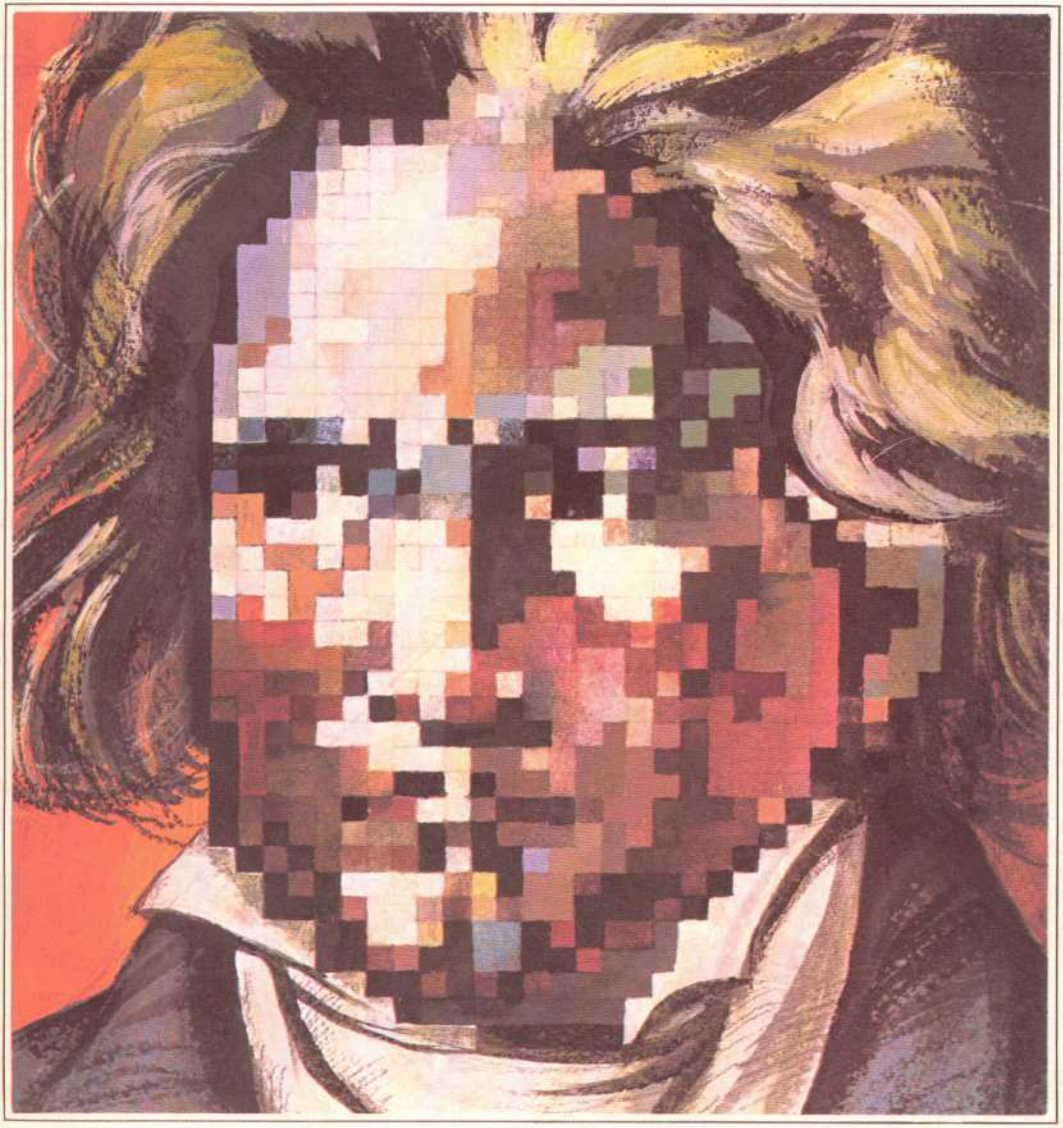
Ára: 30 Ft

μ mikro számítógép magazin



SYMPHONY

1987/9

NOVOTRADE

Hazai termék – külföldi referencia

A BECKERbase PC adatbázis-kezelő program IBM PC XT/AT, illetve ezekkel kompatibilis számítógépekre készült. Az NSZK-ban rövid idő alatt óriási sikert aratott: 6000 példányban kelt el.

ÚJ!

A BECKERbase legfontosabb tulajdonságai:

- gyors fájlkezelés,
- egyszerű fájlmegadás és létrehozás,
- kényelmes segédeszköz a képernyőkezeléshez és összesített táblázatok kinyomtatásához,
- korlátlan hálós adatbázisszerkezet-definiálási lehetőség,
- logikai struktúra egyszerű létrehozása egy könnyen kezelhető, az adatbázis felépítését leíró nyelv (DDL) segítségével,
- programnyelv a felhasználói programok kezeléséhez.

A programnyelv (TDL) könnyen elsajátítható, utasításai leginkább a BASIC nyelvéhez hasonlítanak.

A BECKERbase programlez ára, teljes dokumentációval: 6950,— Ft.
Megvásárolható: 2C Számítástechnikai Áruház
1138 Balzac u. 35. Tel.: 402-954

A BECKERbase program elsajátítását lehetővé tesszük legjobb szakembereink közreműködésével 1 hetes intenzív tanfolyam keretében.

A programlemez + a tanfolyam ára: 20 000 Ft.

A tanfolyam helye:

a HOTEL REGE
Kártyaterme
Budapest II., Pálos u. 2.

Várjuk jelentkezésüket a 122-099 és 122-095-ös telefonszámon.

ÚJ!

Ügyintéző: Bakó Lászlóné
Novotrade Rt.
Budapest, Kresz Géza u. 14. 1136

μ mikro számítógép magazin

5. ÉVFOLYAM
1987/9. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel 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)

**Címképünk:
Ramocsai Imri munkája**

Felelős szerkesztő:
Könyves Tóth Pál

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

Levélcím:
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 hírlapkézbesítő 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.
86—0253

Szikra Lapnyomda
Budapest (87-1147)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

„. . . a cat is a house,
if a mouse is a cat,
but a town is no house,
*if a town is a hat."**

Szófia 1987.

Többszörösen is elnézést kell kérnem a kedves olvasóktól. Először talán ezért a kezdő „bolond" versikéért, amelyet — mentségemre legyen mondva — egy számítógép írt. Nagy sikert aratott a „költeménnyel" Volker Claus, aki a májusi szófiai „Gyermekek az informatika korában" c. nemzetközi oktatási konferencián ilyen és más hasonló példákkal mutatta be, hogy milyen feladatokkal hozták közel az informatikát a középiskolás gyerekekhez. Ugye már nem is olyan bolond ez a versike?!

Kérem, hogy a kedves olvasó azt se hányja a szememre, hogy ez a beszámoló egyáltalán nem lesz objektív, hiszen a magam elfogultságait e tárgykörben nem tagadom meg, és ajánlani szeretném mindenki figyelmébe ezt az egyre komolyabb és nagyon hamar jó hírűvé vált nemzetközi tanácskozást.

A konferencia rendezői: a Bolgár Akadémia, a Minisztertanácshoz rendelt tudományos és fejlesztési bizottságok, a Ljudmilla Zsivkova Alapítvány, valamint az UNESCO, AZ IIASA, a WHO és az IFIP támogatásával válogatott oktatási szakembereket nyertek meg előadóként, akik egyben a konferencia színvonalát is meghatározták. Bolgár barátainkat csak csodálni lehet, hogy mennyit áldoztak a találkozó sikeréért nem csak azzal, hogy a meghívott előadók költségeit vállalták, de azzal is, hogy a „Ljudmilla Zsivkova" nemzeti kultúrpalotában méltó helyet biztosítottak a nyugodt és kényelmes munkához.

A konferencia motorja, szervezője, vezetője és irányítója *Blagowest Sendov* akadémikus volt, megszállottja a gyerekek informatikai oktatásának. A meghívott előadók listáját nem szeretném felsorolni, de talán egy kis ízelítő nem árt ebből a nem akármilyen névsorból, mint Seymourt Pappert (lásd a képet), a LOGO nyelv „atyja"; aki egyébként elfogadta meghívásomat a jövő évi μ '88-ra, Sylvia Charp, az amerikai informatikai szövetség korábbi elnöke, Willfried Brauer, müncheni professzor, Peter Bollerslev, Dániában a szakma egyik felelős vezetője, prof. Shoshi Shiba Japánból, a Tsukuba egyetemről, prof. A. P. Ershov Novoszibirszkből, prof. B. Nag, a bombayi egyetem rektora, N. Rusby Angliából, és a már emlegetett Volker Claus az NSZK-ból, hogy csak a számomra legtöbbet jelentő előadások szerzőit soroljam fel.

A rendezők kb. 600 résztvevőt regisztráltak, akiknek a nagyobbik része külföldről érkezett. A konferenciát megelőzően több párhuzamos rendezvényre is sor került, így az EDIT '87-re, a számítógépes oktatással foglalkozó magazinok és újságok szerkesztőinek találkozójára, a már említett IFIP 3. sz. műszaki oktatási bizottsága (TC 3) 1. sz. munkacsoportjának (WG 3.1) munkakonferenciájára, a jövő évi UNESCO oktatási konferencia programbizottságának ülésére, valamint a fiatalok nyílt nemzeti programozói versenyére, amelyiken bolgár, román NSZK versenyzők mellett egy négyfős magyar csapat is indult.

Az idei „Gyermekek az informatika korában" konferencián az előadók feladata elsősorban az volt, hogy megmutassák a lehetőséget a kreativitásra, az innovációra és új cselekvésekre az informatika által.

A programbizottság felhívása az előadókat arra kérte, hogy elsősorban az iskolai informatikával foglalkozzanak, és annak ne csak az ismert pozitív vonásait vizsgálják, de azt is, hogy a számítógép és a számítástechnika nem veti-e vissza a gyerekek kreativitását, és nem csökkenti-e egyéb területen vagy más tantárgyakban az aktivitásukat.

Nem mondom, hogy — különösen a hozzászólásokban — nem hangzottak el számomra negatívnak ható vélemények, pl. az egyik résztvevő igen csak keményen érvelt és példákat is hozott a maga gyakorlatából, hogy a túl korán kezdett számítógépes foglalkozás elfordítja a gyereket a hagyományos tanulástól, „elfelejt" megtanulni pl. fejben vagy írásban számolni. Ezen azután persze — már a WG 3.1-es előkonferencián is — vita alakult ki, hogy kell-e egyáltalán az informatika korában fejben illetve írásban számolni. Nem lenne-e elég pl. csak becslésre és nem pontos számolásra tanítani a gyerekeket, hogy a számoló- vagy számítógépen kapott eredmények helyességét (mint ahogyan azt annak idején mi is a műegyetemen a logarléces számolásnál tettük) meg tudják becsülni.

Erről persze az is eszembe jut, hogy nem is egy előadó elmondta, hogy a gépesítést — esetleg már az óvodában — a számológépekkel (calculator) kezdik. Egészen biztos szerettem volna lenni, hogy jól értettem-e a dolgot, tehát hogy még ma is használják a zsebszámológépeket a tanításban, ezért ezt az egyik amerikai előadótól meg is kérdeztem. Úgy mondta a „természetesen"-t, hogy azonnal azokra az itthoni iskolákra kellett gondolnom, ahol azt láttam, hogy az 5-6 évvel ezelőtt megvett egy-két tucat számológép valamelyik szekrény mélyén hever, mert a tanárok is és a diákok is a HT-ket, a PRIMO-kat meg a többi számítógépet nyúzzák, a számológépeket pedig nem használják. Az egyik előadó válaszában szép logikusan ki is fejtette, hogy a gyerekeket — főleg a kicsiket — a logikus gondolkodásra, a probléma numerikus megfogalmazására, illette algoritmizálásra kell tanítani. Persze a feladatot azután végig kell számolni, hogy az algoritmus helyességéről a tanuló numerikus számítással bizonyosodjék meg. Kézzel, papíron tenni ugyanezt túl sok időbe kerül, és főleg: nem a számolás a cél! A gyerekek tulajdonképpen akármit használhatnának, amivel a „rabszolgamunka" ideje csökkenthető, teljesen logikus, hogy használják a számológépeket is. A számológépekkel kapcsolatban egy másik előadó még továbbment. Elmondta, hogy megjelent a piacon egy olyan számológép, amellyel bármilyen bonyolult integrálokat kb. úgy lehet megoldani, mint az egyszerű számológépeken a számtani műveleteket. Azt fejtegette, hogy ezek után kell-e vagy szabad-e tanítani pl. az integrálási szabályokat, szabad-e papíron integrálási példákat megoldani? Ez sem akármilyen kérdés, érdemes rajta elgondolkozni.

Ehhez kapcsolódik, hogy a TC 3 WG 3.1 munkakonferencián, amelynek az volt a címe, hogy „Az informatika és a matematika tanítása", nagyon sok előadó foglalkozott a kérdéssel: a számológép és a számítógép széles körű használata milyen mértékben változtatja meg a matematika tanítását? Annak ellenére, hogy a számítógép az információfeldolgozás eszköze, mégis csak egy olyan gép, amivel számolni is lehet, és nem is kis hatékonysággal. Ráadásul a tanár rendelkezésére áll egy sor rendkívül hatékony szoftver eszköz is (pl. spreadsheet vagy magyarul számolótábla), amivel a tanuló nem csak számolni tud, de pl. az algoritmizálást — és így általában a feladatmegoldást — is könnyebben tanulhatja meg. Nagyon sok tanárnak az volt a véleménye, hogy tragikus félreértés a számítógépet és mindent, ami hozzá tartozik a matematika hagyományos módon való oktatására használni, mint pl. egy elektronikus fekete táblát vagy pedig mint egy számítógépes távtanulási eszközt (CBT — Computer Based Training). Ha egy tanár csak ennyire képes, akkor a számítógépet akár ne is használja. Ahogy annak idején — ha jól emlékszem Neumann János is felhívta erre a figyelmet — a számítógép megjelenésével olyan feladatok is számíthatóvá váltak, amelyekre annak előtte nem volt lehetőség, így most a matematikaoktatásban is meg kell keresni a matematika azon speciális fejezeteit, amelyeket most már azért lehet oktatni, mert van számítógép. Elhangzottak példák is, pl. bonyolult számtani és mértani sorokkal való manipulációk, kódrendszerek és azok megfejtésének matematikája, általában olyan stúdiumok, amelyekben az algoritmus megkeresése a fő feladat, illetve az algoritmus helyességének numerikus számítással való bizonyítása.

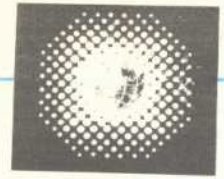
Azt hiszem, itt kell megemlítenem azt is, hogy egyöntetű volt a vélemény: a számítógép sem az elemi, sem a középiskolában, sem pedig az egyetemeken *nem* arra való, hogy programozást oktassanak, nem egy tudós mivelője az informatikának szenvedélyesen ismételgette azt a véleményét, hogy óriási tévedés azt hinni, hogy az egyetemi hallgatók csak akkor tudnak bánni a számítógépekkel, ha legalább a programozás alapjait megismerik. Amelyik egyetemen ez a gyakorlat alakul ki, ott az alkalmazói szoftverben való elmaradottságukat igyekeznek a programozásoktatás erőltetésével takargatni. (A vélemény sokkal markánsabban hangzott el . . .)

Ez a konferencia is, mint a két évvel ezelőtti is a LOGO nyelv oktatási alkalmazásának további erősödését bizonyította. Az alaphangot Seymourt Pappert adta meg, akinek az előadása — főleg a LOGO professzorainak — volt az igazi csemege. Vele töltöttem egy estét, ami után elgondolkoztam: ha pedagógiai szempontból — főleg a kisebb — gyerekek logikai készségének fejlesztésére a világ minden részén a LOGO ilyen kiváló eszköz, akkor nálunk Magyarország-

* *A vers nyers fordítása:*
„. . . a macska egy ház,
ha az egér egy macska,
de egy város nem egy ház,
ha a város egy kalap."

Ez a vers angolul jól hangzik, a program nyilvánvalóan a szavakat csak formálisan vizsgálta, arra ügyelt, hogy a főnevek, az igék, az előljárók, a névelők a helyükön legyenek, és miután versről van szó, a sorok párosával rímeljenek. A szótárból a gép a szótagszámra is ügyelve válogatta ki az egyes szavakat és írta meg a „verset".

(Ismét egy útijelentés)



SECOND INTERNATIONAL CONFERENCE
AND EXHIBITION ON
CHILDREN IN THE INFORMATION AGE:
OPPORTUNITIES FOR CREATIVITY, INNOVATION AND NEW ACTIVITIES
National Palace of Culture, Lyudmila Zhivkova
Sofia-Bulgaria
19 - 23 May, 1987

gon miért nem az? Most, amikor a hazai általános iskolákban már megkezdődött a számítógépek kiosztása, vajon miért nem foglalkozunk a LOGO széles körű megismertetésével és alkalmazásával? Sajnos nem vagyok LOGO-specialista, de azt nagyjából megértettem, hogy a LOGO tulajdonképpen nem csak a magas szintű programozási nyelvek egyike, hanem egyfajta logikus gondolkodásmód megteremtését segítő eszköz, amelyet már egészen kis gyerekek is képesek megismerni és igen bonyolult feladatok megoldására alkalmazni. A LOGO megkönnyíti a gyerekeknek a számítógépnek mint eszköznek a megismerését, a grafikai lehetőségeivel egyszerűbbé teszi a matematika megértését, világossá a technológizált programozást és könnyűvé más magas szintű programozási nyelvek megértését. A LOGO-szakemberek azt mondják, hogy a LOGO a legalkalmasabb eszköz más diszciplinák, pl. fizika, zene és idegen nyelvek tanítására is. Azt remélem, ha Seymourt Pappert jövőre eljön a μ '88-ra, ott találkozni fog tanárokkal és diákokkal, és akkor talán a LOGO-t is sikerül népszerűsíteni, majd alkalmazását főleg az általános iskolákban elterjeszteni. (Lehet, hogy üres kapukat döngetek, bár csak így lenne!)

Egy-egy előadás kapcsán többször is szóba került, hogy milyen hardvert, azaz számítógépeket használjanak az iskolákban, ezen nagyon jóízűeket lehetett vitatkozni még a kávészünetekben is. Persze abban maradtunk, hogy „olyat, amelyik elég nagy kapacitású tárolóval (lemezzel) rendelkezik, és amelyikhez nagyon sok és jó alkalmazói szoftvert lehet kapni". Több hozzászóló is elmondta, hogy az iskola-számítógép projektek irányítói többnyire a mennyiségi szemlélet hibájába esnek, arra törekednek, hogy minél több és olcsó számítógép legyen az iskolákban, ezért azután elhanyagolják a szoftverellátást, nincs határozott elképzelésük arra vonatkozóan, hogy az iskoláknak mire is volnának jók a gépek, pedig az ilyen módon való előregondolkodás az iskolai alkalmazás esetében mindennél fontosabb.

A fejlett ipari országokban egyértelműen az IBM—PC-re és a klónokra szavaznak, az ennél kisebb kategóriát meghagyják otthoni számítógépnek (home computer vagy FAMICOM). Többek, főleg a japánok mondták, hogy ma még nem olyan biztos az sem, hogy a 8 bites gépek házi számítógépekként tovább élnek, meglehetősen nagy a valószínűsége annak is, hogy rövidesen csak 16 bites gépeket vásárolnak még házi használatra is.

Több előadás is foglalkozott az oktatási hálózatokkal, azaz a hálózatok lehetséges szerepével az oktatásban. A hálózat jelentősége abban van — szinte teljesen azonos volt mindenkinek erről a véleménye —, hogy a tanulókat az *adat* értékére, jelentőségére, az adatokkal való manipulációkra (tárolás, visszakeresés, statisztikai számítások stb.) már serdülő korukban megtanítja. Ha a nem létező hazai adatbankokra gondolok, akkor ennek az oktatásnak a jelentősége talán már a mi generációnk szempontjából is óriási.

Többször is szóba került, informatikát kell-e tanítani az iskolában és ha kell, akkor mit, és ha nem kell, akkor hogyan lehet a gyerekeket az informatikával megismertetni. Saját statisztikám szerint jóval többen ellenzik az informatikának, mint tantárgynak a tanítását, mint ahányan támogatják (én is ellenzem). A vita inkább azon folyik, hogy még a támogatók sem tudják megmondani, hogy mit is kellene tanítani. Ilyeneket mondanak, hogy kevés hardverismeretet (mi legyen az a kevés?), szinte semmi programozást, problémamegoldási technológiát, algoritmizálást — mondják —, hiszen az ötödik generációs gépek alkalmazásához ezt kell tudni és nem a programozást. Az egyik matematikatanár véleménye: az utóbbi kettő az nem informatika, az matematika! Egy másik felszólaló szerint az a jó, ha például egy középiskolába elvisznek egy tucat számítógépet, jó sok alkalmazói programot, ezeket megmutatják a tanároknak és a diákoknak is. Következményként általában rövid idő alatt nagyon sok diák és néhány tanár is rákap a gépekre, és kialakul egy meglehetősen magas szintű számítógépes kultúra. A szoftverrel jól felszerelt gépek hatása — mondta a felszólaló — sokkal jobb eredményt hoz, mintha egy kötelező tantárgyat vezettek volna be. Megjegyzem, hogy az OPI is végzett hasonló kísérletet pl. a szekszárdi Garay Gimnáziumban, hasonló módon pozitív eredménnyel egy sokkal gyengébb számítógépes (HT 1080) és egy még rosszabb szoftverkörnyezetben.

Számos előadás foglalkozott a mesterséges intelligencia eszközeinek alkalmazásával. Miután nem vagyok a téma szakértője, csak a benyomásaimat próbálom meg összefoglalni. Miután az utóbbi három évben több hasonló nemzetközi tanácskozáson volt alkalmam részt venni, úgy látom, hogy egyre többen és egyre intenzívebben foglalkoznak intelligens rendszerek létrehozásával. Én azt hiszem, hogy főleg azért, mert a tanítás—tanulás olyan bonyolult, és talán azt is megkockáztatom, hogy sok vonatkozásában (pl. tudáshiányok megtalálásának módszere, a tanulás folyamata, az ember—számítógép kommunikáció stb.) nem is eléggé feltárt folyamat, amelyet csak számítógépekkel és intelligens szoftverrel lehet „megközelíteni". Több előadás szólt az intelligens konzultációs rendszerekről (Intelligent Tutoring Systems), példákon mutatták be az első eredményeket, amelyek — ezt az egyik előadó is megerősítette — biztató kísérleteknek tekinthetők, ma még az alkalmazás nem tart ott, hogy a gyakorlatban jól használható eszközt adhatna a pedagógusok kezébe.

Érdekes beszélgetés bontakozott ki arról, hogy van-e „új műveltség" — angolul „New-Literacy" illetve „új műveletlenség"?

Sokan értettek egyet azzal a véleménnyel, hogy műveletlennek, de legalábbis „új műveletlennek" (New Non—Literate Person) tekinthető az, aki a feladatainak, problémáinak a megoldásához nem használja föl (mert nem ismeri!) az információtechnológia eszközeit.

Talán itt kellene elmesélnem a konferencia egyik derűs pillanatát, az egyik vitadélutánt. A rendezők három témát adtak a résztvevőknek, és 2-2 személyt kértek fel arra, hogy az adott álláspontot védjék meg illetve támadják. Az én felkérésem úgy szólt, hogy védjem meg azt a véleményt — egy bolgár kollégával együtt —, hogy „a számítógépek csökkentik az emberi kapcsolatokat." Tőlem telhetően igyekeztem minden olyan érvet emlékezetemből előrángatni, amelyet néha az itthoni „számítógép-ellenségek" szoktak hangoztatni, nagyon sokszor velem is vitatkozva. (A számítógép elmagányosítja a gyereket, megrontja a szülők és a gyerekek viszonyát, rontja a tanulmányi eredményt és így konfliktust okoz a diákok és a tanárok között stb., bizonyára mindenki ismeri ezeket a vitákat.) Minden rábeszélőkészségemet igénybe vettem, hogy ellenfelemet (S. Charp és B. Nag) véleményem helyességéről meggyőzzem, annyira, hogy még egy kiáltványt is felolvastam — „. . . zárjuk ki az életünkből a számítógépeket!. . ." — a vita végén. Érvelésem végül olyan jól sikerült, hogy a befejezés után odajött hozzám az egyik hallgató és felháborodva mondta: „. . . igazán nem értem, hogyan lehet egy *ilyen* ember Magyarországon a Számítógéptudományi Társaság alelnöke!"

Talán a sok távolabbi téma közül a video—számítógép kombinációról, mint oktatási eszközről érdemes még néhány szót mondani. A nemzetközi statisztikák azt mutatják, hogy mind a videoszalagos, mind pedig a videolemezes technikát — kombinálva a számítógéppel — egyre inkább alkalmazzák mind a tanításban, mind pedig önálló tanulást segítő eszközként, kihasználva a kitűnő videofelvételt és a számítógép interaktivitásának kombinálását. Ezen eszközök — elsősorban a videolemez — elterjedésének ma még talán a legnagyobb akadálya, hogy a lemez előállítása meglehetősen drága, legalább ezer lemezt kell készíteni, hogy egy lemez ára megközelítse a videoszalagét. De éppen az oktatásban lehet ilyen magas darabszámokat elérni, még egy olyan kis országban is, mint Magyarország, ha sikerül olyan „tantárgyat" találni, (ilyen nyilvánvalóan van!) aminek a tanításához a jó minőségű képfelvétel és a számítógéppel való kommunikáció egyaránt szükséges. Az a döntő szempont a kiválasztásnál, hogy ezt a diszciplinát lehetőleg nagyon sokan tanulják, és így legalább ezer lemezt el lehessen adni.

Nem lehet ma már egy nemzetközi oktatási konferenciáról beszámolni úgy, hogy ne essen szó a számítógépi tananyag készítéséről (courseware). Szófiában is sokszor került szóba a probléma, nevezetesen, hogy az oktatóprogramokat készítő szakemberek kapjanak megfelelő szoftvertámogatást a számítógépekhez, ne kelljen programozói segítséget kérni az oktatóprogramok megírásához. Egyértelműen látszik, hogy a számítógépek „bevonulása" az iskolákba együtt jár a szerzői rendszerek — authoring systems — (AUTOOL, Ten CORE stb.) széles körű elterjedésével. Megjelent egy (nekem) új irányzat is, nevezetesen a témaorientált szerzői rendszer, pl. ilyen a GIGO, amellyel nyelvtanító programokat lehet írni. Én azt remélem, hogy a hazai oktatásügy — kihasználva a nagyobb teljesítményű gépeknél (IBM-klónok) várható árcsökkenést, hamarosan megváltoztatja a jelenlegi számítógépbeszerzési politikát, és nálunk is az egyre nagyobb teljesítményű gépek jelennek meg az oktatásban. Azt is hiszem, hogy az iskolákban is hamarosan használatba veszik a videotex hálózatra kapcsolt terminálokat, és így rövid időn belül valamennyi oktatási program már szerzői rendszer támogatásával fog készülni. Azért tartom ezt a váltást fontosnak, mert akkor megszűnik a programozók monopol helyzete az oktatási programok készítésében, ehelyett a megfelelő pedagógiai, módszertani és szakmai ismeretekkel rendelkező szaktanárok fogják előállítani a nemzetközi előírásoknak (Plato szabvány) is megfelelő oktatóprogramokat.

Nagyon sokat lehetne még írni erről a szófiai tíz napról, a konferencián elhangzott előadásokról, távtanulási rendszerekről, az angol Open Teach rendszerről, a multimédia technológiákról, a számítógépes nyelvtanításról, azokról a beszámolókról, amelyek az egyes országokban folyó informatika-tanulási kísérletekről szóltak.

Én azt hiszem, hogy ez a beszámoló már így is túl hosszúra sikerült, remélem, hogy nem a tartalom rovására. El nem titkolt célom, hogy felhívjam erre a színvonalas rendezvényre a pedagógusok figyelmét, a „Gyermekek az informatika korában" konferencia ugyanis elsősorban a pedagógusok konferenciája, igen kiváló és távolságban is közeli lehetőség az oktatási tapasztalatok kicserélésére. Azt kívánom, hogy 1989-ben sokan éljenek vele.

KOVÁCS GYŐZŐ

TANULNI, TANULNI, TANULNI!

Már az első tantárgyi oktatóprogramok elkészülte után kitűnt, hogy egy jó szoftver mind az új anyag elsajátítása, mind a gyakorlás/bevésés során hasznos segítőtárs lehet tanárnak és tanulónak egyaránt. Sok esetben az ismeretek megszerzése (átadása) a pedagógus jelenlétét sem igényli.

Mára a kedvező vélemények igencsak elszaporodtak. Nézzük ennek hátterét, indokait — így derül ki, hogy divatról vagy valódi értékekről van-e szó.

Programok sokasága tartalmaz ún. számonkérő részt, ahol a tanuló maga ellenőrizheti, milyen mértékben sajátította el az anyagot, így az egyéni tanulásnál az eredmény csak a delikvens szívósságától függ — a gép válasza objektív. — Így könnyű! — ma már jogosan mondhatja egyik diák a másiknak. Eddig senkinek sem volt igazán megbízható eszköz a kezében arra nézve, hogy ha megmérettetésre kerülne a sor, hogyan minősülne a tudása. Pedig az ellenőrzés lehetőségére mindenkiben természetes igény él.

Eddig előnyben voltak azok — és szép számmal akadtak ilyenek —, akik elég pontosan tudták, hányszor kell elolvasniuk egy új anyagot, hány példát kell megoldaniuk egy adott témakörben, vagy hány mondatot kell gyakorlásként lefordítaniuk, hogy másnap ötösre feleljenek, jól sikerüljön a felvételi vagy nyelvvizsga. Most már ez az előny behozható!

Akik nem ismerik ilyen jól saját képességeiket, azoknak sem kell segítségért folyamodniuk testvérhez, szülőhöz, kollégiumi szobatárshoz, magántanárhoz — a segítség nincs is mindig kéznél (a tanár nem ér rá, rendszeresen nem fizethető meg stb.). Nem mondható, hogy bárki számára azonnal elérhető, de mégis realitás: forduljon tehát a személyi számítógéphez!

A bizalom, a megbarátkozás félreismerhetetlen jele az is, hogy az egyébként korlátlan felhasználási területű kisgépek iskolaszámítógép elnevezése honosodott meg a hazai gyakorlatban. Elfogadtuk és befogadtuk tehát ezt az oktatást-okulást segítő eszközt, amelyből százezernél is több van már állami és magánkézben.

A számszerű növekedés manapság jószerével csak attól függ, mennyi és milyen oktató szoftver kapható a gépekhez: érdemes-e befektetni a pénzünket? (A „játékidő" már lejárt . . .)

Hol vásárolhatunk, ha tanulni vágyunk? Az Ápisz két *számítástechnikai szaküzlete* a VIII. kerület, Szigony utca 15-ben és a XI. kerület, Budafoki út 7-ben van. (Ez utóbbi igazán jó helyre települt, hiszen szemben áll a műegyetem bejáratával.) A *Magiszter Könyvesboltot* az V. kerület, Városház utca 1-ben, a *Táncsics Könyvesboltot* a VII. kerület, Lenin krt. 17-ben, a *Tankönyvboltot* az V. kerület, Október 6. utca 8-ban, a *Műszaki Könyáruházat* a VI. kerület, Liszt Ferenc tér 9-ben, a *KÖNYVÉRTÉKA* áruházat az V. kerület, Honvéd utca 5-ben, a *Számítástechnikai Áruházat* a XIII. kerület, Balzac utca 35-ben kell keresnünk. (Ez a hat bolt a Belváros szívében, lényegében egy 2 km-es sugarú körön belül helyezkedik el.)

Hasznos-e a koncentráltság a forgalmazás szempontjából? Egyelőre mindenképpen igen! Hiszen ha valaki azzal a céllal indul el a bevásárolókörútra, hogy feltérképezi a kapható oktatóprogramokat, és utána igénye és lehetőségei szerint vásárol, mindezt egy nap leforgása alatt megteheti.

Valószínűleg az árucikk közismertté válásával és a forgalom felfutásával a reményünk is nőhet arra, hogy az oktatóprogramok vásárlási feltételei közelítenek egymáshoz, bármilyen távoli vidékein is lakjunk e hazának. Ez a feltételezésünk két, egymással összefüggő tényen alapul:

— a jelenleg forgalmazó üzletek többsége könyvesbolt;

— a könyvesbolt és/vagy terjesztői ott vannak a legkisebb településen, eljutnak minden iskolába, munkahelyre.

Forgalmazási feltételek

Mai fogalmaink szerint azokkal a boltokkal szemben, amelyek oktatóprogramok árusítására vállalkoznak, az áru jellege néhány dolgot megkíván. Nevezetesen: a boltban legyenek meg a portéka bemutatásának technikai feltételei, vagyis azok a személyi számítógépek, amelyekre készült programokat a boltban kapni. Pontosabban: legyen ott az alapgép, a magnó és/vagy lemezmeghajtó, monitor.

Körsétánkon az alábbi gépekre találkoztunk szoftverkínálattal: HT—16, HT—64, Sinclair ZX 48 k, C16, C64, C Plus/4, Primo. Hogy választani is tudjunk, a boltban találnunk kell olyan szakembert, aki a készletet napra készen ismeri, a programokat a fenti gépeken be tudja mutatni, és kellő mélységben tájékoztat a szoftverek tartal-

máról. Sőt a programválaszték ismeretében — szükség esetén — a gépvásárláshoz is tanácsot ad. Ez, de az alábbi szempontok kielégítése is *kettős érdek*, ezért mindenképpen *jogos elvárás*. ELADÓ és VEVŐ ezen a piacon is együtt van jelen! Tehát:

— A programok legyenek illően adjusztálva, küllemük keltsen bizalmat a vásárlóban.

— Minden programhoz tartozzon írásos ismertető.

— A boltnak legyen a forgalmazott programokról katalógusa. A katalógusban lehetőleg legalább kétféle osztályozás szerint szerepeljenek a programok: rövid ismertetés *tantárgyak* alapján, és egy-egy *géptípusra* kapható szoftverek listája tantárgyak szerinti bontásban.

— Legyen egy rész elkülönítve az üzletben, ahol a programmal való ismerkedést kevésbé zavarják, és a válogató sem akadályoz más vásárlókat.

— Tartson a bolt minél szélesebb választékot, és átmeneti időre se támadjanak „lyukak" a katalógusban.

Nincs két egyforma bolt

Talán a fentiek szükségessége a legfőbb oka, hogy a nagy könyvterjesztők, illetve az effajta kereskedéssel is foglalkozó számítástechnikai vállalatok csak egy-egy boltjukban vezették be ezeknek a programoknak az árusítását. És most ezekben a következő a helyzet. *Gépellátottság* tekintetében jól áll a Budafoki úti Ápisz (4 db gépük van), a Magiszter (szintén 4 db), a KÖNYVÉRTÉKA (5 db). De legalább kétféle géppel a többiek is rendelkeznek. A vásárlásnál csaknem mindegyik üzletben *szakeladók segítenek:* így volt ez a Tankönyvboltban, a Magiszterben, a Műszaki Könyváruházban, a Budafoki úton. A Honvéd utcában *külön szolgáltatás*, hogy naponta — általában a délutáni órákban — tartanak *szerzői bemutatót*, ahol műhelytitkokba is beavatnak bennünket, s választ kaphatunk a legkülönbözőbb szakmai kérdésekre.

A magnókazettán és hajlékonylemezen megjelenő programok a legváltozatosabb külleműek és felszereltségűek. A Novotrade, a TII, az LSI ATSZ és az OKTA a legnagyobb oktatóprogram-szállítók.

Vásárlókkal és eladókkal beszélgetve egyöntetűnek mutatkozik a vélemény, hogy az OKTA programok külső megjelenése — szép kék színek ízléses kombinációja a címkén és a borítón — és felszereltsége — jól szerkesztett katalógus, kazettánként adott programismertető — miatt 1987 derekán méltán ezek a termékek a legkeresettebbek. Nem mellékes, hogy a gusztusos formához igényes tartalom is járul. Örömmel tapasztaltuk, hogy ez utóbbi minőségi összevetőre a fenti megállapítás szinte valamennyi forgalomban levő oktatóprogramnál elmondható.

Legtöbb üzletből, amelyik több gyártó programjait is forgalmazza, az elmúlt tanév végén még hiányzott a teljes kínálatot bemutató lista. Pedig kiábrándító, ha valakinek, akit történetesen a Commodore 64-re készült angol programok érdekelnek, át kell böngésznie közel kétszáz címet.

Az üzletek kivétel nélkül berendeztek egy sarkot az oktatóprogramoknak. Kétségkívül a Magiszternek, a Műszaki Könyváruháznak és a KÖNYVÉRTÉKÁ-nak a legnagyobb az alapterülete, s ezt az adottságukat remekül ki is használják.

Sajátos (de magyar!) jelenség az ideiglenes vagy hosszan tartó áruhiány. Tavasszal éppen a kazetták tűntek el. Az egyetlen hazai gyártó, a Polimer, alapanyaggondokkal küzdött, s akkor nem volt semmi biztató . . . E téren jelentős javulással aligha számolhatunk. Ezért egy jó tanács (tanácsokban egyébként is gazdag ország vagyunk): aki teheti, szerezze be mielőbb a számára legszükségesebb programokat!

Mi van a polcokon?

Örömmel írhatjuk le, hogy van miből válogatni. A választék boltonként, géptípusonként, tantárgyanként változó, de egy-egy időszakban — ha különböző helyekről is — a kívánt készlet összeszedhető. Csak ízelítőül néhány tantárgy és benne két-két tárgykör.

Magyar

MONDATELEMZÉS. Két tagmondatból álló alá- és mellérendelő összetett mondatok

SZÓFAJOK. A magyar nyelv szófaji rendszere

Történelem

CANNAE. A cannae-i csata bemutatása

IDŐJÁTÉK. A gimnázium I. és II. osztályának legfontosabb évszámai

Földrajz

KERESD A TÉRKÉPEN. Magyarország, Európa stb.

PROGRAMCSOMAG VIII. OSZTÁLYOSOKNAK. Tíz program, amely lefedi a tárgy éves tananyagát

Biológia

NÖVÉNYHATÁROZÁS. Zárvatermők fajai

VITAMINOK. A vitaminok leggyakoribb jellemzői

Kémia

ELEK. Atomok elektronszerkezete

OKTE. Kovalens kötésű molekulák elektronszerkezete

Matematika

SZÁMRENDSZEREK. Különböző alapú számrendszerek átszámítása

EGYENLET. Polinomok értékeinek kiszámítása, algebrai függvények

Fizika

IDEÁLIS GÁZOK. A gáznyomás kinematikai értelmezése

PROGRAMCSOMAG VI. OSZTÁLYOSOKNAK. 15 program, amely lefedi az éves tananyagot

Statisztika

FAKTORANALÍZIS. Faktorsúlyrendszer, sajátértékek, sajátvektorok

TESZT- ÉS KÉRDŐÍVSZERKESZTŐ. Számítógépes adatfelvétel, adatrendezés

Angol

CON. Feltételes mondatok alapesetei

PASSIVE. Szenvedő szerkezetek

Francia

LEXI-TRAP. A francia főnév neme

VERBS/-ER. A szabályos -er végű igék ragozása

Német

SUBPLUR. A főnevek többesszáma

DEUTSVER. Az ún. „erős" igék múlt ideje

Orosz

OTRITSANIE. A tagadás gyakorlása

RUSUPRI. Főnevek és birtokos névmások

Árak

Az egyes programok 180 és 590 forint között változnak. (Ez utóbbinál az adathordozó hajlékonylemez.) A tantárgy egész éves tananyagát lefedő programcsomag ára néhány ezer forint, de az összetevő részprogramok önállóan is kaphatók 300 forintos egységáron. Nem drága tehát, különösen, ha összevetjük egy-egy magánóra árával.

Végül hangsúlyozzuk: sok esetben az oktatóprogram értékét már nemcsak az ára viszonyítja; sokszor a gépbeszerzés sem a döntő kérdés. Hanem az a tény válik dominánssá, hogy gyakran általa olyan praktikus oktatási segédeszközt birtoklunk, amely számos szituációban egyenértékűen alig helyettesíthető.

— ij —

Nem szent tehén

Fonalfék, fonalerő

Az alábbi program közreadásával az a célom, hogy bemutassam: a sokak, talán még az alkotók által is csak „játékszernek" minősített Sinclair ZX–Spectrum alkalmas arra, hogy komoly műszaki feladatok megoldását segítse. A program által megoldott feladat a textilipari szakközépiskolák és technikumok kötőipari szakán a tananyag része. A program közzétételétől azt remélem, hogy az érintett iskolák ezt a nehéznek minősített anyagrészt mint számítástechnikai témát is feldolgozzák (esetleg átdolgozzák saját számítógépeikre), és ezzel közelebb viszik azt a tanulókhoz. Egyben szeretném a tanároknak és diákoknak egyaránt bemutatni, hogy az ilyen és ehhez hasonló technológiai számítások számítógéppel milyen „emberközelivé" válnak, mennyire elvesztik „rémisztő" matematikai jellegüket, és számítógéppel megoldva milyen jól segítik a gyakorlati üzemi munkát.

A gyakorlati probléma

A textiliparban gyakran vissza-visszatérő feladat a feldolgozott fonal bizonyos fokú megfeszítése. A teljesen laza fonal éppúgy nem dolgozható fel hibátlanul, mint a túlságosan feszes. Ezért a textilgépeken olyan szerkezeteket használnak, amelyekkel a fonal feszültsége, helyesebben a fonalban előidézett húzóerő a feldolgozás követelményeinek megfelelően beállítható.

Az *1. ábra* egy ilyen szerkezetet, ún. fonalféket mutat. A fonal részben henger alakú csapokat kerül meg, részben ezeken a csapokon még fémtányérok is vannak, amelyek terhelik a fonalat. A csapok elrendezésével változtatható a fonallal körülfogott ív szöge. Minthogy a csap felülete és a fonal között az ún. kötélsúrlódás lép fel, amelynek nagysága az

$\int e^{\mu_k \alpha} \int$

tényezőtől függ (itt e a természetes logaritmus alapszáma, α az említett körülforgási szög, μ_k pedig a csap és a fonal között érvényesülő súrlódási tényező), a csapok helyzetének beállításával a fonalban a súrlódásból keletkező húzóerő befolyásolható.

A féktányérok és a fonal között is súrlódás keletkezik, ezt az

$\int m\mu_n \int$

tényező határozza meg, ahol m a féktányér tömege, μ_n pedig a féktányér és a fonal közötti súrlódási tényező. A féktányérok cserélhetők, így a kívánt fonalerőt megfelelő tömegű féktányérral érhetjük el.

Az elmondottakból látható, hogy a fonalnak a fék csapjain való vezetésével, valamint a különböző tömegű féktányérok alkalmazásával a fonalerő igen széles határok között, a mindenkori technológiai igényeknek megfelelően állítható be.

Mire a fonal a fékszerkezetbe való belépés helyétől a kilépési helyhez ér, a belépés helyén már eleve benne lévő F_o erő a súrlódások következtében megnövekszik és F értéket vesz fel. Az említett kétféle súrlódást figyelembe véve, a szokásos maximum 3 csap és legfeljebb ugyanennyi féktányér esetén, az F erő így írható fel:

$$\int F = F_o e^{\mu_k(\alpha_1+\alpha_2+\alpha_3)} + \mu_n g \int \left[m_1 e^{\mu_k(\alpha_2+\alpha_3)} + m_2 e^{\mu_k \alpha_3} + m_3 \right] \int$$

ahol α_1, α_2, α_3 az egyes csapoknál kialakuló átfogási szög, m_1, m_2 és m_3 a féktányérok tömege, g pedig a gravitációs gyorsulás, amelynek értéke 9,81 m/s². Ha a képlettel számításokat akarunk végezni, az α szögek nagyságát radiánban, az m tömegeket kg-ban kell behelyettesíteni.

A megoldás

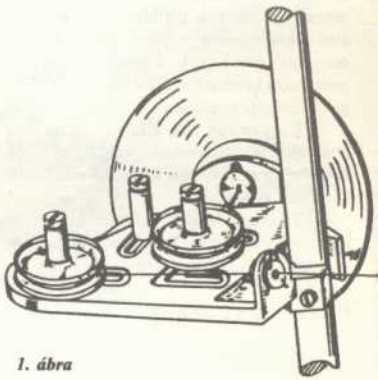
A számítás meglehetősen bonyolult és hosszadalmas, ezért célszerű volt számítógépes programot készíteni hozzá.

A gyakorlatban az α szögeket pontosan nem lehet megmérni, csak a csapok helyzetéből és átmérőjéből lehet kiszámítani. A *2. ábrán* látható vázlaton bejelöltük azokat a koordinátákat, amelyekre ehhez a számításhoz szükség van. A gyakorlatban az is előfordul, hogy — attól függően, milyen fonalerőt kell létrehozni — a fonal befűzésénél egyes csapokat kihagynak, tehát a fonal ilyenkor csak egy vagy két csapot kerül meg, de hogy melyiket, az a befűzést előíró technológus döntésétől függ. Ezt a programban figyelembe kell venni.

