),N,C
50 CIRCLE (125,100),C
60 LET C=C+1
70 NEXT N
80 GOTO 80
```

Ohjelman rakenne on sama kuin edellisessä, paitsi että PAINT-käsky hyödyntää lisäksi FOR... NEXT-silmukkaa. Tässähän maalauskäskystä on tehty muuttuja (C). Väri saa (rivillä 20) aluksi arvon 2, ja joka kerta kun silmukka FOR... NEXT suoritetaan, värin arvo kasvaa 1:llä (rivi 60). Se saadaan aikaan LET-lausekkeella. Ympyrä piirtyy ensin tietyllä värillä ja täyttyy lopulta myös sillä. Palataksemme alkuperäiseen ympyrän piirtämistä koskevaan ongelmaamme lisätään siihen pari riviä:

```
10 SCREEN 2
15 FOR M=60 TO 180 STEP 40
20 FOR N=100 TO 10 STEP -10
30 CIRCLE (M,100),N
40 NEXT N
45 NEXT M
50 GOTO 50
```

Tämä lisää CIRCLE-käskyyn vielä yhden muuttujan ja kehottaa tietokonetta silmukan FOR... NEXT avulla siirtämään ympyröiden keskustan x-koordinaattia 40

pikseliä jokaisen piirtämisen jälkeen. On tärkeää huomata, että silmukat eivät saa mennä ristikkäin – FOR M ... NEXT M sisältyy silmukkaan FOR N ... NEXT N. Silmukoiden pitää aina olla ohjelmissa tällä tavoin, muuten tietokone antaa niistä virheilmoituksen.

Lisätäänpä tämä toinen FOR ... NEXT-silmukka ohjelmaan, joka ensin piirtää ja sitten värittää ympyröitä. Emme olekaan vielä pyytäneet sinua tekemään mitään omatoimista. Yritäpä liittää tuo toinen silmukka, ennen kuin silmäilet seuraavaa ohjelmaa.

```
10 SCREEN 2
20 LET C=2
30 FOR M=60 TO 180 STEP 40
40 FOR N=100 TO 10 STEP -10
50 CIRCLE (M,100),N,C
60 PAINT (M,100),C
70 LET C=C+1
80 NEXT N
90 NEXT M
100 GOTO 100
```

Sinun laatimasi ohjelman pitäisi näyttää suunnilleen tällaiselta. Me numeroimme sen (RENUMber) samalla uudelleen, jotta ohjelmasta ei tulisi liian ahdas. Jos huomasit tehdä saman, niin onneksi olkoon! Jos et tehnyt samoin, ei silti syytä huoleen – kyse on vain harjoituksen puutteesta. Käytetäänpä ohjelmaa, jotta nähtäisiin, mitä se tekee. Tuloksen pitäisi ollakin melko värikäs.

No, mikä meni vikaan? Konehan jatkoi toisen ympyräsarjan puoliväliin, mutta sitten se keskeytti ohjelman ja antoi virheilmoituksen:

```
Illegal function call in 50
```

Vilkaise riviä 50 ja mieti, mikä sen mahtoi aiheuttaa.

Tietokoneelta loppui värit, joilla täyttää ympyröitä.

Värejä voi numeroida vain 0:sta 15:een. Koska ohjelman alussa konetta pyydettiin aloittamaan värillä 2, se neljätoista ympyrää myöhemmin päätyi väriin 15. Kun päästään riville 70, värin arvoksi tulee 16. Niinpä kun ohjelma palaa jälleen riville 15, koneen tulisikin piirtää ympyrä sellaisella värillä, jota se ei kuitenkaan tunne, joten tuloksena on virheilmoitus.

Kuinka sitten ohjelma saadaan toteuttamaan ympyrä-sarja loppuun asti.

IF...THEN

Sekä virheen syntyminen että ohjelman "romahtaminen" aiheutuivat siitä, että annoimme värille arvon, joka ei mahtunut sallittuihin rajoihin. Niinpä pyysimme tietokonetta tekemään jotain sille käsittämätöntä – piirtämään ympyrän värillä 16, jota ei ole olemassakaan. Ratkaisu onkin se, että estetään värinumerointia ylittämästä tuota rajaa. Se tehdään tarkistuksen avulla ja antamalla tietokoneelle toimintaohje rajatapausta varten. Seuraavan rivin lisääminen saakin aikaan juuri tuon:

```
75 IF C=16 THEN C=2
```

Tämä rivi tarkistaa, onko C:n arvo 16, ja vaihtaa myönteisessä tapauksessa arvon 2:ksi. Jos arvo ei ole 16, ohjelma etenee riville 80.

Jos kokeilet ohjelmaa uudelleen tämän rivin lisättyäsi, kaikki toimii vihdoin suunnitelmien mukaan. Ohjelma toimii samoin kuin edellisellä kerralla. Ainoa ero on siinä, että rivin 70 antaessa C:lle arvon 16 rivi 75 tulkitsee sen ja vaihtaa arvoksi jälleen 2, mikä sitten sallii toiminnon alkamisen uudelleen. Sama seikka toistuu joka kerta kun C saa arvon 16, kunnes kaikki ympyrät on piirretty ja täytetty.

Tällä kertaa käytimme IF...THEN-lausetta päästäksemme pinteestä, johon olimme joutuneet. IFiä voi

käyttää monenlaisissa tapauksissa, mutta käyttötapa riippuu todellakin ohjelman laatijan mielikuvituksesta. Tietokoneen voi antaa toimia kahden vaihtoehdon mukaan, jos (IF) jotain tapahtuu.

IF...THEN...ELSE

Ympyrän piirtämisessä käytettyyn ohjelmaan palatakseemme, kiinnostuksen aiheena oli ainoastaan se, oliko C yhtäsuuri kuin 16. Myönteisessä tapauksessa sen arvon tulisi muuttua 2:ksi. Jos ehto täyttyy eli on "tosi", rivi toteutetaan. Jos se on "epätosi" (eli C:n arvo on muuta kuin 16), siirrytään riville 80. Esimerkkitapausta suuremmissa ohjelmissa on "epätosi"-tilanteissa varmaankin näppärintä siirtyä jollekin muulle riville kuin seuraavalle, jolloin IF...THEN...ELSE on mahdollinen. Esimerkiksi:

```
IF C=16 THEN C=2 ELSE 80
```

on toinen tapa kirjoittaa rivi 75. Rivi saa aikaan saman lopputuloksen, mutta se ei annakaan tietokoneen siirtyä riville 80 pelkästään siksi, että se on ohjelman järjestyksessä seuraavana, vaan sen vuoksi, että sinä käskit toimia siten, jos tilanne on epätosi.

HUOM.! Rivinumerot ja ohjelmointiohjeet voi kirjoittaa aivan yhtä hyvin THENin tai ELSEn jälkeen, kuten osoitimme edellä rivin 75 muutoksissa.

IF...GOTO

GOTO-lause onkin jo näkynyt kaikissa ohjelmissa, jotka olet pyynnöstämme kirjoittanut. GOTO on pelkästään varmistanut ohjelman säilymisen ruudussa siten, että viimeinen rivi muodostaa oman silmukkansa, joka ei katkea ennen kuin käyttäjä painaa näppäimiä CONTROL ja STOP. GOTO siis yksinkertaisesti saa ohjelman "hyppäämään" käskyssä mainitulle riville.

GOTOn toiminta selviää, kun ensin kirjoittaa jäljempänä olevan ohjelman ja sitten käyttää sitä. Ennen kuin luet selityksen tapahtumien kulusta, perehdy ohjelmaan ja yritä päätellä, miten kaikki kävi. Se on tähänastisista ohjelmista mutkikkain, joten ei kannata huolestua, jos se tuntuu hieman vaikealta. Ohjelma on periaatteessa sama kuin ympyrän piirtämiseen käytetty. Siihen on ainoastaan lisätty muutama rivi, joiden avulla tietokone kysyy, haluaako käyttäjä sen toimivan. Ohjelmaa käytettäessä tulee vastata "kyllä" tai "ei", mutta yritäpä vastata vaikkapa "miksi?" tai jotain muuta – kuinka käykään?

```
10 CLS
20 PRINT "Piirränkö ympyröitä?"
30 PRINT "Vastaa kyllä tai ei"
40 INPUT A$
50 IF A$="kyllä" GOTO 90
60 IF A$="ei" GOTO 170
70 PRINT "Anteeksi, en ymmärrä. Yritä uu
destaan."
80 GOTO 40
90 SCREEN 2
100 LET C=2
110 FOR N=100 TO 10 STEP -10
120 CIRCLE (125,100),N,C
130 CIRCLE (125,100),C
140 LET C=C+1
150 NEXT N
160 GOTO 160
170 END
```

Vilkaistaanpa rivi riviltä, mitä ohjelma tekee.

Rivi 10 tyhjentää näytön.

Rivi 20 käskee tietokoneen kirjoittaa ruutuun kysymyksen: "Haluatko, että piirrän ympyröitä?"

Rivi 30 pyytää sinua vastaamaan joko "kyllä" tai "ei".

Rivi 40 saa tietokoneen odottamaan sinun antamaasi

sanaa. Se ilmoittaa muuttujan A$, johon vastauksesi sijoitetaan ja joka saa tietokoneen kirjoittamaan näyttöön kysymysmerkin, jotta tietäisit koneen odottavan sinulta jonkinlaista vastausta.

Rivi 50 kertoo tietokoneelle, että jos vastauksesi on "kyllä", tulee siirtyä riville 90 ja jättää rivit 60, 70 ja 80 huomiotta. Sitten se vie läpi meille jo tutuksi tulleen ympyränpiirto-ohjelman. Jos sitten vastaat "ei", rivin 50 GOTO-lause ei toteudu vaan tietokone ottaa käsiteltäväkseen rivin 60.

Rivi 60 ilmoittaa tietokoneelle, että jos vastaus on "ei", tulee siirtyä riville 170, jossa END-lause aiheuttaa ohjelman päättymisen ja OK-valmiusilmoituksen palauttamisen ruutuun.

Rivi 70 kehottaa tietokonetta kirjoittamaan ruutuun: "Anteeksi, en ymmärrä. Yritä uudestaan."; jos vastaat riville 40 jotain muuta kuin "kyllä" tai "ei".

Rivi 80 kehottaa tietokonetta palaamaan taas riville 40, jossa sinun pitää kirjoittaa sellainen vastaus, jonka kone ymmärtää.

Toivottavasti selityksemme auttaa ymmärtämään IF... GOTO-lauseen joustavuutta ja säästöä, joka saadaan, kun ei tarvitse kirjoittaa samaa ohjelman osaa yhä uudelleen; toimitaan kutakuinkin samalla tavalla kuin FOR ... NEXT-silmukoita käytettäessä.

IF... GOTO ... ELSE

IF... GOTO ... ELSE-lause toimii aivan samoin kuin IF... THEN ... ELSE, paitsi että IFin jälkeinen GOTO rajoittaa toiminnon hypyksi jollekin muulle riville.

Lopuksi näitä ehdontarkistuksia voi mainiosti yhdistellä samalle käskyriville, esimerkiksi:

IF... THEN IF... AND ... GOTO ... ELSE

tai jopa

IF... OR IF... AND IF... THEN ... ELSE

Voit laatia ohjelman niin mutkikkaaksi kuin haluat. Emme esittele tässä vaiheessa ohjelmaa, jossa olisi tuonkaltaisia lauseita. Todettakoon vain, että ohjelmointitaitojen karttuessa kykenet kirjoittamaan yhä mutkikkaampia ohjelmia ja alat pian keksiä niitä aivan itse!

Huomasit varmaankin, että viimeisimpään ohjelmaan oli sujautettu muutamia sinulle vieraita käskyjä, nimittäin PRINT, INPUT ja END. Kuten kappaleen alussa mainittiin, eräät ohjeet ovat mielekkäitä vasta ohjelmaan liitettyinä, ja nämä käskyt ovat sellaisia, kuten myös FOR ja IF. On aiheellista kertoa lisää noista lauseista, kun olet tutustunut ohjelman laadinnan perusteisiin.

Ohjelmalauseet

Tämän jakson lauseilla ei ole varsinaista rakennetta. Ne sisältävät lähinnä tietoainesta, jota lause tarvitsee tehtävänsä suorittamisessa.

DATA <sisällysluettelo>

Kaikissa tähän mennessä esitellyissä ohjelmissa tietokonetta on pyydetty prosessoimaan asioita, jotka se tietää jo entuudestaan, kuten värin tai ympyrän halkaisijan muutoksia. Mitä sitten pitäisi tehdä, kun haluaa syöttää tietokoneeseen lisätietoja? Kuinka esimerkiksi saamme kerrotuksi sille kilometrimäärän, jonka auto kulkee tietyllä polttoainemäärällä, kun halutaan saada selville keskikulutus? DATA-lauseen käyttö on yksi mahdollisista ratkaisuista. DATA-lause ei varsinaisesti tee mitään, se ainoastaan kertoo tietokoneelle, että lauseen numerot tai sanat ovat dataa, joka on

saatavissa READ-käskyn avulla. (Siihen käskyyn tutustumme myös aikanaan.) Esimerkkejä DATA-lauseista:

```
DATA 1,2,3,4,5,6,7,8,9,10
DATA tammikuu, helmikuu, maaliskuu, huhtikuu
```

DATA-lause voi sisältää niin monta sanaa tai numeroa kuin niitä pilkuilla erotettuina riville mahtuu, ja ohjelmaan voi liittää vaikka kuinka monta DATA-lausetta. Dataa luetaan kuin pitkää luetteloa, joten on yhdentekevää, mihin kohtaan ohjelmaa DATA-rivit sijoitetaan, kunhan ne vain ovat oikeassa järjestyksessä. Kaikki numerotyypit ovat käytettävissä lukuun ottamatta laskutoimituksia, kuten 2 + 3. Jos käytetään vakioista koostuvia merkkijonoja, joihin sisältyy pilkkuja, pisteitä tai välejä, kuten päivämäärissä, vakiot täytyy sijoittaa lainausmerkkeihin (" "). Merkkijonoja muodostuu esimerkiksi nimestä, puhelinnumerosta tai sanoista.

DATA-lauseeseen sisällytetyn datan täytyy olla sellaisessa muodossa, että READ-lause kykenee lukemaan sen.

DIM <luettelo indeksimuuttujista>

DIM-lauseen myötä käsiteltäviksi joutuvat taulukot, sillä sen avulla määritetään muistissa olevan taulukon koko. Taulukko on erikoinen muuttuja, jonka avulla on helppo tallentaa ja ottaa käsiteltäviksi useita läheisesti yhteen kuuluvia muuttujia.

Yksinkertaisin taulukkotyyppi koostuu muuttujajoukosta, joka on koottu yhden ainoan taulukkonimikkeen alle, esimerkiksi taulukkoon A. Ohjelman alussa täytyy määrittää A:han tallennettavien muuttujien määrä. Määritystä kutsutaan taulukon DIMensioinniksi eli ulottuvuuden määrittämiseksi. A voidaan määrittää koostumaan kymmenestä muuttujasta käyttämällä käskyä DIM A(10).

Jokainen muuttuja saa taulukossa oman tunnusnumeronsa ja on saatavissa taulukosta nimen ja tunnusnumeron perusteella. Esimerkiksi A(3) on taulukon kolmas ja A(7) seitsemäs muuttuja.

Taulukoissa voi numeroita käyttää muuttujien niminä, jos on hankalaa kehittää uusia muuttujanimiä. Jos halutaan tallentaa ja tulostaa kymmenen ensimmäisen luvun neliö, täytyy kirjoittaa kymmenen lausetta: ensin LET A=1*1, sitten LET B=2*2 jne. Se olisi kuitenkin äärimmäisen epäkäytännöllistä. Helpoimmin sen saa tehtyä käyttämällä FOR ... NEXT-silmukkaa ja taulukkoa, kuten seuraavassa ohjelmassa:

```
10 DIM A(10)
20 FOR I=1 TO 10
30 A(I)=I*I
40 PRINT I,A(I)
50 NEXT I
```

Ohjelmassahan annetaan aluksi I=1 ja ensimmäinen lause asettaa A(1)=1*1. Toinen lause asettaa A(2)= 2*2 ja niin edelleen 10:een asti. Taulukkomuuttujia voi sitten käsitellä mielensä mukaan; ne voi vaikka jakaa 10:llä seuraavan rivin avulla:

```
35 A(I)=A(I)/10
```

END

END-lause kertoo tietokoneelle pelkästään ohjelman loppuun pääsystä; se käskee lopettamaan ohjelman käytön, sulkemaan ohjelman takia luodut tiedostot ja palaamaan käskytasolle, mikä saa OK-valmiuden tulemaan näkyviin ruutuun.

ERROR <kokonaisluku>

MSX-BASICiin kuuluu monia virheilmoituksia, joista muutamiin olemmekin jo tutustuneet. Ne on kaikki luetteloitu liitteeseen B. Jokaiseen virheilmoitukseen

liittyy luku 0 – 255. Liitteestä huomaa, että luvut 0 – 60 on jo varattu, joten sinulle jää omia ilmoituksiasi varten luvut 61 – 255. On parasta aloittaa ilmoitusten numerointi 255:stä ja jatkaa siitä alaspäin. Näin virheilmoituksesi säilyvät yhtenevinä niiden laajennusten kanssa, joita MSX-BASICiin mahdollisesti tulee. Seuraava on esimerkki siitä, kuinka voit lisätä ohjelmaan oman virhekoodisi:

```
10 ON ERROR GOTO 1000
100 IF A$="Ei" THEN ERROR 250
1000 IF ERR=250 THEN PRINT "Oletko ehdot
toman varma?"
```

FOR

Katso sivu 56.

GOSUB <rivinumero>

Olemme jo todenneet, kuinka GOTO-lause saa ohjelmassa aikaan hyppäyksen tietylle riville. GOSUB on eräänlainen kehittynyt versio GOTOsta, sillä sen avulla tietokone alkaa käyttää aliohjelmaa. Aliohjelma on erikseen tiettyä tehtävää varten laadittu varsinaisen ohjelman osa, ja sitä voidaan käyttää monta kertaa pääohjelman aikana. Jos esimerkiksi ympyränpiirto-ohjelmaan olisi lisätty kysymys "Piirränkö neliöitä?", sitä olisi voitu käyttää reaktiona, mikäli kysymykseen "Piirränkö ympyröitä?" olisi vastattu "Ei". Jos ohjelmaan vielä lisäisi yhden mahdollisuuden, "Soitanko sävelmän?", huomaisi, että ohjelma alkaisi käydä melko hankalaksi. Jos yhteenkin kysymyksistä vastattaisiin "Kyllä", olisi yksinkertaisuuden vuoksi mielekästä, että tietokone alkaisi suorittaa erillistä GOSUB-lausetta, jonka aliohjelma joko piirtäisi ympyröitä tai neliöitä tai soittaisi sävelmää.

Aliohjelman loppuun tulee lisätä RETURN-lause. Se ilmoittaa tietokoneelle, että tulee palata aliohjelman aloittaneen GOSUB-lauseen jälkeiselle riville.

Kun osa riveistä jätetään kirjoittamatta, erityislaaja ympyröidenpiirto-ohjelma näyttäisi aliohjelmineen seuraavanlaiselta:

```
10 "PIIRRÄNKÖ YMPYRÖITÄ"
20 IF "kyllä" GOTO 100
30 "PIIRRÄNKÖ NELIÖITÄ"
40 IF "kyllä" GOTO 120
50 "SOITANKO SÄVELMÄN"
60 IF "kyllä" GOTO 140
70 IF "ei" GOTO 500
80 PRINT "Anteeksi. En ymmärrä. Yritä uu
destaan."
90 GOTO 10
100 GOSUB 200
110 GOTO 10
120 GOSUB 300
130 GOTO 10
140 GOSUB 400
150 GOTO 10
200 REM Aliohjelma "Ympyröille"
299 RETURN
300 REM Aliohjelma "Neliöille"
399 RETURN
400 REM Aliohjelma "Sävelmälle"
499 RETURN
500 END
```

REM selvitetään sivulla 76.

HUOM.! Kannattaa aina sijoittaa aliohjelmat varsinaisen ohjelman loppuun ja ottaa ne mukaan GOTO-käskyllä, koska silloin välttyy niiden tahattomalta mukaantulolta tavallisen ohjelman keskelle. Ohjelmaa ei kuitenkaan saa missään tapauksessa jättää päättymään GOSUB-aliohjelmaan, jolloin ruutuun tulee virheilmoitus "RETURN without GOSUB", kun RETURN-käskyä noudattava ohjelma ei löydä paikkaa johon palata. Esimerkissämme käytettiin GOTO 10:tä estämään ohjelmaa etenemästä riville 200.

GOTO <rivinumero>

Katso sivu 62.

IF <lauseke> THEN <lauseke> ELSE <lauseke>

Katso sivu 61.

INPUT [I "<valmius>";] <muuttujaluettelo>

Jo aikaisemmin sinulta kysyttiin, haluatko saada piirtymään ympyröitä. Toivottavasti silloin huomasit, mihin INPUT-lausetta käytetään. Se kertoo tietokoneelle, että jokin tieto täytyy saada näppäimistön kautta ennen kuin ohjelma voi jatkua (esimerkissämme se oli "kyllä" tai "ei"). Niinpä tietokone esittää näyttöruudussa kysymysmerkin osoittaakseen sinulle, että se odottaa vastaustasi, ja kysymysmerkkiä edeltää valmiusosoitus, jos se on ohjelmoitaessa kirjoitettu lauseeseen. Ohjelmamme osoitti, kuinka valmiusosoitusta käytetään INPUTissa. Valmius sijoitettiin muuttujaan AS ja tietokoneelle ilmoitettiin, että vastauksen tuli olla joko "kyllä" tai "ei". Muussa tapauksessa kone ilmoittaisi, että se ei ymmärtänyt vastausta, ja pyytäisi aloittamaan alusta. Kun INPUT-lause pyytää antamaan tietoja, vastauksen täytyy aina olla koneen edellyttämässä muodossa. Muuten kone antaa virheilmoituksia, kunnes sen saama tieto on hyväksyttävissä.

Seuraavaksi käsittelemme väärän syöttötiedon aiheuttamat virheilmoitukset.

Vääräntyyppiseen dataan, esimerkiksi numeron tilalle tulleeseen kirjaimeen, kone vastaa kirjoittamalla:

```
Redo from start
```

eli "aloitetaanko uudelleen alusta".

Joskus tietokone saa liian monta syöttötietoa, esimerkiksi 1, 2 ja 3, kun pitäisi saada vain kaksi lukua:

Extra ignored

eli "jätetäänkö ylimääräinen huomiotta".

Ja mikäli INPUTiin annetaan liian vähän tietoja, tuloksena näkyy:

??

ja tietokone jää odottamaan lisätietoja.

INPUTista voi poistua painamalla näppäimiä CONTROL ja STOP samanaikaisesti. CONT-näppäimen painaminen saa INPUT-lausekkeen palaamaan.

LINE INPUT ["<**valmius**>;]<**merkkimuuttuja**>

LINE INPUT toimii samalla tavoin kuin INPUT, paitsi että sen avulla voi ottaa käyttöön koko rivillisen merkkejä (256 kappaletta) merkkimuuttujiksi käyttämättä lainausmerkkejä. Ruutuun ei ilmesty lainkaan kysymysmerkkiä, jollet määritä sitä valmiuteen. Kaikki valmiuden jälkeen kirjoittamasi kuuluu sitten merkkimuuttujaan. LINE INPUT saadaan poistetuksi ja palautetuksi samoin kuin INPUT.

[LET] <**muuttuja**> = <**lauseke**>

LET-lause asettaa lausekkeelle muuttujan antaman arvon, esimerkiksi LET C = 2, LET A$ = "kyllä" jne. Huomaathan, että LETin kirjoittaminen on vapaaehtoista – jo pelkkä yhtäläisyysmerkki riittää, joten C = 2 on aivan yhtä pätevä ilmaus kuin A$ = "kyllä".

LPRINT [<**luetellut lausekkeet**>]
LPRINT USING <**jonolauseke**>;<**luetellut lausekkeet**>

Näitä käytetään tietojen tulostamiseen rivikirjoittimilla. Katso käyttöön liittyviä tietoja kohdista PRINT ja PRINT USING.

MID$ (<jonolauseke 1>,n[,m]) = <jonolauseke 2>

MID$-lauseen avulla voi vaihtaa lausekkeeseen osia toisesta lausekkeesta. Lausekkeen 1 merkeistä vaihtuvat kaikki n:ää edeltäviä lukuun ottamatta lausekkeen 2 merkkeihin. "m":n avulla voidaan määrittää lausekkeesta 2 otettavien korvaavien merkkien määrä. Ei ole mahdollista korvata useampia merkkejä kuin lausekkeessa 1 alunperin on. Esimerkiksi:

```
10 LET A$="Auringonnousu"
20 LET B$="laskupas"
30 MID$(A$,9,5)=B$
40 PRINT A$
```

muuttaa A$:n "Auringonnoususta" "Auringonlaskuksi".

NEXT <muuttuja>

Katso sivu 56.

ON ERROR GOTO

ON ERROR GOTO -lauseen käyttö selvisi jo ERROR-lauseeseen tutustuttaessa. ON ERROR GOTO kertoo tietokoneelle, mitä tulee tehdä virheen sattuessa. Jos et ole ohjelmassa ilmoittanut tietokoneelle, mitä tehdä, tai olet kirjoittanut "ON ERROR GOTO 0", tietokone keskeyttää ohjelman suorittamisen ja tulostaa ruutuun virheilmoituksen. Siten ohjelmoija voi valvoa kaikkia ohjelmaansa mahdollisesti ilmestyviä virheitä. Hän voi jopa selvittää ne virheet, jotka olisivat tavallisesti pysäyttäneet koko ohjelman. Parasta on kirjoittaa tämä lause heti ohjelman alkuun.

ON <lauseke> GOTO <rivinumeroluettelo>

ON ERROR GOTO on vain yksi muoto ON . . . GOTO-lauseesta. Sitä voi käyttää moneen tarkoitukseen, kun haluaa asettaa jonkin muunkin GOTOn, joka riippuu valinnan mukaisesta lausekkeesta. Esimerkiksi jos lau-

sekkeen arvo on 3, ohjelmassa hypätään riville, joka on kolmantena GOTOn jälkeisessä luettelossa.

Jos lausekkeen arvo on 0 tai suurempi kuin listalla mainitut, mutta ei suurempi kuin 255, GOTOsta ei piitata, vaan suoritetaan seuraavana oleva lauseke.

Ainekset kirjoittuvat asemiin sen mukaan kuin ne on luettelossa eroteltu toisistaan välimerkeillä.

Esimerkiksi:

, Kirjoitettavat rivit jaetaan 14 merkin mittaisiin vyöhykkeisiin. Pilkun erottama tietoaines kirjoittuu ensimmäisen mahdollisen vyöhykkeen alkuun.
; Puolipiste liittää jälkeensä tulevan aineksen heti edellisen perään. Yhden tai usean välilyönnin jättäminen ainesten väliin saa aikaan saman lopputuloksen.

Jos pilkkua tai puolipistettä käytetään <rivinumeroluettelon> lopussa, seuraavaksi tuleva PRINT-lause kirjoittuu samalle riville, ja väli jää sen mukaiseksi. Jos pilkkua tai puolipistettä ei käytetä, seuraava PRINT-lause alkaa tulostua vasta seuraavalle riville.

Tulostuvien lukujen jälkeen jää aina tyhjä väli. Positiivisiä lukuja edeltää myös väli, negatiivisia taas miinusmerkki (—).

Jottei tarvitsisi kirjoittaa PRINTiä yhä uudelleen, se voidaan korvata kysymysmerkillä, sillä PRINT-lauseessa ne tarkoittavat MSX-BASICin mukaan samaa asiaa.

Jotta nämä PRINT-lauseen ominaisuudet tulisivat selviksi ja jotta tietokonekirjojen hyvät perinteet eivät unohtuisi, voisit kokeilla seuraavaa:

ON <lauseke> GOSUB <rivinumeroluettelo>

ON ... GOSUB toimii melkein samalla tavalla kuin ON ... GOTO. Kukin luettelon numeroista on oman aliohjelmansa ensimmäinen rivi.

POKE <muistiosoite>,<kokonaisluku>

POKEa käytetään melkein samalla tavalla kuin VPOKEa, jota käsiteltiin grafiikkakäskyjen yhteydessä, ja sitä tulisi käyttää yhtä varovasti. Sen avulla voi antaa tietylle muistipaikalle jonkin arvon.

<muistiosoite> on muistissa oleva paikka, jota käsitellään. Sen arvo on −32768 – +65535. Jos arvo on negatiivinen, se tulkitaan vähennykseksi 65536:sta, joten −1 = 65535.

<kokonaisluku> on käsiteltävä tieto eli data. Sen arvo voi olla 0 – 255. Esittelyjaksosta kävikin jo ilmi, että RAMien enimmäismäärä on 64K (64K = 64 x 1024 = 65536 tavua).

PRINT ["<lausekeluettelo>"]

PRINT-lausetta käytetään tiedon tulostamiseen ruutuun, kuten ympyränpiirto-ohjelmamme lopullinen versio monesti osoitti. Jos PRINTiä käytetään ilman että sitä seuraa lausekeluettelo, tulostuu tyhjä rivi. Jos siihen sisällytetään jokin lauseke, se tulostuu. Esimerkiksi PRINT "Terve" saa sanan "Terve" ilmestymään kuvaruutuun.

