

# mikro számítógép magazin

Ára: 30 Ft



**EGY SZÁMÍTÓGÉP BEN  
MINDEN BENNE VAN.  
MI AZT TUDJUK,  
HOGYAN KELL  
KIHOZNI BELŐLE.**

**KSH Számítástechnikai és Ügyvitelszervező Vállalat**

1989/9.



**PROFESSZIONÁLIS  
SZÁMÍTÓGÉPEK  
PERIFÉRIÁJA  
AJÁNLATUNKBAN**

**DS 400**

**NAGY SEBESSÉGŰ PROFESSZIONÁLIS  
MÁTRIXNYOMTATÓ,  
KÜLÖNLEGESEN NAGY IGÉNYBEVÉTELEKRE**

**JELLEMZŐI:**

- kétirányú és több üzemmódú, 24 órás folyamatos nyomtatás
- 18 tűs ballisztikus nyomtatófej
- sebesség: 400 karakter/s
- kompatibilitások: EPSON MX80/100, grafikus, IBM grafikus, Diablo 630, DS 180
- grafikus nyomtatás, képpontonként címezhető módon
- BAR CODE, OCR nyomtatás
- a felhasználó által definiált karakterek
- Centronics párhuzamos és RS232 soros illesztő
- 16 kb-át puffer
- ASCII, német, francia, svéd, finn, norvég, dán, görög, spanyol és IBM PC-kompatibilis karakterkészletek
- 8 bites, 256 karakterkódot tartalmazó és a felhasználó által tölthető készlet

Ár: 250 000,- Ft + 25% áfa.

**ELŐJEGYZÉST AZ ELŐSZÁLLÍTÁS JOGÁ-  
VAL**

**FELVESZÜNK**

**1989. IV. NEGYEDÉVRE**

**TOVÁBBI TERMÉKEINKRŐL**

**KÉRJE ÁRJEGYZÉKÜNKET!**

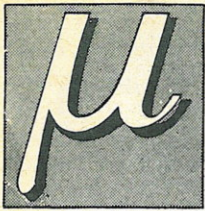
**SCI-L Számítástechnikai Informatikai Fejlesztő Leányvállalat  
Kereskedelmi Iroda**

**Budapest, Iskola u. 10. 1011.**

**Telefon: 154-065, 350-180/180, 181, 182, 183, 184**

**Telex: 22-4599**

**Telefax: 35-39-15**



# mikro számítógép magazin

7. ÉVFOLYAM  
1989/9. SZÁM

## A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság  
vezetője:  
Kovács Győző

A szerkesztőség  
munkatársai:

Bakos Tamás  
(programozástechnika)

Broczkó Péter  
(hírek)

Kovács Győző  
(levelezés)

Nagy Imre  
(tanuljunk együtt)

Petróczy Judit  
(könyvek)

Pinke György  
(Enterprise)

Soltészné Vizi Zsuzsa  
(tervezőszerkesztő)

Szebenszki Sándor

Tamásné Lakó Erika

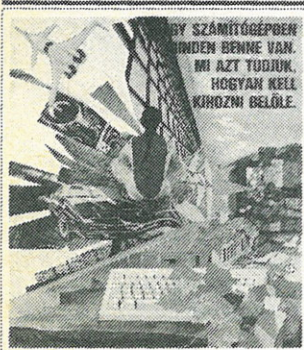
Terebessy Ákosné

Varga János

(olvasószerkesztő)

Címképünk:  
Velekey József Lajos  
munkája

**μ mikro számítógép  
magazin**



Felelős szerkesztő:  
Könyves Tóth Pál

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

Levélcím:  
1371 Budapest  
Pf. 433

Kiadja:  
MTESZ Neumann János  
Számítógéptudományi Társaság  
1054 Budapest, Báthori u. 16.

Levélcím:  
1368 Budapest 5. Pf. 240

Telefon: 329-349

Felelős kiadó:  
Tóth Istvánné ügyvezető  
főtitkárhelyettes

Terjeszti a Magyar Posta  
Előfizethető a hírlapkezelés  
hivataloknál  
és a Posta Hírlap-előfizetési  
és Lapellátási Irodáján  
(1900 Budapest XIII.,  
Lehel u. 10/A)  
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  
88—1135



Szikra Lapnyomda  
Budapest (89—1319)  
Felelős vezető:  
Dr. Csöndes Zoltán vezérigazgató

INDEX: 25 629  
ISSN 0236-6088

### TARTALOM

- 2 Vírus a számítógépben
- 8 Nemes Tihámér Országos Középiskolai Számítástechnikai Verseny
- 10 Gyorstöltők a C64-en
- 13 Feladatok — megoldások
- 41 A Microsoft BASIC-interpretere Amigára
- 45 Programtermék — Ismerd meg anyanyelved!
- 46 Adok — veszek — cserélek

### TANULJUK EGYÜTT!

- 3 A Pascal rejtelméi
- 5 Programtervezés számítógéppel

### CSIPEGETŐ

- 15 ZAK McKRACKEN and the alien mindbenders
- 16 Mire jó az RS232 csatorna?
- 17 Újra Gunship!
- 17 Örökélet
- 18 Változtatás következményekkel
- 18 TOP-lista

### PROGRAMOZÁSTECHNIKA

- 19 Vírusirtó program
- 22 Védőoltás
- 23 Virusgazdák kiméljenek!
- 24 Programozási fogások és melléfogások

### ENTERPRISE

- 25 Diagramszerkesztő
- 28 Érdekes görbék Enterprise gépre
- 30 ENTERPRISE-TOTÓ 1989. II.
- 31 SPOKE-SPEEK
- 31 Mi a manó?

### PÉCÉZZÜNK!

- 33 Nagy felbontású monitorok
- 35 NC — Norton Commander
- 37 Amikor a programokat el szabad „lopni”!!!
- 38 Az MS — DOS 3.30 parancsai funkciók szerint
- 40 Kiterjesztések .@a@a-tól .WRI-ig
- 40 FILL — merevlemezről floppyra

### SAKK

- 44 Támadás és védelem 2.

### KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

3

15

19

25

33

44

47

*„A számítógépvírus valójában néhány olyan programkód, amelyet azért csináltak, hogy imitálja önmagát, vagy ezzel együtt néhány egyszerű feladatot is végrehajtson: például üzenetet küldjön a képernyőre különböző időpontokban. Egy nagyon egyszerű, de rémisztő ötlet.”*

*(Dave Moussond: A Computer Virus)*

# Vírus a számítógépben

Előre kell bocsátanom, nem vagyok vírusszakértő. Azt is el kell mondanom, hogy a számítógépvírusokról szóló első híreket évekkal ezelőtt kételkedve fogadtam, meg voltam győződve arról, hogy megint valami olyasmiről van szó, amit szennzációhajhász újságírók találtak ki.

Később azért utána néztem a dolgoknak, és kénytelen voltam elfogadni, hogy a vírus létezik, egyre több és hatékonyabb vírust készítenek és helyeznek el a gépekben különféle megfontolásból ilyen vagy olyan szakemberek.

Néhány hete az egyik kollégámmal Bulgáriában, Várnában voltunk, ahol bemutatót tartottunk a matematikusok éves konferenciáján. A bemutató második napján kollégám, András, gyanakodva nézegette a képernyőt, gyanús volt az egyik programja. A program által elfoglalt memóriaterület ugyanis egy nap alatt jelentős mértékben megnőtt. Megnéztük néhány órával később, a program ismét gyarapodott néhány kbájttal.

Gyorsan megszületett a diagnózis: a kölcsön kapott gép „fertőzött” volt, és megfertőzte András garantáltan „egészséges” programját is. A helybeli vírusszakértők kollégámmal együtt rögtön konzultációt tartottak, és megállapították, hogy az egyik legismertebb vírusról van szó, amelyik beépül valamelyik tárolt programba, ott elkezd ismételni önmagát, „szaporodik”, és végül tulajdonképpen haszontalan információval tölti fel a rendelkezésre álló szabad tárterületet. A jelenség — talán a leírásból is látszik — nagyon hasonlít a ráksejtek egészségtelen burjánzásához. Az említett számítógépvírus tevékenységének is hasonló az eredménye: a számítógép egy idő után nem fogad el több információt, gyakorlatilag leáll.

Sokakkal beszélgettem arról, hogyan is kerül a vírus a számítógépekbe. Sokféle véleményt hallottam. Az egyik kolléga szerint játékos kedvű szakemberek, főleg fiatal emberek kellemetlenkedéséről van szó; egyfajta vetélkedésről, ki tudja még a legvédehetőbb gépeket vagy hálózatokat is megfertőzni. Különösen becses tréfa például egy „agyonvédett” hadügyi vagy banki számítógépbe juttatni a vírust. Egy ilyen tett felér egy fél Nobel-díjjal. Legutóbb a hannoveri vá-

sáron ki szerettünk volna próbálni egy programot, és odamentünk egy ismerős céghez, hogy „rákérdzünk” a gépre. Régi ismerősöm volt a cég képviselője, ennek ellenére sajnálkozva tárta szét a kezét: sajna, nem fogadhat, a még nagyobb főnök megtiltotta idegen program ráengedését a gépre. Tudniillik az előző napon, mint már sokszor a hannoveri bemutatókon, két ismeretlen fiatal embernek megengedték saját programjuk kipróbálását, akik aztán „hálából” egy elég kemény munkát adó vírust hagytak a gépben. Egyébként a sztorit más cégnél is hallottam, így Hannoverben tényleg arról lehetett szó, hogy a régi szokásokat ismerve ez a két ismeretlen fiatal ember végigfertőzött egy sor gépet, és közben figyelte, hogy megtalálják-e a „vakcinát” a híres professzionális cégek szakemberei.

Nyilvánvaló, hogy az egyre növekvő vírusveszély létrehozott egy új technológiát, a vakcinagyártást, illetve az ezt megelőző vírusdiagnosztizálást. Persze vannak már vírusgyűjtők is, sőt azt mondják, hogy a víruskészítők szigorúan titkos szövetsége is megalakult.

Azt is hallottam, hogy ez a szövetség csak teljesen korrekt „párbajképes” szakembereket fogad tagjai közé, akik fogadalmat tesznek, hogy csak olyan vírust állítanak elő, ami kellemetlenkedik, de nagy bajt nem okoz. Belegondolni is rossz, hogy milyen következményekkel járna egy olyan vírus működése, ami például egy katonai objektumban hamis atomriadót jelez, vagy egy atomerőműben megváltoztatja a vezérlést, megsemmisít kulturálisan, tudományosan értékes adatbázisokat. Még felsorolni is háborzongató. Nem nagyon hiszek az önkéntesen korlátozott károkozásban, ennél sokkal jobbnak tartom a hatékony védekezést.

Mondják, de hiába kérdeztem rá, senki sem igazolta azt a feltevést, hogy a szerzők jó része egy időzített bombával, azaz egy beépített, de nem aktivizált vírussal védekezik szellemi tulajdonának ellopása ellen. Ez a „védekezés” állítólag úgy működik, hogy a legálisan vásárolt szoftverben benne van ugyan a vírus, de be van „tokozódva”. Ha a szoftvert illegálisan másolják, akkor a vírus feléled, és elkezd áldástalan működését. Ha a vírus kiöltöje ravasz és jól kép-

zett, akkor véletlenszerűen időzített vírust tervez a tolvaj megbüntetésére, ami a lopás után egy ideig még alszik, és ezzel elaltatja a szoftver illegális használatát is, hagyja abban a hitben ringatózni, hogy minden a legnagyobb rendben van. Aztán persze kezdődik a haddel-hadd!

Beszélgetve a vírusokról, illetve a vírusveszélyről, voltak olyan partnereim, akik azt mondták, hogy ma már létezik a számítógépek AIDS-e is, ami annyiban különbözik a valódi AIDS-től, hogy ma még nem találták fel a számítógépes kondomot, vagyis a leghatékonyabb védekezést. Egyet tehetünk: nem hagyjuk a számítógépünket „egészségileg” kontrollálatlan programokkal érintkezni. Sokan azt is mondják, mérlegelni kellene a világméretű számítástechnikai hálózatok alkalmazásának a korlátozását is, mert a távkapcsolat az egyik legkomolyabb potenciális veszélyforrás a nagy felhasználók számítógépeire.

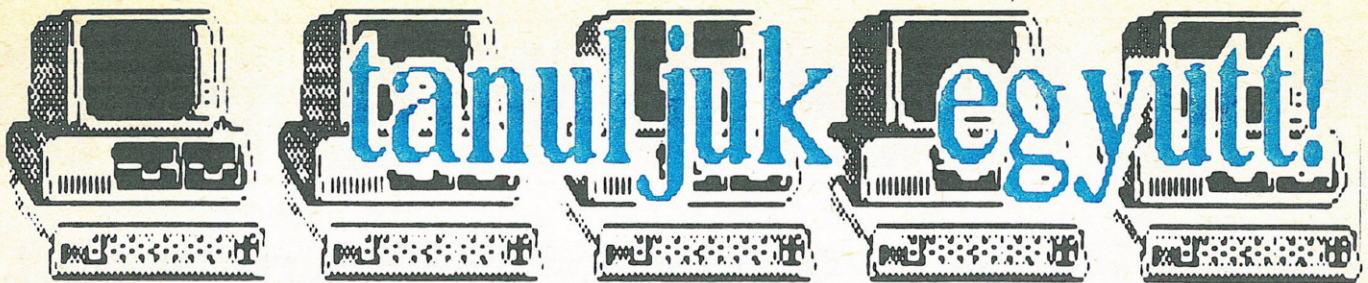
A számítógépes vírus korunk valódi problémája, ami ellen védekezni kell, az újabb és újabb és egyre ellenállóbb vírusok ellen egyre hatékonyabb vakcinák bevezetésével. Talán a hatékony ellenőrzés és védelem nem teszi szükségessé a hálózati összeköttetések elvágását, a gyanakvást, félelmet minden olyan floppytól, amely nem garantált forrásból származik. Ha már a világban, igaz, hogy nagy nehézségekkel, de még a politikusoknak is sikerült elosztatniuk az egymás iránti bizalmatlanságot, akkor talán mi is képesek leszünk ugyanerre a számítástechnika számunkra annyira kedves világában.

*Kovács Győző*

**Az Országos Műszaki Információs Központ és Könyvtár (OMIKK) Referáló Kiadványok Szerkesztősége külső munkatársakat keres külföldi folyóiratok magyar nyelvű referálására az alábbi szakterületeken:**

- elektronika
- elektrotechnika
- fizika (alkalmazott)
- gyártásautomatizálás
- mélyépítés
- számítástechnika
- távközléstechnika
- vízipítés.

*Elsősorban angol-, német-, francia- és orosznyelv-ismerettel rendelkező szakemberek jelentkezését várjuk a 181-994, ill. 382-300/113 telefonszámokon (munkanapokon 8–17 óráig) vagy személyesen az OMIKK anyaggyűjtési és nyilvántartási csoportnál (Budapest VIII., Múzeum u. 17. III. 301. szoba) hétfőn, szerdán és csütörtökön 8 és 17 óra között.*



# A PASCAL REJTELMEI

## 13. Speciális lehetőségek a Turbo Pascalban

Az eddigiekben elsősorban a standard Pascal elemeit alkalmaztuk, a Turbo Pascal specialitásainak említését lehetőség szerint kerültük, illetve ha ilyesmiről esett szó, arra külön felhívtuk a figyelmet. Ebben a részben kifejezetten a Turbo-fordító néhány jellegzetes lehetőségét és eszközét ismertetjük.

### 13.1 Önállóan futtatható programok

Programjainkat eddig mindig a **R** (run) paranccsal fordítottuk le és indítottuk el. Bár ez a procedúra viszonylag rövid programok esetén legfeljebb 3-5 másodpercig tartott, de a fordító alatti futtatás — annak betöltése, elindítása, a **W** és az **R** parancsok kiadása — végső soron bizonyos kényelmetlenséget jelentett. Mindezekről megszabadulhatunk, ha a „belőtt” programot a DOS alatt önállóan futtatható programfájl (COM kiterjesztésűvé) alakítjuk.

A COM fájl létrehozásához alkalmazzuk a parancs főmenü **O**: compiler options parancsát. Ekkor a képernyőn a 41. ábrán látható kiírás jelenik meg.

Használjuk a **C** parancsot, majd **Q**-val lépünk ki a compiler options menüből! A most kiadott **C**: compile parancs hatására a fordító a programot változatlan névvel és COM kiterjesztéssel végrehajtható fájl alakítva helyezi el az aktuális meghajtó lemezén. A **Q** paranccsal kilépünk a Pascal rendszerből, és a lefordított programot nevének bebillentyűzésével elindíthatjuk.

Például:

A Pascal program neve: PROBA.PAS  
A végrehajtható program neve: PROBA.COM

A végrehajtható program indítása:  
PROBA

Ha a PAS és a megfelelő COM fájl méretét megvizsgáljuk, a COM fájl általában lényegesen (2–15 kb-ajttal) hosszabbnak mutatkozik. Ennek magyarázata igen egyszerű: míg a futáshoz szükséges ún. run-time-rutinok a COM fájlhoz annak létre-

hozása során hozzákapcsolódnak, addig a Pascal-fordító alatt futó programnál erre nincs szükség, mert a rutinokat a tárban lévő fordítóból lehet meghívni.

### 13.2 Direktívák a fordítónál

A fordítóval kapcsolatos direktívák — kissé lezserül fogalmazva — a program futására, kezelhetőségére vannak hatással. A pontosabb meghatározás az egyes direktívák részletes ismertetésénél ki fog derülni. Minden direktívának van egy alapértelmezése, ezt a fordító indításakor automatikusan felveszi.

Szintaxisuk igen egyszerű, a kommentekhez hasonlóan kapcsos zárójelek között kell őket elhelyezni. A kezdő zárójel után a \$ karakter áll, ezt közvetlenül (szóköz nélkül) követi a direktíva betűjele, és — ha szükséges — a funkció be- és kikapcsolására utaló + vagy - jel. Ha egyszerre több direktívát kívánunk alkalmazni, az egyes jeleket egymástól vesszővel elválasztva kell leírni. Például: {\$U+}

{\$I-,R+}

A következőkben a legfontosabb direktívákat részletesen ismertetjük.

#### 13.2.1 B: Választás CON és TRM között

Alapértelmezés: B+

Ha a B direktíva aktív, {\$B+}, a CON periféria érvényes, ha passzív, {\$B-}, a TRM, azaz az **input** és az **output** fájlnevek a CON, illetve a TRM készülékekhez vannak hozzárendelve. A direktíva globális érvényű, azaz a futás során nem definiálható újra. Számunkra a kérdés egyébként érdektelen, csak a teljesség kedvéért közöljük.

#### 13.2.2 C: Control-S és Control-C

Alapértelmezés: C+

A C direktíva a Ctrl karakterek értelmezését vezérli a konzolon folyó B/K műveletek közben. Ha aktív, {\$C+}, a Ctrl-C billentyűkombinációval a **read** és **readln**

eljárás végrehajtása megszakítható. A Ctrl-S a képernyőn végzett **write** és **writeln** eljárások végrehajtását függeszti fel; ilyenkor annak folytatása bármelyik billentyű lenyomásával elindítható (ez a funkció azonos a Break billentyű hatásával). A {\$C-} eredményeképpen mind a Ctrl-C, mind a Ctrl-S hatástalanná válik.

A C direktíva aktív állapota kissé lassítja a képernyő kimenetet, tehát ha e folyamat sebességét növelni akarjuk, az aktív állapotot meg kell szüntetni.

A direktíva globális érvényű, a program befejezéséig nem definiálható újra.

#### 13.2.3 I: I/O hibakezelés

Alapértelmezés: I+

Az I direktíva az I/O (B/K) hibakezelés vezérlésére szolgál. Ha aktív, {\$I+}, a B/K folyamatokat a Pascal rendszer ellenőrzi. A {\$I-} előírásával a hibakezelés átvehető a rendszertől. Az esetleges hiba kódja az **IOresult** nevű standard függvény segítségével kérdezhető le. Ha nincs hiba, a hibakód értéke 0. Az **IOresult** hívása a hibaállapotot is törli, enélkül egyébként nem is hajthatók végre újabb B/K műveletek.

#### 13.2.4 R: Indextartomány ellenőrzése

Alapértelmezés: R-

Ha az R direktíva aktív, {\$R+}, a rendszer futás közben ellenőrzi az indexek helyességét, és azt, hogy a felsorolási, intervallum stb. típusú változók a deklarációkkal meghatározott tartományban vannak-e. Ezt a lehetőséget a 8. pontban em-

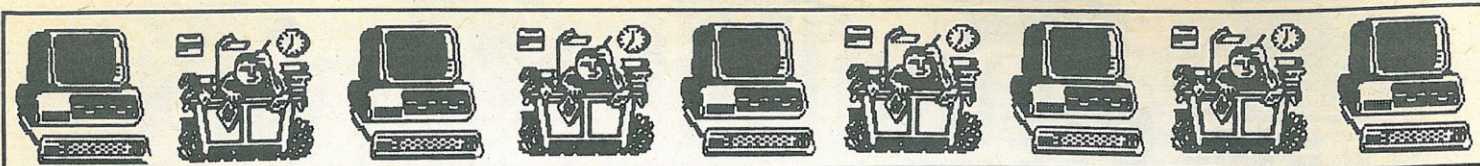
41. ábra

```

compile -> Memory
           Com-file
           cHn-file

command line Parameters:

Find run-time error  Quit
>
    
```



lítettük, amikor arról volt szó, hogy az értéktartományok ellenőrzését elvégeztet-hetjük a Pascal rendszerrel.

A direktíva passzív volta, {\$R-} esetén nincs ellenőrzés, és ez programhibához vezethet. A célszerű eljárás: a program-fejlesztés és kipróbálás alatt a direktíva aktivizálása, a hibamentes program véglegesítése előtt pedig kikapcsolása (ez utóbbi esetben ugyanis a futási idő jelen-tősen csökken).

### 13.2.5 U: Felhasználói megszakitás

Alapértelmezés: U—

A Pascalban kezdők számára talán a legfontosabb direktíva. Alkalmazását nem lehet elég nyomatékosan tanácsolni. Véleményünk szerint bonyolultabb programok írásakor nem szabad a programból kihagyni, még a Pascalban jártasabbak-nak sem.

Az U direktíva a felhasználói megszaki-tást vezérli. Ha aktív, {\$U+}, a program a futás tetszőleges fázisában megszakit-ható Ctrl—C-vel; ha nem, csak B/K folyama-tok közben szakítható meg (ha ezt egyé-bként a C direktívával nem tiltottuk le).

A direktíva aktivizálásával a futási idő nagymértékben növekszik, így alkalmazá-sa hibátlan programokban már nem cél-szerű. Programfejlesztéskor viszont taná-csos aktívra állítani, mert egy végtelen ciklusba kerülű program esetén a gép csak újraindítással (Ctrl—Alt—Del) állít-ható le. Ilyenkor viszont programunk el-vész, hiszen többnyire — ez általános ta-pasztalat — még nem mentettük lemezre.

### 13.2.6 I: Include Files (fájlok belefoglalása)

Az I: Include Files direktívát követő fájl-név megadásával a fordító „befordítja” a programba a megjelölt fájlt. Formája:

{\$I FILENEV, TIP}

Az I után a szóköz alkalmazása kötele-ző, a fájl TIP kiterjesztésének pedig há-rom karakterből kell állnia. Ha ez a feltétel nem teljesül, szóközökkel kell kiegészíte-ni. A direktíva alkalmazására a későbbiek során — a Pascal-grafika kapcsán — többször sor kerül majd. (A Turbo Pascal rendszer a grafikát támogató eljárásokat külön fájlokban — GRAPH.P és GRAPH.BIN — tartalmazza.)

## 13.3 Szerkesztési parancsok a Turbo Editorban

Eddigi munkánk során — elsősorban abból adódóan, hogy csak rövid progra-mokat írtunk — nem lehetett sok gond a programok gépelésével. Hosszabb progra-mok írása közben viszont gyakran fordul elő, hogy a programot hiba, ésszerű-sítés vagy az esztétikusabb kivitel céljából át akarjuk szerkeszteni. Ez nagyobb átala-kítás esetén sok gépelési munkával jár-hat, például amikor sorokat törölünk és írunk új helyekre.

Az ilyen átrendezi munkát nagymér-tékben támogatja a Turbo Editor sokféle blokkművelete. Ezek segítségével össze-függő programrészeket tudunk törölni, máshova áthelyezni, átmásolni, külön (le-mezes) fájlba kiírni vagy már meglévő fáj-lokat a programba bemásolni.

A blokkműveleteket indító parancsok Ctrl—K-val kezdődnek (két kivétellel, ezekre majd az aktuális helyen felhívjuk a figyelmet). A billentyűkombinációt a kép-ernyő bal felső sarkában megjelenő ^K igazolja vissza. Ezután még egy, a pa-rancs típusától függő, többnyire a pa-rancs funkcióját leíró angol szó valame-lyik betűjének megfelelő billentyű lenyo-mására van szükség. Ezeket az angol sza-vakat a parancsok ismertetésénél közöl-jük is, nagybetűvel jelölve a parancsban használt billentyűt.

### Blokkok megjelölése

A kurzort a megjelölni kívánt blokk ele-jére (első karakterére) állítjuk; ezután ki-adjuk a következő parancsot:

**Ctrl—K, B**

A kurzort a megjelölni kívánt blokk vé-gére (az utolsó karakter utáni pozícióra) állítjuk; ezután kiadjuk a következő pa-rancsot:

**Ctrl—K, K**

A megjelölt blokk kiemelten (más ár-nyalattal, színes monitoron a sárgán írt programszövegtől eltérően fehéren) jele-nik meg a képernyőn.

### Blokkműveletek

Ha a megjelölt blokkot törölni kívánjuk, a

**Ctrl—K, Y**

billentyűzetkombináció hatására a blokk sorai a programból törölődnek. Blokkáthelyezés (moVe) esetén a kurzort arra a pozícióra állítjuk, ahová a blokkot át akarjuk helyezni; ezután adjuk ki a

**Ctrl—K, V**

parancsot.

A blokk eredeti helyéről a kurzor által meghatározott helyre kerül, de továbbra is megjelölt formában (más árnyalattal vagy színnel kijelölve) marad.

Blokkok másolásánál (Copy) a kurzort arra a pozícióra állítjuk, ahová másolni akarjuk a blokkot, majd kiadjuk a követke-ző parancsot:

**Ctrl—K, C**

A blokk a kurzor által meghatározott helyre kiemelten másolódik át, de eredeti helyén is — már kiemelés nélkül — meg-marad.

**Ctrl—K, H**

billentyűkombináció használatakor a blokkmegjelölés nem szűnik meg, de a ki-jelzés igen, a blokk a program többi ré-szeitől most nem különböztethető meg.

A kijelzést ugyanezzel a billentyűkombi-nációval állíthatjuk vissza, azaz a megje-lölt blokkot ismét láthatóvá tehetjük.

Az elrejtett (Hidden) blokkokra egyes blokkműveletek igen, mások viszont nem alkalmazhatók. Mivel ennek a kérdésnek csak csekély jelentősége van, részletesen nem foglalkozunk vele. Ha valakinek ked-ve van, próbálja ki.

Blokkot lemezes fájlba a

**Ctrl—K, W**

paranccsal írathatunk ki (Write). A pa-rancs hatására az editor a képernyő leg-felső sorában a fájl nevét kéri. Ha ezt megadjuk, a parancs végrehajtódik, ha név megadása nélkül nyomjuk meg az En-ter billentyűt, a parancs hatástalan. A fájl-név beírásánál az operációs rendszer használatok szokásos meghajtó- és út-vonal- (directory-) megadási módot ter-mészetesen alkalmazhatjuk (például a: \pascal\adatok\blokk.dta).

Ha lemezes fájlról kívánunk beolvasni (Read) egy blokkot, a kurzort arra a po-zícióra állítjuk, ahová a külön fájl tartalmát be akarjuk másolni. Ezután adjuk ki a

**Ctrl—K, R**

parancsot, amelynek hatására a képernyő legfelső sorában az editor a fájl nevére kérdez rá. Ha a fájlnevet megadjuk, a fájl a program megjelölt helyére bemásoló-dik, és ott kiemelten megjelenik. Nem lé-tező fájl megadásakor a képernyő legfel-ső sorában hibaüzenet jelenik meg. Az Enter billentyű közvetlen lenyomására a parancs nem hajtódik végre, a meghajtó és az útvonal megadása ugyanaz, mint a Ctrl—K, W esetében.

A blokk kiírása és beolvasása segítsé-gével az editor támogatja a strukturált programozás logikáját. A programozási munkánk közben keletkező és várhatólag más programokban is jól alkalmazható el-járásokat későbbi felhasználás céljaira egy eljáráskönyvtárba helyezhetjük el, il-letve a már meglévő és az adott problé-ma megoldása során felhasználható eljá-rásokat egyszerűen beépíthetjük készülő programunkba.

### Blokkok kezdete és vége

A pozicionálásra szolgáló két parancs rejtett blokkok esetén is végrehajtható. Az eddigiektől abban térnek el, hogy kez-detüket nem a Ctrl—K, hanem a Ctrl—Q billentyűkombináció jelenti.

**Ctrl—Q, B**

paranccsal lehet a blokk kezdetére pozici-onálni, ilyenkor a kurzor a blokk első ka-rakterére áll.

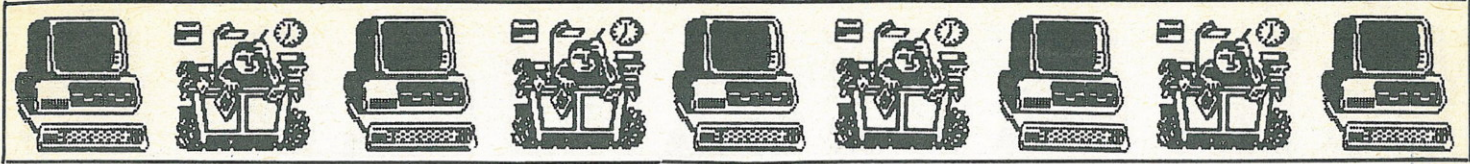
A blokk végének pozicionálására a

**Ctrl—Q, K**

parancs szolgál; ekkor a kurzor termé-szetszerűleg a blokk utolsó karakterére áll.

A Turbo Editor további parancsainak is-mertetésére most nem térünk ki.

Nagy Imre



# Programtervezés számítógéppel

Az alkalmazói szoftverek piacán a szövegszerkesztők, az adatbázis-kezelők és a kiadványszerkesztők mellett a CAD (Computer Aided Design — számítógéppel segített tervezés) rendszerek a legnépszerűbb termékek. Aki a műszaki tervezés munkamenetét akár csak vázlatosan is ismeri, tudja, hogy a kivitelezéshez nélkülözhetetlen grafikus dokumentáció elkészítése időigényes munka. A CAD rendszerek megjelenése előtt kialakult munkamegosztás létrehozta a műszaki rajzoló szakmát, amelynek művelői tehermentesítik az agymunkát végző mérnököket a terv letisztázása, a műszaki rajz tényleges elkészítése alól. A CAD rendszerek azonban ennél is több segítséget nyújtanak, hiszen a monitoron megjelenített ábra már a tervezés közben is pontos, hű képe a tervezett objektumnak. Más szóval a CAD program nem csupán a tervezői folyamat mechanikus segítője, hanem az érdemi munkát támogató szellemi erőforrás is.

Ezt tudva, méltán csodálkozhatunk, hogy a programozók, akik a legváltozatosabb rendszerekkel látják el a műszaki, gazdasági, de még a művészeti felhasználókat is, saját magukra nem gondolnak. Az ilyen CAPD (Computer Aided Program Design) rendszer alap gondolatát vetjük fel, és egy házi használatra alkalmas, egyszerűsített változatának elkészítéséhez adunk útmutatást. A programozásban kevésbé járatosak kedvéért egy BASIC megoldást is mutatunk, amely a legtöbb, nálunk használatos házi számítógépre alkalmas. Maga a minta a CBM BASIC V2.0 verziója szerint készült, így változtatás nélkül használható a Commodore 64, 16, Plus/4 és 128 gépeken.

Bár minden hasonlat sántít, egy ismeretlen terület feltárásánál mégis célszerű valamilyen analógiát használni. A programozási munka esetében ez a modell a műszaki tervezés lehet. A mérnök munkájának célja egy épület vagy gép elkészítése olyan formában, hogy az rendeltetésének megfelelően, használható legyen. A programozásban a cél egy futtatásra, használatra kész program előállítás. A műszaki rajz olyan dokumentáció, melynek alapján a kivitelező — aki legtöbbször nem azonos a tervezővel — az objektumot el tudja készíteni. A program elkészítése (megírása — belövése) tehát olyan dokumentációt igényel, amelynek alapján a kivitelező (programozó) úgy tudja létrehozni a készterméket vagy annak egy részletét, hogy esetleg nem is ismeri a rendeltetését.

Ez a felfogás bizonyára idegen azoknak, akik már önállóan megoldottak néhány programozási feladatot. Ne felejtsék el, hogy ezek az egyszemélyes feladatok rendszerint kis programok, és a tervező-kivitelező munkamegosztás ilyenkor szóba sem kerülhet. Egy nyúlketrec vagy fászkamra tervét elég megálmodni, s ennek alapján a következő hétvégén összefuzítható. A toronyházak esetében ez nem megy.

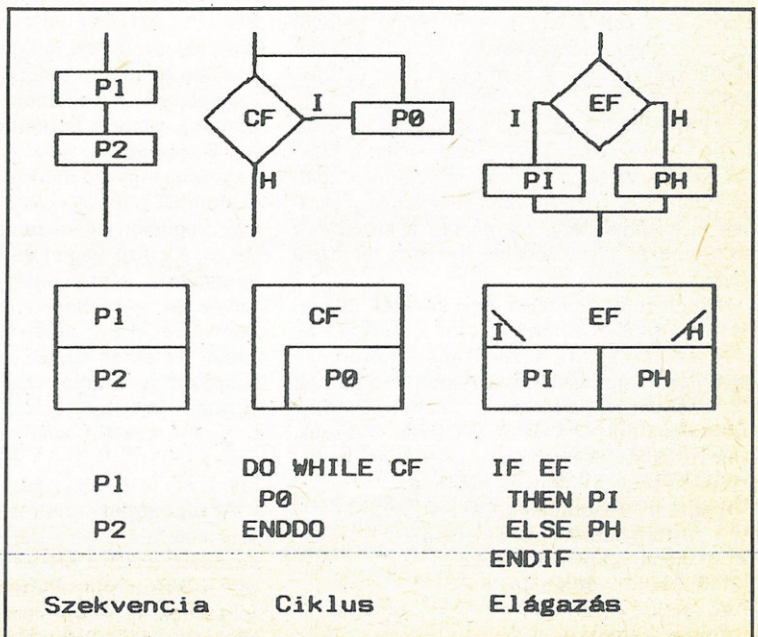
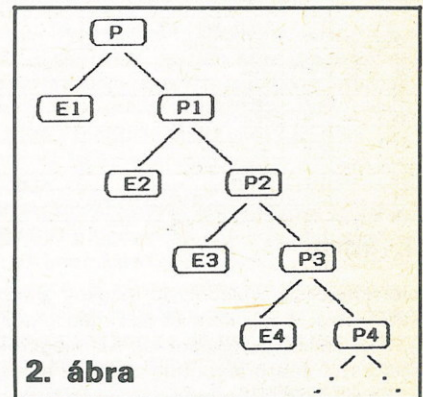
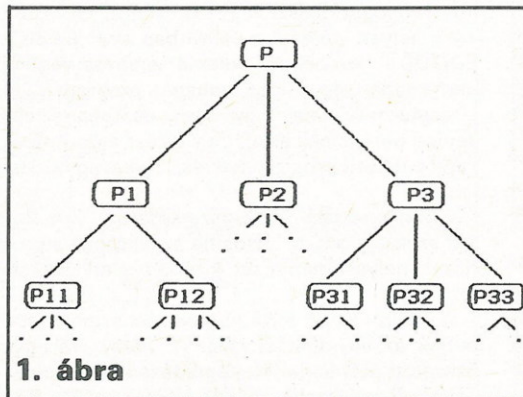
Az alapozási munkában részt vevő vasbetonszerelő szakmúsnak esetleg fogalma sincs az épület egészéről. Neki csak az a fontos, hogy milyen alapanyagból és milyen méretben kell a vázszerkezetet elkészíteni. Ennek megfelelően a programot írónak — a kivitelezőnek — a munkamegosztási hierarchiában felette álló programtervezőtől kell megkapnia azokat a dokumentumokat, amelyek alapján a kódolást el tudja végezni.

Ilyen dokumentum, vagyis az elkészítendő program terve az algoritmus pontos leírása, amely a műszaki tervekhez hasonlóan tartalmazza a szerkezeti elemeket (műveletek) és azok kapcsolatát (programszerkezet). A leírás részletessége függhet a feladat nagyságától, bonyolultságától és a kivitelező szakértelmétől. Ami azonban független a feladattól, a kivitelezőtől, az a leírás pontossága, egyértelműsége.

A műszaki dokumentációnak kialakult az egyezményes formalizmusa, melyet ugyan országoként változóan, de nagyjából egyöntetű szabványok rögzítenek. A programozási gyakorlatban a folyamatábrát, a struktogramot és a szöveges leírást használják az algoritmus lejegyzésére. A többféleség előnye az, hogy a tervező választhatja meg a neki legkedvezőbb formát. Hátránya viszont, hogy a kivitelezőnek mindháromban otthonosnak kell lennie. Az egyszemélyes programozásban ez utóbbi szempont lényegtelen, de az algoritmusok közzététele, cseréje esetén hasonló helyzet áll elő.

Mindhárom kifejezőmódnak vannak előnyei. Neumann János a számítástechnika hőskorában még azon a véleményen volt, hogy a folyamatábra a programtervezés egyetlen és nélkülözhetetlen eszköze. (Illik tudni, hogy elsőként ő javasolta használatát.) Akkor, a kis kapacitású gépek és a szükségszerűen kis programok idején ez a megállapítás igaz is volt. Ma a legkisebb gépek operatív memóriája is olyan hatalmas programokat tud befogadni, melyeknek algoritmusát csak igen nagy papíron lehetne így ábrázolni. Ha egy ilyen „pókháló” kis lapokra tördelünk, akkor viszont az egyes lapok közötti hivatkozások miatt olvashatatlanná, áttekinthetlenné válik a közlendő információ.

A Dijkstra és munkatársai által bevezetett struktogram sokat megőriz a folyamatábra ké-





PROGRAM : Másodfokú egyenlet

```

a=0
DO WHILE a=0
  Input: a,b,c
DO←
d=b2-4*a*c
IF d>=0
  THEN d=sqr(d)/(2*a)
  b=-b/(2*a)
  x1=-b-d
  Print: x1
  IF d<>0
    THEN x2=-b+d
    Print: x2
  ELSE ---
  IF←
ELSE Print: 'Nincs valós gyök.'
IF←
END
  
```

P.	F(P)	T(P)	T - A - R - T - A - L - O - N
1.	2	1	Másodfokú egyenlet
2.	3	1	De: a<>0, b, c
3.	4	0	a=0
4.	5	1	Input a,b,c enig a=0
5.	6	-4	do while
6.	7	-1	a=0
7.	8	0	Input: a,b,c
8.	9	-2	do←
9.	10	1	Megoldás
10.	11	0	d=b <sup>2</sup> -4*a*c
11.	12	1	Gyökök, ha vannak
12.	13	-3	if
13.	14	-1	d>=0
14.	15	-6	then
15.	16	1	d=sqr(d)/(2*a)
16.	17	0	d=sqr(d)/(2*a)
17.	18	1	Gyökök
18.	19	0	b=-b/(2*a)
19.	20	1	Gyökök
20.	21	1	Egyik gyök
21.	22	0	x1=-b-d
22.	23	0	Print: x1
23.	24	1	Másik gyök, ha van
24.	25	-3	if
25.	26	-1	d<>0
26.	27	-6	then
27.	28	1	Van másik
28.	29	0	x2=-b+d
29.	30	0	Print: x2
30.	31	-7	else
31.	32	0	---
32.	33	-3	if←
33.	34	-7	else
34.	35	0	Print: 'Nincs valós gyök.'
35.	0	-3	if←

#### 4. ábra

pies kifejezési erejéből, de megköti a tervező kezét azzal, hogy nem enged meg ugrásokat (GOTO nélküli programozás). Aki megszokta a szerkesztett ágazó algoritmusokat, az a strukturált programozást szigorúsága, kötöttsége miatt nehezen fogadja el. Ám a módszer másik alapelve, a top-down (felülről lefelé haladó) tervezés olyan előnyökkel jár, hogy a kezdeti ellenkezés után senki nem bánja meg az átállást.

A szöveges leírás vagy pszeudokódot elsősorban laikusok számára találták ki, ám a szakma felismerte, hogy nemcsak a dokumentációban, hanem a program tervezésének szakaszában is alkalmazható. Különösen a strukturált programozás elvét követve hasznos tervezési eszköz.

Nem lesz haszontalan egy konkrét feladat tervezését követve megmutatni a módszer alkalmazását. Válasszuk példaként a másodfokú egyenlet megoldására készíthető programot. Pontosabban fogalmazva a feladatot: valódi másodfokú egyenlet valós gyökereit keressük. A tervezés első lépéseként az algoritmust a következő sémával fogalmazzuk meg:

Program: Másodfokú egyenlet

1. Az együtthatók beolvasása
2. Az egyenlet megoldása
3. A megoldás kiírása

Vége

A top-down elvet alkalmazva, minden lépést

külön feladatnak értelmezve, ugyanígy kisebb lépésekre bontjuk. a program terve így alakul:

Program: Másodfokú egyenlet

- 1/1 Az a <> 0 együttható bevétele
- 1/2 A b és c együtthatók bevétele
- 2/1 A diszkrimináns számítása
- 2/2 Ha van valós gyök, akkor gyökképlet
- 3/1 Ha van gyök, akkor kiírni
- 3/2 Ha nincs gyök, akkor kiírni: "NINCS"

Vége

Azt hiszem, felesleges tovább folytatni a példát, hiszen egyrészt a top-down elv lényege, másrészt a következetes végrehajtás technikai problémája ennyiből is látható: minden tervezési lépésben, minden felbontási szinten előről végig le kell írni a terv vázlatát, annak újabb, az előzőtől alig különböző változatát. Különösen így van ez, ha egy harmadik tervezési elvet, a kis lépések elvét is alkalmazzuk. Éppen ezért nem lehet elmarasztalni azokat, akik a tervezés közben, különösen annak kezdeti szakaszában fejben dolgoznak, és csak amikor már sok sorból áll a vázlat, akkor veszik a kezükbe a ceruzát. Mint mindig, most is találunk egy egyszerű megoldást: nagyobb papírt kell használni és ritkábban írni a sorokat. Így a felbontás a sorok közé való beszúrással és a feleslegessé váló sorok kiradírásával megszervezhető.

Ennél még jobb, ha szövegszerkesztőt használunk. Ha valaki ezzel beéri, akkor a további elemzéstől el is tekinthet. Sőt, ha az ún. általános szövegszerkesztők (Easy Script, DELTEX stb.) helyett például a ROM-ban lévő BASIC EDITOR-t használjuk, akkor a tervezés végén nem csupán algoritmus, hanem a program lesz a végtermék. Csakhogy egyik esetben sincs semmi garanciánk arra, hogy a kész algoritmus vagy a BASIC program helyes, logikailag hibátlan.

Olyan speciális szövegszerkesztőre van tehát szükség, amely biztosítja az elkészült algoritmus helyes felépítését, kizárja a szerkesztési hibákat.

A felülről lefelé történő tervezést szemléltethetjük az ún. döntési fával (1. ábra). Ha egy összetett probléma (P) megoldásán dolgozunk, célszerű azt kisebb részproblémákra (P1, P2, P3), majd ezeket ismét részekre bontani (P11, P12, ..., P31, P32, P33) mindaddig, amíg minden részprobléma elég egyszerű, optimális esetben triviálisan megoldható lesz.

Általában a megoldóra van bízva annak eldöntése, hogy a felbontást hogyan végzi. Az egyik véglet az, amikor lehetőleg egyenlő nagyságú, egyező nehézségűnek látszó részekre bontjuk a problémát. Ezt nem mindig könnyű megítélni, különösen nehéz a tervezés kezdetén. A másik véglet ezzel ellentétes stratégiát alkalmaz. A problémát egy egyszerű és egy másik, az eredetinel alig egyszerűbb részre bontva végül a 2. ábrán látható döntési sémát kapja. Az ábrán E1, E2, ... elemi, P1, P2, ... összetett problémát jelöl. Lényegében ezt a "favágó" módszert alkalmazzák azok, akik a programtervezést a következőképpen végzik. Első felbontás: 1. sor + többi. Második felbontás: 1. sor + 2. sor + többi. És így tovább.