A program nagyvonalú felépítését a *3. ábrán* látható folyamatábra szemlélteti, maga a Sinclair Spectrumra készített program a *listán* látható.

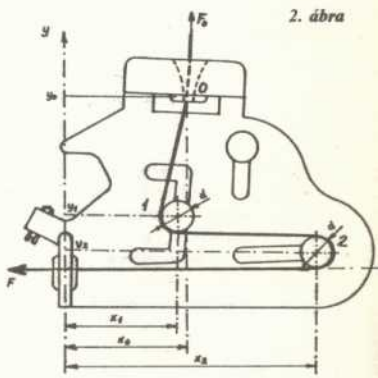
Bonyolult programok esetén természetesen nem célszerű részletes magyarázatot építeni a programba; ilyenkor a programhoz mellékelt kézikönyv vagy használati utasítás nélkülözhetetlen. Mindenképpen helyes azonban, ha a program elején rövid magyarázat tájékoztat arról, hogy a program mire használható és milyen szolgáltatásokat nyújt (17—18-as sor). Kívánatos, hogy ezek a szöveges részek a magyar nyelv teljes betűkészletével készüljenek, azaz a programban — felhasználói grafikus jelek formájában — elő kell állítani az ékezetes magyar betűket is (5000—5150-es sor).

A fonalfék geometriai elrendezéséből meglehetősen bonyolult trigonometriai számításokkal határozhatjuk meg az α szögeket. Ennek részleteire nem térünk ki, a program 150—350-es soraiból kikövetkeztethető. Itt feltételezzük — és ez a gyakorlatban általában így is van —, hogy valamennyi csap azonos átmérőjű.



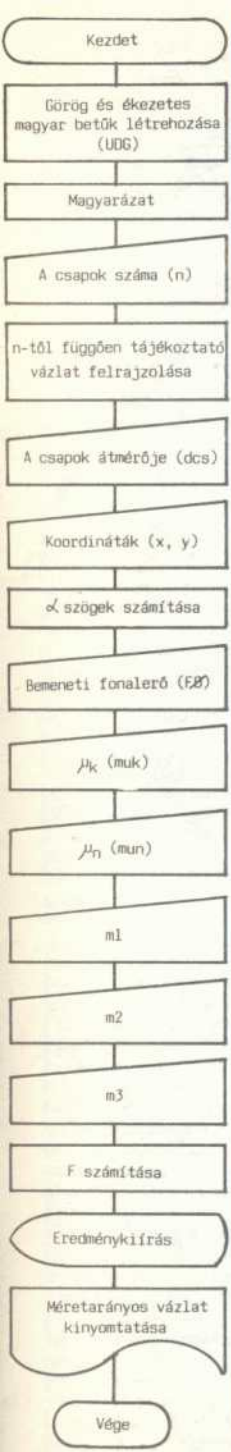
1. ábra

2. ábra



Ha a 420-as sorban feltett kérdésre (amikor a bemeneti fonalerőt kell megadni) 1-gyel válaszolunk, az 500-as sorban F értékeként azt kapjuk meg, hogy a fonalfék az adott beállításnál hányszorosára növeli meg a fonalerőt a kimenetnél. Ugyanezt az adatot adja meg egyébként — azonban F_o bármilyen értékénél — az 510-es sor is.

A 430-as és 440-es sorban a kétféle súrlódási tényezőt kell megadnunk. Erre a szakmai kézikönyvekben találunk adatot, miután ezek az értékek az alkalmazott anyagoktól függnek. Ha ezeket eleve beépítjük, és például a fonal és a csapok, illetve féktá-

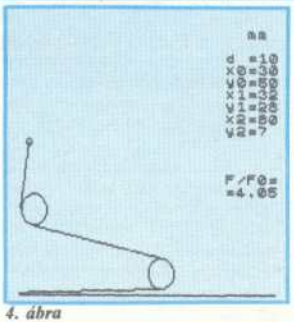


3. ábra

```
  1 CLS
  2 REM **********************
  3 REM *                    *
  4 REM * FONALFÉK,  FONALERŐ *
  5 REM *                    *
  6 REM * © Lázár Károly 1985 *
  7 REM *                    *
  8 REM **********************
  9 REM
 15 GO SUB 5000: REM Görög
                     betűk
 16 GO SUB 5100: REM Ékezetes
                     betűk
 17 CLS : PRINT TAB 7;"FONALFÉK
, FONALERŐ""""A program egy-, két- és három-""csapos tányéros fonalfékekre alkalmazható. A geometriai méretek""megadása után kiírja az átfogási""szögeket, majd megkérdezi a súrlódási tényezőket egyrészt a fo-
 18 PRINT "nal és a csap (μk), másrészt a""fonal és a féktányér között (μn).""""végezetul a féktányérok tömegeit""és a bemenő fonalhúzóerőt kell""megadni.""""Az eredmény (a kimenő fonalhúzó-""erő és annak a bemenő erőhöz viszonyított nagysága) kiírása után""a képernyőn megjelenik a fonalfék méretarányos vázlata, amit""kívánságra kiis nyomtatható."
 19 REM A geometriai méretek megadása
 20 INPUT "Hány csap van? ";n
 25 DIM x(n+1): DIM y(n+1)
 30 CLS : PRINT AT 20,0;"A méreteket milliméterben kérem:"
 35 GO SUB (n=1)*6500+(n=2)*6600+(n=3)*6700: REM Tájékoztató
                     vázlat
 40 INPUT "A csapok átmérője: d =";dcs: LET r=dcs/2
 42 PRINT AT 0,26;"mm"
 45 PRINT AT 2,26;"d =";dcs
 45 LET x2=0: LET y2=0: LET x3=0: LET y3=0
 50 INPUT "x0=";x0
 55 LET x(1)=x0: PRINT AT 3,26;"x0=";x0
 60 INPUT "y0=";y0
 65 LET y(1)=y0: PRINT AT 4,26;"y0=";y0
 70 INPUT "x1=";x1
 75 LET x(2)=x1: PRINT AT 5,26;"x1=";x1
 80 INPUT "y1=";y1
 85 LET y(2)=y1: PRINT AT 6,26;"y1=";y1
 90 IF n=1 THEN GO TO 140
100 INPUT "x2=";x2
105 LET x(3)=x2: PRINT AT 7,26;"x2=";x2
110 INPUT "y2=";y2
115 LET y(3)=y2: PRINT AT 8,26;"y2=";y2
117 IF n=2 THEN GO TO 132
120 INPUT "x3=";x3
125 LET x(4)=x3: PRINT AT 9,26;"x3=";x3
130 INPUT "y3=";y3
131 LET y(4)=y3: PRINT AT 10,26;"y3=";y3
132 IF n=2 THEN LET x3=x2
135 IF n=2 THEN LET y3=y2
140 PAUSE 100: CLS : PRINT AT 10,12;"Egy pillanat!"
145 REM Az átfogási szögek számítása a koordinátákból
150 LET k0=SQR ((x0-x1)*(x0-x1)+(y0-y1)*(y0-y1))
155 IF n=1 THEN GO TO 180
160 LET k3=SQR (x3*x3+y3*y3)
170 LET k2=(n=2)*SQR ((x3-x1)*(x3-x1)+(y1-y3)*(y1-y3))+(n=3)*SQR ((x3-x2)*(x3-x2)+(y2-y3)*(y2-y3))
180 LET k1=(n=3)*SQR ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))+(n=1)*SQR (x1*x1+y1*y1)
190 LET a=ASN ((x0-x1)/k0)
191 IF n<>1 THEN GO TO 200
195 LET a1=(n=1)*ASN ((x1-x0)/k0): LET d1=(n=1)*(ASN (y1/k1)-ASN (r/k1)): LET c1=(n=1)*ASN (y1/k1)
200 LET b=ASN (r/k0)
202 IF n=1 THEN GO TO 278
205 IF n=2 THEN GO TO 240
210 LET c=ASN ((y2-y1)/k1)
220 LET d=ACS (2*r/k1)
230 LET e=ASN ((x3-x2)/k2)
240 LET f=ASN (y3/k3)
250 LET g=ACS (r/k3)
260 LET h=ASN ((y1-y3)/k2)
270 LET i=ACS (2*r/k2)
275 IF n=2 THEN GO TO 310
277 IF n=3 THEN GO TO 280
278 LET a1=PI/2+a1+b-d1: IF n=1 THEN GO TO 330
280 LET a11=PI+a+b+c-d
290 LET a12=PI+c-d-e
300 LET a13=PI+e-f-g
310 LET a14=PI-h-i+a+b
320 LET a15=2*PI-h-i-f-g
330 LET a$=" rad.": LET b$=" fok"
340 LET o=180/PI
350 IF n=2 THEN GO TO 395
355 CLS
360 PRINT "α1=";a11;a$;TAB 19;a11*o;b$
365 IF n=1 THEN GO TO 420
370 PRINT "α2=";a12;a$;TAB 19;a12*o;b$
380 PRINT "α3=";a13;a$;TAB 19;a13*o;b$
390 GO TO 420
395 CLS
400 PRINT "α1=";a14;a$;TAB 19;a14*o;b$
410 PRINT "α2=";a15;a$;TAB 19;a15*o;b$
415 REM Egyéb kiinduló adatok megadása
420 INPUT "F0 [cN]=";f0: PRINT "F0=";f0;" cN": LET f0=f0/100
430 INPUT "μk=";muk: PRINT "μk=";muk
440 INPUT "μn=";mun: PRINT "μn=";mun
450 INPUT "m1 [g]=";m1: PRINT "m1=";m1;" g": LET m1=m1/1000
460 IF n=1 THEN GO TO 485
470 INPUT "m2 [g]=";m2: PRINT "m2=";m2;" g": LET m2=m2/1000
475 IF n=2 THEN GO TO 488
480 INPUT "m3 [g]=";m3: PRINT "m3=";m3;" g": LET m3=m3/1000
482 IF n=3 THEN GO TO 490
485 LET ft=f0*EXP (muk*(a11)+mun*9.81*m1: GO TO 500
486 LET ft=f0*EXP (muk*(a14+a15))+mun*9.81*(m1*EXP (muk*a15)+m2): GO TO 500
490 LET ft=f0*EXP (muk*(a11+a12+a13))+mun*9.81*(m1*EXP (muk*(a12+a13))+m2*EXP (muk*a13)+m3)
500 PRINT : PRINT "F=";ft*100;" cN"
510 PRINT "F/F0=";ft/f0
520 PRINT AT 20,0;"Nyomjon meg egy gombot!": LET w$=INKEY$: IF INKEY$="" THEN GO TO 520
530 CLS : GO TO 6000
5000 FOR i=1 TO 2
5005 READ p$
5010 FOR j=0 TO 7
5020 READ k: POKE USR p$+j,k
5030 NEXT j
5040 NEXT i
5050 DATA "α",0,0,58,68,68,68,58,0,"μ",0,0,66,66,66,120,64,64
5060 RETURN
5100 DATA "á",8,16,56,4,60,68,60,0,"Á",8,60,82,66,126,66,66,0,"é",8,16,56,68,120,64,60,0,"É",8,126,80,64,124,64,126,0,"í",4,8,0,24,8,8,28,0,"Í",4,8,62,8,8,8,62,0
5110 DATA "ó",8,16,56,68,68,68,56,0,"Ó",8,60,82,66,66,66,60,0,"ö",40,0,56,68,68,68,56,0,"Ö",66,0,66,66,66,66,60,0,"ő",20,40,56,68,68,68,56,0,"Ő",20,124,170,130,130,130,124,0
5120 DATA "ú",8,16,68,68,68,68,56,0,"Ú",8,82,66,66,66,66,60,0,"ü
```

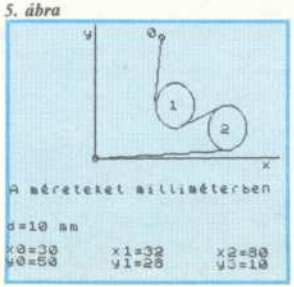
```
:,40,0,68,68,68,68,56,0,"Ü",36,6
5,66,66,66,66,60,0,"Ü",20,40,68,
68,68,68,56,0,"Ü",20,170,130,130
,130,130,24,0
5130 FOR f=1 TO 18
5140 READ p$: FOR m=0 TO 7: READ
 a: POKE USR p$+m,a: NEXT m: NEX
T f
5150 RETURN
6000 REM A méretarányos vázlat
          felrajzolása
6001 IF n=1 THEN GO TO 6200
6005 PLOT 0,0: DRAW 255,0
6010 PLOT 0,0: DRAW 0,175
6015 CIRCLE 2*x0,2*y0,2
6020 CIRCLE 2*x1,2*y1,2*r
6025 IF n=2 THEN GO TO 6040
6030 CIRCLE 2*x2,2*y2,2*r
6040 CIRCLE 2*x3,2*y3,2*r
6050 PLOT 2*x0,2*y0: DRAW -2*(x0
-x1+r*COS (a+b)),-2*(y0-y1-r*SIN
 (a+b))
6055 IF n=2 THEN GO TO 6075
6060 PLOT 2*(x1+r*SIN (PI/2-d+c)
),2*(y1-r*COS (PI/2-d+c)): DRAW
2*(x2-x1-2*r*SIN (PI/2-d+c)),2*(
y2-y1+2*r*COS (PI/2-d+c))
6070 PLOT 2*(x2+r*SIN (PI/2-e)),
2*(y2+r*COS (PI/2-e)): DRAW 2*(x
3-x2-r*SIN (PI/2-e)+r*COS e),2*(
-(y2-y3+r*COS (PI/2-e)-r*SIN e))
: IF n=3 THEN GO TO 6080
6075 PLOT 2*(x1-r*SIN (-PI/2+h+l
)),2*(y1-r*COS (-PI/2+h+l)): DRA
W 2*(x3-x1+2*r*SIN (-PI/2+h+l)),
-2*(y1-y3-2*r*COS (-PI/2+h+l)):
GO TO 6100
6080 LET u=2*(x3+r*SIN (PI/2+e-a
l3)): LET v=2*(y3-r*COS (PI/2+e-
al3))
6090 GO TO 6110
6100 LET u=2*(x3+r*SIN (1.5*PI-h
-l-al5)): LET v=2*(y3-r*COS (1.5
*PI-h-l-al5))
6110 PLOT u,v: DRAW -u,-v
6120 GO SUB 6300
6190 GO TO 7000
6200 LET q=0
6205 IF y1<0 THEN LET q=y1+r
6210 IF y1<r THEN LET q=r
6220 PLOT 0,2*q: DRAW 255,0
6230 PLOT 0,2*q: DRAW 0,175-2*q
6240 CIRCLE 2*x0,2*(y0+q),2
6250 CIRCLE 2*x1,2*(y1+q),2*r
6260 PLOT 2*x0,2*(y0+q): DRAW 2*
(x1-x0+r*COS (a1+b)),-2*(y0-y1-r
*SIN (a1+b))
6270 PLOT 2*(x1+r*SIN d1),2*(y1+
q-r*COS d1): DRAW -2*(x1+r*SIN d
1),-2*(y1-r*COS d1)
6280 GO SUB 6300
6290 GO TO 7000
6300 PRINT AT 0,28;"mm"
6310 PRINT AT 2,26;"d =";dcs
6320 FOR w=1 TO n+1
6330 PRINT AT 1+2*w,26;"x";w-1;"
=";x(w): PRINT AT 2+2*w,26;"y";w
-1;"=";y(w)
6340 NEXT w
6350 PRINT AT 4+2*w,26;"F/F0=";A
T 5+2*w,26;"=";(INT (10*ft/f0+0.
5))/10
6360 RETURN
6500 REM A tájékoztató vázlat
          felrajzolása
6505 PLOT 65,170: DRAW 0,-90: DR
AW 120,0
6510 PRINT AT 1,7;"y";AT 12,22;"
x"
6520 CIRCLE 125,115,15
6530 CIRCLE 115,160,2
6540 CIRCLE 65,80,2
6550 PLOT 115,160: DRAW 23,-38:
PLOT 130,101: DRAW -65,-21
6560 PRINT AT 1,13;"0";AT 7,15;"
1"
6570 RETURN
6600 PLOT 65,170: DRAW 0,-90: DR
AW 140,0
6610 PRINT AT 1,7;"y";AT 12,24;"
x"
6620 CIRCLE 125,115,15
6630 CIRCLE 165,100,15
6640 CIRCLE 115,160,2
6650 CIRCLE 65,80,2
6660 PLOT 115,160: DRAW -5,-42:
PLOT 130,101: DRAW 28,12: PLOT 1
67,85: DRAW -102,-5
6670 PRINT AT 1,13;"0";AT 7,15;"
1";AT 9,20;"2"
6680 RETURN
6700 PLOT 65,170: DRAW 0,-90: DR
AW 140,0
6710 PRINT AT 1,7;"y";AT 12,24;"
x"
6720 CIRCLE 100,115,15
6730 CIRCLE 135,140,15
6740 CIRCLE 165,100,15
6750 CIRCLE 115,160,2
6760 CIRCLE 65,80,2
6770 PLOT 115,160: DRAW -28,-38:
 PLOT 115,110: DRAW 5,27: PLOT 1
50,145: DRAW 30,-40: PLOT 165,85
: DRAW -100,-5
6780 PRINT AT 1,13;"0";AT 7,12;"
1";AT 9,20;"3";AT 4,16;"2"
6790 RETURN
7000 REM Befejezés
7001 PRINT #1;"Kinyomtassam? (i/
n) ": PAUSE 0: LET w$=INKEY$
7002 IF w$<>"i" AND w$<>"n" THEN
 GO TO 7001
7003 IF w$="i" THEN COPY : CLS :
 GO TO 7005
7004 IF w$="n" THEN GO TO 7005
7005 PRINT #1;"Folytatjuk? (i/n)
": PAUSE 0: LET w$=INKEY$
7010 IF w$="i" THEN CLS : GO TO
20
7020 IF w$<>"i" THEN CLS
7030 PRINT AT 10,6;"Köszönöm, be
fejeztük."
7040 STOP
```

nyérok anyagát kérdezné a program, amiből azután — megfelelő tömbökből — kikeresné a súrlódási tényező értékét, ez szükségtelenül bonyolulttá tette volna a programot. Egy-egy üzem gyakorlatában ugyanis viszonylag kevés kombináció fordul elő; ezeket a technológusok — akik a programot használják — általában fejből tudják.



4. ábra

5. ábra

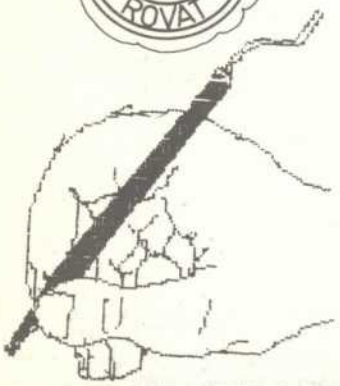


Fontos részét képezik a programnak a 6000—6790-es sorok. Amint az a folyamatábrán látható, a csapok számának tisztázása után a képernyőn tájékoztató vázlat jelenik meg *(4. ábra)*, ahol a csapok be vannak számozva, hogy ennek megfelelően lehessen válaszolni a program kérdéseire. Ezt a vázlatot — a csapok számának megfelelően — a 6500, 6600, illetve 6700-as sorban kezdődő programrész állítja elő. A 6000—6110 és 6200-6270-es sorban hasonló, azonban most már a valóságos fékelrendezésnek helyes méretarányban megfelelő vázlatot készít a program (és ki is nyomtatja) a számítások befejezése után *(5. ábra)*. Erre azért van szükség, mert ennek a vázlatnak a segítségével ellenőrizheti a technológus a fonalfék tényleges beállítását.

Lehet, hogy a szigorúan vett programozástechnika szempontjából az én programom nem igazán „elegáns", de nem is ez a lényeg. A fontos az, hogy a program a feladatnak megfelelően működjön, és segítse a tervezési munkát. Ezt a szemléletet kellene az iskolákban — főleg a szakmai iskolákban — a tanulókban kialakítani, s hozzá természetesen az érvényesítéséhez szükséges ismereteket megtanítani.

LÁZÁR KÁROLY

Rajzoljon Spectrummal!



Talán sokan kiélnék rajzolói kedvüket, művészi hajlamaikat Spectrum számítógépen, ám erre nincs módjuk, mert nem ismernek ehhez programokat. Illetve a külföldön készült gépi kódú rajzolóprogramok igencsak drágák, vagy egyáltalán nem kaphatóak. Az alábbi BASIC nyelvű program viszont a legtöbb rajzolási lehetőséget magába foglalja.

A program variációi:

Vonal rajzolása; kör rajzolása és befestése; doboz kerete, festése; pontok folyamatos törlése vagy rajzolása; papír, a keret és a tinta színének állítása, rajzok kimentése, illetve betöltése magnón, a kurzor lépésszámának változtatása.

```
  10 CLEAR 57999: RESTORE 11: FO
R a=58000 TO 58023: READ c: POKE
 a,c: NEXT a: BORDER 6: PAPER 7:
 INK 0: CLS : GO TO 8000
  11 DATA 33,0,64,17,168,226,1,0
,27,237,176,201,33,168,226,17,0,
64,1,0,27,237,176,201
  20 LET i=0: LET b=6: LET v=7:
LET j=1: LET rp=0: LET x=20:LET
y=20: LET q=x: LET w=y: LET r=2:
 LET xx=x: LET yy=y: PLOT 20,20:
 PRINT #0;AT 0,1;"Mozgas   ";AT
 1,1;"Sprite"
  30 PAUSE 0: LET i$=INKEY$: LET
 q=x: LET w=y
  40 IF i$="p" THEN LET x=x+j: I
F x>255-j THEN LET x=255-j
  41 IF i$="o" THEN LET x=x-j: I
F x<j THEN LET x=j
  42 IF i$="q" THEN LET y=y+j: I
F y>176-j THEN LET y=176-j
  43 IF i$="a" THEN LET y=y-j: I
F y<j THEN LET y=j
  44 IF i$=CHR$ 13 THEN GO SUB 4
000
  45 IF i$="NOT " THEN GO TO 700
0
  46 IF i$="/" THEN CLS : PLOT x
,y
  47 IF i$="-" THEN GO TO 1000
  49 IF i$="d" THEN GO SUB 2000
  50 LET a$="jlcibvs01": FOR a=1
TO 9: IF i$=a$(a) THEN OUT 0,a:
BORDER b: GO TO 100+a
  60 NEXT a: IF i$=" STOP " THEN
 GO TO 6000
  65 PRINT #0;AT 0,1;"Mozgas
  "
  66 LET rp=POINT (x,y): LET wp=
POINT (q,w)
  70 IF r=0 THEN PLOT INVERSE 1,
q,w: PLOT x,y: GO TO 30
  80 IF r=1 THEN PLOT q,w: PLOT
x,y
  90 GO SUB 9000: GO TO 30
 100 INPUT "Lepes:";j: IF j<1 OR
 j>50 THEN GO TO 100
 110 GO TO 30
 200 LET xx=x: LET yy=y: LET r=2
: PRINT #0;AT 0,1;"Vonal      "
 210 PAUSE 0: LET i$=INKEY$: LET
 q=x: LET w=y
 220 IF i$="p" THEN LET x=x+j: I
F x>255-j THEN LET x=255-j
 221 IF i$="o" THEN LET x=x-j: I
F x<j THEN LET x=j
 222 IF i$="q" THEN LET y=y+j: I
F y>176-j THEN LET y=176-j
 223 IF i$="a" THEN LET y=y-j: I
F y<j THEN LET y=j
 224 IF i$=CHR$ 13 THEN GO TO 24
0
 225 GO SUB 9000: GO TO 210
 240 PLOT xx,yy: DRAW x-xx,y-yy:
 GO TO 30
 300 LET xx=x: LET r=2: PRINT #0;
AT 0,1;"Kor       "
 310 PAUSE 0: LET i$=INKEY$: LET
 q=x: LET w=y
 320 IF i$="p" THEN LET x=x+j: I
F x>255-j THEN LET x=255-j
 330 IF i$="o" THEN LET x=x-j: I
F x<j THEN LET x=j
 340 IF i$=CHR$ 13 THEN GO TO 36
0
 350 GO SUB 9000: GO TO 310
 360 CIRCLE xx,y,x-xx
 370 INPUT "Besatirozzam? (I/N):
"; LINE m$: IF m$<>"i" AND m$<>"
n" THEN GO TO 370
 380 IF m$="n" THEN GO TO 30
 390 INPUT "Milyen szinu legyen?
 (0-7):";sa: IF sa<0 OR sa>7 THE
N GO TO 390
 392 INK sa: IF x>xx THEN FOR n=
0 TO x-xx: LET b1=INT (.5+SQR ((
x-xx)^2-n^2)*2): LET a1=y+b1/2:
PLOT xx+n,a1: DRAW 0,-b1: PLOT x
x-n,a1: DRAW 0,-b1: NEXT n
 395 IF x<xx THEN FOR n=0 TO xx-
x: LET b1=INT (.5+SQR ((xx-x)^2-
n^2)*2): LET a1=y+b1/2: PLOT xx+
n,a1: DRAW 0,-b1: PLOT xx-n,a1:
DRAW 0,-b1: NEXT n
 398 GO TO 30
 400 INPUT "Iras szine:";i: IF i
<0 OR i>7 THEN GO TO 400
 410 INK i: GO TO 30
 500 INPUT "Keret szine:";b: IF
b<0 OR b>7 THEN GO TO 500
 510 BORDER b: GO TO 30
 600 INPUT "Papir szine:";v: IF
v<0 OR v>7 THEN GO TO 600
 610 PRINT AT 0,0;: FOR a=1 TO 7
04
 620 PRINT PAPER v; INK 8; OVER
1;" ";: NEXT a: PAPER v: GO TO 3
0
 700 LET r=2: PRINT #0;AT 1,1;"S
prite   ": GO TO 30
 800 LET r=0: PRINT #0;AT 1,1;"T
orles   ": GO TO 30
 900 LET r=1: PRINT #0;AT 1,1;"R
ajzolas ": GO TO 30
1000 INPUT "Kepernyotorles? (I/N
):"; LINE c$: IF c$<>"i" AND c$<
>"n" THEN GO TO 1000
1010 IF c$="i" THEN CLS
1020 PRINT AT 0,0;: LOAD ""SCREE
N$ : GO TO 30
2000 LET xx=x: LET yy=y: LET r=2
: PRINT #0;AT 0,1;"Doboz      "
2010 PAUSE 0: LET i$=INKEY$: LET
 q=x: LET w=y
2020 IF i$="p" THEN LET x=x+j: I
F x>255-j THEN LET x=255-j
2021 IF i$="o" THEN LET x=x-j: I
F x<j THEN LET x=j
2022 IF i$="q" THEN LET y=y+j: I
F y>175-j THEN LET y=175-j
2023 IF i$="a" THEN LET y=y-j: I
F y<j THEN LET y=j
2024 IF i$=CHR$ 13 THEN GO TO 20
40
2025 GO SUB 9000: GO TO 2010
2040 PLOT xx,yy: DRAW x-xx,0: DR
AW 0,y-yy: DRAW -(x-xx),0: DRAW
0,-(y-yy)
 2080 INPUT "Besatirozzam? (I/N):
"; LINE m$: IF m$<>"i" AND m$<>"
n" THEN GO TO 2080
2090 IF m$="n" THEN GO TO 30
2100 INPUT "Milyen szinu legyen?
 (0-7):";sa: IF sa<0 OR sa>7 THE
N GO TO 2100
2110 INK sa: PLOT xx,yy: IF x>xx
 AND y>yy THEN FOR a=yy TO y: PL
OT xx,a: DRAW x-xx,0: NEXT a
2120 IF x<xx AND y>yy THEN FOR a
=yy TO y: PLOT x,a: DRAW xx-x,0:
 NEXT a
2130 IF x<xx AND y<yy THEN FOR a
=y TO yy: PLOT x,a: DRAW xx-x,0:
 NEXT a
2140 IF x>xx AND y<yy THEN FOR a
=y TO yy: PLOT xx,a: DRAW x-xx,0
: NEXT a
2150 GO TO 30
4000 RANDOMIZE USR 58000
4010 BORDER 6: PAPER 7: INK 0: C
LS
4020 PRINT AT 0,11;"Szivarvany"
4030 PRINT "----------------------
-----------0.....fel,1.........
.....rajzolA.....le,0.........
.....torol0...balra,S.........
.....spriteP..jobbra,SSht+V..ke
pernyotorles----------------------
-----------";
4040 PRINT "L....vonal
            C.....kor
            D....doboz
            ----------------------
-----------1.....iras szine
            B....keret szine
            V....papir szine
            ----------------------
-----------";
4050 PRINT "SSht+S.....kep felve
tele magnoraSSht+J....kep betolt
ese magnoroISSht+A.............a
program vege----------------------
-----------";
```

Perifériateszt Commodore 128-ra

A perifériákat is kezelő programok írásakor szükséges lehet ellenőrizni a munkát, vagyis azt, hogy a felhasználó valóban bekapcsolta-e azokat. Az alábbi két rutin mindenesetre ehhez segít.

A 60000—60040 sorokig a lemezmeghajtót, a 60100—60180 sorokig a nyomtatót ellenőrző rutin található. Utóbbi meghívása előtt az egységszámot is be kell állítani. Ezt az EG változó tárolja. Itt az alprogram megvizsgálja, hogy megengedett nyomtató-egységszámot adtunk-e meg, s ha nem, akkor jelzi azt, majd megáll.

Nincs csatlakoztatva vagy bekapcsolva a kérdéses periféria, ha a rutin meghívása után a B változó értéke nulla, készen áll, ha értéke 1.

Mindkét teszt a BASIC V7.0 hibavizsgálásának lehetőségével „operál". Az első megpróbálja kiolvasni a meghajtó hibacsatornáját, ha sikerül, a floppy rendben van, ha nem, akkor elágazik a hibavizsgáló rutinra, amely nullára állítja B-t. A nyomtató tesztje megnyit egy csatornát a megadott egységszámú nyomtatóhoz, és kiír egy soremelést. Ha közben hiba lép fel, vagy a státuszváltozó értéke nem nulla — szintén hiba —, akkor is törli B-t. Majd lezárja a csatornát, kikapcsolja a hibavizsgálatot (TRAP), és befejezi a rutint.

Ha attól félünk, hogy a felhasználó az ellenőrző rutin alatt nyomja meg a BREAK-et, és ez váltja ki a hibát, akkor a RUN-STOP/RESTORE-t letilthatjuk a POKE 808,107 paranccsal.

A két alprogram használatát szemléltetik a 10—60 sorok; futtatás után kiírják a meghajtó, illetve a nyomtató állapotát.

NÉMETH ISTVÁN

```
10 PRINT"FLOPPY ";:GOSUB 60000:GOSUB 50
20 PRINT"NYOMTATO ";
30 EG=4:GOSUB 60100:GOSUB 50
40 END
50 IF B=1 THEN PRINT"RENBEN.":RETURN
60 PRINT"NINCS BEKAPCSOLVA.":RETURN
70 :
60000 REM **** FLOPPY-TESZT ****
60010 TRAP 60040
60020 B=DS:B=1
60030 TRAP:RETURN
60040 B=0:RESUME 60030
60050 :
60100 REM **** NYOMTATO-TESZT ****
60110 IF EG<4 OR EG>7 THEN GOTO 60170
60120 TRAP 60160
60130 OPEN 1,EG:PRINT#1
60140 IF ST=0 THEN B=1:GOTO 60180
60150 B=0:GOTO 60180
60160 RESUME 60150
60170 PRINT"ROSSZ EGYSEGSZAM.":END
60180 CLOSE 1:TRAP:RETURN
```

```
4060 PRINT "J.............a kurzo
r lepesszáma----------------------
-------------"; INVERSE 1;"ENTER.
.........a rajz folytatasa";
4070 IF INKEY$<>CHR$ 13 THEN GO
TO 4070
4080 RANDOMIZE USR 58012: GO TO
30
6000 LET s$="
              Viszlat!
                          ": FOR a=
1 TO LEN s$-31: PRINT #0;AT 0,0;
s$ (a TO a+31);: BEEP .02,a/3: NE
XT a: STOP
7000 INPUT "A kep neve:"; LINE n
$: IF LEN n$=0 OR LEN n$>10 THEN
 GO TO 7000
7010 SAVE n$ SCREEN$ : GO TO 30
8000 PRINT AT 3,11; INVERSE 1;"S
ZIVARVANY"
8010 PLOT 20,156: DRAW 208,0: DR
AW 0,-16: DRAW -208,0: DRAW 0,16
: PLOT 20,148: DRAW 66,0: PLOT 1
66,148: DRAW 61,0
8020 PRINT AT 8,1;"Keszitette Fe
kete Balazs  1987"
8030 LET d$="
            (c) Knight software
          Nyomj meg egy gombot,
 ha indulhat a rajzolas!
                          "
8040 FOR a=1 TO LEN d$-31: PRINT
 #0;AT 0,0;d$(a TO a+31): BEEP .
06,-24
8050 IF INKEY$="" THEN NEXT a
8051 RESTORE 8054: FOR a=1 TO 16
8052 IF INKEY$<>"" THEN GO TO 80
60
8053 READ b,c: BEEP c*2,b: NEXT
a: GO TO 8040
8054 DATA 0,.1,2,.1,4,.2,4,.2,4,
.1,5,.1,4,.2,2,.2
8055 DATA 0,.1,2,.1,4,.2,4,.2,4,
.1,12,.1,11,.2,9,.2
8060 BORDER 6: PAPER 7: CLS: GO
TO 4000
9000 IF r=2 AND rp=1 THEN PLOT q
,w: PLOT X,Y
9010 IF r=2 AND rp=0 THEN PLOT I
NVERSE 1,q,w: PLOT x,y
9020 RETURN
```

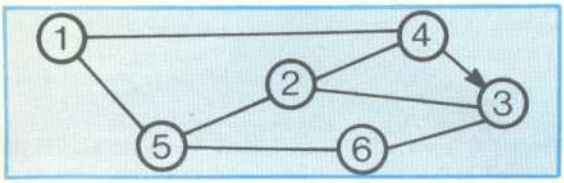
A program felépítése:
10—11: Gépi kódú programrész betöltése, ugrás a címfeliratokra
20: Értékadások
30—90: Irányítás
100—2150: Rajzolási módok és alprogramok
4000—4080: Segítő táblázat kiírása
6000: A rajzolás befejezése
7000—7010: A kép felvétele magnóra
8000—8060: Címfeliratok
9000—9020: A kurzor kirajzolása

A programot a következőképpen lehet felvenni:
SAVE „szivárvány" LINE 10

A program kezelése:
RUN-nal indítunk, majd gombnyomásra megjelenik a tájékoztató kép. Az „ENTER" gombra kezdhetjük a rajzolást. A Q, A, O, P gombokra mozog a kurzor fel, le, balra és jobbra.

FEKETE BALÁZS

Útkeresés gráfban



1. ábra

Bizonyára előfordult, hogy elképzelték, mennyire jó lenne számítógéppel megterveztetni egy utazást úgy, hogy megkapják az útra fordítandó legkevesebb időt, pénzt stb.-t. Igen ám, de azt nem tudják, hogyan lehet az ilyen optimalizálási feladatot számítógéppel megoldani. Az alábbiakban ehhez szeretnénk segítséget nyújtani két rövid programmal. Ezek a kérdések közül az utat keresik meg egy általunk megadott úthálózatban, hiszen ritka az olyan útvonal, ahol valamennyi város egymással össze lenne kötve.

Az úthálózat nem más, mint egy egyszerű gráf. (Matematikusok előnyben!) Legyen modellünk egy hat városból álló rendszer (1. ábra).

Érdekes, hogy a 3. és 4. várost összekötő úton csak a 4.-ből a 3.-ba lehet haladni s az egyirányú. Ezt a programok figyelembe veszik, hiszen az úthálózatot mátrixban tároljuk: A (x,y) jelenti az x-edik városból az y-odikba vezető utat, ahol Ø jelenti az út hiányát, bármilyen más szám pedig a létező utat. Ebben a mátrixban tárolhatjuk az utazási költségeket, amit az optimalizálásnál felhasználhatunk. Mivel azonban egyik program sem dolgozik ilyen adatokkal, az utat itt csak az 1. jelöli. A 2. ábra mutatja modellünk mátrixát. A programok nem kérik be a főátló adatait, hiszen önmagába visszatérő út nincs.

Az első program (1. lista) kiírja az összes olyan utat, amelyben egy város legfeljebb egyszer szerepel, és mi adhatjuk meg, hogy honnan hová vezessen. Az útmátrix bevitele a 10-19. sorban történik. A tulajdonképpeni keresés a 140. sortól kezdődik, miután megadtuk a kezdőhelyet és a végcélt.

F (n) az n-edik város mutatója — 1, ha már jártunk ott. Az U-tömb szolgál az eddig bejárt út tárolására, P pedig az éppen aktuális pozicióra mutat az U-tömbön belül. A program végigjárja a lehetséges utakat, és a megfelelőket kiírja. Hogyan? Legyen H az a város, ahol éppen vagyunk, keressük azt a K várost, amelybe innen el lehet jutni. Ha H és K között nincs út, akkor A/H,K/ = Ø, ha pedig már jártunk K-ban, akkor F/K/ = 1. Mindkét esetben új K-t kell keresni (18Ø). Ha K mindegyik feltételnek megfelel, akkor ez lesz az út következő állomása, a mutatók állítását a 19Ø. sor végzi. Ha NEM TALÁLUNK ILYEN K-t, akkor zsákutcába kerültünk, egy városnyit vissza kell lépni (24Ø). Ezt folytatjuk mindaddig, míg el nem érjük a célt, vagy be nem járjuk az összes lehetséges utat. A 21Ø—23Ø-as sorok a helyes út kiírását végzik, ide kerülhet tehát az útiköltség-számítás, a legvégén pedig csak az optimálist kell kiíratni. A 3. ábra a program egy futását mutatja.

2. ábra

x \ y	1	2	3	4	5	6
1	0	0	0	1	1	0
2	0	0	1	1	1	1
3	0	1	0	0	0	1
4	1	1	1	0	0	0
5	1	1	0	0	0	1
6	0	1	1	0	1	0

3. ábra

```
MELYIK VAROSBOL INDULUNK ?? 1
MELYIK VAROSBA ERKEZUNK ?? 6
Ut : 1 , 4 , 2 , 3 , 6
Ut : 1 , 4 , 2 , 5 , 6
Ut : 1 , 4 , 2 , 6
Ut : 1 , 4 , 3 , 2 , 5 , 6
Ut : 1 , 4 , 3 , 2 , 6
Ut : 1 , 4 , 3 , 6
Ut : 1 , 5 , 2 , 3 , 6
Ut : 1 , 5 , 2 , 4 , 3 , 6
Ut : 1 , 5 , 2 , 6
Ut : 1 , 5 , 6
```

```
10 INPUT"HANY VAROS VAN ?";N
20 IF N<2 OR N<>INT(N) THEN10
30 DIMA(N,N),F(N),U(N)
40 FOR X=1 TO N
50 PRINT
60 FOR Y=1 TO N
70 IF X=Y THEN90
80 PRINT"VEZET UT";X;"-BOL";Y;"-BA ?";:INPUT A(X,Y)
90 NEXT Y,X
100 PRINT
110 INPUT"MELYIK VAROSBOL INDULUNK ?";I
120 INPUT"MELYIK VAROSBA ERKEZUNK ?";E
130 IF I<1 OR I>N OR E<1 OR E>N OR I=E THEN110
140 F(I)=1:P=1:U(P)=I
150 K=1
160 H=U(P)
170 IF K>N THEN240
180 IF A(H,K)=0 OR F(K)=1 THEN K=K+1:GOTO170
190 P=P+1:U(P)=K:F(K)=1
200 IF K<>E THEN150
210 PRINT:FOR Z=1 TO P-1
220 PRINT U(Z);",";
230 NEXT Z:PRINT E
240 K=U(P):F(K)=0:P=P-1:K=K+1
250 IF P>0 THEN160
?60 GOTO100
```

1. lista

```
10 INPUT"HANY VAROS VAN ?";N
20 IF N<2 OR N<>INT(N) THEN10
30 DIMA(N,N),F(N),U(N+1)
40 FOR X=1 TO N
50 PRINT
60 FOR Y=1 TO N
70 IF X=Y THEN90
80 PRINT"VEZET UT";X;"-BOL";Y;"-BA ?";:INPUT A(X,Y)
90 NEXT Y,X
100 PRINT
110 F(1)=0:P=1:U(P)=1
120 K=1
130 H=U(P)
140 IF K>N THEN210
150 IF A(H,K)=0 OR F(K)=1 THEN K=K+1:GOTO140
160 P=P+1:U(P)=K:F(K)=1
170 IF K<>1 OR P<N+1 THEN120
180 PRINT:PRINT"UT :";:FOR Z=1 TO P-1
190 PRINT U(Z);",";
200 NEXT Z:PRINT 1
210 K=U(P):F(K)=0:P=P-1:K=K+1
220 IF P>0 THEN130
230 END
```

2. lista

```
UT : 1 , 4 , 2 , 3 , 6 , 5 , 1
UT : 1 , 4 , 3 , 2 , 6 , 5 , 1
UT : 1 , 4 , 3 , 6 , 2 , 5 , 1
UT : 1 , 5 , 6 , 3 , 2 , 4 , 1
```

4. ábra

A második program (2. lista) nagyon hasonlít az elsőhöz — nem is csoda, hisz az algoritmusa majdnem ugyanolyan, de ez valamennyi olyan körutat kiírja, amelyben egyszer minden város szerepel. Természetesen most nincs szükség indulási és érkezési helyre, hiszen akárhonnan is indulunk, ugyanoda érünk vissza. Induljunk az első városból. Addig kell tehát keresnünk, amíg vissza nem érünk az első városba. Mivel azonban mindegyiken át kell haladnunk, erre is kell egy vizsgálat (17Ø). A 4. ábra mutatja a program egy futását.

Ehhez hasonló programokkal például megkereshetjük bármilyen bonyolult gráf Hamilton-köreit, Euler-vonalait stb. Remélem ezek után senkinek nem jelent gondot majd hasonló feladat megoldása.

PALLAGI LÁSZLÓ

OKTA-TOTÓ

Az IBM személyi számítógépeiben — annak ellenére, hogy éppen a közelmúltban merült fel a módosítás lehetősége — Intel mikroprocesszorokat (8088, 8086, 80286, . . .) használnak (x).

Az IBM PC sorozat elemei (PC, PC XT, PC AT) között az alapvető különbség a sebesség (x).

Az IBM kompatibilis Commodore gép — amint azt a neve is mutatja — a PC 20 (x).

A személyi számítógépek legnépszerűbb programnyelve a BASIC. A professzionális személyi számítógépeket azonban BASIC-ben nem lehet elég hatékonyan programozni. A FORTRANT-t személyi számítógépeken nem igazán használják. A PASCAL viszont az IBM PC-hez különösen sikeres (2).

Mivel ez a kérdés (az 5.) a játékprogramokra nem vonatkozik, a sok helyen publikált első válasz helyes (1).

Az IBM PC sorozatban a felsoroltak közül az AT a legnagyobb (Advanced Technology), így az tudja a legnagyobb fizikai tárat kezelni. (x).

Ehhez 1.2 Mbájt hajlékonylemez és 40 Mbájt winchester is tartozhat (x).

Az NLQ a Near Letter Quality (közel levélminőségű) rövidítése, tehát nyomtatási képre vonatkozik (1).

A streamer mágnesszalagos információrögzítő, de a hagyományos kazettáknál jóval nagyobb kapacitású és gyorsabb (1).

Az IBM PC-k utánzatainak közismert problémája a megbízhatatlanság. A sebességük gyakorlatilag ugyanaz, a programok rendszerint futnak (x).

Az IBM PC sorozat elemei felfelé kompatibilisek (1).

A Framework és a dBASE III. egyaránt az IBM PC-kre készült (x).

A katalógus nyilván az állományneveket tartalmazza (2).

Elég közismert (különböző — publikált — teszteredmények alapján), hogy az Olivetti PC-k gyorsabbak versenytársaiknál (1).

A helyes megfejtés tehát:

1	2	3	4	5	6	7	8	9	10	11	12	13	+1
x	x	x	2	1	x	x	1	1	x	1	x	2	1

A megfejtéseket a pályázati szelvényre kell rávezetni úgy, hogy a kérdésszám sorába a játékos beírja tippjét: 1-est, 2-est vagy X-et. A közölt három válasz közül csak egy a helyes. A kitöltött szelvényt postai levelezőlapra felragasztva a szerkesztőség címére kell beküldeni (1371 Budapest, Pf.: 433).

A beküldési határidő: 1987. szeptember 20.

A helyes megoldás feladói közül havonta öt nyertest sorsolunk ki, akik 200 forint értékű könyvutalványt kapnak. Akinek mind a hat TOTÓ-ja telitalálatos, egy kis szerencsével számítógépet nyerhet! A pályázati szelvény nélkül érkezett megoldás nem vesz részt a sorsoláson!

1. A srukturált programozás
1. felosztja a vezérlőegység szabad memóriáját
2. lehetővé teszi a programozási munka elosztását több programozó között
X. megteremti az optimális arányt a perifériák és a központi egység között.

