Ára: 30 Ft

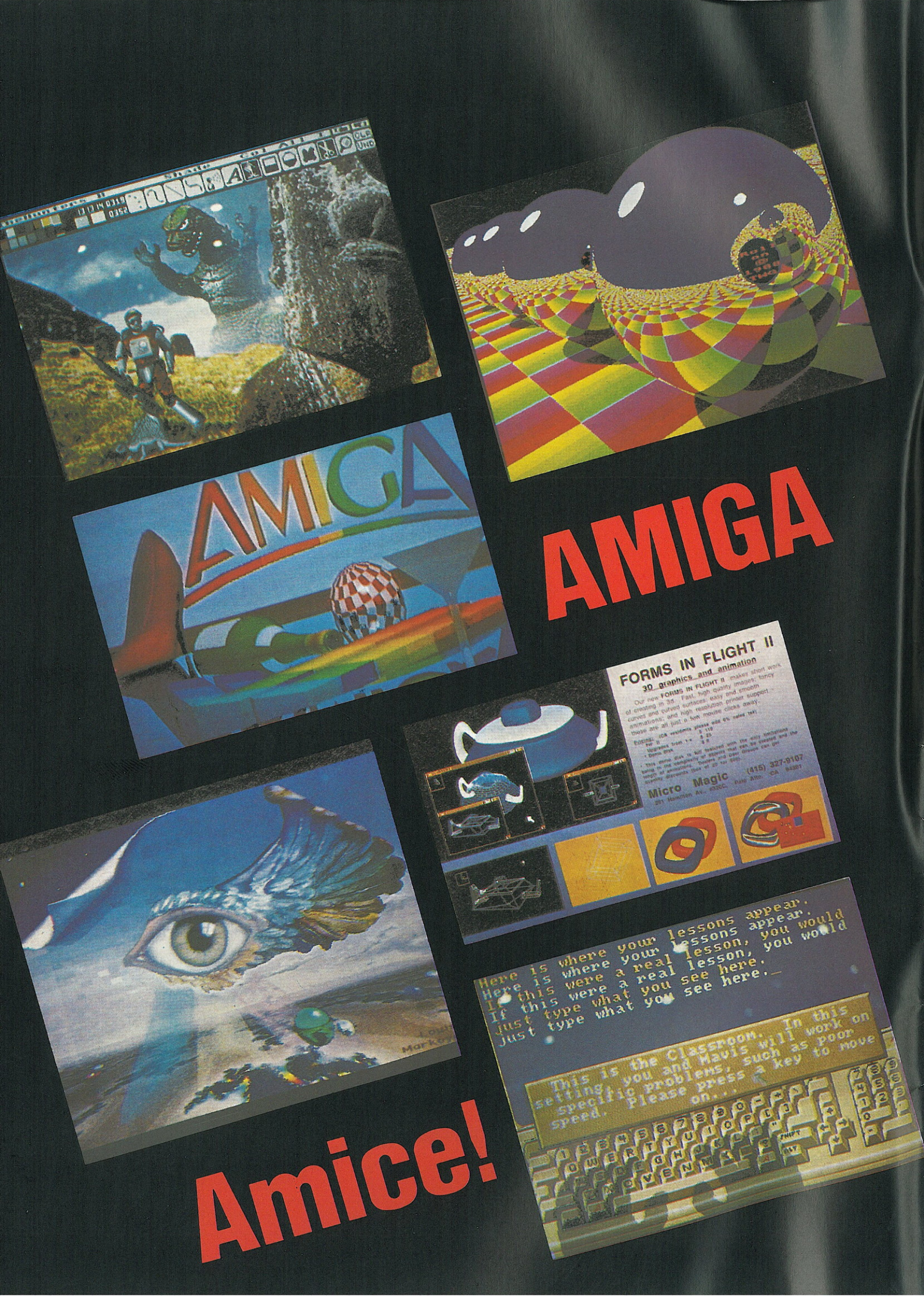
mikro számítógép magazin

Ne félj, csak szellemek!

Miénk a vár: Amiga!

Így írtok ti... programot

1989/6


AMIGA
AMIGA
FORMS IN FLIGHT II
3D graphics and animation
Micro Magic
(415) 327-9107
Here is where your lessons appear.
If this were a real lesson, you would
Just type what you see here.
This is the Classroom. In this
setting you and Mavis will work on
specific problems, such as poor
speed. Please press a key to move
on...
Amice!

mikro számítógép magazin

7. ÉVFOLYAM
1989/6. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

Címképünk:
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 főtitkárhelyettes

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

Megjelenik havonta.
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, Pf. 149
és a Magyar Média
1932 Budapest, Pf. 279
88—1135

Szikra Lapnyomda
Budapest (89—0699)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

TARTALOM

„. . . Jól tudjuk: az emberben minden paradoxon. Biztosítjuk valakinek a megélhetését, hogy alkotni tudjon, és ő elalszik, a győzedelmes hadvezér elpuhul, s a nagylelkű, ha meggazdagszik, fukarrá válik. Mit számíthatnak a politikai elméletek, amelyek fel akarják virágoztatni az embereket, amíg nem tudjuk, milyen fajta embereket fognak felvirágoztatni? Mi fog megszületni belőlük? Mégsem vagyunk hizlalásra szánt csorda, s egy szegény Pascal megjelenésének sokkal nagyobb jelentősége van, mint száz névtelen pocakosnak."

(Antoine de Saint-Exupéry: Az ember földje)

Hétköznapi ünnepeim

Éppen a közepén vagyunk a 4. Országos Mikroszámítógépes Találkozónak, vagy népszerű nevén nevezve az „ünnepet" a μ'89 eseményeinek, amikor a szerkesztőség ultimátuma miatt tovább már nem húzhatom-halaszthatom a szerkesztőségi cikk megírását. Miről is írhatnék ilyenkor, mint a Mikróról, meg egyáltalán arról, hogy szükség van-e még a társadalom informatizálására.

Teszem ezt azért is, mert időközben a szomszédaink is valami hasonlót kezdtek el, tavaly novemberben az osztrákok rendezték meg hasonló, de csak az ifjúságnak szóló összejövetelüket; és persze büszkék lehetünk, mert a meghívójuk szerint a μ'88 adta az ötletet a „Jugend und Computer" kiállítás és konferencia létrehozásához.

Az idén — sok év után — ismét eljutottam a hannoveri vásárra, ahol, nem kis meglepetésemre, a 19-es pavilonban megtaláltam a Mikro német megfelelőjét „Computer Camp" címen.

Az idén ismét sok külföldi vendégünk volt, elsősorban a Szovjetunióból, azután Görögországból, Bulgáriából, de megszólított a pavilonban egy osztrák, egy német és két olasz szakmabeli is, miután megmondta valaki nekik, hogy van némi közöm a találkozóhoz. Szerénytelenség volna ismételni a véleményüket, én annak örültem elsősorban, hogy észrevették a kiállítás mondanivalóját, nevezetesen azt, hogy sikerült egy helyre összehoznunk a szakma legjobbjait, az amatőröket, a diákokat, a számítástechnika-történetet, a képző- és zeneművészetet. Tetszett nekik az is, hogy találkozókat, fontos tanácskozásokat rendezünk, amelyeken a számítástechnika lelkes hívei fejtik ki nézetüket például a szerkesztőségek elektronizálásáról, a művelődési házaknak az informatika elterjesztésében betöltött szerepéről, a számítástechnikának az orvostudományban való alkalmazásáról.

Nekem tulajdonképpen fel sem tűnt, ami németországi látogatónknak rögtön szemet szúrt, hogy a 25-ös pavilonban összehoztuk a számítástechnikát művelők egész társadalmát, sőt azt is megmutattuk, hogy ez így teljes, ahogyan van, így együtt diákok, orvosok, újságírók, programozók, mérnökök és természetesen az érdeklődő nagyközönség.

Nem szeretném elhallgatni a negatív felhangokat sem, hiszen néhányan azt is megjegyezték, hogy csökkent a Mikro népszerűsége, amit az is mutat, hogy az idén a felére csökkentettük a kiállítási területet, a 24-es pavilont ugyanis átadtuk a borászoknak. Igaz, és az is igaz, hogy valamivel kevesebb cég állított ki, mint korábban. Sőt azt is elárulom, hogy a szervezőbizottság egy része az első összejövetelünkön arra szavazott, hogy ne is rendezzük meg az idén a Mikrót, hiszen ebben a gazdaságilag katasztrofális, spórolásos évben csődöt kell hogy mondjunk, mert a kutya se fog eljönni kiállítani. (Csak zárójelben jegyzem meg, a végén kevés volt a kiállítási terület, a 36 kiállítót csak úgy tudtuk elhelyezni, hogy a nagyok területéből egy kicsit lefaragtunk.) A kiállítók összetétele az elmúlt négy évben folyamatosan változott. A rendszeresen kiállító nagy cégek mellett — Videoton, Számalk, SZKI, SZÜV, HT — egyre több kisszövetkezet, gmk, pjt., kft., közös vállalat, leányvállalat és más gazdasági formáció hozta el termékeit, mutatta be azokat a programjait, amelyeket bárki könnyen alkalmazhat. Ezt a találkozót elsősorban a számítástechnikát alkalmazóknak szerveztük, akik a gépben és a programokban eszközt keresnek kézzel már megoldhatatlan feladataik ellátásához.

Nem tagadjuk, a találkozó kivételes alkalom az ifjúság számára is, egyrészt azért, mert itt minden cég szeretettel várja az ifjú alkalmazókat, tehát alkalmuk van „tárgyalni" a szakma legjobbjaival, másrészt ők is bemutathatják, hogy mit tudnak. Az idén az elmúlt évekhez képest a fiatalság érdeklődése — szerény becslések szerint is — megháromszorozódott. Az oktató- és tanulóprogram-pályázatra közel kétszáz jelentkezés érkezett, a szombat-vasárnapot nagyjából erre fordítottuk. Eljöttek az amatőrök is, néhányan már önálló vállalkozóként is bemutatkoztak, berendezéseik körül mindig nagy volt a tolongás, hiszen ezek a munkák azt sugallták, hogy ha egy nem számítástechnikus is képes egy jól működő berendezést megépíteni, akkor az biztosan másnak is sikerülhet. Eljött a „Hámán", itt voltak a művelődési házak, a sakkozók, a világhírű Chess-Base adatbankkal, aminek a bemutatója elhozta hozzánk az élsakkozókat is.

A bevezetőben elmondtam, hogy ma, amikor a cikket írom, még a program felénél tartunk. Visszavan a művelődési házak szakembereinek a találkozója, a Garay-verseny bemutatója, az orvosok megbeszélése. Visszavan még a Tourcomp kerekasztal is, a kiállítás — a számítástechnika az idegenforgalomban — és a kerekasztal az egyik legújabb szín a Mikro palettáján.

Jó szokásunk szerint a Mikrót lezáró szervezőbizottsági megbeszélésen el fog kezdődni a vita ismét, legyen-e vagy ne legyen újabb Mikro, ha legyen, akkor milyen újdonságokkal rukkoljunk elő.

Szerintem persze, legyen, mert Mikro nélkül nem teljes a Budapesti Tavaszi Fesztivál és az Utazás kiállítás – mondta egyik látogatónk. Ráadásul jövőre lesz ötéves a találkozó, tízéves a fesztivál, így még többet vár tőlünk a szakma is és a közönség is. Szeretném egy régi tervemet megvalósítani és sokkal nemzetközibbé avatni a találkozót. Nemcsak a külföldi cégek, de a külföldi társadalmi szervezetek jelentkezését is várjuk. Azt már eldöntöttük, hogy ismét lesz „A gép is ember" karikatúrakiállítás, már most van vagy ötven új rajzunk a világ minden tájáról.

Az idén az UPIC francia zenei központ sajnos — anyagi okok miatt — lemondta a szereplését, helyettük a szlovák rádió számítógépes zenei stúdiója szerepelt. Jövőre talán sikerül a franciákat idecsábítani, az anyagi problémákat megoldani. Szeretnék minél több humán alkalmazást is bemutatni, számítógépes tanulást, addigra talán a nyílt főiskola ügye is elmozdul a holtpontról.

Szóval terveink vannak, és minden reményünk is, hogy teljesüljenek. A biztos pont a közönség érdeklődése, amihez — bízom benne — megkapjuk a szakma, az amatőrök, az iskolák, egyszóval a számítástechnikát művelők támogatását.

Előre is köszönöm.

Kovács Győző

tanuljuk együtt!

A PASCAL REJTELMEI

10. Program másodfokú egyenlet megoldására

Szokásunkhoz híven következő programunk megírásának ürügyén újabb Pascal ismeretekkel hozakodunk elő. Már a kezdet kezdetén megemlítettük a standard adattípusok között a boolean típust is. Használatát eddig szándékosan kerültük, elsősorban a BASIC-en nevelkedettek miatt, mert féltünk, hogy pánikot keltünk vele. Most azonban, a Pascal tudományának bár csak kis részével, de már mégis felvértezve, reméljük, hogy olvasóink hozzá fognak szokni a boolean-fegyver csapásaihoz.

10.1 Boolean változók

Hogy a lényegre rávilágítsunk, válaszszunk egy igen egyszerű feladatot! Egy program folyamatosan egykarakteres adatokat vár mindaddig, míg a karakter nem az x betű; egy bebillentyűzött x hatására befejeződik. Talán nem lesz zavaró, ha először a program BASIC nyelven készült megoldását mutatjuk be *(31. ábra)*.

Ehhez hasonló feladatot már Pascalban is megoldottunk: egy programelágazást attól tettünk függővé, hogy a lenyomott billentyű az ESC vagy valami más volt-e (5.2 rész, 13. ábra). Nézzük meg most a megoldást a **boolean** típusú változó alkalmazásával! *(32. ábra.)*

A BASIC programtól való lényeges eltérés, hogy ott a vizsgálat (IF) egy reláció teljesülésére vagy nem teljesülésére terjed ki. A Pascal megoldásban a reláció kiértékelésének eredményét az **ezegyx** nevű változóban értékadással helyezzük el. Ha a relációban megfogalmazott feltétel (**ch='x'**) teljesül, az **ezegyx** változó értéke **true** (igaz), ha nem, **false** (hamis) lesz. A programelágazás az **until** segítségével valósul meg; a ciklusmag addig ismétlődik, amíg az **ezegyx** értéke hamis. Ha igazzá válik, a program befejeződik.

A **boolean** változók természetesen nemcsak relációk kiértékeléséből, hanem közvetlen értékadásból is kaphatnak értéket. Például az

ezigaz:=true;
tevedes:=false;

utasítások a két **boolean** típusú változónak az igaz, illetve a hamis értéket adják.

Jogos lehet a kérdés, hogy nem egyszerűbb-e a BASIC megoldása. Jelen esetben igen, de gondoljunk arra, hogy az IF után következő reláció sokkal bonyolultabb is lehet, és ha a program különböző helyein többször is ugyanezt a relációt kell elágazáshoz felhasználni, a kiértékelés sok időt vehet igénybe. Ha a Pascal kínálta lehetőséget választjuk, a reláció többszöri kiértékelése elmarad, mindössze a reláció igaz vagy hamis voltát tartalmazó **boolean** változó értékét kell csak megvizsgálni. Mellékesen még két szempont: az említett esetben a Pascal megoldás kevesebb írásmunkát is jelenthet, továbbá egy változó vizsgálata kevésbé bonyolítja, teszi nehezebben emészthetővé a programot, mint egy esetleg két bonyolult kifejezésből álló reláció sorozatos előfordulása.

Ha valaki mégis ragaszkodik a BASIC-re jellemző megoldáshoz, a Pascal ezt is megengedi. A *33. ábrán* bemutatott megoldással a program továbbra is kifogástalanul működik.

10.2 Az if . . then . . else utasítás

Az **if . . then . . else** az **if . . then** szerkezet egy javított változatának tekinthető. Az **if . . then** használata során — ha az **if** által vizsgált feltétel értéke **false** — az utasítás „nem hagy nyomot maga után". A javított szerkezetben az **else** után utasítást írhatunk, amely akkor hajtódik végre, ha a feltétel értéke false.

Használatát tulajdonképpen el lehet kerülni, de bizonyos esetekben alkalmazásával egyszerűbb, áttekinthetőbb programhoz jutunk. Az utasítások egymásba ágyazottan is használhatók. Ilyenkor természetes, hogy az **if** és az **else** szavak száma azonos. Ha egy ilyen programrészben egy **if . . then** is előfordul, jogosan merülhet fel a kérdés, hogy egy **else** mihez tartozik: eredeti szándékunk szerint az **if . . then . . else**-hez, vagy a **then** és az **else** közé beékelt **if . . then**-hez. Ilyenkor a fordító az **else**-t mindig a közvetlenül előtte álló **if . . then**-hez tartozónak tekinti. Programjaink készítésekor erre a tulajdonságra mindig legyünk tekintettel.

Ha már szabályokról esett szó, egy igen fontos szintaxis-kérdésről sem szabad megfeledkeznünk: az **else**-t megelőző utasítás végére nem tehetünk ; karaktert. Ha ezt a szabályt nem vesszük figyelembe, a **pontosvesszőt** a fordító az előtte lévő **if . . then** végének megjelöléseként értelmezi, és a később következő „magá-

```
10 INPUT A$
20 IF A$="x" THEN 30 ELSE 10
30 PRINT "X: Tehát vége"
40 END
```

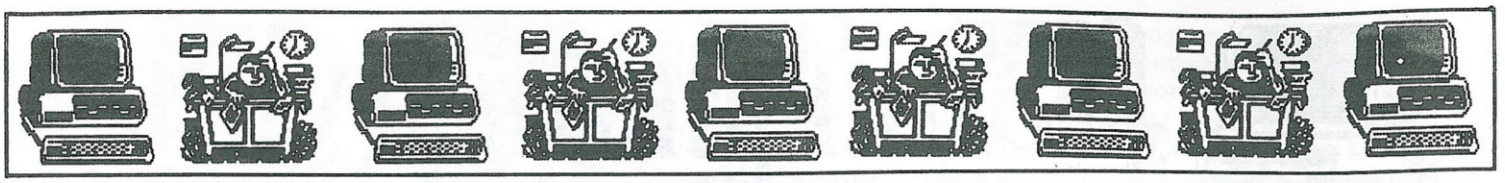
31. ábra

```
program xvagynemx;
var ch:char;
    ezegyx:boolean;
begin
   clrscr;
      repeat
         readln(ch);
         ezegyx:=ch='x';
      until ezegyx;
   writeln('x: tehát vége')
end.
```

32. ábra

33. ábra

```
program xvagynemx;
var ch:char;
    ezegyx:boolean;
begin
   clrscr;
      repeat
         readln(ch);
      until ch='x';
   writeln('x: tehát vége')
end.
```



nyos" **else**-zel nem tud mit kezdeni. Ez az egyetlen eset, amikor egy rossz helyre tett pontosvesszővel szintaxishibát tudunk előidézni.

Végezetül a *34. ábrán* egy igen egyszerű programot mutatunk be, amely két szám algebrai összegzését végzi el a lengyel módszerrel (először az operandus, azután a műveleti jel).

10.3 Az egyenletmegoldó program

A program három egyszerű — az együtthatók bevitelét, a diszkussziót és a megoldást végző — eljárásra épül. Hogy az új ismeretek: a **boolean** típusok és az **if .. then .. else** begyakorlását minél zavartalanabbul végezhessük, a program csak az egyszerű, már sokszor alkalmazott Pascal ismereteket használja, így a kelleténél talán kissé hosszabb is lett.

A lista bemutatása előtt még egy apróságra kell kitérni. A programban négyzetgyökvonást is kell végezni, ezért erre a célra egy még nem ismertetett Pascal eszközre, a függvények használatára is szükség volt. Erről most csak annyit, hogy az **sqrt** függvény az utána zárójelben lévő argumentum négyzetgyökét számítja ki. A programlista a *35. ábrán* látható.

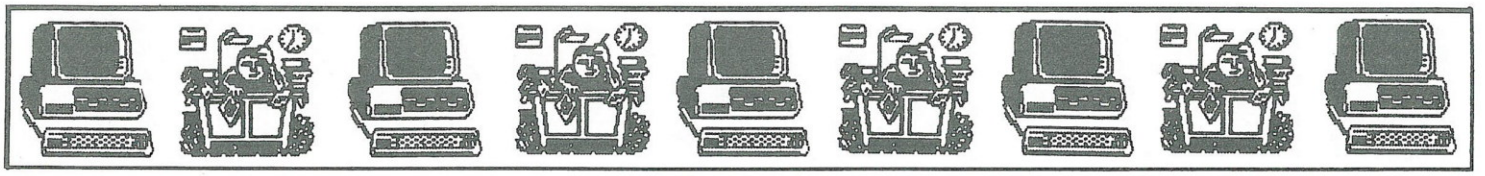
Nagy Imre


```
program lengyel;
var a,b,summa:real;
    jel:char;
begin
    clrscr;writeln('A két adat:');
    readln(a);
    readln(b);
    writeln('Összeadás:+, Kivonás:minden más jel');
    readln(jel);
    if jel='+'
       then summa:=a+b
       else summa:=a-b;
    writeln(summa)
end.
```

34. ábra

35. ábra

```
program masodfoku;
{boolean változók, függvények, if-then-else}
var a,b,c,d,x1,x2:real;
    nemmasodfoku,nincsvalosgyok:boolean;
procedure bevitel;
begin
   writeln('Kérem az A,B,C együtthatókat!');
   write('A=');readln(a);
   write('B=');readln(b);
   write('C=');readln(c)
end;
procedure diszkusszio;
begin
   nemmasodfoku:=a=0;
   d:=b*b-4*a*c;
   nincsvalosgyok:=d<0
end;
procedure megoldas;
begin
   x1:=(-b+sqrt(d))/(2*a);
   x2:=(-b-sqrt(d))/2/a
end;
{föprogram kezdet}
begin
   clrscr;writeln('Másodfokú egyenlet megoldása');
   writeln;bevitel;
   diszkusszio;
   if nemmasodfoku
      then begin
              writeln;
              writeln('Nem másodfokú egyenlet')
           end
      else
         if nincsvalosgyok
            then begin
                    writeln;
                    writeln('Nincs valós gyök')
                 end
            else begin
                    megoldas;
                    writeln;writeln('A gyökök:');
                    writeln('X1=',x1);
                    writeln('X2=',x2)
                 end
end.
```



Más billentyűzet XT-AT-n

vagy amit akartok . . .

Biztosan mással is előfordult már, hogy amikor egy PC-vel elkezdett dolgozni, „nem találta" a billentyűket. Például ha a Z-t nyomta meg, akkor Y, ha a [-t, akkor ö jelent meg a képernyőn. Ilyenkor egyszerű a megoldás, hiszen a DOS lemezen több KEY???.COM program van. Végig kell próbálni, hogy melyik az, amelyik a mi billentyűzetünkhöz illik. Gyakran azonban egyik sem igazán jó, mert egy-két karakter mindegyikben eltér a billentyűzetre rajzoltaktól. Vagy, ha szerencsénk is volt, akkor eszünkbe jut, hogy milyen jó lenne kihasználni a PC-n levő néhány „magyar" betűt (ÁáÉéíóÖöúÜü), vagy á grafikus karaktereket is.

Természetesen kódjuk ismeretében ez nem okoz nagy gondot, hiszen az ALT gomb lenyomva tartásával a numerikus billentyűzeten beírhatjuk, de . . .

A programunk felhasználóját nem illik arra kényszeríteni, hogy megtanulja az ASCII kódtáblát, ezért jobb lenne, ha a billentyűzet gombjaihoz definiálnánk hozzá egy másik helyen ezeket a plusz betűket, ahol vagy könnyebben megjegyezhetők, vagy fölírhatók. Nagyon sok helyünk van erre, hiszen a programok nem szokták használni az ALT gombbal elérhető lehetőségeket, és a funkciógombok nagy részét sem.

A közölt *program* az eredeti ASCII karakterkészletben megtalálható magyar betűket helyezi el a billentyűzeten a következő rendszer szerint:

á — ALT a
Á— ALT A
é — ALT e
É — ALT E
ú — ALT u
í — ALT i
ó — ALT o
ü — ALT y
Ü — ALT Y (az u melletti betű)
ö — ALT p
Ö — ALT P (az o melletti betű)

Természetesen a program listája és a *táblázat* szerint bárki bővítheti vagy új helyre teheti át ezeket a karaktereket. (Az értékek 16-os számrendszerben értendők a táblázatban.)

	SHIFT	CTRL	ALT
F1 — 3B00	F1 — 5400	F1 — 5E00	F1 — 6800
F2 — 3C00	F2 — 5500	F2 — 5F00	F2 — 6900
. . .	. . .	. . .	. . .
F10 — 4400	F10 — 5D00	F10 — 6700	F10 — 7100

ALT:1-7800	2-7900	3-7A00 . . .	--8200	=-8300
ALT: Q-1000	W-1100	E-1200 . . .	[-1A00	]-1B00
ALT: A-1E00	S-1F00	D-2000 . . .	;-2700	'-2800
ALT: Z-2C00	X-2D00	C-2E00 . . .	/-3500	\-3600
CTRL: A-0001	B-0002	C-0003 . . .	Y-0019	Z-001A
A-0041	B-0042	C-0043	Y-0059	Z-005A

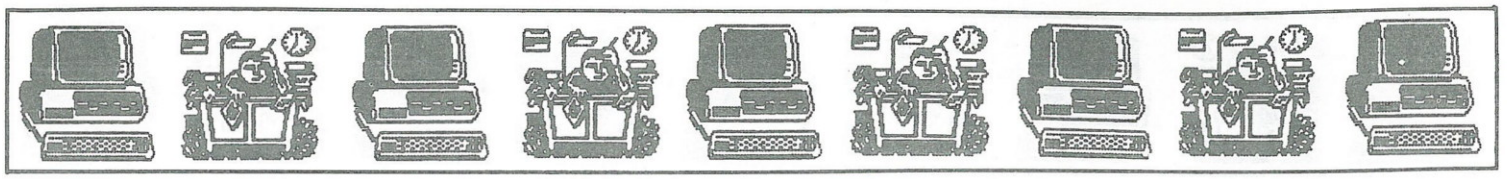
A program bekapcsolására az ALT+, kikapcsolására az ALT- gombokat használtam fel, így ezek nem programozhatók át.

A listán látható programban lévő kódtáblázat első oszlopa a billentyűtől kapott kódot, a második a normál, a harmadik a SHIFT-tel és CAPSLOCK-kal általunk kívánt kódot tartalmazza. Fontos, hogy a ASCII kódtáblázatban lévő kódok hexadecimális alakja elé 00-t írjunk, mint a példában a kis és a nagy Q felcserélésénél. (A Q kódja 51, a táblázatban tehát 0051 szerepel.) Természetesen ez a felcserélés nem hasznos, a programban is csak a bemutatás céljából szerepel.

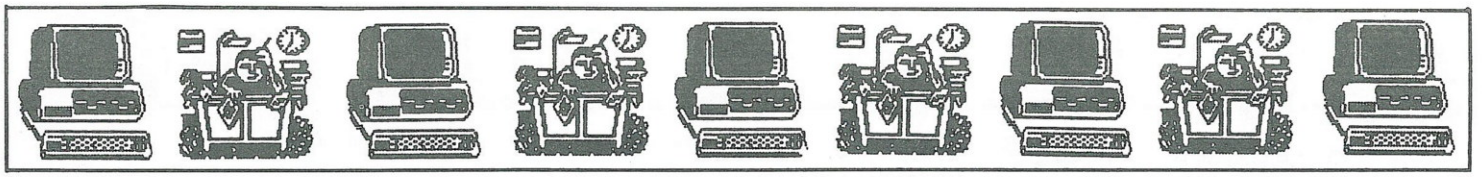
A programot a begépelés után fordítsuk le (MASM KEYMIKRO;), majd hibátlan fordítás esetén indítsuk el a szerkesztést (LINK KEYMIKRO;), és ezután alakítsuk át COM kiterjesztésűre (EXE2BIN KEYMIKRO) (RENAME KEYMIKRO.BIN KEYMIKRO.COM). Az így kapott KEYMIKRO.COM programot az első használat előtt indítsuk el; az a számítógép kikapcsolásáig a memóriában marad.

A program más billentyűzet-átdefiniáló rutinokkal együtt is használható, de a memória jobb kihasználása érdekében célszerűbb a többi változtatást is ebbe a programba beépíteni. A különböző memóriarezidens programok (SIDE KICK, AFD stb.) indítását és futását nem akadályozza, ha a program által felhasznált kódokat nem definiáltuk át. Ha az átdefiniálásukra mégis szükségünk volt, akkor a KEYMIKRO kikapcsolásával (ALT-) ezt semlegesíthetjük.

Krizsák László



```
; Billentyüzet átdefiniáló program
CODE    SEGMENT PARA PUBLIC 'CODE'
  ASSUME CS:CODE,DS:CODE,SS:CODE,ES:NOTHING
        ORG     100H
START:  JMP     START2

IT_CIM  PROC    FAR
        PUSHF
        CMP     AH,2
        JNZ     IT
        POPF
; A régi IT rutin indítása
        JMP     FAR [CS: DWORD PTR VSEG]

IT:     MOV     CS:[PUFF], AH
        POPF
        PUSHF
        CALL    FAR [CS: DWORD PTR VSEG]
        PUSHF
        CLI

; A regiszterek elmentése
        PUSH    BX
        PUSH    CX
        PUSH    DS
        PUSH    SI

        PUSH    CS
        POP     DS

        JNZ     BE
        CMP     [PUFF],0
; Nincs lenyomva semmi
        JNZ     ITRET

; ALT + (bekapcsolás) elvégzése
BE:     CMP     AX,8300H
        JNZ     KI
        MOV     [KI_BE],1
        JMP     SHORT ITRET

; ALT - (kikapcsolás) elvégzése
KI:     CMP     AX,8200H
        JNZ     ITCIM
        MOV     [KI_BE],0

; A KEYMIKRO be van kapcsolva?
ITCIM:  CMP     [KI_BE],0
        JZ      ITRET           ; Nincs

; A Shift gombok leolvasása
        MOV     BX,0
        MOV     DS,BX
        MOV     BH,DS:[0417H]
        PUSH    CS
        POP     DS

; A Shift és CapsLock értelmezése
; SI=2 ha Shift-es feladat
        MOV     SI,2
        TEST    BH,40H
        JZ      H0
H1:     TEST    BH,3
        JZ      BEALL
        JMP     SHORT MIC
H0:     TEST    BH,3
        JNZ     BEALL
MIC:    MOV     SI,0

BEALL:  LEA     BX,TABLA
        MOV     CX,0H
        CMP     AL,32
        JC      KERES
        MOV     AH,0

; Táblázatból keresés
KERES:  CMP     [BX],CX
        JZ      SHORT ITRET
        CMP     AX,[BX]
        JNZ     NOVEL
        MOV     AX,[BX+2+SI]
        JMP     SHORT ITRET
        nop
NOVEL:  ADD     BX,6
        JMP     SHORT KERES

; Regiszterek visszatöltése
ITRET:  POP     SI
        POP     DS
        POP     CX
        POP     BX
        POPF
        STI
        RET     2
IT_CIM  ENDP
VOFF    DW      0
VSEG    DW      0
PUFF    DB      0
KI_BE   DB      1
;               alap kódok,  kód, Shift dekód
tabla   dw      1200H,  0082H,  0090H   ; e - é
        DW      1500H,  0081H,  009AH   ; y - ü
        DW      1600H,  00A3H,  0097H   ; u - ú
        DW      1700H,  00A1H,  0080H   ; i - í
        DW      1800H,  00A2H,  0098H   ; o - ó
        DW      1900H,  0094H,  0099H   ; p - ö
        DW      1E00H,  00A0H,  008FH   ; a - á
;       DW      0051H,  0071H,  0071H   ; Q - q
;       DW      0071H,  0051H,  0051H   ; q - Q
vege:   DW      0000H

; Billentyüzet IT (16) vektor átirása
START2: CLI
        MOV     AX,3516H
        INT     21H
        MOV     AX,[WORD PTR IT_CIM]
        CMP     ES:[BX],AX
        JNZ     INIC
        LEA     DX,VOLT
        MOV     AH,9
        INT     21H
        INT     20H

; új IT vektor elmentése
; új IT vektor beállítása
INIC:   MOV     [VOFF],BX
        MOV     [VSEG],ES
        LEA     DX,IT_CIM
        MOV     AX,2516H
        INT     21H
        STI
        LEA     DX,SZOVEG
        MOV     AH,9
        INT     21h
        LEA     DX,VEGE+2
; Kilépés. A program a memóriában marad
        INT     27H
VOLT    DB      "A program már "
        DB      "installálva volt!"
        DB      0dh,0ah,"$"
SZOVEG  DB      "A KEYMICRO program "
        DB      "installálva.",0dh,0ah,"$"
CODE    ENDS
        END     START
```



Így írtok ti . . . programot

Örvendetesen bővül azoknak a folyóiratoknak a köre, melyek olvasóik által írt programokat tesznek közzé. Ennél is nagyobb szerencse, hogy növekszik a közölhető programokat alkotni tudó amatőrök száma. Ezekben a programokban bőségesen találhatunk sziporkázóan ötletes részleteket, jól szerkesztett, tisztán kidolgozott eljárásokat, amelyeknek leírása, kódolása bárki számára jól érthető. A jó példa segíti az autodidakta fejlődését, újabb ötletetek forrása lehet. Sajnos a rossz példa is ragadós. Az előbbiéhez hasonló, de ellentétes hatást fejt ki a kusza program, a nem kellően tervezett algoritmus.