```
10 CLS
20 PRINT "Terve"
30 PRINT "Terve","Terve"
40 PRINT "Mitä";" täällä"
50 PRINT "oikein"
60 PRINT "on"
70 PRINT "tekeillä"
80 PRINT "?"
```

Jos yrität käyttää tätä ohjelmaa jommallakummalla grafiikkaruudulla, ei tapahdu yhtään mitään. Ongelman voi ratkaista seuraavanlaisella ohjelmalla (mikä kävikin ilmi jo LOCATEa tarkasteltaessa):

```
10 OPEN "GRP:" FOR OUTPUT AS #1
20 SCREEN 2
30 PRINT #1,"Mit haluatkaan tietokoneen
kirjoittavan"
40 GOTO 40
```

PRINT USING <jonolauseke>;<lausekeluettelo>

Helppokäyttöiseen PRINT-lauseeseen verrattuna PRINT USING on melko vaativa. Turhan sekaannuksen välttämiseksi sen käsittely on lykätty tuonnemmaksi.

READ <muuttujaluettelo>

READ esiteltiin jo DATA-laueen yhteydessä. Silloin todettiin, että noita kahta käytetään aina toistensa yhteydessä. Vilkaisepa vielä kerran, mitä DATAsta sanottiin, ja kokeile seuraavaa ohjelmaa:

```
10 PRINT "DATA-lauseessa";
20 PRINT " olevien lukujen summa";
30 PRINT " on";
40 READ A,B,C,D,E,F
50 G=A+B+C+D+E+F
60 PRINT G
70 DATA 1,2,3
80 DATA 4,5,6
```

Ohjelmasta on tarkoituksellisesti tehty hieman pitempi kuin olisi välttämätöntä, koska haluamme osoittaa PRINT- ja DATA-lauseista muutamia seikkoja, joista oli jo puhe, kuten puolipisteen käyttöä, lukujen väliin tulostuksessa jääviä tyhjiä tiloja ja sen, että data luetaan peräjälkeen – onpa se missä tahansa. Jos et huomannut näitä asioita, et ole tainnut olla aivan tarkkana! Huomasithan myös, että vaikka data voi esiintyä missä tahansa ohjelman kohdassa, se on kuitenkin

sijoitettu loppuun, koska silloin se ei sekoitu muuhun ohjelmaan.

READ-lause ei pelkästään kykene hakemaan lukuisia DATA-lauseita, vaan samaa dataa voi hakea monella READ-lauseella. Kokeilemme asiaa aikanaan RESTORE-käskyn yhteydessä. Säilytä siis ohjelma!

REM <huomautus>

REMiä käytetään pelkästään siihen, että voit halutessasi liittää huomautuksia, jotka kertovat kunkin ohjelman osan merkityksen. Käytimme REMiä erityislaajennetussa ympyränpiirto-ohjelmassamme kertomaan alioohjelmien toiminnasta. Tietokone ei suorita REM-lauseita, kuten huomasit ohjelman rakenteesta, vaan niitä voidaan käyttää GOSUBiin saapumisen tai GOTOsta poistumisen osoitukseen. Huomautuksen voi lisätä rivin loppuun joko merkkiä ' tai ; REMiä käyttäen, mutta kannattaa käyttää rivi pelkästään tuohon huomautukseen, jotta ohjelman rakenne säilyisi helppolukuisena.

RESTORE [<rivinumero>]

READ-lauseen yhteydessä jo todettiin, että data tulee tallentaa, jotta sitä voitaisiin käyttää uudelleen. Jos olet säilyttänyt kirjoittamasi ohjelman READissä, muuta se seuraavan mallin mukaiseksi. Silloin ymmärrät, mistä on kyse.

```
10 PRINT "DATA-lauseessa";
20 PRINT " olevien lukujen summa";
30 PRINT " on";
40 READ A,B,C,D,E,F
50 G=A+B+C+D+E+F
60 PRINT G
70 RESTORE
80 PRINT "Lukujen tulo on";
90 READ A,B,C,D,E,F
100 H=A*B*C*D*E*F
110 PRINT H
```

```
120 DATA 1,2,3
130 DATA 4,5,6
```

Jos yrität käyttää ohjelmaa ilman RESTORE-lausetta, saat aikaiseksi virheilmoituksen "Out of DATA in 90". RESTORE-lauseessa on mahdollista ilmoittaa myös rivinumero, jolta rivien lukeminen aloitetaan. RESTORE 120 aloittaisi myös ohjelman, sen sijaan RESTORE 130 ei voisi aloittaa.

HUOM.! Huomasithan, että ohjelmamme rivit 70 ja 90 olivat tarpeettomia. Kun A, B, C, D, E ja F ovat jo saaneet arvonsa, ei ole lainkaan tarpeen lukea dataa uudelleen (rivi 90). Ohjelma on kuitenkin laadittu tällaiseksi, jotta RESTOREn käyttö tulisi selväksi mahdollisimman helposti. Kirjan loppupuolella RESTORE löytyy toki hyvinkin mutkikkaista ohjelmista, joissa esimerkkimme käyttötapa on tarpeen.

RESUME

RESUME-lause ilmoittaa tietokoneelle, mitä sen tulee tehdä virheen korjauksen jälkeen. Mikä tahansa neljästä mallista on käytettävissä sen mukaan, mistä haluat ohjelman jatkavan suoritustaan.

RESUME tai **RESUME 0**	palaa suorittamaan lausetta, joka aiheutti virheen.
RESUME NEXT	alkaa suorittaa virheen jälkeen ensimmäisenä olevaa lausetta.
RESUME <rivinumero>	alkaa suorittaa määrätyltä riviltä.
RETURN	Katso GOSUB.

STOP

STOP-lausetta voi käyttää missä kohtaa tahansa suori-

tuksen päättämiseen ja komentotasolle palauttamiseen. Kun STOP tulee esiin ohjelmassa, viesti on:

```
Break in xxxx
```

jossa xxxx on STOP-lauseen rivinumero. Suoritukseen voi palata antamalla CONT-käskyn. Voit kokeilla asian lisäämällä:

```
75 STOP
```

ohjelmaan, joka on luetteloitu RESTOREen. Toisin kuin END-lause, STOP ei sulje tiedostoa.

SWAP <muuttuja>,<muuttuja>

SWAP vaihtaa kahden muuttujan arvot, joten kun A = 5 ja B = 10, niin SWAP A,B muuttaa ne A = 10 ja B = 5. SWAPilla ovat vaihdettavissa kaikentyyppiset muuttujat, kunhan ne ovat keskenään samaa tyyppiä.

Yhteenveto

Onneksi olkoon! Olet selviytynyt kirjan vaikeimmasta luvusta! Olet oppinut yli seitsemänkymmentä MSX-BASICin tärkeimmistä käskyistä ja lauseista. Samalla olet hankkinut perustiedot niiden yhdistelystä ohjelmiksi. Toivottavasti esityksemme ohjelmoinnin peruskäsitteistä oli sekä mielenkiintoinen että visuaalisesti kiehtova. Viime osan tarkoituksena on toimia myös lähdeteoksena, kun perehdyt jäljellä oleviin lukuihin, joissa käsitellään yhä kehittyneempiä ohjelmia.

Seuraavassa luvussa rakennetaan pohjalle, jonka olet juuri hankkinut, esitellään entistä monimutkaisempia rakenteita ja kuvataan loput käskyt, lauseet ja toiminnot, jotka katsoimme soveliaiksi jättää siihen. Perehdymme yhä tarkemmin datatyyppeihin ja MSX-BASICin käyttökelpoisimpiin matemaattisiin funktioihin. Kun olet oppinut hallitsemaan ne kaikki, tietosi tieto-

koneen laitteista ja kielestä riittää ohjelmanlaadintaan, jolla kykenet miltei mihin tahansa kuviteltavissa olevaan suoritukseen. Seuraaviin jaksoihin on sisällytetty ohjelman runkoja, joita voidaan soveltaa koearvosanojen suhteuttamisessa, peleissä ja musiikissa ja joita voit soveltaa omiin tarkoituksiisi. Kaksi viimeistä lukua käsittelevätkin sitten grafiikan ja musiikin makrokielten kiehtovaa maailmaa.

Numeroilla työskentely

Tietokoneen täytyy siis saada dataa, ennen kuin se kykenee tekemään mitään hyödyllistä. Tässä luvussa tarkastelemme datatyyppejä, joita tietokone kykenee käyttämään, ja niiden soveltuvuuksia.

MSX-BASIC tukee useita datatyyppejä. Niitä on kaikkiaan seitsemän: integer, kiintoluku, liukuluku, heksadesimaali, oktaali, binaari ja merkki. Kukin tyyppi käsitellään vuorollaan.

Data voidaan sijoittaa ohjelmaan kahdella tavalla, **vakiona** tai **muuttujana.** Vakion arvo ei muutu ohjelman edetessä. Muuttujan arvo taas voi vaihtua ohjelmaa suoritettaessa – eli sen arvoa voi muuttaa haluamallaan tavalla.

Vakiot

Kuinka nuo erityyppiset vakiot sitten määritetään vakioiksi? Ensiksi täytyy esitellä jokainen datatyyppi.

Integer tarkoittaa itse asiassa kokonaislukua eli lukua, jossa ei ole desimaalipistettä. Integerluvun arvo voi

olla −32768 – 32767. Luku sijoitetaan ohjelmaan yksinkertaisesti vain kirjoittamalla se MSX-BASIC-lauseeseen. Esimerkiksi luku 5 voidaan lisätä lukuun 10 seuraavalla ohjelmalla:

```
10 PRINT 10+5
```

Ajaapa tuon pikku ohjelman kuinka monta kertaa tahansa, lopputulokseksi kirjoittuu aina lauseen 10 + 5 summa 15. Kun siitä on kerran tullut ohjelman vakio, sen arvo säilyy, haluatpa tai et.

Tutustutaan sitten **kiintolukuun.** Kiintoluku on joko positiivinen tai negatiivinen reaaliluku, joka ei välttämättä ole kuitenkaan kokonaisluku. Jos kiintolukuun liittyy murto-osa, sen täytyy kuitenkin olla desimaalimuodossa. Numerosarja ja yksi desimaalipiste määrittävät ohjelmaan vakion, joka on kiintoluku.

Hyväksyttäviä kiintolukuvakioita ovat esimerkiksi −3.645, 123.87, 0.013, −0.56746 ja .78.

On toinenkin vakiotyyppi, jossa voi olla desimaalipiste. Se on **liukuluku.** Myös liukuluku voi olla positiivinen tai negatiivinen. Se määritetään ohjelmaan seuraavalla tavalla, jota on toisinaan kutsuttu "tieteelliseksi merkinnäksi". Sen luku koostuu kolmesta osasta: **mantissasta,** kirjaimesta **E** tai **D** ja **eksponentista.** Asia käy parhaiten selville esimerkin avulla. Luku 753000 ilmaistaan liukulukuna 7.53E5. Siinä on mantissana 7.53 ja eksponenttina 5. Lause tulkitaan siten, että luku on 7.53 kerrottuna 10 potenssiin 5:llä eli yksinkertaisesti sama kuin jos desimaalipistettä siirrettäisiin viisi numeroa oikealle. Jos eksponentti on negatiivinen, desimaalipistettä siirretään vasemmalle. Eksponentti voi olla vain integertyyppinen ja arvoltaan −64 – 63.

D:n sijoittaminen E:n tilalle lauseeseen tarkoittaa sitä, että luku tallennetaan 14 desimaalin tarkkuudella eikä

6 desimaalin niin kuin E:tä käytettäessä. Hyväksyttäviä esimerkkejä liukuluvuista:

Varsinainen arvo	**Liukulukuesitys**
12400	12.4E3
0.00683	6.83E−3
−0.00004077032	−4.077032D−5
534500210	5.3450021D8

Kaikki numeerisesti esitettävät datatyypit ovat tallennettavissa molemmilla tarkkuuksilla. Yksinkertainen tarkkuus on siis 6 desimaalia ja kaksinkertainen 14 desimaalia. Luvun tarkkuus määritetään kirjoittamalla joko E tai D liukulukuvakioon ja #- tai !-merkki muunlaiseen lukuvakioon. Näin siis käy yksin- ja kaksinkertaisen tarkkuuden määrittäminen liukulukuun. Yksinkertainen tarkkuus integer- ja reaalilukuja varten saadaan aikaan asettamalla huutomerkki numerojonon loppuun, esimerkiksi 145! tai 89.6!

Kaksinkertainen tarkkuus saadaan liukuluvussa käyttämällä kirjainta D, kiinto- ja integerluvussa taas asettamalla haluttaessa loppuun #-merkki. Merkkiä # ei todellakaan ole pakko käyttää, sillä MSX-BASICin mukaisesti tietokone olettaa heti virran kytkettyä, että kaikkia lukuja käsitellään kaksinkertaisella tarkkuudella, jollei koneelle toisin ilmoiteta. Kaksinkertainen tarkkuus on siis MSX-BASICin oletuksen mukainen. Niinpä 5600 on kaksinkertaisella tarkkuudella ilmoitettu arvo, kuten ovat myös 4573.6, 12.93#, −89.9# ja 12314.

Heksadesimaalinen vakio on ehkä monelle uusi tuttavuus. Heksadesimaalilukujen perustana on 16. Tavallisesta kymmenjärjestelmästä (jossa yksikköinä ovat kymmenet, sadat jne.) poiketen heksadesimaalin perusyksikköinä ovat kuusitoista ja sen kerrannaiset, kuten kaksisataaviisikymmentäkuusi (16 x 16). Heksadesimaaliluvut rakentuvat näin:

Desimaaliarvo: 0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16

Heksadesimaaliarvo: 0 1 2 3 4 5 6 7 8 9
A B C D E F 10

Heksadesimaaliluvut muuttuvat siis 9:n jälkeen kirjaimiksi. Niinpä heksadesimaalikoodeissa on sellaisiakin lukuja kuin "FFFF" ja "BCDD". Heksadesimaaleja eli heksoja käytetään usein konekielen ohjelmointiin, eikä sitä kannata turhaan pohtia. Eikä kirjammekaan käsittele niitä muussa yhteydessä. Näin heksat määritetään: aseta etuliite **&H** luvun eteen, esimerkiksi &H3FA, &H1650 ja &H2FE.

Oktaalinen vakio on tietokoneen kulta-ajan muistojen kummajaisia. Oktaalijärjestelmässä käytetään kantalukuna 8:aa. Sitä tapaa vielä toisinaan, mutta harva enää alkaa käyttää sitä. Jos mielii kokeilla sitä, tässä esimerkki:

Desimaali: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Oktaali: 0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20

Myös oktaalivakion osoittamiseen tarvitaan etuliite, joka on tällä kertaa **&O.** Esimerkkejä hyväksyttävistä oktaalivakioista ovat &O12, &O5643 ja &O129.

Binaaristen vakioiden edessä on tunnuksena **&B.** Binaariluvut perustuvat kaksijärjestelmään, kuten ensimmäisessä luvussa kerrottiin. Tietokoneen kanssa siis kommunikoitiin 1:llä ja 0:lla. Sitä yksinkertaisempaa muotoa onkin vaikea keksiä. Binaarivakiot esitetään esimerkiksi näin: &B11110000, &B00101110 ja &B11001101. Yhdellä tavulla voi ilmaista luvut 0 – 255.

Viimeinen vakiotyyppi on **merkkijono.** Merkkijono on mikä tahansa lainausmerkkien väliin sijoitettu sana- tai numerosarja: sekä kirjaimet A:sta Ö:hön ja a:sta ö:hön että numerot 0:sta 9:ään ja kaikki muutkin MSX-BASICin käyttämät merkit, jotka on sijoitettu lainausmerkkien väliin.

Jonot voivat olla jopa 255 merkin pituisia. Tässä muutama malliksi:

"retu", "1234", "$%^*", "Minä olen ketju!", "345,687".

Jonoja käytetään usein PRINT-lausekkeessa, esimerkiksi kun ohjelmasta tulostetaan viesti:

```
10 PRINT "Terve, minä olen vakiomerkkijo
no."
20 PRINT "Minua ei voi muuttaa ohjelmaa
käytet-täessä."
```

Jonovakiokin on nyt esitelty, joten siinä olivat kaikki mahdolliset vakiot. Seuraavassa jaksossa tarkastellaan muuttujia, niiden määrittämistä ja käyttöä.

Muuttujat

Muuttujien avulla ohjelmassa voi ilmaista tiettyjä arvoja. Niitä voi vaihdella ohjelman käytön aikana. Samoin kuin vakioita, muuttujiakin on monenlaisia – kaikkiaan neljää tyyppiä.

Aluksi otamme selvää siitä, kuinka muuttujat määritetään. Kuten ohjelmoinnissa ylipäänsä, tulee muuttujien määrittämisessäkin ensin oppia muutamia perussääntöjä. Muuttuja voi olla vaikka kuinka pitkä, kunhan se ei vain vie koko käytettävissä olevaa muistitilaa! Muuttujan *täytyy* alkaa kirjaimella.

Muuttujaan ei saa jättää välilyöntiä. Muuttujaksi ei voi nimetä eikä edes sen osana saa käyttää BASIC-kielelle varattua sanaa. Kun nuo sanat kerran on varattu MSX-BASICin käyttöön, ei tuo kieli järin suvaitse, että siltä näpistetään! Nuo rajoitukset saattavat tuntua turhankin ankarilta, mutta niihin tottuu hyvin nopeasti. Kannattaa varmuuden vuoksi aloittaa käyttämällä pelkästään kirjaimia A:sta Ö:hön.

Muuttujan tyypin voi ilmoittaa käyttämällä muuttujan

ilmoitusmerkkiä. Tuo merkki sijoitetaan muuttujan nimen loppuun. Käytössä ovat nämä merkit:

% integer-muuttuja
! yksinkertainen tarkkuus
# kaksinkertainen tarkkuus
$ merkkijono

Seuraavassa on esimerkkejä hyväksyttävistä ja hylättävistä muuttujanimistä:

Hyväksyttävä: Paino% A$ MSX! Nimi10$ Pii%
Hylättävä: 15X NAMELISTS$ 2PALKAT DIM%

Muuttujat voi nimetä myös DEF-lausekkeen avulla. Se tehdään seuraavasti:

DEF <muuttujatyyppi>[**<lauseke>**],[**<lauseke>**],...

Muuttujatyypiksi voidaan ilmoittaa jokin seuraavista: **INT, SNG, DBL** tai **STR.** Ne edustavat järjestyksessä integeriä, sitten yksinkertaista ja kaksinkertaista tarkkuutta ja lopuksi merkkijonoa. Lauseessa käytetty ilmaus on joko kirjain tai rajattu kirjainjakso. Esimerkiksi kun lause DEFINT A suoritetaan, kaikki A:lla alkavat muuttujat ovat integertyyppisiä. Jos suoritettaisiin lause DEFSTR A–Z, kaikkien ohjelmassa olevien muuttujien oletettaisiin olevan merkkijonoja. Jos halutaan vaihtaa tietyn muuttujan tyyppiä DEF-lauseen jälkeen, tarvitsee vain käyttää muuttujan ilmoitusmerkkejä.

```
10 DEFSTR A-Z:REM Määritä muuttujat merk
kijonomuuttujiksi
20 E="RETU":S="VIRTA"
30 PRINT E+S
40 PRINT A
50 A%=100
60 PRINT A%*10
```

Jonoihin S ja R ei siis tarvitse liittää $-merkkiä. Kun muuttuja A tulostuu rivillä 40, se osoittautuu tyhjäksi jonoksi, josta tulee integer rivillä 50.

Viimeinen muuttujatyyppi on taulukkoalaindeksi, josta olikin jo puhe ja jota käsitellään vielä tarkemmin sivulla 103.

Kun olet valinnut muuttujalle nimen, voit sijoittaa sen ohjelmaan ja antaa sille arvon. Arvojen antamista muuttujalle sanotaan **muuttujan määrittämiseksi.** Voit kokeilla sitä seuraavalla ohjelmalla:

```
10 REM Muuttujan arvon sijoittaminen
20 PRINT NUMERO%
30 INPUT NUMERO%
40 LET NUMERO%=NUMERO%+1
50 PRINT NUMERO%
```

Ohjelmassa on ainoastaan yksi muuttuja, integertyyppinen NUMERO%. Kun sen arvo tulostetaan rivillä 20, vastauksen NUMERO% on 0. Kun MSX-BASIC-ohjelmassa kohdataan uusi muuttujanimi, sen oletusarvo on aina nolla. Riville 30 sijoitettu INPUT-lause kysyy NUMERO%:lle annettavaa arvoa. Rivillä 40 LET lisää arvoon 1:n, ja vastaus tulostuu rivillä 50.

Jos yrität kirjoittaa ?-valmiuden jälkeen kirjaimen tai sanan, BASIC reagoi vastaamalla: ?redo from start. Ja tuohon on syynä se, että antamiesi ohjeiden perusteella tietokone odottaa integermuuttujaa. Niinpä se odottaa yhä, kunnes annat sille integerin, jota se voi käsitellä.

Muistathan, että LET-osa rivillä 30 ei ole pakollinen. Lause "NUMERO%=NUMERO%+1" olisi kelvannut aivan yhtä hyvin. Kaikkia muuttujia voi määrittää tällä tavoin.

Yritäpä sitten vaihtaa %-merkki $:ksi, jolloin NUMERO vaihtuu merkkimuuttujaksi. Jos sen jälkeen yrität toteuttaa ohjelmaa, saatkin jälleen viestin MSX-BASICilta. Viestiin liittyy beep-äänimerkki, joka ei ole järin miellyttävä. Oletkin itse asiassa saanut aikaan virheilmoituksen "Type mismatch in 30" – yhteensopi-

maton merkkityyppi rivillä 30 (siitä Kohtalosinfonia!). Tuollaisia raivostuttavia pikku viestejä tulee ohjelmoitaessa vastaan. Ne on kaikki luetteloitu liitteeseen A. Ilmoitus tulostui, koska tietokone yritti lisätä kirjoittamasi sanan arvoa 1:llä, mikä ei tietenkään onnistunut. Aivan kuin olisi yrittänyt laskea yhteen omenoita ja appelsiineja: kahta erilajista asiaa ei voi käsitellä samoin. Jos poistat häiritsevän rivin 40, kaikki aineket ovat jälleen oikeissa laareissaan. Kirjoittamasi sana poistuu, eikä tietokoneen tarvitse enää yrittää laskea yhteen "omenoita" ja "appelsiineja". Yritä kuitenkin yhtä asiaa ennen kuin jatkat. Muuta riviä 40 seuraavasti:

```
LET NUMBER$=NUMBER$+"1"
```

Lainausmerkkien asettaminen 1:n molemmin puolin teki siitä jonon, ja senhän tietokone solmii mielihyvin toiseen jonoon.

On hyvä tietää, kuinka paljon muistia tietokone tarvitsee muuttujien arvoja varten. Alla on taulukko, josta selviää, kuinka paljon kukin muuttujatyyppi tarvitsee.

MUUTTUJAT:

Datatyyppi	**Tarpeellinen tavumäärä**
Integer	2
Yksinkertainen tarkkuus	4
Kaksinkertainen tarkkuus	8
Jonot	1 tavu merkkiä ja 3 jonoa kohden; "RETU" on 7 tavua

TAULUKOT:

Datatyyppi	**Tarpeellinen tavumäärä**
Integer	2 ainesta kohden
Yksinkertainen tarkkuus	4
Kaksinkertainen tarkkuus	8

Kun kaikki tyypit on esitelty, tutustutaan kunnolla

ohjelmointipuoleen ja annetaan hieman tilaa helpohkolle matematiikalle.

Matemaattisia ilmauksia

Matematiikka ei taida olla kaikkien mieliaihe. Tietokoneet ovat kuitenkin varsinaisia hakoja matematiikassa, joten aihetta ei toisaalta voi sivuuttaakaan. Helppoon alkuun pääsee kirjoittamalla ensin vain:

```
PRINT 10*10
```

Tietokone vastaa: "100".
Tällä tavoin tietokonetta voi käyttää kuin yksinkertaista taskulaskinta. Pyysit juuri tietokonetta kirjoittamaan 10 kertaa 10:n tulon. Kun käytettävissä ei ole lainkaan kertomerkkiä (x), tietokone käyttääkin asteriskia (*). Kokeile vielä kolmea muuta tehtävää:

```
PRINT 10+10
PRINT 10-5
PRINT 10/2
```

Siinä käsiteltiin yhteen-, vähennys- ja jakolasku eli tavallisimmat aritmeettiset tehtävät. Sijoitetaan sitten noita ohjelmiin. Seuraava ohjelma suorittaa ainoastaan neljä aritmeettista toimintoa kahdella luvulla, jotka voit valita mielesi mukaan.