2. Mit jelent a multiprogramozás?
1. egyszerre több személy programozhatja a számítógépet
2. egymáshoz kapcsolt több központi egységgel bővítjük a gép kapacitását
X. két vagy több program ugyanazon a számítógépen látszólag egyidejűleg hajtódik végre

3. A kiegészítő aritmetikai processzor
1. meggyorsítja az aritmetikai kifejezések végrehajtását, különösen a trigonometrikus függvényekét
2. új függvényeket definiál
X. megnöveli a számábrázolás terjedelmét

4. A kiegészítő aritmetikai processzor használatánál
1. nem kell semmit tenni a gép magától tud mindent
2. a megfelelő fordítóprogramot kell behívni
X. a programba csak speciális aritmetikai kifejezéseket írhatunk

5. A szubrutin használata
1. bizonyos feladatoknál kötelező
2. meggyorsítja a program futását
X. jelentősen csökkentheti a program méretét

6. Mit értünk programozásnál a verem (stack) alatt?
1. olyan tár, amelyből a legutoljára bevitt adat érhető el először
2. a programhibákat tárolja
X. a könyvtári programokat hívhatjuk elő belőle

7. Mi a ciklusok egymásba ágyazása?
1. egy programon belül több ciklust használunk
2. cikluson belül alternatív ciklusváltozókat definiálunk
X. cikluson belül egy vagy több másik ciklust szervezünk

8. Hány ciklust ágyazhatunk egymásba?
1. géptől és programnyelvtől függ
2. bármennyit, hiszen önálló utasítások
X. csak kettőt, mert a mátrix két elemű tömb

9. Logikai elágaztatással szervezhető-e ciklus?
1. nem, erre speciális utasítás van
2. igen, de nehézkes
X. nem, mert végtelen ciklust eredményez

10. Összegzésnél (X=X+Y) meg kell-e adni a változó kezdőértékét?
1. Nem, mert a program indításánál automatikusan nulla az értéke
2. igen, mert egyéb esetben hibajelzést kapunk
X. igen, mert értéke bizonytalan

11. Hány dimenziós tömböt definiálhatunk?
1. bármilyet, a ciklusszervezéstől függ
2. géptől és programnyelvtől függ
X. csak egy dimenziósat, vektort

12. Megegyezhet-e a tömb indexváltozója a ciklusváltozóval?
1. igen, általában meg is egyezik
2. nem, mert a ciklusszervezés elve ezt kizárja
X. nem, mert a program nem találja meg a tömb számára lefoglalt memóriaterületet

13. Mivel lehet egy aritmetikai művelet pontosságát növelni?
1. a művelet többszöri végrehajtásával
2. változók definiálásával
X. semmivel, ez a gép tipusától függ

+1. A DOS-nál az EXE kiterjesztésű fájl
1. futtatható program
2. bemenő adatokat tartalmaz
X. ilyen kiterjesztésű fájl nem hozható létre

PÁLYÁZATI SZELVÉNY 87/9

OKTA-TOTÓ

Kérdés	Tipp
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
+	
13+1.	

Név: _____

Cím: _____

_____

Sz.szám:

Non scholae...

Ne féljünk, meghálálja

Virág Ferenc szolnoki általános iskolai matematika-fizika szakos tanár vaskos tanulmányt küldött szerkesztőségünk címére. Munkáját azért bocsátotta rendelkezésünkre, hogy ha úgy gondoljuk mi is, kedvcsinálónak, használandónak adjuk közre. Tanácsokat, ötleteket, tapasztalatokat rögzített írásában arról, hogyan vezették be náluk az iskolában a számítógépet. Talán nem lesz érdektelen azoknak, akik ilyen-olyan magyarázattal, de húzódoznak ettől a találmánytól. Terjedelmi okokból a teljes tanulmányt nem adhatjuk közre, de az alapgondolatot mindenesetre megmentettük belőle.

Az az iskola, ahol néhány éven belül nem használják a számítógépet, előbb-utóbb lemarad a többitől — írja a fizika-matematika szakos fiatal tanár. Vallja ezt annak az iskolának az oktatója, aki közben jól tudja, hogy diákjainak többsége hátrányos helyzetű családok gyermeke, talán otthonában soha nem tudhat majd magáénak számítógépet. Mégis fontosnak, sőt elengedhetetlenül szükségesnek tartja ezeknek a gyerekeknek is a számítógép világában való jártasságát. Sőt, bizonyára ismeri az örökbecsű latin mondást, non scolae, sed vitae discimus, vagyis, hogy nem az iskolának, az életnek tanulunk . . . De addig az iskolából kell az induláshoz nélkülözhetetlen ismereteket megszerezni. Ehhez viszont pedagógusokra, az új, a korszerű eszközökkel megbarátkozó, azokat használó oktatókra van szükség.

Az alábbiak mottójaként a „ne féljünk a számítógéptől!" felszólítást adhatnánk, s joggal, hiszen a cikkben szereplő iskolában már kipróbálták, náluk bevált, ötleteiket mások is hasznosíthatják . . .

Noha kezdetben nem tulajdonítottak nagyobb jelentőséget a számítógépeknek, elterjedésükre már ők is felfigyeltek. Első időkben némi idegenkedéssel fogadták az ismeretlen masinát, míg napjainkban óriási lelkesedéssel vonják be iskolájuk életébe. Négy idősebb tanár, s néhány frissen végzett oktatja a számítástechnikai ismereteket. Tapasztalataik alapján állítják, hogy rövid idő alatt — napi egy-két órás önképzéssel — jártasság szintjén megtanulható az oktatóprogramok írása, s természetesen a gép kezelése.

Induláshoz két gépük volt — azóta egyre gyarapodnak — ezek elégségesek voltak ahhoz, hogy tanulóik kedvet kapjanak a számítógéphez, megtanulják a billentyűk kezelését, begyakorolják a géphasználatot, és újdonsült tudásukat a tanórákon is kipróbálják. Természetesen jó néhány „kérdőjel" kísérte a „nyitányt". Például, hogy hol helyezzék el a személyi számítógépeket. Stabil, vagy változó helyszíneken használják? Az egyik és a másik mellett is voltak érvek. Végül minimális változtatással egy könyvtárszobát alakítottak át „számítógép-központtá". Ha az élet úgy kívánta, azért az osztálytermekbe is bevihették a készülékeket, illetve az osztályok számítógépes óráit helyezték át a könyvtárhelyiségbe.

A jelenlegi hat számítógép — Primo, ZX Spectrum és Commodore 16 — egyelőre elégséges a foglalkozásokhoz, ám mind több vállalkozó kedvű tanár és diák akad, így minden bizonnyal hamarosan növelni kell az állományt. Szándékuk szerint minél több azonos típusú géppel. Végül is terveik között szerepel egy számítástechnikai kabinet létrehozása . . . Addig a jelennek élnek, mellyel máris dicsekedhetnek, hiszen száznál több a maguk írta programok száma. Elsősorban játékprogramjaik vannak, de felnőttek-gyerekek együttgondolkodásából jó néhány oktatóprogram is készült.

Hogyan alakították és működtetik szakköreiket? A tanulók két csoportban dolgoznak. Az egyikben a kezdők és az „enyhe haladók", a másikban a haladó nyolcadikosok heti kétszer két órában foglalkoznak számítógéppel. Mindegyikre elmondható, hogy a szakkörösök fogékonyak, érdeklődők, rendkívül nyitottak. Kialakított terv szerint haladnak, de ez nem jelent merev tematikát, mert ha valaki társánál gyorsabban fejlődik, az magasabb szintű feladatokat is kaphat.

A vállalkozás a névadáskor a „Micro klub" nevet nyerte el. Bevallott célja pedig az, hogy a tanulók számítástechnikai kultúrával ismerkedjenek meg, a programozás rejtelmeiben jártasak legyenek, dolgozzanak öntevékenyen, s nem utolsósorban szabad idejükben érezzék jól magukat.

A számítógépes ismeretekre nyitott, fogékony tanulók rövid időn belül begyakorolták a gép és tartozékainak összeszerelését, üzembe helyezését. Aztán ahogy megismerkedtek a billentyűk kezelésével, megtanulták a programok szalagra történő kimentését és visszatöltését. Ez utóbbit jól használják a hibás programok utólagos javításánál. Csak az alapvető ismereteket mondták el a diákoknak, a bővebb információkat kézikönyvekből, szakkönyvekből és szakfolyóiratokból olvasták össze. És jól alkalmazták a kész programlisták elemzését, mert bár látszatra mechanikus a művelet, valójában gyakorolhatják vele a billentyűk használatát, s az ilyen típusú munkáknál lehetőség van programelemzésre és tapasztalatgyűjtésre.

Alig négy-öt foglalkozás után kisebb — néhány soros — önálló programot írtak a kezdő számítógépesek. A két-három tanulós csoportok valamennyi programot szalagra rögzítették, hogy a későbbiekben elemezve a munkát, a hibásból is tanuljanak. Óriási népszerűségnek örvendenek a játékprogramok, fél év alatt több mint százat írtak, s cseréltek ki egymás között a szakköri tagok. Játszanak, de ismereteiket is bővítik a foglalkozásokon, a szórakozás mellett a tanulás nem kerülhet hátrányba . . . Noha a szakköri tagság fakultatív, s a részvétel is, havonta egyszer a számítógépeseknek kötelező a megjelenés. Ilyenkor ugyanis az elmúlt hónapot értékelik, elemzik, „bonckés" alá veszik saját és társaik munkáját. Teszik mindezt őszinte, jobbító szándékkal.

Kezdetben számítógéppel csak a szakkörökben dolgoztak. Ám ahogy tudásuk bővült, s nőtt a vállalkozó kedvűek száma, elhatároztatott, hogy a tanórákat is bevonják a számítógépes munkába. A felső tagozaton már a fizika és a matematika tantárgyaknál használnak gépeket a tanításhoz. Az ezekhez szükséges programokat kezdetben a megyei pedagógiai intézettől kapták, majd módszertani folyóiratokból, számítástechnikai szaklapokból és szakkönyvekből állították össze. Végezetül pedig a saját készítésű programokkal egészítik ki az oktatóprogramok gazdag színskáláját.

A diákok ma jobb eredményeket érnek el, hiszen az órákon nemcsak tanulnak, gyakorolnak, hanem tudásukat a programokkal kontrollálhatják is.

Mit érhet el egy iskola, ha bevezeti és alkalmazza a számítástechnikát diákjai körében? Mint a tapasztalatok igazolják, korszerű technikai szemléletet alakíthat ki, munkájukat ésszerűen szervezheti, logikus és kreatív gondolkodás alapjait teremtheti meg, kiépítheti az önművelés igényét a tanulókban, s az önállóságra való törekvést, sőt bekapcsolhatja a közösséggé formálódás folyamatába, miközben esztétikai és etikai értékeket nyújt. Végül segít a szabad időt ésszerűen, hasznosan eltölteni.

Annyiféle formában panaszkodunk, hogy gond van már a tizenévesekkel is. Hazsontalan, léha, nemtörődöm az ifjúság. De vajon így van-e? Vagy csak értelmes programot nem adunk neki? Talán ezek a gyerekek rácáfolnak a vádakra, ha tartalmas, gazdag tanulóéveket tudhatnak maguk mögött . . .

KRASZNAI ÉVA

Összetett állománykezelési feladat megoldása az ATARI 800 XL-en

Előző cikkeinkben a gép ismeretére, kezelésére viszonylag egyszerűbb példákat mutattunk be. Most nézzük meg egy olyan probléma megoldását, amelyet általánosítva, a gép korlátain belül szinte tetszőleges feladatot saját magunk programozhatunk be.

A feladat viszonylag még mindig egyszerű. Ha bármilyen dologról olyan nyilvántartást készítünk, hogy ebből egy adattípus szerint tetszőlegesen tudjunk majd visszanyerni vagy kilistázni adatokat, akkor már nem elégedhetünk meg a lemezen sorban egymás után tárolt és visszakeresthető adatkezelési lehetőségekkel. Előző cikkünkben ugyan már megismerkedtünk az indexelési eljárással, de célszerű lenne az indexek alapján az állományt valamilyen adat (például a név ábécé szerinti) sorrendjébe állítani.

Ezenkívül a különféle állománykezelési műveleteket is hasznos egy programban egyesíteni, és szerkeszteni egy menüt, amely által a gépet és a programot emberközelibbé tehetjük.

Hogy mindenki számára kézzelfogható legyen, vegyük konkrét példának egy telefonkönyv szerkesztését. Természetesen bármely más állomány (könyv- vagy hanglemezgyűjtemény, iratkatalógus stb.) megszerkeszthető ezzel a módszerrel, ha már egyértelműen meghatároztuk, hogy milyen adatokat szeretnénk tárolni és melyek alapján kell az adatokat visszakeresni.

A példában a következő adatokat tároljuk:

— telefonszám (6 karakter),
— a tulajdonos neve (24 karakter),
— lakcíme (20 karakter),
— foglalkozása (12 karakter).

Ezek alkotnak egy adatmondatot vagy rekordot. A rekordot most csak egyetlen adatmező, a tulajdonos neve szerint keressük vissza vagy listázzuk ki. Ez lesz a rekord kulcsa. A megoldás alapján bárki könnyűszerrel visszakeresheti vagy sorrendbe állíthatja a rekordokat más kulcsok szerint is.

A kulcsokat egy tömbbe (indexvektorba) helyezzük el. A kulcsokkal párhuzamosan most is tárolni kell a rekordok helyét jelző mutatókat (pointer vektor). Ezeket a tömböket a program elején beolvassuk egy lemezes állományból a memóriába, és a program befejezése után — mivel futás közben a tömböt változtathatjuk — ki kell vinni az index- és mutatótömböket a lemezre. Természetesen ahhoz, hogy a név szerinti ábécésorrendet biztosíthassuk, a kulcs-

```
10 REM ******************************
11 REM * INDEXELT ALLOMANY KEZELES *
12 REM ******************************
40 DIM T$(6),N$(24),S$(24),C$(20),F$(12),R$(63),N0$(2400),S0(100),B0(100)
42 T$="":N$="":C$="":F$="":R$="":N0$=""
50 REM T:TELEFONSZAM;  N:NEV;  C:CIM;  F:FOGLALKOZAS
60 REM REKORD=T+N+C+F;  ID=INDEXVEKTOR MUTATO;  IO=INDEXVEKTOR OLVASASJELZO
70 OPEN #3,4,0,"K:"
75 GOSUB 8000:REM ALLOMANY MEGHATAROZAS
80 GRAPHICS 16
90 POSITION 12,2:PRINT " TELEFONKONYV "
100 POSITION 4,5:PRINT "1=LETREHOZAS"
110 POSITION 4,7:PRINT "2=BOVITES"
120 POSITION 4,9:PRINT "3=MODOSITAS"
130 POSITION 4,11:PRINT "4=TORLES"
140 POSITION 4,13:PRINT "5=LEKERDEZES"
150 POSITION 4,15:PRINT "6=VEGE"
160 GET #3,C
170 IF C<49 OR C>54 THEN 160
180 ON C-48 GOSUB 1000,2000,3000,4000,5000,6000
200 GOTO 80
1000 REM **************
1001 REM * LETREHOZAS *
1002 REM **************
1005 IO=1
1008 ID=0
1010 OPEN #1,8,0,DX$
1020 REM ***************
1021 REM * ADATBEVITEL *
1022 REM ***************
1025 GRAPHICS 16:POSITION 12,2:PRINT " ADATBEVITEL "
1030 POSITION 4,5:PRINT "TELEFONSZAM";:INPUT T$
1040 POSITION 4,7:PRINT "TUL. NEVE";:INPUT N$
1050 POSITION 4,9:PRINT "TUL. CIME";:INPUT C$
1060 POSITION 4,11:PRINT "FOGLALKOZASA";:INPUT F$
1070 IF LEN(T$)<6 THEN T$(LEN(T$)+1)="      "
1080 IF LEN(N$)<24 THEN N$(LEN(N$)+1)="                        "
1090 IF LEN(C$)<20 THEN C$(LEN(C$)+1)="                    "
1100 IF LEN(F$)<12 THEN F$(LEN(F$)+1)="            "
1110 R$(1,6)=T$:R$(7,30)=N$:R$(31,50)=C$:R$(51,62)=F$
1120 POSITION 6,15:PRINT " TAROLHATOK AZ ADATOK? (I/N) "
1130 GET #3,C
1140 IF C=78 THEN 1210:REM 'NEM'
1150 IF C<>73 THEN 1130:REM 'IGEM'
1160 NOTE #1,S,B
1170 PRINT #1;R$
1180 ID=ID+1:REM ID=AZ ALLOMANY TENYLEGES MERETE
1190 S0(ID)=S:B0(ID)=B
1200 N0$((ID-1)*24+1)=N$
1210 POSITION 6,18:PRINT " VAN MEG ADAT? (I/N) "
1220 GET #3,C
1230 IF C=73 THEN 1020:REM 'IGEN'
1240 IF C<>78 THEN 1220:REM 'NEM'
1250 CLOSE #1
1255 IO=1
1260 GOSUB 7000:REM INDEXTABLA RENDEZES
1300 REM **********************
1310 REM * INDEXTABLA KIIRAS *
1320 REM **********************
1330 OPEN #2,8,0,IX$
1340 PRINT #2;ID
1350 FOR I=1 TO ID
1360 N$=N0$((I-1)*24+1):S=S0(I):B=B0(I)
1370 PRINT #2;N$:PRINT #2;S:PRINT #2;B
1380 NEXT I
1390 CLOSE #2
1400 RETURN
2000 REM ***********
2010 REM * BOVITES *
2020 REM ***********
2040 IF IO=0 THEN GOSUB 6500:REM INDEXTABLA BEOLVASASA
2050 OPEN #1,9,0,DX$
2060 GOSUB 1020
2080 RETURN
```

```
3000 REM *************
3010 REM * MODOSITAS *
3020 REM *************
3030 IF IO=0 THEN GOSUB 6500:REM AZ INDEXTABLA JELENLETET JELZI - IGEN=1
3040 GOSUB 5430
3045 IF N$="*" THEN 3180
3050 POSITION 2,8:PRINT "TELEFONSZAM:";:INPUT T$
3060 POSITION 2,12:PRINT "CIM:";:INPUT C$
3065 POSITION 2,14:PRINT "FOGLALKOZAS:";:INPUT F$
3070 OPEN #1,12,0,DX$
3080 IF LEN(T$)<6 THEN T$(LEN(T$)+1)="      "
3090 IF LEN(C$)<20 THEN C$(LEN(C$)+1)="                    "
3100 IF LEN(F$)<12 THEN F$(LEN(F$)+1)="          "
3110 R$=T$:R$(7)=N$:R$(31)=C$:R$(51)=F$
3120 POSITION 6,17:PRINT " TAROLHATOK AZ ADATOK? (I/N) "
3130 GET #3,C
3140 IF C=78 THEN 3180:REM 'NEM'
3150 IF C<>73 THEN 3130:REM 'IGEN'
3155 POINT #1,S,B
3160 PRINT #1;R$
3180 POSITION 6,19:PRINT " VAN MEG MODOSITAS? (I/N)"
3190 GET #3,C
3200 IF C=73 THEN CLOSE #1:GOTO 3040:REM 'IGEN'
3210 IF C<>78 THEN 3190:REM 'NEM'
3220 CLOSE #1
3999 RETURN
4000 REM **********
4010 REM * TORLES *
4020 REM **********
4030 IF IO=0 THEN GOSUB 6500
4040 GRAPHICS 16
4050 POSITION 4,2:PRINT " TORLES AZ ALLOMANYBOL "
4060 POSITION 2,4:PRINT "KINEK AZ ADATAIT TOROLJUK"
4065 POSITION 2,6:INPUT N$:GOSUB 7530
4068 IF N$="*" THEN 4200
4070 GOSUB 5500:REM A KERESETT NEVU TULAJDONOS KIIRASA KEPERNYORE
4080 POSITION 4,18:PRINT "TENYLEG TOROLHETO (I/N)?"
4090 GET #3,C
4100 IF C=78 THEN 4200
4110 IF C<>73 THEN 4090
4120 GOSUB 7700:REM TORLES INDEXTABLABOL ES POINTERVEKTORBOL:ID=ID-1
4200 POSITION 4,20:PRINT "AKAR-E MEG TOROLNI (I/N)?"
4210 GET #3,C
4220 IF C=73 THEN 4040
4230 IF C<>78 THEN 4210
4235 GOSUB 1330
4999 RETURN
5000 REM **************
5010 REM * LEKERDEZES *
5020 REM **************
5030 IF IO=0 THEN GOSUB 6500
5040 GRAPHICS 16
5050 POSITION 8,2:PRINT " LEKERDEZES "
5060 POSITION 2,6:PRINT "1=NEVSOR SZERINTI LISTA"
5070 POSITION 2,9:PRINT "2=KERESES NEV SZERINT"
5075 POSITION 2,12:PRINT "3=KILEPES"
5080 GET #3,C
5090 IF C=49 THEN GOSUB 5200:GOTO 5040
5092 IF C=50 THEN GOSUB 5400:GET #3,C:GOTO 5040
5094 IF C<>51 THEN 5080
5100 RETURN
5200 REM ********************
5210 REM * LISTA NEVSORBAN *
5220 REM ********************
5225 OPEN #1,4,0,DX$:REM BELEPES A NEV SZERINTI KERESESBE
5230 FOR I=1 TO ID
5240 S=S0(I):B=B0(I)
5250 GRAPHICS 16
5260 POINT #1,S,B
5262 INPUT #1,R$
5264 TRAP 5275
5266 T$=R$:N$=R$(7):C$=R$(31):F$=R$(51)
5268 GOSUB 5600
5269 GET #3,C
5270 NEXT I
5275 CLOSE #1
5280 RETURN
5400 REM ************************
5410 REM * KERESES NEV SZERINT *
5420 REM ************************
5430 GRAPHICS 16
5440 POSITION 2,4:PRINT "A KERESETT NEVE":INPUT N$:GOSUB 7530
5450 IF N$="*" THEN 5599
5500 OPEN #1,4,0,DX$:REM BELEPES A NEV SZERINTI KERESESBE
5510 POINT #1,S,B
5520 INPUT #1,R$
5530 T$=R$:N$=R$(7):C$=R$(31):F$=R$(51)
```

tömböt rendezni kell úgy, hogy a mutatókat a kulcsokkal együtt mozgatjuk.

Az állománykezelési és egyéb műveletek kiválasztását egy-egy menü szerint végezhetjük el. A program indításakor és minden művelet vagy műveletcsoport befejezése után a következő menüt jelenítsük meg és kezeljük le:

1. Létrehozás (alapállomány kialakítása)

2. Bővítés (új tulajdonos adatainak bevitele meglevő állományba)

3. Módosítás (egy tulajdonos adatainak, például telefonszámának megváltoztatása: ennél a műveletnél tételezzük fel, hogy a kulcs — a tulajdonos neve — nem változik)

4. Törlés (a neve alapján kiválasztott tulajdonos adatainak törlése az állományból és az indextábla módosítása)

5. Lekérdezés (névsor szerint vagy választott név alapján egy-egy képernyőn jeleníti meg az egyes rekordokat)

6. Kilépés a programból

A programot bőségesen elláttuk megjegyzésekkel, az egyes blokkokat elkülönítettük.

Sok sikert kívánunk a felhasználáshoz és az igények szerinti módosításához.

Végül emberi nyelven az ATARI-ról

Cikksorozatunkat azzal kezdtük, hogy szeretnénk elfogultság nélkül bemutatni, szubjektív színezetű értékítélet mellőzésével összehasonlítani az ATARI-t és a ma már különböző területeken a gyakorlat részévé vált C64-et. Már csak azért is indokolt volt ez a tartózkodás, mert az ATARI rövid hazai szakmai múltja, kismértékű elterjedtsége nem adott módot számottevő tapasztalati anyag összegyűjtésére. Mégis — most az utolsó szó jogán — befejezésül elmondjuk azokat a tapasztalatokat, benyomásokat, esetleg véleményeket, amelyek sorozatunk főszereplőjével kapcsolatban bennünk megfogalmazódtak. Tesszük ezt azzal a bátorsággal, hogy olvasóink közül azoknak, akik mint a gép birtokosai, használói, nyomon követték sorozatunkat, kipróbálták a leírtakat, már van alapjuk némi személyes értékítéletre; akik pedig a jövő ATARI-sai, azoknak talán lerövidíthetjük a célhoz vezető utat, ha véleményünket elfogadják referenciának.

Mi is az ATARI? Egy a személyi számítógépek, „home computer"-ek közül. Ahhoz azonban, hogy közelebb jussunk az igazsághoz, a célból kellene kiindulni: mi a feladata ennek a gépnek? Mi igazolja a létjogosultságát? Mit ad a hazai amatőr és szakértő számítástechnikusnak? Ezekre a kérdésekre mindvégig nem tudtunk pozitív választ adni.

A számítógép környezetfüggő technika. Ugyan egy „újszülöttnek minden vicc új", egy ma kezdő felhasználónak minden számítógép „a gép", de akiknek előzőleg módjuk volt más gépeket is megismerni, azokon tapasztalatokat szerezni, óhatatlanul ha-

```
5540 GOSUB 5600
5550 CLOSE #1
5599 RETURN
5600 REM **********************
5610 REM * KEPERNYO FORMATUM *
5620 REM **********************
5640 POSITION 2,8:PRINT "TELEFONSZAM: ";T$
5650 POSITION 2,10:PRINT "NEV: ";N$
5660 POSITION 2,12:PRINT "CIM: ";C$
5670 POSITION 2,14:PRINT "FOGLALKOZAS: ";F$
5699 RETURN
6000 REM *************
6010 REM * BEFEJEZES *
6020 REM *************
6030 END
6500 REM ***************************
6510 REM * INDEXTABLA BEOLVASASA *
6520 REM ***************************
6530 OPEN #2,4,0,IX$
6540 INPUT #2,ID
6550 FOR I=1 TO ID
6560 INPUT #2,N$:INPUT #2,S:INPUT #2,B
6570 N0$((I-1)*24+1)=N$:S0(I)=S:B0(I)=B
6580 NEXT I
6590 CLOSE #2
6600 IO=1
6999 RETURN
7000 REM ************
7010 REM * RENDEZES *
7020 REM ************
7030 FOR I=1 TO ID-1
7040 I1=(I-1)*24+1:I2=I*24
7050 J=I+1
7060 FOR K=J TO ID
7065 S$=""
7070 K1=(K-1)*24+1:K2=K*24
7080 IF N0$(I1,I2)<N0$(K1,K2) THEN 7110
7090 S$=N0$(I1,I2):N0$(I1,I2)=N0$(K1,K2):N0$(K1,K2)=S$
7100 N=B0(I):B0(I)=B0(K):B0(K)=N:N=S0(I):S0(I)=S0(K):S0(K)=N
7110 NEXT K
7120 NEXT I
7130 RETURN
7500 REM ***********
7510 REM * KERESES *
7520 REM ***********
7530 N$(LEN(N$)+1)="
7540 N$=N$(1,24)
7550 L=INT(LOG(ID)/LOG(2)+1)
7560 AH=1:FH=ID+1
7570 K=INT((AH+FH)/2)
7580 K0=(K-1)*24+1:K1=K*24
7590 IF N0$(K0,K1)=N$ THEN O=K:S=S0(K):B=B0(K):RETURN
7600 IF N0$(K0,K1)>N$ THEN FH=K:GOTO 7620
7610 AH=K
7620 L=L-1:IF L=0 THEN PRINT "NINCS ILYEN":N$="*":RETURN
7630 GOTO 7570
7700 REM ***************************
7710 REM * TORLES AZ INDEXTABLABOL *
7720 REM ***************************
7740 FOR I=0+1 TO ID
7750 K0=(I-1)*24+1:K1=I*24
7760 N0$(K0-24,K1-24)=N0$(K0,K1)
7770 S0(I-1)=S0(I):B0(I-1)=B0(I)
7780 NEXT I
7782 ID=ID-1
7790 RETURN
8000 REM ***************************
8010 REM * ALLOMANY MEGHATAROZAS *
8020 REM ***************************
8030 GRAPHICS 16
8040 DIM W$(7),AN$(8),DX$(12),IX$(14)
8050 AN$="D:"
8060 POSITION 1,4:PRINT "ALLOMANYNEV (MAX. 6 BETU!)";:INPUT W$
8070 IF LEN(W$)>6 THEN 1004
8080 AN$(3)=W$
8090 DX$=AN$:DX$(LEN(DX$)+1)=".DAT"
8100 IX$=AN$:IX$(LEN(IX$)+1)=".NDX"
8110 RETURN
```

sonlítanak, rangsorolnak. Ebben az értelemben az ATARI egy nálunk merőben idegen technika megtestesítője. Még akkor is, ha sajátos interpreterjével közelebb áll az eredeti BASIC-koncepcióhoz, mint a nálunk már elterjedt Commodore és Sinclair gépek.

Vagy másképpen: Commodore és Sinclair szoftverrel, hardverleírásokkal és kiegészítőkkel akár csak kielégítően — hát még ha jól is! — ellátott környezetben az ATARI szoftvertelensége (és e téren a terjesztők igyekezetének hiánya), egyáltalán: a bántó információhiány nem ad magyarázatot a gép céljait illetően.

Nem hisszük, hogy cél lehet egy olyan országban, ahol valamely kultúrát kisszámú egyed nagy tipusgazdagsága testesít meg, egy újabb típussal növelni az amúgy is nagyszámú, nehezen összehasonlítható (inkompatibilis) technikák számát. Ez mindenképpen a típusrendszerek kiépítésének gátja, a gazdaságtalan szoftveregyediség irányába mutat.

Mesterségesen magas ára (kezdetben csaknem 100 ezer forint, s még a cikk írásakor, június elején is mintegy 50 ezer forint) sem szól amellett, hogy itt egy szorgalmazni való kultúra alkalmas eszközének terjesztéséről van szó. Összevetve az ekkor már piacon lévő — bár legfeljebb csak forintban összehasonlítható — Enterprisezal, melynek ára 20 ezer forint alatt van, de szoftverrel egyelőre éppúgy ellátatlan, ezt a kétséget is megerősítve látjuk. Igaz, hogy az Enterprise csak szalagos egységgel rendelkezik, viszont az ATARI-floppy nem ad annyival többet az amatőrnek, a szakfelhasználónak pedig ezzel együtt kevés.

Nem vagyunk marketing szakemberek, nem pályázunk ilyen babérokra, és nem is vetemedünk arra, hogy a gép hazai „kizárólagos importőr"-ének és terjesztőjének ötletet adjunk. Mindazonáltal nincs kizárva, hogy igen jól szolgálná az üzleti érdekeket, ha felhasználói referenciákról és nagymértékű szoftverellátásról gondoskodna a vállalat, netán gépek ideiglenes kihelyezésével is. Mindenesetre rendszergazdák és szoftverházak tapasztalatai bizonyítják ennek az útnak a járhatóságát.

Cikksorozatunkkal szerettük volna enyhíteni az információhiányt, kezdő szintű útmutatást adva a gép programozásához, felhasználáshoz.

A felhasználási lehetőségeket, területeket a teljesség igénye nélkül tárgyaltuk. Nem tértünk ki például az igen szinvonalas grafikai lehetőségekre vagy a hanggenerálásra. Ezek azonban egy igen sajátos felhasználói területet — a számítástechnikán belül is külön szakmát — képviselnek. Mi az adatkezelési gyakorlatot részesítettük előnyben. Ha ezzel hozzá tudtunk járulni ahhoz, hogy a gép felhasználói megkönnyíthetik napi munkájukat, egyáltalán magának a gépnek a használatát jobban megismerték, abban a reményben búcsúzunk, hogy soraink nem voltak hiábavalók.

JÁNOSA—PAÁL—SÜTŐ

BASIC és gépi kód

Legutóbb a léptető utasításokról volt szó, most a logikai utasításokkal ismerkedünk meg. Mint az előző alkalommal, most is megvizsgáljuk egy — korábban megjelent — gépi kódú rutin működését.

A logikai utasítások

Három utasítás tartozik ebbe a csoportba: az AND, az ORA és az EOR. Ezek az A regiszter és az utasítás operandusa által meghatározott memóriabájt tartalmával bitenként végzik el a kijelölt logikai műveletet. Az eredményt az A regiszterben kapjuk meg.

A logikai utasítások a Z és N állapotbiteket állítják be az eredménynek megfelelően: N a 7. bittel megegyező tartalmú lesz, Z pedig akkor lesz 1, ha a művelet eredménye 0.

Az utasítások ismertetésénél az A regiszterre, mint a logikai művelet egyik operandusára, röviden az operandus szóval hivatkozom.

Az AND utasítás

Hatása hasonló a AND BASIC utasításéhoz. Az operandusok között a logikai ÉS műveletet végzi el bitenként. Az eredménynek az a bitje lesz 1 értékű, melynek a pozícióján mindkét operandusban 1 van, minden más biten 0 lesz.

Az ORA utasítás

Hatása hasonló a OR BASIC utasításéhoz. Az operandusok között a logikai VAGY műveletet végzi el bitenként. Az eredménynek az a bitje lesz 1 értékű, melynek a pozícióján a két operandus közül legalább az egyikben 1 van, az összes helyen 0 lesz.

Az EOR utasítás

Neve az angol Exclusive OR (kizáró VAGY) kifejezés betűiből származik. A két operandus között bitenkét végrehajtja a kizáró VAGY logikai műveletet. Az eredménynek az a bitje lesz 1 értékű, melynek pozícióján a két operandus különböző értéket tartalmaz, vagyis az adott helyen az egyik operandusban 0 van, a másikban 1.

A BASIC-ben nincs az EOR-nak megfelelő utasítás, de a BASIC logikai utasításaival többféleképpen is előállítható. Íme két példa:

X EOR Y:=(X AND NOT Y) OR (NOT X AND Y)

A EOR B:=(A OR B) AND NOT (A AND B)

Az első képlet jobban szemlélteti a kizáró VAGY tulajdonságait, az utóbbi [1] viszont rövidebb.

A NOT BASIC utasításnak az EOR #$FF assembly utasítás feleltethető meg. Ez az A regiszter összes bitjének értékét az ellenkezőjére változtatja.

A kizáró VAGY-tól való megkülönböztetés céljából az OR logikai műveletet megengedő VAGY-nak is nevezik.

Több helyen — például a Z80 assemblerében — találkozhatunk a XOR utasítással. Ez pontosan megfelel az EOR-nak, csak nevében különbözik.

A programokról

Az 1. listán egy korábban közölt program [2] disassemblált változata látható. Annak idején a BASIC betöltő jelent meg, és egyrészt a gépi kódú rutinnak a BASIC RAM végén történő elhelyezésére, másrészt a SYS utasítással az A regiszteren át történő paraméteradásra mutatott be példát. A rutin megszámolja az A regiszter 1-es bitjeit, és a képernyő aktuális kurzorpozíciójába írja.

A lista a C64-en futtatott változatról készült, de a kódrésze pontosan megegyezik a másik két géptípuson futtatható változatokkal.

Lássuk, hogyan működik ez a rutin.

$9FEE: az X regiszterben fogjuk számlálni az 1-es biteket, nullát viszünk a regiszterbe.

$9FFO . . . $9FF2: megvizsgáljuk az A regiszter tartalmát, ha az 0, akkor a regiszterben nincs több 1-es bit, a befejező részre ugrunk.

$9FF4: az A regiszter tartalmát eggyel balra toljuk, a bal szélső bit tartalma a C bitbe kerül, a jobb szélén egy 0 bit lép be. ASL helyett itt LSR utasítást is használhattunk volna, mert a bitek helyértékét figyelmen kívül hagyjuk, csak a darabszám fontos.

$9FF5: ha a regiszterből a C-be tolt bit értéke 0 volt, a számláló megnövelése nélkül a CMP utasításra ugrunk.

$9FF7 . . . $9FF8: eggyel megnöveljük a számláló tartalmát, és a CMP utasításra ugrunk. Itt a BPL feltételes ugróutasításnak látszik, valójában az ugrás mindig megtörténik, ugyanis X tartalma ezen a ponton sohasem lehet negatív. Lehetne itt JMP utasítás is, de nem célszerű.

$9FFA . . . 9FFB: a számláló tartalmát átvisszük az A regiszterbe. Ott így egy 8-nál nem nagyobb pozitív szám van. A számjegyek ASCII kódja $30-cal nagyobb, mint a számjegy értéke. Egyszerű lenne ezt a számot hozzáadni a regiszter tartalmához, de az két utasítást igényelne. Mivel a regiszterben lévő számnak a felső, a $30-nak pedig az alsó félbájtján csak 0-bitek vannak, az ORA művelet ugyanazt az eredményt adja, mint az összeadás.

$9FFD: a BSOUT KERNAL rutinra ugrunk, amely a képernyőre írja az eredményt.

A rutinnak a 2. listán látható változata két utasítással rövidebb. A befejező részben egy BASIC ROM rutinra ugrunk, amely az A és X regiszterek tartalmát egy kétbájtos, előjel nélküli egész számként kezelve az aktuális kurzorpozíciótól kezdve a képernyőre írja. A magasabb helyértékű résznek az A regiszterben kell lennie. Amikor a rutin futásánál ide érünk, az A regiszterben 0 van, X-ben a kiírandó szám.

A rövidítésnek ára van. Az új rutin csak C64-en futtatható. Ha másmilyen típusú gépen akarjuk futtatni, meg kell változtatni a JMP utasítás operandusát. A ROM-rutin a többi gépen is megtalálható. Címe: a VC20-on $DDCD, a C16-on $A45F.

Irodalom

1. Lothar Englisch: Gépi kódú programozás a Commodore 64-esen. Data Becker — Novotrade 1985.

2. BASIC és gépi kód. Mikroszámítógép Magazin 1986/6.

BARNA LÁSZLÓ

1. lista

```
9fee  a200      ldx #$00
9ff0  c900      cmp #$00
9ff2  f006      beq $9ffa
9ff4  0a        asl a
9ff5  90f9      bcc $9ff0
9ff7  e8        inx
9ff8  10f6      bpl $9ff0
9ffa  8a        txa
9ffb  0930      ora #$30
9ffd  4cd2ff    jmp $ffd2
```

2. lista

```
9fee  a200      ldx #$00
9ff0  c900      cmp #$00
9ff2  f006      beq $9ffa
9ff4  0a        asl a
9ff5  90f9      bcc $9ff0
9ff7  e8        inx
9ff8  10f6      bpl $9ff0
9ffa  4ccdbd    jmp $bdcd
```

Z80 programok haladóknak Spectrumra és Primóra

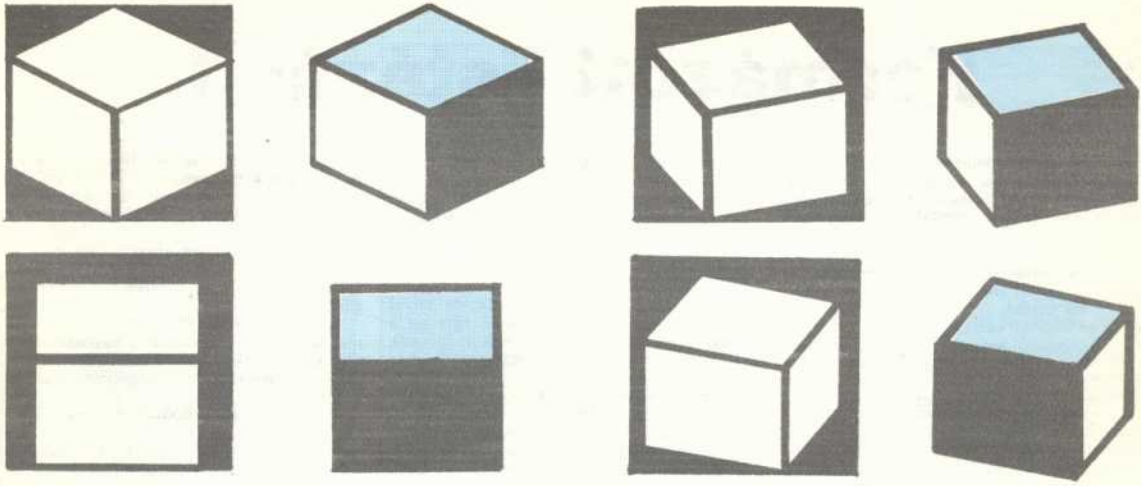
3. A manókezelő program illusztrációja

A program *(1. lista)* látványosan érzékelteti az előző részben szereplő manókezelő program képességeit, és egyúttal jó példája a manókezelő helyes használatának.

A demo tíz manót mozgat; ehhez (teljesen véletlenül) éppen tíz manóalakra van

1. lista

```
;*****************************************
;*          Specsprite demo              *
;*****************************************
DEMO    LD   HL,TABLE      ;manófájl kezdeti a-
        LD   DE,MANOF      ; datait a manófájl-
        LD   BC,32         ; ba tölti.
        LDIR
        CALL ALAPOZ        ;képernyő =) háttér
DEMO_1  LD   HL,MANOF+1    ;első manóelem címe.
        LD   DE,SPEEDS     ;sebességek táblázat
        LD   B,10            ;tíz manót mozgat
DEMO_2  CALL ROT           ;az alak módosítása
        INC  HL            ;manóelem köv.tétele
        CALL XMOVE         ;x koord. módosítása
        INC  HL            ;manóel.köv.tétele y
        INC  DE            ;y seb.a táblázatban
        CALL YMOVE         ;y koord módosítása
        INC  HL            ;köv. manóelemre lép
        INC  DE            ;köv. seb. x-re lép
        DJNZ DEMO_2        ;minden manót így.
        CALL BREAK         ;break-vizsgálat
        LD   HL,MANOF
        CALL SPECSP        ;manókezelő program
        JR   DEMO_1
;-----------------------------------------
BREAK   LD   A,$7F         ;sor kiválasztás
        IN   A,($FE)       ;bill.sor olvasás
        RRCA               ;break b. carrybe
        RET  C             ;ha nincs lenyomva
        POP  BC            ;a demot hívó rend-
        RET                ; szerbe tér vissza
;-----------------------------------------
ROT     LD   A,(HL)        ;manóalakcím
        CP   DS+8          ;8. és 9. kiszűrése
        RET  NC
        LD   A,B           ;cklusv. vizsgálat:
        CP   6             ; manók fele balra
        JR   C,FORG_1      ; fele jobbra forog
        DEC  (HL)          ;köv. alak
        LD   A,(HL)
        CP   DS-1          ;körbefordult?
        LD   (HL),A
        RET  NZ            ; ha nem
        LD   (HL),DS+7     ; ha igen
        RET
FORG_1  INC  (HL)          ;balra forgás
        LD   A,(HL)
        CP   DS+8
        LD   (HL),A
        RET  NZ
        LD   (HL),DS+0
        RET
;-----------------------------------------
XMOVE   LD   A,(DE)        ;seb. x komponens
        ADD  A,(HL)        ;koordináta növelés
        LD   (HL),A
        CP   224           ;a képernyő szélén
MOVE_1  RET  C             ; visszapattan
        LD   A,(DE)
        NEG                ;x seb.előjelet vált
        LD   (DE),A
        ADD  A,(HL)
        LD   (HL),A
        RET
;-----------------------------------------
YMOVE   LD   A,(DE)        ;y irányú mozgatás
        ADD  A,(HL)
        LD   (HL),A
        CP   160           ;képernyő teteje 159
        JR   MOVE_1
;-----------------------------------------
TABLE   DEFB 3             ;manóelem 3 bájt
        DEFB DS+3,  0,  0  ;1. manóelem
        DEFB DS+1,  8, 14  ;2. manóelem
        DEFB DS+6,100, 50  ;...
        DEFB DS+5, 40,150
        DEFB DS+0,200,  0
        DEFB DS+4,130, 95
        DEFB DS+7,200,150
        DEFB DS+2, 70, 21
        DEFB DS+8, 50, 50
        DEFB DS+9,100,100  ;10.manóelem
        DEFB 0             ;manófájl végjel
;-----------------------------------------
SPEEDS  DEFB 1,3           ;1. manó seb. x,y
        DEFB 6,5           ;...
        DEFB 2,8
        DEFB 4,2
        DEFB 5,7
        DEFB 3,1
        DEFB 7,4
        DEFB 8,6
        DEFB 6,1
        DEFB 8,2           ;10. manó sebessége
;-----------------------------------------
DS      EQU  $F1           ;első manóalak címe
;-----------------------------------------
```



2. lista

```
AZ 1. MANOALAK-MASZK

F100:FF,FF,FF,FF,FF,FE,7F,FF
F108:FF,F8,1F,FF,FF,E0,07,FF
F110:FF,80,01,FF,FE,00,00,7F
F118:F8,00,00,1F,E0,00,00,07
F120:C0,00,00,01,80,00,00,07
F128:8C,00,00,19,83,00,00,61
F130:80,C0,01,81,80,30,06,01
F138:80,0C,18,01,80,03,60,01
F140:80,00,80,01,80,00,80,01
F148:80,00,80,01,80,00,80,01
F150:80,00,80,01,80,00,80,01
F158:80,00,80,01,80,00,80,01
F160:C0,00,80,03,F0,00,80,0F
F168:FC,00,80,3F,FF,00,80,FF
F170:FF,E0,83,FF,FF,F8,8F,FF
F178:FF,FE,8F,FF,FF,FF,FF,FF

AZ 1. MANOALAK-KEP

F180:00,01,80,00,00,06,E0,00
F188:00,1D,58,00,00,6A,AE,00
F190:01,D5,55,80,06,AA,AA,E0
F198:1D,55,55,58,6A,AA,AA,AE
F1A0:D5,55,55,55,BA,AA,AA,AF
F1A8:8D,55,55,5F,83,AA,AA,FF
F1B0:80,D5,55,FF,80,3A,AF,FF
F1B8:80,0D,5F,FF,80,03,FF,FF
F1C0:80,00,FF,FF,80,00,FF,FF
F1C8:80,00,FF,FF,80,00,FF,FF
F1D0:80,00,FF,FF,80,00,FF,FF
F1D8:80,00,FF,FF,80,00,FF,FF
F1E0:C0,00,FF,FF,30,00,FF,FC
F1E8:0C,00,FF,F0,03,00,FF,C0
F1F0:00,E0,FF,00,00,18,FC,00
F1F8:00,06,F0,00,00,01,C0,00
```

```
1 ;-----------------------------------------
2 BREAK  IN   A,(63)      ;break gomb cime
3        RRCA             ;bill =) carry
4        RET  NC          ;ha nincs lenyomva
5        POP  BC          ;a demot hivo rend-
6        RET              ; szerbe ter vissza.
7 ;-----------------------------------------
```

3. lista

szüksége. Így sajnos a manóalakok összesen $10 \times 256 = 2560$ bájtos fájlt alkotnak, amelyet hosszúsága miatt nem lehet hexa listában közölni. A következő áthidaló megoldást választottam. Az első négy manóalak rajzolatát (külön a maszkot és a képet) bemutatom az *ábrán*, az elsőt pedig emellett még hexa listán is *(2. lista)*. A második négy manóalak az első négyhez nagyon hasonlóan egy újabb negyedfordulata a kockának. A különbség csak annyi, hogy a fekete és fehér oldalak fel vannak cserélve. A kék fedőlapokat sakktáblamintázattal rajzoljuk meg. A két utolsó manóalak a két állandó alakú manó lesz, megtervezésüket az olvasóra bízom. Az elkészült manófájlt # F100 címen kell elhelyezni. (A sorozat 4. — következő — részétől kezdve egy előnyös manóalak-szerkesztő programot kezdek el bemutatni, az ábrák alapján azzal majd gyorsan és kényelmesen el lehet készíteni a manóalakfájlt.)

