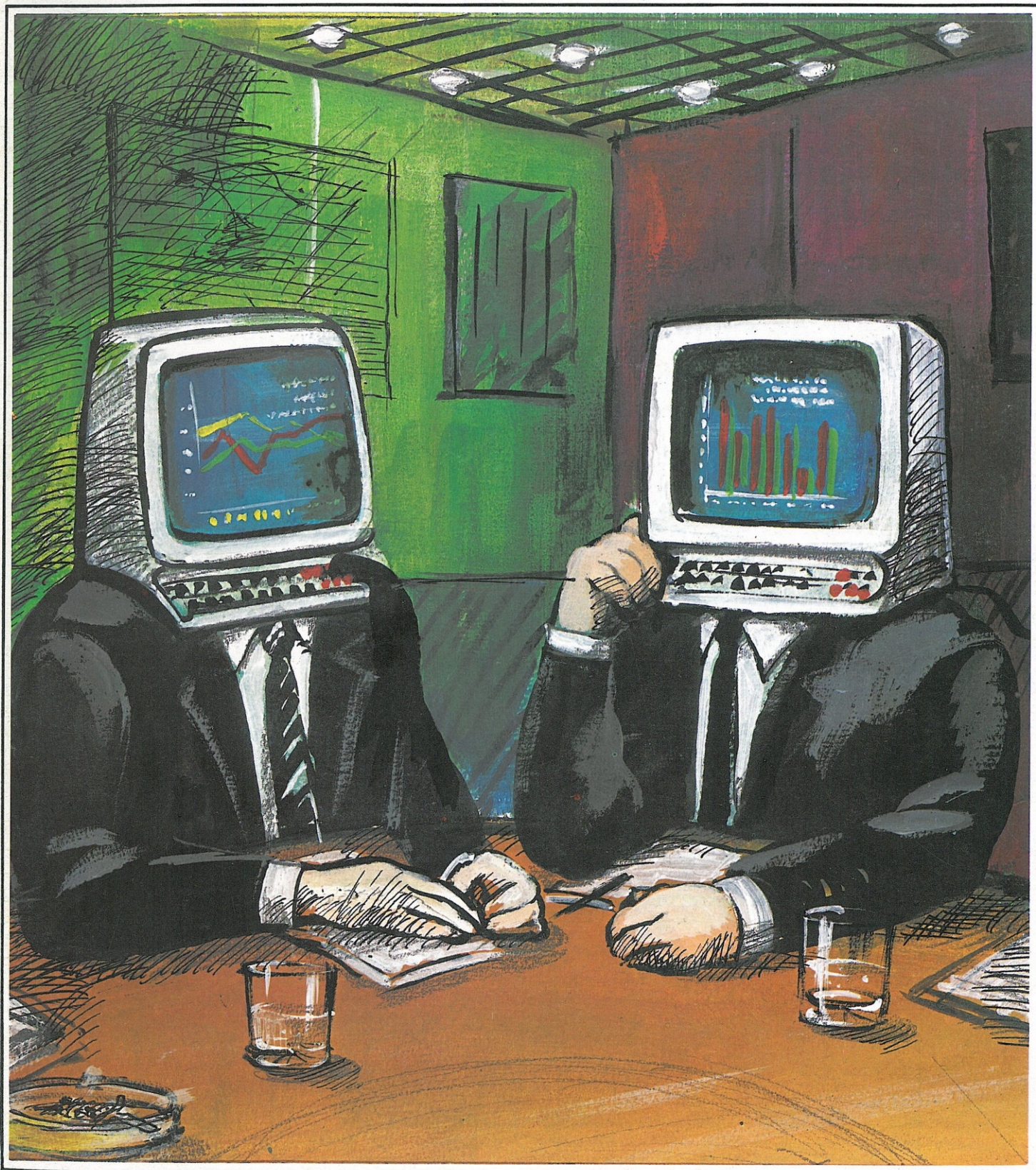


mikro

számítógép

magazin

Ára: 30 Ft



KONFERENCIA

1987/4

SZÖVEG, TÁBLÁZAT, GRAFIKON, GRAFIKA-KÉP készítéséhez sokoldalúan felhasználható a PRT—80 GS GRAFIKUS MOZAIKNYOMTATÓ

**Személyi számítógépekhez, mikro-, minigépekhez
intelligens terminálokhoz, mérőkészülékekhez.**

**A berendezést ajánljuk: PRIMÓ, IBM és IBM-kompatibilis, vala-
mint Sinclair és Commodore személyi számítógépekhez, ame-
lyeknél a csatlakoztatást a megrendelő kívánságára elvégezzük.**



**Műszaki
Vevőszolgálat:
ELEKTROMODUL**

Budapest XIII.,
Victor Hugó u. 11—15.
Telefon: 495-340,
251 sz. mellék.

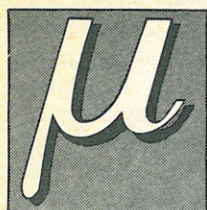
**Árusítás:
ELEKTROMODUL**

2. Sz. Szakboltjában
Budapest XIII.,
Jászai Mari tér 5.
Telefon: 530-800

**Gyártja:
BHG
Híradástechnikai
Vállalat**

1119 Budapest,
Fehérvári u. 70.
H—1509 Budapest, Pf. 2.
Telefon: 453-300





mikro számítógép magazin

5. ÉVFOLYAM
1987/4. SZÁM

**A NEUMANN JÁNOS
SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG
ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA**

A kiadvány
a Tudományos-
és Informatikai
Intézet
együttműködve készül

A szerkesztőbizottság
vezetője:
Kovács Győző

E számunkat
szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

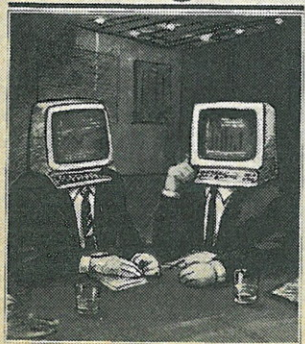
Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Címképünk:
Ramocsai Imri munkája

**μ mikro számítógép
magazin**



KONFERENCIA

1987/4

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.
86-253



Szikra Lapnyomda
Budapest (87-0307)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Gyerekek	2
Program Információs Központ középiskolásoknak	9
Távkonferencia I.	10
Vegyem, ne vegyem?	12
Emlékképek Neumann Jánosról	14
Mit tud a Pascal?	21
RAINBOW a μM-ban	24
Mikrogépes úttörők	31
Vigyázat! Tolvaj!	32
Adok — veszek — cserélek	35
Az ATARI 800 XL	37
μINFORM	41

ISKOLA — SZÁMÍTÓGÉP

Az ismeretlen C16	3
Egy új BASIC fordítóprogram: a PLUS-Comp	5
Gépi kódban	6

DIÁKROVAT

Megszakítás	6
Eljárásdefiniálás	8
Betűzsonglörködés	9

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód	18
OVERLAY—CHAIN— APPEND	19

μPROGRAMOK

Oszlopdiaagram-rajzoló	26
Menük	26

μKLUB

Integrált szoftver	33
Egy jól használható programjavító segédlet	35
Adom a magyarázatot!	36

SAKKPROGRAMOZÁS

Bitek és figurák	43
------------------	----

AZ OLVASÓ ÍRJA

JÁTÉKPROGRAMOK

Úrhajózás	46
Világosság a Sötét Torony körül	46

KÖNYVEK

HÍREK-ÉRDEKESSÉGEK	48
--------------------	----

„A tengerpartot járó kisgyerek
mindig talál a kavicsok közt egy-
re,
mely mindöröktől fogva az övé,
és soha senki másé nemis lenne.”

(Pilinszky János:
Egy szenvedély margójára)

Egy jó barátommal találkoztam, aki rosszkedvű volt. Lehangelve újságolta, hogy nyári szabadságuk alatt, úti ellátmányukból megtakarítva minden megtakaríthatót, elcsípve még egy testhezálló leárazást is, megvették a GYEREKNEK a már nagyon régen kívánt számítógépet. Vettek hozzá fél tucat játékkazettát is, hogy a GYEREK ne sikertelen kísérletekkel kezdje a — minden biznnyal az egekbe tartó — programozói karrierjét, hanem játékokkal szokjon hozzá a gépekhez, és akkor észre sem veszi, hogy már nem is játszik, hanem programoz, sőt versenyeket és pályázatokat nyer.

A papa tervez, a GYEREK végez. Elmúlt néhány hónap, és a GYEREK megunta a játékokat, feléje se nézett a gépnek. „Kiadtam egy csomó pénzt — mondja a papa —, és a helyzet pontosan ugyanaz, mint annak idején a videojáték vásárlásakor volt: a GYEREK azt is megunta és ezt is, azt gyorsabban, ezt lassabban. A különbség szinte semmi!”

Gyanakodva kérdezte egyébként tőlem, miután szorgos olvasója a μ M-nak, hogy az „Olvasó írja” leveleit ki küldi? A kérdésből nem volt néhez kitalálni: föltételezte, hogy én vagyok a levélíró is, meg én is válaszolok magamnak. Nem értette ugyanis, hogy még mindig vannak és egyre többen lesznek lelkes gyerekek és felnőttek, akiknél nem múló divat, hanem tartós „szerelem” a számítógép.

Hosszasan elbeszélgettünk, de még sokáig nem hagytam nyugodni a probléma. Figyelni kezdtem az iskolákban, a szakörökben és a klubokban folyó munkát. Beszélgettem általános iskolásokkal és egyetemi hallgatókkal, tanárokkal és diákokkal. Megpróbálom röviden összefoglalni tapasztalataimat.

Jó ideje folyik már az iskola-számítógép-program, valószínűleg nincs olyan középiskola, ahol ne volna néhány számítógép, de lassan ilyen általános iskola sem lesz már. A felsőok-

Gyerekek

tatásban pedig a számítógép az oktatás és a tanulás legfontosabb (segéd)eszköze.

Mégis az a véleményem alakult ki, hogy a sok-sok biztató jelenség mellett az ifjúság és a számítógép közötti kapcsolatnak már vannak aggasztó tünetei is. Egyik-másik pedagógustól azt hallottam, hogy a drága pénzen vett HT, Primo, Spectrum, C16 és a többi iskolai gépért már nem folyik kézitusa a szünetekben, és már nem kell gépet foglalni a délutáni szakkörre sem. Az NJSZT Közpon-ti Klubjában is azt tapasztaltuk, hogy nincs szükség a gépidőigény előzetes bejelentésére, mindenkinek jut elég gép, ha bejön a Báthori utcába.

Azt hiszem, hogy a GYEREK és a számítógép viszonyában egy új korszak kezdődött, a gyerekek — én így gondolom — nagyon gyorsan „kinőtték” ezeket az egyszerű gépeket. Az egyik kis barátom, már első gimnáziumba jár (!), valahogy úgy fogalmazott, hogy „ezekkel az iskolagépekkel már nem lehet semmi újat csinálni”.

Ő is zsonglóra a programozásnak, aki csak képernyőn dolgozik, a BASIC-en kívül ismeri a PASCAL-t (néha az apuka hivatali IBM-klónjára jár programozni), de bevallotta, hogy hálás lenne, ha egyszerű egy igazi nagy gép termináljához is odaülhetne. (Azóta kívánsága teljesült.) Ez a gyerek angol szakirodalmat olvas, ismeri a lokális hálózatokat, szabad idejében hálózati protokollokat tervez.

Véleményem szerint a számítógépre nem igaz a mondás: „egy újszülöttnek minden vicc új”, tehát majd jönnek az új alsó tagozatos általános iskolások, akik majd ugyanolyan hévvel vetik magukat a mai iskolagépekre, mint négy-öt évvel ezelőtt „nagy elődeik” tették. Ez az új generáció már nagyon sok előismerettel kerül az iskolába, amelyeket klubokban vagy a családi számítógépen, esetleg a barátainál szerzett, általában a mai iskolagépeknél korszerűbb számítógépeken.

El kellene gondolkodni azon, hogy az iskola-számítás-technika hogyan fejlődjek tovább. Mi legyen a cél? Tart-e még a Commodore-korszak, vagy már akkor túlhaladtuk, amikor megkezdjük? Győznek-e az IBM-klónok az iskolában is, vagy pedig ilyen gépek az általános iskolákba és a középiskolákba sohasem jutnak el? Mi ér többet az iskolában, egy IBM-klón vagy 10 db Commodore Plus/4, netán C16? Melyik géppel lehet legjobban a kívánt pedagógiai-nevelési célt elérni, vagy inkább azt kellene megkérdezni, hogy mi is ez a cél?

Az egyik kérdés hozza a másikat, pedig még nem is beszélünk az oktatási szoftverről (courseware), a számítógépi grafikáról, az iskolai robotokról, de mindazokról a lehetőségekről se, amelyeket egy jól átgondolt iskolai számítógép-vásárlási és -alkalmazási politika megteremthet.

Mintha itt kellene keresni a számítógépet megunt gyerek problémáját... Valószínűleg az is igaz — és ezt a pedagógusok biztosan jobban tudják —, hogy a gyerekek többsége nem eléggé kreatív, nem rendelkezik elég kitartással egy ilyen ravasz szerkezetnek, mint a számítógép, titkai kikutatásához, és talán ezért unja meg a gépet. Akkor viszont vezetni kell, meg kell vele ismertetni az informatika egyre fejlődő világát, amire nyilván csak jól képzett és a számítógépért lelkesedni tudó pedagógusok képesek. Nem volna szabad hagyni — és ezért a számítógépes szakemberek is felelősek —, hogy az iskolai rendszerek megmerevedjenek, tovább kell fejleszteni a rendszereket, mert akkor a GYEREK-ben is felébred a vágy az izgalmas iskolai rendszerek otthoni megvalósítására. Legyen az iskola hardver- és szoftverlaboratórium, amelyet az otthoni számítógépes környezet követ, és nem fordítva: otthon van a csodagép, a csodaszoftver, az iskolában pedig „uram bocsá” a tömény unalom.

Én azt hiszem, hogy a jövőben nagyon fontos szerep vár az iskolai fejlesztő, laboratóriumszerű munka kialakításában a pedagógusokra. Nagyobb, mint azt korábban gondoltam, azokban a szép időkben, amikor a tanár és a diák együtt ismerkedett a számítógépekkel. Én azt remélem, hogy ez az idő elmúlóban van, már egyre több diáktól hal-lom: arra várnak, hogy tanítsák őket alkalmazásokra, technológiára, feladatmegoldásra, numerikus módszerek ismeretére és mindazokra a „tudományokra”, amelyeket már nem lehet kitalálni, hanem meg kell tanulni.

Nem vitás, hogy szükség volt erre a néhány éves „anarchiára” az iskolai számítástechnikában, de lassan „múszáj” elkezdődnie a tisztulási folyamatnak. Változatlanul nem hiszek abban, hogy a számítástechnika tantárgy lehet akár az általános, akár a középiskolában, inkább az informatikai ismeretszintek egy nagyon átgondolt választékát képzelem el mint tananyagot: a különböző programozási nyelvektől az adatbázis-kezelő rendszereken keresztül a hardverismeretekig, amiből a diák az érdeklődésének megfelelően választhat. Az alkalmazói és egyéb programokat pedig találja meg vagy az iskolai számítógépen, vagy pedig a géphez kapcsolt külső számítógépeken (pl. más iskolák kompaktibilis rendszerein). Ezeket az után vagy a választott iskolai munkájához használhatja fel, vagy egyszerűen „csak” azért foglalkozzon velük, mert érdeklik.

Mindnyájan tudjuk, végül is nem az a cél, hogy minden diák programozó vagy hardveres szakember legyen. Sokkal inkább az, hogy amikor magasabb fokú tanulmányait végzi, illetőleg munkába áll, akkor könnyedén használja majd a számítógépeket. Ennek a cél-nak az elérésére össze kell fog-nunk a pedagógusoknak és a számítástechnikusoknak, a diáknak is és a tanárnak is, hogy ez a szellemi és fizikai környezet az iskolákban létrejöhön.

Én optimista vagyok, és remélem, hogy ebben a környezetben már csak nagyon kevés diák fogja megenni a számítógépeket.

Kovács Győző

Az ismeretlen C16

Felhasználói tokenek

A C16-tal alaposabban foglalkozó olvasó az ASCII táblázatokban bizonyára többször találkozott a \$FE tokennel. Mellette magyarázatként valamilyen semmitmondó szöveget olvashatott: „üres” vagy „nem használt”. Ez természetesen nem „üres”, hanem úgynevezett felhasználói token, ami szellemes lehetőséget nyújt azok számára, akik BASIC programjuktól gépi kódú rutinokat akarnak hívni. Ezt köztudottan megtehetik a SYS utasítás vagy az USR függvény segítségével, de a BASIC interpreter egy harmadik lehetőséget is szolgáltat: a rutinok beépítését az interpreterbe és ezek meghívását saját képzésű kulcsszavakkal. Gépi kódú rutinokat tehát nem a SYS[cím] utasítással kell meghívunk, hanem egy általunk létrehozott új BASIC típusú kulcsszóval.

A dolog megértéséhez vizsgáljuk meg, hogy mi történik a gépben, amikor valamilyen szöveg — programsor, parancs — beírása után leütjük a RETURN billentyűt.

A billentyűzetről bevitt szöveg karakterei ideiglenesen egy átmeneti tárolóterületre, a beviteli pufferbe (helye: \$0200—\$0258) kerülnek. Ha a szöveg első karaktere szám, akkor az interpreter ezt címformátummá alakítja. A rutin a \$8E3E címen kezdődik, a \$3B—3C mutatónak az átalakítandó szám elejére kell mutatni. Az eredményt a \$14—15 címekre teszi. Ezután a tokenizáló rutin a szövegben található BASIC kulcsszavakat egybájtos tokenekre cseréli. A tokenizálásnak két előnye van: a sor a tárban lényegesen kevesebb helyet foglal (például az SCNCLR utasítás 6 helyett csak egy bajtot), és végrehajtáskor sokkal gyorsabban azonosítható, mivel csak egy bajtot kell ellenőrizni. Ha a tokenizált sor első karaktere szám volt, akkor az interpreter elhelyezi a többi sor közé, egyébként a BASIC végrehajtó ciklusra ugrik, amely gondoskodik a sor végrehajtásáról.

Az általunk létrehozott új kulcsszavakat is érdemes tokenizálni. Ehhez vizsgáljuk meg először, hogyan működik az interpreter tokenizáló rutinja. A rutin a \$8953 címen kezdődik és a \$8A3C címen fejeződik be. Megoldandó feladatunk szempontjából a rutin három részlete érdemel figyelmet.

Az első:
 8953 JMP (\$0304)
 8956 LDA \$3B
 PHA
 LDA \$3C
 PHA
 JSR \$0479
 JMP \$8965
 8962 JSR \$0473
 8965 BCC \$8962
 JMP (\$030C)
 896A .
 .
 .

A rutin a belépési ponton a \$0304 vektor által meghatározott tárcímre ugrik (ez állapotban \$8956), majd a szövegmutató (TXTPTR) értékét a verembe teszi. A CHRGOT (kezdőcíme: \$0479) rutin segítségével az A regiszterbe tölti a szöveg aktuális karakterét, majd a \$8965 címre ugrik. A CHRGOT és a CHRGET (\$0473, a szövegben következő karakternek az A regiszterbe töltése) rutin az átviteljelzőt törli, ha a betöltött karakter szám. Ezt ellenőrzi a BCC utasítás, amely az \$8962 címen keresztül a CHRGET rutinra ugrik, mivel BASIC kulcsszó számmal nem kezdődhet. A JMP (\$030C) utasítás alapállapotban — amíg a \$030C vektor értéke \$896A — a tokenizáló rutin következő utasítására ugrik.

Az eddig elmondottakból emeljük ki azt, hogy saját új kulcsszavaink tokenizálásához a \$030C vektor átírására van szükség.

A \$8A03 címen kezdődik az interpreternek az a rutinja, amely kikeresi a BASIC kulcsszóhoz tartozó tokent. Az interpreter \$818E címétől kezdődik a BASIC kulcsszavak táblázata. Ennek hexadecimális listájából kitűnik, hogy minden BASIC kulcsszó utolsó betűje \$80-nal (128-cal) nagyobb értékű, mint a megfelelő betűkód, valamint hogy a táblázatban az utolsó BASIC kulcsszó (WHILE) után egy \$00 bajt áll.

A BASIC kulcsszavakat azonosító rutin a beviteli pufferben levő „aktuális” karaktert összehasonlítja a szótáblázat első karakterével. Ha egyezőséget talál, akkor a pufferben is és a táblázatban is egy karaktert előrelép és újra összehasonlít stb. Ha olyan karaktereket talál, amelyek nem

egyeznek meg, de a kódjuk különbsége \$80, akkor a beviteli puffer eddigi karaktersora — az aktuális karaktertől számítva — BASIC kulcsszó, az átviteljelző értéke 1 lesz, a token pedig a kulcsszó táblázatbeli sorszáma + \$80, amit a \$0B címre tesz le. Ebben rejlik az utasításrövidítési lehetőség magyarázata is: az interpreter nem vizsgálja, hogy a táblázatbeli vagy a pufferbeli karakter-e a nagyobb, csak azt, hogy a különbség \$80-e. A SHIFT pedig a karakter kódját éppen \$80-nal növeli meg.

Ha nem egyező karaktereket talál és a különbség nem \$80, akkor az interpreter a pufferben ismét az aktuális karakterre áll, a táblázatban megkeresi a következő BASIC kulcsszót, és újra indul az összehasonlító vizsgálat. Ha a táblázatban a \$00 bajt hoz ér, akkor az aktuális karaktertől kezdődő karaktersor nem BASIC kulcsszó, az átviteljelzőt törli és RTS utasítással visszatér a hívó rutinba.

Számunkra még egy hasznos része van az interpreter tokenizáló rutinjának: az, amelyik a \$89D4 címen kezdődik. A rutin a felismert felhasználói kulcsszót tokenjére cseréli, és a kulcsszóból visszamaradó felesleges bajtokat törli. A felhasználói token \$FE bajtból (ez jelzi az interpreternek, hogy felhasználói tokenről van szó) és a saját utasításunk általunk készített táblázatbeli sorszámából áll.

Most már hozzáfoghatunk a saját kulcsszavainkat tokenizáló rutin elkészítéséhez. Először kulcsszavainkból olyan táblázatot kell készíteni, mint amilyen az interpreterben van, majd a következő rutin beírása után annak kezdetére állítjuk a \$030C vektort. PHA

LDA #\$ a saját szótáblázat kezdőcímenek felső bajtja
 LDY #\$ a saját szótáblázat kezdőcímenek alsó bajtja

JSR \$8A07
 PLA
 BCC VEGE
 LDA \$0B
 PHA
 JMP \$89D6

VEGE: JMP \$896C

Az A regisztert azért kell a verembe menteni, hogy az interpreter folytatni tudja a tokenizálást, ha nem saját kulcsszó kezdődik az aktuális karakterrel. Amikor az interpreter a rutinunkra ugrik, akkor az A regiszterbe a puffer aktuális karakterét teszi; ezt a belső tokenizáló rutin a tokenizálásnál felhasználja, ezért a mi rutinunk nem változtathatja meg. A kulcsszót azonosító rutint nem azon a címen hívtuk meg, ahol az interpreter használja, hogy az a saját kulcsszó táblázatunkban keressen.

A tokent elhelyező rutint sem a fent ismertetett címen hívtuk meg. A rutin eleje az eredeti belépési helytől listázva:

89D4 PHA

DEY

89D6 DEY

A rutin két DEY utasítást tartalmaz, nekünk azonban a helyes működéshez az Y regisztert csak eggyel kell csökkenteni. Ezzel tokenizáló rutinunk kész van, de tokenjeinket listázáskor vissza is kell alakítani kulcsszavakká.

A BASIC-ben a LIST utasítás a \$8AFF címen kezdődik. Ebből azonban számunkra csak a \$8B6B címen kezdődő rutin érdekes: ez írja a képernyőre a token alapján a kulcsszót. A rutinban a \$8B85 címen egy JMP (\$030E) utasítás található, amelyre akkor ugrik, ha felhasználói tokent talált. A token az A regiszterben van. A \$8B93-\$8BBB-ig terjedő rész keresi ki a kulcsszót és írja a képernyőre. Érdemes megvizsgálni a rutinnak ezt a részét:

8B93 TAX

STY \$49

LDY #\$81

STY \$23

LDY #\$8E

8B9C STY \$22

Láthatjuk, hogy a BASIC szótáblázat kezdőcímét a \$22-\$23 címre tölti. Ha ezt a részt kikerüljük, akkor a többi saját kulcsszavaink kiírására használhatjuk.

TAX

STY \$49

LDY #\$ a saját szótáblázat kezdőcímének felső bájta

STY \$23

LDY #\$ a saját szótáblázat kezdőcímének alsó bájta

JMP \$8B9C

A rutin működéséhez a \$030E vektort a rutin kezdetére kell állítani.

Ahhoz, hogy saját kulcsszavainkat használni tudjuk, szükség van egy olyan rutinra, amely meghívja a kulcsszóhoz tartozó

gépi kódú rutint. Az interpreterben ezt a feladatot a \$8C25 címen kezdődő rutin végzi. Ebből a számunkra felhasználható részlet:

8C59 SEC

SBC #\$80

BCC \$8C90

ASL A

TAY

LDA \$8384, Y

PHA

LDA \$8383, Y

PHA

8C68 JMP \$0473

A rutinrészletben az A regiszter az aktuális — értelmezni kívánt — bájtot tartalmazza. Ha az átviteljelző törölve van, akkor nem token, a BCC \$8C90 a LET utasítást meghívó rutinrészletre ugrik. Egyébként a kivonás eredményét megszorozza kettővel, majd áttöltve az Y regiszterbe, a tokenhez tartozó rutin kezdőcímét a verembe tölti. Az interpreterben a \$8383 címtől kezdve felsorolásra kerülnek az egyes tokenekhez tartozó rutinok kezdőcímei alsó bájtt—felső bájtt formátumban. A JMP \$0473 utasítás a CHRGET rutinra ugrik, amely RTS utasítással fejeződik be. Ez a veremből a tokenhez tartozó rutin kezdőcímét veszi elő, mivel ez van a verem tetején, és erre a címre ugrik.

A BASIC rutinokat hívó rutinnak van még egy fontos részlete. A \$8C88 címen egy JMP (\$0310) utasítás található. Ez akkor kap vezérlést, amikor a rutin felhasználói tokent talált. A token az A regiszterben van.

Saját rutinaink hívásához létre kell hozni az interpreterben találhatóhoz hasonló címtáblázatot saját rutinaink kezdőcímeinek felsorolására. Ezután a \$0310 vektort a következő rutin elejére állítva már használhatók is új utasításaink:

SEC

SBC #\$80

ASL A

LDA címtáblázat kezdete + 1, Y

PHA

LDA címtáblázat kezdete, Y

PHA

JMP \$0473

A rutin nagyon hasonlít az interpreterben találhatóhoz, mindössze a BCC utasítás maradt ki. Erre nincs szükség, mert itt nem fordulhat elő LET utasítás.

Ahhoz, hogy az interpreter hiba nélkül értelmezze a BASIC programot, a rutinnak RTS utasítással kell befejeződnie, és a \$3B-3C (TXTPTR) mutatónak az utasítást követő \$00 (sorvég) vagy \$3A (kettőspont) tartalmú bájtra kell mutatnia.

GNÁDIG PÉTER

Plus/4, C16, C116

A hatékony programfejlesztés előfeltételei az olyan segédprogramok, amelyek ezt a munkát segítik. A PLUS-Comp egy BASIC fordítóprogram, amely „alulról” kompatibilis a 3.5-ös BASIC verzióval. A programrendszer a B—G—S Comp (P) fordítóprogram család legfiatalabb tagja. Gyorsaságát és üzemmódjainak számát tekintve felülmúl minden Commodore 64-es fordítót.

A BASIC programok fordítását Plus/4-es és C64-es mikrogépen lehet elvégezni. Mindkét gépen előállítható C16-oson futtatható gépi kódú program. Ezek a C16-os programok nem tartalmazhatnak GRAPHIC utasítást, mivel a P kódú program memóriaszükséglete egysoros BASIC program esetén is minimálisan 7,25 kb-ot (29 blokk).

A fordítás eszköze, melyre a fordítóprogram dolgozik, mágneslemez. A kicsit később részletezett lehetséges 4 fordítási üzemmód remélhetőleg a fejlesztőknek és felhasználóknak egyaránt minden kényelmet megad.

A program hardverszükséglete:

- 1 db Plus/4 vagy C64 mikrogép
- 1 vagy 2 db mágneslemez-meghajtó (VC—1541 vagy vele kompatibilis meghajtó),
- 1 db MPS 801-es mátrixnyomtató vagy vele kompatibilis nyomtató, ha a hibaüzenetek megjelenítésére papíron is van igény.

A fordítás közbeni hibánál a hibás sor számát a hibaüzenettel együtt kiírja a képernyőre és a nyomtatóra. A hibaüzenetek magyar nyelvűek.

- SYNTAX ERROR = szintakszis hiba
- TYPE MISMATCH = hibás karakter típus
- UNDEF'D STATEMENT = definiálatlan sorszám
- BAD SUBSCRIPT = hibás leírás
- OVERFLOW = túlszordulás

A fordító INTEGER aritmetikával dolgozik. A valós számokat egész típusúként kezeli, ha a konverzió elvégezhető. Más valós konstansok esetében a P kódú programban lebegőpontos alakban szerepelnek a számok, így a számítási műveletek gyorsabb végrehajtása válik lehetővé. FOR... NEXT ciklus ciklusváltozójaként egész típusú változót használva az utasítás verem-szükséglete 9 bájtra csökken. A változók közvetlen elérésűek, és a vezérlésátadó utasítások közvetlen végrehajtásúak, vagyis közvetlenül az utasítás memóriacímére kerül a vezérlés.

A fordítást egy vagy két meghajtón lehet végrehajtani. Egy meghajtó esetén csak az 1-es üzemmód használható, két meghajtónál automatikus programcsomag-fordításra van lehetőség, és kihasználható az átlapoló (overlay) technika előnye is (2,3,4-es üzemmód).

- A fordító négyféle üzemmódot kínál:
- 1 — Egy program fordítása a 8-as mágneslemez-meghajtón.
 - 2 — Programcsomag automatikus fordítása a 8-as egységben lévő lemezezől a

Egy új BASIC fordítóprogram: a PLUS-Comp

```

10 REM * BENCHMARKOK
15 REM -----
20 S1$="000000"
30 TI$=S1$:S1=TI
40 FORI=1TO10000
50 NEXTI
60 PRINT(TI-S1)/60" SEC"
70 REM -----
80 TI$=S1$:S1=TI
90 FORIX=1TO10000
100 NEXT
110 PRINT(TI-S1)/60" SEC"
120 REM -----
130 TI$=S1$:S1=TI
140 K=0
150 K=K+1
160 IFK<10000THEN150
170 PRINT(TI-S1)/60" SEC"
180 REM -----
190 TI$=S1$:S1=TI
200 K=0
210 K=K+1
220 A=K/K#K+K-K
230 IFK<10000THEN210
240 PRINT(TI-S1)/60" SEC"
250 REM -----
260 TI$=S1$:S1=TI
270 K=0
280 K=K+1
290 A=K/2#3+4-5
300 IFK<10000THEN280
310 PRINT(TI-S1)/60" SEC"
320 REM -----
330 TI$=S1$:S1=TI
340 K=0
350 K=K+1
360 A=K/2#3+4-5
370 GOSUB400
380 IFK<10000THEN350
390 PRINT(TI-S1)/60" SEC"
395 GOTO420
400 RETURN
410 REM -----
420 TI$=S1$:S1=TI
430 K=0
440 DIM M(5)
450 K=K+1
460 A=K/2#3+4-5
470 GOSUB400
480 FORL=1TO5
490 NEXTL
500 IFK<10000THEN450
510 PRINT(TI-S1)/60" SEC"
520 REM -----
530 TI$=S1$:S1=TI
540 K=0
550 DIM M1(5)
560 K=K+1
570 A=K/2#3+4-5
580 GOSUB400
590 FORL=1TO5
595 M1(L)=A
600 NEXTL
610 IFK<10000THEN560
620 PRINT(TI-S1)/60" SEC"
630 REM -----
640 TI$=S1$:S1=TI
650 K=0
660 K=K+1
670 A=K#2
680 B=LOG(K)
690 C=SIN(K)
700 IFK<10000THEN660
710 PRINT(TI-S1)/60" SEC"
720 REM -----
    
```

lemezen szereplő sorrendben, és a lefordított programok a 9-es egység lemezére íródnak.

- 3 — Automatikus programcsomag-fordítás MENÜ technikával. Ugyanaz, mint a 2-es, azzal a különbséggel, hogy csak az első programhoz lesz hozzáfűzve a gépi kódú RUN-TIME rutin, a többi program erre töltődik rá.
- 4 — Automatikus fordítás, átlapoló technikával. Ugyanaz, mint a 3-as, azzal a különbséggel, hogy az első program tartalmazza az összes program változóit, így a következő programok átveszik egymástól a változókat és a tömbökben lévő értékeket.

A BASIC program fordításának 3 menete van: A, B, C. Az A menet csak átlapoló technika használatok hajtódik végre. A B menetben készülnek el azok a munkafájlok, amelyeket a C menetben fűz össze a fordító.

Különleges az az eset, amikor grafikus utasításokat kívánunk használni, mivel a GRAPHIC utasítás áthelyezi a BASIC programot \$1000-ról \$4000-re. Ilyen programok fordítására szolgál a GRAFIKUS fordítási mód. Ekkor a lefordított gépi kódú program közvetlenül \$4000-re kell tölteni.

A fordító „alulról” kompatibilis a BASIC 3.5-ös verzióval, mivel a fordítás elve a 2.0-ás verzióra készült. Ennek az a következménye, hogy bizonyos utasításoknál jelezni kell a fordítónak, hogy a 3.5-ös BASIC utasítás következik. A legtöbb esetben automatikus az utasítások (3.5) felismerése, de néhányat ki kell egészíteni fordítási direktívával.

Háromfajta direktívából lehet választani. A különbség a direktívák érvényességi határaiban van: meddig érvényesül a direktíva által megjelölt különleges kezelési mód.

- :: = egy utasítás
- ← = utasítássor
- #be } = programmodul
- #ki }

A fordító elfogadja a saját bővítéseket is, de ugyanúgy kell eljárni ekkor is, mint 3.5-ös utasítások esetében.

1986 szeptembere óta kapható a NOVOTRADE RT 2C Számítástechnikai Áruházában a B-G-S Comp (P) fordítóprogram család C64-re. A B,G,S betűk jelentése:

- B = BASIC
- G = Supergraphik 64
- S = Simon's BASIC

programok fordítására alkalmas fordítóprogramok egymeghajtós, illetve kétmeghajtós kivételben. A kétmeghajtós kivétel (P) alkalmas a fent ismertetett négy fordítási üzemmód végrehajtására a Commodore 64-esen is. A háromféle fordítási direktíva itt is használható, mivel alkalmazásukkal memória- és végrehajtsiidő-csökkenést lehet elérni.

Az S-Comp (P) nevű Simon's BASIC fordító többet tud, mint az általánosan elterjedt SIMON'S-COMP (Datatronic Wien) fordító mindkét verziója. A C/prg. nevű P kódú programok ugyanis nagy felbontású grafika használatok nem tudják végrehajtani a karakteres képernyőre való visszacsatolást, mert ? NEXT WITHOUT FOR ERROR hibajelzéssel megállnak. Ez a hiba az S-Comp-pal lefordított programoknál nem lép fel. Az S-Comp lehetővé teszi sprite-ok, illetve a DESIGN utasítás használatát is, ami a SIMON'S-COMP-nál teljesen lehetetlen.

A legismertebb fordítókat egy rövid programmal (lásd a listát) teszteltem. A BASIC és P kódú programokat futtatva a táblázat a szerinti időket mértem. A legjobb átlagot a PLUS-Comp érte el, túlszárnyalva az összes C64-es fordítóprogramot.

BÁRÓ CSABA

BASIC BENCHMARKOK

	BASIC		P - kód					
	C-64	+4	Aus.-Comp	Aus.-Speed	Pet-Speed	B-Comp	P	+ Comp
* 1 *	14.8	18.9	12.33	11.81	2.86	11.75	12.96	
* 2 *	-	-	2.90	1.93	2.26	1.82	1.58	
* 3 *	116.5	127.8	19.41	13.10	7.00	12.00	10.85	
* 4 *	213.5	235.0	71.75	55.21	60.53	50.77	45.75	
* 5 *	228.8	237.1	77.45	65.28	74.00	63.12	53.96	
* 6 *	248.9	271.3	79.52	66.80	75.71	64.09	55.23	
* 7 *	386.9	482.0	150.75	131.28	95.16	127.35	125.20	
* 8 *	593.5	722.5	217.45	170.55	120.08	162.98	161.71	
* 9 *	717.3	703.2	532.46	516.98	515.05	500.75	432.96	
* 10 *	315	349.7	129.3	114.7	105.8	110.5	100.0	
* RUN-TIME (blokk) *	18	26	35	27	31			

Gépi kódban



A Primo szoftverkönyve szerint a 00029 címen kezdődő szubrutin figyelni a billentyűket. A megérintett billentyűhöz „tartozó” „ASC” kódot beírja az „A” regiszterbe. A „G5” jelű programban a korábban ismerttetett utasításokon kívül a „CP B” utasítást használjuk, amely a „B” regiszter tartalmát hasonlítja össze az „A” regiszter tartalmával. Az összehasonlítás eredményétől függően a „JR Z” utasítás vagy a következő utasításon, vagy az utána lévő értéknek megfelelő ugrással engedi folytatni a program futását. A BASIC programnak az adatot az „LD(HL)” utasítással adjuk át.

Próbáljuk ki, hogy milyen hatással van a program működésére az „LD(HL),1” utasítás! Azokon a gépeken, amelyekhez nem kapcsolható botkormány, a játékokat billentyűről kell irányítani. A játék hevében gyakran előfordul, hogy a vezérlő billentyű mellé nyúlunk. Célszerű lenne több billentyű ellenőrzésével vezérelni a játékot, de ez erősen lassítaná a program futását. A „G6” jelű program egy „*” mozgását vezérli a képernyőn jobbra és balra, a rajzon látható 10 billentyű ellenőrzésével. A feladat elvégzéséhez szükséges idő közel azonos a 2 billentyűvel, „BASIC”-ből vezérelt program működési idejével.

A programban lévő „CP” utasítással az utána lévő értéket hasonlítjuk össze az „A” regiszter tartalmával. A „JR NZ” utasítások a „JR Z” utasításhoz hasonló módon működnek. A vezérlő billentyűk a megfelelő „ASC” kódok átírásával megváltoztathatók.

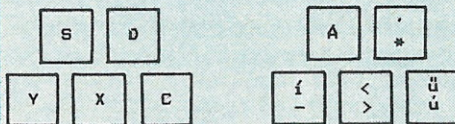
SOMOGYI GYÖRGY

```

1 REM G5
100 C=0 : D=0
105 CLS
110 DIM K%(20)
120 C=VARPTR(K%(0))
140 POKEC,63, 205,29,0,6,0,54,3,184,
    40,1,201,54,2,201
150 D=CALL(C)
160 D=PEEK(D)
170 ON (D-1) GOTO 190,200
190 CLS
195 GOTO 150
200 PRINT "MOST MEGÉRINTETTED."
205 GOTO 150
    
```

A PROGRAM	A KOD
CCF	63
CALL,29	205,29,0
LD B,0	6,0
LD(HL),3	54,3
CP B	184
JR Z,1	40,1
RET	201
LD(HL),2	54,2
RET	201

A PROGRAM	A KOD	MEGJ.
CCF	63	.
CALL,29	205,29,0	.
LD(HL),4	54,4	.D=4
CP,89	254,89	.ASC("Y")
JR NZ,2	32,2	.
LD(HL),2	54,2	.D=2
CP,88	254,88	.ASC("X")
JR NZ,2	32,2	.
LD(HL),2	54,2	.D=2
CP,67	254,67	.ASC("C")
JR NZ,2	32,2	.
LD(HL),2	54,2	.D=2
CP,83	254,83	.ASC("S")
JR NZ,2	32,2	.
LD(HL),2	54,2	.D=2
CP,68	254,68	.ASC("D")
JR NZ,2	32,2	.
LD(HL),2	54,2	.D=2
CP,93	254,93	.ASC("A")
JR NZ,2	32,2	.
LD(HL),3	54,3	.D=3
CP,42	254,42	.ASC("*")
JR NZ,2	32,2	.
LD(HL),3	54,3	.D=3
CP,95	254,95	.ASC("ú")
JR NZ,2	32,2	.
LD(HL),3	54,3	.D=3
CP,62	254,62	.ASC(">")
JR NZ,2	32,2	.
LD(HL),3	54,3	.D=3
CP,45	254,45	.ASC("-")
JR NZ,2	32,2	.
LD(HL),3	54,3	.D=3
RET	201	.



```

1 REM G6
100 C=0 : D=0 : E=0 : I=0 : V=0
105 CLS
110 DIM K%(20)
120 C=VARPTR(K%(0)) : I=C-1
130 I=I+1 : READ E : IF E>400 THEN 150
132 POKEI,E : GOTO 130
135 DATA 63, 205,29,0,54,4
136 DATA 254,89,32,2,54,2 : '89=ASC("Y")
137 DATA 254,88,32,2,54,2 : '88=ASC("X")
138 DATA 254,67,32,2,54,2 : '67=ASC("C")
139 DATA 254,83,32,2,54,2 : '83=ASC("S")
140 DATA 254,68,32,2,54,2 : '68=ASC("D")
141 DATA 254,93,32,2,54,3 : '93=ASC("A")
142 DATA 254,42,32,2,54,3 : '42=ASC("*")
143 DATA 254,95,32,2,54,3 : '95=ASC("ú")
144 DATA 254,62,32,2,54,3 : '62=ASC(">")
145 DATA 254,45,32,2,54,3 : '45=ASC("-")
149 DATA 201,500
150 D=CALL(C)
160 D=PEEK(D)
170 ON (D-1) GOTO 190,200,210
190 IF V>1 THEN V=V-1
195 GOTO 210
200 IF V<40 THEN V=V+1
210 CLS : PRINT#5,V,"*"
225 GOTO 150
    
```

A Primo gépek megszakítása BASIC-ből is letiltható. Sokszor olvashattuk, hogy a BEEP utasítás használatakor kapcsoljuk ki a 16443 cím 7. bitjét, mert ezáltal szebb hangokat fog gépünk kiadni. Mi is ez a 7. bit?

A Primo két megszakítástípust ismer: az NMI-t és az INT-et. Az INT-tel itt most nem foglalkozom részletesen, elég annyit tudnunk róla, hogy ha a bemeneti bit (ez magnóról érkező bit) 0-ba megy át, akkor egy RST 38 utasítás hajtódik végre. Számunkra a másik megszakítás lesz fontos. Ehhez tartozik a 7. bit.

A gép 20 ms-onként kioltja a képet, hogy utána újra felírja a tv-képre a képernyő-memória tartalmát. Ha ugyanakkor, amikor a képkioltás jele elindul, a 16443 cím 7. bitje magas, egy NMI megszakítás keletkezik, amely egyenértékű a mi esetünkben egy JP 16408 (tizes számrendszerben) utasítással. Ezen a 16408-9-10 címsoron egy másik JP 0015 (hexa) utasítás van: ennek a rutinnak a ROM-beli címe, amely lépteti a belső órát és kezeli a NMI-t.

Most néhány szót arról, hogy mi is az a 16443 cím. Valójában nem elég, ha csak itt állítjuk át azt a bizonyos bitet, hanem még az is fontos, hogy ennek a memóriacímnek a tartalmát egy OUT utasítással kiküldjük valamelyik 0-63 portra. Lényegében az is elég, ha az OUT helyett megérintünk egy billentyűt. Az OUT akkor kell, ha a programból végezzük a ki-be kapcsolást. A Primo a hardverből nem tud visszaolvasni, ezért szükséges ez a cím. Itt tartja számon a gép, hogy legutoljára mit küldött ki az adott porton.

Vegyünk egy konkrét alkalmazást. Készítsünk olyan programot, amely egy mánot kezel a képen. A gépi programot az 1. lista tartalmazza.

A program lehetőségei

A program egy ábrát mozgat fix háttér előtt úgy, hogy nem törli azt. Vizszintesen karakteresen, függőlegesen bitenként mozgatja az ábrát. Ha szükséges, saját magát is rajta hagyja a háttéren, amerre halad. Állítható a függőleges hossz és a háttérszellem prioritása.

A szellem bekapcsolása a következő BASIC utasítással történik:
 POKE16443,109:OUT0,109:
 POKE16409,238,67:
 POKE16443,237:OUT0,237

Megszakítás

Kissé bonyolultnak tűnik, de nem érdemes spórolni, mert elszállhat az esetleges BASIC listánk.

A listáról látható, hogy elég keményen a BASIC helyére töltődik a program. Ezért begépelés után vegyük fel a következő adatokkal:

Kezdőcím; 43EE, Végcím; 4475, Indító-cím: 4466; Az összes cím hexadecimális.

