



mikro

számítógép

magazin

Ara: 30 Ft

FORMA-1
HUNGARORING



Raszertrúkkók C64-re
Kulccsal vagy kulcs nélkül
Operációs rendszerek háborúja

ENTERPRISE
— négy oldalon
1989/8.



Új vállalkozásba kezdett az ASZSZ Államigazgatási
Információsrendszerek Fejlesztési Irodája.
A Jólható Megyei Tanács
Oktatási és Továbbképző Központjával együttműködve
az ország egyik vonzó természeti környezetében,

Tengelicen,
létrehozták a hazai informatikai oktatás
egyik új bázisát.

Igényes ellátás, kiváló oktatási feltételek!
Közigazgatási, informatikai
számítástechnikai kurzusok!

További részleteket illetően felvilágosítással
szolgálnak:

Pöszte Lajos
ASZSZ Budapest 1119
Andor utca 47-49., 851-122/348
Horváth Magdolna
Tengelic 7054, 06-74-34033



mikro számítógép magazin

7. ÉVFOLYAM
1989/8. 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
(tanuljuk együtt)

Petróczy Judit
(könyvek)

Pinke György
(Enterprise)

Soltészné Vizi Zsuzsa
(tervezőszerkesztő)

Szebenszki Sándor

Tamásné Lakó Erika

Terebessy Ákosné

Varga János
(olvasószerkesztő)

Cimképünk:
Velekey József Lajos
munkája

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
főtítkárhelyettes

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő
hivataloknál
és a Posta Hírlap-előfizetési
és Lapellátási Irodáján
(1900 Budapest XII.,
Lehel u. 10/A)
vagy átutalással a 215-96 162
pénzforgalmi jelzőszámmal.

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 | Párbeszéd a mindennapi kultúra elméletéről és gyakorlatáról |
| 7 | Nemes Tihamér Országos Középiskolai Számítástechnikai Verseny |
| 9 | Rassterükkök |
| 11 | Feladatok — megoldások |
| 27 | Rendszerfejlesztési eszközök |
| 37 | Centronics szabványú nyomtatók illesztése Atari számítógépekhez |
| 40 | Az AmigaDOS |
| 45 | Programtermék — A nyuszi olvasni tanít |
| 46 | Adók-veszek-cserélek |

TANULJUK EGYÜTT!

- | | |
|---|----------------------------|
| 4 | A Pascal rejtelmei |
| 6 | Kulccsal vagy kulcs nélkül |

CSIPEGETŐ

- | | |
|----|--------------------------------------|
| 13 | BNV-élmény |
| 13 | Miből lesz a cserebogár? |
| 14 | Szedjük szét a Formula One-t! |
| 15 | Örökélet |
| 16 | A novoszibirszki pályázat értékelése |
| 16 | TOP-lista |

PROGRAMOZÁSTECHNIKA

- | | |
|----|--------------------------------------|
| 17 | Rezidens óra IBM AT-n |
| 20 | COM fájlok egyedi védelme |
| 22 | Programozási fogások és melléfogások |

ENTERPRISE

- | | |
|----|-----------------------------------|
| 23 | Programból alkalmazott parancsmód |
| 23 | Információk az Enterprise-piacról |
| 25 | Fej vagy írás |
| 26 | Hol az INPUT? Itt az INPUT! |
| 26 | Mi a manó? |

PÉCÉZZÜNK!

- | | |
|----|-------------------------------|
| 32 | The Norton Guides |
| 33 | Operációs rendszerek háborúja |
| 35 | TODAY: Eseménynaptár-program |

KLUB

- | | |
|----|----------------------|
| 43 | Adom a magyarázatot! |
| 43 | Ki ad magyarázatot? |

SAKK

- | | |
|----|--------------------|
| 44 | Támadás és védelem |
|----|--------------------|

KÖNYVEK — HÍREK — ÉRDEKESSEGEK

- | | |
|----|--|
| 47 | |
|----|--|

mikro számítógép
magazin



PÁR BESZÉD

a mindennapi kultúra elméletéről és gyakorlatáról

Számítógép a mindennapokban

A számítógép munkájával, e munka eredményével Magyarországon „a dolgok végső fázisában” lép érintkezésbe a lakosság döntő része, s így a számítástechnika számukra legtöbbször csak a személytelen formában megnyilvánuló — megfellebbezhetetlennek mutakozó számlák, kitöltendő nyomtatványok stb. mögött rejtőző — hivatali hatalom képzetét kelti. Felhasználóként a számítógépet a felnőtt társadalom csak elenyésző mértékben veszi igénybe. Diósi Páltól (SZÁMALK) azt is megtudhatuk, hogy egy 1986 őszén készült felmérés szerint a 14–29 év közötti fiataloknak is mindössze 14%-a jut hozzá a számítógéphez, közülük is csak 3% otthon (de ezek 70%-a is csak játékgépeket „futtat” rajtuk). A felnövekvő nemzedék tehát a számítástechnikai kultúra két területével találkozik jó esetben: a játékkal és az iskolai oktatás részévé tett programozással. Hazánkban a számítógép igen lassan válik azzá, ami valójában: eszközzé.

A pszichikai idegenkedésen túl (utalhatunk a titkárnők és gépirónk egy részére) az érdektelenség is megfigyelhető (például a városi és városkörnyéki művelődési házak igazgatóinak mintegy 5-6%-a tudja használni az ott elérhető gépeket). Az adatbázisok (könyvtári rendszerek, tájékoztató és információs szolgáltató rendszerek stb.) kezelésére a C64-szintű kapacitás már alkalmatlan, és a PC-k elterjesztése a közművelődésben és az oktatásban anyagi okokból nem olyan erőteljes, mint az előzőek esetén volt.

Amint Tóth Judittól (BME) megtudhattuk, a Magyarországon is beindult informatikusképzés diákjai jelenleg negyedévesek. A hazai bérskála ismeretében azonban nem elképzelhetetlen, hogy idegen nyelvet ismerő részük külföldi programozói segédmunkát keres majd a diploma megszerzése után, tette hozzá Striker Sándor (MTA). Nem csak a bér, az infrastruktúra is hiányzik a számítógép háttéréből: az információs hálózatok kiépítése elbukhat a hiányos és gyenge minőségű telefonvonalak és központok miatt (Simonyi Gyula, Közművelődési Információs Vállalat).

A fentiek tükrében nem különös, hogy a számítógépek használata Magyarországon elsősorban a centralizált intézményeknél (lakosság-nyilvántartás, statisztika, nyugdíjügyintézés, díjbeszedés, adóügyek, betétzámla-kezelés, stb.) és a nagyvállalatok bizonyos részénél terjedt el széleskörűen — természetesen a magával a számítástechnikával foglalkozó vállalatokon kívül —, míg a lakosság, tehát a mindennapi kultúra gyakorlati részét képező felhasználói oldal izmosodása többrendbeli nehézségbe ütközik. Nézzük meg, miért?

Számítógép és társadalom

Interjú Striker Sándorral

M.M. Hogyan került erre a szekcióvitára egy humán érdeklődésű ember?

S.S. A konferencia szervezői felismerték, hogy a számítástechnika a matematikusok szűk szakmai kompetenciáján messze túllépő kérdéseket is felvet — s így került voltképpen a "Párbeszéd a kultúráról" tanácskozas asztalára. Engem egyrészt azért hívtak meg, mert filozófiai

A debreceni Kölcsey Ferenc Megyei-Városi Művelődési Központban az idén ismét lefolytatták — a Művelődési Minisztérium és a SZOT védnöksége alatt, több szervezet támogatásával és társrendezésével — a Párbeszédkonferenciát. A korábbi évek hagyományaihoz hasonlóan a rendezvényen a kultúra elméletéről és gyakorlatáról vitázhattak a téma szakemberei és irányítói. Helyet kapott a tanácskozáson a számítástechnika is: az egyik szekcióvita a Számítógép a mindennapokban címet viselte. A következőkben megpróbáljuk röviden összefoglalni a szekcióban elhangzottakat.

katatörménymunkán során alapvetően kultúraelmélettel foglalkozom, másrészt pedig azért, mert előző munkahelyemen, az akkori Közművelődési Információs Intézetnél (ma Közművelődési Információs Vállalat) magam is részt vettem számítógéphasználattal kapcsolatos feladatok megoldásában.

M.M. Felhasználói oldalról?

S.S. Igen. Az 1980-as évek közepén, részben központi kultúrpolitikai döntések alapján, részben a növekvő magánimport révén jelentős számú C64 vagy ahhoz hasonló kapacitású kiskiszámítógép került az iskolákba, a művelődési házakba és a családokhoz. Ez, mint az a szekcióvitában is elhangzott, nagyrészt kihasználatlanul állt, főleg a közművelődési intézményekben. Hamarosan kiderült, hogy könyvtári és gazdasági célokra ezek a gépek alkalmatlanok, így ahol lehetett, programozást tanítottak rajtuk, vagy csupán játszottak velük, de gyakorta inkább biztonságosan elzárták valahová. Ugyanakkor bizonyos területeken, mint például egy programfüzet vagy egy tanfolyami segédanyag előállítására, ezek az intézmények általában külső, nyomdai segítségre szorultak. Így vált kézenfekvő feladattá egy olyan szövegszerkesztő konfiguráció kialakítása, ami elősegíti az információkibocsátást, s mégsem követel sem elérhetetlenül nagy beruházást, sem speciális szakértelmet.

M.M. És milyen munkamegosztás jöhetett létre ilyen esetben a számítástechnikus és a potenciális felhasználó között?

S.S. A számítástechnikusok elé igen egyértelmű gazdasági és emberi kívánalmak kerültek. A konfiguráció egyetlen darabja sem léphette túl az akkori pénzügyi irányítás által 50 000 forintban meghúzott, külön engedélyhez nem kötött beszerzési árat — közbevetőleg említem, hogy ekkoriban kezdtek élelmesen 49 900 forintért árulni az addig még drágább 1541-es floppykat, így ez a mai szemmel elképesztő, két oldalról is manipulált ár szerint például volt a magyarországi árképződés részint hiány-, részint rendeltettségőségének s egyben a felhasználói kiszolgáltatottságnak. De továbbmenve, az volt a cél, hogy lehetőleg a már meglévő eszközökből lehessen összeállítani az egész berendezést. A játékelemek például kazettán voltak, a legtöbb helyen csak Data-sette magnó állt rendelkezésre, a floppy árának kb. egyötzedéért. A számítástechnikusok így először alkalmassá tették az Easy Script szövegszerkesztőt a kazettás üzemmódba. Ezután a szerencsés módon éppen akkortájt beérkező elektronikus Robotron-írógéphez kellett illesztőt készíteni, s hogy az emberi oldalra visszatérjek, olyan módon átalakítani a szoftvert, hogy az ékezetes magyar betűk az írógép klaviatúrájával azonos helyre kerüljenek. Így ahol már volt C64-es és valamilyen adatrögzítő, ott egy írógép beszerzése és a megfelelő illesztés kialakítása után szinte házi nyomda — legalábbis szedő-tördelő — állt rendelkezésre.

M.M. Elterjedt ez a konfiguráció?

S.S. A Flórián téri Ifjúsági Információs Iroda kért és kapott egyet szinte azonnal, s ott azóta is működik. Kértek tőlünk néhány helyről segítséget, mi magunk akkor, egy gépirónóval s Simonyi Gyula kollégámmal hármában egy több mint 400 kézirataldalmi kötet szedését-tördelését végeztük el rajta úgy, hogy a kézirat beérkezése után hat héttel a könyv kikerült a

nyomdából. De ez már inkább a számítógép társadalomra gyakorolt hatásának kérdéskörébe tartozik, hiszen maga a könyv, a Hungaro-Polonica tanulmánygyűjtemény kitűnő szerzőgárdájának köszönhetően szép kritikát kapott, s jelentősen hozzájárult ahhoz, hogy az ottani rektor elhatározása nyomán Krakkóban az egyetemen összehasonlító lengyel—magyar tanszék jött létre. A C64—Robotron konfigurációt büszkén tüntettük fel a szerény kivitelű kötet kolofonjában. Igaz, mi magunk senkinek sem ajánljuk a napi 12-14 órás, manuális elválasztású szövegszerkesztői rohamunkát ilyen gépeken.

M.M. Lemondtak a C64 használatáról?

S.S. Nem, 10-20 oldalas munkáig kitűnően használható az Editap szoftvert továbbra is. E fölött azonban ajánlatos — ahol van, vagy néha elérhető — egy ugyancsak ügyes kábelrel az 1541-es floppyt egy IBM PC-hez kötni, s átvinni a szövegszerkesztőbe (Microsoft Word, XYwrite). Véleményem szerint nagyon fontos, hogy egy rendszer lépcsőzetesen, felfelé, lefelé is működőképes legyen, s ezt nem társadalmilag értem ezúttal, hanem gyakorlatilag. Ha van IBM PC, lehet írni azon. Ha az foglalt vagy elromlott, lehet a C64-en. Sőt, a Robotron önmagában is kitűnő elektronikus írógép.

M.M. Feltehetően javasol egy mechanikus írógépet is...

S.S. Természetesen. A magyar valóság lefelé nivellál. A számítógép csak mint számítógépes rendszer, csak mint hálózat teljesítheti ki a benne rejlő potenciált. De áramkimaradás esetén kevesebbet ér, mint egy csattogó, elaggott, háború előtti Royal.

M.M. Ennyire borulató a számítógépesítés magyarországi elterjedését illetően? Mi várható nálunk — a humán szakember szemszögéből?

S.S. A kérdés kisértetiesen emlékeztet a gőzgép és gőzvasút 1820—30 utáni elterjedésének problematikájára. Megvadták az állatok, rosszul lesznek rajta az emberek, kiszorítja a lovakat, rosszabb helyzetbe kerülnek a lótenyésztők, nem beszélve arról, hogy Isten elleni gonoszított — mondta az egyik véglet —, míg a másik amelletl kardoskodott, hogy a gőzgép omnipotens, társadalmat átalakító (igazságon átalakító) illetve a fizikai munkát megszüntető berendezés, amellyel az ember a természetet végleg enge-

delmességre kényszeríti. A valóság, a történelem valahol a kettő között adott utat a gőzgépnek — minden kultúrában oly módon, ahogy azt az adott társadalom bevezette. Volt olyan ország, ahol előbb a vasúti vállalkozók gazdagodtak meg, azután a kereskedők, a bányá- és iparmágnások. De ha Móríc Rokonokjának hinni lehet, olyan ország is akadt, ahol legelőször csak a vasutat fel nem építők gazdagodtak meg, maga a vasút pedig csak később kapott helyet a társadalmi életben. Kevésbé virágnyelven fogalmazva: Magyarországon a számítógép az ország, a társadalom habitusának megfelelően kap helyet, fenntartva természetesen azt a szerepet is, hogy megjelenése ugyanakkor bizonyos mértékig formálja is ezt a habitust, visszahat arra.

M.M. Milyen esetekre gondol?

S.S. A szekciósülés vitájának összefoglalója is jelzi, a számítógépesítés mint hálózat nálunk elsősorban a különböző központi apparátusok — díjbeszedés, lakosság-nyilvántartás stb. — szolgálatára épül ki. A lakosság ezeket felhasználóként nem veheti igénybe. De, hogy a fentiekben már említett szövegszerkesztő példát bővítsen, a mai újság-, hírlap- és könyvdömping nem kis mértékben számítógépes háttérre támaszkodik. Ez ezek a kiadványok szintén nem kis mértékben éppen a központi számítógépesítés (pl. a személyi szám) indokolatlan és esetlegesen jogszélt felhasználása ellen s ugyanakkor a társadalom által igénybe vehető számítógépes-telefonvonalas nyilvános hálózat kiépítése érdekében látnak napvilágot. Ez a megosztottság a politikai viszonyok leképezése, hiszen a számítástechnika sem élvez területenkivüliséget, s a matematikai-programozó sem gondolhatja igazán, hogy ő nem humán problémákat old meg vagy okoz munkája során.

M.M. Ön valóban ilyen személyre szóló s egyben ilyen össztársadalmi komplexumnak véli a számítógépesítést?

S.S. Vessen egy pillantást Magyarországra vasúthálózatára valamelyik vasútállomáson. Jóllehet az történelmi okokból alakult így, mégis, ha a számítógépesítést is ilyen rendezőelvek vezérik majd, akkor sem a racionális, gyors egyéni információcsere, sem az ezekből felépülő össztársadalmi nyilvánosság (demokráciát) kialakító kommunikációs s tényleges mobilításra nem lesz lehetséges.

Tamásné Lakó Erika

MUNKA KÖZBEN





A PASCAL REJTELMEI

12. Szébb képernyőt!

Színek és ablakok

Azok az olvasóink, akiknek gépéhez színes monitor csatlakozik, bizonyára nincsenek megelégedve azzal, hogy programjaik egyszínű és fantáziátlan képernyőt állítanak elő. Olyan képet szeretnének tervezni, amelyeket a professzionális alkalmazói programokban láthatnak. A Turbo Pascal számukra is sokféle lehetőséget nyújt, amelyek kihasználásával jól szerkesztett, áttekinthető és látványos kimenetet hozhatnak létre a képernyőn. A következőkben ezekkel a lehetőségekkel részletesen foglalkozunk — egyelőre félretéve a grafikai eszközöket, csupán a karakteres képernyők használatára szorítkozva.

A paraméter nélküli **textmode** az előző állapotra való visszaállást eredményezi. Az eljárás bármilyen paraméter alkalmazása esetén elvégzi a képernyő törlését is.

Mind a háttér, mind az írás színe külön beállítható a **textbackground(n)**, illetve a **textcolor(n)** eljárásokkal. Az egyes paraméterekhez és az előre definiált szövegkonstansokhoz tartozó színek a következő felsorolás tartalmazza:

n	Előre definiált konstans	Szín
0	black	fekete
1	blue	kék
2	green	zöld
3	cyan	cián
4	red	piros
5	magenta	lila
6	brown	barna
7	lightgray	világosszürke
8	darkgray	sötétszürke
9	lightblue	világoskék
10	lightgreen	világoszöld
11	lightcyan	világoscián
12	lightred	világospiros
13	lightmagenta	világoslila
14	yellow	sárga
15	white	fehér

38. ábra

12.1 Színek

A színes grafikus monitorok (CGA, EGA stb.) általában négyféle, esetleg kétféle színes szövegmódbban használhatók. A képernyő üzemmódjainak beállítására a standard **textmode** eljárás szolgál, a következő formában:

textmode(n);

ahol n 0, 1, 2 és 3 értéket vehet fel, illetve a számok helyett a Pascal számára felismerhető előre definiált szövegkonstansokat is alkalmazhatunk. Az egyes paraméterekhez tartozó változatok a következők:

n	Előre definiált konstans	Képernyőmód	Karakteres képernyőméret
0	bw40	fekete-fehér	40x25
1	c40	színes	40x25
2	bw80	fekete-fehér	80x25
3	c80	színes	80x25

```

program szinesszoveg;
var h,i,j:integer;ch:char;
begin
  (képernyőmód 0-tól 3-ig)
  for h:=0 to 3 do
    begin
      textmode(h);
      for i:=0 to 7 do
        begin
          clrscr;
          for j:=0 to 15 do
            begin
              textbackground(i);
              textcolor(j);
              gotoxy(i,j+1);write(i:2,',',j:2);
            end;
          (villogó karakterek)
          textcolor(j+16);
          gotoxy(i0,j+1);write(i:2,',',j:2);
          end;
          textbackground(black);
          textcolor(white);
          gotoxy(10,23);
          write('Folytatás: bármely billentyű');
        (felfényes kijelzés)
          lowvideo;
          gotoxy(10,24);
          write('Folytatás: bármely billentyű');
          read(kbd,ch)
        end
      end
    end..

```




12.2 Ablakok

A képernyőn megjelenő útmutatók és a kimenet könnyebben kezelhető, érthetőbb és esztétikusabb kivitelének a színek mellett másik gyakran alkalmazott eszköze a képernyőablak használat. A karakteres képernyőablakot a

window (x1,y1,x2,y2):

mezőben a háttérszín száma és tőle vesszővel elválasztva az írás színének száma látható. A képernyő bal oldalán a mező nem villogó, tőle jobbra ugyanaz villogó kijelzéssel jelenik meg.

A következő képernyőre való áttérés egy billentyű lenyomásával kezdeményezhető. Az erre utaló üzenet normál és félfényes kijelzéssel is elolvasható a képernyőn.

40. ábra

```

program szinesablakok;
label ide;
var i,j:integer;
    ch:char;
procedure keret (x1,y1,x2,y2,szin:integer);
begin
    textcolor(szin);
    (a keret négy sarka)
    gotoxy(x1-1,y1-1);write(chr(218));
    gotoxy(x2+1,y1-1);write(chr(191));
    gotoxy(x1-1,y2+1);write(chr(192));
    gotoxy(x2+1,y2+1);write(chr(217));
    for i:=x1 to x2 do
    begin
        gotoxy(i,y1-1);
        write(chr(196));
        gotoxy(i,y2+1);
        write(chr(196));
    end;
    for i:=y1 to y2 do
    begin
        gotoxy(x1-1,i);
        write(chr(179));
        gotoxy(x2+1,i);
        write(chr(179));
    end;
end;
procedure ablakbair (szin:integer);
begin
    (az ablakba irt szöveg színe)
    textcolor(szin);
    for i:=1 to 8 do
        for j:=65 to 95 do
            write(chr(j));
        read(kbd,ch);
    end;
begin
    ide:
    window(1,1,80,25);
    clrscr;
    gotoxy(5,25);
    textcolor(white);
    write('Folytatás: bármely billentyű');
    (az első ablak)
    keret(10,5,40,13,green);
    window(10,5,40,13);
    clrscr;
    ablakbair(lightmagenta);
    window(1,1,80,25);
    (a második ablak)
    keret(25,10,55,18,red);
    window(25,10,55,18);
    clrscr;
    ablakbair(yellow);
    window(1,1,80,25);
    (a harmadik ablak)
    keret(40,15,70,23,lightmagenta);
    window(40,15,70,23);
    clrscr;
    ablakbair(cyan);
    window(1,1,80,25);
    gotoxy(45,25);
    textcolor(white);
    write('Vege: ESC');
    read(kbd,ch);
    if ch=#27 then halt (a program vége)
    else goto ide
end.

```

39. ábra

```

program ablak;
label ide;
var i,x1,x2,y1,y2:integer;
    szin,tinta:integer;
    ch:char;
procedure keret (szin:integer);
begin
    (a keret színe)
    textcolor(szin);
    (a megadott ablakot kívülről bekeretezi)
    (a keret négy sarka)
    gotoxy(x1-1,y1-1);write(chr(218));
    gotoxy(x2+1,y1-1);write(chr(191));
    gotoxy(x1-1,y2+1);write(chr(192));
    gotoxy(x2+1,y2+1);write(chr(217));
    (felső és alsó keretvonal)
    for i:=x1 to x2 do
    begin
        gotoxy(i,y1-1);
        write(chr(196));
        gotoxy(i,y2+1);
        write(chr(196));
    end;
    (bal- és jobboldali keretvonal)
    for i:=y1 to y2 do
    begin
        gotoxy(x1-1,i);
        write(chr(179));
        gotoxy(x2+1,i);
        write(chr(179));
    end;
end;
begin
    ide:
    (ablak a teljes képernyő)
    window(1,1,80,25);
    clrscr;
    writeln('Ablak koordináták:');
    write('x1=');readln(x1);
    write('y1=');readln(y1);
    write('x2=');readln(x2);
    write('y2=');readln(y2);
    write('a keret színe:');readln(szin);
    write('az ablakba irt szöveg színe:');readln(tinta);
    clrscr;
    keret(szin);
    (az ablak definíciója)
    window(x1,y1,x2,y2);
    textcolor(tinta);
    (az ablak bal felső sarkára állítja a kurzort)
    gotoxy(1,1);
    repeat
        writeln('Folytatás: bármely billentyű, új ablak: ESC');
        read(kbd,ch);
    until ch=#27;
    (visszaállítja az eredeti írás szint)
    textcolor(yellow);
    goto ide;
end.

```




Kulccsal vagy kulcs nélkül

A szoftver a számítás- és számítógéptech-nika tagadhatatlanul legnagyobb értéke. Különösen igaz ez a HC-PC kategóriára, ahol a relatív értéket kifejező hányados (szoftver/hardver) több száz, sőt több ezer-szeres is lehet. A szoftverállomány tehát mindenképpen sokoldalú védelemre szorul. Ez a védelem többféle — általában egymással kombinált — módon valósítható meg.

Az egyik, mindenképpen szükséges megoldás a teljes szoftverállomány hajlékony-lemezen vagy magnésszalagon történő tárolása, esetleg még külön egy biztonsági másolatot, például BACKUP formában is. A másik, amely nem védi ugyan meg a merevlemez tartalmát, de a személyes felelősség kérdését tisztázza a szándékos vagy véletlen „rongálások” esetén.

A mellékelt programmal ez utóbbi problémának a megközelítést tűztém ki célul. A megoldás igen egyszerű. A PC indítását végző rendszerlemezen, vagy ha a gép a merevlemezről indul, azon (illetve mindkettőn) egy „kulcsszó”-programot kell aktivizálni. A programra újrást az AUTOEXEC.BAT fájlba kell beírni. A rendszerinduláskor tehát a kulcsszóprogram automatikusan elindul, és a gép használatát nem engedélyezi mindaddig, amíg a kezelő a megfelelő kulcsszót meg nem adja.

E megoldás alkalmazása különösen iskolákban előnyös akkor, ha a gépteremben sok gép van, és azokat sokan, rendszeresen és felügyelet nélkül használhatják. Bár a program könnyen „feltörhető” — még akkor is, ha a gépet használók ezen a téren bizonyított képességeit most figyelmen kívül hagyjuk —, de a kulcsszó és a géphasználat illetéktelennek történő átadása egy megfelelően szabályozott környezetben (minden géphasználó csak a neki engedélyezett gépen dolgozhat) senkinek nem lehet érdeke.

Az 1. listán a Turbo Pascal 3.01 verzióban megírt kulcsszóprogram látható. A program színes monitorhoz készült, ennek hiányában a színekre vonatkozó részek kihagyhatók belőle. A kulcsszó megadása a program végén lévő

Password(„kulcsszó”) eljárásnévvel lehetséges.

A programot a Pascal fordítóval „.COM” típusú fájlba fordítjuk, és a rendszerindítást végző lemezen helyezzük el. Ha a lemez könyvtárszervezésű, ügyelni kell arra, hogy az AUTOEXEC.BAT a programhoz hozzáférhessen.

A 2. lista az aktivizáláshoz szükséges AUTOEXEC.BAT program egy lehetséges megoldását tartalmazza.

A program elkészítésével járó munka bőven megtérül, mert használatával jóval szervezettebbé lehet tenni a gépteremben folyó munkát, és nagy részben meg lehet takarítani azt a fáradságot, amelyet egy „feltűrt” merevlemez tartalmának a helyreállítása jelent — a vírusfertőzések lokalizálásáról most nem is beszélve.

Igaz Gábor

eljárással nyithatunk, ahol a zárójelben megadott adatok (ezeknek **integer**-eknek kell lenniük) az ablak bal felső és jobb alsó sarkát összekötő átló végpontjai. Az ablak két meg nem adott sarkának koordinátái értelemszerűen

bal alsó: x1, y2
jobb felső: x2, y1.

Az x1, y1, x2, y2 adatok a karakteres képernyő karakterhelyeire vonatkoznak, így a következő feltételeknek eleget kell tenniük:

0 < x < 81
0 < y < 26,
továbbá a zárójelben megadott koordinátákra fenn kell állnia az
x2 > x1 és az
y2 > y1

relációnak is. Meg kell jegyezni, hogy a 80-as karakterhely és a 25 sor használatra az ilyenkor bekövetkező soremelések miatt az ablakszerkezet elrontja.

A definiált ablak eszentül úgy használható, mint a képernyő, azaz mind a **clscr**, mind a **gotoxy** eljárások csak az ablakra vonatkoznak. Az ablakdefiniáció mindaddig érvényes, amíg új ablakot nem definiálunk. Ha vissza akarunk térni a teljes képernyő használatára, ez is csak ablakdefiniációval oldható meg:

window(1,1,80,25);
ami a maximális méretű ablak, azaz értelemszerűen a teljes képernyő megnyitását eredményezi.

A 39. ábrán egy egyszerű program listája látható, amellyel különböző ablakokat hozhatunk létre a képernyőn. Az eddig elmondottakkal látszólag ellentmondásban van a program 11-es sorában lévő információ (ez egyébként a futás közben a képernyőn meg is jelenik); ennek a korlátozásnak az a magyarázata, hogy a program a definiált ablakot kívülől be is keretezi, így az x1=1 és x2=80, illetve az y1=1 és y2=25 koordinátákat nem használja. A képernyőn megjelenő ablak keretét a program a megadott színnel (szin) rajzolja, az ablakba írt szöveg szintén a bemenetben megadott színnel (tinta) jelenik meg.

A 40. ábra már egy látványosabb programot mutat be, amely három, egymást részben átfedő ablakot nyit, ezek mindegyike más-más keret- és írásszínrel használ. Szokásunk szerint egy, a témához nem szorosan kapcsolódó újdonság is található a programban, éspedig a **halt** eljárás, amely a program leállítására alkalmazható (lásd a program hátulról számított harmadik sorát).

Nagy Imre

```

program Kulccsal;
[MC]
[
  Kulcsszó elnevezés program.
  Hivatás Password('Kulcsszó');
  Visszatérés csak akkor ha a
  kulcsszó egynél.
  A változókbeállításban a
  Password = String(80)-nak
  szerepelnie kell!
  Készítette: Igaz Gábor 1989.11.1.
]
]

```

```

Type Password = String(80); { ne felejtsetd el! }
Procedure Password(Kulcsszo: Password);
Var Jelzobyte: Boolean;

```

```

Procedure Karet(FelsoBalX,FelsoBalY,
  AlsóJobbX,AlsóJobbY: Integer);
Var I: Integer;
Begin
  Gotoxy(FelsoBalX,FelsoBalY);
  Write('!!!');
  For I:=FelsoBalX+1 To AlsóJobbX-1
  Do Write(' ');
  For I:=FelsoBalY+1 To AlsóJobbY-1
  Begin
    Gotoxy(FelsoBalX,I);
    Write('*');
  End;
  Gotoxy(AlsóJobbX,I);
  Write('*');
End;
Gotoxy(FelsoBalX,AlsóJobbY);
Write('!!!');
For I:=FelsoBalX+1 To AlsóJobbX-1
Do Write(' ');
End; {Karet}

```

```

Procedure Init(Passi:Password);
Var I,J : Integer;
Begin
  TextColor(LightGreen);
  ClrScr;
  I:=80-I*Length(Passi) Div 2;
  J:=26-2-2*Length(Passi)+2,12,12);
  Gotoxy(I,J);
  TextColor(LightRed);
  Write('Password: ');
  TextColor(Cyan);
  For I:=1 To Length(Passi) Do
    Gotoxy(I,J,18);
  End; {Init}

```

```

Function OlvPass(Passi:Password): Boolean;
Var I : Integer;
Ch : Char;
Olvasott: Password;
Begin
  Olvasott:=Pass;
  For I:=1 To Length(Passi) Do
  Begin
    Read(Ch,Ch);
    If Ch = #27 Then
      Begin
        Jelzobyte:=False;
        OlvPass:=False;
        Exit;
      End;
    If Ch <> #27 Then
      Begin
        TextColor(Cyan);
        Write('*');
        Olvasott:=Olvasott+Ch;
      End;
    End;
  Jelzobyte:=True;
  OlvPass:=Olvasott+Pass;
  End; {OlvPass}

```

```

Begin {Password}
  Jelzobyte:=False;
  Repeat
    Init(Kulcsszo);
    If Jelzobyte = True Then Write(Ch('???'));
  Until OlvPass(Kulcsszo);
  TextColor(LightGray);
  End; {Password}

```

```

Begin {Clscr}
  Password('*#16#');
  ClrScr;
  End.