Hogyan használja a demo program a manókezelőt?

1. A TABLE táblázat tartalmazza a manófájl kezdeti állapotát. A MANOF nevű, 20 bájtos tároló a manófájl. Indításkor a demo ide másolja be a TABLE tartalmát.

2. Ezután az ALAPOZ rutin hívásával az aktuális képtartalmat a HATTER-be teszi. Ez lesz az a kép, ami előtt a manók mozogni fognak.

3. A manók mozognak és változtatják alakjukat. Ez azt jelenti, hogy a manófájl állandóan változik. Tehát a manókat mozgató ROT, XMOVE YMOVE rutinok a manófájlt módosítják megfelelő módon. A manóknak különböző a sebességük, ehhez a SPEEDS táblázat tartalmazza a sebességvektorok x, y komponenseit. A ROT rutin az alsó 8 manót „forgatja". Ez tulajdonképpen csak annyit jelent, hogy a manófájlba a következő mozgásfázisnak megfelelő manóalakcímet írja be. A 9. és 10. manó nem forog, az ő manóelemükben a manóalakcím állandó. Ha kész a manófájl új állapota, egy breakvizsgálat kell, hogy a demót meg lehessen állítani. Utána HL-be betöltve a manófájl címét, meg lehet hívni a manókezelőt.

4. A manók mozgatásánál folytatódik a végrehajtás.

Befejezésül: Primóhoz csak a breakvizsgáló rutint kell átírni. Ez a *3. listán* látható.

UHERKOVICH PÉTER

Formázott listázás I.

Az itt közölt programmal olyan listákat tudunk előállítani, amelyek az eredeti listáknál jobban olvashatók és megkönnyítik a programok dokumentálását, reprodukálását. Mindez a következő plusz szolgáltatásokkal valósítható meg:

- a nyomtatási kép meghatározása: laponkénti sorok száma, soronkénti jelek száma,
- az oldalak sorszámozása,
- különálló programrészek külön oldalon kezdődő nyomtatásának kijelölése,
- az utasítások sorszámának kiemelése,
- az önálló BASIC utasításszavak automatikus különírása, például THEN-GOSUB helyett THEN GOSUB,
- a grafikus karakterek emlékeztető kódokkal való felcserélése,
- BASIC utasítássoronként egy kontrollösszeg előállítása és felírása a programlistára. (A kontrollösszeg előállításának algoritmusát és felhasználását a következő havi számban olvashatják az érdeklődők.)

Nézzük most az emlékeztető kódokat (lásd a *táblázatot*). A programlistán szögletes zárójelek között megjelenő kifejezések a kívánt vezérlőfunkciók angol nyelvű rövidítéseiből származnak. Nem célszerű ezeket a rövidítéseket magyarra fordítani, mert a billentyűzeten is ezek a feliratok szerepelnek. A billentyűkön azonban egy gombhoz gyakran négyféle funkciót is rendeltek, amelyek közül a megfelelő funkció a következőképpen választható ki:

- a billentyű önálló leütése
- a SHIFT és a kért billentyű egyidejű leütése
- a CTRL és a kért billentyű egyidejű leütése
- a Commodore (C=) és a kért billentyű egyidejű leütése

A táblázat második oszlopa tartalmazza a programlistákon megjelenő kívánt funkciót, a harmadik oszlop pedig az ehhez szükséges leütendő billentyűket adja meg. A program begépelésekor tehát nem kell sem a szögletes zárójeleket, sem a zárójelek között szereplő betűket beírni, hanem a zárójelben lévő funkciót kell a táblázat harmadik oszlopában leírt billentyűk leütésével kiválasztani.

Ha valaki saját részére más emlékeztető kódokat akar szerepeltetni vagy más kódokat is (esetleg a BASIC utasításszavakat is) az eredeti listázóprogramtól eltérően szeretne a nyomtatásban megjelentetni, akkor ezt is könnyen megteheti. Ha áttanulmányozza a program leírását, látni fogja, hogy a programban a gép által tárolt bármelyik BASIC alapszót vagy bármelyik ASCII kódhoz rendelt kiírandó jelet kicserélheti a saját elképzelésének megfelelően.

Programlistakódok

Az eredeti programlisták az ASCII kódokat grafikus ábrákkal jelenítik meg. Ezek az ábrák nyomtatásban nehezen értelmezhetők, ezért azokat szögletes zárójelek közé tett emlékeztető kódokkal helyettesítjük.

A listázó program működése

A program C64-en működik, és mivel BASIC-ben van írva, ez lehetőséget ad a program könnyű megértésére és az igények szerinti módosítására más típusú gépekhez is.

A program működési elve az, hogy a listázó program a tárban helyezkedik el, és egy mágneslemezen tárolt programról készít programlistát. Az is meghatározható, hogy a teljes programról vagy annak csak egy részéről szeretnénk listát készíteni. A program a mágneslemezről beolvas egy utasítássort. Ha a beolvasott utasítássor sorszáma a kiírandó sorszámtartományba esik, akkor azt az utasítássort feldolgozza, és a kívánt formában kinyomtatja. Ezután a következő utasítássort olvassa be, és ez a folyamat folytatódik egészen a program végéig.

A program négy fő részből áll. Az első rész (a 10—310 utasítások) a használt változók kezdőértékeit állítja be, és a listához szükséges paraméterek megadását teszi lehetővé.

A második rész (1000—1280) maga a listázás.

A harmadik rész (2000—3320) tartalmazza a szükséges szubrutinokat.

ASCII Kód	Emlékeztető Kód	Leütendő Billentyűk	Jelentés	
5	[WHT]	CTRL 2	White	Fehér
17	[DOWN]	CRSR DOWN	Cr.down	Cursor le
18	[RVSON]	CTRL 9	RVS on	Invers be
19	[HOME]	HOME	CRS home	CRS alaphelyzet
20	[DEL]	DEL	Delete	Törlés
28	[RED]	CTRL 3	Red	Vörös
29	[RIGHT]	CRS RIGHT	29	CRS jobbra
30	[GRN]	CTRL 6	Grean	Zöld
31	[BLU]	CTRL 7	Blue	Kék
32,160	[SPC]	SPACE	Space	Szóköz
129	[ORAN]	Commodore1	Orange	Narancs
133	[F1]	F1		f1
134	[F3]	F3		f3
135	[F5]	F5		f5
136	[F7]	F7		f7
137	[F2]	F2		f2
138	[F4]	F4		f4
139	[F6]	F6		f6
140	[F8]	F8		f8
144	[BLK]	CTRL 1	Black	fekete
145	[UP]	CRSR UP		CRS fel
146	[RVSOFF]	CTRL 0	RVS off	Invers Ki
147	[CLR]	CLR	Clear	törlés
148	[INST]	INST	Insert	beszúrás
149	[BROWN]	Commodore2	Brown	barna
150	[LT.RED]	Commodore3	Light red	Halvány Piros
151	[GREY1]	Commodore4	Grey 1	Szürke 1
152	[GREY2]	Commodore5	Grey 2	Szürke 2
153	[LT.GRN]	Commodore7	L.grean	Világoszöld
154	[LT.BLUE]	Commodore6	L.blue	Világoskék
155	[GRAY3]	Commodore8	Grey 3	Szürke 3
156	[PUR]	CTRL 5	Purple	Lila
157	[LEFT]	CRSR LEFT	Crsr left	Cursor balra
158	[YEL]	CTRL 8	Yellow	Sárga
159	[CYN]	CTRL 4	Cyan	Cián kék

```
10  REM-PROGRAM-LISTAZAS
20  REM-(PARAMETEREZES.)
30  REM-----------------
40
50  PRINT"{CLR}{DOWN}{DOWN}"; SPC(9);"
    {RVSON}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{RVSOFF}"
60  PRINT SPC(9);"{RVSON}{SPC}{SPC}{SPC}
    PROGRAM{SPC}LISTAZAS{SPC}{SPC}
    {RVSOFF}"
70  PRINT SPC(9);"{RVSON}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{RVSOFF}"
80  PRINT
90  PRINT: IFOK =0 THEN PRINT SPC(8);"EG
    YCSPC}KISCSPC}TURELMETCSPC}KEREK!":
    PRINT: GOSUB3000:OK =1
100
110 OL =1:UP$ ="{UP}":SP$ ="{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
    {SPC}{SPC}{SPC}{SPC}": PRINTUP$;SP$;UP$
120 OPEN15,8,15: GOSUB2310: OPEN4,4
130 INPUT"{SPC}LISTAZANDOCSPC}PRG.{SPC}
    NEVE:";NV$
140 OPEN3,8,3,NV$: GOSUB2310
150 PRINT"{DOWN}": SPC(10);"LISTAZASI
    {SPC}FORMATUM:": PRINT
160 INPUT"KIIRHATOCSPC}SOROK{SPC}SZAM
    {SPC}{SPC}{SPC}{SPC}:{SPC}{SPC}{SPC}72
    {LEFT}{LEFT}{LEFT}{LEFT}";NL: IFNL <
    10 ORNL >100 THEN PRINTUP$: GOT
170 INPUT"SOROKKENTI{SPC}JELEK{SPC}SZAM
    A{SPC}:{SPC}{SPC}{SPC}{SPC}70{LEFT}{LEFT}
    {LEFT}{LEFT}";FN: IFFN <10 ORFN >80
    THEN PRINTUP$: GOTO170
180 INPUT"ELSOCSPC}LISTAZOTTCSPC}SORSZA
    MCSPC}:{SPC}{SPC}{SPC}{SPC}{SPC}0{LEFT}
    {LEFT}{LEFT}{LEFT}";ES: IFES >65535
    THEN PRINTUP$: GOTO180
190 INPUT"UTSOCSPC}LISTAZOTTCSPC}SORSZA
    MCSPC}:{SPC}{SPC}{SPC}{SPC}-1{LEFT}{LEFT}
    {LEFT}{LEFT}";US: IFUS >65535 THEN
    PRINTUP$: GOTO190
200 IFUS <0 THENUS =65535
210
220 INPUT"{DOWN}{SPC}LAPVALTASHAL{SPC}L
    EALL{SPC}{SPC}{SPC}?{SPC}{LEFT}{LEFT}
    {LEFT}";A$
230 IFA$ ="I" THEN< =0: GOTO270
240 IFA$ ="N" THEN< =1: GOTO270
250 PRINT"{UP}{UP}{UP}": GOTO220
260
270 PRINT: PRINT: PRINT: GOSUB1000
280 CLOSE3: CLOSE4: CLOSE15: INPUT"
    {DOWN}{SPC}UJABB{SPC}LISTA{SPC}{SPC}
    {SPC}{SPC}{SPC}?{SPC}{LEFT}
    {LEFT}{LEFT}";A$
290 IFA$ ="I" THEN50
300 IFA$ ="N" THEN END
310
1000 REM---LISTAZAS-----
1010 REM-----------------
1020
1030 SR =0:OL =1
1040 GET#3,A$: GET#3,A$: REMBETOLTESICI
     M
1050 GET#3,A$: GET#3,A$: IFA$ ="" THENZ
     80
1060 GET#3,A$:LN = ASC(A$ + CHR$(0))
1070 GET#3,A$:LN = ASC(A$ + CHR$(0)) *2
     56 +LN: GOSUB2160
1080
1090 IFLN >US THEN280
1100 IFLN <ES THEN2050
1110 GOSUB4030
1120 IFSR >NL THEN GOSUB2100
1130 IFSR =0 THEN GOSUB2220
1140
1150 US# ="": ID =0
1160 GET#3,A$: IFA$ ="" THEN1210
1170 A = ASC(A$): IFA =34 THENID =1 -ID
1180 IFA =32 THEN IFID =0 THEN1160
1190 US# =US# +A$: GOSUB4090: GOTO1160
1200
1210 IF LEFT$(US#,1) = CHR$(143) THEN
     IF MID$(US#,2,1) ="#" THEN GOSUB21
     00: GOSUB2220
1220 SR =SR +1:P1 =0: GOSUB4160
1230 OUT$ = MID$( STR$(LN),2) +"{SPC}":
     GOSUB2400:P2 =P1
1240
1250 ID =0: FORPOZ =1 TO LEN(US#)
1260 A = ASC( MID$(US#,POZ,1)): IFA =34
     THENID =1 -ID
1270 OUT# =KD$(A,ID): GOSUB2400: NEXTPOZ
     : PRINT#4: GOTO1050
1280
2000 REM#---SUBRUTINOK----
2010 REM-----------------
2020
2030 REM-1-SOR-ATLEPESE-
2040
2050 GET#3,A#: IFA# THEN2050
2060 GOTO1050
2070
2080 REM-UJ-OLDALT-VEZD-
2090
2100 IFSR <NL +3 THENSR =SR +1: PRINT#4
     : GOSUB2160: GOTO2100
2110 SR =0:OL =OL +1: IFX THEN RETURN
2120 PRINT"{UP}{SPC}PAPIRCSERE!{SPC}&
     {SPC}RETURN"
2130 GET#3: IFX# ="" THEN2130
2140 PRINT"{UP}{SPC}{SPC}{SPC}{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}"
     RETURN
2150
2160 REM---KIJELZES-----
2170
2180 PRINT"{UP}{SPC}UTASITAS:";LN;"
     {SPC}P,SOR=";SR;"{SPC}{SPC}{SPC}"
     RETURN
2190
2200 REM-A-LISTA-CIME---
2210
2220 PRINT#4, SPC(PN -4);"-";OL;"-"
     REM-OLDAL
2230 PRINT#4, CHR$(16);"10"; CHR$(14);N
     V$; CHR$(15) REM-NEV
2240 PRINT#4; CHR$(13);SR =5: RETURN
2250
2260 REM-UJ-PRINTER-SOR-
2270
2280 PRINT#4,SR =SR +1
2290 P1 =P2: PRINT#4, LEFT$(SP$,P1);:
     RETURN
2300
2310 REM---DISK-HIBA----
2320
2330 INPUT#15,H1,H#
2340 IFH1 <20 THEN RETURN
2350 PRINT"{SPC}{RVSON}{SPC}{SPC}DISK
     {SPC}HIBA ";H#;"{SPC}{SPC}{SPC}"
2360 PRINT#4: GOTO280
2370
2380 REM---PRINT-OUT#---
2390
2400 IFP1 + LEN(OUT#) >PM THEN GOSUB226
     0
2410 P1 =P1 + LEN(OUT#): PRINT#4,OUT#;
     RETURN
2420
3000 REM#-LISTA-KODOK-TABLAZATA
3010 REM-----------------------
3020
3030 KOD# ="ABCDEFGHIJKMFORSX": DIMKD$(25
     5,1): PRINT
3040 FORI =0 TO255:KD$(I,0) = CHR$(I):K
     D$(I,1) = CHR$(I): PRINT"{UP}";
     SPC(16);I;"{SPC}{SPC}{SPC}": NEXT
3050 AD =41110:T1 =128
3060 T# ="{SPC}"
3070 T = PEEK(AD):AD =AD +1:T# =T# +
     CHR$(T AND127): IFT <128 THEN3070
3080 PRINT"{UP}"; SPC(16);T#;"{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}"
3090
3110 KD$(T1,0) =T#:T1 =T1 +1: IF PEEK(AD
     ) THEN3060
3120
3130 RESTORE
3140 READT: READT#
3150 PRINT"{UP}"; SPC(16);T#;"{SPC}
     {SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}{SPC}
     {SPC}"
3160 KD$(T,1) ="{" +T# +"}"
3170 IFT# <>"CYN" THEN3140
3180 RETURN
3190
3200 REM-EMLEKEZTETO-KODOK-
3210 REM-------------------
3220
3230 DATA5,WHT,17,DOWN,18,RVSON,19,HOME
3240 DATA20,DEL,28,RED,29,RIGHT,30,GRN
3250 DATA31,BLU,32,SPC,160,SPC,129,ORAN
     GE,133,F1
3260 DATA134,F3,135,F5,136,F7,137,F2
3270 DATA138,F4,139,F6,140,F8,144,BLK
3280 DATA145,UP,146,RVSOFF,147,CLR,148,
     INST
3290 DATA149,BROWN,150,LT.RED,151,GREY1
     ,152,GREY2
3300 DATA153,LT.GRN,154,LT.BLUE,155,GRA
     Y3,156,PUR
3310 DATA157,LEFT,158,YEL,159,CYN
3320
4000 REM---BOVITES---
4010 REM-A-KOVETKEZO-
4020 REM---SZAMBAN---
4030 RETURN
4090 RETURN
4160 RETURN
```

A negyedik rész (4000—) a már említett kontrollösszeg előállítását és kiírását fogja elvégezni.

A program megértéséhez az a legfontosabb, hogy ismerjük a BASIC utasítássorok tárolásának elveit. Ha egy lemezen tárolt programot bájtonként olvasunk, akkor az első két bájt mindig azt a címet határozza meg, amelytől kezdve az adott program a tárban, a lemezre írást megelőzően elhelyezkedett. Erre a két bájtra most nincs szükségünk, ezért egyszerűen átlépjük az 1040-es utasításban.

A következő bájtok már egy utasítássor tényleges részei, ezért azokat minden utasítássorban azonos elvek szerint vizsgáljuk.

Egy utasítássor felépítése az *ábrán* látható.

Az 1050-es utasítássor olvassa be a mutatót adó két bájtot. Ha ez a két bájt 0, akkor nincs több BASIC utasítás, tehát itt a program vége. Mivel a második bájt adja a cím magasabb helyértékét, de BASIC program nem lehet a 256-os címnél alacsonyabb címeken, így a mutatónak ez a bájtja csak a program végén lehet 0, ezért itt csak ezt az egy bájtot vizsgáljuk.

Az 1060—1070-es sorok az utasítás sorszámát olvassák be és jelzik ki a képernyőn.

Az 1090—1100 sorok megvizsgálják a sorszámot. Ha ez nagyobb, mint az utolsó listázandó sor, akkor a program befejeződik. Ha kisebb, akkor az utasítássor további részét egészen a sort lezáró 0 bájtig a 2050-es szubrutin fogja átlépni.

Az 1150—1270-es sorok dolgozzák fel az utasítássort, ha az kiírásra kerül.

Az utasítássor feldolgozása

Az utasítássor feldolgozása az 1150-es sortól kezdődik. Az US$ nevű változóba olvassuk be az utasítássort bájtonként. A 1170-es sorban a 34 ASCII kódú idézőjelet figyeljük, és jelezzük az ID nevű változóban. Ha a program soronkövetkező bájtja idézőjelek között van, akkor azt az ID=1 jelzi. Ha 0 bájtot olvasunk, ez azt jelzi, hogy vége az utasítássornak, tehát a teljes utasítássort beolvastuk az US$ változóba. Az US$-ban és a lemezen tárolt utasítássor nem mindig teljesen azonos. Ennek oka, hogy mivel az utasításrészben szereplő szóközöknek (SPC) a program végrehajtása szempontjából nincs jelentőségük, a szóközöket a program begépelésekor elhagyhatjuk. Ezáltal rövidebb és gyorsabb lesz a programunk. A listakészítő program viszont a szóközök nélkül írt program listáját

ban is automatikusan BASIC utasításonként egy szóközt ír a jobb olvashatóság kedvéért.

Egészen más a helyzet, ha a szóközök egy idézőjelek között álló kifejezésben vannak. Ilyenkor nem szabad elhagyni ezeket a szóközöket. Az 1180-as sor a 32 ASCII kódú szóközöket figyeli. Ha azok idézőjelek között vannak, akkor elhelyezi az US$-ban, egyébként elhagyja.

Az 1210-es sor az US$ első két bájtját figyeli. Ez a ténylegesen kódolt utasítássor első két bájtja. Tudnunk kell, hogy ez a kódolás az jelenti, hogy minden BASIC utasításszó nem betűnként, hanem kifejezésenként egy egybájtos számmá van átalakítva. Ezt nevezik tokennek. A 143-as szám a REM utasítás tokenje. Ha egy BASIC utasítássor REM-mel kezdődik, akkor a program figyeli a REM utáni bájtot is. Ha itt # van, akkor ezt egy jelzésnek tekintjük, amely azt mutatja, hogy a következő programrész önálló egység, amit nyomtatásban is önállóan, külön oldalon kezdve szeretnénk megjeleníteni. Természetesen más jelölést is lehetne figyelni, de ez ugyanaz a jelölés, amit sorszámozó programban is használtunk a sorszámozás vezérlésére.

Az 1220-as sor végre elkezdi kinyomtatni az aktuális utasítássort. Az SR a nyomtatott sorokat, a P1 a nyomtatás oszloppozícióját számolja. Az OUT$-ba helyezzük el a kinyomtatandó jeleket. Először az utasítássor sorszámát írjuk ki. A sorszám az LN változóban van, ezt átalakítjuk karakterlánccá, levágjuk az értékes számjegyek előtt álló szóközt, és egy szóközt teszünk a számjegyek után, hogy elválasszuk a sorszámot és a BASIC utasítássort egymástól. A P2-ben megőrizzük a sorszám kiírása utáni nyomtatási pozíciót, hogy az utasítássor folytatása többsoros utasítássor esetén ne a sor elején, hanem a sorszámoktól külön álló oszlopban kezdődjön.

Az 1250-es sortól már ténylegesen az utasítások kiírása kezdődik. Az US$-ban tárolt utasítássort bájtonként vizsgáljuk végig. A vizsgált bájt helyét a POZ változó jelzi, a bájt tartalma pedig az A változóba kerül. Az A értéke 1—255-ig változhat. Ha a vizsgált bájt idézőjelek között van (ID=1), akkor az A-ban valamelyik billentyű ASCII kódja van. Igen ám, de éppen ezek között az ASCII kódok között vannak olyanok, amelyek nyomtatáskor grafikus jelként jelennének meg. Ezért ezt a kódot közvetlenül nem küldhetjük ki a nyomtatóra. Ha viszont a vizsgált jel nem idézőjelek között van, akkor lehet betű, szám, jel vagy akár egy BASIC utasítás is. A BASIC utasításokat pedig szeretnénk utasításszavak formájában kiíratni. Ezt a feladatot egy segédtáblázat használatával oldhatjuk meg, amely megmutatja, hogy az A különböző értékeinél mit nyomtassunk ki.

A táblázat

Egy két oszlopból álló táblázatot fogunk összeállítani. A táblázat minden ASCII kódhoz (1—255-ig) két kifejezést rendel hozzá.

A 0-ás oszlopban azok a kifejezések állnak, amelyeket akkor írunk ki, ha nem idézőjelek közötti ASCII kódot vizsgálunk, az 1-es oszlopban pedig az idézőjelek között álló kódoknak megfelelő kifejezéseket helyezzük el. A táblázatból ezután könnyen elővehetjük az éppen kiírandó kifejezést, ha a táblázat sorát az A, oszlopát pedig az ID változókkal határozzuk meg. Ezt végzi el az 1270-es sor.

Most nézzük a táblázat elkészítését. A 3030-as sorban deklaráljuk a KD$(255,1) kétdimenziós mátrixot. A 3040-es sorban ezt a mátrixot feltöltjük a 0—255 közötti számokkal. A számok nagy részének van értelmezhető ASCII karakter megfelelője, amelyet ki is lehet nyomtatni: például a 43-as számnak a + jel felel meg, és ez akár idézőjelek között fordul elő, akár nem, ha az utasításszorban valamelyik bájt tartalma 43, akkor a listázó program egy + jelet fog kiírni.

Más a helyzet, ha például a 147-es kódot nézzük. Ezt ASCII kódnak értelmezve a CLR (képernyőtörlés) gomb kódja, BASIC tokennek pedig a LOAD utasítás. A táblázat 147. sorának 0-ás oszlopába tehát be kell írnunk a LOAD kifejezést, az 1-es oszlopba pedig ha azt akarjuk, hogy a programlistánkon a CLR kifejezés jelenjen meg (az eddig jól ismert grafikus karakter az inverz szivecske helyett), akkor a táblázatba ide a CLR kifejezést kell beírnunk.

A gépünk a BASIC editor és a listázó program számára tartalmazza a BASIC utasításszavakat. Először tehát az utasításszavakat másoljuk át a KD$(mátrix 0-ás oszlopába. A BASIC utasításszavak a 128-as kódtól kezdődnek, ezért a másolást csak a 128. sortól indítjuk.

A gépben tárolt utasításszavak a C64 gépen a 41118-as címtől, a C16 és Plus/4 gépeken a 33166-os címtől kezdve vannak felsorolva. Az utasításszavak minden betűje egy 65 és 90 közé eső ASCII kódnak felel meg. Az utasításszavak különböző számú betűből tevődnek össze, és az egyes utasításszavak között nincs semmilyen elválasztójel. Az egyes utasításszavak szétválasztását úgy oldották meg, hogy a szó utolsó betűjének kódjához 128-at adtak hozzá. Amikor az utasításszavakat átmásoljuk a KD$(mátrixba, mi is ezt a módszert használjuk a szétválasztásra (3050—3110-es sorok).

Különböző típusú gépek különböző számú BASIC utasítást ismernek, ezért nem az átmásolandó utasítások számát határozzuk meg, hanem annyi utasítást másolunk át, ahányat az adott gép ismer. Az utasításkészlet végét egy 0 bájt jelzi.

Ha az utasításszavakat is átmásoltuk, akkor már csak a táblázat azon pozícióit kell átírnunk, amelyeket az eredeti listaképtől eltérően akarunk kinyomtatni. Ezeket a 3230-as sortól kezdődő DATA-kban soroljuk fel, és a 3140—3170-es sorokban írjuk be a táblázatba. Először a táblázat sorát, utána az adott sorba írandó szöveget adjuk meg. Például a data17,DOWN azt jelenti, hogy a 17. sorba a DOWN kerüljön. Így majd az idézőjelek között talált ASCII 17 kód (CURSOR—LE) grafikus jele helyett a program a DOWN szöveget fogja kiírni. A módszerrel tehát bármelyik ASCII kódhoz bármilyen információt rendelhetünk.

ZSOM BÉLA

Vigyázat

Az eddigiekben többféle hibagenerálással és lemezlevédési technikával ismerkedtünk meg. Ezeknek általános jellemzője volt, hogy mindegyik a DOS által használt sávokat írta felül. Ennek hátránya, hogy a lemezkapacitás csökken, és a legtöbb másolóprogram képes már az 1. és a 35. sáv között elhelyezett védelem másolására.

Ezért most megnézzük, hogyan lehet a lemez DOS által nem használt területére — a 36—40. sávra — írni. Ezekre a sávokra alkalmazhatók az eddig már megismert védelmek. Mivel a DOS nem tud átalakítás nélkül írni és olvasni erről a területről, ezért a levédés hatékonysága jelentősen javul.

Régebben a nyugatnémet Data Becker kiadó úgy védte le programjait, hogy ezt a külső lemezterületet is formattálta, és itt helyezte el a program egy részét. Ez a módszer hosszú időn keresztül jónak bizonyult, mert nemcsak egy-két jelet ellenőrzött a program, amit gyakorlott programtörők gyorsan megfejthettek volna, és hatástalaníthatták volna a védelmet. Az itt tárolt programrészlet az algoritmus visszafejtését jelentékenyen megnehezítette. Ámbár egymásfél év elteltével a kiadó formattáló programja kiszivárgott — Magyarországon KINT néven terjedt el —, így lehetővé vált ezeknek a védett programoknak a másolása is. Ezek után a kiadó és más szoftverházak a klasszikus megoldással próbálkoztak, vagyis csak egy-két bájtnyi információt helyeztek el a külső sávokon, és a program ezeket ellenőrizte, illetve ezekkel korrigálta saját magát — például az ezekkel képzett kizáró VAGY művelettel állított vissza egy bizonyos titkosított programrészletet.

Ahhoz, hogy ezeken a külső sávokon is ki tudjuk próbálni a védelmünket, meg kell ismerkednünk néhány, a fej pozicionálását végző programrészlettel és eddig még nem használt memóriacímmel.

A fej pozicionálásához a legfontosabb, hogy vezérelni tudjuk a léptetőmotort és érzékeljük a szinkronjel megérkezését. Ezeket a 6522-es lemezegység vezérlő integrált áramköreinek segítségével irányíthatjuk, illetve ellenőrizhetjük. A lemezegység munkaterülete a meghajtó memóriájában az 1C00 címtől az 1C0F-ig terjed.

Az 1C00 a chip B kapuja, és a hozzá tartozó adatirány-regiszter címe 1C02.

Az 1C00 cím bitjeinek jelentése:

PB0 STP1 az író/olvasó fej pozicionálása

PB1 STP0 az író/olvasó fej pozicionálása

PB2 MTR a motor be- és kikapcsolása

PB3 ACT a piros LED be- és kikapcsolása

PB4 WPS az írásvédelem ellenőrzése

PB5 DS0 írássűrűség-választás

PB6 DS1 írássűrűség-választás

PB7 SYNC szinkronjel detektálása

Az 1C01-es tárcím a chip A kapuja, amin keresztül felírja a rendszer az adatokat a lemezre, illetve elolvassa azokat. A hozzá tartozó adatirány-regiszter az 1C03 címen található.

Az *1. listán* látható programrészlet egy sávval előre, a *2. lista* programrészlete pedig egy sávval visszalépteti az író/olvasó fejet.

Tolvaj!

```
        LDX $1C00               LDX $1C00
        DEX                     DEX
        TAX                     TAX
        AND #$00                AND #$03
        STA TAROL               STA TAROL
        LDA $1C00               LDA $1C00
        AND #$FC                AND #$FC
        ORA TAROL               ORA TAROL
        STA $1C00               STA $1C00
        RTS                     RTS
.BYTE TAROL             .BYTE TAROL
```

1. lista *2. lista*

3. lista

```
0600    JMP $0620
0603    LDA TAROL1
0606    CMP #$01
0608    BEQ $060B
060A    RTS
;
; A FEJ POZICIONALASA ELORE
;
060B    JSR $0670
;
; ATKAPCSOLAS OLVASASRA
;
060E    JSR $FE00
;
; ADATELLENORZES
;
0611    JSR $06BB
0614    BNE $0619
0616    INC TAROL1
0619    INC TAROL2
;
; A FEJ VISSZAALLITASA
;
061C    JSR $064B
;
061F    RTS
;
0620    CLD
;
; A 0600-AS CIMEN LEVO JMP$0620
; UTASITAST ATIRJA JMP$0603-RA
;
0621    LDA #$4C
0623    STA $0600
0626    LDA #$03
0628    STA $0601
062B    LDA #$06
062D    STA $0602
;
; AZ AKTUALIS SAVSZAM A 35
; (IDE POZICIONALJA A FEJET A
; DISK-KONTROLLER)
;
0630    LDA #$23
0632    STA $08
;
; A 3. PUFFER ($600-TOL) KEZDOCI-
; MERE ADJA AT A VEZERLEST. INNEN
; FOLYTATJA AZ INTERRUPT RUTINT.
;
0639    LDA #$0E
063B    STA $03
;
; CIKLUS, AMELY ADDIG VAR, AMEDDIG
; AZ INTERRUPT RUTINHOZ HOZZAFUZOTT
; SZUBRUTIN LEELLENORZI, HOGY A MEG-
; HAJTOBAN AZ EREDETI LEMEZ VAN-E.
;
063D    INC TAROL1
0640    LDA TAROL1
0643    CMP #$02
0645    BNE $0640
;
; TERJEN VISSZA INTERRUPT-BOL IS!
;
0647    INC TAROL1
;
; VISSZATER
;
064A    RTS
;
;*******************************
;
; 5 SAVVAL VISSZALEPTETI A FEJET
;
064B    JSR $065E
064E    JSR $065E
0651    JSR $065E
0654    JSR $065E
0658    JSR $065E
;
; URES CIKLUS, AMELY 22*256-IG
; SZAMOL.
;
065A    JSR $0665
065D    RTS
;
; URES CIKLUS
;
065E    JSR $0665
;
; FEJLEPTETES 1 SAVVAL VISSZA
;
0661    JSR $06A0
0664    RTS
;
; URES CIKLUS 22*256-IG SZAMOL
;
0665    LDX #$00
0667    LDY #$E8
0669    INX
066A    BNE $0669
066C    INY
066D    BNE $0669
066F    RTS
;
; FEJLEPTETES 5-TEL ELORE
;
0670    JSR $0683
0673    JSR $0683
0676    JSR $0683
0679    JSR $0683
067C    JSR $0683
;
; URES CIKLUS
;
067F    JSR $0665
0682    RTS
;
; URES CIKLUS
;
0683    JSR $0665
;
; FEJLEPTETES 1-GYEL ELORE
;
0686    JSR $068A
0689    RTS
;
; A FEJ POZICIONALASA 1-GYEL
; ELORE, A B-KAPU BITJEIVEL
;
068A    LDX #1C00
068D    INX
068E    TAX
068F    AND #$03
0691    STA $06F2
0694    LDA #1C00
0697    AND #$FC
0699    ORA $06F2
069C    STA #1C00
069F    RTS
;
; A FEJ POZICIONALASA 1-GYEL
; HATRA, A B-KAPU BITJEIVEL
;
06A0    LDX #1C00
06A3    DEX
06A4    JMP $068E
;
; HAROM ADAT VETELE A FEJROL,
; HA VALAMELYIK NEM EGYEZIK,
; AKKOR VISSZATER.
; AZ ADATOK: #47, #6E, #FF
;
06A7    JSR $06CF ; 1. ADAT A FEJTOL
06AA    CMP #$47
06AC    BNE $06BA
06AE    JSR $06CF ; 2. ADAT A FEJTOL
06B1    CMP #$6E
06B3    BNE $06BA
06B5    JSR $06CF ; 3. ADAT A FEJTOL
06B8    CMP #$FF
06BA    RTS
;
; A HAROM ELLENORZO ADAT OLVASASA 16-SZOR.
; HA BARMELYIK ROSSZ, AKKOR OSSZESEN
; 16-SZOR PROBALKOZIK UJRA ELOLVASNI.
;
06BB    LDY #$F0
06BD    LDX #$F0
06BF    JSR $06A7
06C2    BNE $06C8
06C4    INY
06C5    BNE $06BF
06C7    RTS
06C8    LDY #$F0
06CA    INX
06CB    BNE $06BF
06CD    INX
06CE    RTS
;
; EGY ADAT A FEJTOL
;
06CF    BVC $06CF
06D1    CLV
06D2    LDA $1C01
06D5    RTS
;
.BYTE TAROL1, TAROL2
;
.END
```

Ahhoz viszont, hogy a fejet pozicionálni tudjuk, ismernünk kell a jelenlegi helyzetet, ami nem is olyan könnyű feladat. Láttuk, hogy az író/olvasó fej mozgatását összesen két biten, a PB0-ás és a PB1-esen keresztül lehet vezérelni. Ez nem teszi lehetővé, hogy a fej aktuális állapotát közvetlenül megtudjuk. A rendszer mindig „beleolvas" a lemezbe, és a blokkok elején levő azonosítóból tudja, hogy hol van. Ezért fordul elő elég gyakran a fej kivezérlése után — ha azt a program nem állítja vissza eredeti helyére, az 1—35. sáv közé —, hogy a DOS a jó lemezt sem tudja elolvasni. Ezért feltétlenül vissza kell pozicionálni a fejet a DOS által használt lemezterületre. Így elkerülhetjük a kellemetlenségeket.

Ezt a jelenséget szokák kihasználni bizonyos programok, amelyek azzal fenyegetőznek, hogy másolásuk esetén hardverhibát okoznak, és így tönkreteszik a meghajtót. Ilyenkor a lemez többszöri inicializálása után e fej általában visszatalál a helyére. Ha mégsem, akkor a készülék fedőlapjának levétele után csak kézi segítséggel javíthatjuk meg.

A meghajtóban azonban valóban szoftver úton is okozhatunk hardverhibát. Ugyanis minden egyes léptetés után „pihentetni" kell a motort. Erre a rendszer egy üres ciklust használ, amely 22-szer elszámol 0-tól 255-ig. Ezt programunkba feltétlenül be kell építeni!

Az író/olvasó fejnek a 36—40. sáv közé pozicionálását egyszerűen megoldhatjuk. Először a lemezvezérlő segítségével a 35. sávra pozicionálunk, azután saját programunkkal sávonként léptetjük a fejet. Az aktuális sávszámot (jelen esetben a 35.) a $80 memóriacímre kell betölteni, majd az akkumulátorba töltjük az aktuális pufferszámot, ahol a programunk folytatódni fog, és meghívjuk a $D6D3 szubrutint, amely átadja a paramétereket a vezérlőnek. Ennek hatására a rendszer a fejet a 35. sávra pozicionálja, és átadja a vezérlést a programunknak. Ennek gyakorlati megvalósítását látjuk a *3. listán*, amellyel egy programot védtek le, és én a védelem hatástalanításakor fejtettem vissza. A program a 40. sávra felírt $47, $6E, $FF azonosító kódokat ellenőrzi.

* * *

Ezzel lezárult adatvédelmi sorozatunk. Reméljük, olvasóink hasznos segítséget kaptak e témakörökkel kapcsolatban.

KOVÁCS P. ATTILA

Mesterséges értelem V.

Gépi érzékelés

A biológiai definícióban, hogy mit értünk az élőlény fogalmán, szerepel az a kifejezés, hogy **képes reagálni** a környezeti ingerekre *(1. kép)*. Ahhoz, hogy bármi vagy

1. kép....képes reagálni....

bárki valamire reagáljon, természetesen először érzékelnie is kell, majd az érzéklet valamilyen módon feldolgoznia, értelmeznie. Ha egy intelligens programot szeretnénk készíteni, de a környezettel való kapcsolatát a konzolra korlátoznánk, akkor ez az érintkezés a külvilággal nagyon szegényes lenne. Az ilyen döntés a program önálló ismeretszerzési, adatgyűjtési képességeinek, tehát általában lehetőségeinek jelentős korlátozását jelentené, vagyis meghiúsítaná a szándékunkat már csak ennélfogva is: a program enyhén szólva „korlátos" lenne.

Az ember számára messze a legtöbb bemenő információt látása és — jóval kevesebbet, de még mindig elég sokat — hallása biztosít. Ezért, és mivel ezeknek az érzékelési funkcióknak a kielégítő megoldása volna a legnagyobb gyakorlati jelentőségű (erről cikkünk végén részletesebben szólunk), az érzékelés területén eddig készült számítógépprogramok is leginkább a látás, **látványértelmezés** *(2. kép)*, valamint a hallott beszéd feldolgozásával foglalkoztak.

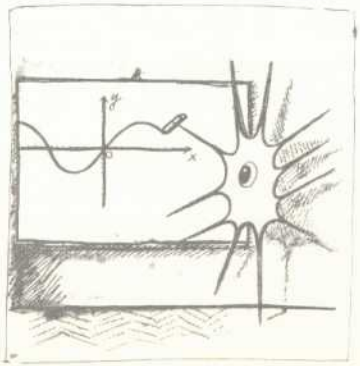
A külvilágból érkező információ feldolgozása viszonylag könnyen két fő lépésre különíthető el. Az ún. korai feldolgozás során a beérkező információban lévő zajokat matematikai módszereken alapuló eljárásokkal szűrik, és a lényeges jellemzőket (kontúrokat stb.) kiemelik és erősítik. Tehát ez a terület lényegében alkalmazott matematikai diszciplina. A sorozat témájához a második lépés — az így létrejött látvány, illetve szöveg értelmezése — inkább illeszkedik, hiszen itt hasznosulnak az eddigiekben már tárgyalt és további módszerek, ezért most a hangsúlyt is inkább ez a lépés kapja.

2. kép....látványértelmezés....

Korai feldolgozás

Amikor az első látványértelmező programok íródtak, az élőlények látásának működéséről még a mainál is kevesebb ismeretünk volt. Részben ezért, részben pedig azért, mert az idegsejtek már akkor is modellezhető működésének szimulálása nagyon számításigényes lett volna, a mesterséges látás, gépi látványérzékelés megvalósítása — akkor úgy tűnt — az élő folyamatok utánzásától független módszerekkel halad. Aztán, ahogy az élőlények érzékelési-reagálási folyamataira vonatkozó ismereteink bővültek, mégis az derült ki, hogy a látókéregben működő idegsejtek is tulajdon-

3. kép....kicsit eltérő mechanizmusokkal....



képpen a vizuális információ számítógépes feldolgozásában — szűrésében — felhasznált Gauss-függvény deriváltjaival történő konvolúciót valósítják meg, ugyan adottságaikból következően **kicsit eltérő mechanizmusokkal** *(3. kép)*.

Mint említettük, ezekkel a kérdésekkel itt részletesebben nem foglalkozunk. Azt már viszont hangsúlyoznunk kell, hogy például a látványra vonatkozó korai feldolgozás által szolgáltatott ún. szürkeségi (színsúly) mátrix, illetve a hangok digitalizálása után az érzékletbeli kisebb logikai egységek (vonalak, hangok) elkülönítésére van szükség. Ezekhez a logikai egységekhez ún. címkéket rendelünk („ez az **a** hang", „ez a vonal egy objektum külső szélét jelöli" stb.), ami nyilvánvalóan már értelmezési tevékenység, még akkor is, ha az érzéklet összességének szempontjából valóban elemi, sőt mint alább megvilágítjuk, sokszor lényegében téves megállapítás is lehet.

Értelmezés

Milyen problémákat kell megoldani a külvilágból érkező információ gépi értelmezéséhez? A folyamatosan érkező információ feldolgozásakor az információ logikai egységekre tagolására és az egységek azonosítására, kategorizálására van szükség. A két probléma szorosan összefügg. Kategorizálni csak a logikai egységeket lehet, de a logikai egységekre tagolás közben adódó bizonytalan esetekben viszont az segíthetne, ha az egységek már kategorizálva, azonosítva lennének. Egy hallott mondat feldolgozása során például a következő szinteken lenne szükséges a kategorizálás:

hangok — szavak — mondatszerkezetek.

Hogy jobban megértsük a szintek közötti kapcsolódást, vegyük a következő, kicsit „sületlen" példát. Tegyük fel, hogy a hangok kategorizálása első szinten már megtörtént (ritka szerencse), és a gép a következő karaktersorral áll szemben: **„kapusült"**. Itt a szóhatár két helyen is lehet, és csak a szövegösszefüggés elemzéséből derülhet ki, melyik a helyes elválasztás *(4. kép)*. Vagy: „A gól után csak az elkeseredett **kapus ült** a földön", vagy: „A pék kemencéjében éppen étvágygerjesztő mézeskalács **kapu sült**".

De hogyan elemezze a program a szövegösszefüggést, amikor eddig még szavakra sem bontotta a bemenő hangsort?

Ehhez hasonló nehézségekkel a kategorizálás minden szintjén találkozunk. A szintek helyes feldolgozásához a fölötte levők ismeretére lenne szükség. Például a szavak nélkül eggyel lejjebb — a hang szinten — a gyors beszédből vagy pontatlan kiejtésből

4. kép....„kapusált"....

adódóan esetleg nem egyértelmű, hogy egy hangnak melyik betű felel meg. Ekkor meg kell nézni, hogy a környező betűk milyen szót alkothatnak. De hogyan, amikor még ezek a betűk meg sincsenek határozva? Az ilyen nehézségek rendszeres felbukkanásának valószínű oka, hogy a mesterséges gondolkodás nem tud úgy haladni, ahogy feltehetőleg az egész az emberi agyban zajlik, ahol a különböző szintek feldolgozása egyidejű, és a feldolgozást végző folyamatok szoros kapcsolatban állnak egymással, közöttük nagy mennyiségű folyamatos információcsere zajlik (már feltéve, hogy a szintek egyáltalán elkülöníthetők).

