

mikro

számítógép

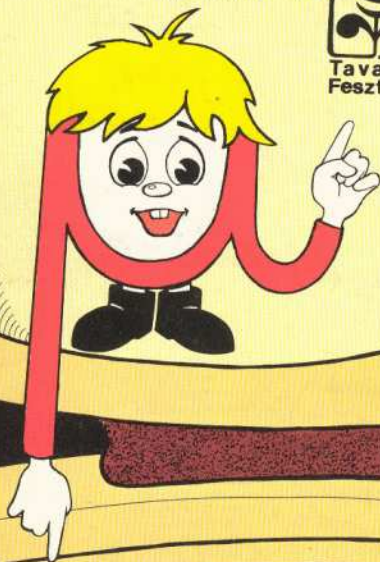
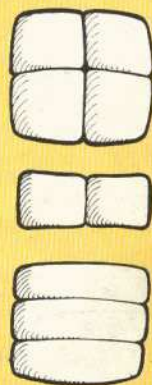
magazin

Ára: 30 Ft

1987. III. 19 - 25.



Tavaszi
Fesztivál



II. ORSZÁGOS
MIKROSZÁMÍTÓGÉPES
TALÁLKOZÓ



87

EGY CHIP-ETNYI AJÁNLAT:

SZÁMÍTÁSTECHNIKAI SZOLGÁLTATÁSOK
SZAKKÖNYVEKTŐL

A

PROGRAMOKIG
SZÁMÍTÓGÉPEK, KIEGÉSZÍTŐ ESZKÖZÖK,
KÉSZ PROGRAMOK, JÁTÉK-, NYELVI,
OKTATÓKAZETTÁK, LEMEZEK.
COMMODORE PC 20 SZÁMÍTÓGÉPEK
ÉS KONFIGURÁCIÓK LEASING ÜGYINTÉZÉSE

EGY CHIP-ETNYI JÖVŐ KÖNYVESBOLTJAINKBAN!

7621 Pécs, Jókai u. 25.

6600 Békéscsaba, Tanácsköztársaság u. 2.

3525 Miskolc, Tanácsház tér 14.

6720 Szeged, Lenin krt. 48.

9021 Győr, Molnár Ferenc u. 9.

4026 Debrecen, Hunyadi u. 8—10.

5000 Szolnok, Ságvári krt. 35.

9700 Szombathely, Mártírok tere 1.

8200 Veszprém, Cserhát u. 7.

MŰVELT NÉP
KÖNYVTERJESZTŐ
VÁLLALAT





mikro számítógép magazin

5. ÉVFOLYAM
1987/3. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány
a Tudományos-
és Informatikai
Intézet
együttműködve készül

A szerkesztőbizottság
vezetője:
Kovács Győző

E számunkat
szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levél cím
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap-
és Könyvkiadó Vállalat

Felelős kiadó:
dr. Petrus György
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkezelésű
hivataloknál
és a Posta Hírlapelőfizetési
és Lapellátási Irodáján
(1900 Budapest V.,
József nádor tér 1.)
vagy átutalással a 215-96 162
pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30. — Ft
Előfizetési díj:
egy évre 360. — Ft
fél évre 180. — Ft
Külföldön terjeszti
a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.



Szika Lapnyomda
Budapest (87-0010)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

μ'87	2
μINFORM	16
Vigyázat! Tolvaj!	28
Adok — veszek — cserélek	29
Olvastunk . . .	30
RAINBOW a μM-ban	34
Ki ad magyarázatot?	41
Egy számítástechnikai-gazdasági konzern	43

ISKOLA — SZÁMÍTÓGÉP

„Folyékony logika”	3
Gépi kódban	5
Megszakításvezérelt digitális óra	6
Pontszerű töltések erővonalképe	9

DIÁKROVAT

Hibaüzenetek olvasása	11
Térhatású ábrák	12

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód	13
OVERLAY — CHAIN — APPEND	13
Többök kezelése FORTH-ban	16

μPROGRAMOK

Házi módosítások a GENS3, MONS3 programokon	19
Kalkulációs program adóelszámolóhoz	24
Hasznos rutinok C64-re	25
Lapdobás	25
Kétütemű motor működése	26
Programbetöltés soros illesztőn keresztül	26
Háromdimenziós ábrázoló geometria	27

μKLUB

ZX Spectrum — helyi hálózatban	36
ZX Spectrum, mint digitális jeleket tároló oszcilloszkóp	37
Külső ROM C64-hez	38
A VC20 hardver adta lehetőségei	39
Menüprogram	40

SAKKPROGRAMOZÁS

Bitek és figurák	42
------------------	----

FÓRUM

	43
--	----

AZ OLVASÓ ÍRJA

	44
--	----

JÁTÉKPROGRAMOK

	45
--	----

KÖNYVEK

	46
--	----

HÍREK — ÉRDEKESSÉGEK

	48
--	----

Címképünk:
Sárközi Antoinette munkája

μ mikro számítógép
magazin



„Az elektronikus digitális számítógépek (sic!) kétségtelenül korunk legnagyobb jelentőségű műszaki alkotásai közé tartoznak. A feladatok, amelyek megoldására már ma is alkalmazták őket, még inkább pedig azok, amelyek megoldására évről-évre alkalmazhatók, megépítők, sokszor — mint például a fordítás és a sakkjáték esetében — egyenesen meghökkenítőek. (...) A hátrókat még korántsem látjuk világosan; annyi bizonyos, hogy az elektronikus számítógépek új fejezetet jelentenek a technika történetében.”

(Tarján Rezső: Gondolkodó gépek 1958.)

Tarján Rezsőtől, sokunk mentorának könyvéből — amely valószínűleg az első volt a magyar informatikai szakirodalomban — vettem a bevezető idézetet. Ő sajnos már nem érthette meg az informatikai gépeket és módszereket azt a társadalmi méretű elterjedését, amelyet az évenként rendezett Országos Mikroszámítógépes Találkozókon bemutatunk.

Amikor a szerkesztőségi cikkhez a rendelkezésre álló Tarján-írások között a megfelelő idézetet kerestem, csodálkozva fedeztem fel, hogy írásaiban — noha műszaki „jósásokra” szívesen vállalkozott — az informatikának mai mértékű elterjedését nem említette. Írt a sakkjátékok és a fordítógépek kivül az automatizált gyárról, a számítógépes árurendelésről, általában a gazdasági életben való alkalmazásokról, de arról, hogy viszonylag nagy teljesítményű gépek jelennek meg majd az iskolákban, sőt a lakásokban is, ennek „jóslására” sem ő, sem nagyon sokan mások sem vállalkoztak.

Az idén másodikban rendezzük meg a népszerű néven μ Találkozót, amelyen — részben a szakintézmények, részben pedig az amatőrök segítségével — azt mutatjuk be, hogyan hódítja meg az informatika mindazokat az alkalmazási területeket, amelyekre nemhogy 1958-ban, de még néhány évvel ezelőtt sem gondoltunk.

A μ '87-et éppen azzal szeretnénk megkülönböztetni például a BNV-től, hogy a látogatók az üzleti céllal rendezett kiállítást mellett az informatikának a kulturális életben, a családi körben, a tanításban és a tanulásban, és még ki tudja, milyen más területeken való alkalmazását is megláthatják.

Már a μ '86-on is meglepetést okozott, hogy milyen sokan érdeklődtek a számítógéppel komponált és a számítógép segítségével előadott zenei alkotások iránt. Egy egész ország csodálta meg az NJSZT számítógépes amatőreinek az alkotásait, amelyek még a professzionális számítógép-konstruktorok elismerését is kiválták.

Az idén egy lépéssel ismét továbblépünk, remélve, hogy a közönség szívesen fogadja nem csak bevált programjainkat, de az újakat is.

A μ '87-en az idén harmadszor rendezzük meg az „A számítástechnika mindenkié, a számítástechnika mindenkiért” — népszerűen rövidítve SzMSZM — kiállítást. Az idén is arra kértük a kiállítókat, hogy azokat az oleós eszközeiket, rendszereiket, programcsomagjaikat hozzák, amelyek széles körű érdeklődésre tarthatnak számot. Az idén először külföldi kiállítókat is meghívtunk, az ismert hazai gyártók és számítástechnikai intézmények mellett. Már a μ '86-on is igen nagy számban vettek részt kisvállalkozások, egy magánkiállítóknak is volt! Idén a kisvállalkozásoknak és a magánvállalkozásoknak különféle kedvezményeket adtunk, hogy a hazai „számítástechnikai paletta minden színe” megjelenjen a μ '87-en.

μ '87, a második Országos Mikroszámítógépes Találkozó

Az idén megrendezzük a megyei NJSZT szervezettek kiállítását is. A területi szervezetek maguk határozzák el, hogy mit fognak bemutatni. Az előzetes hírekéből úgy tűnik, hogy itt is a megyék szakmai eredményein lesz a hangsúly, ami annál is inkább fontos, mert bizony ma még nincs meg az a szervezeti lehetőség vagy piac, ahol egy-egy jó és vidéken készült program vagy műszaki alkotás megfelelő fórumot kaphatna.

A társadalom informatizálásának az egyik legfontosabb programja ma is az iskola-számítástechnika, de valószínűleg évekig az lesz még. Ismét bemutatjuk, hogyan alkalmazták az iskolákban a számítógépeket, pályázatot írtunk ki diákoknak, tanároknak és iskoláknak is, hogy mindnyájan megláthassuk, hogyan lehet az iskolákban, tanórákon, szakkörökben, de foglalkozásokon kívül is a számítógépeket alkalmazni.

Ha már pályázatról van szó, az idén is díjazni fogjuk a számítógépes amatőrök legszellemesebb és műszakilag legjobb alkotásait (számítógépek, perifériák, áramkörök megoldások stb.), hasonlóképpen az amatőr szoftverfejlesztők eredeti munkáit (operációs rendszer-elemek, segédprogramok, alkalmaz... programok stb.) is. Az idén is lehet pályázni az ún. háztartási programokkal. Az elnevezés pontatlan, ti. a nem számítógépes szakemberek pályázhatnak olyan alkalmazói programokkal is, amelyeket például saját hivatali munkájuk megkönnyítésére láttak ki, de elfogadjuk számítástechnikai szakemberek pályázatát is, ha olyan programot hoznak, amelyet ott-honi számítógépükön a ház körüli munkák megkönnyítésére használnak. Ugyanazzal a programmal akár két pályázatot is indulni lehet, például egy nyelvtanár pályázhat mondjuk egy angol segédjegyet gyakorló programmal az iskolai, de a háztartási program pályázatán is.

Hagyományosan nevezhetjük a Szekszárdi Garay János Gimnázium, az NJSZT és a μ M oktatási és játékprogram-pályázat győztesekinek a bemutatóját, ezeket a programokat a kiállítási látogatói természetesen ki is próbálhatják.

Nem hívhatjuk fel először a figyelmet az egészségkárosultak helyzetére, hiszen az informatika az egyik nagyon fontos eszköze lehet az egészségkárosultaknak, amellyel magasan kvalifikált munkát végezhetnek. Szeretnénk, ha egyre több olyan eszköz születne, amellyel a társadalomnak ezen nagyon hátrányos helyzetben élő tagjai számára a munkába állást meg lehet könnyíteni.

A rendezőbizottság úgy véli, hogy nagyon sok látogató fog vonzani az informatika human és művészeti alkalmazásának a bemutatása. Az idén a párizsi IRCAM munkatársait hívtuk meg zenei bemutatásra, akik részben elmondják, de meg is mutatják azokat az eredményeket, amelyeket mind a zene komponálásában, mind pedig zenei művek számítógépen való előadásában elérték.

Az MTA SzTAKI és az Új Impulzus által rendezett számítógépes grafikai kiállítást, a Digitartót a μ '87-en folytatjuk. Reméljük, hogy itt olyan új alkotásokat is be tudunk mutatni, amelyeket már az első, őszi kiállítás, illetve az SZKI — SCI — L művészeti pályázata inspirált.

Késégkívül érdekes színtöltője lesz a μ '87-nek a tv-és hangstúdió, amely a kiállítás tartama alatt folyamatosan műsort „sugároz” mind a μ '87, mind pedig az Utazás '87 látogatói részére. A Magyar Televízió Képzés-stúdiója is részt vesz a μ '87-en, a nézők az aktuális eseményekről a Képzésben kaphatnak folyamatosan tájékoztatást.

Már a μ '86-on is nagy sikere volt a „boltoknak”, ahol a látogatók nem csak nézhették a kiállított tárgyakat, de mindent meg is vásárolhattak. Az idén meghívást küldtünk mindazoknak a cégeknek, amelyek számítógépeket, alkatrészeket, perifériákat, de számítógépes médiumokat (szalag, floppy, leprellő, tisztítószeg stb.) árusítanak, valószínűleg az idén is jó üzleti forgalomra számíthatnak. A μ '86-on a vártnál nagyobb volt az érdeklődés a szakkönyvek iránt, ezért számítottunk a könyvesboltok és a könyvkiadással is foglalkozó cégek részvételére.

Az idén a Műszaki Könyvkiadóval közösen a μ '87-re időzítettük az első informatikatörténeti mű

H. H. Goldstine: „A számítógép Pascaltól Neumann-ig”

kiadását, amelyet először a μ '87-en lehet megvásárolni. A könyv megjelenése alkalmából a μ '87 vendége lesz a szerző, H. H. Goldstine is, akivel találkozót szervezünk.

Az idén is lesz börze, ahol bárki eladhatja vagy elcserélheti számítástechnikai eszközeit, programjait. Azt reméljük, hogy meg tudjuk szervezni a számítástechnikai intézményeknél elfekvő inkkurrens áruk folyamatos eladását is.

Az NJSZT Ifjúsági Bizottsága és Oktatási Szakosztálya a μ '87-tel egyidőben és a találkozó színhelyén rendezei meg az „Ifjú oktatók Kozma László konferenciáját”, amely alkalmából nyílt ülésen tárgyalják meg a számítógépekkel való oktatásnak, illetve az informatika tanításának a még ma is sok vitát kiváltó kérdéseit.

A μ '87 valamennyi programját nem tudjuk felsorolni, a szerkesztőségi cikk írásakor még valamennyit nem is ismerjük. Az idén meglehetősen sok ad-hoc programot is szervezünk, amelyről a napilapokból, illetve a Tv Képzésből értesülhetnek az érdeklődők. Új programok megszervezésére így még az „utolsó pillanatban” is van lehetőség, ezért olvasóink javaslatait is várjuk.

A μ '87 rendezői azt szeretnék, ha mind a „szereplők”, mind pedig a látogatók végül azzal a jó érzéssel összegyűjenek a látottakat, hogy a hazai számítástechnika ismét előbbre lépett a társadalom informatizálásában.

KOVÁCS GYÖZÖ



mennyiséget, amit át kell tölteni a tartályok bármelyikébe. Az áttöltés a szelepek nyitásával irányítható. A képernyő szemlélteti az eddigi műveleteket, a mennyiségeket, és hogy melyik két szelep van nyitva (2. ábra). További szabály, hogy mindig csak a legtöbb folyadékot tartalmazó tartályból lehet áttölteni. Amikor a tanuló elkészült a feladattal, a lépések számát tekintve összehasonlíthatja eredményét a gép minimál-megoldásával. Ha valaki nem tudná folytatni a műveleteket, akkor egy funkcióbillentyűt lenyomva „jó tanácsot” kaphat az éppen aktuális lehetőségek közül, mégpedig a legjobbat.

A feladat – akár a többi oktatóprogram – ismételhető, más adatokkal újraindítható, nehezebb fokozatokon folytatható, illetve egy másik program elérésének érdekében vissza lehet térni a lemezmemóriához.

A „Találd meg az utat!” című játék a megfigyelőképességet és a motorizálást fejleszti. A feladat egy kincsesháza vezető út megkeresése. A C64-re írt feladványban az utak hossza és variációja változó, egy-egy jó megoldás után pedig egyre bonyolultabb. Ha sikerül a botkormány segítségével az utat megtalálni, akkor még háromszor ugyancsak végig kell az úton menni a képen is kijelzett idő lejártá előtt (3. ábra)

Szintén a logikai készséget méri fel a „Csodalabirintus” című C64 szoftver, de itt a jó reflexekre és az áttekintőképességre is szükség van. Egy labirintusból kell minél gyorsabban kijutni, mert

Fogékony korban:

„Folyékony logika”

A Minneapolisban működő JMH SOFTWARE MINNESOTA nevű vállalkozás Apple, Atari és Commodore típusú személyi számítógépekre oktatászoftver-csomagot készített. Bemutatunk néhányat termékeiből.

A legkisebbek számára C64-re készült az Analógiák című program. A logika kifejlesztését is segíti, de a nyelvtanítás egyik segéd-rutinjaként jól hasznosul. Indítás után a feladatok alternatívái közül választhat a gyakorló.

Az egyik feladatsorozat: a mondatokba a hiányzó végződéseket vagy teljes szavakat kell beírni.

Például:

A Hold olyan éjszaka, mint a Nap
(The Moon is tonight as sun is to)

A másik feladatsorozatban szintén hiányos a képernyőn megjelenő mondat, de itt a tanuló négy lehetséges megoldás közül az egyiknek a jelét (A, B, C, D) adja meg válaszképpen.

Például:

Az olló a vágáshoz olyan, mint a toll a -hoz.

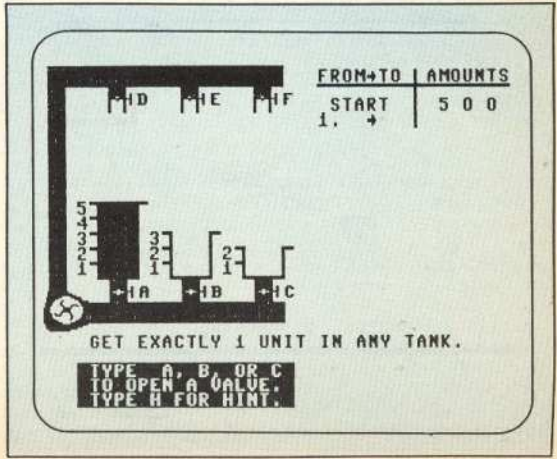
- A/ Írás
- B/ Festés
- C/ Törlés
- D/ Papír

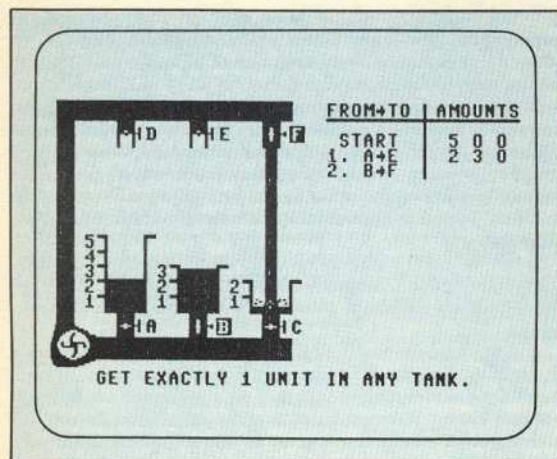
Rossz választásnál a számítógép letörlő a képről az illető tippet, így a diáknak eggyel kevesebb lehetősége marad. Ha a harmadik próbálkozás sem sikerül, akkor a gép a lehetséges és jó megoldásokon kívül más tippeket is megad.

Kilenc-tíz évesek logikai és matematikai képességeit fejleszti a „Folyékony logika” című Atari és Commodore gépre tervezett játék. A gyermek rendelkezésére áll egy szivattyú, a csövek, 3 db tartály, 6 db szelep és a folyadék (1. ábra). A program kijelöl egy viz-

a botkormánnyal irányítható kis figurát egy robot is veszélyezteti. A játékos veszt, ha a robot elfogja, vagy ha az hamarabb kijut a labirintusból. Kezdekör van egy kis idő az áttekintésre. Az első három menetben a robot mindig a „jobbkézszabályt” alkalmazza, azaz minden keresztet folyosón – ha mód van rá – jobbra fordul. A negyedik szinttől már véletlenszerűen változhat a robot iránya. A játék képe a 4. ábrán látható. Funkcióbillentyűk: S – újratekintés vagy START; R – ismétlés; C – folytatás; E – END.

1. ábra





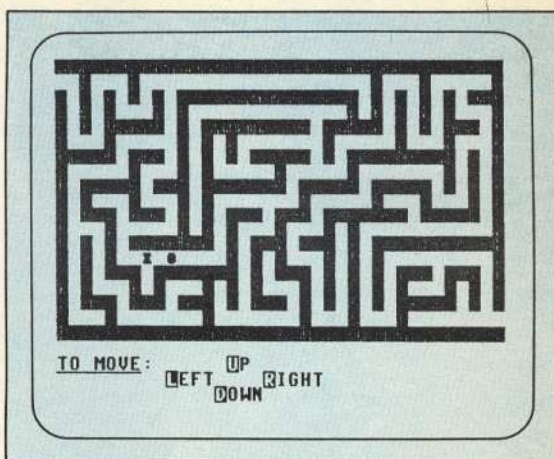
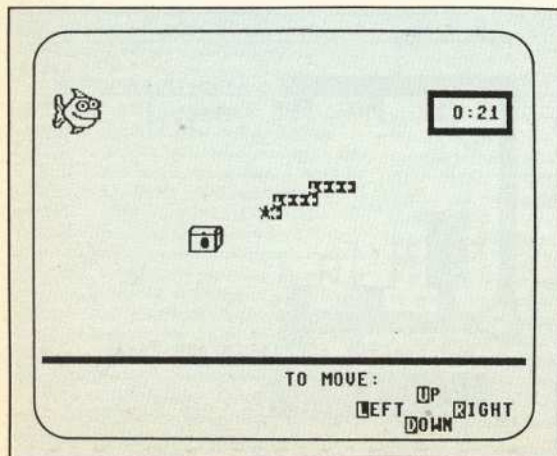
2. ábra

Az idegen nyelvek tanulását a szókincs felmérésével segítik. Összesen hét nyelv, közöttük a francia, a spanyol és a német közül lehet választani. A képernyőre kiírt angol szavak megfelelő idegen nyelvű párját kell megadni, és egyidejűleg két diák is dolgozhat egymással versenyezve. Egy-egy fordulóban 7 angol szót kell párosítani a választott nyelv megfelelő kifejezéseivel. Egy-egy ilyen csoportnak a szavait a számítógép egy nagyobb témakör kifejezőkészletéből emeli ki. A témakörök a következők: ételek; testrészek; napok, hónapok, dátumok; tantermi, szobai bútorok, tárgyak; mellénevek, jelzők.

Kisebbek számára az SI mértékegységrendszer metrikus átváltásait szemlélteti táblázatosan egy Apple gépre írt program, amely a nagyobbaknak már nehezebb feladatokat kínál. Az SI és más rendszerek közötti számításokat kell megoldaniuk (5. ábra).

A földrajzi tájékozódást teszi próbára a mindhárom említett gépre elkészített program, amikor megjelöl egy területet (állam, város, hegység stb.) a vaktérképen, majd megad egy nyílirányt. Ekkor

3. ábra



4. ábra

a tanuló az irány helyes vagy rossz (Y/N) voltát írja be. Ha körülbelül megfelelő a nyíl állása, akkor a botkormányval az adott területig kell elkörmányozni, majd megadni a hosszúsági és a szélességi fokokat.

A halmazelmélettel kapcsolatban is készült feladatok (6. ábra). Itt Venn-diagrammal szemléltetett halmazokon gyakorolhatók a logikai műveletek (konjunkció, diszjunkció, komplementshalmaz képzése). A program elindítása után az üres Venn-diagramok jelennek meg. Ezután a következő adatokat kéri a gép:

- Hány eleme legyen az „A” halmaznak?
- Add meg az „A” elemeit!
- Hány eleme legyen a „B” halmaznak?
- Add meg a „B” elemeit!

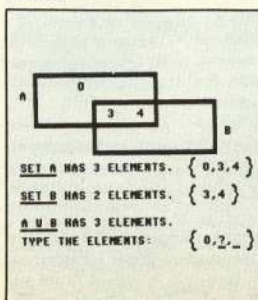
Ha a diák ezzel elkészült, akkor a logikai feladatok következnek. Helyes eredményhalmaz megadását a Venn-diagram látványos módosulása követi.

Az említettekén kívül szerepel a JMH szoftverlistáján összeadás, kivonás, szorzás, osztás (törtekkel is), a koordináta-rendszerrel foglalkozó program, de könyvtárkezelő, helyesírási, a keringési szervrendszert bemutató, sőt még zenei anyag is.

Az oktatás támogatásán kívül azért is fontos ez a kezdeményezés, hogy a számítástechnika az érdeklődők egyre fiatalabb nemzedékeinek gondolkodását csiszoló társává váljék.

HARGITAI PÁL

5. ábra



6. ábra

THE METER IS THE BASIC UNIT OF METRIC MEASUREMENT.

PREFIX	VALUE	TERM	SYMBOL
KILO	1000 METERS	KILOMETER	km
HECTO	100 METERS	HECTOMETER	hm
DEKA	10 METERS	DEKAMETER	dam
	1	METER	m
DECI	.1	METER	dm
CENTI	.01	METER	cm
MILLI	.001	METER	mm

FOR MORE INFORMATION ABOUT THE PREFIXES, SEE THE NEXT PAGE.

Gépi kódban

PRIMO

A Primo gépi kódú programozásának „felderítését” a G3 jelű programmal folytatva (1. lista), egy pontot rajzolhatunk ki a képernyőre. E feladat elvégzéséhez a 00131 címen kezdődő szubrutint hívjuk meg a gépi kódú programunkból.

A gép szoftverkönyve szerint e szubrutin

bemenő paramétereit az E és a D regiszterekben kell megadni: a korábban ismertett módon az (5,5) koordinátájú pontot az LD D, 05 és az LD E, 05 utasításokkal.

A szubrutin meghívása a CALL utasítással és az azt követő címzéssel történik.

Címzés készítése táblázattal. A táblázat

dőlt szedésű (kurzív) számai közül keresse meg azt a legnagyobbat, amelyik a címből levonva (–1)-nél nagyobb maradékot eredményez. (Esetünkben ez 0.) A maradékot kell a CALL utasítás után írni, majd azt a számot, amelyet a kiválasztott kurzív szám alatt látunk.

0	256	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4096	4352	4608	4864	5120	5376	5632	5888	6144	6400	6656	6912	7168	7424	7680	7936
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
8192	8448	8704	8960	9216	9472	9728	9984	10240	10496	10752	11008	11264	11520	11776	12032
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
12288	12544	12800	13056	13312	13568	13824	14080	14336	14592	14848	15104	15360	15616	15872	16128
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
16384	16640	16896	17152	17408	17664	17920	18176	18432	18688	18944	19200	19456	19712	19968	20224
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
20480	20736	20992	21248	21504	21760	22016	22272	22528	22784	23040	23296	23552	23808	24064	24320
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
24576	24832	25088	25344	25600	25856	26112	26368	26624	26880	27136	27392	27648	27904	28160	28416
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
28672	28928	29184	29440	29696	29952	30208	30464	30720	30976	31232	31488	31744	32000	32256	32512
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
32768	33024	33280	33536	33792	34048	34304	34560	34816	35072	35328	35584	35840	36096	36352	36608
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
36864	37120	37376	37632	37888	38144	38400	38656	38912	39168	39424	39680	39936	40192	40448	40704
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
40960	41216	41472	41728	41984	42240	42496	42752	43008	43264	43520	43776	44032	44288	44544	44800
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
45056	45312	45568	45824	46080	46336	46592	46848	47104	47360	47616	47872	48128	48384	48640	48896
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
49152	49408	49664	49920	50176	50432	50688	50944	51200	51456	51712	51968	52224	52480	52736	52992
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
53248	53504	53760	54016	54272	54528	54784	55040	55296	55552	55808	56064	56320	56576	56832	57088
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
57344	57600	57856	58112	58368	58624	58880	59136	59392	59648	59904	60160	60416	60672	60928	61184
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
61440	61696	61952	62208	62464	62720	62976	63232	63488	63744	64000	64256	64512	64768	65024	65280
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Megszakításvez

```

1 REM G3
100 C=0 : CC=0
105 CLS
110 DIM K%(20)
120 CC=VARPTR(K%(0))
128 REM : LDD,05 : LDE,05 : CALL,00131
129 REM : RET
130 POKECC,22,5,30,5,205,131,0,201
140 C=CALL(CC)
.....
A PROGRAM      A KOD
.....
LDD,05         . 22,5
LDE,05         . 30,5
CALL,00131     . 205,131,0
RET           . 201
.....
    
```

1. lista

```

10 REM G:ATSZAMOLO
20 INPUT "MI A SZAM?":S
30 PRINT "AZ ELSO SZAM :";
      S-256*INT(S/256)
40 PRINT "A MASODIK SZAM:";
      INT(S/256)
    
```

2. lista

```

1 REM G4
100 C=0 : CC=0
105 CLS
110 DIM K%(20)
120 CC=VARPTR(K%(0))
127 REM : LDD,05 : LDB,55 : LD(HL),B
128 REM : LDB,(HL) : CALL,00131
129 REM : DJNZ,(-9) : RET
130 POKECC,6,55,22,5,112,94,205,131,0,
      16,247,201
140 C=CALL(CC)
.....
A PROGRAM      A KOD
.....
LDB,55         . 6,55
LDD,05         . 22,5
LD(HL),B       . 112
LDE,(HL)       . 94
CALL,00131     . 205,131,0
DJNZ,(-9)     . 16,247
RET           . 201
.....
    
```

3. lista

Címzés készítése programmal. A 2. listán látható program segítségével a címzés meghatározható.

Természetesen egy pont kirajzolásánál nem érződik a gépi kódú programrész sebessége. A G4 jelű program (3. lista) viszont már egy vonalat rajzol ki a képernyőre. A kirajzoló ciklus ismétlési számát (a vonal hosszát) a B jelű regiszterben adjuk meg: (LD B,55).

A DJNZ utasítás csökkenti a B regiszter értékét 1-gyel. Ha B regiszter tartalma 0, akkor a program tovább fut. Ha B regiszter tartalma >0, akkor a program futása a DJNZ utasítás utáni értéknek megfelelő ugrással folytatódik. Esetünkben ez (-9), ami azt jelenti, hogy az LD D,05 utasításra ugrunk vissza. A D és az E regiszterek ismételt feltöltése azért szükséges, mert a 00131-es címen kezdődő szubrutin e regiszterekben lévő értékeket megváltoztatja.

A negatív számokat a következő módon „kódolva” lehet megadni:
(KÓD)=256+(negatív szám)

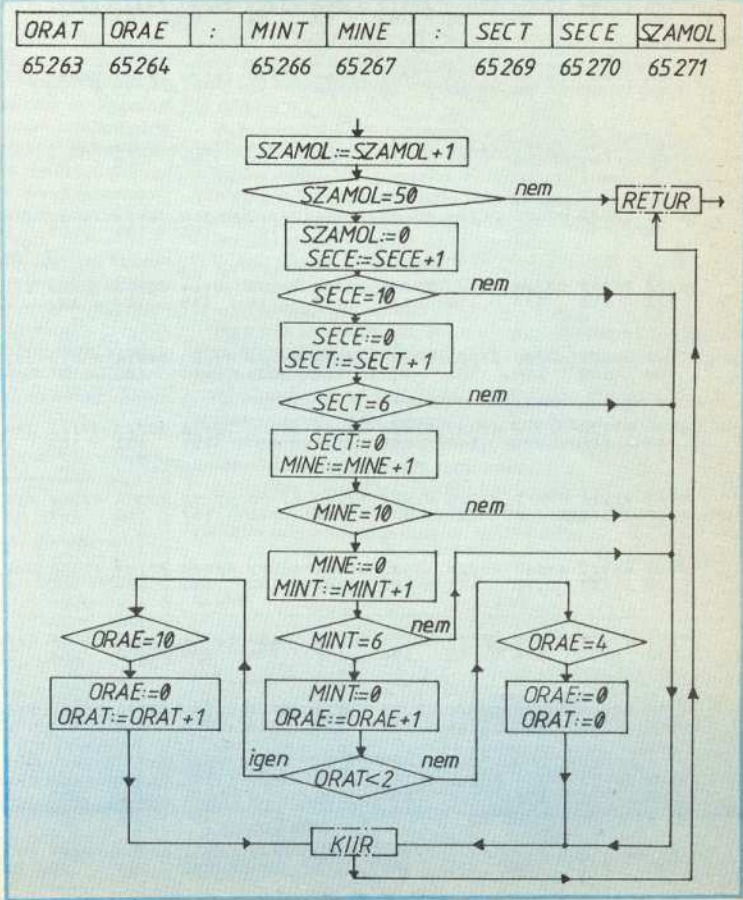
Például esetünkben: 256+(-9)=247
SOMOGYI GYÖRGY

Mikor és mire jó?

A BASIC program (1. lista) elindítása után a képernyő jobb felső sarkában megjelenik a 6 digitos óra alaphelyzetből — 00:00:00 — indítva, és a menü. A számbilentyűk segítségével a következő lehetőségek közül választhatunk: az órát megállíthatjuk vagy újra elindíthatjuk; a kijelzést letilthatjuk vagy engedélyezhetjük, miközben az óra jár; beállíthatjuk (a kijelzés 24 órás); tárolhatunk egy adott időpontot, és ahhoz képest mérhetünk időtartamot. A program csak egy általam kigondolt demonstráció.

Az óra a NEW parancs hatására megáll, tehát ilyenkor újra kell indítani. LOAD,

Az óra vezérlő rész működésének folyamatábrája



SAVE, BEEP esetén a parancs végrehajtása után magától megy tovább. Az óra egyébként független az éppen futó programtól, csak ez a négy parancs van rá hatással. Alkalmos egy programon belül az idő kijelzése, időtartam mérése stb. A későbbiekben leírom a változtatási, beállítási lehetőségeket is.

A gépi kódú rutin

A ZX—Spectrum alaphelyzetben, bekapcsolás után I-es megszakítási módban (IM 1-ben) dolgozik. Ez azt jelenti, hogy 20 ms-onként megszakítja az éppen folyó feladatának végrehajtását és egy „restart” rutinra ugrik (rst 38H), ahol a FRAMES

relt digitális óra

ZX-SPECTRUM

I. lista

```

10 REM az ORA lehetosegeit bemutato prog
ram
15 BORDER 6: PAPER 6: BRIGHT 1: INK 0
20 CLS: PRINT AT 3,10: INVERSE 1:"LEHET
OSEGEK": INVERSE 0
30 PRINT AT 6,0:"1 az ORA (el)inditasa:";
40 PRINT AT 8,0:"2 az ORA megallitasa:";
50 PRINT AT 10,0:"3 a kiiras letiltasa:";
60 PRINT AT 12,0:"4 a kiiras engedelyeze
se:";
70 PRINT AT 14,0:"5 az ORA beallitasa--";
80 PRINT AT 16,0:"6 idopont kiolvasasa-";
90 PRINT AT 18,0:"7 idotartam meres-";
100 LET K=INKEY: IF K="" THEN GOTO 100
110 IF CODE K<49 OR CODE K>55 THEN GOTO
100
120 GO SUB ((CODE K)-48)*100+8000
130 GOTO 100
140 STOP
8100 REM az ORA elinditasa
8110 PRINT AT 8,26:" "": PRINT AT 6,26:
"(-)";
8120 RANDOMIZE USR 65122
8130 RETURN
8200 REM az ORA megallitasa
8210 PRINT AT 6,26:" "": PRINT AT 8,26:
"(-)";
8220 RANDOMIZE USR 65115
8230 RETURN
8300 REM a kiiras letiltasa
8310 PRINT AT 12,26:" "": PRINT AT 10,
26:"(-)";
8320 POKE 65206,24: POKE 65207,45
8330 RETURN
8400 REM a kiiras engedelyezese
8410 PRINT AT 10,26:" "": PRINT AT 12,
26:"(-)";
8420 POKE 65206,0: POKE 65207,0
8430 RETURN
8500 REM az ORA beallitasa
8510 PRINT 0: INPUT "Az idopont 06:15:05
alakban=": LINE T
8520 LET ORAT=65263: FOR I=0 TO 7: PRINT
AT 14,24+I: CHR$(CODE T*(I+1)): POKE ORAT
+1, CODE T*(I+1): NEXT I
8530 RETURN
8600 REM az idopont kiolvasasa
8610 DEF FN P(T)=(PEEK (T)-48)*100+ PEEK
(T+1)-48
8620 DEF FN Q(U)=360*FN P(U)+60*FN P(U+3)
+FN P(U+6)+0.02*PEEK 65271
8630 LET KEZDET=FN Q(65263)
8640 FOR I=0 TO 7: PRINT AT 16,24+I:CHR$(
PEEK(65263+I)): NEXT I
8650 PRINT AT 18,21:" "":
8660 RETURN
8700 REM idotartam meres
8710 LET VEG=FN Q(65263): LET DELTA=VEG-

```

```

KEZDET
8720 LET H=INT (DELTA/3600): LET M=INT (
(DELTA-H*3600)/60): LET S=DELTA-H*3600-M*
60: LET S=(INT (S*100))/100
8730 PRINT AT 18,21:" "": PRINT
AT 18,21:H:"":M:"":S;
8740 RETURN
8998 STOP
8999 CLEAR 65100
9000 LET A="fff3f5c5d5e5dde5fde521f7fe3
47efe323869360021f7fe06022b343e3abe202c36
302b343e36be202336302b10eb"
9010 LET B="2b342b3e32be23200c3e34be201
036302b363018093e3abe200436302b34"
9020 LET C="00000650811184021efff6004fdd
090608dd7e001214dd2310f7d11323c110db"
9030 LET D="fde1dde1eld1c1f1fbc930303a3
0303a303000"
9033 REM *****
9035 LET KONTR=0: LET INV=65129: LET M=A+
B+C+D+*": LET A=INV
9040 LET P=16*(CODE M*-48-(39*(M>2)"E"))
+CODE M*(2)-48-(39*(M>2)"E"))
9050 LET KONTR=KONTR+P: POKE A,P
9060 LET M=M*(3 TO )
9070 IF M="" THEN GOTO 9100
9080 LET A=A+1
9090 GOTO 9040
9100 PRINT "Az ellenorzo szam=":KONTR
9105 REM +++++
9110 DATA 62,63,237,86,237,71,201,62,9,
237,71,237,94,201
9115 RESTORE 9110
9120 FOR I=65115 TO 65128: READ A: POKE
I,A: NEXT I
9130 REM +++++
9150 REM a szamjegyek UDG kodjai
9160 REM #####
9170 DATA 24,36,36,36,36,36,24,0
9171 DATA 4,4,4,4,4,4,4,0
9172 DATA 28,4,4,28,32,32,60,0
9173 DATA 28,4,4,28,4,4,28,0
9174 DATA 32,32,36,60,4,4,4,0
9175 DATA 60,32,32,56,4,4,56,0
9176 DATA 24,32,32,56,36,36,24,0
9177 DATA 24,4,4,4,4,4,4,0
9178 DATA 24,36,36,24,36,36,24,0
9179 DATA 24,36,36,28,4,4,24,0
9180 DATA 0,0,24,0,24,0,0,0
9185 RESTORE 9170
9190 FOR I=65272 TO 65359: READ A: POKE
I,A: NEXT I
9199 REM *****
9200 REM Koozs Tamas Kapuvar 1986.X.21.
9201 REM *****
9210 STOP
9999 SAVE "time" LINE 8999

```

```

1      #Az ORA elindítást és
2      #megállítást végző
3      #rutinok
4      #KOOSZ TAMAS Kapuvar 1986
5
6 FE5B 3E3F      LD A, 63      #IM 1 beallitasa
7 FE5D ED56      IM 1          # az ora kikapcsolasa
8 FEF3 ED47      LD I, A
9 FE61 C9        RET
10 FE62 3E09     LD A, 9      #IM 2 beallitasa
11 FE64 ED47     LD I, A      #az ora elinditasa
12 FE66 ED5E     IM 2
13 FE68 C9        RET
14
15 #IM 2 alkalmazaval a ZX Spectrum
16 #a 9*256+255-os cimben levo
17 #szamnak (ketbyte-os) megfelelo
18 #cimre ugrik es vegrehajtja az
19 #ott talalhato rutint. Ez most
20 #konkretan a 65129.
21 #A gepi kodu program elhelyezese
22 #felott ki kell adni a CLEAR 65100
23 #parancsot, mert ekkor a NEW utan
24 #sem torlodik az ott elhelyezett
25 #rutinunk. NEW utan IM 1-es modba
26 #kerul a gep, ekkor az ora megall.
27 #RANDOMIZE USR 65122 kiadasa utan
28 #azonban ujra elindul.
29
30      END

```

a billentyűzetten keresztül. A regiszterek tartalmának elmentése mindig fontos, mert a rutinból való visszatéréskor hiányuk zűrvárt eredményezhetne. A lista ezután két fő részre bontható: a számláló—vezérlő és a kiíró részre. A rutin végén lévő kilenc bájtból az utolsóban (SZAMOL) tároljuk az eltelt 20 ms-ok számát; az első nyolcban a megjelenő számjegyek és ”.” ASCII kódját. A bemutatott blokkdiagram és a megjegyzések segítik a vezérlő rész egyébként sem nehéz működésének megértését.