```

1. lista

2. lista

```

mch:=#14;
password:=string(80);
repeat
  gotoxy(1,1);
  password('*#16#');
  clrscr;
  ch:=

```


Országos Középiskolai Számítástechnikai Verseny

2. forduló



Készíts programot egy digitális áramkör működésének szimulálására! Az áramkör működésében feltételezzük, hogy minden alaelem kimenetén egységnyi idő múlva jelenik meg a bemeneti jelkombináció hatása, és ez azonnal megjelenik a következő alaelemek bemenetén is. (Azaz az információ a vezetékeken \emptyset , a kapukon egységnyi idő alatt halad át.)

Az áramkör alaelemei a következő kapuk lehetnek: INVERTER (NOT), AND, OR, NAND, NOR, EXOR kapuk. (Az inverternek 1, a többi kapunak pedig 2 bemenete van.) Az áramkörben használható kapuk száma maximum 10 lehet. Kezdetben minden kapu minden bemenetén valamilyen határozott érték (például \emptyset) van.

Az áramkör leírását úgy adjuk meg, hogy az alaelemek bemeneteihez, illetve kimenetéhez rendelünk egy csomópont-azonosító számot, s az azonos csomópontokhoz tartozó be-, illetve kimeneteket tekintjük összekötöttnek (lásd az ábrát).

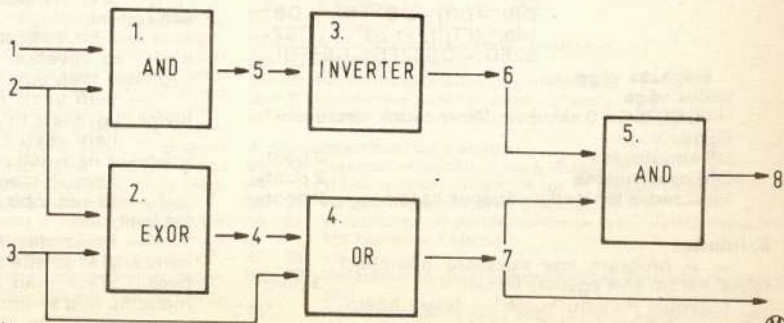
A szimulálandó hálózatról, áramkörörről tegyük fel, hogy véges időn belül kialakul valamilyen stabil kimeneti jelkombináció (például ha N kapu van az áramkörben, akkor ez a korlát legyen N^2 !)

Az egyes kapuk működésének leírása:

<table border="1"> <tr><td>INV</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>1</td><td>\emptyset</td></tr> </table>	INV	\emptyset	1	\emptyset	1	\emptyset	<table border="1"> <tr><td>OR</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>\emptyset</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	OR	\emptyset	1	\emptyset	\emptyset	1	1	1	1	<table border="1"> <tr><td>AND</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>\emptyset</td><td>\emptyset</td></tr> <tr><td>1</td><td>\emptyset</td><td>1</td></tr> </table>	AND	\emptyset	1	\emptyset	\emptyset	\emptyset	1	\emptyset	1			
INV	\emptyset	1																											
\emptyset	1	\emptyset																											
OR	\emptyset	1																											
\emptyset	\emptyset	1																											
1	1	1																											
AND	\emptyset	1																											
\emptyset	\emptyset	\emptyset																											
1	\emptyset	1																											
<table border="1"> <tr><td>EXOR</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>\emptyset</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>\emptyset</td></tr> </table>	EXOR	\emptyset	1	\emptyset	\emptyset	1	1	1	\emptyset	<table border="1"> <tr><td>NOR</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>1</td><td>\emptyset</td></tr> <tr><td>1</td><td>\emptyset</td><td>\emptyset</td></tr> </table>	NOR	\emptyset	1	\emptyset	1	\emptyset	1	\emptyset	\emptyset	<table border="1"> <tr><td>NAND</td><td>\emptyset</td><td>1</td></tr> <tr><td>\emptyset</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>\emptyset</td></tr> </table>	NAND	\emptyset	1	\emptyset	1	1	1	1	\emptyset
EXOR	\emptyset	1																											
\emptyset	\emptyset	1																											
1	1	\emptyset																											
NOR	\emptyset	1																											
\emptyset	1	\emptyset																											
1	\emptyset	\emptyset																											
NAND	\emptyset	1																											
\emptyset	1	1																											
1	1	\emptyset																											

Példa (programod kipróbálásához célszerű ennél egyszerűbb áramköröket választanod):

Az áramkör rajza:



Leírása:

- 1. AND: 1,2,5
- 2. EXOR: 2,3,4
- 3. INVERTER: 5,6
- 4. OR: 4,3,7
- 5. AND: 6,7,8

Az áramkör bemenetei: 1,2,3
Az áramkör kimenetei: 7,8

Bemeneti kombinációk	Kimenetek
1,1,1	1, \emptyset
\emptyset , \emptyset , \emptyset	\emptyset , \emptyset
\emptyset , \emptyset , 1	1, 1

FELADATOK

- A: Add meg, hogyan ábrázolod a memóriában az áramkört!
- B: Olvasd be az áramkör leírását tetszőleges, általad választott módon, s ebből építsd fel az áramkört az általad megválasztott adatszerkezet alapján! (Természetesen nem szükséges kirajzolni a képernyőre az áramkört!)
- C: Olvasd be bemeneti jelkombinációkat és az áramkör működésének szimulálásával határozd meg az ezekre adott kimenetet! (A kimenetet akkor tekintjük késznek, amikor az áramkör minden alaelemének kimenete változatlan marad.)
- D: Állítsd elő automatikusan az összes bemeneti jelkombinációt, és az áramkör működésének szimulálásával határozd meg az ezekre adott kimenetet!
- E: Ellenőrizd, hogy az áramkör leírását helyesen adták-e meg! A lehetséges hibák:
 - egy kapu kimenetét sehova sem kötötték, és az nem is az áramkör kimenete,
 - egy kapu bemenetét sehova sem kötötték, és az nem is az áramkör bemenete,
 - két (vagy több) kapu kimenetét összekötötték,
 - van olyan kapu a hálózatban, amelyet a bemenetről nem lehet elérni,
 - az áramkör valamelyik kimenete nem valamelyik kapu kimenete,
 - az áramkör valamelyik bemenetét nem kötötték egy kapu bemenetére sem,
 - a korlátnak állított időegységszám után sem alakul ki stabil kimenete az áramkörnek.

A verseny első fordulójának

1. feladat

Az eredmény 64, mivel a 20-as és az 50-es sort egyszer, a 60-ast kétszer, a 70-est négyszer, ... hajtja végre a program.

Jó eredmény: 1 pont

Magyarázat: 2 pont

2. feladat

- 1030 DO WHILE J >= 1 AND IS\$ < T\$(J) 1 pont
 1040 T\$(J+1) = T\$(J): J = J - 1 1 pont
 1060 T\$(J+1) = IS\$ 1 pont
 1. → az I-1. elemig a T\$() vektor rendezett 1 pont
 és IS\$ a beszúrandó 1 pont
 2. → a J+1...I intervallumban a T\$() vektor rendezett, 1 pont
 IS\$ helye az 1...J indexű elemek valamelyike 1 pont
 3. → az 1...I intervallumban rendezve van a T\$() vektor 1 pont

3. feladat

- az X tizedes törtet kettedes törtté (azaz kettes számrendszerbeli törtté) alakítja 1 pont
 — befejeződik, ha már előállítottunk 30 számjegyet, 1 pont
 vagy ha a végtelen szakaszos kettedes tört első ismétlődő szakaszának vége van, 2 pont
 vagy ha nulla maradékot kapunk valamelyik lépésben, azaz az utolsó értékes jegynél 1 pont
 — $x = 0,3$ -ra az eredmény 0,01001 1 pont

4. feladat

- K1\$ kulcs szerinti helyére igyekszik tenni a (K1\$,K2\$) sort a táblázatban 2 pont
 — ha az így kiszámított hely szabad, akkor oda teszi, 1 pont
 — ha nem, akkor megkeresi a mögötte levő legközelebbi üres helyet és oda teszi 2 pont
 — ha nincs üres hely, akkor kiírja, hogy "baj!!!!" 1 pont

5. feladat

A D() vektor tartalmazza az egyes érmék darabszámát. Pénzvisszaadás(ÖSSZEG):

Ciklus I = 1-től 5-ig

Ha ÖSSZEG >= FT(I) akkor DB := INT(ÖSSZEG/FT(I)) ha D(I) < DB akkor DB := D(I)
 D(I) := D(I) - DB Ki: DB;
 "db."; (FT(I); "Ft-os" ÖSSZEG := ÖSSZEG - DB * FT(I)

Elágazás vége

Ciklus vége

Ha ÖSSZEG > 0 akkor Ki: "Nem tudok visszaadni"

Eljárás vége.

DB korlátozása 2 pont

D(I) csökkentése 2 pont

Visszaadás lehetetlenségének kezelése 2 pont

6. feladat

- A program egy egyszerű hőembert rajzol, három kört egymás tetején 3 pont
 (egymás melletti körök — fekvő hőember — csak 2 pontot ér)
 — A felső két kör sugara mindig kétharmada az alatta levő kör sugarának 1 pont
 — A legelső kör sugara 30 egység 1 pont

megfejtései



- a GVONAL eljárás egy félkörívet körélez szabályos sokszöggel 1 pont
 — Az :Y paraméter a körív sugarát adja meg 1 pont

7. feladat

- A G1 0-ba ragadásának felismeréséhez $A=0$, $B=1$, $C=1$ szükséges, így a 100-as sor első száma: 6 1 pont
 Ha a kapu jól működik, akkor $Y=0$, egyébként $Y=1$, tehát a második szám: 0 1 pont
 A másik két sor hasonlóan:
 110-es sor: 1,0 vagy 3,0 vagy 5,0 a jó számpár 2 pont
 120-as sor: 7,1 a jó számpár 2 pont
 Indoklás: a tesztelt kapu kimenetének hibája akkor deríthető ki, ha a kapu vezérlése olyan, hogy a kimeneten 1-nek kellene lennie és ez az 1-es ki tud jutni az áramkör Y kimenetére 1 pont
 (Például a G1 kimenete akkor 1, ha $A=0$ és B,C mindegyike 1, mivel ekkor G1 második bemenete 0 lesz. A jel akkor jut ki G3 kimenetére — ami az áramkör kimenete —, ha G3 másik bemenete 0, ez itt teljesül. A hiba tehát kideríthető a fenti vezérléssel, mert G1 hibája esetén 0 helyett 1 jelenik meg az áramkör kimenetén. Itt természetesen kihasználjuk azt, hogy az áramkörben egyszerre csak egy hibás kapu lehet.)

8. feladat

- Problémánként 1 pont (csak algoritmikus problémákra adható), néhányat felsorolunk:
 — a ki-, ill. beszállást 10 másodpercen belül be kell fejezni,
 — a lift, miközben egy szinten áthalad, egyetlen hívást és egyetlen parancsot képes fogadni (hiába nyomnak meg egyszerre több gombot),
 — nem veszi észre, hogy az adott szintre már hívták (ugyanazt többször is megnyomhatják),
 — nem veszi észre, hogy a többszörösen hívtak szintekre nem kell még egyszer elmennie,
 — nem áll meg közbülső szinteken,
 — 32-nél több hívás esetén indexhatár-túllépéssel leáll,
 — ha az utasok utolsó, liftben maradt csoportja nem száll ki az általuk megjelölt szinten, akkor az ajtó becsukódása után mindaddig kénytelenek a liftben maradni, míg kívülről valaki nem hívja a liftet,
 — beszállás után, amennyiben a liftnek nincs több tárolt uticélja, az utasoknak a következő lifthívást meg kell várniuk, mert a lift nem figyeli a PARANCS gombot.

Hogyan és mire használjuk a rasztert a C64-en?

Nilván sokaknak tetszettek a különféle játékprogramok, bejelentkezések látványos vizuális effektusai, nyolcnál több sprite egyidejű megjelenítése a képernyőn, sprite-ok láthatósága a képernyő keretén is, a képernyő egyes területeinek különböző üzemmódbeli használata... Többekben felmerülhetett az a kérdés, hogy mi módon lehet ezeket az effektusokat előállítani.

Az ilyen és ehhez hasonló képernyőkezeléseket vagy a VIC (videovezérlő chip) „becsapását” a raszter használata nélkül nem lehet megoldani.

Itt merül fel az a kérdés, hogy mi a raszter? A képernyőn megjelenő kép ciklikus frissítése a VIC feladata. A kép frissítése soronként történik. Egy kilenc bites számláló, a raszterszámláló mutatja meg, hogy melyik pixelsorban tart éppen. Például egy karakter sor nyolc pixelsorból áll (gondoljunk a nagy felbontású grafikára), tehát a számláló értéke is nyolccal fog nőni, ha egy karakter sor frissítése megtörtént.

A számláló segítségével mindig pontosan a számunkra megfelelő időben hajthatjuk végre a módosításokat a képernyőn. A pontos időzítésre a zene megszüntetésénél is felhasználhatjuk. Így készülnek azok a töltőprogramok, melyek a program töltése közben zenélnék, scrolloznak, és még gyorsabbak is az eredeti töltési sebességnél.

A VIC következő regisztereit kell ismernünk a raszter használatához:

- \$D011 (53265) 7. bitje a raszterszámláló 8. bitje
- \$D012 (53266) a raszterszámláló 0–7. bitjei
- \$D019 (53273) 0. bitje jelzi (=1), hogy rasztermegszakítás történt
- \$D01A (53274) 0. bitje engedélyezi (=1) a raszter megszakítását

Ezekkel a regiszterekkel használható a raszter, de nem árt, ha minden felhasználó felrészíti a C64 megszakítási rendszerére, illetve a VIC többi regiszterére vonatkozó ismereteit, ugyanis ezeket sokszor kell használni.

Most, hogy a megfelelő regisztereket már ismerjük, nézzük meg, mit kell tennünk konkrétan a raszter kezeléséhez. Az egyik lehetőség az, hogy állandóan figyeljük a raszterszámlálót (\$D011/\$D012), és amint a számunkra megfelelő értéket mutatja, azaz azt a sort frissíti a VIC, ahol változtatni akarunk a képernyőn, végrehajtjuk a változtatást, majd újra várunk a következő egyezésig, változtatunk, majd újra várunk...

Ez bizony nem a műveleti idő legjobb kihasználása, főleg ha közben kismillió dolgot kellene csinálni. Nézzünk egy jobb módszert! Irjuk át a megszakításokat (IRQ) egy saját rutinra. Ezt a \$0314/\$0315 címeken található mutató átirásával tehetjük meg (ha a KERNAL ROM-ot kikapcsoltuk, akkor a \$FFFE, \$FFFF címeken). Saját megszakításrutinunkban figyeljük, hogy történt-e rasztermegszakítás, ilyenkor ugyanis \$D019 0. bitjének értéke 1. Ehhez még tudnunk kell azt, hogy ha egy általunk kiválasztott raszterértéket beleurunk a számlálóba, és engedélyezzük a rasztermegszakítást, akkor a rasztermegszakítás megtörténik, amint a számláló eléri a beleirt értéket. Természetesen minden megszakítás után újra be kell állítani a számláló értékét, és törölni kell a megszakításjelző bitet. Ha egymáshoz közeli értékeknek kell módosítanunk a képernyő állapotát, akkor célszerű a két módszert együtt alkalmazni. Arra azonban ügyeljünk, hogy megszakításrutinunk a következő megszakítás előtt véget érjen.

Első példaprogramunkban (1. lista) a rasztermegszakítás segítségével a képernyő színét rasztersoronként változtatjuk meg. A módszerrel látványos érdekességeket és érdekes látványosságokat lehet megjeleníteni. Bármely képernyőn megjelenő alakzatot rasztersoronként lehet színezni, azaz egy karakter vagy egy sprite akár 16 színű is lehet.

A megszakítási rendszer inicializálása

- 9—11 A megszakításvektor módosítása
- 14—20 A munkaváltozók és a munkaterület inicializálása

- 22—25 Rasztermegszakítás helyének beállítása
- 26—28 CIA megszakítások tiltása (Timer, SDR, FLAG)
- 29—30 Rasztermegszakítás engedélyezése

A megszakításrutin kezdete

- 34 Megszakításjelző törlése
- 35—44 A pontos időzítés miatti utasítások

Színbeállító rész

- 45 8 raszter sor egy karakter sor
- 46—49 A színek beállítása a STACK táblázatból, színmutató növelése
- 50—51 Karakter sor vége?
- 52—53 Cím táblázat vége?
- 54 Időzítőrutin meghívása
- 55—56 A színek beállításának folytatása

Időzítőrutin

- 58—71 A színek megváltoztatásának pontos hozzáigazítása a rasztersorokhoz

A színbeállító rész vége

- 73—75 Időzítés
- 76—77 Képernyő- és keretszín fekete

Effektusgenerátor

- 79—110 A szint táblázat módosításával dinamikus képernyő előállítás

A megszakításrutin vége

- 112—113 A rasztermegszakítás helyének beállítása
- 114 Az eredeti ROM rutin vége (kilépés a megszakításból)

Táblázatok

- 116—140 Adattáblázat (a színek kódszámát tartalmazza)
- 142—143 Munkaváltozók
- 145 Szint táblázat (az aktuális színkódokat tartalmazza)

A 2. lista programja a raszter segítségével kiterjeszti a sprite-teret a keret alsó és felső részére is, úgy, hogy 8 sprite-ot használhatunk a keret felső részén és a képernyőn, és nyolcat a keret alsó részén. A VIC chipet a 24/25 soros üzemmód megfelelő idejű átváltásával lehet becsapni. A sprite-teret ki lehet terjeszteni a keret jobb és bal szélére is, de ez egy jóval bonyolultabb programot igényel.

A megszakítási rendszer inicializálása

- 11—15 A 11. sprite-lap feltöltése
- 17—20 A megszakítás átirása a saját rutinra
- 21—25 A CIA megszakítások tiltása (Timer, SDR, FLAG)
- 26 A keretre kerülő információ
- 27 Keretszín
- 28—31 A megszakítás helyének beállítása
- 32—33 Képernyőalapszín
- 34—35 Rasztermegszakítás engedélyezése
- 36—37 Rasztermegszakítás-jelző törlése
- 38—39 Munkaváltozó-inicializálás

A megszakításrutin kezdete

- 45—46 Rasztermegszakítás-jelző törlése
- 48 Ha nem raszter miatt, akkor az eredeti rutin
- 50—55 Rasztermező-mutató növelése
- 56—61 A rasztermező paraméterezése, a következő megszakítás helyének beírása
- 62—64 Időzítés
- 65—101 Rasztermező paraméterezése (nagyítás, szín, koordináták)
- 103 Az eredeti ROM rutin vége (kilépés a megszakításból)

Paramétertábla

- 105 Megszakítások helye
- 106 Képernyőüzemmódok

- 107 Sprite-ok engedélyezése
- 108-109 Nagyítás
- 110 Multicolor üzemmód
- 111 X koordináta MSB
- 112-113 Színek (felső terület)
- 114-115 Színek (alsó terület)
- 116-119 Koordináták (felső terület)
- 120-123 Koordináták (alsó terület)
- 124-125 Sprite-mutatók (felső terület)
- 126-127 Sprite-mutatók (alsó terület)

Programjaink elsősorban ö. tételbészítő céllal készültek; javasoljuk a paramétertáblák átírását, a programok testre szabását.

Bakos Imre—Kelemen Róbert

1. lista

```

0000          ** *C000
0001  ;          .OBJ M
0002          *****
0003          * KEPERNYOSZINEZES *
0004          *
0005          *
0006          * RASZTERSORONKENT *
0007          *****
0008          S          SEI
0009          LDA          #<IRQ
0010          STA          #0314
0011          LDA          #IRQ
0012          STA          #0315
0013  ;
0014          LDY          #5F
0015          LDA          #0
0016          STA          COUNT
0017          STA          DIRECT
0018  LP          STA          STACK,X
0019          DEX
0020          BPL          LP
0021  ;
0022          LDA          #62
0023          STA          #D012
0024          LDA          #1B
0025          STA          #D011
0026          LDA          #7F
0027          STA          #DC0D
0028          STA          #D00D
0029          LDA          #1
0030          STA          #D01A
0031          CLI
0032          JMP          STP
0033
0034  IRQ          INC          #D019
0035          NOP
0036          NOP
0037          NOP
0038          NOP
0039          NOP
0040
0041          LDY          #3
0042          DEX
0043          BNE          W1
0044          NOP
0045          NS
0046          LDA          STACK,X
0047          STA          #D020
0048          STA          #D021
0049          INX
0050          DEY
0051          BEQ          NS
0052          CPX          #5F
0053          BCS          EXTRAS
0054          JSR          WAITRAS
0055          LDA          STACK,X
0056          BCC          STORCOL
0057  ;
0058          WAITRAS          NOP
0059          NOP
0060          NOP
0061          NOP
0062          NOP
0063          NOP
0064          NOP
0065          NOP
0066          NOP
0067          HOF
0068          NOP
0069          NOP
0070          NOP
0071          RTS
0072  ;
0073          EXTRAS          LDY          #5
0074          W2          DEY
0075          BNE          W2
0076          STY          #D020
0077          STY          #D021
0078  ;
0079          LDY          COUNT
0080          BEQ          SET
0081  ;
0082          CPX          #44
0083          BEQ          CLEAR
0084          LDA          DIRECT
0085          BEQ          SET
0086          CLEAR          LDA          #1

```

```

0007          STA          DIRECT
0008          LDA          #44
0009          SEC
0010          SBC          COUNT
0011          TAX
0012          LDA          #0
0013          STA          STACK,X
0014          LDY          COUNT
0015          STA          STACK+45,X
0016          SET          JMP          EX
0017          LDA          #0
0018          STA          DIRECT
0019          LDA          #44
0020          SEC
0021          SBC          COUNT
0022          TAX
0023          LDA          COLS,X
0024          STA          STACK,X
0025          LDY          COUNT
0026          LDA          COLS+45,X
0027          STA          STACK+45,X
0028          INX
0029          JMP          COUNT
0030          EX          STA          COUNT
0031          ;
0032          COLS          .BYTE 11,11,12,12
0033          .BYTE 12,11,12,12
0034          .BYTE 15,07,01,07
0035          .BYTE 15,12
0036          .BYTE 11,11,12,12
0037          .BYTE 10,10,08,08
0038          .BYTE 02,02,09,09
0039          .BYTE 02,02,08,08
0040          .BYTE 10,10,07,07
0041          .BYTE 15,15,12,12
0042          .BYTE 11,11,00,00
0043          .BYTE 15,15,07,07
0044          .BYTE 13,13,03,03
0045          .BYTE 05,05,14,14
0046          .BYTE 06,00,14,14
0047          .BYTE 05,05,03,03
0048          .BYTE 13,13,07,07
0049          .BYTE 15,15,12,12
0050          .BYTE 11,11,12,15
0051          .BYTE 07,01,07,15
0052          .BYTE 15,12,11,12
0053          .BYTE 00,00,00,00
0054          ;
0055          COUNT          .BYTE 0
0056          DIRECT          .BYTE 0
0057          ;
0058          STACK          ** *
0059          .END

```

2. lista

```

0000          ** *C000
0001  ;          .OBJ M
0002          *****
0003          * SPRITE MEZO *
0004          * KERET ALSO ES *
0005          * FELSO RESZEN *
0006          *****
0007          *
0008          S          SEI
0009          ;
0010          ;
0011          ;
0012          ;
0013          LDY          #2
0014          LDA          #4FF
0015          STA          704,Y
0016          DEY
0017          BPL          LP
0018          LDA          #IRQ
0019          STA          #0314

```

```

0019          LDA          #IRQ
0020          STA          #0315
0021          LDA          #7F
0022          STA          #DC0D
0023          STA          #D00D
0024          LDY          #0
0025          STX          #DC0E
0026          STX          #3FFF
0027          STX          #D020
0028          STX          #D012
0029          LDA          #D011
0030          AND          #7F
0031          STA          #D011
0032          DEX
0033          STX          #D021
0034          LDA          #81
0035          STA          #D01A
0036          LDA          #D019
0037          STA          #D019
0038          #0          LDA          #0
0039          STA          COUNT
0040          CLI
0041          JMP          STP
0042  ;
0043          COUNT          .BYTE 0
0044          ;
0045          IRQ          LDA          #D019
0046          STA          #D019
0047          BMI          START
0048          JMP          #EA61
0049          SEI
0050          LDY          COUNT
0051          INX
0052          TXA
0053          AND          #3
0054          STA          COUNT
0055          TAX
0056          LDA          RASTER,X
0057          STA          #D012
0058          LDA          MODE,X
0059          STA          #D011
0060          LDA          SPREN,X
0061          STA          #D015
0062          LDY          #0
0063          DEY
0064          BNE          WAIT
0065          LDA          XSIZE,X
0066          STA          #D01D
0067          LDA          YSIZE,X
0068          STA          #D017
0069          LDA          MULT1,X
0070          STA          #D01C
0071          LDA          XMSB,X
0072          STA          #D010
0073          CPX          #0
0074          BEQ          VRK03
0075          CPX          #3
0076          BEQ          VRK03
0077          LDY          #7
0078          LP1          LDA          SCOLS,Y
0079          STA          #D027,Y
0080          LDA          POINT,Y
0081          STA          #D7F8,Y
0082          DEY
0083          BPL          LP1
0084          LDY          #15
0085          LP2          LDA          COORD,Y
0086          STA          #D000,Y
0087          DEY
0088          BPL          LP2
0089          JMP          EX
0090          VRK03          LDY          #7
0091          LP3          LDA          SCOLS,Y
0092          STA          #D027,Y
0093          LDA          POINT,Y
0094          STA          #D7F8,Y
0095          DEY
0096          BPL          LP3
0097          LDY          #15
0098          LP4          LDA          COORD,Y
0099          STA          #D000,Y
0100          DEY
0101          BPL          LP4
0102          EX          CLI
0103          JMP          #EA61
0104  ;
0105          RASTER          .BYTE 0,2,4F0,4FF
0106          MODE          .BYTE #13,#13,#41B,#413
0107          SPREN          .BYTE 255,255,255,255
0108          XSIZE          .BYTE 0,0,0,0
0109          YSIZE          .BYTE 0,0,0,0
0110          MULT1          .BYTE 0,0,0,0
0111          XMSB          .BYTE #0A,#0C,#0C,#0C
0112          SCOLS          .BYTE 0,0,0,0
0113          .BYTE 1,1,1,1
0114          SCOLSK          .BYTE 1,1,1,1
0115          .BYTE 1,1,1,1
0116          COORD          .BYTE #20,#10,#50,#30
0117          .BYTE #00,#50,#A0,#70
0118          .BYTE #D0,#70,#F0,#50
0119          .BYTE #10,#15,#50,#10
0120          COORDK          .BYTE #00,#10,#A0,#10
0121          .BYTE #00,#00,#F0,#00
0122          .BYTE #10,#10,#30,#10
0123          .BYTE 11,11,11,11
0124          POINT          .BYTE 11,11,11,11
0125          .BYTE 11,11,11,11
0126          POINTK          .BYTE 11,11,11,11
0127          .BYTE 11,11,11,11
0128  ;
0129          .END

```


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 olyat 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 könyvtalálvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

14. feladat: Nyolc királynő

Elhelyezhető-e egy sakk-táblán nyolc királynő úgy, hogy egyik se üsse semelyik másikat? Írjon programot, amely megadja az összes lehetséges elrendezést!

Megoldás

A feladat legegyszerűbb megoldása az lenne, ha a sakk-táblán véletlenszerűen elhelyezzük a nyolc királynőt, majd megvizsgáljuk, ütik-e egymást. Ha igen, akkor vizsgáljuk a következő elrendezést. Ily módon igen egyszerű programhoz jutunk, amelynek csak egy hibája van — nagyon hosszú ideig fut, hiszen 4 426 165 368 a lehetséges elrendezések száma.

A mostani feladat legyen némileg rendhagyó: ne csak az legyen a cél, hogy egy, a problémát megoldó programot írunk, hanem az is, hogy hatékony programot írjunk.

Jobb megoldást kaphatunk, ha kihasználjuk, hogy a feladat megoldását adó elrendezésnél minden oszlopban pontosan egy királynőnek kell állnia, mert különben ütik egymást. Ha ennek figyelembevételével határozzuk meg a vizsgálandó elrendezéseket, már csak 16 777 216 esetet kell vizsgálni. Ez is elég nagy szám, ezért gyakorlati megvalósításra még aligha alkalmas.

Tovább javíthatjuk az algoritmust, ha minden királynő

elhelyezése után megvizsgáljuk, hogy üti-e a már a táblán lévőket. Így a hibás esetek nagy számát időben ki tudjuk zárni, s egy jól használható algoritmushoz jutunk.

Megoldásul két programot közlünk itt: egy Turbo Pascal 5.0 és egy Turbo—C 2.0 nyelvűt. Mindkettő IBM PC számítógépen, Herkules monokróm grafikus adapterrel futott. A két program jól mutatja a két nyelv közötti különbségeket és hasonlóságokat. A programok kissé különbözőek, a feladat két eltérő megoldását jelentik. A Pascal nyelvű program a királynők elhelyezését rekurzívan, a C nyelvű program iteratív módon valósítja meg.

Tekintsük át először a két programban fellelhető, egymásnak megfelelő elemeket. A sakk-tábla tárolására a 'tábla' nevű tömb szolgál. Ennek az egydimenziós tömbnek minden eleme azt mutatja, hogy az adott oszlopban hányadik sorban van a királynő.

A 'kiir' eljárás feladata a sakk-tábla megjelenítése. Ez elég egyszerű formában történik. A vízszintes és a függőleges vonalak meghúzása után a tábla beszámozása következik. Végül az eljárás nyolcszor meghívja a 'királynő' szubrutint, amely meghatározott mezőbe kirajzol egy-egy királynőt. Jelen esetben ez egyszerűen csak egy négyzetet rajzol.

A kiválasztott elrendezés ellenőrzését 'üt' eljárás végzi. Ez true értékkel tér vissza, ha

az os oszlopban álló királynőt üti valamely os-nél kisebb oszlopban lévő.

A Pascal nyelvű és a C nyelvű szubrutin szinte sorról sorra megfeleltethető egymásnak. A feladatot ténylegesen megoldó részek azonban jelentősen eltérnek.

Vizsgáljuk meg először a rekurzívan működő Pascal nyelvű programot. A 'tesz' eljárás elhelyez az 'os'-adik oszlopba egy királynőt. Ezután megvizsgálja 'üt' segítségével, hogy ezt nem üti-e valamelyik másik már táblán lévő. Ha igen, akkor az 'os'-adik oszlopban eggyel odébb teszi a királynőt, és újra próbálkozik. Ha erre nincs lehetőség, akkor visszalép.

Ha az 'os' oszlopba sikerült úgy elhelyezni egy királynőt, hogy azt nem üti semelyik másik, akkor a 'tesz' rutin önmagát hívja meg a következő királynő elhelyezéséhez. Ha sikerült elhelyezni az utolsót is, akkor a 'kiir' eljárás segítségével a tábla megjelenítése következik. Ezzel a program nem fejeződik be, hanem további megoldásokat keres.

A C nyelvű program működése nehezebben követhető, de hatékonyabb kódot eredményez. Itt is fellelhető mind-egyik, a Pascal program leírásánál ismertetett elem. A visszalépés megvalósítása jóval nehezebb, és az összetett feltételek miatt szükség van két logikai változóra is. Ezek használata ugyan goto utasítással elkerülhető lenne, de ez a program olvashatóságát nagymértékben rontaná. A 'köv' változó arra szolgál, hogy választani lehessen az előre- és a visszalépés közül, a 'vége' pedig a program helyes befejezését biztosítja.

15. feladat: Részvektor összege

Egy program bemenete egy N valós számból álló egydimenziós tömb (vektor). Írjon programot, amely ebből kiválasztja azt az összefüggő részvektort, amely elemeinek összege a legnagyobb! Törekedjen arra, hogy a program nagy N értékekre is gyors legyen!