Sokféleképpen lehet jól, helyesen programozni, de ugyanilyen sok vagy még több rossz szokás olvasható ki az említett programokból. A sokféle ok közül most egyet vegyünk szemügyre. Nagyon sok programíró — türelmetlenségtől hajtva vagy csak lustaságból — nem elemzi eléggé a megoldandó feladatot, nem készít előzetesen kellő részletességgel kidolgozott tervet a programhoz. Ehelyett azonnal leül a gép mellé, és írni kezdi a programsorokat. A rossz sakkozóhoz lehet hasonlítani tevékenységét, azéhoz, aki egy, legfeljebb két-három lépéssel tekint előre játék közben. Volt alkalmam látni ilyen módon „szerkesztett", több száz soros programtorzókat, melyek végül sohasem keltek életre.

Egy egyszerű és nagyon kis feladatot választva mintának, hasonlítsuk össze a helytelen és helyes tervezési gyakorlatot!

A feladat

Írjunk BASIC programot, amely a billentyűzeten beírt x,y koordinátájú pontról eldönti, hogy a koordinátarendszer melyik részén helyezkedik el. A program az eredményt szöveges üzenettel közölje.

Az egyik megoldás

A tiszteletre méltó, ám türelmetlen és főleg gyakorlatlan kezdő, néhány egyszerű program elkészítésével szerzett tapasztalatainak birtokában, azonnal beírja a program első sorát:

```
10 INPUT X,Y
```

Ehhez valóban nem kell sok megfontolás, hiszen a feladat szövegéből egyértelmű: be kell kérni két, tetszőleges numerikus adatot. Emberünk ezután kezd gondolkodni: „Mit tanultam a koordinátasík felosztásáról? Van benne négy negyed. Ezeket a pont koordinátáinak előjele szerint különböztethetjük meg. Egyszerre kell a két koordinátát vizsgálni. A vizsgálat eredménye alapján vagy kiírjuk, vagy nem a megfelelő negyed megnevezését." És nosza, már írja is a program következő sorait:

```
20 IF X>0 AND Y>0 THEN PRINT '1. NEGYED'
30 IF X<0 AND Y>0 THEN PRINT '2. NEGYED'
40 IF X<0 AND Y<0 THEN PRINT '3. NEGYED'
50 IF X>0 AND Y<0 THEN PRINT '4. NEGYED'
```

Készen is lennénk, de a tisztesség kedvéért legyen ott:

```
60 END
```

Ezután néhány próba és SAVE! Az első meglepetés akkor éri, amikor barátjának megmutatva új művét, az a 0,0 számpárt adja meg a programnak. Ekkor kezdődik a toldozás-foldozás. A >0 feltételek helyett mindenhová a >=0 kerül. (Miért is ne, erre való a szerkesztő.) Esetleg még egy finomítás a 10-es sorban: INPUT ' X,Y=';X,Y.

A fejlesztés eredménye nem elhanyagolható, hiszen így a program legalább már minden lehetséges bevitelre reagál azzal, hogy a négy lehetséges válasz egyikét és csak az egyiket kiírja a képernyőre. Mesterünk — okulva az elhamarkodott első változat kudarcán — eltöpreng az aszimmetrikus megoldáson. Miért pont az első negyedhez soroljuk az origót? Az origótól különböző tengelypontok osztályozása sem szimpatikus. Az első negyedhez mindkét tengely, kettőhöz csak egy-egy tartozik, a harmadik negyed mindkét oldalán nyitott.

Mi már tudjuk, hogy a „kétesélyes" döntési sémához való kötődés okozhatta az eltévelyedést. A tervező először úgy okoskodott, hogy egy szám pozitív VAGY negatív lehet. Az első javításkor ezt az alternatívát az ugyancsak kétesélyes séma: negatív VAGY nem negatív váltotta fel.

Az elemzés további részletezésétől eltekintve tegyük fel, hogy az alkotó ezen az úton tovább haladva végül a következő megoldáshoz jut:

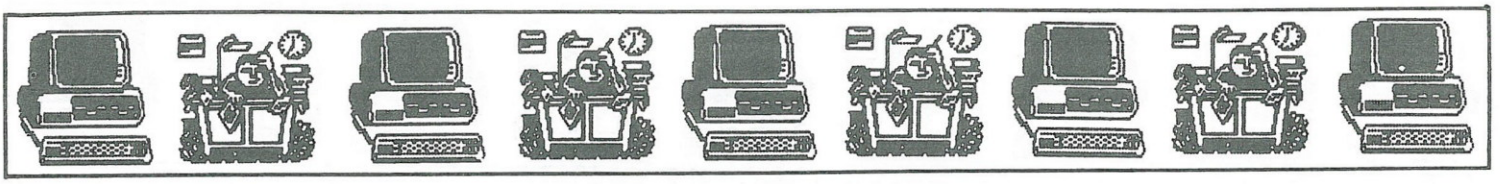
```
 5 PRINT '<clear> PONT HELYE A SÍKBAN'
10 INPUT ' X,Y=';X,Y
15 IF X=0 AND Y=0 THEN PRINT 'ORIGÓ'
20 IF X>0 AND Y>0 THEN PRINT '1. NEGYED'
25 IF X=0 AND Y>0 THEN PRINT '+Y TENGELY'
30 IF X<0 AND Y>0 THEN PRINT '2.NEGYED'
35 IF X<0 AND Y=0 THEN PRINT '-X TENGELY'
```

A sík pontjainak osztályozása — *1. ábra*

	X<0	X=0	X>0
Y>0	(-,+)	(0,+)	(+,+)
Y=0	(-,0)	(0,0)	(+,0)
Y<0	(-,-)	(0,-)	(+,-)

A válaszok táblázata — *2. ábra*

	I=0	I=1	I=2
J=0	2.NEGYED	+Y TENGELY	1.NEGYED
J=1	-X TENGELY	ORIGO	+X TENGELY
J=2	3.NEGYED	-Y TENGELY	4.NEGYED



```
40 IF X<0 AND Y<0 THEN PRINT '3. NEGYED'
45 IF X=0 AND Y<0 THEN PRINT '-Y TENGELY'
50 IF X>0 AND Y<0 THEN PRINT '4. NEGYED'
55 IF X>0 AND Y=0 THEN PRINT '+X TENGELY'
60 END
```

A kész program most már helyesen működik, kifogásunk csak az elkészítés módja ellen lehet. Kezdjük el tehát a megoldást teljesen elölről, de most úgy, ahogyan *mi írunk programot.*

A másik megoldás

Mielőtt a gép közelébe mennénk, elemezzük a feladatot. Első dolgunk tisztázni, hogy milyen adatokat vár a program. A szöveg szerint bemenetként kapja a pont x, y koordinátáit. Ezek tetszőleges valós számok lehetnek, a gépi ábrázolás korlátain belül. Szükségünk van ezenkívül azoknak az üzeneteknek a listájára, melyekből egyet a futás végén a képernyőn látni akarunk. A feladat szövege erre nézve nem ad tájékoztatatást. Ilyen esetben valamilyen általánosan használt megállapodást kell feltételeznie a megoldónak. Ha ilyen sincs, akkor a feladat nem egyértelmű. Megoldani azonban csak egyértelmű feladatot lehet, ezért vagy megkérdezzük a feladat kitűzőjét, hogy mire gondolt, vagy saját hatáskörünkben döntünk egy lehetséges változat mellett.

Válasszuk az osztályozásnak azt a módját, hogy a pont helyzetét két koordinátájának előjele alapján, tehát két szempont szerint döntjük el. Eszerint a pontokat x ÉS y előjele alapján három-három részhalmazba sorolhatjuk. A sík pontjainak ezt az osztályozását az ismert halmazdiagrammal ábrázolhatjuk. (Az *1. ábrán* a szokásos „krumplik" helyett téglalapok szemléltetik a halmazt és részhalmazait.)

Az ábra szerencsés (!) választása azt sugallja, hogy a kettős osztályozással kapott részhalmazok megnevezéseit egy 3 × 3-as karakteres tömbben adjuk meg a programnak, ahogyan azt a *2. ábrán* feltüntettük.

A program adatbevitele tehát két részből tevődik össze: fel kell tölteni a konstans válaszokat tartalmazó tömböt és be kell olvasni a billentyűzetről a koordinátákat. BASIC programban az előbbit DATA-READ, az utóbbi INPUT utasítással oldhatjuk meg.

Az adatbevitel és — ami ennél is fontosabb — az adatszerkezet megtervezése után foglalkozhatunk a feldolgozással. A feladat most már így is megfogalmazható: írassuk ki a válaszok tömbjének azt az elemét, melyet x és y előjelei határoznak meg. Vagyis a feldolgozási lépés feladata a kiírandó elem (i, j) indexeinek kiszámítása. Ha a index lehetséges értékei [0, 1, 2], akkor olyan függvényt kell megadnunk, amely az F: (x,y) → (i,j) hozzárendelést valósítja meg. Mivel az előjelek szerinti osztályozás mellett döntöttünk, kézenfekvő, hogy a keresett leképezést az előjelfüggvény alkalmazásával készítsük el:

```
[i=sgn(x)+1
F:=[
[j=1-sgn(y)
```

Végül a program utolsó lépésében az indexekkel kiválasztott tömbelemet kell egy PRINT utasítással kiírni.

A kész terv birtokában már bekapcsolhatjuk a gépet és megszerkeszthetjük a BASIC programot:

```
 5 PRINT '<clear> PONT HELYE A SIKBAN'
10 DATA '2. NEGYED', '+Y TENGELY','1. NEGYED'
15 DATA '-X TENGELY','ORIGO','+X TENGELY'
20 DATA '3. NEGYED','-Y TENGELY','4. NEGYED'
25 FOR J=0 TO 2 :FOR I=0 TO 2 :READ V$(I,J) :NEXT I,J
30 INPUT ' X,Y=';X,Y
35 PRINT V$(SGN(X)+1,1-SGN(Y))
40 END
```

A tanulság

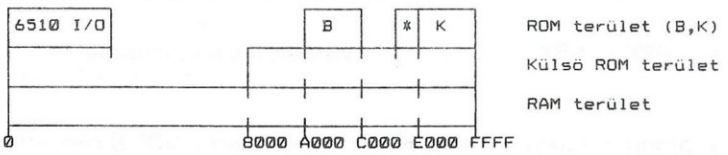
Sapienti sat — a bölcsnek ennyi is elég. Remélem, a példa valóban magáért beszél. Nyomatékul csak ennyit tennék hozzá: képzeljük el a különbséget egy nagy feladat esetén!

Dr. Hack Frigyes

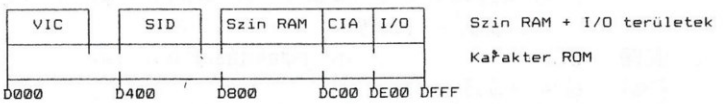
A memóriakezelés csínja-bínja

A C64-ben megtalálható 6510-es processzor csak 64 kbájt memória címzésére képes. Hogyan kezelhető akkor a 65 kbájt RAM, a 20 kbájt ROM és a B/K területek ezzel a processzorral?

Nézzük meg ezeknek a területeknek az elhelyezkedését a 64 kbájton belül.



A $D000—$DFFF-ig terjedő, csillaggal jelölt területet a zsúfoltság miatt külön is bemutatjuk.



Az ábrákból nem pontosan kiolvasható területek kezdő- és végcímei:

6510 I/O	$0000	$0001
VIC	$D000	$D02E
SID	$D400	$D41C
CIA #1	$DC00	$DC0F
CIA #2	$DD00	$DD0F
I/O #1	$DE00	$DEFF
I/O #2	$DF00	$DFFF

Látható, hogy bizonyos területeken átfedés van a memóriák és a B/K területek között. Például a $D000 területen a következő szegmensek közül lehet választani: RAM, külső ROM, karakter ROM és a videovezérlő regiszterei.

Nézzük végig azokat a területeket, amelyek közül mi választjuk ki, hogy melyiket szeretnénk elérni.

— 64 kbájt RAM. A teljes memóriát szabadon használhatjuk.

— Szín-RAM. Karakteres képernyő színinformációit tartalmazó RAM.

— Külső ROM-terület. Ha gépünk bővítő portjába (Expansion port) egy cartridge-et teszünk, akkor a bővítőben levő, programot tartalmazó ROM ezen a területen fog látszani.

— ROM-terület (B, K). A BASIC interpretert és az operációs rendszert tartalmazó ROM-ok.

— Karakter-ROM. A karakterkészletet tartalmazó ROM.

— VIC. A videovezérlő regisztereihez férhetünk hozzá.

— SID. A szintetizátor és az A/D átalakító regisztereihez férhetünk hozzá.

— CIA. A B/K portokat, timereket, órát, botkormányokat, soros buszt vezérlő regiszterekhez férhetünk hozzá.

A külső ROM és a 6510 B/K terület kivételével kiválaszthatjuk, hogy melyik területen melyik egység látszódjon. Ezt a 6510 B/K regisztereinek segítségével tehetjük meg.

$0000 — 6510 adatirány-regiszter
$0001 — 6510 processzor B/K port

A gép bekapcsolása után (ha nincs cartridge), a tárfelosztás a következő:

$0000 — $9FFF — RAM
$A000 — $BFFF — ROM
$C000 — $CFFF — RAM
$D000 — $DFFF — VIC, SIC, Szín-RAM, CIA #1, CIA #2, I/O #1, I/O #2
$E000 — $FFFF — ROM

Az adatirány-regiszter tartalma: 47 = $2F = %00101111
A processzor B/K port tartalma: 55 = $37 = %00110111

A processzor B/K port alsó három bitjének a jelentése a következő:

0 — BASIC ROM be (1 — be)
1 — KERNAL ROM be (1 — be)
2 — karakter ROM ki (1 — ki)

Ha például a processzor B/K tartalmát 54-re ($36 = %00110110) akarjuk változtatni, akkor ezt BASIC-ben a POKE 1,54 paranccsal tehetjük meg. Ne csodálkozzunk, ha a számítógépünkkel a munka nagymértékben megnehezedik, mert kikapcsoltuk a BASIC interpretert tartalmazó ROM-ot. Ha a STOP-RESTORE hatástalan, akkor vagy a RESET-tel vagy a ki- és bekapcsolással járunk sikerrel.

FONTOS! Ha nem akarjuk, hogy esetleg csak a kikapcsolás segítsen, akkor ne hívjunk meg olyan rutint, amely a kikapcsolt ROM-területen helyezkedik el, vagy ne hagyjuk, hogy valamelyik rezidens program vagy megszakítás ezt megtehesse!

A példaprogramokat legegyszerűbben egy monitorprogrammal próbálhatjuk ki. Mi a SMONEX nevű programot használtuk erre a célra.

A BASIC ROM kikapcsolása

Ha szükségünk van a $A000 — $BFFF-ig terjedő területre — például kíváncsiak vagyunk egy programnak a BASIC alatt levő részére, vagy adattárolásra szeretnénk használni —, akkor ezt a következőképpen érhetjük el:

```
                ; A program segítségével a BASIC ROM alatti te-
                ; rületet átmásolhatjuk a $2000 — $4000 címekre
0800    LDA #$36
0802    STA  $01        ; a BASIC ROM kikapcsolása
                ;
0804    LDX #$00
0806    LDY #$20        ; $2000 bájt
0808    LDA  $A000,X    ; forráscím
080B    STA  $2000,X    ; célcím
080E    INX             ;
080F    BNE  $0808      ;                  átmásoló
0811    INC  $080A      ;                  ciklus
0814    INC  $080D      ;
0817    DEY             ;
0818    BNE  $0808      ;
                ;
```

```
081A  LDA #$37
081C  STA  $01        ; a BASIC ROM bekapcsolása
      ;
081E  BRK             ; visszatérés a monitorba
```

Teszteléshez a $0804 címtől futtassuk, ekkor a BASIC ROM-ot másolja át, amit összehasonlíthatunk az eredetivel. Ha többször akarjuk futtatni, akkor ne feledkezzünk meg a forrás- és célcímek beállításáról.

A KERNAL kikapcsolása

A KERNAL kikapcsolásakor a BASIC ROM is kikapcsolódik. A memóriában csak a $D000 — $DFFF-ig terjedő terület lesz lefoglalva. A KERNAL kikapcsolása előtt gondoljuk végig, hogy mi történjen, ha egy megszakítás következik be. Ha egy IRQ vagy NMI hajtódik végre, és a vezérlés a ROM kikapcsolása miatt a RAM-területre kerül, a processzor akkor is megpróbálja végrehajtani az ott megtalálható utasításokat. Ezek a legritkábban felelnek meg a céljainknak.

```
      ; A program a KERNAL alatti területet átmásol-
      ; ja $4000 — $6000 közötti címekre
0800  SEI             ; megszakítások letiltása
0801  LDA #$35
0803  STA #01         ; KERNAL+BASIC kikapcsolva
      ;
0805  LDX #$00
0807  LDY #$20        ; $2000 bájt
0809  LDA  $E000,X    ; forráscím
080C  STA  $4000,X    ; célcím
080F  DEX             ;
0810  BNE  $0809      ;                 átmásoló
0812  INC  $080B      ;                 ciklus
0815  INC  $080E      ;
0818  DEY ;
0819  STY  $D020      ; a keret
                        villogtatása
081C  BNE  $0809      ;
      ;
018E  LDA #$37
0820  STA  $01        ; KERNAL+BASIC
                        bekapcsolása
0822  CLI             ; megszakítások engedélyezése
      ;
0823  BRK             ; visszatérés a monitorba
```

A keretet villogtató sor (STY $D020) nem szükséges a program működéséhez, csak azt mutatja meg, hogy a $D000 — $DFFF területen lévő regiszterek, mint a VIC, a B/K is elérhetők. A tesztelés a $0805 címen futtatva indítható.

A teljes 64 kbájtos RAM-terület kihasználása

A következő példa szerint a teljes RAM-területtel élhetünk. Itt is célszerű figyelembe venni a következőket:
— a nullás lap foglaltságát,
— a processzor veremterületét,
— a képernyő-memóriáig terjedő vektor-, mutató- és táblázatcímeket,
— a képernyőterületet.

```
      ; A program a KERNAL alatti területeket átmá-
      ; solja a $6000 — $8000 közötti címekre
0800  SEI             ; megszakítások letiltása
0801  LDA #$34
0803  STA #01         ; 64 kbájt RAM bekapcsolva
      ;
0805  LDX #$00
0807  LDY #$20        ; $2000 bájt
0809  LDA  $E000,X    ; forráscím
080C  STA  $6000,X    ; célcím
080F  DEX             ;
0810  BNE  $0809      ;                 átmásoló
0812  INC  $080B      ;                 ciklus
0815  INC  $080E      ;
0818  DEY             ;
0819  STY  $D020      ; a keret
                        villogtatása
081C  BNE  $0809      ;
      ;
081E  LDA #$37
0820  STA  $01        ; KERNAL+BASIC
                        bekapcsolása
0822  CLI             ; megszakítások engedélyezése
      ;
0823  BRK             ; visszatérés a monitorba
```

A program futásakor nem villog a keret, mert a $D020 cím nem a VIC egy regisztere, hanem a RAM egy tárcíme

A karakter-ROM olvasása

```
      ; A program a karakter-ROM tartalmát átmásolja
      ; a $8000 — $9000 közötti címekre
0800  SEI             ; megszakítások letiltása
0801  LDA #$33
0803  STA #01         ; a karakter-ROM bekapcsolva
      ;
0805  LDX #$00
0807  LDY #$10        ; $1000 bájt
0809  LDA  $D000,X    ; forráscím
080C  STA  $8000,X    ; célcím
080F  DEX             ;
0810  BNE  $0809      ;                 átmásoló
0812  INC  $080B      ;                 ciklus
0815  INC  $080E      ;
0818  DEY             ;
0819  BNE  $0809      ;
      ;
081C  LDA #$37
081E  STA  $01        ; a karakter-ROM kikapcsolása
0820  CLI             ; megszakítások engedélyezése
      ;
0821  BRK             ; visszatérés a monitorba
```

Bakos Imre — Kelemen Róbert

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 könyvutalvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

12. feladat: Egér a labirintusban 2.

Tegyük az egeret a labirintus bejáratához, a sajtot pedig a kijárathoz. A labirintus térképe ismert. Írjunk programot, amely megmutatja az egérnek a sajthoz vezető legrövidebb utat!

Megoldás

Bár a mostani feladat az előzőtől alig különbözik, megoldása jóval komolyabb erőfeszítést igényel.

A programnak a térkép ismeretében kell eldöntenie bármelyik elágazásnál, hogy az egér melyik utat válassza. Ha a labirintuson csak egy út vezet keresztül, akkor ez valamelyest egyszerűsödik: mindig azt az utat kell választani, amely a labirintusból kivezet, hiszen a másik biztos zsákutca, így annak bejárása csak növelheti a megtett út hosszát.

Ha a feladatot általánosan akarjuk megoldani, akkor először ki kell találni azt a módszert, amellyel a megtett út hossza mérhető, és képzeletben végig kell járatni az egeret az összes lehetséges úton. Így megkaphatjuk valamennyi út hosszát, és ezek közül már könnyen kiválaszthatjuk a legrövidebbet. Az egyszerű és tetszőleges algoritmusnak egyedüli hibája, hogy bonyolultabb labirintusnál sokáig számolna.

Próbáljunk meg ezen segíteni! Egy lehetséges megoldás például a következő.

1. Írjunk be a labirintus első mezőjébe egyet.
2. Írjunk az összes olyan mezőbe kettest, amelyre eljuthatunk az egyes számú mezőből.
3. Írjunk az összes olyan mezőbe hármast, amelyre eljuthatunk a kettes számú mező(k)ről, és még nincs bennük szám. *(1. ábra)*

k. Írjunk az összes olyan mezőbe k-t, amelyre eljuthatunk a k—1-es számú mező(k)ről, és még nincs bennük szám.

Folytassuk ezt mindaddig, amíg a kijárat előtti mezőbe is szám kerül. *(2. ábra)*

A labirintus minden mezőjébe beírt szám megmutatja, hogy az adott mezőre a bejárattól a legrövidebb úton hány lépésben lehet eljutni. Ha a mezőtől a bejáratig visszafelé, a csökkenő számok irányába haladunk, ezt a legrövidebb utat be is jelölhetjük. *(3. ábra)*

Bár a most ismertetett algoritmus alapján működő program a feladat teljes megoldását jelenti, a közölt program mégsem ezt, hanem az elsőként leírt speciális algoritmust használja.

Az egér mozgását bemutató — Turbo Pascal 4.0 nyelvű, IBM PC-n tetszőleges grafikus adapterrel működtethető — program megírásához először a labirintust kellett megrajzolni. Ez a Labirintus szubrutin feladata. A szubrutin az előző számban közölt rutinnal megegyezik. Gondolatmenete és működése a következő.

Induljunk ki a még felosztatlan táblából! Véletlenszerűen választott helyen egy olyan fallal osszuk ketté a téglalapot, amelyen pontosan egy átjáró van!

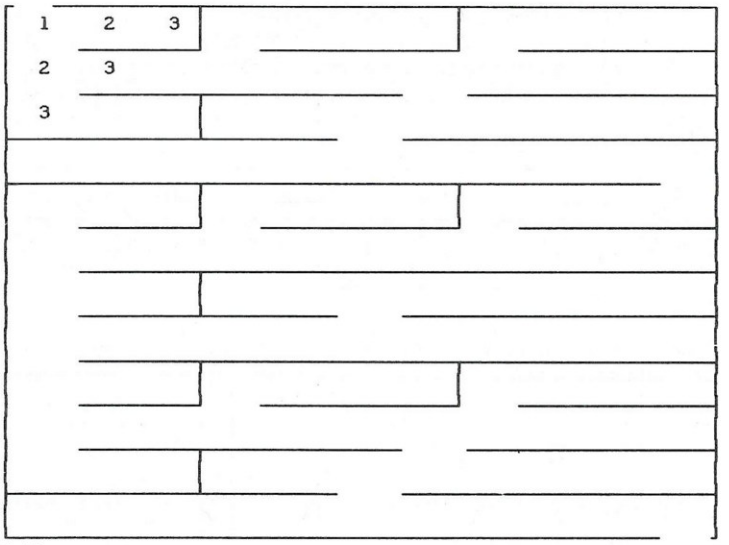
Ismételjük meg ezt az eljárást a jobb és bal oldali téglalapra!

Most végezzük el ugyanezt a keletkezett négy téglalapra!

Folytassuk az eljárást mindaddig, amíg a téglalapok értelmesen tovább már nem oszthatók.

A Labirintus szubrutin fő részét Xfelez és Yfelez eljárások alkotják. Ezek osztják ketté a téglalapot vízszintes, illetve függőleges irányban. A két szubrutin olyannyira hasonló, hogy működésük egységesen tárgyalható.

Az eljárás paraméterként megkapja a kettéosztandó téglalap adatait: a bal felső sarok koordinátáit, a téglalap szélességét és hosszúságát. Ha a téglalap szélesebb vagy magasabb, mint az SZ konstansban tárolt folyosószélesség, az osztófal (oszt) és a falon átvezető átjáró (lyuk) helyének kiszámítása következik. A következő két sor a falat rajzolja meg.



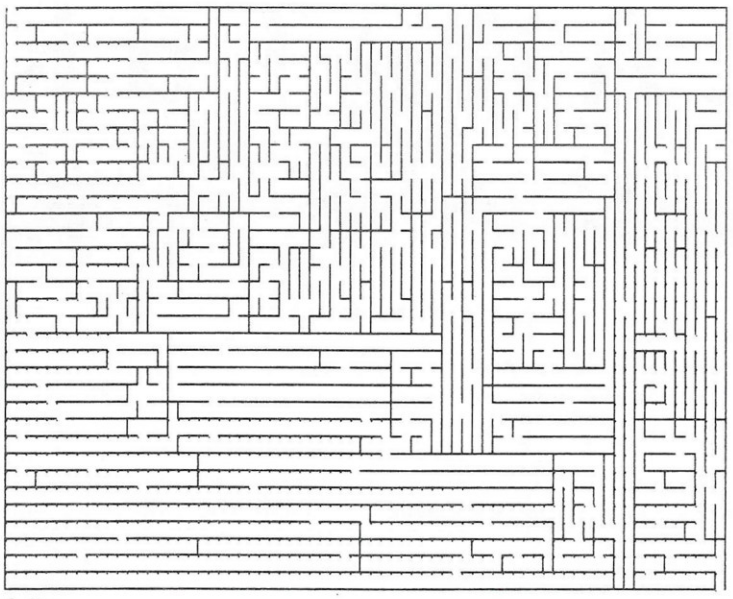
1. ábra

1	2	3	6	7	8	9	10	11	12	13
2	3	4	5	6	7	8	9	10	11	12
3	4	5	12	11	10	9	10	11	12	13
16	15	14	13	12	11	12	13	14	15	16
29	30	31	26	27	28	29	20	19	18	17
28	27	26	25	24	23	22	21	22	23	24
29	30	31	38	37	36	37	38	39	40	41
30	31	32	33	34	35	36	37	38	39	40
31	32	33	36	37	38	39	40	41	42	43
32	33	34	35	36	37	38	39	40	41	42
33	34	35	42	41	40	39	40	41	42	43
46	45	44	43	42	41	42	43	44	45	46

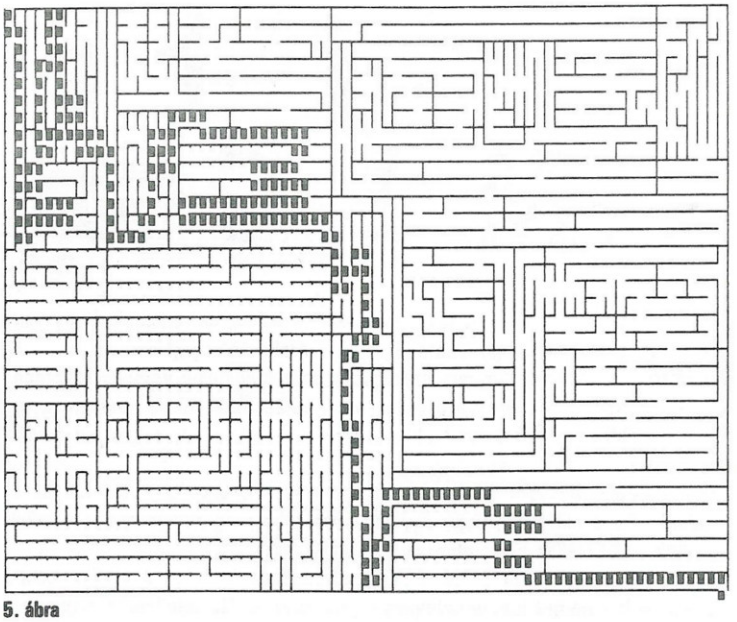
2. ábra

3. ábra

1	2	3	6	7	8	9	10	11	12	13
2	3	4	5	6	7	8	9	10	11	12
3	4	5	12	11	10	9	10	11	12	13
16	15	14	13	12	11	12	13	14	15	16
29	30	31	26	27	28	29	20	19	18	17
28	27	26	25	24	23	22	21	22	23	24
29	30	31	38	37	36	37	38	39	40	41
30	31	32	33	34	35	36	37	38	39	40
31	32	33	36	37	38	39	40	41	42	43
32	33	34	35	36	37	38	39	40	41	42
33	34	35	42	41	40	39	40	41	42	43
46	45	44	43	42	41	42	43	44	45	46