Az így felvett program betöltés után törli a képernyőt, s hátrább teszi a BASIC kezdőcímét. Bizonyára többeknek feltűnt, hogy a program ciklusokkal sok helyen rövidíthető lett volna. Ezt azért nem tettük meg, mert a ciklusokon a gép tovább dolgozik, még gépi kódban is. Erre nagyon ügyeljünk, főleg ha minden 20 ms-ban ide kerül a vezérlés. Nem lenne jó, ha végtelen megszakításba torkollna a program!

Használata

Állítsuk be a megfelelő adatokat a következő címeken: 17526-27 címre kell POKE-olni a sprite memóriacímét, ahonnan kezdve a program kirajzolja azt. A képernyőcímet adjuk meg, ne a raszterpontcímet. Tehát ha a képernyő bal felső sarkába tervezük, akkor ezt írjuk be:

```
POKE17526,0,PEEK(16458)
```

A 17528-29 címekre az a cím kerül, ahonnan kezdve a szellem formája a tárban van. Például ha a 28 672 címtől tettük le a formát, akkor

```
POKE17528,0,112
```

A 17525-ös cím szerepe is fontos. Ha itt 0 van, akkor törli a hátteret, s a szellem marad ott. Ha ilyenkor módosítjuk a szellem helyét, akkor az odébb kerül, de a formája (nyoma) ott marad. Ha azonban nem 0 van ezen a címen, akkor átsiklik a háttér felett.

Fontos! Ha már a helyén levő szellemre akarunk valamit írni, s utána elveszünk onnan a szellemet, akkor is az a háttér marad ott, amely a szellem odatétele előtt volt. Ily módon egy kis ablakot lehet létrehozni a képernyőn.

A 17432,17461 és a 17489 címeken a szellem függőleges méretét állítjuk. Mindhárom címnek egyforma értéket kell képviselnie. Jelenleg ezen a címen 21 van. Egyébként az egész szellem mérete 24 x 21 raszterpont.

Hasznos még a 17434, 17439 és a 17444 cím is. Itt megszakhatjuk, hogy a szellem melyik harmada legyen a háttér előtt vagy mögött, külön-külön. Ha ezeken a címeken

0 van, akkor a szellem látszik, ha ide 182 kerül, akkor pedig a háttér (pontosabban együttlátás van).

Bátran próbálkozzunk! A program legfeljebb akkor száll el, ha nem jól kapcsoljuk be, vagy rossz helyre tesszük a szellemet. Bármikor újraindíthatjuk viszont egy PRINT CALL (17510) utasítással.

Ajánlom a példaprogramok kipróbálását is (2., 3. és 4. lista).

1. lista

```
ORG 43EE
LOAD $
PUSH AF
IN A,(00)
AND 20
JP Z,4430
EXX
LD A,(4475)
OR A
CALL NZ,444A
LD A,01
LD (4475),A
LD HL,(4476)
LD (447A),HL
LD DE,447C
CALL 4434
LD HL,(4478)
LD DE,(4476)
EX DE,HL
LD B,15
LD A,(DE)
OR (HL)
LD (HL),A
INC HL
INC DE
LD A,(DE)
OR (HL)
LD (HL),A
INC HL
INC DE
LD A,(DE)
OR (HL)
LD (HL),A
INC HL
INC DE
LD A,(DE)
OR (HL)
LD (HL),A
INC HL
INC DE
LD A,(DE)
OR (HL)
LD (HL),A
PUSH BC
LD B,01
CALL 4440
POP BC
DJNZ 4419
EXX
POP AF
JP 0057
LD B,15
LD A,(HL)
LD (DE),A
INC HL
INC DE
```

```
LD A,(HL)
LD (DE),A
INC HL
INC DE
LD A,(HL)
LD (DE),A
INC DE
PUSH DE
LD DE,001E
ADD HL,DE
POP DE
DJNZ 4436
RET
LD DE,447C
LD HL,(447A)
LD B,15
LD A,(DE)
LD (HL),A
INC HL
INC DE
LD A,(DE)
LD (HL),A
INC HL
INC DE
LD A,(DE)
LD (HL),A
PUSH BC
LD B,01
CALL 4440
POP BC
DJNZ 4452
RET
START: LD HL,44C2
LD (40A4),HL
DEC HL
LD (HL),0
CALL 1B4A
JP 1A19
```

2. lista

```
20 POKE17528,0,112
30 POKE17432,5
40 POKE17461,5
50 POKE17489,5:POKE17526,0,0:
POKE 17525,0
60 POKE28672,119,81,120,85,91,72,119,95
72,70,85,72,69,81,120
70 POKE16443,109:OUT0,109:
POKE16409,238,67: POKE16443,237:
OUT0,237
80 P=PEEK(16458)
90 FORR=P*256 TO P*256+6143
STEP 32
100 H=INT(R/256):L=R-H*256
110 POKE17526,L,H
120 NEXT
130 POKE17526,15,P
```


3. lista

Az előző programból töröljük a 80—130-as sorokat és írjuk be:

```
80 A=RND(6144)-1:P=PEEK(16458)
90 E=INT(A/256)
100 L=A-E*256
110 POKE17526,L,P,+E
120 GOTO 80
```

4. lista

Ez a gépi kódú program a nyilakkal mozgatja a szellemet. Bárhova be lehet írni a táriba, ahol van hely.

```
LD H,(IX+8)
LD L,0
LD (17526T),HL
LD (min),HL
LD, DE,6143T
ADD HL,DE
INC H
LD A,H
LD (max),A
LD HL,17390T
LD (16409T),HL
LD HL,7000
LD DE,0F
LD B,E
C0 :LD A,(HL)
PUSH HL
ADD HL,DE
LD (HL),A
POP HL
INC HL
DJNZ C0
ELE :LD HL,(17526T)
LD A,(17528T)
LD B,A
LD A,0F
SUB B
LD (17528T),A
IN A,(0;)
RRA
JR NC,LE
LD DE,20
OR A
SBC HL,DE
FEI :LD DE,(min)
RST 18
JR C,ELE
LD (17526T),HL
LE :IN A,(0F)
RRA
JR NC,BA
LD DE,20
ADD HL,DE
LEI :LD A,(max)
XOR H
JR Z,ELE
LD (17526T),HL
BA :IN A,(39)
RRA
JR NC,JO
DEC HL
JR FEI
JO :IN A,(3D)
RRA
INC HL
JR FEI
min :DW 0
max :DB 0
```

FEHÉR CSABA

Eljárásdefiniálás

A strukturált programozás elengedhetetlen kelléke a névvel hívható szubrutin, de ezt még a C16 egyébként kiváló BASIC-je sem ismeri. Az alábbi program e nehézség áthidalására készült. Néhány szó a program használatáról.

A definíció szintaxisa:

sorsz1 procedure <név>

sorsz2 enddef

A hívás szintaxisa:

sorsz3 <név>

A definíciónak mindig meg kell előznie az első hívást!

Ha a főprogramot vagy a szubrutinokat javítottuk, akkor a szubrutinok elhelyezkedése a táriban megváltozik, azaz első utasí-

tásaik más címre kerülnek, és a gép által az előző futáskor megjegyzett cím nem a szubrutin első utasítására mutat. Ilyenkor csak sorszám nélküli RUN paranccsal indítsuk a programunkat; a RUN parancs törli az előző futáskor megjegyzett definíciókat.

A hívással azonos sorban a hívás után olyan GOTO vagy GOSUB utasítás nem állhat, amely a hívás sorszámánál kisebb sorszámú sorra ugrik, mert ez UNDEF'D STATEMENT ERROR-t eredményezne.

A „név”-vel egyező nevű változóval ne dolgozzunk!

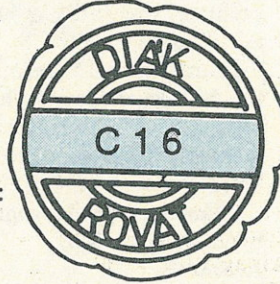
A töltőprogram beírása előtt írjuk be a következő parancsokat:

```
POKE43,1:POKE44,32:POKE
8192,0:NEW
```

A töltőprogramot elindítás előtt mentsük ki a háttértárolóra. Ha a futtatás során a töltőprogram nem jelzett adathibát, akkor monitor üzemmódból a következő paranccsal menthetjük

— magnóra : S "procedure",1,1001,127f
— floppyra : S "procedure",8,1001,127f

A RESET gomb megnyomása és a program visszatöltése után az itt közölt BASIC-bővítés a RUN parancsra indul. G. P.



```
10 rem *****
20 rem *
30 rem * procedure *
40 rem *
50 rem *****
500 x=4096
510 v=4736
515 a=v-x:v=x+8*(int((a-1)/8)+1)
520 c=0
530 for p=x to v-1
540 read a$:a=dec(a$)
550 poke p,a:c=c+a
560 next p
570 read a
580 if a<c then print "hiba !!!"
:else print "ok !!!"
590 end

10000 rem
10001 data 00,0e,10,c2,07,9e,20,28
10002 data 34,31,31,32,29,00,00,00
10003 data a9,53,a0,10,8d,0c,03,8c
10004 data 0d,03,a9,67,a0,10,8d,0e
10005 data 03,8c,0f,03,a9,75,a0,10
10006 data 8d,10,03,8c,11,03,a9,d7
10007 data a0,10,8d,08,03,8c,09,03
10008 data a9,01,85,2b,a9,20,85,2c
10009 data a9,00,8d,00,20,8d,01,20
10010 data 8d,02,20,85,81,20,79,8a
10011 data 4c,d3,8b,48,a9,12,a0,4f
10012 data 20,07,8a,68,90,06,a5,0b
10013 data 48,4c,d6,89,4c,6c,89,84
10014 data 49,aa,a0,12,84,23,a0,4f
10015 data 84,22,4c,9e,8b,38,e9,80
10016 data 0a,a8,b9,78,12,48,b9,77
10017 data 12,48,4c,73,04,24,81,30
10018 data 03,4c,0d,12,a0,00,c9,00
10019 data f0,10,c9,3a,f0,0c,91,d0
10020 data c8,84,22,20,73,04,a4,22
10021 data d0,ec,88,b1,d0,09,80,91
10022 data d0,c8,a5,3b,91,d0,c8,a5
10023 data 3c,91,d0,c8,a9,00,91,d0
10024 data 88,98,65,d0,85,d0,90,02
10025 data e6,d1,a9,80,85,d7,38,a5
10026 data d0,e5,d2,a5,d1,e5,d3,10
10027 data 03,4c,79,04,4c,fd,11,20
10028 data 73,04,f0,0d,24,d7,30,0c
10029 data 10,3f,ea,20,79,04,4c,d9
10030 data 8b,4c,dc,8b,c9,fe,d0,18
10031 data a0,01,20,a5,04,c9,81,d0

10032 data 0f,85,d7,a9,00,85,d7,20
10033 data 73,04,20,73,04,18,90,db
10034 data a0,ff,c8,b1,3b,c9,00,d0
10035 data f9,88,98,65,3b,85,3b,90
10036 data 02,e6,3c,20,79,04,18,90
10037 data c8,2c,eb,02,10,13,24,81
10038 data 10,0f,48,a9,5b,20,b2,90
10039 data 20,5b,a4,a9,5d,20,b2,90
10040 data 68,c9,8a,d0,06,20,1d,12
10041 data 4c,bc,8b,c9,80,10,04,c9
10042 data 5a,30,06,20,3f,8c,4c,dc
10043 data 8b,c9,41,30,f6,a9,12,a0
10044 data 81,85,23,84,22,a0,00,88
10045 data c8,20,a5,04,c9,00,f0,09
10046 data 38,f1,22,f0,f3,c9,80,f0
10047 data 1c,b1,22,30,03,c8,d0,f9
10048 data c8,c8,c8,18,98,65,22,85
10049 data 22,90,02,e6,23,a0,00,b1
10050 data 22,d0,d5,f0,be,98,65,3b
10051 data 85,3b,90,02,e6,3c,c8,b1
10052 data 22,85,d4,c8,b1,22,85,d5
10053 data a0,00,a5,3b,91,d2,20,ed
10054 data 11,a5,3c,91,d2,20,ed,11
10055 data a5,d4,85,3b,a5,d5,85,3c
10056 data 4c,dc,8b,24,81,30,03,4c
10057 data 0d,12,a0,00,20,d5,11,b1
10058 data d2,85,3c,d2,d5,11,b1,d2
10059 data 85,3b,4c,79,04,e6,d2,d0
10060 data 02,e6,d3,a9,ff,c5,d2,d0
10061 data 0b,a9,1f,c5,d3,d0,05,a9
10062 data 00,4c,35,12,60,c6,d2,a5
10063 data d2,c9,ff,d0,02,c6,d3,20
10064 data 02,12,f0,01,60,68,68,4c
10065 data 81,86,a5,d2,c5,d0,04
10066 data a5,d3,c5,d1,60,a2,15,68
10067 data 68,4c,83,86,20,1d,12,20
10068 data 79,04,4c,bc,8b,a9,81,a0
10069 data 12,85,d0,84,d1,a9,fe,a0
10070 data 1f,85,d2,84,d3,a9,00,85
10071 data d7,a8,91,d0,60,48,a9,00
10072 data 85,83,20,c9,c7,68,aa,a9
10073 data 62,85,24,a9,12,85,25,a0
10074 data 00,20,5e,86,4c,d3,86,50
10075 data 52,4f,43,45,44,55,52,c5
10076 data 45,4e,44,44,45,c6,52,55
10077 data ce,00,50,52,4f,43,45,44
10078 data 55,52,45,20,4e,4f,54,20
10079 data 43,41,4c,4c,45,c4,00,84
10080 data 10,ba,11,13,12,00,00,00
10081 data 70182
```




Betűszonglörködés

Régi olvasója vagyok a Mikroszámítógép Magazinak, azonkívül Primo-felhasználó. Így érhető, hogy leginkább a Primóval kapcsolatos cikkeket és programokat figyeltem, böngésztem a lapban. Mivel eddig nem sokat találtam, úgy gondoltam, legalább én megosztom jó, vagy általam jónak tartott ötleteimet, programjaimat a többi Primo-felhasználóval.

Nekem is sok gondot okozott a programok készítésekor a díszítő feliratozás. Bár a Primón sokféle karakterformáció jeleníthető meg, ez mégsem mindig elegendő és megfelelő egy program feliratozásához. Ez indított arra, hogy másfajta karaktereket kiíró szubrutint csináljak. Így jutottam el az 1. listán látható programhoz, mely a következők alapján működik.

A megadott karakter karaktergenerátortábla címéről átmásolja a lerakott bájtokat (amelyek a karakter képét határozzák meg) az A változóba, majd ezeket a bájtokat nagyítja karakter méretűre: egy bit helyére egy karaktert ír ki a képernyőre. Így tehát ugyanazt a karaktert kapjuk nagyban, mint ami kicsiben volt.

Ezzel a szubrutinnal azonban csak hét karakternyi szöveget nagyíthatunk, illetve írhatunk ki a képernyőre, ráadásul kö-

tött méretben (6*8 karakter). Ezután olyan szubrutin írásába fogtam, amellyel tetszőleges méretű és többkarakteres szöveg is kiírható fordított, illetve normál állásban.

A 2. lista szerinti program a megfelelő bájtokat (lásd az előző leírást) szintén az A változóba másolja, majd egy bit helyére a nagyítás mértékének megfelelő számú grafikus pontot rak. Minél nagyobb mértékű a nagyítás, annál kevesebb karakter írható ki. A vízszintes irányú nagyítás legfeljebb 40-szeres, a függőleges pedig legfeljebb 22-szeres (ez esetben azonban csak egy karakter írható ki). Lehetőség van negatív nagyítási érték beírására is, akkor a kinagyított karakter állása fordított lesz. Ezenkívül megoldható a kiírt szöveg pozicionálása is, erre azonban fordított állású karaktereknél figyelni kell; X irányú tükrözésnél érdemes az X koordinátáját 255-nek vagy egy ahhoz közel álló értéknek venni, különben kiszalad a képernyőről és hibát jelez.

Mivel ezek a szubrutinok BASIC nyelven készültek, inkább csak alapot adnak a gépi kódra való átültetéshez. A komolyabb programokhoz a gépi kódú változatot célszerű választani.

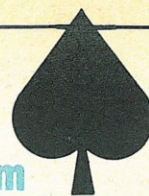
NÓGRÁDI ELŐD

```
10 CLS:Y=1:X=0:T=0
20 BC=12791
30 INPUT A$
40 FOR Z=1 TO LEN(A$)
50 B$=MID$(A$,Z,1):CIM=BC+(ASC(B$)-30)*8
60 FOR K=1 TO 8
70 A=PEEK(CIM)
80 A=A*ABS(AC255)
90 FOR I=6 TO 1 STEP -1
100 B=A AND 2*I
110 IF B<>0 THEN PRINT$Y,X,CHR$(4) "CHR$(20); ELSE PRINT$Y,X," ";
120 X=X+1
130 NEXT I
140 CIM=CIM+1:Y=Y+1:X=X+6:NEXT K
150 T=T+6:X=T:Y=1:NEXT Z
```

1. lista

2. lista

```
10 CLS
20 CLEAR 1000
30 BC=12791
40 INPUT A$,XN,YN,KX,KY
50 IF XN<40 OR YN<22 THEN 30
60 IF XN<40 OR YN<22 THEN 30
70 IF XN<0 THEN H1=-1 ELSE H1=1
80 IF YN<0 THEN H2=-1 ELSE H2=1
90 X=KX+ABS(XN):Y=KY+ABS(YN):T=XN
100 CLS
110 FOR Z=1 TO LEN(A$)
120 B$=MID$(A$,Z,1):CIM=BC+(ASC(B$)-30)*8
130 FOR K=1 TO 8
140 A=PEEK(CIM)
150 A=A*ABS(AC255)
160 FOR I=6 TO 1 STEP -1
170 B=A AND 2*I
180 GOSUB 240
190 X=X+XN
200 NEXT I
210 CIM=CIM+1:Y=Y+YN:X=X+T:NEXT K
220 T=T+XN*6:X=KX+XN:Y=KY+ABS(YN):NEXT Z
230 END
240 FOR Q1=0 TO XN STEP H1
250 FOR Q2=0 TO YN STEP H2
260 IF B=0 THEN RETURN
270 SET(X+Q1,191-Y):SET(X+Q1,191-Y+Q2)
280 NEXT Q2,Q1
290 RETURN
300 END
```



Program Információs Központ középiskolásoknak

A KISZ KB Középiskolai és Szakmunkástanulók Tanácsa program információs központot hozott létre, melynek célja a középiskolák oktatási intézményekben fejlesztett programok elterjesztésének, cseréjének megkönnyítése.

Az információs központ a programok rövid tartalmi ismertetése mellett közli a programfejlesztő nevét és címét is, megkönnyítve ezzel a kapcsolatfelvételt.

Azoknak, akik kérik programjuk felvételét katalógusunkba, térítésmentesen megküldjük az eddig rendelkezésükre álló programok jegyzékét.

A programoktól nem kívánunk profi színvonalat, de szeretnénk meggyőződni használhatóságukról, és adott esetben bemutatni azokat különböző rendezvényeken, ezért kérjük, hogy a programokat az alábbi szempontok szerint küldjék be címünkre:

A programfejlesztő(k) neve

Iskola

Levelezési cím

A program neve

A program „műfaja”

1. oktatást segítő (milyen tantárgyban?)
2. zenei
3. programozást segítő
4. logikai játék
5. ügyességi játék
6. adminisztráció
7. egyéb

Géptípus

Programnyelv

A program rövid leírása

Megjegyzés

Olvasható aláírás

Cím:

KISZ KB KSZTT
„PIKK”
Budapest
Pf. 72.
1388

KISZ KB
Középiskolai
és Szakmunkástanulók
Tanácsa

Távkonferencia I.

Távkonferencián eredetileg zártkörű központi rendezvénynek (értekezlet, kongresszus) a hírhálózat segítségével történő „kibővítését” értik. A rendszerben álló- vagy mozgóképet, hangot, illetve adatokat továbbítanak: a kapcsolatteremtés jellege egy- vagy kétirányú. Az utóbbi esetben az esemény színhelyétől távoli konferenciatermekben — akár más-más városokban is — helyet foglaló hallgatóság látja, hallja a központi rendezvény történéseit, és kérdéseket is tehet fel.

Az egyirányú átvitelrel működő, ún. videokonferencia-rendszerekben erre nincs mód. Ezek között megtalálhatók az eseményeket teljes dinamikájukban közvetítő mozgóképes videokonferencia-rendszerek, melyek a konferencia eseményeiről tv-minőségű képet közvetítenek, valamint az ún. freeze frame típusú rendszerek, melyekben a továbbított kép nem folyamatos, hanem szakaszos mozgású: másodpercenként 8 állóképet továbbítanak. A fentiekben kívül működnek még olyan rendszerek is, melyekben a résztvevők nem látják, csak hallják egymást. Az ilyen megoldásokat audio-konferencia-rendszereknek nevezik.

A következő áttekintést nyújtunk az egyes rendszerek előnyeiről és hátrányairól, a távkonferenciák elterjedését előmozdító és gátló tényezőkről.

A távkonferenciák alapfeltétele a fejlett infrastruktúra, azaz a jó minőségű hírközlő hálózat: például a jól működő telefon- vagy adathálózat, rádiócsatorna, illetve műholdas átvitel.

Történeti áttekintés

Bár a távkonferenciázás kb. 15 évvel ezelőtt az elektronikai ipar egyik legnagyobb ígéretként indult, érdekes módon még ma sem tudja magának a szakmán belül az öt kétségtelenül megillető elismerést kivívni.

A távkonferenciázás irányába az első lépést az USA-ban tették meg az ATT által kifejlesztett képtelefonrendszer létrehozásával. Ez még analóg elven működő rendszer volt, amely azonban — főleg magas üzemeltetési költségei miatt — nem vált népszerűvé. A következő lépést az 1975 után létrehozott Picturephone Meeting Service jelentette, amely már digitális elven működött.

Felismerve az ilyen típusú kapcsolattartásban rejlő óriási gazdasági lehetőségeket, előbb Japán, majd Nyugat-Európa is bekapcsolódott a távkonferencia-kutatásokba. A különböző távkonferencia-fejlesztéseken és a rendszer széles körben való elterjesztésén nagyot lendített a műholdas távközlés szolgáltatással való válása.

A fejlesztések kezdetben főleg az átvitel meglehetősen magas sebességigényének (1,5 Mbps) radikális csökkentésére irányultak, elsősorban a beruházási és üzemeltetési költségek csökkentése céljából. Az új műszaki megoldások alkalmazása végül is lehetővé tette az átviteli sebesség jelentős csökkentését (56 kbps-re) amellet, hogy az átvitt kép minősége is kielégítő maradt (tömörített átvitel).

A távkonferencia-ipar jelentőségére és fejlődésének dinamikájára mi sem jellemzőbb, mint az, hogy míg 1982-ben az USA-ban az iparág forgalma csupán 100 millió dollár volt, addig az évtized végére a forgalom 214 milliárd dollárra való növekedését várják. Távkonferencia-rendszereket ma már az élet legkülönbözőbb területein alkalmaznak, egyes esetekben több, másokban kevesebb sikerrel. Az igazsághoz azonban az is hozzátartozik, hogy a sikeres alkalmazások száma kb. tízszerese a kevésbé sikereseknek.

Hol tartunk?

Hogy a távkonferenciázás mindmáig miért nem tudta mégis beváltani a vele kapcsolatos kezdeti elvárásokat, azt a szakértőlet egyik legnagyobb egyénisége, Elliot Gold elemezte. Szerinte nem a rendszerekkel van baj.

Elliot Gold korábban a NASA (az USA űrhajózási hivatala) alkalmazottja volt, és jelentős szerepet vállalt annak a távkonferencia-rendszernek a létrehozásában, amely a NASA egyes kutató részlegei között teremtett kép- és hangkapcsolatot. Gold felismerte az iparágban rejlő lehetőségeket, és önálló tanácsadó irodát nyitott, amely manapság 36 intézmény számára végez szakértést. Jelentős a publikációs tevékenysége is; olyan kiadványokat jelent meg, mint a Definitive Buyers Guide to Teleconferencing Product Servicing (Távkonferencia-berendezések és -szolgáltatások felhasználói kézikönyve) vagy a Telecommunity Review (tudósítás az újdonságokról). Gold munkastílusára jellemző, hogy üzleti utazásai során állandó telefon- és számítógépes kapcsolatot tart fent irodájával, továbbá munkájához számos olyan műszaki megoldást és szolgáltatást vesz igénybe, melyről újdonság korában éppen ő közölt tudósítást: távkonferencia- és elektronikus levelezési rendszer segítségével továbbítja például az anyagokat a kiadó és a nyomda között.

Gold elsősorban a nem kellő tájékozottságból adódó indokolatlan elvárásokkal magyarázza a sikertelenségeket.

Ami a tájékozatlanságot illeti, sajnos a köztudatban még ma is erősen tartja magát az a tévhit, hogy a távkonferencia elsődle-

ges célja a különböző rendezvényekre (konferenciákra) való utazgatások számának csökkentése. (Ezt erősítette a 70-es évek olajválsága idején felröppentett jelszó is, hogy „olaj takarítható meg az elektronika segítségével”). Márpedig azért sem lehet ez a fő szempont, mert az üzleti háttérű utazásoknak ma még csak kb. 20%-a helyettesíthető a távkonferenciával: sok esetben a személyes kapcsolatokra épülő üzleti tárgyalások — részben jellegükből, részben a hagyományokból eredeztethetően — nem nélkülözhetik a közvetlen érintkezést, mint ahogy a tévézés sem helyettesíti az olvasást.

A távkonferencia-szolgáltatás vevői körében végzett vizsgálatok alátámasztják, hogy ezt a fajta szolgáltatást ma elsősorban azok a szakemberek veszik igénybe, akik egyébként is részt vesznek a különböző rendezvényeken. A távkonferencia-szolgáltatás tehát nem helyettesíti, hanem kiegészíti a konferenciákon való jelenlétet.

Kezdetben ráadásul úgy vélték, hogy a távkonferencia-szolgáltatás elsősorban a nagy távolságú kapcsolatteremtés eszköze, de az USA 22 működő rendszeréről a közelmúltban készített felmérés ennek épp az ellenkezőjét bizonyította. A szolgáltatást főleg olyan kisebb hatáskörzetbeli — például egy városban belüli — intézmények kérik, melyek munkatársait gyakran kell különböző értekezletekre összehívni. Ilyen esetben a távkonferencia alkalmazásának célja elsősorban a zsúfolt városi közlekedésben való utazgatások kiküszöbölése. Tehát a távkonferenciázással kapcsolatos másik „tévhitet” (nagy távolságú kapcsolatteremtés) maga az élet cáfolta meg. Ennek felismerése készítette a XEROX-ot is arra, hogy nagy távolságú távkonferencia-rendszerek fejlesztése helyett kisebb hatáskörzetű rendszerek fejlesztésére térjen át.

Műszaki megoldások

Ami a szolgáltatás minőségi követelményeit illeti, a mozgóképet közvetítő videokonferencia-rendszerektől elsősorban azt várják, hogy a közvetített kép jó minőségű és színes legyen. E követelmény teljesítése egyszerűnek hangzik, de annál nehezebb a gyakorlatban megvalósítani, hiszen a konferenciateremben folytatott tárgyalások gyakran a szünetekben is folytatódhatnak, méghozzá az előcsarnokban, a liftben vagy a büfében. Az ilyen „sétáló tárgyalások” közvetítéséhez szükséges feltételek (például a megvilágítás) nem mindig állnak rendelkezésre, így a közvetítés minőségi követelményei csak aránytalanul drága műszaki megoldással valósíthatók meg.

A mozgóképes távkonferencia-rendsze-

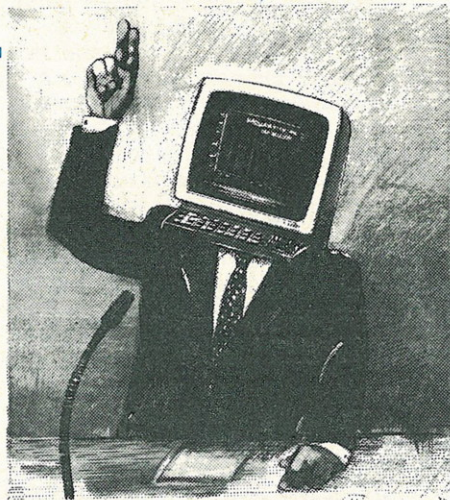
rek a tv-minőségű színes mozgókép mellett igen jó minőségű, nagy felbontású színes állóképet is közvetítenek, elsősorban az előadáson bemutatott dokumentációkról és ábrákról. A konferencia-közvetítéseket elemelve a szakemberek azt is kimutatták, hogy mozgókép-közvetítésre nincs is minden esetben szükség, mivel a közvetítésre érdemes információ zöme (90%-a) hang, és csupán 10%-a kép. Ezek a tapasztalatok hívták életre a hangot és állóképet közvetítő rendszereket, melyek csak az előadó — esetleg a felszólalók — hangját, valamint az ott bemutatott képeket és magyarázó ábrákat közvetítik, mozgó teremképet nem. Az ilyen állóképes videokonferencia-rendszerek beruházási költségei jóval alacsonyabbak a mozgóképes rendszerekénél.

A képközvetítő rendszerek fejlesztésével egyidőben jelentős haladást értek el a rendszerek átviteli sebességének csökkentése terén is, és ennek eredménye a már említett tömörített képátvitel. Különösen azért volt ez nagy jelentőségű, mert így az átviteli csatornák beruházási igénye és bérleti díja jelentősen csökkenthető. A nagy átviteli sebességgel működő rendszer csak nagy sávzélességű hírcsatornán működtethető, ráadásul az 1-2 MHz-es átviteli csatorna bérleti díja közel ötszöröse az 56 kHz-es csatornának.

Ma már egyre több gyártó készít olyan távkonferencia-rendszert, mely tömörített átvittel és az ennek megfelelő alacsonyabb (56 kbps) átviteli sebességgel működik, és jó minőségű képet szolgáltat. A tömörített átvittel működő rendszerek egyik legfőbb előnye, hogy nem igényelnek drága átviteli közeget, hanem a nyilvános telefonhálózaton is működtethetők.

Mindenfajta távkonferencia-rendszernek, mozgó- vagy lassan változó képet, esetleg csak hangot átvivő rendszernek megvan a maga sajátos alkalmazási területe, és célszerűtlen, egyúttal gazdaságtalan is lenne az egyes rendszerek kritikátlan telepítése, csereberélése csupán a divat kedvéért. Nyilván oktalanság — még a gazdag országokban/városokban is — az összes hagyományos telefonkészüléket képtelefonra kicserélni. Annál is inkább, mivel a képtelefon hívási költsége a hagyományosnak két-háromszorosa.

Említettük már, hogy a kapcsolatteremtés jellege szerint a távkonferencia-rendszerek működhetnek egy- vagy kétirányú átvittel (egyidejű adás és vétel). Az iparban mindkét rendszert előszeretettel alkalmazzák; az egyirányú átvittel működő rendszereket pillanatnyilag még nagyobb darabszámban — az ilyen rendszerek egyszerűbb felépítésűek és olcsóbbak —, de az arány várhatóan egyre inkább a kétirányú



átvittel működő rendszerek javára fog megváltozni.

Az egyirányú átvitel kétségtelen előnye a kapcsolat viszonylag egyszerűbb megvalósíthatósága — például a sugárzott adás tőtőantennával történő vétele —, amivel viszont szembeállítható a tömörített (kétirányú) átvittel működő rendszereknek az utóbbi időben tapasztalható jelentős árscökkenése, illetve többletszolgáltatásai. Egyik legfőbb varázs a más távkonferencia-rendszerekhez való egyszerű csatlakoztathatóság, ami egyre megfontoltabb döntés elé állítja a leendő felhasználókat. A választék bővülése, valamint a rendszerek segítségével igénybe vehető szolgáltatások sokfélesége nagymértékben járult hozzá a távkonferencia-rendszerek emelkedő népszerűségéhez és a szolgáltatásba bekapcsolódó előfizetők számának ugrásszerű növekedéséhez. Ilyen körülmények között már lehetőség nyílik arra is, hogy mindenki megtalálja azt a rendszert, amely pontosan az ő igényeit elégíti ki: azaz nem tud se többet, se kevesebbet, mint amire az adott felhasználásban feltétlenül szükség van, és megfizethető.

Személyi számítógépes távkonferencia-rendszerek

Míg az önálló személyi számítógépek (PC-k) a munkahely hatékonyságát elsősorban beépített gépi intelligenciájuk segítségével növelik meg, addig telefonvonalas összekapcsolásukkal a munkahely további — elsősorban a kommunikációból adódó — szolgáltatásokkal is fejleszhető.

A személyi számítógépes távkonferencia-rendszerek elterjedését elsősorban azok a gyártók segítették elő, akik gépeikhez a PC-be dugaszolható kivitelű távkonferencia-csatolóegységeket is kínáltak (IBM, Rascal Milco, Robot Research stb.). Az ilyen rendszerekkel mind álló-, mind mozgóképek, valamint hang is továbbítható. Ami a szakaszosan változó képet és hangot továbbító rendszerek árát illeti, a folyamatosan

mozgó képet továbbító rendszerek árához képest kb. 33%-kal alacsonyabb. A PC-s rendszerek konstrukciós megoldásainak legfőbb előnye, hogy a bonyolultabb funkciók és különlegesebb szolgáltatások utólagos megvalósításához (rendszerbővítéshez) elég a kártyacsere, így felesleges az új készülék megvásárlása. A kártyacsere ui. mindig olcsóbb, mint egy komplett berendezés.

A személyi számítógépes távkonferencia-rendszerek igen közkedveltek a felhasználók körében, és számuk az eladási statisztikák szerint rövidesen meghaladja az összes korábban installált távkonferencia-rendszert. A legolcsóbb kategóriát a hangkapcsolattal működő rendszerek jelentik, mert a kapcsolattartáshoz szükséges készülék (telefon) a legtöbb helyen megtalálható, és a telefonvonalak bérleti díja is az összes között a legalacsonyabb. Az audiokonferencia-rendszerek sokoldalú felhasználhatóságát jól példázza a Deers kereskedelmi hálózatban alkalmazott távkonferencia-rendszer, amely a hálózat egyes kereskedelmi részlegeinek naprakész információval való ellátására használnak — többek között a mindenkori piaci helyzettel és a konkurencia áraival kapcsolatosan.

A különböző rendszerek népszerűségi mutatóinak (például az előadások száma) elemzéséből az is kiderült, hogy a csak hangkapcsolatú rendszerek piaci forgalma jelenleg minden előnyük ellenére csökkenő tendenciát mutat a videokonferencia-rendszerekkel szemben. Egyrészt a szinte mindenhol meglévő tévékészülékek nagy száma, másrészt az egyre újabb műszaki megoldásokból adódó árscökkenés adja a magyarázatot, nem is beszélve a képi kapcsolatteremtésből fakadó egyéb előnyökről. Az egyik „legsikeresebb” terület az oktatás, ahol az 1982-ben létesített National University Teleconference Network (jelenleg 67 oktatási intézményt kapcsol össze egységes rendszerre) segítségével kezdetben még csak négy video- és hat audiokonferenciát rendeztek — évente —, a távlati tervekben viszont már napi két konferencia szerepel.

A videokonferenciázásnak nagy lökést adott a British Telecom által 1983-ban kidolgozott szabvány nemzetközi elfogadása, amely lehetővé tette olyan videokódoló egység kialakítását (a videokódoló a rendszer „lelke”), mely mind a PAL-, mind a SECAM- vagy NTSC-rendszerekben alkalmazható.

A távkonferencia-szolgáltatás népszerűsítésében a fejlett országokban sokat segítene, ha az ottani szaksajtó gyakrabban és részletesebben tudósítana a sikeres alkalmazásokról: ez meggyőzné a ma még várakozó állaspontra helyezkedő vásárlókat.

VEGYEM, NE VEGYEM?



Minősítés rangsorolás — tárgyilagosan

Amikor különböző dolgokat, például termékeket akarunk minősíteni, osztályozni vagy rangsorolni, általában összetett, többtényezős feladatot kell megoldanunk.

Ha csupán egyetlen tulajdonságot veszünk figyelembe, akkor a rangsorolás egyszerű, hiszen a megfelelő mérési adatok, jellemzők révén könnyen elvégezhetjük a

munkát. Személygépkocsik teljesítményét például a mért értékek kielégítően rangsorolják. Nehezebb döntési szituációba kerülünk, ha több tulajdonság egyidejű mérlegelése szükséges. Kényelmes volna, ha a minősítéshez, rangsoroláshoz olyan tulajdonságokat kellene csak figyelembe vennünk, amelyek függetlenek egymástól és jól

jellemzik a minősítendőket. Noha a gyakorlatban ez ritkán van így, a tulajdonságok megválasztásánál lehetőség szerint törekednünk kell arra, hogy ezek a feltételek minél nagyobb mértékben teljesüljenek.

A gyakorlatban szinte mindig több tulajdonság együttese szerinti vizsgálat a feladat. Általában azonban nem elegendő ismernünk az egyes tulajdonságok szerinti rangsorokat külön-külön ahhoz, hogy az „együttes rangsort” kialakítsuk. A tulajdonságonkénti rangsorok ugyanis gyakran eltérnek egymástól. A legnagyobb teljesítményű személygépköci például nem a legolcsóbb.

További nehézség, hogy a tulajdonságok nem azonos mértékben fontosak. A fontosság megítélésénél nem ritkán szubjektív szakértői értékelésekre kell hagyatkoznunk. Egy vállalatnál például a hatékonyság növelése nagyon sok tényezőnek lehet a függvénye. Egyidőben kell mérlegelni több tulajdonságot, például termutatók teljesítését, energiatakarékosságot, önköltségi ár csökkentését, környezetvédelmi előírások betartását stb. E célok fontossága mértékének megállapításánál — mint ezt már említettük — általában teret kell engednünk a

„SZGKOCSI” ALLOMANY VALASZTASI LEHETOSEGEI

Sorszám	MEGNEVEZÉS
1.	HARTBURG 353 W
2.	SKODA 120 L
3.	LADA 1200 S
4.	DACIA 1310
5.	ZASTAVA 1100 GTL

1. táblázat

„SZGKOCSI” ALLOMANY SÜLYTERTEKEI

SOR-SZAM	TULAJDONSAG NEV	SÜLY ERTEK	OPTIMALITAS IRANYA
1.	VETELAR eFt	87.00	A : MIN
2.	UTAZOSEBESSEG km/o	48.00	M : MAX
3.	BESZERZESI LEHETOSEG ev	44.00	A : MIN
4.	Fogyasztás l/100km	88.00	A : MIN
5.	VEGSEBESSEG km/o	28.00	M : MAX
6.	GYORSULAS 80km/orara sec	37.00	M : MAX
7.	MAX TERHELHETOSEG kg	29.00	M : MAX
8.	MAX TELJESITHENY LE	48.00	M : MAX

2. táblázat

„SZGKOCSI” ALLOMANY KIINDULO ERTEKEI

TULAJDONSAG	1. V.L.	2. V.L.	3. V.L.	4. V.L.	5. V.L.
1. VETELAP	118.000	128.000	142.000	139.000	170.000
2. UTAZOSEBESSEG	95.000	90.000	100.000	105.000	95.000
3. BESZERZESI LEHETOSEG	1.000	5.000	4.000	2.000	3.000
4. Fogyasztás	10.000	8.000	8.200	8.000	8.300
5. VEGSEBESSEG	130.000	140.000	140.000	140.000	135.000
6. GYORSULAS	15.000	19.000	20.000	19.000	17.000
7. MAX TERHELHETOSEG	400.000	400.000	400.000	390.000	390.000
8. MAX TELJESITHENY	50.000	52.000	60.000	54.000	55.000

3. táblázat

A(z) „SZGKOCSI” ALLOMANY VALASZTASI LEHETOSEG GRAFJA

A feltart „előnyösebbégi” viszonyok, csokkeno sorrendben. A nyíl (---) a „JÖBB” választási lehetőségre mutat.

1.	5. V.L.	---	4. V.L.
2.	5. V.L.	---	3. V.L.
3.	2. V.L.	---	3. V.L.
4.	1. V.L.	---	4. V.L.
5.	1. V.L.	---	3. V.L.
6.	2. V.L.	---	4. V.L.
7.	1. V.L.	---	2. V.L.
8.	5. V.L.	---	1. V.L.
9.	4. V.L.	---	3. V.L.
10.	3. V.L.	---	4. V.L.
11.	1. V.L.	---	5. V.L.

4. táblázat

7. táblázat

A(z) „SZGKOCSI” ALLOMANY VALASZTASI LEHETOSEGEINEK RANGSORA

RANG	VALASZTASI LEHETOSEG	KIVAL.	SZINT
SOR	MEGNEVEZES	SORREND	SZAM
1.	3. LADA 1200 S	1.	4.
2.	4. DACIA 1310	2.	4.
3.	2. SKODA 120 L	3.	4.
4.	1. HARTBURG 353 W	2.	4.
5.	5. ZASTAVA 1100 GTL	1.	4.

5. táblázat

6. táblázat

A(z) „SZGKOCSI” ALLOMANY VALASZTASI LEHETOSEGEINEK elonye, hatranya es határozatlansaga

VALASZTASI LEHETOSEG	ELONY	HATRANY	HATARAZAT
SZAM	MEGNEVEZES	LANGSAG	LANGSAG
1.	HARTBURG 353 W	41.40	52.45 6.15
2.	SKODA 120 L	44.35	41.73 13.93
3.	LADA 1200 S	53.90	39.45 6.65
4.	DACIA 1310	61.08	26.70 12.23
5.	ZASTAVA 1100 GTL	27.58	67.98 4.45

A(z) „SZGKOCSI” ALLOMANY VALASZTASI LEHETOSEGEINEK RANGSORA VISZONYSZAM SZERINT

RANG	VALASZTASI LEHETOSEG	VISZONY	SOR
SZAM	MEGNEVEZES	SZAM	SZAM
-	ELMELETI „jo”	100.00	-
1.	DACIA 1310	75.65	4.
2.	LADA 1200 S	69.98	3.
3.	SKODA 120 L	63.08	2.
4.	HARTBURG 353 W	49.14	1.
5.	ZASTAVA 1100 GTL	42.54	5.
-	ELMELETI „rossz”	0.00	-

szubjektív, tapasztalatokon alapuló véleményeknek is.

A minősítési, rangsorolási módszerek elmélete ma már terjedelmes, és az igen széles felhasználási területén is nehéz az eligazodás a rengeteg, többé-kevésbé egzakt módszer között. Ha ki is tudunk választani néhány megfelelőnek látszó módszert, kiderülhet, hogy ugyanarra a feladatra alkalmazva ezeket, nem adják ugyanazt az eredményt. A tapasztalat azt sugallja, hogy nincs „optimális”, mindenre egyformán alkalmas, legjobb módszer. A számítógépes rangsorolási algoritmusok (programok) kipróbálása révén a szakemberek több módszert is megismerhetnek. Ezek a kísérletek segíthetik a döntési probléma pontos formába öntését úgy, hogy az szerkezetében és kapcsolataiban egyaránt tükrözze a valóságos helyzetet.

Mivel a minősítés, rangsorolás mostanában országosan is az érdeklődés homlokterében van, úgy gondoljuk, tanulságos lesz, ha főbb vonalaiban ismertetünk egy tipikusnak tekinthető, mikroszámítógépen megvalósított, párbeszédés, minősítő, rangsoroló rendszert.

A SUGAR—16 számítógépes rangsoroló rendszer szolgáltatásait egy szemléltető példa kapcsán mutatjuk be.

Feltételezzük, hogy személygépkocsit akarunk vásárolni és az 1. táblázatba összegyűjtött öt lehetőség valamelyikére szeretnénk befizetni.

A választásnál a 2. táblázatba foglalt tulajdonságokat kívánjuk figyelembe venni. A táblázatban feltüntetett tulajdonságokhoz rendelt súlyértékeket (15 „átlagos vásárló” értékelésének a számtani átlagát véve), valamint azt, hogy a megfelelő tulajdonságoknál az értékek alacsony vagy magas volta előnyös. Ezt A, illetve M betűvel jelöltük, és „az optimalitás irányának” neveztük.

A választási lehetőségeket — minden adott tulajdonság szerint — egy-egy mennyiségi érték jellemzi, amelyeket kiinduló értékeknek nevezünk. A kiinduló értékeket a 3. táblázatban foglaltuk össze. A választási

si lehetőségek (V. L.) az oszlopokban, a tulajdonságok a sorokban állnak.

A programrendszerben használt matematikai módszerekre most nem térhetünk ki, annyit azonban megemlítünk, hogy a választási lehetőség párok közötti kapcsolatok feltárásának lényeges szerepe van.

A táblázatban szereplő adatokkal kapott eredményeket a programrendszer négy eredménytáblázatban foglalja össze.

A választási lehetőségek között feltárt kapcsolatokat a 4. táblázatban láthatjuk, amelyből egy irányított gráfot is szerkeszthetünk.