Pintér Gábor

Egy lehetséges megoldás

								8
							■	7
				■				6
								5
	■							4
			■					3
					■			2
		■						1
a	b	c	d	e	f	g	h	

BNV-élmény



A kőbányai vásárváros ismét megteit élettél. A pavilonokból áradó gépüzgás és a látogatók zsbongása már messziről hirdette, hogy megnyit a tavaszi BNV.

Igazi vásári hangulat uralkodott a hangárok között. Fiatal srácok járták a standokat, s gyűjtöttek be mindent, ami színes és prospektus-szerű. Kis unokák húzták, vonszolták nagymamát, nagypapát, s magyarázták nekik a sok színes technikai „csodabogár” működését.

Nagyon sok — természetesen főleg nyugati — cég reklámjával a fiatalabb korosztályt is megcélozta. A lurkók büszkén mutogatták egymásnak a „zsákmányt”, a Minolta-sörnyit, a Toshiba-jelvényeket, a lufikat, zászlókat, poszterek.

Természetesen a komolyabb érdeklődő is talált magának érdekességeit, újdonságait.

Az IBM-nél a PC-hez szokott felhasználónak is elakadt a lélegzete a PS/2 működése láttán. Az aprócska gép úgy kezelte a képernyőt, hogy szinte nehéz volt elszakadni a látványtól. Egyetlen durvább léptetés, képbeli ugrás nélkül, kimunkált háttérrel, változatosan mutatta be önmagát.

Közvetlenül mellé egy AS/400-ast állítottak ki. Ez egy középkategóriás számítógépcsalád, a legújabb fejlesztések egyike. Néhány kiállítóval odébb szintén egy hasonló jellegű gépcsaládot reklámoztak az Alpha Micronál, az AM-et, de a gépet a kiállítás idejéig nem sikerült behozniuk az országbá, habár — prospektusuk szerint — már Cocom-engedélyes hardver.

Ezeknél a gépeknél főleg a mega- és gigabájtok kápráztattak el, bár már egy táskányi méretű PC, miután végzett a winchester tartalmának listázásával, egy nyolcjegyű számot biggyesztett a lista végére a szabad terület mérete gyanánt (ezt jó lesz megszokni!).

A szoftverekben is megvolt a választék. A szuperfelbontású képernyőkhöz csodálatos

demo-programok társultak, a RANK XEROX-nál pedig bemutatták a Ventura Publisher 2.0 magyar és angol nyelvű változatát. A hazai élvonalba tartozó számítástechnikai korszakvetek is előálltak programújdonságaikkal. A Controll színvonalasan berendezett standja is tükrözte az irányzatot, hogy a piacon az uralkodó géptípus egyértelműen a PC.

A kiállításon persze teret kaptak a robottechnikával, szerszámgyártással foglalkozó cégek is. A bemutató kitöltötte az egész vásár területét, és ha valaki mindent alaposan végig akart nézni, bizony kevésnek bizonyult egy nap.

A BNV hagyományaihoz híven, magas színvonalon, igazán változatos képet vázolt a fejlődés pillanatnyi távlatairól.

(SANY)



Miből lesz a cserebogár?

A következő sorokban négy Commodore 64-es és két Amiga-programról lesz szó. A C64-esre három új játékprogramot és egy új GEOS-családot mutatunk be, míg Amigára egy karakterkészlet-csomagot és egy új akciójátékot.

RAMPAGE

A nemrég készült akciójáték kiváló háromdimenziós grafikával és animációval rendelkezik, bár ez kissé lassítja a futás sebességét. A játékban szereplő szörnyek ismerősek lehetnek a *Monstre-Movie* című hasonló játékból. Itt is megtalálhatók az emberi fantázia óriás szörnyszülöttei King-Kongtól Godzilláig. A játékban egy vagy két játékos vehet részt, végigvonulva az USA különböző városainak 157 helyszínén.

Double Dragon

Megszületett az ismert *Kung-Fu* játék C64-es verziója is. A játékban két ikertestvér méri össze erejét egy garázdalkodó bandával, melynek tagjai elrabolták az egyik testvér barátját. A pergő akciófilmek és játékok kedvelői valószínűleg élvezni fogják a kitünő grafikát és animációt.

GEOS Writer '64

A legújabb GEOS-termék egy új szövegszerkesztő, szótárral egybekötve. Több mint száz ezer szót tartalmazó szótára kiváló-

an használható WYSIWYG-módban is. Forgalmazói levelezéshez ajánlják, de természetesen kompatibilis a *geoPaint* és *geoWrite* GEOS-programokkal is.

Dream Zone

Az amigások újabb fantáziadús játékkal gazdagíthatják gyűjteményüket.

A játék alapötlete kísértetiesen hasonlít a *Freddy-filmekre* (*Nightmare on Elm street*, *Freddy's revenge*), miszerint képtelenek vagyunk félelni, azaz elszabadulni az általunk megáldott szörnyű helyzetekből. Jelen esetben egy örült tudós találmánya tart foga a álomvilágunkban, ahonnan csak különböző szörnyek legyőzésével szabadulhatunk.

ProFonts I—II

Mindkét csomag új karakterkészleteket tartalmaz a *ProWrite* szövegszerkesztő programhoz. A *ProFont Volume I* kiadványokhoz, levelezéshez nyújt további karakterkészleteket különböző méretben. A *ProFonts Volume II*, pedig szép rajzolatú, dekoratív karaktereket tartalmaz, felhívásokhoz, plakátokhoz. Mindkettő használható a *DPaint II*, grafikus- és a *PageSetter* szövegszerkesztő programokhoz.

Tass Csaba

20 LET k=23760
 30 FOR f=34673 TO 34828: POKE
 f, PEEK k: LET k=k+1: NEXT f
 40 STOP
 A versenykörök száma
 10 REM 36504535545145384246192
 646264245
 20 LET k=23760
 30 FOR f=38022 TO 38053: POKE
 f, PEEK k: LET k=k+1: NEXT f
 40 STOP

— Italian — PortugalSpanish — Japanes
 eAustral—

20 LET k=23760: LET g=37890
 30 FOR f=28709 TO 28836: POKE
 f, PEEK k: POKE g, PEEK k: LET k
 =k+1: LET g=g+1: NEXT f
 40 STOP

A pályanevek
 10 REM Jacarepagua ----- D. Fer
 rari — C., Imola — Monaco ----- M
 exico City ----- USA ----- Circuit
 ----- Gilles-Villeneuve ----- Paul-Ricard
 ----- Silverstone ----- Hockenh
 eim ----- Hungaroring ----- Spa
 -Francorchamps ----- Monza -----
 ----- Estoril ----- Jerez -----
 ----- Suzuka ----- Adelaide -----

20 LET k=23760
 30 FOR f=28837 TO 29124: POKE
 f, PEEK k: LET k=k+1: NEXT f
 40 STOP
 A szponzorok
 10REM AGIP ----- BENETTON -----
 ----- BOSCH ----- CAMEL ----- CANON -----
 ----- ELF ----- FIAT
 ----- ICI ----- MARLBORO
 ----- MOBIL ----- RIELLO ----- SHELL
 ----- USF&G -----

A versenykörök számához még van egy kis kiegészítés.
 Ha összehasonlítjuk ezeket a számokat az eredeti körszá-
 mokkal, látjuk, hogy minden most beirt szám 25-tel keve-
 sebb, mint az eredeti volt. Ez az adat is tetszés szerint vál-
 toztatható, de mind az előző, mind a most következő rutin-
 ban a választott értékkel dolgozunk! Hogy miért kell még
 egy rutin? Azért, mert a táblázatból kiolvasott érték csak ak-
 kor kell a gépnek, ha kiírja az egyes futamok hosszát.
 A tényleges futamhossz a 27349—27364 címek közötti 16
 bajton van. Írjuk át ezt is!

10 FOR f=27349 TO 27364
 20 POKE f, PEEK f—25
 30 NEXT f
 40 STOP

Az ártírt Formula One-ban illúzióromboló lenne, ha ver-
 seny közben a program következetesen az 1985-ös dátumot
 írná ki. Ezért írjuk be a következő pár POKE-ot:

POKE 38997,196:POKE 52465,196
 POKE 58957,196:POKE 59608,196

A verseny elején kapott pénz összegét növelhetjük a
 48129 címre beirt értékkel. Véleményem szerint 4 a reális.

Ha ezáltal megvagyunk, szedjünk elő egy üres kazettát,
 és mentjük ki rá a nagy részt, de csak a fejjéc után indítsuk
 a magnót!

SAVE "F1"CODE 26600,38934

Ezután töltsük vissza a programot: a betöltőt és a kép-
 ernyőt az eredeti, a nagy részt viszont az új kazettáról. Ha
 minden O.K., akkor másolóprogram segítségével rakjuk
 össze a Formula One 1989-et!

Kis Piroska Zoltán

Ezúttal Bakos Attila C64-re készült gyűjteményéből adunk közre egy
 csokorra valót.

ACTION BIKER 15489,78
 ANCPITAL 1011,248
 1012,252
 RUN + RESET
 18679,173
 SYS16384
 BEACH HEAD 8567,173
 28486,173
 29768,173
 34074,173
 BOOTY 21063,165
 CAULDRON I. 30759,169
 30760, 9
 CITY COBRA 2191,174
 2192,167
 2411,234
 RUN
 18067,234
 18068,234
 18069,234
 SYS2064
 DALLAS STAR 3271,0
 ESCAPE 5514,27
 FLAK 478,36
 FLASHBIER 9142,173
 GHOST N GOBLINS 2579,12
 GOONIES 3002,173
 16005,173
 GRIDDER 6757,173
 GYRUSS 3999,200
 INFILTRATOR (lemezés verzió)
 LOAD"OUTPUT",8,1
 7252,234
 7253,234

JACK THE NIPPER
 KICKMAN
 LOCO
 LE MANS
 NEMESIS

RAMBO
 TIME RUNNER
 URIDIUM
 WHO DARES WINS
 WHO DARES WINS II
 XERONIX
 YIE AR KUNG-FU
 ZAGA
 ZORRO

7254,234
 3337,234
 3338,234
 3339,234
 9384,234
 9385,234
 9368,234
 9369,234
 9370,234
 9282,234
 9283,234
 9284,234, majd ezek
 után SYS2176.
 ZAPPIT
 7424,230
 26944,77
 2547,255
 10239,166
 43287,0
 42386,12
 SYS32777
 6099,173
 8543,9
 13972,életek
 13973,szint
 4516,252
 15697,173
 6991,0
 2273,238
 41603,208
 9551,9
 5423,127
 6431,125



ÖRÖKÉLET

A novoszibirszki pályázat értékelése



A Mikroszámítógép Magazin ez évi 4. számában jelent meg a most értékelt pályázat. A pályázóknak a programot és a blokkdiagramot április 15-ig kellett beküldeniük. A határidő után érkezett műveket nem vettük figyelembe. A feladat a következő volt:

A felhasználó által adott n számú, mindkét oldalán fehér kártyalapot kiterítünk az asztalra. A kártyákat felfelé néző oldalukon 1-től n-ig sorszámozzuk, majd megfordítjuk és összekeverjük. Az újabb terítést követően a kártyák fehéren maradt oldalát is sorszámmal látjuk el. Ha kész, ismét megkeverjük a lapokat, de közben meg is fordítunk néhányat. Végül az asztalra helyeztük kártyák adatait — külön a felfelé, külön a lefelé néző oldalukon szereplő számokat — balról jobbra sorrendben feljegyezzük.

Beküldendő az a BASIC nyelvű programlista és az az algoritmus, amely a fenti adatok alapján meghatározza, mely kártyákat kell megfordítanunk ahhoz, hogy a kártyák felfelé néző oldalán az 1-től n-ig terjedő számok mindegyike pontosan egyszer szerepeljen. Fontos szempont, hogy az algoritmus, illetve program gyors legyen, és minél kevesebb lépésből álljon.

A feladat egyébként a tavalyi novoszibirszki táborban, az egyik versenyről származik.

A pályázatra 31 pályamű érkezett. Az elbírálás pártatlansága érdekében minden munkát pontoztunk, aminek alapján az első hét helyezett:

- 1—2. Fülöp Attila, Szeged 104 pont
 - 1—2. Maróti Miklós, Szeged 104 pont
 3. Monfera Róbert, Erd 100 pont
 4. Brumár Gábor, Budapest 95 pont
 5. Oravecz Róbert, Miskolc 93 pont
 - 6—7. Haluska László, Budapest 87 pont
 - 6—7. Kórász Tamás, Szeged 87 pont
- Három indult kellett diszqualifikálnunk, mert meglepően hasonló — majdnem karakterről karakterre azonos —

programokat küldtek be. Egyébként kb. 90-95 pontot kaptak volna. A két 0 pontos pályamű írói más feladatot oldottak meg, nyilván félreértették a feladat szövegét.

A pontokat a következőképpen számoltuk. A blokkdiagramért összesen 30 pontot lehetett kapni, 20-at az áttekinthetőségért, és 10-et azért, hogy mennyire jól magyarázza a programot. 40 pontot adtunk, ha a program működik, 0-t, ha nem. (Horváth Attila erre 20-at kapott, mert a program kiszámolja ugyan, de nem írja ki, hogy mely kártyákat kell megfordítani.) A program rövidségéért 30—x/2 pontot adtunk, ahol az x a REM-eken kívüli lépések száma. Ezenkívül 20 pontot lehetett kapni a sebességért. Ezt a pontszámot — objektív mérés lehetőségének híján — a programok egymással való összehasonlítása után ítéltük meg.

Sok pályázó nem vette komolyan, hogy „fontos szempont az elbírálásnál a program (...) hossza”, és igen terebélyes alkotásokat küldtek be. Ezek persze majdnem mind 0 pontot kaptak a hossza. Az algoritmus sebességére vonatkozó figyelemzetest is sokan elengedték a fülük mellett, így kaphattunk például olyan programot is, amely n-köbbel arányos lépésszámmal végzi a dolgot. A pályamunkák — e megjegyzésektől eltekintve — arra utaltak, hogy akik jelentkeztek, komolyan vették a pályázatot, és rendszeresen kidolgozott algoritmusokat küldtek be.

A nyertes programok közül Fülöp Attilát ismertetjük (lásd a listát). A program úgy fut, hogy rögtön az adatbekérés közben jelöli ki a megfordítandó kártyákat. Ha az éppen most beirt kártya tetején levő szám eddig még nem fordult elő (azaz A(D)=0), akkor ezt a kártyát egyszerűen hozzárakja a többihez. Ha már van ilyen, az aktuális kártya megfordítását jelző mutatót állítja (ha Z=0, Z=-1, ha nem, Z=0), és ha megfordítva stimmel, azaz A(E)=0, ak-

kor megfordítva rakja oda. Ha megfordítva sem jó, azaz A(E) < > 0, akkor ezt a kártyát kicserebeli azzal, ami miatt nem jó, és ez utóbbival folytatja a vizsgálatot. A legvégen pedig kiírja, hogy mely kártyákat kell megfordítani. Ha például hat kártyát használunk, és az ezeken lévő számok rendre (1;4), (2;5), (1;6), (2;6), (3;4), (3;5), voltak, akkor az eredmény a következő: 1, 6, 4, azaz az első, negyedik és hatodik kártyát kell megfordítanunk.

Novoszibirszkbe az első négy helyezett mehetett volna, de végül is Brumár Gábor, a negyedik sajnos nem tudta vállalni a részvételt. A Neumann János Számítógéptudományi Társaság úgy döntött, hogy negyediknek Készmári Mátyást küldi, aki a Nemes Tihamér Középiskolai Számítástechnikai Versenyen hetedik helyezést ért el, de az egyetemi felvételin elsős léteére az első tíz helyezett megillető kedvezményben nem részesülhetett. Megjegyezzük, hogy Maróti Miklós (másodikos) a Nemes Tihamér versenyen hatodikként ugyancsak az első tíz között végzett.

Szerencsés utat és jó munkát kívánunk tehát a négy táborozónak. Akinek most nem sikerült nyernie, tanulmányozza szorgalmasan a Mikroszámítógép Magazin számaát, keressen lehetőséget, és próbálkozzon más, újabb pályázattal!

Futó Tibor — Szoldán Péter

```

110 INPUT N
120 DIM A(N),B(N),C(N)
130 FOR B=1 TO N
140 INPUT D,E
150 Z=0:Z1=0
160 IF A(D)=0 THEN GOTO 100
170 A(D)=E: E(D)=B:GOTO 250
180 Z=(Z+D)
190 IF A(E)=0 THEN GOTO 210
200 A(E)=B: B(E)=E: C(E)=Z:GOTO 250
210 Z1=Z+E: E(E)=E: C(E)=Z1
220 A(E)=B: B(E)=E: C(E)=Z
230 D=E: E=Z1:Z1=Z1+E
240 GOTO 180
250 NEXT B
260 FOR B=1 TO N
270 IF C(B) THEN PRINT B;B)
280 NEXT B
READY.
    
```

játék programok	TOP LISTA								felhasználói programok	
	IBM	AMIGA	C-128	C-64	C-416	SPECTR.	ENTERP.	TYC		APPLE
1. Grand Prix C.	•	•	•	•	•	•	•	•	•	1. Geos 2.0
2. Rocket E.	•	•	•	•	•	•	•	•	•	2. dBase IV.
3. UCHlympics	•	•	•	•	•	•	•	•	•	3. Giga Paint
4. Ocean Ranger	•	•	•	•	•	•	•	•	•	4. Amiga Paint
5. Robocop	•	•	•	•	•	•	•	•	•	5. Art Studio 1.2
6. Ritzi	•	•	•	•	•	•	•	•	•	6. Printfox
7. Zak McCrack.	•	•	•	•	•	•	•	•	•	7. Pagefox
8. Bosszu	•	•	•	•	•	•	•	•	•	8. Newerom
9. Battle Chess	•	•	•	•	•	•	•	•	•	9. Windows
10. Shanghai	•	•	•	•	•	•	•	•	•	10. Texter

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

Címünk:
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
Diák szerkesztőség

Rezidens óra IBM AT-n

Az elmúlt alkalomhoz hasonlóan — amikor egy C64-re írt rezidens óra programmal ismerkedtünk meg a bajtók átálakítása kapcsán —, most a 8086-os mikroprocesszor-család gépi kódján írt rezidens óra program kapcsán ejtünk pár szót a bajtók átálakításáról. Az óra, amit le fogunk olvasni, szintén BCD kódban adja vissza az időt. Az IBM AT-nak van egy belső órája, amit a gép kikapcsolásakor is életben tart egy kisméretű gombelém.

A pontos időt a DOS-ban a TIME utasítással állíthatjuk be. A programban a BIOS-nak a rendszeridőt lekérdező 1AH megszakítását (interruptját) fogjuk felhasználni, amely a kézikönyvek tanúsága szerint csak IBM AT-n áll rendelkezésre, tehát IBM XT-n nem. Itt érdemes megjegyezni, hogy a program az IBM PS/2-n is változtatás nélkül futtatható volt.

Előjáróban még annyit, hogy az 1AH BIOS-megszakítás akkor kérdezi le a valós idejű órát, ha az 1AH megszakítás meghívása előtt az AH tárolóba 2-t teszünk. A megszakításrutin lefutása után a rendszeridőt BCD kódban kapjuk vissza: a CH-ban az órákat, a CL-ben a perceket és a DH-ban a másodperceket. A C64-gyel ellentétben itt nem jelzi külön bit a délutánt és a délelőttöt, hanem egy igazi 24 órás időmérővel van dolgunk. Ezért az idő kiírása is egyszerűbb, mint a C64 esetében volt: nem kell a délutánt jelző bitet figyelni és annak függvényében 12-t a BCD számhoz hozzáadni.

Ahhoz, hogy az időt a képernyőn megjeleníthessük, csatlakoznunk kell egy olyan megszakításhoz, amely másodpercenként többször megy végbe. Erre a BIOS 1CH megszakítása nyújt lehetőséget, amely minden másodpercben 18,2-szer hajtódik végre, és amelyet a rendszer a jó kedvű programozók kedvéért tart fent, ugyanis itt alapállapotban semmilyen program nincs, csak egy üres IRET utasítás. Erről mindenki saját maga is meggyőződhet, ha a DEBUG segítségével megnézi a megszakításvektorok táblájában azt a címet, amely az 1CH megszakításhoz tartozik.

A DEBUG d 0:70 L 4 utasításával megjeleníthetjük azt a 4 bajtot, amely a megszakításcímet tartalmazza; ez az én esetemben egy HAT—31 típusú, IBM AT-vel kompatibilis számítógépen a következő volt: 53 FF 00 F0. Mivel a cím első bajt, felső bajt sorrendben tartalmazza előbb az ofszet, majd a szegmens értékét, ezért ez esetünkben az F000:FF53 címet jelenti. Szintén a DEBUG segítségével nézhetjük meg azt is, hogy mi található ezen a címen, ha kiadjuk az U F000:FF53 utasítást. Egy IRET-et kell találnunk.

A rezidens óra gépi kódja a listán látható. A programot úgy írtuk meg, hogy más programokkal együtt is működőképes legyen; olyanokkal is, amelyek korábban már esetleg szintén átírták az 1CH megszakításrutin-címet a BIOS vektorok területén: 0:0070—0:0073. Emiatt az 1CH megszakításrutin eredeti címét a DOS 35H funkcióhívása segítségével előbb lekérdezzük (lásd a lista 01B2—01BE sorait), és a program adatterületén, 0108—010B alatt tároljuk OLD_VEKT illetve OLD_INT néven is. A DOS 35H funkció meghívása előtt a lekérdezendő megszakítás sorszá-

mát, esetünkben az 1CH-t kell az AL tárolóba tenni, az AH-ba pedig természetesen a 35H kerül. Visszatéréskor — az INT 21H után, amely a DOS-funkció hívását valószínűleg meg — ES:BX-ben kapjuk vissza a keresett megszakításvektor címét.

A saját rutinunk címét a 25H-s DOS-funkció hívásával írhatjuk be a BIOS-megszakítás vektortáblájába, amelyhez a DS:DX-ben kell a funkcióhívás előtt elhelyezni az új címet (a lista 01B0—01B1, illetve 01C2—01C9 sorai). Rezidenssé pedig szintén a DOS-funkciók egyikével, a 31H-val lehet tenni a programot, amely-

A rezidens óra gépi kódú programja

```

COMMENT * Ez a program a rendszer időt kérdezi le és írja fel a
színes (CGA) képernyő jobb felső sarkába. *

;----- Az állandók területe -----

= 0020          SPACE          EQU    20h
= 000D          SZINKOD        EQU    0Dh
= 003A          COLON          EQU    3Ah
= B800          SCREEN         EQU    0B800h
= 3100          RESID          EQU    3100h
= 0021          DOSFUNC        EQU    21h
= 001A          BIOS_IDO       EQU    1Ah

;----- Makrok birodalma -----

KEPIRO MACRO REGX              ;; képernyőre írja
MOV     ES:[DI],REGX           ;; ki a bajtot
ADD     DI,2
ENDM

SZAMIRO MACRO REGX             ;; REGX-et számlent
OR      REGX, 0030h           ;; írja ki a
KEPIRO REGX                   ;; képernyőre
ENDM

CHARIRO MACRO CHR              CHR
MOV     BL,CHR
KEPIRO BX
ENDM

;----- Program terület -----

0000          CODE            SEGMENT
ASSUME CS:CODE, DS:CODE, SS:CODE

0100          ORG            100h

0100          E9 01AF R        START: JMP     INIT
0103          B900            KEP     DW     (SCREEN)
0105          0D             SZIN    DB     (SZINKOD)
0106          00             IDO_TMP DW     (0000h)
0108          OLD_VEKT       OLD_VEKT LABEL DWORD
0108          0000 0000      OLD_INT DW     0000h, 0000h

010C          TIMER:
010C          FA             CLI
010D          2E: FF 06 0106 R INC     CS:[IDO_TMP]
0112          2E: 83 3E 0106 R 09 CMP    CS:[IDO_TMP], 0009h
0118          74 06         JE      IDOKI_JELZES
011A          FB             STI
011B          2E: FF 2E 0108 R JMP     DWORD PTR CS:[OLD_VEKT]

```


hez a program helyszükségletét paragrafusokban kell megadnunk (a lista 01CB—01E1 sorai).

Nézünk meg egy kicsit részletesebben, hogyan történik a BCD bájtok átalakítása. Az IDŐKIJELZÉS alprogramnál először elmentjük a tárolók korábbi értékét, majd a 1AH BIOS megszakítással lekérdezzük az órát. Ezután a visszakapott változók értékeit a veremtárolóban helyezzük biztonságba (a lista 0129—0138 sorai). Indirekt módon, hogy később szabadon módosítani lehessen őket, kezdőértéket adunk az ES-nek, a DI-nek és a BH-nak. Az extra szegmens, az ES, a CGA képernyő címét, a B800H-t kapja meg kezdőértékként az adatterület 0103H-s szóterületéről. Amennyiben az IBM AT monitorja nem színes, akkor az ezen az adatterületen lévő szó értékét B000H-ra kell módosítani, mivel a fekete-fehér mono monitorok a kép megjelenítéséhez szükséges információt erről a memóriaterületről veszik. A DI indexregiszternek a 88H kezdőértéket adtuk, amely a képernyő legfelső sorának jobb oldalára esik. A BH tároló tartalmazza a bájtt megjelenítéséhez szükséges színinformációt, az úgynevezett attribútumbájtot.

Itt most pár szót kell szólnunk arról, hogyan is tárolja az IBM a karakteres képernyő megjelenítéséhez szükséges információkat. A karakteres képernyő sorfolytonos számozású, azzal a megszorítással, hogy a megjelenítendő bájtt ASCII kódját az attribútumbájtt követi, amely a bájtt megjelenítéséhez szükséges színinformációkat tartalmazza. Esetünkben a szinkód 0DH-val egyenlő, amelyben a 0 a fekete háttérrel jelenti, a D pedig a karakter intenzív lila színt. A 80 oszlopos megjelenítés esetében ez azt jelenti, hogy egy sor 160 bájtból épül fel. Így most már láthatjuk, hogy a DI-nek a 088H hexadecimális kezdőértéke, amely megfelel a decimális 136-nak, még nem jelenti teljesen a sor végét, hiszen akkor nem férne el mögötte több karakter. Erre a területre még (160-136)/2=12 darab karakter írható. Mi 8 karaktert írunk ki a képernyőre, így a képernyő jobb oldalán 4 karakter helye üresen marad. Ha a DI indexregiszternek a 0088H helyett a 0090H kezdőértéket adnánk, akkor kerülne pontosan az óra az első sor jobb szélére.

Visszatérve a programhoz, az átalakított BCD számokat makrók írják be közvetlenül a képernyő memóriájába, aminek a gyorsaságon kívül megvan még az az előnye is, hogy a DOS nem szerzett róla tudomást, így még olyan szövegszerkesztők esetében is nyugodtan „ketyeghet” az óránk, amelyek nem a grafikus, hanem a karakteres képernyőt használják fel a szövegek szerkesztésére. Ilyen többek között a Norton Editor és az IBM Personal Editor I és II is. Ekkor ugyanis a ketyegő óra ér-

téke nem válik a szerkesztett szöveg részévé, és ily módon nem szemeteli tele a szerkesztendő szöveget az órák, percek és másodpercek fölösleges bájtaival.

A lista 014A-adik soránál a CH-ban lévő órák értékét — amelyről tételezzük fel, hogy 23H-val egyenlő — visszzük át az AL regiszterbe, majd meghívjuk a KEPIR alprogramot, ugyanis ez fogja a BCD bájtot a képernyőre írni. A KEPIR alprogram 017E-edik soránál az AND AX, 00FH utasítás hatására AH=0-val, AL=20H-val lesz egyenlő. A következő két utasítás: a MOV CL,4 és az SHR AX, CL négyszzer jobbra forgatja az AX-et, emiatt az AL most 02H-val lesz egyenlő, amit az alprogram az összehasonlítás után a TIZESEK címkejű soroknál (194-19C) számként ír fel a képernyőre; a 2-es ASCII kód-

ja 32H, emiatt kell a lista 196. sorában a BL-hez 30H-t hozzáadni. Figyelem! A BH állandóan az attribútumbájtot, a szinkód bájttát: 0DH-t tartalmazza, ezért lesz a BL-ben lévő bájtt megjelenített színe intenzív lila a képernyőn.

A lista 199. sorában a MOV ES [DI], BX utasítás az, ami közvetlenül a képernyőmemória területére beírja a BX értéket, mégpedig most a B800:88-as memóriarekeszbe a BL értéket: a 2-es szám ASCII kódját, a 32H-t, a B800:89-es tárolóba pedig a BH értéket, a szinkódot: 0DH-t. A lista 19C-edik sorában az indexregisztert 2 bájttal továbbléptetjük, hogy az új karakter a korábbi mellé kerüljön. Az EGYESEK címkejű 19F-edik sorban újra elővesszük a BCD szám eredeti értékét, a 23H-t, majd az AND AX, 0FH utasítással

```

0120                                IDOKIJELZES:
0120 9C                                PUSHF
0121 1E                                PUSH DS
0122 50                                PUSH AX
0123 53                                PUSH BX
0124 51                                PUSH CX
0125 52                                PUSH DX
0126 56                                PUSH SI
0127 57                                PUSH DI
0128 06                                PUSH ES
0129 16                                PUSH SS
012A 54                                PUSH SP
012B 55                                PUSH BP
012C 2E: C7 06 0106 R 0000          MOV CS:ID0_TMP1,0
0133 B4 02                                MOV AH,2 ; BIOS Interrupt
0135 CD 1A                                INT BIOS_IDO ; az óra lekerdezésére
0137 52                                PUSH DX ; DH = másodperc
0138 51                                PUSH CX ; CH = órák, CL = percek
0139 BB ---- R                          MOV AX,CODE ; DS-nek új kezdő-
013C 8E D8                                MOV DS,AX ; értéket adunk.
013E A1 0103 R                          MOV AX, [KEPI] ; Színes keptar
0141 8E C0                                MOV ES, AX ; ES=SCREEN
0143 BF 0088 R                          DI, 88h ; a felső sor jobboldala
0146 8A 3E 0105 R                       MOV BH, [SZINKO] ; SZINKOD: intenzív lila
014A 8A C5                                MOV AL,CH ; CH = OKA kiírása
014C EB 017D R                          CALL KEPIR ; a képernyőre.
                                           CHARIRO COLON ; kettospont
014F B3 3A                                1 MOV BL,COLON
0151 26: B9 1D                                2 MOV ES:[DI],BX
0154 83 C7 02                                2 ADD DI,2
0157 59                                POP CX
0158 8A C1                                MOV AL,CL ; CL = PERC kiírása
015A EB 017D R                          CALL KEPIR ; a képernyőre.
                                           CHARIRO COLON
015D B3 3A                                1 MOV BL,COLON
015F 26: B9 1D                                2 MOV ES:[DI],BX
0162 83 C7 02                                2 ADD DI,2
0165 5A                                POP DX
0166 8A C6                                MOV AL,DH ; DH = másodperc kiírása
0168 EB 017D R                          CALL KEPIR ; a képernyőre.
016B 5D                                POP BP
016C 5C                                POP SP
016D 17                                POP SS
016E 07                                POP ES ;Vissza Allitjuk
016F 5F                                POP DI ; a tárolók
0170 5E                                POP SI ;korábbi tartalmát.
0171 5A                                POP DX
0172 59                                POP CX
0173 58                                POP BX
0174 58                                POP AX
0175 1F                                POP DS
0176 9D                                POPF
0177 FB                                STI
0178 2E: FF 2E 0108 R /                JMP DWORD PTR CS:[OLD_VKTI]

```


töröljük az AX felső bitjeit, csak az alsó 4 bitet hagyjuk változatlanul, így AX 0003H-val lesz egyenlő, majd ezt a korábbihoz teljesen hasonlóan, számként a képernyőre írjuk, a lista 1A3—1AE sorainál. Ezzel véget is ér a KEPIR alprogram.