Az algoritmus tervezését segítő CAPD rendszer alapfunkciója a helyettesítés. A tervezés kezdetén adott a program, melyet egy összetett műveletnek tekinthetünk. Ezt a műveletet felbontjuk egyszerűbbekre. A rendszernek tehát úgy kell működnie, hogy a kezelő kiválaszt-

hasson a megfogalmazott műveletek közül egyet, és megadhatja annak egy felbontását. A rendszer ezt a felbontást behelyettesíti az algoritmusba, a kiválasztott művelet helyére. Eközben a terv egészének szerkezetét meg kell őrizni. Ezáltal a top-down elv alkalmazását már biztosítottuk.

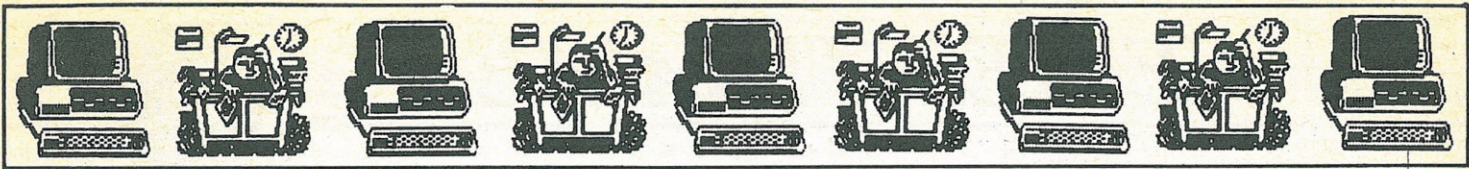
A legegyszerűbb CAPD rendszert úgy építhetjük tovább, hogy a helyettesítést a strukturált programozásban használt három elemi szerkezetre (3. ábra) korlátozzuk. Eszerint a helyettesítést a kezelő a felbontandó művelet kiválasztása után a felbontás típusának kijelölésével írja elő. Ha ezt megtette, akkor e tipustól függően két egymás utáni műveletet, vagy egy ciklusfeltételt és a ciklus magját, vagy az elágazási feltételt és a két alternatív műveletet kell megadnia.

Ahhoz, hogy az így szerkesztett szöveg olvasható, értelmezhető legyen, olyan alapszavakat is be kell iktatni, melyek utalnak a beírt szöveges leírások jellegére: ciklus feltétele, elágazás igaz ága, ciklus vége stb. Kézenfekvő megoldás, hogy ezeket az alapszavakat a rendszer automatikusan illeszse be az algoritmus szöve-

```

0 rem "CAPD-0
10 rem "Alapszavak
20 sw$(2)="do←" :rem"Ciklus vége
30 sw$(3)="if←" :rem"Elágazás vége
40 sw$(4)="do while " :rem"Ciklusfelt.
50 sw$(5)="if " :rem"Elágazás felt.
60 sw$(6)="then " :rem"Igaz ág
70 sw$(7)="else " :rem"Hamis ág
100 tx=200 :rem"A táblázat mérete
110 dim f(tx) :rem"Mutató
120 dim t(tx) :rem"Tipus
130 dim d$(tx) :rem"Listaelem
140 rem =====
150 print "[CLR] < F ő m e n ű >"
160 print " ====="
170 print " 1 Új program
180 print " 2 Szerkesztés
190 print " 3 Betöltés
200 print " 4 Listázás
210 print " 5 Kimentés
220 print " 6 Befejezés
230 :
240 print " ? ← Választás"
250 get f$ : if f$="" then 500
260 if f$="1" then gosub 2000
270 if f$="2" then gosub 1000
280 if f$="3" then gosub 2400
290 if f$="4" then gosub 3000
300 if f$="5" then gosub 2200
310 if f$="6" then goto 2100
320 goto 140
330 :
500 rem...."Képfriállítás"....
510 :t=t(k) : if t>0 then 570
520 :print d$(k):
530 :if t<-3 then 570
540 :s=s+1:print chr$(13)"[Fel]3Inst]":
550 :if t<-1 then print :goto 570
560 :print right$(str$(k+1000)";",4)
570 k=f(k) : if k>0 and s<20 then 510
580 if k=0 then k=1 :s=20 :print"=====
590 return
600 :
1000 rem -----"Szerkesztés"-----
1010 print "[CLR]"; :s=0 :k=1 :gosub 500
1020 get v$ : if v$="" then 1020
1030 : if v$=chr$(13) then return
1040 : if v$="*" then 1100
1050 : s=s-1 :gosub 500 :rem frissites
1060 goto 1020
1100 rem ...."Helyettesítés".....
1110 print "[Home]18Le]"
1120 print "_____";
1130 print "_____";
1140 forx=1 to 16:print"[10Sp]";:nextx
1150 input "[4Fel] p=";p
1160 if t(p)>0 then print"[3Le]";:goto 1150
1200 rem "A p. művelet felbontása"
1210 print "Felbontás: s-c-e"
1220 get t$ : if t$="" then 1220
1230 if t$="s" then 1300
1240 if t$="c" then 1500
1250 if t$="e" then 1700
  
```





gége. Vegyük észre, hogy ez nemcsak kényelmessé teszi a szerkesztést, hanem egyben biztosítja azt is, hogy például egy ciklus lezárása ne maradjon ki a tervből. Ez a fogás az, amivel a CAPD rendszer a szöveg- és program-szerkesztőknél többet nyújt.

Eddig jutva a rendszer tervezésében, döntünk kell, hogy milyen alapszótárt használjunk. A kidolgozott BASIC programban alkalmazott megoldás (10–70 sorok) csupán egy lehetőség. A szótár egyéni ízlés szerint változtatható, magyarosítható. Mivel az alapszavakkal együtt a tervező által beírt mondatok (feltételek és műveletek) az algoritmust egyértelműen leírják, a szerkesztett szöveg kinyomtatásával a kívánt dokumentum birtokába jutunk. A leírást olvashatóbbá, emberibbé tehetjük azzal, hogy a szöveget a képernyőn vagy a nyomtatott sorokra tagoljuk. A mintaprogramban minden feltétel és művelet, továbbá a ciklus és az elágazás végét jelentő alapszó kiírása után új sort kezdünk.

A szerkesztővel készülő vagy már befejezett algoritmus szövege tehát egy olyan lista, amely alapszavakból és „mondatokból” áll.

```

1300 rem ... "Szekvencia"...
1310 if tv+2<=tx then 1340
1320 print "[CLR] Betelt a tábla!"
1330 for x=0 to 999 :next :return
1340 t(p)=1 :fp=f(p)
1350 f(p)=tv+1 :tv=tv+1
1360 d$="" : input "1.blokk":d$
1370 if d$="" then 1360
1380 d$(tv)=d$ :t(tv)=0
1390 f(tv)=tv+1 :tv=tv+1
1400 d$="" : input "2.blokk":d$
1410 if d$="" then 1400
1420 d$(tv)=d$ :t(tv)=0
1430 f(tv)=fp :goto 1010
1440 :
1500 rem .... "Ciklus"...
1510 if tv+4<=tx then 1540
1520 print "[CLR] Betelt a tábla!"
1530 for x= 0 to 999 :next :return
1540 t(p)=1 :fp=f(p)
1550 f(p)=tv+1 :tv=tv+1
1560 d$(tv)=sw$(4) :t(tv)=-4
1570 f(tv)=tv+1 :tv=tv+1
1580 d$="" : input "do while ";d$
1590 if d$="" then 1580
1600 d$(tv)=d$ :t(tv)=-1
1610 f(tv)=tv+1 :tv=tv+1
1620 d$="" : input "mag":d$
1630 if d$="" then 1620
1640 d$(tv)=d$ :t(tv)=0
1650 f(tv)=tv+1 :tv=tv+1
1660 d$(tv)=sw$(2) :t(tv)=-2
1670 f(tv)=fp :goto 1010
1680 :
1700 rem .... "Elágazás"...
1710 if tv+7<=tx then 1740
1720 print "[CLR] Betelt a tábla!"
1730 for x=0 to 999 :next :return
1740 t(p)=1 :fp=f(p)
1750 f(p)=tv+1 :tv=tv+1
1760 d$(tv)=sw$(5) :t(tv)=-5
1770 f(tv)=tv+1 :tv=tv+1
1780 d$="" : input "if ";d$
1790 if d$="" then 1780
1800 d$(tv)=d$ :t(tv)=-1
1810 f(tv)=tv+1 :tv=tv+1
1820 d$(tv)=sw$(6) :t(tv)=-6
1830 f(tv)=tv+1 :tv=tv+1
1840 d$="" : input "then ";d$
1850 if d$="" then d$="----"
1860 d$(tv)=d$ :t(tv)=0
1870 f(tv)=tv+1 :tv=tv+1
1880 d$(tv)=sw$(7) :t(tv)=-7
1890 f(tv)=tv+1 :tv=tv+1
1900 d$="" : input "else ";d$
1910 if d$="" then d$="----"
1920 d$(tv)=d$ :t(tv)=0
1930 f(tv)=tv+1 :tv=tv+1
1940 d$(tv)=sw$(3) :t(tv)=-3
1950 f(tv)=fp :goto 1010
1960 :

```

E lista elemei azonban nem olyan sorrendben kerülnek a rendszerbe, amilyen sorrendben a kezvegen szerepelnek. A rendszerünk a lista elemeit egy D\$(.) szöveges tömbben, a beírás sorrendjében tárolja. A listázáshoz szükséges sorrendet párhuzamosan a mutatók F(.) tömbjében szerkesztjük minden helyettesítőskor. A rendszer e két tömbön kívül egy T(.) típusjel is hozzárendel minden listaelemhez. A szerkesztés pillanatnyi állapotában még fel nem bontott műveletekhez a T=0 típusjellet csatoljuk. A felbontás után a művelet a T=1 típusje-

```

2000 rem -----"Új program"-----
2010 input "[CLR] A program neve":d$
2020 if d$="" then 2010
2030 d$(1)=d$ :t(1)=0 :f(1)=0 :tv=1
2040 p=1 : return
2050 :
2100 rem -----"Befejezés"-----
2110 if tv<2 then print "[CLR]" : end
2120 print "[CLR] Kimentsem? (i/n)"
2130 get q$
2140 if q$="n" then print "[CLR]":end
2150 if q$="i" then gosub 2200 : end
2160 goto 2130
2170 :
2200 rem -----"Kimentés"-----
2210 if tv>1 then 2240
2220 print "[CLR] Nincs program!"
2230 for x=0 to 999 :next :return
2240 input " Eszköz (1/8) " :e
2250 if e<>1 and e<>8 then 2240
2260 d$="00:"left$(d$(1),16)
2270 if e=1 then open 6,1,1,d$
2280 if e=8 then open 6,8,6,d$+","s,w"
2290 print#6,tv
2300 i=1
2310 rem ciklus
2320 :print#6,t(i)chr$(13)chr$(34)d$(i)
2330 i=f(i) : if i>0 then 2310
2340 close 6 : return
2350 :
2400 rem -----"Betöltés"-----
2410 input "[CLR] Eszköz (1/8) " :e
2420 if e<>1 and e<>8 then 2410
2430 input " A program neve:" :d$
2440 if d$="" then 2430
2450 if len(d$)>16 then 2430
2460 if e=1 then open 5,1,0,d$
2470 if e=8 then open 5,8,5,d$+","s,r"
2480 input#5,tv
2490 for i=1 to tv
2500 :input#5,t(i),d$(i)
2510 :f(i)=i+1
2520 next i : close 5
2530 p=1 : f(tv)=0 :return
2540 :
3000 rem -----"Listázás"-----
3010 if tv>1 then 3040
3020 print "[CLR] Nincs program!"
3030 for x=0 to 999 :next :return
3040 input " Eszköz (3/4) " :e
3050 if e<>3 and e<>4 then 3040
3060 be$="[Rvs-On]" :ki$="[Rvs-Off]"
3070 if e=3 then open 4,3 : goto 3130
3080 be$=chr$(145) :rem set/1
3090 ki$=chr$(17) :rem set/2
3110 open 4,4,7
3130 print#4,be$"PROGRAM : "d$(1)ki$;
3140 i=1 : s=0 : n1$=-1
3150 m=2 : m$=chr$(13)+"" : rem margo
3160 rem ciklus
3170 :t(1)
3180 :if t>0 then 3300
3190 :if t=-2 then m=m-1 :m$=left$(m$,m)
3200 :if t=-7 then m=m-5 :m$=left$(m$,m)
3210 :if t=-3 then m=m-6 :m$=left$(m$,m)
3220 :if n1$ then print#4,m$; : s=s+1
3230 :if t<-1 then print#4,be$;
3240 : print#4, d$(1)ki$;
3250 :if t=-4 or t=-5 then m=m+1 :m$=m$+" "
3260 :if t<-5 then m=m+5 :m$=m$+" "
3270 :n1$= t>4
3280 :if e=4 or s<25 then 3300
3290 :get q$ : if q$="" then 3290
3300 i=f(1) : if i>0 then 3160
3310 print#4,chr$(13)be$"END"ki$
3320 close 4
3330 if e=4 then return
3340 get q$ : if q$="" then 3340
3350 return
3360 :

```

let kapja. A feltételek típusa T = -1. Az alapszavak e három típuskódtól eltérő jelet kapnak. A -2, ..., -7 kódokat a tördelt lista előállításiakor használjuk fel.

Hangsúlyoznunk kell, hogy a rendszerben nem írjuk elő a mondatok formáját. Például a tervben szerepelhet az „n páros” megfogalmazás. Amikor a tervet kivitelezük, azaz a program írásához kezdünk, akkor kell ezt a feltételt az adott programnyelv szabályainak megfelelően kódolni. Az „n páros” BASIC-ben  $n/2 > \text{int}(n/2)$  alakú, Pascalban  $\text{mod}(2,n) = 0$  alakú lesz. Ugyanígy kötetlen a műveletek megfogalmazása is. A tervben ugyanis a művelet nem kizárólag egy programbeli utasítást, hanem ennél összetettebb tevékenységet jelent. Ha a felbontást addig folytatjuk, amíg minden művelet elemi utasítás nem lesz, akkor a terv alapján a programozást ezeknek a műveleteknek a kódolásával, egyszerű lefordításával végezhetjük. Mindez persze nem zárja ki, hogy az algoritmusban az elemi műveleteket a programnyelv formalizmusával írjuk le: Input a, b, c;  $x = \text{sqr}(a^2 - u^2)$ ; Print „Nincs megoldás!”.

Nem szükséges azonban minden tervet az elemi műveletek szintjére felbontva elkészíteni. Elegendő addig a megfogalmazásig eljutni, amíg a kódolást egyértelművé nem teszi a leírás. Például az  $S = x(1) + \dots + x(m)$  műveleti leírást egy összegező ciklussal rutinszerűen kódolhatjuk, így a tervben ennek felbontása már felesleges.

A működőképes CAPD rendszerbe a helyettesítésen kívül további funkciókat célszerű beépíteni. A kész vagy készülő algoritmust lemezre/kazettára kell menteni, majd innen a további finomításhoz beolvasni. A BASIC programban alkalmazott megoldás csak a T(.) és D\$(.) táblázatot írja fel az adathordozóra a listázás sorrendjében. Ily módon onnan visszaolvasva, rendezett táblázatokkal folytatható a tervezés.

A BASIC programban külön modul gondoskodik a tördelt lista képernyőre/nyomtatóra való kiírásáról. Ez a lista eltér attól, amely a szerkesztés közben a képernyőn látható. Ennek oka egyszerűen az, hogy a 40 oszlopos képernyőn egy sort könnyen túlléphetünk, ha a sorokat széles margó vezetibe. A szerkesztési listát előállító képrfrissítő rutin a helyettesítendő művelet sorszámát is kiírja, melyre a kész dokumentumban nincs szükség.

Végül a lehetséges fejlesztésről. A bemutatott megoldás csupán a CAPD rendszer minimális funkcióit tartalmazza. Hatásos működő rendszerhez még néhányra szükség lehet: beírt szöveg megváltoztatása, törlés, beszúrás. A fejlesztés kiterjedhet arra is, hogy más szerkesztetek helyettesítését is megengedjük: végén tesztelő ciklus (REPEAT), számlált ciklus (FOR—NEXT), több irányú elágazás (CASE szerkezet).

Azt hiszem, a téma iránt érdeklődők további tippek nélkül is használható ötletekkel láthatnak hozzá saját házi CAPD rendszerük kiépítéséhez. Az itt közölt BASIC program átvétele és használata segíthet a tapasztalatok gyűjtésében, a fejlesztési irányok kitűzésében. A rendszer működését a 4. ábrán egy kidolgozott példán mutatjuk be. Az elkészített program tervében a műveleteket már „majdnem kódolt” formában írtuk. A rendszer által szerkesztett táblázatban olvashatók a közbenső lépésekben használt műveleti leírások.

Dr. Hack Frigyes

# NEMES TIHAMÉR

## Országos Középiskolai Számítástechnikai Verseny



A Nemes Tihamér-verseny második fordulóját 1989. március 11-én tartották. A döntőbe 65 versenyző jutott, köztük mi is. A feladat egy program megírása volt 5 óra alatt. A géptípust és a programozási nyelvet minden versenyző maga választhatta meg.

A program egy áramkör működésének szimulációját végezte, feltételezve, hogy minden alapelem kimenetén egységnyi idő múlva jelenik meg a bemeneti jelkombináció hatása, és ez azonnal megjelenik a következő alapelemek bemenetén is.

Az áramkör elemei a következő kapuk lehetnek: INVERTER (NOT), AND, OR, NAND, NOR és EXOR. (Az inverternek 1, a többi kapunak pedig 2 bemenete van.) Az áramkör maximum 10 kaput tartalmazhatott, kezdetben minden kapu bemenetén valamilyen határozott érték (például 0) volt.

A feladatkiírás meghatározta az adatstruktúra felépítését is: az áramkör leírását úgy kellett megadni, hogy minden alapelem bemeneteihez, illetve kimeneteihez rendelünk egy csomópont-azonosító számot, s az azonos csomópontokhoz tartozó be- és kimeneteket tekintjük összekötöttnek.

A feladatkiírás azt is feltételezte, hogy véges időn belül kialakul valamilyen stabil kimeneti jelkombináció (javasolta, hogy ez N kapu esetén legyen  $N^2$ ). Természetesen e stabil kimenetet a programnak meg kellett vizsgálnia.

A kiírás a program megírását 5 részfeladatra bontotta:

A) Az áramkör ábrázolása a memóriában.

B) A felhasználó által megadott áramkör beolvasása billentyűzetről, ebből az adatszerkezet felépítése.

C) Tetszőleges bemeneti jelkombináció beolvasása billentyűzetről és az áramkör működésének szimulálásával az ezekre adott kimenet meghatározása.

D) Az összes bemeneti kombináció automatikus előállítás és az áramkör működésének szimulálásával az ezekre adott kimenetek meghatározása.

E) Az áramkör leírásának ellenőrzése (helyesen adták-e meg?).

A lehetséges hibák:

(1) egy kapu kimenetét sehova sem kötötték, s az nem is az áramkör kimenete,

(2) egy kapu bemenetét sehova sem kötötték, s az nem is az áramkör bemenete,

(3) két (vagy több) kapu kimenetét összekötötték,

(4) van olyan kapu a hálózatban, amelyre a bemenetről nem lehet elérni,

(5) az áramkör valamelyik kimenete nem valamelyik kapu kimenete,

## A második forduló megoldása

```
10 '
20 ' Digitális áramkör működésének szimulálása
30 '
40 DEFINT A-Z: DIM CSP(30,3), ELEM(10,3), KI(20), BE(20), KAPU(6,1,1), M$(4): GOSUB 750

50 SCREEN 0,0,0
60 CLS: PRINT M$(0)
70 '
80 ' Áramkör megadása
90 '
100 COLOR 1,7: PRINT "Áramkör megadása.": SPACE$(62)
110 COLOR 4,7: PRINT "Áramkör bemeneteinek megadása.": SPACE$(49): COLOR 7,0: PRINT:
X=1
120 PRINT "A(z)"; X; ". bemenet csomópont-azonosítója (ENTER=vége)": INPUT "": BE(X):
IF BE(X)<0 OR BE(X)>30 THEN PRINT M$(1): GOTO 120 ELSE IF BE(X)<>0 THEN IF CSP(BE
(X),2)=3 THEN PRINT M$(2): GOTO 120 ELSE CSP(BE(X),2)=3: X=X+1: IF X<21 THEN 120 EL
SE MBE=20
130 IF BE(X)=0 THEN IF X=1 THEN PRINT M$(4): GOTO 120 ELSE MBE=X-1
140 CLS: PRINT M$(0): COLOR 4,7: PRINT "Az áramkör elemeinek megadása.": SPACE$(49):
COLOR 7,0: PRINT: X=1
150 PRINT "A(z)"; X; ". elem megadása.": PRINT
160 INPUT "A kapu típusa (1=INV,2=OR,3=AND,4=EXOR,5=NOR,6=NAND,ENTER=vége)": ASK:
IF ASK<0 OR ASK>6 THEN 160: ELSE IF ASK<>0 THEN ELEM(X,0)=ASK: GOTO 170 ELSE MAXEL
=X-1: GOTO 210
170 INPUT "A kapu 1. bemenetének csomópont-azonosítója.": ASK: IF ASK<1 OR ASK>30 TH
EN PRINT M$(1): GOTO 170 ELSE ELEM(X,1)=ASK: CSP(ASK,3)=1
180 IF ELEM(X,0)<>1 THEN INPUT "A kapu 2. bemenetének csomópont-azonosítója.": ASK:
IF ASK<1 OR ASK>30 THEN PRINT M$(1): GOTO 170 ELSE ELEM(X,2)=ASK: CSP(ASK,3)=1 EL
E ELEM(X,2)=ASK
190 INPUT "A kapu kimenetének csomópont-azonosítója.": ASK: IF ASK<1 OR ASK>30 THEN
PRINT M$(1): GOTO 190 ELSE ELEM(X,3)=ASK: IF CSP(ASK,2)>0 THEN PRINT M$(2): GOTO
190 ELSE CSP(ASK,2)=1
200 X=X+1: IF X<11 THEN 150 ELSE MAXEL=10
210 CLS: PRINT M$(0): COLOR 1,7: PRINT "Áramkör kimeneteinek megadása.": SPACE$(49):
PRINT: X=1: COLOR 7,0
220 PRINT "A(z)"; X; ". kimenet csomópont-azonosítója (ENTER=vége)": INPUT "": KI(X):
IF KI(X)<0 OR KI(X)>30 THEN PRINT M$(1): GOTO 220: ELSE IF KI(X)<>0 THEN CSP(KI(X)
,3)=1: X=X+1: IF X<21 THEN 220 ELSE MKI=20: ELSE IF X=1 THEN PRINT M$(3): GOTO 220 E
LSE MKI=X-1
230 PRINT: PRINT "Az adatok beolvasása befejeződött.": GOSUB 700
240 '
250 ' ellenőrzés
260 '
270 ' KAPU KIMENET ÉS BEMENET ELL.
280 CLS: PRINT M$(0): X=0: FOR N=1 TO MAXEL: IF CSP(ELEM(N,3),3)<>1 THEN PRINT "A(z) "
N". kapunak a kimenete nincs bekötve.": X=X+1
290 Q=(CSP(ELEM(N,1),2) AND 1)+(CSP(ELEM(N,2),2) AND 1): IF Q<2 THEN PRINT "A(z) "
N". kapu bemenete(i) nincs(enek) bekötve.": X=X+2-Q
300 NEXT N
310 ' KIMENET ELL.
320 FOR N=1 TO MKI: IF (CSP(KI(N),2) AND 2) THEN PRINT "A(z) "N". kimenet nem kapuki
menet!": X=X+1
330 NEXT N
340 ' BEMENET ELL.
350 FOR N=1 TO MBE: IF CSP(BE(N),3)=0 THEN PRINT "A(z) "N". bemenet nincs bekötve!":
X=X+1
360 NEXT N
370 IF X THEN PRINT: PRINT "Az áramkörben "X" hiba van. Kérem adja meg újra a kapcsol
ást!": GOSUB 700: RUN
380 ' ELÉRHETŐSÉG ELL.
390 FOR N=1 TO 30: CSP(N,0)=-1: CSP(N,1)=-1: NEXT N: FOR N=1 TO MBE: CSP(BE(N),0)=0: C
```

(6) az áramkör valamelyik bemenetét nem kötötték egy kapu bemenetére sem, (7) a korlátnak állított időegység után sem alakul ki stabil kimenet.

Tartalmazta a kiírás a felhasználható kapuk kimeneteinek táblázatait is, illetve megadott egy próbakapcsolást.

A feladat közepes nehézsége miatt a rendelkezésre álló 5 óra alatt a versenyzők 60%-ának sikerült a feladatnak legálább 50%-át megoldania.

Az alábbiakban közlünk egy BASIC-megoldást a feladatra, melyet GWBASIC-ben írtunk, de minimális átalakítással bármilyen géptípuson futtatható.

Jellemző, hogy a program (nyugodtabb körülmények között) 3 óra alatt született, s a rá adható pontszám körülbelül 92-97 pont. Ez persze annak az eredménye, hogy a feladatot egyszer már gépre vittük, illetve annak, hogy a verseny után részletesen megbeszéltük a lehetséges technikai megoldásokat.

A feladatkiírás javaslata szerint létrehoztunk egy CSP(30,3) tömböt (30 csomópont minden esetben elegendő 10 ka-

puhoz), amelynek második indexe 0 és 1 esetben a szimuláció során a csomópont aktuális értékét tárolja (szükségszerű a két fázis: az előző szimulációs lépés értékei és az aktuális értékek, gondoljunk csak az életjáték programjára), míg a 2 és 3 érték esetén megkapjuk, hogy az aktuális csomópontnak van-e be- és kimenete (0=nincs, 1=igen: kapu, 3=igen: az áramkör be- vagy kimenete). Ez a két tömb jelentősen megkönnyíti a program hibakeresését.

Szükség van még egy BE(20) és KI(20) tömbre, ezek azokra a csomópontokra mutatnak, amelyek az áramkör be-, illetve kimenetei. A be- és kimenetek számát az MBE és MKI változók jelzik.

Az áramkör elemeinek tárolására egy ELEM(10,3) tömböt hozunk létre, melynek első indexe az aktuális elem sorszámát határozza meg, míg ha a második index 0, a kapu típusát jelöli (1=inverter, 2=or, 3=and, 4=exor, 5=nor, 6=nand), ha 1, illetve 2, az aktuális kapu 1. és 2. bemenetének csomópontszámát adja meg, míg a 3 a kapu kimeneti csomópontszámát ha-

tározza meg. A program az inverter kapunál a 2. bemenetet automatikusan ugyanarra a csomópontra köti, mint az 1-est.

A KAPU(6,1,1) tömb a különböző fajtájú kapuk táblázatait tartalmazza. Az első index a kapu fajtája (1-6, itt a 0 nincs kihasználva), míg a második és harmadik index a kapura érkező két bemenetet adja (mindegyik 0 vagy 1), a tömb megfelelő eleme pedig az erre adandó kimenetet tartalmazza. (Mivel az inverter kapunál a második bemenettől függetlennek kell lennie a kimeneti értéknek, így a táblázatban x 0 vagy 1-re KAPU(1,x,0)=KAPU(1,x,1) fennáll.

Az m\$(4) tömb a gyakori üzeneteket tartalmazza.

Végezetül az összes bemenet előállításakor a program használ egy B(MBE) tömböt, melyben a bemeneteket állítja elő.

A program részekre bontva:  
10-60 Az inicializálás: a képernyő beállítás, a tömbök dimenzionálása, majd a KAPU és M\$ tömbök feltöltése (szubrutinhívás).

70-230 Az áramkör adatainak beolvasása billentyűzetről az adatstruktúrának megfelelően s a (3) hibalehetőség kiszűrése.

240-410 Ellenőrzést végző rutinok: x=a talált hibák száma.

270-300 (1) és (2) hibák kiszűrése a CSP(z,2) és CSP(z,3) tömbök segítségével

310-330 (5) hiba vizsgálata szintén a CSP(z,2) tömb segítségével

340-360 (6) hiba vizsgálata CSP(z,3) tömbbel

370 a hibaszám kiírása, ha volt hiba, visszaugrás az adatbeolvasásra

380-410 (4) hiba vizsgálata. Ez azért áll a hibaellenőrzés végén, mivel csak akkor ellenőrizhető, ha a fentebbi hibák egyike sem fordul elő a kapcsolásban.

A program itt feltölti a csomóponttömböt -1-gyel, a bemeneti csomópontokat 0-val, s végrehajtja a szimuláció egy speciális formáját: végighalad a kapukon, s ha a kapuk valamelyik bemenetén nem -1 van, akkor a kapu kimenetére 0-t ír. Ha egy kapu valamelyik bemenetén N+1 szimulációs lépés múlva is -1 áll, a kapu nem érhető el.

420-500 Szimulációs rutin, amely lekerdezi a bemeneti kombinációt, ha az ENTER, az összes bemenet előállítására ugrik, ha nem, ellenőrzi, a bemenet megfelelő-e, s végrehajtja a szimulációt N<sup>2</sup>-szer, majd ismét bemenetet kér.

510-610 A lépésenkénti szimuláció rutinja: itt M2 az aktuális CSP tömbre mutat, míg M1 az előző lépésnél meghatározott értékeket tároló CSP tömb második indexe. A szimuláció végigjárja a kapukat, és a KAPU tömb segítségével elhelyezi a megfelelő értéket a kapu kimenetén.

620-690 Az összes bemeneti kombináció előállítása. A B tömb bináris léptetését a 670-es sor végzi. A rutin elején dimenzionált B tömböt a rutin végén töröljük.

700-710 Egy gomb lenyomására váró szubrutin.

720-900 A KAPU tömb és az üzenetek tömbjelének (m\$) feltöltése.

```

SP(BE(N),1)=0:NEXT N
400 LEP=MAXEL+1:H=1:GOSUB 540:FOR N=1 TO MAXEL:IF CSP(ELEM(N,1),M2)=-1 OR CSP(ELEM(N,2),M2)=-1 THEN PRINT"A(z)"N". kapu nem érhető el a bemenet(ek)ről!":X=X+1
410 NEXT N:IF X THEN 370
420
430 ' Szimuláció
440
450 CLS:PRINT M$(0)
460 COLOR 7,1:PRINT" Szimuláció"SPACE$(69):COLOR 7,0:PRINT:H=0
470 PRINT:INPUT"Bemeneti kombináció (ENTER=összes bemenet előállítása):",A$:IF A$="" THEN 650
480 IF LEN(A$)<>MBE THEN PRINT MBE"bemenet van!":GOTO 520
490 FOR N=1 TO MBE:ASK=VAL(MID$(A$,N,1)):IF ASK>1 THEN PRINT"Az áramkör bemenete csak 0 vagy 1 lehet!":GOTO 470 ELSE CSP(BE(N),0)=ASK:CSP(BE(N),1)=ASK:NEXT N
500 LEP=MAXEL*MAXEL:GOSUB 540:GOTO 470
510
520 ' Lépésenkénti szimuláció
530
540 M1=0:M2=1:C=0:FOR L=1 TO LEP:FOR N=1 TO MAXEL:B1=CSP(ELEM(N,1),M1):B2=CSP(ELEM(N,2),M1):IF B1=-1 THEN IF B2=-1 THEN 570 ELSE 560 ELSE IF B2=-1 THEN 560
550 C=KAPU(ELEM(N,0),B1,B2)
560 CSP(ELEM(N,3),M2)=C
570 NEXT N
580 Y=0:FOR N=1 TO 30:Y=Y-(CSP(N,0)=CSP(N,1)):NEXT N:IF Y<30 THEN SWAP M1,M2:NEXT L:IF H THEN RETURN ELSE PRINT "Nem alakul ki stabil kimenet!":GOTO 610
590 IF H THEN RETURN
600 FOR N=1 TO MKI:PRINT RIGHT$(STR$(CSP(KI(N),M2)),1)",":NEXT N:PRINT CHR$(29)
"
610 FOR N=1 TO 30:CSP(N,0)=0:CSP(N,1)=0:NEXT N:RETURN
620
630 ' Összes bemeneti kombináció előállítása
640
650 DIM B(MBE):V2=0:FOR V=0 TO 2^MBE-1:V2=V2+1
660 FOR N=1 TO MBE:PRINT RIGHT$(STR$(B(N)),1)",":CSP(BE(N),0)=B(N):CSP(BE(N),1)=B(N):NEXT N:PRINT CHR$(29) " "":LEP=MAXEL*MAXEL:GOSUB 540
670 FOR N=MBE TO 1 STEP -1:IF B(N)=1 THEN B(N)=0:NEXT N ELSE B(N)=1
680 IF V2=20 THEN GOSUB 700:GOSUB 850:V2=0
690 NEXT V:ERASE B:GOTO 470
700 PRINT"Nyomjon le egy gombot!"
710 A$=INKEY$:IF A$="" THEN 710 ELSE RETURN
720
730 ' Logikai kapuk kimeneti értékeinek beolvasása
740
750 FOR N=1 TO 6:FOR M=0 TO 1:FOR B=0 TO 1:READ KAPU(N,M,B):NEXT B,M,N
760 DATA 1,1,0,0
770 DATA 0,1,1,1
780 DATA 0,0,0,1
790 DATA 0,1,1,0
800 DATA 1,0,0,0
810 DATA 1,1,1,0
820
830 ' Gyakori üzenetek beolvasása
840
850 FOR N=0 TO 4:READ M$(N):NEXT N:RETURN
860 DATA " Digitális áramkör működésének szimulálása - CBT Software (c)19
870 DATA"A csomópontszám-azonosítónak 1 és 30 között kell lennie."
880 DATA"A csomópontnak már van bemenete!"
890 DATA"Kell,hogy az áramkör rendelkezzen kimenettel!"
900 DATA"Kell,hogy az áramkör rendelkezzen bemenettel!"

```

# Gyorstöltők a C64-en

A számítógép és a külső egységek közötti adatforgalom a soros vonalon keresztül történik. A soros vonal a következő vezetékeket tartalmazza:

- GND (Ground) — Rendszerföld
- SRQ (Service Request) — Kiszolgáláskérés
- ATN (Attention) — Figyelemfelhívás. A vezérlő aktivizálja ezt a vonalat, ha utasítást akar kiadni
- CLK (CLOCK) — Érvényességjelző
- DATA — Adatvonal

Ezek a vonalak a számítógépben a következőképpen érhetőek el:

- \$DD00 3. bit ATN-kimenet
- 4. bit CLOCK-kimenet
- 5. bit DATA-kimenet
- 6. bit CLOCK-bemenet
- 7. bit DATA-bemenet

A lemezegységben pedig:

- \$1800 0. bit DATA-bemenet
- 1. bit DATA-kimenet
- 2. bit CLOCK-bemenet
- 3. bit CLOCK-kimenet
- 4. bit ATNA
- 7. bit ATN-bemenet

Alapértelmezésben az adattovábbítás az adatvonalon bitenként történik, a megfelelő időzítések betartásával. Köztudott, hogy ez meglehetősen lassú. Gyorsításra ad lehetőséget az a tény, hogy a DATA-vonal mellett a CLOCK-vonal is mindkét irányban hozzáférhető, és le lehet faragni az időzítésekből is.

A lemezegység programozásához a következők ismeretére van szükségünk:

- \$0000 — \$07FF RAM
- \$1800 — \$180F VIA #1 Sorosvonal-kezelő
- \$1C00 — \$1C0F VIA #2 Lemezvezérlő
- \$C000 — \$FFFF ROM

A lemezegységben az író-, olvasó-, fejeziccionáló, szektorkereső funkciókat a megszakítási rendszerrel (JOB) végezhetjük el a legegyszerűbben. A JOB öt puffertérülettel és a hozzájuk kapcsolódó munkaváltozókkal rendelkezik.

PUFFER	CIM	SÁV	SZEKTOR	JOB—CIM
#0	\$0300—\$03FF	\$06	\$07	\$00
#1	\$0400—\$04FF	\$08	\$09	\$01
#2	\$0500—\$05FF	\$0A	\$0B	\$02
#3	\$0600—\$06FF	\$0C	\$0D	\$03
#4	\$0700—\$07FF	\$0E	\$0F	\$04
#5	Nincs RAM!	\$10	\$11	\$05

A következő JOB vezérlő kódok vannak:

\$80 READ A megadott blokk olvasása a puffertérületbe  
 \$90 WRITE A puffertérület kiírása a megadott blokkba

		0002		==	\$C000
		0003		.OBJ	M
C000	A9 06	0004	D0700	LDA	#\$05
C002	85 31	0005		STA	\$31
C004	20 0A F5	0006		JSR	\$(F5)A
C007	50 FE	0007	D0707	BVC	D0707
C009	BB	0008		CLV	
C00A	AD 01 1C	0009		LDA	\$(1C)01
C00D	91 30	0010		STA	\$(30),Y
C00F	C8	0011		INY	
C010	D0 F5	0012		BNE	D0707
C012	A0 BA	0013		LDY	#\$BA
C014	50 FE	0014	D0714	BVC	D0714
C016	B8	0015		CLV	
C017	AD 01 1C	0016		LDA	\$(1C)01
C01A	99 00 01	0017		STA	\$(01)00,Y
C01D	C8	0018		INY	
C01E	D0 F4	0019		BNE	D0714
C020	20 E0 F8	0020		JSR	\$(F8)E0
C023	A5 38	0021		LDA	\$38
C025	C5 47	0022		CMP	\$47
C027	F0 04	0023		BEQ	D072D
C029	A9 04	0024		LDA	#\$04
C02B	D0 5F	0025		BNE	D078C
C02D	20 E9 F5	0026	D072D	JSR	\$(F5)E9
C030	C5 3A	0027		CMP	\$3A

C032	F0 04	0028		BEQ	D0738
C034	A9 05	0029		LDA	#\$05
C036	D0 54	0030		BNE	D078C
C038	B1 30	0031	D0738	LDA	\$(30),Y
C03A	D0 03	0032		BNE	D073F
C03C	EE 01 06	0033		INC	\$(06)01
C03F	B1 30	0034	D073F	LDA	\$(30),Y
C041	AA	0035		TAX	
C042	2C 00 18	0036	D0742	BIT	\$(18)00
C045	10 FB	0037		BPL	D0742
C047	A9 10	0038		LDA	#\$10
C049	8D 00 18	0039		STA	\$(18)00
C04C	2C 00 18	0040	D074C	BIT	\$(18)00
C04F	30 FB	0041		BMI	D074C
C051	8A	0042		TXA	
C052	4A	0043		LSR	A
C053	4A	0044		LSR	A
C054	4A	0045		LSR	A
C055	4A	0046		LSR	A
C056	8D 00 18	0047		STA	\$(18)00
C059	0A	0048		ASL	A
C05A	29 0F	0049		AND	\$(0F)
C05C	8D 00 18	0050		STA	\$(18)00
C05F	8A	0051		TXA	
C060	29 0F	0052		AND	\$(0F)
C062	8D 00 18	0053		STA	\$(18)00
C065	0A	0054		ASL	A
C066	29 0F	0055		AND	\$(0F)
C068	8D 00 18	0056		STA	\$(18)00
C06B	A2 0F	0057		LDX	\$(0F)
C06D	EA	0058		NOP	
C06E	8E 00 18	0059		STX	\$(18)00
C071	AD 00 06	0060		LDA	\$(06)00
C074	F0 1C	0061		BEQ	D0792
C076	C8	0062		INY	
C077	D0 C6	0063		BNE	D073F
C079	A9 05	0064		LDA	#\$05
C07B	85 0C	0065		STA	\$(0C)
C07D	AD 01 06	0066		LDA	\$(06)01
C080	85 0F	0067		STA	\$(0F)
C082	B1 30	0068		LDA	\$(30),Y
C084	C5 0E	0069		CMP	\$(0E)
C086	85 0E	0070		STA	\$(0E)
C088	F0 05	0071		BEQ	D078F
C08A	A9 01	0072		LDA	#\$01
C08C	4C 69 F9	0073	D078C	JMP	\$(F9)69
C08F	4C 04 07	0074	D078F	JMP	\$(07)04
C092	C8	0075	D0792	INY	
C093	CC 01 06	0076		CPY	\$(06)01
C096	L0 A7	0077		BNE	D073F
C098	A9 7F	0078		LDA	#\$7F
C09A	D0 F0	0079		BNE	D078C
C09C	00	0080		BRK	
C09D	00	0081		BRK	
C09E	00	0082		BRK	
C09F	00	0083		BRK	
C0A0	00	0084		BRK	
C0A1	00	0085		BRK	
C0A2	00	0086		BRK	
C0A3	00	0087		BRK	
C0A4	00	0088		BRK	
C0A5	00	0089		BRK	
C0A6	00	0090		BRK	
C0A7	00	0091		BRK	
C0A8	00	0092		BRK	
C0A9	00	0093		BRK	
C0AA	00	0094		BRK	
C0AB	A9 10	0095	D07AB	LDA	#\$10
C0AD	8D 05 1C	0096		STA	\$(1C)05
C0B0	A5 18	0097		LDA	\$(18)
C0B2	85 0E	0098		STA	\$(0E)
C0B4	A5 19	0099		LDA	\$(19)
C0B6	85 0F	0100		STA	\$(0F)
C0B8	A9 05	0101	D07BB	LDA	#\$05
C0BA	85 0C	0102		STA	\$(0C)
C0BC	A9 E0	0103	D07BC	LDA	#\$E0
C0BE	85 04	0104		STA	\$(04)
C0C0	A5 04	0105	D07C0	LDA	\$(04)
C0C2	30 FC	0106		BMI	D07C0
C0C4	C9 7F	0107		CMP	\$(7F)C9
C0C6	F0 30	0108		BEQ	D07FB
C0C8	C9 02	0109		CMP	\$(02)C9
C0CA	90 EC	0110		BCC	D07BB
C0CC	C6 0C	0111		DEC	\$(0C)
C0CE	10 EC	0112		BPL	D07BC
C0D0	A4 0C	0113		LDY	\$(0C)A4
C0D2	C8	0114		INY	
C0D3	D0 07	0115		BNE	D07DC
C0D5	A9 C0	0116		LDA	#\$C0
C0D7	20 8E D5	0117		JSR	\$(D5)8E
C0DA	C6 0C	0118		DEC	\$(0C)
C0DC	A4 0C	0119	D07DC	LDY	\$(0C)A4
C0DE	C0 FB	0120		CPY	\$(FB)C0
C0E0	B0 DA	0121		BCS	D07BC
C0E2	A9 60	0122		LDA	#\$60
C0E4	8D 71 07	0123		STA	\$(07)71
C0E7	A9 FF	0124		LDA	\$(FF)A9
C0E9	20 41 07	0125		JSR	\$(07)41
C0EC	A9 3A	0126		LDA	\$(3A)A9
C0EE	8D 05 1C	0127		STA	\$(1C)05
C0F1	A5 04	0128		LDA	\$(04)A5
C0F3	A2 04	0129		LDX	\$(04)A2
C0F5	4C 0A E6	0130		JMP	\$(E6)0A
C0F8	A9 3A	0131	D07F8	LDA	\$(3A)A9
C0FA	8D 05 1C	0132		STA	\$(1C)05
C0FD	4C 48 D0	0133		JMP	\$(D0)48
		0134			
C100	A0 A0 A0	0135	TABLE	.BYTE	\$(A0)\$(A0)\$(A0)\$(A0)\$(A0)\$(A0)\$(A0)\$(A0)
C103	A0 A0 A0				
C106	A0 A0			.BYTE	\$(50)\$(50)\$(50)\$(50)\$(50)\$(50)\$(50)\$(50)
C108	50 50 50	0136			
C10B	50 50 50			.BYTE	\$(0A)\$(0A)\$(0A)\$(0A)\$(0A)\$(0A)\$(0A)\$(0A)
C10E	50 50				
C110	0A 0A 0A	0137		.BYTE	\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)
C113	0A 0A 0A				
C116	0A 0A			.BYTE	\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)\$(05)
C118	05 05 05	0138			

```

C11B 05 05 05
C11E 05 05

0139 ;
C120 FF FF FF .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C123 FF FF FF
C126 FF FF
C128 FF FF FF 0141 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C12B FF FF FF
C12E FF FF
C130 FF FF FF 0142 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C133 FF FF FF
C136 FF FF
C138 FF FF FF 0143 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C13B FF FF FF
C13E FF FF

0144 ;
C140 20 20 20 0145 .BYTE $20,$20,$20,$20,$20,$20,$20,$20
C143 20 20 20
C146 20 20
C148 10 10 10 0146 .BYTE $10,$10,$10,$10,$10,$10,$10,$10
C14B 10 10 10
C14E 10 10
C150 02 02 02 0147 .BYTE $02,$02,$02,$02,$02,$02,$02,$02
C153 02 02 02
C156 02 02
C158 01 01 01 0148 .BYTE $01,$01,$01,$01,$01,$01,$01,$01
C15B 01 01 01
C15E 01 01

0149 ;
C160 FF FF FF 0150 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C163 FF FF FF
C166 FF FF
C168 FF FF FF 0151 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C16B FF FF FF
C16E FF FF
C170 FF FF FF 0152 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C173 FF FF FF
C176 FF FF
C178 FF FF FF 0153 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C17B FF FF FF
C17E FF FF

0154 ;
C180 80 80 80 0155 .BYTE $80,$80,$80,$80,$80,$80,$80,$80
C183 80 80 80
C186 80 80
C188 40 40 40 0156 .BYTE $40,$40,$40,$40,$40,$40,$40,$40
C18B 40 40 40
C18E 40 40
C190 08 08 08 0157 .BYTE $08,$08,$08,$08,$08,$08,$08,$08
C193 08 08 08
C196 08 08
C198 04 04 04 0158 .BYTE $04,$04,$04,$04,$04,$04,$04,$04
C19B 04 04 04
C19E 04 04

0159 ;
C1A0 FF FF FF 0160 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C1A3 FF FF FF
C1A6 FF FF
C1A8 FF FF FF 0161 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C1AB FF FF FF
C1AE FF FF
C1B0 FF FF FF 0162 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C1B3 FF FF FF
C1B6 FF FF
C1B8 FF FF FF 0163 .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
C1BB FF FF FF
C1BE FF FF

0164 ;
C1C0 00 00 00 0165 .BYTE $00,$00,$00,$00,$00,$00,$00,$00
C1C3 00 00 00
C1C6 00 00
C1C8 00 00 00 0166 .BYTE $00,$00,$00,$00,$00,$00,$00,$00
C1CB 00 00 00
C1CE 00 00
C1D0 00 00 00 0167 .BYTE $00,$00,$00,$00,$00,$00,$00,$00
C1D3 00 00 00
C1D6 00 00
C1D8 00 00 00 0168 .BYTE $00,$00,$00,$00,$00,$00,$00,$00