A rangsort, amelyet a program iteratív módon alakít ki, az 5. táblázat tartalmazza.

A program minden lépésben — meghatározott összefüggésekből — egy-egy „legjobb”, minimális elemszámú, nem üres, választási lehetőség elemű halmazt határoz meg

(1) a „legjobb” választási lehetőséget más választási lehetőség ne előzze meg; illetve

(2) a „legjobb” választási lehetőség minden más választási lehetőséget előzzön meg

feltételeknek megfelelően. Követelmény még, hogy ezek a feltételek csak a „legjobb” választási lehetőségre teljesüljenek.

A „legjobb” választási lehetőség halmazok, amelyeket az (1), illetve a (2) követelményeknek eleget tevően alakítottunk ki, természetesen nem minden esetben lesznek azonosak.

Attól függően, hogy milyen összefüggések teljesülnek az előbbi két halmazra, 1-től 4-ig terjedő „szintszámot” rendelünk a kiválasztáshoz. A példánkban a 3. választási lehetőségre az (1) és a (2) feltétel is teljesül, és csak ez a választási lehetőség tesz eleget a feltételeknek.

Az algoritmus teljesen analóg eljárással választ egy-egy „legrosszabb” választási lehetőség halmazt is, és ellátja ezeket kiválasztási szintszámmal.

Ha a kiválasztási szintszám mindkét esetben 4, akkor az algoritmus leválasztja a „legjobb” és a „legrosszabb” választási le-

hetőséget, ha nem, akkor csak azt, amelyiknek nagyobb a szintszáma, és a megmaradt halmazra megismétli az eljárást addig, ameddig választási lehetőség van.

A program minden választási lehetőségre százalékosan megadja az összes többihez viszonyított ELŐNY-, HÁTRÁNY- és HATÁROZATLANSÁG-értéket (6. táblázat).

A HATÁROZATLANSÁG abból adódik, hogy két vagy több választási lehetőséget ugyanaz az érték jellemezhet. Ezeket nem tartjuk preferencia szempontból rendezhetőnek.

A minősítés eredményeként az emberek általában valamilyen módon mérhető, a választási lehetőségekhez rendelt értékekesség-jellemzőt is várnak. Bár ilyen egyértelműen meghatározható érték nem létezik, a 4. táblázatban ennek az igénynek a kielégítésére a választási lehetőségekhez egy viszony-számértéket rendeltünk a következő módon.

Az összemérendő választási lehetőségek kiinduló értékeiből a program kiválasztja a legkedvezőbb és a legkedvezőtlenebb értéket tulajdonságonként, és ezekkel egy „elméleti jó”, illetve egy „elméleti rossz” választási lehetőséget szerkeszt. A (7. táblázatban megadott) viszonyszám az így kialakított szélső értékekhez mérten egy viszony-mutató, amelynek alapján képzett sorrend — mivel ez az eljárás lényegesen eltér az előzőtől — nem szükségképpen lesz azonos a 2. táblázat sorrendjével.

Befejezésül hangsúlyoznunk kell, hogy gépkocsi-minősítési példánk csupán szemléltetési célt szolgál. Egyik érintett típus gyártója se vegye rossz néven, hogy nem az ő gyártmánya lett az első! Ugyanigy senki se siessen számaink alapján gépkocsit vásárolni! Már csak azért se, mert a minősítésnél nem vettünk figyelembe minden fontos jellemzőt, például a minősítést, a korszerűséget, az üzemeltetés, a javítás költségeit, az átlagos élettartamot stb.

A párbeszédés SUGAR—16 programrendszer számítástechnikai ismeretek nélkül is könnyen és biztonságosan használható.

GAÁL LÁSZLÓ

Az év számítógépe

A CHIP kezdeményezésére — az előző évekhez hasonlóan — 1986-ban is megválasztották az év számítógépét. Hét ország szakújságírói kérték fel a döntés meghozatalára. Először vett részt a választásban szocialista ország szerkesztősége (Svet kompjutera—Jugoszlávia).

A zsűrinek olyan követelményeket kellett figyelembe vennie, hogy a rendszer széles körben legyen elterjedve, műszaki különlegességei és kidolgozottsága révén az átlagból kiemelkedjen, valamint az összes piac felé irányadó tendenciát mutasson.

A versenybírók mindkét szempontot érvényesítették: az egyik minősítő paramétercsoportba besorolták a hétköznapi áruk innovációját, a másikba pedig az év gépének saját új „ékecségeit”.

A HC-knél a Commodore Amiga feltűnően kiemelkedik, nemcsak az USA-ban, hanem világ-szerte. A zsűri döntése is ezt a tényt fejezi ki.

Az Atari hosszú ideig tartotta magát ST családjaival, de győzelmét akadályozta a két különböző modellra való szétválasztása és a nemzeti piacok különböző helyzete. Néhány országban jelenleg az ATARI 520 ST áll az érdeklődés középpontjában, máshol a figyelem sokkal inkább a fejlettebb 1040 ST-re irányul — így osztoznak a második helyen. Komoly pontszámkülönbséggel következnek a konvencionális HC-k: Amstrad 6128 (más néven Schneider 6128), Apple II.

és a kisebb Commodore-ok: C128, C64. Ez is tükrözi a számítógép-alkalmazások új erővonalait: a jelentősebb HC-k semmiképpen nem nyújtanak kisebb teljesítményt mint a professzionális PC-k, sőt bizonyos területeken lényegesen többet tudnak.

A PC-k esetében sem volt kétséges a zsűri döntése, hiszen az IBM PC/AT kétszer annyi pontot kapott, mint a második helyezett.

A professzionális gépek világában az IBM a mértékadó, a további helyezésekre leadott szavazatoknál a technikai finomságok döntöttek. Ebben a kategóriában a tíz helyezett gépből nyolc MS—DOS kompatibilis. Kivételt képez a Schneider—Joyce, amelyik az ún. text-komplett gépek-nél szokatlan koncepciójával minden esetben a második helyet vívta ki, valamint az Apple Macintosh utolérhetetlen felhasználási lehetőségei és grafikai szolgáltatásai miatt.

Commodore Amiga	230 Punte
Atari 1040 ST	140 Punte
Atari 520 ST	140 Punte
Schneider CPC 6128	60 Punte
Apple II	60 Punte
Commodore 128	40 Punte
Commodore 64	40 Punte



IBM PC/AT	190 Punte
Schneider Joyce	70 Punte
Olivetti M28	60 Punte
HP Vectra	50 Punte
NCR P8	30 Punte
Wang APC	20 Punte
Siemens PC-D	20 Punte
Apricot XN	20 Punte
Apple Macintosh +	5 Punte
Compaq Deskpro 286	5 Punte

Emlékképek

A washingtoni Walter Reed Kórház elnöki különszobájában 1957. február 8-án meghalt Neumann János, akit sokan a század nagy matematikusának — néhányan a legnagyobbjának — tartanak. Neumann jelentőset alkotott nemcsak a matematikában, de a közgazdaságtan és a meteorológia matematikai megközelítésében is, és egyedülálló szerepe volt a mai modern számítógépek kifejlesztésében. Számunkra, akik akár a munkánk során, vagy tanulás céljából, de hobbi-ból is a számítógépekkel vagyunk kapcsolatban, valószínűleg ez az utóbbi jelenti a legfontosabbat Neumann életművéből.

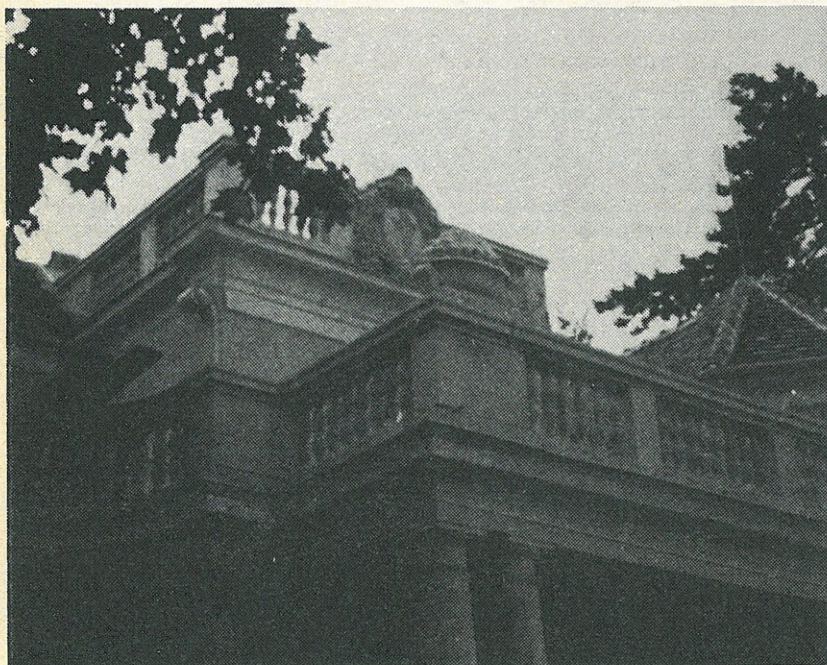
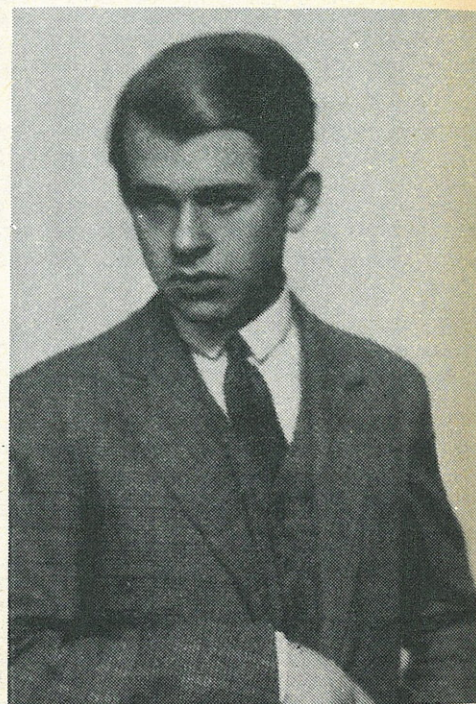
Gondolataink időtállóságát mi sem bizonyítja jobban, mint az, hogy a ma kapható gépek mindegyike — halála után 30 évvel, a felgyorsult fejlődés korában is — az általa megfogalmazott elvek szerint épült, ún. Neumann-elvű gép.

Neumann János emlékének és munkájának — halálának 30. évfordulóján — az életét bemutató képsorozattal tisztelgünk. A családi képek rendelkezésünkre bocsátásáért Neumann János fivérének, Nicolas Vonneumann úrnak köszönjük fejtegetésüket ki.

1. képsor

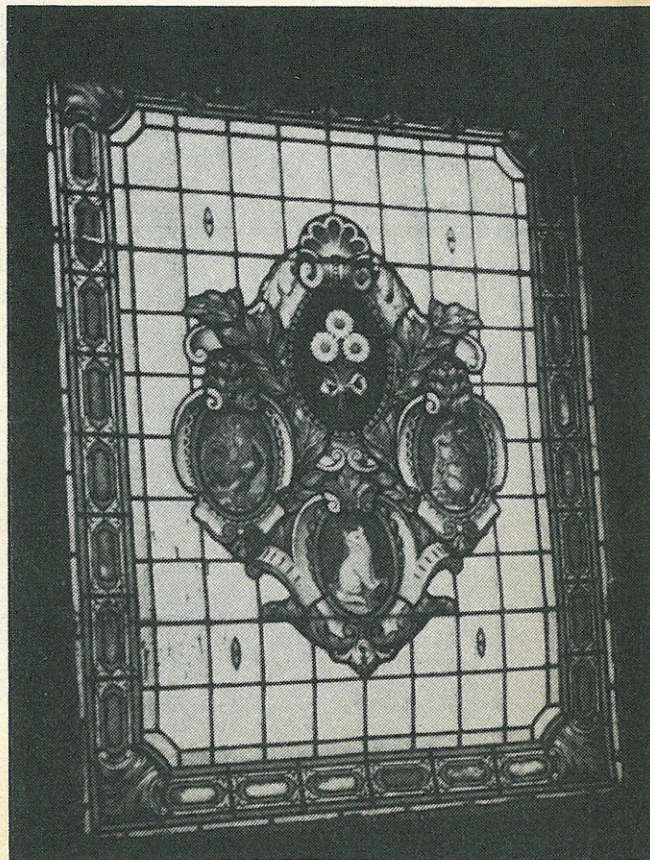


Neumann János Budapesten született, 1903-ban. A képeken: apja, margittai Neumann Miksa dr. bankár, édesanyja, Kam Margit és a fiúk (balról jobbra): Mihály, a



2. kép

A szülőház és a lépcsőház ablakának képe. A három margaréta a Ferenc József által adományozott margittai előnévre utal, a három állatfigura pedig a három gyerek — ma úgy mondanánk, hogy „kabala”-figurája. A kakas volt Jánosé.

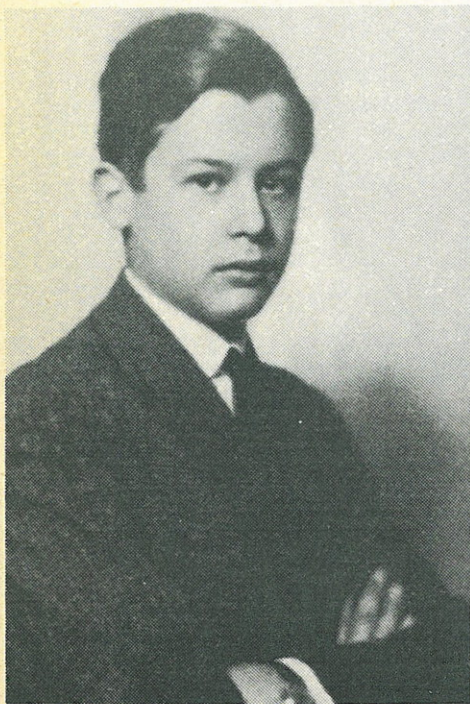


3. kép

Neumann Jánosról

középső, Miklós, a legkisebb, és János, a legidősebb. Neumann Miksa 1929. június 4-én halt meg. Az akkori ipari fellendülés — a régi gárda — progresszív alakjáról a Borsszem Jankó rövid versben emlékezett meg:

margittai Neumann Miksa dr.
Egy munkás élet, sok szép akarat,
Tegnapról mára im kettészakadt.
Alighogy heged, támad újra friss seb,
S a régi gárda kisebb, egyre kisebb.



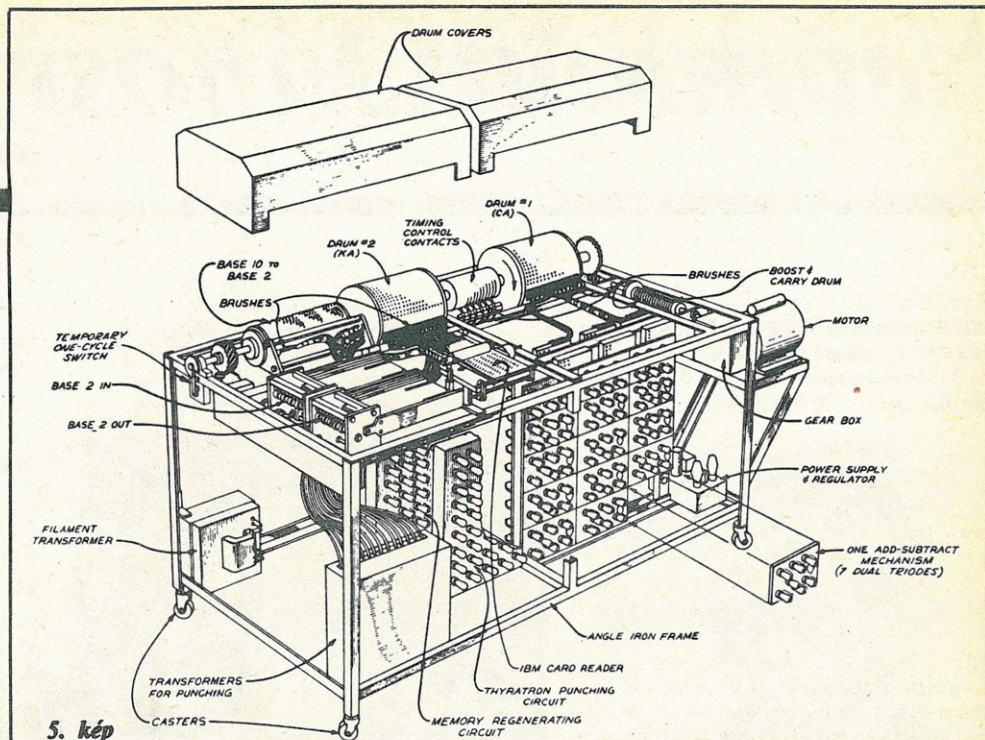
János 12 éves korában unokatestvérével, Miklós fivéré szerint János hihetetlen emlékezőtehetséggel rendelkezett és a legkülönbözőbb területek (nemcsak a matematika) iránt is háttalan érdeklődést mutatott (pl.: történelem, művészettörténet, nyelvek stb.). A fasori gimnázium tanárai gyakorta „panaszolták” a szülőknek, hogy ha megkérdezték Jánost az aznapra kijelölt házi feladatról, azt sohasem tudta, de ha rákérdeztek a feladatra, akkor azt azonnal és pontosan megoldotta. Miklós: „... az utolsó gimnáziumi éveire vonatkozó saját és közvetlen emlékeim szerint is, iskolából hazajövet állva és csak néhány percig nézte át tankönyveit, s ezzel a felkészülés be volt fejezve!” Rácz László, az iskola híres tanára hívta fel apja figyelmét János rendkívüli tehetségére, neveléséhez tanácsot adott Fekete Mihály, Kürschák József és Fejér Lipót is. János, apja tanácsára Zürichben vegyész-mérnöki képesítést, 1926-ban, 23 éves korában Budapesten matematikai doktorátust szerzett. Tudományos pályájának következő állomásai: Berlin, ahol egyetemi magántanár, Göttingen, 1930-ban Oswald Veblen hívására a princetoni egyetem tanára.

4. kép

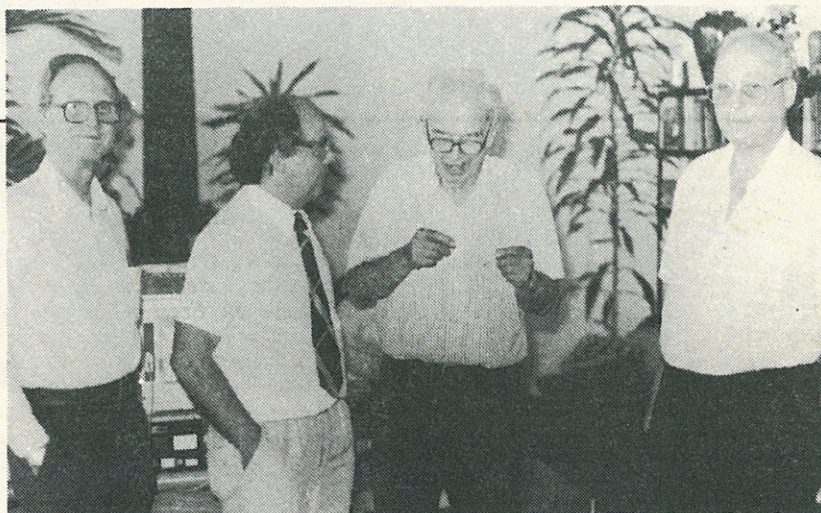


Az első elektronikus elemekből épített számítógépet John Vincent Atanasoff 1936–38 között tervezte. 1940–42 között Clifford E. Berry-vel (†1962) építették meg és ABC gépnek (Atanasoff–Berry–Computer) nevezték el ezt az elektronikus számítógépet. A gép nagyon sok dologban első volt a világon azon kívül, hogy a műveletet végző elemeket elektroncsöves áramkörökből építették. Az ABC kettes számrendszerben számolt, ugyancsak először a számítógépek történetében. Ők használták először a computer szót és az analóg computer elnevezést is, megkülönböztetésül a digitális géptől. Ebben a gépben alkalmazták először a dobmemóriát, igaz, az egyes biteket itt beépített kondenzátorok tárolták. A számítógépben kb. 300 elektroncső működött, a dobmemóriát 3000 bit kapacitására tervezték. Atanasoff különféle bürokratikus bonyodalma miatt az ABC-ben alkalmazott megoldásokra nem kapott szabadalmat. Az ENIAC-ra adott szabadalmak ügyében a Honeywell-cég pert indított. 1973. okt. 19-én került sor az ítélet kihirdetésére. A bíróság egy sor megoldásnál Atanasoff elsőbbségét ismerte el.

Az ABC hatással volt Neumann későbbi IAS-beli munkájára is. Az 5. képen az ABC rajza — Thomas J. Hayes, South Bend Indiana (Annals of the History of Computing, July 1984. Volume 6. Number 3). A 6. képen látogatás J. V. Atanasoff házában (jobbról balra: prof. D. H. Wolbers, J. V. Atanasoff, Kovács Gy., B. Atchison).



5. kép



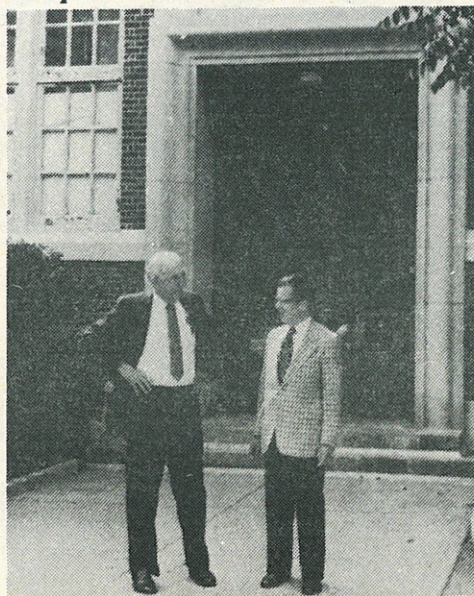
6. kép



7. kép

A közeledő háború az Egyesült Államokban, de más országokban is új lendületet adott a számítógépesítési kutatásoknak. Olyan gyors számológépeket akartak kifejleszteni, amelyekkel a bonyolult tűzvezetési (tűzérési, illetve bombázási) táblázatokat nagyon gyorsan ki lehetett számítani. Az ENIAC-ot (Elect-

8. kép



ronic Numerical Integrator and Computer) katonai célra építették.

A gépet 1943-ban J.W. Mauchly javaslata alapján és vezetésével kezdték el tervezni, az

építés főmérnöke I. P. Eckert volt, a hadsereg részéről H.H. Goldstine irányította a munkát. A 18 000 elektroncsövel és 1500 jelfogóval megépített gép egy összeadást 200 μ s, egy szorzást 3 ms és egy osztást 30 ms alatt végzett el. Elkészültét 1946. február 15-én jelentették be. Neumann-nal a gépet Goldstine ismertette meg, óriási hatással volt rá. Neumann a gép építésében nem vett részt, de fontos továbbfejlesztési javaslatai voltak (pl. a program tárolására).

A 7. képen Saul Gorn — aki programokat készített az ENIAC-ra — a Moore Schoolban (Philadelphia), a szerzőnek a gép működését magyarázza, a 8. képen a Moore School épülete, ahol Neumann dolgozott. Az előtérben Saul Gorn és Nicolas Vonneumann.

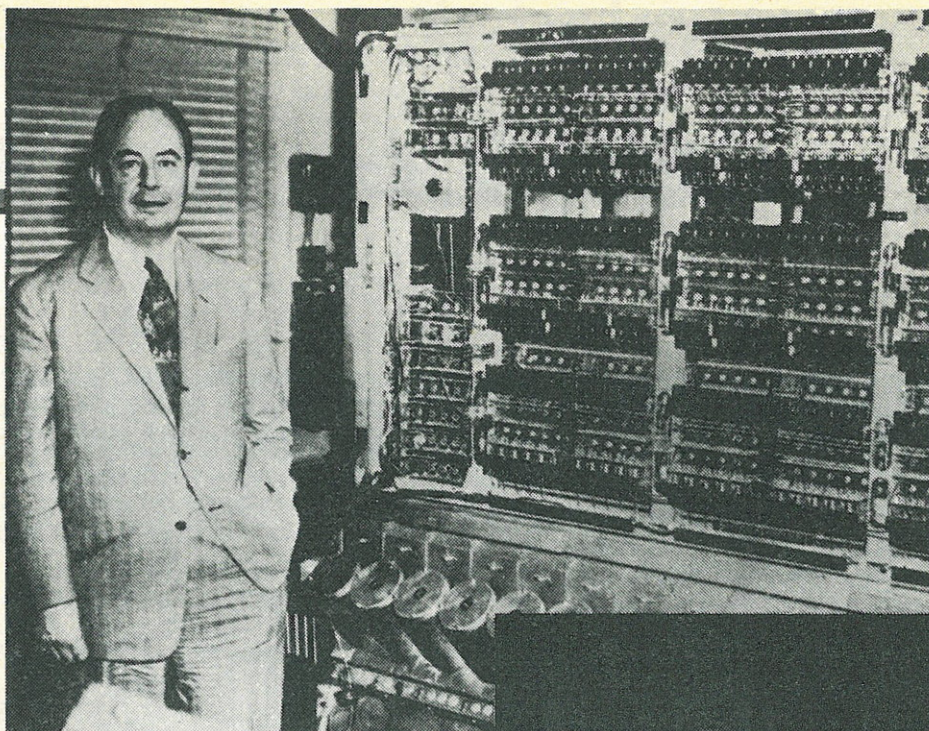
Neumann 1944 augusztusától Goldstine-nel és másokkal egy új gép, az EDVAC (Electronic Discrete Variable Computer) fejlesztésén dolgozott. A „feladatterv” első megfogalmazásában (First Draft) leírja a mai modern számítógépek valamennyi lényeges ismervét. Az EDVAC építésében Neumann már nem vett részt, miután Goldstine-nel együtt az Institute of Advanced Studyban lehetőséget kaptak egy még modernebb elvű, párhuzamos működésű, egycímű, tárolt programú számítógép megépí-

9. kép

tésére. Ezt a gépet általában IAS gépnek vagy Neumann gépnek is nevezik. A gép tárolójaként az akkor leggyorsabban működő Williams csöveket alkalmazták. Technikatörténeti érdekesség, hogy a géphez katódsugárcsöves kijelzőt is használtak, pl. grafikonok megjelenítésére. A 9. képen Neumann az IAS gép előtt.

1955-ben Neumann János megbetegedett. Feleségét, Dán Klárát idézzük, aki Neumann „A számológép és az agy” c. könyvének előszavában erről így ír:

„1955. március 15-én mint az Atomenergia Bizottság újonnan kinevezett tagja, hivatali esküt tett, s röviddel utóbb, május elején, Washingtonba költöztünk. Három hónappal később, augusztusban, tevékeny és feszült életvitelünk, amelynek fő mozgatója férjem fáradhatatlan és lenyűgöző szelleme volt, hirtelen



10. kép

megszakadt. Johnny bal vállában súlyos fájdalmak léptek fel. A műtét során csonttráktot állapítottak meg. A következő hónapok változó reményben és kétségbeesésben teltek el. Egy ideig abban bíztunk, hogy a váll megbetegedése a szörnyű kór egyetlen megnyilvánulása volt, s a tünetek hosszú ideig nem fognak visszatérni, de azután újra jelentkeztek a leírhatatlan fájdalmak és kínok, s halomba döntötték reményeinket. Johnny ez idő alatt lázasan dolgozott. Nappal a hivatalában volt vagy a munkájával kapcsolatos ügyekben utazgatott, éjszaka tudományos cikkeket írt, amelyeket eddig halogatott, mert el akart készülni a bizottság részére végzett határidős munkájával. Most rendszeresen elkezdett dolgozni a Silliman-előadások kéziratán. Annak a nagy része, amit az itt következő oldalak tartalmaznak, a bizonytalanságnak és a várakozásnak ezekben a heteiben íródott. November végén jött a következő roham: a vizsgálat megállapította, hogy a betegség több helyen megtámadta a gerincét és a járásban is súlyos nehézségeket okozott. Ettől kezdve nap mint nap rosszabbodott az állapota, míg végül csak annyi remény maradt, hogy kezeléssel és gondossággal legalább egy időre meg lehet állítani a végzetes betegség elhatalmasodását.

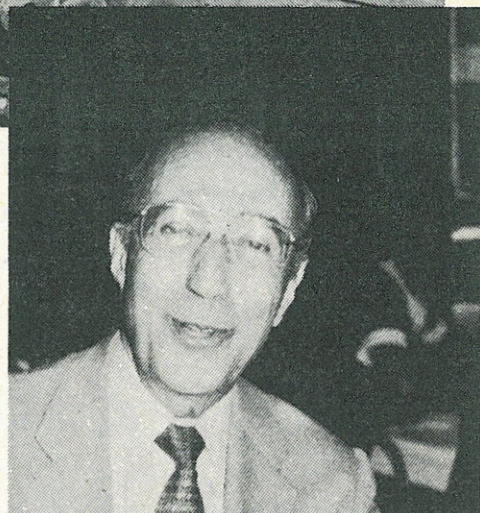
Johnny 1956 januárjától kezdve már csak

tolószékben tudott közlekedni, de még mindig részt vett megbeszéléseken, tolószéken bejárt a hivatalába, s továbbra is dolgozott az előadás-sorozat kéziratán. Ereje szemmel láthatólag napról napra fogyott. Egymás után le kellett mondania minden utazást és vállalt előadást — egyedül a Silliman-előadásokkal tett kivételt. Volt némi remény arra, hogy röntgenbesugárással március végéig legalább annyira rendbe hozhatják beteg gerincét, hogy elutazhasson New Havenbe és eleget tehessen ennek az egy kötelezettségének. Ezért megkerestük a Silliman-előadások rendezőségét, hogy egyre vagy legfeljebb kettőre csökkentse le az előadások számát, hiszen egy teljes héten át tartó erőfeszítés, legyengült állapotában, veszélyes lehetett volna. Márciusban azonban minden csalóka reményünk szertefoszlott, és többé gondolni sem lehetett arra, hogy Johnny bárhova is el tudna még utazni. A Yale Egyetem most is segítőkész és megértő volt, nem törölte az előadásokat, hanem azt javasolta, hogy ha férjem a kéziratot rendelkezésre tudja bocsátani, valaki más felolvassa helyette. Johnny minden erőfeszítése ellenére sem tudta határidőre megírni valamennyi tervezett előadását; a tragikus sors úgy hozta, hogy a kézirat befejezetlen maradt.

Április elején befeküdt a Walter Reed Kórházba, s ezt a helyet 1957. február 8-án bekövetkezett haláláig már nem hagyta el. A Silliman-előadásokra készült befejezetlen kéziratot magával vitte a kórházba; ott még néhányszor megpróbált dolgozni rajta, de azután végleg felülkerekedett a betegség, s még Johnny kivételes szelleme sem tudta legyőzni testének gyengeségét.”

Képünkön Neumann János átveszi Eisenhower elnöktől az egyik legnagyobb amerikai kitüntetését, a Szabadság Emlékéremet.

A számítógépek kifejlesztésének korai szakaszáról az egyik legjobb — alap — mű H. H. Goldstine „A számítógép Pascaltól Neumannig” c. könyve. Goldstine (11. kép) — véleményem szerint — Neumann „számítógépes életének” a legjobb ismerője, az ő érdeme nem-



11. kép

csak az, hogy Neumann érdeklődni kezdett az ENIAC iránt, de az is, hogy Neumann a háború után végleg a számítógépek fejlesztésének alapvető kérdéseivel foglalkozott. A könyvet Goldstine 1979-es látogatása után küldte el nekem, azóta szeretném a magyar fordítását megjelentetni. A Műszaki Könyvkiadó közreműködésével a könyv a '87 idején már kapható lesz a könyvesboltokban. Neumann Miklós a könyvben közölt családi vonatkozású részletekkel kapcsolatban néhány korrekciót javasolt. 1985-ös amerikai utazásom alatt sikerült Philadelphiában találkoznunk és az inkriminált részleteket hármásban megbeszélünk.

Utószó

Neumann János nem kapta meg a sorstól a hosszú élet lehetőségét, alig élt többet, mint fél emberöltő. Összegyűjtött műveinek vaskos kötetei, számunkra pedig leginkább a Neumann-elvű számítógép már több mint negyvenéves fejlődése azt bizonyítja, hogy az informatika korának legjelesebb alakját tisztelhetjük benne, akinek jelentősége az évek múlásával nem halványul, hanem egyre inkább erősödik.

KOVÁCS GYÖZŐ

BASIC és gépi kód

Legutóbb az egyszerű indexelt címzési módokról volt szó. Most újabb két indexelt címzési móddal ismerkedünk meg. Ezeket az indexregiszterek ismertetésekor futólag már említettem.

Az indirekt indexelt címzési mód

Az operandus alakja: (\$nn), y — ahol \$nn egy nulla lapon levő bájtt címét jelöli. Ezen és a következő bájton egy tárcím található, amelyet báziscímnek nevezünk. A műveletben szereplő címet a báziscím és az Y regiszter tartalmának összege adja.

Nagyon fontos programozási eszköz, gyakran találkozhatunk vele. Alkalmazására most a 3. programban láthatunk egy egyszerű példát.

Az indexelt indirekt címzési mód

Az operandus alakja: (\$nn, x) — ahol \$nn egy nulla lapon cím, az indexelt indirekt címzés báziscíme. Ezt úgy használjuk, hogy a báziscímhez hozzáadjuk az X regiszter tartalmát, és az így kapott címen található és az utána következő bájtt tartalmazza a műveletben szereplő címet.

Az indexelt indirekt címzéssel ellentétben ezt a címzési módot nagyon ritkán használjuk, néhány bemutató példán kívül Commodore gépen eddig még nem találkoztam vele. A mellőzés oka feltehetően az, hogy sok helyet vesz igénybe a nullás lapon, és az egyszerű indexelt címzési módok jól helyettesítik.

A mintaprogramokról

A mostani három mintaprogram mindegyike ugyanazt a feladatot hajtja végre: 40 darab fehér A betűt ír a C64-es képernyőjének első sorába. A programok bemutató jellege miatt most csak a disassembler listáját közlöm, BASIC-betöltőt és a másik két géptípushoz tartozó változatot nem.

A programokban két fontos cím található: \$0400 a képernyőmemória, \$d800 a színmemória kezdőcíme. Az A regiszterbe töltött \$01 érték egyszerre az A betűnek és a fehér színnek is kódja: Y-ba a decimális 39-nek megfelelő érték kerül, ugyanis az indexelés 0-ról indul. Az első program egy kezdetleges és eléggé körülményes módszert mutat be, melynek ötletét a VC20 és C64 CHRGET rutinja adta. A módszer lé-

nyege, hogy a program módosítja az egyes utasítások operandusát. Az Y regiszter ebben az esetben csak a ciklusokat számlálja. Az \$C01D...\$C02C címen található utasítások az operandusok eredeti értékét állítják helyre, hogy a program újabb hívásakor nehogy baj legyen. Amikor a 6502-es mikroprocesszor gépi kódú programozásával ismerkedtem, én is alkalmaztam ezt a módszert, mégis azt ajánlom, hogy lehető-

```

c000 a901          lda #$01
c002 a027          ldy #$27
c004 8d0004        sta $0400
c007 8d00d8        sta $d800
c00a ee05c0        inc $c005
c00d d003          bne $c012
c00f ee06c0        inc $c006
c012 ee08c0        inc $c008
c015 d003          bne $c01a
c017 ee09c0        inc $c009
c01a 88            dey
c01b 10e7          bpl $c004
c01d a900          lda #$00
c01f a204          ldx #$04
c021 a0d8          ldy #$d8
c023 8d05c0        sta $c005
c026 8e06c0        stx $c006
c029 8d08c0        sta $c008
c02c 8c09c0        sty $c009
c02f 60            rts
    
```

```

c000 a901          lda #$01
c002 a027          ldy #$27
c004 990004        sta $0400,y
c007 9900d8        sta $d800,y
c00a 88            dey
c00b 10f7          bpl $c004
c00d 60            rts
    
```

```

c000 a900          lda #$00
c002 a004          ldy #$04
c004 85fb          sta $fb
c006 84fc          sty $fc
c008 a900          lda #$00
c00a a0d8          ldy #$d8
c00c 85fd          sta $fd
c00e 84fe          sty $fe
c010 a901          lda #$01
c012 a027          ldy #$27
c014 91fb          sta ($fb),y
c016 91fd          sta ($fd),y
c018 88            dey
c019 10f9          bpl $c014
c01b 60            rts
    
```

leg kerüljük el a használatát. Sajnos, a különféle lapokban közölt programokban már többször találkoztam vele.

A második program ugyanezt a feladatot indexeléssel oldja meg, jóval rövidebb, mint az előző.

A harmadik program az indirekt indexelt címzést mutatja be. A program bevezető részében a képernyőmemória kezdőcímét a \$fb...\$fc, a színmemóriáét a \$fd...\$fe címekre tesszük le, ezeken a helyeken lesznek a báziscímek. A címzési mód előnyeiben a példában nem nagyon látszanak.

Figyeljük meg az utóbbi két programban, hogy az A regiszter tartalmának az egyes tárcímekre történő lerakása „visszafele” történik, azaz csökkenő indexekkel. Ez az eredményt nem befolyásolja.

BARNA LÁSZLÓ

Indul az OKTA-TOTÓ!

Értékes főnyeremények!

Fél éven át szórakozva okulhatnak olvasóink: keressék 1987. májusi számunktól kezdődően novemberig a Mikroszámítógép Magazinban a számítástechnikai alapismeretek

OKTA-TOTÓ-ját!

A helyes tippszelvények beküldői között minden hónapban öt 200 Ft-os könyvtalványt sorsolunk ki. Ez év decemberében a fődíjakat azok közül a játékosok közül húzzuk ki, akiknek a neve bekerült a kalapba, mert telitalálatos totóikat havonta határidőre eljuttatták szerkesztőségünkbe.

Főnyeremények:

- I. díj: 64 K-s számítógép bővítéssel
- II. díj: 64 K-s számítógép programokkal
- III. díj: 64 K-s számítógép

Az OKTA-TOTÓ-ban szakkörök és csoportok is résztvehetnek. A feladatokat az esélyegyenlőséget szem előtt tartva fogalmazzuk meg: a jó megoldásokhoz nincs szükség számítógépre, csak tiszta fejre - s a nyeréshez pedig egy kis szerencsére.

A Szerkesztőség

TAVASZTÓL ŐSZIG JÁTSZUNK - GYERE HÁT TE IS, JÁTSSZUNK!

OVERLAY — CHAIN —

APPEND

A cikk első részében megvizsgáltuk, miképpen lehetséges, hogy egy BASIC program egy másik BASIC programot töltsön a tárb. Most a BASIC és a gépi kódú programok között létrehozható hasonló kapcsolatokat elemezzük.

Gépi kódú program betöltése BASIC programból

BASIC programból gépi kódú program háromféleképpen vihető be.

1. *Betöltés DATA sorokból.* A módszer lényege, hogy a gépi kódú programot bájtontként, decimális számként értelmezve a BASIC program DATA soraiban helyezük el, majd egy READ-POKE ciklussal töltjük a kívánt tárterületre. Ezt az egyszerű módszert alkalmazza az OVERLAY című mintaprogram. Hátránya viszont a READ-POKE ciklus időigénye és a többszörös tár-igény. Ha betöltés után a töltőprogramot nem töröljük, akkor a gépi kódú program bájtontként binárisan és decimálisan kódolva is jelen van a tárb.

2. *Betöltés a BASIC programmal együtt,* úgy, hogy a gépi kódú programot a BASIC program végéhez csatoljuk. (A módszer alapjait lásd a lap 1986. októberi számában, a BASIC RAM felosztás című cikkben.)

Nézzük, hogyan lehet ilyenkor a programokat társítani:

- Betöltjük a BASIC programot.
- Beírjuk a következő sort a program legelső sorába:

```
10 POKE45, : : : POKE46, : : : CLR
— Lekérdezzük és megjegyezzük a SOV értékét (PEEK), ez lesz a gépi kódú rész kezdőcíme. Ehhez hozzáadjuk a gépi program bájtjainak a számát. Ez lesz a SOV új értéke.
```

— A 10-es sorban a vessző utáni kettőspontok helyére beírjuk a SOV új értékének alsó és felső bájtját decimálisan. A megmaradó kettőspontokat nem bántjuk.

— Elkészítjük a gépi kódú program adott tárcímű változatát.

— Parancs módban kiadott LOAD-dal betöltjük a gépi kódú programot a kívánt helyre.

— Kimentjük a kész BASIC-gépi kódú programunkat (SAVE).

Az így elkészült program előnye, hogy a BASIC és a gépi kódú egyetlen LOAD utasítással töltődik be, és a tárban nincsenek nagy helyigényű DATA sorok. Maradt még egy megmagyarázandó feladat: hogyan készítsük el a gépi kódú program megfelelő változatát? Csak azok a programok jöhetnek számításba, amelyek bármilyen tárci-

men működtethetők. Ha ez a feltétel nem teljesül, akkor a gépi kódú programot át kell írni vagy újra fordítani. Ha ez megvan, akkor már akadálytalan a kimentés vagy betöltés. A következő részben arra is választ kapunk, hogyan lehet egy adott tárterület tartalmát kimenteni vagy betölteni.

3. *Betöltés külön LOAD utasítással.* Ha a gépi kódú program megfelelő programfájlban rendelkezésünkre áll, akkor azt a LOAD„filenév”,8,1 utasítással tölthetjük be. Egy dologra azonban nagyon kell figyelni: a LOAD végrehajtása után a program futása nem a LOAD-ot követő

utasításnál, hanem a program első utasításánál fog folytatódni. Ezért, ha a LOAD-ot már egyszer végrehajtottuk, akkor a program újraindulásakor azt többször már ne ismételjük meg, nehogy végtelen ciklusba keveredjünk. Emiatt terjedt el a következő módszer például:

```
10 IF A=0 THEN A=1:LOAD„P1”,8,1
20 IF A=1 THEN A=2:LOAD„P2”,8,1
A program minden LOAD után a 10. sortól indul, de mindig csak azt a LOAD-ot hajtja végre, amelyik eddig még nem futott, majd a soron következő utasításornál folytatódik.
```

Nem tartozik szorosan a programbetöltés témakörébe, mégis érdekes, hogyan menthető egy tárbeli gépi kódú program BASIC utasításokkal lemezre, programfájlba. Erre példa a SAVE nevű program; működése a következő. A SAVE BASIC utasítás a SOB-tól a SOV-ig terjedő tárterületet menti ki. A program ezeket a mutatókat állítja a kívánt értékre, végrehajtja SAVE-et, majd visszaállítja a mutatókat. A helyes működés titka, hogy a SOV elállítása miatt a program nem használhat egyszerű változókat, de használhatja a tömböket. A 10-es sor nem szervezhető a 6. sorhoz hasonló ciklusba.

LOAD gépi kódban

Amikor egy BASIC LOAD utasítást hajtunk végre, akkor ez a ROM-ban lévő gépi kódú rutinokat aktivizálja. A LOAD gépi kódú rutinja a \$FFD5 címen található. Ahhoz, hogy ezt a rutint aktivizálni tudjuk, előbb az input/output fájl adatait és a fájl nevét kell közölnünk. Erre szolgálnak a SETLFS és a SETNAME rutinok. Ha nem akarunk ezzel bajlódni, akkor az adatokat átvehetjük a BASIC programból is a GETNAME rutinnal; címe: \$E1D4. Később még erre visszatérünk, most a táblázat alapján azt nézzük meg, hogy a gépi kódú LOAD és a BASIC LOAD között milyen hasonlóságok vannak.