Kiírásra csak minden másodpercben kerül sor, de ez a 3. lista 19. sorában lévő 69H szám 3AH-ra cserélésével megváltoztatható: ui. ilyenkor a KIIR részre ugunk, azaz minden ötven másodpercben a digitális óra valamennyi számjegye újra kiíródik, ami még „állandóbb” kijelzést eredményez.

Ha a kiírást le szeretnénk tiltani, akkor az 57—58-as sorok két NOP utasítását JR RETUR-ra cseréljük (18H és 2DH). A 60-as sorban a DE regiszterpárban az óratízesek karaktere első bájtnak képernyő RAM-beli címe található (4018H=16408). Ezt átírva, a kijelzés egy-egy képernyőharmadon belül tetszőlegesen áthelyezhető.

A kiírt számjegyek és a kettőspontkaraktereket leíró nyolc-nyolc bájttárolóhelynek kezdőcímét a 66-os sorban az IX regiszterpárba töltjük. Itt saját tervezésű karaktereket használunk; a tárolás kezdőcíme FEF8H=65272 dec. Ha megelőgszünk a

2. lista

rendszerváltozókat állítja (lásd az eredeti gépkönyv 18. fejezetét), majd a billentyűzet figyelése, kódolása következik. Mindezek után visszatér eredeti feladatához.

Mi áttérhetünk egy másik, maszkolható — vagyis szoftverrel beállítható — megszakítási (interrupt) módra, a 2-esre. Ez a programozó számára nagyon hatékony lehet, mivel lehetőséget nyújt saját rutinjainak „állandó”, minden megszakításkor egyszeri futtatására.

Az IM 2-ben a megszakítójel után a PC tartalma elmentődik, majd a gép az (A)*256+255 címen kezdődő kétbájtos számon mint címen tárolt gépi kódú program végrehajtásába kezd. Ennek lefutása után visszatér félbeszakított, eredeti munkájához.

A 2. listán két gépi kódú rutin található. Az első az IM 1, a második az IM 2 beállítását végzi. A ZX-Spectrumnál IM 1-ben az I regiszter tartalma 63. A jelen programban IM 2-ben az I regiszterbe 09H-t töltünk; hatására a gép megnézi, hogy a 9*256+255=09FFH címen milyen kétbájtos szám található. Ezt a PEEK függvényvel magunk is megkereshetjük: LET A=PEEK(2559)+256*PEEK(2560); PRINT A

Az A értéke 65129 lesz; tehát megszakításkor erre a címre ugrik a gép, és az itt található rutint hajtja végre. A-nak ilyen választása azért jó, mert a 48 k-s Spectrum RAM-jának a végére esik. Ha a RAMTOP-ot 65100-ra állítjuk be, akkor kb. 400 bájttal áll a gépi kódú programunk rendelkezésére.

A 3. lista az óra vezérlését és a kiírást végző gépi kódú program. Mindjárt elsőnek az RST 38H utasítás látható; enélkül megszűnne kapcsolatunk a számítógéppel

3. lista

```

1      #A ZX SPECTRUM IM 2-es
2      #megszakításmodjot
3      #felhasználó digitális
4      #kvarcponosságú ORA
5      #KOOSZ TAMAS Kapuvar 1986
6
7      ORG 65129
8      RST 38H
9      DI
10     PUSH AF
11     PUSH BC
12     PUSH DE
13     PUSH HL
14     PUSH IX
15     PUSH IY
16     LD HL, SZAMOL #.02 sec-kent
17     INC (HL)
18     LD A, (HL)
19     CP 50
20     JR C, RETUR #ha (A)<50
21     LD (HL), 0 # (SZAMOL):=0
22     LD HL, SECE+1
23     LD B, 2 #ismetlesek szama
24     CIKLUS: DEC HL #HL a SECE-re mutat
25     INC (HL)
26     LD A, 58 #CHR*(58)="9"+1
27     CP (HL)
28     JR NZ, KIIR
29     LD (HL), 48 #SECE:=0 (vagy MINE)
30     DEC HL #HL a SECT(vagy MINT)-re
31     INC (HL)
32     LD A, 54 #CHR*(54)="6"
33     CP (HL)
34     JR NZ, KIIR
35     LD (HL), 48 #SECT:=0 (vagy MINT)
36     DEC HL
37     DJNZ CIKLUS
38     DEC HL #HL az ORAE-re mutat
39     INC (HL)

```

PTA-4000

Pontszerű töltések erővonalképe

A Coulomb-törvény értelmében a töltések a térben erőhatást fejtenek ki egymásra. A térnek az a része, amelyben ez az erő hat, az elektromos tér vagy mező.

Az elektromos térerősség a teret jellemző vektormennyiség. Iránya a tér valamely pontjában az oda elhelyezett pozitív pontszerű próbatöltésre ható erő irányával egyezik meg, nagysága a fellépő erő és a próbatöltés nagyságának hányadosa. Egyetlen nyugvó pontszerű töltés elektromos térerősségének nagysága t töle r távolságban: $E = kQ/r^2$. Ha a teret több töltés hozza létre, akkor az egyes pontoknál a térerősség nagysága az egyes töltésektől származó térerősségek vektori összege.

Az elektromos erőteret Faraday nyomán erővonalakkal szemléltetjük. Ezek olyan görbék, amelyekhez húzott érintő a tér minden pontjában az ott uralkodó elektromos térerősség irányába esik. A térerősség nagyságát az erővonalak merőleges egységnyi felületet metsző erővonalak száma jellemzi. Az elektrosztatikus erőter erővonalai mindig töltésekből indulnak és töltéseken végződnek. Irányításuk a megállapodás szerint olyan, hogy az erővonalak pozitív töltésekből indulnak és negatív töltéseken végződnek.

```

39 FE9A 2B      DEC HL      ;HL az ORA1-re mutat
40 FE9B 3E32   LD A, 50   ;CHR*(50)="2"
41 FE9D BE     CP (HL)
42 FE9E 23     INC HL     ;HL az ORAE-re mutat
43 FE9F 200C   JR NZ, IDE ;ha ORAT<2
44 FEA1 3E34   LD A, 52   ;CHR*(52)="4"
45 FEA3 BE     CP (HL)
46 FEA4 2010   JR NZ, KIIR
47 FEA6 3630   LD (HL), 48 ;ORAE:=0
48 FEAB 2B     DEC HL
49 FEA9 3630   LD (HL), 48 ;ORAT:=0
50 FEAB 1809   JR KIIR
51 FEAD 3E3A   IDE: LD A, 58 ;CHR*(58)="9"+1
52 FEAF BE     CP (HL)
53 FEB0 2004   JR NZ, KIIR
54 FEB2 3630   LD (HL), 48
55 FEB4 2B     DEC HL
56 FEB5 34     INC (HL)
57 FEB6 00     KIIR: NOP      ;van KIIRAS
58 FEB7 00     NOP      ;ha JR RETUR - nincs
59 FEB8 0608   LD B, 8   ;ORAT-tól SECE-ig
60 FEBA 111840 LD DE, 4018H ;a 0.sor 24.oslop
61 FEBD 21EFFE LD HL, ORAT
62 FEC0 C5     VISSZA: PUSH BC
63 FEC1 D5     PUSH DE
64 FEC2 7E     LD A, (HL)
65 FEC3 E60F   AND 15   ;ASCII-bol szamama alakit
66 FEC5 DD21F8FE LD IX, MINTA
67 FEC9 CB27   SLA A
68 FECB CB27   SLA A
69 FECD CB27   SLA A    ;szorzas 8-cal
70 FECF 0600   LD B, 0
71 FED1 4F     LD C, A

72 FED2 DD09   ADD IX, BC ;a minta kezdete IX-ben
73 FED4 0608   LD B, 8   ;egy karakter = 8 byte
74 FED6 DD7E00 CIKL: LD A, (IX+0)
75 FED9 12     LD (DE), A
76 FEDA 14     INC D    ;egy sorral lejjebb
77 FEDB DD23   INC IX
78 FEDD 10F7   DJNZ CIKL
79 FEDF D1     POP DE
80 FEE0 13     INC DE
81 FEE1 23     INC HL
82 FEE2 C1     POP BC
83 FEE3 10DB   DJNZ VISSZA
84 FEE5 FDE1   RETUR: POP IY
85 FEE7 DDE1   POP IX
86 FEE9 E1     POP HL
87 FEEA D1     POP DE
88 FEEB C1     POP BC
89 FEED F1     POP AF
90 FEED FB     EI
91 FEEE C9     RET
92 FEEF 30303A30 ORAT: DB 48,48,58,48,48,58,48
93 FEF3 303A30
94 FEF6 30     SECE: DB 48
95 FEF7 00     SZAML: DB 0
96 FEF8 00     MINTA: DB 0
97 ;FEF8-tol a 0,1,2,...,9,= karakterek
98 ;10*8 byte-janak tarolasa
99 END
    
```

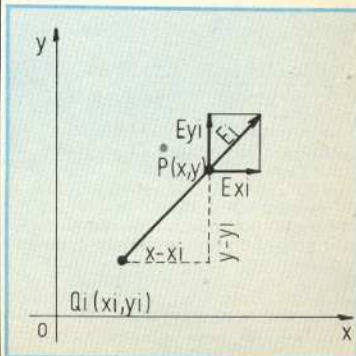
Spectrum saját számjegyeivel, akkor ide 80H, 3DH-t írjunk (3D80H=15744 dec).

Ha mindent kipróbáltunk és tárolni akarjuk a gépi kódú rutinjainkat, akkor ezt a BASIC program begépelése és GO TO 8999 után — ui. ekkor a gépi kódokat elhelyeztük a RAM-ban — SAVE "ÓRA" CODE 65115, 245 paranccsal megtehetjük. A visszatöltés CLEAR 65100:LOAD "ÓRA" CODE 65115, 245-tel lehetséges.

Felhasználói programért cserébe mind a két programot szívesen másolom kazettára annak, aki címre megkért kasszétát küld, megjelölve az üres helyeket.

KOÓSZ TAMÁS

tanár
Kapuvár, Fürdő u. 22.
9330



Ha a teret létrehozó töltések összege nem zérus, az erővonalakat úgy tekintjük, hogy egy megfelelő nagyságú és előjelű, végtelen távoli töltésből is indulnak, illetve ott is végződnek az erővonalak. Az erővonalak

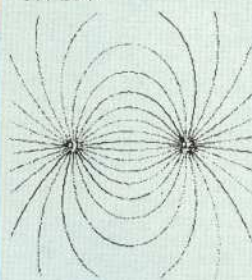
nem valóban létező fizikai objektumok, bevezetésükre csak a szemléletesség céljából volt szükség. Nem a töltések nagyságáról, hanem egymáshoz viszonyított arányokról és elhelyezkedésükről adnak képet.

A program az erővonalakat a következőképpen határozza meg. Adott pontban összegzi az egyes töltések által keltett térerősségek x és y összetevőit, majd az így meghatározott eredő térerősségvektor irányába

```

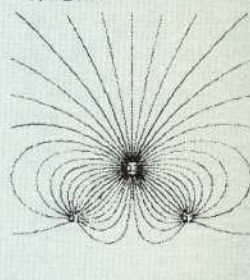
5: CLEAR : ON ERROR GOTO 170
10: INPUT "TÖLTÉSEK SZÁMA="; N; "Y TÁR
TOMÁNY="; W; N=N-1: DIM X(N), Y(N), Q
(N)
20: GRAPH : CSIZE 1: GLCURSOR (107, -W)
: SORGN
30: FOR I=0 TO N: WAIT 0
35: CLS : PRINT I+1; ". X KOORDINATA=";
: INPUT X(I)
40: CLS : PRINT I+1; ". Y KOORDINATA=";
: INPUT Y(I)
50: CLS : PRINT I+1; ". Q TÖLTÉS=";
: INPUT Q(I): QQ=QQ+Q(I)
60: CLS : COLOR (1+2*(Q(I)>0)):
GLCURSOR (X(I)-5, Y(I)-3): LPRINT
"Q"; STR$(I+1): NEXT I
65: INPUT "IC=BOL INDULO EROVONALAK:
"; M; "MOD (0, 1) =" ; G
70: QQ=SGN QQ+(QQ=0): RADIANS : K=5000:
COLOR 2: FOR S=0 TO N: IF SGN (Q(S)
)*QQ<0 GOTO 100
80: FOR J=1 TO M*ABS Q(S): Q=PI/M*2*KJ/
ABS Q(S)
90: X=6*X(I)+Q+X(S): Y=6*Y(I)+Q+Y(S):
GLCURSOR (X, Y): H=0
100: Q=0: P=0: I=0
105: R=(X(I)-X)*(X(I)-X)+(Y(I)-Y)*(Y
(I)-Y): IF R<35.56 GOTO 170
110: E=K*Q(I)/R: R=JR
120: O=O+E*(X-X(I))/R: P=P+E*(Y-Y(I))/
R: I=I+1: IF I<N GOTO 105
125: E=I*(O*O+P*P)
130: DL=3: IF H=1 LET DL=DL*((X*X+Y*Y)/
14000)
140: X=X+QQ*O/E*DL: Y=Y+QQ*P/E*DL
150: REM
160: ON H+1 GOTO 200, 230
170: NEXT J
180: NEXT S: GLCURSOR (-107, -W)
190: TEXT : COLOR 0: LF 2: CSIZE 1: FOR I
=0 TO N
192: LPRINT "Q"; STR$(I+1); "="; Q(I); "
C"; : LCURSOR 11
194: LPRINT "X"; STR$(I+1); "="; X(I); :
LCURSOR 23
196: LPRINT "Y"; STR$(I+1); "="; Y(I)
198: NEXT I: LF 4: END
200: LINE -(X, Y): IF ABS X<108 AND ABS
Y<W GOTO 100
210: IF G=0 LET H=1: GOTO 100
220: GOTO 170
230: PRINT X, Y: IF ABS X<108 AND ABS Y<
W GLCURSOR (X, Y): H=0
240: GOTO 100
    
```

1. ÁBRA



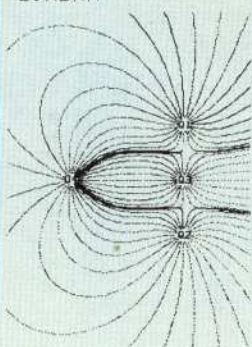
Q1= 1C X1= 50 Y1= 0
Q2= 1C X2= 50 Y2= 0

4. ÁBRA



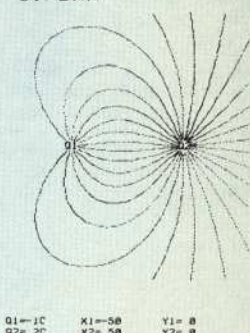
Q1= 2C X1= 50 Y1= 0
Q2= 2C X2= 50 Y2= 0
Q3= 5C X3= 0 Y3= 20

2. ÁBRA



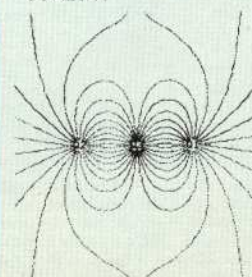
Q1= 1C X1= 50 Y1= 50
Q2= 1C X2= 50 Y2= 50
Q3= 1C X3= 50 Y3= 0
Q4= 1.5C X4= 50 Y4= 0

5. ÁBRA



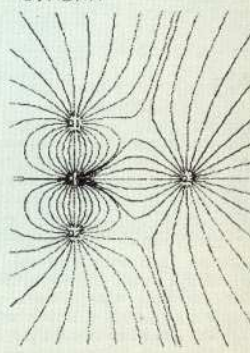
Q1= 1C X1= 50 Y1= 0
Q2= 2C X2= 50 Y2= 0

3. ÁBRA



Q1= 2C X1= 50 Y1= 0
Q2= 2C X2= 50 Y2= 0
Q3= 3C X3= 0 Y3= 0

6. ÁBRA



Q1= 1C X1= 50 Y1= 50
Q2= 1C X2= 50 Y2= 50
Q3= 1C X3= 50 Y3= 0
Q4= 1.5C X4= 50 Y4= 0



HIBAÜZENETEK OLVASÁSA

Gyakran okoz bosszúságot — főleg fájlkezelés közben —, hogy a DOS hibaüzeneteit csak programból lehet olvasni. Ezen segít a rutinom. A gép bekapcsolása után töltjük be a BASIC-betöltőt, és indítsuk el. Ha olvasni akarjuk a DOS üzenetét, adjuk

ki a SYS 49152 utasítást, és a szöveg a képernyőre íródik. Ha csak a gépi kódú részt vesszük fel, majd töltjük be, akkor új program írása előtt adjuk ki a NEW parancsot.

MARÓTI BENEDEK

DISK STATUS

SEITE: 1

```

C000          0          *#49152          ; DISK STATUS - #C000
FFBA         1  SETLFS =65466 ; #FFBA
FFBD         2  SETNAM =65469 ; #FFBD
FFC0         3  OPEN   =65472 ; #FFC0
FFC6         4  CHKIN  =65478 ; #FFC6
FFCF         5  CHRIN  =65487 ; #FFCF
FFD2         6  CHRUT  =65490 ; #FFD2
FFB7         7  READST =65463 ; #FFB7
FFE7         8  CLALL  =65511 ; #FFE7

C000 A9 0F      11      LDA #15          ; LOGIKAI FILE SZAM
C002 A2 08      12      LDX #8           ; EGYSEG SZAM
C004 A0 0F      13      LDY #15          ; MEGNYITASI MOD
C006 20 BA FF   20      JSR SETLFS      ; PARAMETEREK BEIRASA
C009 A9 00      21      LDA #0           ; FILENEV HOSSZA
C00B 20 BD FF   30      JSR SETNAM      ; FILENEV BEIRASA
C00E 20 C0 FF   40      JSR OPEN       ; A FILE MEGNYITASA
C011 A2 0F      41      LDX #15          ; MASODLAGOS CIM
C013 20 C6 FF   50      JSR CHKIN     ; A 15-OS INPUTCSATORNA
C015 20 CF FF   60 W    JSR CHRIN     ; EGY KARAKTER OLVASASA
C019 20 D2 FF   70      JSR CHRUT     ; A KARAKTER KIIRASA
C01C 20 B7 FF   80      JSR READST    ; ST BEOLVASASA
C01F C9 00      81      CMP #0           ; HA ST = 0
C021 F0 F3      85      BEQ W           ; AKKOR UGRAS VISSZA
C023 20 E7 FF   90      JSR CLALL     ; A CSATORNA LEZARASA
C026 60          95      RTS           ; VISSZATERES BASIC-BE
C027          99      .END           ; VEGE
    
```

ZEILEN:26 SYMBOLE:9 FEHLER:0

```

CHKIN =FFC6 CHRIN =FFCF CHRUT =FFD2 CLALL =FFE7 OPEN =FFC0
READST =FFB7 SETLFS =FFBA SETNAM =FFBD W =C016
    
```

```

1 FOR I=49152 TO 49190: READ A:POKE I,A: S=S+A
2 NEXT I: IF S<>5547 THEN PRINT "ADATHIBA": END
3 SYS 49152: CLR: NEW
4 DATA 169,15,162,8,160,15,32,186
5 DATA 255,169,8,32,189,255,32,192
6 DATA 255,162,15,32,198,255,32,207
7 DATA 255,32,210,255,32,183,255,201
8 DATA 0,240,243,32,231,255,96
    
```

READY.

egy lehetőleg kis DL szakaszt vesz, a szakasz másik végpontjaként megkapva az erővonal következő pontját. Ez az iterációs eljárás természetesen csak közelítés, de DL kellően kicsire választásával tetszőlegesen pontossá tehető.

$$r_i^2 = (X - X_i)^2 + (Y - Y_i)^2$$

$$E_i = k \frac{Q_i}{r_i^2}$$

$$E_{xi} = \frac{E_i (X - X_i)}{r_i}$$

$$E_{yi} = \frac{E_i (Y - Y_i)}{r_i}$$

eredők:

$$E_x = \sum E_{xi}$$

$$E_y = \sum E_{yi}$$

$$E = \sqrt{E_x^2 + E_y^2}$$

$$X' = X + \frac{E_x}{E} \text{ DL}$$

$$Y' = Y + \frac{E_y}{E} \text{ DL}$$

Az erővonalak a töltések körüli 6 egység sugarú kör pontjaiból indulnak. Egy erővonal ábrázolásának akkor van vége, ha valamely töltés 6 egység sugarú környezetén belül került, vagy az ábrázolt tartományon kívülre, illetve attól kellően távolra. Az erővonalak a töltések összegének megfelelően vagy csak a pozitív, vagy csak a negatív töltésekből indulnak. Ha a töltések összege zérus, akkor is a pozitív töltésekből indulnak.

A töltéseket egy olyan derékszögű koordináta-rendszerben helyezhetjük el, melyben az X tartomány rögzített (-108, 108), az Y tartományt pedig mi választhatjuk meg (a példánál 120, 150 közötti érték). Mi határozhatjuk meg az egységnyi töltésből induló erővonalak számát is: értékük 26, 24, 13, 9, 15, 24 volt.

A programban választható egy MÓD érték; 0 választásra az 1., 2., 4., 6. ábránál volt szükség; ezekhez tartozik olyan erővonal, melynek egy része a tartományon kívül esett. 0 értéknél a program nem hagyja abba az erővonal vizsgálatát, amikor az elhagyja a vizsgált tartományt, csak akkor, ha az erővonal „végtelenül messze” kerül a tartomány origójától, azaz amikor a távolság értéke túlszorul. A sebesség növelése érdekében ilyen esetekben DL értéke a távolság négyzetével arányosan növekszik.

A program PTA-4000 (PC-1500) számítógépre készült, de kisebb módosításokkal más gépeken is futtatható.

MÉSZÁROS CSABA

Térhatású ábrák



Programjaim írásakor látványos ábrák készítésére törekedtem; arra, hogy kétváltozós függvényeket három dimenzióban ábrázolhassak.

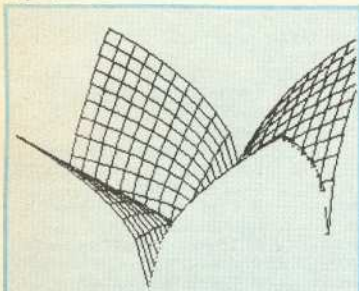
Az 1. program megértéséhez tudni kell, hogy szükség volt egy egyenletrendszerre (nevezük ezt most koordinátatranszformációnak), amely tetszőleges térbeli ponthoz hozzárendeli képernyőbeli megfelelőjét. Ha x, y, z a térbeli és i, j a képernyőpont koordinátái, az

$$\begin{aligned} i &= x + y/2 - \sqrt{z} \\ j &= z + y/2 \cdot \sqrt{2} \quad (2 \cdot \sqrt{2} = 2,8284) \end{aligned}$$

egyenletekkel axonometrikus képet állítunk elő, melyen az x tengely vízszintes, a z függőleges, az y 45 fokos, az y tengellyel párhuzamos szakaszok a képen felszerűkre rövidülnek.

A program annak a felületnek a képét rajzolja ki, melyet az egy sorban megadott kétváltozós függvény állít elő. A függvény értékét csak egy négyzetháló rácspontjaira határozza meg (tízessel halad végig az x és y értékeken), és az így kapott pontokat

1. program



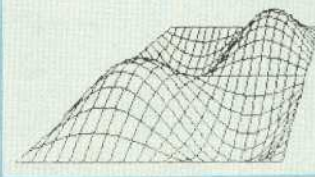
```

1)DEF FN f(x,y)=50R ABS (x-1
00)+y-100))
10 FOR y=200 TO 0 STEP -10
20 FOR x=0 TO 200 STEP 10
30 LET x1=x+10 LET y1=y
40 IF x<200 THEN LET t=1: GO S
UB 100
50 LET x1=x: LET y1=y-10
60 IF y>0 THEN LET t=0: GO SUB
100
70 NEXT x
80 NEXT y
90 STOP
100 LET z=FN f(x,y): LET z1=FN
f(x1,y1)
110 LET i=x+y*(x0-x1)/(y0-y1): LE
T j1=z1-y1*(z0-z1)/(y0-y1): LE
T j=z1-y*(z0-z1)/(y0-y1)
120 LET i1=x+10/2.8284: LET j1
=z1+y1/2.8284
130 IF i<=255 AND i1<=255 AND j1<=0 A
ND j1<=175 AND j<=175 AND j1<=175
THEN PLOT i,j: DRAW i1-j1,j1-j: R
ETURN
1000 LET di=ABS (i1-i): LET dj=A
BS (j1-j)
1010 LET k=di*(di+dj)+dj*(di+dj)
1020 LET e=(i1-i)/k: LET e1=(j1
-j)/k
1030 FOR n=0 TO k
1040 IF i<=0 AND i<=255 AND j<=0
AND j<=175 THEN PLOT i,j: IF t
THEN DRAW INVERSE 1,0,-j
1050 LET i=i+e1: LET j=j+e1
1060 NEXT n
1070 RETURN
2000 DATA 100,-250,200
    
```



```

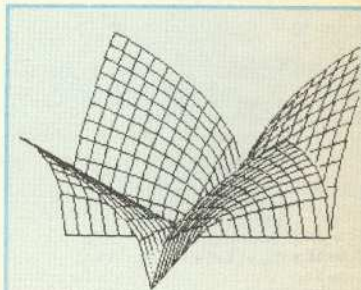
1 DEF FN f(x,y)=30*SIN (PI/10
0*x)+SIN (PI/100*y)
2 READ x0,y0,z0
10 FOR y=200 TO 0 STEP -10
20 FOR x=0 TO 200 STEP 10
30 LET x1=x+10: LET y1=y
40 IF x<200 THEN LET t=1: GO S
UB 100
50 LET x1=x: LET y1=y-10
60 IF y>0 THEN LET t=0: GO SUB
100
70 NEXT x
80 NEXT y
90 STOP
100 LET z=FN f(x,y): LET z1=FN
f(x1,y1)
110 LET i=x-y*(x0-x1)/(y0-y1): LE
T j1=z1-y1*(z0-z1)/(y0-y1): LE
T j=z1-y*(z0-z1)/(y0-y1)
120 LET i1=x+10/2.8284: LET j1
=z1+y1/2.8284
130 IF i<=255 AND i1<=255 AND j1<=0 A
ND j1<=175 AND j<=175 AND j1<=175
THEN PLOT i,j: DRAW i1-j1,j1-j: R
ETURN
1000 LET di=ABS (i1-i): LET dj=A
BS (j1-j)
1010 LET k=di*(di+dj)+dj*(di+dj)
1020 LET e=(i1-i)/k: LET e1=(j1
-j)/k
1030 FOR n=0 TO k
1040 IF i<=0 AND i<=255 AND j<=0
AND j<=175 THEN PLOT i,j: IF t
THEN DRAW INVERSE 1,0,-j
1050 LET i=i+e1: LET j=j+e1
1060 NEXT n
1070 RETURN
2000 DATA 100,-250,200
    
```



2. program

egyes szakaszokkal összekötve alakítja ki a felület hálószerű képét. A 10–50 ciklus határozza meg a soron lévő szakasz végpontjainak x, y koordinátáit, a 100-as sor a $z=f(x, y)$ koordinátákat, a 110–120-as sorok a megfelelő képpont-koordinátákat. 900–1070-ig történik a megjelenítés és a takart részek törlése.

A koordinátatranszformációt megváltoztatva elérhetünk perspektív képet is. A szempontok, melyek alapján az új egyenleteket némi koordinátagéometriával levezettem, a következők: adott a térben egy nézőpont és egy képsík; tetszőleges térbeli pont képe az őt és a nézőpontot összekötő egyenes metszéspontja a képsíknak. Az egy-



```

1030 LET jn=FN f(x+3.5355,y-10)+
(y-10)/2.8284: LET jn1=FN f(x+k*
e1+3.5355,y-10)+y-10/2.8284
1040 LET ejn=jn1-jn)/k
1050 FOR n=0 TO k
1060 IF i<=255 OR j<0 OR j>175 TH
EN GO TO 1110
1070 PLOT i,j: IF y+t=0 THEN GO
TO 1110
1080 IF jn<0 THEN DRAW INVERSE 1
,0,-j: GO TO 1110
1090 IF jn>175 THEN DRAW INVERSE
1,0,175-j: GO TO 1110
1100 DRAW INVERSE 1,0,jn-j
1110 LET i=i+e1: LET j=j+e1: LET
jn=jn+ejn
1120 NEXT n: RETURN
    
```

3. program

szűrőség kedvéért képsíknak az xz síkot választottam, a nézőpont koordinátáit DATA-ként adtam meg. Lehet kísérletezni a változtatásokkal (2. program).

A takarás megoldása mindkét programban éppoly egyszerű, mint amilyen tökéletes: hátulról haladunk előre, és az éppen felrajzolt vonal és az alsó képernyőszegély közötti területet „fehérítjük”, így a később rajzolt részek eltakarják a régebbieket. (Az yz síkkal párhuzamos szakaszoknál nincs szükség törlésre, így a törlést a t változó kapcsolgatja ki-be. Magát a törlést az 1040-es sor végzi el.) Így a felületekre alulról nem láthatunk rá. Változtathatunk ezen az 1030-tól kezdődő sorok 3. program szerinti módosításával. Akkor a program csak az éppen kirajzolt és az egygel későbbi vonal közötti sávot tisztítja meg. Csak az axonometrikus változatot tökéletesíthetjük így, a perspektívus program javításához valamivel több matematikai ügyeskedés kell. A lista könnyebb megértéséhez: $3,5355 = 10/2 \cdot \sqrt{2}, 10$ a lépcsőköz.

Nem titkolom el a program gyengéit sem: 1. lassú; 2. nem szereti a nem folytonos és az értelmezetlen függvényeket, 3. se a túl meredek vonalakat; 4. jól mutat, de mire jó?

SZABÓ PÉTER

BASIC és gépi kód

Legutóbb a feltételes ugróutasításokkal ismerkedtünk meg. Most a használatukra látunk egy egyszerű példát. Megismerünk néhány újabb címzési módot, melyek alkalmazására a következő részben találunk példákat.

A mintaprogram

Mintaprogramunk BASIC betöltőprogramja az 1986/5. számban jelent meg, és akkor az időmérés bemutatására szolgált. A rutin egyszerű feladatot old meg: letörli a képernyőt és 999 darab csillagkaraktert ír fel rá.

A lista disassemblerrel készült. Most ennek alapján lépésről lépésre megvizsgáljuk a rutin működését. A képernyőre a BSOUT KERNAL rutinnal írunk. A BSOUT sok részfeladatot hajt végre, ez okozza, hogy rutinunk viszonylag lassan működik. Előnye viszont e módszernek, hogy a mintaprogram mindhárom géptípuson változtatás nélkül futtatható — eltekintve a VC20 23 x 22 karakteres képernyőméretétől.

A \$033C...\$033E címen lévő utasítások a BASIC-bel PRINT CHR\$(147); utasítás feladatát hajtják végre. A pontososvessző azt jelöli, hogy a BSOUT végrehajtása során a kurzor egy pozíciót lép jobbra, de nem kerül automatikusan a következő sorba, mint egy PRINT utasítás esetén.

A következő két utasítás „felhúzza a ciklust”. A rutin az X és Y regisztereket ciklusszámlálóként használja. Az említett utasítások ezeknek a kezdőértékét állítják be. Hogy miért éppen az itt látható értékekre, arra még alább visszatérek. Megjegyzem, hogy a számlálásra az indexregiszterek helyett két memóriabajt is megfelelne, de ez semmilyen előnnyel nem járna, csak a rutin mérete lenne nagyobb.

A következő utasítás a csillagkarakter ASCII kódját tölti az A regiszterbe.

Két egymásba ágyazott ciklust láthatunk ezután. A belső a \$0347...\$034B, a külsőt a \$0347...\$034E címen lévő utasítások alkotják. A közös ciklusmag a \$0347 című utasítás: ez a BSOUT hívásával egy csillagot ír a következő képernyő-pozícióba.

033c	a993	lda	#93
033e	20d2ff	jsr	#ffd2
0341	a0e7	ldy	#e7
0343	a204	ldx	#04
0345	a32a	lda	#32a
0347	20d2ff	jsr	#ffd2
034a	08	dey	
034b	d0fa	bne	#0347
034d	ca	dex	
034e	d0f7	bne	#0347
0350	60	rts	

Nézünk a belső ciklus végrehajtását. A DEY utasítás egyteljesítő csökkenti az Y regiszter tartalmát, egyúttal beállítja az N és Z feltételbiteket. Minket most az utóbbi érdekel. A Z bit értéke addig 0, míg Y tartalma nullától különböző érték. Ekkor a BNE utasítás a ciklusmagra adja a vezérlést. Ha Y tartalma nullára csökken, a Z bit 1-re állítódik, és a futás a BNE-t követő utasításra folytatódik.

Ugyanez játszódik le a külső ciklusban is, de az Y regiszter helyett az X szerepel benne. Ez összesen négyszer fut le, az X regiszter négy különböző értékénél. Számoljuk meg, hogy a belső ciklus — ezzel együtt a ciklusmag — hányszor hajtódik végre. Amikor X-ben 4 van, a belső ciklus 231-szer fut le. X többi előforduló értékénél a belső ciklus minden alkalommal 256-szor hajtódik végre. A ciklusmag végrehajtásainak száma: $231 + 3 \cdot 256 = 999$. Tehát a rutin ennyi csillagot fog a képernyőre írni.

Megfigyelhető, hogy a DEY végrehajtásakor az Y regiszterben \$00 van. Ez a következő DEY végrehajtások SFF-re változik, aminek következményeképpen a DEY-t követő BNE a \$0347 címre ugat. Ez — mint néhány, különböző helyeken publikált programból kiderült — nem mindenki számára nyilvánvaló.

Azt is érdemes megfigyelni, hogy a disassembler által készített listán a gépi kódban a relatív ugrási távolság látható, míg a BNE emlékeztető kód (idegen szóval mnemonic) után már a tényleges ugrási cím.

Az egyszerű indexelt címzési módok

Az egyszerű indexelt címzési módok az utasításban szereplő memóriabajt címét befolyásolják: az operandusban található értékhez hozzáadódik a művelethez tartozó indexregiszter tartalma úgy, hogy ezt a tartalmat előjel nélküli egész számnak tekintik. A módosítás értéke a \$00—SFF (0—255) tartományba eshet. Az összegzésből létrejövő cím adja a műveletben ténylegesen részt vevő memóriabajt címét.

Négy egyszerű indexelt címzési mód van, ezeket két szempont szerint csoportosíthatjuk: 1. attól függően, hogy a címzést módosító regiszter az X vagy az Y; 2. attól függően, hogy nullalapos vagy abszolút címet módosít az indexelés — más szóval: egy vagy kétbajtos a művelet operandusa.