A továbbiakban a program az órák értéke után egy kettőspontot tesz, majd a veremből előszedi a percek korábbi értékét, és azt az előbbiekhöz teljesen hasonló módon a KEPIR alprogram segítségével a képernyőre írja fel. Ha a kiírandó BCD szám kisebb tiznél, akkor a zisek helyén a nulla helyett egy helyközt (space-t) ír ki a KEPIR alprogram (lásd a lista 185—191-es sorait). A másodperceket is teljesen hasonló módon írja ki a program a 165—168-as soroknál, majd a veremből visszaállítja a változók korábbi értékeit, és

a 178-as sorban egy FAR JMP utasítással visszatér az eredeti 1CH BIOS-megszakításhoz, ha más program nem irányította át, akkor jól tudjuk, hogy csak egy egyszerű IRET utasítást tartalmazott.

Jelenleg a program másodpercenként kétszer írja fel a képernyőre az időt (lásd a lista 10D—11B sorait). Ha a 112-es sorban a 0009H-t lecsökkentjük, akkor gyakrabban fog bekövetkezni az idő kiírása, ha pedig a 9-et megnöveljük, akkor ritkábban. Az 1CH megszakításra másodpercenként 18,2-szer kerül sor (a C64 ebből a szempontból az IBM-nél sokkal „szorgalmasabb”, hiszen ott a rendszermegszakításokra másodpercenként 60-szor kerül sor), így ha a 0009H helyett 0001H-t írunk, akkor másodpercenként 18-szor jelzi ki az időt. Figyelem! 0000H esetében lesz a ki-

rás frekvenciája a legalacsonyabb, ezt ugyanis 1000H-nak értelmezi a program, emiatt csak minden 65536/18,2=3601 másodpercenként, azaz óránként mindössze egyszer jön létre az idő kiírása.

A programot a lista szerint egy szöveg szerkesztővel kell elkészíteni, majd a MASM segítségével lefordítani. A keletkezett OBJ kiterjesztésű fájlt a LINK segítségével alakíthatjuk át futtatható EXE fájlra. Ne is kísérletezzünk az EXE2BIN programmal azon, hogy az EXE fájlt COM típusúra átalakítsuk, mert az nem lehetséges.

A programot több már rezidens program társaságában zavartalanul tudtam használni szövegszerkesztőkkel együtt éppúgy, mint a legkülönbözőbb lemezmuveletek alatt, illetve a DOS-parancsok esetén egy IBM AT-vel kompatibilis HAT-31 típusú gépen, valamint egy eredeti, VGA képernyős, IBM PS/2-50-es számítógépen.

Kísérletezzünk szabadon a programmal akkor is, amikor már rezidens módon a helyére töltöttük, oly módon, hogy a DEBUG-gal megkeressük, a memória melyik területén is helyezkedik el éppen. Mivel mi tudjuk, hogy az 1CH BIOS-megszakítás címét módosítottuk, ennek alapján könnyen megtaláljuk a rendszerbe rezidens módon beépült programunkat, ha megnézzük a DEBUG D utasításával a BIOS-megszakítás vektortáblában az 1CH megszakítás címét, amely mint már korábban említettük, a 00070—00073 területen található. Ha így megtaláltuk a programot, akkor keressük meg például a SZINKÓD helyét — a program legelején, az adatterületen —, és ott a 0DH szinkódot változtassuk meg olyanra, amilyen nekünk a legjobban tetszik. Így az órákat a saját ízlésünk szerint alakíthatjuk. Az attribútum- vagy szinkódbajt egyes bitjeinek a jelentése a következő:

- 0—2. bit: a karakter színe kék, zöld, piros sorrendben
- 3. bit: a karakter színintenzitását jelöli
- 4—6. bit: a háttér színe: kék, zöld és piros
- 7. bit: a karakter villogtatása

A DI indexregiszter kezdőértékét, a lista 0143-as sorában található 0088H-t módosítsa, a képernyő más területére írathatjuk ki az órákat. Figyelem! Ügyelnünk kell arra, hogy a DI értéke mindig páros legyen, mert különben a karakterek ASCII kódja felcserélődik azok attribútum-bájtjával, mivel ilyenkor nem a megfelelő sorrendben történik ezeknek a bájtoknak a képernyő memóriájába való beírása.

Legközelebb a bájtok, vagy ami egyre megy, a kiterjesztett ASCII kódok átalakítását nézzük meg C64-re és IBM XT/AT-re.

Szabó Péter Pál

```

;
; A színes képernyőre ír ki ez az alprogram számokat, egy bájt BCD
; számszámjegyeit. A bájt értéke: 0-59 közé esik.
;

```

```

017D          KEPIR      PROC    NEAR      ; Az eredeti AX-t
017D 50        PUSH AX      ; a verembe teszük, az
017E 25 00F0   AND    AX, OF0H          ; also 4 bit <= 0
0181 B1 04     MOV    CL, 4
0183 D3 E8     SHR    AX, CL           ; AX / 16
0185 3C 00     CMP    AL, 0            ; AL = 0 ?
0187 75 0B     JNE    TIZESEK         ; Ha igen, akkor
                                ; egy helyközt ír ki
                                CHARIRO SPACE
0189 B3 20        1     MOV    BL, SPACE
018B 26: 89 1D   2     MOV    ES: [DI], BX
018E 83 C7 02   2     JMP    DI, 2
0191 EB 0C 90     AND    DI, 2
0194 8A D8     TIZESEK: MOV    BL, AL
                                SZAMIRO BX
0196 83 CB 30   1     OR    BX, 0030h
0199 26: 89 1D   2     MOV    ES: [DI], BX
019C 83 C7 02   2     ADD    DI, 2
019F 58         EGYESEK: POP    AX
01A0 25 00F0   AND    AX, OFH          ; az eredeti AX, az
01A3 8A D8     MOV    BL, AL          ; also 4 bit marad
                                ; amit számként ír
                                SZAMIRO BX
                                ; ki a képernyőre.
01A5 83 CB 30   1     OR    BX, 0030h
01A8 26: 89 1D   2     MOV    ES: [DI], BX
01AB 83 C7 02   2     ADD    DI, 2
01AE C3         RET
01AF          KEPIR      ENDP

```

```

;
; Itt teszi rezidenssé a programot, és a BIOS 1Ch interruptot
; átírányítja a saját, az időt kiíró rutinunkra
;

```

```

01AF 06        INIT:    PUSH    ES
01B0 0E        PUSH    CS
01B1 1F        POP     DS
01B2 B9 351C    MOV    AX, 351Ch          ; ES:BX-ben kapjuk meg a
01B5 CD 21     INT    DOSFUNC           ; megszakítási rutin
01B7 2E: 89 1E 0108 R. MOV    CS: [OLD_INT1], BX ; eredeti címét, amit
01BC 8C C0     MOV    AX, ES            ; kimentünk.
01BE 2E: A3 010A R. MOV    CS: [OLD_INT+2], AX
01C2 BD 16 010C R. LEA    DX, WORD PTR TIMER
01C6 B8 251C    MOV    AX, 251Ch         ; DS:DX-ből átírjuk a
01C9 CD 21     INT    DOSFUNC           ; BIOS 1Ch interruptot
01CB 07        POP     ES
01CC 8C CA     MOV    DX, CS
01CE 8C C0     MOV    AX, ES
01D0 2B D0     SUB    DX, AX
01D2 B8 01AF R. MOV    AX, OFFSET INIT   ; A cím átszámítása
01D5 B1 04     MOV    CL, 4             ; paragrafusra, 4-szer
01D7 D3 E8     SHR    AX, CL            ; jobbra lejtetjük.
01D9 03 D0     ADD    DX, AX
01DB 42     INC    DX
01DC B8 3100    MOV    AX, RESID         ; Rezidens módon fejezi
01DF CD 21     INT    DOSFUNC           ; be a program a
01E1          ENDS
                                ; működését.
                                START

```

0 Warning Errors
0 Severe Errors


```

writeln('bivni a levédett programot, különben csak az aktuális al-könyvtárból hívható. ');
writeln('Ha a program már e védelem előtt futtatott volt, ez a védelem semmit sem ér. ');
halt;
end;

procedure ERRORDierr_code : byte; ( hibáüzenet kiírása )
begin
  textcolor(0);
  case err_code of
  0: writeln('A file nem létezik az adott útvonalon. ');
  1: writeln('A file nagyobb 65000 byte-nál, nem bővíthető tovább. ');
  2: writeln('A file kisebb 100 byte-nál, nem érdekes védeni. ');
  3: writeln('A file már védett. ');
  end;
  textcolor(2);
  close (f);
  halt;
end;

procedure NAMEIadat : sevi; ( string beírása az 'f' file-ba )
var j, l : integer;
begin
  l:=length(adat);
  for j:=1 to l-1 do
  begin
    rc:=ord(adat[j]);
    write(f,rc);
    read(f,rc);
    j:=j+1;
  end;
  rc:=ord(adat[l]);
  write(f,rc);
  writeln(f);
  jel:=1;
end;

procedure ADDRESS(cia : real); ( 2 byte-os érték beírása az 'f' file-ba )
var hi, lo : byte;
    r : real;
begin
  hi:=trunc(cia*256);
  rc:=hi;
  lo:=frunc(cia*256);
  rc:=lo;
  write(f,rc);
  read(f,rc);
  j:=j+1;
  rc:=hi;
  write(f,rc);
  writeln(f);
  jel:=1;
end;

begin
  ( ha nincs paraméter, akkor a program ismertetője jelenik meg )
  if paramcount=0 then INFO;

  ( a programunk neve az útvonallal )
  snev:= '';
  getdir(0,snev);
  snev:=snev+'VSAFE.COM';

  ( a védendő program keresési útvonala, neve )
  pnev:=paramstr(1);

  ( a programfile megnyitása )
  assign(f,pnev);
  (E)-
  reset(f);
  ( ha hiba lépett fel, akkor nem találta meg, hibáüzenet kiírása )
  if ioreult<0 then ERROR(0);
  (E)-
  ( a megtalált védendő program hosszának beolvasása )
  hossz:=longfilesize(f);

  ( ha a file hosszabb 65000 byte-nál, akkor hibáüzenet )
  if hossz>65404 then ERROR(1);

  ( ha a file rövidebb 101 byte-nál, akkor hibáüzenet )
  if hossz<101 then ERROR(2);

  ( vizsgálát : a program le volt-e már védve ilyen útvon ? )
  longseek(f,hossz-3);
  st:= '';
  cst:=chr(4F4)+chr(4F4)+chr(4F4); ( $ db HLT kód )
  while not eof(f) do begin
    read(f,rc);
    st:=st+chr(rc);
  end;

```

```

if st<st then ERROR(3);

( a kiegészítő program abszolút címe )
vcia:=hossz*4101;

( a program kezdő címének beolvasása )
reset(f);
read(f,rc);
read(f,rc);
close(f);
read(f,rc);
ch:=rc;

( az u) kezdőcím kiszámítása )
uh:=trunc(hossz/256);
aa:=uh;
ul:=trunc(hossz-(aa*256));
ul:=ul-3;

( az u) cím beírása )
reset(f);
read(f,rc);
rc:=ul; write(f,rc);
rc:=uh; write(f,rc);

( u) rekordra posicionálás )
longseek(f,hossz-1);

( programunk megnyitása )
assign(ff,snev);
reset(ff);

( posicionálás a programunk kiegészítő részére )
shossz:=filesize(ff);
seek(ff,shossz-250);

( relatív rekord számolás beállítás )
l:=0;
( a kiegészítő program hozzáférése a védelem programhoz )

while not eof(ff) do
begin
  j:=0;
  read(ff,rc);
  j:=j+1;
  case j of
  7: begin ( a 7-8. byte helyett az eredeti ugrás cím kell )
    rc:=cl;
    write(ff,rc);
    rc:=ch;
    write(ff,rc);
    read(ff,rc);
    j:=j+1;
    writeln(ff);
  end;
  21: NAME(' '+pnevchr(0)); ( a 21. byte-tól kezdődve a program neve kell )
    ( a nevet chr(0)-val kell lezárni )
  100: NAME(pnev' '); ( a 100. byte-tól kezdődve a program neve kell )
  159: ADDRESS(vcia+19); ( a 159-168. byte a programnév címe - 1 )
  171: ADDRESS(vcia+83); ( a 171-172. byte a hibáüzenet címe )
  220: begin ( a 220. byte az u) cím felső értéke )
    rc:=uh;
    write(ff,rc);
    writeln(ff);
  end;
  224: ADDRESS(vcia+5); ( a 224-225. byte az u) cím címe )
  239: ADDRESS(vcia+49); ( a 239-240. byte a figyelmeztető üzenet címe )
  end;
  if jel=0 then write(ff,rc);
  end;

( a sikeres végrehajtás üzenete )
textcolor(2);
writeln(pnev,' --> 'hossz*250/70,' bytes');
close(ff);
close(ff);
end.

```


Programozási fogások és melléfogások



A legutóbbi alkalommal a *Mikrovilágban* közölt Form Writer gépi kódú rutinjait előállító programmal kapcsolatos gondokról írtam. Sajnálatos, de egyáltalán nem szokatlan jelenség, hogy a lapnak a hibás program közlése óta — jelen sorok írásának időpontjáig — megjelent három számában hiába kerestem helyreigazítást. Mintha minden a legnagyobb rendben lenne...

A folytatás sem szívdertő. A BASIC nyelven írt főprogram a *Mikrovilág* idei 8. számában jelent meg. A RUN-ban közölt, elég nehezen kezelhető és feleslegesen túlbonyolított program „magyar” változata további programozástechnikai csacskaságokkal bővült. A teljes program elemzéséhez sok háttér-információra lenne szükség, melyek ismertetésére terjedelmi okokból nem vállalkozom. Helyette egy rövidebb — szerkezetileg ismétlődő — programrészt egyszerűsítésének lehetőségét mutatom be.

Az *1/a listán* egy menüpont-kiválasztást kiértékelő programrészlet látható. A teljes programban összesen hat ilyen található, természetesen eltérő sorszámokkal, mindegyiket az aktuális menü képernyőre írása előzi meg. A menüpontok közül valamely számbillentyű leütésével lehet választani, a 0 leütése az eggyel magasabb menüsintze való visszatérést, illetve a főmenüből a program befejezését jelenti. Ha a szám kivül esik a megadott tartományon, a futás az aktuális menü újra kiírásával és a kiértékelés megismétlésével folytatódik. Nézzük a *listán*, miként történik ez.

A 100-as címen kezdődő szubrutin egy billentyű leütésére vár. Utána a leüött számnak megfelelő utasításokra ágazik el a program. Külön sorban kezeli a 0 választást, itt van az egyszerűsítési lehetőség. Az ON utasítás feltételének az *1/b listán* látható apró módosításával egy utasítást megtakaríthatunk. Természetesen az eredeti megoldás sem hibás, a felesleges zá-

rójelek és a „THEN GOTO” ellenére sem, csupán egy apró fogást kívántam bemutatni. Megfontolandó viszont az, hogy helyes-e a nullás menüpont ilyen módon való kezelése. Ugyanis a VAL függvény akkor is 0 értéket ad, ha véletlenül nem számbillentyűt nyomunk le, például ha az 1-es felé nyúlva véletlenül a Q-t nyomjuk le. Ezzel a kérdéssel itt nem akarok tovább foglalkozni, de nem árt elgondolkozni rajta.

A továbbiakban más programokból mutatok újabb példákat a számítógépes szöszaporításra. A *2. listán* a C16-ra és Plus/4-re készült *Helycsere* játék egy részlete látható, amely a *Mikrovilág* 1989/6. számában jelent meg. Ez a hat sor — más adatokkal — a programnak egy más helyén hasonló formában megismétlődik. Az egyszerűsítés módja nyilvánvaló: az IF utasítások feltételének megfordításával az ELSE ág feleslegessé válik, elhagyható. Ebből a program írója is megsejtethetett valamit, mert a lap következő, 1989/7. számában közölt *Lóugrás* című programjában már megfordította az IF feltételét. Úgy, ahogy azt a *3. listán* láthatjuk. Csak kötelezővé ne tegyék ezt a programozási módszert!

Érdekes ezekben a programokban a bevitt adatok kezelésének módszere is. Táblás játékokról lévén szó, a lépések kiindulási és célmezejének koordinátáit egy kétjegyű szám formájában kell megadni, melyet a programban egy öt utasításnyi rutin választ szét két egyjegyű számmá: a sor- és oszlopkoordinátára. A *Lóugrásban* ez a *4/a listán* látható módon történik, a *Helycsere*-ben egy kicsit bonyolultabban, mert ott a LEFT\$ függvény helyett MID\$ szerepel, viszont az utasításcsoport a programnak négy helyén is előfordul. A feladat sokkal egyszerűbben megoldható a *4/b listán* bemutatott módon.

Az itt említett játékprogramoknál számos más — a fentieknél jóval komolyabb — hibái is vannak, de azokra most nem kívánok kitérni. Talán egyszer még sor kerülhet rájuk.

Barna László

1/a lista

```
...
8200 GOSUB 100:ON VAL(A$)GOTO 8300,83
    50,8370,8390,8430
8210 IF VAL(A$)=0 THEN GOTO 3000
8220 GOTO 8000
...
```

1/b lista

```
...
8200 GOSUB 100 : ON VAL(A$)+1 GOTO 3000,
    8300,8350,8370,8390,8430
8220 GOTO 8000
```

2. lista

```
...
500 P=PEEK(3809):IF P=160 THEN 510:ELS
    E 330
510 P=PEEK(3813):IF P=160 THEN 520:ELS
    E 330
520 P=PEEK(3817):IF P=160 THEN 530:ELS
    E 330
530 P=PEEK(3653):IF P=160 THEN 540:ELS
    E 330
540 P=PEEK(3657):IF P=160 THEN 550:ELS
    E 330
550 P=PEEK(3497):IF P=160 THEN 560:ELS
    E 330
...
```

3. lista

```
...
200 CHAR = 26,22, "HOVA?":INPUT N:IF N=0
    0 THEN 300:ELSE 210
210 IF N=0 THEN 290:ELSE 220
220 IF N=0 THEN 290:ELSE 230
230 IF N=0 THEN 290:ELSE 240
240 IF N=0 THEN 290:ELSE 250
250 IF N=0 THEN 290:ELSE 260
260 IF N=0 THEN 290:ELSE 270
270 IF N=0 THEN 290:ELSE 280
280 IF N=0 THEN 290:ELSE 300
...
```

4/a lista

```
...
290 NS=STR$(N):LS=LEFT$(NS,2):L=VAL(LS)
300 RS=RIGHT$(NS,1):R=VAL(RS)
...
```

4/b lista

```
...
290 I=INT(N/10) : P=N-10*I
...
```


Programból alkalmazott parancsmód

A Commodore gépek tulajdonosai ismerik és szellemes programozási ötletként fel is használják számítógépüknek azt a tulajdonságát, hogy a billentyűpufferben több, korábban lenyomott billentyű kódját tárolja. Az Enterprise ezen a téren sem marad le.

Talán észrevették, hogy az Enterprise is „emlékszik” a korábban lenyomott billentyűre. Ez sokszor bosszantó is lehet, de ha céltudatosan felhasználjuk a gépek ezt a képességét, akkor hasznos és érdekes dolgokat művelhetünk vele.

Első kísérletként írjuk be a gépbe az *1. programot*. A semmitmondó program futása közben nyomunk meg egy tetszőleges billentyűt. A program lefutása és az „O.K.” megjelenése után a következő sorba az EDITOR kiírja a megnyomott karaktert. Ha igyekszünk és több billentyűt nyomunk le a program futása közben, akkor csak az utolsónak lenyomott egy billentyű által meghatározott karaktert írja ki a számítógép.

Ha a program futása közben az F2 funkcióbillentyűt érintjük meg, akkor az „O.K.” után a képernyőn megjelenik a program listája. Természetesen ha valamelyik másik funkcióbillentyűt nyomtuk le, akkor az általa meghatározott parancs kerül végrehajtásra.

Mód van arra is, hogy a próbaprogramunk futása közben egy funkcióbillentyűt és egy karaktert tartalmazó billentyűt is lenyomjunk egymás után. Ekkor mind a funkcióbillentyű által meghatározott szöveg-„blokk”, mind a „karakter” a képernyőre kerül a program lefutása után.

Az Enterprise tehát egy karakterre és egy funkcióbillentyű által meghatározott blokkra (szövegre) is emlékszik. Ezt a lehetőséget különböző célokra mi is felhasználhatjuk. Ehhez azonban ismerni kell néhány dolgot, így a pufferek helyét és tartalmát. De hogy ne legyen olyan egyszerű az életünk, ezek a területek a különböző kiépítésű gépek (angol, német) esetében máshol helyezkednek el:

	Német	Angol
Van-e karakter a pufferben	43909	48810
A karakter ASCII kódja	43913	48814
A blokk tárolási címe	43915/16	48816/17
A kijelölt blokk hossza	43917	48818

A táblázat megértéséhez és felhasználásához tudni kell még a következőket. A rendszerprogram angol gépeknél a 0. szegmens, a német gépek esetében (és ezen az UK átkapcsolás sem változtat) a 4. szegmensben lévő KEYBOARD kezelőrutint használja fel a szükséges lekerdezők és kódolások elvégzésére. Ha kiírásra, kiolvasásra karaktert tárol a gép, akkor azt a 43909 (48810) memóriacímre bejegyzett 255 dec értékel jelzi a rendszerprogram. Ha a puffer nem tárol karaktert (nem érkezett billentyűlenyomás), akkor e memóriacím tartalma 0.

A tárolt blokk hosszát, a megnyomott (átprogramozható) funkcióbillentyű által definiált karakterek számát a 43917 (48818) memóriarekesz tárolja. A funkcióbillentyűk maximálisan 23 karakter hosszú szövegre „taníthatók” meg. A funkcióbillentyűkhöz tartozó szövegek tárolása német gépeknél a 43525, angol gépeknél a 48426 memóriacímeken kezdődik. Az aktuális funkcióbillentyűhöz tartozó szöveg tárolási címét tartalmazza a 43915/16, illetve 48816/17 memóriarekesz.

Ennyi ismeret birtokában már értelmes feladatok megoldására is vállalkozhatunk.

A 2. program azt biztosítja, hogy a GET (vagy INKEYS) által beolvasandó karakter tényleg csak a felhasználás helyén érkező karaktert vegye figyelembe. A 150-es sor őríti a billentyűpuffert, így a korábban esetleg lenyomott billentyűre nem fog emlékezni a program.

A 3. program önmagát fejleszti tovább. Az első sorban levő változó segítségével tartja számon a mindig újrainduló prog-

Információk az Enterprise-piacról

Vállalatunk felmérést kíván készíteni az ENTERPRISE-tulajdonosokról, hogy a jövőben hatékonyabb támogatást nyújthasson a gépek minél sokoldalúbb kihasználásához. Kérjük a tisztelt géptulajdonosokat, hogy a mellékelt INFORMÁCIÓS ADATLAP-ot kitöltve szíveskedjenek szeptember 1-jéig vállalatunk címére beküldeni. (A postaköltséget a címzett fizeti.) Amennyiben ezt nem kívánják megtenni, úgy kérjük, hogy az INFORMÁCIÓS KÁRTYA-t küldjék be.

Azok között az ENTERPRISE-tulajdonosok között, akik beküldik a teljes ADATLAP-ot vagy az INFORMÁCIÓS KÁRTYA-t, az 1989. évi karácsonyi vásár alatt számos értékes számítástechnikai ajándékot sorsolunk ki.

Levél cím: CENTRUMNAGYKER
1431 Budapest, Pf. 195.

EP-INFO INFORMÁCIÓS KÁRTYA EP-INFO

Név:

Az EP-tulajdonos címe: irsz
..... helység
..... Pf.
..... utca házszám
Az ENTERPRISE gép típusa: (X) német () angol ()

EP-INFO INFORMÁCIÓS ADATLAP EP-INFO

(Az adatlap kitöltése és továbbítása nem kötelező!)

1. Az ENTERPRISE 128 k számítógép

tulajdonosának neve:

Lakcíme: irsz helység
..... Pf.
..... telefon házszám emelet

- A gép gyári száma: ()
(a gyári szám a gép alaplapjának egy fehér öntapadós négyzetén található)
- A gép típusa (X) német () angol ()
- Adatmagnó (DR) típusa: (X)
ENTERPRISE () OMEGA () CLOUD () PHILIPS ()
- Egyéb ()
- A BASIC INTERPRETER cartridge-on lévő típusjel: (X)
üres () fehér pötty () piros pötty ()
zöld pötty () fehér nyíl ()
- Kérem, jelölje meg, hogy az alábbi hardverkiegészítők közül melyikkel rendelkezik? (X)
egér () „SPEAK EASY” () SPECTRUM EMULATOR kártya () EX-DOS lemezvezérlő kártya () EP-PLUS CARTRIDGE () nyomtató () külső botkormány () CBM MFT () televízió computer-átkapcsoló ()
- Ha van mágneslemezegysége, akkor az mekkora kapacitású? (kbájt) (X)
nincs () 180 kbájt () 360 kbájt () 720 kbájt ()
egyéb () kbájt

ram, hogy milyen sorszámmal vegye fel az újabb DATA sorokat. Itt most az INPUT sorban beírt 9-9 adatot (lehetne ez egy karakterátdefiniáló program része) írja be egy DATA sorba a program.

Még futása közben átdefiniálja az F2-t, és a megfelelő változók átírásával olyan jelzést ad a gépnek, mintha lenyomták volna azt. Az F2 tartalma most azt biztosítja, hogy a kurzor a képernyő bal felső sarkába ugorjon, majd két ENTER kerüljön végrehajtásra a program befejeztével parancsmódban.

A program befejezése előtt a megfelelő szintaktikai szabályok szerint kiírattuk a képernyőre a programba beépített programsort, alá pedig egy RUN szöveget. Az ezt követő END hatására a gép parancsmódba kerül, és végrehajtja a megnyomtnak tekintett F2 által kijelölt kurzormozgatást és soremeléseket. Ahogy a RUN szövegen az ENTER hatására „áthalad” a kurzor, a program újraindul, de már az előbb szerkesztett sort is tartalmazza.

Ennek az eljárásnak egy másik alkalmazási lehetőségét mutatja be a 4. program. Ebben INPUT-ként beírt függvény-sztringet értelmeztünk, mint matematikailag értelmezhető kifejezést. Az eljárás itt is az előzőhöz hasonló. A menü adata lehetőséget kihasználva, az első ágat választva (a BASIC szabályai szerint) beírhatjuk a kívánt függvényt. Ezután a program programsorként (sorszám, DEF) írja ki a képernyőre a sztringet és a RUN szöveget. Az átdefiniált F2 — lehetne természetesen akár-

melyik másik is — és rendszerváltozó-értékek hatására felírja a programot és újraindítja azt. Most már a második menüág alapján beírhatjuk a függvény értelmezési tartományát, mire a program kiírja az értéktáblázatot.

A vállalkozó kedűek persze akár ábrázolást is megvalósíthatnak a mindig megfelelő függvénykapcsolatot tartalmazó program segítségével.

A további kísérletezési lehetőségeknél vigyázni kell, hiszen beavatkoznunk a gép lelkivilágába. A sorszámfelírás, a puffertartalom átírása az adott géptípustól, a programszámtól függ, és ha valamit megdölgődtlenül csinálunk, akkor programunk tönkremeget.

Sz. Lukács János

1. program

```
10 : billentyűpuffer ellenorzes neset
20 FOR I=1 TO 100
30 PRINT PEEK(43913);
40 NEXT
```

2. program

```
100 'karakterpuffer ertes neset
110 FOR I=1 TO 50
120 IF PEEK(43909)<255 THEN PRINT "pufferben van"
130 IF PEEK(43909)=0 THEN PRINT "puffer ures"
140 NEXT
150 PEEK(43909)=!angol gepen 48910
160 PRINT "ezzel puffer ures"
170 GET A$
180 IF A$="" THEN GOTO 170
190 PRINT A$
```

3. program

```
1 LET Q=1
100 ' program feluliras
110 ' DATA sor elallitolo
120 '
130 GOTO 240
140 TEXT
150 PEEK(4837);PEEK(4837)+1
160 PRINT PEEK(4837);"DATA "
170 FOR I=1 TO 9
180 PRINT STR$(I);";";
190 NEXT
200 PRINT CHR$(164)
210 PRINT "RUK"
220 SET FKEY 2 CHR$(177);CHR$(13);CHR$(13)
230 ' angol POKE 48910,66;POKE 48910,3
240 POKE 43915,29;POKE 43917,3 'meset
250 END
260 !
270 CLEAR FREEY
280 ' a polda kedvoert
290 DIM B(9)
300 TEXT
310 FOR I=1 TO 9
320 PRINT I;"adat="
330 INPUT B(I)
340 NEXT
350 GOTO 140
```

4. program

```
100 ' program feluliras
110 ' programozott parancsnod
120 '
130 GOTO 230
140 INPUT PROMPT "y=":Y$
150 TEXT
160 PRINT "250 def y(x)";F$
170 PRINT "260 f(x)";"";F$;"
180 PRINT "run 230"
190 SET FKEY 2 CHR$(177);CHR$(13);CHR$(13);CHR$(13)
200 ' angol POKE 48910,66;POKE 48910,4
210 POKE 43915,29;POKE 43917,4 'meset
220 END
230 !
240 CLEAR FREEY
250 "!!!!!!!!!!!!!!!!!!!!!!"
260 "!!!!!!!!!!!!!!!!!!!!!!"
270 TEXT
280 PRINT " " ; menu";PRINT
290 PRINT "1. u. j. fuggvny"
300 PRINT "2. erteskeslet"
310 PRINT "3. gyokok"
320 PRINT "4. abraolias"
330 PRINT "5. hatarozott integral"
340 GET V$
350 IF V$="5" OR V$="" THEN GOTO 340
360 ON VAL(V$) GOTO 140,430,520,530,540
370 ' értelmezési tartomány
380 PRINT "y=";F$;PRINT
390 INPUT PROMPT "xmin=":XA
400 INPUT PROMPT "xmax=":XF
410 INPUT PROMPT "t=epelos=":DX
420 RETURN
430 ' erteskeslet
440 GOSUB 370
450 FOR X=XA TO XF STEP DX
460 PRINT X;Y(X)
470 NEXT
480 PRINT "bill nyomra tovabh."
490 GET V$
500 IF V$="" THEN GOTO 490
510 GOTO 270
520 'gyokok
530 'abraolias
540 'hatarozott integral
550 PRINT "PRINT "valamit te is csinall!"
560 GOTO 490
```

8. Eredeti gyári kazettáinak száma (játék):

9. Másolt programjainak száma (játék):

10. Felhasználói, illetve utility programjainak száma:

11. Az ENTERPRISE-zal foglalkozó könyveinek száma:

12. Véleménye szerint melyik áruház a legjobban ellátott?

13. Hol vannak Ön szerint a legképzettebb eladók?

14. Ön szerint mi az ENTERPRISE számítógép leggyengébb hardver része?

15. Véleménye szerint mi az ENTERPRISE számítógép leggyengébb szoftvereleme?

16. Milyen elemekkel bővítené még a szoftverválasztékot?

17. Milyen kiegészítőkkel és tartozékokkal bővítené még a hardverválasztékot?

18. Milyenek ítéli a játékprogramok árfekvésését? (X):
olcsó () megfelelő () drága () irreálisan magas ()

19. Milyenek ítéli a felhasználói programok árfekvésését? (X):
olcsó () megfelelő () drága () irreálisan magas ()

20. Milyenek ítéli a hardverkiegészítők árfekvésését? (X):
olcsó () megfelelő () drága () irreálisan magas ()

21. Kérjük, állítsa össze az ENTERPRISE játékprogramok sikerlistáját! (A felsorolásban a Public Domain programok is szerepelhetnek)

EP FUN CLUB'S HOT LINE 10

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.

22. Kérjük, állítsa össze az ENTERPRISE felhasználói, illetve utility programok sikerlistáját!

EP USER'S CLUB CHART

1.
2.
3.