OVERLAY gépi kódban

A továbbiakban bemutatunk egy gépi kódú programot (l. OVERLAY-ASP), mely a múlt havi cikkben ismertetett összes overlay funkciót megvalósítja egy programon belül. Ezenkívül nemcsak a program végére, hanem tetszés szerinti sorszámra is végrehajt APPEND-et. A program tetszés szerinti szabad tárterületre betölthető, és akár parancsból, akár utasításból hívható. Az itt közölt változat hívása a következő:
POKE780,KOD:SYS49152,,nev”,8

OVERLAY

```
10 FOR I = 0 TO 125: READ A: POKE 4915
2 + I,A:B = B + A: NEXT
20 IF B < > 13029 THEN PRINT "HIBAS
DATA ": END
30 PRINT "OVERLAY PROGRAM KESZ": END
40
100 DATA 134, 20, 132, 21, 32
102 DATA 212, 225, 173, 12, 3
104 DATA 10, 144, 71, 166, 43
106 DATA 164, 44, 169, 0, 133
108 DATA 185, 32, 213, 255, 176
110 DATA 50, 173, 12, 3, 41
112 DATA 1, 240, 23, 134, 45
114 DATA 134, 47, 134, 49, 132
116 DATA 46, 132, 48, 132, 50
118 DATA 173, 12, 3, 10, 144
120 DATA 5, 169, 0, 32, 94
122 DATA 166, 32, 51, 165, 173
124 DATA 12, 3, 10, 176, 1
126 DATA 96, 169, 0, 133, 157
128 DATA 32, 142, 166, 76, 174
130 DATA 167, 76, 249, 224, 162
132 DATA 14, 76, 58, 164, 10
134 DATA 176, 32, 10, 176, 13
136 DATA 10, 144, 242, 32, 19
138 DATA 166, 166, 95, 164, 96
140 DATA 24, 144, 170, 166, 174
142 DATA 164, 175, 56, 138, 233
144 DATA 2, 170, 152, 233, 0
146 DATA 168, 24, 144, 154, 166
148 DATA 45, 164, 46, 24, 144
150 DATA 237
```

SAVE MEMORY

```
1 PRINT "[CLR][CD][CD]"; SPC( 10); "
[RSV] SAVE MEMORY [CD][CD]"
2 INPUT "CUJKEZDO TARCIM:";A(4): IF
A(4) > 65535 THEN 2
3 PRINT
4 INPUT "CUJVEGSO TARCIM:";A(5):A(5)
= A(5) + 1: IF A(5) > 65535 THEN 4
5 INPUT " FILE NEVE:";F$(0)
6 FOR I = 0 TO 3:A(I) = PEEK(43 + I):
NEXT
7 POKE 44, INT(A(4) / 256): POKE 43,A(4) -
PEEK(44) * 256
8 POKE 46, INT(A(5) / 256): POKE 45,A(5) -
PEEK(46) * 256
9 SAVE F$(0),8,1
10 POKE 43,A(0): POKE 44,A(1): POKE 45
,A(2): POKE 46,A(3)
```


OVERLAY-ASP-

```

C000          1000      *:=C000
C000 06 14      1010      STX #14 ; SORSZAM
C002 04 15      1011      STY #15 ; TAROLAS
C004 20 04 E1   1020      JSR #E1D4 ; GET NAME
C007          1030      ;
C007 AD 0C 03   1040      LDA #030C
C00A 0A          1050      ASL A ; FUNKCIO
C00B 90 47      1060      BCC APPEND ; VALASZTAS
C00D          1070      ;
C00D A6 2E      1080      LDX #2B ; BETOLTESI
C00F A4 2C      1090      LDY #2C ; CIM=SOB
C011          1100      ;
C011 A9 00      1110      LOAD LDA ##00 ; MASODLAGOS
C013 85 B9      1120      STA #B9 ; CIM=0
C015 20 05 FF   1130      JSR #FFD5 ; LOAD
C018 00 32      1140      BCS ERROR ; HIBA VIZSGALAT
C01A          1150      ;
C01A AD 0C 03   1160      LDA #030C
C01D 29 01      1170      AND ##01 ; SOV ALLITAS
C01F F0 17      1180      BEQ LINKAD ; NEM KELL
C021          1190      ;
C021 06 2D      1200      STX #2D
C023 06 2F      1210      STX #2F
C025 06 31      1220      STX #31
C027 04 2E      1230      STY #2E ; SOV ALLITAS
C029 04 30      1240      STY #30
C02B 04 32      1250      STY #32
C02D          1260      ;
C02D AD 0C 03   1270      LDA #030C
C030 0A          1280      ASL A ; UGRAS
C031 90 05      1290      BCC LINKAD ; HA CHAIN
C033          1300      ;
C033 A9 00      1310      LDA ##00
C035 20 5E A6   1320      JSR #A65E ; CLR
C038          1330      ;
C038 20 33 A5   1340      LINKAD JSR #A533 ; UJRALANCOLAS
C03B          1350      ;
C03B AD 0C 03   1360      LDA #030C ; HA CHAIN
C03E 0A          1370      ASL A ; FOLYTATAS
C03F 00 01      1380      BCS START ; AZ SOB-N
C041 00          1390      RTS
C042          1400      ;
C042 A9 00      1410      START LDA ##00
C044 85 9D      1420      STA #9D ; RUN FLAG
C046 20 8E A6   1430      JSR #A68E ; CHRGET PTR
C049 4C AE A7   1440      JMP #A7AE ; RUN
    
```

```

C04C 4C F9 E0   1450      ERROR JMP #E0F9 ; LOAD HIBA
C04F          1460      ;
C04F A2 0E      1470      ERROR2 LDX #14 ; KOD HIBA
C051 4C 3A A4   1480      JMP #A43A
C054          1490      ;
C054          1500      ;
C054 0A          1510      APPEND ASL A ; APPEND FUNKCIO
C055 00 20      1520      BCS ATOSOV ; VALASZTAS
C057 0A          1530      ASL A
C058 00 0D      1540      BCS ATOE0F
C05A 0A          1550      ASL A
C05B 90 F2      1560      BCC ERROR2
C05D          1570      ;
C05D 20 13 A6   1580      JSR #A613 ; APPEND
C050 A6 5F      1600      LDX #5F ; SORSZAMRA
C062 A4 60      1610      LDY #60
C064 18          1615      CLC
C065 90 AA      1620      BCC LOAD
    
```

OVERLAY-ASP-

```

C067          1630      ;
C067 A6 AE      1650      ATOE0F LDX #AE ; APPEND
C069 A4 AF      1660      LDY #AF ; FILE VEGRE
C06B          1670      ;
C06B 38          1680      MINUS SEC
C06C 0A          1690      TXA ; CIM-2
C06D E9 02      1700      SBC ##02
C06F AA          1710      TAX
C070 98          1720      TYA
C071 E9 00      1730      SBC ##00
C073 A8          1740      TAY
C074 18          1745      CLC
C075 90 9A      1750      BCC LOAD
C077          1760      ;
C077 A6 2D      1770      ATOSOV LDX #2D ; APPEND
C079 A4 2E      1780      LDY #2E ; SOV-RA
C07B 18          1785      CLC
C07C 90 ED      1790      BCC MINUS
C07E          1800      ;
C07E          1810      .END
    
```

```

ZEILEN:04 SYMBOLE:9 FEHLER:0
APPEND=C054 ATOE0F=C067 ATOSOV=C077 ERROR=C04C ERROR2=C04F LINKAD=
LOAD=C011 MINUS=C06B START=C042
    
```

Szemléletes az alábbi táblázat:

Parancs mód	Betöltés kezdőcíme	SOV változás	EOF változás	PRG. folytatás
LOAD"név",8	SOB.	I	I	READY
LOAD"név",8,1	LD.PTR.	I	I	READY
Utasítás mód				
LOAD"név",8	SOB.	N	I	SOB.
LOAD"név",8,1	LD.PTR.	N	I	SOB.
Gépi kód				
JSR \$FFD5	X—Y/LD.PTR a SETLFS által megh.	N	I	PC+1

Új rövidítések:

LD.PTR. a programfájlban tárolt mutató, PC utasításszámláló regiszter
a betöltési címre

X—Y a CPU X Y regiszterei

EOF az utolsó betöltött bajt címe a 174—175 tárcímeken

A KOD a kívánt overlay funkciót adja meg az alábbiak szerint:

bitek: 128 CHAIN
64 APPEND a SOV által mutatott címre
32 APPEND az utoljára betöltött program végére
16 APPEND adott sorszámmra
1 SET SOV

Amelyik bit értékét 1-re állítjuk, azt a funkciót hajtja végre a program. A vizsgálat a 128-as bittől kezdődik. Ha az 1-es bit is 1-re van állítva, akkor a program betöltése után a SOV beáll a program végére, egyébként változatlan marad.

Újdonság, hogy adott BASIC sorra is tudunk APPEND-et végrehajtani. Ehhez az előzőkön kívül még az alábbi módon meg

kell adni azt a sorszámot is, amelyikre már betölthető a hívott program:

POKE780,KOD:POKE781,L:

POKE782,H:

SYS49152,,név",8

Az L és H a sorszám alacsony és magas helyértékű bajtjai. Ha van ilyen sorszám a hívó programban, akkor attól kezdve, ha nincs, akkor az ennél nagyobb sorszámú sorok törlődnek, és helyüket a hívott program fogja elfoglalni. Lényeges könnyebbég még a BASIC APPEND megoldásokhoz képest, hogy a program betöltése után a vezérlés a hívást jelző SYS utasítás utáni BASIC sorra és nem a program elejére kerül.

A gépi kódú program a következő ROM-rutinokat használja:

\$E1D4 GETNAME a fájlnev és egység szám átvétele

\$FFD5 LOAD a gépi kódú load

\$E0F9 DISK hiba kiírása

\$A43A hibajelzés, ha a 780-ra írt kód nem értelmezhető

\$A65E CLR utasítás rutinja

\$A533 BASIC sorok újraláncolása

\$A68E a vizsgált BASIC bajt címbéállítás

\$A7AE RUN

A program használatát szemléltetik az ADATBEVITEL, MENÜ, ÖSSZEGZÉS és KIÍRÁS nevű egyszerű programok.

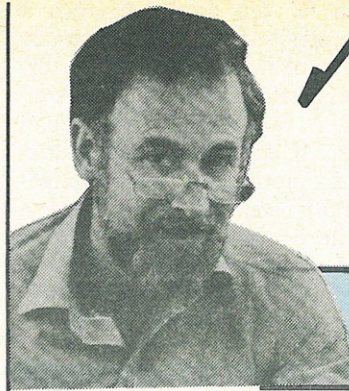
— Töltsük be és indítsuk el az OVERLAY nevű programot.

— Töltsük be és indítsuk el az ADATBEVITEL nevű programot. Az adatbevitelt 3 szám beírása jelképezi. A 150-es sor a program helyére kéri a MENÜ programot, és azt elindítja úgy, hogy a beírt adatok megmaradjanak.

— A MENÜ program elkészíti az adatfeldolgozás menüjét, majd a választásunktól függő alprogramot hívja be vagy a saját helyére, vagy a menüprogram utolsó, most már fölöslegessé vált sorainak helyére.

A program magját jól megfigyelhetjük a 300-410-es sorokban: behívjuk az éppen szükséges szubrutint, azt végrehajtjuk, majd visszatérünk a menühöz, ahol az előbb használt szubrutin helyére egy másikat hívhatunk be. Természetesen az egész eljárás előnye csak akkor jelentkezik, ha az itt alkalmazott szemléltető szubrutinok helyett valódi, komplett feladatokat ellátó programokat kapcsolunk össze rendszerré a bemutatott eljárásokkal.

ZSOM BÉLA



Mit tud

A

PASCAL?

ADATBEVITEL

```
100 POKE 45,0: POKE 46,32: CLR
110 PRINT "[CLR][CD][CD]"; SPC( 10); "A
    DAT BEVITEL"
120 PRINT "[CD][CD]IRJ BE 3 SZAMOT":
    PRINT
130 DIM A(2): FOR I = 0 TO 2: PRINT I
    + 1; ".SZAM": INPUT A(I): NEXT
150 POKE 780,128: SYS 49152"MENU",8
```

MENU

```
100 GOSUB 1500
102 PRINT "[CLR]"
105 PRINT "[HOME][CD][CD][CD]"; SPC( 1
    5); "MENU": PRINT
110 FOR I = 0 TO 3: PRINT I;ME$(I):
    NEXT
120 GET A$: IF A$ = "" THEN 120
130 A = VAL(A$): IF A > 3 THEN 120
135 :
140 PRINT "VARJ": ON A + 1 GOTO 200,30
    0,400,500
150 :
200 POKE 780,129: SYS 49152"ADATBEVITE
    L",8
205 :
300 POKE 780,16: POKE 781,0: POKE 782,
    4: SYS 49152"OSSZEGZES",8
310 GOSUB 2000: GOTO 102
320 :
400 POKE 780,16: POKE 781,0: POKE 782,
    4: SYS 49152"KIIRAS",8
410 GOSUB 2000: GOTO 105
420 :
500 END
510 :
1500 DIM ME$(3): FOR I = 0 TO 3: READ
    ME$(I): ME$(I) = ME$(I) + " ": NEXT
    : RETURN
1510 DATA BEVITEL, OSSZEGZES, KIIRAS, VE
    GE
```

OSSZEGZES

```
2000 B = 0
2010 FOR I = 0 TO 2: B = B + A(I):
    NEXT
2020 RETURN
```

KIIRAS

```
2000 PRINT "ADATOK:"
2010 FOR I = 0 TO 2: PRINT A(I): NEXT
2020 PRINT "OSSZEG:";B
2030 RETURN
```

A Pascal nyelv 1968 és 1973 között alakult ki. Születését az IFIP (International Federation for Information Processing) ALGOL munkacsoportjában létrejött ellentétnek köszönheti. Az új ALGOL-lal kapcsolatos vita eredménye nemcsak — egy azóta már majdnem elfelejtett nyelv — az ALGOL-68 kidolgozása volt, hanem a munkacsoport több prominens tagjának a csoportból való kiválását is kiváltotta. Ezek egyike, N. Wirth, a Zürichi Műszaki Egyetem professzora indította el a Pascalt világhódító útjára. Az alapnyelv végleges formájának kialakulása mintegy 5 évet vett igénybe. A Wirth-féle ún. standard Pascal mellett számos — többnyire bővítések és praktikus eszközöket tartalmazó — nyelvjárás is kialakult. Ezek száma és jelentősége a mikroelektronika előtérbe kerülésével egyre nőtt, és ma már ott tartunk, hogy a Pascal minden professzionális és több hobbi gépen egyaránt használható. Mivel ez (a BASIC-en kívül) nem sok nyelvről mondható el, érdemes vele egy összefoglaló áttekintés erejéig megismerkedni.

Alapelvek

A főfoglalkozásban programozás oktatásával foglalkozó Wirth professzor a jó programok titkát a megfelelő algoritmus és a célszerűen választott adatszerkezet gondos ötvözetében találta meg. Ez azt jelenti, hogy a programozónak elsősorban azt kell tisztáznia, hogy milyen adatokkal, mit akar csinálni. Ha ez a két döntés jó, a „hogyan” (t.i. a használt programozási nyelv) kérdése már kisebb probléma, hiszen nyelvekben nagy a választék. Ha ez így van, akkor jogos a kérdés: milyen objektív indok szolgált egy újabb nyelv, a Pascal létrehozása mellett? A választ a nyelv ismeretében nem nehéz megmondani. A Pascal nemcsak az algoritmusok, hanem az adatszerkezetek létrehozásának eszközeit is tartalmazza elődeinél — és talán eddigi utódainál is — általánosabb, de mégis könnyen használható formában. Ennek a jelentőségét azok értékelik legjobban, akik BASIC-ben, COBOL-ban, FORTRAN-ban vagy PL/I-ben dolgozva a napok neveit kódolva használták és külön utasításokkal kellett ellenőrizni, hogy egy dátumban az év, a hónap és a nap értéke logikailag elfogadható-e. Ilyen esetekben Pascalt használva egyszerűen létrehozhatunk például egy „nap” típust, és ezzel együtt előírhatjuk a majdani nap típusú változók értékészletét is, célszerűen a Hétfő, Kedd, ... Vasárnap felsorolással. Hasonló módon külön típusként kezelhető és rögzített értékészlettel rendelkezhet a többi adat is.

A Pascal pilléreit néhány egyszerű, jól kiforrott és a modern programozás alapelveit szem előtt tartó fogalom alkotja:

— Típusdefinió: A programban szereplő, a nyelv ún. standard típusaitól különböző, tehát a programozó által a feladat megoldásához legjobbnak ítélt típusok leírása. Újabb típust csak a már létező típusra építve hozhatunk létre, kiindulásként az alap vagy standard típusok szolgálnak.

— Deklaráció: Hivatkozás előtt fel kell sorolni egyrészt a programban használt változókat, másrészt a választott algoritmus megvalósításához használt alprogramokat, ha ezek nem tartoznak a nyelv standard alprogram-készletéhez. Az alprogramok eljárások vagy függvények lehetnek. A változók deklarációjában meg kell adni a változó nevét és típusát, egyes Pascal változatokban kezdőértéket is elő lehet írni. Az alprogramok belső szerkezete megegyezik a főprogram szerkezetével. Minden alprogram saját típusdefiniciókat, változó és alprogram deklarációkat, valamint önálló programtörzset tartalmazhat. Azt, hogy egy feladat megoldásához milyen mély programstruktúrára van szükség, a rendszer tervezésekor az algoritmus és az adatszerkezetek meghatározásával együtt kell eldönteni.

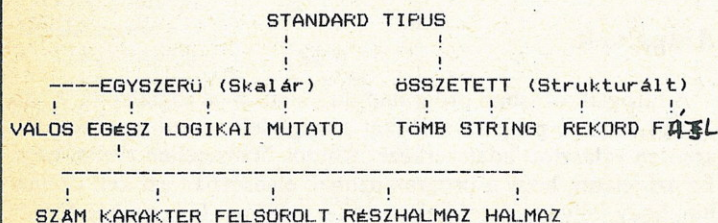
— Jól strukturált programszerkezet: A Pascal utasításai hatékonyan támogatják az áttekinthető, világos szerkezetű és ezért könnyen módosítható programok írását. A strukturált programozás általános elveinek betartását a vezérlőszerkezetek biztosítják. Az utasítás fogalma a programhoz hasonlóan tetszőlegesen mély, strukturált szerkezetet takarhat. A nyelv tartalmazza egyrészt az utasítások egy alapkészletét, másrészt egy arra szolgáló eszközt, hogy több utasításból egyetlen ún. összetett utasítást hozunk létre. Egy összetett utasítás pedig újabb utasítások kialakításánál ugyanúgy alkalmazható, mint a közönséges utasítások. Mivel a típusdefiniciókhoz és a deklarációkhoz hatásköri szabályok is tartoznak, az alprogramok környezetüktől teljesen függetlenek lehetnek, a kapcsolattartás a feladat követelményeihez igazodó paraméterek segítségével történik.

Típusok

A Pascal talán legjelentősebb újítása volt, hogy bevezette a felhasználó által definiálható típusokat. Így a nyelvben használható típusokat két nagy csoportba lehet sorolni:

- Standard, vagyis előre definiált típusok.
- Felhasználói, vagy saját típusok.

Mivel új típust csak a már definiált típusokból lehet felépíteni, először a standard típusokkal célszerű megismerkedni.



A standard típusok nagy része közismert. A valós illetve egész számok, a logikai értékek, a tömbök, stringek (karakterláncok), de még a rekordok és fájlok is ismerősen hangzanak más nyelveket ismerők számára. Ezekkel kapcsolatban a Pascal annyi újdonságot hozott, hogy az ésszerűség határain belül biztosítja az ún. „ortogonalitást”: az összetett típusok komponensei jóformán bármilyen típusúak lehetnek. Például létrehozhatjuk stringekből, logikai változókból és egész számokból álló rekordokból felépített tömbök fájlját, ha ez az adatszerkezet illeszkedik legjobban feladatunkhoz. Az olyan ésszerűtlen kombinációk, mint pl. fájlok tömbje természetesen nem megengedett. Az ortogonalitás értelmében a több dimenziós tömbök kézenfekvő módon tömbök tömbjeként hozhatók létre.

A MUTATÓ típus a gépi kód és az assembly nyelvek szintjén közismert címaritmetikát hozza be a nyelvbe, természetesen a kényelmes használatot biztosító magas szintű eszközökkel együtt. A legfontosabb e típusnál a tetszőleges, előre definiált struktúrájú tárterületek dinamikus (a program futása alatti) lefoglalása és felszabadítása.

Az EGÉSZ-ből származtatott típusok közös tulajdonsága,

hogy kölcsönösen egyértelmű a megfeleltetés az ilyen típus érték-készlete és a természetes egész számok között.

A FELSOROLT típus esetén az érték-készletet egyszerűen fel kell sorolni.

Például, ha a hónapok neveivel akarunk dolgozni:

TYPE hónap = (jan, febr, márc, ápr, máj, jún, júli, aug, szept, okt, nov, dec);

A RÉSZHALMAZ típus csak rövidíti a felsorolást olyan esetekben, amikor egymás utáni értékek következnek.

Tekintsük például a következő típusdefiniciókat:

TYPE nap = (hétfő, kedd, szerda, csüt, péntek, szombat, vas);

munkanap = hétfő .. péntek;

dátum = 1 .. 31;

Természetesen amikor a programban majd munkanap vagy dátum típusú változókat használunk, automatikus ellenőrzés dönti el, hogy értékük a típusdefiniciónak megfelelő-e.

A HALMAZ típus szintén az egészek származékain alapszik és abban különbözik a FELSOROLT, illetve a RÉSZHALMAZ típustól, hogy elemei között sorrend nem értelmezett, a típusdefinició-felsorolással vagy részhalmozokkal csupán azt rögzíti, hogy mely — azonos típusú — elemek tartoznak az adott típusú halmazhoz.

A példák magukért beszélnek:

TYPE Nagy Betű = SET OF "A" .. "Z";

Páratlan = SET OF (1, 3, 5, 7, 9);

A saját típusok bemutatására tekintsünk egy menetrend készítésénél szóba jövő adatszerkezetet:

TYPE Idő = RECORD

Óra: 0 .. 23;

Perc: 0 .. 59

END

Dátum = RECORD

Év: 1900 .. 1999;

Hó: (jan, febr, márc, ápr, máj, jún, júli, aug, szept, okt, nov, dec);

Nap: 1 .. 31

END

Járat = RECORD

Mikor: Dátum;

Honnan: string 25;

Hova: string 25;

Indul: Idő;

Érkezik: Idő;

END;

A Járat típus itt egy további rekordból (Dátum és Idő), valamint stringekből (Honnan, Hova) álló rekord. Most deklarálnak egy Járat típusú változót is, melynek neve pl. Vonat:

VAR Vonat: Járat;

A Vonat egyes mezőire a Vonat.Mikor, Vonat.Hova stb. azonosítókkal hivatkozhatunk.

Már egy ilyen viszonylag egyszerű példa is jól érzékelteti, hogy a saját típusok valóban komoly segítséget nyújtanak az adott feladathoz jól illő adatszerkezet megvalósításához.

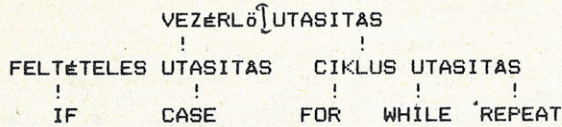
Utasítások

A Pascal utasítások egyik csoportja a standard típusú változókra használható alpműveleteket tartalmazza. Az aritmetikai, logikai és relációs műveletek más nyelvekből ismerősek, mi csak a HALMAZ típusra térünk ki. Itt egy új művelet, az *in* szolgál annak eldöntésére, hogy valami egy adott halmaz eleme-e. Az eredmény természetesen logikai típusú. Halmazokra a +, - és a * a megfelelő halmazműveletet jelöli (unió, különbség és közös rész), az = művelet két halmaz elemenkénti egyenlőségét dönti el, a > = reláció pedig akkor ad igaz (true) eredményt, ha az első halmaz tartalmazza a másodikat.

A változók és műveletek segítségével kifejezések, ezekből és egy újabb változóból pedig értékadó utasítások építhetők fel. Az értékadás műveleti jele a := .

Az utasítások számunkra érdekesebb csoportját az ún. vezérlő-utasítások alkotják. A vezérlőutasítások megértéséhez azonban

szükség van a már említett összetett utasítás fogalmára. Az összetett utasítás definíciója igen egyszerű: ha bármilyen utasítássorozatot a BEGIN és END jelek közé zárunk, abból egyetlen — összetett — utasítás keletkezik. A következőkben, ha utasításról beszélünk, gondoljunk mindig az összetett utasításra is. A vezérlőutasítások csoportosítása a következő lehet:



Az IF utasítás egy kétágú elágazást hoz létre:

```
IF feltétel THEN utasítás-1 ELSE utasítás-2;
```

Az IF utasítás kétágú elágazást hoz létre:

```
IF feltétel THEN utasítás-1 ELSE utasítás-2
```

Ha a feltétel igaz, *utasítás-1*, különben *utasítás-2* hajtódik végre. Ne felejtjük el, hogy ezek összetett utasítások vagy akár újabb IF utasítások is lehetnek.

A CASE utasítás egy tetszőlegesen sokágú elágazást eredményez, melyből legfeljebb egy ág hajtódik végre. A választás egy tetszőleges kifejezés felsorolt értékei alapján történik:

CASE kifejezés OF

érték-1: utasítás-1;

érték-1: utasítás-2;

...

...

érték-n: utasítás-n;

END,

A kifejezést szelektornak is hívják. A felsorolt értékek típusa természetesen csak a szelektoréval megegyező lehet. Ha ez az érték nem egyenlő a felsorolt értékek egyikével sem, az utasítás hatástalan. Ilyen esetek ésszerűbb kezelése érdekében sok Pascal változatban *érték-n* után egy ELSE ágat is megengednek.

A ciklusutasítások egy utasítás, illetve egy programrész ismételt végrehajtását teszik lehetővé. A három ciklusutasítás egymástól elsősorban a ciklus ismétlései számának meghatározásában különbözik. Ha előre ismerjük a ciklus futásainak számát, a FOR utasítást célszerű alkalmazni:

FOR változó:= kifejezés TO érték DO utasítás;

Itt a *változó* egész vagy ebből származtatott típusú, neve: ciklusváltozó. A ciklusváltozó a ciklus elején felveszi a *kifejezés* értékét és a ciklus minden ismétlésekor a típusa szerint meghatározott következő értéket kapja. Az *utasítás* — melyet a ciklus magjának is neveznek — minden végrehajtása előtt ellenőrzésre kerül, hogy a ciklusváltozó nem haladja-e meg a TO és a DO között álló ún. *végértéket*. Ha nem, a ciklus folytatódik, ha igen, a FOR utasítás végrehajtottnak tekintendő.

FOR nap:= hétfő TO vasárnap DO utasítás;

Ha nap egy FELSOROLT típusú változó, ez a ciklus 7-szer fut, miközben nap értéke rendre hétfő, kedd, ... vasárnap lesz.

A WHILE utasítás egyszerűbb:

WHILE feltétel DO utasítás;

Ameddig a *feltétel* igaz, az *utasítás* (ciklusmag) ismételten végrehajtásra kerül, de minden ismétlés előtt megtörténik a *feltétel* újbóli kiértékelése.

A REPEAT utasítás abban különbözik a WHILE ciklustól, hogy az ismétlődést szabályozó feltétel vizsgálata a ciklus végén megy végbe:

REPEAT utasítássorozat UNTIL feltétel;

Itt a REPEAT és az UNTIL között több utasítás is állhat, melyek legalább egyszer végrehajtottak. A ciklus törzse addig ismétlődik, amíg a *feltétel* igaz nem lesz.

Eljárások és függvények

A korábban közös néven alprogramoknak nevezett eljárások és függvények a Pascal leghatékonyabb eszközei közé tartoznak. Az eljárás és a függvény között az a különbség, hogy amíg az eljárás csak paraméterek átvételére és átadására képes, a függvény egy további értéket is hordoz, melyet saját nevével azonosít. Ezért az eljárás hívása önálló utasításban, a függvények pedig kifejezésekben aktivizálhatók. Az alprogramok fontos tulajdonsá-

ga, hogy megvédik belsejüket a környezettől. Ez azt jelenti, hogy egy eljárásról vagy függvényről belül saját típusokat definiálhatunk és saját változókat deklarálhatunk. Ezek még akkor is különböznek az eljárást tartalmazó program típusaitól és változóitól, ha nevük megegyezne. Az alprogram belsejében deklarált, ún. lokális változók természetesen csak az eljárás vagy a függvény végrehajtása folyamán „élnek”, kilépéskor megszűnnek létezni.

Az alprogramok a „külvilággal” paraméterek útján tartanak kapcsolatot. A deklarációnál használt paramétereket formális paramétereknek nevezzük, mivel ezek csak azt határozzák meg, hogy majd az alprogram hívásakor mennyi és milyen típusú aktuális paramétert kell megadni. Pascalban a paraméterek átadásának két módszere használható: az érték szerinti átadásnál az alprogramba az aktuális paraméter értéke kerül, míg a név szerinti paraméterek esetében az alprogram törzsébe az aktuális paraméter neve (vagy erre való hivatkozás) helyettesítődik be. Egy eljárásból eredményt kivinni nyilván csak név szerinti paraméterrel lehet. Bár elvileg az érték szerinti paraméterátadás az egyszerűbb (és a megfelelő program is rövidebb), akkor is érdemes név szerinti átadott paramétereket használni, ha a paraméter valamilyen terjedelmesebb adatszerkezet (pl. egy tömb vagy rekord), mert ezek átmásolása az alprogram minden hívásakor igen idő (és hely-) igényes lehet.

Inkább haladó programozók számára szólunk a rekurzív eljárásokról és függvényekről. Egy alprogram akkor rekurzív, ha saját törzsében önmaga hívását tartalmazza. A Pascal ilyen alprogramok írását megengedi, hiszen a rekurzió gyakran nagyon egyszerű logikájú algoritmusokhoz vezet. Ennek viszont az az ára, hogy a program futási ideje és tárigénye általában megnő.

Tekintsünk például egy rekurzív algoritmust egy string hosszának meghatározására:

— Ha a string üres, a hossza 0.

— Ha nem üres, akkor a hossza eggyel több, mint az egy karakterrel rövidebb string hossza.

Az algoritmus megvalósításához csupán egy olyan függvényre van szükségünk, amelyik egy stringből elhagyja annak első karakterét.

Mint már említettük, a Pascal fegyvertárához számos előre definiált, ún. standard eljárás és függvény tartozik. Standard eljárások szolgálnak egyes adattípusok kezelésére, matematikai függvények értékeinek meghatározására és főleg a külvilággal való kapcsolattartásra a fájlok segítségével.

Az egyes Pascal változatok — egymással versengve — bővebb és bővebb standard alprogramkészletet kínálnak. A jelenleg egyik legnépszerűbb Pascal rendszer, a TURBO PASCAL standard alprogramjai között például egy jól felépített grafikus alrendszer is szerepel.

BAKOS TAMÁS

DATORG

Felajánlunk megvételre egy jó állapotban lévő Siemens 404/2 típusú 16 munkahelyes adatrögzítő rendszert, amely az alábbi egységekből áll:

- 1 db Siemens 404/2, 64 KByte-os központi egység
- 16 db Siemens TD 8153 típusu terminál
- 2 db CDC 564 típusu mágneslemez-egység
- 1 db RCA 432 típusu mágnesszalag-ikeregység
- 1 db Siemens operátori konzol

A számítógép DEOS V 2.10 operációs rendszer alatt adatrögzítési feladatok ellátására alkalmas.

Érdeklődni lehet: Turányi Károly munkatársunknál.

Bp., V. Dorottya u. 6. Tel.: 186 608.

RAINBOW a μ M-ban

Mára a számítógépek egyik fokmérője lett az IBM-kompatibilitás. Ámbár arról, hogy a gyártók által reklámozott kompatibilitás valójában mit takar, általában nem esik szó (lásd következő számunkban A zűrzavartól a zsbongásig c. cikket). Az IBM-kompatibilitás ma — nekünk — sokba kerül. Éppen ezért, de nemcsak ezért, sokan nem engedhetjük meg magunknak, hogy meglevő, nem kompatibilis rendszerünket felszámoljuk, és áttérjünk egy merőben újra. Nos, az e havi számunkban közölt RAINBOW-cikk azoknak lesz hasznos, akik ez utóbbiak közé tartoznak, ugyanakkor szükségük lenne IBM gépen futó anyagok átültetésére saját rendszerükbe.

A közölt program IBM PC-n, MS-DOS-ban írt ASCII szövegfájlokat konvertál a TRS-80 Color Computer által olvasható ASCII szövegfájlokká. A TRS-80 Color Computer sem IBM-kompatibilis. A programmal mégis van két alapvető baj. Az egyik hogy csak Color Computerre konvertál, más géptípusra nem. Ezen a problémán viszonylag könnyű segíteni: elegendő, ha ismerjük a CoCo lemezkezelő struktúráját, valamint a saját, eltérő típusú gépünkét. (A CoCo lemezkezelésének ismertetését a következő számunkban hozzuk. — A szerk.) A másik baj, hogy a program szigorúan a szövegfájlok konvertálására való. Vagyis, ha a mi gépünk nem alkalmas programok letárolására ASCII formátumban, illetve az ASCII formátumban tárolt programoknak futtatható programokként való felismerésére, a konvertálás nem fog sikerülni.

Ha ezek az akadályok elhárultak, és tudunk olvasni a saját gépünkön is MS-DOS lemezeket, futtatható programhoz még mindig nem jutottunk a szintaxishibák miatt. Ezek javítása akkor fog gondot jelenteni, ha a gépünk BASIC-je messze áll az IBM PC BASIC-jétől.

ASCII szövegfájlok konvertálása MS-DOS lemezről CoCo lemezre

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

Akár mennyire is CoCo-rajongók vagyunk, el kell ismernünk, hogy az üzleti életben messze a legelterjedtebb személyi számítógép az IBM PC. Ezt „bűvöljük” a hivatalban, vagy ha mi nem is, valamelyik ismerősünk biztosan. Az IBM PC operációs rendszere az MS-DOS (Microsoft Disk Operating System). Most egy olyan eszközt adunk az érdeklődőknek, amellyel egy MS-DOS-ban készített ASCII formátumú anyagot a CoCo-val olvasható ASCII formátumú anyaggá lehet konvertálni.

A következő számban arra közlünk megoldást, hogyan formáljunk MS-DOS-ban

futtatható lemezt a CoCo-n, és hogyan írjunk szövegfájlt az ilyen lemezre.

A program 64 k RAM-ot és két lemez-meghajtót igényel. A két lemez-meghajtóra a másolási sebesség elviselhető határon belül tartása miatt van szükség. 40 sávós lemez-meghajtók kellene, mivel az MS-DOS is 40 sávot használ.

A fájlok átírásának nehézségei

A Color Computer lényeges eltéréseket mutat az IBM PC-hez képest. Az RS-DOS (Radio Shack Disk Operating System) lemezformátuma 35 sáv, sávonként 18 szektor, szektoronként 256 bájt. Az MS-DOS 40 sávval dolgozik, sávonként 9 szektor van, szektoronként 512 bájt a kapacitása. Az RS-DOS egyoldalú, az MS-DOS kétoldalú lemez-meghajtókat kezel. A CoCo lemezkezelő chipje Western-Digital vagy Fujitsu, az IBM PC-é NEC gyártmányú. Ezek a különbségek együttesen okozzák a nehézségeket. Az a tény pedig, hogy az IBM PC-n futó szövegszerkesztők által előállított szövegek nem pontosan ASCII formátumúak, csak tovább szövi a bonyodalmakat. A legfontosabb tehát, hogy a másolandó anyagról mindenképp meg kell állapítanunk, hogy tisztán ASCII formátumú-e. Szerencsére megvan a lehetőség a legtöbb nem teljesen ASCII formátumú szöveg igazi ASCII formátumúvá való átírására.

Kedvező, hogy mindkét számítógép ugyanazzal a méretű lemezzel (5 1/4") dolgozik, hogy a Microsoft cég írta mind az MS-DOS-t, mind a Color Computer DISK BASIC-jét, továbbá, hogy a Western-Digital vagy Fujitsu lemezkezelő chip képes kezelni bármit, amit a NEC lemezkezelő ír. (Sajnos, fordítva a dolog nem igaz. Így, amikor egy CoCo-anyagot MS-DOS lemezre kívánunk küldeni, figyelembe kell venni a NEC chip korlátait.)

A legtöbb MS-DOS szövegfájl a sorok végén egy kocsi vissza karaktert (\$0D), plusz egy soremelő karaktert (\$0A) tartalmaz. A CoCo-n használt szövegszerkesztők viszont csak kocsi vissza (\$0D) karakterrel záródó sorokat írnak. Ha olyan szövegfájlt akarunk a CoCo-n olvasni, amelyben a soremelő \$0D-vel és \$0A-val záródnak, a szövegszerkesztők egy része automatikusan eltávolítja a sorok végéről a soremelő karaktert. Ilyen például a Telewriter-64. Vannak azonban olyan editorok is (például a Macro 80C), amelyek ilyen esetben nem boldogulnak a fájjal. Általánosságban tehát hasznos, ha a szövegfájl átírásakor a sorok végéről a soremelő karaktert kiszűrjük. A programunk tartalmazza ezt a „szűrőt”, melynek van még egy funkciója. Nevezetesen, hogy az MS-DOS fájl valamennyi karakterének legfelső bitjét θ -ra állítja, mielőtt a CoCo lemezformátumra való átírás megtörténne. A karakterek felső bitjének θ -ra állítása a WordStar-ral vagy más hasonló jellegű szövegszerkesztővel

```
0 REM *****
1 REM *
2 REM *TEXT FILE TRANSFORMATION*
3 REM * FROM MS-DOS TO COCO *
4 REM *
5 REM * THE RAINBOW *
6 REM * JUNE, 1986 *
7 REM *
8 REM *****
9 REM
15 CLEAR 512,&H5DFF
18 DIM LKS(8),NTRYLC(8)
20 IS=&H60:ID=&H6000 *MS-DOS DATA SECTOR
  BUFFER
25 FS=&H62:FD=&H6200 *FILE ALLOCATION TA
  BLE BUFFER
30 DS=&H66:DD=&H6600 *MS-DOS DIRECTORY B
  UFFER
35 DEND=&H71FF *END OF DIRECTORY
40 FOR J=&H7E00 TO &H7E15
41 READ A$:A=VAL("&H"+A$):POKE J,A
42 NEXT J
43 DATA 8E,60,0,A6,B4,B1,0A,26,3,4F,20,2
  ,84,7F,A7,80,8C,62,0,25,EE,39
60 H=PEEK(&HC004):L=PEEK(&HC005):DKON=H*
  256+L

100 CLS:PRINT TAB(8);"MS-DOS ==> COCO"
105 PRINT TAB(6);"TEXT FILE FORDITO":P
  RINT TAB(3);"EGYOLDALAS MS-DOS LEMEZHEZ"
110 PRINT:PRINT" (C) NALAS GOODMAN JAN 1
  , 1986":PRINT
115 PRINT:PRINT TAB(4);"8 VAGY 9 SECTOR/
  TRACK-ES":PRINT TAB(10);"RENDSZERRE":P
  RINT" A "ROOT DIRECTORY" FILE-OKRA"
145 PRINT:PRINT:PRINT"HELYEZZE AZ MS-DOS
  LEMEZT A "DRIVE 1"-BE, MAJD NYOMJ
  A MEG AZ >ENTER< GOMBOT!"
190 IF INKEY$<>CHR$(13) THEN 190

200 REM READ IN FIRST SECTOR OF FAT
210 POKE &HEA,2:POKE &HEB,1:POKE &HEC,0:
  POKE &HED,2:POKE &HEE,FS:POKE &HEF,0
220 EXEC DKON
230 IF PEEK(&HF0) THEN 9000

300 REM 8 VAGY 9 SECTOR?
310 GN=0:GOSUB 15000:T=CV AND 15
320 TS=0
330 IF T=&HOE THEN TS=8:GOTO 400
340 IF T=&HOC THEN TS=9:GOTO 450
370 CLS:PRINT@231,"ROSSZ MS-DOS DISK"
380 PRINT">ENTER< GOMBBAL UJRA INDITH
  AT"
390 IF INKEY$<>CHR$(13) THEN 390
395 GOTO 100
400 REM 8 SECTOR/TRACK DIRECTORY
410 FOR N=4 TO 7
420 POKE &HED,N
430 POKE &HEE,DS+(2*N)-8
435 EXEC DKON
440 IF PEEK(&HF0) THEN 9000
445 NEXT N
447 GOTO 500

450 REM A "FAT" 2. SEKTORA + VALAMENNYI
  DIR
455 FOR N=5 TO 9
460 POKE &HED,N
465 POKE &HEE,DS+(2*N)-12
470 EXEC DKON
475 IF PEEK(&HF0) THEN 9000
480 NEXT N

500 REM DIRECTORY A DISPLAY-RE
510 K=0:LKS(0)=0:Z=0
520 CLS:PRINT TAB(8);"DIRECTORY"
530 PRINT:PRINT" A FILE NEVE HOSSZA
  "
540 GOSUB 13000
550 IF FE=0 THEN 700
555 IF Z=0 THEN 750
560 F$="M" *DIRECTORY KOZEPE
570 PRINT@427,"MELYIKET?"
580 PRINT"(A NYILAKKAL LAPOZHAT)"
600 Z=Z+1:LKS(Z)=K
610 A$=INKEY$
615 IF A$="" THEN 610
620 IF A$=CHR$(10) THEN 680 *LE
625 IF A$=CHR$(94) THEN 660 *FEL
630 IF VAL(A$)=0 OR VAL(A$)>8 THEN 610
635 CLS:VV=VAL(A$)-1:GOSUB 16000:GOSUB 1
  7000
640 PRINT:PRINT NAM$,FZ
650 PRINT:PRINT"HA MEGFELEL, NYOMJA MEG
  AZ >ENTER< GOMBOT, HA NEM, VALAME-
  LYIK MASIKAT."
```



```

656 A$=INKEY$:IF A$="" THEN 656
657 IF A$=CHR$(13) THEN 2000
659 Z=Z-1:K=LKS(Z):GOTO 520
660 IF FE=0 THEN 610
662 K=LKS(Z):GOTO 520
680 IF Z=1 THEN 610
682 Z=Z-2:K=LKS(Z):GOTO 520
700 PRINT@430,"---"
710 PRINT TAB(11);"MELYIKET?"
720 PRINT"(A 'LE' NYILLAL VISSZALAPOZHAT
)";
730 GOTO600
750 PRINT@423,"A DIRECTORY ELEJE"
760 PRINT TAB(11);"MELYIKET?"
780 PRINT"(A 'FEL' NYILLAL LAPOZHAT)";
790 GOTO 600
2000 REM FILE FORDITO RUTIN
2020 HCLU=40*TS
2030 CLS:PRINT:PRINT"HELYEZZE A COCO LEM
EZT A 'DRIVE 0'-BA, MAJD ADJA M
EG A FILE NEVET:"
2050 PRINT"(MAX. 8 KARAKTER HOSSZBAN)"
2060 PRINT:INPUT CFN$
2070 IF CFN$="" THEN 2030
2075 IF LEN(CFN$)>8 THEN 2030
2080 OPEN "0",#1,CFN$+ ".TXT":0
2090 CURCLU=BC
2092 IF TS=8 THEN QQ=5
2093 IF TS=9 THEN QQ=7
2095 ZCLU=INT(FZ/512)+1
2097 CLS:PRINT@232,"MS-DOS ==> COCO"
2098 PRINT@416,"CLUSTER TRACK SECTOR #BY
TE"
2100 FOR M=1 TO ZCLU
2110 IF CURCLU>HCLU THEN M=M+1:GOTO 2260
2120 X=CURCLU+QQ:T=INT(X/TS):S=X-TS*T+1
2130 PRINT@482,CURCLU;:PRINT@500,FZ-(512
*(M-1));
2140 GN=CURCLU:GOSUB 15000:CURCLU=CV
2200 GOSUB 5000
2210 IF CURCLU=0 THEN 190
2220 IF CURCLU<HCLU THEN 2260
2250 IF CURCLU=&HFFF THEN M=M+1
2260 NEXT M
3000 PRINT#1,CHR$(&H0D);
3005 PRINT#1,CHR$(&H1A);
3010 CLOSE #1
3015 PRINT@500," ";:PRINT@450,LB;
3020 SOUND 100,10
3030 PRINT@224," VEGE
"
3040 PRINT:PRINT"HA FOLYTATJA, NYOMJA ME
G AZ >ENTER< GOMGOT, HA NEM, VALAME
LYIK MASIKAT."
3045 A$=INKEY$
3050 IF A$="" THEN 3045
3060 IF A$=CHR$(13) THEN 100
3070 CLS:END

```

```

5000 REM SECTOR A BUFFERBE
5010 PRINT@489,T;:PRINT@495,S;
5020 POKE &HEA,2:POKE &HEB,1:POKE &HEC,T
:POKE &HED,S:POKE &HEE,IS:POKE &HEF,0
5030 EXEC DKON
5040 IF PEEK(&HF0) THEN 9000
5050 EXEC &H7E00
5060 A$=""
5065 P=VARPTR(A$)
5070 POKE P,128
5075 FOR Y=0 TO 3
5080 Z=ID+Y*128
5085 GOSUB 5200
5090 POKE P+2,MSB:POKE P+3,LSB
5100 PRINT#1,A$;
5110 NEXT Y
5130 RETURN

```

```

5200 MSB=INT(Z/256)
5210 LSB=Z-MSB*256
5220 RETURN
9000 REM I/O HIBAJEL
9020 CLS:PRINT@224," LEMEZHIBA
"

```

```

9030 PRINT:PRINT"AZ >ENTER< GOMBBAL A PR
OGRAMOT UJRA INDITHATJA"
9040 IF INKEY$<>CHR$(13) THEN 9040
9050 GOTO 100

```

```

10000 REM
10015 ZZZ$=CHR$(&HE5)+STRING$(7,&HF6))
10020 DLOC=DD+32*NT
10030 IF DLOC>&H71FF THEN F1=4:RETURN
10040 FB=PEEK(DLOC)
10060 IF FB=0 THEN F1=3:RETURN
10070 NAM$=""
10080 FOR N=0 TO 7
10090 NAM$=NAM$+CHR$(PEEK(DLOC+N))
10100 NEXT N
10104 IF NAM$=ZZZ$ THEN F1=3:RETURN
10105 IF FB=&HE5 THEN F1=2:RETURN
10110 NAM$=NAM$+ "."
10120 FOR N=8 TO 10
10130 NAM$=NAM$+CHR$(PEEK(DLOC+N))
10140 NEXT N
10145 F1=0
10150 A=PEEK(DLOC+11)
10155 T=A AND &H10:IF T THEN F1=1

```

```

10160 T=A AND &H0B:IF T THEN F1=7
10200 RETURN

```

```

11000 REM
11010 DLOC=DD+NT*32
11020 FZ=PEEK(DLOC+28)+PEEK(DLOC+29)*256
+PEEK(DLOC+30)*65536+PEEK(DLOC+27)*256
11030 BC=PEEK(DLOC+26)+PEEK(DLOC+27)*256
11040 RETURN

```

```

13000 REM
13010 Q=0:FE=255
13013 SCST=128
13030 NT=K
13040 GOSUB 10000:GOSUB 11000
13050 IF F1 THEN 13200
13100 NTRYLC(Q)=DLOC
13107 PRINT@SCST+32*Q,Q+1
13110 PRINT@SCST+3+Q*32,NAM$
13115 PRINT@SCST+19+Q*32,FZ
13120 Q=Q+1
13200 K=K+1
13210 IF F1=3 THEN FE=0:RETURN
13220 IF DB+32*K>DEND THEN FE=0:RETURN
13230 IF Q>7 THEN RETURN
13240 GOTO 13030
15000 REM
15010 GIN=INT(GN/2)
15020 GCN=3*GIN
15030 GF=GN-2*GIN
15040 B1=PEEK(FS*256+GCN)
15050 B2=PEEK(FS*256+GCN+1)
15055 B3=PEEK(FS*256+GCN+2)
15060 N1=(B1 AND &HF0)/16
15070 N3=(B2 AND &HF0)/16
15080 N5=(B3 AND &HF0)/16
15090 N2=B1 AND &HOF
15100 N4=B2 AND &HOF
15110 N6=B3 AND &HOF
15120 IF GF=0 THEN 15200
15150 CV=N3+N6*16+N5*256:RETURN

```

```

16000 DLOC=NTRYLC(VV):GOTO 10030

```

```

17000 DLOC=NTRYLC(VV):GOTO 11020

```

ranccsal tethetjük meg. A képernyőn megjelenik az üzenet: rakjunk egy üres lemezt a „B” meghajtóba. A rutin ezután a behelyezett lemezt egyoldalásra formattálja. Valamennyi, a CoCo-ra átírandó fájlt másoljuk át erre a lemezre, mert csak az így szervezett lemezt tudja majd a programunk olvasni. Természetesen átírható a program úgy is, hogy kétoldalas lemezt olvasson, ha van kétoldalas lemezmeghajtónk.