Az indexelt címzési mód alkalmazására a következő alkalommal láthatunk példákat. A további két indexelt címzési mód is a következő részben ismerhetjük meg.

(„Az ismeretlen C16” című sorozat első részében jeleztem, hogy a közölt információk helytelenül kapcsolódnak ezekhez az íráskozhoz. Jelen sorozatban idáig elmulasztottam ezt megemlíteni — most pótolom.)

OVERLAY— CHAIN— APPEND

Az 1986. 10. szám 25. oldalán, amikor a program titkosításáról volt szó, találkozunk azzal a feladattal is, hogy miként lehet a titkosítást végző programot a titkosítandó programmal összekapcsolni. A feladatot általánosítva így fogalmazhatjuk meg: hogyan lehet két önálló programot egy egységgé összekapcsolni? A titkosító program után begépelhető a másik, de ez hosszadalmas bibelődéssel járhat. Egyszerűbb megoldásnak tűnt, hogy be kell tölteni a titkosítandó programot, majd a BASIC program-szerkesztő segítségével a viszonylag rövid titkosító programot eleje írni. Sajnos azonban ez az út nem járható, egyrészt mert a titkosító program egy utasításra két képernyősnál hosszabb, másrészt mert nem szabványos BASIC kódokat tartalmaz.

Hasonló feladat előtt állunk akkor is, ha két hosszú program összeolvasztására készülünk úgy, hogy elkerüljük az ismételt begépelést. Ebbe a témakörbe tartozik az is, ha olyan hosszú programot akarunk futtatni, ami egyszerre nem fér el a tárbán, ezért a feladatot részekre bontva vagyunk kénytelenek elvégezni.

Az alábbiakban megvizsgáljuk, hogy milyen kapcsolatba hozható a tárbán elhelyezett program egy külső tárolón elhelyezett programmal. A programok ilyen jellegű kapcsolatának megteremtését OVERLAY vagy átfedéstechnikának nevezzük.

A kapcsolatot minden esetben a külső tárolón elhelyezett program betöltésével hozzuk létre. Ennek kulcsszava a LOAD, ezért egyrészt a LOAD különböző működési lehetőségeit, másrészt a BASIC program térbeli elhelyezkedését és az elhelyezkedést jelző mutatókat elemezzük. (Az utóbbiról részletesebben ismertetés volt az októberi BASIC RAM felosztás, illetve a novemberi Programtitkosítás c. cikkemben.) Most csak a mutatók elnevezését és elhelyezkedését ismételtem meg.

43—44 SOB (Start Of Basic) BASIC-térület eleje alaph: 2049

- 45–46 SOV (Start Of Variables) Változók kezdete alaph: 2051
 47–48 SOA (Start Of Arrays) Tömbök kezdőcíme alaph: 2051
 49–50 EOA (End Of Arrays) Tömbök vége alaph: 2051
 51–52 BOS (Bottom Of Strings) Sztringterület alja alaph: 40960
 55–56 TOM (Top Of Memory) BASIC-terület vége alaph: 40960
 Nézzük tehát a LOAD-ot!

BASIC-ben négyféle LOAD-funkciót valószínűleg meg:

1. Parancsmódban másodlagos cím nélkül. Pl. :LOAD"TEST",8
2. Parancsmódban másodlagos címmel. Pl. :LOAD"TEST",8,1
3. Utasításmódban másodlagos cím nélkül.
4. Utasításmódban másodlagos címmel.

Ez a felosztás talán fölöslegesnek tűnik, pedig komoly gondokat okozhat, ha nem ismerjük az egyes funkciók közötti különbségeket.

CHAIN1-A

```
1000 REM CHAIN1-A
1010
1100 A = 1:AF = "CHAIN1"
1110 PRINT "[CD]";A,AF
1120 LOAD "CHAIN2-A",8
```

CHAIN2-A

```
2000 REM CHAIN2-A
2010
2020 POKE 45,PEEK(174):POKE 46,PEEK(175):CLR
2100 B = 2:BF = "CHAIN2"
2110 PRINT "[CD]";B,BF
2120 END
```

CHAIN1-B

```
1000 REM CHAIN1-B
1010
1100 A = 1:AF = "CHAIN1" + ""
1110 PRINT "[CD]";A,AF
1120 END
```

CHAIN2-B

```
2000 REM CHAIN2-B
2010
2100 B = 2:BF = "CHAIN2"
2110 PRINT "[CD]";B,BF
2120 END
```

CHAIN1-C

```
1000 REM CHAIN1-C
1010
1020 POKE 45,0:POKE 46,64:CLR
1100 A = 1:AF = "CHAIN1" + ""
1110 PRINT "[CD]";A,AF
1120 LOAD "CHAIN2-C",8
```

CHAIN2-C

```
2000 REM CHAIN2-C
2010
2100 B = 2:BF = "CHAIN2"
2110 PRINT "[CD]";B,BF
2120 END
```

Parancsmód

Azt jelenti, hogy a LOAD funkciót közvetlenül a billentyűzetről a kulcsszó beírásával, majd a RETURN lenyomásával aktivizáljuk. Ekkor a következők történnek:

- betöltődik a program a tárba,
- a SOV mutató felveszi a tárban elhelyezett program utáni első szabad bájttal címet,
- egy CLR utasítás hajtódik végre, ami a többi mutatót igazítja a SOV-hoz,
- megjelenik a READY felirat, és a vezérlés visszatér a beviteli várakozó ciklusba.

Utasításmód

Ugyanez akkor kerül utasításmódban végrehajtásra, ha a LOAD kulcsszó a programban van, és a program soron következő végrehajtható kulcsszava éppen a LOAD lesz. Ilyenkor betöltődik a program a tárba, majd a vezérlést a SOB által jelzett program kezdőcímen lévő első BASIC utasítás kapja meg.

Fontos megfigyelés: ilyenkor a mutatók értéke nem változik, és a program nem a LOAD utasítást követő utasításnál, hanem a program első utasításánál folytatódik.

Másodlagos cím

A program betöltésének kezdőcímét fogja meghatározni az alábbiak szerint.

– A másodlagos cím hiánya (Ø) azt jelenti, hogy a program elhelyezését a SOB által jelzett címen kell elkezdeni. Ha a SOB-ot nem változtattuk meg, akkor az új program az előző program helyére kerül.

– Ha a másodlagos cím 1, akkor az elhelyezési címet nem a SOB-ból, hanem a háttértárolón a program előtt tárolt mutatóból vesszük. Ez a mutató a SAVE-kor íródott fel.

A fentiek ismerete magyarázatot adhat néhány rejtélyre. Ilyen szituációkra gondolunk: Parancsmódban betöltünk egy BASIC programot, és azt hibátlanul futtatjuk le. Mivel úgy gondoljuk, hogy szükségünk van egy gépi kódú segédprogramra is, azt szintén parancsmódban betöltjük például a szalagpufferbe (828–1019 tárcimék). Most újra elindítjuk a BASIC programot, de ez rövid működés után furcsa dolgokat produkál. Ha ekkor kétségbeesetten megpróbáljuk legalább SAVE-vel a zabolátlan programunk kimentését, ismét nem várt dolgot tapasztalunk: a néhány blokkból álló programunk kimentése valahogy túl sokáig tart.

Mi a magyarázata mindennek? Lássuk sorban:

– A parancsmódban betöltött gépi kódú program után a SOV áttöltődött.

APPEND1-A

```
1000 REM APPEND1-A
1010
1100 A = 1:AF = "APPEND1"
1110 PRINT "[CD]";A,AF
1120 GOSUB 2100:END
```

APPEND2-A

```
2000 REM APPEND2-A
2010
2100 B = 2:BF = "APPEND2"
2110 PRINT "[CD]";B,BF
2120 RETURN
```

APPEND1-B

```
1000 REM APPEND1-B
1010
1100 A = 1:AF = "APPEND1"
1110 PRINT "[CD]";A,AF
1120 X = PEEK(45) + PEEK(46) * 256: X = X - 2
1130 POKE 44,INT(X / 256):POKE 43,X - PEEK(44) * 256
1140 LOAD "APPEND2-B",8
1150 GOSUB 2100
1160
1170 END
```

APPEND2-B

```
2000 REM APPEND2-B
2010
2020 POKE 45,PEEK(174):POKE 46,PEEK(175):CLR
2030 POKE 43,1:POKE 44,8
2040 GOTO 1150
2050
2100
2110 B = 2:BF = "APPEND2"
2110 PRINT "[CD]";B,BF
2120 RETURN
```

APPEND1-C

```
1000 REM APPEND1-C
1010
1020 POKE 45,0:POKE 46,64:CLR
1050
1100 A = 1:AF = "APPEND1" + ""
1110 PRINT "[CD]";A,AF
1112
1114 X = PEEK(174) + PEEK(175) * 256
1116 X = X - 2
1116 POKE 44,INT(X / 256):POKE 43,X - PEEK(44) * 256
1130
1140 LOAD "APPEND2-C",8
1150 GOSUB 2100
1160
1170 END
```

APPEND2-C

```
2000 REM APPEND2-C
2010
2030 POKE 43,1:POKE 44,8
2040 GOTO 1150
2050
2100
2110 B = 2:BF = "APPEND2"
2110 PRINT "[CD]";B,BF
2120 RETURN
```


— A BASIC program a változóit nem a kívánt helyre kezdi elhelyezni, és így az nem kívánt tárterületeket ír át.

— Amikor a SAVE-vel próbálkozunk, ez a program kimentését a SOB-nál kezdve a SOV-ig hajtja végre, de a SOV értéke most kisebb a SOB-nál, így először a SOB fölötti teljes tárterületet menti ki, majd a számlálást nulláról indítva folytatja a mentést a SOV-ig.

A megoldás: ilyen esetben először a gépi kódú programokat és utána a BASIC programokat töltjük be.

Hasonló buktatót rejt az is, ha egy BASIC programutasítás utasításmódban meghív (LOAD) egy másik programot, de a második program hosszabb, mint az első. Ekkor — mivel a SOV értéke most nem változott — a változók ugyanarra a tárterületre kerülnek, ahol a program vége is elhelyezkedik, így a változók elrontják a program végét.

A megoldás a C64 BASIC-ben nem ismert CHAIN utasítás. Ezt a következő módon szimulálhatjuk.

CHAIN

A CHAIN (láncolás) a programok közötti olyan kapcsolat, amikor a tárban lévő program helyére, annak folytatásaként egy külső tárolóról új programot töltünk be, majd azt elindítjuk.

— Direkt módban a LOAD és a CHAIN közötti különbség csak annyi, hogy LOAD után a programot külön paranccsal kell indítani.

— Utasításmód: CHAIN1—A CHAIN2—A programpár

Utasításmódban a betöltött program automatikusan indul. Az egyetlen tennivaló a betöltött program első sorában olyan utasítás elhelyezése, ami a SOV változó értékét beállítja. Erre példa a CHAIN1—A és a CHAIN2—A programpár. Betöltve és elindítva a CHAIN2—A-t, az behívja és elindítja a CHAIN1—A-t. Bár a CHAIN2—A hosszabb, mint az első program volt, a változók mégsem rontják el a program végét, mert a SOV a program vége utáni bájtra mutat. A futás eredményéből azt is látjuk, hogy az első program változóit a második program nem ismeri.

A CHAIN speciális esete, ha az első program helyére úgy akarjuk betölteni a második programot, hogy ez az első programmal előállított változókat is használja. Ilyenkor külön területet jelölünk ki a programnak és külön a változóknak.

**Direkt mód: CHAIN1—B
CHAIN2—B programpár**

— Betöltjük a CHAIN1—B programot.

— Kijelöljük a program és a változó terület határát. Például: POKE45,0:POKE46,64 (most 16384 a SOV értéke).

— Elindítjuk az első programot. (A változók a SOV fölött lesznek elhelyezve.)

— A program lefutása után a többi változó mutatóit elmentjük egy szabad tárterületre, például 49152-től kezdődő címekre.

— FORX=0TO7:POKE49152+X,PEEK(45+X):NEXT

— Most betöltjük a CHAIN2—B-t.

— Ezután fordított sorrendben visszaállítjuk a mutatókat.

— FORX=7TO0STEP—1:POKE45+X,PEEK(49152+X):NEXT

A második programot nem RUN, hanem GOTO paranccsal indítjuk el:

— GOTO 2000

A program ekkor kiírja az első és a második program változóit is. A programban definált sztringváltozókat csak úgy tudtuk átvinni az első programból, hogy azokat a programterületről áthelyeztük a sztringterületre egy +"" művelettel az 1100-as sorban.

Utasításmód: CHAIN1—C CHAIN2—C programpár

Utasításmódban ugyanezen feladat megoldása a CHAIN1—C és CHAIN2—C programpárral sokkal egyszerűbb. Itt nincs más teendő, mint a CHAIN1—C program betöltése és elindítása. A többi a gép dolga. Érdemes megfigyelni a parancs- és az utasításmód közötti különbségeket ezeken a programpárokon keresztül is.

APPEND

Az APPEND (hozzáfűzés) a programok közötti olyan kapcsolat, amikor a tárban lévő program végéhez egy külső tárolóról egy programrészt fűzünk hozzá. Az összefűzés után a két programrészt a tárban egyéges programot alkot. Az összefűzés feltétele, hogy a két programrészen azonos sorszámok ne forduljanak elő, és a sorszámok az összefűzés után növekvő sorrendben következnek egymást.

Az összefűzés megvalósításához ismerni kell az első program helyfoglalását a tárban. Azt már láttuk, hogy minden BASIC sor első két bájta azt a tárcímét adja meg, ahol a soron következő BASIC utasítássor megtalálható a tárban. Ha ez a két bájtt nulla, akkor ez azt jelenti, hogy nincs több BASIC utasítás, tehát itt van a program vége. Mivel alapértelmezésben egy SOV a két nulla bájtot tartalmazó tárcímek utáni cím-

re mutat, a második programot a SOV—2 címtől kezdve kell betölteni ahhoz, hogy a második program a tárban úgy helyezkedjen el, mintha az előző program folytatása lenne. Az összefűzéshez tehát a következő lépések szükségesek:

— A SOB SOV—2-re állítása.

— A második program betöltése.

— A SOV-ot az összefűzött program végére kell állítani.

— A SOB visszaállítás a eredeti értékére.

A BASIC sorok elején álló mutatókat a LOAD a SOB értékének megfelelően automatikusan átírja, lehetővé téve, hogy egy adott helyről kimentett BASIC programot más tárcímre is be lehessen tölteni.

Nézzük tehát a direkt és az utasításmód közötti különbségeket a mintaprogramok segítségével.

Direkt módban: APPEND1—A APPEND2—A programpár

— Betöltjük az APPEND1—A programot:

LOAD"APPEND1—A",8

— Kiszámítjuk a második program betöltési címét:

X=PEEK(45)+PEEK(46)*256:Y=X—2

— Átírjuk a SOB-ot

POKE44,INT(X/256):POKE43,X—PEEK(44)*256:NEW

— Betöltjük a második programot:

LOAD"APPEND2—A",8

— Visszaállítjuk a SOB-ot:

POKE43,1:POKE44,8

Utasításmódban: APPEND1—B APPEND2—B programpár

Az eljárás hasonló a direkt módhoz, de a szükséges lépéseket most a programba építettük be. Vegyük észre, hogy a második program betöltése után a program futása nem a LOAD-ot követő utasításon folytatódik, hanem a SOB utáni első utasításon, ami nem más, mint a második program első sora! A főprogramba csak úgy lehetséges a visszatérés, ha a második program visszaállítja a SOB-ot, és olyan ugróutasítást tartalmaz, ami az első programra mutat.

APPEND1—C APPEND2—C programpár

Itt a különbség csak annyi, hogy a hozzáfűzött második program használhatja az első program változóit is. Az eljárás már teljesen ismerős kell hogy legyen a CHAIN és az APPEND előző változata alapján.

Tömbök kezelése FORTH-ban

A FORTH nyelv szabványosításával foglalkozó Forth Interest Group (FIG) elnevezésű szervezet 1979-ben fogadta el azt a szabványt, amely 243 FORTH szó definíciójából állt. Ez az alapszókészlet nem kínál tömb kezelésére szolgáló definíciót, de tartalmazza a <BUILDS...DOES> szerkezetet, amelynek segítségével új adatszerkezetek és utasítások értelmezhetők.

Tömb típusú adatszerkezet létrehozásakor a <BUILDS és DOES> közötti utasításokkal foglaljuk le a tömb számára a szükséges területet a memóriában. A terület a tömb szótári elemének PFA mezője. Ezek az utasítások fordításkor hajtódnak végre.

Egy szótári elem PFA mezőjében az alábbiak egyike található:

- végrehajtási cím(ek) a végrehajtás sorrendjében, ha a szótári elem egy másodlagos FORTH szó,
- gépi kódú program primitív szó esetén,
- konstans,
- változó aktuális értéke,
- tömbelemek.

Mivel a tömböt leíró szótári elemet újbabak is követhetik, ezért egy tömb méretét a definíciús után már nem lehet megváltoztatni. Célzerű az egyes tömbelemeknek \emptyset kezdőértéket adni. A DOES > utáni utasítások segítségével számíthatjuk ki egy tömbelem címét. Ezek az utasítások végrehajtáskor működnek.

Egy $m \times n$ méretű kétdimenziós, 2 bájtos szavakból álló tömb $a(i,j)$ elemének távolsága a tömb kezdőcímetől, mint a bázistóló számítva.

$$((i-1) * n + j) * 2 - 1 \text{ bájt.}$$

Látható a képletből is, hogy a tömb elemein kívül az egyes dimenziók maximális méretét is tárolnunk kell. Ezeket célzerű a PFA mező elejére helyezni, a tömb elemeit megelőzően.

n	m	a(1,1)	...	a(m,n)
---	---	--------	-----	--------

↑ PFA mező kezdőcíme

A definíció:
:TOMB
<BUILDS PDUP ,, * 0
DO , LOOP
DOES> DUP @ 4 ROLL 1 - * ROT + 2 * 3 + ;

A PDUP definíciója:
:PDUP (n1 n2 - n1 n2 n1 n2)
OVER OVER;

A ROLL a verem (stack) n-edik elemét a paraméterverem tetejére helyezi; nem szerepel a standard FORTH-ban, de több implementációban megtalálható. Definíciója:

```
:ROLL
  SP @ SWAP 1 - 2 * DUP > R 2 +
+ DUP @ R > SWAP >R >R DUP 1 +
SWAP
  1 - R > 0
  DO DUP 1 - C @ 3 PICK 1
  - C!
  LOOP
DROP DROP DROP R > ;
```

A PICK a verem n-edik elemét a paraméterverem tetejére másolja; akárcsak a ROLL, ez sem standard FORTH szó.

```
:PICK
  SP @ 2 + SWAP 1 - 2 * + @ ;
Hozzuk létre egy 3*4-es tömböt TERULET néven:
  3 4 2TOMB TERULET
Legyen a TERULET(2,3)=10
  10 2 3 TERULET !
```

A tömb egy elemének kiolvasása a beírás-hoz hasonló:
2 3 TERULET @.
10 ok

A fentiek mintájára bevezethetünk 3 dimenziós tömböket is. Egy $m \times n \times o$ szó méretű tömb $a(i,j,k)$ elemének távolsága a tömb kezdőcímetől

$$(((i-1) * n * o) + ((k-1) * n + j)) * 2 - 1 \text{ bájt.}$$

A 3 dimenziós adatszerkezet definíciója:
:3TOMB
<BUILDS DUP , SWAP DUP , ROT
DUP , **
0 DO
0,
LOOP
DOES> DUP @ SWAP 2 + DUP
>R @ * 4 ROLL 1 - *
SWAP 1 - R >
DUP >R @ * ROT + + 2 *
2 + R > + ;

A két- és háromdimenziós tömbök értelmezése és kezelése elvben teljesen azonos:

```
3 4 2 3TOMB KOCKA
Legyen a KOCKA (1,2,2)=14
  14 1 2 2 KOCKA !
```

Egy elem kiolvasása és kiírása a képernyőre:

```
1 2 2 KOCKA @.  
14 ok
```

Ez a tömbkezelési eljárás beépíthető bármely gépen futó FORTH rendszerbe, mivel nincs gépfüggő része.

BALOGH LÁSZLÓ

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

A folyóirat neve	Kódja
64'er Magazin	64er
Chip Magazin	chip
Commodore Horizons	coho
Commodore Microcomputers	comi
Compute!	cute
Computer Persönlich	pers
Happy Computer	happ
hc - Mein Home-Computer	hc
mc - Zeitschrift	mc
Run /USA/	run
Sinclair User	sinc
Your Sinclair	ysin

A tartalomleíró szövegeket permutáltuk, a szövegváriánsokat pedig alfabetikusan rendeztük.

A folyóiratok a SZÁMALK szakfolyóiratában is fellelhetők. A másolás feltételeiről bővebb felvilágosítást a 853-111/251 telefonszámon kaphatnak.

A tartalomleírás egy szövegből áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkezeséséhez a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szövegeiben elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szövegben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karakter sorozatát nyílvzeti be, melyet a / jelleg a folyóiratok azonosítója, a két / jel között az évszám, folyóiratszám és kötőjellel a kezdő oldalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

Európa legjobb Commodore újságjáról, a 64'er Magazinról másféle információszolgáltatás is kérhető. Szolnoki Béla vállalja, hogy havi 20 Ft-ért (+ a postaköltség) elküldi a folyóirat tartalomjegyzékeinek megjegyzésekkel is ellátott fordítását - a lap 1986. 07. havi számától kezdődően bármelyikről, megbízás esetén folyamatosan. Ennek alapján a megrendelő kiválaszthatja, hogy számára melyik cikkek fontosak, és ezek fordítását is megkaphatja oldalanként 5,-Ft-ért (+ a postaköltség), ha ír a következő címre:

"Számítástechnikai információszolgáltatás". Szolnoki Béla, Budapest, Pf.: 400. 1446

UNIFORM

PROGRAMLISTA

64er beviteli utmutato sor-osszeg el
lenorzes (mse) ->64er/86.02-57/2

PROGRAMLISTA

adatavitel (telefonos) c nyelv cp/m-
80 ->pers/86.03-57/4

PROGRAMLISTA

adatbazis lekerdezes atari 400/800; x
l; xe muveletgyorsitas fuzerkezeles
->cute/86.02-58/4

PROGRAMLISTA

animacio commodore 64 sprite-mozgata
s az <impossible mission> nal is alk
almazott technika alapelve
->happ/86.02-69/3

PROGRAMLISTA

apple ii alkalmazasi utmutato <speed
calc> ->cute/86.02-88/14

PROGRAMLISTA

apple ii atari 400/800; xl; xe commodo
re 64 jatekprogram <high rise>
->cute/86.02-49/8

PROGRAMLISTA

apple ii monitorkep tarolas
->cute/86.02-79/2

PROGRAMLISTA

apple ii pascal programozas betutipu
s valtas ->pers/86.03-56/2

PROGRAMLISTA

apple ii alkalmazasi fogasok
->cute/86.02-82/3

PROGRAMLISTA

apple ii grafikus programozas <appl
esoft> modositas felbontas javitasa
->mc/86.02-42/2

PROGRAMLISTA

apple atari commodore compute! bevit
eli utmutato ->cute/86.02-114/10

PROGRAMLISTA

apple jatekprogram <brain trainer>
->hc/86.02-63/7

PROGRAMLISTA

atari 400/800; xl; xe basic programoza
s hibajavitas ->cute/86.02-74/3

PROGRAMLISTA

atari 520st logo programozas rajk
zes egerrel ->cute/86.02-77/2

PROGRAMLISTA

atari monitorprogram lemezegyseg (ata
ri) <duex st> ->happ/86.02-86/4

PROGRAMLISTA

atari beviteli sor osszegezes
->hc/86.02-42/1

PROGRAMLISTA

atari cikksorozat grafikus programoz

as jatekprogram keszites
->happ/86.02-98/3

PROGRAMLISTA

atari jatekprogram <caveman>
->hc/86.02-51/3

PROGRAMLISTA

barKacsolas commodore 64 adatavitel
i lehetosegek buszvezerles
->64er/86.02-93/5

PROGRAMLISTA

basic programozas commodore 64 muvel
etgyorsitas ->cute/86.02-72/2

PROGRAMLISTA

checksum-Kiegeszitesek a <run/85.08>
litaihoz ->run/86.02-118/3

PROGRAMLISTA

cikksorozat commodore 64 <ascompiler
64> ->64er/86.02-68/3

PROGRAMLISTA

cikksorozat commodore 64 eprom beeg
to programozas (eprom)
->64er/86.02-64/2

PROGRAMLISTA

commodore 128 <ultra hi-res>
->run/86.02-34/14

PROGRAMLISTA

commodore 128 saját karakterek gener
alasa ->run/86.02-42/3

PROGRAMLISTA

commodore 64 <softscroll> szoveg-gor
getes ->mc/86.02-41/1

PROGRAMLISTA

commodore 64 fuggvenyabrazolas grafi
kus programozas muveszi grafika
->happ/86.02-107/2

PROGRAMLISTA

commodore 64 grafikon nyomtatasa <pos
ter-hardcopy> alkalmazasi utmutato
->happ/86.02-65/2

PROGRAMLISTA

commodore 64 grafikus programozas mu
veszi grafika sinus-fuggvenyek alkal
mazasa ->hc/86.02-82/2

PROGRAMLISTA

commodore 64 jatekprogram <butterfly
> ->coho/86.02-41/2

PROGRAMLISTA

commodore 64 jatekprogram <fast-food
chef> ->run/86.02-80/3

PROGRAMLISTA

commodore 64 jatekprogram <space sna
ke> alkalmazasi utmutato gepikodu pr
ogramjelso resz ->coho/86.02-38/2

PROGRAMLISTA

commodore 64 karaktermeret valtoztat

```

PROGRAMLISTA
nyomtato(mps 801)KarakterKep finomi
tas ->64er/86.02-75/2
PROGRAMLISTA
nyomtato(mps 801;802;803)KarakterKe
p modositas ->64er/86.02-100/6
PROGRAMLISTA
programozasi fogasokCommodore 16;11
6Rovidprogram-gyujtemeny
->hc/86.02-86/1
PROGRAMLISTA
programozasi fogasokCommodore 64Roi
vidprogram-gyujtemeny
->hc/86.02-84/2
PROGRAMLISTA
programozasi fogasokSinclairRovidp
rogram-gyujtemeny ->hc/86.02-88/1
PROGRAMLISTA
programozasi fogasokUzemeltetesi ta
nacsokAtariRovidprogram-gyujtemeny
->hc/86.02-90/2
PROGRAMLISTA
runBeviteli utmutatoCommodore 64;1
28 ->run/86.02-126/2
PROGRAMLISTA
sinclair spectrumJatekprogramKalie
n ->ysin/86.02-48/4
PROGRAMLISTA
sinclair spectrumJatekprogramKhot
shot ->ysin/86.02-44/2
PROGRAMLISTA
sinclair spectrumMuveletgyorsitasK
quickYKazetta ->happ/86.02-89/2
PROGRAMLISTA
sinclair zx-81Gepikodu programok be
toltese ->chip/86.02-130/3
PROGRAMLISTA
sinclairOktatasi (autovezetes)Utkoze
s szimulacio (fektaovolsag/sebesseg)
->hc/86.02-72/3
PROGRAMLISTA
strukturalt programozasBasic progra
mozasCikk sorozatBegepelo/maszKolo
rutin ->hc/86.02-92/2
PROGRAMLISTA
tavadatvitel (telefon)Terminal u
zemmodCommodore 64KtransbitRrs 23
2 vezeresSoros fileok Kezelese
->happ/86.02-52/5
PROGRAMLISTA
turbo-pascalFilekonvertalasdbase/t
urbo-pascal atteres
->pers/86.03-54/2
PROGRAMLISTA
turbo-pascalMenubeepitesGepfuggetl
en program ->pers/86.03-48/2
PROGRAMLISTA
zeneCommodore 64KeasymusicPoke-h
elyettesito utasitaskeszlet
->coho/86.02-41/1
PROGRAMLISTA
zeneCommodore 64Zongorabilentyuze
t szimulaciojaHangkimenet bovitese
->64er/86.02-80/1
as ->cute/86.02-70/3
PROGRAMLISTA
commodore 64Ilemezegyseg(1541)Muvet
elgyorsitasFuzerbetoltes
->cute/86.02-64/3
PROGRAMLISTA
commodore 64Ilemezegyseg(1541)Progr
amvisztaallitas ->64er/86.02-81/2
PROGRAMLISTA
commodore 64Ilemeztaar Kezeles
->run/86.02-74/2
PROGRAMLISTA
commodore 64MuvetelgyorsitasFuzeru
v altozo tarolasKgarbage 64
->64er/86.02-53/4
PROGRAMLISTA
commodore 64Nyomtato(mps 801)Grafik
us output ->64er/86.02-59/5
PROGRAMLISTA
commodore 64Programozasi fogas(re-c
lear) ->64er/86.02-77/2
PROGRAMLISTA
commodore 64Szovegfile formatalasK
vizauriteKalkalmazasi tanacsok
->64er/86.02-49/3
PROGRAMLISTA
commodore 64Utasitaskeszlet bovitese
Kwork-tool 64 ->hc/86.02-46/3
PROGRAMLISTA
dbase iiFilekezeles
->pers/86.04-48/4
PROGRAMLISTA
dbase iiIiiKeresoprogram(flexibili
s) ->pers/86.03-49/2
PROGRAMLISTA
grafikaPascal programozasTurbo-pas
calPerspektivikus abrazolas
->mc/86.02-36/5
PROGRAMLISTA
grafikaPascal programozasTurbo-pas
cal ->mc/86.02-78/4
PROGRAMLISTA
helyesbites(64er/85.11-85)
->64er/86.02-160/1
PROGRAMLISTA
helyesbites(64er/85.12-156)
->64er/86.02-160/1
PROGRAMLISTA
helyesbites(64er/85.12-82)
->64er/86.02-160/1
PROGRAMLISTA
helyesbites(mc/85.12-10)
->mc/86.02-14/1
PROGRAMLISTA
helyesbitesKcute/85.12-108;86.01-11
0 ->cute/86.02-112/1
PROGRAMLISTA
matematikai programozasMbasic progr
amozasFuggvenybeado alprogram
->pers/86.04-42/3
PROGRAMLISTA
mbasic preprocesszorCimkezett progr
amugrasok ->chip/86.02-123/2

```

Házi módosítások a Gens3, Mons3 programokon

1. Ékezetes üzemmód a Gensben

Itt és a továbbiakban feltételezzük, hogy a Gens a 30000, a Mons az 59000 címre van töltve, más töltési címmel minden értelemszerűen eltolódik; továbbá a Gensben a Buffer size-ra Enterrel válaszoltunk.

Az ékezetes Gens lényege, hogy a kommentekben ékezetes betűket használunk: ha L kurzorunk van, akkor a symbol shifttel együtt lenyomott asdfgqwe2346ui billentyűkre rendre áóüüüÁÉéúüüőÚű kódokat vesz be a program. Hiányzik még az ŐŰŰ, de ezek nélkül is elég magyarosan írhatunk.

A Gens wait-key rutinját kell módosítanunk. Először nézzük meg nagy vonalakban az eredeti működését (a rutin címe #7615=30229):

1. lista

20	#7615	PUSH IV		
30	LD	IV,#5C3A		
40	PUSH BC			
50	PUSH DE			
60	PUSH HL			
70	L761E	EXX		
80	PUSH AF			
85				
90			;; az üzemmódtól függően C vagy L kurzor kiírása	
100			;;inverz; villófény nyomtatásban; majd a normál nyomtatás	
110			;;visszaállítás)	
120				
130	L7648	RES	5,(IV+1) ;ha nem volt billentető - jelzés	
140	CALL	#02BF	;;billentetőzet leolvasása	
150	BIT	5,(IV+1)	;;volt billentetőzomórá?	
160	JR	NZ,L765F	;;ha igen, ugrás	
170	LD	DE,#0A00	;;villófényt kis szöveg jön	
180	L7658	DEC	DE	
190	LD	A,D		
200	OR	E		
210	JR	NZ,L7658		
220	JR	L7648	;;májd újra a leolvasás következik.	
225				
230	L765F	CALL	#7755 ;a kurzor törlése	
240	LD	A,(#5C0B)	;;a lenyomott billentető kódja	
250	CP	#06	;;Caps lock	
260	JR	Z,L767C	;;esetben ugrás	
265				
270	LD	B,#0B	;;11 Párból álló táblázat van	
280	LD	HL,L7689	;;ezen a címen. Ha valahol pár 1.	
290	L766E	CP	(HL) ;a táblával egyezik a bill.kód	
300	INC	HL	;;HL a 2. tábla mutat	
310	JR	Z,L7688	;;akkor ugrás a cseréhez	
320	INC	HL	;;kölabb pár vize9álata)	
330	DJNZ	L766E		
335				
340	L7675	POP	HL ;megyelekt vége a rutinnak.	
350	POP	HL		
360	POP	DE		
370	POP	BC		
380	POP	IV		
390	RET			
395				
400	L767C	LD	HL,#5C6A ;a MODE	
410	LD	A,#0B	;;0-as bitjét (;CAPS LOCK)	
420	XOR	(HL)	;;átváltja	
430	LD	(HL),A	;;újra az új értéket	
440	POP	AF		
450	JR	L761E	;;vissza a wait-key-be	
455				
460	L7688	LD	A,(HL) ;a kód felcserélése	
470	JR	L7675	;;után visszatérés	
480	L7689	DEFB	#08,#18,#0C	;;a 11 Párból álló táblázat
490	DEFB	#05,#0B,#0C		
500	DEFB	#07,#05,#05		
510	DEFB	"E",#05,"I"	;;E, OR, I	
520	DEFB	#E2,"",#03	;;STOP, ;NOT	
530	DEFB	"I",#0D,""	;;I, STEP, ;	
540	DEFB	#0C,"",#0B	;;TO, ; THEN	
550	DEFB	"Y"	;;)	

Ennek alapján a beavatkozás egyszerű: a kódcserehez egy nagyob táblázatot adunk meg a 29944 címen, ebben az ékezetes betűk számára szükséges cseréket is elhelyezzük. Ékezetes betűkre azonban csak L kurzor esetén váltunk át (a kommentekben ügyis kisbetűvel érdemes írni), így C kurzorral elérhetjük a ~/|@#&\$& karaktereket is.

A 29944, illetve 29990 címre az alábbi táblázatokat kell elhelyezni (legélcserűbb ezt előre megcsinálni egy kis BASIC programmal, és kimenteni):

cím	értékek vagy karakterek
29944	8 12
29946	12 8
29948	11 12
29950	7 3
29952	' AND ' 'E'
29954	'<' 'A'
29956	'>' 'G'
29958	'<<' 'E'
29960	'>>' 'E'
29962	'AT' 'I'
29964	'Z' 'Z'
29966	'Y' 'Z'
29968	'Y' 'Z'
29970	'OR' 'O'
29972	' STOP 'S'
29974	'NOT' 'O'
29976	' STEP 'S'
29978	' TO 'O'
29980	' THEN 'O'
29982	'# 'I'
29984	'&' 'O'
29986	'\$' 'O'
29988	'@' 'O'
29990	' ', '~', '/', '@', '#', '&', '\$', '&

1. táblázat

Látható, hogy felcseréltük a mi írógépeinknek megfelelően az y és z billentyűket; akinek ez nem tetszik, könnyen kiiktathatja a cserét úgy, hogy például a táblázatban az yz,YZ,zy,ZY párok helyett yy,YY,zz,ZZ párokat ír.

Ha már a wait-key rutin módosításánál tartunk, akkor érdemes a billentyűzetkattanást is megcsinálni, ami a Gensből szerintem nagyon hiányzik. A kattanást a rutinról való visszatérés előtt adjuk ki. Az 580-as sorban szereplő HL=160 az eredetinel 2 egésszel magasabb hangot eredményez. Bárki az izlése szerint más értékkel is kísérletezhet, a frekvencia ezzel fordított arányban változik.