FEJ VAGY ÍRÁS

A közismert, éremmel játszott szerencsejátékot szimulálja Vicsotka Gyula olvasónk programja. A játék lényege, hogy előre meghatározott téttel játszva az a játékos nyer, aki eltalálja, hogy a feldobott érme melyik oldalára esik. A programmal kitűnően érvényesülnek a gép animációs lehetőségei. Jó szerencsét!

```

1 PROGRAM "FEJ_VAGY_IRAS"
2 REM %zitetto:Vicsotka Gyula
3 RANDMIZE
4 WHEN EXCEPTION USE HIBA
5 LET KERT=0
6 TEXT 40
7 PRINT AT 10,10:"KIS TURELMEZ KER
EK !"
8 SET VIDEO X 12:SET VIDEO Y 15
9 SET VIDEO MODE 1:SET VIDEO COLOR
0
10 FOR I=10 TO 17
11 OPEN #1:"video:"
12 SET #1:PALETTE BLACK,YELLOW
13 NEXT
14 PLOT #10:132,0;252,0;252,16;132,
16;132,0
15 PLOT #11:192,50,ELLIPSE 60,20,PA
INT
16 PLOT #11:192,62,ELLIPSE 60,20,
17 SET #11:INK 0
18 PLOT #11:192,50,ELLIPSE 20,10,PA
INT
19 PLOT #12:192,134,ELLIPSE 60,40,P
AINT
20 PLOT #12:192,146,ELLIPSE 60,40,
21 SET #12:INK 0
22 PLOT #12:192,134,ELLIPSE 30,20,P
AINT
23 SET #12:INK 1
24 PLOT #12:192,134,ELLIPSE 15,8,PA
INT
25 PLOT #13:192,270,ELLIPSE 60,60,
26 PLOT #13:192,270,ELLIPSE 50,50,
27 PLOT #13:192,325,PAINT
28 PLOT #13:192,270,ELLIPSE 20,24,
29 PLOT #13:216,238;192,246;168,238
30 PLOT #13:199,275
31 PLOT #13:185,275
32 PLOT #13:189,252;199,252
33 PLOT #13:176,280;207,280
34 PLOT #13:192,285,PAINT
35 PLOT #13:214,270;214,272
36 PLOT #13:170,272;170,270
37 PLOT #13:192,270;192,262
38 PLOT #14:192,270,ELLIPSE 60,60,
39 PLOT #14:192,270,ELLIPSE 50,50,
40 PLOT #14:192,325,PAINT
41 PLOT #14:185,295;199,295;199,240
;185,240;185,280;175,268;165,264;185,295
42 PLOT #14:192,270,PAINT
43 PLOT #15:192,402,ELLIPSE 60,40,P
AINT
44 PLOT #15:192,390,ELLIPSE 60,40,
45 SET #15:INK 0
46 PLOT #15:192,402,ELLIPSE 40,30,P
AINT
47 SET #15:INK 1
48 PLOT #15:192,380;192,420;175,410
49 PLOT #16:192,490,ELLIPSE 60,20,P
AINT
50 PLOT #16:192,478,ELLIPSE 60,20,
51 SET #16:INK 0
52 PLOT #16:192,490,ELLIPSE 40,10,P
AINT
53 PLOT #17:132,539;252,539;252,523
;132,523;132,539
54 CLEAR SCREEN
55 SOUND PITCH 40,DURATION 30,
56 INPUT PROMPT "Mennyi penzt kersz
? ",AT 10,2:TOKE
57 INPUT PROMPT "Milyen alapon jats
zunk ? ",AT 15,0:ALAP
58 IF ALAP>TOKE THEN 54

```

```

59 LET KERT=KERT+TOKE
60 SET VIDEO X 40:SET VIDEO Y 1
61 SET VIDEO MODE 0
62 OPEN #1:"video:"
63 OPEN #2:"video:"
64 OPEN #3:"video:"
65 SET #1:CORSOR CHARACTER 32
66 SET #2:CORSOR CHARACTER 32
67 SET #3:CORSOR CHARACTER 32
68 CLEAR #1
69 PRINT #1,AT 1,1:"Penzed: ";TOKE,"
Alap: ";ALAP;"
70 DISPLAY #1:AT 19 FROM 1 TO 1
71 DISPLAY #3:AT 18 FROM 1 TO 1
72 SOUND PITCH 40,DURATION 15,
73 PRINT #3,AT 1,1:"Kilépés (ESC)"
;
74 PRINT #2,AT 1,1:"Mire teszel ? (
f,1)";
75 DISPLAY #2:AT 21 FROM 1 TO 1
76 DO
77 GET #F
78 LOOP WHILE #F="" OR #F<>"1" AND
#F<>"f" AND #F<>CHR$(27)
79 IF #F=CHR$(27) THEN 120
80 SOUND PITCH 40,DURATION 15,
81 IF #F="f" THEN LET TET=13
82 IF #F="1" THEN LET TET=14
83 LET N=NRND(2)+13
84 FOR I=10 TO 17
85 DISPLAY #1:AT 1 FROM 1 TO 15
86 FOR T=0 TO 50
87 NEXT
88 NEXT
89 FOR I=17 TO 15 STEP-1
90 DISPLAY #1:AT 1 FROM 1 TO 15
91 FOR T=0 TO 50
92 NEXT
93 NEXT
94 DISPLAY #N:AT 1 FROM 1 TO 15
95 IF #T=TET THEN
96 LET TOKE=TOKE+ALAP
97 ENVELOPE NUMBER 1;6,63,63,40;-
3,-63,-63,40
98 SOUND PITCH 40,DURATION 80,ENV
ELOPE 1
99 END IF
100 IF N>TET THEN
101 LET TOKE=TOKE-ALAP
102 ENVELOPE NUMBER 2;10,63,63,40;
16,-63,-63,40
103 SOUND PITCH 40,DURATION 80,ENV
ELOPE 2
104 END IF
105 IF TOKE=ALAP THEN 68
106 IF TOKE=0 THEN LET ALAP=TOKE
107 PRINT #2,AT 1,1:"Elfogyott a pen
zed! Jatszol meg? (y,n)";
108 DO
109 GET #F
110 LOOP WHILE #F="" OR #F<>"1" AND
#F<>"n"
111 IF #F="n" THEN 117
112 FOR I=10 TO 17
113 CLOSE #1
114 NEXT
115 CLOSE #1:CLOSE #2:CLOSE #3
116 GOTO 6
117 CLEAR #1
118 CLEAR #2
119 CLEAR #3
120 IF KERT<=TOKE THEN PRINT #1,AT 1
,1:"Nyertel!";TOKE-KERT;"forintot!";
121 IF KERT>TOKE THEN PRINT #1,AT 1,
1:"Vesztetel!";KERT-TOKE;"forintot!";
122 PRINT #2,AT 1,1:"BASIC-be (v), R
as jatekos (u) ";
123 LET C=10
124 DO
125 GET #F
126 DISPLAY #C:AT 1 FROM 1 TO 15
127 LET C=C+1
128 IF C=18 THEN LET C=10
129 FOR I=0 TO 40
130 NEXT
131 LOOP WHILE #F="" OR #F<>"v" AND
#F<>"u"
132 IF #F="u" THEN RUN
133 EXT "BASIC"
134 END WHEN
135 HANDLER HIBA
136 RETRY
137 END HANDLER

```


Hol az INPUT? Itt az INPUT!

A rutin, melynek listáját közöljük, a képernyő általunk meghatározott részén kérdez meg a használatól adott hosszúságú szöveget. Ezzel nemcsak látványosabbá tehetjük programunkat, hanem a begépelés hibalehetőségeit is csökkenthetjük.

A rutint CALL INPUT (x, y, hossz, b\$) utasítással hívhatjuk meg. Nem működik a DEL, az INS billentyű és a kurzormozgató. Kilépní az ENTER-rel lehet, ekkor a hívó programban térünk vissza a beolvasott b\$ sztringgel. Törlésre az ERASE billentyű használható.

A 1160 és 1170 számú sorokban az ASCII kódhatárok beállításával változtathatjuk, hogy mely karaktereket fogadják el. Jelenleg a kis- és nagybetűket, a számokat, a vesszőt, a kötőjelet, a pontot, azaz a 44—126 közé eső kódú karaktereket.

A rutin leírása

- 1010 a segédváltozó törlése
- 1020—1040 a beolvasás, ill. kiírás helyére pontokat ír
- 1070 a kurzor pótlására szolgáló nem villogó négyzet kiírása
- 1080—1100 a billentyűzet figyelése
- 1110 ha az ERASE-zel sokat törölünk, azaz DB kisebb lesz mint 0, kezdjük előlről
- 1120—1140 ha a teljes hosszúságú beolvasott sztring végső karaktere nem az ENTER, akkor kezdjük előlről
- 1150 ha az utolsó karakter az ENTER, akkor ugrás a program végére
- 1160—1170 az elfogadott karakterek kiválasztása
- 1180—1230 ha a lenyomott billentyű az ERASE, akkor a kiírt CRSR törlése és egy karakter törlése. Új CRSR és karakter kiírása
- 1250 az új karakter hozzáadása az eredmény-sztringhez

Piros András

```

10 !INPUT RUTIN
20 !HIVAS CALL UTASITASSAL
30 !KESZITETTE:
40 !PIROS ANDRÁS
50 !DEBRECEN
1000 DEF INPUT (X,Y,HOSSZ,REF B$)
1005 CLEAR SCREEN
1010 LET G$=""
1020 FOR I=0 TO HOSSZ-1
1030 PRINT AT X,Y+I: ". ";
1040 NEXT
1050 LET DB=-1:LET B$=""
1060 LET DB=DB+1
1070 PRINT AT X,Y+DB:CHR$(159)
1080 DO
1090 LET D$=INKEY#
1100 LOOP WHILE D$=""
1110 IF DB<0 THEN GOTO 1010
1120 IF DB=HOSSZ THEN
1130 IF D$>CHR$(13) THEN GOTO 1020
1140 END IF
1150 IF D$=CHR$(13) THEN GOTO 1270
1160 IF D$<CHR$(44) OR D$>CHR$(164) T
HEN GOTO 1080
1170 IF D$=CHR$(126) AND D$<CHR$(164)
THEN GOTO 1080
1180 IF D$=CHR$(164) THEN
1190 PRINT AT X,Y+DB: "
1200 LET DB=DB-1:LET G$=G$+D$
1210 LET B$=G$
1220 GOTO 1070
1230 END IF
1240 PRINT AT X,Y+DB:D$;
1250 LET B$=B$+D$
1260 GOTO 1060
1270 FOR I=DB TO HOSSZ
1280 PRINT AT X,Y+I: "
1290 NEXT
1300 END DEF
1400 !
    
```

Mi a manó?

Igen sok olvasónk érdeklődött, hogyan tudja levédeni programját. A kérdések nem hivatásos programkészítőktől érkeztek, ami természetes is. Ők ismerik a különböző módszereket! Mi azonban elgondolkoztunk, vajon miért érdeklő az amatőrök ez a kérdés. Úgy gondoljuk, kíváncsiságból. Ha igazán hatékony módszerekről nem tudunk ismertetőt adni, mert azok legalább olyan titkosak, mint maga a program, amelyet védenek velük.

Török János leírt néhány fogást BASIC program levédésére. Ezeket közkinccsé tesszük.

A STOP billentyűt a SET INTERRUPT STOP OFF utasítással lehet letiltani, vagy lezárjuk a 105-ös csatornát (lásd CLOSE).

A POKE 111140,1 utasítás hatására a program tovább fut, de egyetlen RESET-re is újraindul. Ha ezeket az utasításokat programunkba beépítettük, akkor azt csak úgy tudjuk újraindítani, ha már futott, hogy letöröljük és betöltjük. Ha ellenben a program nem indult el, akkor egyszerűen ki lehet listázni. Ezen már nehezebb segíteni.

Az első sor sorszámát a 4828 és 4825 címeken tárolja a gép. Vigyázat, az ALLOCATE utasítás hátrább csúsztatja a tárban!

Kijátszhatjuk az illetéktelen programolvasót, ha az első sor sorszámát nagyobbra állítjuk, mint 10000. Ezt nem lehet kilistázni, de el lehet menteni és kilistázni. Az illetéktelen beavatkozó a DEF-függvényeket sem tudja hívni a sorszámukkal. Ez a levédés sem tökéletes, mint ahogy egyik sem az, mert a programot a LIST FIRST vagy a LIST LAST paranccsal kilistázhatjuk.

Jó megoldás az „önindító”, de ezt BASIC nyelven nem lehet megoldani.

Reméljük, hogy olvasóink, leendő szerzőink nem védett programokat küldenek be szerkesztőségünkbe.

Várjuk rovatunkba további ötleteiket, tanácsait.



Gaetsch Günterné rajza

Hardver

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot számmázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozók

rendelkeznie kell alapfokú áramkörtudással és hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretekre.

A hardver élesztése és a monitorprogram

A mikroprocesszoros rendszerek fejlesztésének legkritikusabb szakasza az élesztés. Egy új rendszerben egyszerre több hiba is lehet, ezért az élesztést több lépésben célszerű végrehajtani. A rendszer részekre bontását és az egyes részek fokozatos bekapcsolását megfelelő logikai sorrendben kell elvégezni.

Az elkészült (gyártott) nyomtatott áramkört lapot először vizsgáljuk át szabad szemmel és nagyítóval. Nézzük végig, hogy nincs-e áramkört zárlat vagy fölhasználás a lapon. Ezután a tápfeszültség-vezetékét bekötve győződjünk meg arról, hogy nincs-e tápzárlat a kártyán, azaz nulla-e az áramfelvétel. Az integrált áramköröket — prototípusnál vagy kísérleti stádiumban lévő fejlesztésnél — célzerűen foglaljuk helyezni, ezért a következő lépés az óra- és RESET-áramkörökhöz tartozó elemek és az U1, U4 jelű IC-k foglalatának a beforrasztása. A 74LS04 áramkör foglalatba helyezése után az U1/6 lábán megjelenik az órajel, a K1 kapcsolót működtetve az U1/26 lábán alacsony, elegendő magas jelszintet kell érzékelnünk.

A működő hardverbe ezek után csupán azt a programot tartalmazó EPROM-ot kell bedugni, ami az adott feladat megoldását biztosítja, és a fejlesztés kész.

De hát éppen ez az előbbi az egész fejlesztés egyik pillére, és ezt a lépést végezzük el a fejlesztőrendszerrel. A sorozat előbbi részeiben a fejlesztéssel kapcsolatos kérdésekkel — éppen a fontosságuk miatt — részletesen foglalkoztunk, így most csupán a

jelenlegi viszonyokat és lehetőségeket figyelembe vevő, általunk kidolgozott megoldást mutatjuk be.

Hogyan is történik a tényleges fejlesztés? Az assemblerben megírt és lefordított, összeszerkesztett programot EPROM-ba írjuk (égetjük), majd az így előkészített tokot a fejlesztendő rendszerünkben lévő foglalatba helyezzük, és bekapcsoljuk a rendszert. A legtöbb esetben programunk elindul, de nem jól vagy egyáltalán nem működik, és kereshetjük a hibát.

A program futását a legegyszerűbben a processzortokra csipetett állapotanalízissel követhetjük, ilyenkor azonban általában nincs! Ezért célszerűen olyan környezetet kell kialakítanunk, amelynek egyes komponenseit maga a fejlesztendő rendszer tartalmazza, és a másik része is viszonylag egyszerűen megvalósítható.

Maga a rendszer egy IBM PC számítógépből és a hozzá soros vonalon kapcsolódó fejlesztendő rendszerből áll. A fejlesztendő rendszer induláskor egy olyan EPROM-ot tartalmaz, amelyben egy, a programfuttatást segítő monitorprogram van. Ez a program biztosítja, hogy az IBM PC-n fejlesztett programunkat hexadecimális formában írjuk át a fejlesztendő rendszerünk RAM-jába.

Az IBM PC szerepe kettős. *Fejlesztői üzemmódban* a program írható, szerkeszthető, fordítható, linkelhető, majd a tárgykód hexadecimális formában a rendszerbe átirható, és monitor üzemmódban futtatható. *Monitor üzemmódban* a PC a monitor termináljaként működik. A monitornak rajta keresztül adhatók parancsok, a

monitor válaszai pedig a képernyőn jelennek meg.

A rendszerhez természetesen egy EPROM-égető készülék is tartozik, amellyel a végső fázisban beégethetjük a RAM-ban kipróbált programunkat.

A kapcsolatot az IBM PC-vel a fejlesztendő rendszerbe bedugható soros illesztőkártyán keresztül valósítjuk meg. A kapcsolós elemek: egy Intel 8251-es soros periféria-áramkör, egy MC14411-es oszcillátor és baudrate generátor-áramkör, a TTL (+5 V) — RS232 (+12 V, -12 V) szintek illesztését végző MC1488 és MC1489-es áramkörök. A +12 V, -12 V feszültségeket külön tápegység állítja elő. Az illesztőben lévő 8251-es áramkör programozását és kezelését az EPROM-ban lévő monitorprogram egyik része végzi.

A monitor az alsó 32 kb-ot tartalmazó EPROM alsó részén helyezkedik el (0000H—1FFFH), feladata a programbeolvasás.

Az IBM PC mint terminál

A fejlesztőrendszerhez soros vonalon hozzákapcsolt IBM PC/XT számítógép szolgál a monitor termináljaként. A PC-n a TERMINAL.EXE programot kell meghívni, ez vezérli a fejlesztőrendszerrel a kommunikációt. A soros adatátviteli vonalat a TERMINAL program kezeli, és a szükséges karakterkonverziókat is elvégzi. Meghívását követően listát ad a funkció-

billentyűk szerepéről. A "TEXT Vx.x" bejelentkezés után a monitor promptja (".") jelentkezik a képernyőn. Ilyenkor lehet az egybetűs parancsokat beadni, melyek (ha érvényesek) a képernyőn parancsszóra kiegészítve láthatók. Az esetleg szükséges paraméterek beadása után az ENTER billentyűvel lehet a parancsokat terminálni. Az egyes paraméterek között space szolgál elválasztójelként. A számokat hexadecimálisan kell beadni, a program az utolsó négy (illetve két) digitet veszi figyelembe. Szintaktikusan hibás parancs kiadásakor a monitor kérdőjelet ír ki és hangjelzést ad.

- F1: Dir — a DOS dir parancsának felel meg. A "Dir:" kérdésre a szükséges directory-maszkot kell beírni.
- F2: Del — a DOS del (erase) parancsával egyenértékű. A "•Delete" felirat után megjelenő "Filename:" kérdésre a törölni kívánt fájl nevét kell beírni.
- F3: Chdir — a DOS chdir (cd) parancsához hasonlóan a "Path:" kérdésre a kívánt útvonalat kell beírni.
- F4: Load — segítségével Intel hexa formátumú fájlt lehet áttölteni a tesztgép RAM-jába. A "•Loader" felirat után megjelenő "Filename:" kérdésre a fájl nevét kell beírni (xxxxxxx.HEX). A fájl bináris formában betöltődik a rekordokban adott címinformációknak megfelelő memóriaterületekre. Az áttöltés alatt a képernyőn a memóriacímek jelennek meg, az áttöltés végén hangjelzést ad a program.
- F5: Shell — ideiglenesen kiléphetünk a TERMINAL programból és visszatérhetünk a DOS-ba. Az "EXIT" parancs beírásával térhetünk vissza a TERMINAL programba.
- F10: Exit — a TERMINAL programból való kilépés kezdeményezése. A képernyőn megjelenő "Are you sure..." kérdésre adott Y válasz esetén a DOS-ba térünk vissza, N válasza a program ignorálja a kilépési kérést.

A funkcióbillentyűk kizárólag a TEST-

MONITOR futása közben nem. A TERMINAL program a kisbetű—nagybetű konverziót végzi el, így a monitorparancsok kisbetűkkel is hívhatók. A program a főbb hibajelenségek esetén szöveges hibajelentést ad (például "file not found"), egyéb hibáknál "Error:xx" üzenet jelenik meg. A hibák magyarzata a Microsoft QuickBasic v4.00b leírásában található, a TERMINAL program ugyanis ezzel a fordítóval készült.

A TERMINAL program a DOS parancssorában megadott paraméterekkel is hívható. A parancs alakja: `TERMINAL [baudrate] [load_file_name]` ahol baudrate a soros vonal adatátviteli sebessége, load_file_name az F4 funkcióbillentyű kiválasztásával indított "load" funkció során betöltendő Intel hexa fájl neve. Ha a baudrate-et megadjuk, akkor az erre vonatkozó kérdést a program nem teszi fel, ha a load_file nevet is megadjuk, akkor az első F4 parancsnál nem kérdezi meg a hexa fájl nevét. Ha bármely parancssorbeli paramétert megadjuk, a program a funkcióbillentyűk listáját nem fogja kiírni.

A monitor a főmenüből a TEST-MONITOR parancs kiválasztásával

(F) hívható meg. A "TEXT Vx.x" bejelentkezés után a monitor promptja (".") jelentkezik a képernyőn. Ilyenkor lehet az egybetűs parancsokat beadni, melyek (ha érvényesek) a képernyőn parancsszóra kiegészítve láthatók. Az esetleg szükséges paraméterek beadása után az ENTER billentyűvel lehet a parancsokat terminálni. Az egyes paraméterek között space szolgál elválasztójelként. A számokat hexadecimálisan kell beadni, a program az utolsó négy (illetve két) digitet veszi figyelembe. Szintaktikusan hibás parancs kiadásakor a monitor kérdőjelet ír ki és hangjelzést ad.

Az alábbiakban ismertetjük a monitor parancsait. A <> jelek közé zárt adatok paramétereket jelentenek, a [] opcionális paramétert jelent. B — BREAK <addr>: töréspont elhelyezése a tesztelendő, RAM-ban levő programban. Töréspontot természetesen csak az egyes utasítások első bájtvára szabad elhelyezni. A monitor az RST 6 utasítás kódjával helyettesíti az <addr>-en lévő bájtot. A program a G (goto, lásd később) parancsral elindítva a töréspontot áthaladva megáll, a monitor pedig visszaállítja az eredeti bájrt értékét a programban, és kiírja a regiszterek tartalmát. A képernyőn az alábbi két sor jelenik meg:

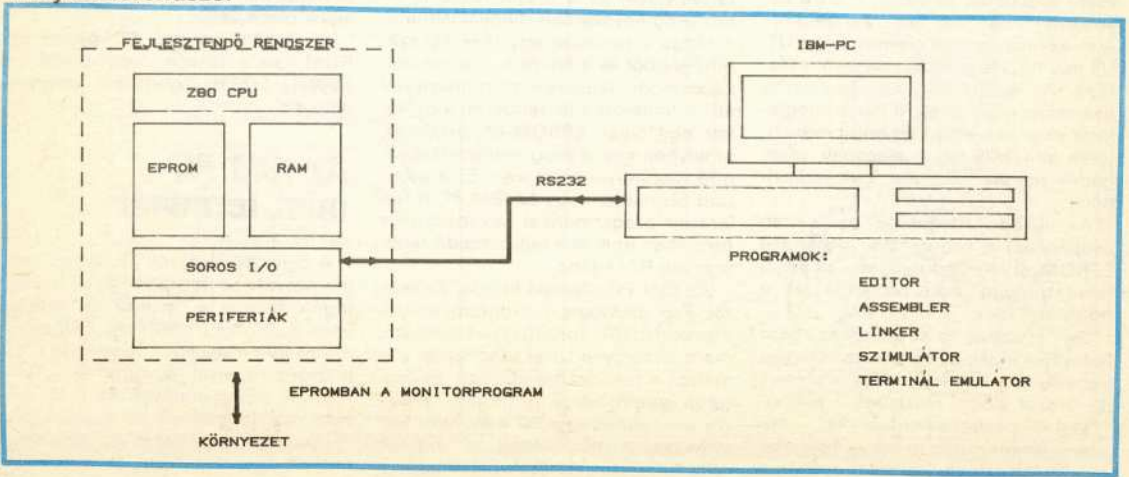
```
* <addr>
PSW = xxxx          BC = xxxx
DE = xxxx HL = xxxx SP = xxxx
ahol xxxx az aktuális regiszter(pár) értéke a töréspont előrésekor. A PSW az akkumulátor és a flageket jelenti. A re-
```

A tesztmonitor

A tesztmonitor a TMON v3.2 felhasználásával készült, abból egyes funkciókat elhagytunk, másokkal a fejlesztési munka igényeinek megfelelően kiegészítettük. A monitor mérete kb. 2,5 kb-át.

A monitor a főmenüből a TEST-MONITOR parancs kiválasztásával

A fejlesztőrendszer



lentyűk segítségével mozoghatunk a listában, és az egyes bájtokat meg is változtathatjuk. A "<" illetve ">" billentyűkkel a monitor 128 bájttal előbb, illetve hátrább kezdődő memóriaterületeket listáz ki. Az ENTER billentyűvel a monitor promptjához térhetünk vissza.

F — FILL <from> <to> <with>: memóriafeltöltés. A <from> kezdőcímtől a <to> végcímig a <with> (bájt) értékkel tölti fel a memóriát.

G — GOTO <addr>: programfuttatás. A monitor a PSW, BC, DE, HL és SP regisztereket az R (registers, lásd később) paranccsal megadott értékekkel feltölti, majd az <addr> címre adja át a vezérlést. Mivel a tesztelendő program felhasználója vermet (stack) használ, amely beállítható valamilyen értelmes értékre, ezért a monitorba való visszatérés a RET utasítással nem lehetséges, erre a célra az RST 1 utasítást kell használni.

I — INP <port>: adott című port beolvasása és a beolvasott adat kiírása hexadecimális és bináris alakban. Ha a port beírása után az ENTER helyett space-t írunk, akkor a beolvasás és kiírás végtelen ciklusban ismétlődik. Ezt az állapotot csak RESET-tel szüntethetjük meg.

O — OUT <port>: az adott című portra bájt(ka)t visz ki. Ha a <port> megadása után RETURN-t ütünk, akkor egyszerű kivitelt következik, míg space-szel folyamatos üzemmódot választatunk. Ekkor a monitor kiírja: "(C.)", ami a folyamatos üzemmódra utal. Ezután a kívendő bájt(ka)t kell megadni, egy bájt beírása után space-t ütve a monitor további bájtok megadására vár, RETURN esetén pedig befejezzük a parancsot. Folyamatos formát választva, az ENTER hatására "." jelenik meg a képernyőn, ami az utolsó bájt végtelen ciklusban történő kivitelt jelzi. Ezt az állapotot csak RESET-tel szüntethetjük meg.

R — REGISTERS: a processzor regisztereinek vizsgálata, illetve módosítása. A "REGISTERS" felirat megjelenése után új sorban a "PSW=xxxx-" jelenik meg, ahol "xxxx" az akkumulátor és a flagek értéke, a "-" pe-

dig jelzi, hogy új értékre vár a monitor, vagyis a regiszterek értéke megváltoztatható. Ha nem akarunk változtatni, és ki akarunk lépni a parancsból, akkor ENTER-t kell ütni, ha további regiszterekre is kíváncsiak vagyunk, space-t kell ütni. Ekkor a BC, DE, HL, SP értékei jelennek meg, amelyek szintén megváltoztathatók. Az értékek a G (goto) parancs kiadásakor kerülnek ténylegesen a processzor regisztereibe.

U — UNASSEMBLE <addr>: a megadott címtől disassembly

listát jelenít meg a képernyőn. A lista formátuma: cím hexa bájt(ok) mnemonik — argumentum(ok)

A következő sor bármilyen billentyű (kivéve a CNTRL-C) lenyomására jelenik meg. A CNTRL-C billentyű a parancs terminálását jelenti.

V — VERIFY <from> <to>: a memóriatartomány ellenőrző összegét számítja ki. A <from> és a <to> memóriacímek közötti tartományban lévő bájtok 16 bite csonkított összegét írja ki "SUMMA:xxx" formában.

Dr. Kónya László

Zárszó

Eljutottunk a sorozat végére. Elbizakodottság lenne azt hinni, hogy csupán az eddigiek alapján, az áramkörtől a rendszer építőelemeinek ismertetésén keresztül eljutva a mikrogéphez, a szorgalmasabb, minden részt elolvasó érdeklődő mikrogépes fejlesztésbe foghatna. A sorozat nem is ezt tűzte ki céljává. Az elsődleges cél annak a megmutatása volt, hogy az ilyen rendszerekben nincs semmi misztikus, fejlesztésük különleges képességeket nem igényel.

A sorozat olvasói — reméljük — azt is felismerték, hogy nem szabad élesen különválasztani a szoftvert és a hardvert, mert ezek a rendszerekben integrálva jelennek meg. A fejlődés a szoftver részarányának növekedése irányába mutat, de a mikroprocesszoros berendezések legújabb áramkörökön keresztül kapcsolódik a külvilághoz.

Nem szögtünk arról, hogy az ilyen mikrogépes rendszereket milyen célokra lehet felhasználni. De ezt a téma szerteágazó volta miatt egy lap hasábjain csaknem reménytelen is lenne összefoglalni. Legfeljebb az olvasók fantáziájában...

*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 kb-ig
- 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.*

BÖRZE



COBRA COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1087 Budapest IX., Illetve út 7. Telefon: 476-160/388

SZÁMLAKÉSZÍTÉS
A KÖNYVELESIG

COBRACONTO

Ügyviteli programrendszer modullei:

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

KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

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

ASY 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 árucikkrel 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

The Norton Guides

Segédinformációk gyors megjelenítése a képernyőn

Melyik programírónak nem volt olyan problémája, hogy programozás közben egy utasítás pontos szintaxisát nem tudta fejből, és különféle programozói kézikönyvekben kellett utánanéznie? Vagy nem tudta egy adott karakter rendszerbeli billentyű- vagy képernyőkódját, és ezt is lapozgatással kereshette meg? Pedig az az igazság, hogy a számítógép mellett az asztalon nem nagyon fér el már semmi, nemhogy néhány kézikönyv!

Peter Norton olyan szakíró és programozó, aki a programozói tapasztalatait nemcsak igen élvezetesen és szakértelemmel írja le, hanem igazi, élő problémákra is ad megoldást.

A következőkben a *The Norton Guides* programot ismertetjük, amellyel egy információs adatbázisból egy adott tématerülethez tartozó adatok hivatkozhatóak a képernyőre oly módon, mintha egy könyvben lapozgatnánk.

A rendszer egy menü szerkezetű megjelenítő programból és a tényleges információt tartalmazó adatbázisokból áll. A megjelenítő program ún. TSR program (terminate and stay resident = befejezés után a memóriában marad), azaz egyszer lefuttatva, a képernyőről billentyűnyomással eltüntethető, majd ismételten előhívható. Helyfoglalása a memóriában 72 kb-ot. A megjelenítő programhoz három menüpont tartozik:

EXPAND — Az adatbázis egy menüpontjának részletes kiírása

SEARCH — Keresés az adatbázisban

OPTIONS — A legfontosabb menüpont az OPTIONS menü. Ebben a következő almenüpontok között választhatunk:

Database — A megjelenítendő (aktív) adatbázis választható ki

Color — A megjelenítés színbeállítása
Full screen — A megjelenítendő ablak mérete: fél/teljes képernyő

Auto lookup — Bekapcsolva a fél képernyős ablak mindig úgy jelenik meg, hogy az eredeti képernyőn lévő kurzor mindig látszik

Hot key — Milyen billentyűkombinációra aktivizálódjon a program

Uninstall — Aktivizálva törli a programot a memóriából

Save options — A beállított paraméterek mentése

Az adatbázisban a mozgás a kurzormozgató jobb és bal nyilakkal, PgUp, PgDn, az oldalt lévő + és - gombokkal, valamint az ESC billentyűvel lehetséges. F1 funkciógomb megnyomása egy rövid segítő képernyőt, valamint a program copyright-ját jeleníti meg.