4. ábra



5. ábra

```
{-------------------------------------}
{                                     }
{        Eger a labirintusban         }
{              terkeppel              }
{                                     }
{             Pinter  Gabor           }
{                                     }
{              1988.10.25.            }
{                                     }
{-------------------------------------}

program EgerT;


uses
  Graph;

const
  SZ=4; { Folyoso szelesseg }
  KOMPL=8; { Labirintus bonyolultsag }

  Fal=7; { A fal szine. }
  Jel=14; { Ezzel a szinnel jelol meg
        minden mezot amelyet erintett.}
  Ut=9; { Ez a szin jelzi a
        kivalasztott utvonalat. }

var
  EgerIrany     :
            (LE, BALRA, FEL, JOBBRA);
  EgerX, EgerY : integer;
  MaxX, Maxy    : integer;
  Ch            : char;
  Gd, Gm        : integer;


procedure Labirintus;


procedure Yfelez(
  x,y,dx,dy : integer); forward;


{ Fuggolegesen ketteosztja a tablat }
procedure Xfelez(
  x,y,  { Bal felso sarok koordinatai }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztofal helyzete: }
  oszt : integer;
  { Az osztofalon a lyuk helye: }
  lyuk : integer;
begin {Xfelez}
{ Kell e tovabb osztani? }
if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
  { Fal rajzolasa: }
  oszt:=x+SZ*Random(dx div SZ -1 )+SZ;

  lyuk:=y+SZ*Random(dy div SZ);
  Line(oszt,y,oszt,lyuk);
  Line(oszt,lyuk+SZ,oszt,y+dy);
  { A fal ket oldalanak levo
    tablak tovabbosztasa: }
  if Random(10)>KOMPL
    then
    Xfelez(x,y,oszt-x,dy)
    else
    Yfelez(x,y,oszt-x,dy);
  if Random(10)>KOMPL
    then
    Xfelez(oszt,y,x+dx-oszt,dy)
    else
    Yfelez(oszt,y,x+dx-oszt,dy)
  end {if}
end {Xfelez};


{ Vizszintesen ketteosztja a tablat }
procedure Yfelez(
  x,y,  { Bal felso sarok koordinatai }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztofal helyzete: }
  oszt : integer;
  { Az osztofalon a lyuk helye: }
  lyuk : integer;

begin {Yfelez}
{ Kell e tovabb osztani? }
if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
  { Fal rajzolasa: }
  oszt:=y+SZ*Random(dy div SZ - 1)+SZ;
  lyuk:=x+SZ*Random(dx div SZ);
  Line(x,oszt,lyuk,oszt);
  Line(lyuk+SZ,oszt,x+dx,oszt);
  { A fal ket oldalanak levo
    tablak tovabbosztasa: }
  if Random(10)<=KOMPL
    then
    Xfelez(x,y,dx,oszt-y)
    else
    Yfelez(x,y,dx,oszt-y);
  if Random(10)<=KOMPL
    then
    Xfelez(x,oszt,dx,y+dy-oszt)
    else
    Yfelez(x,oszt,dx,y+dy-oszt)
  end {if}
end {Yfelez};


begin {Labirintus}

  { Keret a be- es kijarattal }
  MaxX:=SZ*(GetMaxX div SZ);

  MaxY:=SZ*(GetMaxY div SZ);
  Line(0,0,0,MaxY);
  Line(0,MaxY,MaxX-SZ,MaxY);
  Line(MaxX,MaxY,MaxX,0);
  Line(MaxX,0,SZ,0);

  { Labirintus rajzolas }
  Xfelez(0,0,MaxX,MaxY);

end {Labirintus};



{ Ha az eger (EgerX,EgerY) pontrol
  ki tud jutni, erteke true.
  A megtalalt kivezeto utet bejeloli. }
function Mozog : boolean;

begin {Mozog}

{ Jart mar itt? }
if GetPixel(EgerX,EgerY)=Jel
then begin
  Mozog:=false;
  exit;
end;

{ Megjeloli a mezot. }
PutPixel(EgerX,EgerY,Jel);

{ Kiert? }
if EgerY>=MaxY
then begin
  Mozog:=true;
  exit;
end;

{ Balra van-e kiut a labirintusbol? }
if GetPixel(EgerX-1,EgerY)<>Fal
then begin
{ Balra lep }
  EgerX:=EgerX-SZ;
{ Errol a mezorol ki lehet-e jutni? }
  if Mozog
  then begin
{ Berajzolja a mezot, mert resze
  a kivezeto utnak. }
    Bar(EgerX,EgerY,
      EgerX+SZ div 2,EgerY+SZ div 2);
{ Visszalep }
    EgerX:=EgerX+SZ;
    Mozog:=true;
    exit;
  end
  else begin
{ Visszalep }
    EgerX:=EgerX+SZ;
  end;
end;

{ Lefele van-e kiut a labirintusbol? }
if GetPixel(EgerX,EgerY+SZ-1)<>Fal
then begin
{ Lefele lep }
  EgerY:=EgerY+SZ;
{ Errol a mezorol ki lehet-e jutni? }
  if Mozog
  then begin
{ Berajzolja a mezot, mert resze
  a kivezeto utnak. }
    Bar(EgerX,EgerY,
      EgerX+SZ div 2,EgerY+SZ div 2);
{ Visszalep }
    EgerY:=EgerY-SZ;
    Mozog:=true;
    exit;
  end
  else begin
{ Visszalep }
    EgerY:=EgerY-SZ;
  end;
end;

{ Jobbra van-e kiut a labirintusbol? }
if GetPixel(EgerX+SZ-1,EgerY)<>Fal
then begin
{ Jobbra lep }
  EgerX:=EgerX+SZ;
```

A fal két oldalán keletkezett téglalapok további felosztását Xfelez vagy Yfelez végzi KOMPL konstans értékétől függően eltérő valószínűséggel. Ezzel lehet a labirintus bonyolultságát szabályozni.

A szubrutin fennmaradó feladatai már csak a külső keret, a be- és a kijárat megrajzolása. A labirintus rajzolását az innen meghívott Xfelez készíti el.

Az egér útkeresését és mozgását a Mozog eljárás mutatja be. Ez TRUE értéket ad vissza, ha az egér a megadott (EgérX, EgérY) pontról kijut.

Az egér útkeresését jól követhetjük a képernyőn is, mivel minden megvizsgált mezőt megjelöl Jel színnel. *(4. ábra)* Az eljárás a legelején ennek segítségével állapítja meg, hogy mely mezőkön járt már. Ha az egér még nem járt ezen a mezőn, akkor a mezőt megjelöli.

A mező csak akkor hagyható el, ha az közvetlenül a kijárat mellett van, vagy ha valamelyik szomszédos mezőről ki lehet jutni.

Ha a mező a kijárat előtt van, akkor innen egy lépésben kijuthat, és így Mozog=TRUE értékkel tér vissza az eljárás.

Az eljárás önmagát hívja meg, hogy megállapítsa: a szomszédos mezők elhagyhatók-e. Ha igen, akkor ezt a mezőt mint a kivezető út részét berajzolja. *(5. ábra)* Ha nem, akkor visszalép és a további szomszédos mezőkkel próbálkozik.

Minden mezőnek négy szomszédja lehet, ezért az eljárás mind a néggyel megpróbálkozik, és ha mindegyik sikertelen, csak akkor tér vissza mozog=FALSE értékkel a hívóhoz, jelezve, hogy az adott mezőről nincs kivezető út.

A főprogram feladata csak a grafikus mód beállítása, a Labirintus, majd pedig a Mozog szubrutinok meghívása.

```
{ Errol a mezorol ki lehet-e jutni? }
  if Mozog
  then begin
{ Berajzolja a mezot, mert resze
  a kivezeto utnak. }
    Bar(EgerX,EgerY,
      EgerX+SZ div 2,EgerY+SZ div 2);
{ Visszalep }
    EgerX:=EgerX-SZ;
    Mozog:=true;
    exit;
  end
  else begin
{ Visszalep }
    EgerX:=EgerX-SZ;
  end;
end;

{ Felfele van-e kiut a labirintusbol? }
if GetPixel(EgerX,EgerY-1)<>Fal
then begin
{ Felfele lep }
  EgerY:=EgerY-SZ;
{ Errol a mezorol ki lehet-e jutni? }
  if Mozog
  then begin
{ Berajzolja a mezot, mert resze
  a kivezeto utnak. }
    Bar(EgerX,EgerY,
      EgerX+SZ div 2,EgerY+SZ div 2);
{ Visszalep }
    EgerY:=EgerY+SZ;
    Mozog:=true;
    exit;
  end
  else begin
{ Visszalep }
    EgerY:=EgerY+SZ;
  end;
end;

{ Nincs kivezeto ut }
Mozog:=false;

end {Mozog};


begin {EgerT}
Randomize;
  { Grafikus mod }
  Gd := Detect;
  InitGraph(Gd, Gm, '');
  if GraphResult <> grOk then
    Halt(1);

  { Labirintus rajzolas. }
  SetColor(Fal);
  Labirintus;

  SetColor(Ut);
  SetFillStyle(SolidFill,Ut);
  { Eger kiindulasi helyzet. }
  EgerX:=1; EgerY:=1;
  { Az ut keresese es megjelolese. }
  if Mozog then
  Bar(EgerX,EgerY,
      EgerY+SZ div 2,EgerY+SZ div 2);

  readln;
end.
```

13. feladat: Számológép

Írjon programot, amely ellátja egy egyszerű zsebszámológép funkcióit (alapműveletek és négyzetgyökvonás). Feleltessen meg a számológép gombjainak megfelelő billentyűket, és ezek lenyomására a program viselkedjen a zsebszámológéphez hasonló módon!

Pintér Gábor

A μ'89 diákszemmel

Görbe tükör?

Az idén is — az előző évek hagyományai nyomán — megrendezték a μ'89 találkozót. A rendezvény plakátján az állt, hogy ez az egyik legjelentősebb hazai számítástechnikai rendezvény. Lehet, hogy csak a látszat csal, de én bizony nem láttam itt képviselve az összes hazai számítástechnikai céget, vagy az is előfordulhat, hogy ők nem voltak érdekelve a részvételben.

Sok látogató megjegyezte, hogy bizony a tavalyi jobb volt. Ennek az okát egyrészt a területcsökkenésben látják, amely bár nem áll közvetlen összefüggésben a minőséggel, de azért a választékot, a találkozó tarkaságát jelentősen szűkítette. A tavalyi másik számítástechnikai pavilont most borászok foglalták el, finom borokkal lepve meg a számítógépeket kereső látogatókat. A másik ok pedig az általános gazdasági helyzet szorítójában rejlik. Ha nincs pénz, akkor egy cég — logikusan — a kevésbé jövedelmező kiadásokat „függeszti" fel vagy nyirbálja meg. Szemmel láthatóan így volt ez a Mikrofesztiválon is. Sok standnál az érdeklődő a tavalyi, tavalyelőtti reklámanyaggal találta szembe magát, s nosztalgiázott, hogy milyen szépek is lehettek új korukban. (Az aktualitásról ne is beszéljünk.)

A kiállító cégeknél az idén már sablonná vált a PC. Általában az első két-három napon még ügyeltek arra, hogy futkosson bemutatóprogram a gépükön, de később felhagytak ezzel, s az érdeklődőre már csak az üres monitorok bámultak vissza. Szerencsére azért kezdték az idén már feladni a standfelügyelők azt a tavalyi gyakorlatukat, hogy a gépük szent és sérthetetlen, így néha legalább a gépeken prüntyöghettünk egy keveset.

Sajnos, olyan igazán nagy újdonságot nem sikerült felfedeznem a programok kavalkádjában. Ami új, hogy az adatbázis-kezelő programok végre kezdenek új, felhasználóközelibb jelleget ölteni. Színesek, az üzeneteket ablakrendszerben írják ki, s hasonló módon várják a válaszokat is. A helpmenük bárhonnan elérhetőek, s a programok minden lépést magyaráznak. Remélhetően most már eltűnnek azok a programok, melyek használatához külön lexikonokat mellékeltek.

Bár az oktatóprogramok versenyén nem voltam ott, de a bírálók szerint nem túl sok — sőt! kevés — jó program mutatkozott be az idén.

S ha oktatóprogramot említek, meg kell emlékeznem a Linguasoftról is, mely egy nyelvoktató program, s az idén is, immár hagyományosan, megjelent a találkozón.

Ha a látogató szemszögéből akarom vizsgálni a rendezvényt, akkor mindenképpen az itt megtalálható szolgáltatásokra kell helyeznem a hangsúlyt. A NOVOTRADE hagyományaihoz hűen megjelent könyveivel, melyek csakúgy, mint bárhol, meglehetően borsos árukkal alaposan gondolkodóba ejtették a betérő vásárlót. A pavilon emeletén, a galériában a képdigitalizálás, melyet a Hámán SZKI asztalánál csináltak, nagy sikert aratott. S talán szerénytelenség vádja nélkül leírhatom, hogy nekünk is sikerünk volt, úgy mint MMD Stúdió (Mikroszámítógép Magazin Diákrovat) a számítógépes levelezőtárs-keresésünkkel.

A kiállítás teljes képéhez hozzátartozik az is, hogy a fő hangsúly az Utazás '89 kiállításon volt, s a Mikro ennek csak az árnyékában jelent meg. Talán egy kicsit nagyobb propagandával és nagyobb kiállítói táborral a rendezvény kiemelkedhetne ebből az árnyékból, s igazán tükröt állíthatna a hazai számítástechnikának, mert érzésem szerint az idén még egy kicsit görbe volt ez a tükör.

VÁMOS SÁNDOR

Gunship

A Micro Prose újabb, kitűnő harci helikopter szimulációja méltán aratott nagy sikert. A sokszínű feladatok, akciók változatos lehetőségeket nyújtanak a kezdőknek és profiknak egyaránt. A kiváló grafikai megoldások és az áttekinthető menük tovább fokozzák a játékkedvet.

KIVEL, MIVEL, MERRE, MILYEN IDŐBEN?

Az első menüben lehetőségünk van a már esetleges előző játék során sikeresen visszatért pilótánk újbóli bevetésére, valamint új pilóta létrehozására. A folytatás <Continue> során csak egyszer tüzeljünk, és húzzuk valamilyen irányba a botkormányt (Port 2). Másodszori tüzeléssel ugyanis kihagynánk néhány fontos menütételt. A következőkben — portrénk és jelvényünk megcsodálása után — válasszunk ízlésünk szerinti feladatot <Duty>, az azt nehezítő stílust <Style> és a végrehajtást befolyásoló tényezőket <Reality>.

A feladat <Duty> menüben ötféle különböző berepülési, felderítési és egyéb cselekvés közül választhatunk. Ezek a listán, fentről lefelé haladva, egyre nehezebbek. A fenti <Overall Risk> felirat melletti becsmérlő szavak az esetleges kockázatot jelzik <Very Low, Medium> stb. Innen visszaléphetünk az alapmenübe.

A helyzetüket nehezíteni kívánó pilótajelöltek kiélhetik magukat a <Style> menüben, ahol három opció közül választhatnak:

— gyakorlórepülés kis piff-puff kíséretében, biztos túlélés,

— sikeres akcióban nemzeti hőssé válhatunk vagy esetleg elhalálozhatunk,

— visszatérésünkkor garantált a tévészereplés, de erre azért kevés az esélyünk!

Végül a repülést befolyásoló tényezőket — négy ilyen van — állíthatjuk a <Reality> menüben:

1. Akciónkat valós, háborús helyzetben vagy könnyített békeidőben hajtjuk végre.

2. A leszállások lehetnek kiválóan kivitelezettek, vagy esetenként számítani kell rázósabb landolásra is.

3. Az időjárás lehet szép tavaszias, gyenge szellővel, 76 °F, vagy igen változékony, gyakran erős széllel.

4. Az ellenség taktikája, felszereltsége első-, másod- és harmadosztályú lehet.

És itt is kijelződik az általános kockázat. Vigyázat! Ezt a háromféle menü együttesen befolyásolja!

Továbblépés után tömör feladatleírást kapunk. Első helyen az akció minősítése szerepel, majd a feladat jellege olvasható el. Ez fontos, mivel két-három szóban megkapjuk az utasítást: például Search and Destroy — keresd és semmisítsd meg!

Az alattuk levő hosszabb szöveg a feladat részletes leírását tartalmazza. Ebből a lényeget általában más színnel jelölik.

Nagyon fontosak a zárójelben közölt koordináták, mivel ezeknél találjuk az ellenséges célpontokat; legjobb, ha feljegyezzük azokat. A megsemmisítendő célpontok nevei kiemelten szerepelnek mint bunkers, depots stb.

Itt kapunk még az esetleges közelben lévő helikopterekről is fontos információt.

Rövid tájékoztatást ad az időjárásról és a napszakot is kijelöli, hogy nappali (day) vagy éjjeli (night) akciót kell-e végrehajtanunk.

Ha folytatjuk a lekérdezést (Continue), akkor megkapjuk kémeink jelentését az ellenség fegyverzetéről, a géppuskafészkektől a megfigyelő radarokig.

Szívélyes figyelmeztetésben részesülünk a jelszó megjegyzésére. Itt sajnos van egy kis probléma: az akcióból való visszatéréskor ugyanis meg kell adni a közölt jelszó párját, melyet nem tudok ismertetni a kedves olvasóval, mert hiába boncoltam fel a programot, nem találtam meg. Szívesen leközölnénk, ha valaki, aki tudja, elküldené a szerkesztőségbe. De térjünk vissza a játékhoz!

A jelszóra vonatkozó fenyegetés után emlékeztet a koordináták megjegyzésére és részletesen leírja az időjárást.

Ha mindezeken végigrágtuk magunkat, végre kezdődhet a játék!

FELKÉSZÜLÉS

(Arming Your AH—64A). Itt felfegyverezhetjük helikopterünket. A választék igen nagy, de a mértéktelen pakolásnak határt szab a gép teherbíró képessége.

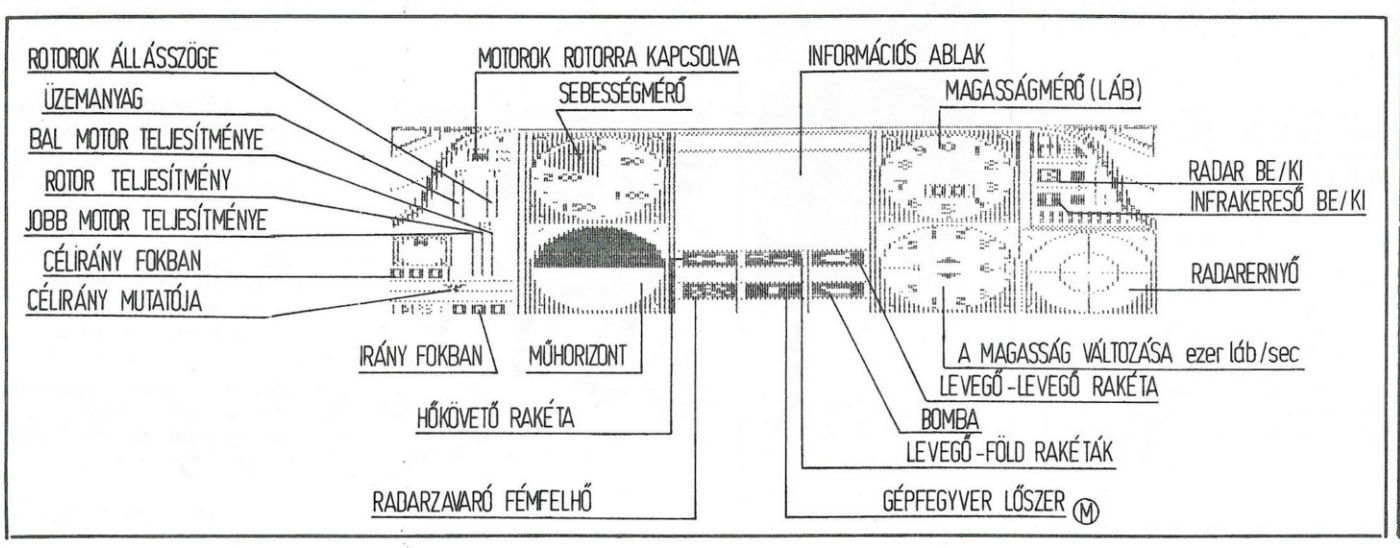
Izgalmas dolog rájönni az egyes fegyverek használatára, ezért nem is írom le őket. Nagyon hasznos a 2,75 FFAR*19 jelű géppuskalőszer, szinte soha sincs elég belőle. Gépünk eleve a maximális üzemanyag-mennyiséggel van feltöltve, így azzal külön nem kell törődnünk. Lehetőség van viszont a gép újrafegyverzésére a (Clear) utasítással.

ÉS AKKOR INDULÁS!

Kapcsoljuk be mindkét motort — a műveletet a motorok felpörgő hangja is jelzi —, majd kapcsoljuk azokat a rotorokra.

Emelkedjünk fel! Ezt a rotorok állásszögének növelésével érhetjük el. Az emelkedés csak maximális hajlásszögnél kezdődik meg. Forduljunk az első célpont irányába. Ezt kétféle módon is megtehetjük: vagy szinkronba hozzuk a célmutatót a saját iránymutatónkkal, vagy ugyanezt a fokban megadott kijelzőn állítjuk be. Az első módszer könnyebb, a második pontosabb. Mozgásnál a botkormány döntésével állítunk.

Induljunk el a sebesség növelésével, a botkormány előredöntésével. Ekkor a sebesség nő, a magasság viszont csökken, amint az a jobb szélső magasságmérő (Alt) műszerről is leolvasható. Ezért ezer láb alatt túlzottan ne növel-



jük a sebességet, mert földhöz csapódhatunk. Igyekezzünk a magasságot a rotorok állásszögével szabályozni!

A CÉLNÁL

Repülésünk közben többször megjelenik a (Target) célpont felirat. Az ehhez tartozó objektum távolságát és jellegét a (Space) megnyomásával olvashatjuk le az információs képernyőről. A kékkel írt, illetve U. S.-szel jelölt célokra ne lőjünk, hiszen azok saját állásaink. Az átkapcsolás után megjelenő célkereszt színe mutatja a találat valószínűségét, úgy, hogy minél világosabb, annál közelebb vagyunk a célhoz, tehát biztosabb a találat. A géppuska használatánál nem kell a célt a kereszt közepébe igazítani, mert a lövés automatikusan oda irányul. Rakéta használatánál azonban ez a mi dolgunk.

Repülésünket a térképen is nyomon követhetjük, ahol alapállapotban az első célponton helyezkedik el a célkereszt, amit azután a célra kell igazítanunk.

A célhoz közeledve elsősorban a megjelölt célpontra lőjünk. Sikeres találatnál rádióüzenetet kapunk. Addig keressük ezt a célt, amíg az üzenet meg nem jelenik, ugyanis csak akkor teljesítettük a feladatot!

A FEGYVERZET

A műszerfalon kijelezve látjuk fegyvereink jellegét és számát. Ezek balról jobbra haladva a következők: hőkövető rakéta, levegő-föld rakéta, levegő-levegő rakéta, radarzavaró fémcsíkok, gépfegyverlőszer, a kiválasztott bomba.

A VISSZATÉRÉS

A feladatok végrehajtása után vissza kell térnünk egy bázisra (U. S. Base). Ezekből három van. Repüljünk a leszállópálya fölé. Ereszkedjünk lassan, nulla sebességgel le, amíg rádióüzenetet nem kapunk. Ebben kérdezik azt a bizonyos jelszót.

Igyekezzünk nagyon alacsonyan repülni, mert a jelszó nélkül biztosan kilőnek, és lezuhanunk. A néhány méter magasból való zuhanásnál nem történik nagyobb baj.

Sikeres leszállás után tájékoztatást kapunk a sikeres feladatok számáról, magunk és gépünk állapotáról.

A BILLENTYŰZET ÉS A MŰSZERFAL

1,2	**motorok ki/be kapcsolása**
3	**motorok rotorra kapcsolása**
4,7	**aktuális fegyver kiválasztása**
\,CLR,DEL	**jobbra, előre, balra tekintés**
O	**infrakereső ki/be**
+	**radar ki/be**
F1, F3	**a két rotor állásszögének növelése**
F5,F7	**a két rotor állásszögének csökkentése**
SPACE	**információs képernyő hívása**
Z	**térkép hívása**
CRSR	**forgás nulla sebesség esetén**

A pilótafülke tetején az ablak fölött megfigyelhetünk 14 betűt. Ezek a gép bizonyos elemeinek állapotát jelzik. Meghibásodáskor sárgán, működésképtelen állapotban pedig pirosan világítanak. A jelzők sorrendben — a műszerfalon balról jobbra — a következők:

R fő rotor
O orroptikák
A orr, repüléselektronikai berendezések
G 30 mm-es gépágyú
F orr-üzemanyagtartály
W jobb oldali szárny
W bal oldali szárny
E jobb oldali motor
E bal oldali motor
F hátsó üzemanyagtartály
A hátsó repüléselektronikai berendezések
L radar álcélpontkilövő
L infra álcélpontkilövő
R forgatónyomaték-kiegyenlítő rotor

Tass Csaba

Mini mutatványok

Harminc sor, három program. Három apró grafikai mutatvány, amelyekhez hasonlókat a Szinte egy lélegzetre... című cikkben a tavaly novemberi számban már bemutattam. Tehát a mostaniak sem virtuóz hosszúak, inkább szórakoztató minik. Persze a mini is lehet tanulságos.

Az *1. listán* egy ötsoros BASIC programocska látható. Aki még nem látott TVC-t garázdálkodni, az el fog csodálkozni, hogy milyen gyors grafikát tittyent az öreg BASIC haló poraiban.

A SET MODE 3 hatását használja fel a program. Ha két vonalat egymásra írunk 3-as MODE-ban, akkor a felső letörli az alsót, és a korábbi hátteret látjuk alatta. Így lehet például célkeresztet mozgatni a táj fölött.

A következő program *(2. lista)* szerencsére nem jó semmire. Futtassuk le: egy mozgalmas grafikát láthatunk. Sokáig el lehet nézegetni. Kipróbálható különböző módokban (Graphics 2, 4, 16), más képernyőképekből indulva. A XOR utasítás okozza ennél is a kalamajkát, azaz gyakorlásra mégiscsak jó. A harmadik szintén gépi kódú program *(3. lista).*A képernyő minden bájtját eltolja biteltoló — SLA — utasítással. Ezt nyolcszor végrehajtva „elporlasztja" a képernyőt, azaz látványosabb CLS-t hajt végre. Más biteltoló utasítással élve hasonlóan érdekes hatást érhetünk el.

Borbás Bence

```
10 **** 1. ****
20 GRAPHICS 4:SET MODE 3
30 X=RND(4)*200:Y=RND(4)*200
40 PLOT X,Y:X+200,Y:X+200,Y+200:X,Y+200:X,Y
50 GOTO 30
```

1. lista

```
10 REM **** 2. ****
20 LOMEM 7000
30 POKE 33,239:POKE 34,25
40 READ A:IF A=555 THEN 100
50 POKE 6639+B,A:B=B+1:GOTO 40
60 DATA 243,62,80,211,2,33,255,127,17
70 DATA 255,127,1,72,63,237,160,226
80 DATA 7,26,174,119,195,253,25,251
90 DATA 201,555
100 REM DEMO
110 GRAPHICS 4
120 LIST
130 EXT0:GOTO 130
```

2. lista

3. lista

```
10 REM **** 3. ****
20 LOMEM 6800
30 READ A:IF A=555 THEN 100
40 POKE 6639+B,A:B=B+1:GOTO 30
50 DATA 243,62,80,211,2,203,38,237,160
55 DATA 234,244,25,251,201,555
100 REM DEMO
110 GRAPHICS 2:LIST
120 FOR I=0 TO 7
130 EXT 0,32767,32767,16200
140 NEXT I
150 END
```

TAPE nyitás és négyesfogat

Akinek kétnyelvű Enterprise-a van, beépítve használhatja a VSAVE és a VLOAD programokat. Az angol és a német cartridge-ban viszont ezek nincsenek benne, ezért külön kell megvásárolni, ami persze költséges móka. Egy trükkel azonban nincs szükségünk ezekre a programokra, ha a saját grafikánkat akarjuk BASIC programból szalagra menteni.

Ez a trükk azon alapul, hogy az Enterprise videokezelője is -VIDEO- escape szekvenciákkal vezérelhető. Ha tehát kazettára akarjuk vinni grafikánkat, nem kell mást tennünk, csak azt a csatornát, amit grafikára nyitnánk meg, TAPE egységként nyitjuk meg írásra.

Valahogy így, a GRAPHICS utasítás helyére:

10 OPEN £101:„TAPE:név"ACCES OUTPUT

Ezután jöhetnek a rajzoló utasítások, a PLOT, PLOT ELLIPSE, PLOT PAINT stb. Végül pedig lezárjuk a magnó csatornáját (xx CLOSE 101).

A grafika visszatöltése:

10 GRAPHICS
20 OPEN £106:„TAPE:név"
30 COPY FROM £106 TO £101
40 CLOSE 106

```
100 SET STATUS OFF
110 SET BORDER 0
120 ALLOCATE 100
130 CODE BUF=HEX$("54,5d,06,50,1b,10,fd,01,a0,05,ed,b0,c9")
140 OUT 177,254
150 GRAPHICS HIRES 16
160 SET PALETTE 0,0,255
170 SET INK 1
180 FOR K=0 TO 8
190   READ A$
200   PRINT #101,AT 10,(20-(LEN(A$)))/2+1;A$
210   CALL LIGHTER
220   FOR I=40652 TO 27693+K*1440 STEP-80
230     CALL USR(BUF,I)
240   NEXT I
250 NEXT K
260 DATA "üdvözlet"
270 DATA "az"
280 DATA "olvasoknak"
290 DATA "és"
300 DATA "minden"
310 DATA "ENTERPRISE"
320 DATA "barátnak"
330 DATA "----"
340 DATA "Ladislas"
350 DEF LIGHTER
360   SET COLOR 1,4
370   CALL WAIT
380   SET COLOR 1,176
390   CALL WAIT
400   SET COLOR 1,60
410   CALL WAIT
420   SET COLOR 1,252
430   CALL WAIT
440   SET COLOR 1,39
450   CALL WAIT
460   SET COLOR 1,231
470   CALL WAIT
480   SET COLOR 1,63
490   CALL WAIT
500   SET COLOR 1,255
510   SET INK 2
520   PRINT #101,AT 10,(20-(LEN(A$)))/2+1;A$
530   SET PALETTE 0,0,255
540   SET INK 1
550 END DEF
560 DEF WAIT
570   FOR WE=1 TO 20
580   NEXT WE
590 END DEF
```

1. lista

```
100 SET 0,0
110 SET VIDEO X 30
120 SET VIDEO Y 100
130 SET VIDEO MODE 1
140 SET VIDEO COLOR 2
150 OPEN #1:"video:"
160 SET #1:PALETTE 0,GREEN
170 FOR I=1 TO 40
180   READ A$
190   PRINT #1,AT I,10-LEN(A$)/2-2:A$
200 NEXT I
210 !
220 FOR W=1 TO 80
230   DISPLAY #1:AT 1 FROM W TO W+20
240 NEXT W
250 GOTO 210
260 DATA ""
270 DATA ""
280 DATA ""
290 DATA ""
300 DATA ""
310 DATA ""
320 DATA ""
330 DATA ""
340 DATA ""
350 DATA ""
360 DATA ""
370 DATA "--------------"
380 DATA ""
390 DATA "Köszöntöm a"
400 DATA ""
410 DATA "µM olvasoit !"
420 DATA ""
430 DATA ""
440 DATA "es"
450 DATA ""
460 DATA "minden"
470 DATA ""
480 DATA "ENTERPRISE"
490 DATA ""
500 DATA "tulajdonost !"
510 DATA ""
520 DATA "Ez egy"
530 DATA ""
540 DATA "UP-SCROLL"
550 DATA ""
560 DATA "program"
570 DATA ""
580 DATA "amit mindenki"
590 DATA ""
600 DATA "felhasználhat"
610 DATA ""
620 DATA "a saját"
630 DATA ""
640 DATA "programjában"
650 DATA "--------------"
```

2. lista

Ezzel a módszerrel nem csak grafikát tudunk kimenteni, hanem hangkezelőre irányuló escape szekvenciákat is. Ilyenkor a SOUND egység helyett nyitjuk meg a magnót. Az eljárás ugyanaz, mint a grafikánál. Ekképpen például kiszedhetjük akár egy gépi kódú játékprogram zenéjét is, csak az ASMON-nal kell egy picit belenyúlnunk a program elejébe. Átírjuk a SOUND:-ot TAPE:-re, és a csatornamegnyitás EXOS funkcióhívásait EXOS 1-ről EXOS 2-re. Aztán kimentjük az átírt programot, és a betöltésénél gyorsan egy üres kazettát behelyezve felvételre állunk. Természetesen a magnó csatornája soha nem záródik le a gépi programból, ezért fájl vége jel sem lesz, de ezen a problémán segíthetünk. A fájl-jel nélküli állományt a SUPER COPY-val ki lehet javítani.