A kiegészítő programot a 29900 címre tettük — lehet máshova is, a címek értelemszerű megváltoztatásával. A lista a következő:

30	#KEZETES GEN3	(kiegészítés és módosítás)	
40			
50	ORG	29900 ;=#74CC; a kiegészítő rutin	
60		=====	
70		;; a Gensből csak akkor kerül ide a vezérlés, ha a kódcsere	
80		;; táblázatban megállította az Input karaktert.	
90		;; a B regiszter mutatja, hogy hátralévő számolva hányadik	
100		;; Pár szerint kell cserélni.	
110		;;#régi kód, (HL)=új kód	
120			
130	LD	C,R ;a régi kód tárolása	
140	LD	A,B	
150	SUB	A	;;az utolsó 10 Pár kivételével
160	LD	A,(HL)	
170	JR	NZ,VISSZA	;;feltétlen csere
180			
190	LD	HL,23658	;;ha a kurzor L
200	BIT	3,(HL)	
210	JR	Z,VISSZA	;;az utolsó 10 Párnál is OK.

```

220 LD A,10 ;egységként ebből a 10 Párból
230 SUB B ;
240 CP B ;az utolsó 4 elemnél (A=B..9)
250 CP B ;
260 CP NC,VISSZ1 ;visszacsinálja a cserét ##00
270
280 LD HL,2990 ;ebből a táblázatból 1-10
290 RRD A,L ;tovább cserél A=B..5 esetén
300 LD C,(HL)
310 LD C,(HL)
320
330 VISSZ1 LD A,C ;a kód az A regiszterbe kerül
340 VISSZA JP #7676 ;vissza a Genshez
350
360
370 ORG #B667 ;#46695
380
390 ;a Gens wait-key rutin 99 részét módosítjuk.
*****
400 ;fordítás után áthelyezendő a #7667=30311 címre
410
420 L761E EOU #B61E
430
440 LD JR Z,L768E ;CAPS LOCK
450 LD B,#17 ;23 Párból álló táblázat van
460 LD HL,#74F8 ;ezen a címen.
470
480 L766E CP (HL) ;kicsésés...
490 INC HL
500 JP Z,29900 ;találatalnál ugrás a kie9. rutinja
510 INC HL
520 LD HL,L766E
530
540 PUSH IX ;IX és AF elmentendő, mielőtt a
550 PUSH AF
560 LD D,#00
570 LD E,(IX-1)
580 LD HL,#00A0
590 CALL #03B5 ;kattantást kiadjuk.
600 POP AF
610 POP IX
620
630 POP HL ;vége a wait-key rutimnak.
640 POP HL
650 POP DE
660 POP BC
670 POP IY
680 RET
690
700 L768E LD HL,#5C6A ;Caps lock rutinja a MODE
710 LD A,#0B ;3. bit (kisbetű/nagybetű)
720 XOR (HL) ;átváltás
730 LD (HL),A
740 POP AF ;vissza őj billentyű leolvasására!
750 JR L761E

```

2. lista

Végül beszéljünk magukról az ékezetes betűkről! Ezek nyomtatósi képtét bármilyen címre tehetjük, és ha a 29944-en levő táblázat BASIC programmal hozzuk létre, akkor még az is lényegtelen, hogy milyen betűt milyen kódhoz rendelünk, hiszen a táblázatba majd a megfelelő kód kerül be. Mivel azonban a keskenykaraktes ékezetes készletre ez már nem igaz (lásd a 2. pontot), ezért jobb, ha itt is ahhoz igazodunk, és az ékezetes betűket a következő sorrendben kódoljuk (144-tól 161-ig):

AEÍŌŌŪŪŪŪaeíóöüüü
 Akinek pedig nincs ékezetes betűkészlete, az alábbi táblázat alapján pótolhatja:

betű	cím	értékek
A	65368	16, 112, 168, 136, 248, 136, 136, 0
E	65376	16, 252, 160, 248, 128, 128, 252, 0
I	65384	16, 32, 112, 32, 32, 112, 0
O	65392	16, 120, 164, 132, 132, 132, 120, 0
U	65400	72, 120, 132, 132, 132, 132, 120, 0
Ö	65408	40, 120, 212, 132, 132, 132, 120, 0
Ő	65416	16, 168, 168, 136, 136, 136, 112, 0
Ü	65424	80, 132, 132, 132, 132, 132, 120, 0
Ū	65432	20, 40, 132, 132, 132, 132, 120, 0
ae	65440	16, 32, 112, 8, 120, 136, 120, 0
ae	65448	16, 32, 112, 136, 248, 128, 120, 0
ae	65456	16, 32, 0, 96, 32, 32, 112, 0
ae	65464	16, 32, 112, 136, 136, 136, 112, 0
ae	65472	80, 0, 112, 136, 136, 136, 112, 0
ae	65480	40, 80, 112, 136, 136, 136, 112, 0
ae	65488	16, 32, 0, 136, 136, 136, 112, 0
ae	65496	80, 0, 0, 136, 136, 136, 112, 0
ae	65504	40, 80, 0, 136, 136, 136, 112, 0
ae	65512	192, 32, 64, 160, 0, 0, 0, 0
ae	65520	192, 32, 192, 32, 192, 0, 0, 0

2. táblázat

2. 64 kar/sor listázási lehetőség a képernyőn

A gépi kódú programozásnál célszerű, ha a programlistát az utasítások mellé írt magyarázatokkal egészítjük ki. Enélkül néhány

hét-műva már saját programunkat sem értjük meg. Sajnos azonban a Gens3 ezt a technikát szinte lehetetlenül teszi, mert a 32 karakter/soros képernyő miatt a jobb oldali kommenteket folytatja a következő sorban a címek és utasítások között, és így teljesen zavaros, áttekinthetetlen lesz a lista. (Aki nem hiszi, próbálja ki!) A megoldás: írjunk 64 karaktert egy sorba! Ehhez célszerűen az L parancsot kell megfejtenünk. Ezért jó, ha tudjuk, hogy a #8ea3 címen van a fő parancsok címtáblázata, amely a következő felépítésű: 19 x 3 bájtból áll, ebből az első a parancs nagybetűjele, utána van a parancs végrehajtási címe (lásd a 3. táblázatot). Megjegyezzük, hogy ugyanilyen táblázat található a #8eeb címen, ami viszont az E (szerkesztő) parancs alparancsait tartalmazza (lásd a 4. táblázatot).

A	32727	= #7FD7	F	31655	= #7BA7
B	30057	= #7569	I	31547	= #7B3B
C	32541	= #7F1D	K	31471	= #79FF
D	31146	= #79FA	L	32620	= #7F6C
E	31257	= #7919	O	32653	= #7F8D
F	31779	= #7C23	S	31569	= #7B51
G	32246	= #7DF6	R	31253	= #7915
I	31104	= #7980	X	31542	= #7B3C
L	31662	= #79BE	Z	31552	= #7B40
K	31740	= #7BFC			
M	32355	= #7E63			
H	32095	= #7E21			
P	32005	= #7D95			
R	31182	= #79CE			
S	31243	= #790B			
T	31881	= #7C89			
V	35839	= #8BFF			
N	31752	= #7C8B			
X	32657	= #7F91			

4. táblázat

3. táblázat

Most már nem okoz nehézséget az L és K parancsok visszafejtése. Előljáróban megjegyezzük, hogy a Gens parancs általános alakja:

parancsjel n1, n2, s1, s2
 ahol n1,n2 kétbájtos szám, s1,s2 pedig max. 20 karakteres sztringek (amit nem adunk meg, azt 0-nak, illetve üres sztringnek veszi a program). Amikor az egyes parancsrutinokra ugunk, akkor már n1, n2, s1, s2 rendre a #8f49, #8f4b, #8f1b, #8f2f címre van le-
 rakva. A legutolsó beadott parancs (editline) mindig a #90ad címen található. Az L parancs általános jelentése: listázás n1-től n2-ig.

```

30 ;a Gens L,K és H Parancsok rutinjai
40 ; visszafejtés
50 ;ha a Gens a 30000h-#7530 címen van
60 ;általános forma bet0,n1-n2,string0,string2
70
80 ;a belépéskor n1,n2 már rendre a #8F49, #8F4B címen van
90
100 ;a listázás n1-től n2-ig L Parancs
110
120 L7BAE LD BC,(#8F49) ;n1 -> BC
130 LD R,B
140 OR C ;meg van azba n1?
150 JR NZ,L7BB7 ;akkor jó
160 INC C ;egységként legyen BC=1
170 L7BB7 CALL #7F35 ;a BC-edik sor mekserésde
180 ;vissza, ha nincs ilyen sor!
190 SET D,(IX+8)
200 L7BF8 LD A,(#8F5B) ;az egy adagban listázandó sorok
210 LD (IX+1),A ;számát a számlálóba teszi
220
230 ;egy sor kiírását végző részpre.
240
250 L7BC5 LD C,(HL) ;a HL a címtáblát, a sorszámot
260 INC HL ;a sorszámtól
270 LD B,(HL) ;a BC-be tesszük
280 EX DE,HL ;a címtáblát DE-ben tároljuk
290 LD HL,(#8F4B) ;n2 -> HL (utolsó listázandó sor)
300 LD R,H
310 OR L ;ha nincs megadva.
320 JR NZ,L7BD3
330 LD HL,#7FFF ;akkor a max sorszámtól veszi
340 L7BD3 SBC HL,BC ;ha ez a sor már nem listázandó
350 RET C ;akkor visszatérés.....
360
370 LD H,R
380 LD L,C ;sorszám a HL-be megy
390 PUSH DE ;a folvó cím tárolása
400 CALL #7F85 ;sorszám kiírása
410
420 POP HL ;a folvó cím
430 PUSH HL ;(sorszám 2. byte)
440 INC HL ;szövege 1. byte
450 LD D,#06 ;a szöveg a 6. oszloptól kezdődik
460 CALL #7806 ;a szöveg kiírása
470
480 POP HL ;címtáblát
490 DEC HL ;sorszám 1. byte
500 CALL #7F4F ;HL<köv.sorz.cím, Z=1,ha vége a

```

```

510      RET Z      ;listának, akkor visszatér.....
;itt helyesebb lenne RET NC alkalmazása, ez megakadályozná
; a Gens elszállását, akkor is, ha valami felütán a lista
;véget jelző mutató nem sorvége utáni keszkeze mutat.
520
530      DEC (IX+1) ;kiírt sorok számlálása
540      LD A,(IX+1)
550      DEC A      ;ha még nem írt ki elég sort,
560      JP P,L7BC5 ;akkor folytatja a listázást
570
580      CALL #76AD ;billentyűt vár, kódiát A-ba teszi
590      CP #03    ;EDIT-et nyomtuk meg?
600      RET Z     ;akkor visszatérés.....
610      JR L7BBF  ;újabb adat listázása következik.
620
630
640 ; A KIV Parancs működése:
650      LD A,(#BF49) ;in mod 256 (első byte)
660      OR A        ;még van adat?
670      JR NZ,L7CB4 ;ha igen, utasás
680      LD A,#0F   ;ha nincs megadva akkor legyen 15!
700 L7CB4 LD (#BFB5),A ;első sor lesz 1 adat listában.
710      RET
720
730
740 ; A M Parancs működése:
750
760      LD A,#10   ;Paraméter a következő rutinhoz
770      CALL #755D ;megadja a Printer csatornát, és
780      JP L7BAE  ;az L Parancsnak megfelelően oda
790              ;fog listázni.

```

3. lista

A List rutint úgy módosítjuk, hogy a sorszámkiíró rutin hívásánál az eredeti #7185 címet saját rutinunk címével helyettesítjük, ami NUMBER = #e100. Ez a rutin először is megnézi, hogy az L után nem 0 áll-e. Ha igen, akkor átér a normális 32 karakteres kiírásra, tehát például L0100 hatására hagyományos, L100 hatására keskenykarakteres lista jön a 100-as sortól.

Ha az L után nem 0 áll, akkor a sorszámot keskeny karakterekkel kiírja, majd visszatér a List rutinba. Ugyanígy járunk el a szöveg kiírást részről is, ahol az eredeti CALL #7806 helyett CALL #e196-ra módosítunk. Ez a LIST64 rutin címe, amely szintén kezeli az L0... esetet. A kiegészítő rutin listája:

```

30  SC #011
40  ;L00 karakteres Gens listához utalunk
50
60      OPB #E100
70
80 ; a sorszám kiírás rutinja
90
100 NUMBER LD A,(#90AE) ; a Parancson 2 karakteres
110      CP 48          ; L0... alak volt-e megadva?
120      JP Z,#7806    ; akkor normál listázás
130
140      PUSH HL
150      LD A,#0640   ; a sor száma kétféleképp 21. sor száma
160      LD C,(HL)    ; tároljuk el
170      LD A,#55     ; 64 karakter sorhossz lehet
180      LD (PMS),A
190      POP HL
200
210      LD DE,-10000 ; a sor szám kiírása a tízezeres
220      CALL DIGIT   ; helyértéktől kezdve
230      LD DE,-1000
240      CALL DIGIT
250      LD DE,-100
260      CALL DIGIT
270      LD DE,-10
280      CALL DIGIT
290      LD DE,-1
300      CALL DIGIT
310
320      LD A,32
330      CALL PR4B    ; a helyező a szám után
340      JDR A
350      RET (TAB),A ; a tabulátorpozíció nullázása
360
370
380 ;a sor száma meg kiírása rutinja
390 DIGIT XDR A      ; nulláról indul a számjávó
400      ADD HL,DE    ; kivonás
410      INC A        ; számjávó hátrésozunk ki
420      INC C,DIGIT+1 ;ha még nem fejezt el visszazuhunk
430      SBC HL,DE    ;egységként utolsó kivonást törli
440      ADD A,47     ; a keresett számjávó kódja
450
460
470 ; A 4 bites Print rutin (ascii és felh.graf. karakterek)
480
490 PR4B CP 3C      ;vezérlő kódoknál
500      RET C       ;visszatérés
510      CP 12B     ;lehet monni az ascii karaktert
520      JP C,ASCII
530      CP 144     ;standard graf.nál vissza
540      RET C
550      CP 165     ;kulcszavattal sew fogad el
560      RET NC
570
580      PUSH DE     ;felh.graf.karakterek esetén

```

```

590      LD DE,UDG4-1152 ;az a bázis cím
600      JP K0205
610
620 ASCII PUSH DE    ; a normál keskenykarakteres
630      LD DE,#CHAPS4-256 ;esetén Pedig ez.
640 K0205 PUSH BC
650      PUSH HL
660      LD H,0
670      LD L,A
680      AND HL,HL
690      AND HL,HL
700      AND HL,HL
710      ADD HL,DE
720
730      LD A,(POS)
740      DEC A
750      LD (POS),A ;kiírás Pozíció lép. ha van hely
760      LD Z,VEGE
770
780      INC H
790      EX DE,HL
800      LD HL,(CIM) ;Dezkarakter címe
810      PUSH HL    ;első lépélem címe
820
830      LD B,8     ;8 lépélem lesz
840      BIT 0,A   ;ha A páratlan, akkor egy 8 bites
850      JP NZ,HATUL ;lép hátsó felébe kell írni)
860
870      LD (HL),A ;eddi kiép törlése
880      LD A,(DE) ;beírás hátulra
890      AND B     ;CHL elejébe mevr.
900      INC DE    ;a kar.lépélem és a
910      INC H     ;lépélem frissítése
920      DJNZ ELOL ;mind a 8 elemet!
930      POP HL    ;a felső lépélem címét lehozzuk
940      JP C,INLEP ;és nem változtatjuk meg.
950
960 HATUL LD A,(DE) ;kar.lépélem -A
970      OR (HL)   ;a hátsó 4 bit beírása a lépélemben
980      LD (HL),A ;kar.lépélem és
990      INC H     ;lépélem frissítése
1000      DJNZ HATUL ;mind a 8 elemet!
1010      POP HL   ;a felső lépélem címe
1020      INC HL   ;most ad.
1030
1040 CIMEP LD (CIM),HL ;a felső lépélem cím beírása
1050      VECE HL
1060      POP BC
1070      POP DE
1080      RET
1090
1100
1110 ; a szöveg kiírás
1120
1130
1140 LIST64 LD A,(#90AE) ; a Parancson 2 karakter
1150      CP 48          ; '0' ?
1160      JP Z,#7806    ; akkor normál kiírás
1170
1180 LIST LD A,(HL)
1190      CP 59          ; aktuális karakter
1200      JP Z,NTABU    ; pontosvessző esetén esetleg
1210      CP 9          ; táblázás szükséges.
1220      JP Z,NTABU    ; táblázó-karakter?
1230      CP 13         ; Enter esetén visszatérés
1240      JP Z,#DFE     ; a scroll rutinon keresztül.
1250      CHAR CALL PR4B ;egységként a karakter kiírható
1260      STEP INC HL   ; táblázás cím növelése
1270      JR LIST       ; és a listázás folytatása
1280
1290 NTABU LD A,(TAB)
1300      INC A          ; a tabulátor ugratása
1310      LD (TAB),A
1320
1330      LD D,#E6
1340      LD E,A         ;DE megcímezi a tabulátor
1350                          ;pozíciójának értékét.
1360      ADD A,(HL)
1370      DEC A          ;tabu.Poz.szám+aktuális karakter
1380      CP 59         ; tábu.Poz.szám+aktuális karakter
1390      JP Z,CHAR     ; pontosv. és 1. pozíció
1400                          ; esetén visszatérés kiírásra
1410
1420 SPACES LD A,(POS) ; az aktuális Pozíció
1430      LD B,A        ; (tárolás B-ben)
1440      LD A,32       ; után széfekeket ír
1450      CALL PR4B    ; (legalább egyet!)
1460      LD A,(DE)    ; amíg a tabulátorpozíció
1470      ADD A,B       ;
1480      CP 67         ;
1490      JR NZ,SPACES ; el nem értük.
1500
1510      LD A,(HL)
1520      CP 59         ; aktuális kar.
1530      JP Z,CHAR     ; pontosvessző esetén
1540      JP STEP       ; kiírás táblázásra, egységként a kar.
1550                          ; táblázására visszatérés
1560
1570      ORG #E200
1580      DEFS 768
1590      UDG4 DEFS 168 ; keskeny karakterkészlet
1600      POS DEFS 1   ; táblázás Pozíció
1610      CIM DEFS 2   ; az ennék címe a képernyőn
1620      TAB DEFS 1  ; a tab. tábl. táblázása
1630
1640      ORG #E600
1650      DEFB 6,13,18,30 ; a tabulátor állászat
1660                          ; a tabulátor állásai

```

4. lista

Amint a listából látható, a 24 karakteres listázásba még egy funkciót bevettünk: a kommentárok tabulálását. Ha a pontosvessző nem a sor elején van, akkor a rutin tabulátornak tekintti. Tehát ahhoz, hogy a megjegyzések szép sorban legyenek egymás alatt, elegendő, ha az utasítás operandusa után egy pontosvesszőt teszünk, még helyköz sem kell. Ezzel időt és memóriát takarítottunk meg.

A végrehajtás egy kis haditervet igényel:

a) A keskeny karakterek nyomtatási képeinek 768 bájtos blokkját a tasword2-ből vehetjük, a 61184 címűtől. Néhány bájtot a save előtt érdemes megváltoztatni, az eredeti karaktereknél szerencsésebb alakokat kapunk így: például jól megkülönböztethető a B,8,0, illetve az m és w, a W és Y (lásd az 5. táblázatot).

b) A felhasznált grafikus karakterek keskeny változatát a 6. táblázat alapján elkészítjük és felvesszük.

c) Betöltjük a Genset 30000-re (ékezeteset 29900-ra), a Monsot 59000-re és a 4. listát Genssel beírjuk. Ezután 5-ös opcióval fordítunk, és okvetlenül ellenőrizzük legalább a NUMBER és a LIST64 címeket (#E100 és #E196).

d) Kilépünk a Gensből, belépünk a Monsba és módosítjuk az alábbi két hívást:

#7BD9 CALL #7F58 ; helyette #E100=NUMBER

#7BE1 CALL #7806 ; helyette #E196=LIST64

e) Betöltjük a keskeny karaktereket 57856-ra, a grafikusakat 58624-re.

f) Ezzel készen vagyunk, most már listázhatunk a Gensben. A kész programot pedig a Monsnal együtt felvesszük.

Végül a szükséges táblázatok következnek.

karakter	cím	értékek
B	57984	0,3,5,5,7,5,6,0.
8	58048	0,2,5,2,5,5,2,0.
N	58224	0,6,5,5,5,5,5,0.
W	58296	0,5,5,7,7,7,2,0.
Y	58472	0,0,7,7,7,7,5,0.

5. táblázat

karakter	cím	értékek
A	58624	1,2,3,5,7,5,5,0.
E	58632	2,7,4,6,6,4,4,7,0.
I	58640	1,2,7,2,2,2,7,0.
O	58648	1,3,5,5,5,5,2,0.
0	58656	5,0,7,5,5,5,7,0.
0	58664	7,0,7,5,5,5,7,0.
0	58672	2,7,7,5,5,5,7,0.
0	58680	5,0,5,5,5,5,7,0.
0	58688	7,0,5,5,5,5,7,0.
4	58696	1,2,6,1,7,5,7,0.
é	58704	1,2,3,5,6,4,3,0.
é	58712	1,3,0,6,2,2,7,0.
ó	58720	0,1,3,5,5,5,2,0.
ó	58728	0,5,0,2,5,5,2,0.
ó	58736	5,5,0,2,5,5,2,0.
ó	58744	0,1,2,0,5,5,7,0.
ó	58752	0,0,5,0,5,5,7,0.
ó	58760	0,5,5,0,5,5,7,0.

6. táblázat

3. A Monsban disassemblált szöveg átalakítása Gens formátumra

A Mons a T paranccsal visszafordított listát állít elő valamely memóriablokkról. A hiba csak az, hogy míg a tabulátorugrásokat a Gens 9-es kódokkal jelzi, a Mons ezt szóközzel teszi. Szerencsére máskor a szövegben nem is fordul elő szóköz, így nincs más dolgunk, mint a 32-es kódokat 9-esre átváltani. A sorszámokat természetesen közben át kell ugrani.

Az átalakító rutin használata a következő. Először a Gensben megtudjuk X paranccsal a végcímét. A Monsban a text kérdésre a végcím (hex!) adjuk be. Kilépvén a Monsból, PRINT AT 21,0;USR 58960 és kész!

```

90 ;3811
95
100 ;TEXT TO GENS
110 ;a visszafordított listában a szpészeket
120 ;chr# 9 -re cserélli
130
140      ORG 58960
150
160 GENS  EDU 30000

```

```

170 MONS  EDU 59000
175
180 ;a MONS a GENS lista mögé tette a szöveget!
185
190      LD  HL,(GENS+70) ;Gens lista vége
200      LD  DE,(MONS+4895) ;Mons lista vége
205
210 NUMBER INC  HL ;az aktuális cím léptetése
220 CHAR  INC  HL
230      LD  A,(HL) ;az aktuális karakter
240      CP  32 ;szpész?
250      JR  Z,TABULA ;akkor azt tabulátorként kezeli.
260      CP  13 ;ha még nincs vége a sornak;
270      JR  NZ,CHAR ;akkor folytatjuk.
275
280      INC HL ;Sor vége: átlépjük az Entert.
290      PUSH HL
300      SBC HL,DE ;vége a listának?
310      POP HL
320      JR  C,NUMBER ;ha nem, tovább dolgozunk.
325
330      LD  (GENS+70),HL ;az egyesített lista végcíme
340      JP  GENS+77 ;melyet starttal a Gensbe!
345
370 TABULA LD  (HL),9 ;a szpészt tabulátorra cserélli
380      JR  CHAR ;folytatja a sort.

```

5. lista

4. A nyomtatóra egy sort küldő rutin

Mivel az eredeti printerfunkciókkal ugyanazok a problémák, mint a képernyőre listázásnál (lásd a 2. pontot), inkább saját magunk fogunk nyomtatni. A most következő gépi kódú rutin egy sort ír ki, ezt majd a BASIC főprogramból fogjuk használni (5. pont). Az indítási cím 58885, és az ITT+1=58891 címen mindig az aktuális sor kezdőcíme van.

```

20 ;GENSB <L>LIST
30
40      ORG 58885
50
60
70      LD  A,3 ;az LPrint csatornát megrövidítük
80      CALL #1691
95
100 ITT  LD  HL,38278 ;A sor vagy lista kezdete.
110      LD  BC,-VISSZA ;ez lesz a visszafordítási cím
120      PUSH BC ;a ROM rutinból
130
140      PUSH DE
150      LD  E,(HL)
160      INC HL
170      INC HL
180      PUSH HL ;szöveg kezdőcím reaktározása;
190      EX  DE,HL ;HL=sorszám.
200
210      LD  E,32
220      LD  BC,-10000
230      CALL #192A ;először a tízezes helyértéket
240      JP  #1A30 ;ezután a többit írja ROM rutinnal
250
260 VISSZA WAP A ;a sorszám nyomtatás után nullázza
270      LD  C,A ;a sorszámot.
280      JR  TABU
290
300
310      ORG 58823 ;a sor karaktereit visszahívja
320
330 LIST LD  A,(HL) ;aktuális karakter
340      INC HL ;címnövelés
350      LD  C,A ;az aktuális kód reaktározása
360      CP  59 ;commentet a jobboldalra!
370      JR  Z,TABU
380      CP  9 ;tabulátor karakter?
390      JR  Z,TABU
400      CP  13 ;vége a sornak?
410      JR  Z,VEGE
420 CHAR RST 15 ;elővből nem nyomtatja a karaktert
430      JR  LIST ;és folytatja a sort.
440
450 VEGE LD  (ITT+1),HL ;a következő sor elejét állítja be
460      RET ;visszafordítás előtt
470
480 TABU LD  A,(TAB) ;a tabulátor leolvasása
490      INC A ;léptetése
500      LD  (TAB),A ;beírása.
510      LD  D,#E6 ;DE) fogja mutatni, hogy hányadik
520      LD  E,A ;oszlopba kell nyomtatni.
530
540      RRD A,C ;a tabulátor állás + aktuális kód
550      DEC A
560      CP  59 ;ha a Pontosv. 1. Pozícióban van
570      JR  Z,CHAR ;akkor minden marad
580
590 SPACES LD  A,32 ;legalább 1 szpész kinomtatása
600      RST 16
610      LD  A,(DE) ;a tabulátor kívánta pozíciót
620      LD  B,A
635      LD  A,(23E80) ;a nyomtatási Pozícióból
640      SUB B ;kivonjuk, ha még a tabu.Poz.

```


650	JR	D,C,SPACES	max9999b, akkor újabb szűrészt!
660			
670	LD	A,C	ha az aktuális kár. nem chr#9.
680	CP	59	hanem Pontozvesztő, akkor kiírás
690	JR	Z,CHAR	
700	JR	LIST	skálában átugrás következik.
710			
720		ellenőrizendő, hogy JEL <= #E500	
730	JEL		
740			
750	ORG	#E600	
760	DEFB	6,13,18,30	
770	TAB		

6. lista

5. BASIC főprogram a módosított Gens-Mons használatára

Az eddigi módosítások alapján van egy ékezetes Gensünk, továbbá egy megnövelt Monsunk, aminek új. hozzácsaptuk a 2—4. pontok rutinjait, és így az ékezetes betűkkel együtt egy CODE 57600,7936 felvételt készítettünk. Mindezek működését a 7. lista BASIC programja a következőképpen vezérli.

10-es sor. Start, kissé gyorsított billentyűzet beállítása. A w(c) függvény kétfajta peeket valósít meg. Ezután a gépi kódú részek beolvasása következik.

100-as sor. A menü kiírása. A válasz egy betű és egy számjegy, ez utóbbi =1, ha nincs megadva; a számjegy a használandó microdrive száma. A tabulátortáblákat mindig visszaállítjuk, mert nyomtatáskor a 934-es sor elállítja. Ezután a megfelelő funkcióra ugrunk.

700-as sor. Ez a részlet a Gens listáját a kijelölt microdrive-ra viszi, és ellenőrzi a felvételt. Először a lista start- és végcímet (ez a lista vége utáni első szabad kezesre mutat) olvassa le, majd ellenőrzi, hogy a lista első sorában kijelöltük-e a hosszmutató bájtoikat. Ennek szabálya: a lista első sorát ";"-vel kezdjük, ezt két tetszőleges karakter követi, majd egy "!"-nek kell állnia, azután akármi lehet. A lista hosszát 200-as számrendszerben, fordított sorrendben írjuk fel (a maximális hossz 39999 bájti), majd a két „számjegyet” 32-vel megnövelve elhelyezzük a ";"- és a "!" közé. Innen tudjuk majd a LOAD-nál a lista hosszát. A 32-es eltolás pedig azért kell, hogy vezérlő karakter ne kerülhessen a listába (717-es sor). Ha a hosszmutató nincs előkészítve, hibajelzés jön (716-os sor). Ha rendben van, akkor felvétel a kijelölt drive-ra, verify, majd visszatérés a menühöz.

800-as sor. Beolvasás a kijelölt drive-ról. A program megkerdezi a felvétel nevét, a lista hosszát — így lehetőség van olyan program beolvasására is, amely nem tartalmaz hosszmutatót. Ha a hossz helyett enterrel válaszolunk, akkor feltételezi, hogy van hosszmutató a beolvasandó listában. Beolvasás után, ha nem adtuk meg a hosszát és nincs hosszmutató sem, akkor hibajelzést kapunk, egyébként kiszámítja az LV végcímet. Ha a tárban már van egy lista, akkor a beolvasást rész ennek folytatásá lesz, az újrászámozást azonban nekünk kell elvégeznünk a Gensben. Előfordulhat, hogy a hossz hibásan van megadva. Ez ellen védekezik a 845-ös sor: addig csökkenteni a végcímet, amíg fenn nem áll, hogy a lista vége egy sor végével egyezik meg. Ellenkező esetben a Gens ugyanis elszáll. A Gens rendszerváltozóba tölti a lista végcímet, majd visszatér a menühöz.

900-as sor. Listázás a nyomtatóra. Feltételezzük, hogy a nyomtató legalább 64 karakter széles. Igazán akkor jön ki jól az ékezetes nyomtatás, ha olyan a nyomtató és a csatlakoztató Epromja, hogy a felhasznált grafikus krakerek nyomtatási képének bal felső 6 x 7-es részét nyomtatja ki (például Seikosha GP 100 és Microteam interfész).

Először a lista kezdőcímet adjuk át a gépi kódú programnak (lásd a 4. részt), majd a végcímet olvassuk le. A 931—934-es sorban lehetőségünk van bal oldalon margót hagyni: annyi szóközt gépelünk be. Ekkor módosul a tabulátortáblázat, amit a gépi kódú rutinok használnak (934-es sor).

Ha a nyomtató nem emel automatikusan lapot, akkor nekünk kell ezt megoldanunk: erre szolgál a 935-ös sor. Ilyenkor a listázást a lap tetején kezdjük el! Ide fog kerülni a fejlécként megadott szöveg és a lapszám, egy sor kimarad, 68 sor lista következik, majd a lap alján két sor újra kimarad (72 sor/lap). Ha a lapemelés automatikus, akkor a fejlécre enterrel válaszoljunk!

Egy sor nyomtatását végzi a 940-es sor. A nyomtatót sorokat számoljuk a 995-ös sorban, és ha már a 71. sorba íránk, akkor új lap, fejléc következik (996-os sor). Ha nincs több listarol, akkor a menü jön.

1500-as sor. Egy memóriablokkot felvesszünk a kijelölt microdrive-ra és ellenőrizzük. Utána visszatérés a menühöz.

5000-es sor. A BASIC főprogram, a módosított Gens és Mons felvétele "m";1;-re.

SLIZ MIKLÓS

7. lista

```

9 REM ----- START -----
10 POKE 23609,20 POKE 23561,15 POKE 23562,4
20 DEF FN W(C)=PEEK C+256*PEEK C+1
30 CLEAR 29999
40 LET Gens=39880 LET Mons=59000
50 LOAD #M:1:;#64,299;CODE
60 LOAD #M:1:;#64,576;CODE
90 CLS : PRINT AT 5,0: @EZETES GENSMONSH3, 6,2:verzió""
64 karakter sor"" @ EpromSoft , 1986.10.09."" Sliz Miklós"
97
98 REM ----- menü -----
99
100 POKE 23558,0 INPUT #1:AT 0,0:"PRINTER CODE-SAVE GENS M
ONS SAVE! LOAD! Text TO GENS: ENTER = STOP :;#
101 IF #M="" THEN STOP
102 LET #=1: IF LEN #>2 THEN LET #=VAL #*(2)
103 RESTORE 933: FOR I=58880 TO 58993 READ A: POKE I+1: NEXT I
105 IF #<1 OR #>5 THEN GO TO 700
106 IF #<1 THEN GO TO 800
110 IF #="G" THEN PRINT AT 21,0: RANDOMIZE USR Gens
115 IF #="P" THEN GO TO 900
116 IF #<1 THEN GO TO 1500
120 IF #="M" THEN RANDOMIZE USR Mons
130 IF #="" THEN PRINT AT 21,0: RANDOMIZE USR 58960
230 GO TO 100
697
698 REM ----- save# -----
699
700 LET LV=FN W(GENS+6695)
710 LET LV=FN W(GENS+70)
715 IF PEEK (LV+2)<=CODE # THEN GO TO
717
716 PRINT "hosszmutató?"; GO TO 100
717 LET hossz=LV-10 POKE 10+4,INT (hossz/200)+32: POKE 10+3,ho
sz+200:INT (HOSZ/200)+32
720 PRINT #; SAVE #M:;#";";
725 INPUT #M: PRINT #M: CODE "L0:;"; LV=L0
730 SAVE #M:;#M;#CODE L0,LV=L0
735 VERIFY #M:;#M;#CODE L0,LV=L0
740 GO TO 100
797
798 REM ----- load# -----
799
800 INPUT " LOAD #M:;";STR# #";";#M
805 LET LV=FN W(GENS+70)
810 INPUT "CODE ";STR# L0:;";#M
830 LOAD #M:;#M;#CODE L0
835 IF #="" AND PEEK (L0+5)<CODE # THEN GO TO 716
835 IF #="" THEN LET LV=10+PEEK (10+3)+32+200*PEEK (L0+4)+32
840 GO TO 845
845 LET LV=L0+VAL #M
845 IF PEEK (LV-3)<13 OR PEEK (LV-2)<13 OR PEEK (LV-1)<13 THEN
LET LV=LV-1 GO TO 845
850 POKE GENS+71,INT (LV/256)
850 POKE GENS+70,LV-256*PEEK (GENS+71)
870 GO TO 100
900
901 REM ----- Printer -----
902
920 POKE 58891,PEEK (GENS+6695)
921 POKE 58892,PEEK (GENS+6696)
930 LET LV=FN W(GENS+70)
931 INPUT "balmar90";#M: LET #=LEN #M
932 RESTORE 933
933 DATA 6,13,18,30
934 FOR I=58890 TO 58993 READ A: POKE I+1: NEXT I
935 LET SOR#="1E# INPUT "fejléc"; enter=#M:;#M: LET LPP#
IF #<0 THEN LET SOR#3 GO TO 996
940 LPRINT #M: RANDOMIZE USR 58885 LPRINT:
945 LET SOR#="SOR+1" IF SOR#7 THEN LET SOR#3: LET LPP#="LPP+1" LPP
950 IF SOR#3 THEN LPRINT #M;#M" : "LPP: LPRINT
997 IF FN W(58891)>LV-1 THEN LPRINT : GO TO 100
999 GO TO 940
1500
1501 REM ----- code save -----
1502
1510 POKE 23558,0
1520 INPUT " SAVE CODE ""from;" TO "C1,C2
1525 LET l=2-c+1+1
1540 INPUT "name ";#M: PRINT " SAVE #M:;#M:;";#M;#CODE "l:c1
";#M";
1550 SAVE #M:;#M;#CODE C1,1
1555 VERIFY #M:;#M;#CODE C1,1
1590 GO TO 100
4990
4991 REM ----- szponzítás -----
4992
5000 CLEAR : SAVE #M:1:"rom" LINE 10
5010 VERIFY #M:1:"rom" LINE 10
5020 SAVE #M:1:;#64,576;CODE 57600,7936
5025 VERIFY #M:1:;#64,576;CODE 57600,7936
5030 SAVE #M:1:;#64,299;CODE 29900,8380
5035 VERIFY #M:1:;#64,299;CODE 29900,8380

```

Kalkulációs program adóelszámolóshoz

ZX-SPECTRUM

Az egész világon terjednek a típusprogramok: kalkulációs, szövegszerkesztő, adatkezelő stb. szoftverek. Ezekkel a gyakorlati életben felmerülő problémák jelentős részét meg lehet oldani. Ily módon egyszerűen és rövid idő alatt, programnyelvek ismerete nélkül is rendkívül hasznos programok állíthatók össze. Hogy ez mennyire igaz, azt egy példával szeretném bizonyítani.

A feladat egy gazdasági munkaközösségen belül a tagok személyi jövedelmének kiszámítása és nyilvántartása, beleértve a TBI-járulékot és a jövedelemadót. Ezt végzi el az a program, amely a ZX-Spectrum számítógépen futó VU-CALC kalkulációs segédprogrammal együtt működik, de az elve minden más gépen használatos kalkulációs programra adaptálható. A VU-CALC se-

gédprogramot a Mikroszámítógép Magazin az 1984/2. számában részletesen ismertette. Jelen leírás épít a VU-CALC-ban szerzett gyakorlatra. Maga a programozási munka csak a celláknak (vagy boxoknak) az 1. lista szerinti feltöltéséből áll. E listán a cellák azonosítóiban a numerikus karakter jelzi a VU-CALC megfelelő oszlopát, a betű pedig a sorát.

A program által készíthető nyilvántartás rész táblázatokból épül fel. Ezek méretét a VU-CALC-beli 18x4 cella (box) határozza meg, amit a számítógépes feldolgozás szempontjából ablaknak nevezünk, valamint a sor- és oszlopkoordináták. Így az egyes rész táblázatok tartalma éppen megfelel a képernyőn látható képek és a kinyomtatás méretének. A tag nevén kívül a rész-

táblázatok személyenként tartalmazzák a nyereségelőlegnek az aktuális (tárgyhavi) számítás/kifizetés időpontjáig járó részét, az ebből korábban felvett összeget, továbbá a tárgyhavi elszámolást, részletezve a levonásokkal:

- költséghányad,
- TBI-járulék,
- jövedelemadó.

A rész táblázatok felépítése

1. ablak 1-4.oszlop = cím és megnevezés
Ez a rész táblázat tetszőlegesen kitölthető. Például: Mikrobi GM. Budapest. Adóelszámolás
2. ablak 5-8. oszlop = számítások elvégzése
Minden egyes tag jövedelmének számítá-

Megjegyzések a programlistához

1. Az 5-ös oszlopban csak szöveges kifejezések szerepelnek.
2. A 6-os oszlopban csak matematikai kifejezések szerepelnek.
3. A 7-es oszlopban a B, F, O, R sorba 0-t kell írni, a többi üres marad.
4. A 8-as oszlopban is csak a B, F, O, R sorban szerepel kifejezés, a többi üresen marad.
5. A program 200 000 Ft adóalapig jó. E fölött további sorok be-
szűrése szükséges.
6. A B6 cellába tetszőleges érték beírható.