```
10 INPUT "Antaisitko kaksi lukua";A,B
20 PRINT "A + B = ";A+B
30 PRINT "A - B = ";A-B
40 PRINT "A * B = ";A*B
50 PRINT "A / B = ";A/B
```

Eipä ole järin mutkikas ohjelma, mutta se toimii! Tartutaanpa vähän vaativampaan ohjelmaan. Uuden ohjelmamme tarkoituksena on hyväksyä käyttäjältään sanoja ja lukuja, joilla tulee suorittaa yksinkertaisia matemaattisia tehtäviä. Koneeseen tulee syöttää tehtävän nimi

isoilla kirjaimilla ja kaksi lukua. Käyttäjä voi kirjoittaa esimerkiksi: KERRO,5,4 — tulos on 20. Tuon tekemistä varten tulee kuitenkin ensin oppia pari seikkaa.

Ohjelma "tuntee" neljä aritmeettisissa tehtävissä yleisesti käytettävää sanaa. Ne ovat ohjelmassa vakioina. Ohjelman täytyy verrata käyttäjän antamia sanoja omaan vakiosanastoonsa. MSX-BASIC vertaa kahta merkkijonoa yksinkertaisesti vain asettamalla yhtäläisyysmerkin kuten kahta lukua käsitellessäänkin.

Niinpä tietokone päättää jo selvitetyn IF-lausekkeen perusteella kertomisesta ja jakamisesta sekä yhteen- ja vähennyslaskusta. Jos matematiikkatehtävän nimen sijasta kirjoitetaan sana LOPPU, ohjelma tulostaa kepeät hyvästinsä ja lopettaa. Muussa tapauksessa se vertaa syötteenä annettua sanaa neljään äsken mainittuun aritmeettiseen termiin. Jos vastaavuutta ei löydy, kone kertoo käyttäjälle, ettei se ole lainkaan ymmärtänyt vaan pyytää muuta tietoa. Ohjelma näyttää tältä:

```
10 REM YKSINKERTAINEN LASKURI
20 REM PRINT OHJESIVU
30 CLS
40 PRINT "LASKURIOHJELMA":PRINT "--------
--------"
50 PRINT "Kirjoita laskutehtävän ";
60 PRINT "nimi ja kaksi lukua.":PRINT
70 PRINT "Ohje ja luvut erotetaan ";
80 PRINT "toisistaan   pilkulla, esim: "
;
90 PRINT "Kerro,5,4"
100 PRINT "Kirjoita 'Loppu,0,0', kun ole
t val-  mis.":PRINT
110 PRINT:INPUT "Syöttötiedot : ";A$,X,Y
120 IF A$="Loppu" THEN GOTO 180
130 IF A$="Lisää" THEN SUM=X+Y:PRINT ;X;
"+n ja";Y;"+n summa on ";SUM:GOTO 110
140 IF A$="Vähennä" THEN SUM=X-Y:PRINT ;
X;"-n ja";Y;"-n erotus on ";SUM:GOTO 110
150 IF A$="Kerro" THEN SUM=X*Y:PRINT ;X;
```

```
"kertaa";Y;"on";SUM:GOTO 110
160 IF A$="Jaa" THEN SUM=X/Y:PRINT ;X;"j
aettuna";Y;":llä on";SUM:GOTO 110
170 PRINT "Anteeksi, en ymmärtänyt.";A$;
" Yritä uudestaan":GOTO 110
180 PRINT "Hei, hei!"
190 END
```

REM-lauseiden käyttö on varsin hyödyllistä, jos ohjelmaa säilytetään nauhalla pitkään eikä muisteta aivan tarkkaan, mitä sillä tehtiin. Sen avulla kyetään myös seuraamaan ohjelman juonta eli logiikkaa (tai pikemminkin sinun omaa logiikkaasi!), jos rakennetta haluaa joskus muuttaa. Luepa ohjelma kokonaan – se on melko helppotajuinen.

Yhteen-, vähennys-, kerto- ja jakolasku eivät ole MSX-BASICin ainoat matemaattiset taidot. Muita ovat luvun **potenssin, modulon** ja **negatiivisen arvon** laskeminen sekä **integerjako.** Kaikki matemaattiset operaattorit noudattavat tiettyä ratkaisujärjestystä. Kun MSX-BASIC-tulkki lukee riviä, jossa on monta aritmeettista operaattoria, sen täytyy ratkaista operaattorien suoritusjärjestys. Esimerkkilauseketta

A=B+C*D

ratkaistessaan MSX-BASICin on vaikea tietää, halutaanko ensin laskettavan yhteen B ja C, joiden summa sitten kerrottaisiin D:llä, vai halutaanko ensin kerrottavan C ja D, joiden tuloon sitten lisättäisiin B. Jos korvaat lauseen kirjaimet luvuilla, huomaat lopputuloksien eron.

Annetaanpa B:lle arvo 2, C:lle 4 ja D:lle 6, ja lasketaan ensin yhteen B ja C, joiden summa sitten kerrotaan D:llä. Silloin suoritus etenee seuraavasti:

Ensin laske yhteen 2 ja 4 = 6.
Kerro sitten keskenään 6 ja 6.
Niinpä A = 36.

Jälkimmäisessä ratkaisujärjestyksessä C * D suoritetaan ensin, minkä jälkeen tuloon lisätään B.

Ensin kerrotaan keskenään 4 ja 6 = 24.
Sitten lasketaan yhteen 24 ja 2.
Niinpä A = 26.

Aivan mutkattomasta lausekkeesta saadaankin siis kaksi eri tulosta. Itse asiassa MSX-BASIC antaisi vastaukseksi luvun 26, koska kieleen on ohjelmoitu järjestys, jonka mukaan kertolasku suoritetaan ennen yhteenlaskua.

Kertolasku saa siis etusijan yhteenlaskulta. Aritmeettisten operaattorien suoritusjärjestys on tämä:

Symboli	**Tehtävä**	**Esimerkki**
^	Potenssilasku	A^B
—	Negatiivisen arvon antaminen	—A
*,/	Kertominen ja liukuluvun jako	A*B A/B
\	Integerjako	A\B
MOD	Modulolasku	A MOD B
+,—	Yhteen- ja vähennyslasku	A+B A—B

Järjestystä voi muuttaa käyttämällä sulkumerkkejä. Jos edellä olevassa esimerkissä halutaan suorittaa ensin yhteenlasku ja vasta sitten kertominen, täytyyy lauseke muotoilla näin:

A = (B + C) * D

Sulkeissa oleva tehtävä suoritetaan aina ensin. Jos B on 2, C on 4 ja D on 6, vastaus on 36.

Potenssilasku on luvun kertomista itsellään, esimerkiksi:

```
PRINT 10^2
```

antaa tulokseksi luvun 100. PRINT-lause siis kehotti tietokonetta "laskemaan luvun 10 toisen potenssin". MSX-BASICin käyttämä merkintätapa vastaa tavallisesti käytettyä 10^2:een. Seuraavassa on lisää esimerkkejä ja niiden tulosteita:

Lause	**Tulo**	**Yleinen käytäntö**
PRINT 10^4	10000	10^4
PRINT 2^8	256	2^8
PRINT 90^2.5	76843.347142016	$90^{2.5}$
PRINT 1.37^40	29431.9730757	1.37^{40}

Integerjako eroaa tavallisesta liukuluvun jakamisesta. Operandeja katkaistaan integerluvuiksi siten, että ensin suoritetaan jakolasku ja sitten tulos katkaistaan kokonaisluvuksi. Katkaisu tehdään desimaalipisteen kohdalta. Katkaistu 2.57 on 2 ja 9.99999 on 9 jne. Käsitelläänpä integerjako vaihe kerrallaan, jotta nähdään sen suorittaminen yksityiskohtaisesti. Suoritetaan integerjako esimerkiksi luvusta 345.978 luvulla 12.866. BASIC etenee askelittain tällä tavoin:

Lause: PRINT 345.978\12.866

1 Katkaise : 345.978:sta 345
2 Katkaise : 12.866:sta 12
3 Jaa : 345 12:lla
4 Tulos = 28.75
5 Katkaise : 28.75:sta 28
6 Tulos = 28

Jos olisimme käyttäneet tavallista jakolaskua (ja sen /-symbolia) tulos olisi ollut aivan toinen (eli 26.890875174879).

Lopulta käsiteltäväksi otetaan **modulolasku.** Modulo osoitetaan MOD-operaattorilla. Sen avulla saadaan integerjaon jakojäännös. Jos kirjoitetaan "PRINT 7.86 MOD 2", tulokseksi kirjoittuu 1. Sen MSX-BASIC saa

aikaan katkaisemalla ensin 7.86:n 7:ksi ja jakamalla 7:n 2:lla, minkä jakojäännös on 1. Alla onkin jälleen joukko esimerkkejä:

Lause	Tulos
1234.4321 MOD 11.31	2 (1234/11 = 112 ja jakojäännös 2)
100 MOD 10	0 (100/10 = 10 ja jakojäännös 0)
76 MOD 13	11 (76/13 = 5 ja jakojäännös 11)
99.99 MOD 31	6 (99/31 = 3 ja jakojäännös 6)

On tärkeää muistaa, että jaatpa mitä tahansa, *älä jaa milloinkaan nollalla!* Tietokone tuottaa virheilmoituksen aina, jos jotain lukua yritetään jakaa nollalla. Varsinkin MOD-operaattorien ja integerjaon kanssa tulee olla varuillaan. Jos esitetaan vaikka lause 7 MOD 0.45 tai 70.45, molempien operaattori 0.45 katkaistaan 0:ksi, mikä aiheuttaa virheilmoituksen. Ole varovainen myös sen käytössä muuttujissa. Saatat yrittää jakaa lukua muuttujalla, jolle ei ole sijoitettu lainkaan arvoa.

Siihen päättyivät aritmeettiset operaattorit, joten voit huokaista helpotuksesta. Mutta operaattorit eivät vielä loppuneet. Kaksi tyyppiä käsitellään seuraavassa jaksossa.

Loogiset ja suhteelliset operaattorit

Nämä operaattorityypit tulevat usein käyttöön pitkiä ohjelmia laadittaessa, ja sinä käyttänet suhteellisia operaattoreita jopa useammin kuin aritmeettisia. Kannattaa siis tutustua näihinkin.

Suhteelliset operaattorit ainoastaan vertaavat asioita keskenään. Ne tarkistavat, ovatko kaksi asiaa identtisiä, onko toinen suurempi vai pienempi kuin toinen vai ovatko ne tyystin erilaisia.

Suhteellisten operaattorien kenties selkein tiivistelmä syntyy taulukon avulla. Niiden käyttö on *erittäin* yksinkertaista. Niitä on yhteensä kuusi, ja ne ja niiden vertailutoiminnot näyttävät seuraavilta:

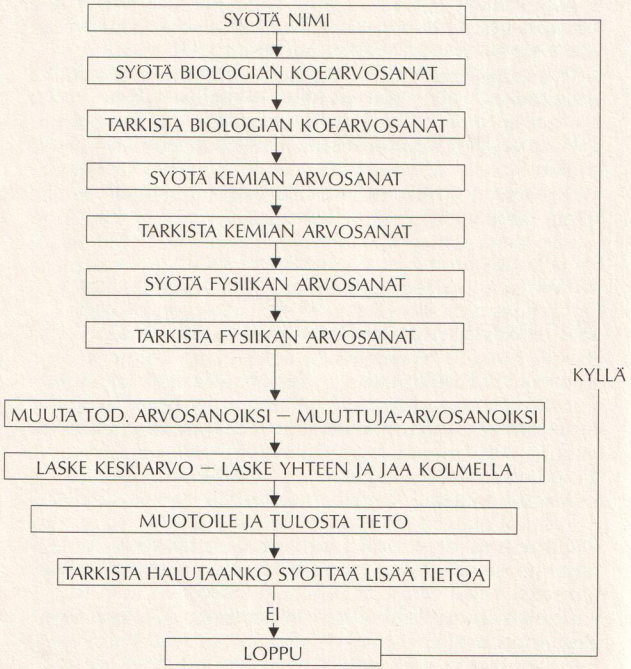
Operaattori	Funktio	Esimerkki
=	Tarkistaa kahden asian yhtäsuuruuden	X=Y
<>	Tarkistaa erisuuruuden	X<>Y
<	Pienempi kuin	X<Y
>	Suurempi kuin	X>Y
<=	Pienempi tai yhtäsuuri kuin	X<=Y
>=	Suurempi tai yhtäsuuri kuin	X>=Y

Nämä operaattorit ovat erittäin hyödyllisiä varsinkin IF...THEN-lauseissa, esimerkiksi jos on tarpeen suhteuttaa luokan koearvosana ja halutaan määrittää kunkin oppilaan oma arvosana. Tuo toteutetaan seuraavassa täysimittaisessa ohjelmassamme. Oppilaiden nimet, oppiaine ja koearvosanat syötetään ohjelmaan. Ohjelma tarkistaa syöttötietojen **oikeellisuuden,** joten on mahdotonta antaa sellaisia arvosanoja kuin −34 tai 230. Ohjelman lopputulokseksi saadaan oppilaan nimi, kunkin kokeen arvosana ja kaikkien aineiden keskiarvo. Kun jokaisen oppilaan tiedot on käsitelty, ohjelma kysyy, halutaanko esittää lisätietoja.

Samalla tapaamme uuden funktion, **INT**in, joka sieventää eli katkaisee luvun. (Katso MSX-BASICin koko funktioluetteloa liite A:sta.) Numerotietojen oikeellisuus tarkistetaan sitten aliohjelman avulla. Koska yhdellä ainoalla aliohjelmalla tarkistetaan kolme koearvosanaa, TEMP!-muuttuja saa tehtäväkseen tallentaa arvosanan ennen kuin hypätään aliohjelmaan. Joka kerta kun arvosana tarkistetaan, TEMP!-muuttujan arvo vaihtuu. Aliohjelmaa käytetään myös arvosanojen sijoittamisessa koetilastoihin, ja TEMP!-muuttujaa käytetään siihen aivan samalla tavalla.

Ohjelman askeleet voidaan tiivistää sanallisesti. Tuota ohjelmansuunnittelun tapaa nimitetään **algoritmin** laatimiseksi, mikä on pohjimmiltaan sama asia kuin jo käsittelemämme lohkokaavion laadinta. Ohjelmamme algoritmi on tällä kertaa:

ALKU Syötä oppilaan nimi.
Syötä biologian, kemian ja fysiikan arvosanat.
Tarkista tietojen oikeellisuus.
Muuta kukin koearvosana oppiaineen arvosanaksi.
Laske kolmen arvosanan keskiarvo ja sievennä lopputulos.
Tulosta nimi sekä biologian, kemian ja fysiikan arvosanat ja kokeiden keskiarvo.
Kysy, halutaanko käsiteltävän lisätietoja, ja jos halutaan, palaa ALKUun. Muutoin lopeta.



6 Lohkokaavio todistuksen arvosanan määritykseen

Seuraavassa on sitten listattu ohjelma. Selitykset ovat tarkat, jotta kävisi tarkasti selville, kuinka algoritmi on muunnettu MSX-BASICiin.

```
10 REM KOEPISTEET/ARVOSANAT
20 REM TYHJENNÄ NÄYTTÖ JA ANNA TIEDOT
30 CLS
40 PRINT "Kirjoita oppilaan nimi":INPUT
S$
50 PRINT "Kirjoita biologian pisteet":IN
PUT BIOL!
60 TEMP!=BIOL!:GOSUB370:BIOL!=TEMP!:REM
PISTEIDEN TARKISTUS
70 PRINT"Kirjoita kemian pisteet":INPUT
KEMI!
80 TEMP!=KEMI!:GOSUB370:KEMI!=TEMP!:REM
PISTEIDEN TARKISTUS
90 PRINT"Kirjoita fysiikan pisteet":INPU
T FYSI!
100 TEMP!=FYSI!:GOSUB370:FYSI!=TEMP!:REM
 PISTEIDEN TARKISTUS
110 REM PISTEIDEN MUUTTAMINEN ARVOSANOIK
SI
120 TEMP!=BIOL!:GOSUB430:G1=ARVO
130 TEMP!=KEMI!:GOSUB430:G2=ARVO
140 TEMP!=FYSI!:GOSUB430:G3=ARVO
150 REM LASKE KESKIARVO
160 KESK%=INT((BIOL!+FYSI!+KEMI!)/3):KES
K=(G1+G2+G3)/3:KESK=INT(10*KESK+.5)/10
170 REM TULOSTA TIEDOT
180 CLS:PRINT "OPPILAAN TULOKSET":PRINT"
------------------":PRINT
190 PRINT"OPPILAAN NIMI : ";S$:PRINT
200 PRINT"ARVOSANAT :":PRINT
210 PRINT"BIOLOGIA  : ";G1:PRINT"KEMIA
   : ";G2:PRINT"FYSIIKKA  : ";G3
220 PRINT
230 PRINT"KESKIARVO : ";KESK%;"/";KESK
240 PRINT:PRINT
250 REM JATKETAANKO ?
260 PRINT"KIRJOITATKO LISÄÄ TIETOJA":INP
UT"K TAI E";VASTAUS$
270 REM JOS VASTAUS ON "K" JATKETAAN, MU
```

```
UTEN LOPETETAAN
280 IF VASTAUS$="K" THEN GOTO 30
290 IF VASTAUS$="E" THEN GOTO 330
300 REM VASTAUS EI OLLUT SALLITTU, YRITÄ
 UUDESTAAN!
310 PRINT"VASTAA 'K' TAI 'E' !":GOTO 260
320 REM VASTAUS OLI 'E'
330 PRINT"TIETOJEN ANTAMINEN ON LOPPUNUT
"
340 END
350 REM ****** ALIOHJELMAT ******
360 REM ALIOHJELMA - PISTEIDEN TARKISTUS
370 IF TEMP!<=60  AND TEMP!>=0 THEN RETU
RN:REM PISTEET OK
380 REM MAHDOLLISUUS KORJAUKSEEN
390 PRINT"PISTEMÄÄRÄ EI VOI OLLA OIKEIN!
"
400 PRINT"ANNA UUDET PISTEET ":INPUT TEM
P!
410 GOTO 370
420 REM ALIOHJELMA - PISTEET ARVOSANAKSI
430 IF TEMP!>=55 THEN ARVO=10:RETURN
440 IF TEMP!>=49 THEN ARVO=9:RETURN
450 IF TEMP!>=43 THEN ARVO=8:RETURN
460 IF TEMP!>=35 THEN ARVO=7:RETURN
470 IF TEMP!>=25 THEN ARVO=6:RETURN
480 IF TEMP!>=15 THEN ARVO=5:RETURN
490 IF TEMP!<15 THEN ARVO=4
500 RETURN
```

Suhteellisia operaattoreita käytetään tässä ohjelmassa pääasiassa kolmeen tarkoitukseen: tarkistamaan syötelukujen oikeellisuus, sijoittamaan arvosana ja tarkistamaan lopulta käyttäjän vastaus kysymykseen: "ONKO VIELÄ KÄSITELTÄVIÄ OPPILASTIETOJA?" Arvosanaa sijoittaessaan ohjelma kysyy ainoastaan muutaman kysymyksen. Jos kokeiden arvosana on SUUREMPI TAI YHTÄSUURI KUIN todistuksen arvosanan rajaksi asetettu, niin ARVOSANAS-muuttujan arvoksi tulee ensimmäisessä tapauksessa "10", seuraavassa "9" jne.

Rivillä 370 oleva lause, jolla tarkistetaan annetun arvon kelvollisuus, saattaa näyttää hieman oudolta.

Tuo rivi sisältää sanan AND, joka on **looginen operaattori.** Se onkin MSX-BASICissa viimeiseksi käsiteltävän operaattorityypin edustaja.

Loogisia operaattoreita käytetään usein IF...THEN-operaatioissa. Ne suorittavat loogisia eli **Boolean-tyyppisiä operaatioita,** ja ne ovat NOT, AND, OR, XOR ja EQV. Tarkastellaan AND-ehtoa, jota käytimme viime ohjelmassamme:

```
370 IF TEMP!<=100 AND TEMP!>=0 THEN RETU
RN
```

Tuon rivin avulla tarkistettiin molempien ehtojen toteutuminen. Jos molemmat ehdot olivat tosia, silloin (THEN) ryhdyttäisiin joihinkin toimiin eli palattaisiin aliohjelmasta.

Tietokoneen logiikan mukaan "tosi" on "kyllä" ja "epätosi" on "ei". Oletetaan muuttuja X:lle arvo 10. Esimerkkimme avulla käy selville, kuinka tosi- ja epätosi-lausekkeita käytetään:

X<20 Tosi
X = 10 Tosi
X>10 Epätosi

Tuohon voisi vielä lisätä muuttujan Y, jonka arvo olisi vaikka 50. Jos lauseke näyttäisi tältä

(X = 10) AND (Y = 40)

ei se olisi tosi vaan epätosi, koska toinen ehdoista ei toteutunut. Ei voi sanoa, että X = 10 **ja** Y = 40, paitsi kun ne molemmat ovat tosia. Jos olisikin kirjoitettu lauseke

IF X = 10 OR Y = 40

sen antama vastaus olisi tosi, koska edes toinen ehdoista olisi tosi.

Ehtojen tarkistaminen on yksi niistä tavoista, joilla tietokone ikään kuin "ajattelee". Se kykenee vertaamaan kahta arvoa ja näyttää päättelevän vertailun tuloksen pohjalta, mitä tulisi tehdä seuraavaksi. Tämä perustuu kokonaan tietokoneen kykyyn tarkistaa yksinkertaisia matemaattisia lausekkeita ja ehtoja.

Loogisten operaattorien toiminta käy parhaiten selville **totuustaulujen** avulla. Totuustaulussa tarkistettavien ehtojen tilaa kuvataan joko merkillä "T" (eli "1"), jos ehto on tosi, tai "E" (eli "0"), jos se on epätosi. Loogisen operaation tulos riippuu siitä, mitä kahta ehtoa kysytään. Seuraava taulukko on tiivistelmä loogisista operaattoreista, jotka on luetteloitu arvojärjestyksessä.

NOT on TOSI vain kun ehto on epätosi. Se on siten EPÄTOSI, kun ehto on tosi. Esimerkiksi:

```
10 X%=0
20 X%=NOT(X%)
```

X% saa arvon −1. Toinen käyttöesimerkki:

```
10 X%=-1
20 IF NOT X% THEN BEEP
```

Ohjelma generoi beep-äänen, kun X% on epätosi (0).

NOTin totuustaulu:

X	NOT X
T	E
E	T

AND on TOSI vain, kun molemmat ehdot ovat tosia:

```
10 X=50:Y=100
20 IF (X=50) AND (Y=100) THEN BEEP
```

Kun molemmat ehdot ovat tosia, kuuluu beep-ääni.

ANDin totuustaulu.

X	Y	X AND Y
T	T	T
T	E	E
E	T	E
E	E	E

OR on tosi vain kun jompikumpi ehto on tosi:

```
10 X=10:Y=40
20 IF (X<100) OR (Y>40) THEN BEEP
```

Vastaus on tosi, koska X:n arvo täyttää toisen ehdoista.

ORin totuustaulu

X	Y	X OR Y
T	T	T
T	E	T
E	T	T
E	E	E

XOR (rajoittava OR) on TOSI, kun toinen ehdoista on tosi ja toinen epätosi, esimerkiksi:

```
10 X=10:Y=40
20 IF (X<100) XOR (Y>40) THEN BEEP
```

Tulos on tosi. Jos Y olisi ollut 100, tulos olisi ollut epätosi.

XORin totuustaulu

X	Y	X XOR Y
T	T	E
T	E	T
E	T	T
E	E	E

EQV on TOSI, kun molemmat ehdot toteutuvat samalla tavalla, ovat molemmat joko tosia tai epätosia. Kummankin esimerkkimme ehtolauseke on tosi:

```
10 X=10:Y=10
20 IF X=10 EQV Y=10 THEN BEEP
```

tai

```
10 X=10:Y=10
20 IF X<>10 EQV Y<>10 THEN BEEP
```

EQV:n totuustaulu

X	Y	X EQV Y
T	T	T
T	E	E
E	T	E
E	E	T

IMP on hieman erikoinen. Ehto on EPÄTOSI vain silloin, kun tarkistetuista ehdoista edellinen on tosi tai jälkimmäinen epätosi. Esimerkiksi:

```
10 X=10:Y=50
20 IF X=10 IMP Y=10 THEN BEEP
```

Ohjelma ei anna äänimerkkiä, koska X=10 on tosi ja Y=10 on epätosi.

IMPin totuustaulu

X	Y	X IMP Y
T	T	T
T	E	E
E	T	T
E	T	T

Näin olemme tehneet opasretken MSX-BASICin lukujen ja operaattorien pariin. Tästedes perehdymme hieman entistä tarmokkaammin ohjelmointiin.

Taulukoiden käyttö

Tässä jaksossa tarkastellaan **taulukoiden** tietorakennetta yksityiskohtaisten käyttöesimerkkien avulla. Taulukothan määritettiin DIM-lauseiden avulla. DIM-lauseessa ilmoitetaan taulukon nimi ja ulottuvuudet eli siihen sisällytettävien ainesten määrä.

Nimi määrittää lisäksi taulukkoon sijoitettavien tietoainesten tyypin. Taulukko voidaan määrittää esimerkiksi näin:

```
DIM A$(5)
```

jolloin taulukkoon voidaan sijoittaa enintään viisi eri ainesta, jotka ovat merkkejä. Samassa taulukossa ei voi käsitellä vain yhtä tyyppiä olevia tietoja, joiden sanotaan olevan taulukon **perustyyppiä.**

Jokaisen taulukon ainekselle tulee antaa oma **alaindeksinsä,** tunnuslukunsa, jonka avulla sen löytää vaivattomasti. Seuraavaa lyhyttä ohjelmaa käytetään MSX-BASICin värinumeroinnin muistiapuna.

```
10 DIM A$(16)
20 FOR I=0 TO 15
30 READ V$
40 A$(I)=V$
50 NEXT I
```

```
60 CLS
70 INPUT "Annatko värinumeron ";N%
80 IF N%<0 OR N%>15 THEN BEEP:GOTO 60
90 PRINT
100 PRINT "Väri";N%"on ";A$(N%)
110 REM Tiedot
120 DATA läpinäkyvä, musta, vihreä
130 DATA vaaleanvihreä, tummansininen
140 DATA vaaleansininen, tummanpunainen
150 DATA sinivihreä, punainen
160 DATA vaaleanpunainen
170 DATA tummankeltainen
180 DATA vaaleankeltainen
190 DATA tummanvihreä, sinipunainen
200 DATA harmaa, valkoinen
210 END
```

Seuraavaksi esitetään taulukko A$:n rakenne. Ennen kuin kirjoittaa luvun, tulee tarkistaa, onko luku hyväksyttävien arvojen alueella. Jos luku on oikea, sen voi kirjoittaa, jotta voisi vilkaista sitä vastaavan värin taulukosta A$.

Taulukon hakemisto	Sisältö
0	"LÄPINÄKYVÄ"
1	"MUSTA"
2	"VIHREÄ"
3	"VAALEANVIHREÄ"
4	"TUMMANSININEN"
5	"VAALEANSININEN"
6	"TUMMANPUNAINEN"
7	"SINIVIHREÄ"
8	"PUNAINEN"
9	"VAALEANPUNAINEN"
10	"TUMMANKELTAINEN"
11	"VAALEANKELTAINEN"
12	"TUMMANVIHREÄ"
13	"SINIPUNAINEN"
14	"HARMAA"
15	"VALKOINEN"

Kuten huomaat, taulukko on todella nimensä mukainen. Sen suurena apuna on nopeus. Kirjoitapa viimeisin ohjelmamme IF...THEN-lauseina, niin nopeusero käy selväksi:

```
80 IF N%=0 THEN PRINT "VÄRI 1 ON LÄPINÄK
YVÄ"
90 IF N%=1 THEN PRINT "VÄRI 2 ON MUSTA"
```

jne.

Esimerkkimme taulukko oli yksiulotteinen. On mahdollista laatia myös kaksi-, jopa kolmiulotteisiakin taulukoita. Seuraava ohjelma käyttää kaksiulotteista taulukkoa kolmen oppilaan kolmen aineen arvosanoja varten, kuten koearvosanaohjelmamme jo osoitti. OP-taulukon ulottuvuudet ovat 3 ainesta kertaa 3 ainesta. (Suluissa olevat luvut osoittavat oppilaiden määrän ja arvosanat.)

Biologia	**Kemia**	**Fysiikka**
OP(1,1)	OP(2,1)	OP(3,1)
OP(2,1)	OP(2,2)	OP(2,3)
OP(3,1)	OP(3,2)	OP(3,3)

Kahta erillistä taulukkoa, ON$ ja AI$, käytetään oppilaiden ja aineiden nimien tallennukseen. Ohjelmalla tallennetaan sekä oppilaiden arvosanat että nimet ja tulostetaan kaikki tiedot ruutuun taulukkona.

```
10 DIM OP(3,3),N$(3),AI$(3)
20 FOR N=1 TO 3
30 READ A$
40 AI$(N)=A$:REM ANNA AINEEN NIMI
50 NEXT N
60 DATA BIOLOGIA,KEMIA,FYSIIKKA
70 FOR I=1 TO 3:REM SYÖTTÖTIEDOT
80 CLS
90 INPUT "OPPILAAN NIMI ";S$
100 N$(I)=S$
110 FOR J=1 TO 3
```

```
120 PRINT AI$(J);" ARVOSANA ";:INPUT SC
130 OP(I,J)=SC
140 NEXT J
150 NEXT I
160 CLS
170 REM TULOSTUSTIEDOT TAULUKKONA
180 FOR I=1 TO 3
190 PRINT "NIMI ";N$(I)
200 PRINT
210 FOR J=1 TO 3
220 PRINT AI$(J),OP(I,J)
230 NEXT J
240 PRINT
250 NEXT I
260 END
```

Asioiden saattaminen oikeaan järjestykseen

Tietojen lajittelu on niitä pitkäveteisiä ja rasittavia tehtäviä, joihin tietokoneiden nopeus ja kärsivällisyys soveltuvat mainiosti. Matemaatikot ja tietokonesuunnittelijat ovat jo vuosia tutkineet lajittelua. Emme turhauta sinua liialla teoretisoinnilla, vaan voit heti tutustua seuraavaan yksinkertaiseen lajitteluoperaatioon. Käytämme samalla taulukoita lajiteltavan tiedon säilytykseen.

Tätä hyvin tunnettua lajittelumenetelmää kutsutaan **kuplalajitteluksi (bubblesort).** Ensiksi lajitteluohjelma, ja sitten selitys.