```

```

C1DB 00 00 00
C1DE 00 00

0169 ;
C1E0 A5 A4 0170 BYTEIN LDA $A4
C1E2 8D 00 DD 0171 STA $DD00
C1E5 AD 00 DD 0172 HWAIT LDA $DD00
C1E8 10 FB 0173 BPL HWAIT
C1EA AD 12 D0 0174 RWAIT LDA $D012
C1ED C9 31 0175 CMP $*31
C1EF 90 06 0176 BCC BITS
C1F1 29 06 0177 AND $*06
C1F3 C9 02 0178 CMP $*02
C1F5 F0 F3 0179 BEQ RWAIT
C1F7 A5 A5 0180 LDA $A5
C1F9 8D 00 DD 0181 STA $DD00
C1FC EA 0182 NOP
C1FD EA 0183 NOP
C1FE EA 0184 NOP
C1FF EA 0185 NOP
C200 EA 0186 NOP
C201 EA 0187 NOP
C202 EA 0188 NOP
C203 EA 0189 NOP
C204 EA 0190 NOP
C205 EA 0191 NOP
C206 AC 00 DD 0192 LDY $DD00
C209 B9 00 C1 0193 LDA TABLE,Y
C20C AC 00 DD 0194 LDY $DD00
C20F 19 08 C1 0195 ORA TABLE+8,Y
C212 AC 00 DD 0196 LDY $DD00
C215 19 10 C1 0197 ORA TABLE+16,Y
C218 AC 00 DD 0198 LDY $DD00
C21B 19 18 C1 0199 ORA TABLE+24,Y
C21E 60 0200 EXIT RTS
0201 ;
C21F A0 00 0202 LOAD LDY $*00
C221 84 9D 0203 STY $*9D
C223 B1 BB 0204 LDA ($BB),Y
C225 C9 24 0205 CMP $*#*
C227 D0 04 0206 BNE SLOAD
C229 98 0207 TYA
C22A 4C A5 F4 0208 JMP $*F4A5
0209 ;
C22D 84 93 0210 SLOAD STY $*93
C22F A5 BA 0211 LDA $*BA
C231 C9 04 0212 CMP $*04
C233 90 E9 0213 BCC EXIT

```

```

C235 A5 B7 0214 LDA $*B7
C237 F0 E5 0215 BEQ EXIT
C239 A5 BA 0216 LDA $*BA
C23B 85 B8 0217 STA $*B8
C23D 20 AF F5 0218 JSR $*F5AF
C240 20 33 F3 0219 JSR $*F333
C243 A6 B9 0220 LDY $*B9
C245 86 02 0221 STX $*02
C247 A9 60 0222 LDA $*60
C249 85 B9 0223 STA $*B9
C24B 20 D5 F3 0224 JSR $*F3D5
C24E A5 BA 0225 LDA $*BA
C250 20 09 ED 0226 JSR $*ED09
C253 A5 B9 0227 LDA $*B9
C255 20 C7 ED 0228 JSR $*EDC7
C258 20 13 EE 0229 JSR $*EE13
C25B A5 90 0230 LDA $*90
C25D 4A 0231 LSR A
C25E 4A 0232 LSR A
C25F 90 03 0233 BCC NOERR
C261 4C 04 F7 0234 JMP $*F704
C264 20 D2 F5 0235 JSR $*F5D2
C267 A9 00 0236 LDA $*00
C269 85 A4 0237 STA $*A4
C26B 20 44 C3 0238 JSR $*DRCOM
C26E A9 57 0239 LDA $*#*
C270 20 DD ED 0240 JSR $*EDDD
C273 A5 A4 0241 LDA $*A4
C275 20 DD ED 0242 JSR $*EDDD
C278 A9 07 0243 LDA $*07
C27A 20 DD ED 0244 JSR $*EDDD
C27D A9 20 0245 LDA $*20
C27F 20 DD ED 0246 JSR $*EDDD
C282 A4 A4 0247 LDY $*A4
C284 18 0248 CLC
C285 A5 A4 0249 LDA $*A4
C287 69 20 0250 ADC $*20
C289 85 A4 0251 STA $*A4
C28B B9 00 C0 0252 LDA $*D0700,Y
C28E 20 DD ED 0253 JSR $*EDDD
C291 C8 0254 INY
C292 C4 A4 0255 CPY $*A4
C294 D0 F5 0256 BNE TLP
C296 20 FE ED 0257 JSR $*EDFE
C299 A5 A4 0258 LDA $*A4
C29B D0 CE 0259 BNE TRLOOP
C29D 20 44 C3 0260 JSR $*DRCOM

```

```

C2A0 A9 45 0261 LDA $*#E
C2A2 20 DD ED 0262 JSR $*EDDD
C2A5 A9 AB 0263 LDA $*#AB
C2A7 20 DD ED 0264 JSR $*EDDD
C2AA A9 07 0265 LDA $*#07
C2AC 20 DD ED 0266 JSR $*EDDD
C2AF 20 FE ED 0267 JSR $*EDFE
C2B2 EA 0268 NOP
C2B3 AD 00 DD 0269 LDA $*D000
C2B6 85 93 0270 STA $*93
C2B8 29 07 0271 AND $*07
C2BA 85 A5 0272 STA $*A5
C2BC 09 08 0273 ORA $*08
C2BE 85 A4 0274 STA $*A4
C2C0 AD 15 D0 0275 LDA $*D015
C2C3 85 97 0276 STA $*97
C2C5 A9 00 0277 LDA $*#00
0278
C2C7 78 0279 SEI
C2C8 20 6D C3 0280 JSR $*SMIN
C2CB A1 04 0281 LDY $*04
C2CD 20 E0 C1 0282 JSR $*BYTEIN
C2D0 F0 2B 0283 BEQ $*FIRSTLAST
C2D2 C9 FF 0284 CMP $*FF
C2D4 D0 06 0285 BNE $*NOBERR
C2D6 A9 02 0286 LDA $*#02
C2D8 85 90 0287 STA $*90
C2DA D0 42 0288 BNE $*ENDLOAD
C2DC 20 E0 C1 0289 JSR $*BYTEIN
C2DF E0 02 0290 CPX $*02
C2E1 F0 03 0291 BEQ $*BLKLP1
C2E3 20 58 C3 0292 JSR $*LOADD
C2E6 20 E0 C1 0293 JSR $*BYTEIN
C2E9 EA 0294 NOP
C2EA 20 75 C3 0295 JSR $*STORAGE
C2ED E6 AE 0296 INC $*AE
C2EF F0 07 0297 BEQ $*INA
C2F1 E8 0298 INP
C2F2 D0 F2 0299 BNE $*BLKLP1
C2F4 A2 02 0300 LDY $*#02
C2F6 D0 D5 0301 BNE $*STBLOCK
C2F8 E6 AF 0302 INA
C2FA 18 0303 CLC
C2FB 90 F4 0304 BCC $*INP
0305
C2FD 20 E0 C1 0306 $*FIRSTLAST JSR $*BYTEIN
C300 E0 02 0307 PHX $*#02
C302 08 0308 CPY $*08
C303 AA 0309 TAX
C304 28 0310 FLP
C305 F0 05 0311 BEQ $*LAST
C307 20 58 C3 0312 JSR $*LOADD
C30A CA 0313 DEX
C30B CA 0314 DEX
C30C CA 0315 LAST DEX
C30D CA 0316 DEX
C30E 20 E0 C1 0317 JSR $*BLKLP2
C311 EA 0318 NOP
C312 20 75 C3 0319 JSR $*STORAGE
C315 E6 AE 0320 INC $*AE
C317 D0 02 0321 BNE $*NOI
C319 E6 AF 0322 INC $*AF
C31B CA 0323 NOI DEX
C31C D0 F0 0324 BNE $*BLKLP2
0325
C31E A9 37 0326 $*ENDLOAD LDA $*#37
C320 85 01 0327 STA $*01
C322 20 46 F6 0328 JSR $*IF646
C325 A5 97 0329 LDA $*97
C327 8D 15 D0 0330 STA $*D015
C32A A5 93 0331 LDA $*93
C32C 8D 00 DD 0332 STA $*D000
C32F A9 00 0333 LDA $*#00
C331 85 93 0334 STA $*93
C333 A5 90 0335 LDA $*90
C335 4A 0336 LSR A
C336 4A 0337 LSR A
C337 A9 40 0338 LDA $*#40
C339 05 90 0339 ORA $*90
C33B 85 90 0340 STA $*90
C33D A6 AE 0341 LDY $*AE
C33F A4 AF 0342 LDY $*AF
C341 A9 1D 0343 LDA $*#1D
C343 60 0344 RTS
0345
C344 A5 BA 0346 $*DRCOM LDA $*BA
C346 20 0C ED 0347 JSR $*ED0C
C349 A9 6F 0348 LDA $*#6F
C34B 20 B9 ED 0349 JSR $*EDB9
C34E A9 40 0350 LDA $*#*
C350 20 DD ED 0351 JSR $*EDDD
C353 A9 2D 0352 LDA $*#*
C355 4C DD ED 0353 JMP $*EDDD
0354
C358 20 E0 C1 0355 $*LOADD JSR $*BYTEIN
C35B 85 AE 0356 STA $*AE
C35D 20 E0 C1 0357 JSR $*BYTEIN
C360 A4 02 0358 LDY $*02
C362 D0 06 0359 BNE $*NOADD
C364 A5 C3 0360 LDA $*#C3
C366 85 AE 0361 STA $*AE
C368 A5 C4 0362 LDA $*#C4
C36A 85 AF 0363 STA $*AF
C36A 60 0364 RTS
0365
C36D 8D 15 D0 0366 $*SMIN STA $*D015
C370 A9 35 0367 LDA $*#35
C372 85 01 0368 STA $*#01
C374 60 0369 RTS
0370
C375 C6 01 0371 $*STORAGE DEC $*01
C377 A0 00 0372 LDY $*#00
C379 91 AE 0373 STA $*($AE),Y
C37B E6 01 0374 INC $*01
C37D 60 0375 RTS
0376
C37E A9 1F 0377 $*START JSR $*#<LOAD
C380 8D 30 0378 STA $*#0330
C383 A9 C2 0379 LDA $*#>LOAD
C385 8D 31 0380 STA $*#0331
C388 A9 00 0381 LDA $*#00
C38A 85 9D 0382 STA $*9D
C38C 60 0383 RTS
0384
C385 0385 $*END

```

\$A0	VERIFY	A puffer összehasonlítása a megadott blokkal
\$B0	SEEK	A megadott sávra pozicionál
\$C0	BUMP	A fej a #1 sávra pozicionálja (fizikai hatar)
\$D0	JUMP	Program végrehajtása a pufferben
\$E0	EXECUTE	Program végrehajtása a puffer elejéről, a fej a megadott helyen áll, a sűrűség beállítva a 6502 diszk kontroller módban dolgozik

A lemezen az információ egy kódolt (GCR) formája található a megfelelő szektorazonosító és ellenőrző bájtokkal bővítve. A kódolás során az adatbájt 8 bitjéből 10 bitet állítunk elő, de az eredményként előállított kódban nem lehet kettőnél több egymás melletti nulla. Erre a mágneses adatrögzítés miatt van szükség, ugyanis a fej csak "1"-eseket tud kiírni a lemezre (a mágnesezési irány megváltoztatásával), a "0"-kat az "1"-esek között eltelt időből, a shiftregiszter segítségével számítja ki egy áramkör.

Bemutatunk egy lehetséges gyorsítási módszert a programok töltésére. A program egy ismert gyorsító kismértékű módosításával készült. Választásunk azért esett rá, mert kismértékű átalakításával töltés közben zenélhetünk, scrollozhatunk és sprite-okat is kezelhetünk.

#### A lemez meghajtóban futó program

4—5	A 3. puffer kijelölése
6	Az adatblokk fejlécének megkeresése
7	Bájt beolvasásának kivárása
8	Bájt kész jelzésének törlése
9	Bájt a fejről
10	Tárolása a pufferben
11—12	Ismétlés 256-szor
14—19	A maradék 69 bájt beolvasása a veremterületre
20	GCR-dekódolás
21—23	Az adatblokk ID = \$07 ?
24—25	Nem. Hibajelzés
26	Adatpuffer paritásának számítása
27—28	Megegyezik a lemezen tárolttal?
29—30	Nem. Hibajelzés
31—32	A fájl utolsó blokkja?
33	Igen. Az értékes bájtok számának kiszámítása.
Az adatblokk átküldése	
34—35	A következő bájt felvétele
36—41	Handshake (szinkronizáció)
42—46	A felső négy bit az alsó négy bitre kerül
47	Az eredeti bájt 5. és 7. bitjének átküldése
48—50	Az eredeti bájt 4. és 6. bitjének átküldése
51—53	Az eredeti bájt 1. és 3. bitjének átküldése
54—56	Az eredeti bájt 0. és 2. bitjének átküldése
57—59	A bájt átvitelének vége
60—61	A fájl utolsó blokkja?
62—63	Nem. Az átküldés folytatása
64—65	A próbálkozások száma öt
66—67	A következő blokk szektorszámának tárolása
68—70	Megváltozik a sávszám?
71	Nem. A következő blokk olvasása
72—73	Vissza a lemezvezérlőhöz
75—76	Az utolsó blokk utolsó bájtja volt?
77	Nem. Az átküldés folytatása
78—79	Vissza a lemezvezérlőhöz, egy szabadon választott fájl végjellel
80—94	Szabad terület
Itt indul a program	
95—96	A fejléptetés gyorsítása
97—100	A kezdő sáv, szektorszám átadása a 4-es puffer vezérlőterületére
101—102	A próbálkozások száma öt
103—104	A 4-es pufferben található program futtatása
105—106	A futás végének megvárása
107—108	Ha az utolsó blokk volt, akkor ugrás a kilépésre
109—110	Ha nem történt hiba, akkor a következő blokk olvasása
111—112	Hiba esetén újra megpróbálni (öt próbálkozás)
113—121	Ha még mindig sikertelen a beolvasás, akkor BUMP és újabb próbák
122—125	Az átküldő rutin módosítása (RTS), hibajelző bájt (\$FF) átküldése
126—127	A fejléptetési idő visszaállítása

128—130	Hibaüzenet
131—132	A fejléptetési idő visszaállítása
133	A BAM beolvasása
A számítógépben futó program	
135—168	Maszk tábla
A \$FF-et tartalmazó területek értékei közömbösek a program számára, méreteik viszont kötöttek. Soronként azért szerepel 8 egyforma bájt, hogy bármelyik videoszeletet használhassuk töltés közben. A visszamaszkolás invertált bitekkel történik. Bájtokat beolvasó rutin	
170—181	Handshake. (Szinkronizáció. A képernyőkezelés miatt nem minden rasztorsorban van megengedve a töltés.)
182—191	Időzítés
192—193	Az eredeti bájt 5. és 7. bitjének vétele
194—195	Az eredeti bájt 4. és 6. bitjének vétele
196—197	Az eredeti bájt 1. és 3. bitjének vétele
198—199	Az eredeti bájt 0. és 2. bitjének vétele
Töltéskor ide kerül a vezérlés	
202—203	A program mód állítása. (Nincsenek üzenetek)
204—208	A tartalomjegyzék töltése az eredeti rutin feladata
210	Load
211—215	Hibás paraméter esetén kilépés
216—217	Logikai fájlok száma = egység szám
218	SEARCHING FOR ... üzenet kiírása (202—203-as sorral letiltva)
219	Az aktív B/K csatorna lezárása
220—221	A másodlagos cím mentése
222—224	Fájl megnyitása
225—226	TALK küldése
227—228	Másodlagos cím küldése
229	Vonal olvasása
230—234	Hiba esetén FILE NOT FOUND
235	LOADING üzenet (202—203-as sorral letiltva)
236—237	Munkaváltozó törlése
238	"M—" küldése
239—240	"W" küldése
242—244	Cím küldése
245—246	\$20 db bájtot küldünk
247—251	A következő csomag kezdőcímének kiszámítása
252—256	\$20 bájt küldése
257	UNLISTEN küldése
258—259	Ha még van a drive programból (256 bájt), akkor folytatás
260	"M—" küldése
261—262	"E" küldése
263—266	Indító cím (\$07AB) küldése
267	UNLISTEN küldése
268—270	\$DD00 mentése
271—272	Videoszelet + TXD mentése
273—274	Videoszelet + TXD + ATN mentése
275—276	Sprite-engedélyezés mentése
277—280	Sprite-tiltás és KERNAL + BASIC kikapcsolása
281	Az első 4 bájt értéktelen az első blokknál
282—283	Egyblokkos program?
284—285	Hiba?
286—288	Igen. Töltés vége
289—292	Ha az első blokk, akkor ugrás a kezdő cím beolvasására
293—295	A beolvasott bájt tárolása
296—304	A következő tárcím, teljes blokk beolvasása, ugrás a következő blokk olvasásához
306—316	Az utolsó (2) vagy az első és utolsó (4) blokknál az értéktelen bájtok kiszámolása
317—324	Az egész blokk beolvasása
326—327	BASIC + KERNAL visszakapcsolása
328	CLOSE
329—330	Sprite-engedélyezés
331—332	\$DD00 visszairása
333—334	LOAD kapcsoló
335—340	Státusz beállítása
341—342	Töltési végcím
343—344	Program vége
346—353	"M—" üzenet küldése
355—364	Kezdő cím beírása
366—369	Sprite + KERNAL + BASIC kikapcsolása
371—375	64 k RAM bekapcsolása
377—383	LOAD vektor átírása a saját rutinra
Ezen a címen kell indítani a programot	

# FELADATOK

## – MEGOLDÁSOK

Sorozatunkat elsősorban középiskolásoknak szánjuk, de reméljük, hogy minden olvasónknak tanulási lehetőséget és szórakozást nyújt.

A feladatok a Nemes Tihamér országos számítástechnikai verseny színvonalának felelnek meg. Minden esetben olyan választunk, amely röviden, gyorsan megoldható, de a megoldáshoz ötletre van szükség. A megoldást mindig a következő számban közöljük.

Mivel változatosságra törekszünk, különböző programozási nyelveket használunk. Az is előfordul, hogy egy feladatra több programnyelven is közlünk megoldást, ezzel is elősegítve az ismeretszerzést.

A szerkesztőség várja az olvasók, a versenyzők leveleit. A legötletesebb program beküldőjét könyvtalvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

### 15. feladat: Részvektor összege

Egy program bemenete egy  $N$  valós számból álló egydimenziós tömb (vektor). A megírandó programnak azt az össze-függő részvektort kell kiválasztania, amelyben az elemek összege a legnagyobb. Törekedjünk arra, hogy a program nagy  $N$  értékekre is gyors legyen!

#### Megoldás

A sorozatban eddig megjelentektől eltérően most a feladatot különböző megoldásainak hatékonyságát hasonlítjuk össze. Négy megoldást ismertetünk, fokozatosan haladva az egyre jobb felé.

A megoldást adó szubrutinok tesztelésére szolgál a közös rutinokat tartalmazó keretprogram. A Turbo Pascal 5.0 nyelven IBM PC-re megírt program két szubrutint tartalmaz. Ezek közül a *RandomVektor* a vektort tölti fel véletlenül választott értékekkel. Az értéktartomány  $-1000$ -tól  $+1000$ -ig terjed. Ilyen értékek mellett a feladatot megoldó szubrutinok működése jól követhető.

A *Kiír* nevű, másik szubrutin a kiválasztott részvektor megjelenítését végzi. A képernyőn sorokra tördelve jeleníti meg a vektort, és kiemeli a kiválasztott részvektort. Az ellenőrzést segíti, hogy a részvektor összegét is kiszámítja.

Ezután nézzük az algoritmusokat. Mind a négy algoritmus azonos fejlécű szubrutinként kapcsolódik a tesztelő keretprogramhoz. A keretprogram sorban meghívja őket, majd kiírja a kapott eredményt.

30	-40	51	20	-10	40	-89	-98	-11
----	-----	----	----	-----	----	-----	-----	-----

3 \_\_\_\_\_ 6

A megoldás rendkívül egyszerű lenne, ha tudnánk, hogy az input vektorban minden szám pozitív. Ekkor a maximális összegű részvektor maga az input vektor lenne. Gond akkor van, ha negatív számok is szerepelnek: vajon érdemes-e bevinnünk egy negatív számot a vektorba, abban reménykedve, hogy utána megint pozitív számok következnek, és kompenzálják a csökkenést? Nyilvánvaló, hogy ha mindegyik szám negatív, akkor a legnagyobb összegű részvektor a legnagyobb számot tartalmazó egyelemű vektor.

A feladatot legkézenfekvőbb megoldása az, hogy az összes lehetséges részvektorok kiszámítjuk az összegét, és ezek közül választjuk ki a legnagyobbat (1. algoritmus). Rövid, egyszerű, könnyen érthető programhoz jutunk, amely viszont rendkívül lassú. A külső ciklus pontosan elemszámra hajtódik végre, a belső ciklus az elemszám négyzetével arányos lépést igényel, az algoritmus pedig az elemszám köbével arányos ideig fut. Ez már 100 elem esetén is kényelmetlenül hosszú.

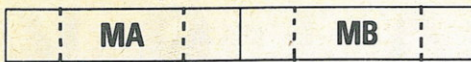
A legtöbb programozó azt mondaná az 1. algoritmusra: „Van egy kézenfekvő módszer, amivel sokkal gyorsabbá lehet tenni.” Ha kihasználjuk, hogy az  $(a..f)$  összeg szoros kapcsolatban van az  $(a..f+1)$  összeggel, akkor négyzetes algoritmushoz jutunk. Ilyen például a 2. algoritmus. Négyzetesnél hatékonyabb algoritmust készíteni már jóval nehezebb. Ha a problémát fel tudjuk osztani két nagyjából egyenlő részre, és a részekre kapott megoldások felhasználásával el tudjuk készíteni a teljes probléma megoldását, akkor egy hatékony rekurzív algoritmust kaphatunk. Esetünkben a vektort feloszthatjuk két nagyjából megegyező méretű részre:

A	B
---	---

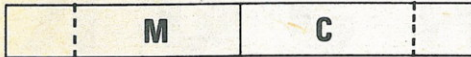
Ezután rekurzívan megkeressük A-ban és B-ben a maximális részvektorokat (MA, MB):

### Közös rutinok a maximális részvektor kereséséhez

```
program Részvektor;  
  
uses  
  Crt;  
  
const  
  elemszám = 100;  
  
var  
  vektor : array [1..elemszám] of real;  
  max    : real;  
  eleje, vége : integer;  
  ee, vv : integer;  
  
procedure RandomVektor;  
{ Feltölti vektort véletlenszerűen  
  választott értékekkel }  
  
var  
  i : integer;  
  
begin {RandomVektor}  
  Randomize;  
  for i:=1 to elemszám  
  do vektor[i]:=Random(2000) - 1000.0;  
end; {RandomVektor}  
  
procedure Kiír(  
  szám : integer;  
  e, v : integer);  
{ Kiírja a vektort es kiemeli az [e..v]  
  részvektort. }  
  
var  
  i : integer;  
  c : char;  
  összeg : real;  
  
begin {Kiír}  
  Writeln(szám,  
    ' algoritmus.           A vektor:');  
  összeg := 0;  
  for i:=1 to elemszám  
  do begin  
    if i mod 10 = 1  
    then Writeln;  
    if (i >= e) and (i <= v)  
    then begin  
      HighVideo;  
      összeg := összeg + vektor[i];  
    end  
    else LowVideo;  
    Write(vektor[i]:7:0);  
  end;  
  LowVideo; Writeln;  
  Writeln('Az összeg: ',összeg:7:0);  
  Writeln; Writeln; c:=Readkey;  
end; {Kiír}  
  
{$I vektor1.pas}  
{$I vektor2.pas}  
{$I vektor3.pas}  
{$I vektor4.pas}  
  
begin {Részvektor}  
  repeat  
    RandomVektor;  
    Választ1(eleje,vége);  
    Kiír(1,eleje,vége);  
    Választ2(eleje,vége);  
    Kiír(2,eleje,vége);  
    Választ3(eleje,vége);  
    Kiír(3,eleje,vége);  
    Választ4(eleje,vége);  
    Kiír(4,eleje,vége);  
  until false;  
end. {Részvektor}
```



Sajnos azt azért nem mondhatjuk, hogy a teljes vektor maximális részvektora MA vagy MB, mivel elképzelhető, hogy átnyúlik a határon (MC):



Az algoritmusnak tehát rekurzívan ki kell számítani az MA-t és MB-t, valamilyen más módon MC-t, és közülük a leg-

## 1. algoritmus

```

procedure Választ1(var e, v : integer);
{ Kiválasztja a maximális összegű
  részvektort }
var
  alsó, felső : integer;
  összeg, max : real;
  i : integer;
begin {Választ1}
  max := vektor[1];
  e := 1; v := 1;
  for alsó := 1 to elemszám
  do begin
    for felső := alsó to elemszám
    do begin
      összeg := 0;
      for i := alsó to felső
      do begin
        összeg := összeg + vektor[i];
      end;
      { összeg most a vektor[alsó..felső]
        összeget tartalmazza }
      if max < összeg
      then begin
        e := alsó; v := felső;
        max := összeg;
      end;
    end;
  end;
end; {Választ1}

```

## 2. algoritmus

```

procedure Választ2(var e, v : integer);
{ Kiválasztja a maximális összegű
  részvektort }
var
  alsó, felső : integer;
  összeg, max : real;
begin {Választ2}
  max := vektor[1];
  e := 1; v := 1;
  for alsó := 1 to elemszám
  do begin
    összeg := 0;
    for felső := alsó to elemszám
    do begin
      összeg := összeg + vektor[felső];
      { összeg most a vektor[alsó..felső]
        összeget tartalmazza }
      if max < összeg
      then begin
        e := alsó; v := felső;
        max := összeg;
      end;
    end;
  end;
end; {Választ2}

```

nagyobb adja a teljes vektor legnagyobb összegű részvektorát.

A 3. algoritmus a Maxösszeg rekurzív szubrutin segítségével végzi ezt. A rutin (alsó..felső) tartományban keresi a maximális összegű részvektort. Ez (e..v) és az összeg "max".

Először két lépésben a középen átnyúló legnagyobb összegű részvektort (MC) keresi: előbb a balra, majd a jobbra levő részét. Ezután rekurzívan meghívja önmagát a bal oldali (MA), majd a jobb oldali maximum (MB) megállapításához. Végül a három részvektor közül a legnagyobb összegűvel tér vissza.

## 3. algoritmus

```

procedure Választ3(var e, v : integer);
{ Kiválasztja a maximális összegű
  részvektort }
var
  max : real;
procedure Maxösszeg(
  var e, v : integer;
  var max : real;
  alsó, felső : integer);
var
  közép : integer;
  ee, vv : integer;
  összeg : real;
  i : integer;
begin {Maxösszeg}
  if alsó = felső
  then begin
    e := alsó; v := e;
    max := vektor[e];
  end
  else begin
    közép := (alsó+felső) div 2;
    e := közép; v := közép;
    összeg := vektor[közép];
    max := összeg;
    { Középtől balra összegzés }
    for i := közép-1 downto alsó
    do begin
      összeg := összeg + vektor[i];
      if összeg > max
      then begin
        max := összeg;
        e := i;
      end;
    end;
    összeg := max;
    { Középtől jobbra összegzés }
    for i := közép+1 to felső
    do begin
      összeg := összeg + vektor[i];
      if összeg > max
      then begin
        max := összeg;
        v := i;
      end;
    end;
    összeg := max;
  end;
  { Baloldali maximum }
  Maxösszeg(ee, vv, összeg,
    alsó, közép);
  if összeg > max
  then begin
    max := összeg;
    v := vv; e := ee;
  end;
  { Jobboldali maximum }
  Maxösszeg(ee, vv, összeg,
    közép+1, felső);
  if összeg > max
  then begin
    max := összeg;
    v := vv; e := ee;
  end;
end; {Maxösszeg}
begin {Választ3}
  Maxösszeg(e, v, max, 1, elemszám);
end; {Választ3}

```

Az így nyert algoritmus n elem esetén n log n-nel arányos lépést igényel. Ennek igazolása megtalálható Jon Louis Bentley: A programozás gyöngyszemei c. könyvének 7. fejezetében (Bp., Műszaki Könyvkiadó, 1988.).

Nehezen hihető, hogy még ennél hatékonyabb algoritmus is létezhet. A 4. algoritmus mégis ilyen — a feladatot az elemszámmal arányos lépésben oldja meg. Az első elemtől kezdve végignézi a tömb elemeit, és mindig számon tartja az addig tapasztalt maximális összegűt. Tegyük fel, hogy a feladatot már megoldottuk (1..i)-re. Hogyan terjeszthetnénk ezt ki (1..+1)-re? A maximális összegű részvektor vagy az első i elemből kiválasztott maximális összegűvel egyezik meg, vagy az i-edik pozícióban végződik.

maxeddig

maxvégig

A program működésének kulcsa az, hogy minden lépés után "maxeddig" az (1..i) vektor legnagyobb összegű részvektorát, "maxvégig" pedig azt a legnagyobb összegű részvektort tartalmazza, amely magában foglalja a legutolsó (i-edik) elemet is.

Hogy milyen különbségeket jelent az algoritmus jó megválasztása, arról saját gépén bárki könnyen meggyőződhet.

## 16. feladat: Sorba rendezés

Írjon olyan programot, amely nagyság szerint növekvő sorba rendez egy N egész számból álló egydimenziós tömböt! Törekedjen arra, hogy a program nagy N értékekre is gyors legyen!

Pintér Gábor

## 4. algoritmus

```

procedure Választ4(
  var alsóeddig, felsőeddig : integer);
{ Kiválasztja a maximális összegű
  részvektort }
var
  alsóvégig : integer;
  maxeddig, maxvégig : real;
  i : integer;
begin {Választ4}
  maxeddig := vektor[1];
  alsóeddig := 1; felsőeddig := 1;
  maxvégig := vektor[1];
  alsóvégig := 1;
  for i := 2 to elemszám
  do begin
    if maxvégig > 0
    then begin
      maxvégig := maxvégig + vektor[i];
    end
    else begin
      maxvégig := vektor[i];
      alsóvégig := i;
    end;
    if maxeddig < maxvégig
    then begin
      maxeddig := maxvégig;
      alsóeddig := alsóvégig;
      felsőeddig := i;
    end;
  end;
  { maxeddig és maxvégig
    érvényesek [1..i]-re. }
end;
end; {Választ4}

```



# ZAK McKRACKEN and the alien mindbenders

A Lucas Filmgames kalandjátéka már hosszú hónapok óta népszerű, és huzamosabb időn át vezette a TOP-listát is. A játék célja az emberiség megmentése idegen lények mesterkedéseitől. Az idegenek különös gépeikkel az emberiség szellemi képességeinek csökkentésére törekzenek. Zaknak a Földre küldött ügynökeiket kell kicseleznie. A Földön tevékenykedő ügynökök teljesen emberszerűek. A feladat végrehajtásához három segítőtársra lesz szükségünk, őket azonban csak később aktivizálhatjuk.

A betöltés utáni lázalmot tekintve érthetővé válik Zak igyekezete... A részletes ismertetés előtt azonban ejtsünk néhány szót az irányításról is. A játékhoz egy kis angol tudásra és nagy helyzetfelismerő képességre van szükség. A menüben szereplő ígéseket kell megfelelően összepárosítani a használni kívánt tárggyal, például: ajtónyitás OPEN DOOR. A használni kívánt tárgyra a képen vagy a menüben mutassunk rá a célkeresztrel, és nyomjuk meg a tűzgombot. Az igéknél hasonló módon járjunk el.

A billentyűk:

SPACE felfüggeszti a játékot

RETURN továbblépés

F2 játékkállás elmentése, töltése

F8 játék újratekintése

Az álmat követően Zak a szobájában látható. Vegyük fel az akváriumot, nyissuk ki az alatta lévő fiókot, és vegyük ki a telefonkártyát. Menjünk az íróasztalhoz, húzzuk ki a fiókot, és vegyük fel a szájharmonikát. Ha becsukjuk a fiókot, az asztal alatt egy műanyag kártya található. Ezt azonban még nem tudjuk felvenni. Mindenesetre tépjük le a tapétát a falról (TORN WALL-PAPER), és emeljük fel a szőnyeg jobb sarkát.

Ezután kapcsoljuk be az íróasztalon lévő üzenetrögzítőt és távozzunk. A szomszéd szobában egy kiváló tévé hívja fel magára a figyelmet. Keressük meg a távirányítóját a heverő párnája alatt. De ezt még hiába használnánk, nem történne semmi, ugyanis a tévé mellett hever a garnitúra hiányzó párnája, mögötte a tévézsinór és a dugaszolóaljzat. Dugjuk be a konnektorba a zsinórt, és most bekapcsolhatjuk a tévét. Ezek után szedjük fel a konyhából a vajkenő kést, nyissuk ki a mosogató alatti szekrényt, és vegyük ki a krétás dobozt. Nyissuk ki a hűtőt, vegyük ki a tojást, és vegyük fel az ajtó mellől a kis kulcsot. Eközben Zak egyszer nagy érdeklődést tanúsít a műsor iránt. Térjünk vissza a hálószobába.

Lépjünk az íróasztalhoz, és bányásszuk ki a hitelkártyát (cash card) a vajkenő késsel. Most térjünk vissza az előző szobába, és azon keresztül menjünk ki az utcára. Induljunk balra, és csöngessünk be a pékhez (push doorbell). Az eligazítás után szemtellenül csöngessünk be még kétszer, mire a pék megdob egy múlt heti kenyérral. Ezt vegyük fel, és menjünk jobbra, a telefonszobához. Itt vegyünk fel egy jelentkezési lapot (hátról a dobozból). Távozzunk, és induljunk el jobbra fel a 14. utcán. Itt nyissuk be a boltba. Vásároljunk meg mindent, amit csak látunk, kivéve a gitárt. Fizetni a hitelkártyával kell, a boltos minden

vásárláskor elkéri. Fontos a szerszámosláda (tool kit), az esőkabát (wet suit), a kalap (hat), az álarc (nose glasses) és a golfütő (golf club). Vehetünk lottót is. Vegyük fel a kabátot, az álarcot és a kalapot — bár nem kényelmes viselet —, és menjünk be a telefonszobához. Használjuk a pult mögötti terminált a telefonszámlával, majd töltsük ki a jelentkezési lapot a sárga ceruzával (use yellow crayon on phone billet).



Térjünk haza, de mielőtt bemennénk a lakásba, nyissuk ki a postaládát a kis kulccsal, és tegyük bele a jelentkezési lapot. Menjünk a konyhába, és nyissuk ki a szerszámosládát (a menü kibővül a szerszámokkal). Vegyük elő a franciakulcsot (monkey french) és szereljük le vele a lefolyót. Ezután tegyük be a száraz kenyeret a mosogatóba, és kapcsoljuk fel a kapcsolót. A mosogató motorja ledarálja a kenyeret, és mi felvehetjük a morzsakupacot alulról. Ballagjunk vissza a hálószobába, kapcsoljuk be az üzenetrögzítőt. A mama kedves figyelmeztetése közben vidáman bontsuk fel a felhajtott szőnyegsarkoknál a padlót a franciakulccsal. Használjuk a kötelet (rope) a leereszkedéshez, különben nagyot koppanunk odalent!

Menjünk ki balra az ajtón, és a társaságtól kilépve jobbra a buszhoz. Itt használjuk a szájharmonikát, ezzel felébresztjük a sofőrt. A szívélyes invitálás után szálljunk be és használjuk a hitelkártyát a jegykezelőnél. A repülőtérén adjuk hitelkártyánkat az ott csellengő buddhista úrnak, mire ő egy igaz tanításokat tartalmazó könyvet ad (42 dollárért!). Szálljunk fel a Seattle-be induló repülőre. A repülőn a stewardess szövegét lelőhetjük RETURN-nel. Repülés közben

fáradjunk a mosdóba és tépjünk egy kis WC-papírt. Ezt helyezzük el a mosdóba, nyissuk meg a csapot, és nyomjuk meg a falon látható stewardesshívó gombot. Siesünk vissza a helyünkre, amíg a stewardess meg nem érkezik a WC-hez. Miközben ő megpróbálja helyrehozni a mellékhelyiséget, helyezzük el a tojást a mikrohullámú sütőben, csukjuk be az ajtaját, és kapcsoljuk be. Menjünk a helyünkre, ugyanis a tojás nemsokára felrobban. A visszasiető stewardess gutaütéstől környékezve igyekszik rendet rakni itt is.

Most cselekedjünk gyorsan. Emeljük fel az előttünk lévő ülés párnáját, és vegyük fel a kieső öngyújtót. Utána nyitogassuk a csomagszekrényeket, amíg nem nem találjuk az oxigénmaszkot. Nyomjuk meg a RETURN-t, és megérkezünk Seattle-be. Menjünk ki a reptérről, és törjük le a faágat. Adjuk oda a reptérről kapott mogorót (peanuts) a hörcsögnek, és miután elment, kapirgáljunk a bottal a lyuk körül. Kisvártatva egy barlang bejárata tárul elénk. Lépjünk be, és keressük meg a madárfészket (birdnest) a WHAT IS paranccsal, és tapogassuk körbe! Szedjük le a madárfészket a bottal, és tegyük a jobbra lévő tűzrakó helyre (fire pit), tegyük rá a botot, és gyűjtjük meg az egészet az öngyújtóval. Menjünk a barlangban jobbra, és egészítsük ki az idegen jeleket (strange markets) a sárga ceruzával (use yellow crayon on strange markets). Lépjünk be a keletkezett ajtón, használjuk a távirányítót. Vegyük fel a kék kristályt.

(Folytatjuk)



# MIRE JÓ AZ RS232 CSATORNA?

A VIC-20 és a C64 számítógépek által használt RS232 csatorna a gép hátulján található. A nyílás fölött felirat van: USER = felhasználói port. Ez egy 24 pólusú illesztőegység, amelyhez a kezelő szoftvert a gépbe gyárilag beépítették.

Mire jó a felhasználói port? Képzeld el, hogy a számítógépünkön csak megnyomunk egy gombot, és máris felgyullad a szobánkban a villany. Vagy hasonló mozdulatra társaloghatunk az alattunk lakó szomszédal. Az RS232 csatorna felhasználói lehetőségei csaknem végtelenek! Ennek tudatában könnyebben rászánhatjuk magunkat, hogy alaposabban megismerkedjünk a gépünkben rejlő nagyszerű hardver adottsággal.

Az RS232 csatorna kezelésének alapvető utasításai a következők.

## A csatorna megnyitása:

OPEN 1f,2,Ø parancs sztring

ahol

1f a logikai fájl száma (melyre később hivatkozhatunk),

2 az RS232 csatorna egység száma,

Ø a csatornán mind INPUT, mind OUTPUT lehet.

A parancs sztringet később részletezzük.

## A parancs sztring leírása

Ez a sztring CHR\$ értékekből áll, mégpedig a következőképpen:

CHR\$(BR + AB + SB) + CHR\$(HM + AM + PE)

ahol

BR átviteli sebesség baudban

AB adatbitek száma

SB stopbitek száma

HM huzalszám

AM az átvitel módja

PE a paritás ellenőrzése

## Az átviteli sebesség lehetőségei

Decimálisan	Sebesség (baud) programban deklarált
Ø	
1	50
2	75
3	110
4	134,5
5	150
6	300
7	600
8	1200
9	1800
10	2400
11-15	nem használható

## Az adatbitek számának lehetőségei

Decimálisan	Adatbitszám
0	8
32	7
64	6
96	5

## A stopbitek számának lehetőségei

Decimálisan	Stopbitszám
0	1
128	2

## Az RS232 huzalozása

Decimálisan	Huzalszám
0	3
1	X

X = max. 25 pólus

## Az átvitel módjának változatai

Decimálisan	Átviteli mód
0	duplex
16	fél duplex

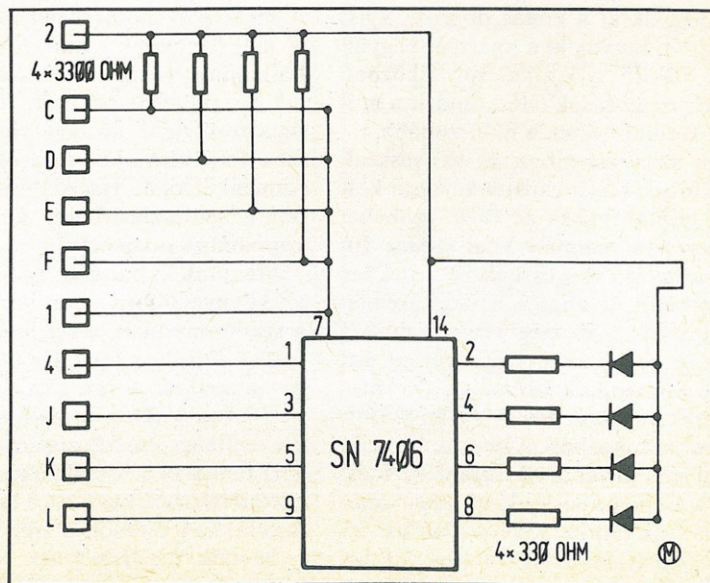
## Paritásellenőrzés

Decimálisan	Paritásellenőrzés
0	nincs, 8. adatbit
32	páratlan
96	páros
160	nincs, 8. adatbit 1
224	nincs, 8. adatbit Ø

Tehát ezeket az értékeket kell a csatorna nyitásánál a parancsba beírni. BASIC-ből a csatorna az ST változóval figyelhető, de az értéke a kiolvasás után azonnal törlődik.

## A programozáshoz még használt tárcímek

Decimálisan	Jelentése
659	vezérlőregiszter
660	utasításszó
661/662	az idő értéke
663	állapotregiszter
664	az adatbitek száma
665/666	az idő OUTPUT-nál
181	a csatorna következő bitje
144	ST státuszszó
247/248	INPUT puffer mutató
249/250	OUTPUT puffer mutató
667	a csatornán fogadott INPUT mutató (bájt)
668	INPUT mutató
669	a csatornán küldendő OUTPUT mutató (bájt)
670	OUTPUT mutató





A csatorna megnyitásánál vigyázni kell, mert a BASIC-terület a végéből 1024 bájtot lefoglal az RS232 puffereinek. És az itt tárolt program elvesz! A változók tartalma nyitáskor törlődik, mert a rendszer végrehajt egy CLR utasítást is. A csatorna nyitását ajánlatos tehát a programunk legelejére tenni.