E rövid trükk után jöjjön a négyesfogat, azaz négy kis program, melyekből az Enterprise-szal ismerkedők tanulhatnak. Tömören a programokról. Az *1. listán* látható program egy nagybetűs felfelé-scroll. A pontonkénti mozgatást lényegében egy gépi kódú rutin végzi. A másik nagybetűs felfelé-scroll olyan felfelé futó feliratokat mutat be, melyek fent kifutnak. A BASIC-ben íródott program az Enterprise videolap-technikáját használja *(2. lista)*. A harmadik program gépi kódú *(3. lista)*. Ez a gép USER-INTERRUPT-jára példa. A scrollozás az elsőhöz hasonlóan pontonkénti. A negyedik programban, mely BASIC-ben íródott, a szöveg szintén pontonként lép balra. A kisbetűs balra-scroll az Enterprise LINE PARAMETER TABLE-jébe nyúl bele *(4. lista)*.

Göbölös László

```
  1 ! Scroll megszakitásban
  2 !
 10 POKE 48896,33
 20 POKE 48897,20+16
 30 POKE 48898,185
 40 POKE 48899,52
 50 POKE 48900,201
 60 !
 80 SPOKE 255,49134,191:SPOKE 255,49133,0
 90 SPOKE 255,49135,255
100 GRAPHICS HIRES 4
110 SET PALETTE 0,GREEN
120 PRINT #101,AT 2,1:"üdvözlem a µM olvasoit !"
```

3. lista

4. lista

```
100 PROGRAM "pixel_scroll"
110 SET VIDEO MODE 1
120 SET VIDEO COLOR 1
130 SET VIDEO X 42
140 SET VIDEO Y 1
150 OPEN #1:"video:"
160 DISPLAY #1:AT 1 FROM 1 TO 1
170 SET #1:PALETTE 0,GREEN
180 PRINT #1,AT 1,1:"PIRANA SOFT CO scroll rutine"
190 POKE 47376+8,RED
200 POKE 47376+9,YELLOW
210 FOR I=0 TO 83
220   POKE 47380,I
230   FOR K=1 TO 20
240   NEXT K
250 NEXT I
260 GOTO 210
```

TOP LISTA

	játék programok	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE
1.	Rocket R.		●		●					●
2.	Ocean R.		●		●					
3.	UGHlympics		●		●					
4.	Grand Prix C		●		●					●
5	Robocop		●		●					●
6.	Battle Chess	●	●							●
7.	Riziko				●					
8.	Bard's Tale				●	●				
9.	Bosszu				●					
10	Sanghai	●								

	felhasználói programok	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE
1.	Giga Paint				●					
2.	dBase IV.	●								
3.	Amica Paint				●					
4.	Giga Cad +				●					
5.	Art Studio				●		●			
6.	Windows	●			●		●			
7.	Printmaster	●			●					●
8.	News Room	●			●					●
9.	Paper Clip 3		●		●					●
10	Texter				●					

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

Címünk:
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
Diákszerkesztőség

A számítógépek motorja V.

Tárkezelés

A tárolóegységek vagy memóriák igen fontos szerepet játszanak minden számítógéprendszer működésében. Ezért az erőforrások kezelésében a tárolóegységekkel való gazdálkodás körülbelül a processzorkezelésnek megfelelő súlyt képvisel. Táron vagy memórián most a gép belső, operatív tárolóját értjük, bár mint látni fogjuk, a külső tárolóegységek szintén kulcspozíciót töltenek be a tárkezelésben. Ebben a részben a fejlődés sorrendjében tekintjük át a különböző szintű tárkezelési módszereket.

Abszolút címzésű rendszerek

Az egyszerű, egyfelhasználós számítógéprendszereknél a tárat két részre osztották. Az egyiket a felhasználó kapta, a másikat az operációs rendszer rezidens része — ami lényegében egy betöltő programból állt — használta. A programok és a memóriacímek közötti kapcsolatot már a program írásakor rögzítették, azaz a program csak a tár egy bizonyos, abszolút címekkel megadott területén volt működőképes. Az utasítások közvetlenül címezhették a teljes memóriát, így a tárkezelésre igen kevés feladat jutott, többek között például az operációs rendszer területének védelme. Erre a célra egy határregiszter szolgált, amely az operációs rendszer által elfoglalt legnagyobb vagy legkissebb címet tartalmazta annak megfelelően, hogy a rendszer a tár alsó vagy felső részén helyezkedett-e el. A privilegizált módban futó operációs rendszer a teljes memóriához — a felhasználó területéhez is — hozzáfért. A felhasználó programjaiban azonban minden generált címet ellenőrizni kellett, hogy nem sérti-e a határregiszter által kijelölt rendszerterületet. Arra számítva, hogy az operációs rendszer mérete az idők folyamán változik (rendszerint növekszik), a határregiszter tartalmát egy — természetes privilegizált — utasítással lehetett módosítani.

Áthelyezhető címzésű rendszerek

Az abszolút címzés igen sok gépies munkát rótt a programozóra, ezért a programozási nyelvek elterjedésével gyorsan feledésbe is merült. Helyét átadta a legegyszerűbb áthelyezhető címzésnek, amit relatív címzésnek is szokás nevezni. Ennek lényege, hogy a fordítóprogram például 0-tól kezdődő relatív címet állít elő, a végleges címek kialakítását pedig a tárgyprogram betöltéséig késleltetjük. Ekkor a betöltő egy alkalmas érték hozzáadásával kialakítja az abszolút címeket. A megoldás előnye, hogy ha változik a határcím, a programot elegendő csak újratölteni, fordítani nem kell.

A következő lépést a tranziens részt tartalmazó operációs rendszerek képviselték, amikor is a rendszer által elfoglalt memória a felhasználói program futása során is változhatott, mivel a ritkán használt szolgáltatásokat nem volt célszerű állandóan az akkor még viszonylag szűkös méretű tárban tartani. A legegyszerűbb megoldást egy régi PDP—11-es rendszerben találhatjuk meg, ahol az operációs rendszer a tár elején a növekvő címek szerint foglalt helyet, a felhasználó programját pedig a tár tetejétől lefelé töltötték be. A kettő közötti szabad területet akár a rendszer, akár a felhasználó elfoglalhatta. Az általános megoldást a CDC 6600-as gépeken valósították meg, ahol a címszámítást egészen a végrehajtás pillanatáig késleltették. Ezt dinamikus áthelyezésnek nevezik, és némi hardvertámogatást igényel. Mint az *1. ábrán* látható, a határregiszter szerepét egy relokációs bázisregiszter veszi át, amelynek tartalma hozzáadódik a futó program relatív címeihez.

A felhasználó nem is látja a fizikai címeket: minden műveletét logikai (relatív) címekkel végzi. Ebben a rendszerben, ha változik a relokációs regiszter tartalma, a felhasználó teljes területét egyszerűen át kell csak másolni a kijelölt részre, a futó program kódja nem módosul. Ennek a viszonylag már régen alkalmazott megoldásnak a lényege — a logikai címek fogalma, melyeket az operációs rendszer fizikai címekre képez le — minden modern tárkezelő rendszer alapja.

Particionált rendszerek

A tárkezelés fejlődése természetszerűleg a memória több felhasználó közötti megosztása felé haladt. Ehhez a logikai és fizikai címek fogalmán kívül az kellett, hogy a tárkezelés háttértárakra is támaszkodhasson. Mivel viszonylag gyors, cím szerint írható-olvasható háttértárra van szükség, eleinte mágnesdobok, később (és jelenleg is) főleg mágneslemezek jöhettek csak szóba.

Az első gyakorlati megoldást az úgynevezett swapping (helycsere) jelentette, amely először egy rezidens operációs rendszerrel és határregiszterrel felszerelt hardveren működött, időosztásos alapon. A swapping lényege, hogy a felhasználói részt mindig egy felhasználó uralja, a többiek teljes memóriaterülete a háttértáron várakozik. Felhasználócserénél a rendszer kimásolja a régi, és behozza az új felhasználó „memóriaképét".

Mivel a helycsere ideje a kivitt, illetve behozott terület méretétől függ, a swapping szükségessé tette a programok memóriaigényének dinamikus kezelését. A rendszer sokkal hatékonyabban működött, ha csak a folyamatok által ténylegesen lefoglalt területeket cserélgette, nem pedig mindig a teljes felhasználói memóriarészt. Ekkor alakultak ki a memóriafoglalásra és -felszabadításra szolgáló rendszerhívások.

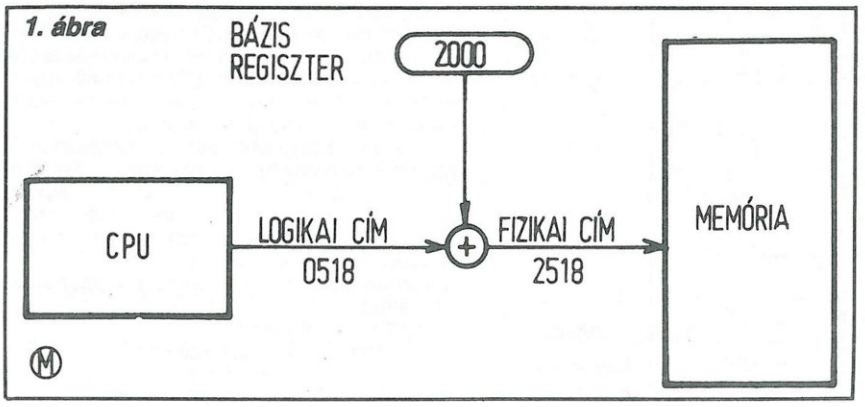
A swapping technika egyik nagy hátránya, hogy a csere csak passzív folyamatokkal hajtható végre. A B/K műveletek esetében azonban megtörténhet, hogy az operációs rendszer több aszinkron lépésben fordul a felhasználó területén lévő pufferekhez, tehát a folyamat passzív volta nem egyértelmű. Két ilyen művelet közötti swapping ugyanis végzetes lehet mindkét folyamatra nézve. A megoldás mindenképpen megszorítást jelent a B/K tevékenységre: vagy megtiltjuk a swappingot a teljes B/K művelet alatt, vagy B/K műveletet csak az operációs rendszer puffereivel lehet végezni, amit egy későbbi belső adatátvitel követ. Talán ez volt az oka annak, hogy a swapping átfedéses változatából kialakult a multiprogramozásnál általánosan elterjedt technika: egyszerre több programot kell a memóriában tartani.

A memória szabad területét több részre (partícióra) osztják, és minden partícióban egy-egy felhasználó programja tartózkodhat, végrehajtásra várakozva. A központi egység igen gyorsan kapcsol át egyik folyamatról a másikra, a memóriakezelés feladata pedig az, hogy a rendszeren kívül várakozó feladatokból folyamatosan feltöltse a partíciókat futásra kész programokkal. A multiprogramozás fokát a partíciók száma szabja meg: valahányszor egy partíció felszabadul, az operációs rendszer kiválaszt és betölt egy feladatot a várakozó sorból.

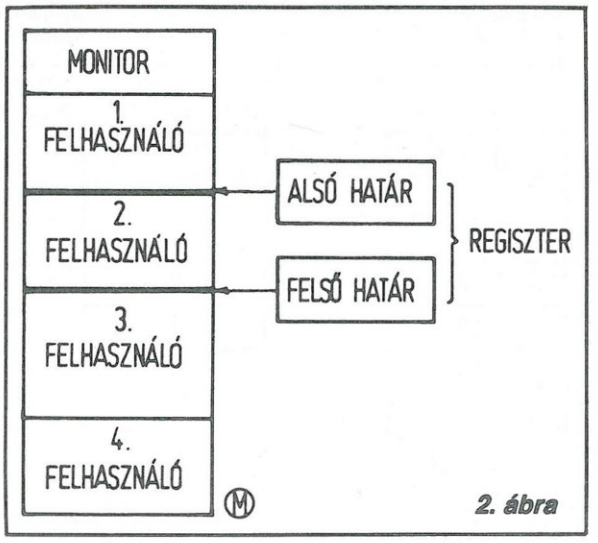
A particionálás kétféle módon oldható meg: statikusan vagy dinamikusan. A statikus megoldásban rögzített méretű partíciók szerepelnek, szemben a másik megoldás változó méretű partícióival. A két változatot az IBM MFT, illetve MVT néven terjesztette (a Multiprogramozás Fix és a Változó számú Feladattal kifejezések rövidítése), ami általánosan elfogadott elnevezéssé vált.

Logikai címterület kijelölése két határregiszterrel

Dinamikus áthelyezés



1. ábra



2. ábra

Mivel minden programot védeni kell az összes többi programtól, egy helyett két határregiszterre van szükség, amint ezt a *2. ábra* is mutatja.

A határregiszterek vagy a partícióban generálható legkisebb és legnagyobb címet tartalmazzák, vagy a legkisebb címet (báziscím) és a partíció hosszát, a limitet. Az első megoldás a programok csak statikus (fordításkor vagy később a betöltéskor rögzített) relokációját engedi meg. A bázis- és limitértékek segítségével már a logikai címek is ellenőrizhetők — nem haladhatják meg a limitet —, a bázis hozzáadásával pedig a relokáció futás közben, dinamikusan végezhető el. Az IBM 360 a statikus, a CDC 6600 és utódai a dinamikus relokációt alkalmazták.

Lapozás és szegmentálás

A patricionálás igen kellemetlen mellékhatása az úgynevezett elaprózódás. A folyamatos munka során olyan kis, nem összefüggő partíciók keletkezhetnek, amelyek egyenként egy feladat befogadására sem elegendőek, de összegük esetleg már használható lenne. Az elaprózódás ellen eleinte úgy védekeztek, hogy a memóriát időnként átrendezték — tömörítették —, az aktív feladatokat folyamatosan helyezték el, és így a szabad terület is egy folyamatos blokkot alkotott. A tömörítés természetesen meglehetősen időigényes, a felhasználó szempontjából teljesen haszontalan tevékenység volt.

Az elaprózódás leküzdésének másik, a tömörítésnél általánosabb megoldása az, ha feladjuk az egy feladatra kiosztott memóriaterület folytonosságnak elvét. A feladathoz több, a memóriában szétszórtan elhelyezkedő blokk tartozik, a rendszer pedig gondoskodik arról, hogy futás közben szükség esetén a vezérlés az egyik blokkból a másikba átkerüljön. A blokkokat lapoknak szokás nevezni, a módszert pedig lapkezelésnek vagy lapozásnak.

Minden, a központi egység által generált címet két részre bontunk: egy lapcímre és egy lapon belüli relatív címre. A lapcím adja a laptábla indexét, ahol minden laphoz megtalálható annak fizikai kezdő (bázis) címe.

A logikai és a fizikai tár közötti leképezés modelljét mutatja a *3. ábra*.

A logikai memóriát azonos nagyságú lapokra, a fizikai memóriát pedig ugyanekkora részekre, keretekre osztjuk. A programot végrehajtás előtt betöltjük az éppen szabad keretekbe, majd a laptáblát úgy állítjuk be, hogy a lapok és keretek aktuális megfeleltetését fejezze ki.

A lapméret hardver paraméter, a gyakorlatban 1 kbájttól vagy szótól 4 k-ig terjed. (Az IBM 370-nél 2 k vagy 4 k, a PDP—10-nél 512 szó.) Általában, ha a lapméret P, akkor az L logikai címhez tartozó lap-, illetve relatív cím:

p = L div P
d = L mod P

A lapméretet azért szokás kettőhatványnak választani, mert így a fenti műveletek a bináris címen igen egyszerűen elvégezhetők: a megfelelő számú felső bit adja p-t, az alsó bitek pedig d-t.

A feladatok ütemezése természetesen alkalmazkodik a lapozáshoz. A feladat méretét lapokban kell kifejezni. Ha az ütemező úgy találja, hogy rendelkezésére áll a feladat által igényelt számú keret, akkor ezeket lefoglalja, a feladatot laponként betölti rendre a keretekbe, és a keret sorszámát minden lap betöltése után beírja a laptábla megfelelő rovatába. Minden feladathoz külön laptábla tartozik, melyet a regiszterek tartalmával együtt a feladatvezérlő blokkban tárol.

Lapozásnál külső elaprózódás nincs, de fellép egy feladatonként várhatóan féllapnyi belső veszteség, hiszen a feladatok mérete általában nem egész számú többszöröse a laphossznak.

A laptábla tisztán szoftver megvalósítása meglehetősen lelassította volna a rendszert, ezért többféle, hardverrel támogatott megoldást alkalmaztak. Eleinte, amikor a logikai tár csak néhány tucat vagy néhány száz lapból állt, a laptáblát egy külön regiszterkészlettel valósították meg, ami laponként egy regisztert jelentett. A lapregiszterek tartalmát természetesen csak az operációs rendszer módosíthatta. A regiszterek a címtranszformációt igen gyorssá tették. Amikor azonban a memória mérete nőtt, a laptáblát kénytelenek voltak a memóriában tartani és bevezetni az úgynevezett laptáblát bázisregisztert (angol rövidítése: PTBR), amely mindig az aktuális laptáblára mutatott. Így a laptáblák váltása viszonylag gyors volt, a logikai-fizikai címtranszformáció azonban legalább két memóriához fordulást igényelt, ami sok esetben elfogadhatatlan sebességcsökkenéshez vezetett.

A probléma általánosan elfogadott megoldása egy kis, speciális hardvermemória beépítése, amelyet tartalom szerint címezhető vagy asszociatív tárnak hívnak. Ez a memória igen gyors, asszociatív regiszterekből állt, melyek együtt tartalmazták a lapcímet és keretsorszámot. Ha megadtunk egy lapcímet, az egyszerre került összehasonlításra az összes tárolt értékkel, a megtalált regiszterben pedig ott volt a megfelelő keretsorszám is. A keresés igen gyors volt, csak mintegy tíz százalékkal több a közvetlen címzés idejénél. A szükséges hardver viszont meglehetősen költséges.

Éppen a magas hardverköltség miatt az asszociatív tárak általában csak a laptábla egy kis részét tartalmazzák. Egy-egy konkrét címtranszformációnál két eset lehetséges: találatnál — ha a keresett lap éppen az asszociatív tárban van — nincs probléma, különben pedig a memóriából kell elővenni a keresett keretsorszámot, amit nemcsak felhasználunk, hanem be is írunk az asszociatív tárba, további hivatkozásokra számítva. A találati valószínűség nyilván elsősorban az asszociatív regiszterek számától függ. 8 vagy 16 regiszterrel általában 80-90 százalékos találati arány érhető el, ami egy átlagos, 750 ns-os memóriaelérési és 50 ns-os asszociatív keresési idővel számolva csak mintegy 16-26 százalékot lassít.

A lapozás másik jelentős előnye, hogy a közös programok több felhasználó között megosztva futtathatók. Akárhányan dolgoznak például egy szövegszerkesztővel, ha annak kódja újrabelépthető, elegendő egyetlen példányt tartani a fizikai memóriában, és mindenki ezt használhatja. Egy program kódja akkor újrabelépthető (angolul: reentrant), ha futás közben nem változik. Ilyenkor az osztott felhasználásnak nincs akadálya, ha minden folyamatnak saját regiszterkészlet-másolata és adatterülete van. Ha n felhasználó megosztva futtat egy programot, a megtakarítás a program memóriaigényének (n-1)-szerese.

A lapozással kapcsolatos memóriavédelmet laponként valósítják meg, egy vagy több védőbit segítségével. A védőbiteket általában a laptáblában tartják. Egy bittel egy lapot csak olvashatónak vagy írható/olvashatónak lehet minősíteni. Amíg a fizikai cím meghatározódik, ellenőrizni lehet a védőbitet is. A védelem megsértése címzési hibát, megszakítást eredményez. A védelmet gyakran tovább finomították — általában hardvertámogatással —, például úgy, hogy csak olvasható, írható/olvasható és végrehajtható minősítéseket lehessen kezelni. A laptáblabeli védőbitet jól lehet érvényes-érvénytelen jelzésként is használni olyan feladatoknál, melyeknél a címterület a feladat méretéből következően nem fedheti le a teljes címtartományt. Így a program címzési hibái könnyebben behatárolhatók.

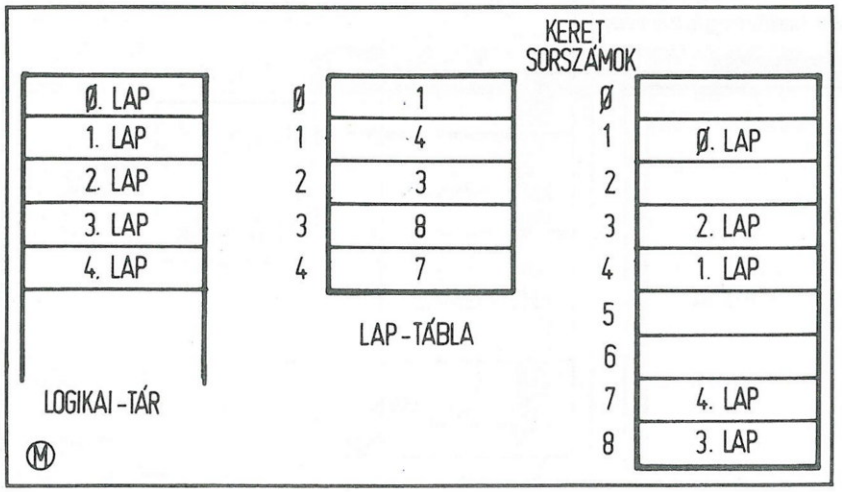
Az a tény, hogy lapozásnál a felhasználó a fizikaitól különböző memóriát lát, további lehetőségeket is tartogat. Mivel a logikai címek fizikaiakra való leképzését az operációs rendszer egy rejtett címfordító eljárása végzi, megvan annak a lehetősége, hogy a logikai és fizikai memória ténylegesen különböző méretű legyen. Ha például a logikai memória a fizikai tár negyedrésze (minden felhasználó csak legfeljebb ekkora tárigényű programot írhat), akkor a tárat tekintve automatikusan biztosítva van a négyszeres multiprogramozás. Ezt a technikát a 60-as években (amikor a memória még drága volt) tervezett kisgépek alkalmazták előszeretettel az olcsó félvezetős memória megjelenésekor.

Mivel a címfordításra mindenképpen szükség van, érdemes a felhasználók kedvében járni azzal, hogy a logikai tár olyan legyen, amilyet ők elképzelnek. Ez pedig nem bájtok lineáris, számozott sorozata, hanem funkcióval bíró, különböző nagyságú, sokszor rendezettséget nem mutató szegmensek gyűjteménye. Egy-egy szegmens lehet egy eljárás, egy modul, egy táblázat, a főprogram stb. A felhasználó számára ezek memóriabeli helye és sorrendje teljesen közömbös, a szegmensek mérete különböző, hivatkozni rájuk a nevükkel, elemeikre pedig egy — a szegmens kezdetétől mért — relatív távolsággal lehet (például egy program vagy táblázat 20. sora).

A szegmentálás olyan tárkezelés, amely ilyen logikai memóriával dolgozik. A logikai címtartomány szegmensek halmaza, melyek mindegyikéhez tartozik egy név és egy hossz. A logikai cím a szegmens nevéből és a szegmensen belüli relatív címből áll. Emlékezzünk vissza, hogy a lapozásnál a felhasználó egyetlen címét a hardver vágta szét lap- és relatív címre, a felhasználó „háta mögött".

Az egyszerűbb megvalósítás érdekében a szegmensekre név helyett egy szegmensszámmal hivatkozunk. A tárgyprogramot általában fordítóprogram állítja elő, amely automatikusan úgy építi fel a szegmenseket, hogy azok a forrásprogram szerkezetét tükrözzék. Egy Pascal-fordító például a következő szegmenseket generálhatja:

— globális változókat,
— vermet az eljárások számára,
— az eljárástörzsek kódját,
— lokális változókat eljárásonként.



3. ábra. Lapozási modell

A felhasználó kétdimenziós logikai címeit természetesen most is fizikai címekké kell alakítani, amit egy szegmenstábla segítségével tehetünk meg.

Az ötlet egyszerűnek tűnik, és sok előnnyel kecsegtet. A legszembetűnőbb talán az, hogy a felhasználó programjai meghaladhatják a fizikai tárméretet. Gyakran előfordul azonban, hogy az egyszerű ötletek megvalósítása nem megy könnyen, amint ezt a virtuális tár esetében is látni fogjuk.

Az a felismerés, hogy egy program végrehajtásához rendszerint nincs szükség a teljes program jelenlétére, nem új. A többmenetes assemblerek vagy egyéb fordítóprogramok jó példák erre, de ha jó a tervezés, akkor az adatfeldolgozó rendszerek is egymással csak adatstruktúrákon keresztül érintkező modulokból állnak. Ma már egyes személyi számítógépeken megvalósított nyelvek (például a Pascal) is megengedik a programok átlapoló (angolul: overlay) végrehajtását.

Az általános megoldás szerint a program egy rezidens és egy tranziens területen helyezkedik el. A vezérlő rész a rezidens területen működik. Fő feladata, hogy a tranziens részre a program logikája szerinti sorrendben behívja az egymást átlapoló programrészeket. Természetesen több tranziens rész is lehet, de az egy részre „beosztott" modulok egymást nem hívhatják. A tranziens részek hívása értelemszerűen egy dinamikus betöltést is magával vonhat, hacsak a hívott modul nincs éppen a tárban.

Ha ezt a megoldást a programozási nyelv kereteiből kiemelve az operációs rendszer szintjén valósítjuk meg, virtuális memóriáról beszélünk. A virtuális tár a rendszer számára legalább három szempontból előnyös:

— A programokat nem köti a fizikai memória mérete, a nagy virtuális címtartomány felhasználásával feltehetően egyszerűsödik a programozók dolga, munkájuk termelékenyebb lehet.

— Mivel az egyes felhasználók kevesebb fizikai tárat kötnek le, több felhasználó szolgálható ki, növekszik a központi egység kihasználtsága és a rendszer átbocsátóképessége.

— Kevesebb B/K műveletre van szükség, elmarad a feladatok ki-be másolása, a rendszer hatásfoka javul.

A virtuális memória megvalósítása több, néha bonyolult tevékenységből áll. Többféle gyakorlati megoldás létezik. Közülük a legelterjedtebb az igény szerinti lapozás (angolul: demand paging). Ez a technika leginkább a swappinggal kombinált lapozásra emlékeztet. A programot végrehajtás előtt esetleges cserével egybekötve olvassuk be. Az eltérés annyi, hogy a beolvasó — a swapper — csak akkor olvas be egy lapot, ha arra a program futása során ténylegesen szükség van. Annak érdekében, hogy a program ne próbáljon még még be nem hozott lapon lévő címre hivatkozni, a laptáblában a már említett érvényességi bitet lehet használni. Legyen ez a bit 1, ha a lap a memóriában van, különben pedig 0. A címtranszformáció természetesen az érvényességi bit vizsgálatával indul. Figyeljük meg, hogy ha a program nem hivatkozik a kérdéses lapra, nem is derül ki, hogy az nincs a memóriában.

Ha az érvényességi bit alapján kiderül, hogy a lap még nem elérhető, „laphibáról" beszélünk, de tudjuk, hogy ez az operációs rendszer által a gazdaságosabb üzemelés érdekében tudatosan elkövetett „hiba", amit a rendszer a következő lépésekben korrigál:

— A folyamat vezérlő blokkjához tartozó táblában megnézi, hogy a kérdéses címet a folyamat jogosan használja-e. Ha nem, akkor igazi címhibáról van szó, a folyamatot meg kell szakítani.

— Ha a cím jogos, a kérdéses lapot be kell olvasni. Ehhez először egy szabad keretre van szükség, majd ütemezni kell a lemezről való olvasást, végül módosítani kell mind a folyamat saját tábláját, mind pedig a laptáblát, hogy az új helyzetet fejezze ki.

— Újra kell indítani a laphibát okozó utasítást.

Az igény szerinti lapozás lényegében nem igényel több hardvertámogatást, mint a swappinggal egybekötött közönséges lapozás, bár arról is gondoskodni kell, hogy a laphiba miatt félbeszakadt utasítás újraindítható legyen. Ez a legtöbb esetben automatikusan teljesül. Vannak azonban olyan utasítások is, amelyek például blokkosított adatátvitellel több cím tartalmát változtatják meg.

A multiprogramozás fokozásával könnyen előállhat az a helyzet, hogy a párhuzamosan futó programok számára együttesen kiosztott memória meghaladja a fizikai tár méretét. Ilyenkor ki vagyunk téve annak, hogy laphiba esetén azért nem lehet behozni a kívánt lapot, mert nincs szabad keret a fizikai tárban. Ebben az esetben két megoldás van: a swapping vagy a lapcsere.

Lapozási stratégiák

A virtuális tárkezelés tulajdonképpen minél gazdaságosabb igény szerinti lapozást követel, amihez viszont két központi problémakört kell megoldani: megfelelő lapcsere-algoritmusokra és jó keretkiosztási módszerekre van szükség.

A lapcsere-algoritmusok célja a laphibák arányának csökkentése. Az algoritmusokat egy címhivatkozási próbasorozattal lehet értékelni, amikor is meghatározzák a laphibák számát. A próbasorozatot mesterségesen vagy valódi programokból vett minták alapján szokták előállítani. Az egyszerűség kedvéért a próbasorozatban elegendő csak a lapcímeket feltüntetni, és értelmetlen egy lapcímet kétszer egymás után szerepeltetni, hiszen a második hivatkozásnál soha nem fordulhat elő laphiba. Természetesen a laphibák számának meghatározásához ismerni kell a rendelkezésre álló keretek számát is.

Az egyik legegyszerűbb algoritmus az úgynevezett „előbb be — előbb ki" (angolul: First-In-First-Out, rövidítve FIFO). A behozott lapokat egyszerűen egy sor végére állítjuk, ha laphiba van, akkor az „áldozatot" a sor elejéről vesszük. Példaként tekintsük a következő próbasorozatot:

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Ha feltesszük, hogy a memória három keretből áll, a *4. ábra* mutatja a laphibákat.

A húsz címhivatkozás tehát 15 lapcserét igényelt. A FIFO algoritmust egyszerű megvalósítani, de hatásfoka néha elég rossz, különösen, ha van egy nagyon gyakran használt, központi irányító lap a programban, amit állandóan célszerű lenne bent tartani, de ez az algoritmus sokszor lecseréli.

Létezik egy optimális lapcsere-algoritmus (OPT): azt a lapot kell lecserélni, amelyet a folyamat a leghosszabb ideig nem fog igényelni. Ez a módszer adott keretszám mellett garantálja a minimális laphibaszámot. Az *5. ábra* az eredeti húsztagú próbasorozattal mutatja be az optimális algoritmust.

Mint látható, az OPT kilenc laphibát ad, de mivel ebből az első három elkerülhetetlen, azt mondjuk, hogy az algoritmus megfelezte a FIFO laphibáinak számát. Mivel azonban az OPT megvalósításához előre ismerni kell a címhivatkozások sorozatát, az algoritmust csak összehasonlító értékelésre használják. A gyakorlatban legtöbbször meg kell elégednünk az OPT valamilyen közelítésével.