```
10 CLS
20 INPUT "LAJITELTAVIEN LUKUMÄÄRÄ";N
30 DIM TAULUKKO(N)
40 FOR I=1 TO N
50 INPUT "ANNA LUKU";X
60 TAULUKKO(I)=X
70 NEXT I
80 CLS
90 PRINT"LAJITELTAVIEN LUKUJEN ALKUPERÄI
```

```
NEN      JÄRJESTYS":PRINT
100 FOR I=1 TO N:PRINT TAULUKKO(I):NEXT
I
110 REM TIEDON LAJITTELU
120 FOR I=2 TO N
130 FOR J=N TO 1 STEP -1
140 IF TAULUKKO(J-1)>TAULUKKO(J) THEN SW
AP TAULUKKO(J-1),TAULUKKO(J)
150 NEXT J
160 NEXT I
170 REM TULOSTA LAJITELLUT LUVUT
180 PRINT:PRINT "LAJITELLUT LUVUT":PRINT
190 FOR I=1 TO N:PRINT TAULUKKO(I):NEXT
I
200 END
```

Tämän kuplalajitteluversion avulla syöttötiedot lajitellaan nousevaan järjestykseen. Jos halutaan lajitella laskevaan järjestykseen, merkki (>) on vaihdettava pienempi kuin -merkkiin (<).

Versiomme selaa ainekset taulukon "yläpäästä" toiseksi alimpaan ja vertaa käsiteltäväksi ottamaansa arvoa heti sen alla olevaan. Jos alla on sitä korkeampi arvo, arvot vaihtavat paikkaa. Kun data on selattu kertaalleen, taulukon alin arvo sivuutetaan ja suoritus alkaa uudelleen ylimmästä arvosta. Otetaanpa esimerkiksi lajittelematon data:

100 7 9 1 514

Lajittelun välivaiheet näkyvät taulukostamme:

I:n arvo	J:n arvo	Datan järjestys
2	5	1 100 7 9 514
3	5	1 7 100 9 514
4	5	1 7 9 100 514
5	5	1 7 9 100 514

Esimerkistämme näkee, kuinka arvo 100 siirtyy lukusarjassa ylöspäin. Kuplalajittelu on vaihtolajittelua, sillä taulukon arvothan vaihdetaan (SWAP) oikeaan järjes-

tykseen. Tätä menetelmää voi käyttää myös nimien, osoitteiden tai muiden merkkijonojen lajitteluun, kunhan vaihtaa taulukoiden datatyypin ja syöttää sopivat muuttujat.

Tässä lajittelutavassa on kuitenkin heikkoutensa. Jos käsiteltävänä on vaikka lukusarja 1 2 3 5 4, se asettuu oikeaan järjestykseen kerralla – kun 5 ja 4 ovat vaihtaneet paikkaa. Mutta siitä huolimatta ohjelma selaa dataa yhä uudelleen, vaikka kaikki onkin jo järjestyksessä. Se vertaa turhan monta kertaa silloin kun luvut ovat alunperin *miltei* oikeassa järjestyksessä.

Miten ohjelman saa ymmärtämään, että kaikki on jo kohdallaan? Voimme ilmoittaa ohjelmalle, että luvut ovat järjestyksessä silloin, kun minkään arvon paikkaa ei tarvitse vaihtaa. Ohjelman viime selauksesta mahdollisesti aiheutuneen paikanvaihdon osoittava muuttujan arvo voidaan vaihtaa 0:sta 1:een. Jos arvo on selauksen jälkeen 1, ohjelma jatkaa selaamista, kunnes luvut ovat järjestyksessä. Tässä tapauksessa muuttujaa kutsutaan **lipuksi,** joka osoittaa jonkin seikan tapahtuneen. Ohjelma tarkistaa lipun arvon, jotta mahdollinen muutos huomattaisiin.

Viime ohjelman tehokkuus paranee, jos mukaan lisätään pari riviä:

```
115 F=0
155 IF F=0 THEN GOTO 170
```

ja muutetaan rivi 140:

```
140 IF ARRAY(J-1)>ARRAY(J) THEN SWAP ARR
AY(J-1),ARRAY(J):F=1
```

Lajitteluohjelmia voidaan käyttää kaikenlaiseen. Vaikka kuplalajittelu ei olekaan lyömätön, se on varmaankin alkuun aivan riittävä.

Matemaattiset funktiot

MSX-BASIC tarjoaa käyttöön monia matemaattisiin suorituksiin suunniteltuja **funktioita.** Kaikkiin ei kuitenkaan kannata perehtyä kovin tarkkaan, ja käymmekin läpi ainoastaan yleisimmät. Funktioon sijoitetaan arvo, ja se vastaa toisella arvolla. Tarkastelemme trigonometrisiä funktioita **SIN, COS** ja **TAN** sekä neliöjuurta **SQR.**

Funktioon täytyy sijoittaa **argumentti.** Argumentti on muuttuja-arvo, jonka funktio laskee. Argumentti kirjoitetaan sulkuihin funktion nimen jälkeen.

SIN-funktion avulla lasketaan kulman sinifunktio – jolla ei ole mitään tekemistä värien kanssa! SIN saa kulman radiaaneina ja antaa vastauksen sininä. Jos et ymmärrä radiaaneja vaan haluat mieluimmin käyttää asteita, käytä seuraavaa yhtälöä muuttamaan asteet radiaaneiksi.

Radiaanit = asteet * pii/180

Pii on suunnilleen 3.141593. Jos siis halutaan laskea 45-asteisen kulman siniarvo, muunnetaan asteet ensin radiaaneiksi ja sitten käytetään SIN-funktiota:

```
10 A=45
20 R=A*(3.141593#/180)
30 PRINT SIN(R)
```

Ruutuun ilmestyy luku 0.7071. Myös COS- ja TAN-funktiot toimivat samaan tapaan radiaaneilla ja antavat vastaukseksi kosini- ja tangenttikulmat.

SQR-funktio ilmoittaa luvun neliöjuuren. Neliöjuuri on aina saatavissa, kunhan laskettavaksi ei tarjota negatiivista lukua tai nollaa. Tietokoneet kykenevät monenmoisiin suorituksiin, mutta mahdottoman neliöjuuren löytäminen ei kuulu niihin. Yritäpä seuraavaa ohjelmaa:

```
10 PRINT SQR(100)
20 PRINT SQR(4)
30 PRINT SQR(16)
```

Tulokseksi ilmoitetaan arvot 10, 2 ja 4. Liite A antaa yksityiskohtaiset tiedot MSX-BASICin sekä matemaattisista että muista funktioista.

Käyttäjän määrittämät funktiot

Jos MSX-BASICin tarjoamat matemaattiset funktiot eivät aivan riitä kaikkiin tehtäviisi, voit tietysti laatia lisäfunktioita. Sitä varten on olemassa lause DEF FN, jonka avulla voi luoda käskysarjoja ja antaa niille omat nimet vastaisen varalle. Ehkäpä haluat saada luvun neliön – ilman että x 2:een olisi käytettävissä. Sellainen funktio ilmaistaisiin näin:

Ohje DEF FN rakentuu seuraavasti:

DEF FN<funktion nimi>(<muuttuja>,<muuttuja>,. . .) = <lauseke>

Esimerkkimme funktion nimi on A, ja lauseke on X*X. Suluissa oleva arvo tunnetaan funktion **valemuuttujana.** Valemuuttuja korvataan varsinaisella muuttujalla, jota **kutsutaan.** Kokeile seuraavaa yksinkertaista ohjelmaa:

```
10 DEF FNA(X)=X*X
20 INPUT S
30 PRINT FNA(S)
```

Ohjelma ilmoittaa neliöarvon mille tahansa luvulle, joka siihen sijoitetaan. Tällä kertaa X on funktion valearvo, joka korvataan rivillä 30 annetulla arvolla S. Funktion argumenttien määrää ei ole rajattu yhteen. Jos haluat kertoa kaksi arvoa keskenään, riittävät seuraavat toimet:

```
10 DEF FNA(X,Y)=X*Y
20 INPUT S,T
30 PRINT FNA(S,T)
```

Käyttäjän määrittelemät funktiot toimivat aivan samaan tapaan kuin MSX-BASICin määrittelemät.

Muistatko Pythagoraan teoreeman? Kertaamme sen niillekin jotka eivät muista. Jos kolmio on suorakulmainen, hypotenuusan neliö on yhtä suuri kuin suorankulman sivujen summan neliö. Niinpä jos halutaan saada selville hypotenuusan (c) pituus kolmiosta, jonka muut sivut ovat a ja b, käytetään seuraavaa yhtälöä:

$$c = \sqrt{a^2 + b^2}$$

Pituuden laskemiseen soveltuva funktio on tällä kertaa sijoitettu seuraavaan ohjelmaan:

```
10 DEF FNA(A,B)=SQR((A^2)+(B^2))
20 INPUT "SIVUN A PITUUS ON ";X
30 INPUT "SIVUN B PITUUS ON ";Y
40 PRINT "SIVUN C PITUUS ON  ";FNA(X,Y)
50 END
```

Tämän funktion muunnelmaa käytetään grafiikkaohjelmassa. Funktioiden käytöstä on useita etuja. Jos vaikka pyrit laatimaan yhtälöitä atomin ytimen halkaisuun, voi pitkien yhtälöiden kirjoittaminen käydä sangen rankaksi.

Funktiot ovat määrityksen jälkeen hyvin tehokkaita ja yksinkertaisia käyttää. Ne ovat myös nopeampia kuin yhä uudestaan ja uudestaan kirjoitettavat matemaattiset lausekkeet. BASIC-tulkin ei tarvitse suorittaa uudelleen yhtälöä, joka säilyy samana kun muuttujat vaihtuvat. Funktiot säästävät myös muistitilaa. Ja kaiken kukkuraksi, niiden avulla ohjelmat avautuvat vielä viikkojen kuluttua entistä helpommin itsellesi ja myös sille viattomalle sivustakatsojalle, joka ohjelmaasi ihastellessaan tokaisee: "No miten se sitten toimii?"

Ohjelmien käyttö

Tässä jaksossa käsitellään eräitä hyödyllisiä ohjelmia sekä perehdytään datan oikeellisuuteen ja näyttävien ohjelmatulosteiden aikaansaamiseen.

Tiedon syöttö

Aluksi vilkaisemme muutamia keinoja, joiden avulla syöttörutiinit muotoutuvat helpoiksi, esittelemme LOCATE-käskyn ja INKEY$-muuttujan käyttöesimerkeillä, joilla luodaan valikkopohjaisia ohjelmia.

Valikkopohjainen järjestelmä toimii tietokoneohjelmassa siten, että valitset luettelosta haluamasi asian kuin ravintolan ruokalistasta; valikon englanninkielinen nimi onkin **menu** eli ruokalista. Tietokoneen tarjoamat vaihtoehdot saa näkyviin ruutuun. Sitten luettelosta valitaan mieleinen asia ja annetaan tietokoneen suorittaa halutunlainen tehtävä. Tämä käy näppärästi kirjoittamalla vaihtoehdon viereen numerot, joista käyttäjän tulee valita mieleisensä. Tämä yksinkertainen ohjelma tuo ruutuun luettelon ja pyytää käyttäjää valitsemaan siitä jonkin vaihtoehdon.

```
10 REM yksinkertainen valikko
20 REM tulosta vaihtoehtoluettelo
30 CLS:PRINT "1. TULOSTA VIESTI"
40 PRINT "2. SOITA SÄVELMÄ"
50 PRINT "3. PIIRRÄ YMPYRÄ"
60 PRINT "4. POISTU"
70 LOCATE 1,20
80 PRINT "VALITSE VAIHTOEHTONUMERO";
90 A$=INKEY$:IF A$="" THEN 90 ELSE PR
INT A$
100 VAIH%=ASC(A$)-48
110 IF (VAIH%<1) OR (VAIH%>4) THEN BE
EP:LOCATE 22,20:GOTO 90
120 REM VALITSE TOIMINTAVAIHTOEHTO
130 ON VAIH% GOSUB 500,600,700,800
500 CLS:PRINT "TERVE. MINÄ OLEN MSX-T
IETOKONE!"
510 FOR I=1 TO 800:NEXT I
520 RETURN 30
600 REM SOITA
610 PLAY "CDEFG"
620 FOR I=1 TO 800:NEXT I
630 RETURN 30
700 REM PIIRROS
710 SCREEN 2
720 CIRCLE (128,92),70,1
730 PAINT (128,92),1
740 FOR I=1 TO 800:NEXT
750 SCREEN 0:RETURN 30
800 CLS:END:REM LOPETA OHJELMA
```

Rivillä 90 tutustuimme erityiseen INKEY$-muuttujaan, joka tallentaa painettavan näppäimen nimen. Rivin 90 lause sijoittaa INKEY$-muuttujan nykyisen arvon muuttujaan A$. Jos A$ on tyhjä (IF A$ = ""), ei mitään näppäintä ole painettu. Silloin ohjelma hyppää takaisin rivin alkuun ja tarkistaa INKEY$-muuttujaa, kunnes se saa arvon. Tuota tapahtuman toteutumisen tarkistusta kutsutaan kiertokyselyksi. Rivi 90 antaa valita näppäimistön avulla.

Kun näppäintä on painettu, A$ tallentaa näppäimen nimen muistiin. Seuraavaksi täytyy tarkistaa, onko pai-

nettu "1", "2", "3" vai "4". Kaikkeen näppäimistön kautta syötettyyn suhtaudutaan kuin merkkiin. Olisi voitu käyttää tavallista INPUT-lausettakin, mutta silloin ohjelmaa voitaisiin käyttää väärin.

Voit esimerkiksi kirjoittaa merkin G, mikä saa BASIC-tulkin tuottamaan virheilmoituksen. Ohjelmien tulisi itse tarkistaa syöttötietojen oikeellisuus. BASICia ei tule vaivata sillä.

Ohjelma muuttaa merkin kokonaisluvuksi ja tarkistaa sitten tiedon soveltuvuuden. BASICin ASC-funktio muuttaa merkit kokonaisluvuiksi. Kaikilla MSX-BASICin merkeillä on ASCII-koodin mukainen tunnus (katso lukua 1). ASC-funktio tuottaa jokaisesta merkistä sekä merkkijonosta ASCII-koodin. ASCII-koodit luvuille 1, 2, 3 ja 4 ovat 49, 50, 51 ja 52. Ja rivi 100 muuttaa syötetyn merkin ASCII-koodin mukaiseksi ja vähentää koodiluvusta 48. Jos siis on painettu näppäintä 1, rivi 100 muuttaa 1:n koodinumeroksi 49, vähentää siitä 48 ja antaa vastaukseksi sen mitä haluttiinkin eli 1:n.

Sitten ohjelma voi tarkistaa syöttöarvon kelvollisuuden eli sen, ettei se ole pienempi kuin 1 tai suurempi kuin 4. Jos arvo ei ole hyväksyttävissä, kuuluu beep-ääni, ja havaitsemme BASICissa uuden ominaisuuden. Kohdistin siirtyy viimeisestä sijaintipaikastaan LOCATE-käskyn mukaiseen uuteen kohtaan. Tällä kertaa kohdistin sijoitetaan rivin "VALITSE VAIHTOEHTONUMERO" loppuun, ja oikaistu tieto voidaan syöttää. LOCATE 22,20 tarkoittaa: "Sijoita kohdistin 20. riville 22. merkin kohdalle." Tekstitilassa merkin 0 voi sijoittaa enintään 40 paikkaan, koska sen enempää merkkejä ei silloin riville mahdu, ja tekstitilassa 1 yläraja on 36. Suurresoluutiotilassakin kohdistimen voi sijoittaa. Suurin ero on siinä, että ruutua käsitellään 256 merkkiä leveänä ja 192 riviä korkeana.

Kun oikea tieto lopulta syötetään, ohjelman täytyy päätellä, mitä käyttäjä haluaa. Silloin tulee mukaan ON ... GOSUB. Se on kuin rivillinen IF ... THEN-lau-

seita. BASIC tarkistaa määritetyt vaihtoehdot – käyttäjän suorittaman valinnan – ja hyppää uuteen kohtaan ohjelmassa sen mukaan valittiinko 1, 2, 3 vai 4. Kirjoittamamme rivin 130 asemesta olisi voitu yhtä hyvin käyttää seuraavaa koodauspätkää:

```
130 IF VAIH%=1 THEN GOSUB 500
132 IF VAIH%=2 THEN GOSUB 600
134 IF VAIH%=3 THEN GOSUB 700
138 IF VAIH%=4 THEN GOSUB 800
```

Kun tietokone on toteuttanut käyttäjän toivomuksen, se odottaa näppäimen painallusta ennen kuin palauttaa jälleen päävalikon. Se tapahtuu INKEY$-muuttujan avulla tapahtuvalla näppäimistö-valinnalla. Kun INKEY$ saa arvon, ohjelma palaa alkuun ja suoritus voi toistua, kunnes käyttäjä valitsee luvun 4, joka lopettaa ohjelman.

Sauvaohjaimen käyttö

Valikkopohjaista ohjelmaa voidaan käyttää myös sauvaohjaimen avulla, siirtämällä kohdistin ruudussa esitetyn vaihtoehdon kohdalle. Kun kohdistin on halutun vaihtoehdon kohdalla, painetaan välilyöntinäppäintä, mikä osoittaa valinnan. On mahdollista laatia yksinkertainen ohjelma, joka pelkästään tulostaa ruutuun sanaluettelon, antaa käyttäjän siirrellä kohdistinta, antaa hänen sijoittaa kohdistimen halutun vaihtoehdon viereen ja tehdä jotain välilyöntinäppäimen painalluksella. Ohjelmassa esitellään kaksi uutta funktiota, erityismuuttuja ja lause, jotka ovat sangen hyödyllisiä. Niiden toiminta käy selville, kun kirjoitat tämän uuden valikkopohjaisen ohjelman:

```
10 REM valikko-ohjelma 2
20 REM määritä näytön tila
30 SCREEN 1:KEY OFF
40 REM anna x ja y
50 X=1:Y=1
60 REM tulosta valikko
```

```
70 LOCATE 18,12:PRINT "LOPPU"
80 LOCATE 18,16
90 PRINT "SÄVELMÄ"
100 REM AKTIVOI VÄLILYÖNTINÄPPÄIMEN H
AVAITSEMINEN
110 STRIG(0) ON
120 REM VIE KOHDISTIN X:N JA Y:N OSOI
TTAMAAN KOHTAAN
130 LOCATE X,Y,1
140 ON STRIG GOSUB 300:REM TARKISTA O
NKO PAINETTU
150 REM TARKISTA SAUVAOHJAIMEN SUUNTA
 (IF STICK(0)=3 THEN 200 JNE)
160 A=STICK(0)
170 ON A GOTO 190,195,200,205,210,215
,220,225
180 GOTO 160
190 Y=Y-1       :GOTO 240:REM "YLÖS"
195 X=X+1:Y=Y-1:GOTO 240:REM "YLÄOIKE
ÄAN"
200 X=X+1       :GOTO 240:REM "OIKEAAN
"
205 X=X+1:Y=Y+1:GOTO 240:REM "ALAOIKE
ÄAN"
210 Y=Y+1       :GOTO 240:REM "ALAS"
215 X=X-1:Y=Y+1:GOTO 240:REM "ALAVASE
MPAAN"
220 X=X-1       :GOTO 240:REM "VASEMPA
ÄAN"
225 X=X-1:Y=Y-1:GOTO 240:REM "YLÄVASE
MPAAN"
240 REM TARKISTA KOORDINAATTIEN OIKEE
LLISUUS. VAIHDA VIRHEELLISET ARVOT.
250 IF X>40 THEN X=40
260 IF X<1 THEN X=1
270 IF Y>24 THEN Y=24
280 IF Y<1 THEN Y=1
290 GOTO 130:REM SIJOITA KOHDISTIN UU
TEEN PAIKKAAN
300 REM ALIOHJELMA - VAHVISTAA JA TOT
EUTTAA VAIHTOEHDOT
310 IF CSRLIN=12 THEN GOTO 500
320 IF CSRLIN=16 THEN PLAY "CDEFGGFED
```

```
C":RETURN
330 REM VIRHEEN ILMOITUS - BEEP
340 BEEP:RETURN
500 CLS:PRINT "LOPPU":END
```

Lause STRIG(0) ON neuvoo tietokonetta odottamaan sauvaohjaimen liipaisimen painallusta. Mikäli sauvaohjainta ei ole liitetty, välilyöntinäppäin toimii liipaisimena. ON STRIG GOSUB -lause neuvoo tietokoneelle, mitä tehdä kun liipaisinta painetaan. Kun aliohjelma on suorittanut liipaisimen painalluksen edellyttämän tehtävän, se toteuttaa itsekseen toisen STRIG(0)-lauseen aliohjelmasta palatessaan. Saman voi tehdä myös lauseilla STRIG(0) OFF ja STRIG(0) STOP. Edellinen ilmoittaa tietokoneelle, ettei tarvitse odottaa liipaisimen painallusta, jälkimmäinen taas ilmoittaa, että tietokoneen tulee huomata painallus mutta olla tekemättä mitään. Hypätään aliohjelmaan, joka tietää mitä tehdä liipaisimia painettaessa, kun STRIG(0) ON -lausetta suoritetaan.

STICK-funktio tarkkailee sauvaohjaimen liikkeen suuntaa. Jos STICK(0):n antama arvo on 0, sauvaohjainta ei ole liikutettu lainkaan. Jos laitteeseen ei ole yhdistetty sauvaohjainta, voidaan käyttää kohdistinnäppäimiä. Jos esimerkiksi halutaan siirtää kohdistinta yläoikeaan, tulee painaa yhtaikaisesti sekä ylöspäin että oikeaan osoittavaa näppäintä. STICK(0) asettaa kohdistinnäppäinten oletukset, jollei laitteeseen kytketä sauvaohjainta.

POS-funktio tuo ruutuun näkyviin kohdistimen kulloisenkin horisontaalisen aseman (rivillä) numeroina. Funktiolle voi sijoittaa minkä tahansa argumentin. Kun argumentilla ei ole mitään varsinaista merkitystä funktiolla saadun tiedon kanssa, sitä kutsutaan **valeargumentiksi.** CSRLIN on erityismuuttuja, joka tarkkailee kohdistimen vertikaalista (eli pysty-) asemaa ruudussa.

Keskeytysten käyttö

Keskeytykset ovat hyvin käyttökelpoinen käskyjoukko, joka on erittäin harvinainen BASIC-murteissa. Keskeytys on nimensä mukaisesti jonkin tapahtuman aiheuttama katkos tavanomaisessa ohjelmankulussa. Keskeytyksiä käytetään jonkin tapahtuman **löytämiseksi,** jotta se huomattaisiin ja jotta sille tehtäisiin jotain.

Keskeytys eroaa kiertokyselystä, jolla tarkistettiin tietyn näppäimen painaminen. Ero käy selville seuraavista kahdesta ohjelmasta, jotka molemmat on laadittu soittamaan tietty sävel, kun välilyöntinäppäintä painetaan.

```
10 REM kiertokysely
20 PRINT "KIERTOKYSELY"
30 GOSUB 90
40 PRINT "EI OLE YHTÄ"
50 GOSUB 90
60 PRINT "TEHOKAS"
70 GOSUB 90
80 GOTO 20
90 REM  TARKISTA ONKO VÄLILYÖNTINÄPPÄ
INTÄ PAINETTU
100 A$=INKEY$
110 IF A$=CHR$(32) THEN PLAY "C":RETU
RN
120 RETURN
```

Keskeytysten avulla ongelman voi ratkaista näin:

```
10 REM KESKEYTYS
20 STRIG(0) ON
30 ON STRIG GOSUB 80
40 PRINT "KESKEYTYKSET"
50 PRINT "OVAT"
60 PRINT "TEHOKKAAMPIA"
70 GOTO 40
80 PLAY "CDEFG":RETURN
```

Kiertokyselyssä tietty jakso koodista tarkistaa, tarvitseeko painettua välilyöntiä kirjoittaa. Jälkimmäisessä

ohjelmassa STRIG(0) ON käskee tietokonetta odottamaan välilyönnin painamista ja ON STRIG GOSUB määrittää rivin, jolle ohjelman tulee painalluksesta haarautua.

Menetelmien ero selviää ehkä seuraavan kuvitellun tilanteen avulla. Oletetaan vaikka, että täytyisi paistaa kakku uunissa ja kirjoittaa samanaikaisesti kirje ystävälle. Kiertokyselymenetelmän mukaisesti toimittaessa tulisi viiden tai kymmenen minuutin välein käydä tarkistamassa, onko kakku paistunut, ja palata sitten kirjettä kirjoittamaan. Keskeytysmenetelmän mukaisesti toimittaessa lieteen olisi laitettu ajastin ja merkkiääni. Kirjettä voisi silloin kirjoittaa murehtimatta kakun kypsymistä, ja kakku voitaisiin ottaa uunista, kun merkkiääni ilmoittaa sen olevan valmis.

MSX-BASICissa on monia keskeytystapoja. Ne pysäyttävät tapahtumia toimintonäppäimen painalluksesta, STOP-näppäimen painalluksesta, virheestä, spriten törmäyksestä tai muusta. Keskeytyksiä varten on kaksi käskyä – <interrupt>ON (jonka juuri tapasimme) ja <interrupt>STOP. Jälkimmäinen muistaa, että jotain on tapahtunut, mutta sen jälkeen ei voi käyttää <interrupt>GOSUB-lausetta. Silloin tapahtuman voi pysäyttää hetkeksi menettämättä sitä hallinnastaan. <interrupt>ON-käskyn suorittaminen vuorostaan saa ohjelman haarautumaan, jos keskeytys on muistettu.

Keskeytyskäskyjä tulee kuitenkin käyttää harkiten. Kun ON ERROR on suoritettu, *kaikki* muut keskeytykset mitätöityvät. Jos siis ON ERROR -lausetta käytetään muiden keskeytysten yhteydessä, on hyvä muistaa, että kaikki muut keskeytykset täytyy virittää uudelleen.

Keskeytyskäskyjen avulla ohjelmat on helppo jäsentää tiiviiksi. Suurin haitta on se, että niitä on hieman hankala jäljittää ja korjata virhetilanteissa, joten niiden selitysten tulee olla selkeitä.

Datan tulostaminen

Tutustuimme jo alkujaksoissa PRINT-käskyyn ja sen käyttöön tulosta ruutuun muotoilevien kaksois- ja puolipisteen kanssa. Mainitsimme myös, että PRINT USING -käskyyn tutustuttaisiin aikanaan läheisesti.

PRINT USING on aivan toista maata kuin PRINT. Tämän lauseen avulla voidaan määrittää tulosteen muotoilu aivan tietyllä tavalla. Siihen käytetään erityisiä muotoilumerkkejä. Merkkidatan tulostukseen on kolme muotoilumerkkiä, !, & ja a. Ensiksi tutustumme huutomerkkiin. Kokeilehan, mitä seuraava ohjelma saa aikaan:

```
10 INPUT A$
20 PRINT USING "!";A$
```

Vain jonon ensimmäinen merkki tulostuu. Tästä on hyötyä varsinkin kun tulostetaan nimiluetteloa. Ohjelmamme avulla voisi etunimet typistää alkukirjaimeen. Seuraava ohjelma vastaanottaa syötettävät nimet ja tuottaa muotoilluksi tulosteeksi sukunimen ja etunimestä ensimmäisen kirjaimen:

```
10 INPUT "ETUNIMI ";E$
20 INPUT "SUKUNIMI ";S$
30 PRINT USING "!";E$;
40 PRINT ". ";S$
50 GOTO 10
```

Jos syötettäisiin tiedot "HARRI NIEMINEN", "RETU VIRTANEN" ja "EERO VIITANEN", saataisiin ohjelman tuloksena:

H. NIEMINEN
E. VIITANEN
R. VIRTANEN

Toinen jonon muotoilumerkeistä on &. Se määritetään hieman toisin:

```
10 PRINT USING "&&";"RETU";"EERO"
20 PRINT USING "& &";"RETU";"EERO"
```

Tuloste on silloin tämä:

```
RETUEERO
RETU EERO
```

Tämä muotoilija tulostaa jonot ja yhtä monta merkkiä kuin niiden väliin jätetään. Siksipä edelliselle riville tulostui "RETUEERO". Jälkimmäisellä rivillä oli jätetty merkin verran väliä &-merkkien väliin, joten nimien "RETU" ja "EERO" väliin jäi vielä tyhjä tila.

Seuraava ohjelma vastaanottaa syötetietona nimen ja tulostaa sen viestinä:

```
10 INPUT "MIKÄ SINUN NIMESI ON";N$
20 PRINT USING "TERVE &, MITÄ KUULUU"
;N$
```

Jos kirjoitat "RETU VIRTANEN", tietokone vastaa:

"TERVE RETU VIRTANEN MITÄ KUULUU".

Luvuille on myös erityismuotoilijat. Ne ovat #, +, —, **, \\, **\ ja ^^^^. Eräistä on tosin harvoin hyötyä, mutta vilkaisemme joka tapauksessa niiden vaikutuksia.

#-merkki ilmaisee numeroa. Jos haluaa tulostaa lukusarjoja hyvin tyylikkäästi, tällä muotoilumerkillä on käyttöä. Kirjoitapa esimerkiksi:

```
10 PRINT USING "###.##";1.646,134.5,.
45,6.91
```

Ohjelma antaa seuraavan tulosteen:

```
 1.65 134.50 0.45 6.91
```

Tarvittaessa luvut pyöristetään tällä muotoilumerkillä. Jos desimaalipiste on määritetty ja luvussa esiintyy piste, sen eteen ilmestyy aina 0. Jos luku on lyhyempi kuin PRINT USING -lause ilmoitti, luvun eteen jää vastaava määrä tyhjää tilaa.

Plus- ja miinusmerkit aiheuttavat luvun eteen tai jälkeen merkin + tai −.