A programnak önálló, de szerves tartozékai az adatbázisok, melyeknek két típusa létezik. Az egyik típushoz tartoznak a Norton cég által elkészített kész adatbázisok, amelyekben a következő programnyelvekhez tartozó ismeretek vannak összegyűjtve: BASICA, MS-DOS, QuickBASIC, Turbo BASIC, Turbo Pascal, Turbo C, Microsoft C, MASM.

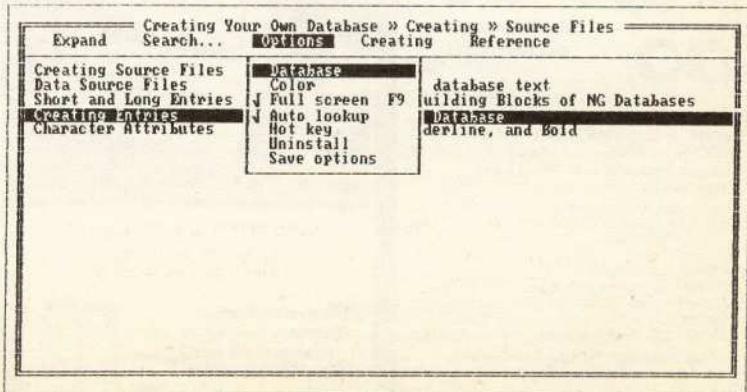
Ezek az adatbázisok készen megvásárolhatók, de újabb adatbázisok készítése

is folyamatban van (például PS/2, Microsoft Windows...). Meg kell vásárolni a megjelenítő programot és azokat az adatbázisokat, amelyekre a felhasználónak szüksége van. Ezekben az adatbázisokban az adott programozási nyelvhez tartozó összes lényeges információ megtalálható. Külön említést érdemel a MASM Macro Assembler adatbázis: az assembler ismertetésén túlmenően tartalmazza a ROM-BIOS és a DOS-függvények pontos és részletes leírását is.

Az adatbázisok másik típusát a felhasználó hozza létre a megvásárolt megjelenítő-, fordító- és szerkesztőprogramok segítségével. Természetesen hogy hogyan kell egy ilyen saját adatbázist létrehozni, az is egy Norton által megírt adatbázisban található, amelynek neve: *Creating Your Own Database*.

A következőkben — mintegy illusztrációként — röviden összefoglaljuk, hogyan épül fel egy ilyen adatbázis, és milyen szempontok szerint kell az információs anyagot előkészíteni. Az adatbázis címszavakból, a hozzá tartozó magyarázatokból és menükből áll. Ezek az egységek két típusú szövegfájlban vannak tárolva: magukban az információkat tartalmazó és a képernyőn megjelenő adatfájlokban, és az információ logikai strukturáját leíró menüvezérlő fájlban.

Az adatfájlokat, miután elkészültek, le





kell fordítani egy tömörített, a megjelenítő program által olvasható alakra, amelynek a a fájlnévkiterjesztése: .NGO. A fordítás azt is biztosítja, hogy az adatbázis bármely része egyforma gyorsan lesz a megjelenítő program számára elérhető. Az adatbázis-generálás utolsó lépése során a menü-összeszerkesztő program segítségével a menüvezérlő fájlt megszerkesztjük. Ez a fájl tartalmazza az adatbázis nevét, a menük felépítését, a menüpontok neveit és a hozzájuk tartozó fájlokat. A szerkesztés végeredménye egy adott nevű, .NG kiterjesztésű, a megjelenítő program által már olvasható formájú adatbázis.

A fordító számára az utasításokat a szövegfájlokban szereplő direktívák (eredeti elnevezéssel: bang commands) segítségével közölhetjük. Mindig új sor elején kell kezdődniük, és az első karakterük a felkiáltójel (!). A következőkben felsoroljuk ezeket a direktívákat, és a jelentésüket is megadjuk.

- !Credits: Az adatbázis szerzői jogát jelezhetjük
- !File: Egy szöveget tartalmazó fájlra hivatkozhatunk
- !menu Menüdefiniáció kezdete
- !Name A generálendő adatbázis neve
- !Sealso: Kereszthivatkozások jelölhetők ki vele
- !Short: Egy fejezet alpontját (címszavát) jelöli

- Még két speciális direktíva használható:
 - * Megjegyzés (comment) kezdetét jelöli
 - * Karaktertípus-váltásra utal (az ESC karakterhez hasonló);
 - * b Vastag írás kezdete (színes képernyőn sárga szín)
 - * n Normál írás
 - * r Inverz videoírás
 - * u Aláhúzás kezdete (színes képernyőn más színű lesz a szöveg)
 - ** A ^ karakter megjelenítése.

A karaktertípus-váltások teszik lehetővé,

hogy egyes szavakat vagy szövegrészeket a szövegben jól látható módon kiemeljünk. A Norton Guides hátránya, hogy nagyobb adatbázis csak merevlemez meghajtóval ellátott gépeken képes futni, és az operatív memóriából is 72 kb-ot fenntart magának. Alkalmazási körét szűkíti, hogy az adatbázis nem tartalmazhat grafikus ábrákat, így szöveggéző grafikákat csak a karakterkészlet felhasználásával alakíthatunk ki.

Az így elkészített adatbázis egy adott tématerülethez tartozó összes információ gyors, szisztematikusan megjelenítését teszi lehetővé. Használható programok alkalmazói útmutatójaként, megtakarítva a külön kézikönyveket, továbbá bizonyos tématerületek oktatásában, tanfolyamok képzési anyagának elsajátításában. Az adatbázis kialakítása nem igényel programozói szakértelmet, hanem pusztán az adott témával kapcsolatos szakismertetet.

— kl —

Operációs rendszerek háborúja

A számítógépek, pontosabban a rajtuk futó alkalmazások potenciális lehetőségeit az operációs rendszer korlátozza vagy kibővíti. Nem oktanal törekvés hát újabb és újabb felfogásban fogant rendszereket kidolgozni, s a felhasználókat ezek előnyeiről meggyőzni. A cél természetesen: állni a versenyt a nagykakal, elsősorban a legnagyobb, a Nagy Kékkal, az IBM-mel.

Az IBM 1981-ben a személyi számítógépekhez a Microsoft cég által kifejlesztett MS-DOS operációs rendszert választotta. Ezzel a választással és az elért piaci sikerekkel ez a rendszer, amit ma már egyszerűen DOS-nak (Disc Operating System) hívunk, a személyi számítógépek szabványos operációs rendszerévé vált.

Ezt megelőzően már létezett egy szinte szabványul elfogadott operációs rendszer, a CP/M (Control Program for Microcomputers), amit a Digital Research cég fejlesztett ki. Sok történet ismert, amely azt meséli el, hogy az IBM miért éppen a DOS-t választotta, nem a CP/M-et. Az egyik legérdekesebb szerint a CP/M fejlesztői éppen golyóznak voltak, mikor az IBM hívta őket, hogy felvegye velük a kapcsolatot.

Két éve kezdte meg az IBM a PS/2 gépek forgalmazását, és egy új operációs rendszert kívánt használni: az OS/2-t. Vételtyásként viszont már régebben ott volt a piacon a Bell Laboratories által kifejlesztett Unix.

Mi is a probléma az MS-DOS-szal? Sajnos van egy nagyon komoly hiányossága: nem képes multitaszkos módban működni. Ez egyszerűen azt jelenti, hogy a számítógép egy időben csak egy felada-

t tud végrehajtani. Ha például egy olyan programot használunk a DOS alatt, mint a Microsoft Windows, a számítógép memóriájában egyszerre több program is van, de amíg az egyik dolgozik, addig a többi nem működik. Ez a körülmény az egyre fejlettebb hardverek hatékonyságát lényegesen korlátozza.

Elégge érthetetlen, hogy az IBM miért ezt az alapvető hiánnyal bíró rendszert választotta, hiszen a hatvanas években már gyártotta a nagy, multitaszkos gépeit, amelyek képesek voltak egy időben több programot futtatni.

A probléma kiküszöbölésére az IBM együttműködve a Microsofttal, létrehozta a DOS-t felváltó multitaszkos operációs rendszert, az Operating System/2-t, vagy röviden az OS/2-t. Ez a rendszer már a multitaszkos futtatáshoz hardverben is felkészített 32 bites mikroprocesszorok olyan új generációját vezérelti, mint például az Intel 80386-os. A valóságban több program futását ez a processzor is a feladatok közötti gyors átkapcsolásokkal oldja meg úgy, hogy a felhasználó nem vesz észre belőle semmit.

Ez a megoldás már régóta ismert a számítógépes rendszerekben, és alkalmazták is, mint például a háttérből való

nyomtatásnál (print spooling). Ilyenkor a számítógép átadja az adatokat a nyomtatónak, és amíg az nyomtat, addig a számítógép más feladattal foglalkozik. Kommunikációs szoftverekben pedig, míg információt visz át a gép a telefonvonalon, addig a felhasználó a gépen más csinálhat.

Az OS/2 rendszer MS-DOS-szal való kompatibilitása érdekében az OS/2 rendszer átkapcsolható, hogy imitálja az MS-DOS rendszert. Így az új rendszer alatt is futtathatók az eredetileg MS-DOS alatt futó programok. Másik megoldásként a gépekben lévő merevlemez is úgy osztható, két részre (particionálható), hogy az egyik részt az MS-DOS, a másikat az OS/2 használja.

Az OS/2 1.0, a rendszer első verziója elég alapfokú volt, és alkalmazása a felhasználókat elég nagy tudást igényelt. Az 1988 végén bejelentett Presentation Manager nevű 1.1 verzió sokkal könnyebben használható, és az IBM új szabványának akarja elfogadtatni.

Az IBM most fanyalag az MS-DOS-szal kapcsolatban, amit a rendszerbe sokat fektető cégek megbízhatatlanságának tekintenek, és ez az OS/2-vel kapcsolatban még visszauteth.

Az IBM-nek jó oka van, hogy az OS/2-t szabványá tette. A kifejlesztett PS/2 számítógépcsaládján elsődlegesen ezt a rendszert akarja futtatni. A család új technológiával készül, az ún. Micro Channel Architecture (MCA = mikrocsatorna-fel-



építés) csatornakezeléssel, amit kimonodottan a 386-os processzorral való együttműködés céljaira terveztek, és az IBM monopolhelyzetét kívánják biztosítani vele.

Bár az IBM a mikrocsatornáját hardvernek tekintti, nem lehet tudni, hogy ténylegesen mi ez és mit csinál. Az IBM analógiákkal — lázasan dolgozó hivatalokkal, csúcsgalanggal és bankok előtt sorban álló tömegekkel — írja le az MCA-t. Magyarázata szerint az új rendszer legnagyobb előnye, hogy megszünteti a hagyományos adatbusz szűk keresztmetszeteit, mivel több ki- és bemeneti csatornát használ a processzor és a külvilág között. Ez a kialakítás lehetővé teszi, hogy az adatforgalom kisebb sebességgel zajljék a buszon, csökkentve a rádiófrekvenciás zavarok lehetőségét. Ezenkívül az adatátvitel 256 bájtis blokkokban történik a szimpla, bájtis átvitel helyett.

Mindzezzel együtt az IBM a PS/2-OS/2-MCA rendszerrel kívánja a másolatokat gyártó cégeket kizárni ebből az üzletből. Az IBM most 5 százalék licencciját kér a PS/2-másolatokat gyártóktól, és 1 százalék díjat viszamenőleg az IBM PC/XT-, AT-másolatokra. Azok a cégek, amelyek ezt a visszamenőleges díjat nem fizet-

tek ki, nem kapják meg a PS/2 gyártási jogát. Az engedély nélküli gyártókat pedig beperlik az IBM-szabadalmak jogtalan használata miatt.

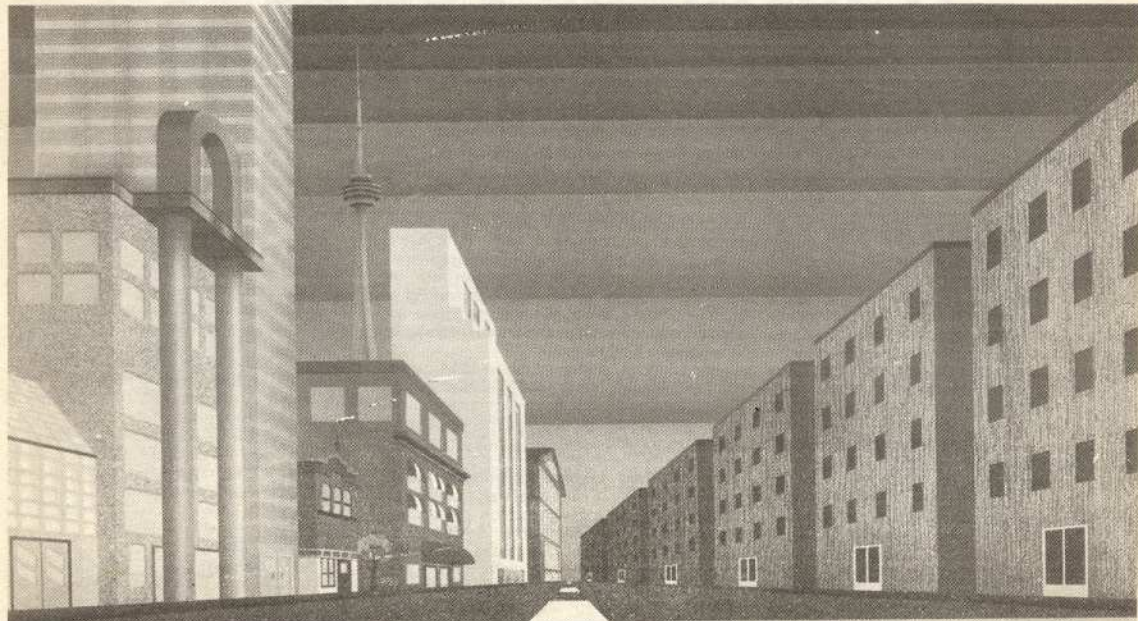
Igaz is, szabadalmak: az IBM évente 500 szabadalmat jegyeztet be, csupán Európában. Néhány ezek közül igen terjedelmes, akár egy háromkötetes könyvnek is beillik, ami 300 szokványos szabadalmi leírásnak felel meg! Ez igen nehézkes és hosszadalmas teszi annak eldöntését, hogy egy PS/2-t másoló által alkalmazott megoldás mennyire tekinthető az IBM-szabadalom eltulajdonításának.

Nehéz megjósolni, mi lesz a felhasználók magatartása az MS-DOS kontra OS/2 kínálatban. Azok, akik valami jobbat akarnak a DOS-nál, választhatják a Unix operációs rendszert. Érdekes módon az IBM más részlege már forgalmazza a szabványos Unix rendszert. Ez, bár kevésbé felhasználóbarát, mint az OS/2, sokkal hatékonyabb annál.

Egyes szakemberek szerint az OS/2 sokkal jobb ugyan az MS-DOS-nál, sokan mégis a Unixra esküsznek. Az OS/2 valahol a DOS és az Unix között helyezkedik el, ezért kihagyható.

A nálunk kevésbé ismert, kanadai Quantum Software Systems szerint azonban mind az OS/2, mind a Unix egy olyan monolitikus operációsrendszer-felfogást testesít meg, amely húsz évvel ezelőtt volt divatban. Az ő QNX operációs rendszerük viszont moduláris szerkezetének köszönheti lényegesen nagyobb teljesítőképességét és rugalmasságát. Sikerét igazolja, hogy 1982 óta 75 000-et adtak el belőle. Előnye az OS/2-vel szemben, hogy nemcsak multitaskingos, hanem többfelhasználós is, egyszerre 32 terminál vagy modem csatlakoztatását téve lehetővé bármely számítógéphez. Azzal pedig még a Unixot is megelőzi, hogy vezérlése alatt integrált hálózat is kialakítható. Ez azt jelenti, hogy minden erőforrást — bármely, a hálózatba kapcsolt processzort, lemezt, nyomtatót stb. — minden felhasználó igénybe vehet.

A QNX (ejtsd kjunix) tehát nemcsak a kiejtésben jelentkező egyetlen *kában* különbözik a Unixtól. Mellette szól az is, hogy mindkét versenytársától eltérően, a PC-től az XT-n és az AT-n keresztül éppen a PS/2-ig bármilyen típusú gépen vagy gépkombinációból álló hálózatot futtatható.



**QNX vs OS/2
UNIX.**



? AJÁNDÉK

Szabad szoftver

TODAY: Eseménynaptár-program

Bizonyára sokszor gondoltunk már arra, hogy jó lenne tudni, milyen érdekes dolgok történtek egy adott napon. Mikor születtek híres vagy kevésbé híres emberek? A PC-re készült TODAY program segítségével megtudhatjuk. A program futás közben megnézi a rendszerdátumot vagy a felhasználó által megadottat, és az adatbázisból kiírja, hogy milyen események történtek azon a napon. Létrehozhatunk saját adatbázist is, a számunkra fontos dátumokból.

Installálás

A TODAY rendszert az alábbi fájlok alkotják:

TODAY.DOC — a program leírását tartalmazó fájl

TODAY.EXE — maga a program

TODAY.xxx — ahol xxx JAN-tól DEC-ig a tizenkét hónapot rövidítse. Ezek a fájlok tartalmazzák az adott hónapban történt eseményeket

TODAY.OWN — a személyes dátumok fájlja (példákkal feltöltve)

Miután kinyomtattuk a DOC fájlt, nem szükséges lemezen megtartani, csak az .EXE és a .xxx fájlokra lesz szükség. Ezeknek azonos alkönyvtárban kell lenniük, amit akár TODAY-nek is lehet nevezni.

A program AUTOEXEC.BAT-ból hívható, például a következőképpen:

```
cd \today
today once clear wait
cd \
```

A today után más paramétereket is megadhatunk.

Hogyan használjuk?

Egyszerűen TODAY-t gépelve, megkapjuk az aznapi csemegét, de megadhatunk egy tetszőleges dátumot HHNN formában, bevezető nullákkal kiegészítve a számokat (például 0102). Ilyenkor a megadott napról regél a TODAY. Ezenkívül még megadhatunk néhány paramétert:

HELP — futtatási paraméterek kiírása

CLEAR — képernyőtörlés kiírás előtt

WAIT — várakozás kiírás után gombnyomásra

ONCE — naponta egyszeri futtatás. Enélkül valahányszor lefuttatjuk a programot, kiírja az üzeneteket. Az AUTOEXEC.BAT-ban használva csak akkor fut a TODAY, amikor először bekapcsoljuk a gépet, az új rendszer-töltéseknél nem.

Ezeknek az opcióknak a nagy része akkor hasznos, ha az AUTOEXEC.BAT-ból hívjuk a programot az első bekapcsoláskor. Segítségükkel megakadályozható, hogy minden rendszertöltésnél ezzel töltsse a gép az időt. Lehet képernyőtörlést kérni, mielőtt a TODAY esetelnie a nap történelmét, és lehet szünetet kérni utána, hogy a következő programok ne töröljenek bele az üzenetekbe. Például: TODAY CLEAR 0628

Saját adatok

A TODAY minden hónap adatait külön fájlban tárolja (TODAY.JAN, TODAY.FEB, TODAY.MAR stb.) Ezeket kívül létrehozhatunk egy saját adatbázist TODAY.OWN néven. Ez azért van, hogy ne kelljen a saját adatokat tizenkét külön fájlban szerkesztgetni. Az összes fájl egyszerű karakteres sorvegfájl. A fájlok sorvég jellel (CRLF=ENTER gomb) lezár 70 karakteres sorokból állnak:

1. karakter — A rekord típusára utaló betű, ami lehet
B (születésnapok),
S (történelmi dátumok, események)
R (emlékeztetők)
* az egész sor csak megjegyzés sor
- 1–5. karakter — Hónap és nap HHNN formában, bevezető nullákkal kiegészítve. Ha a hónap vagy nap helyén nulla áll, akkor az az összes napot vagy hónapot jelenti. Például 0001 minden hónap elsejét, 0400 pedig április minden napját jelenti.
- 6–9. karakter — Az esemény éve négy számjeggyel. Üresen hagyható, ekkor space-ek jelennek meg helyette.
10. karakter — Speciálisrekord-jelző.

Értékei lehetnek:

SPACE: nem speciális rekord

C: az előző sorban kezdett üzenet folytatása. A dátummezőket meg kell ismételní! Ez a funkció alkalmas többsoros üzenetek kiírására.

A: HÉT-NAPJA SZÁMJEGY, ahol 1=vasárnap... 7=szombat. Ezzel lehet elérni, hogy az üzenetet csak akkor írja ki, ha a megadott napra esik. Erre a november havi fájlban található példa a szavazás napján (angol).

11–70. karakter. A megjeleníteni kívánt üzenet.

Példák

*mmdyyyy Birthdays

*

B09011875 Edgar Rice Burroughs, novelist, Ah-ee-ah-ee-ah!
B09021838 Queen Liliuokalani (last queen of Hawaii).

*mmdyyyy Events

*

S09011939 Germany invades Poland, starts World War II.
S0901 2Labor Day, a legal holiday
S09021620 The Mayflower sets sail from Plymouth with
S09021620C102 Pilgrims.

*mmdyyyy Reminders

*

R0001 Pay the mortgage!
R0401 Taxes due in two weeks.
R0415 Last day to pay taxes!



A fejlécek szerepeltetése a formátum megtartása miatt fontos! Figyeljük meg, hogy a Labor Day-nek hétfőre kell esnie, a Mayflower-ról pedig két sorban emlékeznek meg.

A TODAY.OWN fájlnak ugyanez a formátuma. Beirhatunk születésnapokat, évfordulókat, ünnepeket stb. A személyes adatbázis üzenetei valószínűleg születés-

napok és emlékeztetők lesznek. Az emlékeztetőket célszerű némi haladék érdekében néhány nappal a kitűzött dátum elé berakni. Egy kis magyar fájlíráslet:

```
* B=Születésnap R=Megjegyzések E=Események
* hh=hónap (0-12, 0=minden hónap) nn=nap (0-31, 0=minden nap)
* éééé=év, ha hiányzik, minden év
*-----
* Formátum:
* Xhhnnéééé Szöveg, ahol X B,R,E betűk valamelyike
*-----
*
* hhnnéééé Születésnapok
*-----
B08021986 TODAY/PC program ezen a napon született.
B07231952 Feleségem ezen a napon született.
*
* hhnnéééé Események
*-----
S1201988 TODAY/PC programot ezen a napon vettem használatba.
E08021986 TODAY/PC program ezen a napon született.
*
* hhnnéééé Megjegyzések
*-----
R0001 Csekkbefizetések!
R0002 Csekkbefizetések!
R0003 Csekkbefizetések!
```

A programról

A TODAY/PC-t az IBM PC-vel kompatibilis gépekre Datalight C-ben Patrick Kincaid írta 1986 júliusában. Az ötletet egy meglévő program adta, amit Mike Butler írt PL/1-ben egy IBM VM/CMS gépen. Az ő érdeme az eredeti elgondolás és az adatok nagy része.

Felhasználói támogatás

A program írója nem pénzt kér, hanem további adatokat! Szeretné, ha a program ingyen jutna el sok helyre az alábbiakkal: ha valakinek tetszik a dolog, küldjön még adatokat az adatbázisba. Ilyen módon valódi shareware-t (felhasználók által írt és terjesztett szoftvert) készíthetünk. Nem a kiegészített adatbázisokat kell elküldeni, hanem az új adatokat.

A cím: Patrick Kincaid
618 Douglas Drive
Mill Valley, Ca 94941
USA

FIGYELEM!

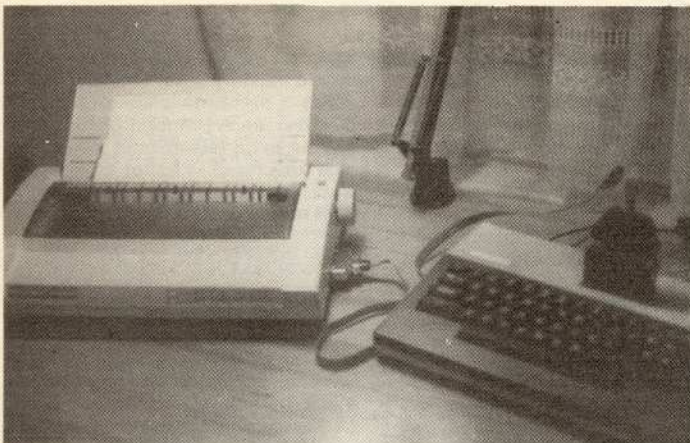
A PÉCÉZZÜNK rovatban megjelent cikkek szövege szöveg-fájlok formájában, valamint az „Ajándék” szabad szoftver 360 kb-átos DS-DD lemezen, utánnvétellel, önköltségi (lemezár, lemezmásolás, postázás) 300 forintos áron megrendelhető.
Cím: Koncz Edit, Budapest, Kunigunda u. 44. 1037

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
- málépítés
- számítástechnika
- távközléstechnika
- vízépités.

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.

Centronics szabványú nyomtatók illesztése



Atari

számítógépekhez

Egy hónappal ezelőtt megjelent cikkünkben a párhuzamos interfész tervezésével és építésével foglalkoztunk, most felhasználásáról szólnunk.

Az Atari cég azzal igyekezett biztosítani az általa gyártott nyomtatók iránti keresletet, hogy a 130 XE, 65 XE, 800 XL és 800 XE számítógéptípusok és nyomtatók kommunikációját bonyolult kétirányú soros átvitelrel valósította meg, s ezzel nagyon megnegyeztette más, gyakran jobb minőségű nyomtatók használatát. Ezen segít a múlt havi cikkben ismertetett párhuzamos interfész és az itt leírt program.

Az operációs rendszer

A felhasználói programnak a perifériák kezelésére hét csatorna (1–7. számozású) áll rendelkezésre. A 0. csatornát az operációs rendszer az editor számára tartja fenn. Ha a felhasználói program meg akar nyitni egy csatornát valamely periféria (képernyő, lemez, billentyűzet stb.) számára, az operációs rendszer először a HATABS táblázatot keresi át (kezdőcíme: \$031A), amely a perifériák azonosító karaktereit (P, C, E, S,

K, ...) és a hozzájuk tartozó kezelőtáblázat kezdőcímet tartalmazza. A HATABS táblázatban 12 periféria definíciója számára van hely. RESET esetén az operációs rendszer átmásolja a ROM-ból az első öt standard periféria definícióját. Ezek a következők:

- P: nyomtató
- C: magnó
- E: editor
- S: képernyő
- K: billentyűzet

Az operációs rendszer a táblázat végéről kezdi a keresést. Ha tehát többször fordul elő a táblázatban ugyanolyan perifériaazonosító karakter, az operációs rendszer a táblázat végéhez közelebb levőt találja meg.

Mint említettük, a HATABS-ban az egyes perifériák kezelőtáblázatának kezdőcíme szerepel. E kezelőtáblázatok formátuma a következő:

- OPEN vektor — 1
- CLOSE vektor — 1
- GET BYTE vektor — 1
- PUT BYTE vektor — 1
- GET STATUS vektor — 1
- SPECIAL vektor — 1

Ha új perifériát akarunk a gépnek definiálni, a táblázat első szabad helyére be kell írni a perifériaazonosító karaktert és utána az új kezelőtáblázat kezdőcímet. Ha már létező perifériát akarunk átdefiniálni, elég a HATABS-ban a megfelelő kezelőtáblázat-kezdőcímet átírni. Mindkét esetben biztosítani kell RESET után az ismételt beírást, mert RESET-nél az operációs rendszer a ROM-ból inicializálja a táblázatot.

A csatornákon történő kommunikáció esetén az operációs rendszer az aktuális kezelőtáblázatból kiolvassa az illető operációt ellátó rutin 1-gyel csökkentett kezdőcímet, verembe tölti, és RTS utasítással ráugrik a rutin kezdőcímeire. Az adatátvitel mindig az A regiszter segítségével valósul meg. A kezelő (driver) rutin munkája végeztével az Y regiszterben üzenetet küld a művelet eredményességéről, illetve az esetleges hibáról. Siker esetén 1, hiba esetén a megfelelő hibakódot tartalmazza, amely 127-nél nagyobb.

Az illesztés

A nyomtatónak az interfészhez való illesztéséhez szükség van egy Centronics és egy 37 pólusú Canon csatlakozóra. A két csatlakozó átkötését a táblázat mutatja, amelyben az egyes jelek nevei is megtalálhatók. A fel-

PC0	13	5TB	1
PA0	23	D0	2
PA1	4	D1	3
PA2	22	D2	4
PA3	3	D3	5
PA4	21	D4	6
PA5	2	D5	7
PA6	20	D6	8
PA7	1	D7	9
PC7	11	BUSY	11
GND	7	GND	19

használt kábel ne legyen hosszabb 1 méternél.

A program

A nyomtatómeghajtó (printer driver) assembly programját (1. lista) ATMAS II-vel írtuk. A 8255-ös IC 0-ás üzemmódban kommunikál, hogy a program a lassúbb nyomtatókkal is megbízhatóan működjön, és azért is, hogy a hosszú vezetón az átmeneti jelenségekből eredő esetleges hibákat elkerüljük.

A legkomolyabb nehézség abból származik, hogy az Atari nem a ha-

nyományos sorvég ASCII-kódot használja, hanem \$9B-t. Ha nyomtatónkkal grafikus üzemmódban szeretnénk nyomtatni, nem írhatjuk át egyszerűen az Atari \$9B sorvégg kódját \$0D, \$0A-ra. A nyomtatókon a grafikus üzemmód definiálása általában a következő bájtsorozattal történik:

ESC, *, horizontális sűrűség, hossz LO, hossz HI. A program úgy íródott, hogy ha ezt a jelsorozatot kapja, akkor a következő *n* bájtot változtatás nélkül küldi a nyomtatóra, mivel grafikus üzemmódról van szó (*n*=hossz LO+256, hossz HI).