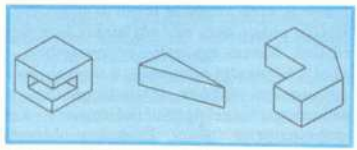
5. kép....magasabb elemzési szintre....

A fentiekhez hasonló bizonytalanságok feloldására két lehetőségünk van. Az egyik, hogy minden kérdéses elemhez hozzárendeljük az összes olyan címkét, amelyik lehetséges az adott elem számára, és amelyek között nem tudunk dönteni. Íly módon a döntést „továbbpasszoltuk" eggyel **magasabb elemzési szintre** *(5. kép)*.

A másik lehetőség, hogy az adott szinten egy kicsit tágabb környezetet vizsgálunk meg, és ezt a környezetet addig tágítjuk, amíg nem sikerül egyértelmű döntésre jutnunk. Tételezzük föl például, hogy egy hangról valóban nem tudjuk megállapítani, melyik betűt is kell hozzárendelni. Ha az elemzési tartományt kitágítjuk, elképzelhető, hogy a mellette lévő két egység (hang) amplitúdófrekvencia-mintázata egyértelműen maghatározza, hogy azok milyen betűk. Eközben kiderülhet, hogy amit eddig a vizsgált minta részének tekintettünk, az tulajdonképpen valamelyik mellette lévő hanghoz tartozik, mert a határt előzőleg rosszul húztuk meg. Így a vizsgált hangról már megszülethetett az azonosítás: így, csonkábban már viszont csak egy betűhöz tartozhat. Ha a szomszédos hangok nem segítenek a döntésben, vagy maguk sem egyértelműek, akkor tovább kell tágítani a kört a távolabbi szomszédos hangok felé, remélve, hogy így előbb-utóbb eredményes lesz a lokális vizsgálat. A „továbbpasszolós" módszer — a magasabb szinten intelligensebb, azaz — bonyolultabb algoritmusok működését igényli, a másik — a „körülnézős" — megoldás viszont számításigényesebb: valószínűleg sokszor kell „körülnéznünk" olyan esetekben, amelyek magasabb szinten, bővebb információk birtokában esetleg igen könnyen eldönthetők lennének.

Az értelmezésben a legutolsó lépés, hogy a felcímkézett objektumból egy összefüggő, **ellentmondásmentes** egységet állítsunk össze.

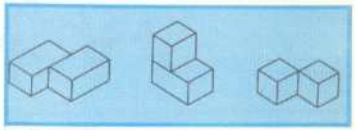
Példaként kövessük végig egy olyan algoritmus működését egy képen, amelyben csak vonalakból álló objektum értelmezése van kidolgozva. A vonalak olyan alakzatokat ábrázolhatnak, amelyeknek mindegyik pontjában három vagy annál kevesebb él találkozik. Ilyeneket mutat az *1. ábra*, illetve a nem megfelelőkre mintákat a *2. ábra*.



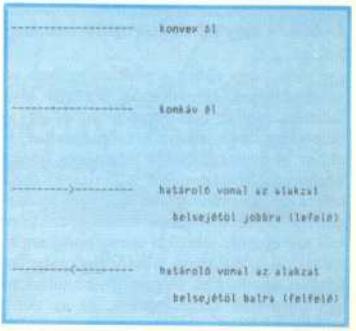
1. ábra. Néhány, a feltételeknek megfelelő alakzat

A megfelelő alakzatokon minden vonal besorolható a *3. ábrán* látható kategóriák valamelyikébe. Egy konkrét objektumot vizsgálva, annak éleit a kategóriák szerint minősíti az értelmező algoritmus (lásd. *4. ábra*).

Az algoritmust megalapozó egyik további megfigyelés, hogy a háromélű objektumok esetében az élek találkozása négyféle lehet *(5. ábra)*. Az összes éltalálkozásnál a vonalak mindegyike tartozhat bármelyik típushoz. Ha utánaszámolunk, megállapít-



2. ábra. Néhány, az algoritmus számára nem megfelelő alakzat

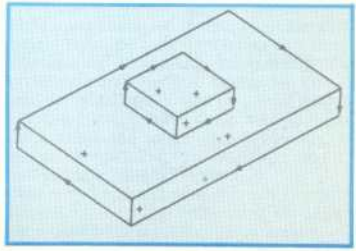


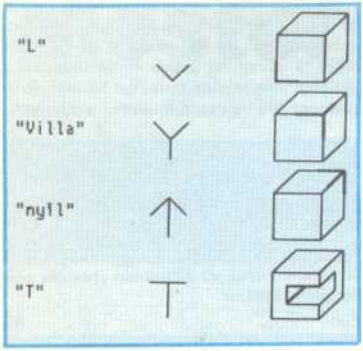
3. ábra. Az alakzatokat alkotó élek típusai

hatjuk, hogy 208 féle vonalcímkézés fordulhat elő. Az „L" alakú két él találkozása $4 \times 4 = 16$ esetet jelent. Amikor három él találkozik, az elvben lehetséges esetek száma $4 \times 4 \times 4 = 64$, így $4 \times 4 + 3 \times 4 \times 4 \times 4 = 16 + 3 \times 64 = 208$. Szerencsére a 208 elképzelhető variációnak csak egy kis része reális a valódi fizikai objektumokat ábrázoló képek tekintetében. Fizikailag csak 18 különböző éltalálkozás lehetséges. Ezt úgy láthatjuk be, hogy végiggondoljuk, hogy a *6. ábra* szerinti egy térnyolcadot kitöltő téglatest környezetéből összegyűjtjük a másik hét szegmensből látható éltalálkozásokat *(7. ábra)*.

Ezzel áttekintettük az algoritmust megalapozó korlátozásokat és észrevételeket, valamint az ezek következtében előállt egyszerűbb helyzetet. Maga a (D. L. Waltz által kidolgozott) algoritmus, amit most példán követünk, a következő. Kiválasztunk az ábrán egy találkozási pontot, és össze-

4. ábra. Egy alakzat felcímkézett körvonalakkal





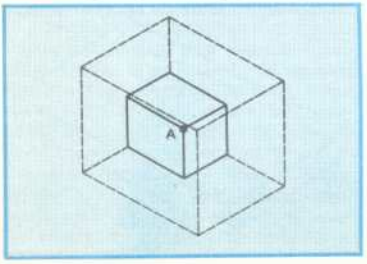
5. ábra. A négy lehetséges éltalálkozási eset

gyűjtjük, hogy ehhez a ponthoz milyen vonalcímkézések rendelhetők. Ezután kiválasztunk egy az első csomópontból egy él távolságra lévő második csomópontot, és ahhoz is összegyűjtjük a lehetséges felcímkézéseket. Ezután annak a korlátozó feltételnek az alapján, miszerint a két csomópontot összekötő élnek azonos címkével kell szerepelnie mindkét csomópontban, kizárjuk a szóba nem jöhető címkézéseket. Ezután az eddigiek szerint továbbhaladunk egy harmadik csomópontba, és az ebből adódó feltételek alapján azután visszafelé is kizárjuk az ezzel nem konzisztens címkézéseket, és így tovább. Az eljárást addig folytatjuk, amíg minden csomópontot be nem jártunk.

Kövessük végig az algoritmus működését a *8. ábrán* látható kockán. (A zárójelben lévő számok címkéket jelölnek, az anélküliek bejárási sorszámot.) Az első csomópontban csak a 13-as sorszámú címke lehetséges, a másodikban csak a 6-os. Ugyanígy az összes határoló élen a címkézés egyértelmű. Ha a 7-es csomópontot csak önmagában tekintjük, akkor a „villák" bármelyike állhatna ott, de amennyiben a környezetet is tekintjük, csak a 8-as sorszámú felel meg a feltételeknek.

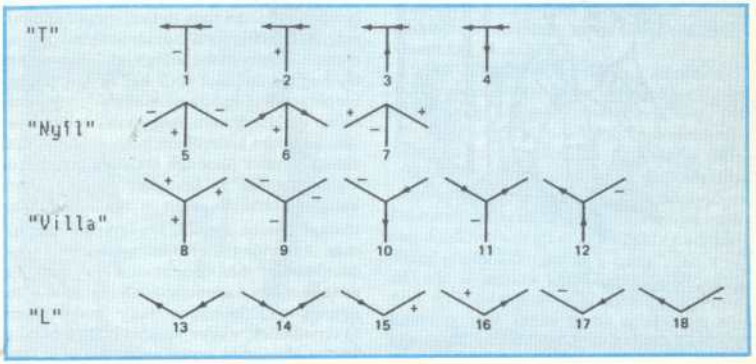
Ennek az algoritmusnak a végrehajtásával a számítógép számára (mesterséges értelemmel) eddig csak vonalhalmaznak tekinthető ábrán azonosítható egy objektum, amelynek osztályba sorolása a tulajdonságai alapján — vagyis végeredményként annak megállapítása, hogy itt most kockáról van szó — egy újabb szint döntési feladata. Ez a következő szint elvezet a szabályalapú

6. ábra. Egy térnyolcadot elfoglaló téglatest

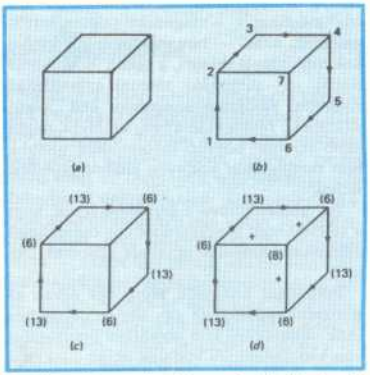


rendszerek kérdéséhez, amiről sorozatunkban már említést tettünk. (1987/6. — A szerk.)

Látható tehát, hogy egy igen korlátozott problémaosztályon belül is milyen sok körültekintést igényel az akár csak olyan egyszerű értelmezési feladat megvalósítása is, mint amelyet a tárgyalt algoritmus old meg. Talán érezhető, hogy mennyivel bonyolultabb például egy olyan program felépítése és működése, amely a robotkamera által érzékelt képekből kellene, hogy kihámozza láthatunk a tévében, főleg mechanikus érzékelőkkel oldanak meg tájékozódási feladatokat. Például egy olyan robot, amely szalagon érkező tárgyakat fog meg és tesz át egy másik helyre, mondjuk egy présgép alá, ma még nem kamerával figyeli a szalagot, hogy érkezett-e új munkadarab, hanem a tárgy mozgása közben egyszerűen csak működésbe hoz egy megfelelő helyre épített mechanikus nyomásérzékelőt, vagy elzárja valahol egy pillanatra a fotocellához irányított fénysugár útját, amire a robot



7. ábra. A fizikailag lehetséges éltalálkozások



8. ábra. Egyszerű példa a címkézési algoritmusra

ezek jelentőségét és szerepét a robotot körülvevő tárgyak világában, majd még vezérelnie is ezek között a robot cselekedeteit. Hiszen ekkor nem egy végletekig leegyszerűsített, hanem egy valós környezetben — ahol a környezet bonyolult, a tárgyak jóval összetettebb alakúak, sőt mozoghatnak is, akár még a megvilágítás is változhat — kell a feladatot elvégezni. Ezek a problémák mind az első feldolgozást, mind pedig az értelmezést rendkívül megnehezítik. Olyannyira, hogy napjainkban még nincs is akár csak éppen robotrendszer sem, illetve annak a néhány és rendkívül drága modellnek, amelyek mégis vizuális érzékeléssel dolgoznak, az ilyen irányú teljesítménye nagyon szűkös tartományban érvényesül. Azok a robotok, amelyeket lassan naponta végrehajtja a számára előírt mozdulatsort. Akkor is, ha nincs munkadarab a megfelelő helyen, vagy esetleg valamiért nem a megfelelő szögben áll, vagy ha a fénysugár bármely egyéb okból szakadt meg — mondjuk kiégett a lámpa.

* * *

A kutatók és fejlesztők világszerte nagy erőfeszítéseket tesznek egy megfelelő vizuális tájékozódó/értelmező rendszer létrehozására, hiszen ez a robotok — már így is óriási! — felhasználási lehetőségeit megsokszorozná. (Nálunk sem véletlenül alakult meg közel két éve a Magyar Robottechnikai Társaság, amely a már évtizedes hazai munkákat és sok igen jelentős eredményt mint társadalmi szervezet is hasznosítani, továbbvinni és az érdekelteket összefogni kívánja. — A szerk.)

Ezzel zárjuk a mesterségesintelligencia-kutatások alapjait tárgyaló sorozatunkat. A mélyebb tanulmányozáshoz is kedvet kapott olvasók jelentős további információkhoz juthatnak a már az előző részekben ajánlott irodalmakon túl az alább felsorolt könyvekből és cikkekből.

SOMOGYVÁRI KÁROLY

Irodalom

(1) Winston, P. (szerk.): The Psychology of Computer Vision. McGraw-Hill, 1975.
(2) Rich, E.: Artifical Intelligence. McGraw-Hill, 1983.
(3) Sántáné Tóth E.—Szeredi P.: Prolog Applications in Hungary. (Szerk. Clark, K. L.—Tärlund, S. A.: Logic Programming. London, Academic Press, 1982. kötetben)

Mit tud a SYMPHONY?

A SYMPHONY egy szép integrált információkezelő rendszer. Olyan szép, mint egy valódi szimfónia. (Sajnos, nem annyira fülbemászó...) A fő dallam megszólaltatásához az e célra szolgáló hangszeren — egy IBM PC/XT, AT számítógépen — aránylag rövid időn belül eljuthatunk ugyan, de a teljes mű eljátszásához a „partitúrát" (a nyelv leírását) még sokáig tanulmányoznunk kell.

Az öt „fő tétel"

Mint minden igazi szimfónia, a SYMPHONY is több tételből áll, és ezek együttese — bár külön-külön is gyönyörűséget okozhatnak — adja a teljességet, teremti meg a harmóniát.

Ezek a mi tárgykörünk szerint, illetve ennek megfelelő szakszerű kifejezéssel ún. *munkakörnyezetek* (work environment), mégpedig az alábbiak:

1. Táblakezelő rendszer (spreadsheet) — SHEET
2. Szövegszerkesztő (word processing) — DOC
3. Grafikoníró (graph) — GRAPH
4. Adatbázis-kezelő (database management) — FORM
5. Adatkommunikációs program (data communication) — COMM

Végleg elszakadva most már kezdő hasonlatunktól, a cikk további részeiben már csak a szakterminológiával élünk. A felsorolt öt *munkakörnyezet* a SYMPHONY-nak mint eszköznek — az angol környezetmegnevezések melletti, nagybetűkkel jelzett — öt üzemmódját képviseli. Már itt fontos megjegyezni, hogy bármely pillanatban átkapcsolhatunk az egyik üzemmódból a másikba, ha a feladat úgy kívánja. Az öt üzemmódot az köti össze, hogy mindegyiknek közös a háttere: egyetlen állomány, amelynek tipikus szerkezete van. Ezt *munkatáblának* (worksheet) nevezzük.

A munkatábla a szerkezetét tekintve egy táblázat, egy téglalaprács (lásd az *1. ábrát*), amelynek egy-egy téglalapjába 240 karakternyi információ fér bele. Egy téglalapot *cellának* nevezünk. A munkatábla 8192 sort tartalmaz, egy sora 256 cellából áll.

A cellák tartalma lehet szám, szöveg, dátum, logikai érték és képlet is. Képleten itt olyan formulát értünk, amelyben cellahivatkozások, a szokásos műveleti jelek és a SYMPHONY speciális beépített függvényei is szerepelhetnek. A memóriában minden egyes cellához a tartalmon kívül (ami lehet formula vagy érték — ebből a szempontból minden, ami nem formula, érték) a megjelenítésre vonatkozó előírásokat, a hozzáférést befolyásoló védettséget tárolja a rendszer (lásd a *2. ábrát*).

Valamennyi üzemmódban tehát ezen az egyazon munkatáblán dolgozunk, az egyes műveleteket cellákon (ezek tartalmain) vagy egy csoportján (tartományán) végezzük el. A műveletek típusa üzemmódonként más és más.

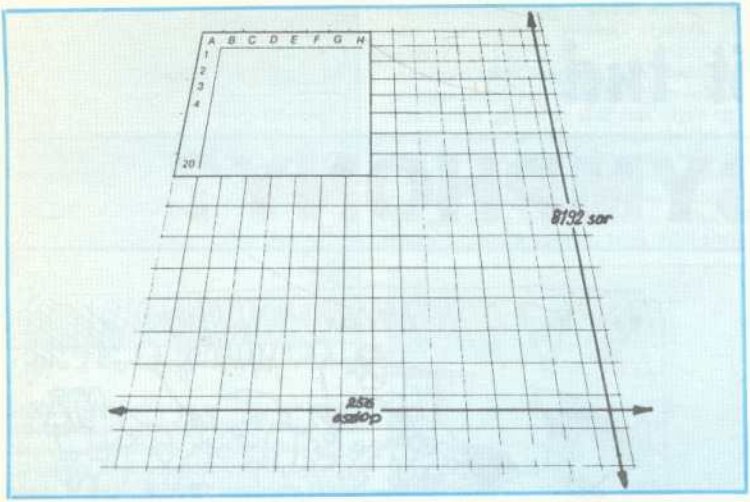
Mivel a SYMPHONY voltaképpen egy táblán dolgozik, leegyszerűsítve *táblázatkezelő programnak* tekinthető.

Az ablak

Hogy a munkatábla valamelyik részén éppen mit „látunk", attól függ, milyen „ablakon" keresztül nézünk rá. Minden üzemmódnak saját *ablaka* (window) van. Ez leegyszerűsíti a dolgunkat: azt is mondhatjuk, hogy az ablaktípus megadásával határozzuk meg, melyik üzemmódba akarunk kerülni.

A monitor képernyője túlságosan kicsi ahhoz, hogy akár a munkatábla minden celláját (minden cella tartalmát) egyszerre láthassuk. Arról, hogy egy képernyőnyi ablakrészen pontosan mit (mennyit) láthatunk, az ablaktípusok ismertetésénél beszélünk. Ami már most fontos: tulajdonképpen a munkatáblán az adatok bevitele, a formulák beírása is az ablakokon keresztül, azok felhasználásával történik.

Ha még az ablaktípusok neveit is felsoroljuk, akkor a leglényegesebb dolgokat már tudjuk az ablakokról. Az öt üzemmód-



1. ábra A munkatábla

hoz tartozó öt ablaktípus neve ugyancsak SHEET, DOC, FORM, GRAPH, COMM. A *3. ábrán* együtt láthatjuk az öt ablak szerkezetét (két SHEET típusú ablakot tettünk rá a munkamezőre).

Parancsok menüből

A SYMPHONY valójában olyan program, amelyben az algoritmusokat parancsokkal írjuk fel. A parancsok (commandok) menürendszerbe vannak összefoglalva. Egy műveletsor végrehajtására parancsokat választunk ki a menüből. A SYMPHONY tehát egyrészt a menürendszer által vezérelhető, másrészt van egy saját speciális programnyelve, amely ún. makrókból áll (erről a következő számunkban olvalhatnak majd — a szerk.).

A SYMPHONY főmenüje a SERVICES MENU (lásd *4. ábra*). Ez minden üzemmódban (tehát mindegyik ablaktípusnál) behívható, és belőle parancsok választhatók. Minden üzemmódnak van saját, többszintes menüje is. Ugyanannak a parancsnak, attól függően, hogy melyik menüből hívtuk meg, más a hatása.

A táblakezelő rendszer (a SHEET üzemmód)

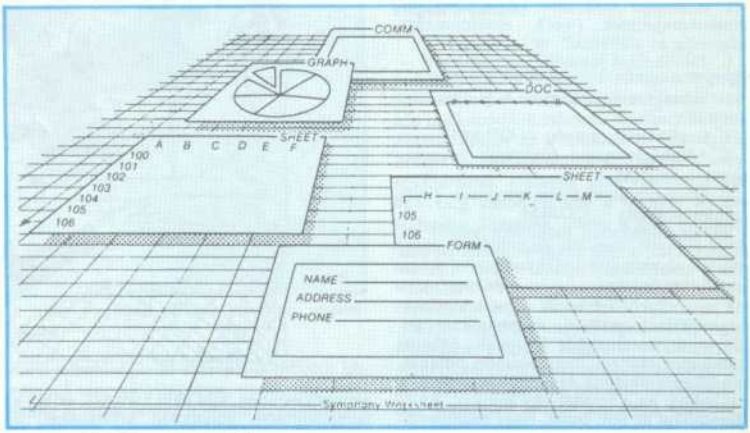
A rendszer alapüzemmódja a táblakezelő üzemmód, ami egyúttal azt is jelenti, hogy ilyenkor a SHEET típusú ablakon keresztül látjuk a munkatáblát. A SHEET üzemmód azért tekinthető alapüzemmódnak, mert erre a többi üzemmód is erősen támaszkodik.

A SHEET-ablak szerkezetét az *5. ábrán* láthatjuk. Ha tehát SHEET üzemmódban vagyunk vagy ebbe kapcsolunk át, akkor a képernyőn egy ilyen ablak jelenik meg.

A SHEET-ablak oszlopait az ábécé betűivel, illetve mivel összesen 256 oszlopot tartalmaz egy ilyen ablak, ezért betűpárokkal jelöljük az ábécé által meghatározott sorrendben (A,B, . . ., Z, AA,AB, . . ., AZ, . . ., BA, . . ., BZ, . . ., IA, . . . IV). A 8192 sor mindegyike egy sorszámot kap. Egy cella címét az oszlopnévből és a sorszámból állítjuk össze: például a felső bal sarokban levő cella címe: A1.

Mint már említettük, minden cella „hossza" 240 bájt; a képernyőn azonban alapértelmezésben minden cellából csak az első kilenc bájt látszik. Természetesen van lehetőség az alapértelmezés megváltoztatására, azaz a látható részek oszloponkénti szélesítésére (ezt például egy erre szolgáló utasítással tehetjük meg).

Igen fontos „mozgó" szerkezeti eleme a SHEET-ablaknak a *cellamutató* (cell pointer). Ezzel egy-egy cellát tudunk kijelölni, a képernyőn „kivilágosítani" a következő módon. Amikor belépünk a SHEET üzemmódba, az A1 cella, azaz az A oszlop és az első sor által kijelölt kis téglalap kivilágosodik, azt jelölve, hogy a cellamutató az A1 cellán áll. Ettől kezdve a cellamutatót a mozgatóbillentyűk (→, ←, ↑, ↓) segítségével tetszőleges cellára rávihetjük többszöri fel, le, jobbra, balra mozgatással). Kellemes tulajdonsága a cellamutatónak a hosszbeli rugalmassága, vagyis hogy pontosan olyan szélességben világítja ki az adott cellát, mint amilyen a cellahossz (tehát ha valamelyik oszlop celláit mondjuk 12 bájtra szélesítettük, akkor ha a cellamutatót egy ilyen oszlop valamelyik cellájára viszszük rá, ott pontosan 12 bájtnyi téglalapot világít meg). Az éppen mutatott cellát *kurrens* (aktív) *cellának* nevezzük.



3. ábra Ablakok a munkatáblán

2. ábra Cellatartalom

Fontos része az ablaknak az *5. ábrán* látható *vezérlő panel* (a képernyő bal felső sarkának három felső sora). Ennek első sora az *állapotsor*, itt az aktív cella címe és tartalma látható. (Ha a cellamutatót elmozgatjuk, ennek megfelelően változik meg e sor tartalma is.) A vezérlő panel második sorában, a *parancssoron* az éppen behívott menü parancsai, illetve a harmadikban bizonyos üzenetek jelennek meg.

A képernyő jobb felső sarkában most a SHEET szó látható. Ez a rész a *módjelző*. Egyfelől itt láthatjuk, hogy milyen üzem-

Window	File	Print	Configuration	Application	Settings	New	Exit
Use	Save	Go	File	Attach	Learn	No	No
Create	Retrieve	Line-Advance	Printer	Detach	Security	Yes	Yes
Delete	Combine	Page-Advance	Communications	Invoke	Global-Protection		
Layout	Xtract	Align	Document	Clear	Auto-Execute		
Hide	Erase	Settings	Window	Quit	Communications		
Isolate	Bytes	Quit	Help		Quit		
Expose	List		Auto				
Pane	Table		Other				
Settings	Import		Update				
Quit	Directory		Quit				

Ablak	Fájl	Nyomtat	Konfiguráció	Alkalmazás	Beállít	Új	Kilép
Használ	Mentés	Indít	Fájl	Hozzákapcsol	Tanulási üzemmód	Nem	Nem
Készít	Betöltés	Soremelés	Nyomtattás	Elválaszt	Biztonság	Igen	Igen
Töröl	Kombinálás	Lapemelés	Kommunikáció	Aktivizál	Általános védelem		
Elhelyez	Részmentés	Igazít	Dokumentum	Töröl	Auto-makró		
Eltüntet	Törlés	Beállít	Ablak	Kilép	Kommunikáció		
Izolál	Bájtok	Kilép	Segítség		Kilép		
Visszahoz	Listázás		Auto				
Vág	Fájl tábla		Egyéb				
Állít	Import		Aktualizál				
Kilép	Tartalomjegyzék		Kilép				

4. ábra A SERVICE menü felépítése és magyar értelmezése

módban vagyunk, de az üzemmódon belüli egyes tevékenységeket is ezen a helyen jelzi a rendszer. Például ha éppen az adott üzemmód menüjét hívtuk be, akkor a MENU szó, ha szerkesztünk, az EDIT szó jelenik meg stb.

A képernyő, illetve az ablak jobb alsó sarkában az ablak neve látszik. Ezt a nevet mi adhatjuk meg. Ha ezt nem tesszük, akkor az ablak neve: MAIN. Végül a bal alsó sarokban a dátum és az idő olvasható.

Mielőtt rátérnénk, hogyan lehet az adatokat a táblán kezelni, még egy fontos fogalommal kell megismerkednünk.

A táblakezelő utasítások nagy része cellákon végez műveleteket. Vannak olyan műveletek is, amelyek cellák téglalap alakú csoportjával dolgoznak. A SYMPHONY-ban ezt *tartománynak* (range-nek) hívjuk. A tartományt definiálni kell. Ezt kétféleképpen tehetjük meg. Az egyik lehetőség az, hogy az egyik sarkot (például a téglalapot alkotó cellák közül a bal felső sarokban levőt) a cellamutatóval kijelöljük, majd egy speciális utasítással, a TAB funkcióbillentyűvel rögzítjük, és utána a cellamutató mozgatásával határozunk meg egy téglalapot. A másik megoldás: bebillentyűzzük a kijelölendő tartomány szemközti csúcsainak cellacímeit, úgy, hogy a két cím közé legalább egy pontot írunk. Például az A5 . . F10 beírással azt a tartományt jelöljük ki, amelynek bal felső sarka az A5-ös, a jobb alsó sarka pedig az F10-es cella.

Műveletek a táblán

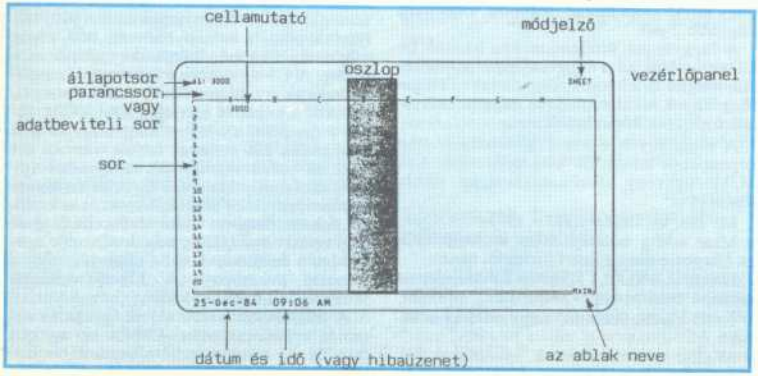
A tábla voltaképpen egy elektronikus űrlapnak tekinthető, amelyen a rubrikákat a cellák jelentik, ahova beírhatunk számokat, szövegeket; a számokat összegezhetjük és az eredményeket beírhatjuk más cellákba stb.

A táblakezelő valamely cella tartalmát értéknek (számnak) tekinti, ha a számjegygyel vagy a +, −, ., @ #, $, (,) jelek valamelyikével kezdődik. Szöveg mindaz, amelynek első karaktere nem a fentiek közül való. Szöveg beírásakor a rendszer alapértelmezésben automatikusan egy felső vesszőt (') ír az első karakter elé.

	A	B	C	D	E	F	G
1			Forgalom				
2			május 11-16				
3							
4		kenyér	tej	vaj	bevétel		
5	Bolt-1	500	300	45.5	8340.25		
6	Bolt-2	700	800	72.5	15268.75		
7	Bolt-3	600	400	36.5	9520.75		
8	összesen:	1800	1500	154.5	33129.75		
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

6. ábra Példa SHEET üzemmódban

5. ábra A SHEET ablak

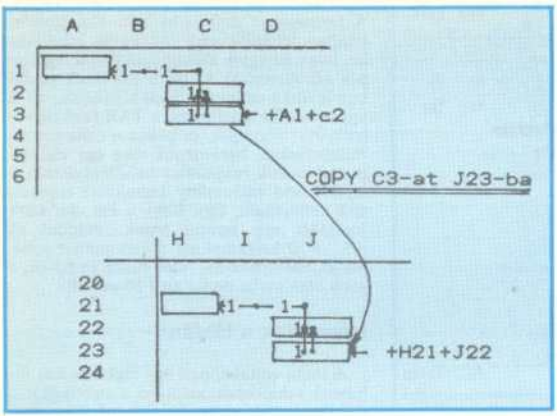


Adatok betöltése a táblába

Ez a művelet nagyon egyszerű: a cellamutatót rámozgatjuk a cellára, bebillentyűzzük a karaktersorozatot, majd megnyomjuk a Return (↵) billentyűt. Ennek hatására az állapotsoron egyidejűleg megjelenik a kurrens cella címe és egy kettőspont után a bevitt tartalom. Például: ha a cellamutató az A5 című cellán áll és a billentyűzeten a Bolt—1 szöveget ütjük le, akkor az állapotsorban az A5: 'Bolt—1 karaktersorozat jelenik meg.

Illusztrációképpen hozzuk létre a *6. ábrán* látható táblázatot. Ezt az alábbi műveletsorral tudjuk elérni (M a cellamutató helyzetét, G a gépelést, a nyilak a cellamutató mozgatását jelentik):

M: A5
G: Bolt—1
G: Bolt—2



7. ábra Példa relatív cellahivatkozásra

G: Bolt—3
G: Összesen
M: B4
G: kenyér
G: tej
G: vaj
G: Bevétel

M: B5
G: 500
G: 700
G: 600

M: C5
G: 300
G: 800
G: 400

M: D5
G: 45.5
G: 72.5
G: 36.5

M: C1
G: Forgalom
G: máj. 11—16.

Formulák a táblában

A *6. ábrán* levő táblázatban az „összesen" sort és a „bevétel" oszlopot nem töltöttük ki. Ezeket *formulákkal* képezzük.

A formulák cellákból, szám- és szövegkonstansokból, beépített függvényekből és ezek közötti műveleti jelekből (+, −, *, /, ↑ (hatványozás), # AND #, # OR #, # NOT #, =, <=, >=, & (konkatenáció)) állnak. Zárójelek itt is alkalmazhatók. A formula kezdőkaraktere a következők egyike lehet:

számjegy, ., +, −, (,), @, #, $. Például: +C4/6.8; +A5+ (B8—C4)* 3.8

A formulát abba a cellába írjuk be, ahol az értékét kapni akarjuk. Maga a formula mint karaktersorozat tárolódik a cellához tartozó memóriarészben, és az érték, ami a formula kiszámításakor létrejön, jelenik meg a képernyőn.

Töltsük ki a nyolcadik sort formulák segítségével. A három oszlopösszeget különböző módon állítjuk elő — ezáltal illusztráljuk is a formulaírás lehetőségeit. Az első oszlopösszeget az alábbi lépésekkel hozzuk létre: M: B8: G: +B5+B6+B7. E lépések hatására az állapotsorban megjelenik a B8: +B5+B6+B7 alakzat, és a B8 cellában 1800.

A formulában operandusként szereplő cellahivatkozásokat nemcsak billentyűzéssel lehet megadni, hanem a cellamutatóval is. Így hozzuk létre — POINT módban — a C8-as cella értékét, az alábbi műveletsorral:

M: C8; G: +; M: C5; G: +; M: C6; G: +; M: C7 ↵

Ennél a megoldásnál a cellamutatóval jelöljük ki a formulában szereplő cellákat (az állapotsoron ugyanaz az alakzat képződik, mintha bebillentyűztük volna). A fenti műveletsor eredménye: 1500 érték a C8-as cellában.

A harmadik oszlopösszeg kiszámítása előtt ismerkedjünk meg a SYMPHONY beépített függvényeivel. E függvények formátuma a következő:

@ függvénynév (argumentumok)

Például: @COS(A5), @SQRT(B6+F4). Bizonyos függvények argumentumai nem cellák, hanem tartomány(ok). Ilyen például a @ SUM ( ) függvény, amely a zárójelben szereplő tartomány celláinak tartalmát összegzi. A SYMPHONY-ban 89 beépített függvény van (a más nyelvekben megszokott elemi függvények, statisztikai függvények, „szövegkezelő" függvények stb.).

Érdekes lehetőség egy cella értékének megadására az IF függvény alkalmazása. Például ha egy cellához az

@ IF(A5>2, 500, 300)

„feltételes" értéket rendeljük, vagyis ezt a formulát „írjuk be", akkor az adott cellába vagy 500 vagy 300 kerül attól függően, hogy az A5 cella értéke nagyobb, vagy nem nagyobb 2-nél.

A függvények értelemszerűen írhatók be a formulákba, és a függvények argumentumai maguk is lehetnek függvények. Tehát a függvények alkalmazásával nagyon bonyolult formulák képezhetők.

A függvények e rövid ismertetése után képezzük a harmadik oszlopösszeget, a @ SUM függvény alkalmazásával az alábbi lépésekkel:

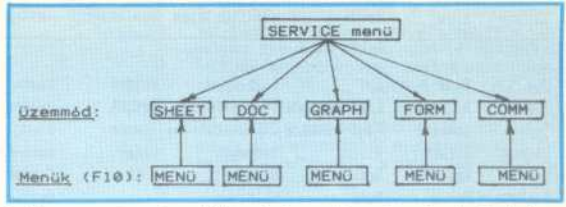
M: D8; G: @SUM(D5 . . D7)

Mint eddig mindig, most is megjelenik az állapotsorban a felírt formula, azaz:

D8: @SUM(D5 . . D7) és a D8 cellában a művelet eredménye: 154.5.

Ezzel kis táblázatunk nagy része el is készült.

Mielőtt továbbmennénk, szeretnénk felhívni a figyelmet a formulák egy fontos tulajdonságára: „dinamikusságukra". A formulák mindig követik a bennük szereplő cellák értékeinek módosulását: ha egy cella értéke megváltozik, akkor a rendszer minden olyan formula értékét (a formulát képviselő cella tartalmát) megváltoztatja, amelyben ez a cellahivatkozás szerepel. A változtatás teljesen automatikus, nekünk semmit nem kell tennünk. Ez egyben azt is jelenti, hogy a formulákat megszerkeszthetjük és beírhatjuk már akkor is, ha még nem minden, a formulában szereplő cellának adtunk értéket. Ilyenkor is kiszámítja a formulát úgy, hogy az üres cellák értékeit 0-nak veszi a rendszer.



8. ábra A menürendszer felépítése

Ismétlődő műveletek egyszerű megoldása a táblán

Táblázatunk ötödik oszlopának kitöltését egy ciklus jellegű eljárással végezzük. Ennek kapcsán be tudjuk mutatni a menüvezérlés lényegét is.

A bevételt úgy számítjuk ki, hogy az eladott mennyiséget megszorozzuk az egységárakkal. Legyen a kenyér egységára 7,50 Ft, a tejé 8,40 Ft, a vajé 120 Ft. Rendeljük hozzá az E5 cellához a Bolt—1 bevételét kiszámító

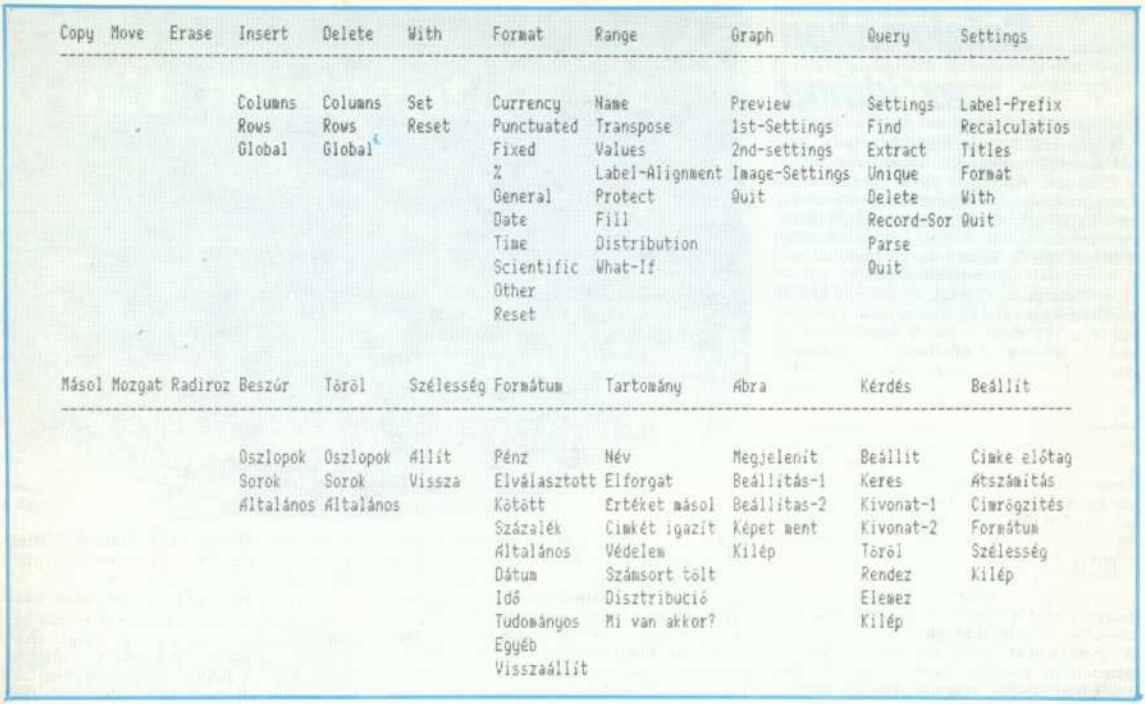
+B5*7.5+C5*8.4+D5*120

formulát, azaz mozgassuk rá a cellamutatót E5-re, és gépeljük be ezt a kifejezést.

A következő három sorba kerülő formulák a fentitől csak abban különböznek, hogy a sorindex mindig eggyel nagyobb. Tehát jól jön, hogy át lehet vinni E5-ből a formulát E6-ba úgy, hogy közben a B5, C5, D5 cellák sorindexe eggyel nő — ezt a Copy nevű másolóutasítással lehet megtenni. Ezt a folyamatot most kissé kiemeljük a tárgyalásból, hogy a relatív cellahivatkozás szerepét részletezhessük.

A formulákbeli relatív és abszolút cellahivatkozások által lesz a táblakezelő program is az, ami. Egy cellába beírt relatív cellahivatkozást a SYMPHONY úgy érti, hogy az aktuális cellától a hivatkozásban szereplő cella viszonylagosan (relatív) milyen távolságra és irányban van. A *7. ábrán* a következőképpen tanulmányozható ez a dolog. A C3-as cellában szeretnénk megjeleníteni az A1, illetve az A2 cellák összegét. Amikor a képletet beírjuk (+A1+C2), akkor a program azt jegyzi meg, hogy az első operandus két cellával felfelé van és két cellával balra. A második operandus egy cellával felette. Ha a C3-as cella tartalmát átmásoljuk a J24-es cellába, ott is a kettővel felette, kettővel balra lévő cella lesz az első operandus (H22), míg a második operandus a felette levő cella (J23) lesz. Így a másolás eredménye a J24-es cellában +H22+J23 formula lesz.

A cellahivatkozások másik fajtája az abszolút cellahivatkozás. Ebben az esetben mindig egy konkrét cellára utalunk, és formulamásolás eredményeként is ugyanarra



9. ábra A SHEET menü felépítése

a cellára fog vonatkozni a képlet. Ezt a cellahivatkozást egy dollárjellel jelöljük az oszlop és/vagy a sorjel előtt, attól függően, hogy az abszolút megkötés melyikre vonatkozik (B5, $B5, B$5).

A Copy parancsot a SHEET üzemmód menüjéből kell kiválasztani. Mivel az eddigiek folyamán nem léptünk ki a SHEET üzemmódból, tehát ha kiadjuk a MENU parancsot, akkor a rendszer a SHEET üzemmód menüjét hozza be (a menünek minden üzemmódban MENU a neve). A klaviatúra egyik funkcióbillentyűjének (F10) benyomása a menü behívását eredményezi (lásd a *8. ábrát*), hatására a parancssorban megjelennek ennek a menünek a parancsai, nevezetesen:

Copy Move Erase Insert Delete Width Format Range Graph Query Settings

Az első parancs ki van világítva, a parancsmutató ide mutat. A kívánt parancsot vagy a kezdőbetű begépelésével, vagy a parancsmutató rámozgatásával (a kurzormozgatók segítségével) és a (↵) billentyű leütésével tudjuk kiválasztani. Nekünk itt éppen a Copyra van szükségünk, tehát amikor a fenti sor megjelenik, elég most a (↵) billentyűt lenyomni. A Copy kiválasztása a menüből csak az egyik lépés: meg kell adnunk az argumentumát is. A Copy másoló parancs, tehát közölni kell vele a „honnan"-t és a „hová"-t. Mivel tulajdonképpen a Copy tartományt másol át tartományba, ezeket kell kijelölni számára. A másolás során formulát tartalmazó tartományt másol át relativ vagy abszolút módon. Ha a cella nem formulát tartalmaz, akkor a konkrét értékeket viszi be a kijelölt tartományba. Bár példánk egy igazán egyszerű esetet reprezentál, jól megvilágítja ezt.

M: E5	kijelöljük az E5 . . E5 tartományt
FB: MENU	behívjuk a menüt
K: Copy	kiválasztjuk a Copy parancsot
M: E6	kijelöljük azt a tartományt, ahová az
FB: TAB	E5 . . E5 tartományt át akarjuk
M: E8	másolni. (A TAB billentyűvel rögzítjük a tartomány bal felső sarkát, majd a ↓ billentyű kétszeri lenyomásával kivilágositjuk az E7, E8 cellákat is (ki van jelölve a tartomány)

Ha a fentiek végrehajtása után lenyomjuk a Return (↵) billentyűt, a Copy utasítás hatása:

az E6 cellához a B6*7.5+C6*8.4+D6*120
az E7 cellához a B7*7.5+C7*8.4+D7*120
az E8 cellához a B8*7.5+C8*8.4+D8*120

formulát rendeli hozzá a rendszer, és mindjárt ki is számítja a megfelelő értékeket. Ezzel táblázatunk teljesen elkészült.

Szerkesztő, beállító, nyomtató műveletek a táblán

A főmenü (a SERVICES) és a táblakezelő menüje az elmondottakon kívül számos más lehetőséget ad a tábla kezelésére. Néhány funkciót külön is megemlítünk közülük (a SHEET menü felépítését a *9. ábrán* láthatjuk):

— az összes cellához illetve a cellák egy tartományához beállítható a megjelenítési formátum, valmint a szövegek elhelyezése a cellában (jobbra vagy balra, vagy középre),

— a tartományok elnevezhetők, és attól kezdve a műveletekben névvel lehet rájuk hivatkozni,

— a cellák egy csoportja védhető a felülírástól, így meg lehet határozni azon tartományok szűk körét, amelyeket a felhasználók változtatnak meg,

— a táblázat egyik helyéről nemcsak átmásolhatunk cellákat, hanem át is mozgathatunk tartalmakat úgy, hogy az eredeti helyen az eltűnik,

— teljes sorokat lehet beszúrni illetve törölni a táblából,

— a tartományok másolását úgy is előírhatjuk, hogy a másolatban csak a cellaértékek kerüljenek át, a formulák nem, illetve hogy a másolás eredményeként a sorokból oszlopok és az oszlopokból sorok legyenek,

— a SERVICES menü Print almenüjének segítségével a táblázatot megszerkesztve, a megkívánt (képernyőn általunk elkészített) formában kinyomtathatjuk,

— stb.