```
10 PRINT USING "+###.##";12.86,-12.86
20 PRINT USING "###.##+";12.86,-12.86
30 PRINT USING "###.##-";12.86,-12.86
```

Tuloste näyttää silloin tällaiselta:

```
+12.86   −12.86
 12.86    12.86−
 12.86    12.86−
```

**-muotoilija jättää lukua edeltävien tyhjien tilojen kohdalle asteriskeja:

```
10 PRINT USING "**#.##";12.86
20 PRINT USING "**#.##";1.83
30 PRINT USING "**#.##";123.67
```

Tuon ohjelman tulosteet näyttävät seuraavilta:

```
*12.86
**1.83
123.67
```

Kirjoittimelle tulostettaessa käytetään aivan samoin toimivia käskyjä LPRINT USING ja LPRINT.

Tiedostojen syöttö ja tulostus

Tiedosto on pohjimmiltaan datakokoelma. MSX-BASIC tarjoaa käyttäjälle monia erityiskäskyjä ja muuttujia tiedoston käsittelyyn. Lisälaitteita on neljää tyyppiä. Vain yhtä voi käyttää sekä tiedon tallennukseen että hakuun: kasettinauhuria. Muut kolme laitetta soveltuvat ainoastaan tulostukseen. Seuraava taulukko sisältää laitevaihtoehdot, niiden nimet (tunnisteet) MSX-BASICissa ja käyttötavat:

Laitteen nimi	**Laitetunniste**	**Syöttö/Tulostus**
Kasettinauhuri	CAS:	Syöttö ja tulostus
Rivikirjoitin	LPT:	Tulostus
TV/monitori	CRT:	Tulostus
Grafiikkaruutu	GRP:	Tulostus

Keskeisimpänä laitteena pidetään kasettinauhuria, sillä se on välttämätön ohjelmien tallennuksessa. Sitä paitsi se on jo monessa kodissa entuudestaan.

MSX-BASICin tiedonkäsittelyä varten täytyy ensin **avata** tiedosto. Seuraava ohjelma pyytää käyttäjältä vain kolme sanaa tallentaakseen ne nauhalle:

```
10 OPEN "CAS:RETU" FOR OUTPUT AS #1
20 INPUT A$
30 INPUT B$
40 INPUT C$
50 PRINT #1,A$
60 PRINT #1,B$
70 PRINT #1,C$
80 CLOSE #1
```

Rivi 10 avaa kasetilta tiedoston "RETU" ja ilmoittaa, että sitä käytetään ohjelman tulostetiedostona. #-merkin edessä oleva luku on **tiedostonumero.** Tiedoston numero #1 viittaa koko ohjelman ajan "RETU"-tiedostoon. Kun rivi 10 on suoritettu, käyttäjää pyydetään painamaan nauhurin play- ja record-näppäintä.

Rivi 30 hoitaa tulostamisen tiedostoon. PRINT # -lause ainoastaan hakee merkkijonon nauhalta. Lause on lähes sama kuin tavallinen PRINT. MSX-BASICista löytyy lisäksi PRINT # USING. Ohjelman viimeinen lause sulkee tiedoston. Jos tiedoston numeroa ei ilmoiteta, kaikki avoimet tiedostot sulkeutuvat. END-lauseella on sama vaikutus. Tiedostoa sulkiessaan ohjelma kirjoittaa nauhaan merkin, joka osoittaa tiedoston päättymisen. Merkki vastaa CTRL- ja Z-näppäimen yhtaikaista painamista.

Koska olet näin tallentanut merkkijonon nauhalle, haluat ehkä löytää sen vielä joskus. Seuraava ohjelma sijoittaa merkkijonoja RETU-nauhatiedostosta:

```
10 OPEN "CAS:RETU" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 60
30 INPUT #1,A$
40 PRINT A$
50 GOTO 20
60 CLOSE #1
```

OPENia muutetaan siten, että data on tällä kertaa **syöttötietona. EOF** on MSX-BASICin erityismuuttujia. Kun tiedosto luetaan nauhalta, CTRL-Z-merkki havaitaan, jolloin muuttuja EOF saa arvon —1. EOF **(End-Of-File)** on **tiedostonloppu**-muuttuja. Jos ohjelma ei totea EOF:n arvon muuttuneen —1:ksi, saadaan virheilmoitus "Input Past End".

Noissa yksinkertaisissa esimerkeissä käytettiin kahta kolmesta tiedostokäskystä. INPUT- ja OUTPUT-käskyjen lisäksi käytettävissä on APPEND. Jos tiedoston avaa APPENDilla, tiedostoa luetaan nauhalta kunnes CTRL-Z osuu kohdalle, minkä jälkeen uutta tietoa voidaan lisätä tiedoston loppuun.

Seuraava hyvin helppo ohjelma on tarkoitettu nimien ja puhelinnumerojen käsittelyyn, jotta nimen avulla löytyisi oikea puhelinnumero.

```
10 CLS
20 PRINT "1. TIEDOSTON MUODOSTAMINEN"
30 PRINT "2. TIEDONHAKU"
40 PRINT:PRINT
50 INPUT "VALITSE (1 TAI 2)";A$
55 PRINT "SYÖTÄ NAUHALUKEMA":INPUT "N
RO : ";C
60 IF A$="1" THEN GOTO 100
70 IF A$="2" THEN GOTO 240
80 BEEP:GOTO 50
90 REM LAADI/UUSI TIEDOSTO
100 OPEN "CAS:PUH" FOR OUTPUT AS #1
110 CLS
120 PRINT "TIEDOSTO VALMIS"
130 PRINT "--------- -------"
140 FOR I=1 TO 800:NEXT I
150 CLS
160 PRINT "POISTU KIRJOITTAMALLA NIME
N SIJASTA *"
170 PRINT
180 INPUT "KIRJOITA NIMI : ";N$
190 IF N$="*"THEN PRINT "LAADINTA PÄÄ
TTYNYT": PRINT "KELAA NAUHA TAKAISIN
KOHTAAN : ";C:GOTO 340
200 INPUT "KIRJOITA NUMERO : ";P$
210 PRINT:INPUT "ONKO HYVÄ? (K/E) ";V
$
220 IF V$="K" THEN PRINT #1,N$,P$
230 GOTO 150
240 REM TIEDONHAKU
250 OPEN "CAS:PUH" FOR INPUT AS #1
260 CLS:PRINT "ALOITA NAUHA KOHDASTA
";C
270 INPUT "KIRJOITA NIMI: ";N$
280 PRINT "HAEN ";N$;"N NUMEROA"
290 IF EOF(1) THEN 330
300 INPUT #1,A$,B$
305 PRINT A$,B$
310 IF N$=A$ THEN PRINT "NUMERO ON ";
B$:F=1
320 GOTO 290
330 IF F=0 THEN PRINT "LUETTELOSSA EI
 OLE NUMEROA NIMELLÄ   ";N$
```

```
340 CLOSE #1
350 END
```

Kasettitiedostojen ongelmia ovat menetelmän uskomaton hitaus ja melkoinen joustamattomuus. MSX-DOS tarjoaa tiedoston käsittelyyn hyvin paljon nopeampia ja käyttökelpoisempia keinoja kuin BASIC ottaen täyden hyödyn levykkeiden nopeudesta ja tallennustilasta.

Tarkista liite A:sta muut muotoilutoiminnot, kuten SPACE$ ja TAB.

Seuraava luku käsittelee tähänastista eloisampia ohjelmoinnin piirteitä, MSX-tietokoneen musiikillisia ominaisuuksia.

MSX:n musiikki ja sointi

Tietokoneella voi tuottaa soivia ääniä monin tavoin. Keskuslaitteiden ja minitietokoneiden parissa työskentelevillä ohjelmoijilla oli vuosia sitten tapana luoda musiikkia ovelasti rivikirjoittimia käyttäen. He olivat huomanneet, että kun kirjoittimelle lähettää merkkejä tiettyinä sarjoina, saa tulostimen äänenkorkeuteen pieniä muutoksia, joita voi kutsua säveliksi. Tuon epätavallisen keinon avulla saattoi esittää joitakin nokkelia vaikkei aivan korviahiveleviä säveliä.

Kotimikrojen saavuttua aikoinaan markkinoille säveliä saattoi soittaa lähettämällä sähkösykäyksiä kaiuttimiin. Jos sykkeet lähetetään tarpeeksi nopeasti, syntyy sävelmä. Yleensä BASICin POKE valvoo soittoa. POKE asettaa lukuja muistipaikkaan, josta assembler-tason sykäys sitten lähtee hyvin nopeasti kaiuttimeen tuottamaan tunnistettavan sävelen.

MSX-mikroilla on vieläkin kehittyneemmät keinot äänien synnyttämiseen. Tarkoitusta varten on oma sirunsa, johon pääsee kahden lauseen ja yhden käskyn avulla: SOUND, PLAY ja BEEP. Siru kykenee tuottamaan kolme eri ääntä samanaikaisesti, ja valittavana on kaikenlaisia ääniä pelien tutuista kilahteluista,

rysähdyksistä ja vinkunoista aina säveliin ja sointuihin asti.

BEEP-käskyyn olemmekin jo tutustuneet. Muistin virkistykseksi voit toki kirjoittaa vielä kerran: BEEP. Todella käyttökelpoisia lauseita ovat SOUND ja PLAY.

SOUND on noista kahdesta vaikeampi oppia, mutta sen avulla tuotettavien äänityyppien käyttö on joustavampaa. SOUND lähettää lukusarjan suoraan äänisirulle. Sirussa on 13 **rekisteriä,** jotka kaikki ovat käytettävissäsi. Kuhunkin rekisteriin tallennetaan luku, joka valvoo sirun tiettyä toimintoa. Toiminnot ja niitä vastaavat numerot löytyvät seuraavasta taulukosta. Kohteena olevan rekisterin ja annetun lukeman arvon mukaan tapahtuu sitten tiettyjä asioita. Sekavaako? No, asia valottunee hieman tutustuttaessa MSX-äänisirun **ohjelmointimalliin,** General Instrumentsin AY-3-8910:een!

Rekisterin nro	Toiminto
0	Valvoo kanava A:n sävelkorkeuden hienovirityksen. Kaikki 8 bittiä ovat käytettävissä, joten arvo voi olla 0 – 255.
1	Valvoo kanava A:n sävelkorkeuden raakavirityksen. Tähän on käytettävissä 4 bittiä, joten sijoitettava arvo voi olla enintään 15.
2	Kanava B:n hienovirityS
3	Kanava B:n raakavirityS
4	Kanava C:n hienovirityS
5	Kanava C:n raakavirityS
6	Tämä moduloi kohinakanavan. Käytettävissä on 5 bittiä, joten arvo voi olla 0 – 63.
7	Tämä rekisteri voi muuttaa äänikanavan tulosteen sävelestä sihinää muistuttavaksi. Enintään 7 arvoa on käytettävissä sävelen tuottamiseksi tällä kanavalla. Sen ylittävät arvot

	aiheuttavat kohinaa kolmelta äänikanavalta.
8	Säätää kanava A:n äänenvoimakkuuden.
9	Säätää kanava B:n äänenvoimakkuuden
10	Säätää kanava C:n äänenvoimakkuuden
11	8 bitin hienosävel
12	8 bitin raakasävel
13	Vaipansäädin

Äänirekisterit

Laitteiston osan ohjelmointimalli osoittaa ohjelmoijalle, mitä kukin sirun rekisteri valvoo. Se käy selville seuraavan ohjelman avulla.

```
10 REM ÄÄNINÄYTE
20 REM VIRITÄ ÄÄNIKANAVA A. ASETA RAA
KAVIRITYS 1 JA HIENOVIRITYS 0
30 SOUND 1,0
40 REM AVAA KANAVA A:N ÄÄNI
50 SOUND 8,5
60 END
```

Kun käytät ohjelmaa, tulet havaitsemaan kaksi seikkaa. Ensiksi televisiosi kaiuttimesta kajahtaa karmeaa ulinaa, ja toiseksi ruutuun ilmestyy "OK", mikä osoittaa ohjelman suorituksen päättyneen. "Vai niin!" kuuluu joku hämmästelevän. "Miksi sitten tuo karmea ulina ei päättynyt yhtaikaa ohjelman kanssa?" Hermojanne raastava meteli jatkuu, koska äänisirulle on kerrottu ohjelmassa, että sen tulee ulista, mutta lopettamisesta ei ole puhuttu mitään. Ulinasta pääsee eroon kahdella tavalla. Voi kirjoittaa "BEEP" – sillä äänisirua käyttäessään BEEP muuttaa arvot ja katkaisee äänen tuottamisen.

Tämä mahdollinen päänsäryn aihe poistuu myös, kun kirjoitetaan SOUND 8,0. Tällöin kanava A:n äänenvoi-

makkuus laskee nollaan. No, kuinka ohjelma sitten synnytti tuon kauhean äänen? Asettamalla sekä arvon 0 rekisteriin 0 että arvon 1 rekisteriin 1 ohjelma sääti kanava A:n sävelkorkeuden. Asettamalla arvon 5 rekisteriin 8 ohjelma kohotti kanava A:n äänenvoimakkuutta. Kokeilepa seuraavaa ohjelmaa:

```
10 SOUND 0,1:SOUND 8,5
20 FOR I=1 TO 14:FOR J=1 TO 100:NEXT
J
30 SOUND 1,I
40 NEXT I
50 SOUND 8,0
60 END
```

Tämä ohjelma tuottaa 14 erikorkuista säveltä. Huomasithan, että katkaisimme äänen ohjelman lopussa.

Ja sitten härkää sarvista: avataan kaikki äänikanavat samanaikaisesti. Näin syntyy sointu:

```
10 SOUND 0,1:SOUND 1,1
20 SOUND 2,1:SOUND 3,3
30 SOUND 4,1:SOUND 5,6
40 FOR I=8 TO 10
50 SOUND I,5
60 NEXT I
```

Tässä olivatkin sävelten soittamisen perusmuodot. SOUNDin kiehtovin piirre on useiden erikoisten sointien synnyttäminen. Niitä saa helpoiten aikaan käyttämällä hienoviritysrekistereitä.

SOUND-ohjelmien näytteitä

Äänisirun keinot käyvät parhaiten selville harjoittelemalla. Siksi olemme valmistaneet joukon ohjelmia. Niihin kannattaa todella tehdä itse muutoksia, jotta niiden vaikutukset tulisivat tutuiksi.

Tässä jaksossa hairahdumme outojen äänten aiheuttamiseen...

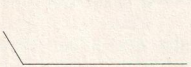
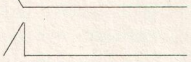
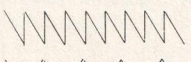
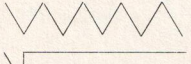
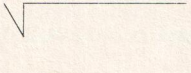
```
10 REM OUTOJA AANIA
20 INPUT N
30 SOUND 0,1:SOUND 1,1:SOUND 8,5
40 FOR I=255 TO 0 STEP -N
50 SOUND 0,I
60 NEXT I
70 GOTO 40
```

Annetaanpa N:lle vaikka sellaisia arvoja kuin 5, 15, .5 jne. ja odotellaan mitä tapahtuu.

Kohinakanavan mahdollisuudet ovat valtavat. Rekisteriä 7 käytetään kohinakanavan aukaisuun. Seuraavien kahden ohjelman tarkoituksena on synnyttää helikopterin ääntä muistuttava ääni.

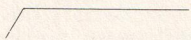
```
10 SOUND 7,5
20 SOUND 8,7
30 FOR I=63 TO 1 STEP -1
40 SOUND 6,I
50 NEXT I
60 GOTO 30
```

Tässä toinen helikopterin möräkkä moottorin ulinoineen.

S:n arvo	**Tuotettu vaippa**
0,1,2,3,9	
4,5,6,7,15	
8	
10	
11	

12

13



14

"S"-käskyn tarjoamat vaippamallit

```
10 SOUND 0,20:SOUND 1,0:SOUND 2,30
20 SOUND 3,0:SOUND 4,0:SOUND 5,9
30 SOUND 6,0:SOUND 7,48:SOUND 8,16
40 SOUND 9,4:SOUND 10,6:SOUND 11,100
50 SOUND 12,2:SOUND 13,10
60 GOTO 30
```

Pikku ohjelmamme käyttää hyväkseen äänigeneraattorin vaipanmuodostuskeinoja. Vaippa määrittää tuotettavan aallon muodon. Siihen on käytettävissä 8 eri muotoa. Edellä esitetty taulukko osoittaa nuo muodot ja rekisteri 13:lle annettavat arvot.

SOUND-lausettakin täytyy kokeilla aikansa ennen kuin siihen tottuu. Jos haluat musiikkia, SOUNDin tarjoamat mahdollisuudet ovat liian aikaavieviä ja edellyttävät ajattelemaan lukuja nuottien sijasta. Eikö olisikin mutkattominta vain pyytää tietokonetta soittamaan jokin sävel, kuten E tai C?

Musiikkimakrokieli

PLAY-käskyn avulla voit sitten pyytää konetta soittamaan jotain säveltä ja esittämään paljon muutakin. Korkeuden lisäksi voidaan määritellä niiden kesto, aaltomuoto ja äänenvoimakkuus. Äänisirun ohjeet annetaan merkkijonoina. Seuraava helppo lause saa kuuluviin soinnun.

```
PLAY "C", "A", "E"
```

Kanava A soittaa C:n, kanava B A:n ja kanava C E:n.

Sävelten korkeusala C:stä C:hen on 8 oktaavia. Merkkijonoihin sisältyy PLAY-käskyn tunnistamia ohjeita. Tästä muodostuu BASICin sisäinen pienoiskieli, **musiikkimakrokieli (Music Macro Language),** josta käytetään lyhennystä **MML** ja jonka ohjeet ovat:

MML-käsky	Toiminto
A to **G**	Soittaa sävelen annetussa oktaavissa. Voidaan myös käyttää merkkejä # tai +, jotka ovat korotusmerkkejä, – taas on alennusmerkki. Näitä vaihtoehtoja voi kuitenkin käyttää ainoastaan pianon mustia koskettimia vastaavissa sävelissä. Niinpä B ei ole hyväksyttävä sävel.
O<n>	Määrittää oktaavin, ja n:lle voidaan antaa arvo 1 – 8. Jokainen oktaavi alkaa C:stä ja päättyy B:hen, ja MML olettaa oktaavin olevan 4, jollei O-ohjetta käytetä.
N<n>	Tarjoaa vaihtoehdon sävelille ja oktaaveille, ja n:lle voidaan antaa arvo 0 – 96, jolloin 0 tarkoittaa lepotilaa, 1 alimman oktaavin C:tä ja niin edelleen.
L<n>	Säätää sävelen kestoksi 1/n, ja siis: **L1** kokonuotti **L2** puolinuotti **L3** yksi triolin osa (1/3 neljän iskun tahdissa) **L4** neljäsosanuotti **L5** kvintoli, viidesosan yksi osa (1/5 4/4-tahdista) **L6** yksi osa neljäsosanuoteista koostuvasta triolista **L64** 1/64 nuotti Keston jälkeen voi kirjoittaa myös nuotin, jos haluaa vain yhden nuotin muuttuvan. Siis L2A ja A2 tarkoittavat yhtä ja samaa asiaa.
R<n>	Säätää paussin eli tauon keston, ja

n:lle voidaan antaa arvo 1 – 64 ja se toimii aivan samoin kuin L:n tapauksessa. Oletusarvo on 4. A. nuotin jäljessä saa sävelen soimaan pidennettynä eli tavalliseen kestoonsa nähden 3/2-kertaisena. Nuotin jälkeen voidaan kirjoittaa monta pistettä, ja kesto asettuu sen mukaisesti: esimerkiksi A... soi 27/8 kertaa normaalia pitempään. Tauon jälkeen voidaan myös kirjoittaa pisteitä, jolloin sen pituus säätyy vastaavalla tavalla.

T<n> Säätää sävelmän tempon määrittämällä, kuinka monta neljäsosanuottia tulee soida minuutissa, jolloin n:lle voidaan antaa arvo 32 – 255. Oletusarvo on 120.

V<n> Säätää äänenvoimakkuuden, jolloin n:lle voidaan antaa arvo 0 – 15. Oletusarvo on 8.

M<n> Säätää aallon keston, jolloin n:lle voidaan antaa arvo 1 – 65535. Oletusarvo on 255.

S<n> Säätää aallon muodon, jolloin n:lle voidaan antaa arvo 1 – 15. Oletusarvo on 1.

X<merkkimuuttuja> suorittaa tietyn ketjun säveliä, taukoja jne.

MML-käskyissä n:n arvo voi olla joko muuttuja tai vakio. Jos käytetään muuttujia, ne täytyy sijoittaa käskyjonossa. Kokeilepa esimerkiksi seuraavaa lyhyttä ohjelmaa, joka toimii arvoilla 1 – 8:

```
10 INPUT X
20 PLAY "O=X;C"
30 GOTO 10
```

Huomasithan puolipisteen muuttujan nimen jäljessä. Jos sitä ei muista, BASIC ilmoittaa "ILLEGAL FUNC-

TION CALL". Ohjelma soittaa C:n haluamassasi oktaavissa.

Tässä pikku melodia, jonka tahdissa voit pyöritellä peukaloitasi.

```
10 PLAY "L405AEC04","L403AEBA","L8AEC
03A04AEC03A"
20 PLAY "L405BEC04G+","L402G+","L803G
+04CEBBEC03G+"
30 PLAY "L406C05EC04G","L402G","L804G
05CE06CC05EC04G"
40 PLAY "L405F+D04AD","L403D","L805F+
D04AD05F+D04AD"
50 GOTO 10
```

Kaikki kolme ääntä otettiin siis leikkiin mukaan. Sävelten kesto on myös yksilöity. Seuraava ohjelma tuottaa näennäisesti satunnaisia säveliä **RND**-toiminnon avulla.

```
10 REM NAENNAISSATUNNAISIA SAVELIA TU
OTTAVA OHJELMA
20 PITCH=INT(RND(10)*100)
30 IF (PITCH>96) THEN GOTO 10 ELSE PL
AY "S8M1500L24N=PITCH;":GOTO 20
```

RND on kuin onnenpyörä. Kun sitä käytetään, tuotetaan jokin arvo 0 – 1. Ohjelma on nimetty näennäissatunnaiseksi, koska se tuottaa joka käytön aikana aina saman jakson satunnaislukuja. Tällä kertaa S- ja M-ohjeita käytetään PLAY-jonossa. S määrittää tuotettavan ääniaallon tyypin ja M määrittää, kuinka kauan aalto muokkaa soitettavia säveliä.

Seuraava ohjelma vuorostaan käyttää kaikkia kolmea ääntä tuottaessaan satunnaissointuja. Sauvaohjainta käytetään nopeuttamaan sävelten esittämistä lyhentämällä ja pidentämällä säveliä.

```
10 REM NAENNAISSATUNNAISIA SAVELIA TU
OTTAVA KOLMIAANINEN NOPEUSSAADETTY OH
JELMA
```

```
20 REM OLETETTU SÄVELEN KESTO
30 L=12
40 REM SAUVAOHJAINVALINNAT
50 IF STICK(0)=1 THEN L=L+1:IF L>64 T
HEN L=64
60 IF STICK(0)=5 THEN L=L-1:IF L<1 TH
EN L=1
70 REM LASKE SATUNNAISLUVUT JA VASTIN
EET ÄÄNILLE KAKSI JA KOLME
80 P1=INT(RND(10)*100):IF (P1>86) OR
(P1<11) THEN GOTO 80 ELSE P2=P1-10:P3
=P1+10
90 REM SOITA SATUNNAISSOINTU
100 PLAY "S8M1500L=L;N=P1;","S8M1500L
=L;N=P2;","S8M1500L=L;N=P3;"
110 GOTO 50
```

Tämän jakson viimeinen ohjelma käyttää X-ohjetta. Ohjelman alussa esitellään kolme merkkimuuttujaa, jotka sitten käsitellään X-ohjeen avulla.

X mahdollistaa musiikkijonon suorittamisen PLAY-lauseessa. Niinpä jos kappaleessa toistuu sävelsarjoja, niitä ei tarvitse kirjoittaa joka kerta erikseen. Oletetaan, että haluttaisiin toistaa sävelsarja "04BGB05CEG", jolloin tuosta jonosta tehtäisiin muuttuja A$, jota kutsuttaisiin sitten PLAY-lauseissa X-ohjeella. Lyhyt ohjelmamme osoittaa, kuinka X-ohjetta käytetään.

```
10 A$="O4BG05CEG"
20 PLAY "XA$;"
30 END
```

Puolipiste on merkkimuuttujan jäljessä välttämätön.

```
10 A$="L2405BGBO6CEGO5BGBO6CEGO5BGBO6
CEG"
20 B$="L2404BGBO5CEGO4BGBO5CEGO4BGBO5
CEG"
30 C$="L2403BGBO5CEGO4BGBO5CEGO4BGBO4
CEG"
40 FOR I= 1 TO 4
```

```
50 PLAY "V4XA$;","V4XB$;","V4XC$;"
60 NEXT I
70 FOR I= 1 TO 2
80 PLAY "V10L103D;","V9L24XA$;","V10L
102A"
90 PLAY "O3D",A$,"O2G"
100 PLAY "O3C",A$,"O2E"
110 PLAY "O3C",A$,"O2A"
120 NEXT I
130 FOR I=1 TO 2
140 PLAY "V12L12O3DACO2GO3GA","V10L20
4C","V10L205C"
160 PLAY "L2404CDEFGBO5CDEFGB","O3G",
"O5G"
170 PLAY "O4BGBO5CEGO4BGBO5CEG","O3F"
,"O5F"
180 PLAY "SM15000O3CCCCCCO5CCCCCC","O4
C","O5C"
190 PLAY "L404DG","O3B","L205D"
200 PLAY "L12CDEFGB","L806DCO5AF","O5
G"
210 PLAY "O4BGBO5CEG","SM15000L2407CC
CCGGGGDDDD","F"
220 NEXT I
230 PLAY "L2404CDEFGBO5CDEFGB"
240 PLAY "L2403CDEFGBO4CDEFGB"
250 PLAY "L2402CDEFGBO3CDEFGB"
260 PLAY "V10L12O3G","V10L12O4B","V10
L1204D"
270 PLAY "O3A","O4E","O5C"
280 END
```

Seuraavan ohjelman avulla yritämme esittää "hallelujakuoron". Rivien 120 – 180 nuottisarjat ovat kuultavissa kaiutettuina.

```
10 A$ = "R8L40T255DR24DR24T120L8ER24D
20 B$ = "T120R18L12DR24ER24D"
30 C$ = "R8L6O3GR18AR18BO4L6CR2403GO4
C"
40 PLAY "T120L6G.","T120L6G.","T120L6
G."
50 PLAY B$,"R18XB$;","R16XB$;"
```

```
60 PLAY A$,"R18XA$;",A$
70 PLAY A$,"R18XA$;",A$
80 PLAY "R6","R6","R6"
90 PLAY "T120L6G.","T120L6G.","T120L6
G."
100 PLAY B$,"R18XB$;","R16XB$;"
110 PLAY A$,"R18XA$;",A$
120 PLAY A$,"R18XA$;",A$
130 PLAY C$,"R18XC$;",C$
140 PLAY A$,"R18XA$;",A$
150 PLAY A$,"R18XA$;",A$
160 PLAY C$,"R18XC$;",C$
170 PLAY A$,"R18XA$;",A$
180 PLAY A$,"R18XA$;",A$
```

Seuraava musiikkinäyte on yksinkertainen yritelmä 12-tahdin blues-soinnuttelusta C:ssä.