1. lista

Jövedelem nyilvántartás
Programlista

A5:NEV	A6:	A7:	A8:
B5:kereset	B6:10000	B7:0	B8:B6-B7
C5:-10%kts	C6:B6*.1	C7:	C8:
D5:	D6:	D7:	D8:
E5:maradek	E6:B6-C6	E7:	E8:
F5:- TBI	F6:E6*.1	F7:0	F8:F6-F7
G5:	G6:	G7:	G8:
H5:Adoalap	H6:E6-F6	H7:	H8:
I5:	I6:H6*.02	I7:	I8:
J5:	J6:(H6-20000)*(H6>20000)*.04		
K5:	K6:(H6-40000)*(H6>40000)*.04		
L5:	L6:(H6-60000)*(H6>60000)*.1		
M5:	M6:(H6-80000)*(H6>80000)*.1		
N5:	N6:(H6-100000)*(H6>100000)*.15		
O5:-adocsz	O6:8I6+N6	O7:0	O8:O5-O7
P5:+10%kts	P6:C6	P7:	P8:
Q5:	Q6:	Q7:	Q8:
R5:kifizet	R6:H6-O5+C6	R7:0	R8:R6-R7

	O5	O6	O7	O8
B	BARNA I			
k	kereset	1200000	630000	660000
-	-10%kts	120000	63000	
mar	maradek	118100	56700	
-	TBI	11810	5670	5940
ad	Adoalap	1044000	610000	
0%	0%	00000	10000	
10%	10%	33700	12000	
20%	20%	5700	441	
30%	30%	4400	0	
40%	40%	4400	0	
45%	45%	6700	0	
-	-adocsz	166100	0	12916
+	+10%kts	120000	63000	
k	K i f i z e t	101772	54628	47144

2. lista

3. lista

	13	14	15	16
B	SZALAI			
k	kereset	269000	54000	215000
-	-10%kts	26900	5400	
mar	maradek	24210	4860	
-	TBI	2421	486	1935
ad	Adoalap	217800	4074	
0%	0%	436	07	
6%	6%	71	00	
10%	10%	0	00	
20%	20%	0	00	
30%	30%	0	00	
40%	40%	0	00	
-	-adocsz	6000	0	419
+	+10%kts	26900	5400	
k	K i f i z e t	23973	4827	19146

Hasznos rutinok C64-re

sa itt zajlik, az 1. lista szerinti programmal. A képernyőn csak az aktuális számítás eredménye jelenik meg (2. lista). A kapott eredményeket minden egyes futtatás után el kell helyezni a nyilvántartás megfelelő résztáblázatában, vagyis a tag sorszáma szerinti ablakba át kell vinni.

3. ablak 9—12. oszlop = 1. tag adatai

4. ablak 13—16. oszlop = 2. tag adatai (3. lista)

n+2. ablak a—(a+3). oszlop = n. tag adatai
Itt a értéke 4n+5.

Az első kivételével minden résztáblázat felépítése egyforma.

A számítás módja az első hónapban

Az A5 cellába a tag nevét, a B6 cellába a tag bruttó nyereségét írjuk be. Ezután CALC parancs kiadását követően a 6. és 8. oszlopban megjelennek a számított értékek. Végül az oszlopokat nyilvántartás céljából transzfer paranccsal átvisszük a megfelelő résztáblázatba:

t, 5, a Az a értéke: 4n+5

t, 6, a+1

t, 7, a+2 (n a tag sorszáma)

t, 8, a+3

Természetesen a kiadott parancsba az „a” kiszámított értékét — ez 9-től 57-ig terjedő természetes szám lehet — kell beírni.

A számítás módja a további hónapokban

Először visszahozzuk a számítandó tag adatait résztáblázatból:

t, a+1, 7

(Az „a” értéke ugyanaz, mint korábban.)

Ez lesz az előző időszak oszlopa.

Innentől a számítás megegyezik az első hónapban követett módszerrel. Megfigyelhető, hogy a 6-os oszlop a teljes, a 7-es oszlop a megelőző, a 8-as oszlop a tárgyidőszak kifejezések tartalmazza.

E program tárolására az s (save) parancs, betöltésére az l (load) parancs szolgál VU—CALC üzemmódban. Kinyomtatása a p paranccsal érhető el. Vigyázni kell, mert csak a képernyő tartalmát nyomtatja ki. Ha bármilyen ok miatt a gép kiesik a kalkulációs üzemmódból, GOTO 3300 segítségével térhetünk vissza.

A VU—CALC betöltése a szokásos: LO-AD, és a program automatikusan indul.

SZALAI GÉZA

A program három fő részből áll: a rutin-
szétválasztóból és a két rutinból.

A rutinsztétválasztó a 828-839 (033C — 0347) címeken helyezkedik el. Először meghívja azt a KERNAL rutint, amely megvizsgálja, hogy vessző következék-e, és ezután betölt az X regiszterbe egy nyolcbites számot. Ezt a számot megvizsgálja: ha 1, akkor a 840-es (0348) címre, a 1. rutinra ugrik, ha 2, akkor a 857-es (0359) címre, a 2. rutinra ugrik. Ha valami más szám van ott, a program megáll.

Az egyik rutin a kurzor pozicionálását teszi lehetővé. Ami egy kicsit furcsa benne, az az, hogy először az Y koordinátát kell megadni. Működése egyszerű. Bekér egy nyolcbites számot, az lesz az Y koordináta, majd bekér egy újabb nyolcbites számot, az lesz az X koordináta. Ezután meghívja a PLOT KERNAL rutint, amely az X, Y regiszterek értéke szerint a kurzort az adott helyre irányítja. A kurzorpozicionáló rutin a 840—856 (0348—0358) címeken helyezkedik el. Hívása SYS 828,1,Y koordinat, X koordinat.

A teljes program

033C	1	+=B2B									
D020	5 KERE	=#D020;KERETSZIN TARCIME									
D021	10 ALAP	=#D021;ALAPSZIN TARCIME									
02B6	15 KARA	=#02B6;KARAKYBERZIN TARCIME									
B7F1	20 SZAM	=#B7F1;EGYBYTE-OS SZAM BEVETELE									
FFF0	25 CURS	=#FFF0;KURZORPOZICIONALAS TARCIM									
033C 20 F1 B7	30	JSR SZAM									
033F E0 01	40	CFX #01									
0341 F0 05	50	BEQ CU									
0343 E0 02	60	CFX #02									
0345 F0 12	70	BEQ PD									
0347 60	80	RTS									
034B 20 F1 B7	90	JSR SZAM									
034B 8A	100	TXA									
034C 4B	110	PHA									
034D 20 F1 B7	120	JSR SZAM									
0350 8A	130	TXA									
0351 AB	140	TAY									
0352 6B	150	PLA									
0353 AA	160	TAX									
0354 1B	170	CLC									
0355 20 F0 FF	180	JSR CURS									
035B 60	190	RTS									
0359 20 F1 B7	200	PO									
035C 8E 20 D0	210	STX KERE									
035F 20 F1 B7	220	JSR SZAM									
0362 8E 21 D0	230	STX ALAP									
0365 20 F1 B7	240	JSR SZAM									
0368 8E 8A 02	250	STX KARA									
036B 60	260	RTS									
036C	270	.END									
ALAP	=#D021	CU	=#034B	CURS	=#FFF0	KARA	=#02B6	KERE	=#D020	PD	=#0359
SZAM	=#B7F1										

Lapdobás

Vállalati C64-es adatfeldolgozásnál gyakran előfordul összefüggő adatblokkok tablózása. A program ezek lapdobásos kiírását oldja meg.

Első része a 251-es bajton számolja a kiadott PRINT #4 utasításokat. A második rész ellenőrzi, van-e elegendő hely a legna-

COMMODORE 64

gyobb adatblokk számára. Ha nincs, lapot dob és nullázza a 251-es bajtot.

A tablózás menete a következő lehet:
1. Betöltjük az 1. lista szerinti BASIC-töltő programot, és a 251-es bajtot nullázzuk (POKE 251,0).

2. A 252-es bajtra az előforduló legnagyobb adatblokk sorainak számánál 2-3-mal nagyobb számot írunk be (például: POKE 252,1 0).

```

10 REM LAPDOBS 'BASIC BETOLTOJE'
20
30 FOR I=828 TO 906
40 READ A:POKE I,A:XX+A+1
50 NEXT I
60 IF X<79383 THEN PRINT"HIRAS":END
90 SYS828:NEW
100 END
110 DATA 162,83,189,70,3,149,114,202,209
120 DATA 248,96,76,74,3,230,122,208,82
130 DATA 230,123,32,121,0,201,152,208,89
140 DATA 152,72,164,251,209,132,251,184
150 DATA 168,76,121,0,169,72,56,229,232
160 DATA 197,251,16,30,139,72,162,4,32
170 DATA 201,255,169,72,56,229,251,133
180 DATA 251,162,0,32,215,170,232,228
190 DATA 251,208,248,169,0,133,251
200 DATA 184,170,96

```

1. lista

```

4
5
6
7
8
9
10 ##833C
20 LDK ##93
30 LF LDA CIM-L,X
40 STA ##972,X
50 DEK
60 BNE LP /20-60 A CHRDET RUTIN
70 RTS /ATIRASA
75 / A PRINT#4 UTASITASOK SZAMOLASA
76 ## 833C SZAMONAK TAROLASA A #B-BEN
80 CIM JPM UJCIIM
90 UJCIIM INC #7A
100 BNE KI
110 INC #7B
120 KI JSR ##979
130 CNP ##9E /A PRINT#4 TONKEJE
140 BNE VISSZA
150 TVR
160 PHA
170 LDV ##0FB
210 INV
230 STV ##0FB
240 PLA
250 TRV
260 VISSZA JMP ##979
305 / A KIIRT SOROK SZAMONAK ELLENORZESE
330 LDA ##48 /EDY LAP 72 SOR
340 SEC
350 SBC ##0FC
380 CNP ##0FB
390 BPL VISSZI
391 / LAPDOBS AS A #FB NULLAZASA
392 TXA
393 PHA
400 LDX ##04
410 JSR ##FC9 /SOREMELES ELOKESZITESE
420 LDA ##48
430 SEC
440 SBC ##0FB
450 STA ##0FB
460 LDK ##00
510 HR JSR ##AD7 /SOREMELES
520 INX
530 CPX ##0FB
540 BNE HR
550 LDA ##00
560 STA ##0FB
560 PLA
570 TAX
600 VISSZI RTS

```

2. lista

```

10 REM PELDAPROGRAM
20 OPEN#4:POKE251,0
30 INPUT"KEREM AZ ADATBLOKK HOSSZAT":X
40 INPUT"KEREM A DATUMOT ":D#
50 POKE 252,X
60 FOR I=1 TO 30
70 GOSUB 170
80 PRINT#4," " SOR-1"
90 PRINT#4," " SOR-2"
100 PRINT#4," " SOR-3"
110 PRINT#4," " SOR-4"
120 PRINT#4," " SOR-5"
130 PRINT#4," =====":PRINT#4
140 NEXT I
150 PRINT#4:CLOSE4:END
170 SYS867
180 IF PEEK(251)<>0 THEN RETURN
190 Z=Z+1
200 PRINT#4,CHR$(16)"30"Z", "D#"
210 PRINT#4:RETURN

```

3. lista

3. Behívjuk a 2. listájú táblózó BASIC programot, melyben minden adatblokkot kiíró rész előtt szerepel a SYS 867, a lapot lapteőre állítjuk, és futtatjuk a programot. A példaprogram (3. lista) minden lap teőjére felírja a lapszámot és a dátumot is. TÓTH GYULA

soros illesztőn keresztül

A programrészt RS232 interfészen keresztül sztring formájában érkező sorokat fűz be a programba. Az ötlet lényege: mivel a C64 billentyűpuffere 10 karaktert tud fogadni, és ha ezt 13-mal töltjük fel, akkor a későbbiekben paraméterezett nyomtatás során a programból kilépe automatikusan be lehet új sorokat vinni, illetve parancsokat lehet végrehajtani.

A 63000-es sorban a kurzor pozicionálása után megnyitjuk a soros illesztőt az adatonval paramétereivel azonosan, és beolvasunk az első sztringet, melynek szerepe kettős. Egyrészt gyakorlatban tapasztalható, hogy a fájl megnyitása után soros illesztő esetén az első karakter gyakran téveszt, ezt szűrjük ki egy fel nem használt sztring átvételével, másrészt ez a sztring adja a szinkronizálást. Nem kezdődhet számmal, és nem lehet olyan, ami értelmezhető parancsként is. Amennyiben ez a sztring nem a helyén érkezik, kiíródik a képernyőre, és a gép szintaxis hiba jelzésével leáll.

9 programsor bevételezése és kiírása után lezárjuk a fájlt, és a gép „RUN 63000” sztringet ír ki, mely a programrészt újbóli indítását, és ezzel újabb 9 sor bevételezését kezdeményezi.

A 63006-os sorban a billentyűpuffert (631–640. memóriacím) CR kóddal (13) töltjük fel, és a puffertömbbe (198. cím) betöltjük a puffertben lévő karakterek számát (10), majd a kurzort a bal felső sarokba vezéreljük, és áttalunk parancs üzemmódba. Ezt követően a sorok bevételezése automatikus.

A betöltés végén egy „%” karaktert vár a

```

63000 PRINT"###"OPEN#1,2,2,CHR$(13) :IMP
UT#1,8:FOR I=1TO9:BE=I+1
63001 GET#1,8:IF BE=""THEN63001
63002 IFR#CHR$(13)THENPRINT#1:GOTO63005
63003 BE=BE+BE:IF BE=CHR$(10)+"%":THENPRINT#
1,8,63007:I=I+1:GOTO63005
63004 GOTO63001
63005 NEXT:CLOSE1:PRINT"R 63000"
63006 FOR I=8TO9:POKE631+I,13:NEXT:POKE19
8,18:PRINT"R" :END
63007 PRINT"###" :FOR I=8TO9:PRINT#1+63000
NEXT:PRINT" C (147)":GOTO63006

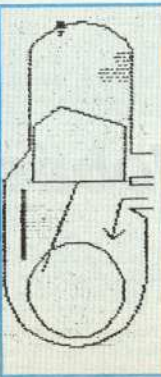
```

programrészt. Ezt kiértékelve a gép kirírja a „RUN 63007” sztringet, mely parancs üzemmódban a teljes programrészt kitörölés után maga után. Vigyázni kell, hogy a kinyomtatható (megjelenítendő) sztringek hossza ne haladja meg a 80 karaktert, mert ez esetben a túlszorduló részt a gép nem értelmezi, a sor bevétele után a kurzor a csonka sorra ugrik, és a következő bevételezésnél szintaxis hibával áll le, vagy ha tudja is értelmezni, kiesik a szinkronizmusból. A hibák leggyakoribb oka, hogy olyankor küldünk adatokat, amikor a bevő fogadituka nincs még kész. A sztringekben rövidített kulcsszavak is lehetnek (PRINT helyett % stb.), mert a gép ezeket is értelmezi.

A programok futás közbeni változtatására is alkalmas az alapötlet. Használatával a futó program a lemezről vagy kazettáról beolvasás közben módosítható, vagy sorokat törölhet önmagában a futás részereményeitől függően. Ilyenkor a változók elvesznek, ezért mentésükről és újbóli betöltésükről gondoskodni kell.

KRAUSS OTTÓ

A kétütemű motor működése



ZX-SPECTRUM

A program metszetben megjeleníti a kétütemű motort, és bemutatja a működés különböző fázisait: szívás, sűrítés, robbanás (munkavégzés), kipufogás. Az s billentyűvel a program megállítható, a c billentyűvel pedig kimásolható a kívánt rajzállapot.

```

1 REM © Nyitrai 86
5 REM 2 ütemű motor működése
10 GO SUB 290: BORDER 7: PAPER
1: INK 6: CLS
20 CIRCLE 125,39,24
30 PRINT AT 19,0: "Ket ütemű" :A
T 20,0: "motor":AT 21,0:"működés
e"
40 PLOT 102,160: DRAW 0,-50: D
RAW -12,-15: DRAW 0,-50: DRAW 70
0,PI: DRAW 0,30: DRAW -6,5: DRA
W 15,0: PLOT PEEK 23677,PEEK 236
78+12
50 DRAW -15,0: DRAW 0,4: DRAW
15,0: PLOT PEEK 23677,PEEK 23678
+12: DRAW -15,0: DRAW 0,53: DRA
W -50,0,2: PLOT 100,90: DRAW 0,-3

```

```

5: DRAW -1,0: DRAW 0,35
60 PRINT AT 0,14; INK 7; BRIGH
T 1;"A"
70 FOR a=(2*PI)-PI/6 TO 0 STEP
-PI/6
80 FOR z=0 TO 1
90 LET b=126+20*COS a: LET c=3
9+20*SIN a
100 PLOT OVER 1;b,c: DRAW OVER
1;126-b,45
110 LET d=c+45
120 OVER 1: PLOT 105,d: DRAW 44
,0: DRAW 0,28: DRAW -32,10: DRAW
-13,-10: DRAW 0,-26: OVER 0
130 IF INT c=59 THEN GO SUB 210
140 IF INT c=56 AND b>126 THEN
PRINT AT 0,0;"Munkavegzes":AT 1,
14;"":AT 2,13;"":AT 3,1
3;"":
150 IF INT c=21 AND b>126 THEN
GO SUB 230
160 IF INT c=38 AND b<126 THEN
PRINT AT 0,0;"Surites"
170 IF INT c=48 AND INT b<126 T
HEN GO SUB 280
180 IF INKEY$="c" AND z=0 THEN
COPY
190 IF INKEY$="s" AND z=0 THEN
GO TO 190
200 NEXT z: NEXT a: GO TO 70
210 PRINT AT 0,0;"robbanás"
AT 1,14; INK 2; BRIGHT 1;"
AT 2,13;"":AT 3,13;"
IF z=0 THEN FOR x=0 TO 10: BE
EP .0110: BEEP .01,20: NEXT x
220 RETURN
230 PRINT AT 0,0;"
OVER 1: PLOT 134,122: DRAW 20,-
20, PI/2: PLOT 155,102: DRAW 16,
0: PLOT 166,106: DRAW 5,-4: DRAW
-5,-4: PRINT AT 9,22;"Kipuffoga
s"
240 PRINT AT 9,0;"Uzemanyag";AT
10,5;"fel": PLOT 96,40: DRAW 0,
55: DRAW 20,22: DRAW -5,0: POKE
23677,PEEK 23677+5: DRAW 0,-5: O
VER 0
250 IF z=0 THEN FOR x=1 TO 100:
IF INKEY$="c" THEN COPY
260 IF z=0 THEN NEXT x
270 RETURN
280 OVER 1: PLOT 150,85: DRAW -
12,0: DRAW -5,-20: DRAW -4,5: PO
KE 23677,PEEK 23677+11: DRAW -6,
-5: PRINT AT 11,21;"Uzemanyag";A
T 12,21;"beszivás";AT 13,25;"ve
ge": OVER 0: GO TO 250
290 RESTORE 300: FOR x=0 TO 14:
READ y: POKE USR "a"+x,y: NEXT
x
291 RETURN
300 DATA BIN 00001110,BIN 00001
110,BIN 00011111,BIN 00011111,32
,BIN 11001110,BIN 00000100,BIN 0
0001100,BIN 01011010,0,BIN 10101
001,0,BIN 01001010,0,EIN 0010010
0,0
310 SAVE "motor" LINE 1
530 LET szamitas=szamitas+1: RE
AD d
540 LET c(szamitas,1)=c+COS (PI
/6): LET c(szamitas,2)=d+(-c*SIN
(PI/6))
550 GO TO 525
610 PLOT 150,0
620 FOR n=1 TO szamitas: DRAW c
(n,1),c(n,2): NEXT n
630 LET e=magassag+COS (PI/6)
LET f=magassag*SIN (PI/6)
640 PLOT 150,0: DRAW e,f
650 FOR n=1 TO szamitas: DRAW c
(n,1),c(n,2): DRAW -e,-f: DRAW e
,f: NEXT n
660 INPUT "Kimasoljam ? i/n ";
LINE k$: IF k$="i" THEN COPY
999 STOP
1000 FOR n=0 TO 7: READ a: POKE
USR "a"+n,a: NEXT n: DATA 48,72,
72,48,0,0,0,0
1005 PRINT TAB 10: INVERSE 1;"3-
D RAJZPROGRAM": INVERSE 0
1050 PRINT AT 20,0;"Nincs egy go
mbot a rajzkeszites
hez": PAUSE 0: GO TO 40
9000 DATA -50,0,-10,20,40,20,
-40,1000
9999 SAVE "3D rajz"

```

Háromdimenziós ábrázoló geometria

A program megkérdezi a kért idom magasságát, nézeti számát, és három nézetben lerajzolja, végül megjeleníti három dimenzió-ban is. Bármelyik nézet kinyomtatható. A 9000-es DATA sort bármilyen megrajzolt egységű ábrára ki lehet cserélni, amelyet a program háromdimenziósról átrajzol.

NYITRAI LÁSZLÓ

```

5 REM © Nyitrai 86'
6 REM ábrázoló geometria
10 BORDER 1: PAPER 2: INK 6: C
LS : RESTORE : GO TO 1000
40 DIM a(100,2): DIM b(100): L
ET szamitas=0: LET a1=0: LET a2=
0: LET a3=0
50 READ c: IF c=1000 THEN GO T
O 61
51 LET a1=a1+c: IF a1<a2 THEN
LET a2=a1
52 IF a1>a3 THEN LET a3=a1
60 LET szamitas=szamitas+1: RE
AD d: LET a(szamitas,1)=c: LET a
(szamitas,2)=d: GO TO 50
61 LET szelesseg=a3-a2
69 REM * FORGASSIKOK *
70 INPUT "Az idom magassaga,(m
in 2,max 50) ? ";magassag: IF ma
gassag<2 OR magassag>50 THEN GO
TO 70
72 CLS
75 PRINT AT 0,0;"Az idom forged
tasa: INPUT "Nézetek száma ?
(min 1)2":nezet: LET nezet=INT ne
zet: IF nezet<2 THEN GO TO 75
76 LET nezet=nezet-1
77 LET pp=(127+szelesseg/2)-a3
80 FOR n=0 TO PI/2+.001 STEP P
I/(2*nezet)
90 FOR m=1 TO szamitas: LET b(
m)=a(m,2)+COS n: NEXT m
95 LET a=(INT ((PI/2-n)*190/P
I+2))*100+.5)/100
100 CLS : PRINT "Szög: ";a;"°";
PLOT pp,70: FOR m=1 TO szamita
s: DRAW a(m,1),b(m): NEXT m: IF
n=0 THEN GO TO 200
150 LET x=magassag*SIN n: PLOT
pp,70: DRAW 0,-x: FOR m=1 TO sza
mitas: DRAW a(m,1),b(m): DRAW 0,
x: DRAW 0,-x: NEXT m
200 INPUT "Kimasoljam ? i/n ";
LINE k$: IF k$="i" THEN COPY
220 NEXT n
225 REM *RAJZKESZITES*
230 LET pp=ABS a2
250 CLS : PRINT AT 0,0;"Az idom
három nézetben: "
260 PLOT pp,70: FOR n=1 TO szam
itas: DRAW a(n,1),a(n,2): NEXT n
270 PLOT 170,70: FOR n=1 TO sza
mitas: DRAW 0,a(n,2): NEXT n: PL
OT 170,70: DRAW 0,magassag: FOR
n=1 TO szamitas: DRAW 0,a(n,2):
DRAW -magassag,0: DRAW magassag,
0: NEXT n
280 PLOT pp,10: FOR n=1 TO szam
itas: DRAW a(n,1),0: NEXT n: PLO
T pp,10: DRAW 0,magassag: FOR n=
1 TO szamitas: DRAW a(n,1),0: DR
AW 0,-magassag: DRAW 0,magassag:
NEXT n
290 INPUT "Kimasoljam ? i/n ";
LINE k$: IF k$="i" THEN COPY
300 REM * FERDE VONAL *
310 LET x=COS (PI/4):magassag*
2: LET pp=120+szelesseg/2
315 CLS : PRINT AT 5,0;"AZ idom
harant nézetben"
320 PLOT pp,10: FOR n=1 TO szam
itas: DRAW a(n,1),a(n,2): NEXT n
: PLOT pp,10: DRAW x,x: FOR n=1
TO szamitas: DRAW a(n,1),a(n,2):
DRAW -x,-x: DRAW x,x: NEXT n
330 INPUT "Kimasoljam ? i/n ";
LINE k$: IF k$="i" THEN COPY
499 REM *3-D RAJZ *
500 CLS : RESTORE 3000: PRINT "
Az idom nézete 3 dimenzióban"
520 DIM c(50,2): LET szamitas=0
525 READ c: IF c=1000 THEN GO T
O 600

```

Vigyázat! Tolvaj!

Az előző alkalommal már foglalkoztunk a mágneslemez sávok szerkezetével és a sávok szektorokra osztásával. Most nézzük, hogyan épül fel egy adatblokk! Az 1. ábra a mágneslemez egy sávjának részét ábrázolja. Láthatjuk, hogy az egyes blokkokat és a blokk fejrészt az adatblokkot szinkronjel választja el. Ez olyan speciális jelsorozat, ami a lemez formattálásakor hardver úton kerül a lemezre. Nem más, mint öt \$FF bájt, azaz 40 1-es értékű bit. Mint látni fogjuk, a GCR kódolás miatt ugyanilyen jelsorozatot a szinkronjelen kívül nincs a lemezen. A DOS a szinkronjel segítségével találja meg a blokkok elejét. Használata nélkül az adatblokkok egybefolynának, így nem lehetne azokat különválasztani.

A blokk fejrésze

A rendszer a blokk fejrészének adatai szerint azonosítja az író-olvasó fej pozícióját, vagyis, hogy a lemez melyik részén dolgozik. Itt hívom fel az olvasó figyelmét, hogy a „1541 User's Manual” című könyv 54. oldalán és a dr. Úry László: Commodore 64 című könyv 180. oldalán található ábra hibás, téves információt tartalmaz. A helyes elrendezés a 2. ábrán látható.

Szinkronjel. Erről ismeri fel a DOS, hogy adatblokk vagy blokkfej következik. Negyven 1-es értékű bit.

A blokkfej azonosítója (header block ID). Minden esetben \$08. Ez tudatja a rendszerrel, hogy a szinkronjel után a blokkfej következik. Az adatrész előtt ez \$07.

Blokkfejellenőrző összeg (header block checksum). A blokkfej adatainak ellenőrzésére szolgál. Magát az értéket a sávszám, szektorszám és a két ID karakter közötti EOR (kizáró vagy) művelet elvégzése után kapjuk. Ha ez nem egyezik meg az adatok olvasása után kiszámított értékkel, akkor hibajelzést kapunk.

Szektorszám (sector number). Megadja, hogy az illető blokk melyik szektorban van.

Sávszám (track number). Megadja, hogy az illető blokk melyik sávban van. A DOS erről állapítja meg, illetőleg ellenőrzi, hogy az író-olvasó fejet helyesen pozicionálta-e.

ID karakterek. Az ID (lemezazonosító) második és első karaktere; erről ismeri fel a DOS a lemezcsere-t. Azért veszélyes több lemeznek ugyanazt az ID-t adni, mert akkor nem tudja megkülönböztetni az egyes lemezeket, és így felülíródnak rajta az adatok.

\$0F bájtok. A blokkfej végét jelölik.

Blokkfejérés (header gap) Ez nyolc \$55 bájt. A rendszer sosem olvassa, ez alatt az idő alatt készül fel az adatok fogadására. A 1541-es és a 4040-es meghajtó esetén eltérnek egymástól, ezért ezek csak olvasáskompatibilisek.

A blokk adatrésze (3. ábra)

Szinkronjel. Nyolc \$FF bájt, azaz negyven 1-es értékű bit.

A blokk adatrészének azonosítója (data block ID). Minden esetben \$07 bájt. Ezzel különbözteti meg a DOS a szinkronjel után a blokkfejet az adatrészről.

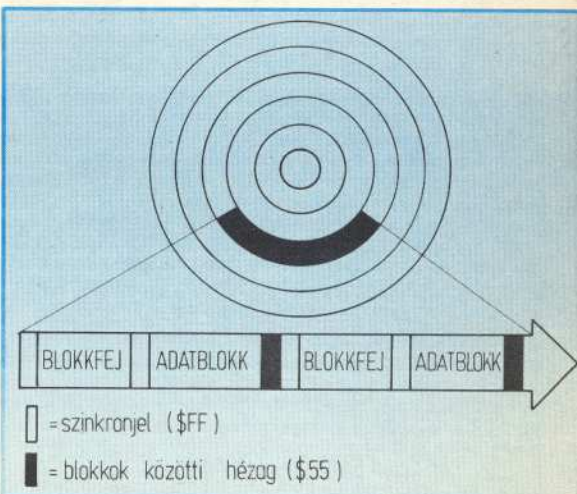
Az adatblokk ellenőrző összege (data block checksum). A 256 adatbájt közötti EOR (kizáró vagy) művelet elvégzéséből adódik.

\$00 bájt. Az adatblokkot záró két bájt. A DOS formattálásakor írja fel őket azért, hogy az adathossz esetleges elcsúszása esetén kiegyenlítse a blokkot, és ne sérüljön meg a szinkronjel.

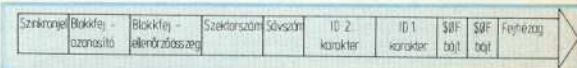
Szektorok közötti rés (inter-sector gap). \$55 bájt, amelynek hosszúsága 4 és 12 bájt között változik.

Hibafelismerő (GCR) kód

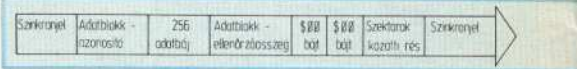
A 1541-es meghajtónak két írásmódja van: a szinkronjel mód és a normál adatbájt mód. A DOS a szinkronjel (\$FF), a szektorok közötti rés, valamint a blokkfej és az adatblokk közötti rés (\$55) kivételével az adatokat GCR (Group Code Recording) kódban tárolja.



1. ábra



2. ábra



3. ábra

GCR konverziójára a rendszer minden bájt alsó és felső négy bitjéből ötöt állít elő, így egy GCR bájt tíz bitből áll. Ezt a konverziós rutint a meghajtó memóriájában a \$F78F címen találhatjuk (lásd: Das grosse Floppy-Buch, 251. oldal). A GCR konverzióhoz szükséges táblázat a következő:

Hexadecimális	Bináris	GCR
\$00 (0)	0000	01010
\$01 (1)	0001	01011
\$02 (2)	0010	10010
\$03 (3)	0011	10011
\$04 (4)	0100	01110
\$05 (5)	0101	01111
\$06 (6)	0110	10110
\$07 (7)	0111	10111
\$08 (8)	1000	01001
\$09 (9)	1001	11001

ADOK-VESEK-CSERÉLEK

\$0A (10)	1010	11010
\$0B (11)	1011	11011
\$0C (12)	1001	01101
\$0D (13)	1101	11101
\$0E (14)	1110	11110
\$0F (15)	1111	10101

Nézzük meg például, hogy a \$12 átváltásához milyen lépésekre van szükség:

1. lépés: hexadecimális-bináris átváltás.
 $\$12 (18) = 00010010$
2. lépés: a bájt felső négy bitjének GCR konverziója.
 $0001xxxx = \$1(1) = 01011$
3. lépés: a bájt alsó négy bitjének GCR konverziója.
 $xxxx0010 = \$2(2) = 10010$
4. lépés: a GCR kód egyesítése (konkatenációja).
 $01011 + 10010 = 0101110010$

Most pedig lépésről lépésre bemutatjuk, milyen bitműveleteket végez a rendszer egy adatblokk felírása előtt. Tekintsük az adatblokk egymást követő négy bájttját. Legyenek ezek a \$08, \$10, \$00, \$12 számok.

1. lépés: hexadecimális-bináris átváltás

\$08	\$10	\$00	\$12
00001000	00010000	00000000	00010010
2. lépés: bináris-GCR átváltás.
 1. A 8 bites adatbájtok alsó és felső négy bitjének elkülönítése:

00001000	00010000	00000000	00010010
0000 1000	0001 0000	0000 0000	0000 0001 0010
 2. A bájtok alsó és felső négy bitjének GCR megfelelői:

01010	01001	01011	01010	01010	01010	01011	001010
-------	-------	-------	-------	-------	-------	-------	--------
 3. A számok 10 bit GCR-bájt megfelelői:

0101001001	0101101010	0101001010	0101110010
------------	------------	------------	------------
3. lépés: 10 bit GCR-bájtról 8 bit GCR-bájtra történő konverzió.
 1. A 10 bit GCR-bájt konkatenációja:

01010010010101101001010010110010

 2. A bitsorozat nyolcbitenkénti szétválasztása:

01010010	01010110	10100101	00101001	01110010
\$52	\$56	\$A5	\$29	\$72
4. lépés: bináris-hexadecimális átváltás

01010010	01010110	10100101	00101001	01110010
\$52	\$56	\$A5	\$29	\$72

Tehát a \$08, \$10, \$00, \$12 számszámok a GCR megfelelője: \$52, \$56, \$A5, \$29, \$72.

Láthatjuk, hogy a \$FF GCR megfelelője 1010110101, tehát a DOS egyértelműen megkülönbözteti a \$FF adatbájtot a szinkronjeltől.

Ezek után nézzük, hány bájtól áll valójában egy adatblokk.

	Adatbájt	GCR-bájt
Szinkronjel	5*	5
Blokkfej	8	10
Fejrés	8*	8
Szinkronjel	5*	5
Adatok	260	325

A *-gal jelzett bájtoknál nincs GCR konverzió. A Floppy Disc Controller (FDC) vagyis a lemez meghajtó vezérlője a GCR kódot hiba- és szinkronjel-felismerésre is használja. Mivel a Hamming-távolság kettő, ezért hibajavításra nem alkalmas, csak az adat rögzítési hibák felismerésére. Hiba esetén az FDC a hibás adatot \$FF bájjal cseréli fel.

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépell soronként (60 karakter) 100,-Ft, magán személyeknek az első sor 50,-Ft, minden további sor 20,-Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseket a szerkesztőség címére várjuk.

C-16-os játékprogramokat cserélnek. Tóth Mihály, E-mód, Akác u. 9. 3432.
Commodore 16-os, 20-as, 64-es, 128-as géppel rendelkezőkkel kapcsolatban keresek kb. 4ezer programot van. Gyermán Sándor, Zrenjanin Rade Koncara 23/V. 23000. Jugoszlávia
16 K-s ZX81 számítógép olcsón eladó. Megvételre ke- resem a Mikroszámítógép Ma- gazin 1986/6. számát. 48 K-s ZX-Spectrumhoz játék-és e- gyéb programokat cserélek, as- semly fordítóprogramot ke- resek. Hidvégi András, Komló, Ifjúság útja 34. 7300.

48 K-s ZX-Spectrumhoz felhasználói és játékpogram- mokat cserélek. Válaszokat listával kérek: Kernbaum Ant- tal, Bicske, Kisfaludy út 25. I. 29. 206G.
ZX-Spectrum 48 K, 14 000 forintért és GP-505 nyomató 15 000 forintért eladó. Varga Zoltán, Székesfehérvár, Hosszúesetér u. 17. III/2. 8000.

ZX-Spectrumra 1000 da- rabszámú játékprogram-kész- letemet cserélem folyamán sze- retném bővíteni. Válaszokat a programok listájával ké- rek: Csah Attila, Debrecen, Kapitány út 16. 4033.
ZX-Spectrum 48 K számí- tógépre játékprogramot és se- gédprogramokat cserélek. Somló Ferenc, Tatabánya, Mártírok u. 68. III/1. 2600.

ZX-Spectrumhoz eladó 16 K-s ROM. Új BASIC parancsok (renum, hexaszám, hexadump, onerror, delline, variables, meleg reset). EPROM-okat be- égetem. Puzsár István, Bu- dapest, Zrínyi 125/A 1196.

48 K-s ZX-Spectrum-ra játékprogramokat cserélek. Farkas Tibor, Sarkad, Tán- csics u. 1. 5720.
**C-64-es játékprogramo- kat cserélek, magán hasz- nálókkal Varga Károly, Ka- pupvár, Lenin u. 11. 9330.
 Commodore 64-es számí- tógép, adat rögzítő magnóval, a teljes Data Becker könyv- sorozattal, programokkal e- gyütt eladó 25 000 Ft-ért.** Sári Gábor, Szolnok, Rózsa Ferenc út 6. 5000.
Commodore 64-es számí- tógépre játékprogramokat cse- rélnék. Válaszokat a progra- mok listájával kérek: Turó- czy Péter, Nyíregyháza, Ko- ranyi Frigyes u. 74. fsz. I. 4400.

C-64-re vagy Spectrumra írt horozkópprogramot ke- resek. Cseréle programokat tu- dok küldeni, esetleg meg- venném. Nagyon sürgős! Tel.: 84/II-494

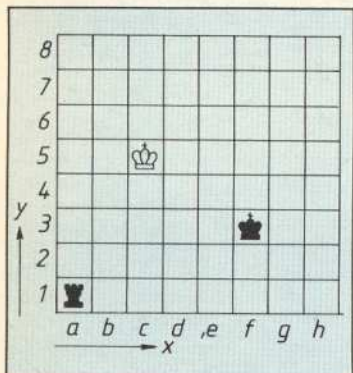
16 K RAM ZX81-hez bőví- tő olcsón eladó. Alig használt játékpogramot is tudok ad- ni. Sörös Róbert, Budapest, XI. Bartók Béla u. 17. 1114.
VC 20-hoz 16 K RAM cart- ridg eladó. C-64-es játék- programokat cserélek kez- tetlen. Kiss János, Hőmezővá- sarhely, Árpád u. 2. IV/2. 6800.
Casio PB-200 -as géphez keresek FA-3 típusú inter- fészet. Beregszászi Attila, Mátészalka, Alkotmány u 13. 4700.