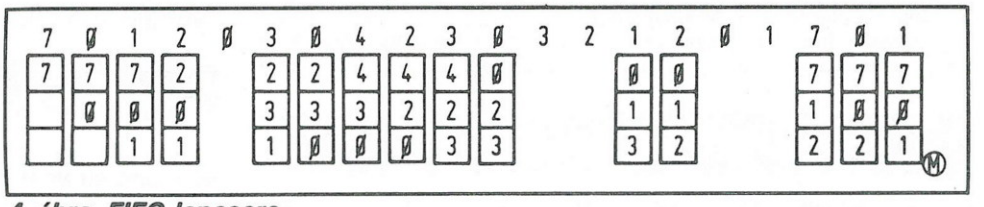
Keretkiosztási algoritmusok

A lapcsere-algoritmus birtokában a memóriagazdálkodás következő megoldandó problémája a rendelkezésre álló keretek szétosztása a multiprogramozásban résztvevő folyamatok között. Az egy folyamathoz rendelhető keretek száma nemcsak felülről, hanem alulról is korlátozott. A felső korlátot a keretek száma, az alsót a számítógép architektúrája határozza meg, mivel — mint már említettük — laphiba esetén a félben hagyott utasítást újra kell indítani. Ha az utasítások egyciműek, legalább két, ha indirekt címzés is megengedett, legalább három keret kell minden folyamatnak, hiszen az egyik lapon levő utasítás címe hivatkozhat indirekten egy másik lapra, ahol a harmadik lapra mutató cím van. Általában legalább annyi keretet kell egy folyamathoz rendelni, amennyi az egyetlen utasítással érinthető különböző lapok mentéséhez szükséges. Az egy szónál hosszabb utasításoknál a helyzetet bonyolítja, hogy maga az utasítás is két lapra kerülhet.

Anélkül, hogy az egyes folyamatokhoz rendelt keretek számát megadnánk, két alapvető stratégiát különböztethetünk meg: a globális és a lokális kiosztást. Globális kiosztásnál minden folyamat a szabad keretek közös halmazából elégítheti ki keretigényét, a lokális esetben pedig a folyamat csak a saját kereteivel gazdálkodik. A két megoldás közötti eltérés — és egyben a globális kiosztás hibája — az, hogy egy feladat nem szabályozhatja a saját laphiba-gyakoriságát, mert ez a többi folyamat viselkedésétől is függ.

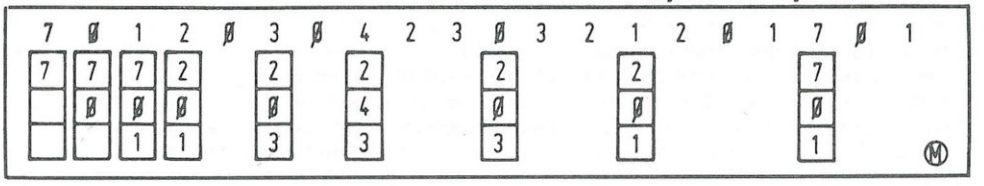
A legegyszerűbb algoritmus az egyenlő kiosztás: minden folyamathoz ugyanannyi (n keret és m folyamatnál n div m) keret tartozik, a maradék pedig a szabad keretek halmazát alkotja. Ha figyelembe vesszük, hogy a nagyobb programok általában több keretet igényelnek, mint a rövidek, az egyes folyamatok virtuális tárigényeinek aránya alapján az egyenlő kiosztást arányos kiosztásra finomíthatjuk.

A rendszer működése során mindkét esetben a folyamatok keretállománya a multiprogramozás fokával fordított arányban változik.



4. ábra. FIFO lapcsere

Optimális lapcsere 5. ábra



A következő lépés lehet a folyamatok prioritásának figyelembevétele: magasabb prioritásnál több memóriával gyorsítani lehet a folyamatot. Hasonló hatást érhetünk el úgy is, hogy lapcserénél az alacsony prioritású folyamatoktól veszünk el memóriát. Ilyen stratégiánál azonban körültekintően kell eljárni, mert ha az egy folyamat rendelkezésére álló keretek száma az abszolút szükséges minimum közelébe kerül, fellép az úgynevezett vergődés (angolul: thrashing), amikor a folyamat több időt tölt lapcserével, mint futással. Globális kiosztásnál az egy folyamat vergődése miatt létrejövő magas laphiba-gyakoriság láncreakciószerűen elterjedhet a rendszerben, ami a központi egység kihasználtságának hirtelen zuhanását eredményezi. Ilyenkor csak a multiprogramozás fokának csökkentése jelent megoldást. Lokális kiosztásnál a vergődés kisebb problémát okoz: a többi folyamat memóriája megmarad, csupán a lapcserére váró sor és ezzel együtt a folyamatok futási ideje növekszik.

A vergődés elkerülésére többféle módszert dolgoztak ki. Az egyik lehetőség azon a feltételezésen alapszik, hogy egy folyamat kis környezetből kis környezetbe vándorolva működik, egy ideig főleg lokális címekre hivatkozik, majd átáll egy másik környezetbe és ott újra a lokális címhivatkozások dominálnak stb. Ezt hívják röviden lokalitásnak. A lokalitás kihasználásával meg lehet keresni és állandóan karbantartani azt a lapcímhalmazt, amely várhatóan az aktív lapokat tartalmazza. Ennek érdekében felvesznek egy H értéket, és a kritikus halmazba besorolják az első H memóriahivatkozásban szereplő lapcímeket. Ezt az eljárást időnként megismétlik. Ha a H értékét jól becsültük — a szakirodalom 10 000-et javasol —, a program lokalitásai követhetők.

Egyéb megfontolnivalók

A lapozási és lapkiosztási algoritmusok igen fontosak lehetnek, de néhány egyszerű ötlet is nyújthat további segítséget. Az igény szerinti lapozásnál nyilvánvaló, hogy a program indulásakor azonnal sok laphiba keletkezik, és ugyanez történik a swapping során is, amikor a folyamat újra visszakerül a memóriába. Ezen segíthet az előlapozás. Ha például a folyamatok aktív lapjainak halmazát a rendszer folyamatosan nyilvántartja, az előlapozás azt jelenti, hogy minden megszakítás utáni folytatáskor az aktív lapokat kérés nélkül betöltik a memóriába, és ezzel egy laphibasorozatot eliminálnak.

Egy másik fontos tervezési szempont a lapok mérete. A két egymásnak ellentmondó cél a laptábla méretének csökkentése, illetve a belső elaprózódás féken tartása, ami a lapok méretének csökkentésével azonos. Ha ehhez még hozzávesszük a lapok kivitelének és behozatalának idejét is, mint újabb szempontot, meglehetősen komplex problémát kapunk, amelynek nincs egyértelmű megoldása. Még azt sem lehet általában eldönteni, hogy a B/K időt a lapméret csökkentése vagy növelése redukálja. Mivel az átvitelnél nem az átviteli, hanem a várakozási idő dominál, nagyobb lapmérettől kevesebb átvitel várható. Ugyanakkor a kisebb lapmérettel a lokalitás jobban kihasználható és több szabad memória marad, mert nem hozunk be egy lappal együtt sok felesleges, soha nem használt bájtot is.

Bakos Tamás

Hogyan szövegeljünk?

A szövegfeldolgozás elemei

Beszúrás — átírás

A szöveg legegyszerűbben úgy módosítható, hogy a kurzort a megfelelő helyre visszük, majd választunk a beszúrás, illetve az átírás mód között. A rendszer alaphelyzetben általában az előbbi módban van — így nem tűnik el szándékolatlanul információ a képernyőről —, azaz a leírt karakterek a meglévő szöveget „széttolva" kerülnek a helyükre. A két üzemmód közötti váltásra valamelyik vezérlőbillentyű (legtöbbször az INS) szolgál. Az üzemmódot a képernyő elkülönített részén jelezni szokták.

Törlés — mozgatás — másolás

Rövid szövegdarabok a BACKSPACE billentyűvel törölhetők. Mivel ilyenkor a törölt szöveg menthetetlenül elvész, gondoskodni kellett a törlés barátságosabb megoldásáról is. Maga a törlés viszonylag egyszerűen kezelhető a kurzor kiterjesztése után alkalmazott DEL billentyűvel. A korszerű szövegfeldolgozó rendszerek tovább tökéletesedtek: összekapcsolták a törlést szövegrészek átvitelével és másolásával. A művelet lényege, hogy a kurzorkiterjesztéssel vagy más módon kijelölt szövegrész a DEL billentyű vagy a *Delete* parancs hatására törlődik az iratból, de tárolódik egy pufferben, ameddig egy újabb törlés felül nem írja. Ezután a szöveg átvitele arra egyszerűsödött, hogy a kurzor megfelelő pozicionálása után a puffer tartalmát az INS billentyűvel vagy egy *Insert* paranccsal beszúrjuk a megfelelő helyre.

Másolni hasonló módon, de a *Copy* parancs hatására lehet. A különbség az, hogy ilyenkor a szövegrész az eredeti helyén is megmarad.

Rövidítések

A szövegmozgatás további általánosítása lehetővé teszi, hogy az iratban gyakran előforduló szavakat, frázisokat vagy akár nagyobb szakaszokat is névvel ellátva (ami tulajdonképpen egy rövidítés) tároljuk, és ismételt leírásuk helyett nevükkel és valamilyen vezérlőbillentyűvel helyezzük el a szövegben. Ennek megvalósításaként a rendszer a törléskor és másoláskor nemcsak egy, hanem több, névvel ellátott puffert használ. Az irathoz tartozó rövidítések természetesen egy állomány formájában is megőrizhetők és folyamatosan felhasználhatók. A rövidítések alkalmazása sokat segít egyrészt a nem magyar szabvány szerinti billentyűzet, másrészt a gépírásban járatlan felhasználók problémáin. A jól használható rendszerek tervezői arra is gondoltak, hogy a felhasználó egy idő után képtelen fejben tartani az egyre bővülő rövidítésgyűjteményt, ezért lehetőséget adtak arra, hogy a gyűjtemény használatakor annak tartalma egyszerűen megszemlélhető legyen. Néhány általános rövidítést (dátum, idő, lapszám stb.) a gyűjteményben a rendszer automatikusan elhelyez.

Keresés

Valamely szövegrész keresése különösen terjedelmesebb iratoknál hasznos művelet. Keresni tetszőleges karakterkombinációt lehet, a művelet a kurzor pillanatnyi helyétől indul előre vagy visszafelé. Kereséskor általában a kérdéses szöveg első előfordulásának felderítése és a kurzor pozicionálása a cél. A felhasználó választhat, hogy a mintát csak önálló szóként vagy egy másik szó részeként is meg akarja-e találni. Hasonlóképpen célszerű azt is a felhasználóra bízni, hogy a nagy- és kisbetűket különbözőnek tekinti-e, amikor keres.

Helyettesítés

Tulajdonképpen a kereséshez kapcsolódó azon művelet, amikor a megtalált karaktersorozatot egy másikkal helyettesítjük. Ezt megtehetjük automatikusan vagy jóváhagyással. Az utóbbi megoldás akkor hasznos, ha nem minden előfordulást kell kicserélni, de mindegyiket szemügyre akarjuk venni.

Ablakkezelés

A modernebb szövegfeldolgozó rendszerekben alkalmazott műveletcsoport, amely elsősorban a képernyő fizikai és funkcionális felosztását teszi lehetővé. A fizikai felosztás azt jelenti, hogy a képernyőt vízszintesen vagy függőlegesen ablakokra osztjuk, melyek mindegyike önállóan is használható. A létrehozott ablakok funkcionálisan is szétválaszthatók, azaz mindegyikben egy-egy önálló irat kezelhető. Az ablakok nyitásához, zárásához, az aktív ablak kiválasztásához és egyéb manipulációjához kiterjedt és változatos parancskészlet tartozik. Itt csak annyit említünk meg, hogy az ablakok akkor használatosak, ha például két iraton párhuzamosan dolgozunk — egy fordításon és a hozzá tartozó szakszótáron —, vagy az egyik iratból szövegdarabokat akarunk átvinni a másikba.

Formázás

A szövegfeldolgozás talán legösszetettebb műveletcsoportja. Az a célja, hogy az irat külső megjelenését a felhasználó speciális igényeinek megfelelően alakíthassa ki. A formázás egyaránt kiterjed a nyomtatott szöveg tördelésére, az iratban használható betűtípusokra, valamint szövegkiemelési megoldásokra (aláhúzás, vastagítás, dőlt betű stb.). A művelet általában az irat szerkezetéhez igazodva hajtható végre. Ennek megfelelően az elterjedtebb formázási lehetőségek a következők.

A *részek* formázása globális lehetőségeket tartalmaz a *lapszélek* (margók), a *lapszámozás,* a lapok *külalakja,* valamint a nyomtatott *sorok számozásának* vezérlésére. A lapszéleknél nemcsak a lap fizikai méretei állíthatók be, hanem a jobb és bal oldali, az alsó és felső lapszél. Ezenkívül a lapok későbbi bekötését megkönnyítő, ún. kiegészítő bal lapszél is szabályozható. A lapszámozásnál egyrészt a lapszám helye, másrészt kezdőértéke és formája (arab, római, betű) adható meg. A nyomtatás külalakja elsősorban az egy lapra kerülő oszlopok számát, valamint a lábjegyzetek kezelését határozza meg.

A *bekezdések* formázása már inkább lokális jellemzők kialakítására alkalmas. Sok — néha speciális — lehetőség közül csak az általánosan használtakat említjük meg. Egy bekezdés megjelenését elsősorban az elhelyezése határozza meg. A sorokat lehet a bal lapszélhez, a jobb lapszélhez, középre és mindkét lapszélhez egyszerre igazítani. Az utóbbi megoldást *sorkiegyenlítésnek* nevezik. A globális lapszélbeállítás mellett egy-egy bekezdésben saját, lokális lapszélbeállítás is végezhető, sőt a bekezdés első sorát ilyen szempontból általában külön lehet kezelni. A bekezdés formájához hozzátartozik még a nyomtatásnál használt sortávolság, illetve a bekezdés előtt és után kihagyandó üres sorok számának megadása, valamint egy olyan jelző használata, amely letilthatja a bekezdés szétvágását abban az esetben, ha az nem férne már el az aktuális lapon.

Ezek után a szerkesztési műveletekről a kurzorkiterjesztés (tetszőleges szövegdarab kijelölési lehetősége) miatt elegendő a *karakterekkel* kapcsolatosan tárgyalni, hiszen az itt elmondottak szavakra, sorokra vagy akár az egész iratra is alkalmazhatók. A karakterek formázása elsősorban a karakter jellemzőire vonatkozik. A sokféle közül a gyakrabban használtak a vastagított, a dőlt, az aláhúzott, az alsó vagy felső indexpozícióba írt és az aláhúzott karakterek. Sokszor megválasztható a karakter típusa, például a pica, roman, elite stb. és mérete is.

Az iratok megjelenését nagymértékben javíthatja egy jól megválasztott, minden lap tetején vagy alján megjelenő, ún. *állandó fejléc.* A fejlécek formázása elsősorban a pozíciójukra, valamint néhány speciális tulajdonságukra vonatkozik. Lehetőség van történetesen arra, hogy a páros és páratlan lapokra különböző fejléc kerüljön a főcím és az éppen futó fejezetcím jelölésére. Jól használható az a lehetőség is, hogy a lapszámot, illetve a dátumot a fejlécbe építve nyomtassuk.

Egyes alkalmazási környezetekben, ahol gyakran azonos formájú iratokra van szükség, jól használhatók az *előre gyártott iratformák.* Ezeknek a segítségével például igen egyszerűen generálhatjuk a különféle bekezdéseket vagy akár a szabványosított leveleket is. Az előre gyártott formák természetesen szintén formázással, speciális parancsokkal hozhatók létre.

Nyomtatás

Az iratkészítés végső célja rendszerint a szöveg kinyomtatása. A megfelelő színvonalú eredmények érdekében a tényleges nyomtatást összetett előkészítő tevékenység előzi meg, amely az irat lapokra tördeléséből, az eredményt hordozó eszköz kijelöléséből, a nyomtatóprogram kiválasztásából, a példányszám beállításából és más hasonló műveletekből áll.

A *laphatárok* manuális beállítására azért van szükség, mert bizonyos felsorolások, táblázatok vagy címek elhelyezéséről alkalomadtán legjobban közvetlenül a nyomtatás előtt lehet dönteni.

Előfordulhat az is, hogy a nyomtató helyett a formázott szöveget egy *lemezes állományba* érdemes kivinni és később, más munkával párhuzamosan az MS—DOS nyomtatóprogramjával (PRINT) kinyomtatni. Ilyenkor az eredményállományba pontosan azok a karakterek kerülnek, melyeket egyébként a nyomtatóra küldtünk volna.

A jó szövegfeldolgozó rendszerek közös jellemzője, hogy alkalmasak *különböző nyomtatókkal* való együttműködésre. Az egyszerű mátrixnyomtatóktól a pillangókerekeseken keresztül a lézernyomtatókig terjedő spektrumon csak úgy lehetett úrrá lenni, hogy a nyomtatórutinokat leválasztották a szövegfeldolgozó rendszerről és a felhasználóra bízták, hogy egy készletből a nyomtatójához melyiket választja. Egyes rendszerek még ennél is tovább mentek, és új *nyomtatórutin generálására* alkalmas eszközt is mellékelnek, amellyel az előre számításba vett nyomtatók is kezelhetők. Számítani kell arra is, hogy nem a teljes iratot, hanem csak *egy részét akarjuk kiírni.* Ilyenkor a választás alapja a lapszám vagy a kijelölt szövegdarab.

Ennek bizonyos értelemben ellentéte, amikor *több iratot egyesítve,* folyamatos lapszámozással, fejlécekkel stb. kell kinyomtatni.

Ha nem leporellóra, hanem különálló lapokra nyomtatunk, és a nyomtatón nincs lapetető, a nyomtatást minden lap végén *fel kell függeszteni* az új lap betöltésének idejére, amit szintén a nyomtatóparancs valamelyik opciója valósít meg. Végül a nyomtatóparancs használati értékét növeli, ha a nyomtatás bármikor egyszerűen *megszakítható,* mivel előfordulhat, hogy az íráskép mégsem azonos a tervezettel, begyűrődik a papír vagy beszorul a festékszalag.

Tárolás — betöltés

Az iratok lemezen való tárolása és beolvasása elvileg egyszerű, valójában azonban összetett tevékenység. Többek között ide tartozik ugyanis az iratok *törlése* (mind a tárból, mind a lemezről), több irat teljes *egyesítése,* valamint az iratok *nevének megváltoztatása* is. A rövidítések és az esetleges előre gyártott iratformák tárolását és beolvasását szintén ide sorolhatjuk. A nagyobb biztonság érdekében az irat új változatának kivitele előtt a rendszer az esetleges előző változatot is megőrzi (például más néven). Ha a felhasználó a módosított irat tárolása nélkül akarna kilépni a rendszerből, üzenet figyelmezteti arra, hogy munkájának eredményét elveszítheti.

Járulékos szolgáltatások

A szövegfeldolgozó rendszerrel együttműködő, de attól független alrendszerek gondoskodnak az előállított iratok további feldolgozásáról. A fejlesztők leleményessége ezen a területen is meglepő; mi az alábbiakban csak rövid ízelítőt adunk a választékból.

Nagyobb lélegzetű tudományos monográfiáknál igen hasznos lehet a *tárgymutató,* az index. Az automatikus tárgymutató-készítő lapokra tördelt iratokkal dolgozik, melyekben speciális jelzéssel látták el az indexbe felveendő szavak egy-egy előfordulását. Eredményül az irat folytatásaként megkapjuk a kijelölt szavak összes előfordulását, a hozzájuk tartozó lapszámokkal együtt.

Az indexkészítéshez hasonló tevékenység az automatikus *tartalomjegyzék-generálás,* amely a megjelölt címekhez táblázatos formában rendeli hozzá a megfelelő lapszámokat.

Mind az index, mind a tartalomjegyzék az eredeti irat egy részeként jön létre, tehát tetszőleges módon tovább szerkeszthető.

Speciális tartalmú iratoknál igen hasznos lehet a *rendező* alrendszer is. Rendezni lehet bekezdésenként vagy soronként, alfanumerikus vagy numerikus értékrend szerint, valamint növekvő vagy csökkenő sorrendben. Nem szabad elfeledkezni arról, hogy a magyar ékezetes betűk használatakor nem számíthatunk arra, hogy a rendezés helyes eredményt ad, hiszen az ékezetes betűk nem illeszkednek az eredeti karakterek rendezettségéhez.

Az angolhoz kötődő szövegfeldolgozók *helyesírást ellenőrző* és szavak elválasztását támogató alrendszerei, sajnos, a magyar nyelvű iratokhoz nem használhatók, de nyelvgyakorlásra vagy angol szövegek előállítására igen hasznosak lehetnek. Ezek az alrendszerek meglehetősen nagy szótárral dolgoznak, és kihasználják az angol nyelv viszonylag egyszerű ragozási szabályait. Egy hasonló célú és teljesítményű magyar rendszer elkészítése jóval bonyolultabb feladat lenne.

Ugyancsak nyelvfüggő, de nyelvtanulásra kiválóan alkalmas az a szolgáltatás is, amely egy adott szóhoz *szinonimákat keres* egy speciálisan szervezett lista, az ún. *tezaurusz* alapján. A felajánlott szinonimák közül a kiválasztott automatikusan az eredeti szó helyére íródik.

A szövegfeldolgozó rendszerek lehetőségeit elsősorban figyelemfelkeltő szándékkal tekintettük át. A téma iránt érdeklődők a kívánt rendszer használatát szaktanfolyamokon vagy a megfelelő dokumentáció önálló tanulmányozásával sajátíthatják el.

Befejezésül megjegyezzük, hogy a lézernyomtatók elterjedésével a szövegfeldolgozás új lendületet kapott, és betört a kiadói tevékenységbe is. A magyarul talán házinyomdának nevezhető DTP (az angol *Desk-Top Publishing* rövidítése) rendszerek, mint például a VENTURA, megfelelő szakértelemmel a professzionális szintet megközelítő nyomdatermékek előállítására alkalmasak. Ezeknek az ismertetése azonban külön cikket érdemel.

—s —s

Programozási fogások és melléfogások

Ez alkalommal egy nagyon elterjedt tévhitet szeretnék bemutatni és — legalábbis a Commodore gépekre vonatkozóan — megcáfolni. A szóban forgó tévhit leírásával először a Mikrovilág 1986. májusi számában találkoztam, ahol a 4. oldalon közölt Programgyorsítás című cikkben a következőket olvastam: „A GOTO utasítással kapcsolatban jó tudni, hogy a program a címet *úgy keresi meg, hogy az első sortól kezdve növekvő sorszám szerint végigmegy.* Ha tehát a gyakran használt szubrutinokat a program elején helyezzük el, akkor a keresésre fordított idő csökken, a program futása gyorsul."

Nem sokkal ezután jelent meg magyarul Data Becker—Novotrade kiadásban Hornig Trapp Weltner *További tippek és trükkök a Commodore 64-eshez* című kötete, melynek 30. oldalán *A szubrutinokat a program elejére tegyük!* cím alatt az előbbihez hasonló tartalmú szövegre bukkantam: „A szubrutinokat célszerű a program elejére tenni, mert a számítógép a *megadott sorszámot mindig a program elejétől kezdve keresi.* Minél később találja meg a gép a sorszámot, annál hosszabb ideig fut a BASIC program." Állításait a szerző két, az *1. listán* lévőhöz hasonló programmal próbálja alátámasztani. Nem sok sikerrel, mint az a programot követő néhány sorból is kiderül: „A bemutatott programok közötti különbség nem figyelhető meg eléggé, mivel a programok nagyon rövidek, és ezért az első programban sem kell nagyon sok sorszámot végignézni. Az összehasonlítást egyszer hosszabb programok esetében is érdemes kipróbálni."

Kipróbáltam a könyvben P21 és P22 sorszámmal közölteket éppúgy, mint az 1. listán itt bemutatottnak több változatát. Az előbbieket itt nem idézem, de az említett könyvben bárki megtalálhatja őket. Apró inkorrektséget is találtam bennük. Ugyanis a P22-ben — amely a program elejére tett szubrutin hívásának gyorsabb voltát hivatott bizonyítani — a hívott sorszám kétjegyű, ellentétben a másik program háromjegyű sorszámával, ezenkívül az összeadásban — valószínűleg elírás miatt — az X változó helyett a nulla értékű Z szerepel. A P21-es program 34 TI-egység alatt fut le, a P22-es 32 alatt, de az imént említett eltérések kiigazításával a két futási idő egyenlővé válik.

Az 1. lista programjával a program elejére tett szubrutin hívásával járó futási időt vizsgálhatjuk; ha a 220-as programsorba „GOSUB180" helyett „GOSUB260"-at írunk, megkaphatjuk a hívás helye mögé tett szubrutin hívásával kapcsolatos időt. A program méretét kettőspontokat tartalmazó sorokkal növeltem meg, a ciklus végértékének megnövelése a pontosabb mérést teszi lehetővé. Az időméréshez a könyvben szereplő algoritmust használtam, ugyanis a szokásos módszerem a TT változó használata és a sok ciklusléptés miatt lényegesen befolyásolja a mérési eredményt.

A listán látható program futási ideje 3617, a 260-as sort hívóé 3389 TI-egység. Ez ellentmond a fentebb idézett állításoknak, hiszen ha az interpreter mindig a program elején kezdené a sorszám keresését, mindenképpen az utóbbi futási időnek kellene nagyobbnak lennie. További próbaként a program elejére beszúrtam még 99 sort. A 180-as sort hívó változat végrehajtási ideje 7824 TI-egységnyire nőtt, míg a 260-ast hívóé változatlan maradt.

A Commodore 64-es belső felépítése című, sokszerzős Data Becker—Novotrade kiadvány 159. oldalán — a GOTO BASIC utasítást feldolgozó interpreter rutin leírásánál — megtaláljuk a magyarázatot: *ha a keresett sorszám nem kisebb az aktuálisnál, akkor a keresés az aktuális sortól kezdődik.* Valószínűnek tartom, hogy ez érvényes más elterjedt géptípusok (HT, Spectrum, TVC, Enterprise stb.) BASIC-jére is, de erre vonatkozóan nincsenek pontos információim.

A *2/a listán* a Mikrovilág 1987. márciusi számának mellékletében megjelent — C16-ra írt — Hanoi-torony program néhány sorát mutatom be. Sajnos az ehhez hasonló „programgyorsító fogások" igen sűrűn fordulnak elő a különféle számítástechnikai kiadványokban.

A program elején — a fenti babona szellemében — egy GOTO-val átugrott szubrutin található. Ennek feladata, hogy ellenőrizze, hogy a K$ változóba INPUT-tal beolvasott érték az egytől háromig terjedő számtartományba esik-e. Kétséges, hogy érdemes a program elejére tenni, hiszen — főképpen a körülményessége miatt — kiértékelése lényegesen több időt vesz igénybe, mint a program bármely részén való megkeresése. Célszerűbb a szubrutinhívások helyére a *2/b listán* látható utasítást beírni. Ezáltal a 2/a lista programrészletéből csak a képernyőtörlő utasítás marad meg.

Barna László

1. lista

```
100 goto 200
110 :
120 :
130 :
140 :
150 :
160 :
170 :
180 a=a+x
190 return
200 ti$="000000"
210 for x=1 to 10000
220 gosub 180
230 next x
240 print ti
250 end
260 a=a+x
270 return
```

2/a lista

```
   ...
10 PRINT"□":GOTO20
12 IFK$="1"THENRETURN
13 IFK$="2"THENRETURN
14 IFK$="3"THENRETURN
15 GOTO1250
20 ...
```

2/b lista

```
...
IF K$<"1" OR K$>"3" THEN 1250
...
```

Különleges ismeretlenek

Még a német nyelvű Enterprise számítógépek azon tulajdonosai, akiknek megvan az EXOS 2.1 leírása, sem ismerhetik pontosan, hogy milyen képességű a gépük. A leírás ugyanis csak az angol számítógépekről vesz tudomást, a német Enterprise-oknak a plusz 16 kbájtnyi ROM-ból eredő többlettudásáról még csak említést sem tesz. E hiányt pótolandó, az információk egy részét közkinccsé teszem.

Az Enterprise német üzemmódban még két speciális funkciót ismer az EXOS leírásban felsoroltakon kívül. E két funkció a két billentyűzeteszközhöz kapcsolódik (keyboard); tehát csak egy billentyűzet-csatornaszámmal működik. A két rutin hívása ugyanúgy történik, mint a §§DFKEY-é vagy a §§JOY-é. A leírásuk a következő (az első két szám az alfunkciókódok értékei hexadecimálisan, illetve decimálisan):

0CEH (206) : §§DPTR
kimenő értékek: DE = a ROM-bővítő változó- és munkaterületének címe a rendszerszegmensen (OOFH=255)

0CFH (207) : §§KEYB
bemenő értékek: 1. C = 0
DE = a billentyűzeteszköz felhasználó által megváltoztatható belső rendszerváltozók új, beállítandó értékeit tartalmazó 16 bájtos blokk kezdőcíme.
A 16 bájt rendre a következő rendszerváltozókat állítja be:

1—2. NORM_TBL
3—4. SHIFT_TBL
5—6. CTRL_TBL
7—8. ALT_TBL
9—10. KBD_XP1_ADDR
11—12. KBD_XP2_ADDR
13. SM_PL_LET
14. BG_PL_LET
15—16. a karakterképek leírásának címe

2. C≠0 (ha C nem egyenlő nullával)
DE = a karakterképek leírásának címe

A billentyűzet megszakítási rutinja (KBD_ISR) a lenyomott billentyűhöz tartozó kódok előállítására négy, úgynevezett billentyűzetdekódolási táblázatot használ: egyet a sima billentyű megnyomásánál, illetve egyet-egyet a SHIFT, a CTRL és az ALT billentyűk együttnyomásánál. E táblázatok címeit tartalmazzák a NORM_TBL, SHIF_TBL, CTRL_TBL, ALT_TBL nevű változók. Minden táblázat 80 bájtot foglal le. Eredeti címeik: NORM_TBL=0E53DH, SHIFT_TBL=0E58DH, CTRL_TBL=E5DDH, ALT_TBL=0E62DH.

A táblázatok a billentyűmátrixnak megfelelően tíz oszlopra és nyolc sorra bomlanak. Az oszlopok jelentik a billentyűzetmátrix sorait, vagyis a 0B5H (181) portra kiírandó értékeket. A bal oldali oszlop felel meg a port 0-ás értékének, a jobb szélső pedig a 9-esnek. A sorok a mátrix oszlopait prezentálják, vagyis a portról beolvasott bájt bitjeit, mégpedig úgy, hogy a felső sor tartozik a 7. bithez, az alsó pedig a 0-hoz.

A rutin kiemeli a táblázat megfelelő sorából és oszlopából az egybájtnyi értéket és legtöbbször ezt adja vissza a billentyű kódjaként. A nem létező sor-oszlop kombinációknál, tehát amelyekhez nem tartozik billentyű, és a kódot vissza nem adó billentyűknél, mint a SHIFT, CTRL, ALT, LOCK, az eredeti táblázatok 2AH (42, a *, szorzásjel kódját) tartalmazzák. Néhány esetben a táblaelem nem a billentyűkódot, hanem a belső kódot tartalmazza. Ezeket meg kell őrizni az új táblázatokban is, ha azt akarjuk, hogy funkciójuk helyes legyen. Igy a STOP billentyűnek megfelelő bájt mind a négy táblázatban 7FH (127) értékű. A billentyűkódként jelentkező három érték csak másodlagos.

Noha úgy tűnik, hogy a PAUSE (HOLD) billentyűnek nincs kódja, ez csak részben igaz, mert belső kódja azért van. Értéke 0C0H (192); igaz, billentyűzetkód ebből nem generálódik. A funkcióbillentyűk helyes működéséhez mind a négy táblázatban 0F0H—0F7H értékeknek (240—247) kell szerepelniük, persze a megfelelő billentyűnél a megfelelő kódnak. A STOP, a PAUSE és a funkcióbillentyűket a KBR_ISR a belső kódjukról ismeri fel. Ha ezeket a kódokat más billentyűhöz írjuk be, az a billentyű is úgy viselkedik, mintha az előbbiek közül lenne egy. A funkciót ezért a billentyűzet átdefiniálására is használhatjuk; ha például a nyomtatónk a 40H (64) kódot értelmezi É karakterként, akkor ezt kényelmesebb a billentyűzetről többek között <CTRL> E-ként bevinni, mint észben tartani, hogy a § karakter jelenti a nagy É betűt.