A csatorna megnyitása után már a megszokott fájlkezelő utasítások használhatók. Ezek a következők:

olvasás: INPUT#LF, STRING VÁLTOZÓ LISTA (nem ajánlott)  
GET#LF, STRING  
írás: CMD#LF, VÁLTOZÓ LISTA  
PRINT#LF, VÁLTOZÓ LISTA

Ezekkel az utasításokkal a teljes adatforgalmat le-bonyolíthatjuk az RS232-n. A csatorna a CLOSE LF utasítással zárható le. De vigyázat, ilyenkor a pufferek azonnal megszűnnek, s a bennük még felhasználatlan adatok is elvesznek! A BASIC az 1024 bájtot visszakapja. Tehát a biztonságos működés érdekében ezt a programunk legvégére ajánlatos tenni.

#### Az állapotregiszter egyes bitjeinek jelentése

- 0. bit paritáshiba
- 1. bit szerkesztési hiba
- 2. bit input puffer túlcsoordulás
- 3. bit input puffer üres
- 4. bit CTS jel nem jött be
- 6. bit DSR jel nem jött be
- 7. bit vonalszakadás, egyéb hardverhibák

Az 5. bitet a rendszer egyéb célokra használja.

Az RS232 csatorna lábkiosztását nem részletezem, mert ez a gépkönyvben megtalálható.

Lássunk most egy ismert RS232 példát. A példaszoftver csak C64-re jó, de a hardver VIC-20-ra is. A példaprogramnak elvileg semmi jelentősége nincs, de a gyakorlatban egy ilyen egyszerű példából kiindulva megoldhatók bonyolultabb feladatok is. A példaprogramhoz szükséges hardver az ábrán látható. Íme tehát az ábrázolt áramkör kezelő szoftvere:

```

0 OPEN 2,2,CHR$(5+0+128)+CHR$(1+0+32)
5 S=53280:L=56577
10 POKE56579,240:POKEL,0
20 PRINT CHR$(147),CHR$(19):POKES,0:
   POKES+1,0:POKE 646,6
30 PRINT "THE LED AND RS232 TASTE
   SOFTWARE"
40 PRINT "TASTE=FUNCTIONS KEY!"
50 POKE 204,0
60 GETA$
70 IFA$=CHR$(133)THEN POKEL,PEEK(L)OR 16
80 IFA$=CHR$(134)THEN POKEL,PEEK(L)OR 32
90 IFA$=CHR$(135)THEN POKEL,PEEK(L)OR 64
100 IFA$=CHR$(136)THEN POKEL,PEEK(L)OR 128
110 IFA$=CHR$(137)THEN POKEL,PEEK(L)AND 239
120 IFA$=CHR$(138)THEN POKEL,PEEK(L)AND 223
130 IFA$=CHR$(139)THEN POKEL,PEEK(L)AND 191
140 IFA$=CHR$(140)THEN POKEL,PEEK(L)AND 127
150 IFA$=CHR$(32)THEN CLOSE2:SYS64738
160 GOTO60
  
```

#### A program leírása

- Csatornamegnyitás (konkrétan ebben a programban nem is lenne rá szükség).
- Színbeállítás, szövegkiírás.
- A funkcióbillentyűk figyelése és értékük szerinti beállítás.

Bartos Gyula

# Újra Gunship!

Az 1989/6-os számunkban közölt Gunship-leírásban felkértem a Tisztelt Olvasókat, hogy aki tudja a helikopter sikeres leszállásához szükséges kódokat, küldje el őket. Nos, a kódok több olvasótól is igen hamar megérkeztek. Például Erős Attilától és Témun Attilától. Mivel a küldött jelszó-kód párok egyeztek több különböző levélben is, így azok biztosan jók. Tehát a passwordkhöz tartozó kódok:

PASSWORD	CODE
ACCENT	TRAMPOLINE
BILLBOARD	KICK BACK
CROMAGNON	MELODRAMA
DAKOTA	ONSTAGE
ELECTRA	VERTICAL
FOOTHOLD	INSOLENT
GRENADIER	NOCTURNE
HEDGEHOG	LOCKSMITH
IVORY	WILLON
KNOCKOUT	PUREBRED
LOZENGE	ROMANTIC
MAZURKA	YELLOW
OVATION	UPSTAGE
PENTHOUSE	SYMPHONY
QUARTZ	ZEBRA

Tass Csaba

# ÖRÖKÉLET

ALLIGATA	3574,44	ANDROID 2	4283,234
BALAGGER	53264,126	BATTLE TIME	22045,255
BLUE MOON	7316,255		
BOMB JACK	RUN RESET	CAVELON	23789,255
	5693,255		15458,255
	5694,255		
		CRAZY	
		CAVEMAN	3332,255
	5695,255		
		DEFENDER	3005,5
	SYS 2096		
		DRAGON HAWK	3477,255
EAGLE EMPIRE	2214,15	ESPLAL	3359,33
GATEWAY			
TO APSHAI	2264,99	HIGHMOON	18033,255
HOUSE			
OF USHER	6721,238	LANCER LORDS	16424,60
LODE RUNNER	7892,255	MONTY ON	
		THE RUN	3994,255
		OCEAN	
MAGGOT MANIA	2532,34	DECATHLON	9450,173
OMEGA RACE	6300,230	PROTECTOR II	16425,6
PETCH	20295,44	SPELUNKER	10407,44

A SPINDIZZY-hez egész kis gépi kódú programot küldött Bakos Attila orosházi olvasónk. Most a program BASIC-betöltőjét közlünk.

```

10 X=673
20 READ A:POKE X,A:X=X+1
30 IF A<>255 THEN 20
40 SYS 679:LOAD .....
50 DATA 120,169,52,133,1,162,6,189,7,8,157,0
60 DATA 233,202,16,247,169,55,133,1,88,96,255
  
```

# Változtatás következményekkel

Az IBM-felhasználók örömmel fogadták a Microsoft új fejlesztését, az MS-DOS és a PC-DOS 4.0-ás verzióját. Az új termék számtalan apró javítás mellett két döntő újítást tartalmaz: a menürendszerrel és egérrel támogatott kezelést, a winchesteren pedig megszűnt a felhasználható tár 32 Mbájtos határa.

Bár a DOS így nagyobb merevlemez kapacitást képes kihasználni, mint elődei, de ennek ára van: az előző változatokkal való kompatibilitási problémák.

Egészen a 3.3-as verzióig az MS és PC-DOS-ok belső címzése korlátozta a winchester összefüggő tárterületének nagyságát. Ez a címzés ugyanis 16 biten történt, 512 bájtos szektor-nagysággal együtt 32 Mbájtos partíciókat engedett meg. Nagyobb tárterület elérésére csak akkor nyílt lehetőség, ha a szektorok méretét megnövelték, de ez a felhasználóknak kompatibilitásbeli problémákat okozott.

A 4.0-ás verziónál a szektorcímzés 32 bites számokkal történik, ez elméletileg 512 bájtos szektorokkal 2048 Gbájtos (sőt, a maximális 64 kbájtos szektorokkal 256 Terabájtos) partíció-nagyságot enged meg.

Határt szab a méreteknek, hogy az 512 bájtos szektorokkal és a 3.0 változat óta megszokott 4 szektoros szektorcsoportokkal egy partíció maximum 128 bájtos lehet. Természetesen nagyobb szektormérettel és több szektoros csoportokkal nagyobb terület is elérhető (2 Gbájtnál 32 kbájtos szektorokra vagy 64-es szektorcsoportokra van szükség).

Ez a megváltoztatott merevlemez-kezelés „felelős” főként a kompatibilitásbeli hibákért. A Microsoft állítása szerint a különösebb trükkök nélkül írt, „jóhiszemű” programoknál ez semmiféle nehézséget nem okozhat, de a probléma az, hogy éppen a leggyakrabban használt programok nem jóhiszeműek ebben az értelemben. Itt főként a merevlemez kezelő, azt újrarendező és a winchestert közvetlenül használó programokról lehet szó. Hibák léphetnek fel ezenkívül az operációs rendszer kiegészítéseinél, például a „Bernoulli-Box” esetében is. Meglepő módon a Microsoft saját programjai sem mind futnak az új rendszerben.

Még egy megjegyzendő újdonsággal lepte meg a felhasználókat a Microsoft az MS-DOS és a PC-DOS új változatának periféria-kezelésével kapcsolatban: minden disk és merevlemez formattálásánál, illetve másolásánál egyértelmű azonosítót kap (kevés a valószínűsége annak, hogy két lemezt másodpercre pontosan egyszerre formázzanak vagy másoljanak). Az operációs rendszer ennek az azonosítónak a változása nyomán állapítja meg, hogy lemezt cseréltek. Ez az eljárás viszont csődöt mondhat egy korábbi DOS verzió vagy más programok által formattált lemezeknél.

Fennáll még az a probléma is, hogy az új MS-DOS, illetve PC-DOS nem kompatibilis az OS/2 1.0-ás változatával sem. Ennek merevlemez-kezelése ugyanis még a régi módon dolgozik, tehát ha a két operációs rendszert egyszerre akarjuk üzemeltetni, akkor be kell érünk 32 Mbájtnál kisebb tárterületekkel is.

Ezek után a felhasználóknak érdemes mérlegelniük, hogy mit nyerhetnek, illetve mennyit veszíthetnek a 4.0-ás verzióval.

Friss Tamás és  
Vámos Sándor



Listánkat felhasználói, illetve játékelemekből állítjuk össze. A legjobbakat, legérdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprise-ra, TVC-re, Atarira és IBM-re készült programrangsorokat várunk havonta.

**Címünk:**

Mikroszámítógép Magazin  
Szerkesztősége  
1371 Budapest, Pf. 433  
Diákszerkesztőség

TOP LISTA																			
Játék programok	Felhasználói programok																		
	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE	
1. Bard's Tale 3	•	•	•	•	•	•	•	•	•	1. dBase IV.	•	•	•	•	•	•	•	•	•
2. Grand Prix C.	•	•	•	•	•	•	•	•	•	2. Geos 2.0	•	•	•	•	•	•	•	•	•
3. Rocket R.	•	•	•	•	•	•	•	•	•	3. DOS 4.0	•	•	•	•	•	•	•	•	•
4. Stealth F.	•	•	•	•	•	•	•	•	•	4. Giga Paint	•	•	•	•	•	•	•	•	•
5. Shanghai	•	•	•	•	•	•	•	•	•	5. Art Studio	•	•	•	•	•	•	•	•	•
6. Rizikó	•	•	•	•	•	•	•	•	•	6. Quick C 5.0	•	•	•	•	•	•	•	•	•
7. Zak McCrack.	•	•	•	•	•	•	•	•	•	7. Paintbrush	•	•	•	•	•	•	•	•	•
8. Robocop	•	•	•	•	•	•	•	•	•	8. Amiga Paint	•	•	•	•	•	•	•	•	•
9. Bosszú	•	•	•	•	•	•	•	•	•	9. News Room	•	•	•	•	•	•	•	•	•
10. Battle Chess	•	•	•	•	•	•	•	•	•	10. Printmaster	•	•	•	•	•	•	•	•	•

## Betegség ellen

A számítógépes vírusok témaköre sajnos továbbra is igen aktuális. Olvasóinkat a következő programok ismertetésével segíteni szeretnénk a vírusok elleni küzdelemben. Bár a programok csak a vírusok „hazai” változataival foglalkoznak, a problémát eltérő módon közelítik meg.

## védekezzezz!

# Vírusirtó program

A vírusok immáron hazánkban is gyors ütemben terjednek, s ezt a folyamatot az illegális programmásolások miatt nagyon nehéz megfékezni. Ha pedig egy vírus már „befészkelte” magát a programokba, általában csak a fertőzött programok letörlésével érhetünk el eredményt. Mivel annak megállapítása, hogy egy program fertőzött-e, nem olyan egyszerű dolog, ezért gyakran csak az összes program letörlése lehet a megoldás, ami elég kényelmetlen módszer. Az itt bemutatott program a hazánkban legelterjedtebb *POTYOGÓS* és az általam *RESETELŐ*-nek hívott vírus felkutatására és kiirtására képes.

## A vírusokról

A vírusfertőzést onnan lehet felismerni, hogy a gép addig nem tapasztalt dolgokat művel: a lemezműveletek idejének hossza megnő (ekkor fertőz a vírus), programjaink tönkremennek, törődnek, a képernyő időnként összezavarodik (például a *POTYOGÓS*-nál a betűk leesnek). De ezek az esetek vírusfertőzés nélkül is előfordulhatnak. Víusról akkor beszélhetünk biztosan, ha ezek a gondok ismétlődnek, szaporodnak, más lemezek is megfertőződnek, és ha fájljaink hossza indokolatlanul megnő, mivel a vírus hozzámásolja magát.

Egy vírus tevékenysége kettős: szaporodik és árt. Lehet pusztán egy jónak vélt tréfa, de vannak komolyan károsító vírusok is. Néhánynak a működése csak a behívás idejére korlátozódik — ilyen a *RESETELŐ* is —, de van olyan is, amelyiknek rezidens része egy-egy megszakításirutin-cím magára írásával bizonyos időközönként vagy lemezműveleteknél lép működésbe. Erre példa a *POTYOGÓS* vírus. Szaporodásukat általában — mint példánkban látni fogjuk — úgy oldják meg, hogy egy kiszemelt futtatható (COM vagy esetleg EXE kiterjesztésű) fájl végéhez hozzáírják magukat, majd a fájl elejére egy önmagukra mutató ugróutasítást írnak. Ahhoz, hogy a programot ezután is tudják futtatni — hiszen enélkül hamar lefűlhetnek lennének —, ezeket az első bájtokat meg kell őrizniük. Egy vírus kiirtásánál ezt a néhány — esetünkben három — bájtot kell megtalálnunk.

## A RESETELŐ vírus

Hossza 648 bájt (a fájlok hossza 648 bájttal nő). Csak COM kiterjesztésű fájlokat fertőz meg. Tevékenysége nem látványos, de annál károsabb: sorra veszi a COM fájlokat, eldönti róluk, hogy fertőzöttek-e, és ha nem, akkor megfertőzi az éppen soron lévő, majd futtatja az eredeti programot. Tehát egyszeri futtatásakor csak egy fájlt fertőz meg. Azt, hogy egy fájl fertőzött-e, úgy állapítja meg, hogy lekérdezi a fájl létrehozásának, változtatásának idejét, és ha a másodpercek száma 31 (\$1F), akkor a fájl már fertőzött. Így az összes fájl idejének ilyen beállításával elérhető, hogy a vírus ne fertőzzön! (1. program) A vírus fertőzése: lekérdezi a rendszeridőt, másodperceinek számát elosztja 8-cal. Ha a maradék 0, akkor a fájl elejére a következő utasítást írja: `JMP F000:FFE0 ($EA $E0 $FF $00 $F0 bájtok)`. Ez pedig a rendszer hidegindítását eredményezi. Az így fertőzött fájlok már menthetetlenek, mivel az eredeti bájtok nem őrződtek meg. Ha a maradék nem 0, akkor szaporodik a vírus: megőrzi az első három bájtot, majd felülírja őket egy ugróutasítással: `JMP { a fájl eredeti hossza - 3} ($E9 XX XX bájtok)`. Így a vezérlés futtatásakor az eredeti program utolsó bájta utáni utasításra kerül. A vírus pedig ide másolja magát.

A 2. program úgy azonosítja a vírusokat, hogy a kezdő bájtoit vizsgálja meg. A *RESETELŐ* a következőképpen kezdődik:

```
51          PUSH CX
BA YY XX   MOV DX,XXYY
FC         CLD
8B F2     MOV SI,DX
81 C6 0A 00 ADD SI,000A
BF 00 01   MOV DI,0100
B9 03 00   MOV CX,0003
F3        REP Z
A4        MOVSB
```

Innen már le is olvasható, hogy az eredeti három bájt hol található: az `XXYY + 0AH` címtől. Mivel a DOS a COM kiterjesztésű fájlokat `100H` (256)-tól tölti a memóriába, ezért a tényleges cím: `XXYY + 0AH - 100H = XXYY - F6H = XXYY - 246`. Így ezeket a bájtokat visszairva és a vírust a fájlból kitörölve kiirthatjuk a vírust.

## 1. program

```

PROGRAM RESETELŐ ELLENI VÉDELEM;
{***** Hornák Zoltán *****)
VAR HANDLE: INTEGER;
UT: STRING(80);
R: RECORD AX, BX, CX, DX, BP, SI, DI, DS,
ES, FLAGS: INTEGER;
END; {8086/8088 REGISZTEREI}
OK: BOOLEAN;

BEGIN
OK:=TRUE;
REPEAT
CLRSCR; HIGHVIDE0; GOTOXY(20,1);
WRITELN('RESETELŐ VIRUS ELLENI VÉDEL
EM V1.00');
IF NOT OK THEN
BEGIN
GOTOXY(1,6);
WRITE('Nem tudom megnyitni a fájl
t!');
END;
GOTOXY(1,4);
WRITE('Megvédendő fájl neve:');
READ(UT);
IF UT='' THEN HALT; R.AX:=$3D02;
IF POS('.',UT)=0 THEN UT:=UT+'.COM';
UT:=UT+#0;
R.CX:=$0003; R.DS:=SEG(UT);
R.DX:=DFS(UT)+1; MSDOS(R);
HANDLE:=R.AX; OK:=TRUE;
IF (R.FLAGS AND 1)=1 THEN OK:=FALSE
ELSE
BEGIN
R.BX:=HANDLE; R.AX:=$5700;
MSDOS(R); R.DX:=R.DX OR $1F;
R.AX:=$5701; R.BX:=HANDLE;
MSDOS(R); R.AX:=$3E02;
R.BX:=HANDLE; MSDOS(R);
END;
UNTIL FALSE;
END.
    
```

## 3. program

```

{*****VAN-E A MEMORIÁBAN VIRUS?*****}
PROGRAM MEMVIR;
TYPE
RESS=RECORD AX, BX, CX, DX, BP, SI, DI, DS,
ES, FLAGS: INTEGER;
END;
VAR R: RESS;
BEGIN
R.AX:=$4BFF; {virus lekérdezése}
R.DI:=0; R.SI:=0; MSDOS(R);
IF R.DI=$55AA THEN
BEGIN
WRITELN('A PÓTYÓGÓS vírus már a memó
riában van!! (Most már csak volt!');
INLINE($FA);
{CLI: megszakítás letiltása}
{**21H-es megszakítás visszaállítás: **}
MEMW0:$64:=R.AX; MEMW0:$86:=R.ES;
{**1CH-es megszakítás visszaállítás: **}
MEMW0:$70:=MEMW0.DX:$13D;
MEMW0:$72:=MEMW0.DX:$13E;
IF MEMW0:$A2=R.DX THEN
BEGIN
{**2BH-es megszakítás visszaállítás: **}
MEMW0:$A0:=MEMW0.DX:$13B;
MEMW0:$A2:=MEMW0.DX:$13D;
END;
INLINE($FB);
{STI: megszakítás engedélyezése}
END
ELSE
WRITELN('Nincs vírus a memóriában.');
```

## 2. program

```

PROGRAM VIRUS_IRTO(V1.00 Hornák Zoltán);
CONST S=$6000;
VAR I, VK, M, HOSSZ: INTEGER;
COMFILE: FILE;
NEV: STRING(80);
VOLTFER, FERTOZES, OK: BOOLEAN;
CH: CHAR;
{***** RESETELŐ VIRUS KIIRTASA: *****}
PROCEDURE RESVIR;
BEGIN
M:=MEMW0:VK+21-246;
FOR I:=0 TO 2 DO
MEMW0:I:=MEMW0:M+1;
FOR I:=VK TO HOSSZ DO
MEMW0:I:=0;
HOSSZ:=VK;
VOLTFER:=TRUE;
END;
{***** PÓTYÓGÓS VIRUS KIIRTASA: *****}
PROCEDURE POTYVIR;
VAR SP, SI: INTEGER;
BEGIN
SI:=VK+34; SP:=$0682;
WHILE SP>0 DO
BEGIN
MEMW0:SI:=MEMW0:SI XOR (SI+256);
MEMW0:SI:=MEMW0:SI XOR SP;
SI:=SI+1; SP:=SP-1;
END;
M:=VK+45;
MEMW0:0:=MEMW0:M;
MEMW0:1:=MEMW0:M+1;
MEMW0:2:=MEMW0:M+2;
FOR I:=VK-1 TO HOSSZ DO
MEMW0:I:=0;
HOSSZ:=VK-1;
VOLTFER:=TRUE;
END;
{***** PROGRAM BEOLVASASA *****}
PROCEDURE BEOLVASAS;
BEGIN
{$I-}
OK:=TRUE; ASSIGN(COMFILE, NEV);
IF IORESULT<>0 THEN OK:=FALSE;
RESET(COMFILE);
IF IORESULT<>0 THEN OK:=FALSE;
BLOCKREAD(COMFILE, MEMW0:0, SI, HOSSZ);
HOSSZ:=HOSSZ#128+128;
IF IORESULT<>0 THEN OK:=FALSE;
CLOSE(COMFILE);
IF IORESULT<>0 THEN OK:=FALSE;
{$I+}
END;
{*** BEOLVASOTT PROGRAM KIÉRTÉKELÉSE ***}
PROCEDURE KIÉRTÉKELÉS;
BEGIN
IF (MEMW0:0)=$FOEA AND
(MEMW0:2)=$00FF AND
(MEMW0:4)=$F0
THEN WRITELN('A fájl JAVITHATATLAN
UL megfertőződött');
IF MEMW0:0<>$E9
THEN
BEGIN
IF NOT VOLTFER
THEN WRITELN('A fájl nem fer
tőzött.');
```

```

);
VOLTFER:=FALSE;
END
ELSE
BEGIN
VK:=MEMW0:11+3;
{VK A LEHETSÉGES VIRUSRA MUTAT}
{***** PÓTYÓGÓS VIRUS? *****}
IF (MEMW0:VK)=$8BFA AND
(MEMW0:VK+2)=$EBEC AND
(MEMW0:VK+4)=0 AND
(MEMW0:VK+6)=$815B AND
(MEMW0:VK+8)=$31EB
THEN
{***** BIZTOS PÓTYÓGÓS VIRUS! *****}
BEGIN
WRITELN('PÓTYÓGÓS VIRUS!');
POTYVIR;
END
ELSE
BEGIN
{***** RESETELŐ VIRUS? *****}
IF (MEMW0:VK)=$B51 AND
(MEMW0:VK+4)=$8BFC AND
(MEMW0:VK+6)=$81F2 AND
(MEMW0:VK+8)=$0AC6
THEN
{***** BIZTOS RESETELŐ VIRUS! *****}
BEGIN
WRITELN('RESETELŐ VIR
US!');
```

## A POTYOGÓS vírus

Hossza 1701 bájtt (a fájlok hossza 1701 bájttal nő). Csak COM kiterjesztésű programokat fertőz. Eredeti IBM gépeken, vagyis olyanokon, ahol F000:E008-tól a 'COPR.IBM' szöveg található, nem fut. Van rezidens része! Tehát bennmarad a memóriában, amíg a gépet ki nem kapcsoljuk, vagy újra nem indítjuk. FIGYELEM! A rendszer újraindításakor a vírus visszatöltődhet a rendszerfájlokkal!

Egy fertőzött fájl behívásakor a következők történnek. A vírus ellenőrzi, hogy már a memóriában van-e. Ha nem, lekérdezi a rendszer dátumot. Ha ez 1988. szeptember utáni, a vírus bennmarad a memóriában, a 21H-s és az 1CH-s megszakítást önmagára írja, a potyogtatásjelzőt beállítja, majd futtatja az eredeti programot. Ha a dátum 1981, akkor az előzőekhez képest annyi változás lesz, hogy a potyogtatásjelzőt nem állítja be (így a vírus nem potyogtat, de fertőz!), és még a 28H-s megszakítást is önmagára írja. Ennek feladata, hogy a rendszeridő átállításakor beállítsa a potyogtatásjelzőt. Ez egyébként egy fenntartott BIOS-funkció, amely nem is minden gépen működik. Ha a rendszer dátum nem ennyi, akkor csak a 21H-s megszakítást írja magára, így nem potyogtat, de fertőz.

Az 1CH-s megszakítást az órakezelő rutin hívja meg, másodpercenként 18-szor. Ha a potyogtatásjelző be van állítva, késleltetések után elkezd potyogtatni.

A 21H-s megszakítás a DOS-funkciók meghívását szolgálja, így a vírus minden lemezkezelésnél akcióba lép. Ha a 4BH-s funkciót — a program futtatását — hívjuk meg, a vírus ellenőrzi, hogy a fájl COM kiterjesztésű-e, vagyis hogy nem 4DH, 5AH bájttal kezdődik, mivel ez az EXE fájlok jellemzője. Ha COM-fájl, akkor megvizsgálja, hogy fertőzött-e, vagy a hossza nagyobb-e, mint 63803. Ha nem, akkor eltávolítja, majd felülírja az első három bájtot, és a fájl végéhez írja magát. Azt, hogy fertőzött-e egy fájl, úgy dönti el, hogy megvizsgálja az első három bájtot, és ha ez egy ugrás a fájl 1700. bájtpárára, akkor a fájl már fertőzött. Ez a rutin így kezdődik:

80 FC 4B	CMP AH,4B	
74 10	JZ 10	
2E	CS:	
FF 2E 37 01	JMP FAR [0137]	
BF AA 55	MOV DI,55AA	
2E	CS:	
C4 06 37 01	LES AX, [0137]	
8C CA	MOV DX,CS	
CF	IRET	
3C FF	CMP AL,FF	
74 F1	JZ F1	
3C 00	CMP AL,00	
75 E8	JNZ E8	

Az eredeti DOS hívása

Itt látható, hogy a vírus úgy kérdezi le magát, hogy a 21H-s funkciót AX=4BFFH (DI=0, SI=0) paraméterrel hívja meg, és ekkor, ha a vírus már a memóriában van, a következő értékeket kapjuk vissza:

DX = { A vírus szegmense }  
 ES:AX = { Az eredeti 21H-s megszakítás címe }  
 DI = 55AAH

A 3. program az így felderített, memóriában lévő vírus hatástalanítására képes úgy, hogy az eredeti megszakításvektorokat visszairja.

Ahhoz, hogy a vírust kiirtsuk, meg kell még néznünk, hogyan fertőz. Mint láttuk, egy fájl futtatásakor fertőződik meg: az első három bájttal egy ugróutasítás lesz: JMP {a fájl eredeti hossza - 2} (\$E9 XX XX). Tehát a vírus a fájl eredeti vége utáni második bájton kezdődik. Az első bájtnak ellenőrző szerepe lehet, értéke 1. A POTYOGÓS kezdete:

FA CLI  
 8B EC MOV BP,SP

E8 00 00	CALL 0000	
5B	POP BX	
81 EB 31 01	SUB BX,0131	
2E	CS:	
F6 87 2A 01 01	TEST BYTE PTR [BX + 012A], 01	Hivatkozás az első bájtra
74 0F	JZ 0F	
8D B7 4D 01	LEA SI, [BX+014D]	
BC 82 06	MOV SP,0682	
31 34	XOR [SI],SI	
31 24	XOR [SI],SP	
46	INC SI	
4C	DEC SP	
75 F8	JNZ F8	

Így látható, hogy a vírus nem listázható formában van kiemelve (átXORolja magát). Tehát amikor az eredeti három bájtot keressük, ezt a műveletet nekünk is el kell végeznünk. Az eredeti bájtok a vírus kezdete után a 45. bájttól kezdődnek. Így már ezt a vírust is ki tudjuk törölni.

## Néhány szó a programról

A program bekéri a kiértékelendő fájl nevét. Ajánlatos e fájlok nevét előre feljegyezni, hogy ne kelljen a programból állandóan kilépni. Ezután beolvassa a fájlt (a fájl hossza max. 511 x 128 = 65408 bájttal lehet!), majd kiértékeli. A kiértékelést a fentebb leírt módon hajtja végre. Megvizsgálja, hogy ugróutasítással kezdődik-e a program. Ha igen, VIRKEZD mutat a vírus kezdetére. Innentől ellenőrzi az első bájtokat, és eldönti, hogy van-e, illetve milyen vírus van a programban. Ha talált vírust, akkor visszairja az eredeti bájtokat, és kitörli a vírust. Ezt a kiértékelést mindaddig folytatni kell, amíg volt fertőzés, mert amikor egy fájl a POTYOGÓS-sal fertőzött, a RESETELŐ megfertőzheti. Ekkor a POTYOGÓS már nem ismeri fel magát, így a fájlt ismét megfertőzheti. Így egy fájl háromszor (ha időközben az idejét megváltoztattuk, akkor többször is) megfertőződhet. A közkezen forgó vírusdetektor programok (például a CHKVIR) az ilyen fertőzést már nem tudják kimutatni.

A programmal problémák is vannak. 256 k-s gépen a program 2. sora: CONST S = \$3000;. A Turbo Pascal 3 csak olyan fájlokat tud megnyitni, amelyek nincsenek írásvédve, elrejtve, így e védelmeket a program futtatása előtt el kell távolítani. Mivel a rendszerfájlok is védettek, ezeket célszerű az eredetivel felülírni. Ha netán a fertőzés annyira elhatalmasodott volna, hogy a Turbo Pascal is fertőzött lenne, akkor tegyük a következőt. Másoljuk le a Turbót egy üres lemezre. Rendszerindítás után állítsuk be a rendszeridőt egy régi dátumra, ekkor a POTYOGÓS nem potyogtat. Indítsuk el a Turbót, írjuk be a programot, majd mentjük ki a lemezre. Futassuk a programot: töröljük ki a Turbóból a vírust. Ezután rakjunk be egy biztosan nem fertőzött rendszerlemezre, és indítsuk újra a gépet (CTR + ALT + DEL). Így már nincs vírus a gépben. Töltsük be a lemezről a Turbót és a programot, majd irtsuk ki a vírust az összes többi fájlból is. A vírusfertőzést csak az összes fájl kiértékelésével lehet megszüntetni!

Mit tegyünk, ha ismeretlen vírussal állunk szemben? Próbáljuk meg a vírus elterjedését megakadályozni. Figyelmeztessük a velünk programcsere-kapcsolatban állókat a vírusveszélyre! A vírusok nem futtatható fájlokat (például szövegfájlokat) nem fertőznek meg, így ezeket mentjük ki üres lemezekre, hogy legalább ezek megmaradjanak, majd ha más megoldás nincs, töröljük minden állományt.

A programot továbbfejlesztettem úgy, hogy most már képes önmagától kiértékelni az összes COM-állományt (az alkönyvtárakban lévőket is), így a vírusirtás egyszerűvé vált. Ha valaki küld egy üres lemezt és egy válaszborítékot, annak szívesen felmásolom ezt a változatot is. Címem: Veszprém, Cholnok u. 25/B. 8200.

Hornák Zoltán

## Védőoltás

```

cseg segment byte public

public maincode,go,copyright,old_21
public install,DOS_21
public uzenet,inic,eredeti

maincode      proc      far
      assume  cs:cseg
      org     100H
go:           jmp      install

old_21        dd 0

copyright     db      13,10
              db      A rendszer                ,13,10
              db      A betüpergető VIRUS ellen ,13,10
              db      immunuzálva van.         ,13,10
              db      ,13,10
              db      (C) Mörk Péter 1988      ,13,10
              db      NME Miskolc             ,13,10
              db      ,13,10
              db      $

install:call   inic
      lea    dx,inic
      mov    cx,4
      shr    dx,c1
      inc    dx
      mov    ax,3100H
      int    21H

maincode      endp

DOS_21        proc

      ; Az új DOS 21H rutin

      cmp    ah,4BH
      jnz   eredeti
      cmp    al,255
      jz    uzenet

eredeti: jmp far [old_21+2]

uzenet: mov   di,b5AAH
      iret

DOS_21        endp

inic          proc

      ; fejléc kiírása

      mov    ah,09
      lea    dx,copyright
      int    21H

      ; régi DOS vektor lekérdesés és mentés

      mov    ax,3521H
      int    21H
      mov    word ptr old_21,bx
      mov    word ptr old_21+2,cs

      ; új DOS vektor beállítás

      mov    ax,2521H
      lea    dx,DOS_21
      int    21H

      ret

inic          endp

cseg          ends
end go
    
```

A programvírusok egyikét a Műszertechnika Kíszövetkezet terminológiája XU/2 néven emlegeti. A fertőzés jele: a karakterek lepotyognak az alsó sorba.

A mellékelt programot elindítva a rendszer immunissá válik a fertőzéssel szemben, de ez a program csak ettől az egyfajta vírustól óvja meg a gépet. Célszerű a programot az AUTO-EXEC.BAT fájl segítségével minden bekapcsolás után lefuttatni. Ez annál inkább fontos, mert a már fertőzött rendszert a program nem védi meg, hiszen ilyenkor az eredeti DOS-vektor már át van írva.

A program működése azon alapul, hogy a vírus saját maga el tudja dönteni a rendszerről, hogy az fertőzött-e. Ez úgy történik, hogy az AX regiszterbe 4BFFH értéket tesz, és meghívja a 21H sorszámú DOS-szolgáltatást. Ha a vírus már rezidens módon bent van a tárban, akkor a válasz: 55AAH kerül a DI regiszterbe.

A Védőoltás program feladata mindössze annyi, hogy a vírussal „elhitesse”: már bent van a tárban, és nem kell újra betöltenie magát. Ennek megfelelően van módosítva a DOS 21H rutin. Ha AX=4BFFH-val történik a hívás, a válasz DI=55AAH lesz. Minden más esetben a vezérlés visszakerül az eredeti DOS-rutinhoz. A programot például makroassembler fordítóval lehet lefordítani, végül pedig COM formátumúra kell konvertálni (EXE2BIN).

A program betöltése után tehát futtathatjuk a fertőzött programokat is anélkül, hogy a fertőzés továbbterjedne róluk. Ezzel megkíméljük magunkat attól a fáradságtól, hogy más programok segítségével időről időre végigbogarásszuk a merevlemezt az esetleges új fertőzések megszüntetésére.

Mörk Péter

### Új szolgáltatásokkal, új helyen várja kedves ügyfeleit az **ISKOLASZÁMÍTÓGÉP SZERVIZ**

**IBM és Commodore számítógépek javítása  
közületek és magánszemélyek részére**

**Éves átalánydíjas szerződések rendkívül kedvező  
feltételekkel.**

**Egyéb szolgáltatások:**

- C16 bővítése 64 kbájtra,
- magyar ékezetes  
karakterkészlet beépítése
- játékprogramok eladása  
és vétele
- Tectronix oszcilloszkóp  
előnyös áron

**A javítás ideje alatt  
szükség szerint cseregépet biztosítunk.  
Címünk: 1088 Budapest, Rákóczi út 25.  
Tel.: 381-121.**



# Vírusgazdák kíméljenek!

El kell oszlatnunk egy tévhitet: a „vírustenyésztés” korántsem egyszerű gyermeki csínytevés. Még a legártalmatlanabbnak számító vírusok is okoznak kárt, ha közvetlen fizikai károsodást nem is. Ilyen a közismert potyogós vírus is, amelynek „ártalmatlan” működéséről azok tudnának mesélni, akiknek már kellett olyan gépen háttáridős munkát végezniük, akár csak egy szövegszerkesztést is, amelyben dühöngött ez a típus. Ha mást nem, de lekötik a kommunikációs vonalakat, lassítják a programok végrehajtását, gépidőt rabolva. Legnagyobb károkozásuk mégis a bizalmatlansággeltetés, ami az agresszív vírusok kártevésével közel egyenértékű. Ezek a kifejezetten ártó szándékkal életre keltett számítástechnikai démonok programokat, adatbázisokat, extrém esetekben akár csatlókártyákat, lemezegységeket is tönkretelhetnek. Ezek után felmerülhet a kérdés: vajon valóban teljesen védtelenek vagyunk?

## A fertőzés elkerülhető

Mint ahogyan egyre több magyarországi számítástechnikai intézmény, a Softinvest is biztosít minden olyan lehetőséget felhasználói részére, amellyel kiküszöbölhető a fertőzés. A kínáló módok közül talán a legfontosabb a megelőzés. Kizárólag eredeti forrásból származó szoftverek forgalmazásával és felhasználásával biztonsággal elkerülhető a fertőzés veszélye. A Softinvest széles skáláját forgalmazza a hazai származású fejlesztői, felhasználói és ügyviteli programoknak. Fejlesztőivel kötött szerződésben jogi garanciákkal védi programrendszerének felhasználóit. Ezek a garanciák többet érnek minden más védelmi rendszernél, hiszen képzeljük el annak a szerzőnek a további sorsát, akire rábizonyosodik egy vírus elterjesztése. Magyarországon nincs olyan szoftverfejlesztő vagy -kereskedő cég, amely ezek után szóba állna vele.

Természetesen vannak helyzetek, amelyekben nem lehet a megbízható, nevükkel is garanciát vállaló szoftverkereskedő cégekhez fordulni. Ezek lehetnek például speciális célfejlesztések, amelyek látszólag egyetlen hivatásos forgalmazó profiljába sem tartoznak. Ekkor sincs minden veszve. Győződjünk meg róla, hogy fejlesztőink jogtiszt szoftverek felhasználásával hozták létre felhasználói rendszerünket. Ha tehetjük, különítsük el egy gépet a fejlesztésre, de a tesztelésre mindenképpen. Ez természetesen önmagában nem véd meg minket a fejlesztő által algoritmikusan beépített vírustól, de mindenféleképpen megóvhat néhány, általában nem is szándékosan okozott kellemetlen meglepetéstől. Hasonló esetben, vagy ha eleve nyitott, többfelhasználós rendszerrel rendelkezünk, ahol nem kontrollálható, ki, milyen származású programokat futtat, feltétlenül ki kell építenünk saját védelmi állásainkat.

## Pajzs a MS-DOS-felhasználóknak

A fő szabály a pCHIGI esetében is a megelőzés. Ellentétben más vírusellenes

„Viruskillereket” természetesen nemcsak az amatőrök, hanem a profi számítástechnikai cégek is kidolgoztak. Ezek egyikét, a Softinvest pCHIGI programrendszerét mutatjuk be a következőkben.

programrendszerekkel, amelyek egy-egy speciális vírus kártételei után, „mentsük, ami menthető” elven működnek, a pCHIGI elsősorban a megelőzésre fordít gondot. Köztudott, hogy tökéletes védelem nincs. A fertőzés viszont a fertőző akció pillanatában detektálható, és a károkozás megakadályozható, de legalábbis minimalizálható. Nagyon fontos a biztonságot többszöröse fokozó rendszabályok betartása is, amelyeket nagy számban tartalmaz a rendszer kézikönyve.

A továbbiakban hangsúlyozottan jelentkeznek az a probléma, ami e cikk írójának fejében és talán másokéban is motoszkál. Ismertetni kellene a védelmi rendszer működési elvét, ezzel kitenni annak a felhasználókat, hogy újabb „na ezen is ki tudok fogyni” vírusokat pattintsak ki láthatatlan ellenfeleink fejéből. Érthetően optimalizálva a feladatot, megpróbálom a rendszer képességeit mindenki számára demonstrálni anélkül, hogy stratégiailag fontos adatok kerüljenek közös ellenfeleink kezébe.

Az összes vagy — legyünk kicsit realisabban — a tapasztalataink alapján elképzelhető összes vírus ellen általános védelmi rendszert csak úgy fejleszthetünk ki, hogy a normál felhasználói rendszerek számára kényelmes, de a vírusok számára kényelmetlen környezetet teremtünk. Erre van lehetőségünk, mivel a vírusoknak jellegzetes működési tulajdonságaik vannak. A rejtőkódos szándéka miatt minél tömörebb kódot kell alkalmazniuk, ezt jórészt csak DOS-hívásokon keresztül tehetik meg. Szintén rejtőkódosra törekvésük leplezi le jelenlétüket, amikor a központi tárban maradvá próbálnak minél több programot megfertőzni.

## Harcban a biológiai fegyver ellen

A pCHIGI programrendszer BIOS- és DOS-hívások ellenőrzésével védi a felhasználók érdekeit. Igény szerint négy, illetve öt elemből áll.

A vHIGI nevű program naplózza a potenciálisan veszélyeztetett rendszerlemeleket. Figyelem!!! Ezek nemcsak az EXE és COM kiterjesztésű fájlok. Hiszen végrehajtható „kódot” tartalmaznak például a BAT fájlok is. Legyünk tehát óvatosak! Természetesen a vHIGI valóban minden veszélyeztetett fájl figyelt, sőt a tapasztalatlan felhasználókat hasznos rendszabályok betartására is felszólítja.

A vírusok rongáló hatására a vLEMEZ lemezevédelmi rendszer kívánja kiszűrni. Speciális, a felhasználói programok szerkezetét nem érintő lemezszervezés mellett írásvédett teszi a veszélyeztetett fájlokat.

A rezidens programokat a vMEM rendszer figyeli. A memóriában már aktivizálódott, de még rejtőkódos vírusok hatásának kiszűrésére alkalmas. Ismerve programjain-

kat, könnyen kiszűrhetők a hasonló típusú vírusok, hiszen felettebb gyanús, ha mondjuk a Lotus 1-2-3 vagy a dBASE rezidensen a tárban kívánna maradni. A program alkalmas arra, hogy felismerje és a memóriából eltávolítsa a nem kívánt rezidens programokat.

Speciális jeleket keres a vDOKI program. A vírusok nagy százaléka a gyors működés eléréséhez különböző jeleket helyez el a lemezekre. Ezek a jelek is árukladóak. A program mellékes tulajdonsága, hogy az ún. „BOOT-vírus” bizonyos változatát gyógyítani is képes.

Az ANTIVÍRUS új fejlesztésű figyelőrendszer, amely a mesterséges intelligenciára épül. Rugalmasan igazodik a felhasználó körülményeihez. Az elméletileg elérhető legnagyobb biztonságot nyújtja a vírusok ellen az MS-DOS operációs rendszer alatt.

Nem állítható, hogy van tökéletes vírus elleni védelem, de az igaz, hogy a pCHIGI programmal a kockázat alaposan lecsökkenthető. A felhasználhatóság érdekében a pCHIGI nincs védve másolás ellen, viszont jogszerű felhasználóinak nevét „bedrótozva” tartalmazza. A rendszert Hollósi József fejlesztette ki, a forgalmazó a Softinvest.

## Házi praktika

Néhány jó tanács azoknak, akiknek még nem áll módjukban beszerezni valamilyen hatékony rendszert a vírusok elleni védelemhez:

1. Soha ne folytassuk a munkát olyan gépen, amelyen előtünk más dolgozott. Ne melegítéssel indítsuk újra a rendszert, mert ebben az esetben maradhatnak a memóriában nem kívánt diverzánsok. Mindig hidegindítást alkalmazunk.

2. Eredeti lemezeinket lássuk el írásvédelemmel, másoljuk át, és a másolatokat használjuk. Általában, ha lemezről hívunk meg egy programot, a lemezt tegyük írásvédetté.

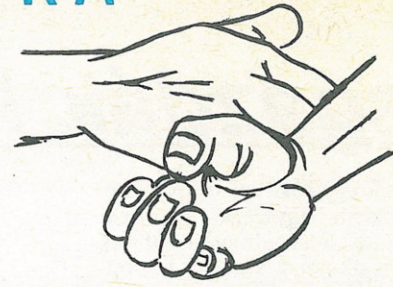
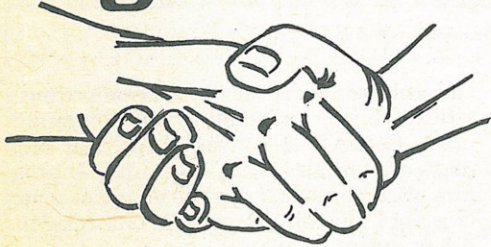
3. Amennyiben lemezhibát jelez az operációs rendszer, ha tehetjük, szakítsuk meg a program végrehajtását. A vírusok egy jelentős része próbál az operációs rendszer üzenetét utánozva időt nyerni.

4. Ne tartsunk a merevlemezünkön nem használt végrehajtható fájlokat. Egyezünk meg munkatársainkkal egy meghatározott, jól áttekinthető tárolási struktúrában. Az idegen, ismeretlen célú végrehajtható fájlokat azonnal töröljük le.

5. Végül a legfontosabb: csak ellenőrzött forrásból származó programokat használjunk. Természetesen ezek közé kell tartozniuk a programvédelmi rendszereknek is. Egyedül így biztosíthatjuk megnyugtatóan immunitásunkat a számítógépes vírusokkal szemben.