A 64'er újság olvasóival szeretném a kapcsolatot fel- venni, programcsere céljáb- ol. Horváth-Papp Imre, De- brecen, Gőrgy u. 4. II/19. 4032.
PRIMO tulajdonosokat ke- resek programcsere céljából, számítógépekkel kapcso- latos könyveket eladok. Kívánságra részletes listát küldök, vá- laszboríték ellenében. Vár- sányi Gábor, Kiskőrös, Árpád u. 12. 6200.
EPROM-programozó készü- lékhez 28 lábás IC-foglala- tok eladók (390 Ft/db). Csá- nyi Gábor, Mezőtér, Ifjúsági lkt. 13. III/16. 5400.
TV-COMPUTER és ZX81 programokat cserélek. Meny- hárt Tibor Debrecen, Gyepü sor, 12. 4033.
LASER 2001-es géppel, valamint német nyelvű le- fással rendelkezem. Keresek hasonló géptípussal rende- lőketek tapasztalatcsere céljából. Cím: Kerekes Mi- hály, Keszthely, Zalika Máté út 19/A/2/3, 8360, tel.: 12-463

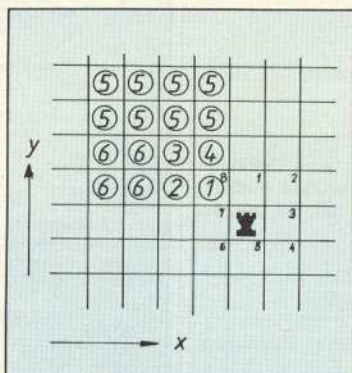
bástya: al, el

pontosabban reméljük, önök olvasták a „Végjáték” nevű program fejlesztésének az előzőekben megkezdett bemutatását (lásd az 1986/11-12. számban). Most ezt folytatjuk. Röviden emlékeztetbe idézzük: a gép királlyal és bástyával játszik, a gép kezelője — az ellenfél — ellen. A kiinduló állás az 1. ábrán látható.

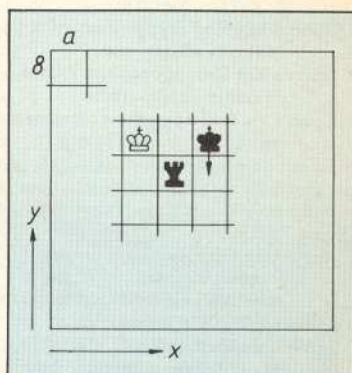
A program megírásához a Pascal nyelvet választottuk. Az előző cikkben röviden bemutattuk a főprogram vázát (vezérlési struktúráját), és adatábrázolást és az ellenfél lépését ellenőrző-elfogadó procedúrát.



1. ábra. A kiinduló állás



2. ábra. A két király relatív helyzete a bástyához képest, ahogy a program látja. A bekarikázott számok az ellenfél királyának, a be nem karikázottak a gép királyának helyzetét jelölik a bástyához képest



3. ábra. Az „12” relatív helyzet, melyben a gép a királyával „y” irányban hátra lép

Fontolgattuk, milyen „stratégiát” válasszunk a gép soron következő lépésének meghatározásához (ez a program leglényegesebb része), és három lehetőség közül — tudva, hogy még sok más, esetleg sokkal jobb megközelítés is létezhet — kikötöttünk amellett, hogy az állásokat a gép szempontjából a két király bástyához viszonyított helyzetére alapján különböztetjük meg (lásd a 2. ábrát).

Utaltunk arra, hogy így majd $6 \cdot 8 = 48$ -féle megkülönböztetett álláshoz be kell programozni a megfelelő lépést. (Mivel a szerző nem sakkjátékos, ezért az olvasóra bízta, hogy a bemutatott programozási megközelítés alapján a saját maga által „legmegfelelőbbnek” tartott lépéssel helyettesítse az itt közölteket.)

Most tehát a lépésen nevével a procedúra (szubrutin) bemutatása következik, melyben a gép lépéseit úgy határozzuk meg, hogy a gép a bástyával és a királlyal fokozatosan az a8 sarkokba szorítja az ellenfél királyát, és ott tartat ad neki. (Ez jogosan vitatható egyszerűsítés: egy nagyobb, bonyolultabb programmal az ellenfél lépéseitől függetlenül lehetne értékelni, hogy melyik lehet a legmegfelelőbb sark.)

A gép lépéseinek megválasztása szempontjából a „parti”-t három részre lehet osztani.

Az első részben a gép a bástyával meglehetősen mechanikusan a saját királya mellé (védelmébe) lép. A gép első lépése — valójában a „parti” 2. lépése, mert először az ellenfél léphet — mindig ez:

A második — a parti 4. — lépésében a gép mindig a bástyával lép előre úgy, hogy

- királya védelmébe kerüljön,
- az ellenfél királya a bástya és a tábla 8. sora között maradjon és
- a bástya a fenti két követelmény betartása mellett annyit haladjon előre y irányban, amennyit csak lehet.

A második részben — mint már szó volt róla — a gép a két király bástyához viszonyított relatív helyzete alapján $6 \cdot 8 = 48$ állást különböztet meg, és az ezekhez az állásokhoz „beprogramozott” lépést teszi meg, fokozatosan az a8 sarkok felé szorítva az ellenfelet. A lépésen procedúra hívja az ezt megvalósító lépgen2 procedúrát.

A harmadik részben az ellenfél mozgásterét már annyira lecsökkent, hogy nem használható tovább a lépgen2, mert egyszerűen figyelni kell, nehogy pathhetely alakuljon ki, másrészt, hogy esetleg márt lehet adni.

A parti harmadik részében a lépésen procedúra hívja a lépgen3 procedúrát.

Most nézzük meg, hogy alakul ez Pascalban (vázlatosan):
PROCEDURE lépésen;

```

IF l < 5 THEN BEGIN (* Ha az első részben vagyunk *)
  IF l = 2 THEN (* Ha a gép első lépése következik *)
    állás [saját bástya, x] := PRED(állás[saját király, x]);
  IF l = 4 THEN (* Ha a gép második lépése következik *)
    állás [saját bástya, y] := PRED(állás[ellenkirály, y]);
  IF állás [saját bástya, y] > '4' THEN
    (* Ha kilépett volna a király védelméből, akkor: *)
    állás [saját bástya, y] := '4';
  (* Akkor visszahúztuk a védelmébe *)
  (* Ide jön a lépés kijelzése a képernyőre *)

```



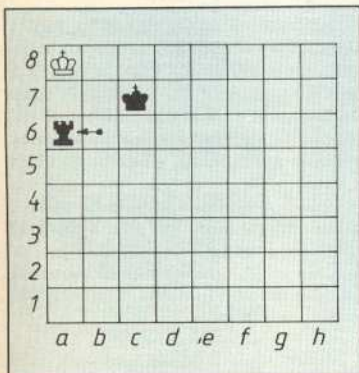
```

END (* Első rész *)
ELSE (* Akkor a második vagy a harmadik részben
vagyunk *)
IF második rész THEN lépgen2 (* Külön procedúra *)
(* Ha nem a második, akkor a harmadik részben vagyunk *)
ELSE lépgen3 (* Külön procedúra *)

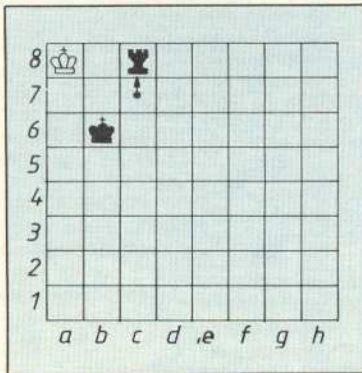
l:=l+1; (* Lépésszámláló *)
END; (* Lépésen procedúra)
    
```

```

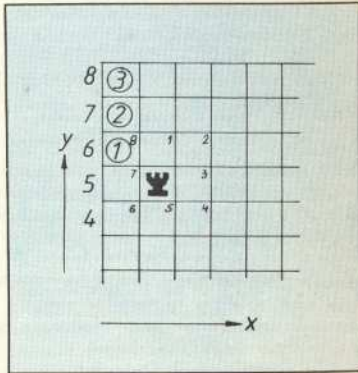
(*13*) IF((xs=SUCC(xb)) AND (ys=yb)) THEN
ys:=SUCC(ys) ELSE BEGIN
.
.
.
END END END END END
END (*1*) ELSE BEGIN
(*2*)
(* És így tovább *)
    
```



4. ábra. Az egyik matthelyzet



5. ábra. A másik matthelyzet



6. ábra. Az állások egyik javasolt megkülönböztetése a „parti” harmadik részében
A bekerikázott számok az ellenfél királyának, a be nem karikázottak a gép királyának helyzetét jelölik a bástyához képest

(* Megjegyzés:
VAR második rész: BOOLEAN;

második rész:=(xb>'b') AND (yb<'7')) *)

Mint említettük, kritizálható a gép első két lépése: lehetne a bástya veszélyeztetése nélkül olyan lépéseket is generálni, melyek az ellenfél mozgásterét már az első lépésekben jobban szűkíthetnék – így a mattig vezető lépések száma csökkenthető lenne. A program felépítése szinte kínálja az utólagos finomítást. Ezt az olvasóira bizzuk.

Most foglalkozunk a lépéseket a parti második részében (a gép harmadik lépésétől) a parti harmadik részének eléréseig generáló lépgen2 procedúrával.

```

PROCEDURE lépgen2;
VAR xc,xs,xb:CHAR;
ye,ys,yb:CHAR;
BEGIN xc:=állás[ellenkirály, x];ye:=állás[ellenkirály, y];
xs:=állás[saját király, x];ys:=állás[saját király, y];
xb:=állás[saját bástya, x];yb:=állás[saját bástya, y];
(* Az xc,ye,xs,ys,xb és yb segédváltozók feltöltése megtörtént *)
(* 0 a "K<>" eset kezelése. Lásd később! *)
(*1*) (* Az ellenkirály 1-es relatív pozíciója esetén *)
IF (xc = PRED(xb) AND (ye = SUCC(yb))) THEN BEGIN
(*11*) (* A saját király nem lehet az 1-es relatív helyzetben,
mert az ellenkirály ütné: kizárt eset *)
(*12*) IF((xs=SUCC(xb)) AND (ys = SUCC(yb))) THEN
ys:=PRED(ys) ELSE BEGIN
    
```

Kicsit másképpen fogalmazva tehát: hogy a lépgen2 az általa elővileg megkülönböztetett 6*8=48 fajta állásban hogyan lépjen, így oldja meg:

```

(*0*) A „K<>0” eset kezelése. Lásd később!)
.
.
.
(*1*)
IF < az ellenkirály az 1-es pozícióban van a bástyához képest >
THEN
(*11*) (* Kizárt eset, mert ha a saját király is az első pozícióban
lenne, akkor a két király ütné egymást *)
(*12*) IF < a saját király a 2-es pozícióban van > THEN
    
```

a saját királlyal lépünk egyet „y” irányban vissza

ELSE BEGIN

A $6 \cdot 8 = 48$ eset közül összesen 7 esetben nem tudott a szerző ellenállni a kísértésnek, és a bástyával úgy lépett a gép nevében, hogy az „leszakadt” a királyától. (Lásd a később részletesen bemutatott lépgen2-ben a 34., 54., 55., 56., 62., 63., 64. állást.) A részletesen kidolgozott lépgen2 ezt a következő lépésnél figyelembe veszi (0-ás eset!), és a királlyal a bástya után lép.

($K=1$, $K=2$ és $K=3$ szerint háromfajta korrekciós lépést tehet ilyenkor.)

Az „IF THEN . . . ELSE” konstrukció egyszerűvé teszi, hogy ha a gép egy állásra ráismert, akkor a többi már ne vizsgálja. Ahogy a gép a keresésben előre halad, a program ugrásmélysége nő. Egyes compilerek a hatékonyság érdekében az ugrásmélységet beállítják valamilyen előzetesen maximált értékre. Ezt túllépni csak akkor tudjuk, ha a megfelelő helyen *direktívát* adunk a compilernek, hogy az adott környezetben engedje az ugrásmélység túllépését. A szerzőnek — az általa használt compilernél — ezt meg kellett tennie a „(* SJ ++ *)” furcsa kommentár formájában. Az adott compiler ebből megérti a kérést, hogy engedje az előzetesen beállított (default) ugrásmélység megnövelését további két fokozattal.

Most pihenésképpen hagyjuk el egy rövid időre a lépgen2-t, és vizsgáljunk meg egy olyan triviális részletet, hogy miként „veszi be” a gép az ellenféltől a következő lépést jelentő koordinátapárt, azaz a két karaktert:

PROCEDURE kérdez; (* Az ellenfelet kérdezi a gép a következő lépésről vagy arról, hogy feladja-e a „partit” *)

VAR xbe, ybe: CHAR;

megfelel: BOOLEAN;

BEGIN

REPEAT (* Amíg a beadott lépés meg nem felel *)

megfelel := FALSE;

WRITELN ('Kérem, lépjen a királlyal, vagy küldjön egy „v” betűt, ha fel kívánja adni.')

WHILE NOT EOLN DO GET (INPUT);

(* Az előző sorra „technikai okokból” van szükség *)

GET (INPUT); xbe := INPUT↑;

GET (INPUT); ybe := INPUT↑;

IF ((xbe = 'v') OR (xbe = 'V'))

THEN BEGIN felad := TRUE;

megfelel := TRUE END

ELSE felad := FALSE;

IF NOT felad THEN BEGIN

megfelel := FALSE;

ellenőriz (xbe, ybe, megfelel);

(* Ez az előző cikkben bemutatott procedúra *)

IF NOT megfelel THEN <kijelzi ezt az ellenfélnek >

END (* nem adta fel *)

UNTIL megfelel; (* Azaz, amíg az ellenfél lépése meg nem felel *)

IF NOT felad THEN BEGIN

<kijelzi az ellenfélnek > ;

l := l + 1; (* lépésszámláló *)

END ELSE

WRITELN ('Tehát ön feladta a partit. Köszönöm a játékot.')

END; (* A „Kérdez” procedúra vége *)

Az ellenfél így a lépését az alábbi módon adhatja be:

(xbe) (ybe) (adatátvitel)

vagy — ha feladja a partit:

(v) (adatátvitel)

Az ellenfél a kis „v” helyett megnyomhatja a nagy „V” billentyűt is. Az „adatátviteli” billentyű gépenként, terminálonként más és más lehet (lehet például egy „RETURN” feliratú billentyű is).

Általános tapasztalat, hogy a Pascal nyelv READLN (xbe) beépített procedúrájával „óvatosan” kell bánni. A bemutatott megoldásra szeretnénk felhívni a kezdő Pascalozók figyelmét, sok felesleges próbálgatást lehet vele megtakarítani.

A „Kérdez” procedúrában xbe és ybe típusát célszerűtlen lett volna így deklarálni:

VAR xbe: 'a'..'h';

ybe: '1'..'8';

Ilyen megoldás mellett ugyanis, ha az ellenfél melléutne, tehát nem a megfelelő billentyűt nyomná meg a klaviatúrán, akkor a program futását biztosító *run-time-system* megszakítaná a futást és hibát jelezne. Ebből a helyzetből csak a program újbóli indításával lenne kiút.

A „VAR xbe,ybe:CHAR;” deklaráció mellett viszont az ellenfél melléütéseit az előző cikkben bemutatott „ellenőriz” procedúra anélkül lekezelné, hogy a program futását meg kellene szakítani. (Megjegyezzük, hogy hasonló problémáink lettek volna „VAR xbe: 'a'..'h'; ybe: INTEGER;” deklaráció esetén is.)

Most pedig lássuk részletesen a lépgen2 procedúrát (lásd a listát).

A *harmadik* rész akkor kezdődik, ha az

(xb = 'b') OR (yb = '7')

feltétel teljesül.

A matthelyzet kétféle lehet (4. és 5. ábra).

Visszafelé indulva, a matthelyzetből könnyen rájöhettünk arra, hogy a matt eléréséhez elsősorban arra kell vigyáznunk, hogy királyunk (a gép királya) „jó ütemben” kísérje „kivülről” a bástyát. Ha nem kísérmé jó ütemben, akkor a gép a király egy oldallépésével korrigálja az ütemet, hogy azután biztos lépésekben haladjon az elkerülhetetlen matt felé, melyet majd a felhasználónak is jelezni kell.

A harmadik részben működésbe lépő *lépgen3* procedúra felépítése, szerkezete a szerző elképzelése szerint lényegében ugyanolyan lehet, mint a lépgen2-é, csak a figyelt állásokat célszerű egy kicsit módosítva megválasztani — például a 6. ábra szerinti módon.

A szerző bevallja, hogy nincs maradéktalanul megelégedve a bemutatott, illetve eltervezett megoldással. Úgy értékeli, hogy az általa „beprogramozott” stratégia *nem elég intelligens*, túlságosan direkt: ugyanis állásonként megmondja a gépnek, hogy pontosan mit lépjen. Valószínűleg azonban egy ilyen egyszerű végjátéknál módszere nem marasztható el teljesen, bár csak egy kicsit is bonyolultabb helyzetben ez az út már nem lenne járható: túl sok lenne a közvetlenül megvizsgálandó eset. Az állások finomabb megkülönböztetésével, a bástya merészebb mozgásával még ez a stratégia is javítható. Nagyon örülnék, ha az olvasók törnék a fejüket, hogyan lehetne az itt közöltnél egyszerűbb, *intelligensebb* és jobb megoldásokat találni.

RAINBOW a μ M-ban

Amint korábban már jeleztük, módunkban áll cikkeket, programleírásokat átvenni az amerikai RAINBOW számítógép magazintól. A magazin teljes egészében a TANDY RADIO SHACK Color Computer (az USA-ban népszerűvé vált nevén: CoCo) számára írt havilap.

A mi mércénkkel mérve szokatlan, hogy egyetlen géptípusnak egy egész havilapot szenteljenek. Pedig a RAINBOW a maga kb. 300 oldalával nem is az egyetlen, kizárólag a CoCo-val foglalkozó újság. Legalább további öt önálló CoCo-lap létezik még, mi legalábbis ennyit ismerünk: Color Computer Magazine, Color Computer News, Micro Color Journal, CoCo Ads, Spectrogram. Ismerünk továbbá olyan folyóiratokat, amelyek nagyobb terjedelmet biztosítanak a CoCo-nak a lapon belül, mint a mi magazinunk a C64-nek vagy a Sinclair gépeknek. Közéjük tartozik a 80 Micro; CoCo-val foglalkozó szekciója a „HOT CoCo”, amelyet 1983-tól 1986 februárjáig azonos címmel önálló kiadványként jelentetett meg a kiadó.

Az elmondottak alapján a gépnek rendkívül népszerűnek kell lennie az USA-ban. Erre utal az is, hogy több mint 1 millió darabot adtak el belőle. Nyilvánvaló ezek után, hogy megszámlálhatatlan hardver- és szoftverház lát üzletet a fejlesztésben a CoCo számára. Példaként említjük a kaliforniai Microworks által kifejlesztett képdigitalizátort, amellyel másodpercenként nyolc kép állítható elő 256x256 bontásban. A szoftverfejlesztések közül kiemelkedő az OS-9 operációs rendszer (Microware), amely a mini- és nagygépeken futó UNIX

rendszer adaptációja 8 bites rendszerre. (A CoCo processzora valójában nem igazán 8 bites: számos 16 bites utasítást tartalmaz.)

A CoCo-t 1980 óta forgalmazzák. Hazánkban nem ismerik, noha a nálunk népszerű, azonos árkeretű típusok egyike sem tud összességében ennél a gépnél többet.

A CoCo-tól kapható többlet csak részben köszönhető a processzornak (6809E). A felhasználó igen könnyen kezelhető, nagy határfokú és sebességű BASIC-et kap kézhez; a BASIC interpretert a Microsoft cég készítette. Bár az 1980-ban készült változat egy-két módosításon átesett azóta, a változtatásokat a kompatibilitási problémákat kizáróan oldották meg. Az interpretert egyébként sok hasonlóságot mutat az IBM PC BASIC-jéhez.

A RAINBOW cikkeiből válogatva természetesen olyan cikkeket megjelentetésére törekszünk, amelyek közérdeklődésre tarthatnak számot. Az elméleti cikkeket feltétlenül ilyenek, szándékunkban áll azonban programokat is közzétenni. A programok valamilyen mértékben nyilván gépfüggőek. Igyekezünk azonban majd a programokat oly mértékben leegyszerűsíteni, továbbá olyan magyarázatot adni, hogy — kivált a MICROSOFT BASIC ismerőinek — ne jelenszögöt a programok adaptálása.

„STYLIST” — Epson-nyomatók kezelője

Reprinted from the Rainbow, The Color Computer Monthly Magazine. Copyright Falsoft, Inc. 1984. P. O. Box 385, Prospect, KY 40 059

Elsőként az Epson- vagy Epson-kompatibilis nyomtatók birtokosainak adunk egy igen jól használható programot, amellyel az Epson valamennyi betűképe könnyen előállítható, szükségeltelne téve a nyomtatói formákhoz tartozó kódok örökös keresgélését. A programot a nem Epson- vagy Epson-kompatibilis nyomtatóval dolgozó olvasóink is kipróbálhatják a kódok átírásával a saját nyomtatójukhoz specifikált kódokra.

A program menüre szervezett. A betűkép a megfelelő billentyű lenyomásával egyszerűen változtatható. A meghívott kód mind addig érvényben marad, amíg a nyomtatót ki nem kapcsoljuk, vagy a program azt meg nem változtatja. A program szövegszerkesztővel együtt is, de írógépként is működhet.

Ahhoz, hogy a kódokat bármikor megváltoztathassuk, a nyomtatónak állandóan fogadóképes állapotban kell lennie. Ezért a program is állandóan figyelni a nyomtató állapotát, és jelzi, ha a printer esetleg nem fogadóképes. A CoCo esetében a \$FF20 (65314)-en található adat jelzi a nyomtató fogadóképeségét (a 200—210-es és a 710—720-as sorokban).

A program kezelési útmutatót nem igényel, a menü mindenről tájékoztat. Rendkívül egyszerű működését az alábbiakban ismertetjük.

A program két képernyőt igényel. Az első a „Print Stylist” menü jelenik meg; itt lehet a megfelelő betűváltozatot kijelölni. A kínálat a következő: pica és elit mint betűtípus, továbbá kurziv, kiemelt, vékonyított, széthúzott, nyomott, kétszeresen kicsinyített és aláhúzott forma. Ha a kívánt betűtíphez tartozó billentyűt lenyomjuk, a megfelelő menüsorban megjelenik egy *** — visszajelezővén a választást. Ugyanannak a billentyűnek az ismételt lenyomására a *** jel eltűnik. Az ENTER (más géptípusokon RETURN, END LINE) gomb érintése szükségtelen.

A nyomtatókódok táblázata

Az Epson nyomtató kódja	Magyarázat
CHRS(27)“2”	Sortávolság: 6LPI (sor/inch)
CHRS(12)	Lapozás
CHRS(27)CHRS(64)	Normál üzem
CHRS(27)“A”CHRS(0)	Sortávolság: 0
CHRS(27)“P”	Betűtípus: pica (elite helyett)
CHRS(27)“M”	Betűtípus: elite (pica helyett)
CHRS(27)“4”	Betűkép: italic (kurziv)
CHRS(27)“5”	Italic ki
CHRS(27)“W1”	Betűkép: expanded (széthúzott)
CHRS(27)“WO”	Expanded ki
CHRS(15)	Betűkép: compressed (nyomott)
CHRS(18)	Compressed ki
CHRS(27)“S1”	Betűkép: subscript (vékonyított)
CHRS(27)“T”	Subscript ki
CHRS(27)“E”	Betűkép: emphasized (kiemelt)
CHRS(27)“F”	Emphasized ki
CHRS(27)“G”	Betűkép: double strike (kétszeresen fedett)
CHRS(27)“H”	Double strike ki
CHRS(27)“-1”	Betűkép: underlined (aláhúzott)
CHRS(27)“-0”	Underlined ki

```

100 REM *****
110 REM *
120 REM * S T Y L I S T *
130 REM * FOR EPSON PRINTERS *
140 REM *
150 REM * THE RAINBOW MAY 1986 *
160 REM *
170 REM *****
180 A$="PRINT STYLIST SELECTION MENU";D$="2
185 TR$="";FOR I=1 TO 90:TR$=TR$+CHR$(172);NEXT I:TR$=TR$+***** END OF
190 CLS:
200 PE=PEEK(65314):AND 1
210 IF PE=0 THEN 230
220 PRINT#194,"** PLEASE TURN PRINTER ON **";GOTO 200
230 X=I*POKE 150,I:CLS
240 PX$="PRINTER IS NOW SET FOR THIS PRINT STYLE"
250 A1$="":B$="9600":BCT=5
260 GOTO 1110
270 REM - LINE INPUT ROUTINE
280 CLS
290 PRINT" LINE INPUT SCREEN"
300 PRINT#32,"1...5...10...15...20...25...30"
310 PRINT#32,"35...40...45...50...55...60..."
320 PRINT#160,"65...70...75...80"
330 PRINT#192,"-----"
340 PRINT#320,"-----CHARACTERS-----"
350 PRINT#648,"ENTER 'I' FOR PRINT STYLIST MENU":
390 PRINT#224,TR$
430 PRINT#224,":LINE INPUT X#
440 IF X#="I" THEN 490
450 PRINT#0,CHR$(27);"2";X#
480 GOTO 340
490 CLS
500 REM - PRINTER DRIVER ROUTINE
510 IF S#1 THEN 1070
520 IF X=32 THEN PRINT#0,A#
530 PRINT#32,A1$;"A"=PICA
540 PRINT#32,A2$;"L"=ELITE
550 PRINT#32,A3$;"I"=ITALIC
560 PRINT#32,A4$;"X"=EXPANDED

```

```

570 PRINT A$;"C=" COMPRESSED'
580 PRINT A$;"S = SUBSCRIPTED'
590 PRINT A$;"E = EMPHASIZED'
600 PRINT A$;"D = DOUBLE-STRIKE'
610 PRINT A$;"U = UNDERLINE'
620 PRINT' R = RESET ALL PRINTER CODES'
630 PRINT' B = BAUD RATE - ,:B#
640 PRINT' P = PRINT STYLE'
645 PRINT' ? = DEMO PROGRAM'
650 PRINT' I = SINGLE-LINE TYPEWRITER'
660 PRINT' Q = QUIT';
670 ON=INKEY#
680 IF X=32 THEN 710
690 PRINT#31-X,LEFT$(A$,X)
700 X=X+1
710 PE=PEEK(65314) AND 1
720 IF PE=0 THEN 780
730 CLS
740 PRINT#195,** PRINTER IS OFF LINE **
750 'PE=PEEK(65314) AND 1
760 IF PE=0 THEN 490
770 GOTO 750
780 IF ON="Q" THEN 1500
790 IF ON="I" THEN 280
800 IF ON="P" THEN 940
810 IF ON="R" THEN 1110
820 IF ON="A" THEN 1140
830 IF ON="L" THEN 1160
840 IF ON="1" THEN 1180
850 IF ON="S" THEN 1210
860 IF ON="C" THEN 1240
870 IF ON="S" THEN 1270
880 IF ON="E" THEN 1300
890 IF ON="D" THEN 1330
900 IF ON="U" THEN 1360
910 IF ON="B" THEN 1390
920 IF ON="?" THEN 960
930 GOTO 670
940 PRINT#DV,CHR$(27);"2";PX#
950 GOTO 500
960 SW=1:PX#="PRINT SYLIST DEMONSTRATION"
970 N=4
980 FOR XY=1 TO 60
990 RN=RND(N)
1000 IF XY=1 THEN RN=3

```

```

1010 IF XY=2 THEN RN=4
1020 IF XY=19 AND A$="S" THEN RN=4
1030 IF XY=20 THEN N=3:RN=5
1040 IF XY=35 THEN N=3:RN=8
1050 IF XY=45 THEN N=7:RN=5
1060 ON RN GOTO 1180,1210,1270,1240,1360,1300,1330,1140,1160
1070 PRINT#DV,CHR$(27);"2";PX#
1080 NEXT XY
1090 PRINT#DV,CHR$(12)
1100 SW=0:PX#="PRINT IS NOW SET FOR THIS PRINT STYLE"
1110 GOSUB 1510:IF A$="":A2#="":A3#="":A4#="":A5#="":A6#="":A7
:A8#="":A9#="
1120 PRINT#DV,CHR$(27);CHR$(64);CHR$(27);"A";CHR$(0)
1130 GOTO 500
1140 GOSUB 1510:IF A1#="*" THEN 500
1150 A1#="*":PRINT#DV,CHR$(27);"P";A2#="*":GOTO 500
1160 GOSUB 1510:IF A2#="*" THEN 500
1170 A1#="*":PRINT#DV,CHR$(27);"M";A2#="*":GOTO 500
1180 GOSUB 1510:IF A3#="*" THEN 500
1190 A3#="*":PRINT#DV,CHR$(27);"A";GOTO 500
1200 A3#="*":PRINT#DV,CHR$(27);"S";GOTO 500
1210 GOSUB 1510:IF A4#="*" THEN 1230
1220 A4#="*":PRINT#DV,CHR$(27);"U";GOTO 500
1230 A4#="*":PRINT#DV,CHR$(27);"D";GOTO 500
1240 GOSUB 1510:IF A5#="*" THEN 1260
1250 A5#="*":PRINT#DV,CHR$(15);GOTO 500
1260 A5#="*":PRINT#DV,CHR$(19);GOTO 500
1270 GOSUB 1510:IF A6#="*" THEN 1290
1280 A6#="*":PRINT#DV,CHR$(27);"S1";GOTO 500
1290 A6#="*":PRINT#DV,CHR$(27);"T";GOTO 500
1300 GOSUB 1510:IF A7#="*" THEN 1320
1310 A7#="*":PRINT#DV,CHR$(27);"E";GOTO 500
1320 A7#="*":PRINT#DV,CHR$(27);"F";GOTO 500
1330 GOSUB 1510:IF A8#="*" THEN 1350
1340 A8#="*":PRINT#DV,CHR$(27);"G";GOTO 500
1350 A8#="*":PRINT#DV,CHR$(27);"H";GOTO 500
1360 GOSUB 1510:IF A9#="*" THEN 1380
1370 A9#="*":PRINT#DV,CHR$(27);"1";GOTO 500
1380 A9#="*":PRINT#DV,CHR$(27);"-";GOTO 500
1390 GOSUB 1510:BCT=BCT+1:IF BCT=7 THEN BCT=1
1400 ON BCT GOTO 1450,1460,1470,1480,1440
1440 B#="9600":POKE 150,1:GOTO 500
1450 B#="6000":POKE 150,18:GOTO 500
1460 B#="1200":POKE 150,41:GOTO 500
1470 B#="2400":POKE 150,18:GOTO 500
1480 B#="4800":POKE 150,7:GOTO 500
1500 PRINT#DV,CHR$(27);"2";CLS:END
1510 PRINT#DV,CHR$(27);"A";CHR$(0)
1520 RETURN

```

Különböző kombinációk alakulnak több lehetőségű egyidejű érvényesítésével. Bármi kor ellenőrizhető a gép által, hogy milyen iráskepet kértünk valójában: ehhez nyomjuk le a "P" billentyűt ("Print Option") rövidítésekként a "P" billentyű kijelölve); hatására egysornyi szöveg (PRINTER IS NOW SET FOR THIS PRINT STYLE) jelenik meg. A baud értékének beállítását a nyomtató által megkövetelt értékre a "B" billentyű lenyomásával érhetjük el. Az "R" billentyűvel a nyomtató alapállapotba kerül (RESET). Ezt a két billentyűt is, a "P" billentyűhöz hasonlóan, a hozzájuk rendelt funkció angol neve kezdőbetűinek megfelelően jelölték ki, és természetesen e funkciók esetében is kijelölhető bármely más billentyű. Gépfüggő viszont maga az átviteli sebességet beállító rutin (1390–1480-as sor). Az 1440–1480-as sorokban a B\$ sztringből kiolvasható a beállítható baud-sebesség; ténylegesen a gép felőli oldalon a 150. (decimális) memóriarekesz átírásával lehet beállítani. A rutin más géptípusokhoz könnyen átírható, de párhuzamos kimenet esetén az egész rutin felesleges.

A menü megjelenésekor azonnal szabad választani, noha a képernyő még nem teljes: a menünek még nincs címe. A cím ui. ("PRINT SYLIST SELECTION MENU"); lásd 180-as sor) csak most kezd jobbról balra „bejagolgolni” az ernyőre (lásd 690–700-as sorok!). Ennek a megoldásnak természetesen nincs a program futása szempontjából jelentősége; ki is hagyható – pusztán látványosság. A "Q" billentyű lenyomásával a programból kiszállunk, visszatérünk a BASIC-hez. A programban kiválasztott kód a nyomtató számára továbbra is érvényes. Ha most betöltünk egy

szövegszerkesztőt — vagy más, a nyomtatót kezelő programot —, printerünk a kiválasztott variációval fog dolgozni.

A másik képernyő egysornos írógépként működik. A program soronként 80 karakter nyomtatását tételezi fel, a CoCo alfanumerikus képernyője (ahol jelen esetben a program kezelhető) viszont 16 sor, 32 oszlop formátumú. A 80 karakter hosszúságú sor tehát a képernyőn három sort igényel. A problémát áthidalandó a program 300–320-as sorai az oszlopokat beszámozzák. A gépelt szöveg az ernyő közepére kerül egy grafikus karakterekkel kijelölt területre, a számolásnak megfelelően pozícionálva. A szöveg LINE INPUT utasítással a billentyűkről a memóriába jut. Ez az utasítás abban különbözik az INPUT-tól, hogy aktivizálásakor nem jelenik meg kérdőjel az ernyőn, továbbá csak sztringet kezel, viszont bármilyen karaktert elfogad, még idézőjelet is. Ha nem gépeltet be semmit, csak megnyomjuk az ENTER gombot, soremelés és kocsis vissza a következőmeny.

Ez az egysornos írógép alkalmas nyomtatványok kitöltésére is. Ha például egy nyomtatványon egy bekeretezett részt kell kitölteni, amely az 50. oszlopban kezdődik, 50 szünet után gépeljük be a kívánt szöveget.

A "!" billentyű lenyomásával visszatérhetünk a menühöz, ahogy erről a képernyőn állandó tájékoztatást ad. A "?" gombbal a programnak a demonstrációs szakaszát aktivizálhatjuk. Egy teljes oldal jelenik meg különböző betűvariánsokkal. A szöveg minden sorban "PRINT SYLIST DEMONSTRATION" lesz, a betűformák viszont találomra és soronként váltakoznak.

A készletből való kiválasztás a véletlenszám-generátorra van bízva, így lehetnek olyan változatok, amelyek egy gépelt oldalon többször is megjelennek, mások viszont egyáltalán nem. Ha több oldalt gépeltünk, az eredmény mindig más és más lesz.

A CoCo véletlenszám-generátora a következőképpen működik. Ha $A = \theta$, a generált szám $0 < n < 1$; ha $A > \theta$, a generált szám $1 \leq n \leq A$; ha $A < \theta$, a véletlenszám-generátor új alapról indul. A syntax: RND(A).

A program a felesleges részek, például a demonstrációs rész elhagyásával szubrutin-ként is csatlakozhat más programokhoz, ahol ilyen jellegű printerkezelésre van szükség. Ha a szubrutin-kénti alkalmazás a cél, és az adott géptípus BASIC-jében nincs újraszámozó utasítás, célszerű az egyes sorokat ennek figyelembevételével számozni.

Néhány további útmutató a CoCo BASIC-jének megértéséhez. A PRINT #, a szöveget pozícionálja. A 32 × 16 karakteres alfanumerikus képernyő karakterhelyeit θ –511-ig sorban megszámozták: az n helyére θ –511 közötti számot írhatnak. Az egyes sorok első karaktere: sorszám (θ –15) + 32. Az "a" jel rövidítés; jelentése: at (-nál, -nél).

A PRINT #n utasítással a kimenetet választhatjuk meg a kitérítéshöz. n = θ : képernyőre; n = -1: magnóra; n = 1: lemezre; n = -2: nyomtatóra.

A programban az n helyén a DV változó (device number) áll. Értéke itt konstans: -2. (Lásd a 180-as sorban!)

A nyomtató a svéd ábécét használta, ezért a listában \$ helyett X és @ helyett É jelent meg. VÁGH ISTVÁN

Nos, igen! ZX-SPECTRUM – helyi hálózatban

Ha Spectrummal kapcsolatban helyi hálózatot emlegetünk, nem szabad a nagy gépek 10 Mbit/s vagy még gyorsabb, nagy hatótávolságú, mindenféle modemmel támogatott rendszerére gondolnunk. Az Interface 1 (a későbbiekben If-1) lehetőségei ennek csupán egyfajta szimulációját adják, de sokszor ez is nagyon hasznos. A mindössze 32 kbit/s sebességgel (mikrogepeknél azért ez is elég tekintélyes!) és a legfőbb 5-6 méteres távolsággal is ügyes dolgokat lehet készíteni. Az If-1-be kétféle, egymástól eléggé különböző soros interfészt építettek be. Az egyik a tényleges helyi hálózathoz szükséges LAN (Local Area Network), a másik pedig egy félig-meddig szabályos RS232C.

A Spectrum LAN rendszerbe egyszerre maximum 64 gép léphet. Az adatforgalom általában két, konkrétan kiválasztott gép között megy végbe. Folyton az utasítás kiadása nem jelenti illetlenül annak végrehajtását; a magnót nem érdekli, hogy a gép vételkész-e. Ha elküldtünk egy írással kapcsolatos parancsot, de a célállomás még nem készült fel az adatok fogadására, mindössze azt látjuk, hogy a gép elveszi a vezérlést: a border besötétül. Alapállapotban feketére vált, de ez igény szerint megváltoztatható. Amikor a másik gép kiad egy ránk vonatkozó olvasóutasítást, a border villogni kezd, és az információ átkerül a másik géphez.

Az információcserre másik formájában az egyik gép tölti be a főállomás szerepét, a többi pedig mellékállomás. Folyton a főállomás íróutasítást küld, de az információ célját nem jelöli meg konkrétan, hanem állomásként 0-t ad, és a mellékállomások is a 0-ás géptől várják az adatokat. A főállomás nem várja ki, amíg a többi jelzi vételkésztségét: az utasítás a kiadás után rögtön végre is hajtódik. Az információban érdekelt gépeknek tehát már az adás elkezdése előtt késznek kell lenniük az adatok fogadására.