Az MS-DOS rendszerben használatos címkéket, subdirectorykat programunk nem kezeli. Emiatt a másolni kívánt fájlok-nak a „Root Directory”-ban kell lenniük. Ajánlatos továbbá, hogy a másolandó lemezen letörölt fájlt, subdirectory, címke egyáltalán ne legyen.

A fájlok hossza tetszőleges lehet. Az egyszerűség miatt a konvertálás szektoronként halad, és ez az utolsó szektort is teljes egészében érinti — függetlenül attól, hogy teljesen vagy csak részben van-e kitöltve. Az átírt fájlok végére tehát „szemét” kerül vagy kerülhet, és a „szemét” gyakran azt a látszatot kelti, mintha valódi része lenne a fájlnak, ugyanis legtöbbször a fájl végén levő szövegrészről áll össze. Ha egy kicsit előrébb is megnézzük az anyagot, megtaláljuk a fájl valódi végét. Mivel a szektorok hossza 256 bájt, a „szemét” hossza 0—255 bájt között változhat, és tartalma a szektor elején levő hasznos szövegrész annyiszori ismétlése, ahányszor még ez a hasznos szövegrész belefér a szektor fennmaradó részébe.

A program használata

Először is a másolandó anyagokat ASCII formátumban vegyük fel egy egyoldalú MS-DOS rendszerű lemezre, a Root Directoryba. Vigyázzunk, hogy a lemezen valóban ne legyen letörölt fájl, subdirectory vagy címke. Helyezzük az MS-DOS lemezt a Drive 1-be, a fordítóprogramot (MS2COCO.BAS) pedig a Drive 0-ba, majd töltsük be a fordítóprogramot és futtassuk. A címlap megjelenése után nyomjuk meg az ENTER gombot: a Root Directory tartalma megjelenik a képernyőn. Válasszuk ki a fordítandó fájlt, majd ismét nyomjuk meg az ENTER gombot.

A Drive 0-ban levő lemez formattálása RS-DOS szerinti legyen. A másolni kívánt fájl kiválasztása után meg kell adni azt a nevet, ahogy a fájl az RS lemezen fog szerepelni. A név max. 8 karakterből állhat; a program automatikusan hozzáírja a megadott névhez a „/TXT” bővítményt.

A másolás sebessége kb. 2400 baud, ami szövegfájl másolásáról lévén szó, gyorsnak mondható. Ez a sebesség egy, a VARPTR (Variable pointer) utasítást kihasználó programozási trükk eredménye. Az eljárást az 5000—5200-as sorok tartalmazzák. Áttanulmányozva ezt a programrészletet láthatjuk, hogyan lehet a VARPTR-rel egy csapásra előállítani egy 256 bájt hosszú sztringet.

A fordítás közben számok jelennek meg a képernyőn alján. Ezek az MS-DOS éppen aktuális, olvasás alatt levő rekeszeit azonosítják, a képernyő jobb alsó sarkában megjelenő szám pedig a még le nem fordított bájtok mennyiségét mutatja. Ez utóbbi nagyságából következtetni lehet a másolás még hátra levő időtartamára. A fordítás befejeztét a program hangosan jelzi, és lehetőséget ad az eljárás megismétlésére.

VÁGH ISTVÁN

készült fájlnak a CoCo szövegszerkesztőivel való kompatibilitását szolgálja.

Ezt a szűrést elvégző rutint az 5050-es sor hívja meg: EXEC&H7E00. (Az EXEC paranccsal gépi kódú programot lehet futtatni.) Maga a rutin a 40—47-es sorokban található, jelen esetben a CoCo processzorára (6809E) írva, az alábbiak szerint:

7E00	8E6000	LDX	\$6000
7E03	A684	LDA	0,X
7E05	810A	CMPA	\$0A
7E07	2603	BNE	\$7E0C
7E09	4F	CLRA	
7E0A	2002	BRA	\$7E0C
7E0C	847F	ANDA	\$7F
7E0E	A780	STA	,X+
7E10	8C6200	CMPX	\$6200
7E13	25EE	BCS	\$7E03
7E15	39	RTS	

Ha nem akarjuk az átírandó fájlnak ezeket a módosításokat elvégezni, töröljük az 5050-es sort.

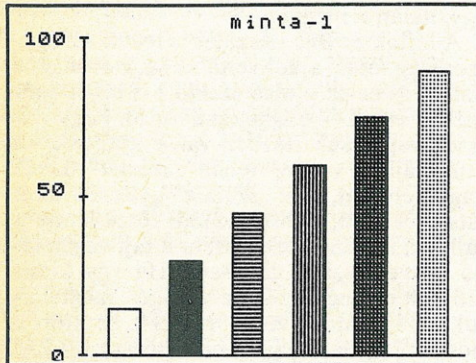
A program gyengéi

A programnak van néhány korlátja. Az első, hogy csak egyoldalú lemezt tud olvasni (kevés CoCo-tulajdonosnak van ugyanis kétoldalas lemezmeghajtója). Ugyanakkor valamennyi MS-DOS felhasználó kétoldalas lemezeket futtat. Az MS-DOS azonban eredetileg egyoldalú lemezek kezelésére készült, és a kompatibilitás fenntartása végett az MS-DOS mai változata az egyoldalú MS-DOS lemezek kezelésére is alkalmas. Ha tehát a Color Computer egyoldalú lemezmeghajtóján is olvasható MS-DOS lemezt kívánunk készíteni, mindenképp egy egyoldalú lemezt kell formálni az MS-DOS gépen. Ezt a „B” meghajtón a FORMAT B:/1 pa-

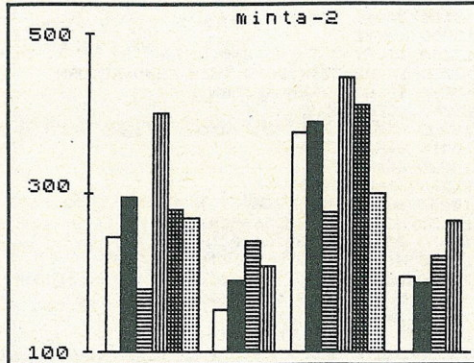
Oszlopdiagram-rajzoló

A program egyenletesen elosztott (1. ábra) vagy csoportokba sorolt oszlopokból álló (2. ábra) oszlopdiagramokat tervez az oszlopok, illetve a csoportok és oszlopok száma alapján, a ZX-Spectrum grafikus lehetőségeinek kihasználásával. Bemetenként kéri az ábra elnevezését, a diagram tető- és alappontját, majd az ábrázolandó értékeket. Minden oszlop ötféle mintázattal színezhető ki.

AUTH FERENC



1. ábra



2. ábra

```

10 DIM c(50): DIM v(100,2): LET m$=""
   "": LET n$=""
   " "
20 CLS: FOR x=26 TO 146 STEP 120: FOR
y=172 TO 84 STEP -88: PLOT x,y: DRAW 0,
-66: DRAW 100,0: NEXT y: NEXT x
30 FOR a=0 TO 120 STEP 120: FOR x=32 T
O 112 STEP 16: IF x<80 THEN LET y=x-8:
GO TO 50
40 LET y=136-x
50 PLOT x+a,106: DRAW 0,y: DRAW 8,0: D
RAW 0,-y: IF a=120 THEN PLOT x+a+4,106+
y: DRAW 0,5: DRAW 1,0: DRAW -2,0
60 NEXT x
70 FOR b=46 TO 82 STEP 36: FOR c=0 TO
2: LET x=b+8*c: IF x<80 THEN LET y=x-8:
GO TO 90
80 LET y=136-x
90 PLOT x+a,18: DRAW 0,y: DRAW 8,0: DR
AW 0,-y: IF a=120 THEN PLOT x+a+4,18+y:
DRAW 0,5: DRAW 1,0: DRAW -2,0
100 NEXT c: NEXT b: NEXT a
110 FLASH 1: PRINT AT 9,9;"a":AT 9,24;"
b":AT 20,9;"c":AT 20,24;"d": FLASH 0: PR
INT AT 10,12;"Valasszon!"
120 LET i$=INKEY$: IF i$="a" OR i$="b"
OR i$="c" OR i$="d" THEN GO TO 140
130 GO TO 120
140 CLS: PLOT 47,164: DRAW 4,0: PLOT 4
9,164: DRAW 0,-152: PLOT 47,88: DRAW 4,0
: PLOT 47,12: DRAW 4,0: PLOT 55,12: DRAW
200,0
150 PRINT AT 1,0;"teto":AT 20,0;"alap"
160 FLASH 1: PRINT AT 0,7;" A diagram
am neve "": FLASH 0: INPUT "A diagram
neve: ";b$: PRINT AT 0,7;m$: PRINT AT 0
,19-(LEN b$)/2;b$
170 FLASH 1: PRINT AT 1,0;"teto": FLASH
0: INPUT "tetopont: ";te: LET tet=LEN S
TR$ te: PRINT AT 1,0;" ";AT 1,5-tet;
te
180 FLASH 1: PRINT AT 20,0;"alap": FLAS
H 0: INPUT "alappont: ";al: LET ala=LEN
STR$ al: PRINT AT 20,0;" ";AT 20,5-a
la;al
190 LET fe1=LEN STR$(al+(te-al)/2): PR
INT AT 10,5-fe1;al+(te-al)/2
200 IF i$="c" OR i$="d" THEN GO TO 220
210 PRINT AT 21,0;n$: INPUT "Az oszlopo
k száma: ";n: LET iv=200/n: FOR a=1 TO n
: LET v(a,1)=55+INT (a*iv-3*iv/4): LET v
(a,2)=55+INT (a*iv-iv/4): NEXT a: GO TO
320
220 INPUT "A csoportok száma: ";cs: IF
cs=1 THEN GO TO 310
230 PRINT AT 21,0;n$: "Az elemek száma ugya
naz (i/n)?"
240 LET a$=INKEY$: IF a$="1" THEN PRIN
T AT 21,0;n$: GO TO 310
250 IF a$<>"a" THEN GO TO 240
260 PRINT AT 21,0;n$: LET n=0: FOR a=1
TO cs: INPUT ("Az elemek száma, ";a;".cs
oport: ");c(a): LET n=n+c(a): NEXT a
270 LET iv=400/(n+cs): LET i=0: LET j=0
: LET ind=1: FOR a=1 TO cs: FOR b=1 TO c
(a): LET i=i+1: LET j=j+1: LET v(j,2)=55

```

```

+INT (i*iv/2+iv/4): IF j=1 THEN LET v(1
,1)=55+INT (i*iv/2-iv/4): GO TO 300
280 IF ind=0 THEN LET ind=1: LET v(j,1
)=55+INT (i*iv/2-iv/4): GO TO 300
290 LET v(j,1)=v(j-1,2)
300 NEXT b: LET i=i+1: LET ind=0: NEXT
a: GO TO 320
310 LET n=0: INPUT "Az elemek száma: ";
a1: PRINT AT 21,0;n$: FOR a=1 TO cs: LET
c(a)=a1: LET n=n+c(a): NEXT a: GO TO 27
0
320 FOR x=34 TO 234 STEP 40: PLOT x,6:
DRAW 16,0: DRAW 0,-6: DRAW -16,0: DRAW 0
,6: NEXT x
330 FOR b=6 TO 0 STEP -1: PLOT 74,b: DR
AW 16,0: NEXT b
340 FOR b=6 TO 0 STEP -2: PLOT 114,b: D
RAW 16,0: NEXT b
350 FOR b=154 TO 170 STEP 2: PLOT b,6:
DRAW 0,-6: NEXT b
360 FOR b=194 TO 210 STEP 2: PLOT b,6:
DRAW 0,-6: NEXT b: FOR b=6 TO 0 STEP -2:
PLOT 194,b: DRAW 16,0: NEXT b
370 FOR b=234 TO 248 STEP 2: PLOT b,2:
PLOT b,4: NEXT b
380 FOR a=0 TO n-1: INPUT "ertek: ";ert
390 IF i$="a" OR i$="c" THEN LET hi=0:
GO TO 410
400 INPUT "hiba: ";hi
410 LET fp=INT (12+152*(ert-al)/(te-al)
): LET hip=INT (152*hi/(te-al))
420 PLOT v(a+1,1),12: DRAW 0,fp-12: DRA
W v(a+1,2)-v(a+1,1),0: DRAW 0,12-fp: PLO
T (v(a+1,1)+v(a+1,2))/2,fp: DRAW 0,hip:
DRAW 1,0: DRAW -2,0: IF ert=al THEN GO
TO 510
430 FOR b=3 TO 28 STEP 5: PRINT AT 21,b
p;"": NEXT b: FLASH 1: FOR b=2 TO 27 STE
P 5: PRINT AT 21,b;(b+3)/5: NEXT b: FLAS
H 0
440 LET a$=INKEY$: IF a$="1" THEN GO S
UB 520: GO TO 510
450 IF a$="2" THEN GO SUB 520: FOR b=v
(a+1,1)+1 TO v(a+1,2)-1: PLOT b,13: DRA
W 0,fp-14: NEXT b: GO TO 510
460 IF a$="3" THEN GO SUB 520: FOR b=f
p TO 12 STEP -2: PLOT v(a+1,1)+1,b: DRA
W v(a+1,2)-v(a+1,1)-2,0: NEXT b: GO TO 51
0
470 IF a$="4" THEN GO SUB 520: FOR b=v
(a+1,1)+2 TO v(a+1,2)-1 STEP 2: PLOT b,f
p: DRAW 0,12-fp: NEXT b: GO TO 510
480 IF a$="5" THEN GO SUB 520: FOR b=v
(a+1,1)+2 TO v(a+1,2)-1 STEP 2: PLOT b,f
p: DRAW 0,12-fp: NEXT b: FOR b=fp TO 12
STEP -2: PLOT v(a+1,1)+1,b: DRAW v(a+1,2
)-v(a+1,1)-2,0: NEXT b: GO TO 510
490 IF a$="6" THEN GO SUB 520: FOR b=v
(a+1,1)+2 TO v(a+1,2)-1 STEP 2: FOR c=fp
TO 12 STEP -2: PLOT b,c: NEXT c: NEXT b
: GO TO 510
500 GO TO 440
510 NEXT a: PRINT AT 21,0;n$: STOP
520 FOR b=2 TO 27 STEP 5: PRINT AT 21,b
;"": NEXT b: RETURN

```

Menük

Mi az a menü? Programrészlet, amelynek segítségével a felhasználó különböző funkciókból választhat — valamilyen beavatkozás (például gombnyomás) által. A program szempontjából a menü nem más, mint egy bemenő adattól függő elágazás. Egyedi megoldásban általában kissé terjengős (1. lista).

Mivel menüket lépten-nyomom használunk, nagyobb programokban akár tucatnyit is, célszerű, ha csinálunk egy általános Menü rutint, amely úgy oldja meg a feladatot, hogy csak az okvetlenül szükséges információt kelljen vele közölnünk. Az előbbi példában csak a kiírandó szövegek és a GOTO sorszámok voltak „informatívak”, a többit megspórolhatjuk. Természetesen a rutin egyszerű megírása nagyobb munka, de később ez megtérül, nem beszélve arról, ha egy programon belül többször is használjuk.

A feladatot több nyelven is megoldjuk, bár inkább csak ötleteket adunk; ki-ki kialakíthatja a neki legmegfelelőbb menükészletet.

BASIC

A rutin a 20—240-es sorokban van, utána következik egy példa az alkalmazására: a 400—410-es sor. A szabály az, hogy a GO SUB utáni első DATA sorban kell megadni a választékot, a hozzá tartozó ugrási címekkel. A választékot megnyomandó billentyű mindig a szó kezdőbetűje: kis- és nagybetű egyaránt lehet.

Működése azon alapul, hogy ennek a DATA sornak a címét megkeressük, pontosabban elég a GO SUB sor sorszámát ismerni, ez viszont a gosub veremben van. Az ERR_SP mutatja azt a rekeszt, ahol a BASIC hibacímre van elhelyezve, kettővel följebb található a legutolsó GO SUB vizsgáztatási címe — ez kell nekünk. A 20-as sor megoldja a DATA-ra való ráállást, leolvashatjuk a tételeket. A beolvasás addig tart, amíg a DATA_ADD további tételt jelez (70-es sor; vegyük figyelembe, hogy a vessző kódja 44).

Kiírtuk a választékot, most billentyűt várunk a 120-as sorban (de még előbb a 110-es sor véd a funkcióismétlés ellen), amit kisbetűre állítunk azután.

A kapott kódot sorban összehasonlítjuk minden tétel első karakterével (a 100-as sor ismét a DATA-ra állt), és ha nincs egyezés, akkor a 100-as sorhoz visszamegyünk új billentyűért.

ZX-SPECTRUM

Egyezés esetén a 200–220-as sorban a kívánt értéket tesszük a gosub verembe az eredeti visszatérési cím helyett, az utasításszámot pedig 1-re állítjuk. Így a RETURN a kívánt sorra való ugrást eredményezi.

A 240-es sorban az INPUT"" törli a képernyő alját (2. lista).

BETA BASIC 1.8 és 1.0

Nagyjából ugyanezen az elven működik a következő, BETA BASIC 1.8-ra írt rutin, amely elegánsan rövid: 2 sor. Használati szabálya éppúgy az, hogy a hívás utáni első DATA-ban kell megadni a funkcióbillentyűket és a hozzájuk tartozó ugrási címeket. Fontos, hogy az elválasztójelként használatos "OR" a végén is legyen! Érdekes a várokozás megoldása: ha nem jó billentyűt nyomtunk meg, akkor found=0, SGN found+1=1, ezért a 2. sorra ugrunk, azaz új billentyűre várunk. Jó billentyű esetén az előbbi érték 2, ezért a hosszú képlettel megadott címre ugrunk, ami viszont épp a ">" és az "OR" közötti részt értékelte ki (3. lista).

Felmerül a kérdés, miért nem elég, ha előírjuk, hogy a menüből 1, 2, 3, ... stb. számokkal kell választani, és akkor egy GO TO ON-nal elintézhető az egész? Egyszerű, csak kicsit igénytelen megoldás, például nehezebb megtanulni a program használatát, mert semmi logikus kapcsolat nincs a funkció és az öt kiválasztó karakter között. Ezért nem is terjedt el.

BETA BASIC 3.1

Ebben a verzióban már módunk van igazi eljárás írására. A DEF PROC után megadott változólista formális paramétereket tartalmaz: ezeknek a változóneveknek semmi közük a főprogramban használt változókhoz. A híváskor adjuk meg, hogy oda milyen változónevet vagy -értéket kell helyettesíteni. REF kulcsszó után a változónevé szerint hívott lesz, egyébként érték szerint. Példánkban az option név szerint hívott, ez tehát kimenő paraméter is lehet. A LOCAL után megadott változóknak semmi közük a főprogramhoz (blokkon belüli deklaráció).

Eljárásunk kiírja a választékot, majd megadja, hogy a lenyomott gomb hányadik funkciónak felel meg (option). Ha egyiknek sem, akkor option=0. Utána GO TO ON option-nel ugrunk. Ez azért jobb megoldás, mert átsorszámozás esetén a GO TO ON utáni sorszámlista is átszámozódik (az

```
50 PRINT "PRINT TEXT FILE      P"
60 PRINT "SAVE TEXT FILE      S"
70 PRINT "LOAD TEXT FILE      J"
... stb.
130 INPUT X$
140 IF X$>="a" AND X$<="z" THEN LET X%=CHR$(CODE X%-32)
150 IF X%="P" THEN GO TO 200
160 IF X%="S" THEN GO TO 1000
.... stb.
230 GOTO 130
```

1. lista

```
5 DEF FN p(x)=PEEK x+256*PEEK (x+1)
10 REM menu rutin
20 LET esp=FN P(23613)
   LET addr=FN P(esp+2)
   RESTORE addr
50 READ X$,naddr
60 PRINT #1;X$;" ";
70 IF PEEK FN P(23639)=44 THEN GO TO 50
100 RESTORE addr
110 IF INKEY$>"" THEN GO TO 110
120 IF INKEY$="" THEN GO TO 120
130 LET kod=CODE INKEY$
140 IF kod>64 AND kod<91 THEN LET kod=kod+32
150 READ X$,naddr
160 IF CODE X$(1)=kod THEN GO TO 200
170 IF PEEK FN P(23639)=44 THEN GO TO 150
180 BEEP 0.4,0
   GO TO 100
200 POKE esp+2,naddr-256*INT (naddr/256)
210 POKE esp+3,INT (naddr/256)
220 POKE esp+4,1
230 BEEP 0.2,30
240 INPUT ""
   RETURN
400 GO SUB 10
410 DATA "printer",1000,"load",800,"save",700
700 PRINT "save"
   STOP
800 PRINT "load"
   STOP
1000 PRINT "print"
   STOP
```

2. lista

3. lista

```
1 DEF PROC menu
   POP caller
   RESTORE caller
   READ m$
   POKE 23658,0
2   GET g$
   LET g%=g$+">"
   LET found=INSTRING(1,m$,g%)
   GO TO ON 1+SGN found;2,VAL m$(found+2 TO INSTRING(found+2,m$," OR ")-1)
   END PROC
80
88 REM demo start
90 PRINT "input/list/microdrive/range/stop"
100 PROC menu
   DATA "i>270 OR l>550 OR m>8000 OR r>1200 OR s>2000 OR "
270 PRINT "input"
   GO TO 90
550 PRINT "list"
   GO TO 90
1200 PRINT "range"
   GO TO 90
2000 STOP
8000 PRINT "microd."
   GO TO 90
```



```

10 DEF PROC menu2 pr, REF option
20
   LOCAL cim, a$, g$
30 LET cim=DPEEK(DPEEK(23513)+2)+7429
   DPOKE 23539, cim-1
40 DO
   READ LINE a$
   PRINT #pr; a$
   LOOP UNTIL ITEM()=0

50 DPOKE 23539, cim-1
60 GET g$
   LET g$=SHIFT$(2, g$)
   LET option=0

70
   READ LINE a$
   LET option=option+1
   IF a$(1)=g$ THEN GO TO 90
80 IF ITEM()=0 THEN GO TO 50
   ELSE GO TO 70

90
   IF INKEY$("<") THEN GO TO 90
100 END PROC

110 menu2 0, opt
   DATA q question, m microdrive, i inp AND lista
   GO TO ON opt, 520, 300, 830

```

4. lista

5. lista

```

100 PROGRAM MENUDEMO1;
110
120 VAR MENU:BOOLEAN;
130
140 FUNCTION GETW:CHAR;
150 VAR I:INTEGER;
160 GOMB:CHAR;
170 BEGIN
180 REPEAT GOMB:=INCH UNTIL GOMB>CHR(0);
190 IF (GOMB='a') AND (GOMB<='z') THEN GOMB:=CHR(ORD(GOMB)-32);
200 FOR I:=1 TO 50 DO IF INCH>CHR(0) THEN I:=1;
210 GETW:=GOMB
220 END;
230
240 BEGIN
250 REPEAT {DEMO CIKLUS}
260 WRITELN; WRITELN('MENU-DEMO:');
270
280 REPEAT
290 MENU:=TRUE;
300 WRITELN('INPUT LIST DRIVE TESTING');
310
320 CASE GETW OF
330 'I':WRITELN('INPUT RUTIN');
340
350 'L':WRITELN('LISTAZGATUNK');
360
370
380 'D':WRITELN('DRIVE MUVELETEK');
390
400
410
420 'T':WRITELN('KIKERDEZLEK') {nincs pontosvesszo!}
430
440 ELSE MENU:=FALSE
450
460 UNTIL MENU
470 {KILEPES CAPS+BREAK, MAJD CAPS+1}
480 UNTIL FALSE
490 END.

```

előző két megoldásnál a DATA sorokat nekünk kell átírni újraszámozáskor).

A megoldás kb. ugyanaz, mint a sima BASIC-ben, de itt a gosub veremben nem a visszatérési sor száma, hanem a címe -7429 van (ez érvényes a 1.8 verzióra is). Ezért RESTORE helyett közvetlenül írunk a DATA_ADD rendszerváltozóba (30-as sor). A többi lépés tulajdonképpen ugyanaz, csak kihasználja a BETA 3 kényelmesebb utasításait és függvényeit.

Megadhatjuk még azt is, hogy a kiírást hová kérjük (lásd 40-es sor, 4. lista).

Pascal

Itt a legkönnyebb a dolgunk, mert a CASE karakter utasítással szépen megoldható az elágaztatás. Viszont problémák vannak, ha a választ INCH-rel várjuk (lehet READ is, de az megint nem elegáns a feleslegesen megnyomandó ENTER miatt).

Például WHILE INCH=CHR(0) DO;

GOMB:=INCH;

eredménye GOMB=CHR(0), amit nem váránk. Hasonlóképpen a következő ciklus, melynek illene addig várakoznia, amíg a billentyűt fel nem engedjük:

REPEAT UNTIL INCH=CHR(0) DO;

ez bizony nem vár egy pillanatig sem. A probléma magyarázatát meglátjuk, ha ezt futtatjuk:

REPEAT WRITE (INCH) UNTIL FALSE;

eredmény például k-t lenyomva ?K???K???K???K???K???K stb. A "?"-ek esetén az INCH 0 kódot vett, aminek az oka feltehetően az, hogy a PASCAL akkor másolja át a LAST K-t az INCH-be, ha a lenyomást a funkcióismétlési ROM (#310) rutin elfogadta.

Tehát az INKEY\$ és az INCH között alapvető különbség van! A problémát a 180-as sor oldja meg: ez megfelel egy GETKEY\$-nek. A 200-as sor pedig kissé illegális módon a billentyű felengedésére vár (az 50-es érték ki van kísérletezve, kisebbet ne adjunk).

Magyarázatként talán még annyit, hogy a MENU változó azért kell, hogy rossz válasz esetén a menü ismételjen. Ilyenkor ugyanis a MENU=FALSE lesz, míg egyébként TRUE. Azzal, hogy az egyes funkciókban MENU értékét megváltoztatjuk-e van nem, szabályozni tudjuk, hogy a funkció végrehajtása után a menühöz vagy az azt követő utasításhoz térjünk-e vissza (5. lista).

Gépi kód

A menü rutin gépi kódú programban a leghatásosabb. Bemutatunk egy olyan megoldást (MENUR rutin), amely

— elmenti a képernyő középső harmadát, majd törli;

— itt azután kiír egy fejléct, utána a választékot (max. 8 sor);

— billentyűt vár, ha jót nyomtunk meg, akkor ugrik a megfelelő funkcióra; ha rossz, akkor vagy tovább vár, vagy átugorja a menüt (beállítható);

— mindenképpen kilépés előtt visszaállítja az eredeti képernyőt.

Híváskor semmi mást nem kell megadni, mint a kiírandó szöveget és az ugrási címet. Hívási példák a 63-90-es sorokban. „Szintaktikája” a következő. A CALL MENUR után megadandó

— a fejléc (lehet üres is, végét ";" jelzi),
— a menüválaszték (egymástól ";"-vel elválasztott tételekből áll, végét ";" jelzi),
— az ugrási címek.

A vessző amellett, hogy elválasztó jel, soremelést is jelent. Természetesen a szövegben belül egyébként nem használhatjuk a vesszőt és a pontosvesszőt. Szükség esetén az elválasztó jelek megváltoztathatók, módosítani kell minden CP 44, illetve CP 59 utasítást a kívánt kódokra, és kész.

Vigyázzunk arra, hogy a tételek száma (t) és az ugrási címek száma (u) között szigorú kapcsolat van; ha ezt nem tartjuk be, úgy elszállunk, mint a huzat. Vagy

A) $t = u$, ha az utolsó tétel nem üres,

B) $t = u + 1$, ha az utolsó tétel üres.

Ennek megfelelően még két üzemmódból is választhatunk: az A) esetben, ha nem jó gombot nyomunk meg, akkor vár a rutin újabb gombnyomásra, a B) esetben rossz gombnyomásnál azonnal kilép a rutinból és a hívás utáni utasítással folytatja a futást.

A rutin működése

A 300-as sortól kezdődik; először elmentjük a képernyő például középső harmadát az 51000-es címre, majd törölünk. Igényesebbek az attributumokkal is megtehetik ugyanezt, 22784-től 256 bajt. A print pozíciót az „ablak sarkába” tesszük.

Az 560-as sorban kiemeljük a veremből a visszatérési címet. Ez éppen megegyezik a fejléc első karakterének címével. Kétféle olvasó ciklus jön: először a fejléct írjuk ki (ott van vége, ahol ";"-t találunk), majd a 740-es sorban elraktározzuk a választék kezdőcímét. A második menetben a választékot írjuk ki, ennek a végét újabb ";" jelzi, utána rögtön jönnek az ugrási címek, tehát a 740-es sorban most az első ugrási cím címét tesszük el.

A 800-as sorban egy INKEY\$ karaktert olvasunk be, de csak akkor, ha ezt a ROM repeating key (#310) megengedi, amit a 23611 című FLAGS rendszerváltozó 5-ös bitjének NZ állapota (1) jelez. Ez azért lényeges, mert ha több menü van egymás után, és az egyikből mondjuk K billentyűvel választottunk, lehet, hogy a funkció végrehajtása olyan gyors (gépi kódban vagyunk!), hogy fel sem tudtuk még engedni

```

30 ;gépi kódú menü demo 8, 86.12.13.
40
50     ORG 50000
60
61     LD  HL,$           ;mindig ide fog visszatérni
62     PUSH HL           ;az egyes "funkciók" végrehajtása
                           után
63     CALL MENUR        ;a menürutin hívása
64     DEFM " , EZ A BEMUTATÓ, FOGTAD AGYILAG?, AKKOR "
65     DEFM "NYOMJ MEG EGY GOMBOT! ;;"
66
70     CALL MENUR        ;újabb menühívás
79     DEFM " , , FUNKCIÓK: , ;"
80     DEFM "s-szóbeírás,a-adattárolás,k-kikérdezés,e-exit
                           ;"
90     DEFW INPUT,DRIVE,QUEST,STOP
100
110 STOP CALL SCREEN    ;visszajön az eredeti kép, azután
120     RST 8
130     DEFB 8           ;8-es hibaüzenettel leáll.
140
150 INPUT LD  A,"s"
160     RST 16
170     RET
180
190 DRIVE LD  A,"a"
200     RST 16
210     RET
220
230
240 QUEST LD  A,"k"
250     RST 16
260     RET
270
280
290
300 ; < gépi kódú menürutin ver2.0 86.12.13. (C) Slíz Miklós >
310
320
330 PUFFER EQU 51000    ;ide mentjük a képernyő 1/3-át (20
                           48 byte)
340
350 ;A képernyő középső harmadát elmentjük és nullázzuk.
360 MENUR LD  A,2
370     CALL #1601       ;az 'S' csatorna megnyitása
380     LD  HL,18432     ;a képernyő középső harmad címe
390     LD  DE,PUFFER   ;a raktár címe
400     LD  BC,2048     ;1/3 képernyő
410     LDIR           ;elmentés
420
430     LD  HL,18432     ;középső harmad
440     LD  DE,18433     ;
450     LD  (HL),0       ;letöröljük
460     LD  BC,2047     ;a középső harmadot.
470     LDIR
480
490     LD  A,22
500     RST 16
510     LD  A,8
520     RST 16
530     LD  A,0
540     RST 16         ;print at 8,0
550
560 ;A menüszöveg kiírása
570
580 MENU POP  HL        ;szöveg kezdete
590     LD  B,2         ;két menet: fejléc, aztán a válasz
                           ték
600
610 CHAR LD  A,(HL)     ;aktuális karakter beolvasása
620     CP  44         ;','?
630     JR  Z,NEXT     ;akkor új tétel jön.
640     CP  59         ;pontosvessző esetén szöveg vége
650     JR  Z,KONEC
660 CHAR1 RST 16       ;az aktuális karakter kiírása
670     INC HL         ;új karakter
680     JR  CHAR       ;folytatjuk.
690
700 NEXT LD  A,13      ;új tétel - új sor
710     JR  CHAR1

```



```

720
730 KONEC INC HL
735 ;1. menetben HL a választék elejére, a 2. menetben
736 ;az első ugrási címre mutat.
740 PUSH HL
745 DJNZ CHAR ;2 menet
750
760 ;Egy karakter a billentyűzetről
770
800 INKEYS BIT 5,(IY+1) ;FLAGS
810 JR Z,INKEYS ;elfogadott billentyűre vár
845 LD A,(#5C00B) ;LAST-K
850 ;a lenyomott billentyű kódja A-ba került
855 RES 5,(IY+1) ;törli az új gomb jelet
860
870 CP 65 ;key<'A' esetén
880 JR C,MARAD
890 CP 91 ;és key>'Z'-nél
900 JR NC,MARAD ;marad, azonban
910 ADD A,32 ;egyébként kisbetűre állít
920 MARAD LD C,A ;végül a kódot a C regiszterbe tes-
;szük

930
940 ;A beolvasott karakter ellenőrzése
950
960 POP DE ;az 1.rutin címére mutat
970 POP HL ;a szöveg 1.tételre mutat
980 PUSH HL
990 PUSH DE
1000 JR NEXT2 ;belépés az 1.tétel 1.karakter ell-
;enőrzésére

1010
1020 KERES LD A,(HL) ;a szöveg aktuális karaktere
1030 CP 44 ;','?'
1040 JR Z,NEXT1 ;ugrás új tételhez
1050 CP 59 ;pontosvessző esetén vége,
1060 JR Z,INKEYS ;új billentyűt kérünk!
1070
1080 INC HL ;következő karakter
1090 JR KERES

1100
1110 NEXT1 INC HL ;a ',' átugrása
1120 INC DE
1130 INC DE ;a következő rutin címére mutat
1140 NEXT2 LD A,(HL) ;a tétel 1.karaktere
1141 CP 59 ;hiányzik az utolsó tétel?
1142 JR Z,CONTIN ;akkor folytatjuk a DEFW utáni uta-
;sitással
1150 CP C ;egyezik a beolvasott billentyűvel
?
1160 JR NZ,KERES ;ha nem, tovább vizsgál.
1170
1180 EX DE,HL ;most HL mutat a rutincímre
1190 LD E,(HL)
1200 INC HL
1210 LD D,(HL) ;DE=rutincím
1220 CONTIN POP HL
1230 POP HL ;kidobtuk a felesleget
1240 PUSH DE ;a kiválasztott rutinra közvetett
;ugráshoz

1250
1260 ;Visszahozzuk az eredeti képet
1270
1280 SCREEN LD HL,PUFFER
1290 LD DE,18432 ;képernyő középső harmad címe
1300 LD BC,2048 ;1/3 mérete
1310 LDIR
1320
1330 LD A,22
1340 RST 16
1350 LD A,0
1360 RST 16
1370 LD A,2
1380 RST 16 ;print at 0,2 ;(csak a demo kedvéé
;rt)

1390
1400 RET ;ugrás a kiválasztott rutinra

```

a K billentyűt, máris egy újabb menüben vagyunk, és akaratlanul kiválasztottuk a K funkciót stb. Megoldásunk ezt megakadályozza, funkcióismétlés csak a REPDEL-nél lenyomott gombnál lehet.

Mivel a ROM billentyűzet rutinját használjuk, a megszakítást nem szabad letiltani!

A 870-es sorban a beolvasott karaktert kisbetűre állítjuk.

A 960-as sorban kezdődik az input karakter vizsgálata. Elővesszük a választék címét, és újra végigmegegyünk rajta. Mindig a tételek 1. karakterével hasonlítjuk össze a beolvasottat (1140-es sor). Előbb azonban megnézzük, hogy az 1. karakter nem pontosvessző-e (ez a helyzet, ha az utolsó tétel üres; ugyanakkor a beolvasott karakter is lehet ";"). Itt mindig a HL a szöveg cím-mutató, DE pedig az aktuális rutincímre mutat. Figyeljük meg, hogy a ciklusból háromféleképpen lehet kilépni:

- az 1060-as sornál, ha nem volt egyezés, és az utolsó tétel nem üres, és ekkor tovább várunk jó billentyűre (A üzemmód);

- az 1142-es sornál, ha nem volt egyezés, és az utolsó tétel üres (tehát 1. karakter helyett ";"-t találunk), és ekkor a DE-ben a menühívás utáni utasítás címe van (B üzemmód);

- az 1180-as sornál, ha megtalálta a keresett kódot, és ekkor DE-be kerül a kiválasztott rutin címe.

Az 1220-as sorban találkozunk a két eset: eltávolítjuk a veremből a felesleget, majd PUSH DE. Ez volt az egész rutin értelme! Az 560-as sorban kivettük a veremből a visszatérési címet, most a DE-t tesszük a helyére. Tehát a RET után ide fog ugrani a processzor. Az A és B esetnek megfelelően ez lehet egy funkció rutinra ugrás, illetve egyéb billentyű megnyomása esetén a menü átugrása (folytatás a CALL MENUR, DEFM ..., DEFW ... utáni utasítással).

Az 1280-as sorban még a visszatérés előtt visszatesszük a képernyő közepét, hátha valami fontos volt rajta.

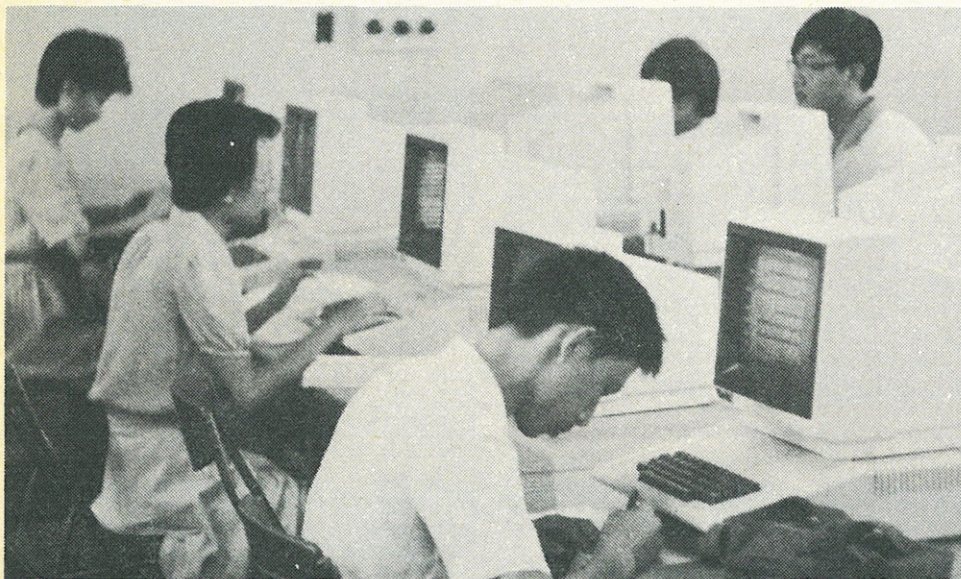
A demo részről (60–280-as sor) két menühívást csinálunk. Az első B típusú: a választék egyetlen üres tételből áll, tehát bármilyen gombnyomásra folytatódik a program; ugrási cím nincs is. A másik A típusú: az egyes billentyűkkel válaszhatunk, és képzeljük azt, mintha lenne is olyan rutin. Most csak egy betű jelenik meg AT 0,2;, hogy lássuk, mi történt.

A demo végtelenített, amit a 61-62-es sornak köszönhet. Ez egyben arra is példa, hogy a menüt nemcsak ugró funkciókra lehet használni, hanem CALL jellegű is lehet, ha előtte a megfelelő visszatérési címet betesszük a verembe. Kilépés csak exit-tel, ez a hibakezelő rutint hívja, STOP statementtel áll meg.

Hogy a BASIC-ben kipróbáljuk, írjunk a képernyőre valami szöveget, majd PRINT AT 0,0;" > < "; PRINT " " RANDOMIZE USR 50000.

Mikrogepes úttörők

Sétálgatva a Pekingi Úttörőpalotában, mindenütt munkában, alkotásban elmerült gyerekeket láttunk: a repülőmodellezők nyüzsgő termétől kezdve a szabadtéri színpadon gyakorló úttörőzenekarig. Hirtelen ötlettől vezérelve megkérdeztük az egyik tanárt, hogy vannak-e itt mikroszámítógépek? A szemközti pagodára mutatott, teljes természetességgel. Hitetlenkedve, bizalmatlanul mentünk az évezredes épülethez, és nyitottuk ki ódon ajtaját. Földbe gyökerezett a lábunk: az utcai trópusi hőség után a hazai gépterem hűvöse fogadott; a helyiség tele volt, már az első pillantásra is láthatóan, IBM PC-vel kompatibilis mikrogepekkel. Az előttünk ülő gyerekek a munkában elmerülve, pillantásra sem méltattak bennünket, európaiakat, pedig az utcán mindenütt kíváncsi tekintetek keresztüztében jártunk.



Az IBM PC-vel kompatibilis gépek a laboratóriumban

Huang Hai-penget, az egyik tanárt rövid időre elszakítva a gyerekektől, faggatni kezdtük.

— *Mióta üzemelnek itt ezek a mikroszámítógépek?*

— 1985 decemberében nyitottuk meg a géptermet. Japán gyártmányú NEC PC 9801 típusú gépekkel szereltük fel, melyek kompatibilisek az IBM PC-vel. Kis, csak hajlékonylemezes tárolót tartalmazó konfigurációkat szereztünk be, mivel oktatási célokra ezek is megfelelnek. Ugyanilyen megfontolásból csak néhány géphez vásároltunk nyomtatót, hiszen hajlékonylemezen át lehet vinni a kinyomtatandó programokat.

— *Tudnak-e kínaiul ezek a gépek?*

— Egy pillanat, tessék megnézni! — És Huang Hai-peng a legközelebbi, géphez lépve, lemezről elindított egy programot.

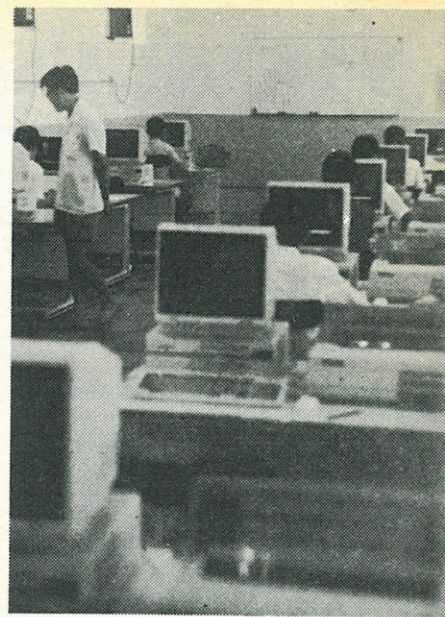
Az addig latin betűket író képernyőn megjelentek a kínai írásjelek.