Boros László

## Programozási fogások és



# melléfogások

A *Magazin* 1987/5. számának 32. oldalán *Különleges input* címmel mintaszerűen dokumentált gépi kódú rutin jelent meg C64-re. Idézem a kísérelő leírást:

„A rutin csak olyan billentyűk lenyomását fogadja el, melyeket az idézőjelek között megadtunk, s a változó értéke a lenyomott billentyű sorszáma lesz.

A rutin legelőnyösebb felhasználási lehetőségei: menüknél annak eldöntése, hogy melyik »fogást« választotta a felhasználó. Ennek bemutatására szolgál a BASIC példaprogram (...), mely a betöltőprogramot is tartalmazza a magyarázatokkal együtt. Az A\$=CHR\$(PEEK(880)) értékadással megkaphatjuk a karaktert is.”

Az említett BASIC példaprogram egy része az 1. listán látható, csekély módosítással: az idézőjeleken belüli vezérlőkarakttereket — a jobb olvashatóság érdekében — kapcsos zárójelek közé zárt emlékeztető kódjukkal helyettesítettem. Az itt bemutatott részt az eredeti programban jól tagolt magyarázó sorok és a gépi kódú rutin betöltéséhez szükséges DATA sorok előzik meg, melyeknek részletezése elemzésünk szempontjából felesleges.

A programlistán néhány apró szépség hibát figyelhetünk meg. A 310-es sorban az S *valós* változó kap 0 kezdőértéket, a következő sorokban viszont az ellenőrző összeg az S% *egész* változóba kerül, feltehetően figyelmetlenségből. Ez bajt nem okoz, mert a Commodore gépek BASIC-interpreterje minden újonnan megjelenő változónak automatikusan 0 kezdőértéket ad. A felesleges számkonverziók elkerülése érdekében hasonló esetekben célszerűbb *valós* változókat használni. A ciklus záróértékében is jobban tetszene a "828+72" kifejezés helyett a kiszámított 900-as érték.

Hibának látszik a PRINT utasítások végén a záró idézőjel hiánya, de nem az. A Commodore BASIC-jében a programsor vége a nyitva hagyott karakterlánc-konstansokat is lezárja, éppúgy, mintha ott lenne a hiányzó idézőjel.

A felsorolt valódi vagy vélt hibákon kívül van még egy apró szépség hibája a programnak. Az, hogy teljesen *feles-*

leges a gépi kódú rutin alkalmazása. A feladat BASIC-ben is gond nélkül megoldható. Egy lehetséges megoldást a 2. listán mutatok be.

A módosított program a 380-as sorral kezdődik, mivel a gépi kódú rutin és annak betöltése elmarad. A menükiíró részben a „kurzor le” vezérlőkaraktterek használata helyett a TAB függvények argumentumát módosítottam a megegyező hatás elérése céljából. Nem tudom, miért nem alkalmazzák gyakrabban ezt a kézenfekvő módszert a nyomtatásban közzétett programokban. Lehet, hogy nem ismerik?

Az eredeti 440-es sorból négy sor lett (440—446). Ez a négy sor oldja meg a gépi kódú rutin feladatát. A gép az ezekben lévő utasításokat hajtja végre, a megadott billentyűk egyikének leütéséig. A GET utasítás közvetlenül az A\$ változóba olvassa be a leütött billentyűhöz tartozó karaktert. Az elfogadható karaktereket a MID\$ függvény első argumentumába kell tenni, a kívánt sorrendben. A gépi kódú rutinnal ellentétben itt nemcsak karakterlánc-konstans, hanem változó is megadható. Arra feltétlenül ügyelni kell, hogy a ciklusváltozó végértéke ne legyen nagyobb, mint a karakterlánc hossza. Az ON GOTO-t — ha elfér — írhatjuk közvetlenül az IF utasítás THEN-je után is. Példánkban ezt csak azért nem tettem meg, mert igyekeztem az eredeti program felépítéséhez igazodni.

Az eredeti gépi kódú rutin csak C64-en működik, de a rendszerrutinok címeinek átírásával könnyen adaptálható más Commodore gépekre is. A most bemutatott algoritmust csak a feladathoz kell hozzáigazítani, de változatlanul alkalmazható bármely Commodore gépre, a VC20-tól a C128-ig.

Mi van a futási sebességgel? — kérdezheti bárki. Valóban, a gépi kódú rutin sokkal gyorsabban fut, de ez a sebességnövekedés *egyáltalán nincs kihasználva*. A billentyű leütése és a kiértékelés befejezése között eltelt idő BASIC használata esetén is a másodperc tört része, amely a program használója számára nem is érzékelhető.

A gépi kód felesleges használatára mutattam be példát, ennek ellenére az a véleményem, hogy az idézett program közlése nem volt hiábavaló, mert — a BASIC-betöltőprogram apró hibá-

itól eltekintve — egy jól megírt gépi kódú rutin jelent meg, melynek elemzését mindenkinek szívből ajánlom, aki most ismerkedik a gépi kódú programozás rejtelmeivel.

Befejezésül egy tanács: az algoritmus, illetve a gépi kódú rutin végrehajtása előtt ürítsük ki a billentyűzetpuffert, nehogy egy korábbi „érvényes” billentyűlenyomás hamis eredményt adjon. A puffer kiürítésének egy lehetséges módját a 3. listán láthatjuk. Egyszerűbb, de gépfüggő a puffer logikai törlése: POKE 198,0 (VC20-on és C64-en), illetve POKE 239,0 (C16-on és PLUS/4-en).

Barna László

### 1. lista

```

...
290 REM --- BEOLVASAS ---
300 :
310 S=0:FORI=828TO828+72
320 READA:POKEI,A:S%=S+A:NEXT
330 IFS%=9962THEN380
340 PRINT"*** DATA HIBA ***":END
350 :
360 REM --- PELDA ---
370 :
380 PRINT"(CLR) (5 DOWN)";TAB(12);"-----"
390 PRINT"(2 DOWN)";TAB(17);"(RVSON)F1(RVSOFF) RUN
400 PRINT"(2 DOWN)";TAB(17);"(RVSON)F3(RVSOFF) LIST
410 PRINT"(2 DOWN)";TAB(17);"(RVSON)F5(RVSOFF) NEW
420 PRINT"(2 DOWN)";TAB(12);"-----"
430 PRINT"(3 DOWN)";TAB(12);"KEREM VALASSZON!"
440 SYS828,"(F1)(F3)(F5)",A
450 ONAGOTO460,470,480
460 RUN
470 LIST
480 NEW
    
```

### 2. lista

```

...
380 PRINT "{CLR}";TAB(212);"-----"
390 PRINT TAB(97);"(RVSON)F1(RVSOFF) RUN
400 PRINT TAB(97);"(RVSON)F3(RVSOFF) LIST
410 PRINT TAB(97);"(RVSON)F5(RVSOFF) NEW
420 PRINT TAB(92);"-----"
430 PRINT TAB(132);"KEREM VALASSZON!"
440 GET A$
442 FOR A=1 TO 3
444 IF A$=MID$( "(F1)(F3)(F5)",A,1) THEN 450
446 NEXT : GOTO 440
450 ON A GOTO 460,470,480
460 RUN
470 LIST
480 NEW
    
```

### 3. lista

```

...
375 GET A$ : IF A$<>" THEN 375
...
    
```

## Diagramszerkesztő

Ha egy előadásban, tanulmányban adatok vagy adathalmazok időbeni vagy valamilyen összefüggés szerinti változását szemléletes módon kívánjuk bemutatni, gyakorta élünk a táblázatok, különböző diagramok nyújtotta lehetőségekkel. Az Enterprise számítógép grafikai adottságai lehetővé teszik hisztogramok egyszerű szerkesztését, jól megkülönböztethető vonaltípusokkal.

Az elkészített diagramokat szalagra mentve vagy kinyomtatva munkánk során bármikor felhasználhatjuk.

Ilyen célra szerkesztett programot mutatok be. A programot áttekintve minden bizonnyal lesznek olyan olvasók, akikben felmerül, hogy lehetett volna egyszerűbben, szerkezetében jobban elrendezetten megírni ezt a programot. Igazuk van. A program magán viseli mindazon jegyeket, amelyek bizonyítják: egy program sohasem lesz kész, mert használatba vételekor újabb és újabb igények és ötletek merülnek fel, amiket meg kell oldani, ha másért nem, hát a próba kedvéért. Így készült ez a program is.

```

VÁLASZTÉK I ----- I
I OSZLOPDIAGRAM -> 1 I
I I I
I KÖRDIAGRAM -> 2 I
I I I
I DIAGRAMRAJZOLÓ-> 3 I
I I I
I VEGE -----> 4 I
I I I
I ----- I
    
```

ÜZEMMÓD (1, 2, 3, 4)

VÁLASZTÁS --> =

Betöltés után a megjelenő menüből választhatunk (lásd az ábrát): ha 4-nél nagyobb, illetve 1-nél kisebb számot gépelünk be, a kurzor a helyén marad. Oszlopdigramot választva (a

410-es sortól) a gép megkérdezi, álló vagy vízszintes elrendezést akarunk-e. A kérdésre H vagy V betű leütésével a program az 1260-as vagy a 460-as sortól fut tovább. A 480-as és az 1280-as sor utasítása a KOOR nevű függvényt hívja meg, s az ebben foglalt utasításoknak megfelelően jelenik meg a képernyőn a koordináta-rendszer két féltengelye. A koordinátatengelyek felrajzolása után a program a függvényt hívó utasítás utáni sortól folytatódik (490-es vagy 1290-es). A grafikus képernyő alatt az első sorban megjelenik a kilépésre vonatkozó utasítás (M1=0).

Ezután következik az adatok begépelése. Az oszlop magassága és hossza le van határolva az 550-es sor és az 1360-as sor utasításainak megfelelően.

Az adatok begépelése után a gép megkérdezi, hogy sík- vagy térbeli legyen-e (S/T) az ábrázolás. S betű leütése után a SOSZLOP, T után a TOSZLOP függvényben meghatározottak, vízszintes oszlopok esetében a VSOSZLOP és a VTOSZLOP függvényben meghatározottak szerint rajzolja fel a gép az oszlop képét.

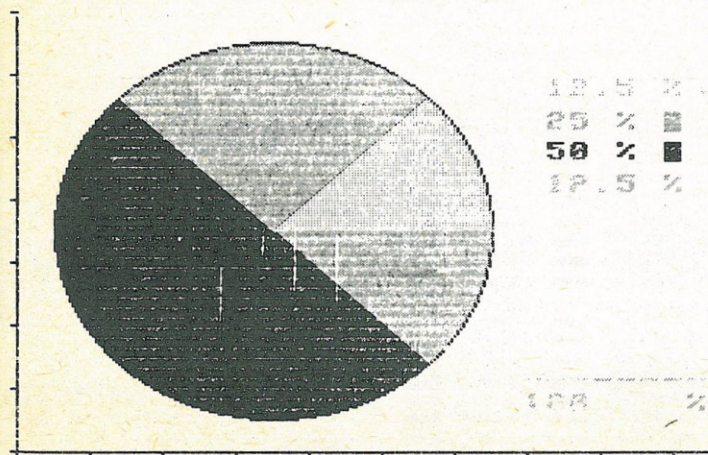
Ha befejeztük a szerkesztést, az M1=0 begépelésével kilépünk.

### Kördiagram

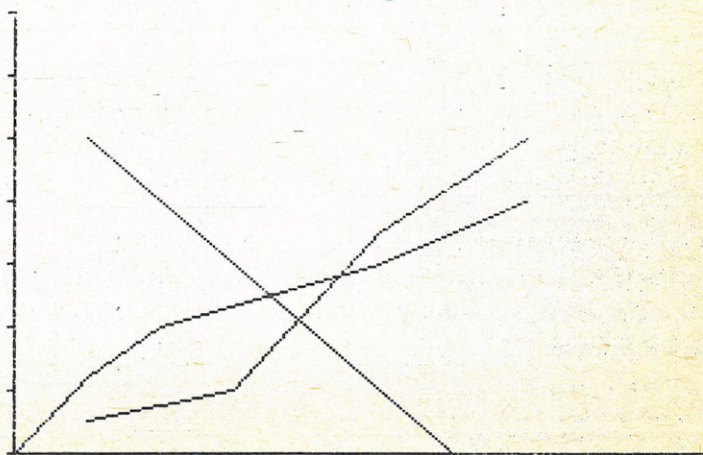
Kördiagram-szerkesztő programja a 780-as sortól az 1210-es sorig tart. A 840-es sor hívja a KOOR függvényt. A koordinátatengely felrajzolása után a kilépés módjára ad utasítást (S=0) a képernyőn megjelenő szöveg. Ezek után kéri be a gép a szerkesztéshez szükséges adatokat, majd felrajzolja a megadott adatok alapján a körcikket, és az 1120-as sor utasítása meghívja az SZ jelű függvényt, amely kiszámolja a körcikk százalékos értékét, s kiírja a képernyőre. Ezután a program végrehajtása a CALL utasítás utáni sortól folytatódik.

Az 1160-as sor utasítására a program visszatér a 950-es sorra. Így a szerkesztés tovább folytatható. Ha a kör teljes területét felhasználtuk, akkor a gép kiírja az egyes körcikk százalékaiknak összegét, a 100%-ot és kilép a programból (1210-es

Kördiagram



Vonaldiagram



sor). Ha a körcikk nyílásszögeinek összege  $\Sigma S = K > 360^\circ$ , akkor a szerkesztést újra kezdjük (1140-es sor).

## Grafikonrajzoló

A program (1570—1880-as sor) alkalmas arra, hogy különböző értékpárokkal meghatározott pontokat összekötő görbét (két pont között egyenes szakasszal) felrajzoljunk. Ez igen hasznos lehet mérésnél kapott értékpárok kiértékelésénél, gépek munkapontjának meghatározásánál, jelleggörbék felvételénél.

Az értékpárok tárolására két numerikus tömböt T (1 RO Q) és Z (1 TO Q) hozunk létre. A gép megkérdezi, hogy hány értékpárunk van, melyet feldolgozunk (Q).

A megadott értékpár számának megfelelő mélységű lesz a "T" és "Z" tömb. (Q =) 0 begépelésekor kilépünk a programból. Ezután a gép bekéri az értékpárokat (1650—1750-es sorok), melyeket az előzőekben létrehozott tömbökben tárol.

Gépeléskor a beírt adatok helyességét meg kell erősítenünk az "Y" billentyű leütésével. Ezután írja ki a gép, hogy P érvényes adatpár beírása megtörtént. Az utolsó érvényes (Q) értékpár beírása után következik a rajzolás, ami előtt azonban még a gép kérdésére meg kell adni a vonal típusát (1810-es sor). Az 1830-as sortól kezdődő ciklusban foglalt utasításoknak megfelelően rajzolódik meg a görbe.

## Új adatpárok bevitele

Miután a rajzolás befejeződött, az 1880-as sor utasítására a program az 1590-es sortól újra futni fog, s egyben törli a tömbökből az előzőleg beírt adatokat. Így válik lehetővé új adatpárok bevitele és új görbe rajzolása, az előzőleg felrajzolt koordináta-rendszerbe. Még egy ötlet: ha értékpáronként szerkesztünk, kedvünk szerinti ábrát rajzolhatunk rövid, egyenes szakasszal — ha van türelmünk. Ez megoldható. Az 1790-es és az 1800-as sorok utasításait az 1580-as és az 1590-es sorok közé (1790→1584; 1800→1586) írjuk be, majd az 1790-es és az 1800-as sorokat átírjuk. Az 1790-be egy kérdőjelet, s ha kell, megjegyzést írunk. Az 1800-as sorba írjuk: DEF RAJZOLÓ.

Az 1880-as sorba a RUN 1590 helyett END DEF-t írunk. Ezzel hívható függvényé tettük a rajzolóciklust. Az 1740-es és az 1750-es sorok közé a következő utasításokat írjuk:

```
1742 INPUT PROMPT "Rajzol v. tovább ? R/N = ":E $
1744 LET E$ = UCASE$(E$)
```

```
1746 IF E$ = "R" THEN CALL RAJZOLÓ
```

Az 1750-es és az 1760-as sorok közé a következőket:

```
1754 INPUT PROMPT "Felrajzolja ? I/N = ":C $
```

```
1756 C$ = UCASE$(C$)
```

```
1758 IF C$ = "I" THEN CALL RAJZOLÓ
```

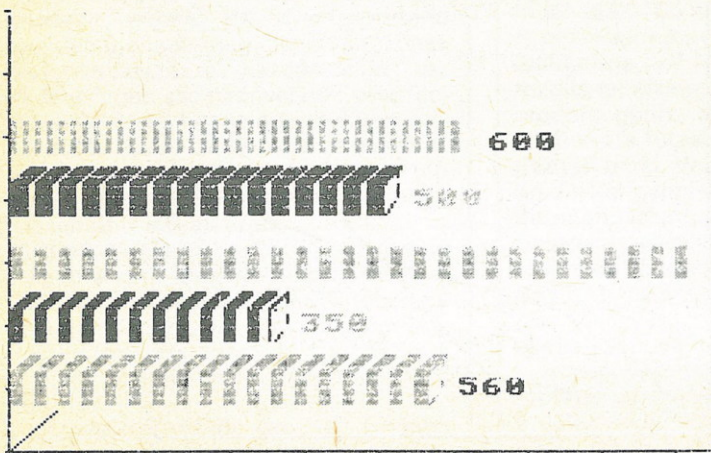
S végül az 1760-as sorba a ! jel helyett írjuk be: RUN 1590. Ezzel a kis kiegészítéssel szemünk előtt, szakaszonként rajzol a gép, amely minden beolvasási ciklusban megkérdezi, fel kívánjuk-e rajzolni a megadott értékpárhoz tartozó egyenes szakaszt. Ha a válasz R, akkor felrajzolja, ha nem (N), akkor a következő értékpár bekérése következik. Ha a tömbök megteltek, az 1754-es sor utasítása megkérdezi: felrajzolja-e a görbét. Ha a válasz "I", akkor a rajzolást végrehajtja, ha "N", akkor az 1760-as sor utasítására törlődik a beírt adathalmaz, és a program ismét az 1590-es sortól folytatódik. Próbáljuk ki!

Remélem, hogy ez a kis szerkesztőprogram, amely még javítható, bővíthető, igény szerint átalakítható — egyszerűen „be-fejezhető” —, segítségére lesz mindazoknak, akik hasonló feladatok megoldására vállalkoznak.

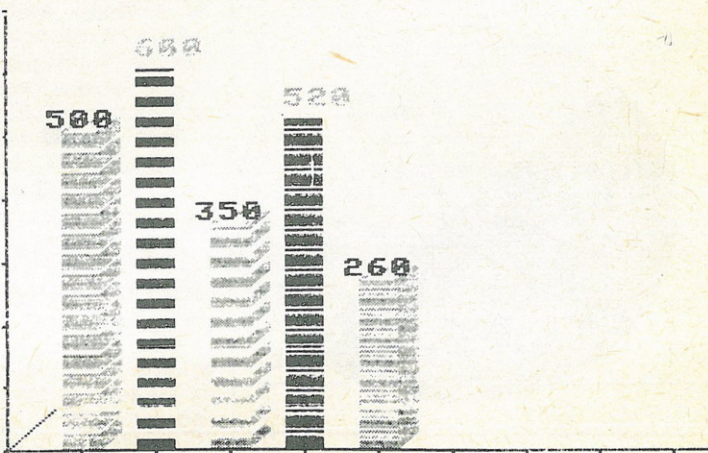
Szabó Gyula

```
100 'DIAGRAM_SZERK-3. nyomtava 89.01.25
110 TEXT
120 PRINT AT 3,12: "-----"
130 PRINT AT 4,12: "I"
140 PRINT AT 5,12: "1 OSZLOPDIAGRAM --> 1 I"
150 PRINT AT 6,12: "I"
160 PRINT AT 7,12: "I KÖRDIAGRAM --> 2 I"
170 PRINT AT 8,12: "I"
180 PRINT AT 9,12: "1 DIAGRAMRAJZOLÓ-> 3 I"
190 PRINT AT 10,12: "I"
200 PRINT AT 11,12: "1 VEGE -----> 4 I"
210 PRINT AT 12,12: "I"
220 PRINT AT 13,12: "-----":PRINT
230 PRINT AT 3,2: " "
240 PRINT AT 4,2: "VALASZTEK"
250 PRINT AT 13,5: " "
260 PRINT AT 16,5: "UZEMMOD (1,2,3,4)"
270 PRINT AT 17,5: " "
280 INPUT AT 18,5,PROMPT "VALASZTAS --> ":V1
290 IF V1=0 OR V1>4 THEN 260
300 SELECT CASE V1
310 CASE 1
320 GOTO 410
330 CASE 2
340 GOTO 790
350 CASE 3
360 GOTO 1570
370 CASE ELSE
380 GOTO 2540
390 END SELECT
400 !
410 'oszlodpdiagram
420 INPUT PROMPT "Vízszintes vagy álló oszlopokat akar H/V =":M$
430 LET M$=UCASE$(M$)
440 IF M$>"V" AND M$<"H" THEN 420
450 IF M$="V" THEN 460
450 IF M$="H" THEN 1260
```

Vízszintes oszlopdiagram



Függőleges oszlopdiagram



```

460 GRAPHICS 4
470 SET INK 3
480 CALL KOOR
490 PLOT 0,0;75,75
500 'Szerkesztes
510 '
520 PRINT AT 1,26:" KILEPES M1=0"
530 INPUT PROMPT "Oszlop mag. M1=":M1
540 IF M1=0 THEN 2540
550 IF M1<0 OR M1>600 THEN 530
560 INPUT PROMPT "Oszlop X Koord. E1=":E1
570 IF E1<50 THEN 560
580 INPUT PROMPT "Oszlop szine (1-3) S1=":S1
590 IF S1<1 OR S1>3 THEN 580
600 INPUT PROMPT "Vonal tipus(1-14)U=":U
610 IF U<1 OR U>14 THEN 600
620 SET INK S1
630 SET LINE STYLE U
640 LET H=E1
650 LET Z=M1
660 LET K1=Z+50
670 LET T=S1+1
680 IF K1<650 THEN LET K1=650
690 INPUT PROMPT "Abrazolas sik v.terblj legyen S/T=":V$
700 LET V$=UCASE$(V$)
705 IF V$<>"S" AND V$<>"T" THEN 690
710 IF V$="S" THEN CALL SOSZLOP
720 IF V$="T" THEN CALL TOSZLOP
730 PRINT :PRINT :PRINT :PRINT
740 '
750 PRINT :PRINT :PRINT :PRINT
760 GOTO 520
770 '
780 !KORDIAGRAM
790 '
800 GRAPHICS 4
810 '
820 LET O=0
830 SET INK 3
840 CALL KOOR
850 'SZERKESZTES
860 OPTION ANGLE DEGREES
870 PRINT AT 1,25:"kilepes S=0"
880 INPUT PROMPT "Kör középpontja X->":X
890 IF X>500 OR X<350 THEN 880
900 INPUT PROMPT "Kör középpontja Y->":Y
910 IF Y>350 OR Y<350 THEN 900
920 INPUT PROMPT "Kör sugara R->":R
930 IF R>350 THEN 920
940 LET X=X+10:LET Y=Y+10
950 PLOT X,Y,ELLIPSE R,R
960 PLOT X,Y,K+R,Y+X,Y
970 OPTION ANGLE DEGREES
980 INPUT AT 1,1,PROMPT " NYILAS-SZÖG ->=:S
990 IF S=0 THEN 2640 SZIN(1-3-ig)->=:1
1000 INPUT AT 2,1,PROMPT " SZIN(1-3-ig)->=:1
1010 IF I<1 OR I>3 THEN 1000 ELZÖ SZÖG ->=:S1
1020 INPUT AT 3,1,PROMPT "
1030 SET INK 1
1040 LET K=S+51
1050 PLOT ANGLE K
1060 PLOT FORWARD R;
1070 PLOT BACK R,
1080 PLOT ANGLE S1+S/2
1090 PLOT FORWARD R/2,PAINT
1100 PLOT BACK R/2,
1110 LET O=O+50
1120 CALL SZ
1130 IF K<360 THEN 1170
1140 IF K<360 THEN 790
1150 PRINT AT 3,20:"Ek=" K,
1160 GOTO 950
1170 PLOT 700,150,
1180 PRINT #101:"-----"
1190 PLOT 670,100,
1200 PRINT #101:(K/K)*100 " Z"
1210 GOTO 2540
1220 !Képtelvetel ":vsave" utasissal.
1230 !Keszult 1988.11.06. Dorogon Sz.Gy.
1240 !Vizszintes "oszlodiagram"
1250 '
1260 GRAPHICS 4
1270 SET INK 3
1280 CALL KOOR
1290 PLOT 0,0;75,75
1300 'SZERKESZTES
1310 PRINT
1320 PRINT AT 1,26:" KILEPES M1=0"
1330 PRINT
1340 INPUT PROMPT "Oszlop hossz M1=":M1
1350 IF M1=0 THEN 2540
1360 IF M1<100 THEN 1340
1370 INPUT PROMPT "Oszlop Y koord. E1=":E1
1380 IF E1<50 OR E1>600 THEN 1370
1390 INPUT PROMPT "Oszlop szine (1-3) S1=":S1
1400 IF S1<1 OR S1>3 THEN 1390
1410 INPUT PROMPT "Vonal tipus(1-14)U=":U
1420 IF U<1 OR U>14 THEN 1410
1430 SET INK S1
1440 SET LINE STYLE U
1450 LET H=E1
1460 LET Z=M1
1470 LET K1=Z+10
1480 LET T=S1+1
1490 IF K1<1100 THEN LET K1=1100
1500 INPUT PROMPT "Abrazolas sik v.terblj legyen S/T=":H$
1510 LET H$=UCASE$(H$)
1515 IF H$<>"S" AND H$<>"T" THEN 1500
1520 IF H$="S" THEN CALL VSZGZLOP
1530 IF H$="T" THEN CALL VTOSZLOP
1540 PRINT :PRINT :PRINT :PRINT
1550 GOTO 1520
1560 '
1570 !grafikon rajzolo
1580 GRAPHICS 4
1590 INPUT PROMPT "Grafikon hany osszetartozo ertekekkel rendelkezik ? kilepes
>0=0; 0=":0
1600 IF 0=0 THEN 2540
1610 '
1620 NUMERIC T(1 TO 0),Z(1 TO 0)
1630 '
1640 '
1650 PRINT "Kerem az osszetartozo ertekeparokat X,Y"
1660 FOR W=1 TO 0

```

```

1670 INPUT PROMPT "X=":T(W)
1680 INPUT PROMPT "Y=":Z(W)
1690 LET T(W)=T(W)+10:LET Z(W)=Z(W)+10
1700 LET R=W
1710 INPUT PROMPT "az adatok jok ? Y/N=":Y$
1720 LET Y$=UCASE$(Y$)
1730 IF Y$<>"Y" THEN 1670
1740 PRINT "P=" R
1750 NEXT W
1760 '
1770 !rajzolas
1780 '
1790 SET INK 3
1800 CALL KOOR
1810 INPUT PROMPT "Vonal tipus 1-14 S=":S1
1820 SET LINE STYLE S1
1830 FOR D=1 TO R
1840 LET L=D+1
1850 IF L>R THEN LET L=R
1860 PLOT T(D),Z(D);T(L),Z(L)
1870 NEXT D
1880 RUN 1590
1890 '
1900 !KORDINATA FELRAJZOLASA
1910 DEF KOOR
1920 PLOT 10,10;1200,10,
1930 PLOT 10,10;10,700,
1940 FOR N=10 TO 1210 STEP 100
1950 PLOT N,0;N,10,
1960 NEXT N
1970 FOR M=10 TO 710 STEP 100
1980 PLOT 10,M;0,M,
1990 NEXT M
2000 END DEF
2010 '
2020 DEF SZ
2030 LET C=0
2040 LET A1=100*S/360
2050 LET B1=ROUND(A1,2)
2060 LET N=650-C
2070 PLOT 700,N,
2080 PRINT #101:B1 "% B"
2090 END DEF
2100 '
2110 DEF TOSZLOP
2120 FOR R=1 TO 50
2130 PLOT H-15+R,12;H-15+R,12+Z
2140 PLOT H+35,12;H+35+R/2,12+R/2;H+35+R/2,Z+12+R/2
2150 PLOT H-15,Z+18;H+12,Z+35;H+60,Z+35;H+35,18+Z
2160 NEXT R
2170 IF T>3 THEN LET T=2
2180 SET INK T
2190 PLOT H-70,K1,
2200 PRINT #101:Z
2210 END DEF
2220 '
2230 DEF VTOSZLOP
2240 FOR R=1 TO 50
2250 PLOT 12,H-15+R;12+Z,H-15+R
2260 PLOT 12,H+35;12+R/2,H+35+R/2;Z+12+R/2,H+35+R/2
2270 PLOT Z+18,H-15;Z+35,H+12;Z+35,H+60;18+Z,H+35
2280 NEXT R
2290 IF T>3 THEN LET T=2
2300 SET INK T
2310 PLOT K1+10,H+25,
2320 PRINT #101:Z
2330 END DEF
2340 '
2350 DEF SOSZLOP
2360 FOR R=1 TO 50
2370 PLOT H-15+R,12;H-15+R,12+Z
2380 NEXT R
2390 IF T>3 THEN LET T=2
2400 SET INK T
2410 PLOT H-50,Z+60,
2420 PRINT #101:Z
2430 END DEF
2440 '
2450 DEF VSZGZLOP
2460 FOR R=1 TO 50
2470 PLOT 12,H-15+R;12+Z,H-15+R
2480 NEXT R
2490 IF T>3 THEN LET T=2
2500 SET INK T
2510 PLOT Z+20,H+15,
2520 PRINT #101:Z
2530 END DEF
2540 END

```

**Minden kedden 17-től 20 óráig**  
**HCC ENTERPRISE klub**  
**a VSZM**  
**Közösségi Házban**  
**(Bp. XI., fehérvári út 120.)**  
**Klubvezető: Romvári Gábor**  
**Telefon: 810-950/473**

## Érdekes görbék

Az Enterprise személyi számítógép megjelenésével igen olcsó és az árához képest jó képességű gép került forgalomba. A gép BASIC-interpretere igen sokat tud, de lassú. Ennek ellenére szerintem elterjedése csak a forgalmazókon múlik. Különösen kiemelendőnek tartom a gép grafikai képességeit. BASIC-ből is 640 x 720 felbontású grafikus lapot tud kezelni. A Nick chip ki tudja használni a felhasználói kézikönyvben írt 1280 x 720-as felbontást is. (Ezt egy későbbi cikkben ismertetjük majd.) A közzölt programok grafikus képernyője nem egészen ilyen felbontású. Az INIT alprogram megnyit egy grafikus csator-

## Enterprise gépre

nát a két színű, nagy felbontású grafikának, amit a 101-es csatornához rendel (lásd az 1. listát).

A sorszámok a programoknál nem fedik egymást, tehát külön-külön beírhatók, majd MERGE utasítással fűzhetők össze. Ha nem mindegyik programot próbáljuk ki, akkor az INIT alprogramot mindig hozzá kell írni a programhoz. De megpróbálhatjuk saját grafikus képernyő készítését is.

A programok mind paraméteres görbékét ábrázolnak. Természetesen néhol lenne egyszerűbb megoldás is, például a teknőcparancsok alkalmazása. Én azért nem használtam ezeket, hogy szemléltessem két szögfüggvény összetételével, milyen bonyolultnak tűnő görbék lehet kapni.

### 1. lista

```

3170 DEF INIT
8180 SET VIDEO COLOR 0
8185 ! 2 szín 640 keppont
8190 SET VIDEO MODE 1
8195 ! nagyfelbontasu grafikus lap
8200 SET VIDEO X 38
8210 SET VIDEO Y 24
8220 OPEN #101:"VIDEO:"
8222 ! grafikus csatorna
8224 ! alapertelmezes
8226 ! így nem kell a csatornaszamot
8228 ! kiírni a grafikus utasitasok ele
8230 SET PALETTE BLACK,WHITE
8240 DISPLAY AT 1 FROM 1 TO 24
8250 LET MAX_Y=828
8260 LET MAX_X=1214
8270 END DEF
    
```

### 2. lista

```

8000 DEF SIN_COS
8010 TEXT
8020 INPUT AT 5,5,PROMPT "Amplitudo>":AMPLITUDO
8030 IF ABS(AMPLITUDO)>2 THEN 210
8040 INPUT AT 7,8,PROMPT "Omega>":OMEGA
8050 INPUT AT 9,8,PROMPT "Fazis>":FAZIS
8060 LET FAZIS=FAZIS*PI/180
8080 LET LEPESKOZ=.2/OMEGA
8090 CALL INIT
8100 FOR X=0 TO 2*PI STEP LEPESKOZ
8110 PLOT 193*X,207*AMPLITUDO*SIN(OMEGA*X+FAZIS)+MAX_Y/2;
8120 NEXT X
8130 END DEF
    
```

### 3. lista

```

8360 DEF LISS8370 TEXT
8380 PRINT AT 2,8:"Az elso rezges adatai"
8390 INPUT AT 5,3,PROMPT "Amplitudo>":A1
8400 IF ABS(A1)>3 THEN 8390
8410 INPUT AT 7,7,PROMPT "Omega>":O1
8420 INPUT AT 9,7,PROMPT "Fazis>":F1
8430 PRINT AT 12,8:"A masodik rezges adatai"
8440 INPUT AT 15,3,PROMPT "Amplitudo>":A2
8450 IF ABS(A2)>4 THEN 8440
8460 INPUT AT 17,7,PROMPT "Omega>":O2
8470 INPUT AT 19,7,PROMPT "Fazis>":F2
8480 LET F1=F1*PI/180
8490 LET F2=F2*PI/180
8500 LET LEPESKOZ=.3/((O1+O2)+ABS(O1-O2))/2
8510 CALL INIT
8520 FOR X=0 TO 2*PI+LEPESKOZ STEP LEPESKOZ
8530 PLOT A1*193*SIN(O1*X+F1)+607,207*A2*SIN(O2*X+F2)+414;
8540 NEXT X
8550 END DEF
    
```

### Sinus- és cosinus-görbék

E két függvény csak annyiban különbözik, hogy egymáshoz képest 90 fokkal el vannak tolvá, pontosabban:

$$\sin(X+90) = \cos(X)$$

Ennek alapján elég, ha csak az egyik függvény ábrázolására írunk programot. Mert ha például a  $\sin(x)$  görbét ábrázoljuk, akkor a  $\cos(x)$  görbét úgy kaphatjuk meg, hogy az  $x$  ciklusváltozóhoz hozzáadunk 90 foknak megfelelő értéket.

Ezek után már csak az van hátra, hogy a görbékét jól helyezzük el a képernyőn. A PLOT utasításban szereplő konstans szorzók szerepe az, hogy a SIN függvényt megfelelően nagyra nyújtsák meg a képernyőn. A 193 éppen a MAX\_X-nek a  $2\pi$ -ed része, a 270 pedig a MAX\_Y-nak a negyede.

### Bemenő paraméterek

Amplitúdó: a görbét nyújtja  $y$  irányban.

Omega: a függvényt nyújtja  $x$  irányban.

Fázis: a függvény induló értéke.  $\cos(X)$ -nél 90 fok.

A lépésközt azért tettem az omegától függővé, hogy a rajzolás viszonylag gyors legyen, de a kép még ne legyen túlságosan szögletes (2. lista).

### Lissajous-görbék

Az előbbi sinus-, illetve cosinus-görbékét felfoghatjuk harmonikus rezgőmozgás grafikonjaiként is. Két, egymásra merőleges rezgés eredőjének általában bonyolultabb görbék felelnek meg. Ennek a demonstrálására is lehetne alkalmazni a következő programot.

A program először bekéri az első rezgés amplitúdóját, körfrekvenciáját, fázisát, majd ugyanezt teszi a másik rezgéssel is. Ezzel a programmal előállítható kör és ellipszis egyaránt. Az omegák arányának változtatásával hasonló jellegű görbékét kapunk (3. lista).

## Egyéb paraméteres görbék

A paraméterrel megadható görbék három alapesete — a körön kívül — a ciklois, a szivgörbe és a csillaggörbe. A három görbét a következőképpen származtathatjuk:

**Ciklois:** egy egyenesen csúszás nélkül gördülő körnek egy kerületi pontja cikloist ír le (4. lista).

**Szivgörbe:** egy  $r$  sugarú kör külső oldalán csúszás nélkül gördülő másik  $r$  sugarú kör egy kerületi pontja szivgörbét ír le (5. lista).

**Csillaggörbe:** egy  $r$  sugarú kör belső oldalán csúszás nélkül gördülő  $r/4$  sugarú kör egy kerületi pontja csillaggörbét ír le (6. lista).

Valamivel bonyolultabbak a következő görbék. Nemcsak a képzési szabályuk miatt, hanem azért is, mert igen nehéz eltalálni helyüket a képernyőn.

Az első három egy téma három változata. Igen nehéz a második kettőnél megtalálni azt a konstans, amelyiket hatványozunk. Ha túl nagyra választjuk, akkor a koordináták túlságosan gyorsan nőnek, és így még körbe sem ért, már kimegy a képernyőről a következő pont. Érdekes, hogy milyen szemléletesen látszik a két görbéből az exponenciális és a logaritmus-függvény közötti lényeges különbség.

A 7. lista egy euklideszi spirált ábrázol, a 8. lista logaritmikus, a 9. lista exponenciális spirált ábrázol.

A 10. lista programja által létrehozott görbét a következőképpen lehet megkonstruálni. Vegyük kiindulásnak azt a Lissajous-görbét, amelynek paraméterei a következők:

Omega1=1  
Fázis1=90  
Omega2=2  
Fázis2=0

Az amplitúdók egyenlők ( $a$ ).

Így a kapott görbe paraméteres egyenlete:  $x = a \cdot \cos(t)$   $y = a \cdot \sin(2 \cdot t)$ . Hogy nyolcas legyen belőle, cseréljük fel a két koordinátafüggvényt. Így az  $x = a \cdot \sin(2 \cdot t)$  és  $y = a \cdot \cos(t)$  lesz a két egyenlet.

Ahhoz, hogy a görbe periódusonként csak egyszer messe önmagát, először tükrözzük a nyolcast az  $y$  tengelyre ( $x = -a \cdot \sin(2 \cdot t)$   $y = a \cdot \cos(t)$ ).

Hogy a görbét  $x$  irányban széthúzzuk, a következő módosítást hajtjuk végre az egyenleteken:

$x = -a \cdot \sin(4 \cdot \pi \cdot t) + a \cdot 3 \cdot t$   
 $y = a \cdot \cos(2 \cdot \pi \cdot t)$ .

Ha a görbénél a sinus- és a cosinus-függvények argumentumában szereplő konstans szorzótényezőt megváltoztatjuk, érdekes aszimmetrikus görbéket kaphatunk.

Koller Pál

## 4. lista

```
8600 DEF CIKLOISZ
8610 RX,RY=1
8620 IF RX<=.5 THEN LET ST=8
8630 IF RX<=1 AND RX>.5 THEN ST=4
8640 IF RX>1 THEN LET ST=2
8650 FOR X=0 TO ST*PI STEP .1
8660 PLOT 85*RX*(X-SIN(X)),103*RY*(1-COS(X))+414;
8670 NEXT X
8680 END DEF
```

## 5. lista

```
8690 DEF SZIVGORBE
8700 LET RX,RY=1
8710 FOR X=0 TO 2*PI STEP .1
8720 PLOT 80*RX*(2*COS(X)-COS(2*X))+607,75*RY*(2*SIN(X)-SIN(2*X))+414;
8730 NEXT X
8740 END DEF
```

## 6. lista

```
8750 DEF CSILLAG
8760 RX,RY=1
8770 FOR X=0 TO 2*PI STEP .1
8780 PLOT 193*RX*COS(X)^3+607,207*RY*SIN(X)^3+414;
8790 NEXT X
8800 END DEF
```

## 7. lista

```
8810 DEF SPIRAL
8820 RX,RY=1
8830 FOR X=0 TO 16*PI STEP .1
8840 PLOT 7*X*SIN(X)+607,7*X*COS(X)+414;
8850 NEXT X
8860 END DEF
```

## 8. lista

```
1000 DEF LG_SPIRAL
1010 CALL INIT
1020 FOR X=0 TO 16*PI STEP .1
1030 PLOT 75*LOG(X+.9)*SIN(X)+607,75*LOG(X+.9)*COS(X)+414;
1040 NEXT X
1050 END DEF
```

## 9. lista

```
1060 DEF EXP_SPIRAL
1070 CALL INIT
1080 FOR X=0 TO 8*PI STEP .1
1090 PLOT 3*1.2^X*SIN(X)+607,3*1.2^X*COS(X)+414;
1100 NEXT X
1110 END DEF
```

## 10. lista

```
1200 DEF SUJTAS
1210 CALL INIT
1220 FOR X=0 TO PI/2+.04 STEP .02
1230 PLOT -100*SIN(10*PI*X)+600*X+100,100*COS(5*PI*X)+414;
1240 NEXT X
1250 END DEF
```

## 11. lista

```
1300 DEF CSAVAR
1310 CALL INIT
1320 FOR X=-1 TO 13*PI+1 STEP .2
1330 PLOT 50*COS(X)+25*X+100,80*SIN(-X)+414;
1340 NEXT X
1350 END DEF
```

## ENTERPRISE—TOTÓ 1989. II.

A CENTRUM Áruházak Vállalat a Mikroszámítógép Magazinval közösen kétfordulós játékpályázatot hirdetett. A kérdésekre a TOTÓ ismert szabályai szerint 1, 2, illetve X jelöléssel kell válaszolni. Beküldendő a 14 helyes válasz a kérdések sorrendjében a következő címre:

### CENTRUMNAGYKER

1431 Budapest, Pf. 195

Minden megfejtést értékelünk.

A játékosok mindkét fordulóban külön-külön nyerhetnek.

A telitalálatos szelvények kitöltői között a CENTRUM fordulónként az alábbi nyereményeket sorsolja ki:

**1 db Spectrum-emulátort,  
3 db Enterprise-eget (mouse),**

**10 db új programcsomagot (kazetta).**

Aki mind a két szelvényt helyesen töltötte ki, az a szuper-sorsoláson is részt vesz, és az őszi BNV-n

**EPSON RX 80 típusú nyomtatót**

nyerhet. Vigaszdíj is lesz.

A második TOTÓ beküldési határideje 1989. szeptember 8. Ne felejtse a megoldással együtt beküldeni nevét és pontos címét. A vigaszdíj sorsolásán a találatoktól függetlenül az vesz részt, aki gépének gyártási számát is feltünteti.

1. A nyugati prospektusok állítása szerint az Enterprise 128K számítógép meddig bővíthető?

- 1 576 kb-át       2 4080 kb-át       3 320 kb-át

2. A CP/M operációs rendszer mely verziójával kompatibilis az Enterprise-on futtatható dBASE, WordStar, Supercalc stb.?

- 1 2.2       2 2.0       3 2.1

3. Mekkora az Enterprise Logo által definiálható maximális pixel-méret?

- 1 672 x 512       2 672 x 256       3 1280 x 720

4. Mekkora az Enterprise 128K számítógép fizikai méretei?

- 1 40 x 27 x       2 45 x 27 x 3,5 cm       3 40 x 20 x 2,5 cm 2,5 cm

5. Mi az Enterprise őseinek tekintett sakkszámítógép neve?

- 1 Simultano       2 Mephisto       3 Conquistador

6. Melyik állításnak felel meg a Zzip BASIC Compiler verziója?

- 1 integer       2 non integer       3 turbo

7. A felsorolt programok közül melyik jelent meg áruházi forgalomban?

- 1 CAULDRON       2 BEATCHA       3 DRUID

8. Mennyi a hálózatba kapcsolható Enterprise gépek maximális száma?

- 1 32       2 30       3 16

9. Melyik az eddigi legnagyobb példányszámban eladott program?

- 1 Star Strike 3D       2 Cauldron       3 Magic Ball

10. Mikor jelent meg az első Enterprise 64K számítógép a nyugat-európai számítógéppiacon?

- 1 1985       2 1984       3 1987

11. Melyik cég készítette a Batman nevű programot?

- 1 Bubble Bus Software       2 Entersoft Ltd.       3 Ocean Software Ltd.

12. A „The Last Ninja II.” szintjeinek száma?

- 1 6 szintes       2 7 szintes       3 8 szintes

13. Az Enterprise PLUS cartridge-ban rejlő EPROM-foglalatok száma?

- 1 2       2 1       3 3

13+1. Mely érintkezők nincsenek bekötve a SERIAL/NET csatlakozófelületén?