```
10 FOR I=1 TO 4
20 PLAY "L1203D","O4F+","O4A"
30 PLAY "O3D","O4G","O4B"
40 PLAY "O3D","O4A","O4C"
50 PLAY "O3D","O4G","O4B"
60 NEXT I
70 FOR I=1 TO 2
80 PLAY "O3G","O4B","O5D"
90 PLAY "O3G","O5C","O5E"
100 PLAY "O3G","O5D","O5F"
110 PLAY "O3G","O5C","O5E"
120 NEXT I
130 FOR I=1 TO 2
140 PLAY "O3D","O4F+","O4A"
150 PLAY "O3D","O4G","O4B"
160 PLAY "O3D","O4A","O4C"
170 PLAY "O3D","O4G","O4B"
180 NEXT I
190 FOR I=1 TO 2
200 FOR J=1 TO 4
210 PLAY "L1205EC+O4A","O6E","O5C+"
220 NEXT J
231 FOR J=1 TO 4
240 PLAY "O5DO4BG","O6D","O5B"
250 NEXT J
```

```
260 NEXT I
270 PLAY "O3D","O4F+","O4A"
280 PLAY "O3D","O4G","O4B"
290 PLAY "O3D","O4A","O4C"
300 PLAY "O3D","O4G","O4B"
```

MML:n viehätys perustuu siihen, että se muistuttaa melkoisesti yleistä musiikin merkintätapaa. Musiikkia opiskelleet saavat pian tuntuman siihen, ja muutkin havaitsevat musiikin opiskelun varsin helpoksi.

MSX-BASICin grafiikka

Tietokoneen kiehtovimpia ominaisuuksia on piirustustaito. Ohjelmoijat ympäri maailmaa ovat viehättyneet piirrosmenetelmään, jossa kuva saadaan aikaan asettelemalla tiedoston teksti halutun kuvan näköiseksi. Tietokoneasennusten yhteydessä on lähes kaikkialla sitten tuotettukin silmäniloksi Ressu-koiran kuvia. Kun kirjoitin tulostaa moneen kertaan samalle riville (overprint), saadaan aikaan sävyeroja. Tällaisen tietokonetaiteilun kuuluisin teos on varmaankin kaikille tuttu versio Leonardo da Vincin *Mona Lisasta*. Jos käytettävissäsi on kirjoitin, voit kokeilla tällaistakin piirtelyä. Kirjoitin on silloin kuitenkin melkoisessa rääkissä! Kuvia voi tuottaa toisinkin, hyvin yksinkertaisesti.

Kaikissa MSX-tietokoneissa on ruudun grafiikkaesitykseen erikoistunut siru, jonka yhteyteen pääsee monin tavoin. Dataa voi esimerkiksi lähettää näyttösuoritukseen suoraan VPOKElla (ks. lukua 2). Se on kuitenkin yleensä melko hidasta eikä siis sovi kärsimättömille ohjelmoijille. MSX-BASICiin sisällytetyt grafiikkakäskyt mahdollistavat paljon näppärämmän piirtämisen kuin VPOKE.

Grafiikkatilojahan oli kaksi: pien- ja suurresoluutio.

Tässä luvussa keskitytään noista kahdesta tarkemmin erottelevaan ruutuun 2. Ruutu 2 koostuu yhteensä 49152 pienestä pikselistä. Ruudun erottelukyky perustuu 256 vaaka- ja 192 pystypikseliin. Nuo pikselipisteet säätyvät suurresoluutiotilan alussa kaikki samanvärisiksi eli taustan värisiksi.

PSET- ja PRESET-grafiikka

PSET ja PRESET ovat yksinkertaisimmat grafiikkakäskyt. PSET virittää pikselin ja PRESET poistaa sen. Pikselin väri voidaan määrittää PSETin värivaihtoehtojen ja PRESET-käskyjen avulla. Seuraava lyhyt ohjelmalistaus virittää kaikki pikselit.

```
10 SCREEN 2
20 FOR I=0 TO 256
30 FOR J=0 TO 192
40 PSET(I,J),15
50 NEXT J
60 NEXT I
```

Jokainen pikseli muuttuu valkoiseksi (väri 15). Kaikkien pikselien asemahan määriytyi kahden koordinaatin mukaan. PSETiä ja PRESETiä käytettäessä voi virittää yksittäisiä pikseleitä ja siten vetää viivoja tai piirtää mitä tahansa kuvioita. Viritettävät pikselit voidaan rajata helposti sauvaohjaimen avulla. Seuraava ohjelmamme osoittaa, kuinka yksittäisiä pikseleitä voidaan sauvaohjaimen avulla virittää ruutuun kuin kynällä piirtäen.

```
10 REM PIIRUSTELUOHJELMA
20 SCREEN 2
30 REM ILMOITA ALKUKOORDINAATIT
40 X=128 : Y=96
50 STRIG(0) ON
60 D = 1 : E = 0
70 ON STRIG GOSUB 200
80 IF D=1 THEN PSET (X,Y),15 ELSE LOC
ATE X,Y
```

```
90 REM VALINTA SAUVAOHJAIMELLA
100 IF STICK(0)= 1 THEN Y=Y-1
105 REM 6-158
110 IF STICK(0)=2 THEN Y=Y-1:X=X+1
120 IF STICK(0)=3 THEN X=X+1
130 IF STICK(0)=4 THEN Y=Y+1:X=X+1
140 IF STICK(0)=5 THEN Y=Y+1
150 IF STICK(0)=6 THEN Y=Y+1:X=X-1
160 IF STICK(0)=7 THEN X=X-1
170 IF STICK(0)=8 THEN Y=Y-1:X=X-1
180 GOTO 80
200 REM VAIHDA D:N ARVO
210 SWAP D,E:RETURN
```

Muuttujan D avulla ruudussa voidaan siirtyä paikasta toiseen jälkeä jättämättä. Kun välilyöntinäppäintä painetaan, D:n arvo vaihtuu SWAP-lauseella 0:sta 1:een tai 1:stä 0:aan. Jos D:n arvo on 1, pikselit syttyvät, mutta jos arvo on 0, pikselit eivät syty vaan kohdistin siirtyy.

PSETin ja PRESETin esittelyn jälkeen olemmekin valmiit laatimaan ensimmäisen "animaatio-ohjelmamme". Ensin kuitenkin vilkaisemme liikkeen jäljittelemistä ruudussa.

Kaikkien animaatioiden pohjana on kuvan esittäminen, sen muuttaminen jollain tavoin ja uuden kuvan esittäminen niin nopeasti, että katsoja luulee näkevänsä yhtenäistä liikettä. PSETin ja PRESETin avulla mekin voimme jäljitellä liikettä.

Ensiksi valitaan ruudusta piste – esimerkiksi PSET (10,10). Sitten asetetaan sen viereinen piste ja alkuperäinen piste häivytetään samalla PRESETin avulla. Niinpä ohjelma virittää pisteen ja häivyttää sen naapurin. Tämä toistuu niin kauan kuin halutaan. Liikkeen jatkuminen hoituu FOR . . . NEXT-silmukalla.

```
10 SCREEN 2
20 FOR I=10 TO 250
30 PRESET (I-1,100)
```

```
40 PSET (I,100)
50 NEXT I
60 FOR I=250 TO 10 STEP -1
70 PRESET (I+1,100)
80 PSET (I,100)
90 NEXT I
100 GOTO 20
```

Tämä saa pisteen siirtymään ruudun vasemmasta reunasta oikeaan ja palaamaan sitten takaisin. Panithan merkille, että vasemmalle ja oikealla siirryttiin eri FOR... NEXT-silmukoilla. Katsopa vielä kerran myös PRESETin rakennetta. Siirryttäessä vasemmalta oikealle PSETin jälkeisen pisteen kohdalla tulee pisteen sammua; sen vuoksi PRESET on juuri I–1,100.

Tee tästäkin ohjelmasta kokeilemalla muunnelmia! Yritä saada tietokone antamaan äänimerkki suunnan vaihtuessa. Tai yritä saada piste siirtymään joka suunnanvaihdossa uudelle riville (siihen tarvitaan ehkä uusi muuttuja).

Myös LINE on näppärä grafiikkalause. Sen avulla voidaan määrittää alku- ja loppupisteen koordinaatit, joiden väliin LINE vetää yhdistävän viivan, esimerkiksi näin:

```
10 SCREEN 2
20 LINE (50,50)-(100,50),15
30 LINE (100,50)-(100,100),15
40 LINE (100,100)-(50,100),15
50 LINE (50,100)-(50,50),15
60 GOTO 60
```

Tällä ohjelmalla piirtyy valkoinen neliö. On vieläkin helpompi tapa piirtää laatikko vain yksirivisellä LINE-lauseella. Kuten hetki sitten havaitsit, ruutuun piirtyy vinoviiva, jos määrittää vasemman ylänurkan alkukoordinaatiksi ja oikean alanurkan loppukoordinaatiksi. Jos kuitenkin käytetään vaihtoehtoa B (2. luku), tulos on aivan toinen. Kirjoitapa se mukaan ja kokeile ohjelmaa:

```
10 SCREEN 2
20 LINE (20,20)-(40,40),,B
30 GOTO 30
```

LINE-lauseen loppuun sijoitettu B osoittaa, että haluatkin neliön etkä viivaa. LINEn avulla voidaan myös värittää neliö samalla kun se piirtyy. Silloin tulee käyttää B:n sijasta BF:ää lauseen lopussa. Se saa tietokoneen "maalaamaan" neliön.

Seuraava ohjelmamme piirtää neliöitä, joiden koko ja väri ovat sattumanvaraisia.

```
10 SCREEN 2
20 DEFINT A-Z
30 V=INT(RND(1)*256):W=INT(RND(1)*192
)
40 X=INT(RND(1)*192):Y=INT(RND(1)*192
)
50 C=INT(RND(1)*15)
60 LINE(V,W)-(X,Y),C,BF
70 GOTO 30
```

Se neliömäisistä aiheista. Voimme myös piirrellä ympyröitä, kuten jo kävi ilmi. Tässä muutamia ohjelmia, jotka piirtävät eräitä mielenkiintoisia kuvioita CIRCLE-lauseen avulla.

```
10 REM ornamenttia
20 REM SYöTTöTIEDOT
30 INPUT "X:N VÄLIT ";X
40 INPUT "Y:N VÄLIT ";Y
50 INPUT "HALKAISIJAN SUPISTUMINEN ";
Z
55 SCREEN 2
60 FOR I=1 TO 256 STEP X
70 FOR J=1 TO 192 STEP Y
80 FOR K=100 TO 5 STEP -Z
90 CIRCLE(I,J),K,15
100 NEXT K
110 NEXT J
120 NEXT I
130 GOTO 130
```

Sijoita aluksi vaikka 50, 30 ja 10. Ohjelman avulla syntyy mukavia kehäkuvioita. Tässä sitten toinen riipustelu:

```
10 REM YMPYRÄN PIIRUSTELUA
20 SCREEN 2
30 FOR I=1 TO 192
40 CIRCLE(128,I),I,15
50 NEXT I
60 GOTO 60
```

Voidaan tietysti käsitellä muitakin muotoja kuin ympyröitä. Ellipsi voidaan määritellä CIRCLEn mittasuhdevaihtoehtojen avulla, kuten jo olet havainnut. On vain määritettävä ympyrän korkeuden ja leveyden välinen suhde. Jos siis valitset luvun 1/3, saat tulokseksi litteän ellipsin, jos taas valitset luvun 3/2, ellipsi on korkea ja kapea. Seuraavassa ohjelmassa esitellään CIRCLEn käyttöä eri mittasuhteiden avulla:

```
10 SCREEN 2
20 FOR I=1 TO 32
30 CIRCLE(128,96),70,15,,,I/8
40 NEXT I
50 GOTO 50
```

CIRCLEn keinoihin kuuluu vielä kaaren piirtäminen. CIRCLE-lause tarvitsee silloin kaaren alku- ja loppukohdan radiaaneina. Kokeilepa seuraavaa:

```
10 SCREEN 2
20 CIRCLE(128,92),70,15,5,4
30 CIRCLE(128,92),50,15,4,3
40 CIRCLE(128,96),40,15,3,2
50 GOTO 50
```

Nähtäväksi piirtyy kolme vajaata ympyrää. Jokainen niistä on eri kohdasta vajaa. Syynä on kunkin alku- ja loppukohdan radiaanierot.

Kun olet piirtänyt ympyrän, ellipsin tai neliön, voit värittää sen PAINT-lauseen avulla. Ainoa ehto on se,

että suurresoluutiotilassa muodon maalaamiseen *täytyy* käyttää samaa väriä kuin sen piirtämisessä alunperin käytettiin. Tämä ohjelma piirtää ympyrän ja maalaa sen mustaksi.

```
10 SCREEN 2
20 CIRCLE(128,96),70,1
30 PAINT(128,96),1
40 GOTO 40
```

Yritä sitten piirtää valkoinen ympyrä mustan sijaan. Koko ruudun pitäisi pyyhkiytyä valkoiseksi. Tietokoneen näet tulisi lopettaa maalaaminen vasta kun kohdataan valkoinen rajaus. Pienresoluutiotilassa PAINT maalaa kaikki mahdolliset kuviot, onpa rajausväri mikä tahansa.

Väreistä puheen ollen, ruudun jokaisen pikselin väri selviää POINT-toiminnon avulla. POINT ilmoittaa värin, joka ruutukoordinaatilla määritetyllä pikselillä on. Se on myös näppärä apu helppojen pelien ohjelmoinnissa. Sillä voidaan esimerkiksi osoittaa ohjuksen osuminen maaliin. Kaikki maalit voidaan maalata tietyllä värillä, esimerkiksi mustalla. Kun ohjuksen koordinaatit vastaavat mustan pikselin koordinaatteja, ohjus on osunut maaliinsa. Seuraavassa yksinkertainen esimerkki POINTin käytöstä:

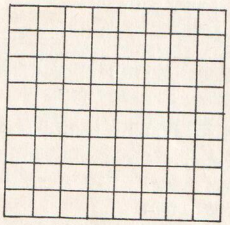
```
10 SCREEN 2
20 CIRCLE(50,50),25,15
30 CIRCLE(100,100),25,8
40 CIRCLE(150,150),25,1
50 PAINT(50,50),15:REM VALKOINEN
60 PAINT(100,100),8:REM PUNAINEN
70 PAINT(150,150),1:REM MUSTA
80 FOR I=1 TO 5000:NEXT I:REM HETKEN
VIIVE
90 X=POINT(50,50)
100 Y=POINT(100,100)
110 Z=POINT(150,150)
120 SCREEN 0
130 PRINT X,Y,Z
```

Spritet

Spritet ovat vekkuleita pikku grafiikkahahmoja, joita voi määritellä ja liikutella molemmissa grafiikkaruuduissa. Niiden olemassaolo ei näytä lainkaan riippuvan muista kuvioista, jotka on piirretty CIRCLEn, LINEn, PSETin tai jonkin muun avulla. Ne soveltuvat peleissä määrittämään ohjuksia, pommeja, avaruusaluksia ja kaikkia muitakin outoja hahmoja, joita piileksii useissa tietokonepeleissä – ja tuo soveltuvuus selviää sinullekin pian.

Ensiksi tulee kuitenkin hieman perehtyä sprite-sanastoon. Grafiikkaruutuihin voi suhtautua kuin pöytälevyyn, johon voi sommitella kuvia tavallisilla grafiikkakäskyillä. Tuota kutsutaan **nollatasoksi.** Kuvitellaanpa sitten sen ylle kerroksia, lisää tasoja graafisten hahmojen piirtämiseen. Lisätasot ovat spritejen käytettävissä. Jos sitten sijoitetaan yksi sprite nollatasolle ja toinen ykköstasolle, jälkimmäinen näyttää olevan edellisen yllä. Näppärää! Miltä nuo spritet näyttävät ja kuinka niitä määritetään?

Spritet koostuvat 8 binaariluvusta, ne ovat joukkoja joissa on yhteensä 64 kappaletta 1:iä ja 0:ia. Kun nuo luvut asetetaan 8 x 8 ruudukoksi, muodostuu kuvan 7 mukainen perusluonnosvihko, johon voi piirtää spriteja.



7 Spriten määrittelyruudukko.

Jos sitten tuollaiseen ruudukon neliöön sijoittaisi "1":n, tuo spriten osa syttyisi eli tulisi näkyviin ruutuun. Jos neliöön sijoitetaankin nolla, tuo spriten osa ei tule näkyviin ruutuun. Otetaan valmiin esimerkin avulla kunnolla selkoa asiasta. Seuraava lyhyt ohjelma asettaa säännöllisen neliön suurresoluutioruudun keskelle. Sprite näyttää kiinteältä neliöltä, sillä kaikki datalauseiden arvot ovat ykkösiä ja kukin 64 osasta "syttyy".

```
10 SCREEN 2,0,0
20 FOR I=1 TO 8
30 READ B$
40 SHAPE$=SHAPE$+CHR$(VAL("&B"+B$))
50 NEXT I
60 SPRITE$(0)=SHAPE$
70 PUT SPRITE 0,(128,96),15,0
80 GOTO 80
90 DATA 11111111
100 DATA 11111111
110 DATA 11111111
120 DATA 11111111
130 DATA 11111111
140 DATA 11111111
150 DATA 11111111
160 DATA 11111111
```

Rivin 60 SPRITE$ on hyvin erikoinen MSX-BASICin muuttuja. Sitä käytetään erityisesti spritejen määrittelyyn. FOR-silmukka ottaa DATA-lauseiden lukusarjat yksi kerrallaan ja liittää ne merkkimuuttujaan. Kuten varmaan huomasit, rivillä 40 tapahtuu melkoisesti muuttamista. Siinä **VAL**-toiminto muuttaa lukuja esittävät merkit numeerisiksi, esimerkiksi PRINT VAL("1232") antaisi yksinkertaisen tarkkuuden mukaisesti tuloksenaan luvun 1232.

Ensin rivi 30 lukee kunkin DATA-lauseen luvut merkkijonoksi, sitten rivi 40 liittää binaariluvun etuliitteen &B, muuttaa jonon varsinaiseksi binaariarvoksi ja muuttaa lopulta vielä koko binaariluvun merkkijonoksi CHR$( ) -toiminnon avulla. Kun FOR-lauseeseen on tuotu SHAPE$-muuttujan arvo, sitä voi käyttää erityisessä

SPRITE$-muuttujassa. Tuossa vaiheessa sprite täytyy numeroida. Valittavana on 32 numerointivaihtoehtoa 0 – 31. Valinnan mukaista numeroa käytetään sen jälkeen aina tuon spriten yhteydessä. Rivi 60 sijoittaa tunnusnumeron 0 juuri kasatulle spritekuviolle.

Kun sprite on määritetty, se varmaankin halutaan sijoittaa grafiikkaruutuihin. Sen voi tehdä PUT SPRITE -lauseella. Lause ei pelkästään määritä *mihin* kohtaan ruutua sprite sijoittuu, vaan se määrittää myös *mikä* sprite sijoitetaan ruutuun, *minkä värisenä* ja *mille tasolle*. Esittämässämme tapauksessa sprite 0 sijoitettiin valkoisena nollatasolle ruudun keskustaan. PUT SPRITE -lause voidaan kuvata seuraavasti:

Laita SPRITE tasoon numero XX asemaan (X,Y) YY:n värisenä, ja käyttöön otettavan SPRITEn numero on ZZ.

Ruutuun sijoittuminen määriytyy spriten vasemman yläkulman eikä suinkaan sen keskustan mukaan, joten tuo kulma tulee esimerkissämme ruudun keskelle. Tämä on jokseenkin tärkeää muistaa!

Jos yrität sijoittaa kaksi spritea samalle tasolle, tapahtuu melko mielenkiintoinen seikka. Spritet syttyvät ja sammuvat vuorotellen. Spritet välkkyvät, koska samalla tasolla voi kulloinkin olla vain yksi sprite. Näyttösiru sijoittaa yhden spriten kerrallaan tietylle tasolle. Jos siellä kohdataan toinen sprite, siru poistaa sen ja sijoittaa tilalle uuden. Seuraavaksi opetellaankin sitten liikuttelemaan spritea ruudussa.

Käytämme oppimaamme sauvaohjainmenetelmää spriten siirtelyyn. Ensimmäisen keskustassa on paikoillaan yksi sprite. Näet mustan ristin liikkuvan valkoisen ristin *yli*. Jos mustan ristin sijoittaa ensimmäiselle tasolle samalla kun valkoinen risti on nollatasolla, mustaa ristiä voi liikuttaa valkoisen alla.

Jos ohjelman datalauseita tarkastelee oikeassa valossa,

niin kykenee hämärästi erottamaan ykkösistä muodostuvan ristin. Ja sitten ohjelmaan:

```
10 SCREEN 2,0,0
20 FOR I=1 TO 8
30 READ B$
40 SHAPE$=SHAPE$+CHR$(VAL("&B"+B$))
50 NEXT I
60 SPRITE$(0)=SHAPE$
70 PUT SPRITE 0,(128,96),15,0
80 SPRITE$(1)=SHAPE$
90 PUT SPRITE 0,(128,96),15,0
100 X=128:Y=92
110 DIR=STICK(0):IF DIR=0 THEN 110
120 GOSUB 150
130 PUT SPRITE 1,(X,Y),1,0
140 GOTO 110
150 IF DIR=1 THEN Y=Y-1
160 IF DIR=2 THEN Y=Y-1:X=X+1
170 IF DIR=3 THEN X=X+1
180 IF DIR=4 THEN Y=Y+1:X=X+1
190 IF DIR=5 THEN Y=Y+1
200 IF DIR=6 THEN Y=Y+1:X=X-1
210 IF DIR=7 THEN X=X-1
220 IF DIR=8 THEN Y=Y-1:X=X-1
230 IF X>256 THEN X=256
240 IF Y<1 THEN Y=1
250 IF Y>192 THEN Y=192
270 RETURN
280 DATA 10000001
290 DATA 01000010
300 DATA 00100100
310 DATA 00011000
320 DATA 00011000
330 DATA 00100100
340 DATA 01000010
350 DATA 10000001
```

SCREEN-käskyyn sijoitetulla valinnalla voi spriteja suurentaa 16 x 16 pikselin kokoisiksi. Vaihda sitten rivi 10 muotoon SCREEN 2,1,0. Siten suurennat ohjelman ristejä.

On toki mahdollista määrittää erillisiä 16 x 16 pikselin spriteja ilman että tarvitsisi suurentaa 8 x 8 spriteja. Ensiksi, täytyy valita ruututila, joka hyväksyy 16 x 16 spritejen käytön. Sen saa aikaan kirjoittamalla SCREEN 2,2,0. Jos jotakuta ihmetyttää näppäinkilahduksen vaimentaminen SCREEN-käskyssä, niin syynä on yksinkertaisesti se, että sauvaohjainta reippaasti käytettäessä syntyvä meteli on äärimmäisen ärsyttävää!

16 x 16 spritejen määrittelyyn tarvitaan 32 DATA-lausetta. Ensimmäiset 16 määrittävät spriten vasemman puoliskon muodon ja toiset 16 oikean.

Muutoin asettelu tapahtuu jokseenkin samalla tavalla kuin 8 x 8 spriteilla. Spriteja voi suurentaa jopa 32 x 32 pikselin kokoisiksi käyttämällä käskyä SCREEN 2,3,0.

Sitten seuraakin muutamia sangen alkeellisia pelejä, joiden avulla pääset jyvälle asiasta. Ensimmäisessä pelissä käytetään 16 x 16 spritea avaruusaluksena, 8 x 8 spritea ohjuksena ja maaleina punaisia neliöitä. Peli jatkuu, kunnes osut maaleihin viisi kertaa. Se on sangen yksinkertainen peli, mutta antaa hyvän suunnitteluvalmiuden omien pelien laadintaan. Siihen on sisällytetty muutamia erikoisuuksia. Kun sauvaohjaimella ei liikuta mihinkään suuntaan, alus poukkoilee satunnaisesti.