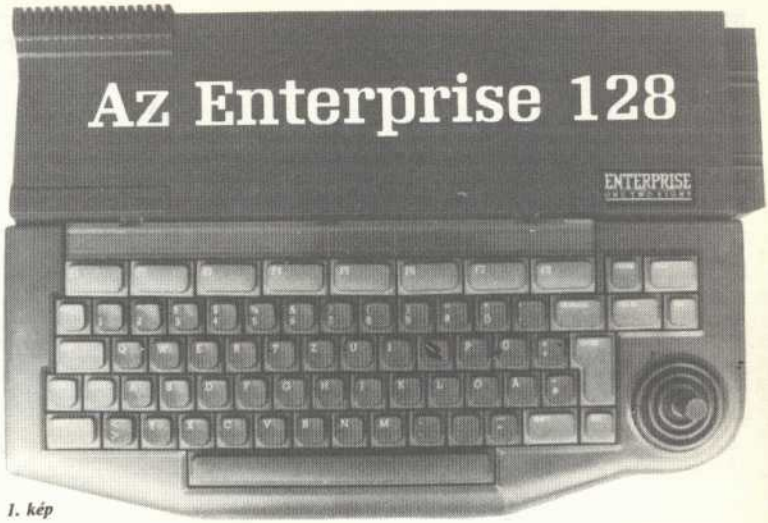
Ezeket itt nem részletezzük: az eddigiek, reméljük, elegendőek a táblakezelő koncepciójának megismeréséhez.

A további üzemmódokat a következő számban ismertetjük.

GERŐ JUDIT
DR. SZELEZSÁN JÁNOS

Szinkronban az idővel

Az Enterprise 128



1. kép

A gép szülőhazája Nagy-Britannia. Ennek ismeretében egy kicsit meglepett, hogy a Centrum Áruházak által forgalmazott Enterprise-oknak német billentyűzetük van. Ez azt jelenti, hogy a számítógépeknél megszokott billentyűzethez képest a Z és az Y helyet cserélt, és néhány jel (például a :, ;, <, >, =, →) is máshova került.

Az Enterprise 128-nak egyébként kiváló tulajdonságai vannak, hiszen mint a neve is mutatja, 128 kbájt RAM-ot tartalmaz, s ez akár 4 Mbájtig is bővíthető. A központi egysége egy Z80A mikroprocesszor, amelynek órajele 4,37 MHz. A képernyőfelbontás sem csekély: 672 × 512 képpont. Mindehhez 256-féle színárnyalat használata lehetséges, és ha úgy tetszik, a szineket magunk is kikeverhetjük a három alapszínből. Beépítettek egy négycsatornás sztereó hanggenerátort is, csatornánként nyolc oktávval. Továbbá: első ránézésre láthatjuk, hogy mint a Videoton TV-Computernél, e gépnél is beépített nyolcirányú botkormány szolgálja a kényelmes kezelést.

Ez a hasonlóság egyébként nem véletlen, hiszen a TV-Computert is az Enterprise 128 alkotója tervezte, igaz, egy kicsit korábban. A perifériákat igen rugalmasan kezeli, nemzetközi karakterkészlet is van benne, amelyben számos magyar ékezetes betű is fellelhető. A BASIC utasításokat átnézve, a BASIC fordítója is kitűnőnek látszik, hiszen kb. 300 utasítást, illetve függvényt ismer. Beépített szövegszerkesztőt is tartalmaz, és akárcsak a Spectrumok, hálózatba is köthetők a gépek (összesen 32 darab).

Az utasítások számának megfelelően sokféle hibaüzenetet kaphatunk. Alapállapotban ezeket németül írja ki a gép, de ha kiadjuk az :UK parancsot, akkor angolul kapunk visszajelzést, sőt ekkor a billentyűzet is normál angol formátumot vesz fel.

Különleges érték, hogy a nagyobb gépekhez hasonlóan egyszerre több (maximálisan 128) BASIC programot tárolhatunk a memóriában. Természetesen csak akkor lehet egyszerre ennyi program a gépben, ha elegendő a memória, mert amikor egy program szerkesztését megkezdjük, a rendszer lefoglal kb. 10 kbájtot. És bár az első program mindössze 42 kbájt, a többi pedig csupán 32 kbájt lehet, de a programok hívhatják egymást, és paraméterek átadására is van mód.

Első benyomásra tehát ez a gép az eddigi legjobb a házi számítógépek kategóriájában, és ehhez képest az ára sem túl magas: magnóval együtt 16 900 forint.

A gép kívülről . . .

Bevallom őszintén, első pillantásra kicsit megijedtem, de már megbarátkoztam az alakjával *(1. kép)*. Nem úgy a billentyűzettel. A már említett helyváltozásokon kívül kényelmetlennek tartom a billentyű megnyomását is. A billentyűzet tartósságának kipróbálására még nem volt lehetőségem, de úgy érzem, nem való tartós igénybevételre (játék). Felül 8 darab funkciógomb kapott helyet, amelyeknek a C16-hoz hasonlóan alapállapotban is szerepük van, azaz bizonyos utasításokat nem kell begépelnünk, hanem ehelyett elég a megfelelő funkciógomb kezelése.

A funkcióbillentyűk mellett található a PAUSE gomb, amivel megállítható, majd újraindítható a rendszer (listázás, programfutás stb.). A jobb felső sarokban van a STOP gomb, amely a programmegszakításra szolgál. Jobb oldalon helyezkednek el továbbá a program szerkesztésére szolgáló ERASE, DEL, INS és ENTER billentyűk is; ezekről később még szó lesz. Ugyancsak itt látható az ALT gomb. Ha ezt egy másik billentyűvel egyszerre nyomjuk le, megjeleníthetjük a nemzetközi karaktereket. A SHIFT gombból szokás szerint kettő van. Bal oldalon kapott helyett a CTRL, a TAB és az ESC gombon kívül a LOCK billentyű, amely nagymértékben megkönnyíti a munkát, ugyanis a SHIFT-tel vagy az ALT-tal együtt lenyomva egy olyan módba léphetünk, ahol csak egy billentyűt kell lenyomnunk, és mégis úgy funkcionál, mintha egyidejűleg nyomnánk a SHIFT-et vagy az ALT-ot

Ha a CTRL-t és a LOCK-ot nyomjuk meg egyidejűleg, akkor az ún. CAPS módba térünk át, ami azt jelenti, hogy mindent nagybetűkkel írhatunk, míg az egyéb billentyűk (például a számok) lenyomásakor a normál kép (a szám) jelenik meg.

A billentyűzet hátrányát kompenzálja bizonyos mértékig az, hogy a különböző funkciókat ellátó gombok különböző színűek, valamint az is, hogy a kurzor a beépített botkormánnyal irányítható.

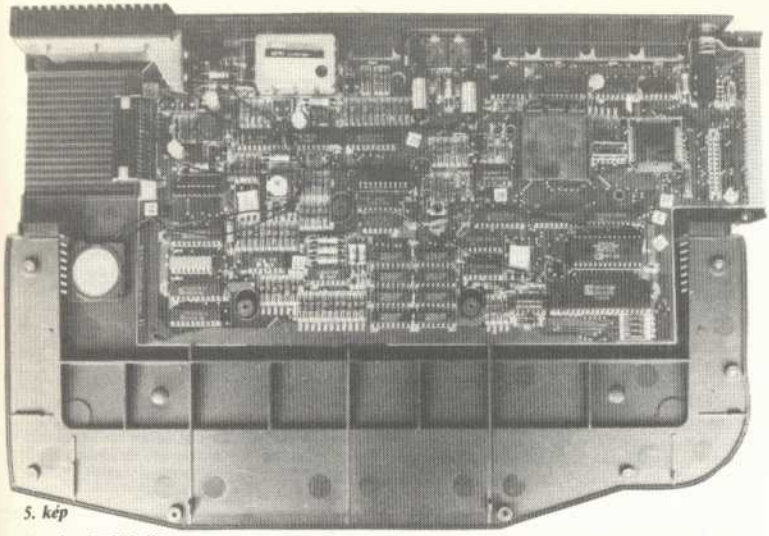
A csatlakozók a gép két oldalán és a hátsó részen helyezkednek el, és jó részük igen nyitott, hiszen maga a panel szolgál erre a célra. Bal oldalon van a BASIC modul helye *(2. kép)*. A BASIC fordító ugyanis — a hazánkban elterjedt házi számítógépektől eltérően — nincs beépítve a gépbe. Jobb oldalon kivezették a rendszer minden vezetékét *(3. kép)*. Ide csatlakozik a lemezmeghajtó vezérlőkártyája, a memóriabővítő és az egyéb programnyelveket (Pascal, LISP, FORTH stb.) tartalmazó modulok. Hátul található a két külső botkormány, a nyomtató, a monitor és a tévécsatlakozó, és itt van lehetőség a hálózatba kötésre is. *(4. kép)*. Ugyancsak a hátsó részen van a magnócsatlakozó, amin keresztül a gép képes megállítani és elindítani az erre alkalmas magnók motorját. A magnókimenetről erősíthetjük fel a gép által keltett hangot, amit egyébként csak a beépített hangszórón hallhatunk. Ugyancsak hátul van a RESET gomb, amelyről még lesz szó a későbbiekben.

Már csak egyetlen csatlakozó maradt hátra, a tápegységé, amely szintén a gép hátulján van, és egyben a bekapcsolásra is szolgál, hiszen nincs külön ki/be kapcsoló. Megvallom, nem tudom, milyen megfontolásból hagyták ezt el. Ilyen megoldást még csak a Spectrumnál alkalmaztak.

2. kép

3. kép

4. kép



5. kép

. . . és belülről

A Z80-as mikroprocesszoron kívül két fontos, speciális integrált áramkört építettek be *(5. kép)*. Az egyik a NICK chip, ami egy grafikus processzor, órajele 14 MHz, és feladata a képernyőszerkesztés. A másik a DAVE, amely több funkciót lát el. Van benne egy hanggenerátor, frissíti a memóriát, kezeli a billentyűzetet. Az utóbbi kettő nem túlzottan érdekes, de a hanggenerátor igen. Négy csatornát használhatunk, és sztereó hangkeltésre is képes, olyan módon, hogy külön megadhatjuk a jobb, illetve a bal csatorna hangerejét. A beépített hangszóró persze csak monó hangok kibocsátására képes, de a magnókimeneten is megjelenik a hanggenerátor jele, így ide fejhallgatót vagy erősítőt csatlakoztatva, már élvezhetővé válik a sztereó hang.

A hanggenerátor különlegessége, hogy a hullámformát mi magunk állíthatjuk be. A C64-en például közismerten csak néhány fajta hullámforma közül választhatunk, itt viszont számtalan variáció lehet. Bár először egy kicsit bonyolultnak tűnik a hangkeltés, de ha valaki megszokja, akkor — úgy gondolom — jól tud vele dolgozni. Részletesebben a BASIC leírásánál írok erről, hiszen először mindenképpen valamilyen magas szintű programnyelvből érdemes kipróbálni, és talán a megértése is könnyebb.

A videoprocesszor néhány tulajdonságát érdemes közelebbről megismerni. Hasonló dolgokra képes, mint az Atari 800XL ANTIC chipje, de talán annál is sokrétűbb, és sokkal nagyobb felbontást tesz lehetővé.

Ötféle grafikus üzemmód van: normál karakteres, 80 karakteres, melyet szoftver segítségével állít elő a NICK, nagy felbontású (HIRES), alacsony felbontású (LORES) és attributum (ATTRIBUTE) grafikai módok. A karakteres módokban kevés szín közül lehet választani, de a grafikus módokban döntés kérdése, hogy 2, 4, 16 vagy 256 színű üzemmódot akarunk-e. Természetesen a színek számának növekedésével csökken a felbontás. A LORES és a HIRES grafika között is csak a felbontásban van különbség. Az attributum üzemmód azt jelenti, hogy külön színmemória van, és így minden bájtra külön háttér- és rajzolásszín választható.

Hasznos tudni, hogy a keretet csak az operációs rendszer teszi a képernyőre, és ennek kikerülésével olyan képet szerkeszthetünk, amelyen a keretre is lehet írni.

A videoprocesszor utasításai egy ún. Line Parameter Table-ben vannak. Itt minden sorra kiválasztható a grafikus üzemmód, a színek száma, a paletta és az információ memóriabeli helye.

A gép felépítésében azonban nem tetszik, hogy néhány helyen átkötést alkalmaztak két kivezetés között.

A BASIC

Az Enterprise 128 BASIC fordítója több mint 300 utasítást ismer. Már ebből is látszik, hogy az eddig elterjedt gépek fordítóihoz képest nagyobb teret biztosít fantáziánk kibontakoztatására.

Kezdjük talán azzal, hogy programunk írása közben egy szövegszerkesztőben dolgozunk, amit az operációs rendszer tesz lehetővé számunkra. Ez többek között azért kényelmes, mert például amikor egy programot listázunk és az nem fér ki a képernyőre, nyugodtan hagyhatjuk kifutni a listát: később a kurzor felfelé való mozgatásával bármikor visszamehetünk a program első sorára is — persze csak meghatározott hosszúságú program esetében. Még nagyon sok más előnye is van ennek a szövegszerkesztő rendszernek: például a tetszés szerint beállítható tabulátor, a margók változtatása, vagy a teljes sorok képernyőn való mozgatása.

Ide tartozik még az is, hogy egy betűt kétféleképpen is törölhetünk. Az egyik módszer az ERASE billentyűvel — úgy, mint a C64-en: a betű mögé állítva a kurzort —, a másik pedig a DEL billentyűvel, ahogy a Spectrumon: a betű elé állva a kurzorral. Természetesen ugyanilyen törlést alkalmazhatunk egy teljes szóra (a legközelebbi üres karakterig töröl) az ALT vagy a CTRL gomb és a törlőbillentyű egyidejű lenyomásával. Sorokat is törölhetünk, ha a törlőbillentyűt a SHIFT-tel együtt nyomjuk le.

A szerkesztőnek egyébként még számos jó tulajdonsága van, de ezekre itt nem térek ki, hiszen a felhasználói kézikönyvben megtalálhatók.

Mivel ez a gép összességében lényegesen több lehetőséget nyújt, mint az eddigiek, ezért nem írok az általános utasításokról, csupán az újdonságokat emelem ki, illetve rámutatok a gyengékre is.

A BASIC maximum 31 karakteres változóneveket képes azonosítani. A változóknak azonban nincs kezdeti értékük, tehát ezt egy értékadással vagy egy definiálással be kell állítanunk. A változók definiálása a NUMERIC és a STRING utasításokkal történik, és ekkor 0 a kezdeti érték. Az értékadásnál nem kell beírnunk a LET utasítást, a gép azonban automatikusan beírja ezt a programba.

A szöveges változók maximális hossza 127 karakter. Ez nagyon kevés, mert például a Commodore gépeken 255 karakter, a Spectrumon és az Atari 800XL-en pedig bármilyen hosszú karakteres változókat használhatunk. Arra azonban ügyelni kell, hogy a változók hosszúsága egy karakteres tömb dimenzionálásakor a maximális értéket veszi fel, ezért ajánlatos például a DIM A$(40) utasítás helyett a STRING a$(40)*8 utasítás kiadása, ahol a * utáni szám a változó maximális hosszúságát jelenti.

Ciklusok szervezésére rendelkezésre áll a DO—LOOP és a FOR—NEXT típus egyaránt.

Feltételszabásra a már megszokott IF—THEN—ELSE-en kívül az IF—END—IF szerkezet is megfelel, ahol nem kényszerülünk arra, hogy az utasításokat egyetlen sorba írjuk. Segítségünkre van még a SELECT CASE—END SELECT szerkezet is, amivel egy szám változó értéke alapján tudunk elágaztatni.

A DEF név utasítás szintén újdonságnak számít BASIC-ben. Ezzel egy utasássornak nevet adhatunk, majd a CALL névvel bármikor végrehajtathatjuk azt. Természetesen a szokásos függvénydefiníciót is ismeri a gép. Mindkét esetben lehetőség van paraméterátadásra. Arra azonban vigyázni kell, hogy ha egy változót csak egy ilyen függvényben definiáltunk, akkor azt a főprogram nem tudja használni.

Ritka a hasonló kategóriájú gépek körében, hogy a BASIC megszakításkezelésre alkalmas. Az Enterprise 128-on ez lehetséges. Először is írnunk kell egy hibakezelő rutint, majd meg kell adnunk a főprogramban, hogy melyik rutint használja a gép hiba esetén. Ezután be kell állítani egy adott változót, így megadhatjuk, hogy hány másodperc múlva kérjük a megszakítást. A számláló lejártával egy bizonyos kódú hibát kapunk, és a vezérlést a hibakezelő rutin kapja meg. Ily módon persze lehetőség van egyéb hibák kezelésére is. Ha például a program egy hibánál hibaüzenettel leállna, ez nem következik be, hanem mi dönthetjük el, hogy mi történjék. Segítségünkre van ebben az, hogy speciális változókban megkapjuk a hiba sorát és típusát.

A grafikát teljes mértékben támogatja a BASIC. Pontok, vonalak és ellipszisek rajzolhatók, de nem akárhogyan. Kiválasztható ugyanis a vonalhúzás módja (normál, AND, OR, XOR) és stílusa, ami szaggatott vonalakkal való rajzolásra ad lehetőséget. Módunkban áll zárt alakzatok kifestése, és

használhatjuk a LOGO nyelvből ismerős teknősbéka-grafikát is, ahol az elfordulás szöge fokokban és radiánban egyaránt megadható.

Szintén a grafikát támogatja az ablaktechnika is. Tetszőleges méretű és grafikus üzemmódú ablakokat definiálhatunk, s miután egy videoablaknak egy számmal ellátott csatornát kell nyitnunk, az ablakra úgy rajzolhatunk, hogy a csatornaszámot beírjuk a rajzolóutasításba. Egy ablakot bárhol elhelyezhetünk a képernyőn, de mivel erre külön utasítás szolgál, ezért megtehetjük, hogy úgy rajzolunk egy ablakra, hogy az nem látható, majd amikor kész a rajz, láthatóvá tesszük. Ezáltal gyors mozgások és változások jeleníthetők meg, igen látványosan.

Nemcsak a videoablakokat, de szinte minden eszközt csatornákon keresztül érhetünk el. A csatornák így igen fontos szerepet töltenek be ezen a gépen, s úgyszólván el sem hagyhatók egy-egy programból.

```
10 FOR I=1000001 TO 1000003 STEP 2
20   FOR D=3 TO SQR(I)
30     IF I/D=INT(I/D) THEN 50
40   NEXT D
50   PRINT D
60 NEXT I
```

1. lista

```
 10 LET SI=7001
 20 DIM FL(7001)
 30 PRINT "ONLY ONE ITERATION"
 40 LET CO=0
 50 FOR I=0 TO SI
 60   LET FL(I)=1
 70 NEXT I
 80 FOR I=0 TO SI
 90   IF FL(I)=0 THEN 170
100   LET PR=I+I+R
110   LET K=I+PR
120   IF K>SI THEN 160
130   LET FL(K)=0
140   LET K=K+PR
150   GOTO 120
160   LET CO=CO+1
170 NEXT I
180 PRINT CO,"PRIMES"
```

2. lista

```
10 FOR I=1 TO 1000
20 NEXT I
```

3. lista

```
10 FOR I=1 TO 1000
20   PRINT "ENTERPRISE 128"
30 NEXT I
```

4. lista

```
10 FOR I=0 TO 100
20   LET S=I)I
30 NEXT I
```

5. lista

```
10 GRAPHICS HIRES 2
20 FOR A=0 TO 1279 STEP 4
30   PLOT A,0;A,719
40 NEXT I
```

6. lista

7. lista

```
10 GRAPHICS HIRES 2
20 FOR I=1 TO 1000
30   LET A=RND*1279
40   LET B=RND*719
50   PLOT A,B
60 NEXT I
```

	3. LISTA	4. LISTA	5. LISTA	6. LISTA	7. LISTA
C-64	1.60	41.00	5.50		
PLUS/4	1.60	43.00	4.79	23.00	25.90
ATARI 800XL	2.42	30.79	22.63	34.13	49.16
DRAGON	1.17			7.55	
ENTERPRISE 128	2.17	99.40	1.46	17.92	114.00

Az eredmények másodpercben

1. táblázat

géptípus	10⁵ ng-ben
C-64	255
APPLE IIe, APPLESOFT	230
IBM PC BASICA	190
IBM PC SBASIC	1
PLUS/4	325
ATARI 800XL	211
DRAGON	200
ENTERPRISE 128	412

2. táblázat

A hanggenerátor programozására csupán két utasítás szolgál, de ezek annyira rugalmasak és változatosak, hogy elegendők a hangkeltéshez. Az egyik, az ENVELOPE a hullámformát választja ki, a másik, a SOUND pedig a hangkeltésre szolgál. A gép arra is képes, hogy zenélés közben rajzoljon. Ha többszólamú zenét írunk, egyszerre is indíthatjuk őket. A hang stílusának kiválasztásával különböző torzításokat érhetünk el. Így állítható elő például a szintetizátor hangja.

A beépített függvények mennyiségétől elámultam. Olyanokat találtam köztük, amilyeneket még sehol sem láttam. Most csak néhányat sorolok fel.

Itt van mindjárt az ANGLE, amely egy egyenes iránytangensét számítja ki. Beépítették a trigonometrikus függvények ellentettjét és hiperbolikus függvényeit. A fokokat átszámolja a gép radiánba és vissza. Számok felfelé és lefelé kerekítésére egyaránt lehetőség van. A természetes alapú logaritmuson kívül a tízes és a kettes alapú is rendelkezésre áll. A maradékos osztást kétféleképpen kérhetjük: az egyik egész, a másik tört maradékot számol. Sokféle új függvény szolgál karakteres változók és tömbök kezelésére is. Egy „új" függvényt is találtam: aki ismeri a BASIC nyelvet, biztosan találkozott már az ASC függvénnyel; ennek itt csak a neve más (ORD), egyébként mindenben megegyezik a megszokottal.

A gépi kódú rutinok használata is kényelmes. A rutinoknak nevet adhatunk, s utána ezzel a névvel hivatkozhatunk rájuk. A HEX$ függvény segítségével a hexadecimális bájtsorozatban adhatjuk meg a gépi kódú programot.

Az EXOS

Az EXOS az Enterprise 128 ki/bemeneti rendszere. Rutinjainak hívása az RST 30h gépi utasítással történik, ami után egy adatnak kell állnia. Ez az adat határozza meg a funkciót. Az EXOS változóinak írása egyébként BASIC-ből is lehetséges a SET utasítással.

A perifériák

Perifériák tekintetében is rugalmas a gép. Bármilyen magnetofon csatlakoztatható hozzá, s amelyik távvezérelhető, az irányítja is.

A lemezmeghajtó vezérlőkártyájához mindenféle párhuzamos lemezmeghajtót minden beavatkozás nélkül tudunk csatlakoztatni. Ha soros lemezmeghajtót (például VC—1541-et) akarunk csatlakoztatni a géphez, akkor egy programra van szükség, amely kezeli azt. A DOS egyébként nagyon intelligens, hiszen többféle lemezformátumot képes olvasni, többek között az IBM-ét is.

A Printer feliratú csatlakozón keresztül szinte mindenféle nyomtató hozzáköthető az Enterprise-hoz: például az EPSON nyomtatókat is képes kezelni.

A vizsgálóprogramok

A vizsgálóprogramok főként a BASIC fordító sebességét mérik, és összehasonlítási lehetőséget teremtenek a gépek között. A Mikromagazin 1987. áprilisi számában a 40. oldalon közölt eredményeket egészítettem ki az Enterprise 128-on mértekkel.

Az első program *(1. lista)* kivételesen nem a gyorsaságot, hanem a számítási pontosságot vizsgálja. A gép akkor számol pontosan, ha a program a belső ciklusból csak D=101 értéknél ugrik ki. Ez az Enterprise-on így is történik.

A *2. listán* látható program az ún. prímszámteszt. Az eredményeket az *1. táblázatról* olvashatjuk le. Figyelembe kell azonban venni, hogy az Atari csak ötezer elemű tömb dimenzionálására volt képes. A többi gépen ez a szám hétezer.

A következő 5 program futási eredménye a *2. táblázatban* látható. A legegyszerűbb *(3. lista)* csupán egy üres ciklus, amit ezerszer hajt végre a gép. A következő program *(4. lista)* csupán egy PRINT utasításban különbözik az előzőtől, de lényegesen tovább tart (az Enterprise-on különösen). Gondoljunk viszont arra, hogy ezen a gépen minden kiíratást a szövegszerkesztő dolgoz fel, ami sok időt vesz igénybe.

Az *5. lista* programja a hatványozás sebességét méri. Ez az Atari 800XL hátrányainak bemutatására készült, de bizonyára segít a teljesebb kép kialakításában.

Az utolsó két program a grafika sebességét teszteli. Az első *(6. lista)* a legnagyobb felbontást megengedő grafikus üzemmódban tölti fel a képernyőt vonalanként. Azt hiszem, az Enterprise eredménye figyelemre méltó, ha azt is tekintetbe vesszük, hogy itt négyszer akkora a képernyő mérete, mint a többi gépnél.

A másik grafikai program *(7. lista)* ezer pontot helyez el véletlenszerűen a nagy felbontású képernyőn. Az Enterprise 128 rossz eredménye azt mutatja, hogy az RND függvény lassítja le a programot, hiszen az előzőekben láttuk, hogy a vonalhúzás nem lassú. A véletlenszám-generálás egyébként elég szorosan kapcsolódik a grafika használatához, így nem fölösleges ennek tesztelése sem.

Ábrakészítés előadásokhoz, cikkekhez

Beszámoló egyes szám első személyben

Indokaim

Tudományos cikkek, előadások ábráinak elkészítése minden szerző keresztje. Nem elég jól megfogalmazni a mondandót, ezt illusztrálni kell a könnyebb érthetőség kedvéért. A legritkább esetben van olyan grafikus, fotós, aki a sok munkát és türelmet igénylő ábrák elkészítéséhez érdemi segítséget képes nyújtani. Arról nem is beszélve, hogy a megfelelő alapanyagok nem olcsók, és a munkahely többnyire nem tud anyagilag hozzájárulni az effajta „minőségemeléshez". A szerző tehát még alaposan a zsebébe is nyúlhat, ha valóban szép ábrákkal akarja mondanivalóját kiegészíteni.

Miért kell ezeket az ábrákat lefényképezni? Az előadások tartásakor lényegesen egyszerűbb automata vetítővel diákat vetíteni, mint akár episzkóppal, akár más módon az ábrákat mindenki számára láthatóvá tenni.

Mi a legolcsóbb megoldás? Az ember fia elővesz egy írógépet, papírt fűz bele, az ábrát jól—rosszul legépeli, majd lefényképezi. Ha tudja, a kész negatívot kékíti, színezi. Ha nem, az egész marad fekete-fehér.

Ezen a problémán azon az osztályon, ahol dolgozom, a mikroszámítógép alkalmazása alapvetően segített. Ábráinkat Spectrum 48k mikroszámítógép segítségével, rajzolóprogram alkalmazásával készítem. A program a *Melbourne Draw* nevű grafikai tervező program általam módosított változata, és használata igen rövid idő alatt elsajátítható.

Hogy miért az Mdraw programot választottam? Sajnos sem a Leonardo nevű, sem sok más jobb program nincsen meg a programkönyvtáramban. Így maradok az ősi igazságok mellett: „Addig nyújtózkodj, amíg a takaród ér", illetve „Szegény ember vízzel főz." Habár az *Mdraw* messze van a víztől!

Tetteim

Annak ellenére, hogy az igen kiváló *The Artist* nevű programom megvan, a kettő közül az előbbit tartom igazán alkalmasnak ábrák készítésére. Hogy miben különbözik a két program? Nos, az *Artist* tud kört, ívet rajzolni, különböző mintával lehet alakzatokat kitölteni, az „ecset" mérete változtatható. Rajzoláshoz valóban sokkal jobb. Az előadásokhoz viszont gyakran kell „álló" ábra, amihez több irányban kell hozzáírni. Gyakran előfordul, hogy a megtervezett képen módosítani kell, ilyenkor van szükség a scroll funkcióra, hogy az ábra elhelyezése ismét arányos legyen. És mivel az ábrák többsége szöveges, így számomra egyértelműen az Mdraw a kényelmesebb. Természetesen, ha valami szép rajzos ábra kívánkozik az anyaghoz, a rajzot az Artist segítségével készítem. A kimentett képet ezután a Mdraw-ba töltve megírom a szöveget.

A módszer rendkívüli előnye, hogy az ábrák szövege percek alatt kiváló minőségben, költség nélkül elkészíthető. A televízió képernyőjéről az itt közölt eljárással a diák vagy negatívok az igény szerint — fekete-fehérben vagy színesben (tetszőleges színekben) elkészíthetők.

Az ábrák elkészítéséhez természetesen bármilyen típusú számítógép megfelel, amennyiben az erre való program rendelkezésre áll. Mindennek jelentőségét az időszerűség adja: a számítástechnika eszközeivel való kapcsolatunk egyre inkább mindennapos, több és több helyen használnak számítógépet.

A tudnivalók

Az eljárás menete a következő. A számítógéppel elkészített ábrát nagyképernyős televízióról fényképezem. A színek beállítása után a felvételeket tükörreflexes fényképezőgéppel 20 DIN-es filmmel, 5,6-os blendével, 1 sec expozíciós idővel, állványról készítem. (A tévékép vibrálása miatt ajánlatos az 1 másodperces expozíciós időt választani.) A felvételeket sötétben (hogy a képernyő ne tükröződjék) exponálom.

A kész diák (expressz hívás esetén) 1 nap múlva keretezhetők, illetve a képek ragaszthatók. Eddig több poster készült, és 2 éve előadásaink diáit is így fényképezzük. Ez a módszer nagyobb lehetőséget kínál az előadóknak előadásaik illusztrálására. Néhány képpel több elkészítése nem jelent lényegesen nagyobb munkát, az előadások viszont élvezetesebbé válnak.

Eredményeink magukért beszélnek. Az előadásokhoz, posterekhez szükséges 10-15 ábrát néhány óra alatt elkészítem. Előzetesen kockás papíron skiccelni csak komplikált vagy több részből álló táblázatoknál van szükség. Nem kell szabni, rajzolni, ragasztani. A betűk egyformák, több betűtípus szerepelhet akár egy ábrán belül is. Az ábra kiosztása, elhelyezése könnyen megváltoztatható. Az ábrák percek alatt átszínezhetők, hiszen nem egyszer hasznos ugyanazon ábra többszöri kivetítése más-más gondolat kiemelésével. A kép tárolható, bármikor könnyedén módosítható, új adatokkal kiegészíthető. Az elkészített „alapábrák" többször felhasználhatók.

Számítógépes táblázataink és ábráink az elmúlt 2 év orvosi kongresszusain voltak láthatók. Kivitelük, színeik semmiben nem maradtak el a többi előadó ábráitól.

Az eredeti Melbourne Draw program rövid ismertetése

A Melbourne Draw nevű program készítője Philip Mitchel, forgalmazója a Melbourne House. A rajzoló képernyő a 32768-as címnél kezdődik, hossza 7680 bájt. A gépi kódú program 40960-tól 6600 bájt.

A program betöltése után megjelenik a menü (*1. ábra*). A *p* gombbal indíthatjuk a gépi kódú programot .A többi betű gomb a kimentő illetve betöltő funkciót aktiválja. Az *e* megnyomásával BASIC-be léphetünk ki. A program GO TO 30 utasítással indítható újra.

A gépi kódú program indítása után üressé válik a képernyő, és az alsó két sorban tájékoztató feliratok jelennek meg. Az *x* a vízszintes, az *y* a függőleges irányú koordinátáját jelöli a rajzoló pontnak, illetve az éppen írt betű bal alsó sarkának. A MOD utasítás után két szó áll.

Az első:

SKIP	— rajzolás nélküli mozgatás	(SPACE)
SET	— rajzolás	(ENTER)
RESET	— törlés	(0)
INVERT	— ahol tinta van, törli, ahol nincs, ott ír	(i)
SCROLL	— scroll	(k)

A második:

SCREEN	— rajzolás	
ATTR.	— attributum színezés be/ki	(h)

Rajzolni a QWE
A D
ZXC betűk használatával lehet.

BILLENTYŰ	PARANCS
m	nagyítás
n	kicsinyítés norm. méretre
b 0—7	border
0—7	ink
csh + 0—7	paper
B	bright be/ki
V	flash be/ki
F	alakzat kitöltése
g	sakktábla raszter ff. be/ki
G	sakktábla raszter színes be/ki
L	tükrözés
R	képernyőtörlés ill. színbeállítás
csh + 9	információsor fent/lent
csh + 8	kicsinyítés/nagyítás
csh + space	vissza a menühöz
u	az aktuális betűhely betűmintájának elhelyezése valamelyik UDG karakterre

A *t* megnyomása után a TEXT felirat jelenik meg, és a képmezőn üres nyíl látható. A program nagybetűt ír, a kisbetű caps shift + betű lenyomásával írható. A delete nem működik. A csh + 5,6,7,8 vándoroltatja a nyilat, a grafikus üzemmód használható. A csh + sys együttes lenyomása után a nyíl fekete lesz, az írás iránya beállítható csh + 5,6,7,8 billentyűkkel.

Módosításaim

Mivel a program betöltése után más nem található a memóriában, megpróbáltam a maradék

Végezetül

Mindent összevetve a gép kiemelkedik a házi számítógépek kategóriájából, s valahol a C64 és az Amiga között helyezkedik el. Az egyetlen nagyobb probléma az, hogy a kiegészítő egységek és perifériák drágák és nehezen szerezhetők be. Igaz, hogy bármilyen tévére ráköthető, de az ilyen nagy felbontás megjelenítésére alkalmas monitor ára hazánkban több mint 100 ezer forint. A nyomtató és a lemezmeghajtó sem olcsó. Valószínűnek tartom azonban, hogy ha nagyobb mennyiségű, tizezernél több Enterprise-t sikerül eladni, akkor valamelyik cég importál hozzá lemezmeghajtót is, hiszen abból is elég sokat tudnának eladni ahhoz, hogy viszonylag olcsón lehessen forgalomba hozni. Nyomtatónak megfelel a Seikosha SP180 is. Ezt a kis nyomtatót, amely tudásban megközelíti az EPSON 80-ast és 100-ast, már nálunk is lehetett kapni 24 300 forintért. Persze ki tudja, lesz-e még belőle ezután is.

Ami a dokumentációt illeti: az a felhasználói kézikönyv, amelyet a géphez adnak, messze felülmúlja a megszokott szintet, bár egy-két hiba ebben is felfedezhető. Ha azonban valaki nem akar BASIC-ben programozni, vagy esetleg szeretné belülről megismerni a gépet és gépi kódban programozni, akkor nehéz dolga van. Az Enterprise-ról ugyanis külföldön is kevés dokumentáció jelent meg.

A géphez egyelőre 49 program kapható. Ezek között található felhasználói és játékprogram egyaránt. Tulajdonképpen kezdetnek ez jó, hiszen a Commodore is így kezdte. A további szoftverek megírása a magyar programozókra vár.

Az Enterprise-nak, tartozékainak és programjainak forgalmazói a Centrum Áruházak. A szervizhálózat kialakítását megszervezték, kiterjedése az eladott gépek számától, azaz az igényektől is függ.

Az Enterprise 128 nagyobb gépek termináljaként is működtethető, az összekötés nem túl bonyolult. A gép kiváló programozás tanulására, többféle magas szintű programnyelven. No és természetesen tökéletesen megfelel otthoni szórakozásra, játékra.

ROMVÁRI GÁBOR

```
      MELBOURNE DRAW
           by
      Philip Mitchell
Extended by Gresz Miklos 1985

Main Menu :
  (p) > edit picture 2
  (e) > exit program
  (s) > save picture 2 to tape
  (l) > load picture 2 from tape
  (v) > verify picture 2 on tape
  (S) > save UDG area to tape
  (L) > load UDG area from tape
  (V) > verify UDG area on tape
  (b) > letter change
  (k) > SCREEN$  change
  (m) > SCREEN$ XOR, OR

Press the key in the brackets
for the desired command.
```

1. ábra *1/a. ábra*

```
>C $
BDB0  1100E2    LD    DE,#E200
BDB3  210080    LD    HL,#8000
BDB6  0618      LD    B,#18
BDB8  C5        PUSH  BC
BDB9  0600      LD    B,#00
BDBB  C5        PUSH  BC
BDBC  7E        LD    A,(HL)
BDBD  F5        PUSH  AF
BDBE  1A        LD    A,(DE)
BDBF  47        LD    B,A
BDC0  F1        POP   AF
BDC1  B0        OR    B
BDC2  77        LD    (HL),A
BDC3  23        INC   HL
C
BDC4  13        INC   DE
BDC5  C1        POP   BC
BDC6  10F3      DJNZ  #BDBB
BDC8  C1        POP   BC
BDC9  10ED      DJNZ  #BDB8
BDCB  C9        RET
BDCC  00        NOP
BDCD  00        NOP
BDCE  1100E2    LD    DE,#E200
BDD1  210080    LD    HL,#8000
BDD4  0618      LD    B,#18
BDD6  C5        PUSH  BC
BDD7  0600      LD    B,#00
C
BDD9  C5        PUSH  BC
BDDA  7E        LD    A,(HL)
BDDB  F5        PUSH  AF
BDDC  1A        LD    A,(DE)
BDDD  47        LD    B,A
BDDE  F1        POP   AF
BDDF  A8        XOR   B
BDE0  77        LD    (HL),A
BDE1  23        INC   HL
BDE2  13        INC   DE
BDE3  C1        POP   BC
BDE4  10F3      DJNZ  #BDD9
BDE6  C1        POP   BC
BDE7  10ED      DJNZ  #BDD6
BDE9  C9        RET
BDEA  00        NOP
BDEB  00        NOP
BDEC  2100E5    LD    HL,#E500
C
BDEF  110040    LD    DE,#4000
BDF2  01001B    LD    BC,#1B00
BDF5  EDB0      LDIR
BDF7  C9        RET
BDF8  00        NOP
BDF9  00        NOP
BDFA  00        NOP
```

```
1100>REM
1101 REM Betukeszlet valtas
1102 REM
1110 IF PEEK 23607=60 THEN POKE 23607,185: POKE 23676,189: GO TO 30
1120 IF PEEK 23607=185 THEN POKE 23607,190: POKE 23676,194: GO TO 30
1130 IF PEEK 23607=190 THEN POKE 23607,194: POKE 23676,198: GO TO 30
1140 IF PEEK 23607=194 THEN POKE 23607,198: POKE 23676,159: GO TO 30
1150 IF PEEK 23607=198 THEN POKE 23607,60: POKE 23676,190: GO TO 30
1170 GO TO 30
2000 REM
2001 REM Kepernyo masolas
2002 REM
2100 PRINT AT 5,0;"  k ";AT 7,2;"                              "
2110 PRINT "                                  "
2120 PRINT "  SCREEN$ 2 TO 1--------------1"
2130 PRINT "  SCREEN$ 3 TO 1--------------2 "
2140 PRINT "  SCREEN$ 4 TO 1--------------3 "
2145 PRINT "                               "
2150 PRINT "  SCREEN$ 2 TO 3--------------4 "
2160 PRINT "  SCREEN$ 2 TO 4--------------5 "
2165 PRINT "                               "
2170 PRINT "  SCREEN$ 3 TO 2--------------6 "
2180 PRINT "  SCREEN$ 4 TO 2--------------7 "
2190 PRINT "                               "
2210 LET b$=INKEY$: IF b$="" THEN GO TO 2210
2215 IF CODE INKEY$>55 OR CODE INKEY$<49 AND CODE INKEY$<>13 THEN GO TO 2210
2216 IF CODE INKEY$=13 THEN GO TO 30
2220 LET a=48622: LET b=48625: IF VAL b$=1 THEN POKE a,128: POKE b,64: RANDOMIZE
 USR 48620: PAUSE 0: PAUSE 0
2230 IF VAL b$=2 THEN POKE a,202: POKE b,64: RANDOMIZE USR 48620: PAUSE 0: PAUSE
 0
2240 IF VAL b$=3 THEN POKE a,229: POKE b,64: RANDOMIZE USR 48620: PAUSE 0: PAUSE
 0
2250 IF VAL b$=4 THEN POKE a,128: POKE b,202: RANDOMIZE USR 48620
2260 IF VAL b$=5 THEN POKE a,128: POKE b,229: RANDOMIZE USR 48620
2270 IF VAL b$=6 THEN POKE a,202: POKE b,128: RANDOMIZE USR 48620
2280 IF VAL b$=7 THEN POKE a,229: POKE b,128: RANDOMIZE USR 48620
2290 GO TO 30
2489 REM
2490 REM OR es XOR masolas
2491 REM
2500 PRINT AT 5,0;"  m ";AT 7,2;"                              "
2510 PRINT "                                  "
2520 PRINT "  SCREEN$ 3 OR 2---------------1"
2530 PRINT "  SCREEN$ 4 OR 2---------------2"
2540 PRINT "                               "
2550 PRINT "  SCREEN$ 3 XOR 2--------------3"
2560 PRINT "  SCREEN$ 4 XOR 2--------------4"
2570 PRINT "                               "
2580 PRINT "                          "
2590 PRINT "                          "
2600 PRINT "                          "
2610 LET b$=INKEY$: IF b$="" THEN GO TO 2610
2615 IF CODE INKEY$<52 OR CODE INKEY$<49 AND CODE INKEY$<>13 THEN GO TO 2610
2616 IF CODE INKEY$=13 THEN GO TO 30
2620 LET a=48562: LET b=48565: IF VAL b$=1 THEN POKE a,202: POKE b,128: RANDOMIZ
E USR 48560
2630 IF VAL b$=2 THEN POKE a,229: POKE b,128: RANDOMIZE USR 48560
2640 LET a=48592: LET b=48595: IF VAL b$=3 THEN POKE a,202: POKE b,128: RANDOMIZ
E USR 48590
2650 IF VAL b$=4 THEN POKE a,229: POKE b,128: RANDOMIZE USR 48590
2660 GO TO 30
```

2. lista
3. lista

A memória kiosztása

	40960 - gépi kód kezdete
	47560 - gépi kód vége
16384 - képernyő kezdete	47616 - 1. betükészlet
22528 - ATTR. kezdete	48384 - 1. UDG
23296 - képernyő vége	48560 - OR
.	48590 - XOR
.	48620 - képernyő áthelyezés
.	48640 - eredeti UDG
32768 - rajzolómező kezdete	48896 - 2. betükészlet
38912 - ATTR. kezdete	49664 - 2. UDG
39680 - rajzolómező vége	49920 - 3. betükészlet
39681 - ATTR. tárolás kezdete	50688 - 3. UDG
40448 - ATTR. tárolás vége	50944 - 4. betükészlet
.	51712 - tartalékképernyő I.
40704 - 4. UDG	58624 - tartalékképernyő II

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 szóláncokat permutáltuk, a szóláncvariánsokat pedig alfabetikusan rendeztük.