- 1 2 és 7       2 1 és 8       3 2 és 8

**KÉT FORDULÓ,  
HÁROM SORSOLÁS!**

**ENTERPRISE  
COMPUTERS**

Az 1989/7-es számunkban megjelent első forduló megfejtése a következő: 2 1 1 X 2 X 1 2 1 2 2 X 2 X



## SPOKE-SPEEK

Aki olyan programot szeretne írni, amely szöveges adatokat tárol (szótár, lexikon, adatnyilvántartás stb.), előbb-utóbb szüknek érzi az IS BASIC STRING utasításával lefoglalható tömbméretet. Mindent megpróbál, hogy minél több helyet biztosítson egyre szaporodó adatainak.

A kézikönyv tanulmányozásakor felfigyel arra, hogy a gép alapkiépítésben 128 k-s, és kissé értetlenül rázza a fejét. Az ugye tény, hogy az IS BASIC is lefoglal egy kis helyet, s ha floppyt is csatlakoztatunk, ez a lefoglalt terület csak nő, noha nem végzetesen. Helyet foglal továbbá az interpreter, adminisztrációs célokra is. Ezt az INFO billentyű megnyomásakor tapasztalja. Nem érti azonban, hogy az IS BASIC miért csak 64 k-t kezel. Az IS BASIC végül is gondol ránk, s két kulcsszóval segít gondjainkon, hogy a még szabad 64 k-t is hasznosíthassuk. Ez a két kulcsszó a jól ismert POKE-PEEK testvére, egy S betűvel megtoldva.

Mi a különbség a két POKE, illetve PEEK között?

A POKE-PEEK utasítás, az első (nevezük elsőnek) 64 k memóriába történő (0-255 nagyságú) számok beírására és kiolvasására ad lehetőséget, ami annyit jelent, hogy a 0. memóriacím-től a 65536. memóriacímig írhatunk, illetve olvashatunk. Ez formailag így néz ki:

POKE a,b illetve PEEK (a)

a=0-65536

b=0-255

A SPOKE-SPEEK utasítás a teljes 128 k memória kezelését teszi lehetővé, azzal az apró nehezítéssel, hogy a szegmenskiostást figyelembe kell venni.

Az Enterprise 128 memóriája 8 db, egyenként 16 k-s (16384 bájtos) szegmensre van felosztva. A szegmensekre 0-255 értékű számmal hivatkozhatunk. A SPOKE formailag így használható:

SPOKE a,b,c illetve SPEEK (a,b)

a=a szegmens címe (0-255)

b=a szegmens belüli memóriarekesz címe (0-16383)

c=a memóriában elhelyezendő szám (0-255)

Az első 8 szegmenst végignézhetjük az alábbi módon:

10 FOR i=0 TO 7

20 FOR ii=0 TO 16383

30 PRINT SPEEK (i, ii)

40 NEXT ii

50 NEXT i

Elárulom, hogy nincs sok értelme, ugyanis azt láthatjuk csak, hogy telis-tele van írva mind a 8 szegmens, tehát végig kell nézni mind a 255 szegmencímet. Erre azonban az alábbi kis program talán célszerűbb:

10 FOR i=0 TO 255

20 Print i, SPEEK (i, x)

30 NEXT i

Ha az x értékét megváltoztatva többször futtatjuk a programot, akkor a következő megállapításra juthatunk. Az Enterprise nyolc használatos szegmense a 248, 249, 250, 251, 252, 253, 254, 255. Ezek szerint a szegmensek címei memóriabővítés esetén lefelé bővülnek. Az IS BASIC a 248-251 szegmenstartományt használja. A számunkra lényeges plusz 64 k a 252-255-ös szegmenseken helyezkedik el. A lényeg, hogy rendelkezésre áll a szabad, „szűz” 64 k, valamint a kezeléséhez két utasítás.

Hogyan használjuk? Segítségül még álljon itt egy minta-probléma:

A 252-es szegmens 0. rekeszében szeretnénk tárolni például egy „A” betűt. Ez nem okozhat nagyobb fejtörést, mert tudjuk, hogy mindegyik karakterünk megfeleltethető egy, a 32-159 értékű számtartományba tartozó számmal. Ennek ismeretében pedig a megoldás:

SPOKE 252,0,ORD („A”)

Még egy megjegyzés. Ekkora tárterület kezelése már felveti egy fordító (compiler) szükségességét is. Hiszen aki megszerette a BASIC-programozást, az már csak a szerelem miatt sem mond le róla, noha tudja, hogy vannak sokkal hatékonyabb célnyelvek is.

Papp Miklós

## Mi a manó?

A Centrum Áruházakban 498 forintos áron megjelent a Quelle katalógusban is fellelhető TV-Computer átkapcsoló szerkezet. Ez az eszköz lehetőséget nyújt a hazai számítógépet használók népes táborának arra, hogy a tévékészüléket, illetve számítógépet úgy használhassa, hogy a csatlakozókábeleket ne kelljen minduntalan ki-be húzogatni a tévékészülékből. A 75 ohmos antennakábelt kell bedugni a TV feliratú csatlakozórészbe, a szabadon maradó, Computer feliratú aljzatba pedig számítógépünk RF antennacsatlakozóját kell bedugni. A kis doboz saját antennacsatlakozóját ezután biztonságosan dughatjuk be a tévékészülékünk antennabemenetébe. Hasznos dolog, hogy a négy színben forgalomba hozott átkapcsoló felragasztható a bútor oldalfalára.

\*

Kedvező fogadtatásra talált az Úttörő Áruházban a Public Domain kazetták másolása (az ott megvásárolt kazetták B oldalára ingyen másolják a programot). Az eddig így eladott kazetták száma megközelíti az ezret. A sikeren felbuzdulva rövidesen Miskolcon, Szegeden és Győrött is megteremtik a lehetőséget a Public Domain kazetták másolására.

\*

A Kerminél, a MEEI-nél és a Postánál már vizsgálják a 180 kb-át kapacitású hajlékonylemez-meghajtót, amelyet a Novotrade szándékozik forgalmazni.

\*

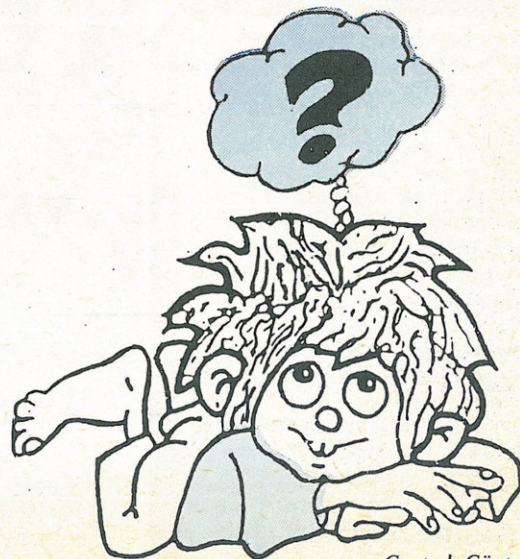
A Novotrade Rt. és a Centrum Áruházak különválása óta a Novotrade két új programmal, a Centrum pedig tíz újabb jattékkal jelentkezett a hazai Enterprise-piacon.

\*

Meg nem erősített hírek szerint újabb 2000 darab Enterprise 128K jelű számítógép érkezik az országba, de az importőr nem a Novotrade Rt. Ugyanakkor 4000 darab EP-64K jelű gép hamarosan behajózik Egyiptomba.

\*

Néhány kérdés, amelyre eddig nemigen kaphattunk választ. Miért kell a vevő személyi száma a License Agreement kártyára? Miért kell pótlólagos költség megfizetésére, illetve minimális értékvesztésre kényszeríteni azt, akinek egy 27256 EPROM-ja van a cartridge-ban? Miért hemzsegnek az angol kifejezésektől a dokumentációk? Csupán divat, vagy mélyebb értelmi indíttatású a csomagoláson a sok angol kifejezés?



Gaetsch Günterné rajza

# BÖRZE



## COBRA COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET  
1067 Budapest IX., Ilmtos út 7. Telefon: 478-180/388

**SZÁMLAKÉSZÍTÉSTŐL  
A KÖNYVELÉSIG**

### COBRACONTO

Ügyviteli programrendszer moduljai:

- számlakészítő,
- számlanyilvántartó,
- bér- és jövedelemszámfejtő,
- főkönyvi könyvelő,
- anyagnyilvántartó,
- általános könyvelési programok.

KÉRJEN RÉSZLETES TÁJÉKOZTATÓTI

## TUTTI

### ELECTROCOOP KISSZÖVETKEZET

- Eredeti számítógépek
- Kompatibilis XT/AT,  
386 konfigurációk
- Szünetmentes tápegységek
- Egyedi megrendelésfelvétel

Cím: Bp., Üllői út 81. 1091  
Tel.: 334-354  
Telex: 22-7230  
Telefax: 149-869



## ENET



Lokális hálózat  
IBM kompatibilis gépekre  
Külön hardver  
nem szükséges  
Ára gépenként:  
19 900 Ft + áfa  
ECOSOFT KISSZÖVETKEZET  
Tel.: 863-677

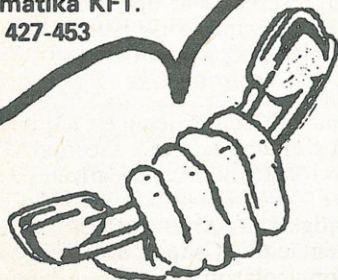
## ESZ ELEKTRONIKA

A legújabb ajánlatunkból:

- ASY-16 SZUPERMIKRO számítógépek (12 terminál, UNIX operációs rendszer)
  - IBM PC/XT-AT-kompatibilis számítógépek
  - megrendelő által definiált betűkészlettel rendelkező billentyűzetek
  - elektronikai elemek (integrált áramkörök, ellenállások, tranzisztorok)
  - monitordobozok, műszerházak
  - szoftvertermékek és fejlesztések
- KERESKEDELMI IRODA:  
1061 BUDAPEST  
LISZT FERENC TÉR 10.  
TEL.: 415-166, TELEX: 22-4378

## ARECO KFT.

a Mikropo KSZ fejlesztési témáinak jogutódja, felajánlja szabad fejlesztőmérnöki kapacitását. ARECO Informatika KFT.  
Tel.: 427-453



## NÁLUNK MÉG KAPHATÓ

Első könyvem a mikróról 30,— Ft  
Első könyvem a programról 30,— Ft:



Kereskedelmi Iroda  
1145 Budapest,  
Szugló u. 9-15.  
Telefon: 642-000/176-os,  
177-es mellék

ISKOLÁKNAK OKTATÓKNAK SZAKKÖRÖKNEK

## procontrol



IRÁNYÍTÁSTECHNIKA  
KISSZÖVETKEZET

MÁRKABOLT, BEMUTATÓTEREM  
62/21-165, 28 985  
Szeged, Kazinczy u. 8. Tel.: 12-259  
Telex: 82-726

- |                                    |                |
|------------------------------------|----------------|
| 386 AT 25 MHz toronygép            | 299 000 Ft-tól |
| 286 AT 16 MHz TURBO AT gép         | 98 000 Ft-tól  |
| - 86 XT 12 MHz                     |                |
| TURBO XT gép                       | 69 000 Ft-tól  |
| LAPTOP hordozható                  |                |
| LCD XT gép + modem + akku          | 149 000 Ft     |
| 12" zöld monochrom monitor TUV M31 | 4 900 Ft       |
| 5 1/4" DS DD Floppy lemez          | 79 Ft          |

és még 2000 árucikkről árjegyzék, azonnali kiszolgálás.



**BROTHER AX 15 típusú,**  
margarétafejes,  
elektronikus írógép

Megvásárolható:  
Budapest VI.,  
Népköztársaság útja 2.  
Tel.: 531-231



1146 Bp.,  
AJTÓSI DÜRER SOR 10.  
Levélcím:  
1393 Pf.: 319.  
Telefon: 421-974  
Telex: 22-6544

### SZAKÜZLETEK:

1. Bp. VI., Szinyei Merse u. 1.  
Tel.: 127-628. Tx.: 22-6684
2. 9022 Győr, Lukács S. u. 18.  
Tel.: 96-14880. Tx.: 02-4679
3. 3100 Salgótarján, Ady Endre u. 1.  
Tel.: 32-10971. Tx.: 22-9380

## Nagy felbontású monitorok

**Amikor az IBM elkezdte a CGA kártyák és monitorok forgalmazását, a szövegek megjelenítése nagyon egyszerű volt a 25 soros 80 karakteres képernyőn. A videojátékokban a színek és a grafika felbontása már abban az időben is finomabb volt, de az IBM eredeti, 25x80-as monokróm monitorának helyettesítésére a CGA által nyújtott felbontás még megfelelt.**

A játékprogramokhoz a CGA 320 x 200 pontos négy színű grafikája nem volt ugyan igazán jó, de a szoftvergyártóknak és felhasználóknak egy ideig megfelelt. Ugyanakkor ahogy fejlődött a technológia, úgy vált egyre nyilvánvalóbbá a grafika szerepének felértékelődése. Logikus lépésként a PC képernyőjének felbontása és a színek kezelése fejlődött, megjelentek a ma használatos EGA, VGA, MCGA szabványok, és ezt az irányt igazolta az első olyan, nagy mennyiségben gyártott személyi számítógép — a Macintosh —, amelynek csak grafikus képernyője van.

Ma már a színes monitorok széles skálája áll rendelkezésre a szabványos IBM PC vagy Macintosh számítógépekhez. Ezek a megjeleníthető színek számában és a felbontásban különböznek egymástól.

### Megközelítés — kétfelől

A nagy felbontású képernyőkkel kapcsolatban mind a Macintosh, mind az IBM kialakította a maga filozófiáját. Az a mód, ahogy a Macintosh kezeli a képernyőt, világosan tükrözi a gép tervezőinek grafikaközpontú alapállását. Valójában a Macintosh által kezelt tényleges képernyőnek és felbontásnak nincs szerepe a megjelenítés kialakításában. A képernyőn megjelenő szöveg és grafika egyaránt a ROM-ban tárolt, QUICKDRAW nevű, grafikai alapeljárásokat tartalmazó programcsomag segítségével jön létre. A gépen futó minden program ezeket a rutinokat használja fel a képernyő kezeléséhez. A grafikát egy — 32768 és +32767 között megadható logikai koordináta-rendszerben helyezi el, amit fizikailag ablakként — például az általánosan alkalmazott 512 x 342-es felbontású képernyőn — jelenít meg. A QUICKDRAW rutinok a logikai grafikai sík bármely részén képesek működni. Az aktív szegmens kijelölése a GrafPort eljárással lehetséges, ennek végrehajtása után minden szöveg és grafika az így kijelölt részre kerül.

Indításkor minden program ezt a GrafPort rutint aktivizálja, biztosítva, hogy a logikai sík is megjelenjen a képernyőn. A programok azonban másként is használhatják ezt a grafikát. Az animáció például nagyon egyszerűen megvalósítható olyan módon, hogy a logikai sík nem látható részén építjük fel a képeket, és más-más paraméterrel aktivizálva a GrafPort rutint, jelenítjük meg őket a képernyőn.

Ez a fajta működés teszi lehetővé, hogy a Macintoshon futó programok — a felbontástól függetlenül — bármilyen méretű képernyővel fussanak. A szabványos képernyőmemória méretét a Macintosh a hardvertől függetlenül tárolja a memóriájában, így ez bármikor megváltoztatható, ha egy programot betöltünk, vagy más monitort használunk. Ha a betöltött program ellenőrzi a képernyő méretparamétereit, akkor könnyen, automatikusan alkalmazkodik az adott felbontáshoz. Természetesen nem mindegyik program képes elvégezni ezt az ellenőrzést, de a legfontosabbak — mint a PageMaker, az Excel, a Word, a Filemaker Plus — ennek köszönhetően működnek egyszerűen bármely képernyőméret esetén.

### Kell a hely a grafikának

Korántsem ilyen egyszerű a nagy felbontású képernyős monitorok illesztése. Itt nincsenek eszközfüggetlen megjelenítő grafikai és szövegrutinok, és ha a felbontás felülmúlja az EGA kártyák 640 x 350, illetve a VGA kártyák 640 x 480-as felbontását, akkor nincs szabványos megoldás a képernyő kezelésére.

A PC-n futó grafika úgy működik, hogy a RAM-területből egy nagy tartományt lefoglal, s ezt felelteti meg a képernyő bit-térképének. A képernyő pontjainak a memória egy vagy több biteje tartozik. Monokróm (fekete-fehér) képernyő esetén elegendő egy bit, színes képernyőnél a

színek számától függő bitszóport felel meg a képernyő egy pontjának. A szükséges RAM mérete a felbontástól és a színek számától függ. 1 000 x 1 000-es képernyő kezelése 1 000 000 bitet, azaz kb. 128 kb-ot igényel monokróm képernyő esetén, míg 64 szín megjelenítéséhez az előbbi felbontásnál már 768 kb-ajtos memória szükséges, ami több, mint a DOS által használt terület! A közönséges EGA felbontáshoz  $640 \times 350 \times 4 = 896 000$  bit, azaz 128 kb-ajtos memória kell.

A VGA szabványon túlmutató szabványos PC-grafika hiánya alapvető gond a monitorgyártók számára. Míg az új, nagyobb felbontású grafikus kártyákat és a hozzájuk tartozó monitorokat viszonylag egyszerű előállítani, nagyon nehéz a megfelelő programokat, illetve képernyőmeghajtókat megírni. Más megoldási lehetőség azonban nincs: meg kell írni az új speciális meghajtókat, és a programokat is ennek megfelelően kell módosítani.

### CAD és DTP

Szerencsére ez nem azt jelenti, hogy a gyártóknak minden PC-s alkalmazói programhoz külön meghajtót kell írniuk: a nagy felbontású képernyőket használó programok száma korlátozott, és szinte szabványosnak tekinthető. Alapvetően két terület — a CAD-alkalmazásoké és a kiadványszerkesztő programoké — igényli ezt a képernyőtípust. A CAD területén egy meghatározott program van, az Autodesk cég által készített és forgalmazott AutoCad; a legtöbb igény kielégíthető, ha ehhez a programhoz elkészítik a meghajtó rutint. Ezenkívül az AutoCad használ még egy egyszerű, szabványos illesztést, az ADI-t, amely a nem szabványos felbontású monitorok illesztésére szolgál.

A kiadványszerkesztő programoknál szerencsére még kedvezőbb a helyzet, mivel többnyire vagy a Microsoft Windows vagy a Digital Research cég GEM programját használják a szöveg és grafika kialakításánál. Következésképpen a monitorgyártóknak csak ezekhez a programokhoz kell elkészíteniük azokat a meghajtókat, amelyek támogatják például a Ventura, a Pagemaker használatát. További előny, hogy a GEM és Windows alá írt egyéb programok — GEM Draw, Crossroads, Quartz, Excel — is képesek használni a finom felbontású képernyőt.

A GEM és a Windows tulajdonképpen utánozza a Macintosh QUICKDRAW-já-



nak az eszközfüggetlenségét, s ezzel biztosítja, hogy az egyéb programok felhasználják az előre definiált grafikai és szövegrutinokat, amelyekkel a menük, ablakok és egyéb felhasználói grafikák működnek. Minden hardverfüggést tartalmaz a GEM és a Windows. Ezeknek a meghajtóknak a megírása azonban nem csupán abból áll, hogy megmondjuk a GEM-nek és a Windowsnak: most tessék ezzel és ezzel a felbontással dolgozni. Meg kell oldani az új képernyőn lévő karakterek kezelését és a nyomtató-képernyő megfeleltetés kialakítását.

Persze a felhasználók nem túl boldogok, ha csak azt a néhány programot tudják futtatni az új képernyőjükön, és egy második grafikus képernyőt kell használniuk más alkalmazások futtatására, mivel ezekhez nincsenek meg a megfelelő meghajtók. A megoldás gyakran az, hogy több, szabványos grafikát kezelni tudó áramkört építenek a meghajtó kártyára, amely például tudja kezelni a szabványos CGA és az EGA grafikát is a nagy felbontású grafikus kártyával, és képes ezeket a kimeneteket megjeleníteni a kapcsolódó nagy felbontású monitor képernyőjén. Ez vagy úgy történik, hogy megjeleníti a CGA-, illetve EGA-képet egy ablakban, a képernyő közepén (egy képpont egy nagy felbontású képpontnak felel meg), vagy úgy skálázzák az EGA-képet, hogy minden EGA-képpont megfeleljen például a nagy felbontású képernyő négy képpontjának. Ilyen módon közel kitölthető a teljes képernyőfelület.

A választott megoldás függ a képernyő teljes felbontásától. Például egy EGA-képernyő, amely négy képpontot használ fel a nagy felbontású képernyőnél, azt igényli, hogy a felbontás 1280 x 700-as legyen.

### „Grafi(fi)ka”

A másik fő probléma, amivel a nagy felbontású képernyők tervezőinek szembe kell nézniük, a rajzolási sebesség. A Macintoshban és a PC-ben kis felbontás esetén minden képpontot a központi egység rajzol meg, és minél jobban megnöveljük a megrajzolandó képpontok számát, annál jobban lelassul a gép teljes működési sebessége. Hogy ezt elkerüljék, a gyártók valamilyen grafikai társprocesszort alkalmaznak a nagy felbontású illesztőkártyákon. Ezek teszik lehetővé, hogy a processzor a rajzolási feladatokat átadja a grafikai processzoroknak.

A grafikai társprocesszorok komplexitása a fejlett CRT-vezérlőktől (például a Hitachi HD63484 ACRTC = Advanced CRT Controller) az igazi 32 bites grafikai társprocesszorokig (Texas Instrumens TM34010) terjed.

De hogyan válasszák meg a gyártók a színeket és a monokróm felbontást, amikor megtervezik az új, nagyméretű képernyőket és a grafikai kártyákat? Milyen legyen a képernyő optimális mérete, hogy olyan kellemes legyen a szemnek, mint a ma kapható EGA és VGA monitorok képernyője?

A Macintosh esetében eddig nem volt választási lehetőség. A Macintosh II volt az első, amely színes monitort használt, de a hozzá kapható monitorok többsége így is nagy felbontású, monokróm display, a kiadványszerkesztők piacát célozva meg. Jelenleg a legtöbb gyártó úgy határozott, hogy olyan képernyőfelbontást biztosít, amely egy szabványos papírméretnek felel meg.

A legszemléletesebb példákat ezekre a képernyőkre a Radius cég adja, és még a képernyők márkaneve is ezt jelzi: FPD (Full Page Display = teljes oldalas display), TPD (Two Page Display = kétoldalas display).

Az FPD felbontása vizszintesen 640, függőlegesen 864 képpont, amely a Macintosh 72 képpont/inches felbontásának felel meg, és egy teljes 8,5x11 inches szabványos oldal fér rá a képernyőre. A Radius cég azonban nem feledkezett meg sem az európaiakról, sem a valós követelményekről. A 640 x 864-es felbontás y ablaként jelenik meg a 640 x 1024-es logikai képernyőn, így hosszabb „papír” használható, noha egy kis mozgató igényel minden lap tetején. Hasonló módon a TPD felbontása 1152 x 864, ami két egymás melletti teljes oldal megjelenítését teszi lehetővé.

A legérdekesebb, hogy az FPD és TPD képernyők felhasználják a Macintosh Plus és a Macintosh SE speciális grafikai tulajdonságait úgy, hogy a Macintoshhoz kapcsolva, annak eredeti képernyője is használható legyen. A Pagemaker futtatásakor ez az eredeti képernyő használható fel a később a nagy felbontású képernyőn beszerkesztendő szövegek és grafika tárolására. A Macintosh II-nél ez még hatékonyabb lehet, mivel a gépbe hat képernyőmeghajtó dughat be, s ezáltal a logikai képernyősik hat területe jeleníthető

meg, amelyek közül mindig csak egy az aktív.

### A drága az olcsó?

A PC-hez is kaphatók már nagy felbontású képernyők, bár igen borsos áron. A kiadványszerkesztők elterjedése miatt a nagy felbontású képernyők iránti igény ezzel együtt egyre nő. Lehetséges nagy felbontású monokróm monitor és kártya vásárlása 1000 font alatti áron — és ez viszonylag olcsó —, ami már a nyomtató szöveg hú megjelenítését adja a képernyőn. Ráadásul ezt a monitort az AutoCad és a Lotus 1-2-3 is képes használni.

Hasonlóan a Macintosh piacához, a nagy monokróm képernyők megjelenése részben a CRT technológia, részben a piaci követelmények következménye. Ezeknél a felbontás vizszintesen 1280, függőlegesen 800, illetve 1200 képpont lehet, amely nem csupán két A4-es lap egymás melletti megjelenítését teszi lehetővé, de képes a CGA és EGA képernyők skálázott szimulálására is.

Meglepően olcsó például az 1280 x 800-as felbontású Wyse WY-700, amelynek meghajtója van az Autocad, a Windows, a GEM és a Lotus 1-2-3 programokhoz. A Moniterm Viking 1 1280 x 960-as felbontással ugyanezt tudja, a Taxan Crystal View-nak pedig a Venturához is van meghajtó programja. A csúcs a Sigma LaservieW 1664 x 1200-as felbontással és az előbb említett meghajtó programokkal.

Ezeknél a képernyőknél a különbségeket elsősorban a vezérlőkártyák hordozzák. A Wyse a legolcsóbb, mivel részben a CGA és Hercules monokróm illesztőkben már alkalmazott Motorola 6845-ös áramkört használja fel, és rajzolási sebessége sem túl nagy. A másik véglet a Hitachi ACRTC áramkörét használó Moniterm vezérlőkártyája, ami extra teljesítő

***A monitorok és a vezérlőkártyák technológiai fejlődése lehetővé tette, hogy az IBM PC és a Macintosh számítógépekhez 19 inchnél nagyobb képernyőjű, nagy felbontású monitorokat illesszünk. A Macintosh kialakítása egyszerűvé teszi a nagy felbontású monitorok illesztését. A PC-k esetében azonban ez nem olyan egyszerű, kivéve, ha nem szabvánnyá vált alkalmazói programokat — amelyekhez megírják a meghajtókat — vagy a GEM, illetve a Windows alatt futó programokat használunk.***

***Amikor azonban az elemek — a monitor, az illesztőkártya és az alkalmazói program — egysége létrejön, a hatás lenyűgöző. Az IBM a CGA monitor kifejlesztésével egy olyan folyamatot indított el, ami zsákcuccokkal, tévedésekkel és inkompatibilitásokkal terhelt, de az elért eredmények igazolták az erőfeszítéseket.***



képességű, igaz, dupla áron. Ilyen mérvű felbontással már színes képernyők is kaphatók — exponenciálisan növekvő költségek árán. A NEC Multisync XL és az IBM 8514 típusok az első ilyen próbálkozások, ezekkel 1024 x 768-as felbontás mellett 256 szín jeleníthető meg, de általános elterjedésüknek a szinte elérhetetlenül magas ár egyelőre még gátat szab.

A monokróm képernyőket használó CAD-alkalmazásokkal ellentétben a kiadványszerkesztők színeit mindaddig nem lehet kinyomtatni, amíg a nagy felbontású színes nyomtatók meg nem jelennek a piacon.

Jelenleg a CAD a kis darabszámú, de igen drága szoftverek piacát jelenti. A kiadványszerkesztőket sokkal szélesebb körben használják, s ez a nagy felbontású monokróm monitorok árát csökkenteni fogja.

### „Hogy látva lássanak”

A nagy felbontású monokróm monitorok fejlesztése függ a képcsövekben alkalmazott anyagoktól is. Közismert, hogy a kép minden katódsugárcsőben a pászta-  
zó elektronsugarak segítségével jelenik meg úgy, hogy a képernyő belső oldalán lévő világító foszfort gerjesztik. A monokróm rendszereknél a kép színe az alkalmazott foszfortól függ, jelenleg zöld, fehér és sárga foszforokat alkalmaznak. A sötét alapon világos megjelenítésnek azonban számos ergonomiai hátránya van.

Először: ha sokáig nézünk egy sötét képernyőt, a szemünk pupillája kitágul — hasonlóan ahhoz, mikor huzamosabban tartózkodunk egy sötét szobában —, s ez igen megnehezíti az összpontosítást a képernyő kis darabjára.

Másodszor: a sötét képernyő sokkal jobban tükröződik, még akkor is, ha nincs fényforrás a háttérben.

Harmadszor: a másoláskor fellépő váltás a sötét alapon világos szövegű képernyőről a világos alapon sötét szövegű papírra rendkívül fárasztó a szem számára — a szemnek gyorsan kell alkalmazkodnia a világos és sötét tónusok váltakozásához.

Az igazi papírféher foszforok, amelyek lehetővé tennék a képernyők világos alapon sötét használatát, még kevésbé használhatóak és igen drágák. A piac diktálta szükségletek miatt azonban ezek az igények egyre jobban előtérbe kerülnek.

A fehér alapon fekete képernyőhöz speciális megoldások kellenek. Az éles képek megjelenítése gyors elhalványulást biztosító foszforanyagokat igényel, mivel a lassan elhalványuló képernyőn az utánvilágításból adódó szellemkép alakul ki. Az ilyen gyors foszforos képernyők a villódzás kiküszöbölése miatt gyors képrögzítést igényelnek; a képrögzítés sebességének meg kell haladnia a 70 Hz-et.

(A Desktop Publishing World c. folyóirat cikke nyomán)



## PROGRAMISMERTETŐ



# NC—Norton Commander

*Bár az IBM PC DOS operációs rendszere, a DOS parancskészlete széles körű lehetőséget nyújt a rendszerhasználójának, szinte nélkülözhetetlenek az olyan programok, amelyek segítségével kényelmesen kezelhetjük az operációs rendszert és a rendszerben levő állományokat. Az ilyen programokat szokták az angolból átvett kifejezéssel DOS-héjnak (shell) is nevezni, mert mintegy a rendszert „átfogva, keretezve” biztosítják annak kiterjesztett működését. A jó DOS-héj nemcsak gyors és könnyen alkalmazkodik a felhasználóhoz, hanem akkor sem akadályozza őt, ha munkája során közvetlenül magát a DOS-t akarja használni.*

A jól kialakított, teljes körű szolgáltatást nyújtó DOS-héj nem tiltja meg a DOS parancsaihoz való hozzáférést sem. Ha a DOS szintjén akarunk parancsot adni a gépnek, akkor azt egyszerűen a szokott módon begépelhetjük.

Az IBM PC-t használók körében három ilyen DOS-héj terjedt el: a Pathminder, a PcTools és a Norton Commander. Ezúttal az utóbbi rövid bemutatására vállalkozunk, de fontosságuk miatt a másik kettő ismertetését is célul tűztük ki. Meg kell említeni, hogy az új DOS-verzió, a DOS 4.0 is a DOS-héjak legjobb ötleteit vette át, ablakos-egeres, felhasználóbarát ember—gép kapcsolatot biztosítva.

A Peter Norton Computing Inc. cég által forgalmazott Norton Commander 2.0 (a továbbiakban NC) program a cégtől megszokott minőségben készült, és ötletgazdag megoldásokkal rendelkezik. Messzemenően figyelembe veszi a felhasználók igényeit. Az NC kétféleképpen futtatható:

tárban maradó rezidens programként vagy ismételten betöltődőként. Az első esetben az elfoglalt tárméret 138 kb-ot, a másodikban mindössze 13 kb-ot. Ilyenkor a programnak csak egy rezidens része (magja) marad a tárban, és a parancsok aktivizálásakor lép be a lemezen kint levő megfelelő programkészlet. Az NC vagy billentyűzettel, vagy egerrel, esetleg mindkettővel képes működni, s akkor is igen gyors, ha az ismételten betöltődő programváltozatot használjuk. Merevlemezről a betöltés mindössze egy-két másodpercig tart.

EGA és VGA adapterkártyákkal ellátott rendszereknél választhatunk 43, illetve 50 soros üzemmódot is. Az NC monokróm vagy színes megjelenítésű, bár az utóbbi esetben szint nem lehet választani. Táskaszámítógépekhez külön üzemmódja van.

A képernyőn két, egymástól független tartalomjegyzék jeleníthető meg. Közülük billentyűnyomásra bármelyik eltüntethető, illetve megjele-



Name	Size	Date	Time	Right
..	DIR	88-12-20	8.31	EXE
bcd	pas	4847	88-08-29	5.00
graph3	doc	1987	88-08-29	5.00
graph3	tpu	7280	88-08-29	5.00
turbo3	doc	895	88-08-29	5.00
turbo3	tpu	1744	88-08-29	5.00
upgrade	dta	25290	88-08-29	5.00
upgrade	exe	43104	88-08-29	5.00

Right
EXE
PCB
CMOS
DISCRET
INTEL
TTL
DIR
TP
DOC
TURBO3
X80
PAS
ARC
NORTON
TMP
ARCIO
DRK
TESZT
TEXT

C:\REDAC

C:\TP\TURBO3>

1)P) 2)CP) 3)Jew 4)dit 5)Op) 6)en)ov 7)dir 8)de)te 9)enu 10)uit

níthető. Mód nyílik az állományok mindenféle, akár ablakok közötti átrendezésére is. A mozgás a tartalomjegyzékekben a CTRL+PgUp (tartalomjegyzék bezárása és egy szinttel feljebb lépés), illetve a CTRL+PgDn (tartalomjegyzékben levő fájl megjelenítése) billentyűkkel történik. A CTRL+PgDn billentyűkombináció helyett az ENTER-t is használhatjuk. Lehetséges, hogy az egyik vagy mindkét ablakban a könyvtár fa szerkezetét látva dolgozzunk. Egy másik szolgáltatás segítségével egy-egy soros összefoglalót jeleníthetünk meg, amely megmutatja, hány külön állományt választottunk ki, és mennyi ezek összmérete. A tartalomjegyzékekben szereplő fájlokat név, kiterjesztés, létrehozási időpont vagy méret szerint sorba rendezve jeleníthetjük meg. A fájlra vonatkozólag megjeleníthető csak a név és kiterjesztés (brief) vagy ezeken kívül a generálási dátum és méret (full) is. A program a tartalomjegyzékben levő összes fájl kilistázza, a rejtett attribútumúakat is. Ezeket a fájlnev-kiterjesztés előtti külön grafikus karakter jelzi.

A kapcsolat a felhasználóval a „mutass rá, és tüzelj” elven valósul meg: a kurzorral rámutatunk a megjelenített tartalomjegyzék egyik elemére vagy csoportjára, ez megkülönböztethető módon átszíneződik, majd végrehajthatjuk rajta az összes megszokott DOS-funkciót, például törlést, másolást vagy átnevezést. A csoportkijelölés egérrel, a + vagy az INS billentyűvel lehetséges.

A szokásos DOS-parancsok mellett az NC-ben a MOVE utasítás biztosítja az állományok átmozgatását egyik tartalomjegyzékből a másikba. A kije-

lölést vagy a billentyűzet, vagy az egér segítségével végezhetjük el.

A képernyőn a jobb felső sarokban az idő folyamatosan megjeleníthető.

Az NC emlékszik az általa utoljára végrehajtott 15 DOS-parancsra, így újrainrás nélkül, menüből adhatjuk ki őket ismét. Az ilyen régi parancsok szerkeszthetősége is biztosítva van. A CTRL+ENTER billentyűk együttes megnyomásakor az ablakban kiválasztott fájl nevét és kiterjesztését a DOS-parancssorba másolja.

Speciális utasítással összehasonlíthatjuk két könyvtár állományait. Ez megjelöli mindazokat az állományokat, amelyek a másik könyvtárból hiányoznak, vagy azokat, amelyeket legutoljára hoztunk létre. Így naprakész állapotban tarthatjuk a könyvtárakat.

Fájlokat a lemezen nagyon egyszerű megkeresni: egy utasítással (Find File) megtalálhatjuk az összes olyan fájlt, amely megfelel egy általunk megadott maszknak. Például a W\*.COM maszk alkalmazásakor az összes W-vel kezdődő és .COM kiter-

jesztésű fájl kereshetjük meg. Sikeres keresés után lehetőség van arra, hogy a megtalált fájlt tartalmazó tartalomjegyzék legyen az aktuális tartalomjegyzék.

A beépített, WordStarhoz hasonló szövegszerkesztő maximum 30 kb-átos állományokat képes kezelni, ami legtöbbször elegendő. A szerkesztett állományok tetszőleges, akár bináris fájlok is lehetnek, amelyekbe beleírhatunk, törölhetünk. Használatukra éppen ezért nagyon kell ügyelni! Ha valaki a megszokott szövegszerkesztőjét akarja használni, kijelölheti a sajátját. Ekkor a program egyszerűen ezt a saját szerkesztőt indítja, és automatikusan betölti hozzá a kiválasztott állományt is. A főmenü VIEW funkciójával nagyobb méretű, tetszőleges állomány jeleníthető meg, de a megjelenített állomány nem szerkeszthető.

Létrehozhatunk állományt az egyes alkalmazói programjainkhoz tartozó állománynév-kiterjesztéseknek is. Így, amikor kiválasztunk egy adatállományt a könyvtárból, az NC automatikusan a hozzá tartozó alkalmazási programmal együtt tölti be azt.

A felhasználói menü készítése nagyon egyszerű. A főmenün kívül minden könyvtárnak lehet saját menüje, az egyes menüket bonyolult rendszereket alkotva össze lehet fűzni.

A programhoz két könyvet adnak: az egyik a felhasználói kézikönyv, a másik egy rövid képes termékleírás. Ezekben a dokumentációkon kívül munka közben az F1 funkcióbillentyű megnyomásával is kaphatunk segítséget. Ilyenkor az a képernyő jelenik meg, amelyen a CTRL billentyűvel kombinált parancsokról láthatunk összefoglalót. Egy további segédképernyő tartozik a szerkesztő részhez. Az egér használatát és a menüfunkciók magyarázatát ilyen online help nem támogatja.

K. L.

*Összefoglalva: az NC nagyon jól használható program. Nagyon sok szolgáltatása van, alkalmazása egyszerű, könnyű megtanulni. Ha egeret használunk, szinte nem is kell a billentyűzethez nyúlnunk: minden funkció megvalósítható a kurzorral történő rámutatással és az egér két nyomógombjának segítségével. Igen ötletes, hogy másolás közben a már átmásolt állomány nagyságával arányos hosszúságú csík látszik, vizuálisan mutatva, hogy éppen melyik stádiumban tart a másolás. A program eleget tesz mindannak, amit a felhasználó elvár mind a menüstruktúra, mind a felhasználóval való kapcsolattartás tekintetében.*



## SZABAD SZOFTVER



# Amikor a programokat el szabad „lopni”!!!

**Nincs túl jó hírük a számítógép-alkalmazóknak és programozóknak hazánkban. Már jóformán egyik hazai profi szoftveres vállalkozásunk sem akarja termékeit védelem nélkül a piacra dobni. Fél attól, hogy egyetlen példányt tud csak eladni; így még a fejlesztés költsége sem térül meg, hogy az elmaradt haszonról ne is beszéljünk. S a helyzet egyre súlyosbodik.**

A hazai felhasználó irtózik a hazai — sokszor bizony komoly károkat okozó — szoftvervédelmeiktől, vírusoktól. Ugyanakkor a másolás megakadályozására a programozók is egyre durvább módszereket találnak ki. Természetesen ők sem ártatlanok. Nem egy termékről derült ki, hogy a védelem csak a „stiklit” leplezte, az „alkotó-forgalmazó” más tollával ékeskedett.

Vajon milyenek a másolás legális lehetőségei? Vajon tényleg ilyen rossz a helyzet, hogy nem egy szoftverlaboratóriumban ki kellett írni a gépek fölé: „Hazai eredetű szoftvert a gépbe tölteni, másolni, használni szigorúan tilos!”

### Szoftver ajándékba

Röviddel a számítógépek elterjedése után megjelentek az úgynevezett szabad szoftverek is. Ezek részben nagy cégek ingyenes szoftverajándékai, részben forgalmazási jelentőséggel nem bíró apró segédprogramok, programozási segédeszközök.

Elterjedésükhöz a cégeknek is érdekük fűződött, mert ha bevált egy szabad program, akkor a felhasználó nyilván nagyobb bizalommal vásárolta meg a drága, nagy programrendszereket is. A szabad szoftverek alkotóinak másik fele olyan amatőrökből került ki, akik hobbijukkal nem akartak keresni. Az általuk fejlesztett programokat önzetlenül bocsátották a köz rendelkezésére. (Nálunk ez a fajta mentalitás sajnos nem igazán honosodott meg; jól példázza ezt a Mikroszámítógép Magazin által elindított, majd kényszerűségből — felajánlott ingyenprogramok híján — berekesztett Közprogramok sorozat sanyarú sorsa is.) Emellett még egy

kategóriát találhatunk: az olyan szabad szoftvereket, ahol csak a magáncélú felhasználás jogát engedik át a programozók, az államigazgatási vagy egyéb professzionális célú felhasználásért minimális térítést azért kell fizetni. Végül van még egy kategória. Itt az alkotók arra szólítják fel a használókat, hogy ha elégedettek a programjukkal, akkor ezt valamilyen névleges díjjal honorálják. Ennek fejében aztán felveszik a pénzt küldőt az úgynevezett alkalmazói listára, s legtöbbször lemezárban vagy másoknál előbb kapja meg az újabb verziót. Természetesen a nagy cégek nagy rendszerei azok, amelyekből sohasem lesznek — sajnos — szabad szoftverek.

Ha megvizsgáljuk a hazai forgalomban található, a közhit szerint illegális másolatként keringő szoftvereket, nagyon meglepő dolgokat tapasztalhatunk. A rendszeresen használt és jogtalanak tűnő szoftverek nagy része a szabad szoftverek közé tartozik!

A legtöbb ilyen szabad szoftver angol, amerikai vagy német közvetítéssel került hazánkba. Az önköltségen árusító cégek (például a Public Domain Software Inc.) hirdeteiből, katalógusaiból egyértelműen kideríthető az ide tartozó szoftverek köre. Ez rendszeresen bővül, változik. Mert sokszor előfordul, hogy amikor kijött valamilyen új programverzió, az előző változatot alkotói szabadon terjeszthetővé teszik. Így sokszor egy-egy szoftvernek csak bizonyos verziószámú változatai esnek ebbe a kategóriába. Ilyenek például egyes Norton segédprogramok régi változatai is, vagy a széles körben alkalmazott TERMULTR, azaz a termulátor kommunikációs szoftver.

Hivatásos alkalmazásokért pénzt kér a SeaWare Inc. az ARC nevű tömörítő szoftveréért. Ezzel szemben a PkWare Inc. PKX35A35, az előbbivel teljesen kompatibilis programrendszere szabad szoftver, de névleges díjért az alkalmazói listára bárki felvétetheti magát.

### Mint egy telefonkönyv

A szabad szoftverek egyik legteljesebb katalógusa (ami nem tartalmazza a német nyelvterület és a frankofon országok készletének nagy részét) az USA-ban jelenik meg. Évente publikálják az újabb kiadásokat, amelyekből mindig kimaradnak a már senki által nem használt „öskori kövületek”, hogy helyüket átvegye az újabb termés. Érdekes még olvasmányának is ez a vastag, telefonkönyvszerű kiadvány. Ezt végignézve látható csak át a számítástechnikai kultúra külhoni szintje: szinte minden feladathoz található megfelelő szintű szabad szoftvert is. Aki teheti, feltétlenül szerezze be ezt a hasznos kiadványt: The IBM PC (and compatibles) free software catalog and directory. The what, where, why and how of selecting, locating, acquiring and using free software. Red.: Robert A. Froelich. Published by Dilithium Press Ltd. ISBN 0-517-56112-3 921 S. W. Washington. Ára: 17,95 USD. Portland, Oregon 97205 USA. Distributed by: Crown Publishers Inc. 225 Park Avenue South, New York, 10003.

A második kötet a CP/M operációs-rendszer-családba tartozó gépek szabad szoftvereihez ad angol—amerikai piacra kiterjedő tájékoztatót. Címe: The free software catalog and directory for all computers capable of running CP/M operating system including Apple, Commodore, Digital Equipment, Atari and the other 250 computers using CP/M. Red.: Robert A. Froelich. Published by Dilithium Press Ltd. 921 S. W. Washington. Ára: 9,95 USD. Portland, Oregon 97205 USA. Distributed by: Crown Publishers Inc. 225 Park Avenue South, New York, 10003.