Egy komolyabb kiépítésű hálózatban gyakran előfordul, hogy egyszerre két vagy több gép is bejelenti igényét ugyanarra az állomásra. Ekkor a kisebb állomászámú gépeknek van prioritása. Ebből is látszik, hogy nem érdemes és nem is szabad azonos állomászámokat használni, mert ebből bizonytalom származhat: általában egyik gép sem kapja meg a vezérlést.

A LAN tehát általános célú, aránylag nagy sebességű kapcsolatot létesít a gépek között. Hibája, hogy az egyik géphez kötött meghajtó vagy más periféria a vezérlő gépe kihagyása nélkül, vagyis közvetlenül nem kezelhető. Bár ez nem is általános LAN funkció, de mivel majdnem annyit érne,

mint az egész együtt, és mert egyhamar nem várható hardverfejlesztés Amstardéknál, „boldogult” Sinclairék – még annak idején – tehettek volna az If-1-be egy ilyen célú vezérlőáramkört...

A másik interfész, az RS232C használhatóságában elmarad a LAN lehetőségeitől. Az, hogy nem szabványos, már kizárja az ilyen interfészt használó perifériák többségét: esetleg néhány rendszer és a kevés szabályellenes nyomtató jöhet szóba. Két gép között megvalósulhat információs vonal is, de ez sokkal lassabb és kényelmetlenebb, mint a LAN.

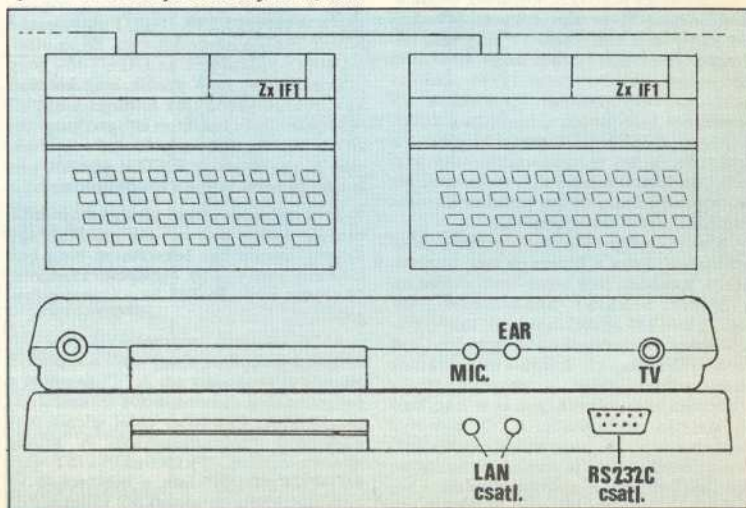
Ezek után nézzük meg utasításait! Mivel a LAN rendszerben 64 gép szerepelhet, a kavarodások elkerülésére mindegyikhez hozzá kell rendelni egy állomászámot: ez 1-64-ig terjedhet. A későbbiekben tárgyalat adatsere-forgalomban majd mindig ezekre fogunk hivatkozni. Bekapcsoláskor a rendszerben egyetlen gépnek sincs állomáskáma (tulajdonképpen még If-1-e sincs), de ha az If-1-et valamilyen tekintetben érintő utasítást adunk ki, tehát a külső ROM-ot felélesztjük, a gépünk száma 1. lesz. Ha eredetileg nem az 1-es állomászám volt bevezetve, akkor ezt az előzőekben leírtak miatt érdemes megváltoztatni. Az azonosító szám beállítását a FORMAT utasítással hajtjuk végre. Ennek szintaktikája: FORMAT "n"; a. Itt az "n" jelenti a LAN (Network) csatornakódját és az állomászámot. A pontosvessző helyett If-1 utasításoknál

megfelel a vessző is. Sajnos a saját gépünk számát kulcsszóval nem tudjuk lekérdézni, így ha a számot elfelejtettük, akkor a 23749-es címen lévő NET STATION rendszerváltozót kell kiolvasnunk.

A LAN funkciók közül először nézzük a legáltalánosabbat; a programok átvételét. A funkció nagyon hasonlít a kazettás egyseggel kapcsolatos utasításokhoz. Amit megtehetünk a magnónál, itt is szabad. A LOAD, SAVE, MERGE, VERIFY kulcsszavak után csillagot kell tennünk, majd "n", vessző vagy pontosvessző, valamint állomászám következik. Ezek után minden magnóutasítást használhatunk: CODE, SCREEN\$, LINE, továbbá az adathalmazra vonatkozó utasításokat mint DATA a() vagy DATA a\$(). A vonalon az adatok csak sorosan továbbíthatók, mivel a vezeték sodrott érpárú. Azzal a különbséggel, hogy a magnókezelő rutin csak 8 bites puffert használ (L regiszter), a LAN a továbbítás előtt egy, a memóriában létrehozott 2 kbit-es puffert tölt fel.

A LAN másik funkciójában adatokat továbbít a célállomás(ok) felé. Ez kicsit bonyolultabb a programátvitelnél. Itt már nem elég kijelölni a NETWORK üzemmódot, ehhez még egy streamet is hozzá kell rendelni (egyszerre többet is lehet). Miután kijelöltük az „n” csatornát és az állomászámot, megnyitunk egy streamet a LAN számára: OPEN #X; "n"; a., ahol X a stream (0-15), az a pedig az adatátvitelben a

Spectrumok összekapcsolása helyi hálózatban



ZX-SPECTRUM mint digitális jeleket tároló oszcilloszkóp

Előfordulhat, hogy ismeretlen digitális jelsorozatot oszcilloszkópon kell elemeznünk. Ha a vizsgálandó jel nem periodikus, tároló oszcilloszkópra van szükség. A magnetofonbemenet segítségével egy csatorna a hardver átalakítása nélkül vizsgálható ZX-Spectrumon. Az 1. programmal 8 másodpercnyi jelet tárolhatunk a memóriában 50 µs felbontással. Ez azt jelenti, hogy a vizsgálandó jel néhány kHz frekvenciájú lehet.

A program a memóriában formál működik, de célszerű annak felső részében elhelyezni. A vizsgált jel a memóriában bitfolytonosan helyezkedik el a 40000-tól 59999-ig terjedő tartományban.

FF78	F3	0010	ORG	65400	
FF78	F3	0020	DI		
FF79	21409C	0030	LD	HL, 9C40H	
FF7C	11F4F	0040	LD	DE, 4F1FH	
FF7F	0E06	0050	LD	C, 6	
FF81	0E06	0060	NAGY	LD	B, 8
FF83	C5	0070	KIS	PUSH	BC
FF84	41	0080	LD	B, C	
FF85	10FE	0090	DJNZ	-2	
FF87	DBFE	0100	IN	A, (254)	
FF89	C607	0110	RLC	A	
FF8B	C607	0120	RLC	A	
FF8D	C616	0130	POP	(HL)	
FF8F	C1	0140	POP	BC	
FF90	0E06	0150	LD	C, 6	
FF92	10FE	0160	DJNZ	KIS	
FF94	23	0170	INC	HL	
FF95	1B	0180	DEC	DE	
FF96	0E02	0190	LD	C, 2	
FF98	3E90	0200	LD	R, 0	
FF9A	82	0210	OR	D	
FF9B	00	0220	NOP		
FF9C	00	0230	NOP		
FF9D	20E2	0240	JR	NZ, NAGY	
FF9F	FB	0250	EI		
FFA0	C9	0260	RET		
		0270	END		

1. program

```

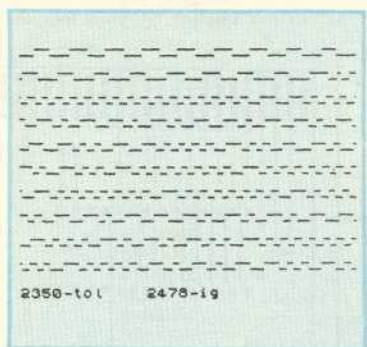
1 CLEAR 09999: LOAD "SZKOP.CO
DE 65400
2 CLS : PRINT "Barmely govb i
ndi jre a felvétel" : PAUSE .5
3 IF INKEY="" THEN GO TO 3
5 RANDOMIZE USR 65400: LET e =
0: GO TO 15
15 LET i="kezdet, a s " : e
16 INPUT i:INT (40000+e*.25)
20 CLS : PRINT AT 20,0: i;-4000
0) : 4 -" -tol"
30 FOR k=16384 TO 16576 STEP 6
4: GO SUB 100: NEXT k
40 FOR k=16432 TO 16624 STEP 8
4: GO SUB 100: NEXT k
50 FOR k=20480 TO 20576 STEP 6
4: GO SUB 100: NEXT k
55 PRINT AT 20,12: i;-40000 : .4
i;"-19"
60 LET a$=INKEY$: IF a$="" THE
N GO TO 60
62 IF a$="z" THEN COPY =
70 IF i<60000 THEN GO TO 20
75 STOP
100 FOR j=0 TO 31: LET c=PEEK i
: POKE k+j,c: POKE l+j,1024+255-
c: LET i=i+1: NEXT j: RETURN
    
```

2. program

10-zel újraindítjuk. Így tetszőleges időpillanattól kezdhetünk. Új adatokat GO TO 2-vel olvashatunk be.

Az ábrán a ZX-Spectrum magnójele látható. Az első és a második sor a bevezető jel, a második sor végén fejléc kezdődik. A gépi kódú program decimális listáját DATA sorokban elhelyezve, a programot bevihetjük a memória megfelelő helyére: 234,33,64,156,17,31,79,14,6,6,8,197,65,16,254,219,254,203,7,203,7,203,22,193,14,6,16,239,35,27,14,2,62,0,178,0,0,32,226,251,201

DR. MAROSSY KÁLMÁN



A ZX-Spectrum magnójele

A kezdő cím a HL regiszterpárba kerül, a DE-be pedig a feltöltendő bájtok számánál 255-tel nagyobb szám. Ha ezeket az adatokat meg akarjuk változtatni, vegyük figyelembe, hogy a program csak a 32768 cím fölött működik helyes időzítéssel, alatta az ULA prioritása miatt értékelhetetlen eredményt kapunk. A programban levő két NOP utasítás biztosítja, hogy minden bit feltöltése ugyanannyi időközönként történjék, függetlenül attól, hogy a bájt első vagy közbelső bite. A program egy körülfordulása 173 órajelnyi. Ez 3,5 MHz órajel esetén 49,43 µs; 1,2 százalékkal kevesebb, mint 50. Ez a hiba általában elhanyagolható.

A megjelenítésre a 2. program szolgál. Egy ernyőre 10 sorban 128 ms-nyi jelet rajzol fel, amelyet nyomtatóra is kivihetünk. A 8 másodpercnyi felvételnek általában csak egy részletre vagyunk kíváncsiak. Ezt az ernyőn előrelepozással megkereshetjük, és az érdekes időtartományokat feljegyezzük. A programot leállítjuk és GO TO

cél- vagy az adóállomás száma, — attól függően, hogy mi vesszük vagy adjuk az adatokat. Ha ez rendben, akkor az X streambe írhatunk: PRINT #X, ... , illetve onnan olvashatunk: INPUT #X, Az adatok először egy 2 kbit-es pufferben gyűlnek, és az átvitel csak akkor kezdődik, ha ez megtelt.

Egy stream — hacsak nem zárjuk le és nyitjuk újra — csak egyféle funkcióra alkalmas. Ha először írunk bele, akkor később nem olvashatunk, és fordítva. Tehát egy stream funkcióját mindig az első utasítás határozza meg. Ha mégis kiadunk funkcióellenes utasítást, akkor »READING TO A „WRITE FILE“«, vagy »WRITING TO A „READ FILE“« hibaüzenetet kapunk. Ha a streambe írunk, akkor dolgunk végeztével feltétlenül zárjuk le a csatornát: CLOSE #X. Ha ezt nem tesszük meg, akkor a vevő továbbra is adatokra fog várni.

A másik soros átviteli lehetőség az RS232C. Ennek a csatlakozója az If-1 hátoldalán a LAN aljzatoktól balra található. Az If-1-féle RS232C-nek két nagy hibája van. Az egyikén kis barkácsolással lehet segíteni: a szabványos RS232C csatlakozó egy 25 pólusú D, az If-1-beli viszont a botkormánynál megismert 9 pólusú. A másik probléma már kicsit bonyolultabb: a szabvány a logikai 0 szintnek +15 V-t, a logikai 1 szintnek -15 V-t ír elő, míg az If-1 a TTL szinteket használja. Ezen csak körülményesen, szintváltóval lehet segíteni. Az adatátvitel itt is soros formátumú, de még maximális sebességnél is sokkal lassabb, mint a LAN, mert a nyolc adatbitet egy startbit előzi meg, és 2 stopbit zárja.

Az RS232C átvitelnél a LAN-hoz hasonlóan itt is először egy FORMAT utasítást kell kiadni: FORMAT "", v. az idézőjelbe itt is a csatornakód kerül; v a sebesség baudban. Az If-1 csak kilencféle sebességet ismer, így az általunk adott adatot mindig kerekíti. Csatornakódként a t és a b betűt választhatjuk a következők szerint.

A t (TEXT) egy szövegcsatornát jelöl ki. Ez az üzem nyomtató használatkor előnyös lehet, de egyébként bonyodalmas, mivel a nem nyomtatható karaktereket és a vezérlőkódokat el sem küldi.

A b bináris csatorna nem vizsgálja az információt, így minden adat eljut a vevő géphez. Ezt a csatornát általános célú adatforgalomra jelöljük ki. Ha programot szeretnénk egyik gépből a másikba továbbítani, akkor gépünket a b csatornára állítsuk, majd az utasításokat a LAN-hoz hasonlóan adjuk ki. Például:

```
LOAD="n" CODE helyett LOAD="b" CODE.
```

Az RS232C-n is mehet tisztán adatforgalom, de ennek formátuma szintén megegyezik a LAN-éval.

VARGA ISTVÁN

Külső ROM C64-hez

Ezzel a cikkel azoknak szeretnénk segíteni, akik programjaikat a gyári kártyákhoz hasonló külső ROM-ban akarják tárolni.

A külső ROM-kártya a gép amúgy is szokásos RAM-jából 8 kb-ot leválaszt. Két lehetőségünk van: vagy a teljes 8 kb-ot RAM-ot helyettesítjük ROM-mal, vagy csak egy kis részét, ami a tulajdonképpen felhasználandó program szabad területre másolásához szükséges. Ez akár a kártya által kiválasztott RAM is lehet. Az átmásolás után a kártya lekapcsolható.

Az első esetben a program a leválasztott RAM helyén a ROM-ban futhat. A most szóban forgó kártya a BASIC interpreter alatt levő 8 kb-ot használja. Ez megegyezik a szabad BASIC munkaterület tetejével. A maximális programhossz tehát szintén ennyi.

A kártya megépítésénél szinte bármilyen

EPROM, PROM szóba jöhet. A gép hátulján a „cartridge port” vezetékai között találhatjuk meg azokat a vonalakat, amelyek az EPROM megcímzését, engedélyezését és olvasását teszik lehetővé (1. ábra). A címzésre és az adatok fogadására az ábrán látható vonalakra van szükség.

Az EPROM szelektálására a ROML vonalat használjuk, mely \$8000-tól \$BFFF-ig aktív 0, ha közben az EXROM vonalat földre kötöttük. A leköttést kapcsolóval választjuk meg.

Ha megcímztük a kártyát, és a CS jelet is kiküldtük, akkor időzítmény problémák nem léphetnek fel, ezért az OE-t földre kötjük.

A kártyacsatlakozó sora a szabványos 0,1 inch osztású kell hogy legyen. (A nyomdai rajz itt ettől méretben eltérhet!)

Az általunk készített NYÁK a leggyak-

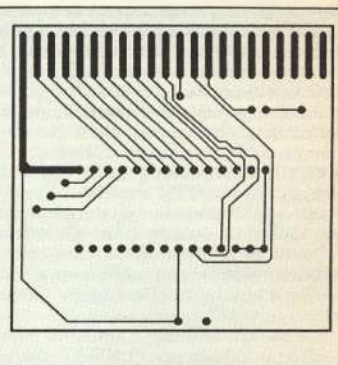
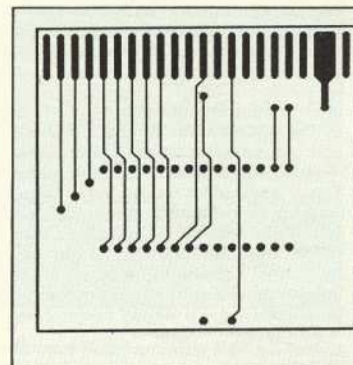
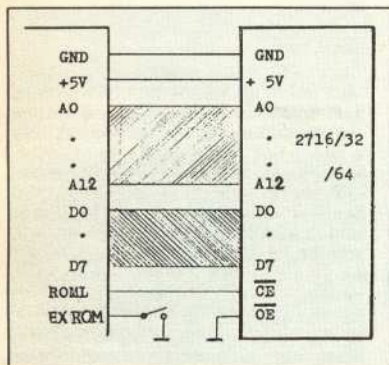
rabban használt EPROM-ok (2716, 2716 A, 2732 A és ezek CMOS változatai: 2516, 2732, 2764) befogadására képes (2. a, b, c ábrák). A NYÁK-ot a 3. ábrán mutatjuk be.

A kártyát működtető szoftverrel kapcsolatban néhány megjegyzés. Ha olyan kártya készítése a cél, amely a gép bekapcsolása után azonnal, automatikusan indul, akkor vegyük figyelembe, hogy a gép reset rutinja vizsgálja \$8000-tól a tárat. Ha itt a CBM 80 emelt karakterek ASCII kódjai találhatók (\$ C3, \$ C2, \$ CD, \$ 38, \$ 30), akkor a programfutás a (\$8000—\$8001) címre ugrik. A másik két bájt az NMI vektora (RUN STOP/RESTORE).

A gép tulajdonképpen még inicializálандó; ez legegyszerűbben a KERNAL reset rutinjának némi módosításával érhető el.

MARTOS

Számítástechnika Kör Zalaegerszeg



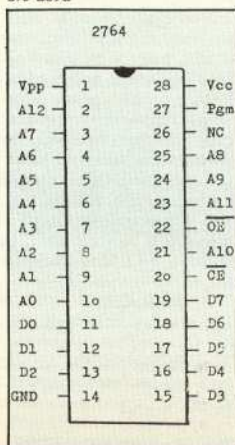
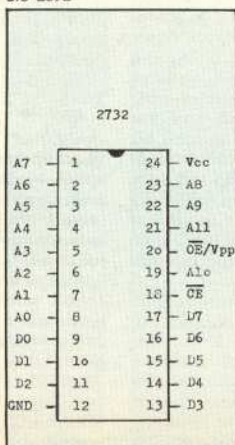
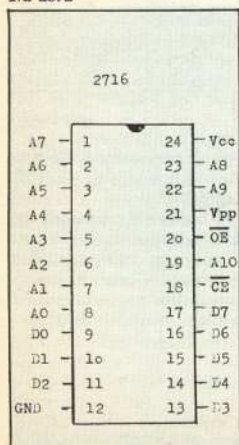
1. ábra

3. ábra

2/a ábra

2/b ábra

2/c ábra



Közületek
figyelem!

**Mikroszámítógépet
akarnak vásárolni?
Tájékoztódnak
előtte
a naprakész
piaci helyzetről!**

Díjtalan ismertető!
MESZ
Számítástechnika
1368 Budapest
Pf. 193.

A VC20 hardver adta lehetőségei

Hanggenerálás

A 6561-es chip öt regiszterének használatával lehet hangadásra kényszeríteni a VC20-as gépet. Ezek a regiszterek a következők.

36874-es tárcim: audioszcillátor 1. (mély): a 0–6. bitek a frekvenciát, a 7. bit pedig a be- (=1), illetve a kikapcsolt állapotot jelzi.

36875-ös tárcim: audioszcillátor 2. (közepes): bitfelosztás a 36874-es tárcimnél.

36876-os tárcim: audioszcillátor 3. (magas): bitfelosztás a 36874-es tárcimnél.

36877-es tárcim: zajgenerátor (fehérzaj): a zaj milyenségét határozza meg, a 7. bit mint az előzőknél.

36878-as tárcim: az előző négy regiszter hangerejét határozza meg. Értéke 0–15 lehet. Többszínű üzemmódban (lásd itt később) vigyázni kell arra, hogy csak és csak is az alsó 4 bit tartalmát változtassuk meg.

A kiválasztott hang megszólaltatásához az alábbi értékek valamelyikét kell az első három regiszter egyikébe beírni, természetesen a hangerő és a zaj értékének meghatározása után.

OKTÁVOK: 1. 2. 3. 4.

HANG:	135	195	225	240
C	135	195	225	240
C#	143	199	227	241
D	147	201	228	
D#	151	203	229	
E	159	207	231	
F	163	209	232	
F#	167	212	233	
G	175	215	235	
G#	179	217	236	
A	183	219	237	
A#	187	221	238	
H	191	223	239	

Többszínű üzemmód

A képernyő egy 8 × 8-as méretű karakterhelyén megjelenő pontok kétszínű üzemmódban vagy háttérszínűek (nem látszanak), vagy a színmemóriában levő kódznak megfelelő színűek. Többszínű üzemmódban összesen négy szín lehet egy karakterhelyen belül, a háttérszínnel együtt.

A többszínű üzemmód bekapcsolása: a színmemóriában a karakter helyének megfelelő bajt 3. bitjét magasra állítjuk. A színmemória alapkiépítésben a 38400–38884 címeken található, sorfolytonosan. Ha bővítve van a gép, akkor a 37888–38399 címeken lesz a színmemória. Többszínű üzemmód esetén a karakter alakját definiáló biteket párosával kell tekintenünk, és a két bit helyén a bitpár értékének megfelelően két azonos színű pont fog megjelenni a

képernyőn. A bitpárok a következő színek tartalmazzák:

00 = háttérszín: a 36879-es cím felső 4 bitje
 01 = keretszín: a 36879-es cím alsó 3 bitje
 11 = segédszín: a 36878-as cím felső 4 bitje
 10 = a színmemóriában levő szín (alsó 3 bit)

Igy például az „A” betű alakjai így néznek ki:

két színnel:

```

XX
XXXX
XX XX
XXXXXX
XX XX
XX XX
XX XX
    
```

több színnel:

```

DDAABBDD
DDCCCCDD
AABBAABB
AACCCBBB
AABBAABB
AABBAABB
AABBAABB
DDDDDDDD
    
```

bittérkép:

```

00011000
00111100
01100110
01111110
01100110
01100110
01100110
00000000
    
```

A többszínű módnál az azonos színű pontokat egyféle betűk jelölik.

Grafikus lehetőségek

Mivel a grafika a képernyő és a karaktergenerátor áthelyezésén alapszik, először ezek áthelyezésének problémáit kell tisztázni.

A képernyő áthelyezésére a következő utasítást kell használni:
 POKE 36869, (PEEK(36869)AND15) OR X:

POKE 36866, (PEEK(36866)AND127) OR 128*Y
 ahol X és Y értéke a következőképpen alakulhat:

X	Y	A képernyőmemória kezdőcíme
128	0	0
128	1	512
144	0	1024
144	1	1536
160	0	2048
160	1	2560
176	0	3072
176	1	3584
192	0	4096
192	1	4608
208	0	5120

208	1	5632
224	0	6144
224	1	6656
240	0	7168
240	1	7690

A karaktermemória áthelyezésére a következő utasítást kell használni:

POKE 36869, (PEEK(36869)AND240) OR X

ahol X értéke a következő lehet:

X	A karaktermemória helye
4	4096
5	5120
6	6144
7	7168
8	8192
9	9216
10	10240
11	11264
12	12288
13	13312
14	14336
15	15360

Ha X értéke 0, 1, 2 vagy 3, akkor a gép a ROM-ból veszi a karaktereket. A nagyfelbontású grafikánál ugyanazokat az elveket használjuk, mint a felhasználói karakterek definiálásánál. A kapcsolatot a VIDEO-RAM és a karaktermemória között a következő. Ha a karaktermemóriában minden bit nullára van állítva, akkor a képernyőn sincs egy pont sem kilágitva. A nagyfelbontású grafika eléréséhez a VIDEO-RAM-ba betöltjük mind a 256 karakter kódját, majd a karaktergenerátor kezdőcímét egyenlővé tesszük a VIDEO-RAM kezdőcímével. Mivel 1-1 bit helyének meghatározása eléggé bonyolult, ezért minden további magyarázat nélkül leírok egy rövid grafikai programot.

- PRINT "SHIFT+CLR/HOME"
- POKE 36867,128
- POKE 36865,60
- F(8)=θ:F(θ)=128:F(1)=64:F(2)=32:F(3)=16
- F(4)=8:F(5)=4:F(6)=2:F(7)=1
- FOR Q=θ TO 255
- POKE 7680+θ,Q
- POKE 38400+Q,2
- NEXT Q
- FOR Q=5120 TO 7160
- POKE Q,θ
- NEXT Q
- POKE 36869,253
- POKE 36866, PEEK(36866) OR 128
- POKE 36867,150
- FOR C=θ TO 175
- L=45+40*SIN(C/10)
- A=5120
- LR=L/8
- LA=INT(LR)
- A=A+(LA*176)
- LR=(LR-LA)*8

THE USERS PORT

San Fernando Valley Commodore Users Group

Menüprogram

Amerikai utamon ígéretet kaptam a San Fernando Valley Commodore Users Group (Los Angeles) nevű klub vezetőjétől, David L. Oseas úrtól, hogy The Users Port című lapjukat küldeni fogják, és abból cikkeket vehetünk át. Ez meg is történt, így ezután rendszeresen közlünk ebből a lapból átvett cikkeket, rövid programokat. (SIMONYI ENDRE)

1. lista

```

10 PRINT "COMODORE"
15 FOR I=1 TO 100:GOTO 20:FOR J=1 TO 10
20 PRINT "LOMB"CHR$(44)CHR$(47)PRINT CHR$(1)CHR$(2)
25 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
30 PRINT "TELE"
35 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
40 PRINT "MAGDA"
45 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
50 PRINT "PASCAL"
55 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
60 PRINT "BASIC"
65 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
70 PRINT "ZOOM"
75 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
80 PRINT "VIC"
85 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
90 PRINT "RAM"
95 FOR I=1 TO 100:PRINT CHR$(44)CHR$(47)
100 PRINT "VIC"
    
```

Sokakat érdekelhet, hogyan lehet rövid menüprogramból hosszabb programokat bevitelni. Ezt nemcsak BASIC, hanem ZOB.

```

10 PRINT "VIC"
15 PRINT "RAM"
20 PRINT "VIC"
25 PRINT "VIC"
30 PRINT "VIC"
35 PRINT "VIC"
40 PRINT "VIC"
45 PRINT "VIC"
50 PRINT "VIC"
55 PRINT "VIC"
60 PRINT "VIC"
65 PRINT "VIC"
70 PRINT "VIC"
75 PRINT "VIC"
80 PRINT "VIC"
85 PRINT "VIC"
90 PRINT "VIC"
95 PRINT "VIC"
100 PRINT "VIC"
    
```

OM PASCAL programokból is megtehető.

Az 1. lista az újság eredeti programja, magyar nyelvű üzenetekkel. A továbbiak ennek az elgondolásnak alapján készült saját programok.

A 2. lista egy BASIC menüprogram, az előzőnek alkalmazására. Azt mutatja be, hogyan lehet tetszőleges hosszúságú BASIC programokat változó felhasználásával behívni. BASIC programok esetén törli a változó listát.

A 3. listán az első program ZOOM PASCAL változata látható, azzal a módosítással, hogy megkérdezi a behívandó program nevét.

PÁL MAGDA

3. lista

```

PROGRAM MENU
VAR NEV:STRING
CH:CHAR
L:INTEGER
BEGIN
  REPEAT
    WRITE(0,"BE TÖLTENDO PROGRAM NEVE");
    REPEAT UNTIL
      L=>#LEN(NEV);
    UNTIL L<2;
    IF REPEAT THEN BEGIN
      WRITE("EGYSEB SZINT");
      REPEAT(0);
      NEV:=CHR(0);
      WRITE(0,"BE TÖLTENDO PROGRAM NEVE");
      REPEAT UNTIL L=>#LEN(NEV);
      UNTIL L<2;
      IF REPEAT THEN BEGIN
        WRITE("EGYSEB SZINT");
        REPEAT(0);
        NEV:=CHR(0);
        WRITE(0,"BE TÖLTENDO PROGRAM NEVE");
        REPEAT UNTIL L=>#LEN(NEV);
        UNTIL L<2;
      END;
    END;
  END;
END.
    
```

- 23 CR=C/8
- 24 CA=INT(CR)
- 25 A=A+(CA*8)
- 26 A=A+LR
- 27 CR=INT((CR-CA)*8)
- 28 POKEA, PEEK(A) OR F(CR)
- 29 NEXT C
- 30 GET AS:IF AS="" THEN 30
- 31 POKE 36869,240
- 32 POKE 36866,150
- 33 POKE 36867,174
- 34 POKE 36865,38

A VIC chip 16 regiszterének lehetőségei

- 0. regiszter 36864-es tárcím: A bal felső karakter első pixeloszlopának távolságát állítja be a képernyő bal szélétől.
- 1. regiszter 36865-ös tárcím: A bal felső karakter felső pixelsorának távolságát állítja a képernyő tetejétől.
- 2. regiszter 36866-os tárcím: A 0-6. bitek az egy sorban levő karakterek számát mutatják, a 7. bitnek a képernyőmemória helyének meghatározásában van szerepe.
- 3. regiszter 36867-es tárcím: Ha a 0. bit=1, akkor 8x16-os karaktermérettel dolgozik a gép, egyébként normál, 8x8-as karakterméretet használ. A 7. bit a raszterszámláló legkisebb helyértékű biteje. Az 1-6. bitek a képernyő oszlopszámának kétszeresét adják.
- 4. regiszter 36868-as tárcím: Kiegészítve a 3. regiszter 7. bitjével, az utóljára frissített raszterszor számát adja.

- 5. regiszter 36869-es tárcím: A VIDEO-RAM és a karaktergenerátor kezdőcíme.
- 6. regiszter 36870-es tárcím: Fényceruza vízszintes koordinátája.
- 7. regiszter 36871-es tárcím: Fényceruza függőleges koordinátája.
- 8. regiszter 36872-es tárcím: Játék I/O X potméter érték.
- 9. regiszter 36873-as tárcím: Játék I/O Y potméter érték.
- 10-14. regiszterek 36874-36878-as tárcímek: Lásd a hanggenerálásnál, a cikk elején.
- 15. regiszter 36879-es tárcím: A felső 4 bit a háttérszínkódot, az alsó 3 bit a keretszínkódot tartalmazza. Ha a 3. bit értéke 1, akkor a kiírás inverz, ellenkező esetben normális. A háttér, a keretszín, valamint az invertálás megválasztása:

POKE 36869,16*háttérszínérték+8*invertálási mód+keretszínérték.

Az ismertetés Nick Hampshire: VIC Reveal-ed c. könyve alapján készült.

(td)

KI AD MAGYARÁZATOT?

Feladatom volt, hogy rajzoljak a DRAGON számítógép képernyőjére egy bogarat és nyomtassam ki, a nyomtatónak azt a lehetőségét kihasználva, hogy egy sorba 1920 pontot képes rajolni.

A DRAGON képernyőjén 256 × 192 pont jeleníthető meg. A képernyőn elő is állítottam, azonban a papíron más jelent meg. Mivel a nyomtatás nagyon lassan haladt, módosítottam a programot.

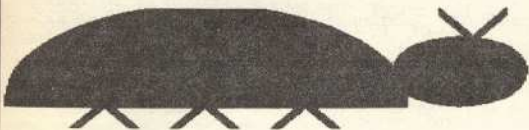
Mit változtattam? Kihagytam az X változót; így elmaradt több mint 5000 értékadás. Kiemeltem a második ciklusból a szorzást a ciklus elé: így elmaradt több mint 5000 szorzás. A hatványozás helyett osztottam: így elmaradt több mint 80 000 hatványozás, helyette lett több mint 40 000 osztás. Végül a rajz feletti és alatti üres területet elhagytam: így a rajzterület kb. 40 százalékkal csökkent. Meglepetésemre nemcsak a sebessége nőtt meg, hanem a nyomtatott rajz is megváltozott. Miért?

Az 1. lista és ábra az első, a 2. a második változat. A PRINT utasítások mind a nyomtatóra küldtek adatokat, az ESC Q az 1920 pont/sor beállító, ESC T 01 a sorközt minimumra állítja, az ESC W 0008 az utána következő 16 bit által megadott 16 függőleges pontot nyolcszor megismétli. Végül a 13 és 10 a kicsivissza/soremelés. Ezek sem változtak. A DRAGON Microsoft—Basic változatot használ.

A kinyomtatás képernyőpontoként, azaz bitenként történt, ehhez bájtonként kiolvastattam a képernyőtartalmat, és a bájtot bitekre bontottam — először a hatványozással, majd az osztással. Ezután minden bitértéket nyolcszorozva nyomtattam ki, így az ábrát nyolcszorosára felnagyítottam.

Ellenőrizve mind a számítógép, mind a nyomtató hibátlanul működött.

SIMONYI ZSUZSA



```
10 PMODE4,1:PCLS:SCREEN1,1
20 CIRCLE(100,126),90,,1,.5,1:CIRCLE(214,
,96),30:CIRCLE(220,84),6,,1,.5,1
30 LINE(10,126)-(190,126),PSET:LINE(214,
84)-(226,84),PSET:LINE(40,152)-(55,126),
PSET:LINE-(70,152),PSET:LINE(85,152)-(10
0,126),PSET:LINE-(115,152),PSET:LINE(130
,152)-(145,126),PSET:LINE-(160,152),PSET
:LINE(205,40)-(220,66),PSET:LINE-(235,40
),PSET
40 PAINT(214,96):PAINT(100,100)
50 PRINT#-2,CHR$(24);CHR$(27);"Q";CHR$(2
7);"T01";:A=1536:FOR I=0TO 191:FOR J=1 T
O 30:X=PEEK(32*I+J+A):Y(0)=X:FOR K=1 TO
8:B=(2*(8-K))*INT(Y(K-1)/(2*(8-K))):Y(K)
=Y(K-1)-B:PRINT#-2,CHR$(27);"W0008";
60 IF B<0 THEN PRINT#-2,CHR$(255);CHR$(
255);:GOTO70 ELSE PRINT#-2,CHR$(0);CHR$(
0);:GOTO70
70 NEXT K:NEXT J:PRINT#-2,CHR$(13);CHR$(
10);:NEXT I:STOP
```

```
10 PMODE4,1:PCLS:SCREEN1,1
20 CIRCLE(100,126),90,,1,.5,1:CIRCLE(214,
,96),30:CIRCLE(220,84),6,,1,.5,1
30 LINE(10,126)-(190,126),PSET:LINE(214,
84)-(226,84),PSET:LINE(40,152)-(55,126),
PSET:LINE-(70,152),PSET:LINE(85,152)-(10
0,126),PSET:LINE-(115,152),PSET:LINE(130
,152)-(145,126),PSET:LINE-(160,152),PSET
:LINE(205,40)-(220,66),PSET:LINE-(235,40
),PSET
40 PAINT(214,96):PAINT(100,100)
50 PRINT#-2,CHR$(24);CHR$(27);"Q";CHR$(2
7);"T01";:A=1536:FOR I=36TO 152:D=32*I+A
:FOR J=1 TO 30:Y(0)=PEEK(J+D):C=256:FOR
K=1 TO 8:C=C/2:B=C*INT(Y(K-1)/C):Y(K)=Y(
K-1)-B:PRINT#-2,CHR$(27);"W0008";
60 IF B<0 THEN PRINT#-2,CHR$(255);CHR$(
255);:GOTO70 ELSE PRINT#-2,CHR$(0);CHR$(
0);:GOTO70
70 NEXT K:NEXT J:PRINT#-2,CHR$(13);CHR$(
10);:NEXT I:STOP
```

A 16 kbájtos HT—1080Z iskolaszámítógépeken nagy sikerrel használt **zFORTH** után közreadjuk az azonos filozófiával készített, a 16 kbájtos Commodore 16 gépeken használható

cFORTH-ot

A cFORTH mind kazettával, mind hajlékonylemez-egységgel használható, s mindkét eszközt tudja kezelni. Ha bővített táru C16-on vagy C plus/4-en használjuk, akkor automatikusan 32 kbájtig bővített tárterületet kezel.

Megrendelhető csak iskolák részére
a Tudományszervezési és Informatikai Intézetnél,
Budapest, Pf. 454. 1372.
Ügyintéző: Bánó György, tel.: 864-011/2857.

A dinamikus értékelés jelentősége

Mint tudjuk, amíg a sakkozó megtesz egy lépést, a változatok sokaságát gondolja át, rövidkeli, és ezek alapján dönt, hogy melyik lépést válassza az illető állásban. A számítógép az állás értékelésénél tulajdonképpen ugyanezt a műveletet végzi gépi eszközökkel. Ezt a minimax algoritmus ismertetésénél majd részletesen tárgyaljuk. Egyelőre csak azt kell tudnunk, hogy a gép is változatokat számít — hasonlóan az emberhez —, de nagy különbség, hogy a gépnek nincsen tapasztalás alapú intuíciónak képessége, szemben a sakkozók. Mivel az eddig tárgyalt statikus értékelőfüggvény nem lehet tökéletes, ezért az állásértékelés finomítása miatt alkalmazzuk az úgynevezett dinamikus értékelőfüggvényt.

A dinamikus értékelőfüggvény a statikus értékeléssel szemben nem az előre számolt változat végpontjában kialakuló állást értékeli, hanem azokat a lépéseket, amelyekkel az az állás létrejön. Vagyis a végállás elemzésétől teljesen függetlenül, csak a kezdőállás függvényében a végállásig megtett lépéseket veszi egyenként figyelembe.