— *Hogyan juthatnak itt géphez a pekingi gyerekek?*

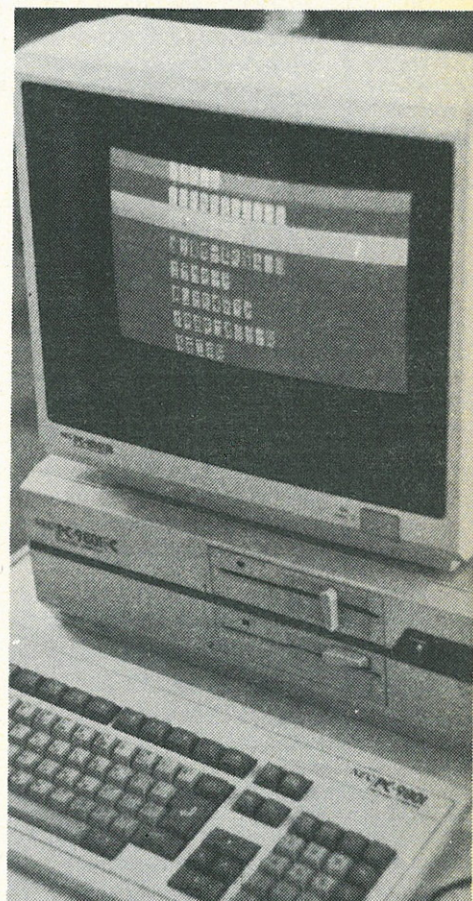
— Újságban hirdetjük ezt a lehetőséget, és a jelentkezők felvételi vizsgát tesznek. Nem a konkrét számítástechnikai ismereteket kérjük számon, hanem inkább a logikai képességeiket és matematikai tudásukat. A gyakorlat azt mutatja, hogy a kiválasztott gyerekek megbecsülik a kapott lehetőségeket, és igen komolyan foglalkoznak a mikrogepekkel.

— *Még egy éve sincs, hogy megnyílt ez a labor, és már gyakorlatról lehet beszélni?*

— *Hogyne! Mi most az új gépek laboratóriumában vagyunk. Néhány épülettel távolabb 1984 szeptemberében helyeztük üzembe a „régit” mikrogepeinket. Azok Apple II-vel kompatibilis hongkongi gyárt-*



Az Apple II-vel kompatibilis gépek terme



A NEC PC 9801 közelről

mányúak, és legalább olyan népszerűek a gyerekek körében, mint ezek a nagyobb teljesítményűek. A két gépteremben egész napos a mikrogepek terhelése, annak ellenére, hogy a pekingi iskolák 80-90 százalékában van már mikroszámítógép. Rendkívüli ütemben nő ugyanis a fiatalok körében a számítógépek népszerűsége. Persze ennek nagyon örülünk, hiszen ezért dolgozunk. Nem is munkának, inkább küldetésnek tekintjük azt, amit csinálunk.

További sikert kívánva búcsúztunk a lelkes tanártól és tanítványaitól.

DR. BROCKÓ PÉTER

A pagoda, amelyben a mikrogepek vannak. Előtte Huang Hai-peng a szerzővel



Eddig elméletileg foglalkoztunk a programlevédés technikáival és a VIC-1541 meghajtó működési elvével. Jelenleg egy egyszerű gyakorlati példát mutatunk, amelynek áttanulmányozásával az Olvasó könnyebben megérti majd az ezután ismertetendő bonyolultabb eljárásokat.

Az FDC (Floppy Disk Controller) hat különböző típusú olvasási hibát tud felismerni. Ezek a következők:

Hibakód	A hiba típusa
21	Nincs szinkronkarakter
20	Az FDC nem találja a blokkfejet
27	Ellenőrzőösszeg-hiba a blokkfejben
29	Diszk ID csere
22	Nincs adatblokk
23	Ellenőrzőösszeg-hiba az adatblokkban

Most a 21-es típusú hibával fogunk részletesen foglalkozni. Az FDC az adott sávon (track) keresi a szinkronjelet, és ha 20 millisekundum időn belül nem találja, akkor hibát jelez.

Ezt az olvasási hibát legegyszerűbben úgy használhatjuk fel a védelem céljára, hogy a levédendő lemez bizonyos sávjára vagy blokkjára olyan információt viszünk, amelynek hatására olvasáskor 21-es típusú hibaüzenetet kapunk. A levédendő programba egy olyan szubrutint kell beépíteni, amely megpróbálja elolvasni ezt az általunk átírt sávot vagy blokkot, és ha a várt hibaüzenetet kapjuk, akkor valóban az eredeti lemez van a meghajtóban.

```

100 REM 21-ES HIBA
105 REM PARAMETER: SAVSZAM (TRACK NUMBER)
110 PRINT "CLR121 HIBA"
120 PRINT "LEJIA LEVEDENDO LEMEZT A MEGHAJTÓBA!"
130 INPUT "LEJITORLENDŐ SAV":T
140 IF T<1 OR T>35 THEN END
150 INPUT "LEJ VALÓBAN? (I/N)":Q#
160 IF Q#<<"I" THEN END
170 OPEN15,0,15
180 PRINT#15,"10"
190 INPUT#15,EN#,EM#,ET#,ES#
200 IF EN#="00" GOTO 250
210 PRINT "LEJ"EN#,"EM#","ET#","ES#"
220 CLOSE15
230 END
240 REM A PROGRAM ÁTÍRASA A MEGHAJTÓ
245 REM MEMÓRIÁJÁBA
250 JOB=176
260 GOSUB 400
270 FOR I=0 TO 23
280 READ D
290 D#=#D#+CHR$(D)
300 NEXT I
310 PRINT#15,"M-W"CHR$(0)CHR$(4)CHR$(24)D#
320 REM VEGRÉHAJTÁS
330 PRINT "LEJITORSZÁM:SAVSZAM":T
340 JOB=224
350 GOSUB 400
360 PRINT "LEJKEZSI"
    
```

Most pedig tanulmányozzuk át a „Dos Inside”-ban megjelent programot, amellyel

21-es típusú hibát eredményező információt tudunk felírni a lemez bármely sávjára. A programban a kurzorpozíció egyértelmű jelölésére szögletes zárójelben az irányt írjuk. Természetesen a programban azt értelemszerűen kell helyettesíteni.

```

370 CLOSE15
380 END
390 REM*****
400 TRY=0
410 PRINT#15,"M-W"CHR$(0)CHR$(0)CHR$(2)CHR$(T)CHR$(0)
420 PRINT#15,"M-W"CHR$(1)CHR$(0)CHR$(1)CHR$(JOB)
430 TRY=TRY+1
440 PRINT#15,"M-R"CHR$(1)CHR$(0)
450 GET#15,E#
460 IF E#="" THEN E#=CHR$(0)
470 E=ASC(E#)
480 IF TRY=500 GOTO 510
490 IF E)127 GOTO 430
500 RETURN
510 CLOSE15
520 PRINT "LEJIRVSNEM SIKERULTIROFF1"
530 END
540 REM 21-ES HIBA
550 DATA 32,163,253,169,85,141,1,28
560 DATA 162,255,160,48,32,201,253,32
570 DATA 0,254,169,1,76,105,249,234
    
```

A meghajtó memóriájában futó program forráslistája:

```

100 : *****
110 : 21-ES HIBA LEMEZREÍRÁSA
120 : *****
130 :
140 **$0500 : A PROGRAM A $0500-TOL KEZDŐDÖ
150 : PUFFERBA KERÜL
160 JSR $FDA3 : AZ IRÁS BEKAPCSOLÁSA
170 LDA#$55 : NEM SZINKRONBAJT
180 STA $1C01
190 LDX#$FF
200 LDY#$48
210 JSR $FDCB : 18432 NEM SZINKRONBAJT FELÍRÁSA
220 JSR $FE00 : AZ OLVÁSÁS BEKAPCSOLÁSA
230 LDA#$01
240 JMP $F969
    
```

Most nézzünk ugyanerre a levédési technikára egy másik programot, amely a sávnak csak egy megadott blokkját írja át, és így csak ezt olvasva kapunk 21-es hibaüzenetet.

```

100 REM 21-ES HIBAGENERÁTOR
110 DIM B$(7)
120 PRINT "CLR121 LEVEDENDO LEMEZT HELYÉZZE A MEGHAJTÓBA!"
130 INPUT "LEJIAZ ÁTÍRANDÓ BLOKK SAV- ES SZÉKTORSZÁMA":T,S
140 IF T<1 OR T>35 THEN END
150 NS=20+2*(T)17)+(T)24)+(T)30
170 IF S<0 OR S>15 THEN END
180 INPUT "LEJIVALÓBAN EZ A TORLENDŐ? (I/N)":Q#
190 IF Q#<<"I" THEN END
200 OPEN15,0,15
210 PRINT#15,"10"
220 INPUT#15,EN#,EM#,ET#,ES#
230 IF EN#="00" GOTO 280
240 PRINT "LEJ"EN#,"EM#","ET#","ES#"
250 CLOSE15
260 END
270 REM *****
280 IF S=0 THEN S=NS:GOTO 300
290 S=S-1
300 JOB=176
310 GOSUB 570
315 REM *****
320 REM OLVÁSZÁS
330 JOB=128
340 GOSUB 570
350 FOR J=0 TO 7
360 FOR I=0 TO 7
370 READ D
380 D#=#D#+CHR$(J)+CHR$(I)
390 NEXT I
400 NEXT J
    
```

```

410 I=0
420 FOR J=0 TO 7
430 PRINT#15,"M-W"CHR$(I)CHR$(S)CHR$(6)D#(J)
440 I=I+2
450 NEXT J
460 REM *****
465 REM VEGRÉHAJTÁS
470 PRINT#15,"M-W"CHR$(2)CHR$(0)CHR$(1)CHR$(224)
480 PRINT#15,"M-R"CHR$(2)CHR$(0)
490 GET#15,E#
500 IF E#="" THEN E#=CHR$(0)
510 E=ASC(E#)
520 IF E)127 GOTO 460
530 CLOSE15
540 PRINT "LEJKEZSI"
550 END
560 REM *****
570 TRY=0
580 PRINT#15,"M-W"CHR$(0)CHR$(0)CHR$(4)
    
```

```

CHR$(T)CHR$(S)CHR$(T)CHR$(S)
590 PRINT#15,"M-W"CHR$(1)CHR$(0)CHR$(1)CHR$(JOB)
600 TRY=TRY+1
610 PRINT#15,"M-W"CHR$(1)CHR$(0)
620 GET#15,E#
630 IF E#="" THEN E#=CHR$(0)
640 E=ASC(E#)
650 IF TRY=500 GOTO 680
660 IF E)127 GOTO 600
670 IF E=1 THEN RETURN
680 CLOSE15
690 PRINT "LEJIRVSNEM SIKERULTIROFF1"
700 END
710 REM AZ ADATBAJTOK
720 DATA 32,16,245,32,86,245,162,0
730 DATA 80,254,184,202,208,250,162,69
740 DATA 80,254,184,202,208,250,169,255
750 DATA 141,3,28,173,12,28,162,0,169
760 DATA 9,132,141,12,28,162,0,169
770 DATA 85,30,254,184,141,1,28,202
780 DATA 208,247,80,254,32,0,254,169
790 DATA 1,76,105,249,234,234,234,234
    
```

Most pedig nézzük meg az assembly forráslistát, hogyan működik.

KOVÁCS P. ATTILA

```

100 : *****
110 : 21-es hibagenerálás egy blokkra
120 : *****
130 :
140 : Az $520-es puffert használjuk
150 :
160 **$0500
170 :
180 JSR $F510 : A BLOKKFEJ MEGKERESÉSE
190 JSR $F556 : A SZINKRONJEL KERESÉSE
200 :
210 : VAR AMIG AZ ADAT MEGJÖN
220 :
230 LDX #500
240 READ1 BVC READ1
250 CLV
260 BEX
270 BNE READ1
280 :
290 LDX#$45
300 READ2 BVC READ2
310 CLV
320 BEX
330 BNE READ2
340 :
350 LDA#$FF :ADATIRÁNY KI (OUT)
360 STA $1C03
370 LDA $1C0C :IRÁSMÓD BEKAPCSOLÁSA
380 AND #51F
390 ORA #5C0
400 STA $1C0C
410 :
420 LDX#$00
430 LDA#$55
440 WRITE1 BVC WRITE1
450 CLV
460 STA $1C01
470 BEX
480 BNE WRITE1
490 :
500 WRITE2 BVC WRITE2
510 :
520 JSR $FE00 :OLVÁSÁSMÓD BEKAPCSOLÁSA
530 :
540 LDA#$01
550 JMP $F969
    
```


Integrált szoftver

A játékprogramozás technikája c. sorozat után hasonló felfogásban új témát indítottunk. A fenti sorozatcím alatt olyan szoftvereszközöket ismertettünk, mint az adatbázis-kezelő, szövegszerkesztő, adat-összefésülő, postázó és táblázatkezelő (spread sheet) programok.

A programcsomag elemei, programjai egymásra épülnek, egymás „termékeit” használják. Az integráltságot egy egyszerű példaként felismerhetjük a programcsomagban, ugyanis az azonos funkciójú változók elnevezése a különböző programokban azonos, és az azonos célú szubrutinok különböző programokban ugyanazon sorszámok alatt találhatók. A programok tárgyalásánál ezért az ilyen szubrutinokat csak egyszer, legelső előfordulásuknál tárgyaljuk, a későbbiekben csak utalunk rájuk.

Törekvésünk ezeknek az egyszerűen elkészíthető és használható programoknak a bemutatása. Minden programnál először az általános tudnivalókat közöljük, majd a programot, kezelésének előírásaival és a hozzá tartozó táblázatokkal együtt. A könnyebb érthetőség céljából, de a jól használhatóságot szem előtt tartva, mindig egyik program BASIC nyelven íródott, DRAGON BASIC változatban. A DRAGON gépen tulajdonképpen a Microsoft BASIC a megfelelő változat, és így a program könnyen átirható például VT, TRS, HT, IBM PC stb. gépekre. A Commodore és a Sinclair gépeknél egy kicsit több módosítás szükséges. (Következő számunkban egy segéd táblázatban bemutatjuk a gépek közötti eltéréseket.) A programok a közölt formában, változtatás nélkül csak a DRAGON gépen futtathatók ugyan, de az apró változtatások, módosítások az alábbiak — a szokatlan, más gépektől eltérő értelmezésű utasítások, kódok, adatok — alapján elvégezhetők.

- CLEAR *n* — helyet foglal le a tárolóban a karaktersorozatnak;
- CLS *n* — a képernyőt az *n* színkóddal megadott színre „festi”;
- PRINT & *n* — a képernyő adott pontján kezdi a megjelenítést, melynek koordinátái:
Y=INT(*n*/32)
X=*n*-INT(*n*/32)
- STRING\$(*ab*) — *a* számú *b* ASCII kódú karaktert képez;
- POKE 329,0 — kisbetűk bekapcsolása (255 a kikapcsolás);

- PRINT USING "##,###" — például 5 számjegyet ír, az ezresek után vesszőt tesz;
- INKEY \$ — egy bebillentyűzött karaktert olvas be;
- IF... THEN... ELSE — a feltétel teljesülésénél THEN és az ELSE közötti részzel, a nem teljesülésnél az ELSE utáni ággal folytatódik;
- LINE INPUT — karaktersorokat visz be (a sor eleji üres karaktereket is beleértve);
— a felfelé mutató nyíl (↑) kódja;
— azonos az előzővel + shift;
— a lefelé mutató nyíl (↓) kódja;
— azonos az előzővel + shift;
— CR/LF kódja;
— a visszalépés kódja (←);
— a jobbra mutató nyíl kódja (→);
— a sárga színű fénypont (kurzor);
- IF PEEK(342) THEN... — azt vizsgálja, hogy le lett-e nyomva a 247 kódértékű billentyű vagy sem;
- SOUND (*ab*) — hangkeltés, ahol az *a* értékkel a hang magasságát adjuk meg (1—255; az alapskála C-je 89), a hang hosszúságát *b* értéke adja meg (1—255 között); 16 egyenlő 1 s);
- INSTR *a*, *b*\$, *c*\$ — *c*\$ karaktersorban kikeresési a *b*\$ karaktersorozatot az *a*-adik pozíciótól kezdve;
- OPEN "a", # — 1 — állomány megnyitása kazettán: kiírásra (0) vagy beolvasásra (1);
- PRINT # — 1 — kiírás a kazettára;
- CLOSE # — 1 — az állomány lezárása a kazettán;
- MOTORON — kazettamotor bekapcsolása;
- INPUT # — 1 — adatbeolvasás a kazettáról;
- EOF (# — 1) — sorvégjelzés a kazettán;
- POKE & H3FF,*n*-soros (1) vagy párhuzamos (0) csatolás-e a nyomtató, # — 2 a nyomtatócsatorna száma;
- CHR\$(12) — a képernyőtörlés kódja;
- PCLEAR *n* — tárfoglalás a képernyő számára (1,5 kb-át egységekben);

- CSAVE "név" — a program tárolása kazettán;
- CLOAD "név" — a program beolvasása kazettáról;
- MOTOROFF — a magnómotor kikapcsolása.

A képernyőn a telexszövegeknél megszo-

- Á = AA
- É = EE
- Ö = OE
- Ü = UE

Ez a rendszer nem ismer se hosszú, se ket-

tős ékezeteket. A programokban mindenütt kazettás tároló szerepel, de nagyon egyszerű a program átírása lemezrendszerre. Az egyes programok részletes ismertetésénél erre még visszatérünk.

Az adatbázis-kezelő rendszer

A program egy ún. „kulcsszó a szövegben” típusú adatbázis-kezelő rendszert valósít meg. Ez azt jelenti, hogy az egyes adatokhoz tartalmuk — nem pedig egy index — alapján lehet hozzáférni. Ez a módszer is elengedően gyors keresést és adatvisszanyerést eredményez. Maga a program, de más adatbázis-kezelő programok is a vizsgált adatbázist a gép belső tárolójában tartja. Így a DRAGON-nál mintegy 300 rekorddal, rekordonként átlagosan 100 karakterrel képes dolgozni. A maximális karakterhossz 240, a 100 átlagérték. Ha valaki kevesebb, de hosszabb rekorddal kíván dolgozni, akkor a programban az AV változó értékét kell megváltoztatnia (60-as sor) a kívánt, de 240-nél nem nagyobb értékre. Ez (és a többi program is) tartalmaz egy másolatkészítő részt (itt lásd a 9999. sorszámú sort).

A kezdet

A program elindítása után kéri az adatbázis nevét. Háromféle válasz adható:

1. Ha a válasz „ÚJ”, akkor az új adatbázis nevét kell beadni. Kezdetben, amikor még nincs kialakítva az adatbázis, csak ez a válasz helyes. Az adatbázis neve ennél a

géptípusnál és programnál legfeljebb 8 karakterből állhat.

2. Ha egy adatbázisnevet adunk meg, akkor a program ezt a kazettán megkeresi és betölti (a könnyebb működéshez a magnó számlálóját figyelve állítsuk be a kazettát egy adatbázis közelében lévő pozícióra). Egy adatbázisnevet keresésre természetesen csak akkor adhatunk meg, ha már van adatbázisunk, és annak neve megegyezik a kazettán lévő adatscsoport nevével.

3. Lenyomjuk az ENTER gombot — az ADATOK elnevezésű adatscsoportot keresi ki a kazettán és betölti. Ezután a program megkérdezi, hogy soros vagy párhuzamos

csatolású nyomtatót használunk-e, majd a választ követően a képernyőn megjelenik a főmenü.

A főmenü

A főmenüben négy választási lehetőségünk van.

1. *Adatbeadás (B)*. Ebben az esetben egy új rekordot adhatunk meg, és ezt ismételhetjük mindaddig, amíg be nem adjuk az új (U) választ. A rekordok beadásánál két segédkarakter segítségével dolgozhatunk. A soremelő hatása ismert. A nagyobb jel (>) 10 üres hely beiktatását jelenti a re-

kordba. A már korábban megadott hosszig minden ASCII karaktert fogad a rekord.

2. *Az adatok előkeresése és kinyomtatása*. A képernyőre kiírt kérdésre válaszként be kell írni, mi a kikeresendő. Ezután a program végigvizsgálja az adatbázist, és az egyezőnek talált rekordokat kinyomtatja. Ha a keresendőre válaszolva az ENTER gombot nyomjuk le, akkor a teljes adatbázis kinyomtatása az eredmény.

3. *Ha a válaszunk (U), az adatbázist eltárolja a kazettára*. Ha a tárolás befejezése után újra az U betűt adjuk a kérdésre válaszként, visszatérünk a főmenübe; bármely más betű esetén másolatot készít az adatbázisról. Fontos, hogy előbb a magnetofont bekapcsoljuk, vagyis felvételre állva és a kazettát a megfelelő helyre állítva kell várnunk a motorbekapcsolás és az adatátvitel megkezdését.

4. *Kikeresés az adatbázisból, elővétel (E)*. A program megkérdezi, hogy mi az, amit keresünk. A válaszként megadott kód, illetve kódcsoporthoz a szövegben való elhelyezkedéstől függetlenül érvényes. Így például ha a válasz "MA", akkor kikeresi mindazokat a rekordokat, amelyekben ez előfordul, vagyis ha találkozik ALMA, HAGYMA, MALOM, MAMA stb. szavakkal vagy ezeket tartalmazó rekordokkal, mindegyik előkerül. Ha nem talál egy megfelelőt sem, a képernyőn megjelenik egy almenü.

Almenü

A főmenühöz hasonlóan itt is négy választási lehetőségünk van.

1. *Törlés (K)* — a teljes rekordot törli.
2. *Nyomtatás (N)* — a rekordot kinyomtatja a talált formában.
3. *Az adatbázis mozgathatása (!)*. A DRAGON gépnél a ↓ és ↑ által tudjuk az adatbázist mozgatni.
4. *Szerkesztés (S)* — e válasz után megjelenik egy újabb, alacsonyabb szintű almenü.

A lehetséges három válasszal: betoldás (B), csere (C), törlés (K) módosíthatjuk a szöveget. Az ENTER gomb lenyomásával visszatérhetünk az eggyel magasabb szintű menühöz. A három (B, C, K) szövegmódosító lehetőség nem különbözik a szöveg-, illetve képernyőszerkesztőnél szokásos módszertől, vagyis a fénypont (kurzor) helyéhez képest jobbra engedélyez módosítást vagy törlést. Itt is használhatjuk a már korábban említett két kontroll kódot, az üres hely betoldást és a soremelést. A bemutatott lista tartalmazza az adatbázis-kezelő programot, amelyre a továbbiakban még visszatérünk.

GALINA FERENC

```

TYPE ADATBAZIS
10 CLEAR300:CLS0:GOTO 1030
20 FOR L=1 TO 1000:NEXT L:RETURN
30 AS=INKEY$
40 AS=INKEY$:IF AS="" THEN 40 ELSE
RETURN
50 PRINT &481,">"=10 UERES HELY,
"&"=UJ SOR":RETURN
60 CLEAR32000:AV=100:T=INT(30000/AV):
N=T:DIM T$(T):CLS:H$="ADATOK":
+STRING$(10,223)+STRING$(11,239):
GOSUB 740:PRINT:INPUT"ADATBAZIS
NEEV":DS$
70 IF DS="" THEN S$="ADATOK"
80 IF DS="UJ" THEN INPUT"UJ ADAT
BAZIS NEEV":DS$:IF DS="" THEN
DS$="ADATOK":GOTO 200 ELSE
GOTO 200
90 PRINT DS$:"-KERESEES"
100 OPEN "I",-1,DS$:PRINT:PRINT"BE
TOEL
TEES":DS$:FOR X=0 TO T:LINEINPUT#-1,
T$(X):IF EOF(-1) THEN X=T
110 NEXT X:CLOSE:X=0
200 GOSUB 730:PRINT:PRINT"SOROS,
VAGY
PAARHUZAMOS A NYOMTATO?":INPUT
"<S> VAGY <P>":AS:IF AS="S" THEN
POKE &H3FF,1 ELSE POKE &H3FF,0
500 P$=""GOSUB 730:PRINT &418,"<N>
ELOVEEDEL EES KINYOMTATAAS":PRINT
TAB(2)"<E>LOVEEDEL <B>EADAAS <U>
J":GOSUB 30:GOSUB 730
510 IF AS="U" THEN 600 ELSE IF AS="B"
THEN 650 ELSE IF AS="N" THEN AS=""
:AP$="N"
520 GOSUB 740:PRINT &128,"":LINEINPUT
"KERESENDO?":S$
530 X=0:XD=1
540 IF T$(X)="" THEN GOSUB 740:GOTO 610
550 P=INSTR(1,T$(X),S$):IF P=0 THEN 360
560 GOSUB 730:GOSUB 940:PRINT:PRINT T$:
IF AP$="N" THEN 600
570 PRINT &481,"<K>ITOEEROL <N>YOMTAT
VAGY":CHR$(24):GOSUB 30:GOSUB 730
580 IF AS="K" THEN T$(X)=""
590 IF AS="S" THEN GOSUB 750:GOTO 560
600 IF AS="N" OR AP$="N" THEN PRINT &33,
"NYOMTAT":GOSUB 940:GOSUB 8200:OPEN
"O",-2,"W":PRINT#-2,T$:PRINT#-2:
CLOSE-2:GOSUB 730
610 IF AS=CHR$(10) THEN XD=1 ELSE IF AS
=CHR$(94) THEN XD=-1
620 IF AS="U" THEN IF XD=-1 THEN X=0
ELSE IF XD=1 THEN X=T-1
630 X=X+XD:IF X < 0 OR X > T THEN AP$=""
ELSE GOTO 540
640 X=0:GOSUB 730:PRINT:"AZ":GOSUB 20:
GOTO 500
650 FOR X=0 TO T
660 IF T$(X)="" THEN GOSUB 730:GOSUB
50:PRINT &65,"ADATBEADAAS, VAGY <U>
J":LINEINPUT T$(X):GOSUB 730
670 IF T$(X)="" THEN T$(X)=""GOTO 500
680 NEXT X:GOTO 500
690 GOSUB 730:PRINT"ELTAAROLAAS":GOSUB
7110:GOSUB 730
700 PRINT &136,"ELTAAROLAAS":DS$:FOR X=
1 TO 6000:NEXT:OPEN "O",-1,DS$:FOR
X=0 TO T:IF T$(X)="" THEN 710 ELSE
PRINT#-1,T$(X)
710 NEXT X:X=0:CLOSE:I$=INKEY$:GOSUB 730
:PRINT:PRINT"<U>J, HA MAAS BILLENTY
UET NYOM LE MEEG EGY MAASOLATOT KEE
SZIT"
720 I$=INKEY$:IF I$="" THEN 720 ELSE IF
I$="U" THEN END ELSE 690
730 CLS
740 PRINT &0,H$;
742 PRINT &32,X;N;"LEHETSEGESBOEL":
RETURN
750 LL=0:T$=T$(X)
760 GOSUB 730:PRINT&416,"<B>ETOLDAAS
<C>SERE <K>ITOEERLES"
770 PRINT &448,"VISSZATEERES <ENTER>":
GOSUB 50
780 PRINT &96,MID$(T$,1,LL):PRINT &96
+LL,CHR$(159):PRINT&97+LL,MID$(T$,
,LL+1):I$=INKEY$:I$=""
790 IF I$=CHR$(8) THEN IF LL>0 THEN LL=-
1:GOTO 780
800 IF I$=CHR$(9) THEN IF LL < LEN(T$)
THEN LL=LL+1:GOTO 780
810 IF I$=CHR$(94) THEN IF LL>32 THEN
LL=-32:GOTO 780
820 IF I$=CHR$(10) OR PEEK(342)=247 THEN
IF LL+32<=LEN(T$) THEN LL+32:GOTO 780
830 I$=INKEY$:IF I$="" THEN 830
840 IF I$=CHR$(8) OR I$=CHR$(9) OR I$=
CHR$(94) OR I$=CHR$(10) THEN 790
850 IF I$<"K" THEN 880
860 I$=INKEY$:IF I$="" THEN 860 ELSE IF
I$<"K" THEN 780
870 SOUND100,1:IF LEN(T$)>LL THEN T$=LE
FT$(T$,LL)+RIGHT$(T$,LEN(T$)-LL-1):
PRINT&97+LL,MID$(T$,LL+1):SOUND140,1
:GOTO 860
880 IF I$<"B" THEN 900
890 PRINT &97+LL,STRING$(100,32):PRINT &
192+LL,MID$(T$,LL+1):PRINT &LL+96,"
":LINEINPUT I$:T$=LEFT$(T$,LL)+I$+
RIGHT$(T$,LEN(T$)-LL-LL+LEN(I$)
:GOTO 760
900 IF I$<"C" THEN 920
910 PRINT&LL+96,"":LINEINPUT I$:IF
LEN(I$)>LEN(T$)-LL THEN T$=LEFT
$(T$,LL)+I$ ELSE T$=LEFT$(T$,LL
)+I$+RIGHT$(T$,LEN(T$)-LL-LEN
(I$):LL=LL+LEN(I$):GOTO 760
920 IF I$=CHR$(13) THEN T$(X)=T$:
GOSUB 730:RETURN
930 GOTO 780
940 P=1:T$=T$(X)
950 P=INSTR(P,T$,"&")
960 IF P=0 THEN 980
970 T$=LEFT$(T$,P-1)+CHR$(13)+RIGHT$(T$,
,LEN(T$)-P):GOTO 950
980 P=1
990 P=INSTR(P,T$,">")
1000 IF P=0 THEN 1020
1010 T$=LEFT$(T$,P-1)+STRING$(10,32)+
RIGHT$(T$,LEN(T$)-P):GOTO 990
1020 RETURN
1030 CLS2:PRINT&195,"ADATBAAZIS":
M=MEM
1040 GOSUB 20:GOSUB 20
1050 PCLEAR1:GOTO 60
7110 FOR X=1 TO 20:SOUND200,1:PRINT &320
,"":PRINT &325,"A MOTOR BEKAPCSOLVA
":NEXT X:MOTORON:FOR X=1 TO 8000:
NEXT X:X=0:RETURN
8200 ZY$=""FOR X=1 TO LEN(T$):ZX$=MID$(
T$,X,1):IF(ZX$<"") AND (ZX$<"$")
AND (ZX$<"&") AND (ZX$<"#") THEN
8250
8210 IF ZX$="#" THEN ZX$=CHR$(125):GOTO
8250
8220 IF ZX$="#" THEN ZX$=CHR$(124):GOTO
8250
8230 IF ZX$="#" THEN ZX$=CHR$(96):GOTO
8250
8240 ZX$=CHR$(126)
8250 ZY$=ZY$+ZX$:NEXT X:T$:RETURN
9999 MOTORON:FOR X=1 TO 8000:NEXT:CSAVE
"ADATOK":MOTORON:FOR X=1 TO 999:
NEXT:CSAVE"ADATOK"

```


ADOK — VESZEK — CSERÉLEK

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

ADOK

Commodore VIC-20-as gépemet eladom programokkal. Tel.: 420-863. Cím: Patai Gábor, Budapest VII. Damjanich u. 38. 1071.

C-16-ra, C plus/4-re és C-116-ra jutányos áron játékok programok eladók. Dombóvári Árszló, Kiskőrös, Majakovszkij utca 25. 6200.

Sinclair Spectrum megkoptott /fém/ fedőlapját ócsón, gyorsan újjávarázsolom. Képjó Robert Budapest, IX. Imre u. 5. II.14. 1093. Tel.: 378-075.

VESZEK

ZX-Spectrum (16 vagy 48 K-s) számítógépet vennék. Billentyűzethiba gép vagy üzembeképes nyák is megfelel. Tel.: 12-901 Ajka.

CSERÉLEK

C-64-es programokat cserélek lemezen és kazettán. Listát kérek. Boldizsár Gábor, Rém, Petőfi u. 23. 6446.

Commodore 16 és Plus/4 programokat cserélnék. A programlistát a következő címre várom: Fazekas Attila, Kemece, Kecskéshegy u. 15. 4501.

HomeLab-3 tulajdonosokat keresek, tapasztalat-, és programcsere céljából. Szijjártó Ferenc, Óroshegy, Juhász Gy. u. 20. 2/10. 5900.

Játékok programokért cserébe a következő programokat kérem: PROFI ASS 64, SYNTAX 64, GRAPHIC BASIC, PROFIT MON 64, SUPERGRAFIK 64, vagy ehhez hasonló BASIC bővítések, segítőprogramok. Madaei Tamás, Pécel, Kőr u. 19. 2119.

LASER 210-es Color Computer tulajdonosokkal szeretnék kapcsolatot felvenni, tapasztalat- és programcsere céljából. Cím: Venczel Árpád, Vác, Lenin u. 82. VII/30. 2600.

VC 20 alapgépen és (8-32 K-s) bővítővel ellátott gépen futó programokat cserélnék vagy vennék. Program Cartridge 1000 Ft-ért eladó. Programlistákat Marincsak János, Kisvárda, Lenin út 20. 4600 címre kérek. Felhasználói programok is érdekelnek. VC 20-al foglalkozó folyóiratokat /külföldi/ keresek.

VC 20 programokat cserélnék (32 Kbyte-ig) kazettán. Török Csaba, Komárom, Lengyár-telep V. 2900.

ZX-Spectrum (48 K-ra) készült színvonalas programokat cserélnék. /pl. MICROBOT, TOMAHAWK, CAULDRON, BOOTY/ A választásokat programlistával várom. Óhidi Olivér, Szombathely, Engels u. 64/C, 9700.

ZX-Spectrum felhasználói és játékok programokat cserélek. Leveleket programlistával a következő címre kérem: Szarka Endre, Veszprém, Stadion út 22. 8200.

ZX-Spectrum programokat cserélek. Válaszokat a programok listájával erre a címre kérem: Frank Gábor, Iatabánya II. Vadász u. 68. 4/2. 2800.

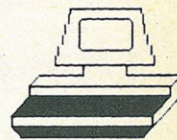
ZX-Spectrumra 1000 darabszámú játékprogram készletet cserélem folyamán szeretném bővíteni. Válaszokat programlistával a következő címre kérem: Cseh Attila, Debrecen, Kapitány u. 16. 4033.

280-as egykártyás mikrogép (billentyűzettel, display-el, software-vel) adok, cserébe Vermona dobogókat kérek a következő címre. Balogh György, Budapest, Bertói u. 14. 1172.

THE USERS PORT

San Fernando Valley Commodore Users Group

Egy jól használható programjavító segédlet



Nagyméretű BASIC programokban hibák keresése gyakran nagyon munkaigényes, és esetleg újabb hibákat okozó művelet. Ez különösen akkor igaz, ha ezt a munkát manuálisan végezzük.

A legtöbb C64 használó ma már rendelkezik vagy az eredeti EASYSRIPT programmal, vagy annak valamilyen hasonmásával. Ennek használata sok esetben segít a következők szerint:

— Készítsünk a BASIC programról egy az EASYSRIPT által olvasható formátumú másolatot.

Ennek módja: Beadandó:

OPEN8, 8, 8, "Ø: új fájlnev, S, W": CMD8:
LIST

PRINT #8

CLOSE8

(Amit mi külön sorba írtunk, azt külön sorba is kell írni!)

— Töltsük be az EASYSRIPT programot.

— Töltsük be az „új fájlnev” elnevezésűt (ami természetesen nem ilyen furcsa nevű, hanem új fájlnev alatt bármilyen EASYSCRIPT-olvasható fájlnev szerepelhet).

— Az EASYSRIPT parancsait használva keressünk.

Mire használható ez? Sok mindenre. Ebből itt csak néhány példát emelünk ki. Azokat, amiket talán a leggyakrabban használhatunk.

— Minden olyan utasítás kikeresése, ami egy adott szubrutin sorszámra hivatkozik.

— Minden olyan hely kikeresése, ahol egy adott változónév előfordul.

— Annak a helynek a megkeresése, ahol egy változót definiáltunk.

— Annak a helynek a megkeresése, ahol egy tömböt definiáltunk.

— Annak a helynek a megkeresése, ahol egy tömböt dimenzionáltunk.

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)

— Annak a helynek a megkeresése, ami a következő típusú hibaüzenet tényleges okozója: BAD SUBSCRIPT ERROR LINE aaa. (Általában az aaa-adik sorban semmiféle hibát nem találunk, ezért van erre szűkség. A tényleges ok lehet pl. az, hogy a dimenzionálásnál nem megfelelő paramétert, paraméterértéket használtunk, vagy a program futása során ezt az értéket módosítottuk.)



Adom a magyarázatot!

Legutóbb egy nyomtató-rajzoló program két változatát közöltük, melyek — szerzőjük szándéka szerint — csak futási idejükben különböztek volna, azonban az előállított rajzok is különböztek.

A két program egyetlen sor egy részletében tért csak el, melyeket a jobb tárgyalhatóság céljából elkülönítve közlünk (1. és 2. lista 50. sor). A sorban még egy nyomtatást is beleírtam, melynek magyarázatát később adom.

A két változat látszólag ugyanazt teszi, csak kétféle módon. Ez pedig az, hogy az Y(0)-bajtot nyolc bitértékre bontja, minden egyes bitszámjegyet megszorozva százhuszonhéttel. (Ez utóbbi -E- a nyomtatáshoz kellett.)

Az első változatnál ez úgy történt, hogy először a bajt számértékét, a továbbiakban a maradványértéket osztotta kettő megfelelő hatványával és annak egészrészét megszorozta ugyanezzel a hatvánnyal. (A svéd abc-t használó nyomtatón ü jelent meg hatványjel helyett.) Az így kapott helyiértékkel szorozott számjegyet (B) levonta az első ciklusnál a bajt számértékéből, a továbbiakban a maradványértékéből, előállítva ezzel az új maradványértéket $Y(K)/Y(K)$. Végül a számjegyet szorozta százhuszonhéttel. (Bár ez utóbbi helyett egy feltételes utasítás áll, az azonban ezt teszi,

$$127 \times 0 = 0$$

$$127 \times 1 = 127$$

A második változatnál kettő hatványainak előállítására először kettőszázötvenhatnak, a továbbiakban az előző osztásértéknek (C) kettővel történő osztása szolgált. Más eltérés a két változat közt nincs. Mindkét változatot kiegészítettem a bajtértékmegadással (1. sor és a 80. sor első két utasítása). Ez 0 és 255 lett. Rajzolás helyett pedig kiírtam E értékét (70. sor). Az 1. és 2. futtatás eredménye különböző eredményt szolgáltatott. Amennyiben a bajtérték 0 volt, azaz a bitérték

00000000

úgy a két futtatás azonos eredményt adott. Kettőszázötvenöttnél, azaz

11111111

bitértéknél a 2. futtatás valóban minden egyes helyén százhuszonhetet adott, az 1. viszont a legalacsonyabb helyiértéknél nem. A helyes értéket a 2. lista szolgáltatta, vizsgáljuk meg ezért a 1-t részletesebben. Átalakítva a listát, a részletszámítások eredményét táblázatos formában kapjuk meg (3. lista és 3. futtatás). Itt a 10. sorral előállítottam a táblázat fejlécét, az 50. sorból elhagytam a bajtérték kiírását, és a 70. sorban minden részeredményt kiírtam.

A táblázat szolgáltatja a magyarázatot. Látható, hogy a hatványozás eredménye kettő ötödik és hatodik hatványánál igen kis értékkel, de eltér a helyestől. Ez ott még nem ad a rajzolás szempontjából hibát. Továbbviszi azonban a számítás a pontatlan értéket, most még hibát nem okozva. A baj az utolsó B-számításnál következik be. Itt

Y(7) helyes értéke 1 lenne, melyet kettő nulladik hatványával osztva egyet kapnánk. Ebből egészértéket képezve továbbra is 1 adódna, amelyet eggyel szorozva

$$B = 1$$

lenne az eredmény (és így $E = 127$).

```
1 Y(0)=0
50 PRINT#-2:PRINT#-2,"Y=";Y(0):C=256:FOR
K=1T08:C=C/2:B=C*INT(Y(K-1)/C):Y(K)=Y(K-
1)-B:IFB<>0THEN E=127:GOTO70 ELSE E=0:GO
T070
70 PRINT#-2,E;:NEXTK
80 IFA=0THENA=1:Y(0)=255:GOTO50
90 PRINT#-2:STOP
```

1. lista

Y = 0
0 0 0 0 0 0 0 0
Y = 255
127 127 127 127 127 127 127 0

1. futtatás

```
1 Y(0)=0
50 PRINT#-2:PRINT#-2,"Y=";Y(0):FOR K=1T08
:B=(20(8-K))*INT(Y(K-1)/(20(8-K))):Y(K)=
Y(K-1)-B:IFB<>0THEN E=127:GOTO70 ELSE E=
0:GOTO70
70 PRINT#-2,E;:NEXTK
80 IFA=0THENA=1:Y(0)=255:GOTO50
90 PRINT#-2:STOP
```

2. lista

Y = 0
0 0 0 0 0 0 0 0
Y = 255
127 127 127 127 127 127 127 127

2. futtatás

```
1 Y(0)=0
10 PRINT#-2,"Y(K)", "B", "E"
50 FORK=1T08:B=(20(8-K))*INT(Y(K-1)/(20(
8-K))):Y(K)=Y(K-1)-B:IFB<>0 THEN E=127:G
OTO70 ELSE E=0:GOTO70
70 PRINT#-2,Y(K),B,E:NEXTK
80 IFA=0THENA=1:Y(0)=255:GOTO50
90 PRINT#-2:STOP
```

3. lista

3. futtatás

Y(K)	B	E
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
127	128	127
62.9999997	64.0000001	127
30.9999997	32.0000001	127
14.9999997	16	127
6.99999966	8	127
2.99999966	4	127
.999999658	2	127
.999999658	0	0

```
1 Y(0)=7
10 PRINT#-2,"Y(K)", "B", "E"
50 FORK=1T08:B=(20(8-K))*INT(Y(K-1)/(20(
8-K))):Y(K)=Y(K-1)-B:IFB<>0 THEN E=127:G
OTO70 ELSE E=0:GOTO70
70 PRINT#-2,Y(K),B,E:NEXTK
80 IFA=0THENA=1:Y(0)=170:GOTO50
90 PRINT#-2:STOP
```

4. lista

4. futtatás

Y(K)	B	E
7	0	0
7	0	0
7	0	0
7	0	0
7	0	0
3	4	127
.999999997	2	127
.999999997	0	0
41.9999998	128	127
41.9999998	0	0
9.99999978	32.0000001	127
9.99999978	0	0
1.99999977	8	127
1.99999977	0	0
1.99999977	0	0
.999999773	1	127

E helyett azonban

$$Y(7) = .999999658$$

vagyis nagyon kevéssel, de kisebb, mint 1. Ezt eggyel osztva is egynél kisebb számot kapunk, amelynek egész része 0.

Más bajtértékkel (7, 170 a 4. listában) kaphatunk hasonlóan a legalacsonyabb helyiértéken hibát (4. futtatás 8. sor), de kaphatunk kettős hibát is (4. futtatás két utolsó sor, ahol a bitek számértéke felcserélődött).

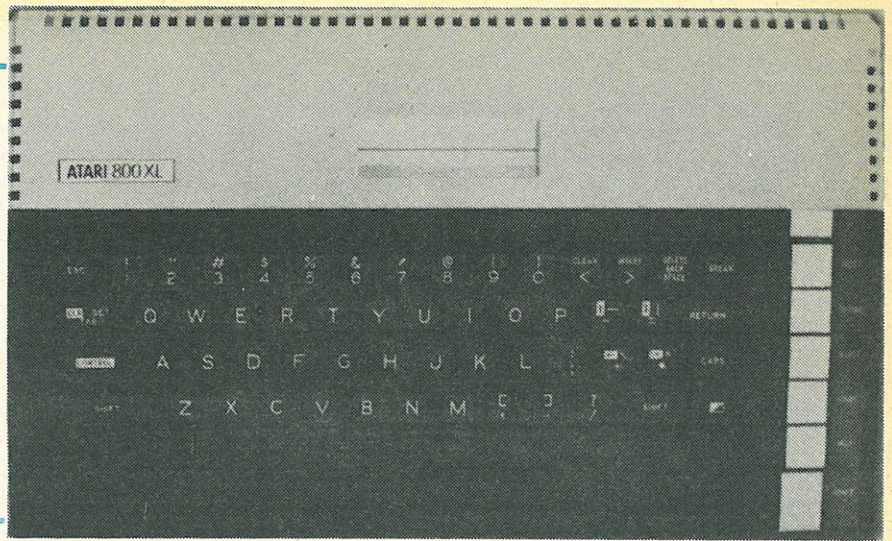
Az ok tehát az, hogy a hatványozás látszólag csekély (mintegy tízmilliomod százalék) pontatlansága téves döntést eredményezett, mert egy ennél pontosabb (itt tízenkét jegyre pontos) egészértékképzési művelettel ellenőriztük. A számítógép sem abszolút pontos eszköz, és előfordulhat (mint esetünkben is), hogy a látszólag nagy pontosság sem elég nagy. Célszerű tehát megvizsgálunk minden esetben a számítások pontatlanságának a hatását, ha nem akarunk pl. tücsköt-bogarat rajzoltatni bogár helyett.

dr. SIMONYI ENDRE

Új mikroklub

A HCC-n belül a Húsiipari Műszaki Fejlesztés Közös Vállalat (HMFKV) támogatásával megalakult az Alpha-Micro klub. — Minden érdeklődőt szeretettel várunk a klub rendezvényein és azok előkészítésére. Az összejövetelek helyszínéről és időpontjáról tájékoztatást ad Papp György a 337-350/24 vagy 329-580/328 telefonszámon. Levélcíme: HMFKV, Budapest, Gubacsi u. 6/b 1097.