Bár egyik kiadvány sem másolható szabadon, mégis megéri beszerezni. A szerző teljes körű áttekintést igyekszik adni a szabad szoftverek választékáról és be-



szerzési lehetőségeiről. Most, hogy a Magyar Posta kevésbé akadályozza adminisztratív eszközökkel a telefonon történő adatátvitelt, igen figyelemreméltó, hogy ingyenes adatbankok telefonszámjai is megtalálhatók a könyvben. Így lehetővé válik, hogy magunk hívassunk le ingyenes szoftvert. Figyelembe kell azonban venni a magas telefonszámlát, és azt, hogy az USA-rendszerek Bell-szabványú modemeket használnak. Európában a CCITT az uralkodó norma. A könyv emellett határozóként is felhasználható, ha egy ismeretlen szoftver kerül a kezünkbe, vagy éppen el szeretnénk dönteni valamelyik programról, hogy szabad-e vagy sem. Közli minden programról — amelyek szabad szoftverként szerepelnek — az egyes fájlok nevét, hosszát, dátumát, CRC-jét, forgalmazóját, készítőjét, valamint azt, milyen célt szolgál a program.

### **Ezt is szabad . . .**

Nemcsak a szabad szoftverként forgalmazott szoftvereket kell azonban szabadon felhasználható szoftvereknek tekinteni. Ezekkel egyenlő elbírálás alá esnek a *könyvekben megjelent forráskódú programok* (de a programlemezek, ha nem adják mellé a forráskódot és/vagy másolás ellen védik, akkor már nem szabadok). Így például szabad szoftver Peter Norton: *Inside in the PC* könyvének lemez melléklete, de az LSI-ATSZ kiadásában megjelent PC-DOS kézikönyv oktatólemeze már nem! Ez utóbbit ráadásul az Éltető-féle ELTGUARD rendszerrel védik a másolás ellen.

Hasonló a helyzet a *folyóiratokban megjelent forrásnyelvi programokkal*, és

szabad szoftvernek tekintendők a *programok, programnyelvek, nyelvi rendszerek forrásnyelvi példaprogramjai* is (a könyvtári rutinok programváltozatai nem!).

*Egyes programrendszerek futtató moduljai* akkor szabad szoftverek, ha a szoftverforgalmazó nem rendelkezik másképp. Így például szabad a Flash Up vagy az IBM Story Teller futtató rendszere, de nem szabad szoftver a BASIC-futtató (Basrun), a Fox futtató modulja, hogy a legismertebbeket említsük.

A szabad szoftverek sem használhatók fel teljesen szabadon, mondhatni gátlástalanul. A szerzői jogok személyhez fűződő jogok. Így a szoftver a szerző tulajdonát képezi. Hiába alkalmazható szabadon, a felhasználó pénzt nem kérhet érte, csak egyéb, befektetett munkáját fizettetheti meg. Például az UNESCO ingyen bocsátja a felhasználók rendelkezésére a PC ISIS adatbázis-kezelőjét. A felhasználást kérőnek azonban írásban is köteleznie kell magát, hogy a programrendszerért nem kér anyagi ellenszolgáltatást, csak a vele elkészült adatbázist fizettetheti meg. Így például a PKX35A35-ért sem lehet külön pénzt kérni, de ha egy jogtisztá felhasználói programot tömörítünk vele, a saját munkánkat már megfizettethetjük.

### **DE!**

A szabad szoftverek is bizonyos szabályok figyelembevételével használhatók, mint az előbbiekből már kitudnik: csak arra a célra használhatjuk őket, amire engedélyezték (a szabad szoftverek ilyen megkötéseit a program bejelentkezéskor közli); nem írhatjuk át például sajátunkra a copyrightot; csak a teljes rendszert (dokumentációval stb.) adhatjuk tovább ellenszolgáltatás nélkül (a lemez árát és a méltányos munkadíjat a szabadprogram-forgalmazók is felszámolják, ők ebből élnek); a rendszert csak akkor írjuk át saját nyelvünkre, ha ezt kifejezetten felkínálja (például forrásnyelven van, vagy a szövegek külön fájlban vagy egyéb elérhető helyen vannak. A hozzáértés nélküli vagy a forráskód ismerete nélküli változtatás kárt okozhat a gépben lévő egyéb programokban is); bármit irtunk is át, változtatunk az eredetihez képest valamilyen szoftveren, azt a lemezen is dokumentáljuk, s ezt a verziót így adjuk tovább (erre szolgálnak például az egyezményes Read.me, Olvass.el stb. nevű szövegfájlok).

(A Floppy lapban megjelent cikk nyomán)

***Ha kiderül, hogy sajátunkként adtunk akár szabad szoftvert, akár egy kevésbé ismert profi szoftverház termékét, megsértettük a szerző jogait, így az jogosan perelhet. S most, hogy nálunk is komolyan kell végre venni ez irányú kötelezettségeinket, szinte biztosan megnyeri a pert, mint azt több jogbitorlási per mutatta a közelmúltban.***



# Az MS-DOS 3.30 parancsai funkciók szerint

Az IBM PC számítógépek operációs rendszeréről — a Microsoft cég által kifejlesztett MS-DOS-ról — és parancsairól már magyar nyelven is igen bőséges irodalom áll rendelkezésre. A következőkben bemutatjuk a parancsok egy olyan funkcionális felosztását, amely mélyebb áttekintést ad az MS-DOS belső felépítéséről, mint az általában alfabetikus sorrendben ismertetett parancsok. Mivel a DOS-t állandóan továbbfejlesztik, ezért a jelen cikkben

az aktuális, Magyarországon is elterjedt 3.30-as változat parancsait ismertetjük.

Az MS-DOS parancsok az információt tároló bájtsorozatokat (a fájlok) kezelését, az adott rendszernek megfelelő konfiguráció és a perifériák beállítását, valamint a rendszer számára adott parancsok feldolgozásának megkönnyítését szolgálják. Az általuk végrehajtott tevékenységek szerint több csoportra oszthatók:





### Rendszerkonfiguráló és -vezérlő parancsok

BREAK	CTRL-BREAK billentyűk figyelésének bekapcsolása
COMMAND	Másodlagos parancsfeldolgozó program indítása
DATE	Dátum beállítása
EXIT	Visszatérés az elsődleges parancsfeldolgozóhoz
PROMPT	Rendszerbejelentkezési formátum beállítása
SELECT	Rendszerlemez készítése adott nemzeti kódhoz
SET	Programkörnyezeti változó beállítása
SHARE	Fájlmegosztási környezet kezelése
TIME	Idő beállítása
VER	DOS-verzió megjelenítése

### Karakter típusú eszközök (egy karakter ki/be) kezelése

CLS	Képernyő törlése
CTTY	Aktuális B/K terminál kezelése
GRAFTABL	Grafikus karakterkészlet betöltése
GRAPHICS	Képernyő grafikus nyomtatása
KEYBxx	Nemzeti billentyűzet definiálása
MODE	Perifériák működési módjának beállítása
PRINT	Nyomtatás háttérből (spooling)

### Fájlok kezelése

ATTRIB	Fájljellemzők beállítása
BACKUP	Fájlok elmentése (archiválása) háttértárolóra
COMP	Fájlok összehasonlítása
COPY	Fájlok másolása
DEL/ERASE	Fájlok törlése
FC	Fájlok összehasonlítása
RECOVER	Sérült fájlok helyreállítása
RENAME	Fájlnev átnevezése
REPLACE	Fájlok aktualizálása
RESTORE	Elmentett fájlok visszaállítása
TYPE	Fájl megjelenítése
XCOPY	Fájlok csoportos átmásolása

### Szűrők

FIND	Karakterfüzér keresése
MORE	Egy képernyőnyi tartalom kijelzése
SORT	Fájl vagy karaktersorozat alfabetikus sorba rendezése

### Tartalomjegyzékek kezelése

APPEND	Adatfájl keresési útvonalának beállítása
CHDIR	Aktuális tartalomjegyzék váltása
DIR	Tartalomjegyzék megjelenítése
MKDIR	Altartalomjegyzék létrehozása
PATH	Keresési útvonal definiálása
RMDIR	Altartalomjegyzék megszüntetése
TREE	Tartalomjegyzék struktúrájának megjelenítése

### Lemezkezelés

ASSIGN	Meghajtó-hozzárendelések megváltoztatása
CHKDSK	Lemez vizsgálata
DISKCOMP	Lemezek összehasonlítása
DISKCOPY	Teljes lemez másolása
FORMAT	Lemez formázása
FDISK	Fix lemezegység konfigurálása
JOIN	Lemez hozzárendelése tartalomjegyzékhez
LABEL	Kötetazonosító megadása
SUBST	Tartalomjegyzék helyettesítése lemezegység betűjével
SYS	Rendszerfájlok másolása
VERIFY	Írás utáni ellenőrzés
VOL	Kötetazonosító megjelenítése

### Installálható eszközmeghajtók

ANSI.SYS	Kibővített képernyő- és billentyűzetkezelő
DRIVER.SYS	Külső lemezmeghajtók konfigurálása
RAMDRIVE.SYS	Memórialemez (virtuális diszk) használata
VDISK.SYS	Memórialemez (virtuális diszk) használata

### Rendszerkonfigurálás

BREAK	CTRL-BREAK billentyűk figyelésének beállítása
BUFFERS	Belső lemezpufferek számának beállítása
COUNTRY	Országra vonatkozó nemzeti kód megadása
DEVICE	Perifériakezelők installálása
DRIVPARM	Blokk típusú eszközök paramétereinek beállítása
FCBS	Egyszerre megnyitható fájlok száma (CP/M-szerű fájlkezelés)
FILES	Egyszerre megnyitható fájlok száma (DOS-fájlkezelés)
LASTDRIVE	A logikailag használható meghajtók max. száma
SHELL	Felhasználói parancsfeldolgozó beállítása
STACKS	DOS veremhasználat változtatása

### Köteget (batch) fájlok

AUTOEXEC.BAT	Rendszer indításakor aktív parancsfájl
ECHO	Üzenetek megjelenítése
FOR	DOS-parancsok ismételt végrehajtása
GOTO	Ugrás címkére a parancsfájlban
IF	Feltételtől függő végrehajtás
PAUSE	Parancsfájl végrehajtásának felfüggesztése
REM	Megjegyzéssor
SHIFT	Változtatható paraméterek léptetése



# Kiterjesztések -tól .WRI-ig

Az MS-DOS által használt fájlok maximálisan 8 karakterből álló azonosítójuk és három karakterből álló fájlnevük kiterjesztésük van. Ez utóbbi megadása nem általánosan kötelező, de a DOS három kiterjesztést megkülönböztetve kezel: az .EXE, a .COM és a .BAT kiterjesztésű fájlok futtathatók a DOS alatt. Más kiterjesztésre nincs kikötés, de a szervezett, egységes munka érdekében a DOS rendszerben bizonyos kiterjesztések egységes jelentéssel bírnak. A névkiterjesztéseknek határozott rendeltetésük van: a fájlok kategóriájának és osztályozásának jelzésére szolgálnak. A kiterjesztéseket a Microsoft vezette be, és jelenleg szinte szabványosnak tekinthetők. Minél következetesebben használjuk ezeket a kiterjesztésnév-mintákat, annál könnyebbé teszik állományaink (fájljaink) használatát. A következőkben ezeket a legfontosabb kiterjesztéseket táblázatos formában ismertetjük.

KITERJESZTÉS	PROGRAM	LEÍRÁS
@@	MS-DOS	Backup azonosító fájl
\$\$\$	EDLIN	Hiba-fájl
.ASC	ÁLTALÁNOS	ASCII-szövegfájl
.ASM	MASM	Assembly-forrásfájl
.BAK	ÁLTALÁNOS	Eredeti elmentett fájl (backup)
.BAS	BASIC	BASIC forrásfájl
.BAT	MS-DOS	Parancsfájl
.BIN	ÁLTALÁNOS	Bináris fájl
.C	C nyelv	C nyelvű forrásfájl
.CAL	WINDOWS	Naptárfájl
.COB	COBOL	COBOL forrásfájl
.COD	ÁLTALÁNOS	Object listafájl
.COM	MS-DOS	Futtatható fájl
.CRD	WINDOWS	Kártyafájl
.CRF	MASM	Kereszthivatkozási fájl
.DAT	ÁLTALÁNOS	Adatfájl
.DBG	COBOL	Hibafájl
.DEF	WINDOWS	Moduldefiníciós fájl
.DOC	ÁLTALÁNOS	Dokumentumfájl
.DRV	ÁLTALÁNOS	Eszközmeghajtó fájl
.ERR	ÁLTALÁNOS	Hibafájl
.EXE	MS-DOS	Futtatható fájl
.FNT	ÁLTALÁNOS	Betűtípus-(font-) fájl
.FON	ÁLTALÁNOS	Betűtípus-(font-) fájl
.FOR	FORTRAN	FORTRAN forrásfájl
.GRB	WINDOWS	Képernyőkép-fájl
.H	C	Include fájl
.HEX	MS-DOS	INTEL HEX formátumú fájl
.HLP	ÁLTALÁNOS	Segítség tartalmazó fájl
.INC	ÁLTALÁNOS	Include fájl
.INI	WINDOWS	Inicializációs fájl
.INT	COBOL	Object-fájl
.LIB	ÁLTALÁNOS	Könyvtárfájl
.LST	ÁLTALÁNOS	Listafájl
.MAP	ÁLTALÁNOS	Címtérkép-fájl
.MOD	ÁLTALÁNOS	Modulfájl
.MSG	COBOL	Üzenetfájl
.MSP	WINDOWS	Fájl Windows-rajzhoz
.OBJ	ÁLTALÁNOS	Áthelyezhető tárgymodul
.OVL	ÁLTALÁNOS	Overlay-fájl
.OVR	COBOL	Fordító overlay-fájlja
.PAS	PASCAL	PASCAL forrásfájl
.PIF	WINDOWS	Programinformációs fájl
.QLB	ÁLTALÁNOS	MS Quick-termékek könyvtárfájlja
.RC	WINDOWS	Forrásszöveg-fájl
.REF	CREF	Kereszthivatkozási listafájl
.RES	WINDOWS	Lefordított fájl
.SCR	ÁLTALÁNOS	Leíró fájl
.SYM	ÁLTALÁNOS	Szimbólumfájl
.SYS	ÁLTALÁNOS	Rendszerfájl vagy eszközmeghajtó
.TMP	ÁLTALÁNOS	Segéd fájl
.TRM	WINDOWS	Terminál fájl
.TXT	ÁLTALÁNOS	Szövegfájl
.WRI	WINDOWS	Write-fájl

## ? AJÁNDEK

# FILL — merevlemezről floppyra

Amikor a merevlemezről olyan állományokat akarunk hajlékonylemezre másolni, amelyek összmérete nagyobb, mint a hajlékonylemez kapacitása, bajban vagyunk, hogy milyen sorrendben másoljuk át a fájlokat, hogy egy lemezre maximális mennyiségű adat kerüljön. A fenti, a DOS COPY parancsához hasonló program ezt a feladatot végzi el: úgy csoportosítja a fájlokat, hogy egy lemezre maximális mennyiség kerüljön.

A program hívása:

FILL [-q] source-path [source-path] [source-path . . .] destination-path  
ahol

-q: a paraméter megadása esetén a program nem ad üzenetet akkor, ha a céllemezen egy már létező nevű állományt írunk felül. Megadása nélkül az üzenet megjelenik.

source-path: a merevlemezről átmásolandó állományok forrásalkönyvtárainak elérési útvonala.

destination-path: a hajlékonylemez célalkönyvtárának útvonala.

Korlátozás: egy megadott forrásalkönyvtárban nem lehet 1024 fájlnál több. Ez a korlátozás módosítható, ha a program C nyelvű forráslistáját módosítjuk és újra fordítjuk.

**Megjegyzés.** A program három fájlból áll: a C forrásnyelvi listából, a lefordított programból és az eredeti angol nyelvű dokumentációból.

(Dave Rand, USA 72 Longfellow St., Thousand Oaks, CA 91360 12/09/86)

## FIGYELEM!

A PÉCÉZZÜNK rovatban megjelent cikkek szövege szövegfájlok formájában, valamint az „Ajándék” szabad szoftver 360 kb-átos DS-DD lemezen, utánvétellel, önköltségi (lemezár, lemezmasolás, postázás) 300 forintos áron megrendelhető. Cím: Koncz Edit, Budapest, Kunigunda u. 44. 1037

## AmigaBasic

# A Microsoft BASIC-interpretere Amigára

Az AmigaBasicet teljes egészében a Motorola 68000-es mikroprocesszor — az Amigák központi egysége — assembly nyelvén írták. Ennek köszönhető, hogy a betöltődés után a memóriának csak viszonylag kis részét (kb. 80 kb-át) foglalja el. A Microsoft állítása szerint az interpreter magját három éven keresztül tesztelte a gyakorlatban. Véleményem szerint az AmigaBasic — legalábbis az 1.2-es verzió — még mindig nem mentes a fatális programhibáktól. Aki komolyabb feladat megoldásába fog, jól teszi, ha minden jelentősebb változtatás után elmenti munkáját. Engem meglepetések mindig csak programszerkesztés közben értek, így azt hiszem, hihetünk a Microsoft tesztsjeinek.

### SAY: Amiga

Az AmigaBASIC előnyei közé tartozik, hogy felülről kompatibilis a jelenleg IBM PC-ken elterjedt BASIC-ekkel (GW-BASIC, Turbo BASIC). Támogat majdnem minden olyan funkciót, amelyben az Amiga többet tud nyújtani más számítógépeknél. Kihasználhatjuk az egér-, a menü- és az ablaktechnika nyújtotta lehetőségek zömét. Definiálhatunk képernyőt akár 640 x 512-es felbontás mellett is. A maximálisan 32 darab palettaszínt 4096 közül válogathatjuk össze. A négy darab sztereó hangcsatornán az általunk szabadon definiált hullámformájú hangokat hozhatjuk létre, és élhetünk a szinkronizáció lehetőségével is. A kiviteli utasítások között megtalálhatjuk a SAY-t, amelynek révén tetszőleges szövegeket képes kimondani a számítógép. A SAY az egyik legfontosabb előrelépés, s egy új számítógép-generáció tagjává avatja az Amigát.

Az AmigaBasicnek három fő üzemmódot van: a program üzem-mód a program futása közben él; parancs üzemmód esetén a program nevét viselő ablakba vihetünk be parancsokat, s ezeket az interpreter a RETURN billentyű lenyomása után azonnal végrehajtja. Végül a szerkesztési üzemmódban a LIST nevű ablakot használhatjuk a program írására, módosítására.

Az első négy menüszoport az AmigaBasic számára van fenntartva. A menüszoportok nevét a menüpontok nevétől / jellel választottam el.

Project/New:	törli a memóriában levő programot és az összes változót
Project/Open:	új program betöltését kezdeményezi
Project/Save:	a memóriában levő program elmentése az eddig használt néven
Project/Save As:	a program elmentése egy új néven
Project/Quit:	kilépés az AmigaBasicből
Edit/Cut:	a megjelölt programrészletet kivágja a programból és áthelyezi az átmeneti tárbá
Edit/Copy:	a megjelölt programrészletet átmásolja az átmeneti tárbá
Edit/Paste:	az átmeneti tár tartalmát a kurzor helyénél beszúrja a programba
Run/Start:	elindítja a memóriában levő programot
Run/Stop:	leállítja a programfutást
Run/Continue:	folytatja a STOP-pal megállított program végrehajtását
Run/Suspend:	Felfüggeszti a programfutást addig, amíg egy tetszőleges billentyűt le nem nyomunk
Run/Trace On/Off:	be-, illetve kikapcsolja a lépésenkénti program-végrehajtási üzemmódot
Windows/Show List:	előtérbe helyezi és aktivizálja a LIST nevű ablakot
Windows/Show Output:	előtérbe helyezi és aktivizálja a program nevét viselő parancsablakot

### Sorszámok nélkül

Az AmigaBasic programlistáinak legszembetűnőbb sajátossága, hogy nincsenek sorszámok. Mégis, hogyan hivatkozhatunk a szerkeszteni kívánt programrészekre, a GOTO-val, GOSUB-bal megcélzott sorok-

ra? Szerkesztési módban ez az egér segítségével elmozgatott kurzorral történhet, program módban pedig címkékkel jelölhetjük meg az elérni szándékozott sorokat. A címke egy az utasításor előtt elhelyezett tetszőleges karaktersorozat, amelyet kettősponttal fejezünk be, és ugyanúgy használhatjuk, mintha egy sorszám lenne.

Lényeges megkötés, hogy a programsorok hossza maximálisan 254 karakter lehet.

A változók különböző típusokba sorolhatók, a típust a változó neve után írt speciális karakter jelzi. A változótipusok jellemzőit foglalja össze az alábbi táblázat:

Típus	Jelző-karakter	Rövidítés	Értéktartomány
2 bájtós egész	%	INT	-32768 ... 32767
4 bájtós egész	&	LNG	-2147483648 ... 2147483647
egyszeres pontosságú lebegőpontos	!	SNG	$1.18 \times 10^{-38}$ ... $3.4 \times 10^{38}$
dupla pontosságú lebegőpontos	#	DBL	$2.23 \times 10^{-308}$ ... $1.79 \times 10^{308}$
sztring	\$	STR	0 ... 32767 karakter

Rendkívül praktikus az a könnyítés, hogy a 10-es méretig semmilyen tömbváltozót nem kell dimenzionálnunk a DIM utasítással, ugyanis ezt a rendszer automatikusan elvégzi, amikor a tömbre hivatkozunk.

Az aritmetikai operátorok között a következő — bizonyára sokaknak újdonságot jelentő — lehetőségek is feltárolnak:

/	egész-osztás (a lebegőpontos osztással szembeni előnye a gyorsaság)
MOD	modulo-osztás (eredménye az osztáskor keletkező maradék)
XOR	KIZÁRÓ-VAGY művelet (bitenként)
EQV	EKVIVALENCIA művelet (bitenként)
IMP	IMPLIKÁCIÓ művelet (bitenként)

### BOB-é a képernyő

Az Amigán az animáció legjelentősebb bázisai az ún. BOB-ok és a náluk sokkal inkább háttérbe szorított sprite-ok. A sprite-ok mérete kötött: 16 x 16 képpont, a BOB-oké viszont szabadon választható; egy BOB akár a teljes képernyőt is elfoglalhatja. Mindezt csak azért említem meg ezen a helyen, mert az AmigaBasic nem tesz különbséget a BOB-ok és a sprite-ok között. Mindkettőt az OBJECT (tárgy) névvel illeti.

Jó tulajdonságuk a grafikus utasításoknak, hogy lehetővé teszik a turtle (teknősbéka) grafikai felhasználást, amely a BASIC-nél magasabb szintű nyelvek irányába mutat. Ha ugyanis egy STEP kulcsszó irunk az illető grafikus utasítás után, akkor a megadott koordinátákat az aktuális grafikus kurzorpozícióhoz előjelesen hozzáadódó relatív koordinátáként (eltolásként) értelmezi az interpreter.

A be- és kivitellel használható logikai eszközökre (készülékekre) az itt megadott rövidítésekkel hivatkozhatunk az utasításokban.

Rövidítés	Jelentés
SCRN	képernyő
KYBD	billentyűzet
LPT1	nyomtató
COM1	soros (RS232) port

### Az AmigaBasic utasításkészlete

y=ABS(x)

Az argumentum abszolút értékét szolgáltatja.

AREA [STEP] (x,y)

Az AREA utasítások sorozatával egy sokszöget definiálhatunk, amelyet azután az AREAFILL-lel tölthetünk ki. A kitöltéshez használt mintázatot a PATTERN utasítás segítségével adhatjuk meg.

AREAFILL [n]

Az AREA utasítással leírt sokszöget tölti ki a PATTERN-nel megadott mintázattal.

n=0 kitöltés a megadott mintázattal  
 n=1 kitöltés a megadott mintázat inverzével  
 y=ASC (x\$)  
 Az argumentum ASCII kódját megadó függvény  
 y=ATN(x)  
 Az argumentum arkusztangensét adja vissza.

**BEEP**  
 Az utasítás kiadása egy rövid hangjelzést eredményez a bal csatornán. Hatása megegyezik a 7-es ASCII kódú karakter kiíratásával.

**BREAK ON/OFF/STOP**  
 A BREAK billentyűkombináció (CTRL-C) figyelését, illetve az általa kiváltott programágazást engedélyezi vagy tiltja.

**ON:** a CTRL-C nem szakítja meg a programfutást, és a BREAK kísérlet az ON BREAK GOSUB utasítással programból észlelhető

**OFF:** A CTRL-C megszakítja a program végrehajtását, és ez a programból nem érzékelhető

**STOP:** ha CTRL-C-t adunk, akkor azt eltárolja a rendszer, és csak egy BREAK ON kiadása után ugrik el az ON BREAK GOSUB rutinjára

[CALL] címke [(argumentumlista)]

[CALL] memóriacím [(argumentumlista)]

Az első szintaxis szerint kiadott CALL utasítás szubrutinként hívja meg a megadott nevű alprogramot, átadva az argumentumlistában felsorolt paramétereket (lásd SUB utasítás). A kezdőcímet megadva gépi kódú szubrutin is meghívható CALL-lal. Érthetőbbé tehető a programlista, ha ilyenkor is címkéket alkalmazunk. Ezt a ROM-rutinok esetében tehetjük meg a legkönnyebben, mégpedig úgy, hogy a LIBRARY utasítás segítségével betöltjük a használni kívánt rendszerkönyvtárak címkelistáját.

(Ehhez persze elengedhetetlenül szükséges a Kickstart-ROM ismerete.) Mellékesen megjegyzem, hogy ha nincs átadandó paraméter, akkor a CALL kulcsszó el is hagyható.

y=CDBL (x)

Az argumentumot dupla pontosságú lebegőpontos („valós”) számmá alakítja át.

**CHAIN [MERGE] fájlnev [, [induló sorszám] [, [CALL] [DELETE tartomány]]]**

A CHAIN egy olyan sokoldalú utasítás, amellyel ún. overlay programfutást érhetünk el, vagyis a programok képesek betölteni és elindítani egymást. Ha a CHAIN után paraméterként csak egy fájlnev áll, akkor az új program kitér az őt hívó programot, majd elindul. Ha számozott sorokból áll a betöltendő program, akkor megmondhatjuk, hogy melyik sortól szeretnénk elindítani azt. ALL esetén a vezérlést átadó program összes változóját átveszi a betöltött program (nemcsak a COMMON-nal deklaráltakat). A DELETE kulcsszó segítségével a betöltőprogramunk egy részét törölhetjük ki a megadott sorszámok között. Ennek a műveletnek csak MERGE-dzsel együtt van értelme, mert így a betöltőprogram is a memóriában marad. Nem győzőm hangsúlyozni, hogy a MERGE csak számozott sorokat képes megkülönböztetni!

**CHDIR útvonal**

Az aktuális útvonalat beállító utasítás. Az útvonalmegadás formátuma semmiben sem tér el az AmigaDOS-nál megszokottól.

y=CINT(x)

Az argumentumot 16 bites előjeles egészé alakítja át.

**CIRCLE [STEP] (x,y), r [,szin[,kezdőszög, végszög[,arány]]]**

Kört vagy ellipszist rajzol a megadott középponttól vett r sugárral. Szinként valamelyik palettaszín szerepelhet, a rajzolás kezdeti és befejezési szöge radiánban értendő. Az arányon az ellipszis kis- és nagytengelyeinek arányát kell értenünk.

**CLEAR [, [munkaterület] [, stack]]**

Törli az összes változó tartalmát, ezen túlmenően pedig a BASIC munkaterületének és a FOR-NEXT ciklusok, valamint a GOSUB-RETURN hívások számára fenntartott veremnek (stacknek) a méretét állíthatjuk be vele. Maximálisan 1024 lehet mindkét paraméter értéke.

y=CLNG(x)

Az argumentumot 32 bites előjeles egészé alakítja át.

**CLOSE[#]logikai fájlszám 1[,[#]logikai fájlszám2] . . .**

Lezárja a felsorolt fájlokat.

**CLS**

Törli az aktuális képernyőt, és a bal felső sarokba állítja a kurzort.

**COLLISION ON/OFF/STOP**

Engedélyezi, illetve tiltja, hogy az objectek egymással vagy az ablak szélével ütközzenek.

**ON:** ütközés esetén az ON COLLISION GOSUB után megadott sorra adódik a vezérlés

**OFF:** tiltja az ütközések detektálását

**STOP:** ez az opció nem engedi, hogy ütközéskor azonnal életbe lépjen az ON COLLISION GOSUB, viszont az első COLLISION ON kiadása után „bepótolja a mulasztást”.

y=COLLISION (n)

A függvény a legutoljára bekövetkezett objectütközésről szolgáltat információkat.

n=0 az utoljára ütközött object számát kapjuk meg

n= -1 esetén a függvény értéke annak az ablaknak a száma, ahol az ütközés bekövetkezett

n < > 0 a függvény annak az objectnek a számát adja meg, amelylyel az n számú object ütközött.

Negatív eredmény esetén az n-edik object:

-1: a keret felső szélével ütközött

-2: a keret bal szélével ütközött

-3: a keret alsó szélével ütközött

-4: a keret jobb szélével ütközött

**COLOR [előtér][,háttér]**

Az előtér és a háttér színét képes beállítani a palettaszínek valamelyikére.

**COMMON változó [,változó] . . .**

Az így deklarált változók tartalma a CHAIN utasítás során nem vész el.

**CONT**

Megszakítás (STOP vagy CTRL-C) után folytatja a program végrehajtását.

y=COS(x)

Koszinuszfüggvény.

y=CSNG(x)

Az argumentumot egyszeres pontosságú lebegőpontos számmá alakítja át.

y=CSRLIN

A szöveges kurzor y pozícióját megadó rendszerváltozó.

y=CVI (2 bájtos karakterlánc)

y=CVS (4 bájtos karakterlánc)

y=CVL (4 bájtos karakterlánc)

y=CVD (8 bájtos karakterlánc)

A felsorolt függvények az argumentumként szereplő karakterláncot numerikus értéké alakítják. A CVI 16 bites, a CVL 32 bites, a CVS 4 bájtos lebegőpontos, a CVD 8 bájtos lebegőpontos számot hoz létre.

**DATA konstans [,konstans] . . .**

A jó öreg DATA utasítás használata annyival egyszerűsödött, hogy a sztringeket csak akkor kell idézőjelbe tenni, ha vesszőt, kettőspontot vagy szó eleji/szó végi szóközt tartalmaznak.

y\$=DATE\$

Rendszerváltozó, amely az aktuális dátumot tartalmazza hónap-nap-év elrendezésben.

**DECLARE FUNCTION név [(paraméterlista)] LIBRARY**

Egy könyvtárban szereplő gépi kódú rutint mint függvényt deklarál, és a név után álló formátumjelző karakter mondja meg a visszakapott eredmény típusát.

**DEF FN név [argumentum][,argumentum2] . . .]=függvénydefiníció**

A felhasználó egyéni igénye szerint deklarálhat függvényeket ennek az utasításnak a jóvoltából. A függvény akár numerikus, akár karakteres változókat kezelhet.

Itt van viszont az AmigaBasic — és sok más fejlett BASIC-interpreter — egyik gyenge pontja: nem vihető be INPUT segítségével tetszőleges függvény a futó programba. (Pedig ezt már a ZX-Spectrum is tudta!)

**DEF típus változó1 [-változó2][,változó3[-változó4]] . . .**

Az intervallumokba eső változókat a megadott típusúnak deklarálja. Megadható:

INT: 16 bites előjeles egész (%)

LNG: 32 bites előjeles egész (&)

SNG: egyszeres pontosságú lebegőpontos (!)

DBL: dupla pontosságú lebegőpontos (#)

STR: sztring

A változónevek csak egy betűből állhatnak!

**DELETE [kezdőcímké][–[befejezőcímké]]**

Törli a két címke közé eső programrészt, amelyek közül az egyik szükség esetén elhagyható.

**DIM [SHARED] tömbváltozó 1(index1)[,tömbváltozó2(index2)] . . .**

Változótömböket dimenzióval. A dimenziók száma 255, az egy dimenzió belüli méret 32767 lehet maximálisan. Ha kitesszük a SHARED opciót, akkor a tömbök a fő- és az alprogramok (SUB) által egyaránt elérhető, ún. globális tömbök lesznek.

**END**

A program vége, kilépés parancsmódba.

# Kiadványszerkesztő munkahely

<b>Hungarian Ventura Publisher 1.2</b>	149.000,—
<b>Gem 1st Word plus (Gem 3.0 - val) szövegszerkesztő</b>	49.800,—
<b>Billentyűzetkezelő</b>	15.000,—
<b>PC-AT alapkonzfiguráció:</b>	169.000,—
<ul style="list-style-type: none"> <li>• CPU (80286/13 MHz)</li> <li>• 1 Mbyte memória</li> <li>• 1,2 Mbyte floppy drive</li> <li>• 40 Mbyte winchester disk</li> <li>• soros/párhuzamos interface</li> </ul>	
<b>Monitorok csatolóval</b>	
EGA 14"	85.000,—
SIGMA Laser View (A3)	385.000,—
<b>Lézernyomtatók</b>	
STAR Laser-8	360.000,—
HP Laser Jet II	449.000,—
NEC LC-890 Postscript Laser	690.000,—
<b>Scannerek</b>	
DFI HS-3000 Handy Scanner	87.500,—
Microtek MSF-300C	243.000,—
Microtek MSF-300G	395.000,—
Microtek MSF-400G	485.000,—
Microsoft InPort Mouse	32.000,—

A jó ráma minden kép esztétikumát emeli.  
A DTP keretét — **hardverben is** —  
mi biztosítjuk.

1137 Budapest, XIII., Kun Béla rkp. 8.  
Levél cím: 1391 Budapest, Pf.: 218 Tel. 129-230, 328-769



**TRIGON**  
software - hardware

IPARI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET

Telephely:  
1118 Budapest, Bodajk u. 29.  
Telefon: 858-293



# Támadás és védelem 2.

A támadás és védelem értékének numerikus megadása nagyon nehéz feladat. Az egyes programok a legkülönbébb módszerekkel dolgoznak, ugyanis nincs kiforrott, egységesen elfogadott számítási eljárás. A támadás mértékének meghatározására több módszert is nyilvánosságra hoztak a sakkprogramok készítői. Ezek közül az előző alkalommal kettőt már ismerttettem. Elsőként azzal a módszerrel foglalkoztunk, amely az összes megtámadott mező számából állapítja meg a támadási értéket, majd pedig azzal, amelyik csak a saját térfélen lévő megtámadott mezők számát veszi figyelembe. A sort az ellenfél térfelén lévő megtámadott mezők számának elemzésével folytatjuk.

A kezdő állásban ez az érték — az eddig ismertetett módszerekkel ellentétben — mindkét fél számára zérus. A továbbiakban ezt a függvényt a következőképpen jelöljük:

$F_{8p}(W)$  és  $F_{8p}(B)$ . Az  $op$  index az oppo-  
nent szó rövidítése, a  $W$  a white a  $B$  pedig a black szóból származik. Az előbbi jelzések bevezetésével az 1. ábrán látható hadállásra a függvény értékei a következők:  $F_{8p}(W)=14$ ;  $F_{8p}(B)=13$ . Az  $F_{8p}(W)$  függvény kiszámításánál a következő lépéseket kellett figyelembe venni:

$Hc3-b5$ ,  $Hc3-d5$ ,  $Hf3-e5$ ,  $Hf3-g5$ ,  
 $b4 \times a5$ ,  $b4 \times c5$ ,  $Vh4 \times h5$ ,  $Vh4-g5$ ,  $d5 \times c6$ ,  
 $d5 \times e6$ ,  $Fg5-f6$ ,  $Fg5-e7$ ,  $Fg5-d8$ ,  $Fg5-h6$ .

Az  $F_{8p}(B)$  értékének meghatározásához a következő lépésekkel kell számolni:

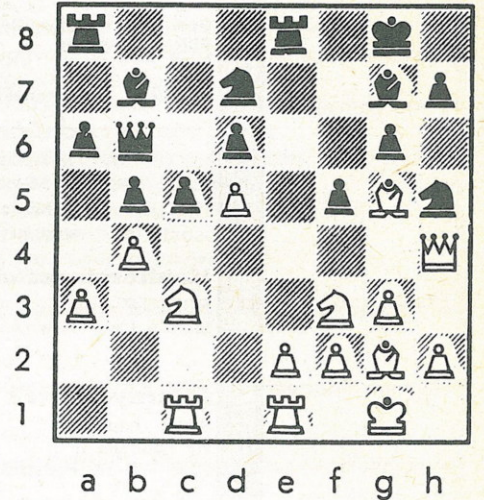
$b5 \times a4$ ,  $b5 \times c4$ ,  $c5 \times b4$ ,  $c5 \times d4$ ,  $f5 \times e4$ ,  
 $f5 \times g4$ ,  $Hh4 \times g3$ ,  $Hh4 \times f4$ ,  $Fg7-d4$ ,  
 $Fg7-c3$ ,  $Be8-e4$ ,  $Be8-e3$ ,  $Be8 \times e2$ .

A 2/a ábrán tanulmányozhatjuk, hogy is alakul ez az érték a játszmák zömében. Az analízisnál az eddigiekben is felhasznált 832 nagymesterjátszmát vettük alapul, amelynek átlagértékeit láthatjuk a féllépcsorszám függvényében.

A függvény kezdetben lassan emelkedik, ugyanis az ellenfél térfelének ellenör-

zése lassan kezdődik (csak a bábuk kifejlődése után), majd a középjátékban stagnál. Azonban ha a nyerő fél és a vesztes fél függvényértékeit összehasonlítjuk, akkor szembevetendő, hogy a nyerő játékos a játszma kezdetétől a végéig folyamatosan minden esetben több mezőt ellenőrzött az ellenséges térfélen, mint az ellenfele.

Érdekes összehasonlítani ezt a függvényt az előző alkalommal ismertetett saját térfélen lévő megtámadott mezők függvényével (3. ábra). Ahol az egyik függvénynek magas az értéke, ott a másiknak alacsony és fordítva. Nagyon hasonlóan viselkednek, mintha az  $x$  tengelyre tükrözve megkapnánk a másikat. Dap Hartman holland sakkprogramozó szerint a harmadikként ismertetett eljárást érdemes a sakkprogramba beépíteni, szerinte ez növeli legjobban a program nyerőesélyeit.

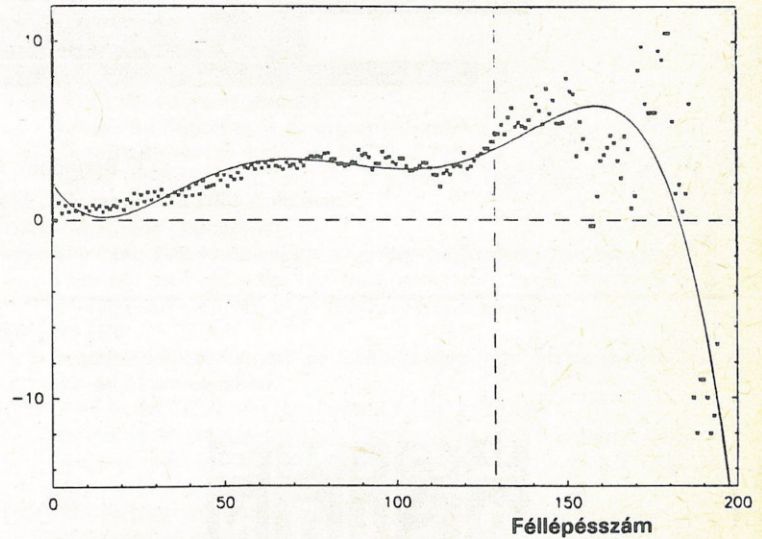


1. ábra KASZPAROV — Fjodorovics. Graz, 1981 A 42. féllépés után

Kovács P. Attila

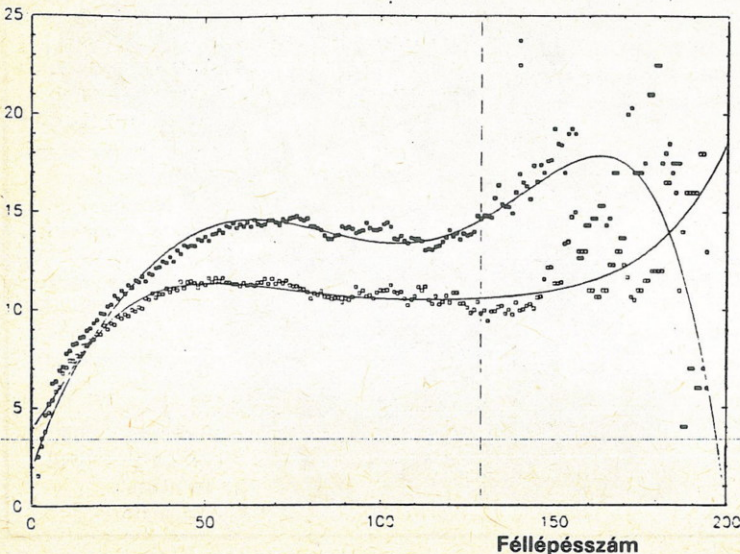
2/b ábra

A nyerő és a vesztes fél függvényértékeinek különbsége



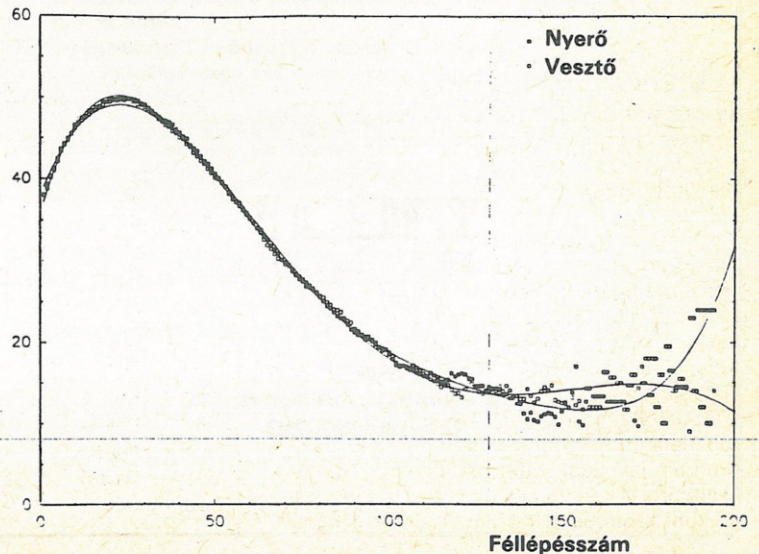
2/a ábra

Az ellenfél térfelén lévő megtámadott mezők száma



3. ábra

A saját térfelén levő megtámadott mezők száma



# PROGRAMTERMÉK

## Ismerd meg anyanyelved!

Meggyőződésem szerint sokkal többen hiszik, hogy jól beszélnek a magyar nyelvet, mint ahányan ezt tényleg elmondhatnák magukról. Már számtalanszor vettem részt vérforraló, de halott vitákban, amelyek arról folytak, hogy ez vagy az a kifejezetten magyar szó mit is jelent pontosan. A jövővényszavakról ne is beszéljünk. További gond, hogy néha nagy nevű vagy nagy buzgalmú aktívák „megszólalásig” szimpatikusak. Azután már nem — a katasztrofális beszédhibák miatt. Mintha az alapismereti tárgyak oktatásánál hiányoztak volna az iskolából. Persze az oktatást sem dicsérik, hogy csak ennyit tudott ráragasztani az érintettekre.

A fenti „morgás” természetesen megint valami program kapcsán jutott eszembe. Még hozzá nem is egy, hanem két programról van szó. Mindkettő anyanyelvünk jobb megismerését célozza. A programok önmagukban elég rövid lélegzetűek, de jelentőségüket ez nem ki-sebbiti. Szerintem a két program együtt kitenne egy nagyobbat, ha szembe kellene állítani őket azokkal, amiket eddig vizsgáltam.

### Szójáték

A Szójáték célja a magyar szókinccs mintegy 400 szavának felismertetése, olykor kicsit meghökkenítő szómeghatározások alapján, három, egyre nehezített megfejtési móddal. A játékot legfeljebb négyen játszhatják. A kérdések száma fordulónként egytől nyolcig terjedhet.

Az első fordulóban megkapjuk a megfejtendő szó összes betűjét ábécérendben (ezt ugye egy profi számítástechnikus az anagrammák kanonikus alakjának nevezné), továbbá egy meghatározást. Az idő telik. Lejárta után megkapjuk a választ. Ha az idő lejárta előtt sikerül megfejteni a szót, akkor a program a versenykiértékeléshez feljegyzi a felhasznált időt. Rossz válasz esetén büntetőidőt számít fel. A kiértékelés pontos algoritmusát sajnos nem lehetett kifigyelni, a leírás pedig homályos.