Egyes hadállásokban többféle nem egyenértékű lépéssorozat is eljuthatunk egy adott végálláshoz. Nézzük például az 1. ábrát. Világos a bemutatott állásból látszólag két-féle úton érheti el a mattot. Az első — kényszerítő erejű — változat:

1. Vd2 × g5 + Kg7 — h7
2. Bf3 — h3 (Hd7 — f6)
3. Vg5 × h6 matt.

A második — nem kényszerítő erejű — lehetséges út:

1. Bf3 — h3 (Hd7 — f6)
2. Vd2 × g5 + Kg7 — h7
3. Vg5 × h6 matt.

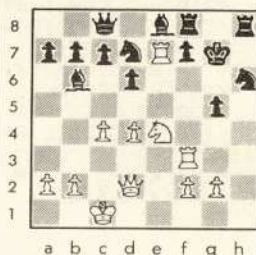
Láthatjuk, hogy kétféle úton is eljuthatunk a mattig, de valóságos partiban csak az első esetben tudja világos bemutatott sötétet, ugyanis 1.Bd1 — h1 lépésre sötét növeli királya szabadsági fokát, azaz mozgékonyaságát a Bf8 — g8 lépéssel. Így királya az f8 mezőn el tud bújni a matt elől, és anyagi előnyének érvényesítésével megnyeri a játszmát:

1. Bf3 — h3? Bf8 — g8
2. Vd2 × g5 + K7 — f8!

Tehát a dinamikus értékelőfüggvénynek külön értékelnie kell a forszirt lépéseket az aktív és a passzív fél szemközé-

BITEK ÉS FIGURÁK

Dinamikus értékelőfüggvény



1. ábra

ből. Ilyen esetben megtörténhet, hogy hiába azonos a kezdő- és a végállás, az eltérő lépéssorozat miatt a dinamikus értékelőfüggvény más és más értékeket mutat.

A dinamikus értékelés hátránya, hogy dinamikus értékelőfüggvényünk közel egyenértékű lépéssorozatok esetében is nagyon különböző értékeket mutathat.

Felvetődhet a kérdés, hogy ezek szerint nem a végeredmény az érdekes? De igen. A dinamikus értékelőfüggvény használata ennek ellenére szükséges, mert statisztikusan sok hibás lépés kiszűrésére képes.

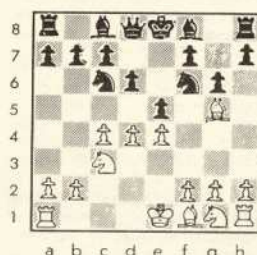
Milyen legyen a dinamikus értékelőfüggvény?

E függvényben a lépésszámtól, a táblán lévő figurák számától és anyagi értékük összegétől függően több feltételt teljesülését vizsgálhatjuk meg. Ilyen feltételek közé sorolhatjuk a sakkjáték több alapelvét és szabályát, amelyek közül a legfontosabbak:

1. A játszma elején ne lépünk többet egy figurával, ameddig a többi tiszttel ki nem fejlődöttünk.

2. Figyelembe vehetjük a sáncolást. Amelyik fél a lépéssorozat közben elsáncolt, pluszpontot kap.

3. Számon tarthatjuk az 50



2. ábra

lépés szabályt, vagyis azt, hogy az utolsó 50 lépésben szerepelt-e gyaloglépés vagy ütés, mert ha nem, akkor az állás döntetlen.

4. Figyeljük a háromszori lépéssimulést. Ugyanis ilyenkor szintén döntetlen.

Egy lehetséges példa a sok közül (csak fiktív pontszámokat tüntettünk fel):

Gyaloglépések esetén:

1. ha a lépésszám < 10:
 - ha a, b, c, f, g, h gyaloglépet: -4;
 - ha d2—d4, e2—e4, d7—d5, e7—e5 történt: +4;
2. 10 < a lépésszám < 24:
 - ha a1(a8), b1(b8), c1(c8), f1(f8), g1(g8), h1(h8) mezőn álló király előtti gyaloglépet: -4;
3. 24 < lépésszám esetén

nincs külön értékelés.

Huszár és futó lépése esetén:

- ha mozdulatlan figura lépett: +10;
- ha már mozdult a figura és a lépésszám < 9: -4 (ha a lépésszám > 9 akkor: 0);
- ha a lépésszám < 15 és a tisz: d3-ra, d6-ra, e3-ra vagy e6-ra lépett úgy, hogy a megfelelő d2, d7, e2 vagy e7 mezőn gyalog áll: -15.

Bástya vagy vezér lépése esetén:

- ha a lépésszám < 7: -10;
- ha a lépésszám > = 7: 0.

Királylépés esetén:

- ha a királlyal lépett, de még nem sáncolt és a figura értékösszeg > 9 vezérnél: -8;
- ha rövidre sáncolt: 12;
- ha hosszúra sáncolt: 6.

Az ilyen értékelésnek az a hibája, hogy megbontja a szimmetriát. Ugyanis a statikus értékelésnél csak olyan szempontokkal számoltunk, amelyek mind világos, mind sötét figurákra egyszerre kiszámíthatóak voltak, és így könnyen képezhették a kettő arányát. Jelen esetben ez képtelenség, mert egyszerre csak egy bábuval léphetünk. A függvényt kiegyenlíthetjük, ha csak páros fellépés után vesszük figyelembe a dinamikus értékelés eredményét. A tapasztalat viszont azt igazolta, hogy ennek az aszimmetriának nincs olyan nagy jelentősége.

Az előbb felsorolt lépéserértékelést kiegészíthetjük más komponensekkel is, csak arra vigyázunk, hogy a sok feltételezsgát ne legyen ellentmondó és ne lassítsa le programunkat. Így például figyelhetjük, melyik figura áll kötésben, illetve melyik figura ellépésével hozunk létre kötéset. A 2. ábrán látható állásban sötétnek a játszma folyamán az e7—e5 gyaloglépés megtételére kell törekednie, ugyanis csak így juthat a centrumban ellenjátékhoz. Ez viszont most azonnal nem megy, mert a lépéssel az f6 huszárját lekötne, majd a figuracserék után elvesztené. A program a figuraváltást csak a 2. d4 × e5, d6 × e5 3. Vd1 × d8, Hc6 × d8 4. Fg5 × f6 lépések megvizsgálása után venné észre, ami nagyon sok időbe telne. Ezért a dinamikus értékelés használatával 7 fellépés kiértékelése helyett már hamarabb abbahagyja az elemzést, mert az f6 huszár lekötése miatt — és mert egyedül a vezér védi — a létrejövő állást már korábban lepottozza.

A dinamikus értékelés feladata az is, hogy ütőskor csökkentse a figurák számának értékét ugyanúgy, mint az anyagi érték összegét, gyalogátváltás esetén viszont ez utóbbit növelje. Gyorsabbá tehetjük így programunkat, mivel nem kell minden állás értékelésénél mind a figuraszámot, mind a figuraösszeget újra kiszámítani, csak az ütéseknél. Ezek száma viszont a legtöbb esetben elenyésző az egyéb lépések számához képest.

COMMODORE—64

Játék POKE-ok

Nagy sikerük volt a Mikromagazin 1985/6. és 1986/1. számában megjelent Commodore „játék POKE”-oknak, ezért valószínűleg a C16-os játékok módosítására is kíváncsi a lap közönsége. Az itt közölt szerint számos „életet” hosszabíthatnak meg olvasóink.

Az alábbi két játék tulajdonképpen BASIC program gépi kódú részekkel, ezért bennük a szokásos módon BASIC editorral javíthatunk, mint valami hibát.

RIG ATTACK A 300-as sorban
HL=HL-1 helyett
HL=HL-0 szerepeljen.

SPACE MISSION A 750-es sorban a
ME=ME-1 helyett
ME=ME-0 a jó.

A táblázatban felsorolt programok teljesen gépi kódúak, ezért az átállításuk előtt MONITOR parancsral gépi kódú monitorba kell lépünk, majd a táblázat szerinti sorok beírása után X parancsral ki kell lépni a monitorból. Ugyanezt egyszerűbb lenne POKE utasítással elintézni, de a játékok egy része nem szereti ezt, a módszert, és OUT OF MEMORY ERROR hibajelzéssel jutalmaz érte. Természetesen a POKE utasítást nem is hajtja végre.

Fontos, hogy a táblázat sorait mindig külön sorba kell írni, és a sorok között — minden sor után — le kell nyomni a RETURN billentyűt. A > jel a billentyűzet jobb alsó sarkában található, a . felett.

A táblázatban egyes játékoknál megjegyzéssel ellátott sorok is vannak. Ezek nem tartoznak az „örökéletre” állításhoz. Hatásuk a következő:

négyzetek (MAJOR BLINK) Eltűnteti a szaladgáló négyzeteket, melyek zavarják a kezdő játékosokat.
pályaszám (ROBIN HOOD) Ez annak jön jól, aki már minden reményt feladott. Ezzel végignézhetheti az összes pályát.
Az energiát állítja energy végtelenítése (BABY BERKS)

készülék elterjedve. Pedig a kereskedelemben kapható, a SKALA-COOP forgalmazza.
Erről az akciónál nem hallottunk, levelét elküldtük a Skála-Coop illetékes vezetőjének.

Bálint Attila, Nyíregyháza,

Gábor Béla köz. 2. IV/19. 4400

7. osztályos általános iskolás tanuló vagyok, és a számítástechnika BASIC nyelvet már nagyjából elsajátítottam. Számítógépm nincs, de ha a 7. osztályos bizonyítványom is jó lesz, akkor kapok egy COMMODORE—64-est és szeretnék minél többet tudni róla, persze nemcsak erről az egy számítógépről, hanem általánosan mind egyiről, és továbbtanulni is ilyen téren szeretnék. Ezért szeretnék segítséget kérni leendő számítógépm megismeréséhez, hogy iradalmat ajánljanak főleg a gépi kódú programozás, de a BASIC nyelven kívül még más programozási nyelv elsajátításához is.

Mindenekelőtt C—64-et termő bizonyítványt kívánok. A géphez nagyon sok könyvet árusít pl. a Novotrade 2C Számítástechnikai Áruház (Bp. XIII., Balzac u. 35. Tel.: 402-954) és a Műszaki Könyvruház (Bp. VI., Liszt F. tér 9. Tel.: 420-353). A C—64-hez kapható szakirodalmat felsorolni túlságosan hosszú lenne, az egyik legutóbb megjelent könyv a Data-Becker—Novotrade kiadásában. Hornig—Trapp—Welter: Tippek és trükkök Commodore 64-eshez. 239 Ft. (A μ M 86/11—12. számában közzétűnk a könyvről rövid ismertetést.)

KOVÁCS GYÖZÖ

μ87 ORSZÁGOS MIKROSZÁMÍTÓGÉPES TALÁLKOZÓ programja

Márc. 19.

Megnyitó és fogadás
— csak a meghívott vendégek részére.

Márc. 20.

Amatőrök napja
— bemutatjuk a magyar informatikai amatőrök eredményeit.

Márc. 21.

Diák nap
— tiétek a μ87!

Márc. 22.

Családi nap
— Mire használható a számítógép otthon?

Márc. 23.

Kulturális és humán nap
— Zene, képzőművészet, irodalom;
Vakok és gyengénlátók, mozgássérültek napja.

Márc. 24.

Társasági nap
— a MTESZ-egyesületek képviselőinek és az NJSZT tagjainak találkozója.

Márc. 25.

Felsőoktatási nap
— a találkozó bezárása.

a Commodore Plus 4-es gép ismertetőjét. A cikk kapcsolatban néhány kérdésem lenne, mégpedig:

a) hozzájuthat-e magánszemély a BAV-on kívül valamilyen formában? (ha lehet, akkor budapesti címen kívül szolnoki, ill. miskolci forgalmazói címeket is kérnék),

b) konkrétan vagy legalább nagyságrendileg milyen áron kerül forgalomba az alapgép és a hozzá illeszthető magnetofon?

c) ha megrendelés útján megvásárolható, hova forduljak igényemmel?

Üzletiökéssel nem foglalkozunk, ezért levelét a Novotrade-hez továbbítottam. (Címük: Budapest XIII., Fürst S. u. 24/26. 1136.) Úgy tudom, hogy a SKALA Computer-S üzletei árulják időnként a Commodore+4 gépet 8000 forint körüli áron.

Pallagi László, Százhalombatta,

Rózsa Ferenc u. 3. 2440

Örömmel olvastam a Mikroszámítógép Magazinban, hogy ismét megrendezésre kerül a Garay pályázat. Különösen az a rész volt számomra kedvező, amely lehetővé tette, mint előfelvételis egyetemista, részt vehetők a pályázaton. Mivel a kiírt gépekhez katonai szolgálatom ideje alatt nem jutok hozzá, egy régebbi — 86 májusi — logikai játékprogramomat adom elbírálásra.

Örülök, hogy részt vesz a Garay pályázaton (nem is rossz ötlet így rövidíteni kedves gimnázium pályázatának az elnevezését). Én azt hiszem, hogy katonai szolgálata alatt is tud számítástechnikával foglalkozni, hiszen a laktányában működnek a μ klubok. Azt javaslom, hogy kérdezze meg a parancsnokait, hogyan tud — ha ilyen épület nélkül — klubfoglalkozást szervezni az alakulat érdekében katonái részére. Sok sikert!

Gulyás András, Pomáz,

Eötvös u. 6. 2013

Az alkalmat felhasználva szeretném kifejezni elismerésemet a lap változatlanul magas színvonalára és érdeklődésre számot tartó cikkei iránt. A számítástechnikával már közelebbi kapcsolatba került, de azt nem hivatászerűen művelő olvasó számára kiválóan alkalmas az ismeretek bővítésére és a kíváncsiság kielégítésére.

Talán az egész oldalas, csak gépi kódban írt listákhoz (játékok) lehetne némi eligazító magyarázatot fűzni. Érdekes lehetne a hazai számítástechnikai folyóiratok, időszakos kiadványok áttekintő bemutatása.

Ez volt a célunk, a szerkesztőség nevében köszönöm, hogy törekvésünket nyugtázza.

Horváth Sándor, Kistélek,

Zrinyi u. 6. 6760

Ne haragudjanak, hogy levelemmel zavarom Önöket. Segítségét szeretnék kérni a lapjuktól. A lényegre térek, vettem a kereskedelemben kapható SELTROW COLOR COMPUTER 200 és a hozzá való 2 db JOYSTICK-et. De nem lehet hozzá kapni játékprogram-kazettát sehol. Még hozzá való magyar nyelvű szakkönyvet sem. Tehát most itt állok tehetetlenül, mert nem értek hozzá. A magyar prospektus, amit kaptam, nem éppen kielégítő.

Mindenekelőtt játékkazettát szeretnék venni, vagy egyáltalán megtudni, hogy hol lehet hozzájutni. Ebben kérem az Önök segítségét. Nagyon kérem, válaszoljanak levelemre (válaszborítékot mellékelek).

Nagyon kérem, legyenek segítségemre, mert már sehova sem tudok fordulni.

Mindenütt azzal fogadnak, sajnos nincs ez a

idő (MAJOR BLINK, TUTTI FRUTTI, ROCKMAN)
 energy (BABY BERKS 2)
 fuel (BAN)
 pajzs (DEFENCE)

„örökenergiára”.
 Hatására az idő nem fogy a játék közben.
 „Örökenergiára” állítás.
 Az üzemanyag fogyasztását akadályozza meg.
 Soha nem fogy el a pajzs.

Egyes játékok különböző néven ismertek. Ezeknél zárójelben a másik név is szerepel.
 Más a helyzet a BMX RACER-nél, ugyanis előfordulhat, hogy az olvasó nem kalózmásolat útján jutott hozzá, és BASIC betöltőrésszel használja. Ebben az esetben a következő a megoldás:
MONITOR
 L "L" 01 (lemezről 08)
 >2360 EA EA EA
 >FF16 CE A1
 >2 01 (botkormány használata esetén 01 helyett 00-t kell irni)
 >21E1 EA (örökenergia)
 X
 SYS8330
 A " " -ben a játék nevének kell állnia, a gépi kódú rész nevének. Nem biztos, hogy ez L jel.

1. táblázat

NÉV	TARTALOM "játék POKE"
DARK TOWER	>1FCA EA EA EA
TIMESLIP	>1404 EA EA
SPECTIPEDE	>1A87 EA EA
BABY BERKS	1A20 00 >2494 00 >1A5E EA (energy végtelenítése)
JETPAC	>1CF0 EA EA 4C F8 1C
(CUTHBERT IN SPC)	>119E életek száma
XZAP	>3665 EA EA 4C 7E 36 >2CFF életek száma
SKRAMBLE	>390B EA EA EA 3A80 életek száma
MAJOR BLINK	>298C 00 >12B2 00 >131D EA EA (idő)
négyszeket:	>1323 EA EA (idő)
EVIL	>1BD1 FF >326A 4C E0 3F EA >3FE0 A6 92 A9

TYCOON TEX	05 95 B8 4C 6E > 3FES 32 >249D EA EA EA EA EA 4C C6 24 >25A3 EA EA >1100 életek száma
GALAXIANS (INVASION)	>19E0 EA EA EA >2669 EA EA EA >2676 EA EA EA >2FAE EA EA EA >2268 EA EA >2105 00 EA >1CBI nehézségi szint (legfeljebb 04)
MAYHEM	>1FD2 00 >2061 00 >121E EA EA EA
STELLAR WARS	>108A EA EA EA >102F életek száma >38E6 EA EA EA EA EA 4C 0A 39 >1117 EA EA EA
SQUIRM	>2375 EA EA EA >261F 00 >12CE 00 >134F EA EA (energy)
ROBIN HOOD pályaszám	>1F50 EA EA EA >1A27 A9 (idő) >354A EA EA EA >1E00 EA EA >17D6 EA EA >1964 EA EA EA >19F8 EA EA EA
ZODIAC	>2D3C EA EA >3AC8 EA EA >1FFC EA EA >3367 EA EA EA
MENEKÜLÉS	>13CB EA EA >3050 EA EA EA >1F2C EA EA >32BA EA EA EA >2B5C EA EA >3C55 EA EA (fuel)
PUNCHY	>3861 EA EA >11A0 EA EA (pajzs) >1C1F A9 04 >32FE A9 05 >2194 EA 2110 EA (idő)
XARGON	TUBA IMRE
SPACE FREEKS	
SPACE SWEEP	
BABY BERKS 2	
TUTTI FRUTTI	
BLAGGER	
WILD WEST	
CUTHBERT IN LAB	
ATLANTIS	
MR PUNIVERSE	
OUT ON A LIMB ZAPPEM	
BIG MAC	
BAN (HALL OF FAME)	
DEFENCE (DEFENDER)	
FIRE ANX DORKS DILEMMA ROCKMAN	

Az ismeretlen C16

Amikor a címbeli állítást megváltoztatandó, e cikkek írásába belefogtam, a C16 belső világról csak elvétve volt fellelhető magyar nyelvű információ. Azóta sokat változott a helyzet. Az clöz szamban ismertett könyv megjelenése után alig egy hónappal, előttem van egy második ugyanerről a témáról, de még ez sem az, amelyekre számítottam.

Tóth Viktor:
A Commodore 16-os belső felépítése (Budapest, 1986. NOVOTRADE, 429 oldal. Ára 99,— Ft.)

E könyvben minden benne van, amit én a jelen sorozatban — eredeti terveim szerint — közlésre szántam. És még sok más. Olyasmí is, amit én nem mertem határozottan leírni.

A BASIC és gépi kód című írások első részében utaltam rá, hogy az általam ismert Commodore gépeknél négy különböző processzortípussal találkoztam. Személyes találkozásról szó sincs, az egyes típusmeghatározásokat az irodalomból vettem. Dr. Úry László Commodore C16 könyvében határozottan 7501-es típusszám szerepel, ezzel szemben Székely Béla kollégám, aki velem egy időben és azonos helyen vette C16-osát, mesélte, hogy ő a gép belsejébe pillantva egyebek között egy 8501 felirátú áramkört látott. Ugyanerről a témáról egy zárójeles mondat Tóth Viktor könyvéből: „(Egyes dokumentációk ettől eltérő — pl. 7501 — típusszámot közölnek; én eddig a nálunk forgalomba került gépeknél csak a 8501-essel találkoztam.)”

A kótet nagy részét itt is a ROM-lista teszi ki. A magyarázatok a nagyobb karakterméret miatt még rövidebbek, mint az előző szamban ismertett könyvben, néhol akadnak a tapasztalatlan olvasót félrevezetőek is, de általában elég jónak mondhatók. Ebben a könyvben nincs keresztreferencia-lista, de nem is hiányzik. Van viszont a rendszer RAM-területéről leírás, amelyben azok az információk is megtalálhatók, melyek a Rendszerváltózik és 1/0 címek című füzetből hiányoznak (lásd az 1986/10. számban).

Az egész könyv — a borítótól eltekintve — a DATA BECKER cég INTERN kötetire emlékeztet, tartalmi felépítését és kivitelezését tekintve egyaránt. A szövegszerkesztővel írt, mátrixnyomatón előállított anyag jól olvasható lenne, de sajnos a nyomdai munka színvonala egyetlen, jó néhány hibás példánnyal találkozott. A bevezető részben a szerző elárastt bennünket a hardver- és szoftverinformációk tömegével, hibát csak elvétve talál, és általában nem túl jelentőseket. (Néhány mutatoba: a BASIC utasításszámok felső

korlátját 65 000-nek írja 64 000 helyett, a fogalommagyarázatoknál az interface szó t „arc-közi”-nek fordítja.) Részletesen ismerteti a gépi kódú programozás elemeit, a KERNAL rutinokat, a monitor utasításait, a TED regiszterképletét, a B/K különleges lehetőségeit és még számos dolgot, amit a cikk leadásáig rendelkezésünkre álló egy hét alatt nem volt időm áttanulmányozni.

Itt találtam meg a SETTMO első tisztességes leírását. A CLALL-hól szólva figyelmezteti az olvasót arra, hogy a rutin csak logikailag zárja le a fájlokat, a háttértárolón a fájl lezárása nem történik meg. Azért emelem éppen ezt ki, mert annak idején én elmulasztottam felhívni a figyelmet erre: egyik olvasóm kifogásolta is.

Összegezve: mindenkinek ajánlom a könyvet, aki a C16-ot nemcsak BASIC-ben akarja programozni. Ez az a kötet, amelyet kezembem tartva azt mondtam magamnak, hogy most már szép lassan befejezhetem a sorozatot, mert a C16 egyáltalán nem ismeretlen többé. Mindez 99 forintért!

A sorozat azonban mégsem fejeződik be. A könyv 66. oldalán röviden olvasható a BASIC-bővítéseknek egy új lehetősége, de a szerző a megoldás felderítését az olvasóra bizza. Erről a témáról a következő számban ír Gnädig Péter, aki megoldotta ezt a problémát.

BARNA LÁSZLÓ

EASY file-től a MASTER 64-ig Adatfeldolgozó programcsomagok Commodore 64-re (Budapest, 1986. LSI ATSZ, 254 oldal. Ára: 185,— Ft.)

A kötet öt adatfeldolgozó programcsomagot közöl módszertani útmutatóval. A programcsomagok általános ismertetését példákon mutatja be, és segítséget nyújt a felhasználók saját feladatainak megoldásához is. Külön fejezet foglalkozik az adatfeldolgozás néhány főbb szempontjával, a lemez- és az adatvédelemmel.

Radnai Rudolf: Mikroprocesszoros berendezések vizsgálata (Budapest, 1986. Műszaki Könyvkiadó, 259 oldal. Ára: 70,— Ft.)

1971-ben az Intel félvezetőgyár előállította az első mikroprocesszort. A cég 4004 típusú áramkörének megjelenését ma már egy új ipari forradalom kezdetének tartjuk.

A mikroprocesszorok megjelenésével a számítástechnika végképp „emberközelbe” került. Felhasználásukkal gyártják az olcsó, nagy teljesítményű személyi számítógépeket és a programozható asztali kalkulátorokat, mikroprocesszorok vezérlik az intelligens terminálokat stb. Mivel alkalmazásuk

által szoftver szinten alakíthatók ki egyedi tulajdonságok, ezért az igen nagy szeriában gyártható hardverelemekből — programozással — egymástól eltérő tulajdonságú rendszerek is előállíthatók.

Nyilvánvaló előnye mellett a mikroprocesszorokra épülő technika együtt jár azzal, hogy a felhasználóknak fel kell készülniük ezeknek a bonyolult digitális eszközöknek a vizsgálatára. A könyv ehhez nyújt segítséget, összefoglalva a mérési alapelveket, bemutatva a különböző mérőműszereket, ismertetve működésüket és kezelésüket.



Pataki Attila—Tallér Ferenc: Fűtési rendszerek számítása személyi számítógéppel (Budapest, 1986. Műszaki Könyvkiadó, 278 oldal. Ára: 158,— Ft.)

A szerzők összeállításukkal segítséget kívánnak nyújtani a családi házak, társasházak fűtési rendszerének számítógépes méretezéséhez és ellenőrzéséhez — a hőszükséglet meghatározásától egészen a rendszer vezérléséig. A könyv az ipari felhasználóknak is hasznos ismereteket ad, de mindazoknak is, akik nem rendelkeznek számítástechnikai és hőtechnikai ismeretekkel. A közzétett programok a Spectrum, a ZX 81 és a Commodore 64-en is futtathatók.

Főbb fejezetek: Hőtechnikai alapfogalmak. Családi házak hőtechnikai méretezése számítógéppel. Kazánok és kazánházak kialakítása személyi számítógépek segítségével. Kazánok, tüzelőberendezések méretezése, ellenőrzése személyi számítógéppel. A fűtési rendszer vezérlésének lehetőségei személyi számítógéppel.

Ada-Winter Péter—Ada-Winter Dávid: A ZX—Spectrum hardverleírás, gépi kódú programozás (Budapest, 1986. Műszaki Könyvkiadó, 358 oldal. Ára: 57,— Ft.)

A könyv a Magyarországon egyik legelterjedtebb személyi számítógéptípus, a Sinclair ZX—Spectrum azon felhasználóihoz szól, akik megismerkedtek gépük

„könnyen hozzáférhető” szolgáltatásaival, BASIC nyelvű programozásával, és mélyebben is bele akarnak nyúlni a rendszerbe. A szerzők ismertetik a Spectrum, mint elektronikai rendszer tulajdonságait, a gép hardverfelépítését és szoftverrendszerét. A könyv alapján az olvasó maga is kísérletezhet saját illesztéssel.

A főbb fejezetek: A ZX—Spectrum rendszerlemelei. A rendszerkivezetések. A ZX-pinger. A Spectrum ROM.

A ZX-függelék a Z80 mikroprocesszor utasításkészletét tartalmazza.

Rucz Lajos: Rutinról rutinra. Bepillantás a Sinclair Spectrum gépi kódú világába (Budapest, 1986. LSI ATSZ, 129 oldal. Ára: 149,— Ft.)

A könyvet a szerző elsősorban azoknak szánja, akik a ZX—Spectrum tulajdonosaként már megismerkedtek a gép kezelésével, a BASIC nyelvvel, és néhány játékprogram lejátszása után szeretnének a számítógéppel kapcsolatban mélyebb ismereteket szerezni.

A logikusan felépített könyv segít eloszlatni mindazoknak az idegenkedését, akik nem tudták eddig rászánni magukat, hogy megtanulják a gépi kódú programozást. Az ismerkedés megkönnyítésére az első fejezet részletes áttekintést ad az alapismeretéről és a Z80 processzor működéséről. A második fejezetben az olvasó választ kap arra a kérdésre, hogy mikor előnyösebb a BASIC helyett a gépi kódot használni. A harmadik rész a felhasználói programokban gyakran előforduló rutinokat ismerteti, így a Beta Basic és a Master Toolkit rutinjait is.

Tokodi Jenő: A Laser mikroszámítógép-család (Budapest, 1986. LSI ATSZ, 144 oldal. Ára: 140,— Ft.)

A kötet a Laser mikroszámítógépek bemutatásán túl általános mikroszámítógépes ismereteket is nyújt. Az egyes témakörök feldolgozásánál mindig az általános elvekből indul ki, és az általánosan alkalmazható módszereket tárgyalja az utasítások és eljárások kapcsán. A Laser mikroszámítógép adta lehetőséget kihasználva a könyv anyaga alkalmas arra, hogy az olvasók áttekinthessék a „home computer” kategóriájú gépek jellemzőit, továbbá a beépített BASIC microsoft sajátosságai miatt megismerkedjenek az IBM PC/XT és az Apple II gépek BASIC programozásával.

A függelékben az olvasó kapcsolási rajzokat talál, melyek alapján házilag, itthon kapható alkatrészekből memóriabővítőt, botkormányt, kazettás egységet építhet.

Szuper metró

Automatikus vonatvezetési rendszert állítanak munkába a budapesti metró észak-déli, vagyis a 3-as, kék vonalán. Az ún. Interlektív vonatszabályozási rendszert a BKV és a francia Matra konzern szakemberei közösen dolgozták ki. Lényege: a metrószervevények haladása állomástól állomásig, az indulástól a következő megállóig automatikus, nincs szükség a motorkocs vezetőjének beavatkozására. Bármilyen veszélyhelyzetben — amiről a biztosító berendezésben keresztül kapott információból értesül az automatika — megállítja a vonatot. A mikroprocesszor által vezérelt fedélzeti rendszert különféle érzékelők tájékoztatják az éppen aktuális helyzetről, s ez kiváltja a beavatkozást.

Az új megoldás előnyei közé sorolják, hogy emelkedik a szervevények szállítóképessége, nő az utazási kényelem. Növekszik a forgalom biztonsága, és a becslések szerint 10-20 százalékos energiamegtakarítás érhető el. Ezenkívül a metrókocsok személyzetét egy főre lehet csökkenteni.

Diétázóknak, fogyózóknak

Akiknek szigorúan be kell tartaniuk diétájukat, jól tudják, milyen nehéz úgy összeállítani a napi étrendet, hogy az változatos is legyen, és megfelelő mennyiségben tartalmazzon szénhidrátot, káliort, különféle vitaminokat, ásványi sókat. A Diabéteszes Társaság által Commodore 64-re kidolgozott program ezen a gondon próbál segíteni. Továbbra is az orvos határozza meg, hogy a páciens mennyi szénhidrátot, káliort fogyaszthat egy nap. A diabéteszeseknek például hatszor kell enniük naponta, és az sem mindegy, hogy egy étkezésre mennyi kalória, szénhidrát jut. Ezen adatok birtokában lehet összeállítani a napi étrendet. A számítógép sora veszi az étkezéseket, s közben menüből lehet választani, mit is akarunk fogyasztani.

A program jó szolgálatot tehet a kórházaknak is: mentesítheti az orvosokat, diétásnővéreket a betegek étrendje összeállításának rutinmunkájától. Már készíti a programnak azt a változatát is, mely egész-

séges emberek, sportolók és a fogyni kívánók számára ad étrendvariánsokat.

Itthon: új C16 — IBM PC illesztő

Már sokan vállalkoznak a C64-nek az IBM PC-vel való összekapcsolására, de a C16 illesztése újdonságszámba megy. Elsősorban az IBM PC-vel kompatibilis nagyobb számítógépeknek az egyszerűbb adatátviteli és szövegszerkesztési munkák alól történő tehermentesítése céljából fejlesztette ki ezt a kapcsolatot a Data Manager Kiszövetkezet. Ez a termék három programból és egy illesztőből áll.

A Text nevű szövegszerkesztő program segítségével a C16-on lehet szövegeket bevenni vagy módosítani. Az így előállított szöveg kezeltára vagy hajlékonylemezről tárolható ki. A C16-on működő adatátviteli programmal lehetőség van az állományok illesztőre küldésére, illetve az illesztőn érkező állományok fogadására. Az IBM PC kompatibilis gépen működő másik program kezeli az illesztőn érkező vagy átmenő jeleket.

A mintegy 30 ezer forintos illesztőnek már elkészült az idén egyre jobban terjedő Commodore Plus/4 és IBM PC közötti kapcsolatot biztosító változata is.

Átszakította a vonalat: első a magyar vonalkódolvasó!

A korszerű áruházi rendszerekben ma már világszerte elterjedt a vonalkódos árujelzés. A számítógépekhez csatlakoztatott vonalkódolvasó ceruzával dolgozó gyors pénztárosok látványra még hétköznapi dolgoknak számít a Skála-Metroról áruházban is. Ezek az olvasók azonban még tökéletes importból származnak.

A szocialista országok első vonalkódolvasó ceruzáját a hazai Szint Kiszövetkezet fejlesztette ki. Az őszel nagy sikerrel bemutatott termék alkalmas mind a hagyományos, nyomdai úton előállított, mind pedig a hőnyomatós papírra készített vonalkódok olvasására. Szabványos RS232 illesztőjén keresztül 300—9600 baud sebességgel tudja továbbítani a

jeleket, s az illesztőn lévő tesztolesztés feszültségénél alkalmazható. Elkészült a vonalkódolvasónak egy IBM PC bilentyűzetfeletle is, amely minden szoftver nélkül bármilyen meglévő rendszerhez alkalmazható, amely eddig bilentyűzetről fogadta az adatokat.

Az olvasó intelligenciáját egy Z80 típusú mikroprocesszor biztosítja, s egy 64 kb-ot méretű szoftver szolgál a vonalkódok felismerésére. A készülék több vonalkódtípus olvasására alkalmas, így például a Bar-kód, az EAN (európai) és az UPC (USA-beli) szabvány szerintiekre is.

A vonalkódolvasó kifejlesztésénél fontos szempont volt a csaknem százszázalékos olvasási biztonság mellett a ceruza ütésálló kialakítása. A tapasztalatok szerint ugyanis az áruházakban alkalmazott vonalkódolvasók legtöbbször a leeseűk következtében romlanak el. A hazai változatnál magában a ceruzában csak a legszükségesebb és újtésre nem érkező elemeket helyezték el, a kényesebb alkatrészek az illesztőben kaptak helyet.

Már folyik a vonalkódolvasó továbbfejlesztése. Készül a pástázó olvasósugarú változat. Ez esetben a pénztáros csak ráhelyezi az olvasófejet a vonalkódra, és az egy mozgó sugárral „letapogatja” a vonalkódot. A másik fejlesztési irány az igen vékony, 7 mm átmérőjű ceruza kifejlesztése.

A mindössze 20 márka tökéletes anyaganyagot tartalmazó vonalkódolvasó hazai ára ötven ezer forint alatt van. Összehasonlítással meg kell jegyezni, hogy az NSZK-ban jelenleg forgalomban lévő vonalkódolvasó ceruzák ára 1700 márkánál kezdődik.

Osztrák Szilícium-völgy?

Salzburg körzetében a Siemens cég leányvállalata szoftverfejlesztő központot létesít. A telephely megválasztását az indokolta, hogy a salzburgi egyetemem új számítástechnikai intézetet alapítottak. A központ munkájának megindulását a salzburgi városi tanács 400 ezer schillinggel támogatja.

Az első szakaszban a szoftvertermelés 10-20 munkahelyre kezdődik. A továbbiakban a kutatási és oktatási tevékeny-

ség felfutásától függően fogják a központ létszámát növelni. Maga az építkezés fél évvel előtt kezdődött meg.

Hazai videoprocesszorok

A Datacoop Kiszövetkezet gyors képfeldolgozó rendszert fejlesztett ki, mely alkalmas tetszőleges forrásból érkező videójel digitalizálására és a digitalizált képek tárolására memóriában. A bemenő, folyamatosan digitalizált kép és a memóriában tárolt kép is megjeleníthető elenőző képernyőn.

A beépített mikroszámítógép lehetővé teszi a tárolt kép (vagy képek) feldolgozását. Ez leggyorsabb esetben a tárolt kép mátrixnyomatón való kinyomtatását jelenti. A csatlakoztatott megjelenítő és bilentyűzet segítségével a mikrogép számára vezérlőutasítások adhatók, így a kinyomtatott kép alá feliratok írhatók, sőt a képtartalom is módosítható.

Számtalan — egészségügyi, műszaki, építőipari — alkalmazási területre közül a textil- és a ruhaipari felhasználási lehetőségeket vizsgáljuk:

— Rajzok, textilminták, szabásminták tervezése, módosítása, optimalizálása;

— A megmunkáló gépen lévő álló vagy mozgó anyagoknak, illetve munkadaraboknak a gyártás folyamatában, annak megskatítása nélkül történő minőség-ellenőzése;

— Automatizálási, szabályozástechnikai alkalmazás (optikai vezérlés, szabályozás);

— Gyors folyamatok „sorozatfényképezése”, a képek kiértékelése;

— Káros részek, a rendelkezéses működés érzékelése;

— Infrakamerával a káros helyi melegek kimutatása.

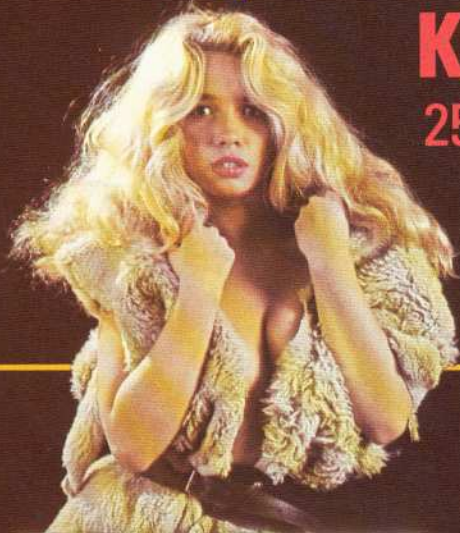
A különböző alkalmazásokhoz természetesen más-más rendezésképzítés és szoftverrelátás tartozik; a fejlesztés folyamatos.



A rovatvezető — a képfeldolgozó által elemi pontjaira bontva

VÁRJUK ÖNÖKET A μ '87

KIÁLLÍTÁS
25. PAVILONJÁBAN!



**KERESSÉK
A**



EMBLÉMÁT!



JÖN!

ATARI

800 XL