Az ATARI[®] 800 XL



1. kép

Ez a gép az ATARI 400-as, 600-as számítógépek utódja. Az ATARI amerikai érdekeltségű cég, ezeket a gépeket azonban a világ különböző pontjain létesült leányvállalatok állítják össze. A 800XL a fent említett gépekkel felülről kompatibilis. A gép 64 Kb-ot tartalmaz, ezen felül beépített BASIC interpreter található benne. Nagy előnye még, hogy ismeri a nemzetközi karaktereket, melyek között számos ékezetes magyar betű is megtalálható. A ROM-ba beégettek még egy kis különálló programot (SELF-TEST), melynek segítségével tesztelhetjük a billentyűzetet, a memóriát és a hanggenerátort. Már az eddigiekből is látszik hogy a gép a házi számítógépek kategóriájába tartozik, komoly ipari és nyilvántartási feladatok ellátására nem alkalmas.

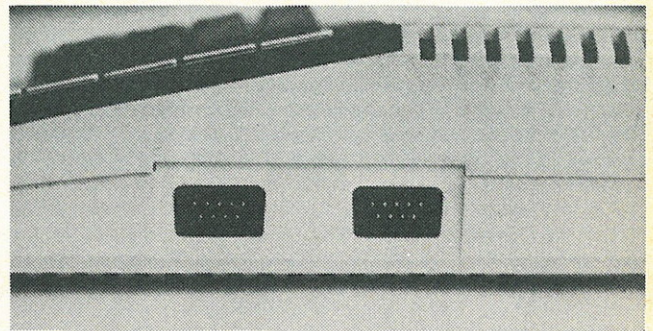
A gép kívülről

Formája esztétikus, a tervezők törekedtek a legmegfelelőbb alak kiválasztására. A billentyűzet kialakítása és elrendezése kényelmessé teszi a gép kezelését. A számítógépeken általában használt billentyűkön kívül, a jobb oldalon öt darab fémbillentyű található (1. kép). A legfelső a RESET, mely többféle módon funkcionál. A másik négy billentyű használatáról a későbbiekben írunk.

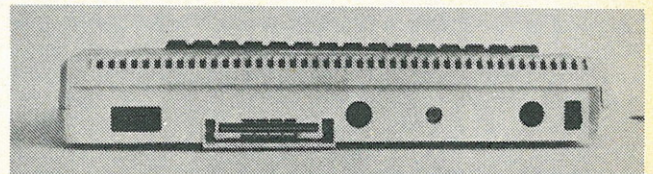
A billentyűzet fölött található kivezetés szolgál a cartridge-ok csatlakoztatására, melynek kialakítása nagyon célszerű, mivel a valódi csatlakozó fölé billenőajtós porvédő került. A géphez kapható programok egy része (főleg játékok), cartridge-on kerül forgalomba. Jobb oldalon helyezkedik el a két joystick kivezetés (2. kép), amely természetesen megfelel az ATARI cég által bevezetett szabványoknak. A COMMODORE cég ugyanezt alkalmazta a C-64 megépítésekor.

Hátul öt darab csatlakozó, és a ki/be kapcsoló kapott helyet (3. kép). Balról jobbra haladva az első csatlakozó a gép, a lemez meghajtók és a nyomtató közötti kapcsolat megteremtésére szolgál. Amint az a 4. képen is látható, eltér a Magyarországon elterjedt típusoktól. A második egy szabványos RS232 csatlakozó. A következő a MONITOR, s mellette az antenna kimenet. Az utolsó a tápfeszültség fogadására szolgál.

A dobozon több szellőzőnyílást helyeztek el, melyek biztosítják a megfelelő hűtést (5. kép).

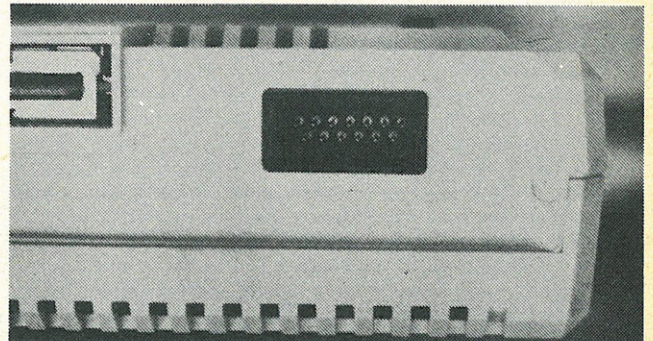


2. kép



3. kép

4. kép



Belső felépítés

A 6502-es mikroprocesszoron kívül négy darab speciális integrált áramkört helyeztek el (6. kép), melyek funkciója a következő:

- ANTIC: -képernyőszerkesztés (DMA, NMI kontroll)
-vízszintes és függőleges finomscroll
-szinkronizáció
-fényceruza pozíciójának lekérdezése
- CTIA: -a grafika és a sprite-ok közötti prioritás szabályozása
-szín és fényerő kontrollálása
-sprite-ok kezelése

- ütközések vizsgálata és lekezelése
- tűzgombok vizsgálata

- POKEY:**
- billentyűzet vizsgálata, kezelése
 - soros ki/bemenet vezérlése
 - potméterpozíciók megállapítása
 - belső órák vezérlése
 - perifériák megszakításrendszerének kiszolgálása
 - hanggenerátor
 - véletlenszám generátor

- PIA :**
- joystick lekérdezés
 - perifériák kezelése
 - perifériák megszakításrendszerének kiszolgálása

A gép NTSC és PAL rendszerű tévékkel egyaránt képes üzemelni. Az ATARI 800XL grafikai lehetőségei messze felülmúlják a hasonló kategóriájú gépekeit. Ezt bizonyítja, hogy 16 féle grafikus üzemmód közül választhatunk, de nem akárhogyan. Ugyanis egy ún. displaylist segítségével minden egyes tévéssorra más-más grafikus üzemmódot választhatunk. Az elérhető legnagyobb felbontás 384x240. A felbontás csökkentésével egyre több szint használhatunk egyszerre. 16 féle szín, 16 féle fényerő, így összesen 256 féle árnyalat áll rendelkezésünkre. Ez is sokkal nagyobb teret biztosít mint akár a C-64, akár a ZX-Spectrum grafikája, hogy magyar gépekről ne is beszéljünk. A gép felépítése sprite-ok használatának lehetőségével segíti elő a programozói munkát. Kétféle sprite-ot különböztetünk meg, melyek csupán méreteikben térnek el egymástól. Az egyik az ún. PLAYER, amely 240 pont magas és 8 pont széles. A másik sprite a MISSILE mely szintén 240 pont magas, de csak 2 pont széles. Mivel a sprite-ok teljes képernyő magasságúak, sőt a keretre is kilóghatnak, így hardware úton csak vízszintes irányú mozgításukra van lehetőség. Külön regiszterekben adhatjuk meg, hogy egy sprite eltakarja-e a másikat, vagy a képernyőn lévő grafikát, vagy sem. Továbbá lehetőségünk nyílik az ütközések egyszerű lekérdezésére. A sprite-ok méreteit külön-külön szabályozhatjuk. Vízszintesen három, függőlegesen két féle méret létezik.

A zenei kibontakozásnak szintén tág teret biztosít a négy egymástól teljesen független hangcsatorna. Mindegyiken 3.5 oktávot foghatunk át, de ha kettőt összevonunk, ez 8 oktávra bővül. Itt már különböző szűrők segítségével állíthatjuk be a végleges hangot. Így természetesen bármilyen zajt, zörejt elő lehet állítani. A hangcsatornák nagyfokú variálhatósága és egymástól való függetlensége nagymértékben megkönnyíti a zenei részek kidolgozását.

A gépi kódú programozást elősegíti a 65xx processzorokra épülő mikroszámítógépekéhez képest meglepően jól kialakított megszakításrendszer.

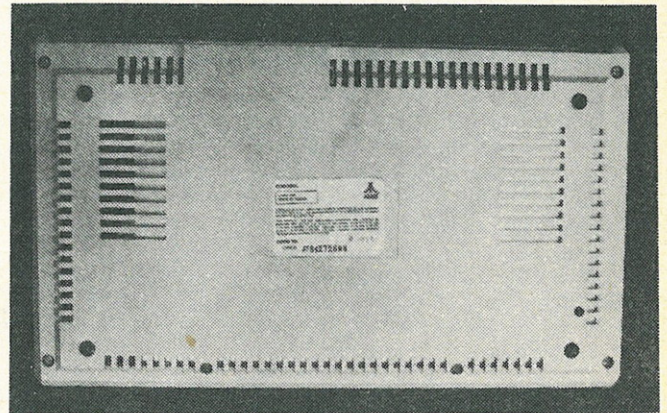
BASIC

Az ATARI 800XL hátrányai nagy részben a BASIC interpreterben vannak. Ezek közül is a legnagyobb az, hogy karaktertömböket nem ismer. Pl. a DIM A\$(200) utasítás csak egy kétszám karakter hosszúságú A\$ nevű változó definiálására szolgál, enélkül nem is használhatnánk ilyen nevű változót. A másik nagy hátrány, hogy hatványozása minden képzeletet felülmúlóan lassú. Ezenkívül sajnos nem ismeri a tangens függvényt -annak ellenére, hogy minden leírás ennek az ellenkezőjét tartalmazza. Még egy lényeges

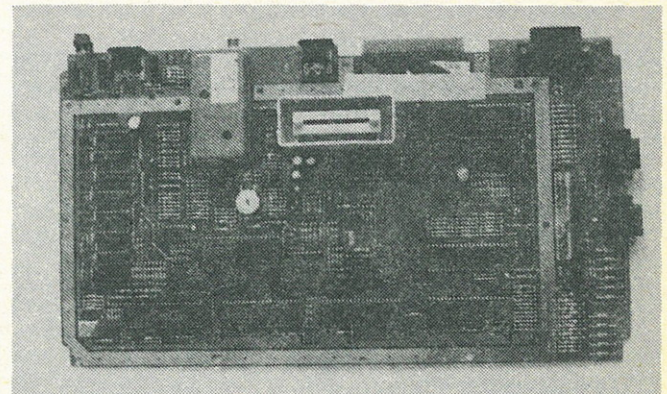
hibát kell megemlítenünk, hogy alapállapotban hibáüzenetként csak a hiba kódját írja ki. Azt már sikerült megállapítanunk hogy a ROM tartalmazza az angol nyelvű hibáüzeneteket is, de ezeket még nem sikerült előcsalnunk.

Hátrányai mellett sok előnye is van ennek a BASIC fordítónak. Könnyebbséget jelent az, hogy minden sor begépelése után szintaktikailag ellenőrzi azt, és a hibát annak helyének feltüntetésével jelzi. A C-64-től eltérő módon, de itt is rövidíthetők a BASIC parancsok. A hanggenerátorok programozására egyetlen univerzális utasítás szolgál, ez a SOUND. Ezzel a hang minden egyes paraméterét beállíthatjuk. A grafika használatát is segíti a BASIC. Így pl. kigyújthatunk, elolthatunk vagy akár ellenőrizhetünk egy pontot. A már más gépekről jól ismert DRAWTO utasítás segítségével történik a vonalhúzás, jónak mondható sebességgel. A BASIC-ből elérhető legnagyobb felbontás 320x192 pont. Ebben az üzemmódban azonban a háttérrel kívül csupán egy színt használhatunk. A legtöbb szín használatát megengedő üzemmódban 16 színnel dolgozhatunk, de a felbontás ekkor csak 80x192.

A GOTO, GOSUB, LIST, RESTORE és TRAP

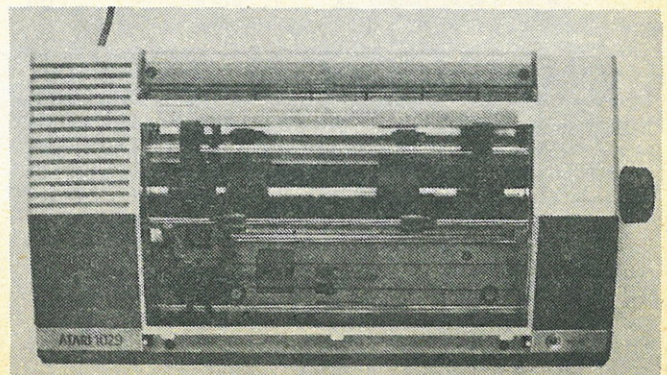


5. kép



6. kép

7. kép



utasítások után álló sorszám helyett változó is állhat. A POP utasítás lehetővé teszi, hogy egy ciklusból vagy egy szubrutinból a normálistól eltérő módon lépünk ki. A természetes alapú logaritmuson kívül a tizes alapú logaritmus használatára is lehetőség nyílik. A gép radiánban és fokokban egyaránt képes számolni. A joystickok és potméterek állapotát szintén egy-egy BASIC utasítás segítségével állapíthatjuk meg.

Ismeri az LPRINT utasítást, amivel közvetlenül csatornamegnyitás nélkül nyomtathatunk. A LIST-el kilistázhatjuk a programunkat képernyőre, nyomtatóra és lemezre egyaránt. A lemezre kilistázott programok betöltése az ENTER utasítással történik. Ezzel elérhetjük két BASIC program összekapcsolását is úgy, hogy az egyik már a gépben van, a másikat pedig ennek az utasításnak a felhasználásával rátöltjük. Vigyáznunk kell azonban arra, hogy az azonos sorszámú sorokat felülírja.

Az ADR utasítás használatával megállapíthatjuk egy karakteres változó kezdőcímét. Ez sokszor segíti a programozást. Gépikódú rutinok BASIC-ből való felhasználására szolgál az USR utasítás melyben több paramétert is átadhatunk a gépikódú résznek. A BYE parancs begépelésével léphetünk be a már említett tesztelő programba, de vigyáznunk kell, mert ezzel a BASIC program törlődik a memóriából.

A perifériák kezelése szintén könnyen megoldható néhány BASIC utasítással (LOAD, SAVE, GET, PUT, POINT, NOTE). A billentyűzet, a kazettás egység, a képernyő, a képernyőszerkesztő, a nyomtató, a lemezmeghajtók és az RS232 csatorna egyaránt elérhető ezekkel az utasításokkal.

A képernyőszerkesztő ún. full screen rendszerben dolgozik, amely szinte teljesen azonos a C-64-esével. Az ATARI-nak azonban van egy kényelmet szolgáló plusz funkciója, ez pedig a tabulátor, melyet tetszés szerint állíthatunk.

Gépikódú programok kazettáról való betöltéséhez egy kis trükköt kell ismernünk. A gép bekapcsolásakor a START gombot nyomva kell tartani, ennek hatására egy betöltő rutinra adódik át a vezérlés.

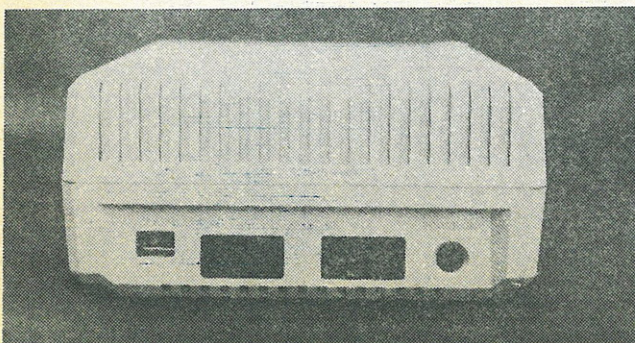
Szintén érdemes megjegyezni, hogy ha bekapcsoláskor az OPTION gombot tartjuk nyomva, akkor az operációs rendszer kikapcsolja a BASIC interpretert. Nagy terjedelmű gépikódú programok beolvasására csak így nyílik lehetőség.

Perifériák

A perifériáknak igen széles választéka áll rendelkezésünkre.

A joystickok helyére csatlakoztathatunk fénykereszt, rajzolótáblát. Ugyanitt kerülhet sor a 8 darab különálló potméter csatlakoztatására, ami azt jelenti, hogy a gépnek 8 analóg bemenete van. A billentyűzet

8. kép



helyére nemzetközi tizes billentyűzet is kerülhet. Az RS232 csatornán keresztül csatlakoztatható memóriabővítő és ramdisk is. A soros vonalon négy féle periféria lehet. Legritkábban használt a 1020-es típusjelzésű színes plotter melynek segítségével gyönyörű grafikák megjelenítésére van lehetőség. A 1029-es mátrixnyomtató (7.kép) mind karakteres, mind grafikus üzemmódban fölülmúlja a commodore gépekhez csatlakoztatható nyomtatókat (MPS sorozat, SEIKOSHA). A nyomtató a géphez hasonlóan ismeri a nemzetközi karaktereket. 50 karakter/sec sebességgel nyomtat. Nagy előnye a CBM nyomtatókkal szemben, hogy grafikus üzemmódban egy sort egyszerre nyomtat ki, minthogy pufferének nagysága ezt megengedi. Szép megoldás, hogy nemcsak a szabványos perforált printerpapírra tud nyomtatni, hanem akár különálló A/4-es lapokra is. Széles és aláhúzott karakterekkel változatosabbá tehetjük a nyomtatási képet.

Adattárolásra kazettát és mágneslemezt használhatunk. A kazettás egység száma 1010. Ha a magnó használatát választjuk, fel kell készülnünk arra, hogy így munkánk nehezekebb, az adattárolás kevésbé biztonságos lesz mint a lemezmeghajtó esetén. A géphez csak speciális magnetofon csatlakoztatható, normál közhasználatú magnót csak kisebb hardver átalakítással illeszthetünk.

Az ATARI 800XL-hez két általánosan elterjedt lemezmeghajtó létezik, a régebbi típus a 830-as, az újabb pedig a 1029-es. Mivel Magyarországon az utóbbi kerül forgalomba, ezért erről írunk bővebben. A meghajtóban egy controller chip látja el a szükséges funkciókat (írás, olvasás, inicializálás, ellenőrzés, hibaüzenet küldése). Lemezszerkezése 1040 darab 128 bájtos szektor használatát engedélyezi, így egy lemezre 130 Kbájt írható. A lemez katalógusszerkezésről itt nem írunk, mert ezt a gépben lévő, lemezről beolvasott disk operációs rendszerek eltérő módon végzik. A lemezmeghajtó nagy előnye, hogy a hátulján található kapcsolók segítségével kiválaszthatjuk az egység számát (8. kép).

A DOS

A legtöbb ilyen kategóriájú mikroszámítógép, amelyhez szorosan kapcsolódik lemezmeghajtó, valamilyen formában tartalmazza a lemezkezeléshez szükséges programrendszert. Ez vagy magába a gépbe van beépítve, vagy a lemezmeghajtó egység tartalmazza. Az ATARI 800XL kivétel ezek alól, mivel sem a gép sem a lemezmeghajtó nem tartalmazza a szükséges programot. A lemezkezelés itt a következő módon történik. A gép bekapcsolása előtt be kell kapcsolni a lemezmeghajtót, és egy speciális programot (FMS.SYS) tartalmazó lemezt kell helyezni. Ilyen például a lemezmeghajtóval együtt szállított DOS 3.0 lemez. A gép bekapcsolása után a rendszer automatikusan betölti az FMS.SYS-t, és így már dolgozhatunk a lemezegységgel.

A többi gépnél megszokott lemezkezelő parancsokat csak a DOS utasítás kiadása után használhatjuk. Ennek hatására a rendszer betölti a lemezmeghajtóban lévő DOS-t tartalmazó lemezről az operációs rendszert. Ez a rendszer nagyon sokféle funkciót lát el. Mivel többféle DOS létezik, most bemutatjuk azokat a parancsokat, amiket mindegyik ismer.

1. Lemez inicializálás. Ezzel előkészítjük a lemezt a további használatra.

2. Lemzekatalógus. A lemezen lévő fájlok nevének és nagyságának listázása képernyőre vagy nyomtatóra.

3. LOAD. Gépi kódban írt programok betöltése.

4. SAVE. A memória egy kijelölt részének lemeze mentése.

5. Átnevezés. Már lemezen lévő fájlok nevének megváltoztatása.

6. Törlés. Főlösslegessé vált fájlok lemezeiről való eltávolítása.

7. PROTECT. Fontosabb fájlok mevédésére szolgál. Ez azért lényeges, mert az ATARI felülírja az azonos néven szereplő fájlokat.

8. UNPROTECT. A védelem feloldása.

9. DUPLICATE. Egész lemez másolása.

10. COPY. Fájlok másolása a katalógus alapján.

11. Gépi kódú programok elindítása egy általunk megadott címtől.

12. Kilépés a DOS-ból.

A programozók mindig tökéletesíteni próbálják a már kész programokat. Így egyre többet tudó DOS verziókat készítettek, és ezeket megkülönböztető számmal látták el. Most felsoroljuk azokat a különleges funkciókat, amiket csak az adott DOS rendszer ismer.

DOS 1.0

Ez az elsőnek elkészült DOS. Csupán két, az általánostól eltérő dolgot tud. Az egyik, hogy egy saját formátumú lemeze önmagát kiírja (a továbbiakban WRITE DOS FILES). A másik pedig az aktuális lemez meghajtó definiálásának lehetősége.

DOS 2.0

Mint az előző 1.0-ás DOS, ez is ismeri a WRITE DOS FILES funkciót, és van még egy újabb parancsa. Ez a CREATE MEM.SAV. Ez a parancs mem.sav néven egy fájlt generál a lemezen, és minden esetben amikor DOS-t hívunk, kimentti ebbe a fájlba azt a memóriaterületet, ahová a rendszer kerül.

DOS 2.5

Ismeri a WRITE DOS FILES, és a CREATE MEM.SAV parancsokat, valamint képes szimpla sűrűséggel formázni a lemezt. Ezenkívül négy féle kiterjesztése van, melyek a következő feladatokat látják el. RAMDISK kezelése, programok 3.0-ás verzióból 2.x-be való konvertálása, aktuális lemez meghajtó kiválasztása, letörölt fájlok visszaírása, egész lemez leellenőrzése, programok önindító programmá való átalakítása.

DOS 3.0

Ismeri a MEM.SAV funkciót, képes 2.x DOS-ból 3.0-ásba konvertálni fájlokat, a felhasználó által definiált kiterjesztéseket betölteni és futtatni. Mindezekon kívül segítséget tud nyújtani önmaga használatában a parancsok részletes angol nyelvű ismertetésével, amit képernyőre és nyomtatóra egyaránt képes kiírni.

DOS 4.6

A WRITE DOS FILES és MEM.SAV parancsokon kívül talán ennek a DOS-nak van a legtöbb új funkciója. Tud általunk meghatározott mennyiségű szektorot másolni és ellenőrizni, egész lemezt ellenőrizni, lemez meghajtó sebességét mérni, valamint számokat tízesből tizenhatos, tizenhatosból tízes

számrendszerbe átalakítani. Minden lemez művelet után kilistázhatjuk az érintett hibás szektorokat képernyőre és nyomtatóra egyaránt, s ezeket a hibás szektorokat ki is nullázhatjuk.

A gép BASIC-ból a következő DOS parancsokat tudja, a XIO utasítás felhasználásával. Lemezinicializálás (szimpla és dupla sűrűséggel), fájlok átnevezése, törlése, védelme és a védelem feloldása. Ezek használata azonban jóval nehezekebb, mintha munkánkat a DOS segítségével végeznénk.

Vizsgálóprogramok

Ezek a programok a gép sebességét és pontosságát hivatottak lemérni. Mivel erre a feladatra szabványos programcsomag nem létezik, ezért két korábban használt és néhány általunk kiválasztott programot alkalmaztunk. Az első két programot az újság 1986/10-es számában található Commodore Plus/4 termékismertetőjéből oltottuk, a jobb összehasonlíthatóság végett. Ugyanebből a célból az általunk kiválasztott programok más gépeken mért futási eredményeit is közöljük (1. táblázat). Az 1. listán látható HCC NEWSLETTER-teszt a pontosságot méri. Ezt a programot egy kicsit át kellett alakítanunk, hogy funkcióját betöltse, mivel az említett cikkben a programnak egy olyan változata szerepelt, amit nem mikroszámítógépekre készítettek. Normális esetben csak D=101 értéknél ugrik ki a belső ciklusból a program, ami a Plus/4-essel ellentétben az ATARI 800XL-en így is történik. A program futási ideje azonban hosszabb ezen a gépen (14.06 mp, míg a Plus/4-en 08.10 mp).

A következő program, amely a 2. listán

1 táblázat

	3. LISTA	4. LISTA	5. LISTA	7. LISTA	7. LISTA
C-64	1.60	4.00	5.50		
PLUS/4	1.60	43.00	4.79	32.00	25.90
ATARI	2.42 / 1.91	30.79	22.63 / 16.74	34.13 / 22.65	49.16 / 36.23
DRAGON	1.17 / 593			6.55 / 4.33 *	7.55 / 3.85

* LINE III DRAW utasítással
AZ EREDMÉNYEK MÁSODPERCENBEN

2 táblázat

GÉP ÉS BASIC	IDŐ (s)
C-64	255
APPLE II, APPLESOFT	230
IBM PC, BASICA	190
IBM PC, ZBASICA	1
PLUS/4	325
DRAGON	280
ATARI 800XL	211

A PRIMSZÁMTESZT EREDMÉNYÉNEK TÁBLÁZATA

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

1. lista

```
5 POKE 559,0
10 SI=5001
20 DIM FL(5001)
30 PRINT "ONLY 1 ITERATION"
40 CO=0
50 FOR I=0 TO SI
60 FL(I)=1
70 NEXT I
80 FOR I=0 TO SI
90 IF FL(I)=0 THEN 170
100 PR=I+1+3
110 K=I+PR
120 IF K>SI THEN 160
130 FL(K)=0
140 K=K+PR
150 GOTO 120
```

```
150 CO=CO+1
170 NEXT I
180 PRINT CO,"PRIMES"
200 POKE 559,34
```

2. lista

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

3. lista

```
10 FOR I=1 TO 1000
20 PRINT "ATARI 800XL"
30 NEXT I
```

4. lista

```
10 FOR I=0 TO 100
20 S=I*2
30 NEXT I
```

5. lista

```
10 GRAPHICS 8+15
20 COLOR 1
30 FOR I=0 TO 319
40 PLOT I,0
50 DRAWTO I,191
60 NEXT I
```

6. lista

7. lista

```
10 GRAPHICS 8+15
20 COLOR 1
30 FOR I=1 TO 1000
40 A=INT(RND*(319)+1)
50 B=INT(RND*(191)+1)
60 PLOT A,B
70 NEXT I
```


látható, szintén átalakításra szorult, mivel a gép nem volt képes egy hétezer elemű tömb dimenzionálására. Ezt a programot a BYTE című számítástechnikai szaklap közölte először, néhány futási eredménnyel együtt. Ezt mi is közöljük a Plus/4-es és az Atari-n mért ATARI 800XL eredményekkel kiegészítve (2. táblázat). Vegyük figyelembe, hogy az Atari-n csak egy ötezer elemű tömbből van szó, míg a többi gépen ez a szám hétezer.

A következő pár programot mi választottuk ki, mivel úgy gondoltuk, hogy ezek a legalkalmasabbak a gép tulajdonságainak bemutatására. A legegyszerűbb (3. lista) egy üres ciklus. Futási ideje 02.37 mp, kikapcsolt képernyővel 01.78 mp.

A 4. listán látható program csak egy PRINT utasítással bővült az előzőhöz képest, futási ideje azonban lényegesen megnőtt (30.76 mp). Ebből is látszik, hogy mint általában az összes mikroszámítógép esetében, itt is sok időt vesz igénybe a PRINT utasítás végrehajtása.

Az 5. listán látható program segítségével a már említett hatványozási hibát szeretnénk megvilágítani. Mindenki beláthatja, hogy a 22.63 mp-es futási idő lehetetlenül hosszú. A képernyő-elsötétítésével is csak 16.74 mp-re sikerült leshorizálni a futási időt.

Az utolsó két programmal a gép grafikáját kívántuk tesztelni. Az egyik (6. lista) a legnagyobb felbontású grafikus képernyőt vonalanként tölti fel. Ennek futási ideje 34.13 mp, míg elsötétített képernyő esetén 25.56 mp, ami meglehetősen jó eredmény. A másik (7. lista) ezer darab pontot helyez el véletlenszerűen ugyanebben a grafikus üzemmódban. Ehhez 36.23 mp-et vett igénybe a gép.

Véleményünk szerint jó lenne egy általánosan használható tesztelő

programcsomag kidolgozása, a gépek jobb összehasonlíthatóságának érdekében.

Összefoglalás

Már több mint fél éve dolgozunk tíz darab ilyen típusú géppel, ezért úgy véljük elegendő információ áll rendelkezésünkre a termékismertető megírásához. Ezen idő alatt komolyabb problémánk nem volt, annak ellenére hogy minden konfiguráció (lemez meghajtó is) naponta 10-12 órát üzemelt megszakítás nélkül. Szerintünk egy igen sok igénybevételt elviselő géppel állunk szemben, amely Magyarországon sajnos csak kevéssé elterjedt. Mivel korábban más géptípusokkal dolgoztunk (Sinclair gépek, C-64, C-16, C-128, Primó, HT 1080-Z, APPLE II, IBM PC), nyugodt szívvel állíthatjuk, hogy az ATARI 800XL programozása kényelmes.

A gépet perifériákkal együtt Magyarországon a SKALA-COOP COMPUTER-S részlege forgalmazza.

Szervizelését a NOVOTRADE-FOTOELEKTRONIK Magyar utcai szerveze végzi.

Már működik Budapesten a NJSZT HCC keretén belül az ATARI szekció, amely minden hónap első keddjén sok szeretettel várja az ATARI barátokat. Itt minden kezdő segítséget kaphat az induláshoz.

Ajánljuk a gépet azoknak, akik a számítástechnikával még csak most ismerkednek, és azoknak is akik bizonyos ismeretekkel rendelkeznek. A gép alkalmas otthoni szórakozásra, játékokra, tanulásra és egyszerűbb számítástechnikai feladatok megoldására.

Mindent összevetve, ezt a gépet leginkább a C-64-hez lehet hasonlítani, de sok tekintetben annál többet nyújt.

Romvári Gábor Sós Gábor

MUNIFORM

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

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

A tartalomleíró szövegeket permutáltuk, a szövegváltozatokat pedig alfabetikusan rendeztük.

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

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

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

Európa legjobb Commodore újságjáról, a 64'er Magazinról másféle információszolgáltatás is kérhető. Szolnoki Béla vállalja, hogy havi 20 Ft-ért (+ a postaköltség) elküldi a folyóirat tartalomjegyzékeinek megjegyzésekkel is ellátott fordítását - a lap 1986. 07. havi számától kezdődően bármelyikről, megbízás esetén

folyamatosan. Ennek alapján a megrendelő kiválaszthatja, hogy számára melyik cikkek fontosak, és ezek fordítását is megkaphatja oldalanként 5,-Ft-ért (+ a postaköltség), ha ír a következő címre:

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

PROGRAMLISTA

```
64er||beviteli utmutato||data-sor elle
norzes||checksummer 64 v3)
->64er/86.03-35/1
```

PROGRAMLISTA

```
adatbazis szervezes(hashing)||szort t
arolas alapelve||peldak
->happ/86.03-148/4
```

PROGRAMLISTA

```
apple ii||commodore 64||ram particiona
las||multimemory)||programbetoltes
stria Nyerges M.0309
->cut/86.03-93/3
```

PROGRAMLISTA

```
apple ii||formatalt listanyomtatás
->hc/86.03-87/2
```

PROGRAMLISTA

```
apple ii||jatekprogram||switchbox)
->cute/86.03-47/2
```


PROGRAMLISTA
apple ii lemezhiba javitas
->pers/86.05-82/1

PROGRAMLISTA
atari 400/800 xl lemezegység (atari 1050) happy-dos ii+d operacios rendszer ->happ/86.03-91/8

PROGRAMLISTA
atari 400/800/xl/xle grafikus programozas animacio ->cute/86.03-85/6

PROGRAMLISTA
atari 400/800/xl/xle jatekprogram (switchbox) ->cute/86.03-45/3

PROGRAMLISTA
atari 400/800/xl/xle muveletgyorsitas bepegelési segedprogram ->cute/86.03-107/3

PROGRAMLISTA
atari 520 st jatekprogram (switchbox) ->cute/86.03-51/2

PROGRAMLISTA
atari 800 xl (mini-dos) lemezegység-vezeres filekezes ->hc/86.03-56/3

PROGRAMLISTA
atari 800 xl/xle lemezblokk kijeloles ->cute/86.03-102/3

PROGRAMLISTA
atari st muveletgyorsitas lemezkezes ->pers/86.05-87/3

PROGRAMLISTA
atari hc-main home-computer beviteli utmutato data-sor osszegellenorzes ->hc/86.03-55/2

PROGRAMLISTA
c nyelv cikksorozat alprogram behivas ->pers/86.06-64/8

PROGRAMLISTA
c nyelv cikksorozat programozasi fogasok szintaxis grafok (peldak) ->pers/86.05-94/4

PROGRAMLISTA
comodore 128 (interface 5120) centrionics illesztés hardver nélkül ->64er/86.03-84/2

PROGRAMLISTA
comodore 128 (spriteseer 2.0) sprite-editor ->comi/86.01-90/3

PROGRAMLISTA
comodore 128 grafikus programozas ->coho/86.03-44/2

PROGRAMLISTA
comodore 128 muveszi grafika (softpaint) ->happ/86.03-86/3

PROGRAMLISTA
comodore 16;128;plus/4 (datastater) basic data-sorok generalasa gepi ru tinbol ->comi/86.01-61/3

PROGRAMLISTA
comodore 16;64;128;vc20;plus/4 comp ute! beviteli utmutato (proofreader) ->cute/86.03-91/2

PROGRAMLISTA
comodore 64 (control-q) reverz kara karakterek iktatasa programozaskor ->comi/86.01-52/1

PROGRAMLISTA
comodore 64 (husky-basic) basic bovites (grafika/hang) ->happ/86.03-77/7

PROGRAMLISTA
comodore 64 (view picture) szines kep beillesztes basic programokba ->64er/86.03-91/1

PROGRAMLISTA
comodore 64 (boulderdash) cimkep gorgetesenek utanzasa ->64er/86.03-80/3

PROGRAMLISTA
comodore 64 alfas karakterek numerikus alakitasa data-bevitelhez ->run/86.03-50/4

PROGRAMLISTA
comodore 64 basic bovites nyomtato (1520) vezereshez ->chip/86.03-194/2

PROGRAMLISTA
comodore 64 futo szalagfelirat generalasa ->64er/86.03-83/1

PROGRAMLISTA
comodore 64 gepnyelvu program lemezre mentese basic segedprogram ->cute/86.03-81/1

PROGRAMLISTA
comodore 64 grafikus programozas (k-oala-painter) hasznalata (print shop) Kiegészites ->64er/86.03-75/4

PROGRAMLISTA
comodore 64 grafikus programozas random grafok generalasa ->hc/86.03-84/2

PROGRAMLISTA
comodore 64 jatekprogram (space snakes);2.resz ->coho/86.03-50/1

PROGRAMLISTA
comodore 64 jatekprogram (tennis) ->hc/86.03-43/7

PROGRAMLISTA
comodore 64 maszkgeneralas ablaktechnika alkalmazasi utmutato ->64er/86.03-71/2

PROGRAMLISTA
comodore 64 matematika fuggvenyabrazolas (Kudiplo 64) ->64er/86.03-57/5

PROGRAMLISTA
comodore 64 muveszi grafika szines nyomtatás koalapainterhez kapcsolodo hasznalat ->hc/86.03-70/5

PROGRAMLISTA
comodore 64 szovegszerkesztes gepi nyelven (run script 64) alkalmazasi utmutato ->run/86.03-40/11

PROGRAMLISTA
comodore 64;128 jatekprogram (copter-fight) ->happ/86.03-69/5

PROGRAMLISTA
comodore 64;128 jatekprogram (switchbox) ->cute/86.03-43/3

PROGRAMLISTA
comodore 64;128 oktatóprogram (csillagaszat) (unsere sonnen system) ->happ/86.03-84/2

PROGRAMLISTA
comodore microcomputers beviteli utmutato ->comi/86.01-123/3

PROGRAMLISTA
cp/m illesztés (lemezformatum) lemeziparameter modositas alkalmazasi peldak (osborne) ->pers/86.05-74/7

PROGRAMLISTA
programozas cikksorozat comodore to Ken-Konverzio modellcserehez ->cute/86.03-82/3

PROGRAMLISTA
rejtjelezés adatvedelem comodore 64 (scrambler) ->comi/86.01-54/3

PROGRAMLISTA
run beviteli utmutato (ml perfect typist) data-sor ellenorzes ->run/86.03-97/3

PROGRAMLISTA
sinclair spectrum (data creator) data-sor generalas kijeloljeto tar tar talombol ->zx/86.01-99/1

PROGRAMLISTA
sinclair spectrum (machine code trace) ->zx/86.01-45/3

PROGRAMLISTA
sinclair spectrum (adattallomany Kezels) (multi-file) ->zx/86.01-82/4

PROGRAMLISTA
sinclair spectrum cikksorozat grafikus programozas ->zx/86.01-32/4

PROGRAMLISTA
sinclair spectrum filekezes modulok (microdrive file analyzer) (file copier) (file sorter) ->zx/86.01-49/5

PROGRAMLISTA
sinclair spectrum jatekprogram (alien) ->zx/86.01-18/3

PROGRAMLISTA
sinclair spectrum jatekprogram (astro balloons) ->zx/86.01-59/3

PROGRAMLISTA
sinclair spectrum jatekprogram (battle fleet orion) ->ysin/86.03-46/4

PROGRAMLISTA
sinclair spectrum jatekprogram (bounce down) ->zx/86.01-114/2

PROGRAMLISTA
sinclair spectrum jatekprogram (coin drop) ->zx/86.01-102/4

PROGRAMLISTA
sinclair spectrum jatekprogram (dungeon danny) ->sync/86.03-102/2

PROGRAMLISTA
sinclair spectrum jatekprogram (goblin dungeon) ->zx/86.01-28/4

PROGRAMLISTA
sinclair spectrum jatekprogram (megabert) ->ysin/86.03-49/3

PROGRAMLISTA
sinclair spectrum jatekprogram (mission 12) ->sync/86.03-99/2

PROGRAMLISTA
sinclair spectrum jatekprogram (spiders) ->zx/86.01-22/5

PROGRAMLISTA
sinclair spectrum matematika oktatóprogram (train race) ->zx/86.01-56/3

PROGRAMLISTA
sinclair spectrum modem (vtx5000) basic bovites ->zx/86.01-74/2

PROGRAMLISTA
sinclair spectrum szovegkezes (keyword ii) bovites ->zx/86.01-76/3

PROGRAMLISTA
sinclair spectrum tablázat szamitas/generalas (tabcalc) ->zx/86.01-86/8

PROGRAMLISTA
sinclair spectrum zene grafikus hang megjelenites ->hc/86.03-59/3

PROGRAMLISTA
sinclair zx81 haztartasi konyveles ->zx/86.01-79/3

PROGRAMLISTA
sinclair zx81 jatekprogram (alphanumeric) ->zx/86.01-116/2

PROGRAMLISTA
sinclair zx81 jatekprogram (letter puzzle) ->zx/86.01-118/2

PROGRAMLISTA
sinclair zx81 jatekprogram (starfighter) ->zx/86.01-54/2

PROGRAMLISTA
speedcalc atari 400/800/xl/xle alkalmazasi utmutato ->cute/86.03-65/13

PROGRAMLISTA
statistika atari comodore grafika (szemlelteto) (msx) (hc superfact) ->hc/86.03-86/5

PROGRAMLISTA
termeke ismertetes apple ii eger ->cute/86.03-99/3

PROGRAMLISTA
terminal uzemmod (unix) cp/m illesztő program eof-felismeres ->pers/86.06-50/4

PROGRAMLISTA
turbo-pascal (aramkortervezes) szimulacio (szabalyozo (pid)) ->pers/86.06-114/2

PROGRAMLISTA
turbo-pascal matrix (otdimenzios) (adattbasis struktura) editorprogram ->pers/86.06-55/4

PROGRAMLISTA
zene atari st zongora szimulacio ->cute/86.03-110/3

PROGRAMLISTA
zene comodore 64 hanghatasok keltese ->cute/86.03-96/3

A sakktablán a legkisebb egység a gyalog, mégis nagy szerepet játszik az ütközetben. A tisztektől eltérően csupán előre megy, de előnyomulással teret nyer a tisztek számára. Fontos feladat hárul rá a centrummezők elfoglalásában és az ellenfél tiszteinek visszaszorításában. Gyengesége egyben erősségét is jelenti: amelyik mező az ütközőbe esik, arra nem léphet ellenséges tiszte.

A gyalogok egy vagy két mezőt ellenőrizhetnek. Ha egy gyalogot az őrhelyéről eltávolítunk, akkor gyenge pontok, esetleg ezek egész sora keletkezhet. De e pontok gyengeségét csakis a tisztek sikeres közreműködésével lehet kihasználni.

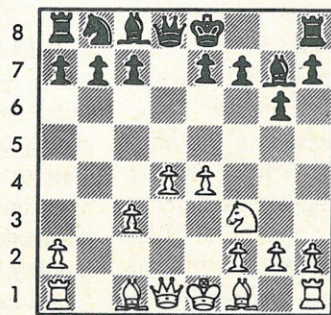
A játék folyamán nagy jelentőségük van a gyalogáttöréseknek, amelyeknek célja, hogy nyílt vonalakra tegyünk szert, s tiszteinkkel behatoljunk az ellenfél táborába. Ilyen esetekben kiemelt szerep jut a gyalogságnak.

A mai gyakorlatban mind nagyobb figyelmet fordítunk a gyalogok helyzetére és mozgékonyására, a gyalogtámadás lehetőségére a centrumban és a szárnyakon. A mozgékony gyalogcentrum, amelyet tisztek támasztanak alá, korlátozza az ellenfél tiszteinek mozgékonyágát, kettéosztja az ellenséges haderőt, és térelőnyt szerezhethet. A centrumgyalogok előnyomulása — ha visszaszorítják az ellenfél tiszteit — nagyon veszélyes lehet. Különösen akkor, ha az előrehaladó gyalog beékelődik az ellenfél táborába, és megosztva annak haderejét, megfosztja az együttműködés lehetőségétől.

A gyalogpár nem mindig előnyös a centrumban, olykor támadási célpontul szolgál. Ez látható például a Grünfeld-védelem egy-egy változatában, ahol sötét megengedi világsónak a gyalogcentrum képzését,

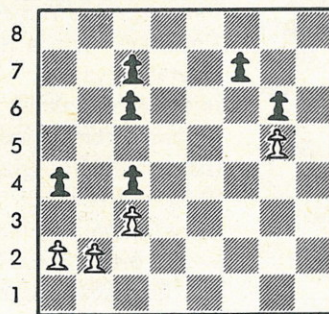
BITEK ÉS FIGURÁK

Gyalogszerkezet



a b c d e f g h

1. ábra



a b c d e f g h

2. ábra

de azután nyomást gyakorol rá a nagylón és a félig nyílt d-vonalon a következőképpen (lásd az 1. ábrát):

1. d2—d4 Hg8—f6
2. c2—c4 g7—g6
3. Hb1—c3 d7—d5
4. c4xd5 Hf6xd5
5. e2—e4 Hd5xc3
6. b2xc3 Ff8—g7

A stratégia egyik legfontosabb problémája a centrumban levő izolált gyalog, amely nagy cselekvési lehetőséget jelent, fontos pontokat ellenőriz (például a d4 gyalog a c5 és e5 pontot). Akié ez a gyalog, annak van nagyobb esélye a támadásra. De vannak hátrányai is!

Az izolált gyalog bizonyos mértékig gyenge, és az ellenséges haderő támadhatja. A középjátékban nagyon fontos körülmény, hogy az izolált gyalog előtt levő mezőt egy tiszte elfoglalhatja és ezáltal leblokkálhatja-e. Az izolált gyalog

gyengesége a tisztek cseréje után egyre érezhetőbbé válik.

A félig nyílt vonalakon visszamaradt gyalog a legérezhetőbb gyengeségek egyike. Gyalog nem védi, az ellenfél nehéz ütegeinek szórásába esik, állandó gondot jelent, nem ritkán el is vész.