A második fordulóban csak a meghatározást kapjuk meg, a betűszám és a szó számára pedig egy kipontozott maszk jelenik meg. Ha nem tudunk válaszolni, akkor segítséget kérhetünk. Minden segítségkérésre véletlenszerűen egy-egy betűt kapunk meg a megfejtésből, még hozzá a megfelelő szópozícióban. A segítségért persze büntetőidő jár. Ezért súlyozni kell, mi éri meg jobban, kicsit törni még a fejet, vagy kapásból segítséget kérni. Néhány elég nehéz szóval is összeakadtam.

A harmadik forduló már igazán nehéz. A feltevételek elvileg azonosak a második fordulóbellekkel, de a segítség ravaszul csak a kanonikus anagramma betűit adja meg véletlenszerűen. Azokkal aztán néha nem sokra megy az ember.

### Rakéta indul!

A harmadik forduló után a program értékeli a játékosok eredményét, és a legkevesebb időt felhasználó játékos rakétája egy kis animációt beiktatva elindul. Amint a rakéta elhúz, megjelenik a versenyzőnkénti kiértékelés, de érthetetlenül lassan. A versenyzők között van mód a visszavágóra.

A 400 szavas szótár, amelyet a program tárol, nem látszik túl nagynak, és ráadásul a program jelenlegi változatában nincs is mód a cseréjére. A szerző azonban úgy tervezi, hogy további szótárakat fog mellékelni a program-

hoz, és így újabb változatokat fog megjelentetni. Ha ez tényleg bekövetkezik, akkor a szerző teljesítménye magasabbra értékelendő. A játék közben egyébként a 400 szóból a választás véletlenszerűen történik, és a program arra is vigyáz, hogy elég hosszú legyen az ismétlődés ciklusideje. Újraindításakor a szósorozat elvileg nem ismétlődik.

A program kezelése általában egyszerűnek mondható, bár van néhány apróság, ami bosszantott. Például a kérdésszám 1–8 beállításakor gond van, ha a szám körbepördül. A 8-as után valami érthetetlen okból nehéz elkapni az 1-est. Rendkívül kellemetlen, hogy a törlőbillentyű helyett a csillag gombot kell használni javításra, és ráadásul az egész szót újra kell kezdeni. Ha az ember tévedésből mégis a törlőgombbal javít, akkor igen pórul jár. A törlőgomb ugyanis működni látszik, de a választ a program rossznak értékeli. A kezelhetőséghez tartozó problémának itélem, hogy a program meglehetősen lassan dolgozik.

A program dokumentációja elég rövid, szövege homályos, és ráadásul nem is elég magyaras, jóllehet a cél éppen a magyar nyelv is-tápolása. A programban elhelyezett ismertetés egy klasszissal jobb a dokumentációnál. Igazából a dokumentáció nem is nyújt többet nála.

A Szófejítő program kifejezett célja tehát az, hogy a játékosok egymással versengve észrevétlenül mélyítsék el a magyar nyelv kiválasztott szókinccsének árnyalatait, és ezzel gyarapítsák intelligenciájukat. A program segítheti a szabadidő értelmes eltöltését akár az iskolában, akár otthon. A program az összértékelésben csak jó minősítést kapott, mint ahogy részleteiben az értékelő táblázat első oszlopából kitűnik. A szerző törekvését mégis nagyra becsülöm, már ami a témát illeti, mert a magyar nyelv ápolására minden befogadható eszközt szükségesnek tartok.

### Pótold ki a szavakat!

A fenti program az olvasási nehézségekkel küszködő — dyslexiás — tanulók beszédhibáinak és helyesírási hibáinak korrekciójára szolgáló drill programként született. A szerző profi

	MINŐSÍTŐ ADATOK	
	(1)	(2)
Kezelhetőség:	közepes	jó
Teljesség:	jó	kiváló
Dokumentáltság:	közepes	közepes
Használhatóság:	kiváló	kiváló
Ár/teljesítmény:	jó	jó
Összbenyomás:	jó	jó

gyógypedagógus (de sajnos kicsit gyenge programozó, talán jobb lett volna neki némi segítséget nyújtani a programozáshoz). A program elsősorban alsó tagozatos gyerekek oktatására alkalmas. A kísérletek szerint leginkább az értelmileg fogyatékos tanulóknak van hozzá a legnagyobb türelmük. A probléma oka valahol a program sebességében keresendő...

A program szótára 600 szavas. Egy-egy menetben a tanulóknak tíz szót kell megfejtenie. A program indulásakor vagy újraindításakor választani lehet 8 gyakorolható „problémakör” közül. A nyolc problémakör a következő: B-P-M, N-M-G, K-G, L-R, GY-NY-LY, L-R-N-J, R-M-N-B és B-D. Nem lennék meglepve, ha a program néhány felnőttnek is okozna egy kis meglepetést. Ahogy a zenében fontos a hallás, ugyanúgy a beszédben is fontos az úgynevezett belső hallás. Mintha belülről valaki súgna nekünk, hogyan is kell kimondani és leírni egy-egy szót. Jaj nekünk, ha a súgás rossz! A korrekció bizony igen keserves munkával jár. Nos, ebben a munkában igen sokat segíthet a kifogyhatatlan türelmű számítógép.

### Kell egy ügyes tanár

A program kezelése végtelenül egyszerű, de még ez a végtelenül egyszerű kezelés sincs jól leírva sem az igen lakonikus dokumentációban, sem a program képernyőszövegeiben. A program szinte rá van utalva arra, hogy egy ügyes tanár, aki már kiismerte a működését, szükség esetén ki tudja segíteni a tanulókat, hogy most éppen mit kell csinálni. Egyébként rajtakaptam a programot: két egymás utáni menetben előfordult a tizből egy ismétlődő szó. A 600-as szótár és a nem túl nagy tárfoglalás (kb. 13 kb-át) alapján ez nem látszik indokoltnak. Érezhetően nincs beiktatva ismétlődésszűrő. Ez a már említett hiányos számítástechnikai ismeretek számlájára írható.

A számítástechnikai ügyetlenségek ellenére a program jó koncepciójú, a témában alapos felkészültséget mutató taneszköznek minősíthető, már amennyire ezt egy fizikus meg tudja ítélni. Ezért a program összműködése szintén jó, sőt részleteiben jobban is tetszett, mint a Szójáték. A részletes adatokat a minősítés második oszlopa mutatja.

A két program mind az iskolai munkát, mind az önképzést támogathatja, s véleményem szerint ott a helyük minden iskolában és művelődési házban, ahol a gépi feltételek adottak!

Zsadányi Pál

#### ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Tudományszervezési és Informatikai Intézet, Kereskedelmi Iroda
Terméknév:	(1) Szójáték (2) Pótold ki!
Szerzők:	(1) Gombás Károly (2) Rezes Ferenc
Géptípus:	(1) TVC, (2) Plus/4 és TVC
Hordozó:	kazetta vagy floppy
Dokumentáció:	(1) öt oldal, (2) két(!) oldal
Ár:	(1) 300 forint (floppy) (2) 190 forint (kazetta)

# ADOK—VESZEK—CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetések közlünk. A díjszabás: kereskedelmi tevékenységet folytatóknak gépelt soronként (60 karakter) 100,- Ft, másoknak az első sor 50,- Ft, minden további sor 20,- Ft. Kérjük, hogy a hirdetés díját az NJSZT OTP V.ker. fiókjánál (218-98055) vezetett 508-8609 rendezvényszámára vagy 1054 Bp., Báthori u. 16. címére fizessék be, rőzsaszínű postautalványon (jelölve, hogy apróhirdetés), a befizetést igazoló szelvényt pedig csatolják a hirdetéshez. Hirdetéseiket a szerkesztőség címére várjuk (1371 Bp., Pf.: 433). Az NJSZT tagjai továbbra is kedvezményesen hirdethetnek (az első sor ingyenes), de kérjük, hogy adják meg tagsági számukat. Azokat a hirdetéseket, amelyek a hónap első napjáig megérkeznek, már a hónap végén viszontláthatják lapunkban.

## ADOK

**Amiga** programok nagy választékban eladók! Érdeklődni lehet: Keresztes Gábor, Bp., Laky köz 11. 1142. Tel.: 643-452

**Amiga!** Legjobb külföldi programok! Ballaságymat, Pf.: 118. 2661

**Atari 800XL** és hozzávaló magnó, valamint joystick eladó 10 000 Ft-ért. Dorgai Csaba, Székesfehérvár, Velinszky L. u. 20. 8000

**Atari 800XL**-re új programok eladók. Tel.: 643-452

**C Plus/4**-es programok nagyon olcsón (5 Ft/db) eladók. Válaszborítékért listát küldök. Pápes Zoltán, Maglód, Ady E. u. 14. 2234

**C16** és **C64**-es programokat adok kazettán (8 Ft/db). Válaszborítékot kérek! Csóka István, Győr, Pablo Neruda u. 3. 9023. Tel.: (96) 15-974

**C Plus/4**, **C64**-es programok eladók kazettán és lemezen (10 Ft/db). Hábel Péter, Szolnok, Versegly u. 38. 5000

**C64**-es kazettás programok olcsón (5 Ft/db)! Ifj. Berkovits Imre, Sárvár, Alkotmány u. 17. 9600

**C64**-re programokat adok kazettán és lemezen. Cím: Horváth Tibor, Győr, Tas vezér u. 35/A. 9012

**Commodore 64**, datasett és joystick 18 ezer Ft-ért eladó. Keremem: Printfox, Disk-Demon, GEOS 2.0 és Newsroom nevű programokat. Pajzer Ferenc, Bp., Tóvirág u. 2. VI/26. 1108

**C64**-es lemezek eladók (200 db) 100 Ft/db - szuper programokkal. Kérésre listát küldök. 1351-es egér, Citizen 1200 + interfész, magnó (6000, 25 000, 2000 Ft) eladók. Némethi Ferenc, Bp., Nagyenyed u. 8/a. 1182

**Commodore 64** számítógép, 1541 floppy, magnó, 801 printer, CP/M cartridge és 600 játék eladó. Dikó István, Tel.: napal 851-344/156, este 373-193

**C64**-es programok eladók, 16 Ft/db. Kérésre listát küldök. Vass Viktor, Zalalövő, Szabadság tér 14. 8999

**Commodore 64** számítógéphez új programok, lemezzel és kazettával olcsón eladók. Tel.: (62) 27-530

**C64c** számítógép, VC 1541 II. floppy, 32 kb-ás cartridge, 2 db joystick, könyvek, játékok, felhasználói és segédprogramok lemezen, csak együtt 32 000 Ft-ért eladók. Molnár Ferenc, Eger, Hell Miksa u. 2. 3300

**C64** és 1541-es lemezegység 4000 programmal (400 db lemez, 300 db kazetta) eladó külön-külön is. Járóka László, Bp. Mogyoródi-Jerney ltp. LK. 1007 ép. L. lh. I/2. 1141

**C64**-hez Final Cartridge III. olcsón eladó. Tel.: 645-442

**C64** sürgősen eladó 2 joystickkal, magnóval, 1541/II floppyval és kb. 300 programmal. A gép és a floppy akár külön-külön is. Irányár: 35 000 Ft. Érdeklődni lehet: Nagy Tibor, Győrújbarát, Fő u. 31. 9081

**C64** programokat nagy választékban és 5.25-ös lemezeket adok. Kérjen tájékoztatást! Kasza Viktor, Siófok, Fenyves s. 11. 8600

**C64** Fastload, Turbotape, Help Plus Cartridge 650 Ft-tól, letilthatatlan Reset gomb 250 Ft, Enterprise joystick illesztő 351 Ft és Enterprise programok 10 Ft/db. És még sok más szolgáltatás! Rendkívüli árak! Szállítás postán is! Répás Sándor, Bp., Fegyvernek 10. VI/61. 1116

**Enterprise 128** + Sanyo DM2112-es monochrom monitor (zöld) + 2 db joystick + könyvek + játékok és felhasználói programok kazettán eladók. Irányár: 20 000 Ft. Balla Tibor, Tatabánya II. Erdész u. 26. II/3. 2800

**Enterprise** programokat adok, cserélek, olcsón (10-20 Ft), rövid határidővel. Németh Antal, Bonyhád-Majos, VIII. u. 26. 7187

**Enterprise 128** programok eladók, Magyarországon a legolcsóbban, 50 Ft/db. Válaszboríték esetén listát küldök! Cím: Bekéssy László, Bp., Szilágyi u. 62. 1174

**Enterprise** programokat adok, cserélek (lemezen, kazettán egyaránt) olcsón (20 Ft), rövid határidővel. Cím: Kato Zoltán, Majos, II. u. 17. 7187

**Enterprise** programokat nagyon gyorsan és olcsón adok, ill. cserepartnereket keresek. Széles programválaszték. Cím: Sándor József, Bonyhád, Bezerédi u. 41. III/5. 7150

**TV-Computer** 64 k-ra válogatott programokat olcsón adok vagy cserélek. Csere esetén listát kérek. Csatlós Béla, Mezőtúr, Ifjússági ltp. XIX. ép. 5400

**TV-Computer** programokat adok-veszek-cserélek. Listát, választborítékot kérek! (750 db program) Molnár János, Szolnok, Jászi F. út 10. VI/25. 5000. Tel.: (56) 31-085

**Primo SSD**-hez 32 k-s komplett felhasználói programcsomag égetése. Egyéb Eprom égetések masterből vagy lemezzről 24 órán belül. Vidékre is. Horváth László, 583-544 napközben.

**Spectrum** 48 k programokat (jó minőségben) adok, veszek és cserepartnereket keresek hosszú távra. Listát kérek, választboríték esetén küldök is! Vágó Zoltán, Bp., Sörgyár u. 64/D. II/2. 1104

**ZX-Spectrum+** 48 k-s számítógép, joystick interfésszel, 2 db joystickkal, programokkal, szakkönyvekkel eladó. Szentmiklósi Csaba, Szeged, Selyem u. 12/A. 6723

**ZX-Spectrum** (48 k) számítógép magnóval, joystick-illesztővel, irrodalommal eladó. Csak egyben! Ár: 19 000 Ft. Asztalos István, Vác, MN SIHKK, Pf.: 271. L-1, 2601

**ZX81** 32 k-s memóriával magyar és angol nyelvű gépkönyvvel, szakirodalommal eladó. Ajánlatokat levélben az alábbi címre kérek: Luczi Péter, Debrecen, Varga-kert u. 3. I/13. 4031

**Citizen 1200** printer sürgősen eladó! Érdeklődni levélben: Bátorfi Zoltán, Rákócziútfalva, Dózsa Gy. u. 17/a. 5085

2 datasettel (**C64**) akar programot másolni? Rendelje meg szuper olcsó másoló-készülékemet! Válaszborítékért tájékoztatást küldök! Tóth Lajos, Székesfehérvár, Vöröshadsereg út 82. 8000

Fordításokat készíték a **64'er Magazinból**. Több mint 3000 oldalnyi kész

anyag! Kedvező árak. Textomat+, GEOS 1.2, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Bp., Pf.: 400. 1446. Válaszborítékot kérünk!

**Joystick szerviz** a Flórián Áruházhoz közel! Javítás, magnófej-beállítás. C64 játékprogramok kazettán és floppy 10 Ft/db. Bp. III. ker. Kerék u. 36. IV/24. Hétfőn és szerdán 17-től 19 óráig.

**Jutányos feltételekkel vállalkozók középfokú német nyelvvizsgára való felkészítést.** C64 vagy C128 és floppy szükséges. Érdeklődni: Szolnoki Béla, Bp., Pf.: 400. 1446

**MPS-801** nyomtató eladó. Sramó András, Pécs, Pintér István u. 9. 7636  
5'25"-os mágneslemezek eladók. Ár: 700-3000 Ft-ig. Itt kaphatók a legújabb, legjobb C64 programok. Cím: Kótai Balázs, Sopron, Margitbányai u. 37/a. 9400

150 db lemezemet **C64**-es játékprogramokkal eladom. Ár 105 Ft/db. Eladó 15 db kazetta is játékprogramokkal. Ár: 300 Ft/db. Kazettánként 40-50 program. Boldizsár Gábor, Rém, Petőfi u. 23. 6446

**Színvonalas programok** eladók kazettán 10 Ft/db. Kb. 1200 programom van. Szappanos Péter, Debrecen, Nagyerdei krt. 32. 4028

**Vegye a legjobbakat a legfrissebbet!** Magyarországon legolcsóbban az I LOVE CEMENT CREW-től veheti meg a pár hetes törésű C64-es programokat és demókat. Megfelelő partner esetén csere is lehetséges, lemezen és kazettán is. Cím: Kozák Zoltán JR. Sopron, Schärmár K. u. 2. 9400. Üdvözlöt a következőknek: ADT, CYNDI, FBI REDS, TGS, HV, QLT, DEY, TKC.

## VESZEK

**C64**-re Shoot'em up construction kit c. programot vásárolnék. Cím: Kriskó István, Koskonyú, Rákóczi út 16. 6211

Megvételre keresem a Mikroszámítógép Peller Imre eddig megjelent összes számát. Peller Imre, Nyíregyháza, Kert u. 6. 4400

## CSERÉLEK

**Amiga** programokat cserélek. Sierra on line szoftvereket keresek. Válaszokat listával kérek. Szivovics Ernő, Szőreg, Szerb u. 30. 6771

**C16** és **C Plus/4** programokat cserélek. Listát kérek! Forgó Gyula, Cseengele 6765, Tel.: 631

**C64**-re játékok és más egyéb programokat cserélek, olcsón eladok elsősorban kazettán, de lemezen is. 800 programmal rendelkezem. Minden levélre válaszolok, listát kérek! Várhegyi István, Nyírbátor, Derzsi u. 31. 4300

**Commodore 128**-as és **64**-es programokat cserélek és mások lemezen. Minden levélre válaszolok! Hegedűs Gábor, Budapest, Menyecske u. 1. 8. em. 49. 1112

**Enterprise, Atari 800XL** programokat cserélek, veszek. (Enterprise-ra kazettán!) listát kérek! Minden levélre válaszolok! Felhasználói programokhoz leírást keresek. Fazekas Dániel, Komló, Viola u. 31. 7300

**Enterprise** programcsere lemezen is. Listát kérek! Cserealapom: kb. 500 program. Kusznyik Gábor, Debrecen, Virág u. 4. 4029

**Spectrum** programokat cserélek! Listát kérek! Keremem a Last Ninja 2. című programot. Fülöp Ferenc, Kaposvár, Laborfalvi u. 54. 7400

Ez a rovatunk **KODEX 2000** szövegszerkesztővel készült.

## Szó, ami szó

Angol-magyar interaktív fordítást segítő programmal (AMI) jelentkezett tavasszal az SZKI. A program két fő részből áll: az Ország László-féle kézisótáron alapuló, mintegy 37 ezer angol címszót és a kb. 23 ezer angol, amerikai kifejezést tartalmazó szótárból, valamint a szótárát kezelő memóriarezidens programból, amely bármikor, tetszőleges környezetből indítható, s lehetővé teszi az aktuális képernyőn

lévő szavak fordítását. A program a rendhagyó és a ragozott alakokat is kezeli.

## Műanyag védi a morzsákat

Az elektronika parányi alkatrészeit, a morzsákat nagyon óvatosan kell szállítani. A legkisebb, szemmel nem látható mechanikai sérülés működésképtelenné teheti azt a készüléket,

amelybe az ilyen morzsát beépítik. Még veszélyesebb, ha a „belső sérülést” elektrosztatikus kisülés okozza, különösen, ha a morzsát számítógépbe építik be. Ezért az elektronikus alkatrészeket különleges csomagolásban szállítják a gyártól a beépítés színhelyéig. Az NSZK-ban erre a célra Ultramid márkanéven ásványi szállal erősített olyan poliamid műanyagot fejlesztettek ki, amely egyfelől nagyon szilárd és mérettartó, másfelől villamos jellemzői révén biztosan védi a morzsákat az elektrosztatikus kisülésektől.



# KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

**Orbán Katalin:**  
**COBOL az IBM PC-n. Professional COBOL**  
(Budapest, 1988.  
Novotrade, 381 oldal.  
Ára: 319,— Ft.)

A COBOL nyelv a legelterjedtebb adatfeldolgozó programozási nyelvek közé tartozik szerte a világon. Népszerűségét többek között beszédes nyelvezetének, nem túl bonyolult szintaktikájának is köszönheti, valamint annak, hogy a hibák felderítése — más nyelvekkel összehasonlítva — viszonylag egyszerű a COBOL-ban. A COBOL programozási nyelv kezdetben a nagyszámítógépek nyelve volt. A személyi számítógépek elterjedésével létrehozták a klasszikus adatfeldolgozási nyelveknek, így a COBOL-nak is, személyi számítógépekre kidolgozott változatát. A könyv a Micro Focus cég által kidolgozott PROFESSIONAL COBOL nyelvet és rendszert ismerteti meg az olvasóval.

Ez a rendszer számos olyan szolgáltatást is nyújt, amely a nagyszámítógépes, hagyományos COBOL nyelv használatánál nem lehetséges.

A könyvet szerkezete alkalmassá teszi az önálló tanulásra is, azaz viszonylag egyszerű feladatokon keresztül ismerkedhet meg az olvasó a COBOL nyelvvel és magával a szoftverrel is.

**Knuth, Donald E.:**  
**A számítógép-programozás művészete 3.**  
**Keresés és rendezés.**  
(Budapest, 1988.  
Műszaki Könyvkiadó, 761 oldal.  
Ára: 249,— Ft.)

Ez a könyv a sorozat első kötetében szereplő, az információs struktúrákról szóló második fejezet folytatásának tekinthető, mert a lineárisan rendezett adat fogalmával bővíti az alapvető adatszerkezetek körét. A „Rendezés és keresés” cím azt sugallja, mintha e könyv csak olyan rendszerprogramozók számára készült volna, akik általános célú rendező rutinok létrehozásával vagy az információ-visszakeresés alkalmazásával foglalkoznak. Ezzel szemben a rendezés és keresés témaköre ideális keretet biztosít igen sok általános, fontos kérdés megvitatásához, pl. hogy hogyan készíthetők jó algoritmusok. Hogyan tökéletesíthetők adott algoritmusok és programok? Hogyan vizsgálhatjuk matematikai módszerekkel az algoritmusok hatékonyságát? Miként lehet az egy feladatra írt különféle algoritmusok között ésszerűen választani? Hogyan használhatók hatékonyan a külső táruk — a mágnesszalagok, -dobok vagy -lemezek — nagy adatbázisok kezelésére?

A szerző szerint szinte minden, a programozásra vonatkozó fontos kérdés valahol a rendezéssel és kereséssel kapcsolatban merül fel!

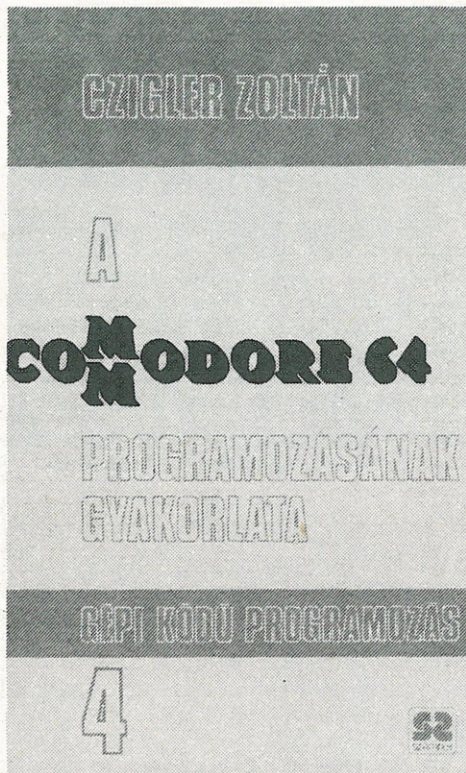
**Germain, Clarence B.:**  
**IBM PC XT/AT programozói**  
**kézikönyv**  
(Budapest, 1988.  
Novotrade, 388 oldal.  
Ára: 390,— Ft.)

A kötet szerzője áttekintést ad a számítógépek történetéről, ismerteti a számítógép legfontosabb részeit. Részletesen írja le a személyi számítógépek felépítését és tartozékait. Ismerteti a DOS operációs rendszert, áttekintést ad a FORTRAN nyelvről. Külön fejezetben tárgyalja a FORTRAN utasításokat. Bevezet a COBOL nyelvű programozásba, igen részletesen közli a COBOL sajátosságait. Hasonlóan foglalkozik a BASIC nyelvvel is. Ismerteti a gépi kódú programozás fontos elemeit és az ASSEMBLER nyelvet is.

A könyv számos hasznos függelékkel is tartalmaz, köztük a különböző billentyűzetkódokat és a különböző számítógépes nyelvekkel kapcsolatos táblázatokat.

**Czigler Zoltán:**  
**A Commodore 64 programozásának**  
**gyakorlata. 4. Gépi kódú programozás**  
(Budapest, 1989. SZÁMALK,  
211 oldal. Ára: 198,— Ft.)

A könyv a C-64 gépi kódú, illetve assembly szintű programozásáról szól. Az assembly-programozás a BASIC-nél lényegesen gépközelebb, ezért kicsit körülményesebb, ám igen hatékony módja a C-64 programozásának.



A programozás körülményessége és nehézsége buszán megterül, mivel az assembly nyelven megírt program több lehetőséget ad a gép kedvező tulajdonságainak kihasználására, ugyanakkor gyorsabban fut, és kisebb memóriaterületet igényel, mint egy ugyanezerre a feladatra írt BASIC program. A C-64 géphez készült megannyi népszerű játék- és rendszerprogram is szinte kivétel nélkül mind assembly nyelven íródott.

A gépi kódú vagy assembly szintű programozáshoz legalább annyira nem szükséges felsőfokú matematikai végzettség, mint a BASIC-hez, csupán logikus gondolkodás és következetesség. A kötet feltételez ugyan egy kevés BASIC alapismeretet, de nem kell túlzottan gyakorlottnak lenni a BASIC-ben ahhoz, hogy bevezethessen a gépi kódba. Ennél többre ez a könyv nem vállalkozik. Elvezeti az olvasót addig a pontig, amíg megérzi ennek a sajátos és szép világnak a lényegét, és ahonnan már egyedül tanulhat tovább.

**Pirkó József:**  
**Turbo Pascal kezdőknek — haladókknak**  
**4.0 verzióig**  
(Budapest, 1989. LSI ATSZ,  
245 oldal. Ára: 192,— Ft.)

A kötet bevezetője részletesen, történeti áttekintésben ismerteti a Pascal magas szintű programnyelv megalkotásának körülményeit, az alkotó Nicklaus Wirthról és a nyelv nevét adó Blaise Pascalról is ír. A Turbo Pascal programnyelv és elsősorban az IBM PC-kre készült változata az egyik legsikeresebb verziója. En-

nek a nyelvnek a Borland Intézet (USA) volt a kifejlesztője, újabb és újabb verziókat jelentettek meg.

A kötet a 4.0 verzió részletes leírását tartalmazza.

**IBM PC DOS. 1—3. köt.**  
**1. A PC DOS használata**  
**2. A PC DOS felépítése**  
**3. PC DOS programozói segédlet**  
(Budapest, 1988. LSI ATSZ, 3 db.  
Ára: 692,— Ft.)

Az IBM PC számítógépekre készült PC DOS operációs rendszer a számítógépet interaktívan kezelni akaró programozók számára készült. Funkcióállománya jóval bővebb a CP/M operációs rendszerénél, nagy megszakítás-készlethez biztosít hozzáférést. A fájlokat bonyolult fa-hierarchiába lehet rendezni, amely nemcsak a felhasználóknak, hanem az egyes feladatok fájlhalmazainak kényelmes, némi biztonságát is adó strukturálását is lehetővé teszi.

A könyvet az IBM PC-vel most ismerkedők és a már tapasztalt programozók egyaránt haszonnal forgathatják. A kötet a DOS 2.10, 3.00, 3.10, 3.20 és 3.30 verzióval foglalkozik.

A programozói segédlet az IBM PC XT/AT-kompatibilis mikroszámítógépek egyik leghatékonyabb programfejlesztő eszközének, a macroassemblernek a használatához szükséges ismereteket foglalja össze. A DOS kézikönyvben a megszakítások és funkcióleírások megértéséhez szükséges legfontosabb információk találhatóak a 8086-os mikroprocesszor családról, a 3. kötet a processzor felépítését és az utasítások struktúráját is tartalmazza.

**Murray III., W. H. — Pappas, Ch. H.:**  
**A 80386-/80286-os processzor Assembly nyelvű**  
**programozása**  
(Budapest, 1988. McGraw—Hill — Novotrade Rt.,  
409 oldal. Ára: 392,— Ft.)

A szerzők azt a célt tűzték maguk elé, hogy bevezetik az olvasót a hatékony gépi kódú programozás világába, megtanítják az egyszerű, de hatékony assembly programozást, és egyúttal referencia kézikönyvet készítenek, amely a programozási utasításokon kívül mintaprogramokat is tartalmaz.

A könyv az assembly programozást példák bemutatásával oktatja. A programok listái minden szükséges fejléccel tartalmaznak, tehát a programkódoknak a listákról való pontos beírása után a programok futtathatók.

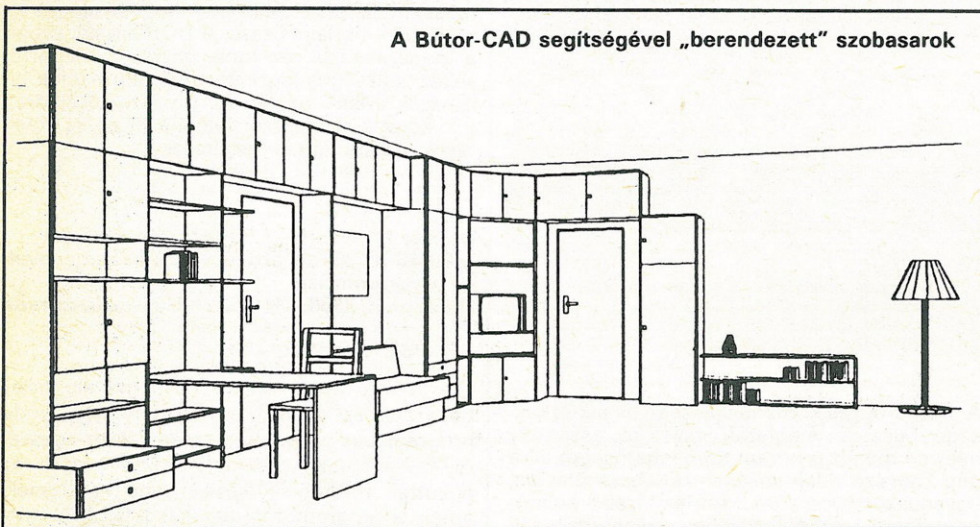
A szerzők fő törekvése az volt, hogy a programokat bemutassák, elmagyarázzák és dokumentálják azért, hogy az olvasó, akár kezdő, akár gyakorlott, az egyes utasításokat és programozási stílusokat minél jobban megértse.

**REXLIB Resident Extended Library**  
**Képernyőkezelő, file-kezelő rezidens**  
**rutin- és utility-csomag MS-DOS és Novell**  
**Netware—286**  
**környezetben fejlesztők számára**  
(Budapest, 1988. LSI ATSZ—SZÁMSZÖV,  
189 oldal.  
Ára: 205,— Ft.)

A REXLIB rezidens működése lehetővé teszi a hardver- és szoftverkörnyezet interaktív módosítását. Csak olyan gépen fut, amelyen megtalálható az ehhez szükséges keycard. A kötet melléklete a SZÁMSZÖV programozói által kidolgozott fejlesztői programcsomag, amely a Turbo Pascal 4.0 leírását is tartalmazza.

## Hová kerüljön a Bútor?

A Bútor-CAD program első verziójával modulrendszerű bútorelemeket lehet tetszés szerint összerakni az ugyancsak tetszőlegesen választható alaprajzú, magasságú szobában. Az egyes bútorelemek egyetlen mozdulattal előhívhatók, és néhány mozdulattal a helyükre illeszthetők, így akár a boltban megjelenő vevő elképzeléseit is azonnal látványosan, szemléletesen lehet képi formába önteni, szükség esetén módosítani, gazdagítani. Emellett az egyes elemek maguk is könnyen módosíthatók, alakíthatók, s gyorsan ellenőrizhető, miként illeszkednek az új elemek a régiekhez. A részben vagy teljesen kész tervet, bútorösszeállítást bármely nézőpontból meg lehet jeleníteni, ki lehet rajzoltatni. A kész összeállításban szereplő elemekről listát lehet készíteni, ami a számlázás, illetve a rendelés alapjául is szolgálhat. A Controll Kisszövetkezet által forgalmazott programot bútortervezéssel, lakberendezéssel foglalkozó szakembereknek és bútorboltoknak ajánlják.



A Bútor-CAD segítségével „berendezett” szobasarak

## Iskoláknak 8 bit

Az oktatási intézmények számára készült az NDK-ban az A 5105 típusú mikroszámítógép. Nem csupán a klasszikus oktatási feladatokra alkalmazható, hanem mérő-, szabályozókészülékekhez kapcsolva is. A gép három alapegységéből épül fel: a hordozható, lapos billentyűzetben rejlő alapgépből, a hajlékonylemez tárolót és a perifériaillesztőket tartalmazó egységből, valamint a megjelenítőtől. A készülék SCPX 5105 nevű operációs rendszere tartalmazza a BASIC-fordítót, a mágnesszalagos és a hajlékonylemez tárolók vezérlőrendszerét is. A gép legnagyobb hátránya, hogy még mindig a 8 bites, U880 típusú, a Z80-nal kompatibilis mikroprocesszort tartalmazza, s így eltér a világszerte elterjedt, IBM PC-vel kompatibilis gépektől.

## Autóteszt

Az amerikai MPG International Corporation és a budapesti Primaút Gépjárműjavító Vállalat vegyes vállalat létrehozását fontolgatja a Computest VDT-04D mikroszámítógépes autótesztelő gyártására.

A Computest ezer gépjárműtípus alapvető adatait tartalmazza a tárolójában. Használata során ezeket a bázisadatokat hasonlítja össze az általa mért eredménnyel. A különbséget megjeleníti a képernyőn, de szükség esetén a nyomtatón papírra is kiírhatja. A Computest alkalmazásához nem szükséges megbontani a gépkocsit, használata mindössze két csatlakozási pontot igényel. A mérési idő tíz perc. Alkalmazása igen sokrétű: a szervizektől, a hatósági vizsgán át a Forma-1-es versenystállóig terjed.

A vegyes vállalat a Computest berendezéseket hazai és kelet-európai piacokon kívánja értékesíteni.

## Notesz vakoknak

Apró elektronikus készüléket fejlesztettek ki a bécsi műszaki egyetem kutatói, hogy megkönnyítsék a vakok életét. A Notaphon nevű szövegfeldolgozó a zsebben is elfér. Tulajdonosa, használója a vakok számára szolgáló Braille-írással jegyzetelhet bele, az elektronikus notesz egyszerű gombnyomásra beszédhang formájában adja vissza a szöveget. Így

## Gépet ajándékba

Szombathelyen, a Savaria Közlekedésgépészeti Szakközépiskolában 1986 óta képezik rendszeresen a tanulókat — tantervi előírások alapján — a számítástechnikai ismeretek alkalmazására. A gépjármű-technikai, a vasútgépész és az építőgépezés tanulók heti két órában tanulják a számítástechnikát. Jelenleg 40 gépből áll a számítógépparkjuk, 16 darabot az Econorg Számítástechnikai Közös Vállalattól kaptak ajándékba. A legkorszerűbb technikát egy vásárolt Multitech 710 típusú, XT kompatibilis gép jelenti.

## Környezetvédelem

Az ÉGSZI Kutató és Szervező Kft. Települési Környezetvédelmi Információs (TKI) rendszert dolgozott ki IBM PC-vel kompatibilis gépekre. A rendszer településsoros formában tartja nyilván a települési környezetvédelmi elemek állapotát és a káros környezeti hatások jellemző értékeit. Célja, hogy a tervezés, a szabályozás és a szakterület központi, közép- és helyi szintű irányítása számára kellő mélységű, rendezett információt biztosítson. A több éve működő rendszer jelenleg az ország 208 városának, városi jogú nagyközségének, továbbá az országos jelentőségű üdülőterülettel rendelkező településeknek az adatait tartalmazza. Az egyes településeket 132 adat jellemzi, a következő fontosabb adatcsoportok szerint:

- a települések általános jellemzői (belterület és a beépített terület nagysága, a lakónépesség száma),
- az altalaj, a földfelszín és a talaj jellemzői,
- levegő- és vízminőség,
- települések zaj-, rezgés- és vibrációs szintje,
- településtisztaság,
- környezetvédelmi problémák.

A TKI-programcsomag támogatja az adatok tárolását, karbantartását és lekérdezését.

óráként, naptárként, határidőnaplóként, ébresztőként éppúgy használható, mint például figyelmeztetőként azok számára, akik bizonyos időnként rendszeresen gyógyszert szednek. Hasznosítható a Notaphon számítógépként, telefonkönyv vagy címtár gyanánt is. A walkman nagyságú készüléket rövidesen sorozatban gyártja a Caretec nevű osztrák cég.

## Fly, fly away...

Az OMF B/G-6-os programja mágneses adattárolók hazai fejlesztését is támogatja. Ilyen munkát folytatnak többek között az ikladi Ipari Műszergyárban is. A program keretében kialakított háromféle, FLY-350, FLY-525 és FLY-800 típusszámú motor — mint nevük is sugallja — sorrendben a 3,5, 5,25 és 8 hüvelykes átmérőjű lemezek meghajtóihoz készült. Májusban már előzetesen bejelentették az STR-525 jelzésű folyamatos mágnesszalagos tároló (streamer) motorjának az elkészültét is. Ennek külön érdeme, hogy szocialista országban még nem készült ilyen termék.

Csak reménykedni tudunk, hogy ezek a fejlesztések nem jutnak a MOM winchester-tárainak sorsára, s termékként is megismerkedhünk velük.

## ISMÉT Egy nap — sok gép

A csákvári Kossuth Művelődési Ház és a TIT Fejér megyei szervezete ismét találkozót szervez a számítógép-felhasználók és számítógépparatók részére a Csákvári Napok keretében.

Mindazokat várják, akik Commodore, Sinclair és TV-Computerekkel dolgoznak, amatőröket és profikat egyaránt.

A program keretében tág lehetőséget biztosítanak a szoftverek csereberéjére is. Akik programokat akarnak cserélni, számítógépet és monitort vigyenek magukkal, asztalt és elektromos csatlakozást térítés ellenében kaphatnak.

A találkozót a csákvári sportcsarnokban rendezik meg, 1989. augusztus 27-én 10 órától délután 4 óráig.



**PROFESSZIONÁLIS  
SZÁMÍTÓGÉPEK  
PERIFÉRIÁJA  
AJÁNLATUNKBAN**

# **KX—P 1540**

**PANASONIC MÁTRIXNYOMTATÓ  
(18 havi garanciával)**

- 24 tűs, kétirányú nyomtatású, terminálhoz is kapcsolható
- nyomtatási sebesség 240 karakter/s
- 3 alap- és 6 opcionális karakterkészlet
- grafikus üzemmódban felhasználói diagramok, grafikonok, illusztrációk
- 96 ASCII karakterkészlet hagyományos vagy dőlt betűkkel
- Centronics párhuzamos és RS232C soros illesztő
- belső 13,5 kb-ajos szabvány puffer
- 3 parancskészlet: 1. EPSON LQ—1500<sup>TM</sup>  
2. IBM PROPRINTER<sup>TM</sup>  
3. DIABLO<sup>R</sup>

Ár: 98 000,— Ft + áfa

**TOVÁBBI TERMÉKEINKRŐL  
KÉRJE ÁRJEYZÉKÜNKET!**

**SCI—L Számítástechnikai Informatikai Fejlesztő Leányvállalat  
Kereskedelmi Iroda**

**Budapest, Iskola u. 10. 1011.**

**Telefon: 154-065, 350-180/180, 181, 182, 183, 184**

**Telex: 22-4599**

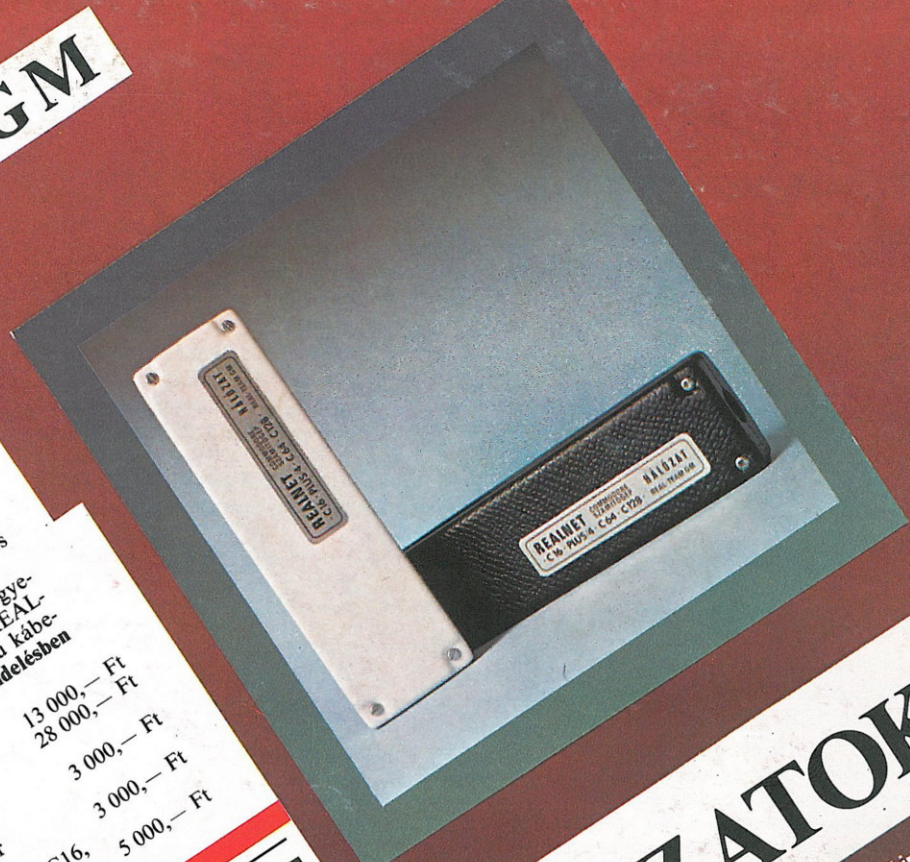
**Telefax: 35-39-15**

# REAL-TEAM GM

A REALNET-re 6 db vagy 16 db Com-  
modore gép kapcsolható. A hálózat tagjai  
lehetnek vegyesen a C16-os, +4-es,  
C64-es és C128-as gépek is. A perifériákat  
a gépek a jelenkezés sorrendjében egy-  
más után használhatják. A kiszolgálás au-  
tomatikusan történik.

- REALNET-6
- REALNET-16
- HS+4 hálózati szoftver
- +4 és C16 gépekre
- HSC64 hálózati szoftver
- C64-es gépekre
- HSCOM hálózati szoftver C16,
- +4 és C64-es gépekre

- A REALNET és a TVCNET hálózati szolgáltatásai:
- A nyomtató és a lemezegység használata
- Gyors programbetöltés
- Gyors programbetöltés tanári gépről a tanulói gépekbe



## REALNET

13 000,— Ft  
28 000,— Ft  
3 000,— Ft  
3 000,— Ft  
5 000,— Ft

## ISKOLAI HÁLÓZATOK

## TVCNET

- A tanári gépről betekintés és beavatkozás a tanulói gépekbe
- Üzenetváltási lehetőség a tanár és a tanulók, valamint tanuló és tanuló között
- Változóknak értékbás és változó olvasása
- A tanári gépről

A TVCNET 16+1, ill. 32+1 Videoton TV-Com-  
puter hálózatra kapcsolását teszi lehetővé. (A  
+1 a tanári gép. A hálózaton kevesebb gép is le-  
nyomtató- és lemezegység.)

TVCNET kábelgarnitúra: 16 db, ill. 32 db 2 mé-  
teres kábel. Ezzel a gépek egy lánc mentén 2 mé-  
terenként helyezhetők el. Ha a terem adottságai  
más során megadott teremvázlat vagy méret esetén  
a megfelelő kábelkkel szállítjuk a rendszert.

- TVCNET-16
  - TVCNET-32
- Termékeinket kulcsrakész állapotban szállítjuk. Az eszközökre 1 év garanciát vállalunk. Áraink ÁFA nélkül értendők.
- 24 000,— Ft  
32 000,— Ft



Gyártja és forgalmazza: REAL-TEAM GM  
1039 Budapest, Bálint Gy. u. 16.  
Telefon: 873-598