```
10 REM DEMO PELI
20 DEFINT A-Z
25 SCREEN 2,2,0:COLOR 15,15,15:CLS
30 REM MUUTTUJIEN ALKUARVOT
35 ON STRIG GOSUB 860:REM POMMIN PUDO
TUS
40 SC=0:REM OSUMIEN LKM
50 X=128:Y=30:REM AVARUUSALUKSEN PAIK
KA ALUSSA
60 REM PIIRRÄ LAATIKOT
70 LINE(40,180)-(50,190),6,BF
80 LINE(80,180)-(90,190),6,BF
90 LINE(120,180)-(130,190),6,BF
```

```
100 LINE(160,180)-(170,190),6,BF
110 LINE(200,180)-(210,190),6,BF
115 LINE(0,96)-(256,96),1
120 REM ALUS-SPRITE
130 FOR I=1 TO 32
140 READ A$
150 S$=S$+CHR$(VAL("&B"+A$))
160 NEXT I
170 SPRITE$(0)=S$
180 REM POMMI-SPRITE
190 REM RESTORE 1260
200 FOR K=1 TO 8
210 READ B$
220 T$=T$+CHR$(VAL("&B"+B$))
230 NEXT K
240 SPRITE$(1)=T$
250 REM VALILYÖNNIN TARKISTUS
260 STRIG(0) ON
270 REM ALUS PAIKALLEEN
280 PUT SPRITE 0,(X,Y),1,0
300 REM ONKO PELIOHJAINTA LIIKUTETTU?
310 REM JOS EI, LASKE SATUNNAINEN SUU
NTA
320 N=STICK(0):IF N=0 THEN N=INT(RND(
1)*8)
330 REM LASKE SIIRTYMÄ
340 IF N=0 THEN GOTO 280
350 ON N GOSUB 760,770,780,790,800,81
0,820,830
360 REM TARKISTA X:N JA Y:N ARVOT
370 IF X>256 THEN X=256
380 IF X<1 THEN X=1
390 IF Y>104 THEN Y=104
400 IF Y<1 THEN Y=1
410 GOTO 280
420 REM ALIOHJELMAT/LIIKKUMISSUUNTA
760 Y=Y-1:RETURN
770 Y=Y-1:X=X+1:RETURN
780 X=X+1:RETURN
790 X=X+1:Y=Y+1:RETURN
800 Y=Y+1:RETURN
810 Y=Y+1:X=X-1:RETURN
820 X=X-1:RETURN
830 X=X-1:Y=Y-1:RETURN
```

```
840 REM ALIOHJELMA/VALINÄPPÄIMEN TARK
ISTUS,
850 REM POMMIN PUDOTUS JA OSUMISEN TA
RKISTUS
860 V=Y+1
870 PUT SPRITE 1,(X+4,V),1,1
880 IF POINT((X+12),(V+8))=6 THEN PLA
Y"SM1500L2406CDEFG":SC=SC+1:IF SC>4 T
HEN SCREEN 0:COLOR 15,4,7:CLS:END ELS
E :RETURN
890 IF V=192 THEN PLAY "C","F+","B":R
ETURN
900 V=V+1:GOTO 870
910 REM SPRITETIEDOT
920 REM 1-AVARUUSALUS
930 DATA 11110000
940 DATA 11110000
950 DATA 11000000
960 DATA 01000010
970 DATA 01000111
980 DATA 01001010
990 DATA 01111111
1000 DATA 01101011
1010 DATA 01101010
1020 DATA 01111111
1030 DATA 01001010
1040 DATA 01000111
1050 DATA 01000000
1060 DATA 11000000
1070 DATA 11110000
1080 DATA 11110000
1090 DATA 00001111
1100 DATA 00001111
1110 DATA 00000011
1120 DATA 10000010
1130 DATA 11100010
1140 DATA 01010010
1150 DATA 11111110
1160 DATA 01010110
1170 DATA 01010110
1180 DATA 11111110
1190 DATA 01010010
1200 DATA 11100010
1210 DATA 10000010
```

```
1220 DATA 00000011
1230 DATA 00001111
1240 DATA 00001111
1250 REM 2-POMMI
1260 DATA 00011000
1270 DATA 11111111
1280 DATA 11111111
1290 DATA 10000001
1300 DATA 01000010
1310 DATA 00100100
1320 DATA 00111100
1330 DATA 00111100
```

Seuraava ohjelma on kuuhunlaskeutumispeli, jossa pyritään kolmelle valkoiselle laskeutumisalustalle. Välilyöntinäppäimen painaminen hidastaa laskeutumista, mutta kuluttaa samalla polttoainetta. Yrityksessä on tietenkin omat vaaransa. Täytyy varoa sekä liikkuvia että liikkumattomia "möhkäleitä". Aluksesi räjähtää, jos törmäät möhkäleeseen. Alus tuhoutuu myös, jos karahdat kivikkoon tai polttoaine loppuu kesken. Kun laskeutuminen vihdoin onnistuu, pieni mies hyppää ruudun alareunassa olevaan neliöön. Ohjelmassa käytetään tutun ON STRIG GOSUBin kaltaista käskyä ON... GOSUB. Sen avulla voidaan välttää sprite-törmäykset eli aluksesi väistää möhkäleet. SPRITE ON -käsky täytyy antaa ensin, jotta BASIC kykenee havaitsemaan sprite-törmäyksen. SPRITE OFF ja SPRITE STOP ovat samanlaisia käskyjä kuin STRIG ON ja STRIG OFF. Näiden lauseiden käyttöperiaatteet selviävät seuraavan ohjelman avulla:

```
10 SPRITE ON :REM TÖRMÄYKSEN HAVAITSE
MINEN PÄÄLLE

70 ON SPRITE GOSUB 100:REM ONKO SPRIT
E TÖRMÄNNYT

100 REM TÖRMÄYKSEN KÄSITTELYRUTIINI
110 SPRITE OFF:REM  TÖRMÄYKSEN HAVAIT
SEMINEN PÄÄLTÄ POIS
```

```
180 SPRITE ON:REM TÖRMÄYKSEN HAVITSEM
INEN UUDELLEEN PÄÄLLE
190 RETURN
```

Tässä luvattu ohjelma:

```
10 REM KUUHUN LASKEUTUMINEN
20 REM
30 SCREEN 2,0,0
40 CLEAR
50 FLAG=1
60 COL=15
70 DIR=4:K=63
80 STRIG(0) ON
90 ON STRIG GOSUB 1500
100 INC=1
110 FUEL=100
120 FFLAG=0
130 K=100
140 F=0
150 ACC=3
160 X=128:Y=10
170 COLOR 15,4,1
180 REM SPRITEARVOJEN LUKEMINEN
190 N=0
200 GOSUB 820
210 N=1:RESTORE 1050:OBJ$="":GOSUB 820
220 N=2:RESTORE 1140:OBJ$="":GOSUB 820
230 GOSUB 350
240 X=128:Y=10:PUT SPRITE 0,(X,Y),8,1:TL
=0:SPRITE ON
250 ON SPRITE GOSUB 1570
260 REM PELIOHJAIN
270 SOUND 6,K:S=STICK(0):IF S=0 OR S=1 O
R S=8 THEN S=DIR
280 GOSUB 880
290 GOSUB 1230
300 GOSUB 1270
310 GOSUB 1440
320 REM
330 GOTO 270
340 REM MAISEMAN PIIRTÄMINEN
350 LINE(0,160)-(24,184),1
```

```
360 LINE(24,184)-(32,144),1
370 LINE(32,144)-(40,160),1
380 LINE(40,160)-(48,152),1
390 LINE(48,152)-(56,168),1
400 LINE(56,168)-(64,152),1
410 LINE(64,152)-(56,144),1
420 LINE(56,144)-(80,144),1
430 LINE(80,144)-(72,152),1
440 LINE(72,152)-(88,160),1
450 LINE(88,160)-(96,104),1
460 LINE(96,104)-(104,128),1
470 LINE(104,128)-(120,128),1
480 LINE(120,128)-(112,144),1
490 LINE(112,144)-(128,168),1
500 LINE(128,168)-(136,152),1
510 LINE(136,152)-(144,168),1
520 LINE(144,168)-(160,136),1
530 LINE(160,136)-(160,160),1
540 LINE(160,160)-(176,120),1
550 LINE(176,120)-(184,144),1
560 LINE(184,144)-(200,128),1
570 LINE(200,128)-(216,128),1
580 LINE(216,128)-(232,144),1
590 LINE(232,144)-(236,128),1
600 LINE(236,128)-(208,96),1
610 LINE(208,96)-(224,80),1
620 LINE(224,80)-(256,136),1
630 LINE(56,142)-(80,142),15
640 LINE(104,126)-(120,126),15
650 LINE(200,126)-(216,126),15
660 REM PAIKALLAAN OLEVAT LOHKAREET
670 PUT SPRITE 10,(32,56),3,2
680 PUT SPRITE 12,(40,92),3,2
690 PUT SPRITE 13,(148,54),3,2
700 PUT SPRITE 14,(104,50),3,2
710 PUT SPRITE 15,(208,50),3,2
720 PUT SPRITE 16,(112,92),3,2
730 PUT SPRITE 17,(184,90),3,2
740 PUT SPRITE 18,(160,20),3,2
750 CIRCLE (0,0),70,8:PAINT (1,1),8
760 PAINT (128,191),1
770 LINE(200,184)-(224,192),15,B
780 LINE(208,184)-(208,192),15
790 LINE(216,184)-(216,192),15
```

```
800 RETURN
810 REM SPRITEARVOT
820 FOR I=1 TO 8
830 READ A$
840 OBJ$=OBJ$+CHR$(VAL("&B"+A$))
850 NEXT I
860 SPRITE$(N)=OBJ$:RETURN
870 REM PELIOHJAIMEN LUKEMINEN
880 IF S=4 THEN 900
890 DIR = S
900 IF S=3 THEN X=X+2:RETURN
910 IF S=4 THEN X=X+1:Y=Y+ACC:RETURN
920 IF S=5 THEN Y=Y+ACC:RETURN
930 IF S=6 THEN Y=Y+ACC:X=X-1:RETURN
940 IF S=7 THEN X=X-1:Y=Y+1:RETURN
950 REM AVARUUSALUS
960 DATA 00011000
970 DATA 00011000
980 DATA 00111100
990 DATA 00111100
1000 DATA 00111100
1010 DATA 11111111
1020 DATA 10000001
1030 DATA 10000001
1040 REM MIES
1050 DATA 00000000
1060 DATA 00011000
1070 DATA 00011000
1080 DATA 01111110
1090 DATA 00011000
1100 DATA 00111100
1110 DATA 00111100
1120 DATA 00000000
1130 REM LOHKARE
1140 DATA 00111000
1150 DATA 01111101
1160 DATA 00111110
1170 DATA 01111111
1180 DATA 11111111
1190 DATA 01111100
1200 DATA 00111110
1210 DATA 00011111
1220 REM AVARUUSALUKSEN JA LOHKAREIDEN S
IJOITTAMINEN
```

```
1230 IF X<0 THEN X=0:RETURN
1240 IF Y<0 THEN Y=0:RETURN
1250 IF X>256 THEN X=256:RETURN
1260 IF Y>192 THEN Y=192:RETURN
1270 REM LIIKKUVAT LOHKAREET
1280 PUT SPRITE 0,(X,Y),8,0
1290 PUT SPRITE 2,(F,80),3,2:F=F+1
1300 PUT SPRITE 3,(F+100,100),3,2
1310 PUT SPRITE 4,(F+10,40),3,2
1320 PUT SPRITE 5,(F+60,128),3,2
1330 PUT SPRITE 6,(F+112,116),3,2
1340 PUT SPRITE 7,(F+224,124),3,2
1350 PUT SPRITE 8,(F+72,72),3,2
1360 IF FLAG=1 THEN RETURN
1370 GOTO 1400
1380 PUT SPRITE 1,(X,Y+4),15,1
1390 PUT SPRITE 1,(X,Y+4),0,1
1400 FUEL=FUEL-2:K=K-4:IF K<0 THEN K =63
1410 IF FUEL=0 THEN GOTO 1570
1420 REM
1430 REM ONKO AVARUUSALUS LASKEUTUNUT ?
1440 IF (POINT(X,Y+9)=15) AND (POINT(X+9
,Y+9)=15) THEN BEEP:PLAY"L2006CDEFG":X=1
28:Y=10:FFLAG=0:FLAG=1:FUEL=100:SC=SC+1
1450 IF SC=1 THEN PUT SPRITE 23,(200,184
),15,1
1460 IF SC=2 THEN PUT SPRITE 23,(208,182
),15,1
1470 IF SC=3 THEN PUT SPRITE 23,(216,184
),15,1:SC=0
1480 IF POINT(X,Y+9)=1 OR POINT(X+9,Y+9)
=1 THEN GOTO 1570
1490 RETURN 330
1500 SWAP FLAG,FFLAG:SOUND 7,5:SOUND 8,7
1510 ACC=1:IF FFLAG=0 THEN PUT SPRITE 1,
(0,0),0,1
1520 IF FFLAG=0 THEN ACC=3:SOUND 8,0
1530 RETURN
1540 END
1550 REM TÖRMÄYKSEN TARKASTAMINEN
1560 TL=1
1570 BEEP
1580 FOR I=1 TO 4
1590 PUT SPRITE 0,(X,Y),1,0
```

```
1600 PUT SPRITE 0,(X,Y),15,0
1610 FOR J=1 TO 255 STEP 5:SOUND 8,9:SOU
ND 0,J:SOUND 8,0:NEXT J
1620 NEXT
1630 IF TL=1 THEN GOTO 240
1640 X=128:Y=10:FUEL=100:FLAG=1:FFLAG=0
1650 RETURN
```

Draw

DRAW-käskyä käytetään piirtämiseen grafiikkamakrokielessä (GML). Se toimii samoin kuin edellisessä luvussa oppimamme MML. Piirrosohjeet esitetään GML-käskyjä sisältävinä merkkijonoina.

Kokeile esimerkkiämme:

```
10 SCREEN 2
20 DRAW "R100D100L100U100"
30 GOTO 30
```

Tämä ohjelma siis piirtää ruutuun neliön. Neliö sijoittuu sen mukaan, oletko jo käyttänyt jotain grafiikkatilaa. Jos syötät tämän ohjelman ensimmäiseksi koneen käynnistämisen jälkeen, neliö piirtyy ruudun vasempaan yläkulmaan. Jos sitten oletkin jo käyttänyt jotain grafiikkaruutua, neliön vasen yläkulma sijoittuu viimeksi käsitellyn pikselin kohdalle.

DRAW-käskyllä tietokone piirrättää joukon viivoja: ensin 100 pikseliä oikealle, sitten 100 alas ja vasemmalle ja lopuksi takaisin ylös, ja niin päädytään takaisin lähtöpisteeseen – siinä kaikki! Muuta nyt rivi 20 kokeeksi:

```
20 DRAW"D100F100E100H100G100"
```

Ensin piirtyy 100 pikselin pystyviiva alaspäin, sitten vinoviiva alaoikeaan, yläoikeaan, ylävasempaan ja lopuksi alavasempaan, missä käytettiin neljää uutta GML:n merkintää: F, E, H ja G. Täydellinen GML-luettelo:

U<n> Ylöspäin
D<n> Alaspäin
L <n> Vasempaan
R <n> Oikeaan
E <n> Yläoikeaan
F <n> Alaoikeaan
G<n> Alavasempaan
H<n> Ylävasempaan

Jokaisen käskyn liike alkaa viimeiseksi mainitusta kohdasta. Niinpä liike alkaa ruudun vasemmasta yläkulmasta ja jatkuu sitten aina viimeksi suoritetun käskyn päätöskohdasta. <n> ilmoittaa piirrosmatkan, joten R100 tarkoittaa, että täytyy siirtyä 100 pikseliä oikeaan. F100 tarkoittaa 100 pikseliä alaoikeaan jne.

Jos halutaan aloittaa piirtäminen muualta kuin vasemmasta yläkulmasta, voidaan käyttää seuraavaa käskyä:

M<x,y>

jolloin x ilmoittaa liikkeen vaaka- ja y pystytasossa. Muuta kokeeksi rivi 20 tähän muotoon:

```
20 DRAW "M100,10R100D100L100U100"
```

Tällä kertaa piirtäminen alkaakin kohdasta, joka on 100 pikseliä ruudun vasemmasta reunasta ja 10 pikseliä yläreunasta.

Kaikkiin noihin käskyihin voidaan kirjoittaa etuliite B tai N. B siirtää, mutta ei jätä merkkiä mihinkään kohtaan, N vuorostaan palaa piirtämisen jälkeen aina alkupisteeseensä. Muut käskyt ovat seuraavat:

A<n> Tämän avulla voidaan määrittää, mihin kulmaan viiva piirretään. <n>:lle voidaan antaa arvo 0 – 3, jolloin 0 on 0, 1 on 90, 2 on 180 ja 3 on 270 astetta.

```
    0
1-- + --3
    2
```

C<n> säätää piirrettävän viivan värin, joka voi COLOR-käskyn mukaan olla 0 – 15.

S<n> säätää U,D,L,R,E,F,G,H- ja M-käskyjen liikkeiden pituuden.

X<merkkimuuttuja> mahdollistaa toisen piirroskäskyn DRAW-käskyn sisällä. Se on käyttökelpoinen esimerkiksi kun halutaan erottaa jokin osa koko piirroksesta tai toteuttaa DRAW-käskyjä, joissa on yli 255 merkkiä.

Värilaatikko-ohjelma

Seuraava ohjelma on nimensä mukaisesti yksinkertainen, monikäyttöinen piirrosohjelma. Grafiikkaruutu jaetaan siinä kahteen alueeseen: piirustus- ja värivalinta-alueeseen. Kohdistinta ohjataan joko sauvaohjaimella tai kohdistimen näppäimillä.

Värivalinta-alue sisältää värilaatikon, josta ohjelman nimi on peräisin. Paletissa on neljätoista väriä, joista haluttu osoitetaan sijoittamalla kohdistin sen ylle ja painamalla välilyöntinäppäintä. Paletista puuttuvat värit 0 (läpinäkyvä) ja 15 (valkoinen).

Käytettävissä on maalisivellin, suorakulma, harppi, neliönpiirrin ja pyyhin. Kuvioiden täyttämiseen voidaan käyttää PAINT-vakiokäskyn lisäksi neljää välinettä: sumutinta, vaaka- ja pystyviivoitusta ja pisteyttämistä. Kohdistin vaihtaa haluttuun kuvion piirtämis- tai täyttä-

misvälineeseen, kun sitä siirrellään piirrosalueella. Mainittakoon, että kohdistimen hallitsemia kuvioita kutsutaan **ikoneiksi.**

Nuo yhteensä 20 piirrostoimintoa ovat näppäimistöllä valittavissa. Käskyjen tunnuskirjaimet ovat seuraavassa luettelossamme (ne ovat kaikki suuraakkosina). Käskykirjaimen tulisi vastata englanninkielistä toimintoa, mutta siinä tuli laatijoille lopulta seinä vastaan!

Siveltimen käyttö

D valitsee siveltimen ja painaa sen alas ("down"). Kohdistinta alaspäin ohjaavaa näppäintä painettaessa piirtyy sitten viiva. Aluksi sivellin vetää hyvin kapean viivan, mutta kun D:tä painetaan toistamiseen, väri leviää koko siveltimen leveydelle.

U poimii siveltimen ylös ("up") jälkeä jättämättä.

Sumuttimen käyttö

A valitsee sumuttimen (spraymaalin).

Välilyöntinäppäin sumuttaa maalia.

Harpin käyttö ympyrän piirtämiseen

C valitsee harpin ja määrittää piirrettävän ympyrän keskiön siihen kohtaan, jossa kohdistin on.

R määrittää ympyrän säteeksi viivan, joka piirtyi C:n painamiskohdan ja kohdistimen uuden aseman väliin. Ympyrä piirtyy sen jälkeen.

Suorakulman käyttö viivan piirtämiseen

L valitsee suorakulman ja määrittää kohdistimen aseman viivan alkupisteeksi.

E määrittää viivan loppupisteen – ja vetää viivan.

Neliömallin käyttö

B valitsee neliön ja määrittää neliön piirtämisen aloituspisteeksi yhden neliön kulmista.

X määrittää neliöstä aloituskulmaan vinottaisesti olevan kulman ja piirtää sitten neliön.

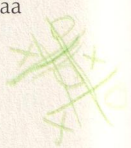
Neliön tai muun rajatun alan maalaaminen

P maalaa kokonaan rajatun alan, johon kohdistin on sijoitettu. Mikäli alaa ei sitten olekaan rajattu, koko ruutu muuttuu samanväriseksi.

Neliöiden sävyttäminen

Näitä käskyjä näppäilemällä valitaan pieniä kuviosävytyskeinoja. Välilyöntinäppäimen painaminen sävyttää kohdistimen alla olevan neliön valitulla kuviolla.

V pystyviivoitus täytekuviona
Y vaakaviivoitus täytekuviona
K pisteytys täytekuviona



Pyyhkimen käyttö

W valitsee pyyhkimen. Kohdistimen siirto pyyhkii maalaten kaiken valkoisilla pikseleillä.

Muut käskyt

Q valitsee valkoisen värin (piirrosalueella oltaessa).
T kiepauttaa kohdistimen – muuttaa sen värin valkoisesta mustaksi tai mustasta valkoiseksi.
H "kotiin" – siirtää värivalinta-alueelle.
S "ruutuun" – siirtää piirrosalueen keskustaan.
Z nollavaihtoehto – pyyhkii ruudun piirrosalueen valkoiseksi. Jotta taideteokset eivät tuhoutuisi tahattomasti, Z-käskyä täytyy käyttää kahdesti ennen kuin se tehoaa.
F poistaa ohjelmasta. Kuten Z-käskyä, tätäkin täytyy tehdä kahdesti, jos halutaan toteuttaa käsky.

Tämä ohjelma käyttää spriteja tyhjentävästi määrittämään kohdistinmuotoja, joita on yhteensä kymmenen kappaletta. Jokaisen kohdistinmuodon viitekohta on vasen yläkulma. Neliön piirtäminen alkaa siitä kohdasta, jossa kohdistimen vasen yläkulma on.

Yhtä käyttäjän määrittämää toimintoa käytetään laskemaan ympyrän säde Pythagoraan teoreeman avulla.

Kun kohdistin siirretään värivalinta-alueelle, se muuttuu nuoleksi. Kun se palautetaan piirrosalueelle, se

muuttuu viimeksi käytetyn piirrosvälineen näköiseksi. Perusoletuksena on, että piirrosvälineenä on sivellin.

Ohjelmassa suoritetaan melkoisesti tarkistuksia, joilla varmistetaan, että vain piirrosalueella olevat kohteet piirtyisivät. Useimmiten käskykirjaimet kelpaavat vain kun kohdistin on piirrosalueen rajojen sisällä.

Jos kohdistin on ruudun jotain alaa maalattaessa vaikka yhdenkin sellaisen pikselin yllä, joka on samanvärinen kuin maalattava väri, ei maalaamista huomaa. Ennen alan värittämistä kohdistin tuleekin nostaa ja siirtää toisenväristen pikselien kohdalle.

Alaa maalattaessa ohjelma havaitsee vain ne viivat, jotka ovat samanvärisiä kuin ääriviivat. Niinpä tulee varmistaa, että väritettävän kuvion ääriviivat ovat samanvärisiä kuin käyttöön on valittu. Jotta ohjelma ei värittäisi muuta kuin piirrosalueen, rajaus piirtyy automaattisesti uudelleen valitun väriseksi ennen kuin alueen värittäminen alkaa. Jos P:tä painetaan ennen kuin mitään on piirretty, koko piirrosalue muuttuu valitun värin mukaiseksi, jolloin "kankaana" voidaan käyttää mitä tahansa väriä paitsi valkoista.

```
10 REM PAINTBOX
20 DEFINT A-Z
30 REM JOYSTICK-ARVOT
40 DIM Z(8,2)
50 FOR I=1 TO 8
60 FOR J=1 TO 2
70 READ A:Z(I,J)=A
80 NEXT J
90 NEXT I
100 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0
,-1,-1
110 SCREEN 2,0,0
120 COLOR 15,15,15:CLS
130 STRIG(0) ON
140 REM PYTHAGORAAN LAUSE
150 DEFFNC(M,N,O,P)=SQR((ABS((M-O)^2))+(
ABS((N-P)^2)))
```

```
160 F=1:M=2:N=1:Q=1:P1=0:P2=1
170 ON STRIG GOSUB 1400
180 X=40:Y=172
190 GOSUB 320:REM NÄYTTÖ
200 GOSUB 410:REM SPRITET
210 W=15
220 REM KOHDISTIN
230 C5=1
240 PUT SPRITE 1,(X,Y),Q,N
250 IF N=3 THEN PSET(X,Y),15
260 IF X>57 AND FLG=1 AND N=2 THEN GOSUB
 2090
270 IF STICK(0)=0 THEN GOTO 290
280 GOSUB 1470
290 B$=INKEY$:IF B$="" THEN GOTO 240
300 GOSUB 1560
305 GOTO 240
310 REM NÄYTÖN PIIRTÄMINEN
320 C=1
330 FOR I=8 TO 32 STEP 24
340 FOR J=8 TO 152 STEP 24
350 LINE (I,J)-(I+16,J+16),C,BF:C=C+1
360 NEXT J
370 NEXT I
380 LINE (56,8)-(248,184),1,B
390 RETURN
400 REM SPRITEARVOJEN LUKEMINEN
410 FOR I=1 TO 10
420 S$=""
430 FOR J=1 TO 8
440 READ A$:S$=S$+CHR$(VAL("&B"+A$))
450 NEXT J
460 SPRITE$(I)=S$
470 NEXT I
480 RETURN
490 REM NUOLI
500 DATA 11110000
510 DATA 11000000
520 DATA 10100000
530 DATA 10010000
540 DATA 00001000
550 DATA 00000100
560 DATA 00000010
570 DATA 00000001
```

```
580 REM MAALARIN SIVELLIN
590 DATA 11110000
600 DATA 11110000
610 DATA 11110000
620 DATA 00000000
630 DATA 11110000
640 DATA 01100000
650 DATA 01100000
660 DATA 01100000
670 REM PYYHEKUMI
680 DATA 11110000
690 DATA 10011000
700 DATA 10010100
710 DATA 11110010
720 DATA 01000001
730 DATA 00100001
740 DATA 00010001
750 DATA 00001111
760 REM KOMPASSI
770 DATA 11000000
780 DATA 00100000
790 DATA 00010010
800 DATA 00001111
810 DATA 00001111
820 DATA 00010010
830 DATA 00100000
840 DATA 11000000
850 REM SUMUTIN
860 DATA 11100000
870 DATA 00111110
880 DATA 00100010
890 DATA 11111010
900 DATA 10001000
910 DATA 10001000
920 DATA 10001000
930 DATA 11111000
940 REM VIIVA
950 DATA 10000000
960 DATA 10000000
970 DATA 10000000
980 DATA 10000000
990 DATA 10000000
1000 DATA 10000000
1010 DATA 10000000
```

```
1020 DATA 10000000
1030 REM LAATIKKO
1040 DATA 11111111
1050 DATA 10000001
1060 DATA 10000001
1070 DATA 10000001
1080 DATA 10000001
1090 DATA 10000001
1100 DATA 10000001
1110 DATA 11111111
1120 REM VAAKASUORAT PYLVÄÄT
1130 DATA 00000000
1140 DATA 11111111
1150 DATA 00000000
1160 DATA 11111111
1170 DATA 00000000
1180 DATA 11111111
1190 DATA 00000000
1200 DATA 11111111
1210 REM PYSTYSUORAT PYLVÄÄT
1220 DATA 10101010
1230 DATA 10101010
1240 DATA 10101010
1250 DATA 10101010
1260 DATA 10101010
1270 DATA 10101010
1280 DATA 10101010
1290 DATA 10101010
1300 REM TÄPLIKÄS KOHDISTIN
1310 DATA 10101010
1320 DATA 01010101
1330 DATA 10101010
1340 DATA 01010101
1350 DATA 10101010
1360 DATA 01010101
1370 DATA 10101010
1380 DATA 01010101
1390 REM VALI/KESKEYTYSRUTIINI
1400 IF X>57 AND N=5 THEN GOSUB 1930:RET
URN
1410 IF X>57 AND N=8 THEN V1=1:V2=2:GOSU
B 2000:RETURN
1420 IF X>57 AND N=9 THEN V1=2:V2=1:GOSU
B 2000:RETURN
```

```
1430 IF X>57 AND N=10 THEN V1=2:V2=2:GOS
UB 2000:RETURN
1440 IF X>57 THEN RETURN
1450 IF POINT(X,Y)=15 THEN PLAY "L1402C"
:RETURN
1460 C5=POINT(X,Y):PLAY "L2405C06C":RETU
RN
1470 X=X+Z(STICK(0),1):Y=Y+Z(STICK(0),2)
1480 IF X>246 THEN X=246
1490 IF Y>183 THEN Y=183
1500 IF Y<9 THEN Y=9
1510 IF X<9 THEN X=9
1520 IF X<56 THEN N=1
1530 IF X>56 THEN N=M:F=0
1540 RETURN
1550 REM KOMENTOSANAT
1560 IF (ASC(B$)<65) THEN RETURN
1570 IF B$="F" THEN BEEP:E1=E1+1:IF E1=2
 THEN SCREEN 0:COLOR 15,5,4:END
1580 IF X<56 AND B$="S" THEN X=152:Y=96:
N=M:RETURN
1590 IF B$="T" THEN SWAP Q,W:RETURN
1600 IF X<56 THEN RETURN
1610 CL=CW
1620 IF B$="Q" THEN C5=15:RETURN
1630 IF B$="U" THEN FLG=0:N=2:M=2:SWAP P
1,P2:RETURN
1640 IF B$="D" THEN FLG=1:N=2:M=2:SWAP P
1,P2:RETURN
1650 IF B$="H" THEN X=40:Y=176:N=1:RETUR
N
1660 IF B$="R" AND F2=1 THEN GOSUB 1810:
F2=0:RETURN
1670 IF B$="P" THEN LINE(56,8)-(248,184)
,C5,B:PAINT(X,Y),C5:RETURN
1680 IF B$="Z" THEN BEEP:Z=Z+1:IF Z>1 TH
EN LINE(56,8)-(248,184),15,BF:LINE(56,8)
-(248,184),1,B:Z=0:RETURN
1690 IF B$="C" THEN C1=X:C2=Y:M=4:N=4:F2
=1:RETURN
1700 IF B$="Y" THEN M=8:N=8:RETURN
1710 IF B$="V" THEN M=9:N=9:RETURN
1720 IF B$="K" THEN M=10:N=10:RETURN
1730 IF B$="W" THEN N=3:M=3:CW=CL:CL=15:
```

```
FLG=1:RETURN
1740 IF B$="L" THEN L=1:L1=X:L2=Y:N=6:M=
6:RETURN
1750 IF B$="B" THEN B=1:B1=X:B2=Y:N=7:M=
7:RETURN
1760 IF B$="X" AND B=1 THEN LINE(B1,B2)-
(X,Y),C5,B:B=0:RETURN
1770 IF B$="E" AND L=1 THEN LINE(L1,L2)-
(X,Y),C5:L=0:RETURN
1780 IFB$="A" THEN N=5:M=5
1790 RETURN
1800 REM SATEEN LASKEMINEN
1810 R = FNC(X,Y,C1,C2)
1820 D1=C1:D2=C2
1830 GOSUB 1860
1840 CIRCLE (C1,C2),R,C5,,,8/7
1850 RETURN
1860 REM SATEEN TARKISTUS
1870 IF D1+R>246 THEN BEEP:RETURN
1880 IF D1-R<57 THEN BEEP:RETURN
1890 IF D2+R>183 THEN BEEP:RETURN
1900 IF D2-R<9 THEN BEEP:RETURN
1910 RETURN
1920 REM SUMUTTAMINEN
1930 IF (X-8<57)OR(Y-8<9) THEN RETURN
1940 RV=5*RND(1):IF RV<2 THEN 1940
1950 FOR I=1 TO RV
1960 PSET(X-(8*RND(1)),Y-(8*RND(1))),C5
1970 NEXT I
1980 RETURN
1990 REM VIIVA/TAPLA-RUTIINI
2000 IF (X+8)>246 OR (Y+8)>183 THEN BEEP
:RETURN
2010 FOR I=X TO X+7 STEP V1
2020 FOR J=Y+1 TO Y+7 STEP V2
2030 PSET(I,J),C5
2040 NEXT J
2050 NEXT I
2060 RETURN
2070 REM PIIRTO
2080 IF P1=1 THEN PSET(X,Y),C5:RETURN
2090 WL=3
2100 IF X+WL>246 THEN WL=WL-1:GOTO 2120
2110 FOR I=0 TO WL
```