Mindezeket a szempontokat célszerű figyelembe venni az értékelőfüggvényben. Igaz, ez nagyon nehéz feladat, ezért a sakkprogramoknál általában kerülni szokták az ilyen finomításokat; többnyire csak a gyalogszerkezet épségét pontozzák, ami sokkal egyszerűbb.

A gyalog a sakk lelke. Ezért nagyon fontos az ép gyalogszerkezet kialakítása. Ehhez azonban tudnunk kell, hogy mit nevezünk épnek és mi az, ami a legtöbbször előnytelen. A nem ép gyalogszerkezet három jellemzője: a dupla gyalog, az izolált gyalog és a hátramaradt gyalog. A 2. ábrán

mindegyikre mutatunk példát.

Dupla, tripla stb. gyalognak nevezzük azt a szerkezetet, ha két vagy több azonos színű gyalog ugyanazon a vonalon helyezkedik el, mint például az ábrán a c4, c6 és c7 gyalogok. A dupla gyalog több szempontból is gyengeséget jelent. Ha izolálttá válnak, akkor különösen a végjátékban könnyen sebezhetőek. Egy bástya mindkettőt egyszerűen fel tudja számolni. Az is jelentősen csökkenti erejüket, hogy nem tudják az ellenféllel szemben ugyanazt az erőt kifejteni, mint amikor két összekötött gyalog előnyomul.

Izolált gyalognak nevezzük azt a konstrukciót, ha a szomszédos vonalokon nincs ugyanolyan színű gyalog; például az ábrán az a4 gyalog. Az izolált gyalog a szomszédos védőgyalogok nélkül gyenge. Ezért az ellenfél célpontként használja és könnyen megtámadhatja. Az izolált gyalogot csak a tiszteji tudják megvédeni, ezzel viszont azokat sokkal értékesebb helyről vonja el, így passzívakká válnak.

A hátramaradt gyalogot esetleges előrehaladása esetén sem tudja a szomszédos gyalog megvédeni. Példa erre az ábrán az f7 gyalog.

Az intelligens sakkprogramok figyelembe veszik a gyalogszerkezet gyengeségeit: ha hátrányos gyalogszerkezet jön létre, akkor a program büntetőpontot von le az állásértékből.

A végjátékban viszont jutalom jár szabad gyalog esetén, sőt, ha ezt egy másik gyalog védi, akkor a jutalom növekszik. Ugyancsak nő a jutalom, ha a gyalog közel van egy átváltozási mezőhöz. A pontos értékeket az értékelőfüggvény többi komponenséhez kell viszonyítani, és végül is csak kísérletezéssel lehet megállapítani.

KOVÁCS P. ATTILA

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

cFORTH-ot

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

Megrendelhető csak iskolák részére a Tudományszervezési és Informatikai Intézetnél, Budapest, Pf. 454. 1372. Ügyintéző: Bánó György, tel.: 664-011/2657.

Január elején a szokottnál kevesebb levelet kaptunk, úgy látszik az ünnepek alatt olvasóink inkább ünnepeltek, mint leveleket írtak. Az újságok számára pedig a legfontosabb dolog az élő és „interaktív” kapcsolat az olvasókkal. Nagyon szívesen olvasnánk például arról, hogyan működnek a klubok, mennyire fejlődik a számítógépes (hv és szv) amatőrmozgalom. Használják-e a számítógépeket az iskolákban, esetleg még mindig vannak olyan iskolák, ahol csak szobadisz a gép és nem a tanítás és az oktatás eszköze. Szeretnék hallani és hírt adni az eredményekről és a problémákról is. Elmúltak az ünnepek — kérem — lehet levelet írni!

Ludván Zsolt, Pécs,

Dr. Veress E. u. 9. III. 10. 7633

Nagy kéréssel fordulok Önökhöz. A μM azon számaikat szeretném megszerezni, melyekben ZX-81-hez közölnek programokat, tanácsokat. Az antikváriumi keresgélés sokkal könnyebb lenne, ha tudnám, melyek ezek.

Kérem, amennyiben lehetséges, küldjék el a kérdéses lapok megjelenési dátumait. Igen sokat segítenének vele. Cserébe nem fogom követelni, hogy a lap ezentúl is közöljön ZX-81-gyel kapcsolatos írásokat.

A μM eddigi számainak tartalomjegyzékét rövidesen megjelentetjük (talán, amikor az olvasók a lap ezen számát kézhez veszik, a tartalomjegyzéket már meg is kapták). Nem bánjuk, ha ZX-81-gyel kapcsolatos írások közlését kéri, de kérését sajnos nem tudjuk teljesíteni, t. i. ezzel a géppel csak egészen egyszerű feladatokat lehet megoldani, amelyek olvasóink nagyon nagy részét már egyszerűen nem érdeklik.

Sajnáljuk.

Turay Gábor, Budapest,

Kőrösi Csoma út 35—37. 1105

Nem olyan régóta vagyok rendszeres olvasója lapjuknak, de ebben az évben sikerült minden számot megszerezni.

Alapjában véve tetszik a lap, és az egyik legjobb magyarországi lapnak tartom. A lapban én nem a hirdetések kifogásolom, hiszen egyrészt ezek tartják el a lapot, másrészt a magazinjelleghez hozzátartoznak a hirdetések is. Egy dolgot azonban

nem tudok megérteni: miért nem közlik a reklámozott termékek árát? A tartalomnál maradván: miért tűnt el a „ZÖLDSÉGESKERT”?

A formátumhoz is lenne egy észrevétel. Alapvető figyelmetlenségnek tartom, hogy az alig olvasható printelt listákat (melyek olvashatatlanságát növeli a betűk mérete) két alapon nyomják!!!

Ha már a programoknál tartunk, bevalom, nekem is van egy 48 k-s Spectrumom. Én is keveslem a Spectrum programokat. Tisztában vagyok vele, hogy bimbózó számítógépiparunknak kell a reklám. De az utóbbi időben minden számban 4-5 oldal (ha nem több) foglalkozik a PTA 4000-rel. Nem irigylem tőle az ingyenreklámot, de Spectrumból legalább egy „nullával” több van az országban. Mindezeket kívül a lapnak nem kész programokat, hanem ötleteket, trükköket kellene közölnie. Néha közül is, pl. nagyon örültem a „Spectrum rendszerváltozói” című cikknek. Ötleteket kell adni a programozáshoz! A jó lapnak műzsának kell lennie, amely olvasás közben „megcsókolja” az olvasót, és saját program megalkotására ösztönzi.

Egyébként a lap sokszínűségével meg vagyok elégedve. Mindenki talál magának valót, az idő múltával egyre többet. (Ezt úgy értem, hogy ahogy bővül az olvasó „tudománya”, úgy ért meg egyre több és több cikket.)

Mint már említettem, van egy Spectrumom. Ezért ha valakinek Spectrum programokhoz cseretársra lenne szüksége, szívesen állok rendelkezésére.

Végezetül egy kérdés: szeretnék látni a lapban egy TURBO LOAD és egy TURBO SAVE rutint, ha lehet magyarázattal. (Mi határozza meg a sebességet stb.) Természetesen a rutint Spectrumhoz gondoltam, lehetőleg assembly-ben.

Azt bizonyára minden olvasónk észrevette, hogy a hirdető cégek általában nem adnak árakat, ennek az oka, hogy a lap átfutási ideje (a kézirat leadásától a megjelenésig) még mindig nagyon hosszú, 2-3 hónap. Arra törekszünk, hogy a kiadó segítségével ezt az időt lerövidítsük. Ha sikerül, akkor egészen biztosan — az elavulás veszélye nélkül — a hirdetésekben megjelennek az árak is.

Zöldsegeskert. — Az olvasók egy részének nem tetszett a cikk stílusa. Nekem se.

Ismét a Spectrumról. Talán nem járok messze az igazságtól, ha feltételezem, hogy a Spectrumok azoknál vannak, akik a lapot olvassák és nem azoknál, akik írják, t. i. ha fordítva lenne, akkor sokkal több cikket és kevesebb reklamációt kapnánk.

Ami a dicséretet illeti, pirulva olvastuk, de jólesett, ha nem is hisszük el, hogy így van. (De szeretnénk!)

„Gyakorló” — jelige

Kezdő kis „számítástechnikus” vagyok, de számítógép hiányában nem tudok gyakorolni. Kérem segítségét számítógép-tulajdonos keresésében, aki használt HT 1080—Z-t vagy más olcsó gépet eladna, vagy ha ilyen jelenleg nincs, akkor új gépet hol és mennyiért lehet kapni.

Ha „Az olvasó írja” rovatban kérésemet szerepeltetik, akkor kérem, hogy nevem helyett — gyakorló — jeligét használjanak, és mivel nem járatom magazinukat, nem tudom mindig megszerezni, ezért írja meg, mikor jelentetik meg.

A kérését megjelentettük; én nem hiszem, hogy HT gépet tud vásárolni, ezeket a számítógépeket az iskolák kapták, akik — ha jól tudom — egyelőre még használják és nem adják el ezeket a gépeket. A μM — mindenféle szerződésünk szerint — minden hó első hetében kellene, hogy megjelenjék. A decemberi „Százoldalas” egy kicsit felborította a terveket, nagyobb feladat volt a nyomdának, mint gondoltuk, és így a lap kb. 2 hét késéssel jelent meg. Ezt a választ január közepén írom, amikor a 87/1. szám még nem jelent meg, tehát a késést még nem tudtuk leküzdeni. Talán ez a szám már pontos lesz.

ifj. Schäffer András, Paks,

Kodály Z. u. 1. 3/7. 7030

Első osztályos számítástechnika szakos tanuló, a lap rendszeres olvasója vagyok. A százoldalas mikromagazinban érdekes cikket olvastam a BBS rendszerekről. Kérdésem a következő lenne, megvalósítható lenne-e itthon egy ehhez hasonló rendszer? A különböző gépekhez az emulátor programot és a modemet bárki megvehetné a boltban. Magyarországon egy ilyen rendszer segítené a számítástechnika elterjedését.

A véleményünk egyezik.

Csupor Roland, Pécs,

Egri Gy. u. 34. 7632

Közreadom ezt a kicsi programot.

A 0-dik sor törli a képernyőt, és szint ad neki 1—7 kirajzol egy sakkbábút, a daták segítségével.

Én most egy problémát szeretnék közölni. A probléma: a program 7. sorában kirajzolja a sakkbábút, ez egy paraszt, és ha ezt

a prg-t tovább folytatjuk és kirajzoltatunk egy lovat, akkor a paraszt is átváltozik lóvá.

No most én a VC 20-as gazdáktól várok segítséget. (Az a betű, a sakkbábunak a bal felső oldalát rajzolja, a b a jobb felső, a c bal alsó és a d jobb alsó.

Én megragadom az alkalmat, hogy köszönetet mondjak Hajnal Csabának, amiért az 1986/11-6 Mikroszámítógépes magazinban közzétette a folttervező nevű prg-t, ugyanis ebből szerkesztettem meg ezt a prg-t.

És ha lesz rá alkalom, akkor szeretnék prg-t cserélni.

Egy kicsit bolond ügynek tűnik, hogy közlünk egy programleírást és a programot nem. Sajnos olvasónk kézzel írt listát küldött, amelynek a külalakja és olvashatósága — enyhén szólva — kifogásolható, nem tudtuk megfejteni, pedig szívesen helyet adtunk volna a kérésének. Sajnos.

Nagyon tetszett viszont a levél befejezése, azt hiszem, hogy a μM történetében először fordul elő, hogy valaki — olvasói levélben — egy másik programozónak köszönetet mond. Ebből még a profik is tanulhatnak.

Stadler Gábor, Veszprém,

Stadion út 24/b 8200

A rejtvénypályázatuk 3. feladatának a megoldását írom le. De mielőtt ezt megtenném, hadd mondjam el, hogy az újságjukat 1986. december 21-én kaptam kezembe. (Egyébként amint megérkezik a városba, az elsők között vagyok, akik kézbe veszik a lapot). Amint föllapoztam a 3. feladatot, egyből szemet szúrt, hogy már tegnap, azaz 20-án meg kellett volna érkezni a megfejtésnek.

Az esetből tanultunk, a pót — pót-kötél is elszakadhat. Legközelebb nem tűzünk ki ilyen rövid határidőt.

Dékány Katalin, Casa Pionierilor,

Salonta 3650 R. S. Románia

Általános iskolába járok, VIII. osztályos tanuló vagyok. A helyi úttörőház informatika körére járok; lapuk régi olvasója vagyok.

A napokban a következő jelenséggel találkoztunk. A ZX Spectrum programozása közben vettük észre:

10 PRINT "AAKÁRMI";

20 GO TO 10

RUN parancs után a program fut, majd megjelenik a „scroll” kérdés (a PRINT után bármilyen karakterlánc következhet).

Ha ekkor megnyomjuk a CAPS SHIFT és SIMBOL SHIFT-et egyszerre, RUN(E)

jelenik meg. ENTER után a képernyő teleíródik töredékes utasításokkal. Ugyanez játszódik le GO TO-val indítva, csak GO TO (E) jelenik meg.

Szívesen adom közre a levelét, olvasóinktól is várjuk a megoldást.

Dr. Rátay Csaba, Budapest

XI., Zólyomi köz 14. 1112

Szeretném Sátoraljaújhelyen és a környező falvakban elősegíteni a távoktatás lehetőségeinek a kialakítását.

A Hegyköz és Bodroghöz két olyan tája Magyarországnak, amelyet különösen súlyosan érint az iskolák körzetesítése, a gyerekek állandó utaztatása. A régi és az új veszedelmek, a családokat károsító tényezők kivédésére családgondozó hálózatot is szervezünk, most folyik a társadalmi önkéntesek képzése.

A távoktatás lehetőségei pompásan illeszkednek a mi elképzeléseinkbe, és a műszaki feltételek adottak.

A művelődési ház technikai személyzete

készen áll a városi tv-hálózat beindítására, melyhez szerintünk minden feltétel adott. Ez szolgálhatná a műszaki feltételeit az együttműködésünknek — meg talán az értelem és segíteni akarás — ami az ott élőkben található.

Ezt a levelet dr. Rátay Csaba voltaképpen nem a μM -nek, hanem nekem írta, azért közlöm, hogy jelezzem, az informatikai távoktatás napirendre került, a közeljövőben szeretnénk egy minden megyére kiterjedő távoktatási hálózatot létrehozni. A következő számaink valamelyikében a megvalósítási tervezett távoktatási rendszerről részletes tájékoztatást közlünk. Az a kérésem, hogy akik (intézmények és szakemberek is) úgy érzik, hogy szívesen részt vennének egy ilyen távoktatási—tanulási rendszer megszervezésében, levélben jelezzék szándékukat.

Köszönöm a leveleket és egyben minden levelezőnkől elnézést is kérek, ha a levelét nem tudom közzétenni, vagy ha az írását csak néhány hónap késéssel közöljük.

KOVÁCS GYŐZŐ

*** HardSoft * HardSoft ***

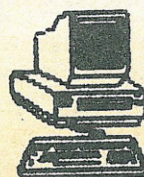
OPTIMER

**Számítás-
technikai GMK.**

**7624 PÉCS,
Jakabhegyi út. 2.**



**TERMINÁL
EMULÁCIÓ
ÉS
FILE-
TRANSFER
ILLESZTÉS**



**IBM PC/XT/AT
VT 16
PROPER 16
VT 20/IV
TPA
VPPC
VPC
CBM 64
CBM 128
és más mikro-
rogépekhez!**

OPTIMER

ÜRHAJÓZÁS

A programban egy űrhajóval kell a meteoritokkal teli úton eljutni a bázisra. A program újdonságai:

- a) kétirányú (bal-jobb) finom scroll
- b) a kép invertálása
- c) karakteres scroll balra

A játék irányítása:

„A”=le „Q”=fel „Enter”=lövés

A pálya 500 km-es, amelyen az 50 lövedékkel takarékoskodni kell. Ha elértünk a bázisig, vagy az öt élet elfogyott, kiíródik a pontszám, és a gép megkérdezi, hogy kérünk-e új játékot.

A program felépítése:

20—30: értékadás, kiírások.

40—60: a billentyűk lekérdezése.

65—80: az ütközés vizsgálata, scroll.

700—740: a bázisra érkezés.

800—810: lövés.

900—980: a játék vége, új játék kérése.

1000—1040: a gépi kódú programrészek betöltése.

```

10 REM URHAJÓZÁS
20 BORDER 4: INK 7: CLS : PRINT
AT 10,0: INK 6: "A":PLOT 0,166: D
RAW 255,0: PLOT 0,4: DRAW 255,0:
PRINT INK 7:AT 0,0:"PONTSZAM:0
ELETEK:4 UT:0 KM"
30 LET X=10: LET UT=0: LET PONT
=0: LET ELET=4: LET SOK=25: LET T
UZ=50
40 LET I$=INKEY$: LET X=X+(I$="
A" AND X<20)
50 LET X=X-((I$="Q" AND X>2)
60 IF I$=CHR$ 13 AND TUZ>0 THEN
LET TUZ=TUZ-1: GO SUB 800
65 PRINT AT 0,0:" 32 SPACE ": L
ET UT=UT+2: IF UT>500 THEN G
O TO 700
66 RANDOMIZE USR 32470: IF SCRE
EN$ (X,0)<>" " THEN FOR A=0 TO 9
: BEEP .02,21-A: RANDOMIZE USR 32
520: NEXT A: LET ELET=ELET-1: PR
INT AT X,0:" ": GO TO 900
70 RANDOMIZE : PRINT AT 0,0:"P
ONTSZAM:";PONT;AT 0,11:" ELETEK
:";ELET;AT 0,22:" UT:";UT;AT 0,3
0:"KM";AT X,0: INK RND*7+1;"A":
INK 7: FOR A=1 TO INT (UT/50)+
1: PRINT AT INT (RND*19)+2,31;"B
": NEXT A
72 PLOT 0,166: DRAW 255,0: PLO
T 0,4: DRAW 255,0: PRINT AT 21,1
0:" LOVES:";TUZ
75 PRINT INK RND*7+1;AT X,0;"A
"
80 GO TO 40
700 FOR A=2 TO 20: INVERSE 1: P
RINT AT A,27;" " : NEXT A: PR
INT AT 10,27;"BAZIS": INVERSE 0:
FOR A=0 TO 35: RANDOMIZE USR 32
470:PRINT AT X,0;"A": NEXT A
720 PRINT AT 5,10: FLASH 1;"SIK
ERULT!!!" : PONT: "PONTOD VAN
,EZ " : IF PONT>SOK THEN LET SOK
=PONT: PRINT "UJ REKORD!": GO TO
740
730 IF PONT<=SOK THEN PRINT"MEG
NEM REKORD!" : "A LEGJOBB EREDME
NY:";SOK
740 GO TO 935
800 FOR A=1 TO 30: PRINT AT X,A
;" " : IF SCREEN$ (X,A+1)=" " THE
N NEXT A: PRINT AT X,1;" 30 SPAC
E : GO TO 40
810 BEEP .07,12: RANDOMIZE USR
32520: BEEP .07,0: RANDOMIZE USR
32540: LET PONT=PONT+1: FOR B=1
TO A+2: PRINT AT X,B;" " :NEXT B:
GO TO 40
900 IF ELET<0 THEN GO TO 920
910 GO TO 40

```

ZX-SPECTRUM

1050—1110: tájékoztató szöveg kiírása.

1120—1130: az UDG betöltése.

9000: a program felvétele.

A listán lévő dőlt betűk a megfelelő grafikus karaktert jelentik! Fekete Balázs

```

920 FOR A=0 TO 21: BEEP .05,A:
BORDER A/3: RANDOMIZE USR 32500:
PRINT AT A,0;"AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA": NEXT A: PRINT
#0;"BBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBB"
930 PRINT AT 11,2;"A PONTJAI D S
ZAMA:";PONT: IF PONT>SOK THEN PRI
NT " EZ UJ REKORD!": LET SO
K=PONT
934 PRINT " A REKORD:";SOK
935 PRINT AT 8,4;"VEGE A JATEKNA
K!";AT 17,8;"UJ JATEK? (I/N)"
940 FOR A=1 TO 16: IF A<13 THEN
RANDOMIZE USR 32520: BEEP .02,-
30
950 IF A>12 THEN RANDOMIZE USR
32540: BEEP .02,-10
960 IF INKEY$="I" THEN GO TO 20
970 IF INKEY$="N" THEN PRINT AT
1,12: FLASH 1;"VISZLAT!": STOP
980 NEXT A: GO TO 940
1000 CLEAR 32469: RESTORE : FOR
A=1 TO 89
1020 READ B: POKE 32469+A,B: NEX
T A
1022 DATA 33,0,64,22,0,62,192,6,
31,35,94,43,115,35,16,249,114,35
,61,32,242,201,0,0,0,0,0,0,0,0
1025 DATA 33,0,64,22,192,6,32,62
,255,150,119,35,16,249,21,32,244
,201,0,0
1030 DATA 33,255,63,6,192,14,32,
35,203,30,13,32,250,55,63,5,32,2
43,201,0
1040 DATA 33,0,88,6,192,14,32,43
,203,22,13,32,250,55,63,5,32,243
,201,0,0
1050 FOR A=0 TO 24: BORDER INT (
A/10): BEEP .005,A/1.8:RANDOMIZE
USR 32500: NEXT A
1060 BORDER 4: PAPER 0: INK 7: C
LS
1070 LET S=1: LET A$=" HELLO!
EBBEN A PROGRAMBAN EGY URHAJÓVAL
KELL ELJUTNI A TAMASZPONTRA. AZ
UTAT SZEMBEJÓVÓ METEORITOK NEHE
ZITIK. Ezeket KERULGETHETED LEFE
LE AZ 'A', FELFELE A 'Q' GOMBOKK
AL, AZ ENTER-REL PEDIG LÖHETED Ö
KET. 5 ELETED VAN, A TAMASZPONT
PEDIG 500 KM-RE TALALHATÓ. SOK S
ZERENCSE! (INDULAS:ENTER)
"
1075 PRINT AT 3,10;"URHAJÓZÁS";A
T 15,2: INK 6;"KESZITETTE " ; INK
4;"FEKETE BALAZS" TAB 10; INK 5
;"1986.XII.": RANDOMIZE USR 3250
0
1080 FOR A=1 TO 3: RANDOMIZE US
R 32540: RANDOMIZE USR 32540
1090 IF INKEY$=CHR$ 13 THEN GO
TO 1120
1100 NEXT A: LET S=S+1: IF S=LE
N A$ THEN LET S=1
1110 PRINT INK 0;AT 10,31;A$(S)
: GO TO 1080
1120 RESTORE 1130: FOR A=1 TO 1
A: GO TO 20
1130 DATA 128,96,125,126,125,96
,128,12,62,126,255,255,255,126,
9000 SAVE "URHAJO" LINE 1000: PA
USE 0: VERIFY "URHAJO": PRINT
FLASH 1;"Q.K.": STOP

```

COMMODORE PLUS/4

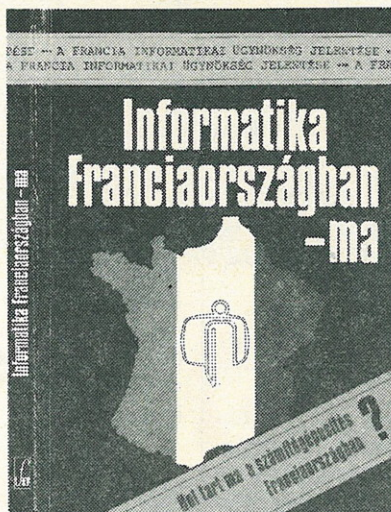
Világosság a Sötét Torony körül

Tavaly ősszel sok C+4 került országunkba. Ezeknek a gépeknek nagyobb a tárkapacitásuk, mint a C16-nak, ezért sokkal alkalmasabbak komoly feladatok megoldására. De akinek van, azért játsszon is vele!

És most játsszunk együtt! Mivel kompatibilis a C16-tal, tegyük a gépbe 16-os programot. Nagy a csalódás: a legérdekesebb játékok nem működnek, kuszan csíkos a képernyő. Első gondolatunk — hogy hibás a program — helytelen, mert a másik gépen futott! Gondolkodjunk tovább: hogyan is alkot képet egy C16-os játék? Karaktereket definiál magának, és így alakítja ki a főhős alakját. Tehát a karakterek adatai hibásak, vagy máshonnan veszi őket a gép, mint ahonnan kellene. Állítsuk át azokat a bajtákat, amelyek a karakterkészlet helyét adják meg a gépnek. Írjuk be POKE 65298,192:POKE 65299,17 és már — csíkos, esetleg gömbölyded formákkal — teli is a képernyő. Most vakon gépeljük be a program elindítását. Lám, már élvezhető az eddig rossznak vélt program!

Jómagam nagyon körülményesnek tartom e két utasítás begépelését játék előtt, ezért néhány javítást ajánlok a programokban. Az alábbiak POKE címet és értéket jeleznek.

TOWER	5123	76	DARK	7748	76
OF	5124	150	TOWER	7749	32
EVIL	5155	62		7750	57
	16022	133		14624	173
	16023	246		14625	224
		133			22
		247			160
		133			192
		248			140
		160			18
		192			255
		140			160
		18			17
		255			140
		160			19
		17			255



**Informatika
Franciaországban - ma.**
(Budapest, 1986.)
**Statisztikai
Kiadó
Vállalat,**
207 oldal. Ára: 150,-Ft)

A számítógépek tömeges megjelenése és zavarba ejtően sokféle alkalmazása egész sor megválaszolásra váró kérdést vet fel. Milyen új lehetőségek nyílnak, és hogyan aknázzhatjuk ki azokat a legelőnyösebben? Az útkeresés, a fejlesztés optimális iránya nemcsak szakmai körökben vitatéma, hiszen a haladásban mindnyájan érdekeltek vagyunk. A francia tapasztalatok közreadásával kiadónk — a két ország hasonló adottságaiból kiindulva — hasznosítható példát kíván a társadalom elé tárni.

A kötet fő erénye az aktualitás. A jelentés elkészítésének elsődleges oka, hogy az 1978. évi helyzetet feltáró emlékezetes Nora-Minc jelentés (amely 1979-ben „A számítógépesített társadalom” címmel jelent meg) bizonyos megállapításai napjainkra átértékelődtek. Szükségessé vált a célok és a cselekvési program újrafogalmazása.

A kiadvány átfogó helyzetjelentés a francia számítógépesítési programról. Részletes statisztikai adatokkal alátámasztva vizsgálja a 80-as évek fejlődését. Az elemzések rendkívül tárgyilagosak, bemutatják a hiányokat és árnyoldalakat is. Rávilágítanak a kritikus pontokra, a társadalom különböző szféráiban felmerült érdeklentésekre, ugyanakkor kiemelik a folyamat erős egy-egy mutató vonásait. A kötet árnyalt képet ad az eszközkínálatról és a felhasználási módok széles köréről. Ismerteti az egyes részterületek szakértőinek véleményét az elterjedést gátló tényezőkről. Elemzi a különböző szektorok és vállalatok viselkedésmin-táit, végül a vizsgálat néhány évre előrete-kintő végkövetkeztetéseit fogalmazza meg. Ajánlást vezérfonalat jelenthetnek a köve-tendő út és a társadalom előtt álló felada-tok meghatározásához.

A jelentés tanulságos olvasmány a számítástechnika iránt érdeklődők és a szakma-

beliek számára egyaránt. A francia és a jelenlegi magyar elektronizálási program rokon vonásainak összehasonlítása, tapasztalatainak kamatoztatása új lendületet adhat terveink megvalósításához.

**Mikroszámítógépes
kiállítások
tapasztalatai.**
**Hannoveri Vásár,
Párizsi SICOB,
Müncheni Productroni-
ca.**
(Budapest, 1986.)
LSI ATSZ,
271 oldal. Ára: 228,-Ft)

A kötet a címben szereplő három, 1985-ben megrendezett kiállítás tapasztalatait foglalja össze tematikus rendben. „Eszközök — technológiák — alkatrészek” címmel a félvezető memóriák, a berendezésorientált eszközök, a hibridáramkörök stb. területén látott újdonságokat ismerteti. „Készülékek” címmel a lézertárolók, a mágnesszalagos, mágnesszalagok háttértárolók, a nyomtatók és a modemek legfontosabb paramétereit elemzi. A „Rendszerek” című fejezet a személyi számítógépek, az adatmegjelenítő képcsöves terminálok és a többterminálos mikroszámítógépek fejlődési trendjét foglalja össze. Végül az „Alkalmazás” című rész áttekintést ad az alkalmazási és a szoftver újdonságokról, az ergonómia területén bekövetkezett változásokról és a piaci helyzetről is.

**C programozási
útmutató.**
Szabvány és ajánlás.
Készítette:
Szilassy Bertalan
(Budapest, 1986.)
LSI ATSZ,
58 oldal. Ára: 64,-Ft)

A szerző összefoglalja mindazokat az ismereteket a C programozási nyelvről, melyek a nemzetközi szabvány megjelenéséig a szakirodalomban rendelkezésre álltak. A kötet a C nyelv de facto ipari szabványának tekinthető használatát tartalmazza az AT and T Bell Laboratóriumának publikációi, a különböző ANSI X3J11 munkabizottsági jelentések és az X/OPEN Group közleményei alapján.

A kötet megjelentetését az indokolta, hogy 1984-ben a legjelentősebb európai számítógépgyártó cégek által megalakított X/OPEN Group által definiált ún. Egységes Alkalmazási Környezet (Common Applications Environment) az AT and T UNIX operációs rendszerére épül, melynek preferált nyelvi rendszere a C programozási nyelv.

	140		76
	19		71
	255		30
	76		
	9		
	20		
VEGAS	8226,96	ROCK	4156 160
	8231,56	MAN	4157 192
			140
(Ehhez a programhoz csak ez a javítás jó)			18
			255
			160
			17
			140
			19
			255
SAKK	7165 160	PULSAR	10074 76
	7166 192		10075 41
	140		58
	18		234
	255	14889	160
	160	14890	192
	17		140
	140		18
	19		255
	255		160
	76		17
	202		140
	31		19
	224		255
	224		76
	224		58
	224		21
	224		
	224		

A pontok a mindig eggyel nagyobb bájtot jelentik. Ha a bájtokat átállítottuk, vegyük fel az új változatot kazettára, és kezdődhet a játék.

ORMOS ZOLTÁN

Közületek figyelem!

**Mikroszámítógépet
akarnak vásárolni?
Tájékozzanak
a naprakész
piaci helyzetről!**

Díjtalan ismertető!

**MESZ
Számítástechnika**

1368. Budapest, Pf. 193.

Bolgár 32 bites

Bulgária már a harmadik szocialista ország, amely bemutatja 32 bites miniszámítógépét. Az Izot 1055C jelzésű gép a VMS operációs rendszerrel működik és a VAX 11/730-cal kompatibilis. 4 Mbájtos operatív tára is jelzi, hogy elsősorban grafikai, tervezés-támogatási rendszerként kívánják üzemeltetni. 1986 szeptemberére a nullszéria során 5 gép készült el, a sorozatgyártás idén kezdődik.

Elektronika a sertéshizlalásban

A Mezőkovácsházi Új Alkotmány Tsz-ben 26 millió forint értékű rekonstrukciót hajtottak végre az állattenyésztő telepen, melynek során teljesen automatizálták a sertéslelepszolgáltatását. Az etetőgép a számítógépes receptúra alapján összekeveri a takarmánykomponenseit a takarmánykonyha tartályában. Ezzel főlegessé teszi a külön takarmánykeverőt, csak az anyagról kell gondoskodni. A számítógép nyilvántartja a készletet, továbbá azt, hogy az egyes takarmányfajták még hány etetésre elegendők.

A számítógéppel lehet azt is meghatározni, hogy az egyes rekeszekbe, melyekben különböző számú sertés van, dekára mennyi moslékot adagoljon a szivattyú. Etetés után egy csőrendszer a mosogatást is elvégzi.

A rekonstrukció után egy kilogramm élősúly előállítására 2,88 kg abrakot használnak s a 12 ezer sertést nevelő ágazat évente 18-20 millió forint nyereséget hoz.

Zenélő winchester

A brit AMS-cég Audiofile rendszere a hangszalagos zenetárolást winchester-tárral helyettesíti. A digitalizált formában tárolt hanganyag ezzel a megoldással rendkívül kényelmesen szerkeszthető: nincs

szükség az időrabló szalagsérvelésre, az anyag bármely része pillanatok alatt elérhető. Az Audiofile mono, sztereó és több csatornás felvételt-lejátszást tesz lehetővé.

A házasság számítógéppel köttetik

Szulejmán Guresci a törökországi Izmirben hat esztendőn át hadakozott a paragrafusokkal, hogy elválhasson feleségétől. A hosszan tartó pereskedés Guresci minden megtakarított pénzét felemésztette. A válás után ő is, s a volt felesége, Neslin Caglasa is — egymás tudta nélkül — ugyanahhoz az izmiri házasságközvetítő irodához fordult. Az űrlapban megfogalmazott igények alapján aztán Guresci számára Caglasát, Caglasa számára pedig Gurescit emelte ki a számítógép a tárolt több ezer fős adatállományból. Mindketten meglepődtek az eredményen, majd hallgatva a gépre, újból összeházasodtak.

Iskolagépek

Megnégyszereződött tavaly az iskola-számítógépek száma. 1985 végén az általános és középiskolákban még csak három ezer hatszázhatvannyolc számítógép szolgált a tanuló-ifjúságot, 1986 végére pedig már tizenötezer. Ez egyben azt is jelenti, hogy 1985-ben több mint négyszáz diákra jutott egy számítógép, tavaly viszont már alig több mint kilencvenre. S a jelenlegi ötéves terv során az iskolagépek száma tovább nő: az évtized végére a tervek szerint ötvétezer számítógép lesz a közoktatásban.

A rendkívüli ütemű növekedésben jelentős szerepet játszott a ráckevei Aranykalász Mezőgazdasági Termelőszövetkezet által kezdeményezett „Számítógépet az iskoláknak!” mozgalom: 1986 elején az ország valamennyi gazdálkodó szervezetéhez elküldte felhívását. Ez széles körben visszhangra talált. Több száz intéz-

mény, vállalat és termelőszövetkezet, budapesti kerületi és megyei tanácsok vásároltak a vonzaskörükben működő iskolák számára számítógépet. Az állam az akciót 30%-os dotációval támogatta.

Az idén a gépek hatékonyabb alkalmazása a fő cél. Ennek érdekében már a könyvelést, az ügyvitelszervezés oktatását és tanulását segítő programcsomag forgalmazását is megkezdték.

Újságírók

Az év utolsó napjaiban avaták fel a Bálint György Újságíró Iskola számítógépes kabinetjét. Az öt Proper-16 típusú géppel felszerelt laboratóriumban a hallgatók megismerkedhetnek a számítógépes szövegszerkesztés lehetőségeivel. Az „elektronikus” cikkírásról túlmenően a gépek máris szinte elektronikus szerkesztőségként üzemeltethetők: a Magyar Távirati Iroda rendszerével összekapcsolva a megyei szerkesztőségekhez eljuttatott napi belpolitikai, külpolitikai és sporthírek is rendszeresen megérkeznek.

A laboratórium idején programja igen feszes. Tavasszal továbbképző tanfolyamok indultak, szeptembertől pedig az Újságíró Iskola tíz hónapos képzési idejéből 90 tanórát a szövegszerkesztés foglal el.

A kabinet megnyitásával a Központi Statisztikai Hivatal anyagi segítségével megvalósított beruházás első szakasza zárult le. A következő lépcsőben megvalósul mindaz, amit a MTESZ Nyomdaipari Egyesülése által kidolgozott tanulmányterv célul tűzött ki: a gépeket fényszedőhöz csatlakoztatva a szerkesztés eredményeképpen kiszedett filmet kapnak, melyről csak lemezt kell készíteni, s indulhat a nyomás.

Kétkarú

IBM PC által felügyelt, koordinált mozgású, általános célú kettős manipulátorkar-rend-

szer kifejlesztésén dolgoznak az Egyesült Államokban. A rendszer két, egyenként hét szabadságfokú manipulátorkart fog összehangoltan, terhelés alatt működtetni. Az előre elkészített programon kívül manuális vezérlésre is lehetőség lesz: az operátor a mozgást botkormány segítségével távirányíthatja.

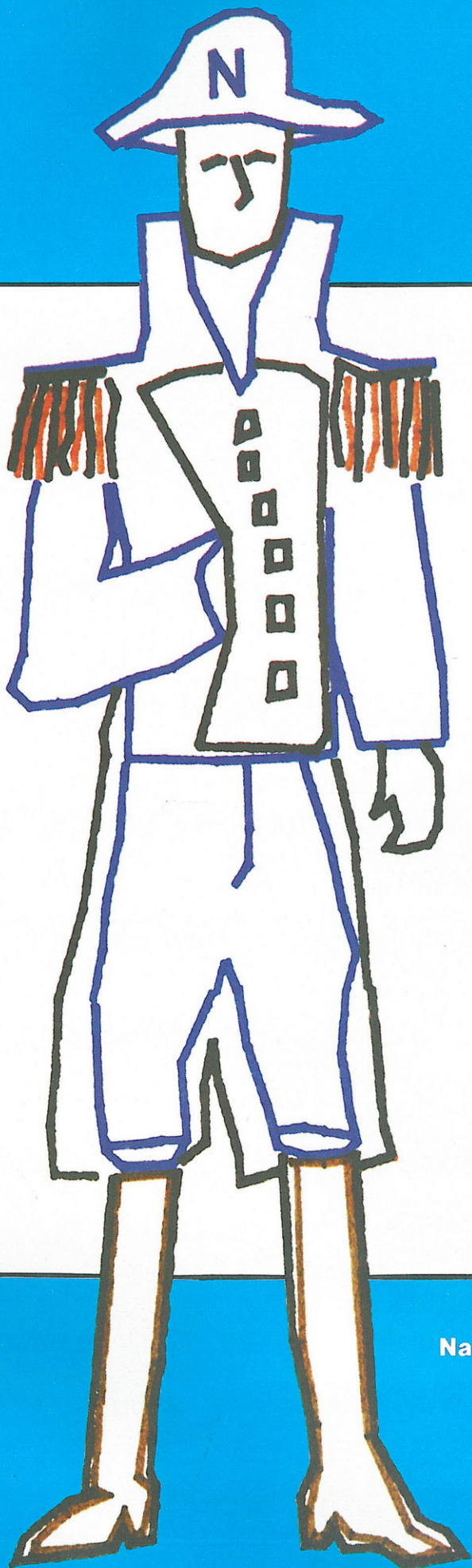
Alkalmazási területe lehet a veszélyes környezetben (atomerőmű, űrállomás) végzett távirányítású karbantartás. A rendszernek az egyszerű manipulátorokkal szembeni lényeges előnye, hogy egyszerűbb szerszámokkal dolgozhat.

Kubai XT-kompatibilisek

Három új, az IBM PC XT-vel kompatibilis számítógéptípussal jelentkezett Kuba az elmúlt ősszel. Közülük a legkisebb teljesítményű az LTEL/16, mely a hagyományos IBM PC XT-nek felel meg; az LTEL/18 a 4,77 MHz-es meghajtás mellett 8 MHz-cel is üzemelhet; az LTEL/20 pedig 8 beviteli-kiviteli csatornával és 8 db EPROM-csatlakozóval készül. Mindegyik típus 640 kb-ig bővíthető operatív tárral, 360 kb-ot hajlékonylemez tárolóval, 10-20 Mb-ot winchester-tárral rendelkezik.

Tisztul a zavaros

A szó szoros értelmében minőségi előrelépést jelentett az elmúlt év őszén a számítástechnika-alkalmazás terén a minőségkérdésnek a figyelem középpontjába kerülése. Meghirdették például a Kiváló Áruk Fóruma emblémára szóló pályázatot a szoftvertermékekre, sőt kisvállalkozás is alakult a minőségértékelésre: az SQS fő tevékenysége a mikroszámítógépes szoftverárak minősítése, az ellenőrző, összehasonlító eljárások kifejlesztése és alkalmazása. Ezen a fő tevékenységen alapul az SQS szoftver- és hardverkiválasztási szolgálata, mely így semleges, pártatlan — a fejlesztőtől, a forgalmazótól és a felhasználótól függetlenül —, korszerű módszerekkel végzi a minőségértékelést.



GRATIS

Széles körű grafikai alkalmazásra készült a Grafikus Adatbáziskezelő és Tervező Interaktív Szoftver (azaz GRATIS). Ez a rendszer közel 140 grafikus funkciót valósít meg, amely a teljes 2D és 3D grafikus tervezési dokumentálási, nyilvántartási és archiválási feladatokat támogatja egy felhasználóbarát, könnyen kezelhető és adaptálható felhasználói illesztőre támaszkodva.

Hierarchikus adatbázisa a grafikus primitívek, alakzatok, blokkok, képek 2D és 3D adatszerkezeteit, grafikus fájlait kezeli. Mind a hardver, mind pedig a szoftver rendszer egy minimál és egy professzionális kiépítettséghez készül.

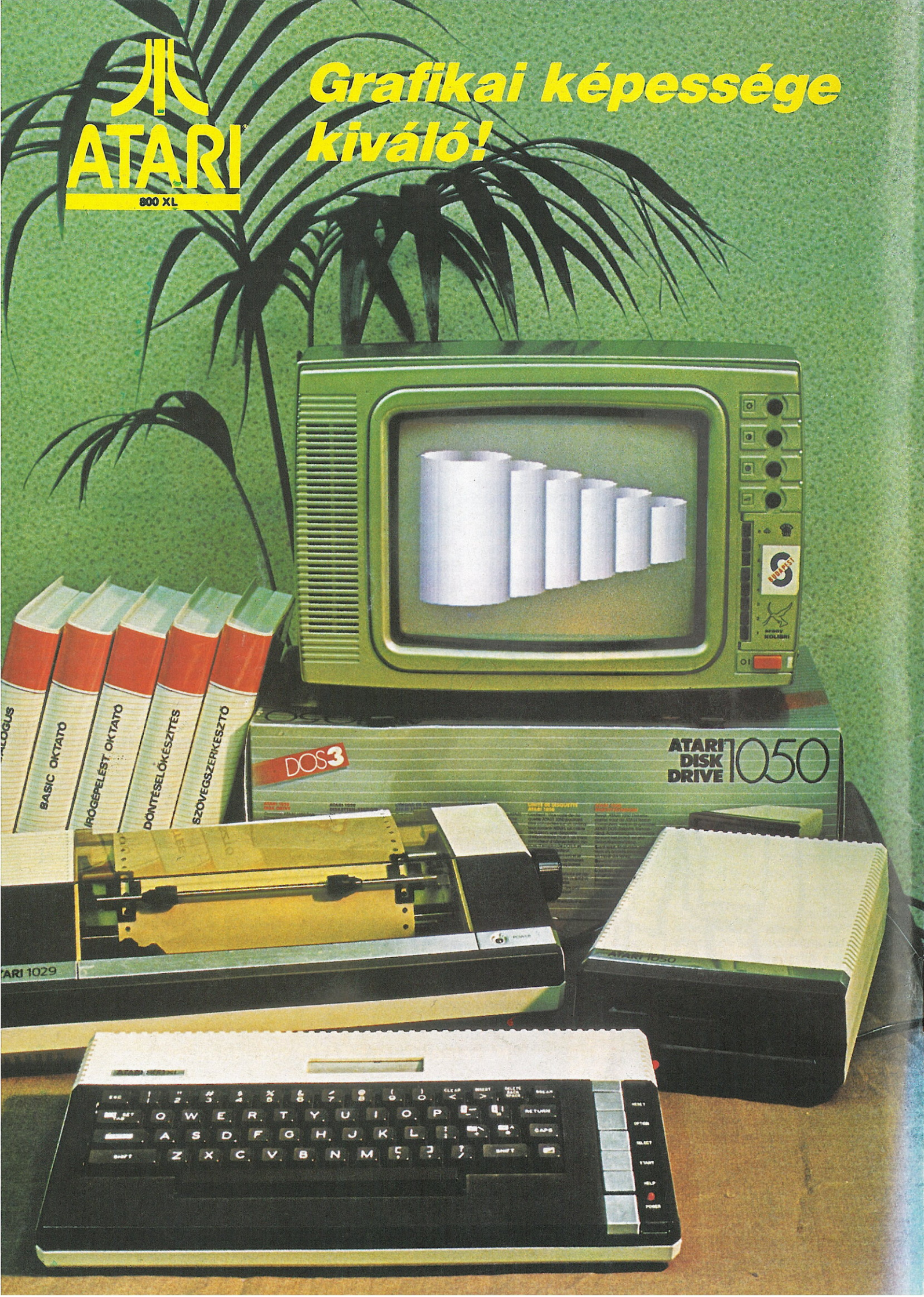
Tipikus felhasználási területei az építészet, a könnyűipar, az elektronika, a mezőgazdaság és az oktatás.

A GRATIS az IBM PC és az azzal kompatibilis gépeken fut.

Napoleon — a GRATIS segítségével

ATARI
800 XL

**Grafikai képessége
kiváló!**



DOS 3

ATARI
DISK
DRIVE 1050

ATARI 1029