A 2. listán látható BASIC program

1. lista

```
*****
*                               *
*   PRINTER DRIVER             *
*   #42501#                     *
*                               *
*****
```

* KAZETTAS VERZIO

```
ESC EPZ 27 ;ESC KODJA
ES EPZ 20
C64 EPZ 30 ;20
BRKKEY EPZ 511 ;BREAK BILL.UZENETE
HATAB5 EQU $0310
MEMLO EQU $0E7
PA EQU $D500 ;A PORT CIME
PC EQU $D502 ;C PORT CIME
CMR EQU $D503 ;IRANYITO REG.CIME
BLK5ZAM EQU (VEGE-FEJLEC+127)/128
```

```
ORG $0700
```

```
FEJLEC DFB 0
DFB BLK5ZAM
DFW FEJLEC
DFW HARMST
```

```
INIT LDA #53C
STA $D302 ;MOTOR STOPP
CLC ;SIKERES JOOT
RTS
```

```
HARMST LDA #VEGE&255
STA MEMLO
LDA #VEGE/256
STA MEMD+1 ;UJ MEMLO
LDA #HATAB&255
STA HATAB+1
LDA #HATAB/256
STA HATAB+2 ;UJ PRINTERT DEFINIALO
;TABLAZAT
```

```
*
RTS
```

```
CLOSE LDA #X10011011 ;MODE 0,IMPUT
STA CMR
PROV LDY #1 ;OK UZENET
WAIT RTS
```

```
OPEN LDA #0
STA ES ;NORN.UZENMOD
LDA #X10001010
STA CMR ;8255 BEALLITASA
LDA #500 ;CR
STA PA
LDX #X00000001
STX PC
```

```
PRTEST LDA PC ;BUSY->CY
ROL ;0=>OK
BCR PROK ;1=>ISMETELD
LWX ;MAX.255-SZOR
BNE PRTEST ;ERROR 130
LDY #130
RTS
```

```
PUTB LDY ES ;GRAF.MOD=>UGORJ
BMI GR ;<0=>SPEC.MOD
BNE SPEC
```

```

CMP #59B ;EOL?
BNE P1 ;NEM,UGORJ
LDA #500 ;CR
JSR PUTB1
LDA #50A ;LF
BNE PUTB1 ;JMP PUTB1
CMP #50C ;ESC?
BNE PUTB1 ;NEM,KULDD KI
INC E5 ;IGEM,ES:=1
BNE PUTB1 ;JMP
CPY #1 ;LEGUTOB B ESC VOLT?
BNE SPEC2 ;NEM,UGORJ TOVABB
INC E5 ;IGEM,ES:=2
CMP #'M' ;M?
BEQ PUTB1 ;IGEM,KULDD KI
LDY #'0' ;HA MAS VOLT MINT ESCM,
STY E5 ; NULLAZD AZ ES VALTOZOT
BEQ PUTB1 ; ES KULDD KI (JMP)
SPEC2 BNE SPEC3 ;LEGUTOB B VOLT?
INC E5 ;NEM,UGORJ TOVABB
BNE PUTB1 ;IGEM,ES:=3
CPY #3 ;ES KULDD KI (HORIZ.SURUSEG)
BNE SPEC4 ;NEM,UGORJ TOVABB
STA C64 ;CNT LO KOVETK?
INC E5 ;ES:=4
BNE PUTB1 ;ES KULDD KI
SPEC4 STA C64+1 ;CNT HI- TEDD EL
LDY #255 ;GRAF.MOD JELZESE
STY E5 ;KULDD KI CNT HI-T
BNE PUTB1
GR LDY C64 ;C64<>0=>GR2
BNE GR2
LDY C64+1 ;C64=0 ES C64+1<>0 => GR1
BNE GR1 ;C64=0 ES C64+1=0 => ES:=0
INC E5
GR1 DEC C64+1
GR2 DEC C64
PUTB1 LDY #128
STA PA
JSR WAIT
DEC PC ;STBNOT:=0
NOP
INC PC ;STBNOT:=1
PUTCYX LDA BRKKEY ;VOLT BREAK?
BEQ BREAK ;IGEM,UGORJ
LDA PC ;NEM,TESTELD
ROL ; A BUSY-T
BCS PUTCYX ;BUSY=>ISMETELD
LDY #1
RTS
BREAK LDY #128 ;BREAK ERROR
RTS
NOTIMP LDY #146 ;ERROR 146
RTS
*** PERIFERIA-DEFINIALO TABLAZAT:
TAB DFW OPEN-1
DFW CLOSE-1
DFW NOTIMP-1 ;GETBYTE
DFW PUTB-1
DFW NOTIMP-1 ;GETSTAT
DFW NOTIMP-1 ;SPECIAL
VEGE DFB 0
```



```

5 REM AUTOBOOT GENERATOR
10 REM EPSON driver-kazettás verzió
20 ? "Kész:tsd el a magnót !"
30 OPEN #1,0,128,"C:";C=0
40 FOR I=1 TO 202:READ A:C=C+A
50 PUT #1,A:NEXT I
60 CLOSE #1
70 IF C(>22031 THEN ? "Hiba az adatokban !"
80 END
100 DATA 0,2,0,7,13,7,169
110 DATA 60,141,2,211,24,96,169
120 DATA 202,141,231,2,169,7,141
130 DATA 232,2,169,190,141,27,3
140 DATA 169,7,141,28,3,96,169
150 DATA 155,141,3,213,160,1,96
160 DATA 169,0,133,28,169,138,141
170 DATA 3,213,169,13,141,0,213
180 DATA 162,1,142,2,213,173,2
190 DATA 213,42,144,228,232,208,247

```

```

200 DATA 160,136,96,164,28,48,65
210 DATA 208,21,201,155,208,9,169
220 DATA 13,32,156,7,169,10,208
230 DATA 64,201,27,208,60,230,28
240 DATA 208,56,192,1,208,12,230
250 DATA 28,201,42,240,46,160,48
260 DATA 132,28,240,40,192,2,208
270 DATA 4,230,28,208,32,192,3
280 DATA 208,6,133,30,230,28,208
290 DATA 22,133,31,160,255,132,28
300 DATA 208,14,164,30,208,8,164
310 DATA 31,208,2,230,28,198,31
320 DATA 198,30,160,128,141,0,213
330 DATA 32,41,7,206,2,213,234
340 DATA 230,2,213,165,17,240,9
350 DATA 173,2,213,42,176,246,160
360 DATA 1,96,160,128,96,160,146
370 DATA 96,41,7,33,7,186,7
380 DATA 72,7,186,7,186,7

```

2. lista

neve autoboot-generátor. Funkciója csak annyi, hogy elindítása után kazettára küldi a nyomtatómeghajtó programot, autoboot formában. Az így szerzett autoboot fájl úgy tölthető be a számítógépbe, hogy a gép bekapcsolásakor lenyomva tartjuk a

START billentyűt, majd a gép kérésére betöltjük az autoboot programot. Betöltés után a BASIC jelentkezik, de már úgy, hogy az összes nyomtatóval kapcsolatos utasítás az új meghajtót használja. A RESET nem árt a nyomtatómeghajtónak.

A program működését csak Commodore MPS 1250 nyomtatóval tudtuk kipróbálni, de feltételezzük, hogy bármilyen más, Centronics szabványú nyomtatóval (EPSON, STAR stb.) is ugyanolyan jól működik.

Máder Indira — Szijjártó Jenő



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

A BLAGUSS-VOLÁNBUSZ

Utazási Iroda
felajánlja

- vállalati szakmai utazások, kirándulások, tanulmányutak
- külföldi kiállításokon, vásárokon való részvétel
- és minden egyéb külföldi szakmai rendezvényre történő csoportos utazás (szállás, ellátás, kirándulások, szakmai programok) komplett megszervezését.

Várjuk szíves
megkeresésüket!

BLAGUSS-VOLÁNBUSZ
1051 Budapest,
Engels tér,
Autóbusz-pályaudvar
Telefon: 177-777
Telex: 22-3179



Az AmigaDOS

Előző (1989/7-es) számunkban bemutattuk az Amiga multitasking szervezésű operációs rendszerét általában, és elkezdtük a DOS utasításkészletének ismertetését. Ezt folytatjuk, felsorolva a szövegszerkesztő parancsait és a hibáüzeneteket is.

Az AmigaDOS utasításkészlete

EXECUTE utasításfájl neve [argumentumok]

Végrehajtja az AmigaDOS szabályai szerint megírt utasításfájlt (programot), behelyettesítve az argumentumokat. Az utasításfájl első soraiban szokás elhelyezni azokat az ún. direktívákat, amelyek az EXECUTE működését befolyásolják. Egy direktíva előtt jelzőkarakterként — alapértelmezés szerint — egy pontnak (.) kell állnia:

.K argumentumnév.

Az EXECUTE paranccsal beadott argumentumra az utasításfájlban ezen a néven hivatkozhatunk. Például direkt módon ki szeretnénk iratni a „Program” nevű fájl paramétereit. Ezért „Lister” néven megírjuk a következő szöveges fájlt:

.K fájlnev

LIST < fájlnev >

Ezután az „EXECUTE Lister Program” parancs már el is végzi a feladatot. (Az utasításfájlban lévő „fájlnev” olyan, mint egy szöveges változó, amelynek a tartalma jelen esetben „Program” lesz.)

.DOT jelzőkarakter — a pont (.) jelzőkaraktert másra állítja át

.BRA nyitó karakter — az argumentum nyitó karakterét (<) változtatja meg

.KET záró karakter — az argumentum záró karakterét (>) változtatja meg

.DOLLAR karakter — a dollárjellet (\$) más karakterre állítja át

FAILAT [hibakód]

Ha a parancs kiadása után bármely működés közben az itt megadottal egyenlő vagy ennél nagyobb kódszámú hiba következik be, leáll a program futása. A hibakód pozitív egész szám lehet, az alapbeállítás a 10-es kódú hiba, amely az ERROR feltételhez tartozik (lásd: IF utasítás).

FAULT [hibakód, ...]

Kiírja a felsorolt hibakódoknak megfelelő hibáüzeneteket. Legfeljebb 10 hibakód adható meg egyszerre.

FILENOTE [FILE] fájlnev COMMENT megjegyzés

A megadott nevű fájlhoz hozzáfűzi a legfeljebb 80 karakterből álló megjegyzést (kommentárt, alcímet, jegyzeteket). A LIST utasítás segítségével a megjegyzések is kiírhatók a képernyőre.

FORMAT DRIVE meghajtó NAME lemeznév [NOICONS]

Megformázza a kijelölt lemezmeghajtóban lévő mágneslemezt. A lemez neve legfeljebb 30 karakterből állhat. Ha kitesszük a NOICONS opciót, az újonnan formázott lemezre nem kerül fel az alapértelmezés szerinti „Trashcan” (szeméttgyűjtő) ikonja (Trashcan, info nevű fájl).

IF [NOT] [WARN] [ERROR] [FAIL] [szöveg1 EQ szöveg2] [EXISTS fájlnev]

[ELSE]

[ENDIF]

A BASIC-ból jól ismert IF-THEN-ELSE utasításhármas feladatát látja el, vagyis feltételes elágazásokat hozhatunk létre a segítségével. A feltételek értelmezése:

NOT — negáció (tagadás, invertálás)

WARN — akkor igaz, ha a hibakód > = 5

ERROR — akkor igaz, ha a hibakód > = 10

FAIL — akkor igaz, ha a hibakód > = 20

szöveg1 EQ szöveg2 — igaz, ha a két szöveg megegyezik

EXISTS fájlnev — igaz, ha a fájl megtalálható valamelyik lemezen vagy a RAM-díszken

INFO

Információkat ír ki a rendszerről és a meghajtó(k)ban levő lemezekről.

INSTALL [DRIVE] meghajtó

Az utasítás segítségével a meghajtóban levő lemezből rendszerindító (boot) lemezt készíthetünk. Az INSTALL a lemez 0-ás és 1-es szektorát (a két boot-blokkot) írja felül, elhelyezve bennük a rendszerindításhoz szükséges gépi kódú programot. Ez az utasítás felhasználható lemezeink vírusmentesítésére is, mivel a programvírusok többsége a boot-blokkban helyezi el magát (SCA, Byte Bandit és a magát antivírusként hirdető North Star).

JOIN forrás fájlnev [forrás fájlnev2 ...] AS cél fájlnev
Az utasítás összegzi a forrásfájlokat (maximum 15-öt), majd az összeget elmenti a cél fájlba. A művelet a felsorolás sorrendjében történik.

LAB címke

Segítségével címkéket helyezhetünk el az AmigaDOS programban. A címkékre az EXECUTE, az IF és a SKIP utasításokkal hivatkozhatunk. A címkével megjelölhetjük a program bizonyos pontjait, amelyekre ezután például vezérlésátadást (ugrást) hajthatunk végre.

LIST [DIR útvonal] [P/PAT maszk] [KEYS] [DATES] [NO-DATES]

[TO fájlnev] [S névrészlet] [SINCE dátum] [UPTO dátum][QUICK]

A megadható opciók:

DIR útvonal — a megadott útvonalon végzi a listázást
P/PAT maszk — csak azokat a fájlokat listázza ki, amelyekre illik a maszk (a maszkokról a bevezetőben mint rövidítési lehetőségekről esett szó)

KEYS — kiírja azokat a szektorszámokat, ahol a fájlok kezdődnek

DATES — kiírja a létrehozás dátumát is

NO-DATES — sem a dátumokat, sem az időpontokat nem írja ki

TO fájlnev — a lista a fájlba kerül ASCII formátumban S névrészlet — kilistázza azokat a fájlokat, amelyek nevében szerepel a megadott névrészlet

SINCE dátum — a megadott dátum után létrehozott fájlokat írja ki

UPTO dátum — a megadott dátum előtt létrehozott fájlokat írja ki

QUICK — csak a neveket sorolja fel

LOADWB

Betölti és elindítja a Workbenchet, az Amiga magas szintű, ikonorientált operációs rendszerét.

MAKEDIR könyvtárnév

Új alkönyvtárat nyit a megadott névvel és a megadott lemezen vagy meghajtón.

MOUNT [DEVICE] készülék

Egy új készüléket aktualizál az AmigaDOS számára. Az aktualizálandó készülék kezelési paramétereit a DEVS könyvtárban elhelyezett Mountlist nevű fájlban adhatjuk meg. A gyári rendszerlemez Mountlistje példákat tartalmaz, amelyeken keresztül megtanulható ennek a fájlnak az elkészítése.

NEWCLI [CON: x/y/szélesség/magasság/ [név][FROM fájlnev]]

Elindít egy új CLI taskot, és kinyit számára egy ablakot. Az ablak bal felső sarkának koordinátái (x és y), szélessége, magassága és neve tetszőleges lehet. A megadott fájlban lévő utasításokat a CLI-ablak kinyitása után végrehajtja a DOS. Figyelem! A törtvonal jelen esetben nem választási lehetőséget jelöl, ugyanis a helyes működéshez a törtvonalaknak is szerepelniük kell az utasítás megadásában.

PATH [[SHOW][ADD]/RESET útvonal [útvonal] ...]

Deklarálja azokat az útvonalakat, amelyeken a DOS a betölteni kívánt fájlt keresi, ha azt az aktuális könyvtárban nem találja.

Opciók:

SHOW — kiírja az eddig beállított útvonalakat

ADD — újabb útvonal megadását jelenti, az eddig megadottak megtartása mellett

RESET — már meglévő útvonalakat töröl

Az önmagában kiadott PATH a PATH SHOW funkcióját látja el.

PROMPT promptjel

Megváltoztatja a CLI-task promptjelét, amely bármilyen karaktersorozatból állhat. (A promptjel azt jelzi a felhasználónak, hogy a DOS végrehajtotta az összes kiadott parancsot, és új parancs fogadására kész.) Ahol a karaktersorozatban „%N” jelzés áll, ott az illető CLI-task száma fog megjelenni.

PROTECT [FILE] fájlnev [FLAGS] attribútumok

A fájl védelmi attribútumait állítja be. Az attribútumok helyén a következő rövidítések állhatnak:

R — olvasható (Readable)

W — írható (Writable)

D — törölhető (Deletable)

E — végrehajtható (Executeable)

A — archivált (Archived)

QUIT [hibakód]

Befejezi az AmigaDOS program végrehajtását és a megadott kódú hibával tér vissza parancsmódba. Az alapértelmezés a 0 hibakód, amely hibamentes kilépést jelent.

RELABEL [DRIVE] meghajtó [NAME] lemeznév

Átnevezi a meghajtóban lévő lemezt. A név legfeljebb 30 karakterből állhat, és ha szóközt is tartalmaz, akkor idézőjelbe kell tenni.

RENAME [FROM] régi fájlnev [TO AS] új fájlnev

Átnevezi a megadott fájlt.

RUN utasítás + ... + utasítás

Végrehajtja a felsorolt utasításokat oly módon, hogy egy új taskot nyit meg részükre. Induláskor kiírja a taskok számát.

SEARCH [FROM] fájlnev/maszk [SEARCH] karakterlánc [ALL]

Megkeresi a karakterlánc összes előfordulását a megadott fájl(ok)ban. Ha a karakterlánc szóközt is tartalmaz, akkor idézőjelbe kell tenni. A maszk felépítése olyan lehet, mint amiről a bevezetőben, rövidítések címszó alatt szó volt. A második SEARCH kulcsszó kiírása (vagy elhagyása) nem befolyásolja a működést, ALL opció esetén minden elérhető fájlban végigmegy a keresés.

SETLOCK OPT LOAD/SAVE

Beállítja a rendszerórát és a hardverórát. LOAD esetén a rendszeróra állása átíródik a hardverórába, SAVE esetén pedig éppen fordítva. Ez az utasítás csak hardverórával ellátott Amigán működik!

SETDATE fájlnev nap-hónap-év [óra:perc]

Megváltoztatja egy fájl létrehozási dátumát és időpontját.

SETMAP országcód

Beállítja a billentyűzet-konfigurációt.

Országcódok Konfigurációk

cnd kanadai

ch1 német svájci

ch2 francia svájci

d német

dk dán

e spanyol

f francia

gb brit

i olasz

is izlandi

n norvég

s svéd

usa0 USA (alternatív)

usa1 USA (szabványos)

usa2 USA (alternatív)

Ahhoz, hogy a SETMAP működhessen, a felsorolt országcódokkal megegyező nevű fájloknak kell lenniük a DEVS könyvtár KEYMAPS nevű alkönyvtárban.

SKIP [címké]

Vezérlésátadás (ugrás) arra az utasításra, amely előtt a „LAB címké” áll. A SKIP utasítás argumentum nélkül a legközelebbi LAB utáni utasításra ugrik.

SORT [FROM] forrás fájlnev [[TO] cél fájlnev]

[COLSTART kezdőoszlop]

A forrásfájl (célszerűen szöveges fájl) tartalmát ábécésorrendbe rendezi, és az eredményt elmenti a cél fájlba. A COLSTART után megadható a sorba rendezés kezdőoszlopa is. Az utasítás nem tesz különbséget a kis- és nagybetűk között! A rendszer veremtárának alapértéke 4000 bájt, amely kb. kétszáz soros fájl kezelésére elegendő. Ha ennél nagyobb fájl akarunk rendezni, használjuk előbb a STACK utasítást. Ez azért fontos, mert a verem telítődésekor a rendszer biztosan elszáll!

STACK [mérete bájtokban]

Argumentum nélkül esetben kijelzi a rendszer veremtárának méretét, argumentummal pedig beállítja azt. A veremtár kiindulási mérete 4000 bájt.

STATUS [taszkszám] [FULL] [TCB] [SEGS] [CLI/ALL]

Információkat ír ki egy taszkról. FULL opcióval minden adatot kiír. TCB megadásakor kiírja a taszk prioritását, veremtárának méretét és a globális vektorok méretét. SEGS opcióval feltünteteti a szegmenslistában a szegmenscsoportok neveit. CLI-t vagy ALL-t megadva az utasítás az aktuális CLI-taszokra vonatkozik.

TYPE [FROM] forrás fájlnev [[TO] cél fájlnev] [OPT N/H]

Opció nélkül szöveges fájlok egyszerű kiírására használatos. OPT N esetén megszámozza a kiírt sorokat,

OPT H-val pedig 16 bites hexadecimális formában írja ki a fájl tartalmát. A kiírást egy tetszőleges billentyű lenyomásával megállíthatjuk, majd RETURN-nel vagy CTRL+X-szel folytathatjuk. A CTRL+C a TYPE utasításból való kilépést eredményezi.

VERSION

A Kickstart (az Amiga ROM-ja) és a Workbench verziószámát adja vissza.

WAIT [érték] [SEC/SECS] [MIN/MINS] [UNTIL óra:perc]
Várakozás a megadott időtartamig vagy időpontig (vagy argumentum nélküli esetben 1 másodpercig). Az érték SEC vagy SECS előtt állva másodperceket, MIN vagy MINS előtt állva perceket jelent. UNTIL opcióval az ez után megadott időpontig vár a DOS.

WHY

Visszakérdez, hogy miért következett be a legutóbbi hiba.

Az AmigaDOS ED nevű szövegszerkesztőjének parancsai

Direkt vezérlőparancsok:

CTRL+A üres sor beszúrása
CTRL+B egy sor törlése
CTRL+D scroll le
CTRL+E felváltva az ablak bal felső és jobb alsó sarkába viszi a kurzort
CTRL+F a kurzor helyén álló betűt nagybetűről kicsire vagy kisbetűről nagyra váltja át
CTRL+G megismétli az utoljára beadott indirekt vezérlőparancsot
CTRL+H a kurzortól balra egy karaktert töröl
CTRL+I a kurzort a következő tabulátorpozícióra helyezi
CTRL+M megegyezik a RETURN billentyűvel
CTRL+O levágja a szónak a kurzortól jobbra eső részét
CTRL+R az előző szó végére állítja a kurzort
CTRL+T a következő szó elejére helyezi a kurzort
CTRL+U scroll fel
CTRL+V letisztazza a kijelzett szöveget
CTRL+Y törli a sornak a kurzortól jobbra eső részét
CTRL+[indirekt módba vált át (megegyezik az ESC-pel)
CTRL+] felváltva a sor elejére és a végére helyezi a kurzort

Indirekt vezérlőparancsok (indirekt módba az ESC billentyűvel léphetünk):

A „szöveg” a kurzor alatt új sort nyit, és ide írja be a szöveget
B a szöveg végére állítja a kurzort
BE megjelöli a blokk végét
BF „szöveg” megkeresi a beadott szöveget, a kurzortól visszafelé
BS megjelöli a blokk kezdetét
CE a sor végére állítja a kurzort
CL kurzor balra
CR kurzor jobbra
CS a sor elejére állítja a kurzort
D törli a sort
DB törli a blokkot
DC törli a kurzor helyén álló karaktert
E „szöveg1” „szöveg2” az első szöveget kicseréli a másodikra
EQ „szöveg1” „szöveg2” az első szöveget kicseréli a másodikra, és minden csere előtt visszakérdez
EX törli a jobb oldali margót
F „szöveg” megkeresi a beadott szöveget a kurzortól előrehaladva
I „szöveg” beszúrja a beadott szöveget
IB a kurzor sorától kezdődően bemásolja a megjelölt blokkot



Born a Champion

IF „fájlnev” a kurzor alatti sortól kezdődően beszúrja a megadott szöveges fájl tartalmát
J egyesíti az aktuális sort a következővel
LC kereséskor különbséget tesz a kis- és a nagybetűk között
M sorszám a megadott számú sorra állítja a kurzort
N a következő sor elejére állítja a kurzort
P az előző sor elejére állítja a kurzort
Q kilépés, a fájl elmentése nélkül
RP az addig ismétli az utasítást, amíg valamilyen hiba be nem következik
S kettévágja a sort a kurzor pozíciójánál
SA [„fájlnev”] elmenti a szöveget
SB a megjelölt blokk elejére állítja a kurzort
SH kiírja az ED beállított paramétereit
LS oszlop a bal margót a megadott oszlopra állítja be
SR oszlop a jobb margót a megadott oszlopra állítja be
ST oszlop beállítja a tabulátor lépésmagyságát
T a szöveg elejére állítja a kurzort
U semmisnek tekint az aktuális soron végrehajtott változtatásokat
UC kereséskor nem tesz különbséget a kis- és nagybetűk között
WB „fájlnev” elmenti a megjelölt blokkot
X [„fájlnev”] elmenti a szöveget, és kilép az editorból

Az AmigaDOS hibaüzenetei

Kód	Angol név	Magyar meghatározás
103	insufficient free store	Kevés a rendelkezésre álló memória
121	file is not an object module	A fájl nem futtatható program
202	object in use	Jelenleg egy másik program dolgozik az elérni kívánt fájljal
203	object already exists	A fájl már létezik ezen a néven
204	directory not found	Nincs meg a keresett könyvtár
205	object not found	Nincs meg a keresett program
210	invalid stream component name	Nem érvényes a fájlnev
213	disk not validated	A DOS nem tud olvasni a lemezről (mert hibás a lemez, vagy azért, mert adatforgalom közben kivettük a lemezt a meghajtóból)
214	disk write-protected	Írási kísérlet írásvédett lemezen
218	device not mounted	Az elérni kívánt eszköz nincs csatlakoztatva, vagy nem üzemképes
221	disk full	A lemez megtelt
222	file is protected from deletion	A fájl védelmi attribútuma nem engedi meg a törlést
225	not a DOS disk	Nem AmigaDOS-formátumú a lemez
226	no disk in drive	Nincs lemez a meghajtóban

Tóth Zoltán

Adom a magyarázatot!

A Mikroszámítógép Magazin 1989/5. számában jelent meg dr. Zana János kérdése. Erre kétféle választ is kaptunk.

A „létező” állományok a Commodore DOS-ban nem kell hogy törvényszerűen adatot is tartalmazzanak. Próbáljuk ki:

OPEN8,8,B,PROBA,S,W":CLOSE8

A tartalomjegyzékben megjelenik egy egyblokkos, PROBA nevű soros fájl, amely azonban egyetlen bájt kiolvasható adatot sem tartalmaz. Hogyan lehetséges ez?

Az OPEN utasítás hatására a katalógusba bekerül a fájl neve, típusa és annak az első szabad blokknak sáv- és szektorszámát, ahová az első adatok kerülnek. Ennek az adatblokknak az első két bájtja azonban a DOS számára hordoz információt: azt mutatja meg, hogy a fájl hol folytatódik (folytatóblokk sávja, szektora). Ha ez az utolsó, a fájlhoz tartozó blokk, akkor az első bájt nulla, a második pedig az ebben a blokkban lévő, de még a fájlhoz tartozó bájtok számát adja meg. (Ad abszurdum, mint példánkban, ez is nulla.) Így adódhat elő, hogy a katalógusban egyblokkos fájl valójában nem tartalmaz kiolvasható adatot.

Hogy ezt a körülményt a BASIC nem ismeri fel és nem kezeli, az csak azt bizonyítja, hogy a hibafelismerésben (és -csinálásban) a felhasználók pályahosszal verik a programozókat. A kérdésben leírt hibajelenséget tehát egy olyan fájl olvasásának kísérlete okozta, amelyet adatok írása nélkül zártak le.

Itt a helye egy másik Commodore BASIC-sajátosságról is szólni. Ha a CHR\$(0)-t írjuk ki lemezre, majd onnan visszaolvassuk bájtontként, azaz mondjuk GET-tel, akkor nem CHR\$(0)-t kapunk vissza, hanem az úgynevezett nullsztringet, azaz a nulla hosszú karakterláncot (""). Ez akkor okoz problémát, ha az ASCII értékeire vagyunk kíváncsiak. Ilyenkor ILLEGAL QUANTITY ERROR üzenetet kapunk. A helyes eljárás tehát:

100 GET #8,Q\$

110 IF Q\$="" THEN Q\$=CHR\$(0)

120 Q=ASC(Q\$)

A 110–120-as sorokat egyszerűbben és lényegesen gyorsabban megoldhatjuk így is:

110 Q=ASC(Q\$+CHR\$(0))

Zoltai Péter

A hiba a mellékelt kis programmal könnyen reprodukálható. Ebben a 110-es sor létrehoz egy szekvenciális fájlt, amelyik két rekordot tartalmaz. Az utóbbiakat a 130-as és a 150-es sor egymástól némileg eltérő INPUT# utasításai hibátlanul visszaolvassák.

De, ha az első sorban B\$="" értéket adunk meg, már elő is állítottuk a kérdéses hibát. A fájl írásakor semmi rendellenességet sem észlelünk. A visszaolvasást azonban csak a 130-as sor utasításai hajtják végre; a 150-es sor önálló INPUT#2,B\$ utasításra keres egy gyakorlatilag nem létező rekordot és ezzel a rendszer le is merevedett.

A hibára tehát olvasáskor derül fény, de az valójában a fájl írásakor keletkezik, mert az „üres rekord” létrehozásával a rekordokat elválasztó CHR\$(13) jelek közvetlenül egymás mellé kerülnek. Ha ez nem a fájl végén történik, tulajdonképpen nincs baj, mert a CHR\$(13)-ak kiolvasásával egyidejűleg az ST állapotjelző értéke 0; ez további „valódi rekord” létezését feltételezi, így az INPUT# utasítás tovább olvas. A fájl végén azonban a második CHR\$(13) kiolvasásakor ST=64 — ami a fájl végét jelzi —, az INPUT#2 viszont nem tudott értékes adatot kiolvasni, így a rendszer leáll.

A tanulság: a fájl végén sohasem állhat „üres rekord”! A megoldás egyik módja, hogy az üres sztringek helyett valamilyen nem zavaró jelet viszünk fel a lemezre: pl. * vagy CHR\$(160). Ehhez egy IF-es utasítás kell, ami hosszabb ciklus esetén BASIC-ben észrevehető időtöbbletet jelent.

A második lehetőség, hogy a beolvasó ciklus lefutása után utolsó adatként beviszünk egy ilyen jelet. A mellékelt program 110-es sorában ez a CLOSE2 előtt még pl. egy PRINT#2,CHR\$(160) utasítást jelentene.

Barabás Miklós

```
100 A$="111";B$="222":PRINT"☐"
110 OPEN2,B,2,"☐:PROBA,S,W":PRINT#2,A$:
    PRINT#2,B$:CLOSE2
120 GOSUB180
130 OPEN2,B,2,"PROBA,S,R":INPUT#2,A$,B$:
    CLOSE2
140 GOSUB180
150 OPEN2,B,2,"PROBA,S,R":INPUT#2,A$:
    INPUT#2,B$:CLOSE2
160 GOSUB180:PRINT"☐":LIST
170 :
180 PRINTA$:PRINTB$:A$="" :B$="" :
    PRINT,"KESZ"
190 POKE198,0:WAIT198,1:RETURN
```

Ki ad magyarázatot?

Ebben a rovatban eddig bonyolultnak látszó, valójában többé vagy kevésbé egyszerűen megmagyarázható problémákat vetettek fel az olvasók. Most az egyszer fordítsuk meg ezt a viszonyt: íme két meglehetősen egyszerűen induló eset, azonban ezek megoldói kiérdemlik „A BASIC Sherlock Holmes-a” címet.

1. Írjuk be ezt a nem túl értelmes, de legalábbis ártalmatlannak tűnő sort:

PRINT 1+ "2"+ -3

Az 1 és 3 helyett bármilyen szám állhat, "2" pedig bármilyen szöveg lehet. A RETURN megnyomása után ne nagyon zaklassuk gépünket, jelke már a Nirvánába jutott: a RESTORE gomb püffölése sem segít. Hasonló jelenséget tapasztalunk bármilyen 8 bites Commodore gépen, a C16-tól a PET-en át a CBM 8000 szeriáig. A PLUS/4 például a moni-

torban „landol”. A ki- és bekapcsolás után érdemes kísérletezni, a lényeg a karakterláncot követő "+ -"-ban van.

2. A gép bekapcsolása után írjuk be a következő programsort:

10 LOAD "\$"8

Indítsuk el RUN-nal, a floppy rövid tekergetése után „?SYNTAX ERROR IN 0” hibaüzenettel leáll a programunk. Eddig minden tiszta ügy, azonban próbáljuk meg mondjuk a 0. sort kitorolni a <0> és <RETURN> beírásával! Pillanatszerű várakozás után a keretszín fehérre, a háttérszín feketére vált, a képernyő alján ismeretlen karakterek tűnnek fel, majd C64-esünk mély álomba sanderül, melyből csak a <RUN/STOP + RESTORE> ébreszti fel. A kiméletlen pofoztatás után is meglehetősen zavarosak gondolatai, erről a LIST parancs segítségével győződhethünk meg.

(„Sherlock Holmes” díj ezüst fokozata)

Zoltai Péter

Támadás és védelem

A támadás és védelem közötti egyensúly fenntartása a középjáték fontos feladata. Legtöbb esetben a támadó félnek kedvező alkalma nyílik szép kombináció kezdeményezésére és annak lebonyolítására. A támadás és védelem mértékének meghatározása közvetlen kapcsolatban van a mozgékonyssággal.

A következőkben a támadási érték három különféle számítási módjával foglalkozunk, és ezeket hasonlítjuk össze.

1. számítási mód: a különböző megtámadott mezők száma.

A támadás alá eső különböző mezők száma 3 és 64 közé esik. 3 abban az esetben, ha csupán királya van az egyik félnek és az az egyik sarokmezőn áll. A „támadás alatt lévő mező” kifejezést akkor használjuk, ha azt az egyik fél birtokolja, függetlenül attól, hogy ott áll-e figura — akár ellenséges, akár saját —, a „megtámadott mező” kifejezést pedig azt jelenti, hogy arra az ellenséges király nem léphet rá, mert ott sakknak lenne. A két értéket nagyon hasonló, de nem ekvivalens egymással!

Példaként tekintünk a bemutatott hadállást, ahol a világos figurákkal megtámadott mezők száma 41, a sötét figurákkal megtámadottaké pedig 42. Ha egy mezőt több figura támad, akkor azt többször kell számítani.