Az Intelligent Software cég a német bővítést olyan lehetőséggel szerelte fel, amit a számítógép alapállapotban sosem használ, de ezzel a speciális funkcióhívással mi ezt megtehetjük. A dolog lényege, hogy a felhasználó két billentyűt különleges SHIFT billentyűként (<XT1> és <XT2>)definiálhat. Ezek a SHIFT-ek a PAUSE billentyűhöz hasonlóan nem adnak vissza kódot, inkább várnak egy következő billentyű megnyomásáig. Ekkor sem következik közvetlen kódvisszaadás, hanem a másodiknak megnyomott billentyű kódját a KBD_ISR rutin keresni kezdi egy a felhasználó által definiált táblázatban. Ha megtalálta, az eredeti billentyűkódra következő bájt tartalmát adja vissza kódként.

Egy táblaelem tehát két bájtból áll. Az első bájtok között keresi a eredeti billentyűkódot, és a neki megfelelő második bájtot adja vissza kódként. A táblázat végét jelzi, ha az egyik első bájt értéke 0.

A rendszer azt a billentyűt tekinti XT1 SHIFT-nek, melynek a dekódolási táblázatban található kódja 0C1H (193), és azt XT2-nek, amelyiknek belső kódja 0C2H (194). Mivel az EXOS önmagában nem definiál ilyen SHIFT-eket, nincs is olyan billentyűkombináció, amelynek ilyen kódjai lennének. Ezenkívül a KBD_XT1_ADDR és a KBD_XT2_ADDR rendszerváltozók értéke is nulla. Ezek a változók tartalmazzák ugyanis a kiterjesztő SHIFT-ek táblázatainak kezdőcímeit. A KBD_XT1_ADDR értéke jelöli a táblázatot, ha az XT1 SHIFT-et nyomtuk meg, illetve a KBD_XT2_ADDR használódik fel XT2 nyomásakor.

CAPS-LOCK üzemmódban a betűbillentyűk megnyomása nagybetűt szolgáltat. Vagyis a kódot a SHIFT-es dekódolási táblázatból veszi a rutin, a többi billentyűnél pedig a normál táblázatból. A KBD_ISR rutin azonban csak a 61H—7AH (97—122, a———z) tartományba eső kódokat tekinti betűhöz tartozó kódoknak. Ugyanakkor német üzemmódban nagybetűs mód esetén az Ö billentyű megnyomására mégiscsak Ö betű jelenik meg, vagyis betűként értelmezte. Ez pedig úgy lehet, hogy ha a kód nem szabványos angol betűnek felel meg, a rutin megvizsgálja az SM_PL_LET és a BG_PL_LET rendszerváltozók értékét, és ha a kód az általuk kijelölt intervallumban van, akkor a rendszer ezt is betűként kezeli.

A változók tartalma is beleértendő az intervallumba. Így a német módban az SM_PL_LET értéke — amely az intervallum alsó határa — 7BH (123, ä), a BG_PL_LET-é pedig — a felső határt jelöli ki — 7DH (125, ü). E változók használatával a betűkészletnek magyar karaktereinket is szerves részévé tehetjük.

A legutolsó két bájt azt a táblázatot jelöli ki, amelyik tartalmazza azon karakterek mátrixát, melyeket a törölt karakterkészlethez képest kívánunk megváltoztatni. A karakterkészletet törölni BASIC-ből a CLEAR FONT utasítással, gépi kódból a §§FONT speciális funkcióval lehet, de ez nem azonos az angol mód karakterkészletével! A karakterek törlését a rutin végzi, ezzel nem is kell törődnünk. Itt egy táblaelem tíz bájtból áll. Az első bájt a karakterkód, a többi kilenc a karakter bitmátrixát adja, úgy, ahogy például BASIC-ben is SET CHARACTER utasí-

tással definiálni kell. A táblázat végét az jelzi, ha a karakterkód ØFFH (255) értékű. Ha a §§KEYB speciális funkciót nem nulla értékű C regiszterrel hívjuk meg, a rutin csak ezt az utolsó funkciót látja el, a DE regiszterpárnak erre a táblázatra kell mutatnia.

E speciális funkció tehát mindazt szimulálja, amit az EXOS elvégez, ha német üzemmódra térünk át. Igy például tökéletes magyar üzemmódot kreálhatunk. Ennek kapcsán van azonban egy probléma: a §§KEYB ugyanis WP_MD_FL_ nevű belső billentyűzet-rendszerváltozóba 2 értéket ír. De ha e bájt értéke nem nulla, akkor a szövegszerkesztőt az EXOS nem indítja be. Ennek kiküszöböléséhez egy rövid gépi kódú programra van szükség. Az alábbi lista ezt tartalmazza. Ha az USR_KBD_DAT címkének értéket adunk, és ott elhelyezzük a megfelelő 16 bájtot, akkor ez egy működő programrészlet. Vigyázzunk, az adatoknak a funkcióhíváskor a nullás vagy az első lapon van a helyük, mert az EXOS a 2., illetve a 3. lapra más szegmenset lapoz be esetleg, és így a cím nem az adatainkra mutat. A program a BASIC-ben érvényes billentyűzetcsatornát használja. Előfordulhat, hogy ezt meg kell változtatni, ha a saját billentyűzetcsatornánkon van, vagy a szövegszerkesztőből bővítésként hívjuk meg a programot. A szövegszerkesztő billentyűzetcsatornájának száma ØC8H=2ØØ.

```
USR_KBD_DAT   EQU NNNN ; a 16 bájtos adat címe
KBD_CH        EQU 69H ; a bill.-csatorna
SPEC_FUNC     EQU ØBH ; a speciális funkció
                      ; EXOS-rutin kódja
§§DPTR        EQU ØCEH
§§KEYB        EQU ØCFH
WP_MD_FL      EQU Ø1A4H ; a rendszerváltozó relativ címe a német billentyűzet-
                        eszköz munkaterületén
CHNG_KBD:     ; a billentyűzet felhasználói módba való átalakítása
3E 69         LD A,KBD_CH
Ø6 CF         LD B,§§KEYB
ØE ØØ         LD C,ØØ ;teljes átdefiniálás
11 NN         LD DE,USR_KBD_DAT
NN
F7 ØB         EXOS SPEC_FUNC
CHNG_FLAG:    ; az átállítandó változó tényleges címének kiszámítása és törlése
3E 69         LD A,KBD_CH ; DE = bővítő munkaterület címe
Ø6 CE         LD B,§§DPTR ;
F7 Ø6         EXOS SPEC_FUNC
21 FB FF      LD HL,ØFFFBH ; HL = DE — 5
19            ADD HL,DE
5E            LD E,(HL); DE = a billentyűzet munkaterület címe
23            INC HL;
56            LD D,(HL)
21 A4 Ø1      LD HL,WP_MD_FL ; a vált. relatív címe
AF            XOR A ; A regiszter törlése
19            ADD HL,DE ; a tényleges cím
77            LD (HL),A ; a változó törlése
```

GOTO parancs módban

Az Enterprise 128 számítógép rendkívül nagy szókincsű és sokoldalú BASIC-jének mégis akad néhány hiányossága, amelyek elsősorban azoknak tűnnek fel, akik más személyi számítógépek nyelvjárásaihoz szoktak — például a ZX-Spectrum igencsak elkényeztette a felhasználókat bizonyos területeken.

Az egyik legzavaróbb — és kicsit érthetetlen — sajátosság, hogy a gép a GOTO utasítást csak programfutás során hajtja végre. Ha parancs módban próbálunk beírni például egy GOTO 1ØØØ utasítást, a számítógép hibaüzenettel leáll: „Programm-Befehl im Direktmodus", illetve „Statement in immediate mode". Ennek közvetlen oka természetesen megadható az interpreter ismerete alapján (a GOTO kulcsszóhoz tartozó és vele együtt a ROM-ba beégetett jelzőbájtban aktív a „direkt módban nem alkalmazható" üzenetet hordozó bit), de a tiltásra valódi magyarázatot nemigen találunk.

Az tudja csak igazán, milyen boszszantó is ez a hiányosság, aki már vesztett el — esetleg hosszú programfutás során kiszámított — adatokat egy szintaktikai hiba vagy éppen egy átgondolatlanul beírt STOP utasítás miatt. A tárolt adatok leállás után még elérhetők, de gyakorlatilag lehetetlen a programot tovább futtatni, illetve egy másik — például az adatokat kiíró vagy háttértárra mentő — programrészletet elindítani adatvesztés nélkül.

Különösen nagy szükség van a program bármely részének adattörlés nélküli indíthatóságára a nagy adattömegekkel dolgozó, hosszadalmas mátrixműveleteket készítő matematikai statisztikai programoknál vagy a számítógép méréstechnikai alkalmazásánál. Itt az adatvesztés következménye esetleg nehezen reprodukálható mérési eredmények elvesztése.

A probléma részleges megoldását jelentheti, ha a programban előrelátóan DEF-fel definiált modulokat helyezünk el, amelyek elvégzik a szükséges feladatokat, például az adatmentést. Az interpreter parancs módban is elfogadja, és adattörlés nélkül végre is hajtja a CALL utasítást. Természetesen így nem ugorhatunk a program tetszőleges sorára. A valódi megoldást az jelenti, ha megvalósítjuk a parancs módban is kérhető GOTO n műveletet. Erre mutat példát az alábbi gépi kódú rutin:

```
LD A, Ø5
OUT (B3), A; belapozza a BASIC szegmensét
CALL C3FD; megkeresi a HL sorszámú BASIC sort
JP CF9B; elugrik a végrehajtásra
```

A rutint betöltő sorokat célszerű a BASIC programunk legelején elhelyezni:

```
1ØØ ALLOCATE 1Ø
11Ø CODE GOTO= HEX$("3E,Ø5,
    D3,B3,CD,FD,C3,C3,9B,CF")
```

A program futtatása után a GOTO nevű gépi rutin akár parancs módban, akár programban hívható a CALL USR(GOTO,n) utasítással, ahol n helyére a célsorszámot kell írnunk. Az utasítás hatására az interpreter az n. sortól hajtja végre a tárolt BASIC programot, a változók törlése nélkül. Ha nincs a megadott számú sor, akkor a végrehajtás a következő létező sorral indul.

Külön előny, hogy sorszámként nemcsak számkonstanst adhatunk meg, hanem változót, sőt tetszőleges aritmetikai kifejezést is, amit azután az interpreter a CALL végrehajtása során kiértékel. Ez még programban is — a „beépített" GOTO helyett — különösen praktikussá teheti a rutin alkalmazását.

Felföldi József

Enterprise és Datacoop CDC PRT—42G

Magyarosan...

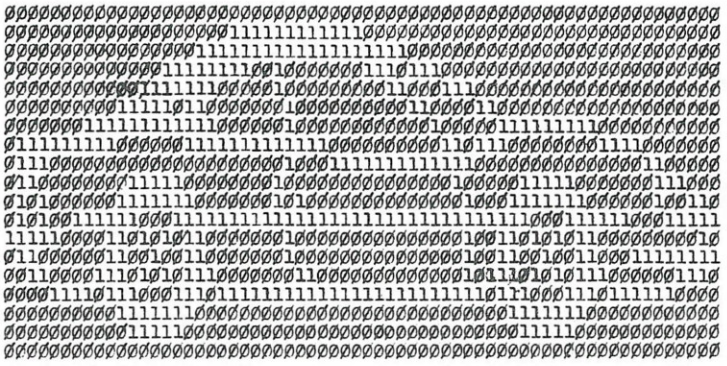
A program német billentyűzetű Enterprise géphez és az elterjedt Datacoop CDC PRT—42G nyomtatóhoz készült. Ha printert nem használunk, a NYOMTATÓ rutint nem kell begépelni. Futtatás után az összes magyar ékezetes betű megjeleníthető és nyomtatható. A billentyűzet az írógépekéhez hasonló elrendezésű. A legfontosabb nyomtatófunkciók tetszés szerint választhatók.

A megváltozott karakterek listája:

é = <ö>	É = <SHIFT> + <ö>
á = <Á>	Á = <SHIFT> + <Á>
ü = változatlan	Ü = változatlan
ö = <'>	Ö = <SHIFT> + <'>
' = <ß>	@ = <SHIFT> + <3>
ú = <ALT> + <U>	ú = <ALT> + <Y>
í = <ALT> + <I>	í = <ALT> + <X>
ó = <ALT> + <O>	Ó = <ALT> + <C>
ű = <ALT> + <Ü>	Ű = <ALT> + <V>
ő = <ALT> + <'>	Ő = <ALT> + <B>

Bozai Gábor

```
  100 PROGRAM "TELJES_MAGYAR_ABC"
  110 ! ENTERPRISE & DATACOOP DCD PRT-42G
  120 ! Bozai Gábor * 1988
  130 !
  140 ALLOCATE 227+78 ! 227+78 byte
  150 LET A$="" ! globális változó lesz
  160 CLEAR SCREEN
  170 EXT "BRD"
  180 CODE KAR=HEX$("3E,66,01,D8,00,11,E6,
12,F7,08,C9,1B,4B,27,66,00,3C,66,66,66,3C,
00,00,1B,4B,40,3C,66,6E,6E,6E,60,3E,00,00,
1B,4B,5B,0C,18,3C,66,7E,66,66,00,00,1B,4B,
5C,1C,7E,60,7C,60,60,7E,00,00,1B,4B,60,66,
3C,66,66,66,66,3C,00,00,1B,4B,7B,1C")
  190 CODE =HEX$("00,3C,06,3E,66,3E,00,00,
1B,4B,7C,1C,00,3C,66,7E,60,3C,00,00,1B,4B,
7E,18,18,18,00,00,00,00,00,00,1B,4B,80,77,
00,66,66,66,66,3C,00,00,1B,4B,82,77,3C,66,
66,66,66,3C,00,00,1B,4B,83,0E,3C,66,66,66,
66,3C,00,00,1B,4B,89,1C,00,38,18,18")
  200 CODE =HEX$("18,3C,00,00,1B,4B,8F,1C,
00,3C,66,66,66,3C,00,00,1B,4B,95,1C,00,66,
66,66,66,3C,00,00,1B,4B,96,77,42,66,66,66,
66,3C,00,00,1B,4B,98,1C,00,3C,18,18,18,3C,
00,00,1B,4B,99,1C,42,66,66,66,66,3C,00,00,
1B,4B,9E,77,00,3C,66,66,66,3C,00,00")
  210 CALL USR(KAR,0) ! ékezetes karakt.
  220 PRINT AT 2,1:"Nyomtatni is akar (i/n
) ? ";
  230 CALL GET
  240 IF A$="I" THEN CALL NYOMTATO
  250 PRINT :PRINT "A BASIC-böl a szövegsz
erkesztöbe (i/n) ? ";
  260 CALL GET
  270 IF A$="I" THEN EXT "WP"
  280 END
  290 !
  300 DEF GET ! karakter input
  310   DO
  320     GET A$
  330     LET A$=UCASE$(A$)
  340   LOOP UNTIL A$="I" OR A$="N"
  350   IF A$="I" THEN PRINT "igen"
  360   IF A$="N" THEN PRINT "nem"
  370 END DEF
  380 DEF NYOMTATO
  390   PRINT AT 2,1:"Kapcsolja be a nyomt
atót!
  400   LPRINT CHR$(17); ! on-line
  410   PRINT AT 2,1:"Van papir a nyomtató
ban (i) ? ";
  420   DO
  430     DO
  440       GET A$
  450     LOOP WHILE A$=""
  460     IF UCASE$(A$)="I" THEN EXIT DO
  470     PING
  480   LOOP
  490   LPRINT CHR$(17); ! on-line
  500   PRINT "igen"
  510   LPRINT CHR$(27);CHR$(64); ! alaphe
lyzetbe
  520   CALL OPCIO("80 karakter/sor","W",0
)
  530   IF A$="N" THEN CALL OPCIO("Vizsz.
megerösitett","E",-1)
  540   CALL OPCIO("Dölt betüs irás","4",-
1)
  550   CALL OPCIO("Arányos betüközök","p"
,1)
  560   CODE PR=HEX$("3E,68,01,3F,00,11,C9
,13,F7,08,C9,1B,72,27,27,9F,1B,72,5B,5D,60
,7B,7D,1B,72,60,60,7C,1B,72,7B,7E,7E,7F,AF
,27,1B,72,80,80,A1,1B,72,82,83,A8,A5,1B,72
,89,89,92,1B,72,8F,8F,97,1B,72,95,96,94,BF
,1B,72,98,99,8B,B3,1B,72,9E,9E,93")
  570   CALL USR(PR,0) ! nyomtató kódtábla
zat módositása
  580 END DEF
  590 DEF OPCIO(X$,Y$,Z)
  600   PRINT :PRINT X$;" (i/n) ? ";
  610   CALL GET
  620   IF A$="I" THEN
  630     LPRINT CHR$(27);Y$;
  640     IF Z<>-1 THEN LPRINT CHR$(Z);
  650   END IF
  660 END DEF
```



Alakzattervezés széltében-hosszában

Az Enterprise-on mozgó ábrát akár grafikus, akár szöveges képernyőn a karakterek átalakításával készíthetünk. E segédprogram előnye kettős: meggyorsíthatjuk vele a folyamatot, ugyanis a képernyőn egyszerre látható egy 2×8 karakterből álló alakzat, és használatával elmaradnak a hosszadalmas számítások is.

A program alkalmazása igen egyszerű. A futtatás után a 80 karakteres képernyőn megjelenik nullákkal kitöltve az alakzat képe. Nincs más dolgunk, mint hogy ahová az alakzaton belül pontot akarunk, ott a nullát átírjuk egyesre. A rajzolás után minden sort ENTER-rel elfogadunk, majd újra futtatjuk a programot.

A program második része számításokat végez és a karaktereket el is helyezi a tárban, méghozzá attól a betűtől kezdve, amelyiket a 120-as sorban az idézőjelek közé írunk. Az alakzatot ezt követően kedvünk szerint használhatjuk fel.

Előfordulhat, hogy szükség van 2×8-nál nagyobb, illetve kisebb alakzatra. Ezeket az alábbiak szerint állíthatjuk elő.

Vízszintesen nem növelhetünk, hiszen több karakter már nem fér a 80 karakteres képernyőre. Csökkentésnél viszont a 16. sor nulláit "fogyasztjuk" annyiszor nyolcra, ahány karaktert vízszintesen fel kívánunk használni, majd a 170-es sor ciklusában a 64-et annyira írjuk át, ahány nullát a 16-ban beírtunk.

Ha függőlegesen akarjuk a karaktereket növelni, akkor a 14-es sorban a 27 helyére annyiszor 9-et írunk, ahány karaktert függőlegesen szeretnénk, a 130-as sorban a 2-t annyiszor írjuk át, amennyi betűt függőlegesen szeretnénk.

Az *ábrán* egy Ferrari Dino 206 GT képét láthatjuk abban a formában, ahogyan azt a programba be kell írni.

Stumphauser Tamás

```
   10 !
   11 ! xxx ONGYILKOS PROGRAM xxx
   12 !
   13 TEXT 80
   14 FOR I=10 TO 27
   15   LET K$=STR$(I)(1:3)
   16   PRINT K$;"DATA 00000000000000000
00000000000000000000000000000000000000000
000000"
   17 NEXT
   18 END
   19 !
   20 ! xxx MEGMARADO PROGRAM xxx
   21 !
  100 DIM A(9)
  110 DIM A$(9)
  120 LET B=ORD("A")-1
  130 FOR K=1 TO 2
  140   FOR I=1 1 0 9
  150     READ A$(I)
  160   NEXT
  170   FOR J=1 TO 64 STEP 8
  180     FOR I=1 TO 9
  190       LET A(I)=BIN(VAL(A$(I)(J:J+7
)))
  200     NEXT
  210     LET B=B+1
  220     SET CHARACTER B,A(1),A(2),A(3)
,A(4),A(5),A(6),A(7),A(8),A(9)
  230   NEXT
  240 NEXT
```

Mi a manó?

A Magazin 1988/11. számában Erdei András EXOS 2.1 példaprogramjait közöltük. Racskó Tamás olvasónk — immár rendszeres szerzőnk — néhány lényeges észrevételt tett a listákkal és az EXOS fordítóval kapcsolatban. E tanácsok azoknak is érdekesek lehetnek, akik esetleg elmulasztották megvenni azt a lapszámunkat.

Az EXOS kezelését szemléltető programlista két hibát tartalmaz, bár ez az olvasáskor nem tűnik fel, noha emiatt a program nem futtatható. Az első: rossz a kezdőcím megválasztása, ugyanis a lefordított kód az editor pufferébe kerül, ott pedig a megszakítás rutin megváltoztatja, az ASMON/SIMON a szöveget tömörített formában tárolja, és a változtatások a lefordított kódon is végbemennek.

Azt ugyan megtehetnénk, hogy a Z parancson belül a "Memory offset:" kérdésre például 4000-rel válaszolva, a kód a 4800H címre fordítódik le, innen kimenthetjük szalagra, majd írunk egy BASIC programot, amely a CODE utasítással bevitt kis gépi kódú programon keresztül betölti és futtatja azt. Azonban így a program a BASIC munkaterületre kerül, és itt, ha nem is törvényszerűen, de felülíródhat. Viszont ha az ORG direktíva utáni címet például 2800H-ra írjuk át, és kiküszöböljük a másik hibát, akkor a lefordított kód futni fog az ASMON/SIMON égisze alatt éppúgy, mint a BASIC programból híva.

A másik hiba az, hogy a negyedik lista "p megsemmisítése" szövege előtt nincs pontosvessző, így azt a fordító címkeként és utasításként próbálja értelmezni. Mivel azonban ez nem sikerül, "Fenntartott szó" és "Ismeretlen utasítás" hibaüzenetet ad a képernyőn. Ezeket kiküszöbölve a futtatás eredményeként egy hibaüzenet jelenik meg, amelyet az EXOS leírás nem említ. A 0CFH (207) kódú hibaüzenetről van szó, ez a copyright üzenetet adja vissza:

"Copyright INTELLIGENT SOFTWARE LTD".

Megemlítjük — a példaprogramoktól függetlenül —, hogy az ASMON ismer a 128 játék című könyvben megjelent SIMON direktívákon kívül egyebeket is, például: . ASEG . . CSEG . . INIOFS . . SETRIP . . RESRTP . Ismertetésükre később visszatérünk, addig annyit, hogy ezek segítségével relokálható fájlokat lehet létrehozni.

Hardver

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. 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 programozónak rendelkeznie kell alapfokú áramköri 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.

Töréspont

Ezt a megoldást a legtöbb monitorprogram alkalmazza. A töréspont, vagyis a programfutás adott pontban való felfüggesztése igen sok feltételtől függhet. Milyen feltételek adhatók meg? Megadható az adott cím (címvonalak adott állapota), bizonyos adat (az adott vonalak adott állapota), adott vezérlővonal-kombinációk és az előzőek kombinációi.

Hasonlóan az előző módszerhez, itt is mind hardver-, mind szoftvermegoldás lehetséges.

A hardvermegoldásnál a vonalak elektromos — logikai — állapotát összehasonlítják (digitális komparátor) a monitor által a kimeneti portokra írt értékekkel, amit a töréspont megadásakor töltünk fel. Amikor a portokon tárolt érték és a vonalak állapota megegyezik, a komparátor kimenetén megjelenő jel WAIT-be vezérli a processzort. Ez a megoldás nem lassítja le a program futását.

A szoftvermegoldásnál minden utasítás végrehajtása után lefut egy töréspont-kiértékelő program, ami azt vizsgálja, hogy nem alakult-e ki a törésponti feltétel. Ha nem, akkor a program futása folytatódik, különben visszaadódik a vezérlés a monitornak.

A töréspontot a legegyszerűbben úgy helyezhetjük el, hogy a törésponti címre egy JP, CALL vagy RST utasítást írunk. Utóbbinak a használata a legjobb, mert egybájtos, és csupán a címen levő bájt értékét kell helyettesíteni az RST utasítással. A töréspont törlésekor az eredeti bájtot kell visszaírni. Ez a módszer is lassítja a program futását, és ilyen módon csak RAM-ban futó programot láthatunk el töréspontal. Azért, hogy a programban lévő ciklusokon minél gyorsabban lehessen áthaladni, a töréspont megadásán kívül meg szokták adni a töréspont mélységét is. Ez az a szám, ami azt jelöli, hogy a töréspontra való hányadik ráfutás után álljon le a program.

A hibakeresés fejlettebb eszközei a szoftverszimulátor és a logikai analizátorok. A szoftverszimulátor egy számítógépen futó, általában valamilyen magas szintű nyelven megírt program, ami az eredeti mikroprocesszor minden utasítását szimulálja, figyeli a regiszterek, jelzőbitek, memóriahelyek tartalmát, azok változását. Ez természetesen papírral és ceruzával is megtehető, de nagyon fáradságosan. Egy ilyen szimulátor legfontosabb tulajdonságai:

— töréspont elhelyezése a legkülönfélébb feltételek teljesülésétől függően,

— regiszterek és memória listázása,

— nyomkövető képesség, ami egy regiszter vagy memóriahely tartalmát írja ki, ha azt a program használja,

— a regiszterekbe tetszőleges érték tölthető és onnan folytatható a program futása.

A módszer legnagyobb hátránya a lassúsága, és nem tudja modellezni a program tényleges be/kimeneti részét sem.

A legfejlettebb hardveres hibakereső megoldást a logikai állapotanalizátorok teszik lehetővé. Ezek tulajdonképpen egy sokcsatornás memóriaoszcilloszkópnak tekinthetők, amelynek bemenetein keresztül képesek a processzor vonalainak időbeli változását rögzíteni egy adott időtartományra visszamenőleg. Megadható a figyelés indításának vagy befejezésének a feltétele, ami azt jelenti, hogy a feltétel teljesülése utáni vagy előtti néhány száz utasításciklus az analizátor memóriájába kerül, ahonnan kiolvasható, megjeleníthető, analizálható.

A korszerű fejlesztések elképzelhetetlenek az ilyen — sajnos igen drága — eszköz nélkül.

A programok tesztelése

A programok tesztelése és a hibakeresés nagyon szoros kapcsolatban van egymással. A tesztelés folyamán olyan adatsorozatokkal vizsgáljuk meg a program helyes működését, ami magában foglalja a triviális, a speciális és az egyszerű (papírral, ceruzával is meghatározható) esetek vizsgálatát.

Az assembler programok teszteléséhez — az esetleges teszt-adatsorozatokon túl — ugyanazokat a szoftver- és hardvereszközöket használjuk fel, mint a hibakeresésnél. Mivel nem mindegyikről esett még szó, röviden felsoroljuk ezeket.

— A be/kimeneti szimulátor. A be/kimeneti eszközök — nyomtató, soros vonal, digitális be/kimenetek — széles skáláját képes működési szempontból helyettesíteni.

— Az in-circuit (ejtsd: inszörkit) emulátor. A kivezetéseit a CPU helyére dugva képes a CPU működését elvégezni (emulálni).

— A ROM szimulátor. A rendszerben lévő ROM helyére dugva tartalmuk módosítható, átírható.

— A tesztadat-generátorok. Véletlen adatsorozatokat képesek generálni a program számára.

Dokumentálás

A programozók többsége igazán nem tekinti fontosnak saját munkájának dokumentálását. Az igaz, hogy ez védelmet is nyújthat az illetéktelen kíváncsiskodók ellen, de sajnos munkastílussá is válhat, és idővel ez a programozó saját és kollégáinak a munkáját is nehezíti.

A programfejlesztés egyik legfon-

tosabb része a pontos, a programot jól leíró dokumentáció. A dokumentációnak a programfejlesztés egész folyamatát végig kell kísérnie, a programhasználatot és a program továbbfejlesztését is támogatnia kell. Egy rosszul dokumentált programot nagyon nehéz karbantartani, használni vagy továbbfejleszteni. Előfordulhat, hogy egy program mérete túl nagy lesz, vagy a program nagyon lassan fut. A tervezőnek ilyenkor a programot tovább kell fejlesztenie. Ezt hívjuk a program újratervezésének.

Azért, hogy egy program jól dokumentált legyen, néhány tanácsot érdemes megszívlelni.

— A program világos, egyszerű felépítésű legyen, minél kevesebb vezérlésátadást (ugrást) tartalmazzon.

— Olyan azonosítókat, címkéket és neveket használjunk, amelyek jól tükrözik a funkciójukat.

— Kerüljük a betűszavakat, mert nem mindenkinek nyilvánvalóak a rövidítések (például SBR=soros bemeneti rutin).

— Ha lehet, teljes szavakat használjunk, ne csak az elejüket, és az elnevezések ne hasonlítsanak egymáshoz, mert összetéveszthetők.

Az assembler programokban szereplő megjegyzésekkel kapcsolatban pedig a következőkre figyeljünk:

— Ne ismételjük meg az utasítás jelentését, például INC B; B növelése eggyel.

— A program minden fontos pontjához — legjobb mindegyik sorhoz — írjunk megjegyzés sort.

— A nyilvánvaló dolgokat ne kommentezzük. Például az INC HL vagy DJNZ ODA programsorok nem igényelnek magyarázatot, hacsak valami speciális jelentésük nincs.

— A megjegyzések mindig az aktuális programhoz tartozzanak; ha a program változik, akkor a megjegyzésnek ezt követnie kell.

— A program fejrészében átfogó magyarázatot kell adni a programról, mert enélkül csupán a soroknál szereplő megjegyzések nem mindig igazítanak el.

— A program egy-egy önállónak tekinthető része előtt írjuk le annak funkcióját és a lényegesnek tartott megjegyzéseket is.

— Követendő az a gyakorlat, hogy a program elején a program íróját, az aktuális verzió írásának időpontját is feltüntetik.

Bár az itt leírtak mennyisége nem tükrözi eléggé, kifejezetten szeretnénk hangsúlyozni a jó dokumentáció fontosságát, mert vele időt, fáradságot és munkát takarítunk meg. Különösen, ha egy fél év múlva a programot valamiért elővesszük . . .

A fejlesztőrendszer

Egy mikroprocesszort tartalmazó rendszer fejlesztése sprciális „szerszámot" is igényel: a fejlesztőrendszert.

Mi is ez? Nem más, mint egy számítógép, ami lehetővé teszi a hardvert működtető program megírását és szerkesztését assemblerben (ez az editor), az assembly program lefordítását gépi kódra (ez az assembler), a lefordított program ellenőrzését (ez a monitor), a lefordított program átvitelét a megépített hardverbe (ez az EPROM-égető).

Ezek a fejlesztőrendszer legfontosabb elemei. Fejlesztőrendszer a legtöbb személyi számítógépből kialakítható. Az imént felsorolt programok általában nem okoznak gondot, csupán egy EPROM-égetőt kell a számítógéphez készíteni. A professzionális fejlesztőrendszereknek még számos egyéb szolgáltatásuk van. Ezeket az előzőekben, a rendszerfejlesztéssel kapcsolatban már bemutattuk.

Az editorral megírt assembly nyelvű programot az assemblerrel gépi kóddá lefordítjuk, majd az így lefordított program működését a monitorprogrammal ellenőrizzük. Itt kezdődnek a nehézségek. A monitorral a programnak csak azok a részei ellenőrizhetők, amelyek nem kapcsolódnak a külvilághoz (nem használnak ki/bemeneti kapukra vonatkozó utasításokat). Sajnos a legtöbbször a programok igen intenzíven használják a ki/bemeneti egységeket. A lefordított és részben kipróbált programot ezek után át kell vinni a hardverünkbe, és a továbbiakban ott kell pórbára tenni. Mivel a program a rendszer EPROM, illetve RAM memóriáját használja, az átvitel a következőképpen lehetséges.