A tartalomleírás egy szólánc6l áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkereséséhez a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szóláncában elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szűkebben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karaktersorozatát nyíl vezeti be, melyet a / jelig a folyóiratok azonosítója, a két / jel kozott az évszám, folyóiratszám és kötőjellel a kezdő oldalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

A folyóiratok a SZÁMALK szakkönyvtárában (Budapest XI., Szakasits Á. út 68. Nyitva: 8-tól fél 5-ig. Tel.: 853-111/251) is föllelhetők. A kiválasztott anyagról másolat rendelhető az alábbi formában:

SZÁMALK Szakkonyvtára
Budapest, 112. Pf.: 146. 1502

Megrendelem a Mikroszámítógép Magazin 1987/ sz. alapján a következő folyóirat-oldal-másolatokat:

Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____

A megrendeléshez csatolom az oldalankénti 8,- Ft-os szolgáltatási díj befizetését igazoló csekkszelvényt.
Dátum, név, pontos cím.

```
PROGRAMLISTA
   apple iipascal nyelvteknocgrafika
   ->hc/86.08-63/3
PROGRAMLISTA
   atari 400/800/xl(vbi-spritemover)
   ->hc/86.08-83/2
PROGRAMLISTA
   cikksorozat commodore 128 grafika 3d
    effektus az ultra-hiressel
   ->run2/86.08-51/6
PROGRAMLISTA
   commodore 128(bootmaker 128/64) boo
   t-szektor kialakitasa a lemezen auto
   matikus atkapcsolas es behuzas c64 p
   rogram eseteben ->run/86.08-42/2
PROGRAMLISTA
   commodore 16 grafika tanacsok a (gra
   phic 1) modus hasznalatahoz
   ->run2/86.08-108/4
PROGRAMLISTA
   commodore 16/116/plus4 grafika statis
   sztikai adatok szemleltetese
   ->hc/86.08-48/2
PROGRAMLISTA
   commodore 16/116/plus4 jatekprogram
   (black jack) ->hc/86.08-49/3
PROGRAMLISTA
   commodore 16/116/plus4 jatekprogram
   (diam) ->hc/86.08-45/4
PROGRAMLISTA
   commodore 16/116/plus4 jatekprogram
   (steinwurf) ->hc/86.08-51/2
PROGRAMLISTA
   commodore 16/116/plus4 szinmodositas
    idegen programokban f billentyukkel
   ->hc/86.08-69/1
PROGRAMLISTA
   commodore 64 (disk keeper) 8-teletes
    menu lemezkezeleshez
   ->run/86.08-34/7
PROGRAMLISTA
   commodore 64 (input)-modosito rutin
   kettospont/veszzo hasznalathoz
   ->chip/86.08-129/1
PROGRAMLISTA
   commodore 64 (superpacker) programto
   morites tobbreszes jatekok egyetlen
   fileba egyesitese
   ->run2/86.08-80/11
PROGRAMLISTA
   commodore 64 (vokabel-trainer) szota
   rkeszites angol nyelvtanulashoz
   ->64er/86.08-53/6
PROGRAMLISTA
   commodore 64 grafika atszamitas hire
   s-rol lores-re automatikus karakterk
   eszlet modositas ->64er/86.08-66/3
PROGRAMLISTA
   commodore 64 grafika jatekprogram ke
   szites (zaxxon)-tipusu lores rajzgra
   fika ->run2/86.08-70/6
PROGRAMLISTA
   commodore 64 hyperbasic utasitaskesz
   let bovites strukturalt programozash
   oz ->64er/86.08-91/3
PROGRAMLISTA
   commodore 64 jatekprogram keszites (
   game-tool) utasitaslista alkalmazasi
    utmutato ->hc/86.08-37/9
PROGRAMLISTA
   commodore 64 lemezegyseg(1541) lemez
   -dos onallo hasznalata kereso rutino
   k ->run2/86.08-46/5
PROGRAMLISTA
   commodore 64 tomoritett programok sz
   ethuzasa es tesztelese
   ->run2/86.08-66/4
PROGRAMLISTA
   commodore 64 (makechars) felhasznalo
   i karakterek szerkesztese a kepernyo
   n ->run/86.08-54/5
PROGRAMLISTA
   commodore 64/128 input-get kombinaci
   o fuzerbeolvasashoz
   ->run/86.08-64/3
PROGRAMLISTA
   commodore amiga grafika (amiga paint
   er) ->run2/86.08-28/5
PROGRAMLISTA
   grafika nyomtato(mps802) 600x400 pon
   tos felbontas ->64er/86.08-91/1
PROGRAMLISTA
   sinclair spectrum jatekprogram (clas
   sroom adventure) ->zx/86.08-82/7
PROGRAMLISTA
   sinclair spectrum jatekprogram (real
   ms of interaction) ->zx/86.08-35/3
PROGRAMLISTA
   sinclair spectrum jatekprogram (wate
   rline) ->zx/86.08-94/2
PROGRAMLISTA
   sinclair spectrum programvirus hatas
   anak szemleltetese redcode nyelv alk
   almazasa ->hc/86.08-71/5
PROGRAMLISTA
   sinclair spectrum szabad tarolokapac
   itas folyamatos kijelzese
   ->happ/86.08-54/3
PROGRAMLISTA
   sinclair zx81 gepi kodu input
   ->zx/86.08-81/1
PROGRAMLISTA
   strukturalt programozas cikksorozat
   commodore 64/128 (comalchen) peldapr
   ogram ->64er/86.08-128/8
PROGRAMLISTA
   szimulacio commodore 64 digitalis ar
   amkorok tesztelese
   ->64er/86.08-59/7
PROGRAMLISTA
   wordstar adatvisszamentesi trukkok
   ->chip/86.08-128/2
```

helyet kihasználva bővíteni a lehetőségeket. Az ábrák szépítéséhez első lépcsőként négy újabb betűkészletet helyeztem a memóriába. Egy kis BASIC kiegészítéssel így öt betűtípus közül lehet választani.

Az ábrák fényképezése az eredeti változatban kissé komplikált. A kész képet ki kell menteni, majd a 16384-es címre betölteni. Mindezt azért, mert az információs sor a kirajzolt képernyőről nem tűnik el. Ezen egy rövidke BASIC toldalékkal, valamint egy nyúlfaroknyi gépi kóddal lehet segíteni. Így a rajzolómező tartalma áthelyezhetővé válik.

Az előbbi számítások után még mindig maradt üres hely a memóriában. A fényképezés megkönnyítésére gondolva azt a megoldást választottam, hogy még kijelölhető legyen két képmező. Ismét egy néhány soros BASIC kozmetika kellett, és bármelyik memóriarész tartalma bárhová áthelyezhető. Így három ábra tölthető egyszerre a gépbe, és fényképezhető.

Humorom

S a programozó látá, hogy műve tökéletes. Megpihenhet babérjain, és élvezheti munkája gyümölcsét. (Avagy: „Minden cigány a maga lovát dicséri!")

Lőn este, s lőn reggel. Sokadik nap.

Beütött a krach!

Két ábrát össze kellett fésülni. Azaz egy kész rajzot egy kész szöveggel kellett egy képmezőbe helyezni. Ismét néhány BASIC sor, valamint egy kevés gépi kód, s a probléma megoldhatóvá vált. (*2. és 3. lista*)

„Magánügyeim"

Ha bárkit további részletek érdekelnek, készséggel állok rendelkezésre. (Természetesen . . .) Én viszont a következő kérdésre várnám valakinek a válaszát:

— Hogyan lehet a Spectrum „színzászló"-ját eltüntetni? Ha valakinek van erre kapcsolása vagy ötlete, örömmel venném. (Természetesen . . .)

Itt a memória vége, fuss el véle! A programozó pedig boldogan él, amíg megint valami újabb marhaság nem jut eszébe. HM hm hm . . .

DR. GRESZ MIKLÓS

1. Turing tesztje — mi a mesterséges intelligencia?

Alan Turingot, a neves matematikust bosszantották azok a terméketlen viták, melyek akörül folytak, mi az *intelligencia*, mit jelent az, hogy *gondolkodni*. Ezért definíció helyett 1950-ben egy próbát (tesztet, kísérletet) ajánlott. Ennek alapján javasolta majdan eldönteni (ha egyszer valaki előáll azzal, hogy szerkesztett egy intelligens gépet), hogy a művileg előállított intelligenciára irányuló teljesítmény, illetőleg a konstruktőr próbálkozása sikeresnek tekinthető-e, vagy sem.

Íme a Turing által publikált teszt, mely „imitációs" (utánzási) játékon alapul. Az eredeti játéknak három résztvevője van, akik nem ismerik, nem is látják egymást — távírógépen keresztül „beszélgetnek". A résztvevők közül kettő ún. „tanú" lesz a játékban, mégpedig az egyik tanú férfi, a másik pedig nő. A harmadik játékos a „kérdező", aki csupán a távírógépen keresztül érkező válaszok alapján próbálja meg kitalálni, hogy melyik tanú a nő és melyik a férfi. A játék trükkje az, hogy az egyik tanú (pl. a férfi) megpróbálja félrevezetni a kérdezőt (megpróbál úgy válaszolni, mintha nő válaszolna), míg a másik (a nő) mindent megtesz annak érdekében, hogy segítse a kérdezőt. Ha a kérdező felismeri, hogy melyik tanú a férfi, akkor a játékban a nő nyert, ha nem, akkor a férfi nyert.

Turing ötlete: állítsuk be mondjuk a férfi helyére a tesztelni kívánt számítógépet, és nézzük meg, hogy a gép az átlagos női vetélytársakkal szemben átlagosan legalább annyiszor képes-e félrevezetni a kérdezőt (aki szintén átlagos képességű ember), mint ahányszor erre ugyancsak átlagos férfijátékos lenne képes.

Ha a gép kiállja a fenti próbát, akkor *intelligens*. Azaz akkor azt mondhatjuk, hogy létrehoztuk a mesterséges intelligenciát.

A kísérlet kulcsa: a nyelvi kommunikáció. Úgy beszél-e (pontosabban távírógépen keresztül úgy levelez-e) a gép, mint ahogy az ember? Igaz-e, hogy ez a „beszéd" az intelligencia fokmérője? Turing szerint azért van a nyelvi kommunikációnak olyan különleges szerepe az intelligenciában, mert ez a kérdés-felelet módszer alkalmas az emberi értelem és igyekezet minden oldalának bemutatására: bármiről is akarunk társalogni, akárcsak egy kicsit is mélyebbre hatolva a felszín alá, nem elég csak a szavakat ismernünk, ismernünk kell a témát is. Jó példa erre az alábbi párbeszéd:

Kérdező: Az ön szonettjének első sora így hangzik: „Olyan vagy, mint a nyári nap". Nem lenne a nyári nap helyett ugyanolyan jó, vagy még jobb az, hogy „tavaszi nap"?
Tanú: Ez elrontaná az ütemet, más a szótagszám.
Kérdező: És az, hogy „téli nap"? Így megmaradna az ütem.
Tanú: Az ütem igen, de senki sem szeretné, ha téli naphoz hasonlítanák.
Kérdező: Lehetne azt mondani, hogy a címzett önt a karácsonyra emlékezteti.
Tanú: Bizonyos értelemben lehetne.
Kérdező: Karácsony pedig téli nap, és úgy vélem, a címzett nem bánna ilyen összehasonlítást.
Tanú: Nem hiszem, hogy ön ezt komolyan gondolja. Ha azt írom, „téli nap", akkor ezen általában egy átlagos téli nap értendő, sokkal inkább, mint egy olyan különleges téli nap, mint a karácsony.

Az a vizsgázó (tanú), aki így felel, nemcsak a nyelveket (a magyart, az angolt) ismeri jól, hanem elfogadható ismeretei vannak a költészetről, az évszakokról, az emberi érzelmekről stb. Mindezekről csupán a (például akár távírógépen folytatott) „beszéd" alapján meggyőződhetünk. Választhattunk volna valamilyen más témát is, pl. a politikát, a jó borokat, a filozófiát is stb.

Mi lenne, ha egyszer egy gép kiállná ezt a vizsgát? A Turing által javasolt teszt mindenesetre hatalmas kihívást jelent a kutatóknak és a konstruktőröknek.

2. Intelligencia — problémamegoldó képesség

Elképzelhető-e olyan *általános* problémamegoldó rendszer, amelyik úgy működik, hogy egymás után generálja és ellenőrzi a megoldási javaslatokat egy tetszőleges feladatra, és ha valamelyik javaslat megfelel, akkor sikerrel befejezi a működést,ha nem, akkor kimerülésig folytatja a javaslatok generálását? Válasz helyett nézzük meg, mi lenne, ha így működne egy sakkozógép. A gép valamilyen mattállásból visszafelé indulna el, és megkeresné azt az „utat" (utakat) a nyitóállomásig, amely az ellenfél bármilyen lépése esetén is biztosan győzelemre vezet. Hamar kiderül: ez a módszer teljesen járhatatlan, mert egy-egy állásban a legális lépési lehetőségek átlagos száma 30-40, melyre legalább ennyi legális válaszlépés lehetséges átlagosan. Ezért egy lépés-válaszlépés pár így legalább mintegy 1000 lehetőséget tartalmaz. Mintegy 40 lépéspárból álló átlagos játszma esetén ez 10^{120} darab lépés generálását és ellenőrzését/kiértékelését igényelné. A feladat ilyen megoldása nyilvánvalóan lehetetlen: ennyi megkülönböztető állapota fennállása óta nem volt az egész világegyetemnek! Ezt a problémát, amelybe így ütköztünk, *kombinatorikus robbanásnak* is szokás nevezni.

A mesterséges intelligenciával foglalkozók egyik fő törekvése ezeknek a problémáknak valamilyen módon való kiküszöbölése. A teljes körű keresés helyett meg kell találni a szelektív (válogató) keresés célszerű módját, amely persze egyrészt részleges, másrészt kockázatos lesz, mert nem tudhatjuk teljes bizonyossággal, hogy a keresett legcélszerűbb megoldás nem éppen a nem vizsgált lehetőségek között van-e.

Mára eljutottunk odáig, hogy tudjuk, a korábbi évtizedek nagy álma — az „általános problémamegoldó" (GPS — General Problem Solver) feltalálása, kifejlesztése — szertefoszlott, ahogy az általános nyelvi (mondjuk angolról németre) fordító gépé is, melyet egy memorandumban Weaver javasolt még 1950-ben. (Valószínűleg hasonló sorsra jut, mint az örökmozgó, de érdemes elgondolkozni azon, milyen soká kereste az emberiség az örökmozgó problémáját.) Helyettük fokozatosan megjelennek (lassan kereskedelmi forgalomban is) az egy-egy szűk területre specializálódó *szakértői rendszerek* (Expert Systems). A szakértői rendszereket úgy tekinthetjük, mint a mesterséges intelligenciával foglalkozó tudományos kutatások alkalmazott termékeit (v. ö. a Mesterséges értelem sorozat II. részével).

A jövő fogja eldönteni, hogy a szakértői rendszerek valóban olyan jelentős hozzájárulást jelentenek-e majd a gondolkodás, az intelligencia, a tanulás lényegének megértéséhez vezető úton, ahogy azt most hisszük.

3. Neumann János alapvető felismerése

Az eddig bemutatott gondolatok inkább a számítástechnika, az emberi megismerés jövőjét érintették, most pillantsunk vissza a múltra. A mai számítógépek döntő többsége felépítését, architektúráját tekintve ún. Neumann-féle gép. „Történelmi" ismeretek és megfelelő összehasonlítás hiányában ma már sokan nem is tudják, mi volt a Neumann János nevéhez fűződő forradalmi újítás e szá-

egy jó könyvet (John Haugeland, „Artificial Intelligence: The Very Idea", Bredford Book, The MIT Press, 1986. második kiadás). Most ebből szemeltünk ki néhány érdekességet a mesterséges intelligenciáról, az általános problémamegoldóról, a veremtárakról és arról, mint jelent az a szó, hogy „digitális".

zad negyvenes évtizedének végén. (Az 1987/4. számbeli megemlékezésünk sem tér ki erre.)

A Neumann-architektúrák jellemzője, hogy a memória egy-egy helyét (címét) két különböző módon is el lehet érni (meg lehet találni, írni és/vagy olvasni lehet az adott címre/címről): relatív cím, illetőleg abszolút cím alapján.

A relatív címzés olyasmi, mint amikor egy idegent igazítunk el az utcán: „menjen tovább két utcasarokkal, majd forduljon balra stb." A postás ezzel szemben abszolút címek alapján kézbesíti a leveleket, mondjuk Budapest, Sas utca 7-be.

A Neumann-architektúrájú gépek memóriájának másik fontos tulajdonsága, hogy kétféle feladatot lát el:

— tárolja az adatokat és

— tárolja a programokat is.

Ennek az újításnak a korszakalkotó jelentőségét elsősorban az adta, hogy lehetővé vált az ún. *szubrutinok* alkalmazása. A szubrutin — tudjuk — nem más, mint egy „konzervprogram", melyet „levehetünk a polcról": egy másik program bárhol, egyetlen lépésként, bármikor meghívhatja. Egy gyakran ismétlődő feladat megoldását elég így csak egyszer programozni. Valahányszor a fő programban ehhez a részfeladathoz érünk, elugrunk a szubrutinra, és a szubrutin végén visszaugrunk a hívás helyére (közvetlenül a hívóutasítás utáni utasításra).

A lelemény éppen a két *ugrásban* van — különösen a másodikban — mivel a szubrutin mindig ugyanott helyezkedik el, hívása viszont a legkülönbözőbb helyekről történhet. A visszaugrás célpontja emiatt változik. Honnan fogja tudni a program, hova kell visszaugrania? Neumann János kortársainak ez egyáltalán nem volt triviális. Találmánya — az adatok és a programok együttes kezelése a tárban — megoldotta ezt a problémát: az ún. *visszatérési címet* egy erre a célra kijelölt memóriarekeszben (címen) lehet tárolni.

A korábbi gépekkel *nem* lehetett megvalósítani ilyesmit. Részben azért, mert ezek nem tudtak abszolút címekkel dolgozni (pl. Babbage Analytical Engine-je sem tudott), részben, mert a programot (a visszatérési címét) nem lehetett módosítani, mert pl. a program lyukkártyán volt rögzítve. De még az olyan korai elektronikus gép, mint az ENIC is hasonlóan működött e vonatkozásban, mint Babbage (1872—1971) korai mechanikus gépe.

A szubrutinból való visszatérés kérdése bonyolultabb, mint elsőre látszik; mi van akkor, ha egy szubrutin egy másik, további szubrutint hív és így tovább. Világos, hogy ilyenkor nem elég *egy* visszatérési címet tárolni — annyit kell, ahány szubrutint hívnak „egymás hasából" (amekkora az ún. „hívásmélység"). Neumann János idejében az volt a gyakorlat, hogy minden egyes szubrutinhoz kijelöltek egy-egy memóriarekeszt, ahol az adott szubrutinból való visszatérés címét tárolták. De ez sem volt igazán jó módszer, mert nem tette lehetővé, hogy egy-egy szubrutint többször mint egyszer meghívjanak, így nem tette lehetővé azt sem, hogy egy-egy szubrutin hívhassa saját magát is, pedig ez az ún. rekurzív megoldásokhoz igen fontos lett volna.

Az általános megoldást az ötvenes évek közepén találta meg Alan Newell és Clif Shaw. Ez az ún „veremtár", angol nevén a „stack". Ez így működik: valahányszor a program hív egy szubrutint, a visszatérési címét elhelyezzük a *verem* tetejére, az alatta lévő cím pedig eggyel „lesüllyed". A szubrutin befejeztével a visszatérési címet mindig a verem tetejéről „emeljük le". A leemelés után a következő visszatérési cím automatikusan a verem tetejére kerül. A veremtár realizálásának egy módját az ábrán látható kis programocska mutatja.

A szubrutinok jelentősége nemcsak a helymegtakarításban van, hanem abban is, hogy lehetővé teszik a programok moduláris felépítését is. Ha van egy működő, kipróbált szubrutinkészletünk, akkor ezekből, mint félkész elemekből építkezhetünk.

Lényegében a szubrutin-elvnek köszönhetjük az ún. *magas szintű gépek* (nyelvek) létrehozását is. Az általános, gyakran használt szubrutinokat automatizált *könyvtárban* gyűjthetjük össze, és programunkat kizárólag e könyvtárban tárolt szubrutinok hívását jelentő magas szintű utasításokból állíthatjuk össze. Az így létrejövő magas szintű (Pascal, BASIC, COBOL stb.) gépek is lényegében Neumann-elvűek — annyira, hogy szinte el is feledkezhetünk erről a tényről.

Napjainkban kezdenek elterjedni az újabb — a Neumann-architektúrájúaktól lényegükben különböző — magas szintű architektúrájú LISP-, PROLOG- stb. gépek, melyek mind a tár szervezésében, mind a vezérlésben különböznek a ma megszokott számítógéptől.

Érdemes megjegyezni: attól, hogy magas szinten (a LISP, a PROLOG stb. nyelvek szintjén) egy gép architektúrája eltér a Neumann-gépek architektúrájától, még nem következik szükségszerűen, hogy a *„csupasz gép"* („amin az egész fut") is új architektúrájú legyen. Ma nagyon gyakori eset, hogy hagyományos gépeken valósítsanak meg PROLOG-, LISP- stb. architektúrákat. Sőt — vannak, akik azt hirdetik, hogy a Neumann-gépek általános elterjedése és emiatt — viszonylag alacsony ára miatt ez a gazdaságos út.

4. Digitális technika

A digitális technika ma annyira elterjedt, annyira körülvesz bennünket, hogy nem szoktunk azon gondolkodni: mi is a lényege. Annyit majdnem mindenki tud, hogy ha valami digitális, akkor az számjegyes működésű.

A digitális fogalmi ellentétpárja az analóg. Otthoni személyi számítógépeink digitálisak, a falon lógó, örökölt ingaóra számlapos, mutatós kijelzőjére pedig ma azt mondjuk: analóg. Mitől függ, hogy valamely technika (művelet, eljárás, eszköz) digitális-e, vagy sem?

Kezdjük azzal, hogy felosztjuk az eljárásokat (a technikákat) két nagy csoportra:

— az elsőbe soroljuk azokat, melyek mindig *teljesen* pontos, egyértelmű eredményre vezethetnek,

— a másodikba azokat, amelyek legfeljebb csak *közelítően* pontos eredményre vezethetnek.

Nevezzük az első csoportba tartozó eljárásokat *pozitív* eljárásoknak. Mitől függ, hogy egy eljárás melyik csoportba tartozik? Elsősorban attól, hogy mit tekintünk *teljesen* pontos eredménynek, hogyan *definiáljuk* a maradéktalan sikert.

Lássunk néhány példát! Legyen a feladat pl. egy hosszú szál deszkából 2 méter hosszú darab lefűrészelése. Az eljárás: vesszük az 1 méteres collstokkot, a deszka egyik végére illesztjük, ceruzával bejelöljük a méteres szakasz végét, majd e ceruzajelhez illesztjük ismét a collstokk egyik végét, és a másik végénél asztalosderékszög segítségével a deszka oldalára merőlegesen berajzoljuk a fűrészelés helyét. Pozitív-e ez a technika? Ez attól függ, hogyan definiáljuk a művelet sikerét, mi az átvételi követelmény. Amennyiben azt mondjuk, hogy a leszabás akkor lesz sikeres, ha a leszabott deszkadarab hossza 1,97 m és 2,03 m között van, akkor ez a feladat szinte

```
PROGRAM verem-tár (input, output),
CONST N=a verem-tár mérete,
VAR nemüres, nincstele: BOOLEAN, n: O..N,
      verem: ARRAY O..NN—1 OF INTEGER,
      x,y: INTEGER,
.
.
PROCEDURE elvermel,
BEGIN
IF n N THEN BEGIN
verem n :=x, n :=n+1, nincstele:= n N, nemü-
res:=TRUE,
END
END,
PROCEDURE kivesz,
BEGIN
IF n O THEN BEGIN
n:=n—1, x:=verem n, nemüres:= n O, nincstele:=TRUE,
END
END,
.
.
BEGIN (A főprogram kezdete)
.
.
n:=O, nemüres:=FALSE, nincstele:=TRUE,
.
.
IF nincstele THEN BEGIN
X :=az „elvermelendő" érték,
elvermel (procedúra-hívás) END,
IF nemüres THEN BEGIN
kivesz, (procedúra-hívás)
yy:=x END,
.
.
END. (A főprogram vége)
```

A veremtár egy lehetséges megvalósítása Pascal nyelvű programban. A verem maximum N darab tételt — pl. címet, mely mindig egész szám (INTEGER) — képes fogadni.

bárki által *tökéletesen* sikeresen végrehajtható. De ha valaki ragaszkodna *pontosan* 2 méteres hosszúsághoz, akkor a feladat *végrehajthatatlan* lenne, illetőleg csak *közelítő* pontossággal lenne végrehajtható. Pontosan 2 m hosszú deszka leszabása tehát *nem* minősül pozitív eljárásnak.

Az, hogy a technika pozitív, még nem garancia arra, hogy egy konkrét esetben feltétlenül sikeres is. A pozitív minősítés csak a *tökéletes* végrehajtás *lehetőségét* jelzi. Arra vonatkozóan, hogy mekkora a valószínűsége annak, hogy egy pozitív eljárást ténylegesen sikeresen végre is hajtanak, egy másik fogalmunk van, a technika *megbízhatósága*.

Ha a deszkát pl. kézifűrésszel 2 m + 1 mm pontossággal kívánnánk méretre szabni, akkor ez a fűrészelési technika még pozitívnak lenne tekinthető, de már nem lenne *megbízhatóan* végrehajtható (a mérce, a derékszög pontatlansága, a felület szálkássága stb. miatt). $2\ 10^{-6}$ m követelmény esetén a technika már sem nem pozitív, sem nem megbízható, mert deszkát *elvileg* nem lehet vágni ilyen pontossággal.

Sok pozitív és egyben megbízható technikát ismerünk. A kosárra dobás (különösen közelről) pl. egy jó kosárlabda-játékos számára pozitív és megbízható is. Egy dobozban lévő ceruzák megszámlálása is pozitív, ilyen egy villanykapcsoló bekapcsolás/kikapcsolás is. Ezekkel szemben pl. egy kemence felfűtése pontosan 680 °C-ra nem pozitív, mert a legjobb szabályozóval is csak *közelítően* pontosan tudjuk beállítani ezt az értéket. Nincs technikánk arra sem, hogy egy doboz ceruza súlyát *teljes pontossággal* megállapítsuk. Ez tehát nem pozitív eljárás.

Digitális technika esetén szimbólumokat *„irunk"* és *„olvasunk"*. Íráson azt értjük, hogy egy előre meghatározott szimbólum (jel)-készletből egyet előállítunk. Olvasásnál egy szimbólumról (egy jelről) állapítjuk meg, hogy ez melyik az adott készletből. Az írás/olvasás ciklus akkor sikeres, ha ugyanazt a szimbólumot ismerjük fel az olvasásnál, amit az írás során előállítottunk.

Digitális technikának ezek után az olyan írási/olvasási technikák összességét (halmazát) tekintjük, melyek pozitívak és megbízhatóak.

A digitális technika jelentősége abban van, hogy lehetővé teszi igen nagy és bonyolult rendszerek megbízható felépítését valamilyen fokon *tökéletlen* elemekből is. Gondoljunk egy pillanatra Rembrandt portréinak és Shakespeare szonettjeinek eltérő sorsára. A festmények a leggondosabb kezelés mellett is fokozatosan kifakulnak, megfeketednek, elvesztik eredeti varázsukat. A költeményeket ezzel szemben sikerül *tökéletesen* megőrizni akár hosszú évszázadokon át is, hacsak nem vesztik el ők teljesen, vagy nem másolják őket hibásan. A legtöbb megőrzött, régi költemény nagy valószínűséggel pontosan olyan, amilyennek annak idején megírták.

A természetben mindig van egy kis változás, „hiba". Nincs két egyforma „A" betű. „Semmi sem tökéletes". A digitális rendszerekben ennek ellenére megvan a képesség a tökéletes működésre. Ennek magyarázata — emlékezzünk vissza példánkra — az, hogy jól megválasztott *hibahatárok* figyelembevételével építjük fel: a tervezők előre számoltak a hibákkal, az ingadozásokkal.

Bár két „A" betű soha nem egyforma, a gyakorlatban kialakult hibahatárok között mégsem szoktuk eltéveszteni őket. Bonyolult, nagy rendszerek megbízható felépítését az teszi lehetővé, hogy a digitális technika megakadályozza, hogy a rendszerek, az elemek szükségszerű hibái összegződve megzavarják az *egész* működést. Mivel a mesterséges intelligencia megvalósítása minden valószínűség szerint igen bonyolult módszert fog igényelni, ezért szinte biztos, hogy megvalósulása digitális technikán alapul majd.

Befejezésül: a digitális technika nem kötődik egyetlen konkrét megvalósítási közeghez sem. A közeg lehet akár elektronikus, akár mechanikus, akár hidraulikus, vagy valamilyen egyéb, ma még nem ismeret közeg is.

—KE—

A játékfa és kiértékelése

● *Amint legutóbb említettem, a sakkprogramozás alapvető ismereteinek könnyebb elsajátítása érdekében megjelent a* Sakkprogramozásról mindenkinek *című könyvem. Érdekesebb pontjait e lap hasábjain igyekszem saját kutatásaim alapján bővebben kifejteni.*

Definíciók

A játékfa 0. csomópontjának nevezzük azt az állást, ahonnan kiindulva a lehetséges lépések gráfját felépítjük, vagyis a gráf gyökerét.

A *mélység* (depth) megadja, hogy a 0. csomóponttól számolva hányadik szinten vagyunk, vagyis a gyökértől kezdve hány féllépést tettünk meg „gondolatban" az illető állásig.

A *szélesség* (width) megadja, hogy az illető levélen hány pszeudolegális lépés van.

A *pozíció* olyan számsorozat, amellyel egyértelműen meg lehet határozni egy adott állást, vagyis az állás valamilyen kódformátumban való rögzítése, meghatározása (például a Mikroszámítógép Magazin 1986. februári és márciusi számában megadott módon).

A *terminális pozíció* olyan állás, ahol az értékelés történik. Ha fix féllépéses mélységnél megy végbe az értékelés, akkor az illető mélység elérésénél létrejött állások a terminális pozíciók. Ha azon a szinten már nincs legális lépés, mert előtte matt vagy patt állás jött létre, a terminális pozíció természetesen kisebb mélységben van. A jobb programok viszont ezt sokkal bonyolultabban határozzák meg; külön algoritmusuk van annak eldöntésére, hogy a mélység növelésével létrejövő állást terminálisnak minősítsék-e vagy sem. Ezek az algoritmusok megvizsgálják az állást, hogy mennyire mozgékony, illetve nyugodt, és ennek függvényében döntenek: értékelni kell-e vagy tovább kell növelni a lépésmélységet.

A *pontérték* az értékelőfüggvény által meghatározott érték, amelyre az állást minősítette. Nagysága általában az illető számítógép által ábrázolható legnagyobb és legkisebb szám közé esik.

1. lista

```
Negamax( p: pozicio; melyseg: integer) :integer
var
     ertek,sz,t,m : integer;
     lepes : array[1..MAX_SZELESSEG]

begin
    if melyseg = 0 then  { terminalis csomopont? }
        return( Pontertek(p));

    sz := Lepesgenerator(lepesek);
    if sz = 0 then       { nincs legalis lepes? }
        return( Pontertek(p));

    { Megkeresi a maximalis erteket a faban }
    ertek := -oo;
    for m :=1 to sz do
        begin
            t := -Negamax( p.lepes[m], melyseg-1);
            if t > ertek then
                ertek := t;
        end;

    return( ertek );
end.
```

```
AlfaBeta(p: pozicio; alfa,beta,melyseg: integer): integer
var
    ertek,sz,t,m : integer;
    lepesek : array[1..MAX_SZELESSEG] of integer;

begin
    if melyseg = 0 then  { terminalis csomopont? }
        return( Pontertek(p) );
    sz := Lepesgenerator( lepesek );
    if sz = 0 then       { nincs legalis lepes? }
        return( Pontertek(p) );

    ertek := -oo;
    for m := 1 to sz do
        begin
            t := -AlfaBeta( p.lepes[m], -beta,-alfa,melyseg-1 );

            if t > ertek then
                ertek := t;

            if ertek >= beta then
                { Levagja a fa tobbi reszet }
                return( ertek );

            alfa := Max( alfa,ertek );
        end;

    return( ertek );
end.
```

2. lista

Lépésgenerátornak nevezzük azt a szubrutint, amely az illető állásban megtehető félig legális lépéseket (lásd a lap 1986. áprilisi számában) listába fűzi. A következő példánknál a lépésgenerátor csak a valóban legális lépéseket teszi a listára, ugyanis a program áttekinthetőségét és így megértését nehezítené, ha minden részletet a működő programoknak megfelelően építettem volna fel. Általában a féllépés növelésénél iktatják közbe a lépéslegalitás ellenőrzését, és így választják le a programok a félig legális lépések listájáról az illegális lépéseket. Ennek jelentőségét a későbbiekben látjuk majd: az alfa-béta algoritmusnál és a program gyorsításának módszereinél. Ugyanis teljesen felesleges olyan állásokban is megvizsgálni a sakkot, amelyek az esetleges levágások miatt létre sem jönnek. Ezzel rengeteg időt veszítenénk. Képzeljük el, hogy minden megvizsgált lépésre (amely alig egy-két művelet) a királytól kiindulva mind a nyolc irányban végig és a huszártámadások irányában is meg kellene vizsgálni az állást, hogy az illető lépés megtétele miatt nem támadja-e ellenséges figura a királyt. Tehát egy-két művelet helyett egy-két nagyságrenddel nagyobb műveletsort kellene végrehajtani, ami, tekintettel az exponenciálisan terebélyesedő fára, jelentős időveszteséget okozna.

Negamax értékelés és alfa-béta algoritmus

A negamax eljárás attól különbözik a minimaxtól, hogy az állás pontértékét mindig minden szinten a lépésen levő fél szemszögéből nézzük, vagyis minél nagyobb ez a pont, annál jobban áll az, aki lép, és minél kisebb a pont, annál roszszabbul.

A Neumann János által kidolgozott minimax elmélet viszont úgy működik, hogy a féllépésszinttől függetlenül, minél nagyobb egy állás pontértéke, annál jobban áll a 0. féllépésszinten lépésre következő fél, és

minél kisebb, annál roszszabb az állása. Ezért páratlan szinteken a maximális, páros szinteken pedig a minimális pontértéket választja az algoritmus.

A negamax eljárásnál nem kell páratlan vagy páros féllépésszinttől függően kettéválasztani ezt a két esetet, hanem mindig csak a maximumokat kell kiválasztani. Ezért egységesebb, rövidebb és tömörebb programmal megoldható az értékelés. Az egységesítés lehetővé teszi a rekurzió használatát, ami elegánssá is teszi a programot. Igaz, bizonyos programnyelvekben (például a FORTRAN-ban) nem alkalmazhatunk rekurziót, de ezekben átírhatjuk ezt a programrészletet egyszerű ciklussá.

A következőkben bemutatok egy példát arra a módszerre, ahogyan kidolgoztam a negamax és az alfa-béta eljárás pszeudokódját. Ez a pszeudokód a Pascal programnyelvhez áll a legközelebb, de tartalmaz olyan karaktereket, amelyek használatát a nyelv szintaktikailag nem engedi meg (*1. lista*).

Tudjuk, hogy a minimax algoritmus a játékfa összes csomópontját átvizsgálja és egyben értékeli is. Ismeretes, hogy már a kutatások korai stádiumában továbbfejlesztették ezt az elméletet és kidolgozták az alfa-béta eljárást, amely a minimax értékeléssel azonos eredményre vezet, de a fa nagy részénél nem értékeli, nem pontozza a pozíciót. Kiderült, hogy a fa nagy részét felesleges értékelni, és így annak pontozása elhagyható, ami jelentős időmegtakarítást tesz lehetővé. Az elhagyott lépéseket levágásnak nevezzük. Az előbb említett időmegtakarítással elérhetjük, hogy a program mélyebben számoljon, és ezzel hosszabb kombinációkat is észrevegyen.

A *2. lista* ennek a megvalósítását mutatja. A szubrutin első meghívásakor a paraméterlistán szereplő alfának és bétának $-\infty$ és $+\infty$, illetve az adott számítógéppel ábrázolható legkisebb és legnagyobb szám az értéke. (Folytatjuk.)

KOVÁCS P. ATTILA

Aprócska játékos trükkök, melyek szórakoztatnak, és egy picit a matematikai ismereteket is felelevenítik.

Petik

Petikét kifejezetten dühítette, hogy április 4. és november 7. szombatra esik. Úgy érezte, elvesznek tőle két potya szünnapot, amelyre joggal számított.

Vajon milyen napra esnek a jövőben ezek az ünnepek? Ez is indokolt kíváncsiság, ahogy az is: bizonyos események milyen napon történtek. És az sem érdektelen, hogy ő — mármint ő-Petikesége — meg a szülei milyen napon születtek.

Munkához látott tehát, hogy olyan naptárt készítsen, amelyből bármelyik dátumról megtudhatja, hogy milyen napra esett vagy fog esni.

A hét egyes napjaihoz hozzárendelte a következő számokat: a vasárnaphoz az 1-et, a hétfőhöz a 2-t, . . . a szombathoz a 7-et. Az 1987. évi naptárban megnézte, hogy január 1. csütörtökre, a hét 5. napjára esik. Ebből már következik, hogy február 1. hárommal későbbi napon, azaz vasárnap lesz. Mivel február 28 napos, ezért március 1. szintén vasárnap. A hónapok eleje ezután szintén későbbi napra tolódik. Felírta 1987-ben a hónapok első napjaihoz tartozó napokat:

január 1. csütörtök (5)
február 1. vasárnap (1)
március 1. vasárnap (1)
április 1. szerda (4)
május 1. péntek (6)
június 1. hétfő (2)
július 1. szerda (4)
augusztus 1. szombat (7)
szeptember 1. kedd (3)
október 1. csütörtök (5)
november 1. vasárnap (1)
december 1. kedd (3)

A következő év január 1-je egy nappal tolódik előre 1987-hez képest, tehát 1988. január 1. a hét hatodik napjára esik. Amennyiben az év nem szökőév, ez minden hónap első napjára is igaz. 1986 hónapjainak első napjai pedig egy nappal visszafelé tolódnak. A szökőévben a február 29 napos. Így márciustól kezdve a hét napjainak sorszáma nem eggyel, hanem kettővel tolódik előre. Ebből következik, hogy a szökőévet követő év első napja kettővel követi az előző évit.

Petike, miután ezt végiggondolta, táblázat formájában felírta az összefüggést az évek hónapjainak első napjai között (*1. ábra*).

1988 — mint minden néggyel osztható év — szökőév; itt a képzési szabály megváltozik, amint az Petike táblázatában jól látszik. A táblázatot folytatva meg lehetne mondani bármelyik dátumról, hogy az a hét melyik napjára esik, illetve fog esni. Ha nem lennének szökőévek, akkor hét év múlva ugyanaz a naptár újra használható lenne, de a szökőévek miatt négyszer hét, vagyis 28 év a naptár periódusa. Az évszázadok közül csak azok szökőévek amelyekben a százasok száma is osztható néggyel. Így az 1800. és 1900. év nem szökőév, de például a 2000. év igen. A periódus tehát csak századon belül igaz. Ezzel nem lehet sokat kezdeni!

Miután Petike rájött, hogy milyen szabály szerint határozhatók meg az egymást követő évek első napjai, újabb táblázatot készített. Ebből az egymást követő hónapok első napjainak különbségét tudta leolvasni szökőév és nem szökőév esetén (*2. ábra*). Majd tovább gondolkodott: ha ki tudom számítani, hogy egy adott év első napja hányadik a héten, akkor — mivel ismerem a hónapokra vontakozó képzési szabályt — nyert ügyem van. Csak azt kell még megvizsgálnom, hogy milyen típusú az év.

Kiindulásnak 1988-at választotta, mert az szökőév, és így könnyebb rá felírni az algoritmust. Visszafelé számolva az évek első napjai mindig eggyel esnek előbbre, kivéve minden negyedik évet. Tehát vette az 1988. év és az őt érdeklő dátum közötti különbséget, és ehhez hozzáadott annyit, ahány szökőév volt a két dátum között, vagyis ahányszor a négy megvan azok különbségében. Ha például 1945 első napjára kíváncsi, akkor a következőképpen kell számolnia.

$DV = 1988 - 1945 = 43$

Negyvenháromban a négy tízszer van meg, tehát 1988 első napjából (azaz 6-ból) 53 napot kell levonni, hogy megkapja az 1945. év első napjának sorszámát. Igen ám, de hogyan vonjon le hatból (1988. év első napja a héten a hatodik) 53-at, hiszen negatív sorszám nincs.

A Petike jelölése sok esetben hasonlít a nyolcas számrendszerre, de mégsem az. A számrendszerek ugyanis negatív és pozitív irányban végtelenek, továbbá a nullát mint számjegyet tartalmazzák. Petike jelölése egytől hétig terjedő periodikus számsor, ahol például ha egyből elveszünk egyet, akkor hét lesz az eredmény (mert vasárnap előtt szombat van).

Petike tudta, hogy csak ismétlődést nem tartalmazó értéket kell hatból kivonnia. Ezért az 53-at annyival csökkentette, hogy ne legyen benne a periodikus ismétlődés, tehát lefelé kerekítette héttel osztható számmá, és ezt levonta belőle. Így 1945. évre.

$6 - (53 - 49) = 2$

eredményt kapta, azaz január elseje a hét második napjára esett, vagyis hétfőre. Változókkal felírva a programban:

$ISZ = DV + INT(DV/4)$
$H(1) = 6 - (ISZ - 7*INT(ISZ/7))$

ahol DV a két év különbsége, ISZ a levonandó összeg, H(1) a keresett nap sorszáma. Mivel a jelölésrendszernél Petike nem használt nullát, ezért értelemszerűen $H(1) = 0$ 7-nek felel meg.

Ez a számítási módszer azonban csak egy évszázadon belül igaz, mert Petike az ISZ változó kiszámításánál minden néggyel osztható évet, így 1900-at is szökőévnek tekintette. A módszer kiterjesztéséhez ezért le kell vonnia azon századfordulók számát, amelyek nem szökőévek. Ezt Petike I változóval jelölte, és felírta az általános eljárást:

$ISZ = DV + INT(DV/4) - I$

Az előrefelé számításnál hasonló meggondolásból indult ki. Programját ebben az esetben kicsit módosítania kellett, mert elő-

röknaptárt készít

```
1 REM OROKNAPTAR
10 DIM H(12),N$(7)
20 DATA VASARNAP,HETFO,KEDD,SZERDA
21 READ N$(1),N$(2),N$(3),N$(4)
22 DATA CSUTORTOK,PENTEK,SZOMBAT
23 READ N$(5),N$(6),N$(7)
29 REM DATUM BEOLVASASA
30 INPUT "EV?",EV
31 IF 1582<EV<=5000 GOTO 35
32 PRINT "NEM ERVENYES IDOSZAK"
33 GOTO 30
35 INPUT "HO?",H
40 INPUT "NAP?",N
45 IF EV=1988 THEN H(1)=6:GOTO 500
50 DV=1988-EV
60 IF DV<=0 GOTO 200
70 I=INT((DV+12)/100)
71 REM EVSZAZAD KISZAMITASA
80 ISZ=DV+INT(DV/4)-I
90 H(1)=6-(ISZ-7*INT(ISZ/7))
91 REM AZ EV ELSO NAPJA
100 IF H(1)=0 THEN H(1)=7
110 GOTO 400
200 DV=ABS(DV)
205 T=INT((EV+12)/100)
210 REM SZOKOEV SZAMAINAK MEGHATAROZASA
220 X=T+20 : I=0
225 IF T=20 GOTO 290
230 FOR J=20 TO T
240 X1=X/4
250 X1=X1-INT(X1)
260 IF X1<0!0001 GOTO 280
270 I=I+1
280 NEXT J
290 ISZ=DV+INT((DV-1)/4)+1-I
300 H(1)=(6+ISZ)-7*INT((6+ISZ)/7)
310 IF H(1)=0 THEN H(1)=7
400 REM HONAPOK ELSO NAPJA
410 REM SZOKOEV-E?
420 E1=EV/100
430 E1=E1-INT(E1)
431 REM SZAZADFORDULO-E?
432 IF E1=0 GOTO 436
435 E1=E1*100 +.0001
436 E1=E1/4
437 E1=E1-INT(E1)
440 IF E1>.0001 GOTO 550
500  EM SZOKOEV !
510 H(2)=H(1)+3 :IF H(2)>7 THEN H(2)=H(2)-7
512 H(3)=H(2)+1 :IF H(3)>7 THEN H(3)=H(3)-7
520 GOTO 560
550 REM NEM SZOKOEV !
555 H(2)=H(1)+3 :IF H(2)>7 THEN H(2)=H(2)-7
556 H(3)=H(2)
560 H(4)=H(3)+3:IF H(4)>7 THEN H(4)=H(4)-7
561 H(5)=H(4)+2:IF H(5)>7 THEN H(5)=H(5)-7
562 H(6)=H(5)+3:IF H(6)>7 THEN H(6)=H(6)-7
565 IF H<6 GOTO 600
566 H(7)=H(6)+2:IF H(7)>7 THEN H(7)=H(7)-7
567 H(8)=H(7)+3:IF H(8)>7 THEN H(8)=H(8)-7
568 H(9)=H(8)+3:IF H(9)>7 THEN H(9)=H(9)-7
569 H(10)=H(9)+2:IF H(10)>7 THEN H(10)=H(10)-
570 H(11)=H(10)+3:IF H(11)>7 THEN H(11)=H(11)
575 H(12)=H(11)+2:IF H(12)>7 THEN H(12)=H(12)
600 REM A NAP MEGHATAROZASA
610 K=H(H)+N-1
620 K=K-7*INT(K/7)
630 PRINT EV;".EV",H;".HO",N;"NAP",N$(K)
640 INPUT "TOVABB? I/N",V$
650 IF V$ <>"N" THEN GOTO 30
660 END
```

re haladva a szökőévek száma eggyel több. Ezért ISZ-hez egyet hozzá kell adni, az eltolódás miatt a különbséget a néggyel való osztásnál pedig eggyel csökkenteni kell. Most is le kell azonban vonni a nem szökő századfordulók számát. A periódusok figyelembevétele is másképpen alakul egy kicsit, mivel az induló hónaphoz hozzá kell adni valamennyit.

Hogy az év első napja hányadik a héten, azt az alábbi programsorok számolják ki:

ISZ = DV + INT((DV − 1)/4) + 1 − I

H(1) = (6 + ISZ) − 7*INT((6 + ISZ)/7)

Ebben az esetben is H(1) = 0 a hét hetedik napjának felel meg.

Petike táblázat nélkül kiszámolta, hogy melyik napra esett 1948. március 15. Mivel 1900 nem szökőév, ezért I = 1. Tehát

ISZ = 140 + 35 − 1 = 174

H(1) = 6 − (174 − 168) = 0 azaz 7

vagyis 1948. január elseje szombatra esett. De milyen nap volt a többi hónap első napja? A szökőévre vonatkozó képzési szabály szerint:

H(1) = 7

H(2) = H(1)3 = 7

H(3) = H(2) + 1 = 4

1948. március a hét negyedik napján, vagyis szerdán kezdődött. Ehhez kell 14 napot hozzáadni (nem 15-öt, mert akkor az első napot kétszer vennénk figyelembe) és a periódusok számát kivonni. Ha K-val jelöljük:

K = 4 + 14 − 14 = 4,

azaz a nevezetes nap szerdára esett.