Világossal a következő negyvenegy lépést számítjuk:

1. Bc1-a1, 2. Bc1-b1, 3. Bc1-d1, 4. Bc1-e2, 5. Be1-d1, 6. Be1-f1, 7. Kg1-f1, 8. Kg1-h1, 9. e2×d3, 10. f2×e3, 11. Fg2-f1, 12. Fg2-h1, 13. Fg2-h3, 14. Hc3-b1, 15. Hc3-a2, 16. Hc3-a4, 17. Hc3-b5, 18. Hc3-e4, 19. Hc3-d1, 20. Hf3-d2, 21. Hf3-d4, 22. Hf3-e5, 23. g3×f4, 24. b4×a5, 25. b4×c5, 26. Vh4-c4, 27. Vh4-d4, 28.

Vh4-e4, 29. Vh4-f4, 30. Vh4-g4, 31. Vh4-h3, 32. Vh4×h5, 33. d5×c6, 34. d5×e6, 35. Fg5-h6, 36. Fg5-f4, 37. Fg5-e3, 38. Fg5-d2, 39. Fg5-f6, 40. Fg5-e7, 41. Fg5-d8.

Az 1. ábrán láthatjuk ennek a függvénynek a képét, melynek kiszámításánál az eddigiekben is felhasznált 832 nagymesterjátszmát vettük alapul, és ezek segítségével készülték az átlagértékek. A függvény a 35. féléppéig meredeken növekvő, itt éri el a maximumot, majd folyamatosan csökken a 129. féléppéig. A 2. ábra azt mutatja, hogy a nyerő játékos figurái átlagban hány mezővel ellenőrzik többet a vesztes fél figuráinál. A maximális érték majdnem 3.

2. számítási mód: saját mezők támadása.

A sakktablát két részre osztjuk: világos-térffélre, amely az első négy sort foglalja magába, és sötét-térffélre, amelyhez az ötödik, a hatodik, a hetedik és a nyolcadik sor tartozik. Ez a felosztás mindkét színnel meghatározza a saját térfél és az ellenfél térfelének részét. Ennél a számítási módszernél csak azokat a mezőket vesszük figyelembe, amelyek a saját térfélen vannak, de az előbb ismertetett módszerrel ellentétben, az ellenőrzött mezőkhöz azokat is hozzászámítjuk, amelyeknek saját figurák állnak. Ez a függvény a bemutatott hadállásra világossal és sötéttel egyaránt ugyanazt az értéket adja, ötvennégyet.

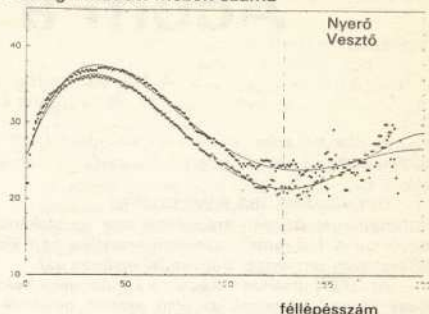
Nézzük, hogy ebbe az értékbe mely lépéseket számoltuk bele: 1. Bc1-a1, 2. Bc1-b1, 3. Bc1-d1, 4. Bc1-e1, 5. Bc1-c2, 6. Bc1-c3, 7. Be1-d1, 8. Be1-c1, 9. Be1-f1, 10. Be1-g1, 11. Kg1-f1, 12. Kg1-f2, 13. Kg1-g2, 14. Kg1-h2, 15. Kg1-h1, 16. e2×d3, 17. e2×f3, 18. f2×e3, 19. f2×g3, 20. Fg2-f1, 21. Fg2-h1, 22. Fg2-h3, 23. Fg2-f3, 24. h2×g3, 25. a3×b4, 26. Hc3-b1, 27. Hc3-a2, 28. Hc3-a4, 29. Hc3-e4, 30. Hc3-e2, 31. Hc3-d1, 32. Hf3-d2, 33. Hf3-d4, 34. Hf3-h4, 35. Hf3-h2, 36. Hf3-g1, 37. Hf3-d1, 38. g3×f4, 39. g3-h4, 40. Vh4-b4, 41. Vh4-c4, 42. Vh4-d4, 43. Vh4-e4, 44. Vh4-f4, 45. Vh4-g4, 46. Vh4-h3, 47. Vh4-h2, 48. Vh4-g3, 49. Vh4×h5, 50. Fg5-f4, 51. Fg5-e3, 52. Fg5-d2, 53. Fg5-c1, 54. Fg5-h4.

A 3. ábrán a függvény vonalán láthatjuk, hogy egészen a végjátékig szinte együtt halad a világos és a sötét gráfja. A végjátékban viszont leg több esetben a gyalogbevitel támogatása miatt a két király szinte helyet cserél, és ezért a világos és a sötét függvénye kettéválik.

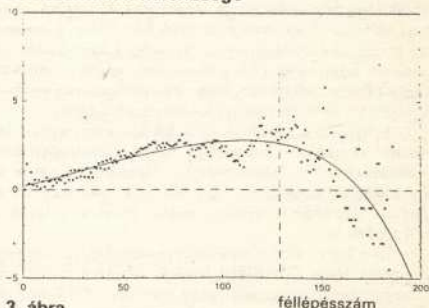
A nyerő és a vesztes játékos függvényének különbségét mutatja a 4. ábra.

(Folytatjuk)
Kovács P. Attila

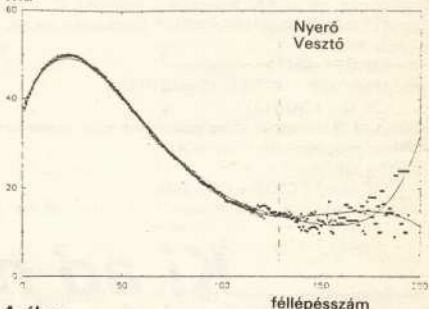
1. ábra
A megtámadott mezők száma



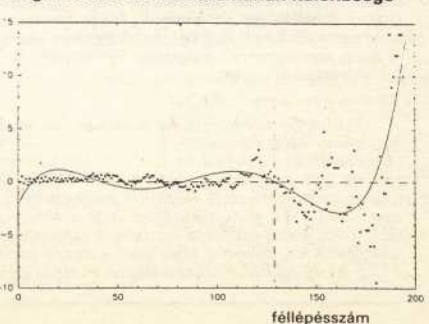
2. ábra
A nyerő és a vesztes fél által megtámadott mezők számának különbsége



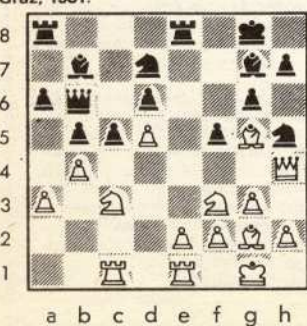
3. ábra
A saját térfélen lévő, megtámadott mezők száma



4. ábra
A nyerő és a vesztes játékos saját térfélen lévő megtámadott mezők számának különbsége



Kasparov — Fedorovics,
Graz, 1981.



PROGRAMTERMÉK

A nyuszi olvasni tanít

A „kicsik”-ről hajlamosak vagyunk megfeledkezni, különösen, ha már nem az első gyerekjáv van szó! A kicsik különben is a maguk sajátjánál inkább zárkózva élnek, erősen elkülönülve az okos felnőttektől, akik képtelenek megérteni őket. Pedig a technika iránti vonzalom a kicsik körében talán még erősebb, mint az idősöbbs korosztályokban. Nincs az a felnőtt, aki úgy ismerne a legújabb autómárkákat, mint némelyik kicsi. Úgy tűnik, hogy a kicsik képességeit jobbra csak elrontani tudjuk.

Mindez annak kapcsán jutott eszembe, amit a Novotrade Játékstúdió vezetője amerikai úti beszámolójában említett. Egyik amerikai partnerük a játékprogramok mellett olyan oktatási programok forgalmazásából él, amelyek a kicsiknek írtak. Feltehetőleg magának a kérdést, vajon nálunk, Magyarországon mi a helyzet ezen a téren. Gondolunk-e mi eléggé a kicsikre? A helyzet nem túl rózsás, még jó, hogy nem teljesen nemleges a válasz.

A Novotrade katalógusát böngészve, a tized sem éri el a kicsik „fogysztására” alkalmasnak tűnő programok száma, s ezeknek is csak a feléről derül ki, hogy azokat a kicsiknek szánták. A témák szerencsére elég változatosak: olvasás, számolás, söt, idegennyelv-tanulás. Más terjesztőknek pedig egyáltalán nem lehetett ilyen programokat felderíteni. Vajon miért nem írunk a kicsiknek?

Az okok két irányba vezetnek. Az egyik ok szerintem az a tévhit, ami egy felmérésen alapul. Eszerint a gyerekek az 5–7. osztályos korban kapja el a „számítógépkort”, ha egyáltalán elkapja, előbb tehát nem is érdemes a dolgot erőltetni. A felmérők figyelme, úgy érzem, akkor irányult, mikor kezdenek el a gyerekek programozni. Nos, biztosan akkor, amikor értelmi fejlődésük előttr odáig (ameddig némelyik „okos” felnőttnek sem ér el akár élete végéig), hogy algoritmusok pontos megfogalmazásával és szigorú betartásával meg tudják oldani a környezetükben felmerülő problémákat — legyen az házi feladat, a videokazetták nyilvánartása, vagy a haverok nevének diszkrétessen ki nyomtatása.

Mindez nem lehet ok arra, hogy a legifjabbak kapcsolatkeresését a számítógéppel későbbre halasszuk, amikor az iskolában már jókora adag kudarcélmény gyűlik össze bennük. Az újabb kudarcokkal fenyegető számítógép az új technika iránti lelkesedést méltán elhoszta. Pedig éppen az első osztályosok azok, akik még mindent „rözsaszinny” látnak, amit jobban ki kellene használni a számítógép befogadtatására. A cél úgysem lehet az, hogy mindezentől programozót csináljunk (de ezt a cént most már többen és sokszor lerágták).

A másik ok, ami miatt nem szaporodnak nálunk a kicsiknek írt programok, jóval alapvetőbb, és visszavezethető az első bekezdésben mondottakra. A felnőttek és a kicsik egészen más világban élnek és gondolkodnak. A kicsik világa mesevilág, amelyben a reális és az irrális határai elmosódnak. A kicsik gondolataikat szövegekben, metaforákban fogalmazzzák meg, amit ok egymás között jól megértének, mi okos felnőttek pedig szenvedünk tőlük. A felnőttek nincsenek tekintettel a kicsik lassú reakcióidőire, erőszakkal gyorsabb tempót diktálnak, amit a kicsik dacolva fogadnak, s csak még jobban leblokkolnak. Amit a gyerek végtelesszer örömmel ismételt, az a felnőtt az második, ismétlés után már halálra idegesíti. Ami a felnőttnek érdekes, az a gyereknek végtelesen unalmas lehet, pillanatok alatt türelmet veszti tőle.

A sort még lehet folytatni. Számunkra most ebből az a fontos, hogy aki a kicsiknek akar

programot írni, az nagyon kösse fel az alsóneműt az említett különbözőségek feloldásához. Nem mindennapi teljesítmény beelőni magunkat a gyermeki világba. Ezek után talán érthetőbb, miért vállalkoznak olyan kevesen kicsiknek szánt programok írására. Egyben kijelenthető az is, hogy aki ilyen programokat sikerrel ír, az méltán érdemel nagy dicsőretet.

Ehavi vizsgálatunk tárgya a Novotrade egyik sikeres, kicsiknek szóló, olvasást tanító programcsomagja, ami a katalógusban érdekes módon a játékprogramok közé sorolot, pedig egészen becsületes oktatóprogram. (Megjegyzem, egy hasonló amerikai terméket is szívesen megnéznék, de erre pillanatnyilag nincs mód.)

A programcsomag tervezői messzemenően figyelembe vették a korosztály képességeit: a kezelés felettebb egyszerű, igazodik a kicsik egyneihez. Kereshi a kurzorvezérlő gombokkal lehet, kiválasztani pedig a térköz (SPACE) billentyűvel. A képernyő bal felső sarkában egy igen élethű, animált számítógép-billentyűzt képeke folyamatban jelzi, hogy mit vár a program. A kurzorvezérlők egy nyilat mozgatnak. A nyil mozgása „méltóságjel”, ami tökéletesen illeszkedik a kicsik elvárható reakcióidejéhez. A programcsomagot az iskolában persze mágneslemezzel lehetne jól használni. Sajnos, meg kell említenem, hogy a kazetták betöltésével szokatlanul sok bajom volt: szinte ahány kazetta, annyi fejbállítás igényelt. A kazettasokszorosítás minőségén — úgy tűnik — volna még javítani való a Novotrade-ben.

A dokumentáció elég rövid létegetű, de a megcélzott felhasználók még úgy sem tudnak olvasni, hiszen éppen most tanulják. Rajtuk a program képekkal megvalósított öndokumentáló jellege segít, amit a készítőik sikerrel meg is oldottak. Tovább javítja a helyzetet, hogy a programcsomag teljesen az általános iskola első osztályos tananyagához igazodik. Ez ugye akkor alkalmasnak teszi arra, hogy a napközis foglalkozásokat számítógépes gyakorlatokkal lehessen színesíteni. (Vagy én túl naiv vagyok?)

Az olvasást a képernyő bal felső sarkában figyelő — a gyerekeknek feltétlenül kedves — állat, egy nyuszi segíti. Először a kis- és a nagybetűket tanuljuk a képernyőn kirajzolt képekkel összekötve, majd szövegek betűzésen keresztül jutunk el a szövegekhez és szavakhoz, hogy aztán a képekkéhez kapcsolódó más szavak felismerésével az olvasási készség mind jobban elmélyüljön. A csúcspont az utolsó program, amelyben három kép alapján egy-egy mesét olvashatunk el. A mese szövegeiből kiemelt szavak értelme alapján rá kell mutatni a képen látható tárgyra, személyekre. Az öt-

let nagyon szellemes, s mivel néhány tárgy a képernyő több pontján is szerepel, amelyeket tehát mind el kell fogadni helyes megoldásként, a program írói eléggé meg is izdadhattak érte.

A képecskék kifejezetten szépek, ami a kicsik szempontjából megint elég fontos körülmény. Valamivel meg kell fogni fantáziájukat, hogy ne hagyják ott a programot rögtön az első próbálkozás után. A képek ráadásul meg is mozdulnak. Az animáció egyes képeknél azonban nem valami tökéletes, sőt, erőltetett is időrabló. Emiatt a kicsikben veszélyesen megjelenő a türelmetlenség faktor! A hat program egyébként határozott fejlődést mutat abban, hogy mennyire idomul a kicsik egyneihez. Mindenképpen az utolsó program a legsikerültebb.

A jutalmazási és büntetési rendszer a megszkott. A jó megoldást a nyuszi rögtön mosollyal nyugtatja, rossz megoldás és ügyetlenkedés miatt morog és mérges arcot váz. A kicsiket valószínűleg nagyon bosszanthatja ugyan („ez nem igazság!”), hogy néha azért is megbüntetik őket, mert a nyilat nem jó pontra irányítják: a sző közelébe, de nem pontosan rá. Persze nem baj, ha a program menet közben pontosságra is nevel. Egy kép vagy képeges helyes megoldása után kapunk egy répát ajándékba. Ha túl sok hibát vétünk — az elkövetett hibák száma programonként változó —, akkor a program a feladatot megismételteti, s addig nem enged tovább, amíg az előrt időn belül gyakorlatilag hibátlanul meg nem oldottuk a feladatot. A jutalmazási rendszernek még fokozata is van. Maximálisan négy répát lehet összegyűjteni, azután a programok előlr kezdődnek. Az újrakérdést erőltethetjük a RESET gomb megnyomásával. Ilyenkor egyes programok véletlenül adnak új feladatot.

A programcsomagban kevés kifejezett hibát lehetett felfedezni. Az egyike az animációval kapcsolatban már említettet. Az első programban a képében találtam hibát. A Lyuk képe a Nyuszi után következett, ami enyhén szólva meglepő, hiszen az ly az l betű után következik. Az ábcébol persze hiányzott a dz, ds, q, x és y, amin igazán nem lehet csodálkozni, hiszen ezekkel a betűkkel főleg csak idegen szavak kezdődhetnek.

A programcsomag egyetlen komoly hátránya, hogy drága. Talán inkább két kazettára kellett volna összehasználni a programokat, így kisebb lenne a csomagolási költség. A programcsomagot ezzel együtt kiválóan ítélem, mint a mellékelt minősítő adatok is mutatják. Ebben az ítéletben nem az elért kasszasiker befolyásolt, hanem a kicsik lelkeletének megértését bizonyító megoldások.

A programcsomag az általános iskolai munkát és a kicsik önálló gyakorlati eredményes támogatását. Jogosan aratot nagy sikert, mert jó példája annak, hogy a felnőtt is belehelyezkedhet a gyermeki világba, amiért külön köszönet illeti a szerzőket.

Zsadányi Pál

MINŐSÍTŐ ADATOK

Kezelhetőség:	kiváló
Teljesség:	kiváló
Dokumentáltság:	jó
Használhatóság:	kiváló
Ár/teljesítmény:	közepes
Összbenyomás:	kiváló

ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Novotrade
Terméknév:	A Nyuszi olvasni tanít 1-6.
Szerzők:	Fekete Mónika Ferencz András Papp György
Géptípus:	C Plus/4
Hordozó:	hat kazetta
Dokumentáció:	két-két oldal a kazettaborítókön 1-5. rész: 306 Ft/kazetta 6. rész: 490 Ft Együtt: 1165 Ft
Ár:	

Bakó András:
AutoCAD grafikus tervező rendszer
(Budapest, 1989. SZÁMALK, 243 oldal.
Ára: 280 Ft)

Az AutoCAD egy összetett, több részből álló programcsomag. Alapja az AutoCAD Drafting Package, mely számítógéppel tervezett kétdimenziós ábrák elkészítését teszi lehetővé. Többféle kiegészítése is létezik a programrendszernek, az újabb verziókkal már háromdimenziós ábrák is rajzolhatók. Az egyik kiterjesztés új szolgáltatása, hogy tartalmaz egy AutoLISP nevű fordítóprogramot, melynek segítségével speciális felhasználási területeken is alkalmazható az AutoCAD.

Bartha Attila:
NORTON Utilities, Integrator, Editor,
Commander Guide
(Budapest, 1989. LSI ATSZ,
96 oldal. Ára: 150 Ft)

Ez a könyv olyan segédesszközöket ismertet, amelyek nélkül szinte elképzelhetetlen a személyi számítógéppel dolgozók élete. „PC-s” körökben, főleg a rendszerprogramozók között, a Norton név fogalom. Norton mindenütt megtalálható: a könyvespolcon és az autoexec-ben is. A módszer zseniális, ahogy bevezet a számítógép rejtelmeibe. Bitekre boncolja a gépet, hogy megismeressen a működésével. Programjai szinte mindent kifürkészenek, a legváratlanabb helyzetekre is felkészültek, és mindezek ellenére rendkívül egyszerűen és jól kezelhetők.

A könyvben található leírások megismertetik az olvasót a Norton-szoftverekkel, mégpedig úgy, hogy azok bármilyen szinten használhatók legyenek. A Norton-szoftverek nemcsak hasznosságuk miatt népszerűek, hanem azért is, mert nagyon jól kezelhetők, könnyen megtanulhatók, és a DOS használatát kényelmessé és gyorsá teszik. E programok segítségével egyúttal közelebb kerülhet az olvasó a PC-hez.

Úry László:
SYMPHONY 2. Tartományok,
adatállományok, ablakkezelés,
szövegszerkesztő
(Budapest, 1988. LSI ATSZ,
90 oldal. Ára: 133 Ft)

A több füzetből álló sorozat második könyvecskéjét tartja a kezében az olvasó. Az előző füzet a SYMPHONY 2. Tartományok és programozásával foglalkozik, ez a füzet a főmenük még nem ismertett lehetőségeivel, köztük a tartományok használatával, a SYMPHONY 2. füzetével és az ablaktechnikával ismerteti meg az olvasót. Ezen túlmenően részletesen tárgyalja a SYMPHONY 2. szövegszerkesztőjét, a

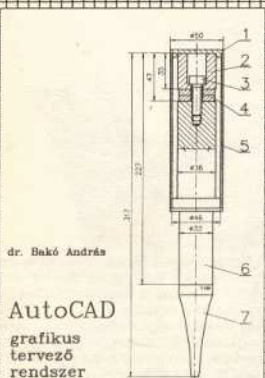
szövegszerkesztő és a kalkulációs lap közös használatát.

Számítástechnikai statisztikai zsebkönyv 1988.
(Budapest, 1989.
Statisztikai Kiadó Vállalat,
152 oldal. Ára: 85 Ft)

A statisztikai zsebkönyvcsalád legújabb kötete a számítástechnika hazai alkalmazásáról nyújt átfogó tájékoztatást. Ismerteti a különböző teljesítményű, eredetű, korú és funkciójú számítógépek állományadatait és megoszlását, a hozzájuk kapcsolódó berendezéseket. Képet ad az alkalmazói tevékenység gazdasági eredményeiről és a programforgalomról. Összehasonlító adatokat közöl a legfontosabb nyugat-európai országok számítógépparkjáról és prognosztizálja a következő években a miniszámítógépek és a szoftvertermékek várható piaci forgalmát.

Orosz Judit:
IBM PC-hálózatok áttekintése.
LAN rendszerek: Novell NetWare,
PC Network, 3Com3+
(Budapest, 1988. LSI ATSZ,
51 oldal. Ára: 123 Ft)

A kiadvány célja, hogy az IBM személyi számítógépen alkalmazott lokális hálózatok alapjaival ismertesse meg az olvasót. Az első három fejezet a kezdőknek szól, betekintést nyújt a hálózatok alapvető tulajdonságaiba. Kiter felépítési szabványaikra, piaci viszonyaikra, valamint a hálózatok alapját képező operációs rendszerrel, az MS-DOS-szal való kapcsolataikra. A könyv második fele a haladó ismereteket foglalja ki kiválóan felépített rendszerbe, erős elméleti alapot szolgáltatva ezáltal. Foglalkozik hálózatok egymáshoz kapcsolásával, a NETBIOS-szal. Az alapok ismertetését a különböző hálózati implementációk leírásával teszi teljessé a szerző.



dr. Bakó András

AutoCAD
grafikus
tervező
rendszer

A könyv célja, hogy bemutassa az AutoCAD rajzolási, rajzszerkesztési funkcióit. Az egyes utasításokat kezdő és haladó szintű felhasználók számára is érthetően tárgyalja.

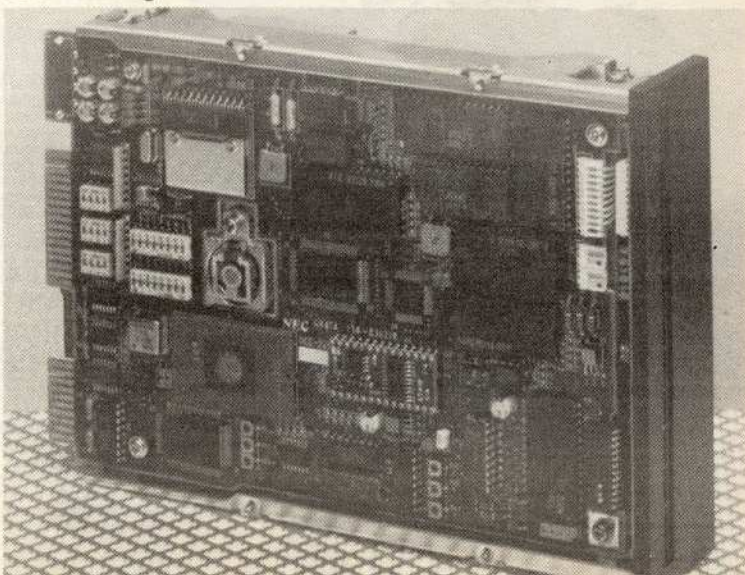
Az AutoCAD-et többféle számítógépen is lehet futtatni. A könyv a hazánkban nagyon elterjedt IBM XT/AT családon való működését mutatja be. A rendszer különböző perifériákat tud kezelni, egér, digitálisizáló, rajzgep is csatlakoztatható az alapgéphez. Külön fejezet foglalkozik a rendszer üzembe helyezésével és installálásával.

Soltész Erzsébet — Sipos Győző
Szövegszerkesztők az IBM PC számítógépekhez.
WordStar, ChiWriter
(Budapest, 1989. LSI ATSZ,
289 oldal. Ára: 248 Ft)

E könyvsorozat darabjai egyaránt szólnak kezdőkhoz és haladókhöz. Azokhoz, akik szövegeket, cikkeket, tanulmányokat vagy könyvet akarnak írni, illetve szerkeszteni. A kötet első része azokat az alapismereteket tartalmazza, amelyekkel a kezdő programozóknak vagy a gépiroknak feltétlenül meg kell ismerkedni a szövegszerkesztők használata előtt.

A könyvben bemutatott két szerkesztőrendszer eltérő stílusú. Nagyobb anyagok, könyvek szerkesztéséhez több opciója miatt inkább a WordStar ajánlható. A ChiWriter jobban használható rövidebb anyagok, elsősorban matematikai vagy más, speciális szimbólumokat alkalmazó szövegek írásához.

180 Mbájtos winchester



A winchester-meghajtók az utóbbi évtizedekben a számítógépek három legfontosabb építőeleme közé kerültek. A merevlemez kapacitása, hozzáférési ideje és mérete a számítástechnikai alkalmazások szempontjából meghatározóvá vált. Nemrégiben még örültünk, ha gépünkhöz 40 megás meghajtót vásárolhatunk, ma már a 100 megásak sem számítanak ritkaságnak. A NEC Deutschland GmbH az idei hannoveri CeBIT kiállításon egy új winchester-családot mutatott be. Az 5,25 inches félmagas (mindössze 41 mm, tolvajnyelven slimline, ejtsd szímlájn) meghajtó kapacitása 179,8 Mbajt (nem formázott), átlagos hozzáférési ideje 18 ms, ami rendkívül kicsinek számít.

Írjunk a képernyőre!



Fénylemeztorony

Az egyesült államokbeli CBIS nevű cég rendszere helyi hálózatba kapcsolt DOS gépek számára maximum 14 fénylemezhez nyújt közvetlen hozzáférési lehetőséget. A fénylemez (angolból vett szakzsargonban: CD-ROM — compact disc read only memory) mint tudjuk, 600 Mbajt tárolására alkalmas „High Sierra” (ISO 9600) adatformátumban. A képen látható torony hét CD-ROM-meghajtót és egy 5,25 inches légylemezjátszót tartalmaz. Egy-egy fénylemez kisebb könyvtárnyi információt ölel fel.

Szoktalan számítógépet hozott forgalomba a Linus Technologies. A 4 kg tömegű, hordozható gép tulajdonképpen egy lapos doboz, amit az asztalra fektetve használnak: felnyitva a tetejét, a belőle kihúzható ceruzával lehet a fekvő képernyőre írni. (Lásd a képet.)

A képernyőt borító írófelület — az átátlátszóságtól eltekintve — nem különbözik a rajzdigitalizálóknál megszokottól. A képernyőn ott jelenik meg a rajz vagy a kézírás, ahol a ceruzával dolgozunk, és közben el is tárolódik. Az IBM-kompatibilis gépeknek a szokásos be/kimenetei vannak, így művünk ki is nyomtatható.

Az új gép fő alkalmazási területe az egészségügy, ahol a képernyőre kivettített kérdőívet kell kitölteni. Minden új felhasználó kézírását egy 30 perces betanulási folyamat során sajátítja el a gép, a régiéket pedig használat előtt lemezről kell betölteni.

A berendezés tömeges elterjedését egyelőre az akadályozza, hogy ára hatszorosa egy XT-kompatibilis gépének.

Mit érnek a mikrofloppyk?

A Memory Control Technology nevű cég megvizsgálta az Egyesült Államok piacán legnagyobb példányszámban forgalmazott 25-féle, 3,5 inch méretű ún. mikrofloppykat. (Az 5,25 inches lemezekre itthon használt hajlékonylemez megnevezés helyett itt „hajlékonylemez merev tokban”-t kellene mondani, de ez olyan hosszú, hogy vagy a mikrofloppy, vagy a 3,5 inches floppyt célszerű használni.)

A vizsgálat során csak négy — a C. Itoh, az IBM, a Sony és a TDK — gyártmány felelt meg az előírásoknak. A leggyengébbnek a Xidex, a Wabash, a Dysan és az SKC bizonyult.

Az itthoni felhasználók figyelmét nyomtatékosan felhívjuk arra, hogy a nálunk kaphatók közül egyedül a Sony gyártmánya jó. Az időnként ugyancsak kapható Wabash és Dysan igen gyenge, és — ami talán a legfontosabb — az itthon csaknem egyeduralkodó 3M, amelyet szintén vizsgáltak, nem szerepel az amerikai szabványok megfelelők között.

Digitális papír mint tömegtartó

A brit ICI Electronics egy egészen új fajta optikai adahordozót fejlesztett ki, amely ötszázszor több információt képes rögzíteni, mint a jelenleg használatban lévő optikai (fény)lemezek vagy mágnesszalagok. A digitális papírnak nevezett új adathordozó rugalmas és ellenálló, mégis olcsón gyártható. A fajlagos tárolási költség mindössze 0,3 penny (kb. 0,30 Ft) megbájtónként, vagyis tízszer olcsóbb, mint a tárolás a mágnesszalagon. Az információörögzítés módja megegyezik a fénylemezével, vagyis a bináris 1-nek egy parányi, lézérfényvel beégetett gödörccske, a 0-nak ennek hiánya felel meg. Az újdonságot az a különleges gyártási eljárás jelenti, amely lehetővé teszi, hogy a fényérzékeny réteget hajlékony alapanyagra vigyék fel, amely így egészen új alkalmazási területeket jelent.

Az Olvasó írja című rovatunk anyagtorlódás miatt e számunkból kimaradt, de leveleiket természetesen továbbra is várjuk. A szerkesztőség



SCI-L

*IBM PC XT/AT-vel
kompatibilis professzionális
személyi számítógépek*



Ajánlatunkban

P—16/AT ALAPKONFIGURÁCIÓ

1 Mbájt RAM-mal 188 000,— Ft
640 kbájt RAM-mal 180 000,— Ft

P—16/XT ALAPKONFIGURÁCIÓ

640 kbájt RAM-mal 120 000,— Ft

Rövid határidőre szállítjuk az alábbi perifériákat:

PANASONIC KX—P 1540 mátrixnyomtató (24 tús)
EGA monitor+ csatolóegység
VGA monitor+ csatolóegység
lézernyomtató, digitalizálók, plotterek, scanner, egér

GARANCIA NÉLKÜLI REKLÁMÁRAINK:

TRS aszinkron terminál: 14 000,— Ft
TRS mátrixnyomtató: 19 500,— Ft
FX—100 mátrixnyomtató: 49 000,— Ft
UNIBOARD billentyűzet: 9900,— Ft

TOVÁBBI BŐVÍTÉSI LEHETŐSÉGEINKRŐL KÉRJÉK ÁRJEGYZÉKÜNKET

Sci—L Számítástechnikai Informatikai Fejlesztő Leányvállalat
Kereskedelmi Iroda
Budapest, Iskola u. 10. 1011. Telefon: 154-065 vagy 350-180/180, 181, 182, 183, 184.

Telex: 22-4599. Telefax: 35 39 15



079
LASER 2077

A KERSZI nagyszámítógépei

2 x 8 megabájt központi tárral,
1,2 millió művelet/másodperc se-
bességgel, 4 millió gépelt oldal
hátterkapacitással az Önök
„álomigényeit” is valósággá va-
rázsolják.

EXTRA szolgáltatások:

ION—LASER nyomtatón: kiad-
ványok, szórólapok, bizonyla-
tok rövid határidős nyomtatá-
sa. **IRÁNYÁR:** 4 Ft/lap

TÁV-ADATFELDOLGOZÁS

IBM PC-kapcsolat

**FLOPPY-konvertálás extra se-
bességgel**

szalagról floppyra, floppyról
szalagra, 800 Ft/floppy

MI ÖNÖKÉRT DOLGOZUNK!