A gépi kódú, tehát lefordított programot az EPROM-égetővel EPROM-ba írjuk be, majd ezt, a programot tartalmazó EPROM-ot a rendszerünkben lévő EPROM-foglalatba dugjuk. Mivel a mikroprocesszor bekapcsoláskor, illetve RESET jel kiadásakor a 0. címtől kezdi az ott lévő utasítások végrehajtását, a rendszerekben a nullás címtől kezdődik az EPROM címtartománya. Az EPROM behelyezése után megkísérelhetjük a bekapcsolást, amivel programunkat indítjuk.

Sajnos itt van a másik nehézség: azonkívül, hogy a működtető program hatásait megfigyeljük, nincs más lehetőségünk a fennálló hibák megkeresésére.

Ha megtaláltuk a hibát, ezt a programunkban kijavítjuk, újra EPROM-ba égetjük, ismét rendszerünkbe dugjuk, majd kipróbáljuk. Ezt az ismétlődő iteratív tevékenységet addig folytatjuk, míg rendszerünk hibátlanul nem működik.

A fejlesztési folyamat blokkvázlatát 1989/4. számunk ábráján mutattuk be. Az ábrával kapcsolatban feltétlenül meg kell említeni, hogy a rendszerfejlesztés nem mindig ilyen szép, egymás utáni lépésekből áll, hanem elképzelhető — és a gyakorlat ezt sokszorosan igazolta —, hogy a fejlesztés bizonyos fázisaiban olyan tények, korlátozások merülnek fel, amelyek és az elengedhetetlen korrigálások miatt a fejlesztés korábbi fázisaiba vissza kell lépni. Ezért a rendszerfejlesztés iteratív tevékenység.

Hogy a rendszer létrehozásának folyamatát valahogy lerövidítsük és egyáltalán sikeressé tegyük, ismét csak azt tanácsolhatjuk, hogy

— első lépésként — akár önálló kis programokkal — feltétlenül győződjünk meg hardverünk helyes működéséről;

— ha a programban meg akarunk változtatni részeket, ezt mindig csak egymás után, részenként tegyük, és mindig próbáljuk ki a keltett hatást. Ha egyszerre több dolgot változtatunk meg, és hiba fordul elő, nem tudjuk behatárolni, hogy melyik változtatás okozta;

— ha van rá lehetőségünk, építsünk be a fejlesztendő rendszerbe diagnosztikai funkciókat, amikkel az egyes részek hibátlan működéséről meggyőződhetünk.

A következőkben konkrét, olcsó és könnyen megvalósítható megoldásként egy már elkészített, Z80 alapú fejlesztőrendszert ismertetünk, és bemutatjuk a rendszerhez illeszkedő, általános célokra használható, Z80 alapú, egykártyás mikroszámítógépet.

Dr. Kónya László

BÖRZE

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

COBRA laser COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1097 Budapest IX., Illatos út 7. Telefon: 476-169/388

SZÁMLAKÉSZÍTÉSTŐL A KÖNYVELÉSIG

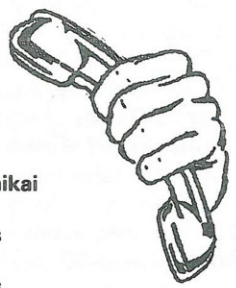
COBRACONTO

ügyviteli programrendszer moduljai:

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

KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

MIKROPO BUDAPEST



- Komplex számítástechnikai szolgáltatás!
- Szünetmentes áramforrás
- Új árkatalógus

Iroda: Bp. VI.,
Nagymező u. 51.
Tel.: 325-768
Telex: 22-7842
Telefax: 124-431

ASY ELEKTRONIKA

A legújabb ajánlatunkból:
UNIX alapú ASY—16 szupermikro számítógépek
- IBM PC/XT kompatibilis számítógépek
- IBM PC/AT kompatibilis számítógépek
- különleges igényeket kielégítő billentyűzetek
- különleges terminálok (pl: VT—52, Siemens 8160)

ASY Kereskedelmi Iroda
Telefon: 415-166
Telex: 22-4378

PERIFÉRIA

Elektronikai Fejlesztő és Szolgáltató Kisszövetkezet
Bp. VII., Peterdy u. 30.
Telefon: 213-588

AJÁNLATA:

- P—XT: 140 E Ft-tól + áfa
- P—AT: 200 E Ft-tól + áfa

igény szerinti konfigurációk

- FX—1000 printer: 75 E Ft + áfa
- 40 MB-os WINCHESTER: 86 E Ft + áfa

procontrol pc

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 — 86 XT 12 MHz	98 000 Ft-tól
TURBO XT gép	69 000 Ft-tól
LAPTOP hordozható LCD XT gép + modem + akku	149 000 Ft
12" zöld monochrom monitor TUV M31	4 900 Ft
5 1/4" DS DD Floppy lemez	79 Ft

és még 2000 árucikkről árjegyzék, azonnali kiszolgálás.

BROTHER AX 15 típusú,
margarétafejes,
elektronikus írógép

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

MIGERT

ECONORG

SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

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

Szervezési szolgáltatások
Számítástechnikai rendszerek fejlesztése
Szerviz, kereskedelmi forgalmazás
Oktatás (vállalati, gazdasági és szervezési tanácsadó szolgálat)
Szellemi export
Az ACER teljes gépcsaládja

A hagyomány él

Évről évre kiemelkedően fontos eseménye a magyarországi számítástechnikai életnek a mikroszámítógépek szerelmeseinek tavaszi találkozója a BNV területén.

Az idén sem volt ez másként, bár a kiállítói és látogatói létszám — ez utóbbi hasonlóan a párhuzamosan megrendezett Utazás '89 kiállításhoz — némi visszaesést mutatott az előző évekhez képest. Az egy pavilonra, a 25-ösre zsugorodott kiállítási terület adott otthont a már hagyományos SZMSZM (A számítástechnika mindenkié, a számítástechnika mindenkiért) és az első ízben megrendezett Tourcomp '89 kiállításnak, és a találkozóhoz kapcsolódó társadalmi események zömének. A másik tavalyi helyszínen, a 24-es pavilonban most bort mértek . . .

Hogy mégis sikeres rendezvényként lehet elkönyvelni a µ '89-et, annak oka, hogy mindenki, aki eljött a kiállításra, alaposan kitett magáért, színvonalas kiállítások, szolgáltatások fogadták a látogatókat. Elmaradt ugyan egy-két előzetesen beharangozott program, de ezeket spontán kezdeményezések pótolták, állandósítva például a külön programként jegyzett szoftverbörzét.

Néhány sorban

Igen sok jó rendszert mutattak be a Tourcomp '89 (Számítástechnika az idegenforgalomban) kiállítás résztvevői, az International House Budapesttől, az Adatrend Kisszövetkezeten és Számalkon keresztül egészen az Európa Szálló és a Hardware-Software Gmk közös kiállításáig, s lehetne még folytatni a sort.

Az SZMSZM kiállítás ez alkalommal talán kissé színtelenebb volt a korábban megszokottnál, de így is találkozhattunk néhány igen figyelemreméltó bemutatóval. Nagy sikerük volt a hagyományos vásári attrakcióknak, a megszokott könyv- és szoftvervásároknak.

Sok érdeklődőt vonzott az Informatika és a számítástechnika a közművelődésben című állandó bemutató, a hasonló címet viselő megbeszélésen pedig fontos bejelentésre is sor került. Németh Lajos (Szolnok) felvetette, hogy a Népművelők Egyesületén belül alakuljon meg a számítástechnikával foglalkozók szakmai szervezete. A résztvevők előzetesen egyetértettek a szervezet létrehozásával, a Számalk-Menü képviseletében megjelent Bíró András pedig azt is kilátásba helyezte, hogy ha ténylegesen is megkezdődik a munka, akkor vállalata szoftverrel és szakmai támogatással járul hozzá a szervezet eredményes működéséhez.

Más. Az érdeklődők napról napra figyelemmel kísérhették, miként bővül a Chess-Base sakk-adatbank (a Budapest Sportcsarnokban folyó verseny játszmái már a következő napon tanulmányozhatók voltak az adatbankban). Jelzi a sakk iránti érdeklődést, hogy az első napi ember—gép partik megismétlésére is jócskán lett volna igény, olyannyira, hogy egy vérmes sakkbarát még komoly fenyegetéseket is kilátásba helyezett, ha nem ülhet asztalhoz egy partira . . . A zenei programok kiemelkedő eseménye volt a Digitális zenetanár bemutatkozása (magáról a programról lapunk 1989/4-es számában már jelent meg ismertetés), s a Bratislavai Rádió elektroakusztikai számítógépes hangversenye is nagy sikert aratott.

Úgy érzem, a találkozó csúcspontja minden szempontból az oktatóprogramok versenye volt, nem csupán a résztvevők minden eddigit meghaladó száma, hanem az alkotások értéke miatt is. Ugyanakkor jelzi a megváltozott szemléletet, hogy az egyik díjnyertes csapat tagjai másnap iskolájuk igazgatójától kérték: ugyan tenne még rá néhány ezer forintot a kiérdemelt pénzjutalomra. Sajnos az oktatóprogram-üzletág — úgy tűnik — mostanság nem éppen a konjunktúra idejét éli, s félő, hogy a sok, rendkívül értékes munka üzleti hasznosítására csak sokára kerülhet sor. Pedig több vezető számítástechnikai intézmény már felismerte — köztük a Softinvest is —, hogy a jövő felhasználóinak megfelelő orientálása nem csupán presztízs-, hanem sokkal inkább jól felfogott üzleti érdek.

A tanuló- és oktatóprogramok országos versenyének eredménye:

Csapatverseny

I. díj: Kilián György Gimnázium, Budapest
Móricz Zsigmond Gimnázium, Budapest

II. díj: Landler Jenő Műszaki Középiskola, Budapest
Pollák Antal Erősáramú Ipari Szakközépiskola, Szentes

III. díj: Ifjúsági Ház Neumann Klub, Szentes
Ifjúsági Ház Mikro Klub I., Szentes

Összesen 16 csapat indult, az első helyezettek 10 000, a másodikok 8000, míg a harmadik helyezettek 6000 forintos pénzjutalmat kaptak a versenyt szponzoráló Softinvest jóvoltából.

Egyéni

I. díj: Bakó László, Orosz igeragozás
Ládonyi Ferenc, Függvényábrázolás
Szabolcsi Imre, Polimerek
Szeghő István, Statisztikák
Fullajtár Pál, Automatika
Gulyás László, PAS
Viczián Gergely, Rezgőmozgás

II. díj: Bakó László, A mesék birodalmában
Kirisits József, Hangolt erősítő
Gonda Zoltán, Szakmai oktatás
Göőz István, Kémia egyenlet
Tímár Ferenc, Jelleggörbe-rajzoló
Sándorfi Gábor, Függvényábrázolók
Budai Zoltán, Hisztogram
Sőtér Zoltán, Nevezetes számok
Vidovics Ferenc, Lejtő
Agócs Zoltán, Angol
Fischer Erik, English

III. díj: Háry András, English games
Visnyovszki Vendel, Eszperantó
Mayer Győző, Olasz
Bedő Attila, Project English
Kovács József, Függvénytrió
Szili Imre, Törtek
Nagy Imre, Ütközések
Papp László, Ábrázolás
Tímár Ferenc, Honvédelmi oktatás
Szabolcsi Szabolcs, Hullámforma
Lugosi Antalné, Történelem
Lőrincz Attila, Minden út Rómába vezet
Csajági Sándor, Atomreaktor
Benkő Dávid, Csillag
Antal József, Fizika
Spányik Balázs, Hammurapi
Gulyás István, Atari-földrajz

A díjak megoszlása ebben a kategóriában 5000 — 3000 — 2000 forint volt.

Az egyszemélyes csapat

Mit tegyen egy családanya, akinek két nagyobb gyermeke már iskolás, és úgy véli, az iskola túl sokat vár el a szülőtől a tanulás terén (rendszeres kikérdezést, leckeellenőrzést, gyakoroltatást stb.)? A Lugosi Antalné által adott válasz sokak számára bizonyára kissé meglepő, de vitathatatlanul szellemes: a család tavaly vásárolt C Plus/4-es gépére elkészített — mintegy 300 munkaóra alatt — egy oktató/feleltető keretrendszert, s azon melegében három alkalmazást is, egyet matematikából, egyet történelemből, egyet pedig kémiából.

Az oktató/feleltető keretrendszerrel szemben támasztott követelmény az volt, hogy többcélú programsorozat legyen; nyújtson segítséget az otthoni tanuláshoz, összefoglaláshoz, a tananyagban való elmélyedéshez, a felzárkóztatáshoz. Ezzel együtt a rendszer alkalmas iskolai használatra is, a tanulók tudásának felméréséhez, helyettesítve vagy kiegészítve az írásbeli felmérőket.

Lugosi Antalné — lévén a gyest rövide-

sen ismét a programtervezői munkával felváltó családanya — úgyszólván véletlenül jutott hozzá az oktatóprogramok pályázati felhívásához, s úgy érezte, két év munkája a nagyobb nyilvánosság előtti bemutatkozásra is érdemes, s kiderülhet az is, milyen irányban kell esetleg tovább lépni, miben kell szaktanárok segítségét kérni. A programcsomag jelenleg 20 képernyőtervet tud, mindegyiknek 5-6 variációja lehetséges. A tantárgyankénti feltöltéshez most már csak az hiányzik, hogy egy-egy kiválasztott témakörből rendelkezésre álljanak azok a kérdések, amelyek az adott program megírásához szükségesek.

A programcsomag mindenesetre csaknem korlátlanul, és minden irányban bővíthető.

„Properity" a szerkesztőségekben

Bemutatkozott a kiállításon a MUOSZ és az SZKI által kifejlesztett szerkesztőségi rendszer. Szauder Mihály, a MUOSZ Elektronizálási Bizottságának vezetője elmondta, hogy a Bálint György Újságíróiskolán három éve kezdték el a számítógépes oktatást, speciálisan azon a területen, amelyet az újságírók hasznosítani tudnak (szövegszerkesztés, tördelés, saját dokumentáció készítése stb.). Tavaly jutott el az együttműködés arra a szintre, hogy hazai hardveralapon, hazai szoftverrel két megyei szerkesztőségben is komplett rendszer jött létre. A komplett rendszer azt jelenti, hogy egészen a nyomás előtti forma elkészítéséig minden munka a szerkesztőségen belül folyik. Először a Szolnoki Néplapnál állították üzembe a Ventura Publisher magyar verzióját, az ugyancsak az SZKI által gyártott Proper gépekre telepítve. A szolnokiakat a Mai Nap követte, és még ebben az évben üzembe áll további három rendszer, Baranya, Hajdú-Bihar és Veszprém megyében.

A folyamatosan kiépülő rendszerek révén teljesülhet a megyei szerkesztőségek régi álma: az MTI közbeiktatása nélkül, akusztikus modemeken keresztül egyik megye a másiknak közvetlenül adhatja át híreit. (Megjegyzés. Szerettünk volna hírt adni arról, hogy rövidesen szerkesztőségünk is belép a HVP áldásait élvezők táborába, de sajnos — anyagiak híján — a mi lapunk továbbra is javarészt hagyományos technológiával készül.)

Az Alcatel standján egy hotel- és egy éttermi számlázórendszert mutattak be az érdeklődőknek, mi ez utóbbi részleteire voltunk kíváncsiak. Kocsis András — aki a szoftvert előállító SDS grazi szoftverházat, illetve annak rövidesen megalakuló magyarországi vegyesvállalatát képviselte — elmondta, hogy a rendszer képes egy tetszőleges, több értékesítőhelyből kiépített éttermi komplexum számlázását elvégezni, igazodva a magyar számlázási előírásokhoz: változtatható például áfa-hozzárendelés az árakhoz.

Szem-szájnak ingere

Bármely értékesítőhelyükön fogyaszt a vendég (például a szálloda sörözőjében, bárjában, éttermében), a rendszer az adott helyszínnek megfelelően kalkulálja az árakat, lehetőséget teremtve a számla tetszőleges fizetési nemben történő kiegyenlítésére: fizetni lehet hitelkártyával, különböző valutanemekben, és vállalatok számára külön hitelszámla is nyitható. A rendszer akkor sem jön zavarba, ha valamely étel- vagy italféleség nem szerepel az étlapon, de a vendég azt kéri. Ilyen esetben főcsoportokból rendeli meg a hozzávalókat, végrehajtva a raktárkészlet megfelelő módosítását is. Ugyancsak elvégzi a rendszer az egyes pincérek, brigádok forgalmának nyilvántartását is, a rendszerhez kapcsolt nyomtató pedig egyedi lapok nyomtatására is alkalmas, így egy szépen előnyomott étlapra pillanatok alatt rávihető az éppen aktuális választék.

A hardvert a Hewlett-Packard gyártja, a központi egység egy HP Micro LX 3000-es gép, 4 megabájtig bővíthető 2 megabájtos központi memóriával, 150 megabájtos winchesterrel (még 150 megabájt járulhat hozzá), ehhez maximum 8 periféria csatlakoztatható. Egy négy munkahelyes rendszer, két nyomtatóval mintegy 400 000 schillingbe kerül, egy terminálnyi bővítés 20 000 schilling. A Hotel nevű helyfoglaló, információs rendszer 300-500 ezer schillingbe kerül, az éttermi szoftver 150 000 schillingért kapható.

Két Hotel rendszer működik már Budapesten, az egyik a Grand Hotel Hungáriában, a másik pedig a Béke Szállóban, és több megrendelése van az SDS-nek az éttermi rendszerre is.

Természetesen ebben a cikkben minden témát nem érintettünk, de összegzésként — úgy vélem — elmondhatjuk, hogy a μ '89 híven elődeihez, ismételten hozzájárult a számítástechnika társadalmasításához. A hazai számítástechnikai rendezvények közül a Mikro az, amely ezt a társadalmi töltetet a legteljesebb mértékben vállalja, még abban a helyzetben is, amikor a mai Magyarországon a társadalmi akciók finanszírozására egyre kevesebb pénz jut. A μ-találkozók jövőjének biztosítéka lehet, hogy talán mind többen ismerik fel a hosszú távú befektetések megtérülésének elvitathatatlan jelentőségét.

Varga János

Beérett a gyümölcs

1. kép

Idén hatodszor hirdette meg a szekszárdi Garay Gimnázium, az NJSZT és lapunk szerkesztősége immár évenkénti pályázatát. A rövid határidő — mindössze másfél hónap — ellenére 42 pályamű érkezett a szerkesztőségbe. A legtöbben Commodore Plus/4-re, Spectrumra, C64-re, Atarira írták meg programjukat. Nem szerepelt viszont HT-re és Videoton gépre írt alkotás.

Előzsűrizés után a döntőbe oktató- és játékkategóriában egyaránt tíz-tíz program jutott be, melyek alkotói március 13—14-én Szekszárdon, a Garay János Gimnáziumban mutatták be műveiket a közönség szórakoztatására és a zsűri munkájának megnehezítésére. Ugyanis az előző években mindkét kategóriában eléggé szórt volt a mezőny. Az idén azonban nagy volt a tömörülés, és nemcsak az élbolyban! Az elérhető 100 pont ellenére az oktatóprogramok első és hatodik helyezettje között mindössze 7 pont, a játékkategória ugyanezen helyezettjei között pedig 10 pont volt a különbség!

A kialakult holtversenyek mellett újdonság volt az idén ifj. Gulyás László szereplése, aki négy önálló programjával is döntőbe jutott, egy döntős programban pedig társszerzőként szerepelt. A zsűri tagjai előtt első nap az oktatóprogramok szerzői mutatták be alkotásaikat. E kategóriában első díjat nyert ifj. Gulyás László Szentesről C Plus/4-re írt Profi Assembler programjával, mely a zsűri egyöntetű véleménye szerint mindent tud, amit kell, sőt még annál is többet. Jutalma a Neumann János Számítógéptudományi Társaság 6000 Ft-os vásárlási utalványa, amit a zsűri elnöke, Kovács Győző adott át, elismerve egyben a többi döntős oktatóprogramjainak magas színvonalát is.

Második helyezést ért el Eső Péter Budapestről, Spectrumra írt, az emberi tápcsatorna működését bemutató programjával, melyet az NJSZT 4000 Ft-os vásárlási utalványával jutalmazott a zsűri. A harmadik helyezett Csókás Csaba Tamásiból, C Plus/4-es Függvényábrázolás című munkájával. Programja tetszőleges függvényt ábrázol, a kép bármely részére ablak rajzolható és ez nagyítható a képernyő teljes méretére. Az eljárás több lépésben ismételhető. Díja az NJSZT 2000 Ft-os vásárlási utalványa, valamint a Tolna Megyei Tanács 2000 Ft-os különdíja.

A különdíjasok közül kiemelkedik Vidovics Ferenc Szentesről C Plus/4-re írt lejtőprogramja, mely az egyenletesen változó mozgás tanításához nyújt segítséget. Figyelemre méltó, hogy a program mellé mintegy 30 Ft-ból némi barkácsolással előállította a lejtőt is. Az egyméteres útszakaszon 17 helyen méri a gép az adatokat. A mérési eredményeket táblázatban, grafikusan ábrázolja, az eredmények nagyon szemléletesek és nyomtathatók is. Munkájával a Tolna Megyei Tanács 3000 Ft-os, mellékelt dokumentációjával a KISZ KB KSZT 1000 Ft-os különdíját érdemelte ki. A Garay Gimnázium KISZ-szervezete által felajánlott 1000 Ft-os közönségdíjat ebben a kategóriában ifj. Gulyás István és ifj. Gulyás László Atari monitorprogramja kapta. A két fiú Szentesről érkezett, jó barátok, de csak névrokonok. Programjuk a Commodore-hívők táborát csalogatja az Atari gépek elé, hisz a monitor működése nagyon hasonlít a C Plus/4 beépített monitoráéhoz.

A játékkategória versenye izgalmakban bővelkedett nemcsak a közönség körében, de a zsűri asztalánál is. Elsődíjas és egyben a versenyek nagy öregje, elsőéves egyetemi hallgató, hisz ötödször szerepelt a döntőben: Rátkai István (1. kép) Esztergomból. C64-re írta a Bosszú című kalandjátékát, melynek mintegy százötven, képekkel illusztrált helyszínén középkori japán környezetben akár hetekig bolyonghat a játékos. A program több száz szót ismer, sőt megérti azok ragozott alakját is. Nehéz helyzetben, vagy más elfoglaltság miatt az állás háttértárra menthető, később onnan visszatölthető és úgy folytatható akár heteken át, míg végre teljesítheti valaki a küldetést. Munkájáért a Novotrade RT 6000 Ft-os vásárlási utalványát, valamint az iskolai KISZ-szervezet 1000 Ft-os közönségdíját kapta.

Megszokott második díjas lett Jurkovics Károly Budapestről, aki C Plus/4-re írta és alakította át a közismert Impossible Mission játékot, végig gépi kódban dolgozva, a gép szűkös (60 k-s) memóriájába zsúfolva be azt. A mellékelt játékleírás is olyan, amilyen eddig csak a zsűri tagjainak álmaiban fordult elő. Megérdemelten vihette haza a Novotrade RT és a KISZ KB KSZT 2000-2000 Ft-os díjait.

Holtversenyük másik résztvevője, Spányik Balázs ugyancsak Budapestről érkezett C64-re írt programjával. A játékosnak bölcs, előrelátó uralkodói tulajdonságokkal Hammurapi uralkodásának nehéz körülményeit kell bemutatnia. 4000 Ft-os jutalmát ugyancsak a Novotrade Rt.-től kapta.

A Tolna megyei NJSZT 3000 Ft-os különdíját ifj. Schaffer András, Paksról érkezett versenyző C Plus/4-re készített, Harc az életért című programja nyerte. A program a közismert életjáték egy szellemesen módosított változata, ugyanis minden generációváltás után a játékos is és a program is elhelyezhet egy-egy új sejtet a 25-ször 23-as mezőben, s csak ezt követi az újabb generációváltás. A program nem lebecsülendő ellenfél; kedvezőtlen helyzetében úgy helyezi el saját új sejtjét, hogy élő sejtjeinek számát a lehető legtöbbel növelje, ellenkező esetben viszont kíméletlenül támad. Minden generációváltáskor úgy rakja le új sejtjét, hogy a lehető legtöbbet pusztítson el a játékos sejtjei közül.

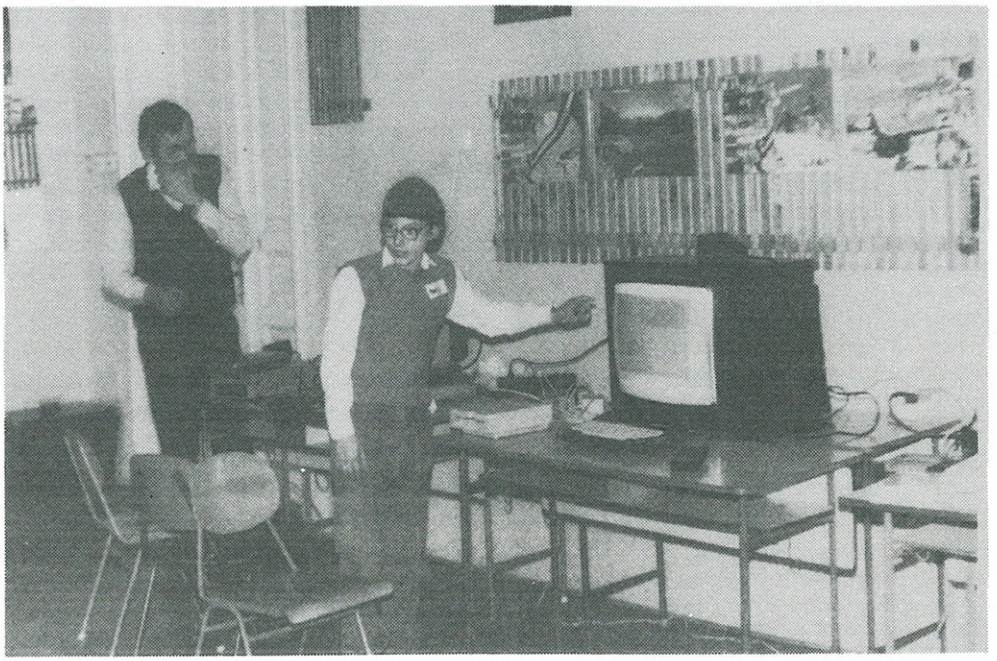
A KISZ KB KSZT 2000 Ft-os különdíját a Somogysámsonból érkezett Kovács Zoltán (2. kép), a mezőny legfiatalabb tagja kapta. A mindössze 7. osztályos tanuló a zsűri előtt magabiztosan, frappáns módon mutatta be Spectrumra írt, tulajdonképpen a Rubik kocka síkbeli változatának mozgatásaihoz hasonló kirakós programját.

Az elért eredmények kihirdetésekor, a díjak kiosztásakor minden versenyző átvehetett egy emléktárgyat a μ'89-re szóló meghívóval együtt, hisz a versenyzők március 21-én az ottani közönségnek is bemutathatták programjaikat.

A pályázat jövőre is folytatódik! A programok szerzőinek sok sikert kíván a szerkesztőség.

Énekes Ferenc

2. kép



Asztali mikroszámítógépek

Sorozatunkban azokat az új hardver- és szoftvertermékeket ismertetjük, amelyek várhatóan általánosan elterjednek, és meghatározó szerepük lesz a fejlődés irányainak kialakításában.

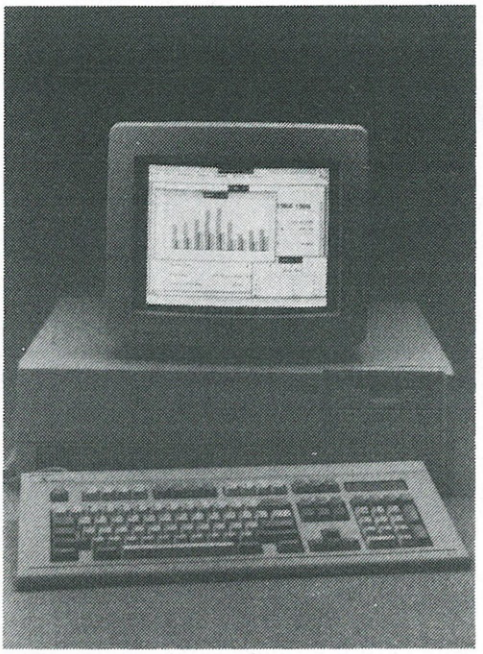
A számítástechnika fejlődése értelemszerűen magával hozta, hogy megjelentek újfajta alkatrészek, majd az ezekkel megépített új típusú gépek, melyeknél vagy a teljesítmény volt nagyobb az addigiakkal összehasonlítva, vagy a méretük volt kisebb. Eleinte általában a teljesítmény változott először, és utána a méret. Később fordult a kocka. Egyrészt megváltozott a sorrend, másrészt néha egyszerre következett be mindkettő.

Nemhiába várták tehát a házi számítógépek teljesítményének növekedése után, hogy következik a zsebszámítógépek megjelenése, ahogy ez a számológépeknél is történt. Ehelyett azonban a nagy cégek inkább abbahagyták a házi számítógépek fejlesztését — a legtöbb, a Commodore kivételével a gyártását is —, és áttértek a személyi számítógépek erőteljes fejlesztésére. Persze azért a miniatűrizálás mégiscsak hatott, létrehozva az úgynevezett lap-top csoportot, ami talán úgy fordítható, hogy ölbe vehető. Így jelenleg hagyományos asztali személyi számítógépek és ölbe vehető gépek vannak.

Nevezhetnénk egyébként ezeket az írógépekhez hasonlóan táskagépeknek is. Ez annál is indokoltabb, mert vagy táskával együtt árulják őket, vagy maga a gép táska formájú, hordfüllel ellátva.

Mitsubishi gyártmányok

Itthon ez a cég nem számítástechnikai termékeiről ismert. Ismertek viszont — bár inkább csak hírből — személygépkocsijai, háztartási elektronikai berendezései, gép- és vegyipari berendezései, termékei, valamint banktevékenysége.



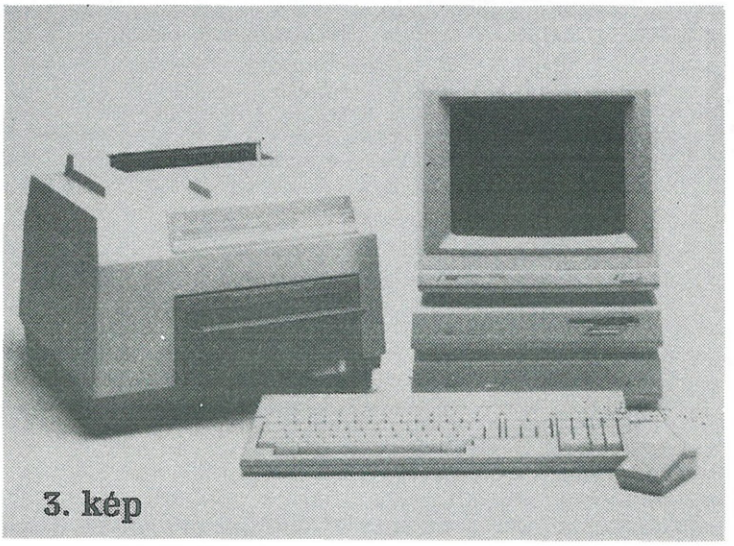
1. kép

Számítástechnikai berendezéseket félvezető termékeire alapozva gyárt ugyan már tizenkét éve, de a többi profilhoz képest kisebb intenzitással. Mégis, a professzionális felhasználóknak eddig 400 ezer személyi számítógépet adott el a cég az Egyesült Államokban.