Petikének nem volt kenyere a sok számolás; írt tehát egy olyan programot, amely a Gergely-féle naptár érvényességi határán belül (1582—5000) bármelyik dátumra kiszámítja, hogy milyen napra esik.

A program 60-as sora megvizsgálja, hogy visszafelé vagy előre kell-e számolni. A 70-es sor az évszázadok számát állapítja meg. Visszafelé számolásnál nem kell vizsgálni, hogy a századfordulók szökőévek-e, mert a naptár érvényességi idején belül nem volt ilyen; előrefelé viszont a program 230—280-as sora vizsgálja és számolja ezt. A többi szökőévet, vagyis amelyek utolsó két számjegye osztható néggyel, a program 410—440-es sora keresi meg.

A program MS BASIC nyelven készült PC-re, de könnyen átírható más gépre, mivel csak alaputasításokat tartalmaz.

PINKE GYÖRGY

év/hó	I.	II.	III.	IV.	V.	VI.	VII.	VIII.	IX.	X.	XI.	XII.
1985	3	6	6	2	4	7	2	5	1	3	6	1
1986	4	7	7	3	5	1	3	6	2	4	7	2
1987	5	1	1	4	6	2	4	7	3	5	1	3
1988	6	2	3	6	1	4	6	2	5	7	3	5
1989	1	4	4	7	2	5	7	3	6	1	4	6

1. ábra

2. ábra

év/hó	I.	II.	III.	IV.	V.	VI.	VII.	VIII.	IX.	X.	XI.	XII.
Nem szökő	-	3	0	3	2	3	2	3	3	2	3	2
Szökő	-	3	1	3	2	3	2	3	3	2	3	2

● Néhány levélre késve válaszolok. Részben a μ'87-tel kapcsolatos teendők, részben pedig külföldi tanulmányutam miatt az elmúlt néhány számban az olvasói levelekre a szerkesztőség tagjai válaszoltak, ezúton is köszönöm a segítségüket.

Virág István tanár, Szolnok,

Mikszáth K. u. 17. III. 74. 5000

Egy másik levél — egy másik kórházból. A furcsa címet azért választottam, mert 10 perce olvastam a „Levél a kórházból" című írását. Nagyon tetszett, a leírtakkal egyetértek. Kár, hogy ma még kevesen olvassák a lapot.

Sajnos én is egy kórteremből írok, Szolnokról, a Hetényi Géza Kórházból.

Másrészt örömömben is írok, mert tegnapelőtt vehettem át egy harmadik díjat a mellékelten elküldött pályamunkámért. (A címe: A SZÁMÍTÁSTECHNIKA BEVEZETÉSE A SZOLNOKI ABONYI ÚTI ÁLT. ISKOLÁBAN).

Amióta ezt megírtam, már sok minden történt. Többek között pályáztunk, ill. kiállítottunk a MIKRO'87-en a 25-ös pavilonban, számítógépes játékdélutánokat tartottunk.

A harmadik díjhoz gratulálunk. Ha már a μ'87 szóba került, tudatom, hogy megkezdtük a μ'88 szervezését, amelyet szeretnénk még nagyobb társadalmi közreműködéssel, még sikeresebben megrendezni. Itt hívom fel az „informatika szerelmeseinek" a figyelmét, hogy szívesen fogadunk minden jó ötletet, javaslatot, ami az Országos Mikroszámítógépes Találkozót érdekesebbé, tartalmasabbá teszi. Az ötleteken kívül várjuk azoknak a jelentkezését is, akik szívesen vállalják valamilyen eseménynek az előkészítését és a megrendezését is. A μ'88 rendezőbizottsága nyitott, mindig szükségünk van új és aktív tagokra.

Miszlai Imre, Orosháza,

Hajnal u. 10. f. 1. 5900

Videopac+G 7400-as gépem van. Érdekelne, hogy programozható-e, és ha igen, akkor milyen nyelven.

A másik kérdésem, hogy eredeti gyári kazettát hol lehet kapni az országban, s mennyi darabja.

S végül szeretném, ha megírná, hogy ehhez a géphez hol lehet kapni műszaki leírást vagy használati utasítást. Vagy ha tud, legyen szíves küldjön egyet.

Nem szívesen ismerem be, de még a hirét sem hallottam ennek a gépnek. Egy röpke körkérdést tettem fel néhány barátomnak, a gépet ők sem ismerik. Az egyetlen, amit tehetünk, közöljük a levelét, talán olvasóink közül valaki segíteni tud.

Emszt Mihály, Budapest,

Nagy Lajos kir. u. 21/B. 1148

Tavaly augusztus óta van egy Commodore 64-es gépem.

Most már a játékon kívül „próbálgatok" kisebb programokat írni, melyek a munkámat könnyítik. Az 1986/8. számban találtam meg az UNIN példaprogramot.

A listát és a hozzá tartozó szűk magyarázatot (nyilván többen értik is, én sajnos nem) úgy gondoltam, hogy más jellegű adatokkal tudnám használni.

A beírás majd a futtatás során a monitoron többszöri villogás után a következő hibaüzenetet írta ki: OUT OF MEMORY 1180.

Miután sok mindent még nem értek (pl. 1000—1015 vagy 1020—1057 sorok), Önhöz fordulok segítségért, amit előre is köszönök.

Tudom, hogy Önök csak közlik a beérkezett és közlésre alkalmas anyagot, így felelősséget sem tudnak vállalni.

Ezt az anyagot csak én szeretném felhasználni. Meg szeretném jegyezni azt is — lapjukban erről is többször volt szó — *az inverz jeleket célszerű lenne írás formájában leírni.*

Levelét elküldtem a cikk szerzőjének, remélem időközben a választ már megkapta. Ami az inverz jeleket illeti, az egyik legnagyobb gondunk. Miután a programlistákat számítógépen nyomtatják és nem szedetjük, okos megoldást erre a problémára még nem találtunk.

Fischer Kornél, Guttamási,

Kossuth u. 7. 8045

Nemrég kaptam egy C—64-et floppyval. Mivel magnóm nincs, úgy gondoltam, nem veszek, hanem építek egyet. Ehhez kérném az Önök segítségét. Ha van rá mód, a 1530. adatmagnó kapcsolási rajzát szeretném Önöktől megkapni.

Érdekelnek még a hardverbővítések is! Ha van birtokukban fényceruza működési leírás és rajz, kérem, küldjék el címemre. A felmerülő költségeket természetesen megtérítem. Megcímzett válaszboritékot mellékelek.

Levelét elküldtem a Novotrade-nek és az NJSZT HCC Commodore szekció vezetőjének, dr. Simonyi Endrének, talán segíteni tudnak.

Kovács Tamás, Kecskemét,

Reile G. 10. 6000

Azért fordulunk Önökhöz, mert nem várhatunk máshonnan segítséget. Reméljük, hogy tudnak nekünk segíteni — ha másként nem, kérjük, közöljék le ezt a kis levelet a magazinban.

A „Hubbit" nevű programról van szó, melynek megfejtésével hiába próbálkozunk már két hónapja.

A program megértéséhez elolvastuk:

1. Tulkien: A Bábu c. könyvét (ez a Hubbit c. könyv magyar nyelvű kiadása),
2. Sinclair Spectrum Játék és Program c. könyv 77. oldalán található programleírást.

A játék során több problémánk akadt:

1. Mire használható a „golden key"?
2. Mit lehet tenni ott, ahol ezt írja ki: „The place is too full for you to enter."
3. Hogyan használjuk a „shoot", a „turn", a „fill" és a „CROSS" parancsszavakat?
4. Hogyan használható a „Magic door" a Tündérkirály Palotájában?
5. Ha határozószót teszünk a mondatba, akkor az semmit nem finomit vagy változtat a mondat jelentésén, pl.: run quickly+run vagy say viciously=say.

A játék állása pillanatnyilag:

```
You are carrying
     a curious map.
     the Large key.
     the short strong sword.
     the valuable golden ring.
     the rope.
     the valuable treasure.
     the golden key.
Thorin waits.
You go east.
Lower halls
Visible exits are: south east up
You see:
          Nothing
Thorin enters.
The red golden dragon enters.
You are in the halls where the dragon
sleeps
Visible exits are: south east up
You see:
          the red golden dragon.
     Thorin. Thorin is carrying
     the small curious key.
The dragon says: „Well thief your cunning
has failed you this time. Prepare to die"
You have mastered 65.0% of this adventure.
```

Leköszöltük.

Schuman Ádám, Budapest,

Csorna u. 1. 1121

A lapot már közel egy éve vettem rendszeresen. A szerkesztésével, cikkeivel meg vagyok elégedve. De szerintem van egy nagy hibája: szinte kizárólag csak C 64 és ZX Spectrumhoz vannak benne programok, írások. Bár tudom, hogy ez a két legelterjedtebb típus, azért azt hiszem, hogy a C 16 és a +/4 is megérdemelne egy kis helyet.

Nekem C 16-osom van, és a legutóbbi 3-4 számban nemhogy keveset, de semmit nem találtam a gépemhez a Magazinban.

Már sokszor megírtam, hogy azokból a programokból tudunk gazdálkodni, amelyeket a programozók küldenek. Én sem értem, hogy miért kapunk kevés anyagot a C 16-hoz és a +/4-hez, amikor ezeket a gépeket még az iskolákban is használják. Remélem ez a felhívás is ösztökélni fogja a programok szerzőit, hogy munkáikat a magazinnak is elküldjék.

Gyenge László, Debrecen,

Soó R. u. 33. 4032

A magazin 87/4 számában olvastam felhivását a távoktatási -tanulási rendszer megszervezésében való részvételről. Nagyon szívesen részt vennék ebben a munkában. Több okból is: mindig vonzott az új, a még járatlan területek felfedezése, másokkal való megismertetése.

A tanítás, bár nem főhivatásban, hosszú évek óta az életem szerves része.

Munkahelyem 1970 óta a Magyar Posta. Jelenleg a Debreceni Postaigazgatóságon dolgozom mint szervezési és számítástechnikai osztályvezető.

Az igazgatóság számítástechnikai tevékenységének elindítása, irányítása volt és feladatom ma is, valamint az üzemszervezés minden területe. Mint valamikori ifjúsági vezető mikroszámítógépes kiállítást szerveztem másokkal közösen Debrecenben a Kölcsey Művelődési Központban, amit Ön nyitott meg. Akkor személyesen is találkoztunk.

A postás szülők gyermekeinek minden nyáron megszervezzük a számítógépes tábort.

Jelentkezésemet mint magánember teszem, de a koncepció ismeretében kollégáim támogatását is igyekszem megnyerni az ügynek.

Talán nem sértődik meg, hogy a levelét közlöm. Őszintén örülök, hogy segíteni kívánja az országos távtanulási rendszer megszervezését. Most — június elején — az érdekelt intézmények közötti együttműködés megszervezésével foglalkozunk és megpróbáljuk a rendszer létrehozásának és üzemeltetésének személyi és nem kevésbé fontos anyagi feltételeit megteremteni. A feladat — minél jobban megismerjük — annál nagyobb. Így a segítséget, a közreműködést mindenkinek köszönjük.

KOVÁCS GYÖZŐ

ADOK–VESZEK–CSERÉLEK

ADOK

C64-re angol nyelvvizsga-előkészítő szótanító program. 11 témakörben 1400 szót tanít, kiejtéssel. 2-3-szor gyorsabban tanulhat, mint szótárból. Ára magánszemélyeknek és közületeknek floppyn 3600 Ft. Kívánságra részletes tájékoztatót küldök. Cím: Tóth András számítástechnika, Post Restant, 1364 Budapest 4.

Commodore Plus/4 számítógép kazettás egységgel, programokkal, könyvekkel 18 000 Ft-ért eladó. Horváth Kálmán, Békéscsaba 2, Pf.: 25. 5602.

Junosty televíziót videomonitorra alakítok (kép+hang), az eredeti funkciók megtartása mellett. C128 esetén 80 oszlopos megjelenítésre is alkalmas! Páll Miklós, Budapest, II. Rákóczi F. u. 345. K7-10. C lépcső IV/9. 1214.

Laser 310-es számítógép bővítővel, magnó és Sanyo monitor eladó. Tóth Ferenc, Gönyü, Deák F. u. 1. Tel.: 96/53-036.

MPS-802 printer, MS-1265A típusú zöld, forgatható monitor eladó. Árajánlatokat a következő címre kérek: Pásztor Sándor, Miskolc, Repülőtéri út 4. 3526.

SELTRON/LASER 200 számítógéphez 64 k-s bővítő és floppymeghajtó áron alul sürgősen eladó. Sipos Attila, 851-011, munkaidőben.

Sinclair QL számítógép (töltő+csatlakozó+tápegység) gyári csomagolásban eladó. Tel.: 560-025.

TV-Computer (32 k és 64 k) eladó játékprogramokkal (kazettán). Érdeklődni: Lánycsók, Petőfi u. 26.

ZX81 4000 Ft-ért eladó. Cím: Dunaújváros, Erdősor 39. III/3. 2400.

ZX-Spectrum (48 k) + 300 db program (játék-felhasználói) eladó. Cím: Szendeffy Szilárd, Pécel, Rákóczi u. 5. 2119.

SZOFTVERFEJLESZTŐK ÉS FELHASZNÁLÓK! Szeretné programját eladni? Mi adatbázisunkban nyilvántartjuk. Problémájára nem talál programot? Keresse nálunk! Hardvert keres vagy kínál? Forduljon hozzánk! Programot, dokumentációt, tapasztalatot cserélne? Mi segítünk. Szaktudását, számítástechnikai ismereteit kamatoztatni szeretné? Szakembert keres speciális problémájára? A megfelelő információt nyilvántartásunkból biztosítjuk. Kérje ingyenes tájékoztatónkat! BCS, Kecskemét, Pf.: 298. 6000.

MBC-64 BASIC compiler /C64, C 1530/. BASIC programjait a memóriában fordíthatja lemezegység nélkül! Maximális programméret: 16 kbájt /64 blokk/. A lefordított program 10 soros BASIC program esetén már kisebb, mint az eredeti! Kérjen díjtalan ismertetőt! BCS, Kecskemét, Pf.: 298. 6000.

VESZEK

Schneider /Amstrad/ lehetőleg új CPC 664 vagy CPC 6128 /ártól függően/ számítógépet keresek megvételre. Fehér Csaba, Keszthely, Fürst Sándor u. 10. 8360.

CSERÉLEK

ZX-Spectrumra színvonalas játékprogram készletemet (kb. ötszáz darab) csere folyamán szeretném bővíteni. Csak színvonalas programokat cserélek (pl. Bomb Jack II., Infiltrator). Válaszokat a programok listájával kérek: Pintér Roland, Szombathely, Szegfű u. 1. 9700.

ZX-Spectrum (48 k) játék- és felhasználói programokat cserélek. A válaszokat listával együtt erre a címre kérem: Zsolnay József, Eger, Rákóczi út 85. fsz. 3. 3300.

Keresem a THE CASTLES OF DR. CREEP, EXPLODING FIST, FLIGHT SIMULATOR II., IMPOSSIBLE MISSION, SEVEN CITIES OF GOLD, SOLO FLIGHT, SPACE TAXI, MUSICALC, SAY IT, valamint egyéb komoly, oktató, zene- és szimulátorprogramokat kazettán. Cserébe más játékokat ajánlok. Tóth Balázs, Pétfürdő, Berhidai út 103. 8100.

C16 és PLUS/4 programokat cserélek kazettán. Egy programért kettőt küldök cserébe. Válaszokat listával kérek a következő címre: ifj. Éliás Sándor, Mátrafüred, Pálosvörösmarti út 32. 3232.

C64, C128 programokat cserélek. Lennardt Tamás, Győr, Kurkándi út 75. 9023.

C16 és PLUS/4 játékprogramokat cserélek. Maros Gábor, Tab, Rudnay u. 3. 8660.

Commodore 128-as gépre és 1571-es floppyra felhasználói programok – adatfeldolgozó, szövegszerkesztő stb. – cseréjéhez kapcsolatot keresek. A programok kézikönyvei, leírása is érdekel. Somori József, Budapest, Lujza u. 1/b. 1086.

Spectrum 128 k-s gépemet Commodore 64-re cserélném. 351-856.

48 k-s Spectrumhoz játékprogramokat cserélek, sok színvonalas programmal rendelkezem. Válaszokat listával erre a címre kérek: Brumbauer Zoltán, Mór, Münnich Ferenc út 15. 8060.

Spectrum programokat és programleírásokat cserélek. A válaszokat a programok és dokumentációk listájával kérem. Pataki János, Kiskunfélegyháza, Horváth Zoltán u. 2. 6100.

Egy hézagpótló mű problémái

Mit jelent az, ha valaki „mikroszámítógépekről" ír? Vagy azt, hogy magukról a gépekről, vagy az alkalmazásukról, vagy mindkettőről van szó. Az első esetben valóban „kemény", pontosan adatolható tényekről van szó: ember legyen a talpán, aki a gépek mai dzsungelében megbízható adatokat tud közölni. Mert amúgy általánosságban kiadni egy 156 oldalas könyvet csak a gépekről, nem valami telitalálat, hiszen a szóba jöhető olvasók elsősorban a számítástechnikai szakemberek közül kerülnek ki.

A hétköznapok laikus felhasználója a gépeket a felhasználásukon keresztül tudja igazán megközelíteni. Neki – még oly egyszerű formában – se a Vásárhelyi tárgyalta áramköri típusok, se a feldolgozás impulzusokban mért sebessége nem sokat mond: de még *a* programokra vonatkozó se. Mert nem *a* programnyelvekre, *az* operációs rendszerekre kíváncsi igazán, hanem az alkalmazói programokra és *azok* kiválasztási szempontjaira. Az alkalmazásról ugyanolyan profi módon kell ismereteket közvetíteni, akár a hardverről, de a „profi" jelző elsősorban nem a számítástechnikai ismeretek tényszerű közlésével érdemelhető ki, hanem eme ismeretek megfelelő, felhasználói szempontú *értékelésével.*

Persze van harmadik jelentés is: hogy mind a hardverről, mind pedig a szoftverről szó van – de ekkor az átfogó tárgykör és a rendelkezésre álló terjedelem legfeljebb az önképzőköri ismereteket engedi meg. Azaz a „mikroszámítógépekről" szólva nagyon pontosan el kell dönteni; kinek is szóljon a mű?

Mindebből talán érzékelhető, mennyire nem egyszerű ma a mikroszámítógépekre irányuló olvasói igényeket kielégíteni. Vásárhelyi Pál könyvével* az információs munka tevékenységi körét vette célba. Ha meggondoljuk, hogy e gépeket ma a játékprogramok és a számítási feladatok mellett túlnyomórészt ezen a területen használják, ezzel a körülhatárolással aligha könnyítette meg a dolgát.

Egy ilyen igényű könyv szellemi közvetítő – szakmai „interfész" – lehetne, mivel a felhasználót és a gépekhez értő programozókat ma szakadék választja el egymástól. A programozó – alkalmazói ismeretek hiányában – nemcsak tájékozatlan a megoldandó speciális problémák világában, de

* Vásárhelyi Pál: Mikroszámítógépek az információs munkában. Budapest, 1986. OMIKK, 175 oldal

többnyire kulturálisan is idegen e világtól, s ezért alig érzékeli magától a komfort iránt támasztandó követelményeket. Javaslatai ezért gyakran leegyszerűsítőek. A felhasználó számítástechnikai tájékozatlansága miatt nem képes megítélni a kapott program minőségét (csak a használatba vétel során jön rá, hogy kényszerzubbonyt kapott munkaruha helyett), és azzal sincs eléggé tisztában, mennyire határozottan követelheti meg szakterületén a programtól a manuális munkájában addig kialakult kultúrát. Magyarán: nem az a baj, hogy megrendelőként nem tudja azt, amihez az eladó ért, hanem az, hogy nem tud eleget saját alkalmazói igényeiről.

Vásárhelyi könyve tehát hézagpótló, de az első kísérletek gyermekbetegségei jellemzik. Már felépítése is elárulja, hogy mekkorát igyekezett markolni:

● Tárgyalja a gép- és szoftverkiválasztás szempontjait (1. és 2. fejezet), sajnos a legáltalánosabb szinten, ami a felhasználó számára viszont a legspeciálisabb: őt aligha az egyes operációs rendszerek előnyei érdeklik, ha nem programozó, s a Unixről vagy a CP/M-ről a könyv elolvasása után is csak annyit jegyez meg, hogy operációs rendszerek.

● A felhasználó szempontjából részletesen foglalkozik a szövegfeldolgozás és táblázatkészítés kérdéskörével (3. és 4. fejezet). Ez a könyv legsikerültebb része — meg kell jegyezni, hogy e területen találhatók ma a kereskedelmileg is legkiforrottabb alkalmazó termékek.

● Az 5. fejezetben az információtároló és -kereső programokra kerül sor. Itt a fájlszervezésről, az adatkezelés és adatbázis-kezelés közötti különbségről is szó van. Az ismertetett három példarendszer — a Superdoc, a Star és a dBASE III — azonban csak ahhoz a felhasználóhoz szól igazán, aki maga profi módon akar megtanulni programozni. Mert komolyan senki sem állíthatja, hogy például a dBASE III-mal — dibéz-láz hin és her — vállalati „éles" körülmények között, a napi munkát hivatásos módon támogató adatbázis-kezelő rendszert bárki amatőr szinten kialakíthat. Azaz az igazi alkalmazót — a vevőt — ez a rész nem nagyon érdekli, a hozzáértő programozónak meg túlságosan is kevés. Lényegében ez vonatkozik az integrált rendszerekre is, amiken Vásárhelyi részben a továbbfejlesztett operációs rendszereket, részben pedig a szövegszerkesztést, táblázatkezelést és grafikus képességeket egyesítő termékeket érti (6. fejezet).

A 7. fejezetben e problémák aztán hatványozottan jelentkeznek. A könyvtári rendszerek tárgyalása alkalmazói szempontból már annyira felületes, hogy legfeljebb azt elégíti ki, akinek a könyvtári munkáról semmiféle fogalma nincsen. (Az igazsághoz persze az is hozzá tartozik, hogy erről sokszor a könyvtárosoknak sincs elég fogalmuk.) De a könyvtári rendszerekre vonatkozó tengernyi — és részben magyarul is hozzáférhető — irodalomra gondolva fel kell tételezni, hogy a könyv terjedelme szabott itt ennyire szűk korlátokat.

● Az utolsó, 8. fejezetben a táv-adatfeldolgozást bizonyára már a következetesség jegyében tárgyalja a szerző: valóban mindent érinteni akart.

Kétségtelenül az irodai alkalmazás a könyv erőssége. Erre utal a szövegszerkesztéssel és a táblázatkezeléssel foglalkozó két fejezet jól adagolt részletessége. A könyv más helyén említett példák is elsősorban az ügyviteli, bér- és készletnyilvántartási alkalmazás világából származnak. A dokumentációs és könyvtári alkalmazás példái nemcsak ritkábbak, de velük kapcsolatban a könyvből jóformán hiányzik a problémaérzékenység. Az alkalmazási területek szelekciója talán nagyobb elmélyülést biztosíthatott volna, s ezáltal megfogalmazódhattak volna a felhasználót annyira nyomorító mai problémák is. Mert „mikroszámítógépekről" írni felhasználóknak igazán csak e problémák fényében érdemes.

Mert mivel magyarázható, hogy a hardverrel foglalkozó fejezetben futólag szóba kerül a karakterek gépi megvalósításának kérdése, s ezzel összefüggésben — egyedül — az ASCII kód. De a felhasználót nem e kód konkrét felépítése érdekli, hanem többek között az előnyei és hátrányai, összevetve a többi kóddal. Az, hogy ha egy adott program e kóddal függ össze, mi ennek a következménye a jelsorrendre, szükségképpen a rendezési képre, a betűkészletre, a magyar ékezetekre és a kiterjesztett latin jelekre. A karakterkészlet dokumentációs és könyvtári szempontból elsőrendű fontossága e könyv alapján mintha a világon se volna.

Más. Az adattár létrehozásával összefüggésben éppen a fájlmeghatározás és a beviteli és kiadási formátum megállapításának a problémái azok, amelyekkel a felhasználó elsősorban szembekerül, amikor belekerül a megrendelő szerepébe. Égetően hiányzik az olyan kézikönyv, amely érthetően írja le, hogyan kezdjen egyáltalán valaki hozzá tervezett adattárának vagy adatbázisának az elképzeléséhez. Hogy miben gondolkodjék? Hogyan szedje össze az adattípusait, és kezdjen gondolkodni bevitelben és kiadásban? Hogy egyáltalán képes legyen elképzelni beviteli formátumát a képernyőn vagy adatlapon. Hogy jó áttekintése legyen arról, mi minden követelhető meg egy-egy mezőn belül, a mezők között, a rekordon belül, a rekordok között? Hogy mi jellemzi a legismertebb alkalmazói programokat e téren? Hogy mi az almező, az ismételhetőség egy mezőn belül, hogy a halmazműveleti operátorokon kívül még mi minden létezik (szövegkörnyezeti, relációs stb. operátor)? Mert ha erről általánosságban fogalma van, ezerszer jobban megítélheti, jobb-e neki egy felajánlott program, mint ha azt tudja, mi jellemzi az operációs rendszert vagy a segédprogramot.

Ha jobban megnézzük a példaként bemutatott Superdoc és Star termékeket, feltűnik, hogy még a legigényesebb megjelenítési formátumuk sem közelíti meg azokat az előírásokat és nemzetközileg elfogadott szabványokat, amelyektől a könyvtáros napi rutinmunkája során egyszerűen nem tekinthet el. Vásárhelyinek teljesen igaza van, amikor — sajnos nagyon is röviden — utal arra, hogy valóban komoly feladatra alkalmas program megírása olyan tapasztalatot igényel, hogy az esetek többségében bele sem érdemes fogni: mert hazai körülmények között, ahol még nem alakultak ki a feladatokra specializált szoftverházak, olcsóbb valójában akár devizáért is — s erről nem esett szó — kulcsrakész rendszereket megvásárolni, mint saját erőből vagy rendelésre elölről kezdve megoldani olyasmit, amit már megoldottak. Megfizetve minden alkalommal a tapasztalatlanság tanulópénzeit. De a tájékoztatás akkor lett volna még teljesebb, ha kiderül az is — mert így van —, hogy elsősorban a dokumentáció és könyvtárügy területén ma egyszerűen nincs az az igényeket maradéktalanul kielégíteni képes kész alkalmazói program sem a piacon, amely versenyezhetne a manuális munkával készült termék — bibliográfia, katalógus stb. — minőségével. Legalábbis a hazai piacon — hangsúlyozzuk! Hogy jelenleg minden gépesítés azzal a veszéllyel fenyeget, hogy a termékek színvonala csökken, a hiányos megjelenítési lehetőségek, a gyatra kezelési komfort miatt inkább durvulásról, mint finomodásról beszélhetünk. (Az igényeket leginkább még a Microisis elégíthetné ki, de róla szó sem esik a könyvben.)

Mivel nincs megfelelő szoftverimport! Mivel beruházói szinten még mindig ott tartanak, hogy számítógépen csak a hardvert értik — nem pedig a hardver és szoftver elválaszthatatlan együttesét!

Vásárhelyi könyve — akarva-akaratlan — annak a hazai felfogásnak a tükre, amely először a hardvert határozza meg, s csak azután nézi meg, van-e hozzá szoftver. A könyv olvastán mindazoknak, akik az előre megadott hardver kalodájában már vergődtek, feltámad a vágyuk egy olyan írásra, amelyben előbb tárgyalnák végre azokat a problémákat, amelyek irodai, ügyviteli, dokumentációs és könyvtári területen ma sikerrel automatizálhatók, azt követné a szoftverek ismertetése, és végül, mintegy függelékben, telefonkönyvszerűen, a hardverek rövid, tömör specifikációja, ára, eladója (nem gyártója!) és a beszerzés pontos forrásai. És hozzájuk azok az alkalmazói programok, amelyeket kezelni tudnak.

Valószínű, hogy ilyen igényeket a „mikroszámítógépekről" általában nehéz kezelhető terjedelemben megfogalmazni. Mindenképpen célszerű a két fő alkalmazási terület — az ügyviteli-vállalati, illetve a dokumentációs-könyvtári — elkülönített tárgyalása, annál is inkább, mert a programkészítés mai szintjén valóban nagyon eltérően megoldandó feladatokról van szó. Legalábbis felhasználói szempontból. És feltehetően arra is nagy szükség van, hogy a hazai szakirodalmat is behatóbban figyelembe vegyék — már csak azért is, mert számos kérdés már úgy-ahogy megfogalmazódott. A hazai szakirodalmat a könyvben egyetlen cikk képviseli — ráadásul e cikknek az a változata, amely nem is a dokumentációs szaklapban — a *Tudományos és Műszaki Tájékoztatás*ban —, hanem lényegében perifériákusabb területen, a *Vezetés, Szervezés*ben jelent meg.

Mindebből az is következik, hogy Vásárhelyi az úttörők nem mindig hálás feladatát vállalta. Bizonyára kedve is volt hozzá, s ezért nemcsak kritika, hanem köszönet is illeti. A kritika címzettje ugyanis nemcsak ő: munkájához a jelek szerint több segítséget érdemelt volna. Erre utal egyrészt, hogy Brückner Huba értő számítástechnikai lektorálása mellett nagyon hiányzott az alkalmazói oldalról is a lektor, helyesebben hiányoztak a lektor*ok!* Hiányuk éppen annak lebecsülésére utal, akinek érdekében e könyv íródott: a vevő szerepét tanulni kényszerülő felhasználó lebecsülésére. És erre a következtetésre kényszerülünk, ha a könyv olvashatatlan és aligha reprezentatív képanyagára emlékezünk vissza enyhe borzongással. Az ilyen hézagpótló vállalkozás szervezésébe illett volna ennél többet befektetni. A könyv az OMIKK házi sokszorosítóüzemében készült.

UNGVÁRY RUDOLF

A NOVOTRADE RT

1987, 1988 évi számítógépes tanfolyamai IBM PC és AT kompatibilis gépekre:

- Rendszerhasználat és operációs rendszer	5 nap	7.000.-
- BASIC programozási nyelv	5 nap	7.000.-
- Macroassembler a Commodore PC-n	5 nap	7.000.-
- Cobol programnyelv a Commodore PC-n és a kompatibilis gépeken	5 nap	7.000.-
- A C programozási nyelv	5 nap	7.000.-
- A Commodore PC-XT professzionális személyi számítógép alkalmazása	5 nap	7.000.-
- Fortran programozási nyelv Commodore PC-n és kompatibilis gépeken	5 nap	7.000.-
- PC-20 szövegfeldolgozás - WORDSTAR	3 nap	6.000.-
- dBASE-III adatbáziskezelő	3 nap	6.000.-
- Többmunkahelyes rendszerek Commodore PC-n és kompatibilis gépeken	5 nap	7.000.-
- Lokális hálózat Commodore PC-n és kompatibilis gépeken	5 nap	7.000.-
- Supercalc és Pluscalc táblázatkezelő programok	3 nap	6.000.-
- Symphony	3 nap	6.000.-
- Rendszerkészítés számítógépen dBASE adatbáziskezelővel	5 nap	6.000.-
- LOTUS táblázatkezelő alkalmazásai személyi számítógépen	5 nap	7.000.-
- A Commodore PC Turbo Pascal nyelven	5 nap	7.000.-

10 fős csoportok esetén 20 %-os árkedvezményt biztosítunk!

Commodore típusú személyi számítógépekre alaptanfolyamok!

- tanfolyamonként.....1.500.- Ft

A tanfolyam helye kellemes környezetben a

"HOTEL REGE" kártyaterme
Budapest II., Pálos u.2.

Várjuk jelentkezésüket a 122-099; 122-095 telefonon.

Ügyintéző: Bakó Lászlóné (NOVOTRADE RT, 1136 Bp.Kresz Géza u.14.)

Beszédfelismerés

Az IBM új öntanuló beszédfelismerő rendszere a beszédhang alapján 20 ezer szót képes felismerni. Mivel minden ember hangja sajátos, a számítógép kéri, hogy használója először mintegy 20 perces, előre meghatározott szöveget olvasson be a gépbe. Ez alapján a gép megismerkedik az illető beszédének egyéni karakterével (ritmus, kiejtés), és megjegyzi azt. Ezután a dolog már egyszerű, mindössze egy kis mikrofonba kell beszélni — kis szünetet tartva minden egyes szó után —, s a számítógéppel értelmezett szavak megjelennek a képernyőn. A diktálással felvitt szöveget a fogadó szövegszerkesztő programmal fel lehet dolgozni és ki lehet nyomtatni. Az eredményben elsősorban a 20 ezer szavas szókészlet a kiemelkedő. Ezt az üzleti életből válogatták, s a gépet először az üzleti levelezés automatizálásánál és gyorsításánál kívánják alkalmazni.

Díszítés az ebédlőn . . .

A Zala Bútorgyár tömbfa-megmunkáló részlegének új büszkesége egy CNC-marógép, amelyet egy Siemens gyártmányú mikroszámítógép vezérel. Ez lehetővé teszi különféle díszítőelemek (például rozetták) sorozatgyártását. Így gazdaságosan állíthatók elő a ma divatos, kecsesebb formák is. A gép üzemeltetéséhez szükséges programok írásához egy önálló programozási csoportot hoztak létre, akik alig pár hónapos munkájukkal már nem csupán több sikeres programot, hanem több sikeres terméket is magukénak vallhatnak.

Autóelektronika

Prometeus néven új kutatási programot indít az Eurêka programban résztvevő 13 nyugat-európai autógyár és 40 kutatóintézet. A következő nyolc évben több mint egymilliárd nyugatnémet márkás beruházással olyan „fedélzeti számítógép" létrehozását tervezik, amely megbízható robotpilótaként funkcionálna személygépkocsikban: ellenőrizné működésüket, szabályozná az üzemanyag-felhasználást, a károsanyag-kibocsátást és — többek között a külső veszélyforrásokról — információkkal látná el az autóvezetőt. A szakértők szerint a gépkocsi-elektronikai piac hihetetlen tempóban nő: tavaly körülbelül egymiliárd dollár értékben (ezen belül csak Nyugat-Európában 380 millió dollárért) vásároltak a nyugati autógyárak autókba építendő elektronikai alkatrészeket, és az évtized végére ez a mennyiség a duplájára nőhet.

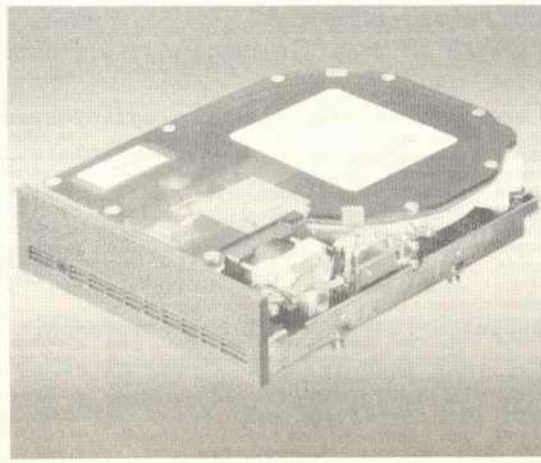
Rutinprogramok géppel

Az ötödik generációs számítógépek kifejlesztésének egyik újabb fontos állomásához érkeztek a mesterségesintelligencia-kutatások gyakorlati alkalmazásával foglalkozó japán kutatók. Ennek eredményeképpen a NEC cégnél elkészült egy új kísérleti rendszer, amely Cobol nyelven programokat ír. A kezelő szóban vagy a billentyűzet segítségével közli kívánságait a felállítandó programmal kapcsolatban, aminek alapján a gép önállóan készíti el a Cobol programot. Az esetleg még tisztázandó kérdéseket szóban kérdezi vissza a kezelőjétől. A rendszer jelentősen hozzájárulhat a jelenlegi programozóhiány felszámolásához s hatékonyan meggyorsítja a jó minőségű programok készítését, elsősorban az olyan időigényes rutinterületeken, mint például a táblázóprogramok írása.

172 Mbájt

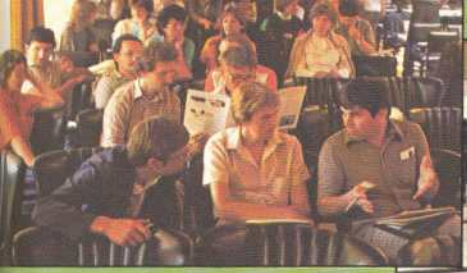
Az év krónikájába való, hogy tavasszal a Hitachi megjelent a piacon a 172 Mbájtos (formattáltan 134 Mbájtos) tárkapacitású, 5,25 hüvelyk átmérőjű hajlékonylemezes tárolójával. Rendkívül kedvező az átlagos elérési idő is, mindössze 23 msec. A 3,7 kp súlyú tár meghibásodásai közötti időt 20 ezer órára garantálják.
A szintén japán NEC cég félmagas, 51 Mbájtos (formattáltan 40 Mbájtos) kapacitású winchester-táránál az átlagos hozzáférés ideje 85 msec. Az 5,25 hüvelyk átmérőjű, 1,8 kp súlyú tár két meghibásodás közötti ideje szintén 20 000 óra.

A NEC cég D51XX típusú winchester-tára



Görcsöny

Milyen legyen a vetésszerkezet? Mivel a döntésnél sok mindent és azok együttes hatását is figyelembe kell venni, ma már számítógép a segédeszköz több tsz-ben az optimális szerkezet megállapításához. A görcsönyi tsz januárban mutatta be a Gödöllői Agrártudományi Egyetem oktatóival közösen kidolgozott vetésszerkezet-optimalizáló számítógépes programját.
A tulajdonságokból indultak ki. A vetésszerkezet meghatározásánál lényeges a korábbi növényvédő szerek és a tápanyagellátottság ismerete, valamint az adott területre (növényre) vonatkozó tápanyagigény: a számítógépes feldolgozás eredményeképpen például a tavalyi tizennégymilliós műtrágyaköltség idén már néhány millió forinttal csökken. Megállapítást nyert, hogy a tsz földjei foszforban és káliumban gazdagok, nitrogénben azonban igen szegények, és az ismeretek szerint csak a szükséges tápanyagot juttatják a talajba.
Az idén kukoricát vetettek a tervezetben jelölt négyszáz hektár napraforgó nagyobb részén is. Hogy ennek mi az oka? Ahová a napraforgót szánták, ott korábban kukorica volt, amit triazin tartalmú vegyszerrel kezeltek — s ezt nem bírja el a napraforgó. A számítógépes vetésszerkezet-meghatározás az ilyen bakik előfordulásának a lehetőségét csökkenti azzal, hogy a program figyelembe veszi a vegyszerezések talajmaradványait. Ez sokak számára nem új, de a korszerű technika mégis elsőrendű látványt kínál.
A tsz vezetése abban bízik, hogy a számítógépes vetésszerkezet-optimalizálás az objektív lehetőségek jobb kihasználásával lényegesen jobb eredmények elérését teszi lehetővé.



Fonyód '87

A konferencia színhelye

Az idén ötödízben volt számítástechnikai konferencia színhelye Fonyód: a Karikás Frigyes Gimnázium és Szakközépiskola kollégiumában június 24. és 28. között tartották a számítástechnikai oktatók országos tanácskozását és tapasztalatcseréjét. A négynapos rendezvény szervezője a III. Budapesti Szervezetének matematikai szakosztálya, az NJSZT Somogy Megyei Szervezete és a Balaton parti középiskola volt.

A programokban gazdag konferencián mintegy százhúsz közép- és szakközépiskolai tanár vett részt. A számítástechnika terén gyűjtött tapasztalataikat cserélték ki egymással, és segítették a megyék középszintű iskolái közötti információáramlást. A tanárok beszélgetéseikben igyekeztek közzétenni megfigyeléseiket, hogy elkerüljék azt a gyakori jelenséget, miszerint nem ismerik a másik által elért eredményeket. A négy nap alatt személyes kapcsolataikat arra használták, hogy megvitassák a számítástechnikára vonatkozó elképzeléseiket. A lehetőségek megismerésével keresték a továbblépés módját.

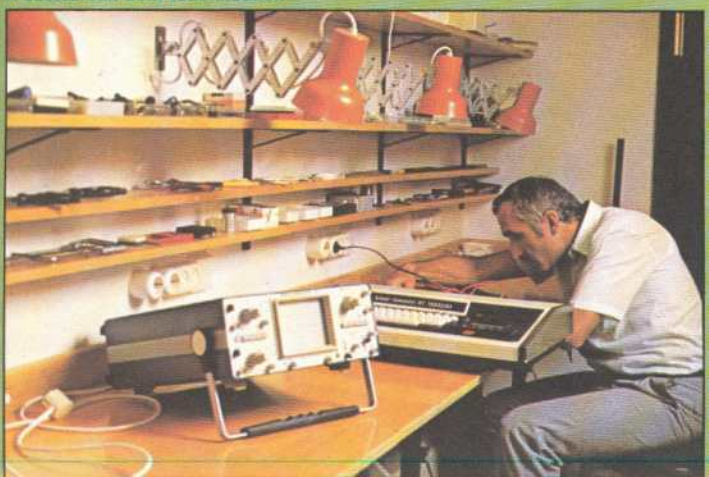
Az élénk eszmecserék mellett számítástechnikai szoftver- és hardverkészítő cégek mutatták be termékeiket: a Balatonboglári Mezőgazdasági Kombinát, a Color Ipari Szövetkezet, a Dombóvári Híradástechnikai Szövetkezet, az International House Language School és a Videoton.

Tartalmas előadások színesítették a konferencia programjait. Többek között előadás hangzott el arról, hogy hazánkban miként terjed a számítástechnikai kultúra, milyen a számítógép és a nyelvtanulás kapcsolata. Szóba került az iskolaszámítógép-program helyzete és a lokális hálózat, mint oktatástechnikai segédeszköz problémaköre. A résztvevők szekcióüléseken bővíthették tudásukat, és éjszakába nyúlóan cserélhettek szoftvereket.

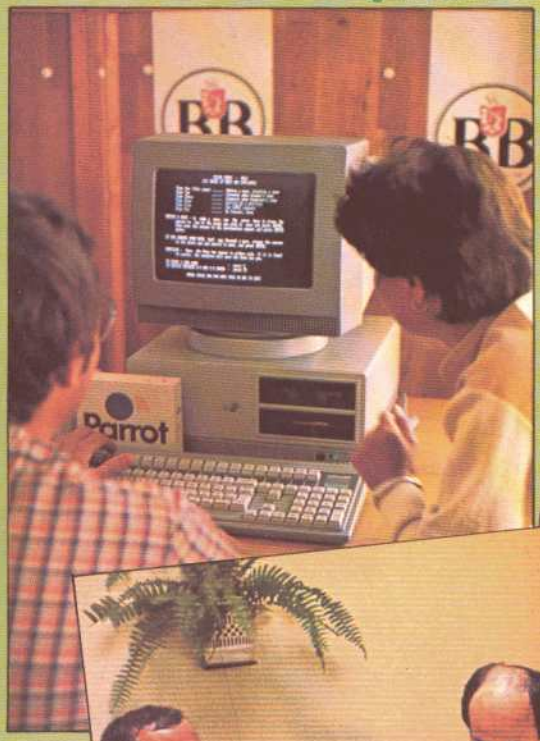
A rendezvényen adtuk át a Mikro '87 díjnyertes versenyzőinek, Horváth Istvánnak és Németh Lászlónak — a fonyódi konferencia két házigazdájának — szerkesztőségünk különdíját.

Egymásra nézve az oktatóteremben

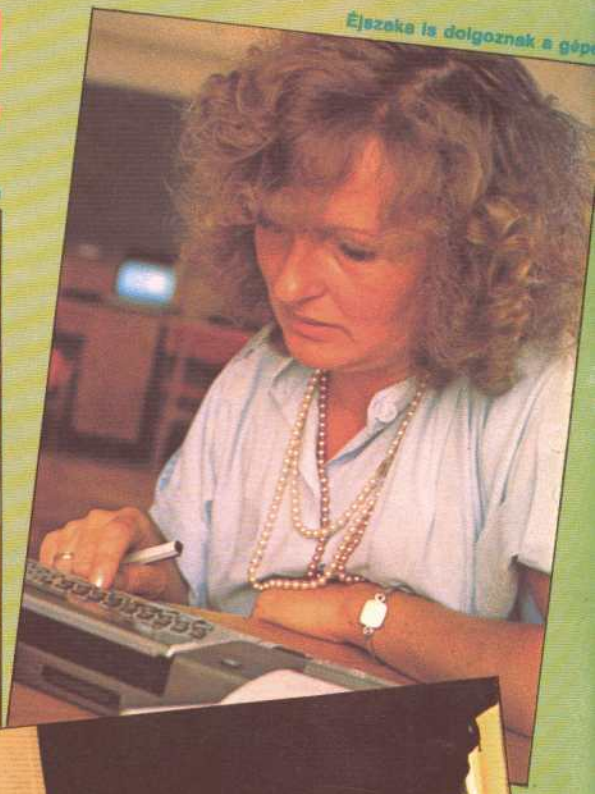
A szertárban sem tétlenkednek



Fonyód '87



Program a szünetben



Éjszaka is dolgoznak a gépek

Szerkesztőségünk díjátadása