```
2120 PSET(X+I,Y),C5
2130 NEXT I
2140 RETURN
```

Värilaatikon parannusehdotuksia

C-käskyä käytettäessä voi piirrosalueella piirtää vain täysiä ympyröitä. Piirrosalueen rajalla ympyröitä voi jokseenkin rouheasti mutta myös tehokkaasti leikata piirtämällä sen toisella kertaa alueen *ulkopuolelle*. Seuraava aliohjelma ja muut pikku muutokset saavat sen aikaan:

```
3000 REM PEITERUTIINI
3010 LINE(0,0)-(56,192),15,BF
3020 LINE(56,0)-(256,8),15,BF
3030 LINE(56,184)-(256,192),15,BF
3040 LINE(248,0)-(256,192),15,BF
3050 GOSUB 320
3060 RETURN
```

Ympyrän mittoja tarkistava aliohjelma (rivit 1860 – 1900) tulee sitten muuttaa tällaiseksi:

```
1860 REM YMPYRÄN SÄTEEN TARKISTUS
1870 IF D1+R>246 THEN FC=1:RETURN
1880 IF D1-R<57 THEN FC=1:RETURN
1890 IF D2+R>183 THEN FC=1:RETURN
1900 IF D2-R<9 THEN FC=1:RETURN
```

Rivi 1910 poistetaan. Viimeinen muutos on rivin 1845 lisäys, jolla kutsutaan aliohjelma:

```
1845 IF FC=1 THEN GOSUB 3000:FC=0
```

Vaikka ruudun uudelleen piirtämiseen käytetään runsaasti grafiikkavoimia, aliohjelma toimii yhä hyvin nopeasti.

Toinen parannus olisi virhetarkistus. Siinä ohjelma osoittaisi virheen pelkän äänimerkin sijasta jotenkin

kuvallisesti. Kullekin virheelle voisi määrittää omat spritensa. Sitten vastaava sprite ilmestyisi ruudun vasempaan alakulmaan aina virheen sattuessa. Jos esimerkiksi yrittäisit maalata jotain alaa, kun kohdistin on samanvärisellä alustalla, näkyviin voisi tulla oikealle osoittava nuoli, joka kehottaisi siirtämään kohdistinta.

Kehittynyt väritysvaihtoehto saataisiin siten, että neliöiden ja ympyröiden väritykseen voitaisiin käyttää vaaka- ja pystyviivoja tai pisteitä. Joskin neliöille sen voi tehdä helposti, mutta ympyröiden kanssa tulee aikamoisia hankaluuksia. Seuraavassa on ehdotelma neliöiden värittämiseen. Ensin muutetaan rivi 1760:

```
1760 IF B$="X" AND B=1 THEN LINE(B1,B
2)-(X,Y),C5,B:B=0:GOSUB4000:RETURN

4000 C$=INKEY$:IF C$="" THEN 4000
4010 IF ASC(C$)<48 OR ASC(C$)>51 THEN
 BEEP:RETURN
4020 IF C$="0" THEN RETURN: REM EI TE
KSTIÄ
4030 IF C$="1" THEN V1=1:V2=2:GOSUB50
00:RETURN
4040 IF C$="2" THEN V1=2:V2=1:GOSUB50
00:RETURN
4050 IF C$="3" THEN V1=2:V2=2:GOSUB50
00:RETURN
4060 RETURN
5000 IF B1>X THEN V1=-V1
5010 IF B2>Y THEN V2=-V2
5020 FOR I=B1 TO X STEP V1
5030 FOR J=B2 TO Y STEP V2
5040 PSET (I,J),C5
5050 NEXT J
5060 NEXY I
5070 RETURN
```

Kun X-käsky on annettu, ohjelma odottaa, että kirjoitetaan luku 0 – 3. Ne vastaavat seuraavia:

0 Älä väritä neliötä.
1 Väritä vaakaviivoilla.
2 Väritä pystyviivoilla.
3 Väritä pistekudoksella.

Aliohjelman rivit 5020 – 5070 ovat miltei samat kuin 2010 – 2060. Jotta rivit eivät toistuisi, kannattaa yrittää muuttaa sekä aliohjelmaa kutsuvia rivejä 1410, 1420 ja 1430 että riviltä 4000 alkavaa aliohjelmaa. Uuden aliohjelman rivit 5000 ja 5010 täytyy kuitenkin säilyttää.

Ympyröiden värittäminen tällä tavoin aiheuttaa valtavia ongelmia. Niihin liittyy melko raskasta laskentaa, mutta sopii yrittää, jos rahkeissa tuntuu olevan varaa!

Olet varmasti huomannut, että tätä ohjelmaa voi laajentaa melkoisesti. Toivottavasti se auttaa alkuun. Näyttää soveliaalta lopettaa tähän kirjan viimeiseen (ja pisimpään) ohjelmaan.

Liite A

MSX-BASICin funktiot

ABS(X)
Ilmoittaa lausekkeen X itseisarvon.

ASC(X$)
Ilmoittaa jonossa X$ ensimmäisenä olevan merkin arvon ASCII-koodissa. Jos X$ on tyhjä jono, ilmoitetaan "Illegal function call".

ATN(X)
Ilmoittaa radiaaneina sen kulman, jonka tangentti on X. Tulos on väliltä −pii/2 – +pii/2. Lauseke X voi olla mitä tahansa numeerista tyyppiä. Tämä funktio toteutuu aina kaksoistarkkuudella.

BIN$(X)
Ilmoittaa desimaaliluvun binaariarvon jonona. X:n täytyy olla numeerinen lauseke väliltä −32768 – +65535. Jos n on negatiivinen, käytetään kahden komplementtimuotoa – – BIN$(−n) on BIN$!(65536 − X).

CDBL(X)
Muuttaa X:n kaksoistarkkuuden mukaiseksi luvuksi.

CHR$(X)
Ilmoittaa X:ää vastaavan ASCII-koodijonon.

CINT(X)
Muuttaa X:n integerluvuksi. "Overflow"-virhe ilmenee, kun X ei ole lukuarvojen −32768 – +32767 mukainen.

COS(X)
Ilmoittaa kulman X kosinin arvon; X on radiaaneina. Se lasketaan kaksoistarkkuuden mukaan.

CSNG(X)
Muuttaa X:n kertatarkkuuden mukaiseksi luvuksi.

CSRLIN
Ilmoittaa kohdistimen pysty(sarake)koordinaatin.

ERL/ERR
Virhemuuttujat. Kun virhe on saatu tavoitetuksi, ERR saa virhekoodin arvon ja ERL sisältää virheellisen rivin rivinumeron. Jos virheen aiheuttanut lause oli suoran käyttötavan seurausta, ERL on 65535.

EXP(X)
Eksponenttifunktio, kantalukuna Neperin luku e = 2.72, matematiikassa vastaava merkintä e^x.

FIX(X)
Ilmoittaa X:n (katkaisussa poistetun) kokonaisosan.

FRE(" ")
Ilmoittaa niiden muistissa olevien tavujen määrän, jotka ovat BASIC-ohjelmien, -muuttujien ym. käytettävissä.

HEX$(X)
Ilmoittaa X:n heksadesimaalijonona. X:n arvon täytyy olla −32768 – +65535.

INKEY$
Ilmoittaa joko yksittäisen merkin, joka on syötetty näppäimistöllä, tai tyhjän jonon.

INPUT$(X)
Ilmoittaa näppäimistöltä syötetyn jonon, joka on X-merkkimäärän pituinen.

INSTR(IX,IA$,B$)
Hakee ensimmäisenä A$:sta löytyvän jonon B$ ja ilmoittaa A$:n kohdan, josta vastaavuus havaittiin. X on käytettävissä haun aloituskohdan määritystä varten. X:n täytyy olla arvoalueella 0 – 255.

INT(X)
Ilmoittaa suurimman kokonaisluvun <=X.

LEFT$(A$,X)
Ilmoittaa jonon A$ vasemmanpuoleisimmat X merkkiä. X:n täytyy olla arvoalueella 0 – 255.

LEN(X$)
Ilmoittaa X$:n merkkien määrän. Kaikki nekin merkit ja välit, joita ei voida kirjoittaa, lasketaan.

LOG(X)
Ilmoittaa X:n luonnollisen logaritmin. X:n täytyy olla suurempi kuin nolla.

LPOS(X)
Ilmoittaa kohdan, jossa rivikirjoittimen kirjoitinpää puskurin mukaan on. Tämä funktio ei kuitenkaan välttämättä ilmaise kirjoitinpään todellista sijaintia. X on valeargumentti.

MID$(A$,XI,YI)
Ilmoittaa A$:sta jonon, joka on Y-merkkiä pitkä ja alkaa X:nnen merkin kohdalta. Sekä X:n että Y:n täytyy olla arvoalueella 1 – 255. Jos Y:tä ei käytetä tai X:nnen merkin oikealla puolella on merkkejä vähem-

män kuin Y:n verran, kaikki oikeanpuoleiset merkit ilmoitetaan.

OCT$(X)
Ilmoittaa desimaaliargumentin X oktaaliarvojonona. X:n täytyy olla numeerinen arvo −32768 – +65535.

PEEK
Ilmoittaa muistipaikassa X olevan tavun (kokonaislukuarvo 0 – 255). X:n täytyy olla arvoalueelta −32768 – +65535.

POS(X)
Ilmoittaa kohdistimen sijainnin vaakatasossa. Vasemmanpuoleisin kohta on 0. X on valeargumentti.

RIGHT$(A$,X)
Ilmoittaa jonon A$ oikeanpuoleisimmat X merkkiä. Funktiota käytetään samoin kuin LEFT$:ia.

RND(X)
Ilmoittaa satunnaisen luvun 0 – 1. Sama satunnaislukusarja generoituu joka kerta ohjelmaa käytettäessä.

SGN(X)
Ilmoittaa 1 (kun X>0), 0 (kun X=0) tai −1 (kun X<0).

SIN(X)
Ilmoittaa kulman X sinifunktion arvon; x on radiaaneina. SIN(X) lasketaan kaksoistarkkuuden mukaan.

SPACE$(X)
Ilmoittaa jonon, jossa on X:n pituudelta välejä. X:n täytyy olla arvoalueelta 0 – 255.

SPC(X)
Tulostaa ruutuun X:n verran tyhjiä merkkejä. SPC:tä saa käyttää ainoastaan PRINT- ja LPRINT-lauseissa. X:n täytyy olla arvoalueelta 0 – 255.

SQR(X)
Ilmoittaa X:n neliöjuuren. X:n täytyy olla >=0.

STR$(X)
Muuttaa numeerisen arvon X merkkijonomuotoon.

STRING$ (X,Y) ja **STRING$ (X,A$)**
Joko X$ on STRING$(X,Y), jolloin jonoon X$ otetaan X kappaletta merkkejä joiden ASCII-koodi on Y, tai X$ on STRING$(X,A$), jolloin jonoon tulee X kappaletta jonon A$ ensimmäistä merkkiä.

TAB(X)
Tulostaa välejä konsolin kohtaan X asti. Jos kirjoituskohta on jo siirtynyt X:n ohi, TAB ei tee mitään. X:n täytyy olla arvoalueella 0 – 255. TABia saadaan käyttää ainoastaan PRINT- ja LPRINT-lauseiden yhteydessä.

TAN(X)
Ilmoittaa kulman X tangentin arvon; X on radiaaneina. TAN(X) lasketaan kaksoistarkkuuden mukaan.

USR[<luku>](X)
Kutsuu käyttäjän assemblerkielisen aliohjelman, jossa on argumentti X. <luku> on arvoalueella 0 – 9 ja vastaa ohjelman DEFUSR-lauseen lukemaa. Jos <luku> jää antamatta, oletetaan tarkoitettavan USR(0):aa.

VAL(X$)
Ilmoittaa jonon X$ numeerisen arvon. Funktio karsii myös jonon alusta tyhjät merkit, tabuloinnit ja rivijaot.

VARPTR(<muuttujan nimi>) ja
VARPTR(#<tiedoston n:o>)
Ilmoittaa <muuttujan nimi> -datan ensimmäisen tavun osoitteen. Arvo tulee sijoittaa <muuttujan nimeen> ennen kuin VARPTR suoritetaan, muutoin tulokseksi saadaan jälleen "illegal function call". Ilmoitettava integerarvo on väliltä −32768 – +32767. Jos ilmoitetaan negatiivinen arvo, siitä saa todellisen osoitteen, kun siihen lisätään 65536.

Liite B

Virhekoodit ja -ilmoitukset

1 NEXT without FOR
Et muistanut sijoittaa FORia silmukan FOR . . . NEXT alkuun. NEXT-lauseen muuttuja ei vastaa mitään jo suoritettua paritonta FOR-lauseen muuttujaa.

2 Syntax error
Ohjelmassa on tavattu rivi, jossa on jokin virheellinen merkkisarja, esimerkiksi pariton sulkumerkki, väärin kirjoitettu käsky tai lause, virheellinen välimerkki tms.

3 RETURN without GOSUB
Ohjelmassa on kohdattu RETURN-lause, jolle ei löydy edeltävää paritonta GOSUB-lausetta.

4 Out of DATA
Ohjelmasta ei löytynyt suoritetulle READ-lauseelle DATA-lausetta, jossa olisi lukematonta tietoa.

5 Illegal function call
Laskenta- tai jonotoiminnolle on annettu arvoalueeseen sopimaton parametri. Virheilmoituksen voi aiheuttaa myös:

1. negatiivinen tai suhteettoman suuri indeksi
2. negatiivinen tai nolla argumentiksi LOGiin
3. negatiivinen argumentti SQR:ään
4. sopimaton argumentti jollekin seuraavista: MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRIG$, SPACE$, INSTR$ tai ON ... GOTO.

6 Overflow
Laskennan tulos ylittää BASICin lukuilmauksen rajat.

7 Out of memory
Ohjelma on liian laaja, sisältää liian monta tiedostoa, FOR-silmukkaa tai GOSUBia, muuttujaa tai ilmausta, jotka ovat liian mutkikkaita.

8 Undefined line number
GOTO, GOSUB tai IF...THEN...ELSE viittaa riville, jota ei ole ohjelmassa.

9 Subscript out of range
Taulukossa on yritetty käyttää indeksiä, joka joko ylittää taulukon mitat tai jonka numerointi ei sovi.

10 Redimensioned array
Samalle taulukolle esitetty kaksi DIM-lausetta tai taulukolle on asetettu oletusarvo 10 jo ennen kuin DIM-lause esitetään.

11 Division by zero
Lausekkeessa on havaittu nollalla jakaminen tai olet yrittänyt korottaa nollan negatiiviseen potenssiin.

12 Illegal function call
Suorassa käyttötilassa yritetty syöttää siihen soveltumaton käsky.

13 Type mismatch
Lukuarvolle on yritetty antaa merkkimuuttujan nimi tai päinvastoin; numeerista argumenttia edellyttävälle toiminnolle on tarjottu merkkiargumentti tai päinvastoin.

14 Out of string space
Merkkimuuttujat ovat saaneet MSX-BASICin ylittämään vapaan muistitilan. MSX-BASIC käyttää merkkijonojen tilaa dynaamisesti, kunnes muisti täyttyy.

15 String too long
On yritetty laatia jono, jonka pituus on yli 255 merkkiä.

16 String formula too complex
Jonolauseke on joko liian pitkä tai mutkikas. Lauseke tulisi jakaa osiin.

17 Can't continue
On yritetty jatkaa ohjelmaa joka on

1. pysäytetty virheen vuoksi,
2. muutettu suorituksessa tapahtuneen katkon aikana
3. tai jota ei ole ensinkään olemassa.

18 Undefined user function
On kutsuttu FN-toimintoa määrittämättä sitä ensin DEF FN -lauseella.

19 Device I/O error
Syöttö/tulostusvirhe nauhurin, kirjoittimen tai näyttölaitteen kanssa. Se on kohtalokas virhe, toisin sanoen BASIC ei selviä siitä.

20 Verify error
Käytössä oleva ohjelma poikkeaa kasetille tallennetusta ohjelmasta.

21 No RESUME
Syötetyssä virheentavoittamismenettelyssä ei ole RESUME-lausetta.

22 RESUME without error
On kohdattu RESUME-lause, jota ennen ei ole syötetty virheentavoittamismenettelyä.

23 Unprintable error
Virheen aiheuttajalle ei ole omaa virheilmoitusta. Syynä on yleensä ERROR, jolle ei ole määritetty virhekoodia.

24 Missing operand
Jossain lausekkeessa on operaattori, jonka perään ei ole sijoitettu operandia.

25 Line buffer overflow
Syötetyssä rivissä on liian monta merkkiä.

26 - 49 Unprintable errors
Näitä koodeja ei ole määritetty ja ne tulisikin varata BASICin laajennuksia varten.

50 FIELD overflow
FIELD-lause yrittää sovittaa useampia tavuja kuin OPEN-lauseessa oli hajasaantitiedostoon määritetty, tai FIELD-puskurin loppu tulee vastaan, kun yritetään suorittaa peräkkäis-I/O (PRINT# tai INPUT#) hajasaantitiedostoon.

51 Internal error
Sisäinen virhetoiminto; ehkä olisi tarpeen antaa asiantuntijan tarkistaa laite.

52 Bad file number
Lauseen tai käskyn viitenumeron mukainen tiedosto ei ole auki tai sitten viite ei mahdu MAXFILE-lauseen määrittämien tiedostonumeroiden alueelle.

53 File not found
Lause (LOAD, KILL tai OPEN) viittaa tiedostoon, jota ei ole muistissa.

54 File already open
Peräkkäistulostustilan OPENia käytetään tiedostoon, joka on jo avattu, tai KILLiä käytetään avoinna olevaan tiedostoon.

55 Input past end
INPUT-lause on suoritettu, kun tiedoston kaikki tiedot on jo syötetty tai tiedosto on tyhjä. Virheen välttää käyttämällä tiedoston loppumerkintänä EOF-toimintoa.

56 Bad file name
On yritetty käyttää sellaista tiedoston nimeä, jota ei voi käyttää esimerkiksi LOADin, SAVEn, KILLin, NAMEn tai muun kanssa.

57 Direct statement in file
ASCII-muotoista tiedostoa ladattaessa on kohdattu suoran käyttötilan lause. LOAD suoritetaan päätökseen.

58 Sequential I/O only
Hajasaantilause on osoitettu peräkkäistiedostoon.

59 File not OPEN
PRINT#:ssä, INPUT#:ssä tai muualla määritettyä tiedostoa ei ole avattu.

60 – 255 Unprintable Errors
Koodeja ei ole määritetty. Käyttäjä voi määrittää omat virhekoodinsa luettelon lopusta alkaen.

Liite C

Ruutueditorin toimintonäppäimet

KONT-ROLLI-NÄPPÄIN	ERITYIS-NÄPPÄIN	TOIMINTO
A		Ei suorita mitään, toiminnoton.
*B		Siirrä kohdistin edellisen sanan alkuun.
*C		Keskeytä kun MSX-BASIC odottaa syötettä.
*D		Toiminnoton
*E		Katkaise rivi (poista teksti loogisen rivin loppuun asti).
*F		Siirrä kohdistin seuraavan sanan alkuun.
*G		Beep-äänimerkki
H	Backspace	Palauttaa kohdistimen edelliseen asemaan poistaen merkin.
I	Tab	Tabuloi (siirrä seuraavaan tabuloituun kohtaan).
*J		Rivin siirto
*K	Home	Siirrä kohdistin "kotiin" ruudun vasempaan yläkulmaan.

***L**	CLS	Tyhjennä ruutu.
***M**	Return	Telanpalautus (syötä tuolloinen looginen rivi)
***N**		Liitä rivin loppuun.
***O**		Toiminnoton
***P**		Toiminnoton
***Q**		Toiminnoton
***R**	INS	Lisää väliin/päällekirjoitustila.
***S**		Toiminnoton
***T**		Toiminnoton
***U**		Tyhjennä looginen rivi.
***V**		Toiminnoton
***W**		Toiminnoton
***X**	Select	Toiminnoton
*		Toiminnoton
***Z**		Toiminnoton
[	ESC	Toiminnoton
*\	Nuoli oikeaan	Kohdistin oikeaan
*]	Nuoli vasempaan	Kohdistin vasempaan
*	Nuoli ylöspäin	Kohdistin ylöspäin
*–	Nuoli alaspäin	Kohdistin alaspäin
***DEL**	DEL	Poista kohdistimen kohdalla oleva merkki.

Kaikki asteriskilla (*) osoitetut näppäimet peruuttavat väliinkirjoitustilan, kun editori on väliinkirjoitustilassa.

Liite D

SV-BASICin ja MSX-BASICin erot

Spectravideon tietokoneiden SV318 ja SV328 kielimääritykset ovat miltei samat kuin MSX-BASICin. Eroja on vain kahdeksan. SV-BASIC käsittelee hieman eri lailla ruututiloja, ruudun leveyttä, toimintonäppäimiä, äänikanavaa, näppäinkilahdusta ja kirjoitinta. Lisäksi siinä käytetään kahta MSX-BASICille vierasta grafiikkakäskyä, GET ja PUT. Käymme ne lävitse järjestyksessä:

SCREEN<tila>[,<vaihtoehto>]

Vaikka MSX-BASICissa voi käyttää neljää ruututilaa, SV-BASICissa voi käyttää ainoastaan kolmea, jotka on määritelty seuraavasti:

SV-BASIC	**MSX-BASIC**	**Tila**
SCREEN 0	SCREEN 0	40 x 24 tekstitila
SCREEN 1	SCREEN 2	Suurresoluutiografiikkatila
SCREEN 2	SCREEN 3	Pienresoluutiografiikkatila

SV-BASIC tarjoaa vain yhden vaihtoehdon SCREEN-käskyssä. Vaihtoehto on käytettävän ruututilan mukainen.

Tekstitilan nollavaihtoehdossa toimintonäppäinnäyttö poistuu. Jos arvoksi annetaan jokin toinen, toimintonäppäinnäyttö on havaittavissa. Ja jälkimmäinen vaihtoehto onkin oletusarvona.

Kummassakin grafiikkatilassa vaihtoehdon avulla valitaan esitettävien spritejen koko.

0 valitsee 8 x 8 suurentamattomat spritet.
1 valitsee 8 x 8 suurennetut spritet.
2 valitsee 16 x 16 suurentamattomat spritet.
3 valitsee 16 x 16 suurennetut spritet.

Grafiikkaa käsittelevässä luvussa selvitetään tarkasti, mitä tarkoittavat spritet 8 x 8, 16 x 16 suurennettu ja suurentamaton.

WIDTH [**39**][**40**]
MSX-BASICissa on tekstitilaan valittavissa mikä tahansa rivileveys nollasta neljäänkymmeneen merkkiin. SV-BASICissa leveydeksi on ainoastaan vaihtoehdot 39 ja 40 merkkiä.

KEY [**ON**][**OFF**]
MSX-BASICissa KEY-käskyn avulla vaihdetaan näppäinten toimintaa osoittava rivi ruudun alareunasta joko näkyviin tai pois näkyvistä. SV-BASICissa käytetään edellä mainittuja ruutuvaihtoehtoja.

SOUND[**ON**][**OFF**]
SV-BASICin SOUND-kanavan voi kytkeä pois, mikä ei ole mahdollista MSX-BASICissa. Jälkimmäisessä SOUND on rajattu tuottamaan outoja, kiehtovia ääniä!

CLICK[**ON**][**OFF**]
MSX-BASICin SCREEN-käsky mahdollistaa näppäinäänen kytkemisen ja poistamisen. SV-BASICissa tätä tarkoitusta varten on oma käskynsä: CLICK ON saa äänen kuuluviin ja CLICK OFF poistaa sen.

PRINT
SV-BASICissa voi PRINT-lauseen avulla kirjoittaa mihin tahansa ruutuun mitä mieleen tulee. MSX-BASICin PRINT toimii tekstiruuduissa, mutta grafiikkaruuduissa se ei toimikaan, ja silloin täytyy turvautua tavallista mutkikkaampaan menettelyyn (katso PRINT-lauseen tarkkaa kuvausta 2. luvusta).

GET(<x1,y1>)–(<x2,y2>),<taulukon nimi>
Kun SV-BASICin grafiikkaruutuun on piirretty kuva, voi osan kuvasta tallentaa GET-käskyn avulla taulukkoon. Kun sitä käytetään PUTin kanssa, voi tuon tallennetun osan palauttaa haluamaansa paikkaan ruutuun. Asia tulee kaikkein selvimmin kuvatuksi lyhyellä ohjelmalla:

```
10 SCREEN 2
20 DIM A(45,45)
30 FOR X=10 TO 100 STEP 10
40 CIRCLE (X,X),X
50 NEXT X
60 GET (10,10)-(100,100),A
70 PUT (125,100),A,PSET
80 GOTO 80
```

Rivi 10 valitsee käyttöön suurresoluutiografiikan. Rivi 20 määrittää A-nimiselle taulukolle niin suuret mitat, että siihen mahtuu ruudusta tallennettava osa. Rivit 30 – 50 kertovat tietokoneelle, että sen tulee piirtää sarja ympyröitä, joista ensimmäisen säde on 10 pikseliä ja keskiö kohdassa 10,10; toisen säde on 20 ja keskiön kohta 20,20 ja niin edelleen, kunnes piirrettävän ympyrän säde on 100 ja keskiön sijainti 100,100. Rivi 60 käskee sitten ottamaan osan kuvasta – 10,10 – 100,100 – ja tallentamaan sen taulukkoon A. Rivillä 70 A asetetaan takaisin ruutuun vasempaan yläkulmaan kohtaan 125,100. PSET on yksi PUTin kanssa käytettävistä vaihtoehdoista, joilla tallennettu kuvio on muokattavissa ruutuun palauttamista varten.

PUT(<x,y>),<taulukon nimi>[,<vaihtoehto>]
PUT-käsky jäljentää kuvasta GETillä määritetyn osan haluttuun paikkaan ruutuun. PUTin rakenne on melko helppo: x ja y määrittävät ruudun vasemmasta yläkulmasta kohdan, johon jäljennettävä kuvio on sijoitettava, ja tiedoston nimi on sama kuin GETissä käytetty kuvion nimi. Kuvion uudelleenpiirtämistä varten voidaan käyttää seuraavia vaihtoehtoja:

PSET
PSET palauttaa kuvion ruutuun juuri sellaisena kuin se tallennettiin taulukkoon.

PRESET
PRESET piirtää kuvion ruutuun aivan samoin kuin PSET, paitsi että se vaihtaa taustan ja etualan värit.

AND
AND yhdistää kuvion värin ruudun väriin siten, että kuvio piirtyy vain niillä kohdin, missä molempien väri on sama.

OR
Piirrettävä kuvio limittyy entuudestaan ruudussa olevan kuvion kanssa siten, että molemmat ovat nähtävissä.

XOR
Jos piirrettävän kuvion pikseli on sama kuin ruudussa entuudestaan olevan, näkyviin tulee taustan väri. XOR on vaihtoehdon oletusarvo.

Vaikka vaihtoehdot tuli näin selvitettyä sanallisesti, ne käsittää parhaiten kokeilemalla itse. Helpoiten se käy palaamalla GETin näyteohjelmaan ja vaihtamalla rivin 70 PSET vuoron perään muihin vaihtoehtoihin.

MSX-standardi on kiistatta tärkeä tapahtuma kotitietokoneiden kehityksessä. Sen ansiosta jollekin MSX-tietokoneelle laaditut ohjelmat ja pelit sopivat kaikkiin muihin koneisiin, sillä ne kaikki ymmärtävät samaa MSX-BASICia.

Kaikkien MSX tutustuttaa sekä aloittelijat että kokeneet ohjelmoijat MSX-koneisiin, MSX-BASICin käskyjen nimiin, merkityksiin ja kykyihin sekä MSX-makrokieleen. Grafiikan käyttöön ja musiikin tuottamismahdollisuuksiin perehdyttyään kuka tahansa voi itse laatia pelejä ja säveltää.

Kokeilemalla kirjan piirros- ja maalausohjelmia, spriteja ja hyödyllisiä vihjeitä saa lisätietoja MSX:n vallankumouksellisista ominaisuuksista.

61

ISBN 951-35-3340-9