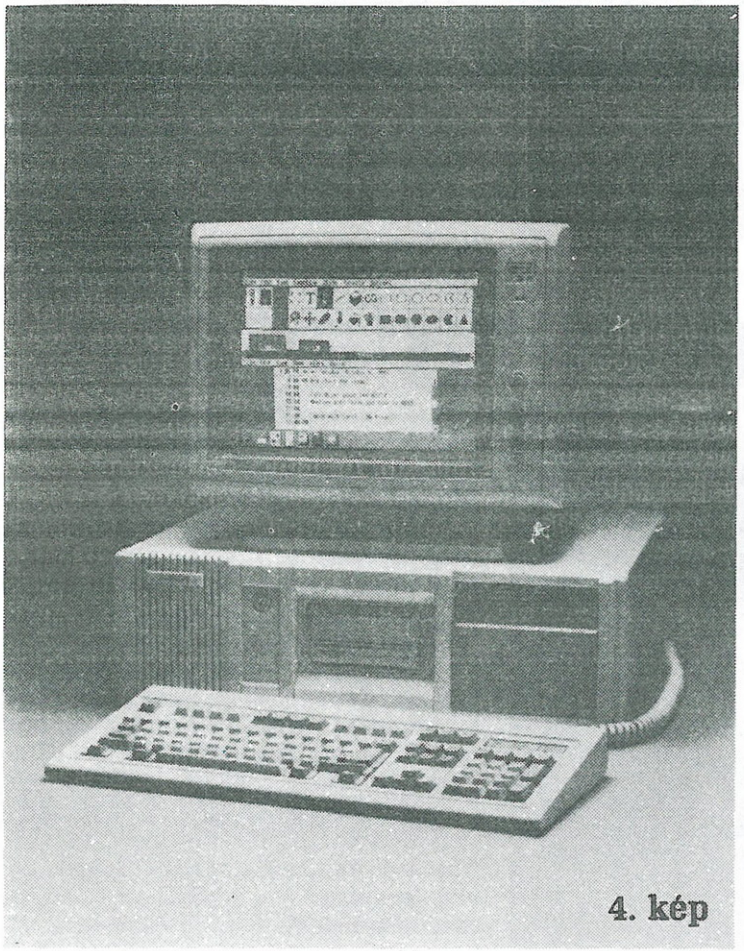
Az MP286-os sorozat rendkívül változatos, összesen 24 különböző kiépítésű típusa létezik. A változatosság a lemezegységek számában (1,2), méretében (5,25" 1,2 Mbájt, illetve 3,5", 1,44 Mbájt), a merevlemezegységek kapacitásában (20, illetve 40 Mbájt), a monitorok színes és monokrom voltában — a színesek lehetnek EGA vagy VGA kompatibilisek — mutatkozik meg. Mindegyik típusnak a tárkapacitása 640 kbájt, ami kiterjeszthető 5,5 Mbájtra, és cserélhető BIOS ROM-ja van (gondoltak a jövőre is). 12 MHz órafrekvenciájúak, négy AT és két XT kompatibilis kártyát fogadhatnak, ezenkívül egy soros és egy párhuzamos csatornájuk van. A nagy változatossággal igyekeznek elérni azt, hogy bármilyen felhasználó igényét kielégíthessék.

Az MP386 sorozatnak *(1. kép)* három alaptípusát gyártják. Közös bennük, hogy 1 Mbájt tárkapacitásúak, ami az alapkártyán megkétszerezhető, és kiterjesztő kártyákkal 10 Mbájtig növelhető. Az előző sorozat kártyabefogadó kapacitását kiegészítették úgy, hogy két további 8/16/32 bites buszú kártya fogadására alkalmas legyen. A ki/bemeneti vonalak itt is azonosak az előzőével. A lemezegységek egységesen 5,25" méretűek, a merevlemezek mérete 40, illetve 70 Mbájt. A monitorok típusa ugyanaz.

2. kép



3. kép



4. kép

5. kép

Atari gépek

Az 520ST sorozatot úgy módosították, hogy ellátták 720 kbájtos beépített lemezegységgel *(2. kép)*.

A nagy előrelépést azonban nem ez, hanem a Professional Office DPT sorozat megjelenése jelentette, ami Mega 4 monokrom gépet, 30 Mbájtos merevlemezegységet, lézernyomtatót, a Timeworks DTP és a VT100 Terminal emulációs szoftvert tartalmazza *(3. kép)*. A rendszer fontos jellemzője az egyes egységek közötti nagy átviteli sebesség (1,33 Mbájt másodpercenként), a GEM-környezet, amelyek együttese lehetővé teszi azt, hogy egy szöveget és rajzokat tartalmazó lapoldal előállítási ideje mindössze 30 másodperc legyen. Mindennek alacsony az ára. Az úgynevezett ajánlott ár — amelynél a kiskereskedelmi ott mindig alacsonyabb — 4400 dollár.

Tandon mikrogépek

A vállalat nevet igazán akkor szerzett magának, amikor nyilvánosságra hozta, hogy az IBM PC-kben az ő lemezegységei találhatók. Ezt követően először a lemezegységekkel, majd a mikrogépekkel aratott sikereket. Miért esett az IBM választása a Tandonra? A céget a névadó Sirjang Lal Tandon 1975-ben alapította. Egyetlen év elég volt ahhoz, hogy a lemezegységek fejeinek legnagyobb gyártójává váljanak. Ők készítették el az első kétoldalas fejeket. 1979-ben a piac kétharmadát tartották a kezükben. 1984-től kezdve gyártottak számítógépeket, először az európai piacra. Egy év alatt 60 ezer darabot adtak el. Ők kezdték el a cserélhető merevlemezegységek gyártását is. (Ezekre a sorozat megfelelő helyén visszatérek.)

A Targa elnevezésű sorozat után most a PAC 286 Plus a sláger (lásd a hátsó belső borítón), 1 Mbájt tárolójú (5-re kiterjeszthető), két Tandon Personal Data PAC helyezhető el benne, 1,2 Mbájtos lemezegysége van . A merevlemezegység vezérlője RLL típusú, kapacitása növelhető, 128 kbájtos gyorstároló javítja az adatforgalmat a merevlemezegységgel. A soros csatorna az alapkártyán található, tartozéka egy párhuzamos csatorna, egy XT és egy AT kompatibilis kártyát fogadhat, a monitor pedig monokrom, Hercules kompatibilis.

A Tandon 386 *(4. kép és borító)* öt változatban készül, melyek a merevlemezegységek kapacitásában (40—70, illetve RLL illesztővel 110 Mbájt, 16—20 MHz órafrekvencia) különböznek egymástól. A tárkapacitás 1 Mbájt, ami kiterjeszthető 8 Mbájtig. Egyebekben azonos az előző sorozattal, kivéve azt, hogy két XT és hat AT kompatibilis kártyát fogadhat.

Panasonic IBM kompatibilisek

Ezt a céget jól ismerik idehaza, de nem mint számítógépgyártót, hanem magnetofonok, háztartási gépek készítőjét. Ezeket a gépeket nem egyedül, hanem a Tandyvel együttműködve gyártja.

A termékeknek háromtagú sorozata adja az XT kompatibilis részt. Mindháromnak a tárolója 640 kbájtos és öt kártyát fogadhat, 4/8 MHz órafrekvenciája és 720 kbájtos lemezegysége van. Az egyes tagok abban térnek el egymástól, hogy hány lemezegységük van és van-e vagy nincs 20 Mbájtos merevlemezegységük. Az *5. képen* az FX—1650FD2 (két lemezegység) látható, aminek éppen nincs ilyen merevlemezegysége.

Az AT kompatibilis háromtagú sorozat szintén 3,5" lemezegységet használ. Az előzőktől egyedül a mikroprocesszorában tér el, ami a *6. képen* látható FX—1750FD2-nek az 5. képpel való összehasonlításából azonnal észrevehető: külsőre mindkettő ugyanolyan.

A 20 MHz órafrekvenciájú, 2 Mbájt tárkapacitású (16 Mbájtig kiterjeszthető), hat AT és két XT kompatibilis kártya fogadására alkalmas két típus

6. kép

7. kép

1,44 Mbájtos 3,5" lemezegységű. Az FX—1950 *(7. kép)* nem, a másik azonban tartalmaz még egy 40 Mbájtos merevlemezegységet is. Mindkettő OS/2 kompatibilis.

Mindegyik géphez gyártanak CGA, EGA, VGA és monokrom illesztőt, valamint monitort. Ezekről a sorozat egy másik részében lesz szó.

Egy Amstrad gép

A PC2086 *(8. kép)* önállóan vagy hálózat részeként használható. Beépítve 640 kbájt, amelynek bővíthetőségéről nincs információm. A gép XT kompatibilis, 8 MHz órafrekvenciájú, egy vagy két 3,5" méretű, 720 kbájtos lemezegységű, esetleg egy 30 Mbájtos merevlemezegységű. Külső, 5,25" méretű lemezegység csatlakoztatható hozzá. Billentyűzete AT típusú, beépített soros és párhuzamos csatornája van. Tartozéka egy Microsoft kompatibilis egér, illesztővel együtt, továbbá a szokásos MS—DOS szoftveren kívül a Microsoft Windows V. 2. 03 is. Monitora és illesztője VGA kompatibilis, de használhatók hozzá MDA, Hercules, CGA és EGA kártyákat igénylő szoftverek is. Csak három szabad kártyahelye van, mert a többit a gép maga használja.

Simonyi Endre

8. kép

Miénk a vár: AMIGA!

Előszó

Elhatároztatott: új sorozat indul, melynek témája az Amiga lesz. Lehetőségeinkhez mérten megpróbáljuk csökkenteni azt az űrt, amelyet a magyar nyelvű szakirodalom hiánya jelent. Mindenből csak a leglényegesebbeket, az önálló munkával nehezen felfedezhető alapokat tárjuk az érdeklődők elé, és építünk a felhasználók egyéni kreativitására is.

Az első két részben az Amiga általános hardver jellemzőit mutatjuk be, amihez egy blokkvázlat is tartozik. A továbbiakban az operációs rendszert tárgyaljuk, valamennyi AmigaDOS utasítás magyar nyelvű összefoglalásával. Leírást adunk az AmigaBASIC-ről, ismertetve az utasításkészletet. Bemutatunk egy kiváló assemblert, a SEKA-t. Szó lesz még felhasználói programokról is, az Amiga grafikus üzemmódjairól, lemezformátumáról — a sávok és szektorok felépítéséről —, a rendszer hibaüzeneteiről (Guru-Meditations) és sok egyéb hasznos tudnivalóról.

A sorozat címe: The Amiga, Born a Champion. Az üzenet e számítógép alkotóitól származik. Igen furcsa módon bukkantunk rá. Az Amiga Workbench nevű magas szintű operációs rendszerében a két SHIFT, a két ALT és a funkcióbillentyűk egyidejű lenyomásakor más-más üzenetek fedezhetők fel a képernyő fejlécében. A címbeli szöveg akkor íródik ki, ha az öt gomb lenyomása mellett (harmadik kezünkkel!) kivesszük a meghajtóból a lemezt. A mondatot így lehetne lefordítani: az Amiga született bajnok. Nagyképűség? Így önmagában minden bizonnyal az. Ha viszont valaki figyelemmel kíséri sorozatunkat, rájöhet akár az ellenkezőjére is . . .

Mit tud az Amiga?

A forradalmian új dolgok — bármiről legyen is szó — szélsőséges érzelmeket váltanak ki. Ugyanez mondható el a Commodore cég Amiga típusú személyi számítógépéről is. A szakértők jobbára elismerik az Amiga igen nagy teljesítőképességét; igaz, akadnak olyanok is, akik — általában helytelen információk alapján vagy éppen konzervatív szemléletük miatt — lebecsülik ezt a gépet. Felhasználóként gyakran hallom a tábor érveit és ellenérveit. Ezért úgy vélem, ideje végre meghatározni az Amiga helyét, összemérni a különféle számítógépekkel, természetesen a realitások alapján végezve az összehasonlítást.

Az Amiga központi egysége egy MC 68000-es mikroprocesszor. Órajelének frekvenciája 7,16 MHz. A 68000-es ennél gyorsabb működésre is képes. Az alapfrekvencia a képelőállító chip miatt ekkora, mert a PAL színes rendszerrel kompatibilis kép létrehozásához 28,64 MHz-es frekvencia szükséges. Ha ezt néggyel osztjuk, előáll a 7,16 MHz-es frekvencia. Ez az érték egyáltalán nem kevés, ha összemérjük az Amigát az akkoriban — 1985-ben — piacra dobott PC-kkel vagy a Commodore 64 1 MHz-es, a C128 1 vagy 2 MHz-es és a Sinclair Spectrum 3,5 MHz-es órajel-frekvenciájával. Az MC 68000-es adottságait a februári számban már ismertettem, ezért itt csak néhány mondatban foglalkozom vele.

Ennek a mikroprocesszornak 32 bites címés adatregiszterei vannak, noha a külvilág a címbitekből csak 24-et, az adatbitekből pedig csak 16 bitet lát. A címbusszal tehát maximálisan 16 Mbájt memóriaterületet címezhet meg közvetlenül.

Tovább növeli a lehetőségeket — az utasítások száma végül is meghaladja az ezret — a 14-féle címzési mód. Azért mégse riadjon vissza senki ennek a processzornak a programozásától; a 68000-es nem olyan bonyolult, mint amilyennek az induláskor tűnik.

A Commodore cég háromféle Amigát gyártott és gyárt. Az első az Amiga 1000-es volt, majd utána következett az Amiga 2000-es és az Amiga 500-as. Az utóbbi két típusban több olyan VLSI áramkör van, amelyet az Amiga 1000-ben még nem integráltak ilyen magas szinten. Az Amiga 1000 256 kbájt, az Amiga 2000 1 Mbájt, az Amiga 500 512 kbájt RAM-ot tartalmaz alapkiépítésben. Eltérő a mechanikai felépítésük is. Az 1000-es és a 2000-es típusok még PC kinézetűek — a lemezmeghajtót és az elektronikát tartalmazó dobozhoz lengőkábellel csatlakozik a billentyűzet —, az 500-as már jobb helykihasználású, a hordozhatóságot szem előtt tartó kivitelben készült. A billentyűzet, az elektronika és a lemezmeghajtó közös dobozban van. Az Amiga 1000-es az áramkörök alacsonyabb integráltsági foka miatt igen drága volt, és a viszonylag kis memóriakapacitás is nehezítette a gép használatát.

Az Amiga ROM-ja a 256 kbájtos, úgynevezett Kickstart ROM, a rendszerprogram fő moduljait tartalmazza, vagyis valamennyi grafikát, lemezt, portokat és perifériákat, a hanggenerálást kezelő rutint, a beszédszintézishez szükséges fonémákat is beleértve.

A gyártó szerint a gép RAM-ja 8 Mbájtig bővíthető, ez az érték azonban nem valós. Véleményem szerint nincs akadálya a 15 Mbájtos RAM kiépítésének sem, hiszen még így is megmarad az operációs rendszernek fenntartott 1 Mbájt.

Az AmigaDOS — az Amiga operációs rendszere — lemezről tölthető a memóriába, ugyanúgy, mint például az MS-DOS. Utasításainak száma az 1.2 verziónál 51, az 1.3-nál 64. A lemezen tárolt operációs rendszerek nagy előnye, hogy azok szabadon javíthatók, tökéletesíthetők, bővíthetők a számítógép pályafutása során, nem úgy, mint a fix programmal ellátott ROM-ok. Ezt a koncepciót a lemezmeghajtó nagy sebessége teszi kivitelezhetővé. A programozó kétféle környezetben kezelheti az Amigát: alacsonyabb szinten, ahol az úgynevezett CLI (Command Line Interface) ablakon keresztül gépelheti be a DOS-parancsokat, illetve magasabb szinten, a Workbenchben.

A Workbench ikonorientált fájlkezelő rendszer, amely AmigaDOS-ból a LOADWB parancs kiadásával hívható a lemezről. A Workbenchben minden lemezt, könyvtárat és fájlt a saját grafikus szimbóluma képvisel. Ezt röviden és találóan ikonnak hívják. Helyezzünk be például egy lemezt a meghajtóba. Ennek ikonja megjelenik a képernyőn. Ha az egérrel mozgatható nyíl (pointer) segítségével rámutatva az ikonra, egymás után kétszer megnyomjuk az egér kiválasztó gombját, a rendszer kinyit egy ablakot, amelyben — szintén ikonok képében — megjeleníti a lemez tartalmát.

Az egyes ablakok mérete tetszés szerint változtatható, a sok ablak közül bármelyik előtérbe vagy háttérbe helyezhető, szabadon elmozgatható. Az Amiga multitasking képességének jóvoltából az ablakokban különféle programok futhatnak egyidejűleg grafikával, zenével és lemezkezeléssel. Különleges élmény például egy zeneprogram betöltése, elindítása valamilyen lemezről, majd betöltve mondjuk az AmigaBASIC-et, halk zene mellett programot írni. A képernyők és ablakok számának csak a memóriakapacitás szab felső határt. Miután a Workbenchben dolgozva olyan rendetlenséget hozhatunk létre a képernyőn, mint egy íróasztalon, jogos a „munkaasztal" elnevezés.

Az Amiga egyéniségét négy speciális, nőkről elnevezett VLSI áramkör adja. PAULA felel a hangokért és a perifériakezelésért, FAT AGNUS (Kövér Ágnes) az animációért, DENISE a grafikáért, GARY pedig a lemezkezelés és a vezérlőbusz bizonyos funkcióiért.

A titokzatos nevek egyébként az áramkörök feladatának rövidítései. PAULA: Ports, Audio and UArt; AGNUS: Address GeNerator circuit; DENISE: Display ENcoder; GARY Gate ARraY. Bonyolultsági fokukra jellemző, hogy ezek az áramkörök egyenként mintegy 20 ezer tranzisztort tartalmaznak.

PAULA négy hangcsatornás: kettő a bal, kettő a jobb oldalon szól sztereóban. Igaz, ez önmagában nem túl sok, viszont az Amiga multitasking rendszerű gép. A hangcsatornák multiplexelésével akár 32 — oldalanként 16 — hangszert is megszólaltathatunk, a fázisviszonyok szabályozásával pedig a kvadrofonikus hangzás is elérhető. A frekvenciatartomány 9 oktványi, tehát frekvenciakorlátokról nem eshet szó. A kiadott hullámforma általunk definiált, nem köt előre megadott jelalak, mint például a C64-nél a színusz, háromszög- vagy négyszögjelekhez. A hang modulációja mind amplitúdóban (AM), mind frekvenciában (FM) lehetséges. A multitasking itt is új lehetőségeket kínál: a hangerősség 64 fokozatban állítható. A DC szintek kapcsolásakor fellépő — és a C64 esetében létező — kattanások hiánya és a zajtalanság teszi az Amiga hangját valóban hifi minőségűvé.

Két korlátról azért beszélni kell, amikor PAULA-t tárgyaljuk. Az egyik ki is védhető. Nevezetesen az, hogy PAULA nyolcbites digitál-analóg konverteket használ, vagyis az úgynevezett kvantálási hiba 256-szorosa a 16 bites rendszerű CD lemezjátszókénak. A második hiba a mintavételezésből adódik: a hasznos hanghoz hozzáadódik egy, a mintavételezés frekvenciájának megfelelő magasságú hang.

Ezzel a gép tervezői is számoltak: aluláteresztő szűrőket helyeztek el a hangfrekvenciás kimeneteken. A szűrők az egyik 8520-as CIA chip regiszterén keresztül be- és kikapcsolhatók. Ha a mintavételezési frekvenciát akkorára választjuk, hogy az ne essen a hallható tartományba, a hibát nem vesszük észre, és így akár ki is kapcsolható a szűrő. A piros színű, POWER feliratú LED egyébként a szűrő bekapcsolt állapotát jelzi.

A lemezvezérlő, az általános célú B/K regiszterek és a digitál-analóg átalakító regiszterei is PAULA részei. A Commodore cég végre szakított saját interfészrendszerével, és az Amigát már felszerelte szabványos RS232C, párhuzamos Centronics, valamint analóg és digitális RGB be- és kimenetekkel. Ezek a portok mind szabványos Canon-D típusú csatlakozókat használnak.

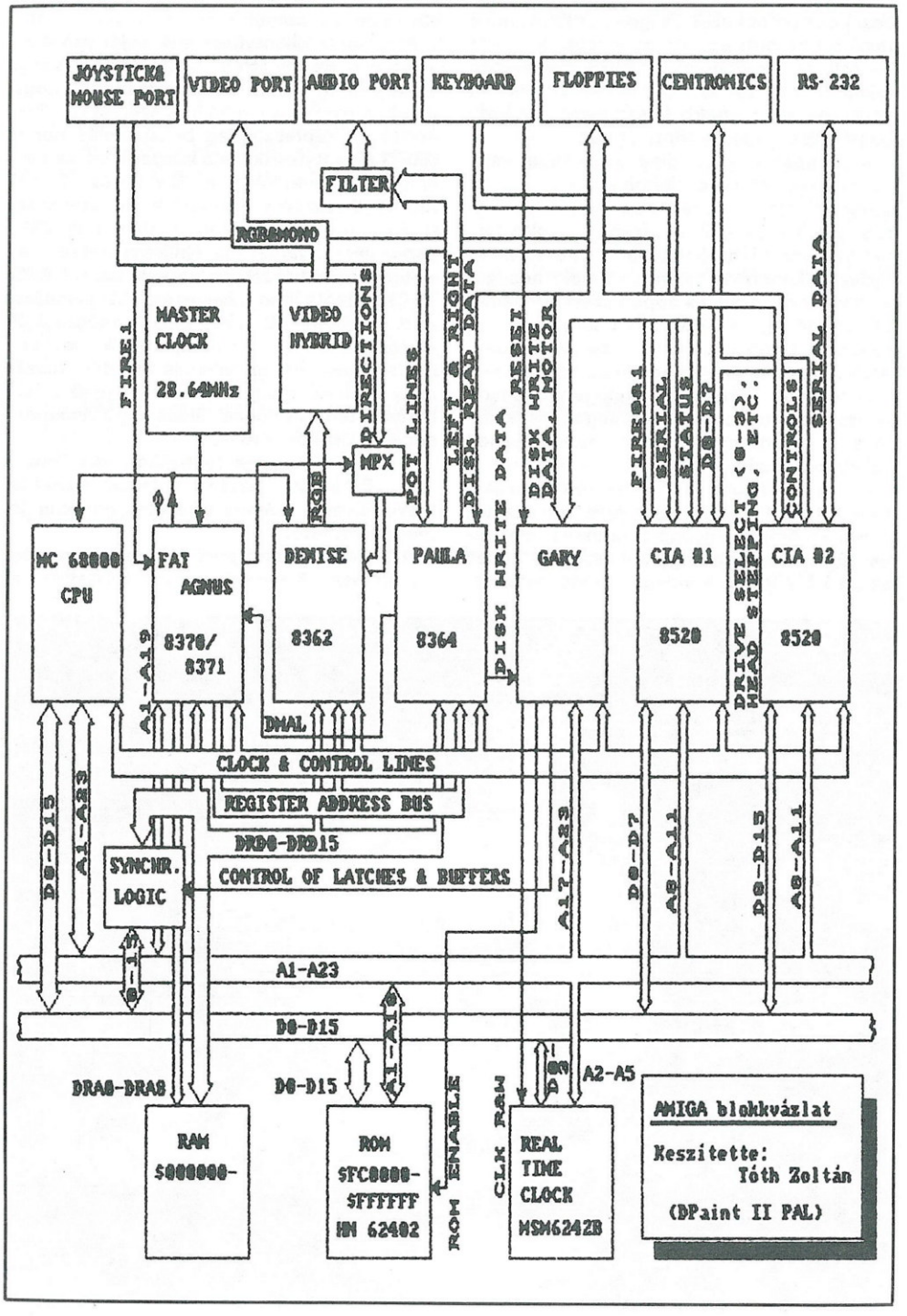
Az alapkiépítésű gépen csak monokrom összetett videojel és az RGB jelek találhatók meg. Ha színes videojelre vagy tévé antennajelre van szükségünk, vásárolnunk kell egy A520 típusú modulátort. Ez olcsón beszerezhető. Az Amiga kiváló gyrafikus lehetőségei csak egy RGB bemenetű tévékészüléken vagy monitoron bontakoznak ki igazán.

DENIS, a grafikus IC, tetszőleges számú grafikus üzemmódot képes egyidejűleg megvalósítani. A felbontási tartomány 320 x 200-astól 640 x 400-asig terjed az NTSC gépeknél és 640 x 512-ig az Európában használatos PAL rendszerűeknél. Mivel az Amiga a teljes képernyőre is rajzolhat — a kijelzési mező méretének megnövelésével és pozíciójának eltolásával a keret megszüntethető —, az egész képernyőre vonatkozó felbontás meghaladja a 700 x 550-et. A különböző ablakokban lehet szöveg — a 40-től a 80 oszlopos kijelzésig minden méretben — vagy kis- és nagy felbontású grafika. Az Amigánál ugyanis nincs olyan karakter-ROM, mint a C64-nél, viszont a karakterek megjelenítése is bittérképes.

DENISE regiszterei a három alapszínre (R, G és B) egyenként négybitesek, tehát a rendelkezésre álló színek száma $2^{3\times4}=4096$. Normál üzemmódban 2, 4, 8, 16 vagy 32 színű palettát használhatunk, az egyes színek a 4096-ból szabadon választhatók. Maximális felbontásnál egyébként legfeljebb 16 színű paletta hozható létre.

Ha egyszerre mind a 4096 színre szükség van, akkor a HAM (Hold And Modify = tartás és változtatás) üzemmódot állítjuk be. Itt az egyes képpontok színét vagy egy 16 színű palettából választhatjuk ki, vagy átvesszük a képpont bal oldali szomszédjának színét, továbbá megadjuk, hogy a színen melyik összetevőt és milyen mértékben kívánjuk módosítani. A HAM üzemmód az összes szín alkalmazása mellett gazdaságos memóriafelhasználást nyújt. Különösen digitalizált képeknél használható előnyösen, mert a valóságot ábrázoló képeken az ugrásszerű változásokkal szemben a folyamatos színátmenetek dominálnak.

A függőleges felbontást kiterjesztő üzemmód az INTERLACE. Normál üzemben a függőleges felbontás felső határa mintegy 280, ezt csak INTERLACE módban léphetjük túl. Ilyenkor a két fél kép már más információt tartalmaz, ezért a képernyő az eddigi 50 Hz-



es félkép-frekvencia helyett a 25 Hz-es képfrekvencia ütemében íródik újra. Ez a képcső utánvilágítási idejétől függő mértékű villódzást okoz. Sokan az Amigát okolják a nagy felbontású képek villogása miatt, noha a 25 Hz-es képfrekvencia európai szabvány és minden közönséges tévékészülék ennek megfelelően működik. A gép tervezőinek — nagyon helyesen — a kompatibilitás volt a legfontosabb, ezért is tartották magukat ehhez a normához.

DENISE tartalmazza a sprite-regisztereket is. 16 sprite-regiszter írja le a nyolc sprite-ot, amelyek négyszínűek, méreteik nem változtathatók. A sprite-okat leíró adatok az alsó 512 k RAM-ban bárhol elhelyezhetők.

FAT AGNUS a legsokoldalúbb az Amiga speciális áramkörei között. Ez a chip nem kevesebb, mint 25 DMA (Direct Memory Access) csatornát kezel. Ez igen gördülékennyé teszi a központi egység és a többi áramkör közötti adatforgalmat. AGNUS belsejében található a Blitter és a Copper — más néven koprocesszor —, nekik köszönhetők az Amiga páratlan grafikus adottságai.

A Copper gyakorlatilag az Amiga valamennyi regiszterének feltöltésére és kiolvasására alkalmas. Vezérlésére mindössze három utasítás szolgál, egyszerű, de sokoldalú programozást lehetővé téve. A Copper tartja nyilván a képernyőt telerajzoló elektronsugár x és y koordinátáit, és éppen ezért kiemelkedő jelentőségű a HAM üzemmód létrehozásában. A Copper a WAIT utasítás hatására addig vár, amíg az elektronsugár nem éri el a megadott pozíciót. Ez a funkció tehát nagyon pontos időzítésre ad lehetőséget, és alkalmas a számítógép teljes működésének szinkronizálására.

Mégis AGNUS és az Amiga igazi sztárja a Blitter. Ez egy igen hatékony grafikai eszköz, innen az Amiga lenyűgöző grafikus adottságai. Ez az áramkör nagy sebességgel képes az alsó 512 k-ban a videomemória tartalmának mozgatására. Ezt olyan gyorsan teszi, hogy ha ez a feladat MC 68000-re hárulna, akkor annak 70 MHz-es órajel-frekvencián kellene működnie.

Ebben rejlik az Amiga igen gyors grafikai kezelésének titka. A Blitter egy ilyen áthelyezés során a memóriatartalommal logikai műveleteket is végezhet. Egyetlen korlátja, hogy e műveletekre csak az aktuális videomemóriával képes. A Blitter segít a gyors területkitöltésben, vonalak és körök rajzolásában, valamint az animációban a GEL (Graphic ELement) rendszeren keresztül. GEL-en a BOB-ok (Blitter OBjects) értendők, amelyek tulajdonképpen tetszőleges méretű sprite-ok, és a VSprite-ok (Virtual Sprites), amelyekkel a nyolc sprite-os készlet bővíthető. E képességek nyerik meg az Amigának az Atari 520ST-vel és az Apple Macintosh-sal szemben a versenyt.

Az Amiga billentyűzetének saját processzora van. Ez egy 6500/1-es központi egység, 64 bájt RAM-mal, 2 kbájt ROM-mal és négy darab, egyenként nyolcbites szélességű B/K porttal. A tápfeszültség bekapcsolásakor a 6500/1 billentyűzetkezelő központi egység ellenőrzőösszeget készít a ROM-ról és a RAM-ról, majd rászinkronizálódik a rendszerbuszra. Ezután a számítógépben folytatódik a bekapcsolási folyamat. Ha a billentyűzet tesztje valamilyen okból sikertelen volt, azt a CAPS LOCK billentyűben elhelyezett LED villogása jelzi. A villogás üteme a hiba természetéről ad tájékoztatást: egy villanás ROM-, két villanás RAM-, három villanás időzítési hibát, négy villanás pedig rövidzárlatot jelez a billentyűzeten. A villogási ciklus egy másodpercenként érzékelhető.

A billentyűzet elrendezésében és külalakjában PC-stílusú. Az ALT billentyű egyidejű lenyomásával számos ékezetes karakter is megjeleníthető.

A lemezmeghajtó port a Commodore cég előrelátását bizonyítja, mert folytatták a 1571-es meghajtónál érvényesítetteket. A lemezvezérlő 8520-as CIA áramkör egyaránt képes a GCR és a MFM formátumú lemezek kezelésére, jelentősen megkönnyítve az emulátorprogramok Amigára írását. Emiatt — no és persze az MC 68000-nek köszönhetően — láthattak napvilágot a C64, az Atari ST és az IBM XT emulátorprogramok. Az Amiga beépített lemezegysége 3,5"-os, kétoldalas lemezeket kezel, amelyeket 880 kbájtra formáz. További három külső floppymeghajtó is csatlakoztatható a géphez, kettőnél több külső lemezegység használatakor azonban külön tápellátásról kell gondoskodni. A meghajtók daisy chain kapcsolatban állnak egymással. Mivel az Amiga nem használ intelligens meghajtókat, a lemezméret sem kötött, lehet 3,5"-os vagy 5,25"-os. Csatlakoztatható továbbá még winchester is, de ehhez már vezérlőkártya szükséges.

A géphez kínált hardverbővítések skálája igen széles. Kapható IBM XT és AT emulátorkártya, alkalmazásuk teljes kompatibilitást valósít meg az IBM PC-vel. Egyetlen hátrányuk, hogy igen drágák. Az XT kártyáé csaknem az Amiga 500-as árával egyenlő, az AT kártya pedig drágább, mint az Amiga 2000.

Beszerezhető többféle kép- és hangdigitalizáció. A digitalizált hangmintákat a zenei programok hangszerként használják, ezért megtehetjük, hogy kedvenc zeneszerzőink hangját magnószalagról digitalizáljuk, majd a beírt kottát rajtuk játszatjuk el. Ugyanezt tehetjük az emberi hangokkal is, így akár énekelhet is a számítógép!

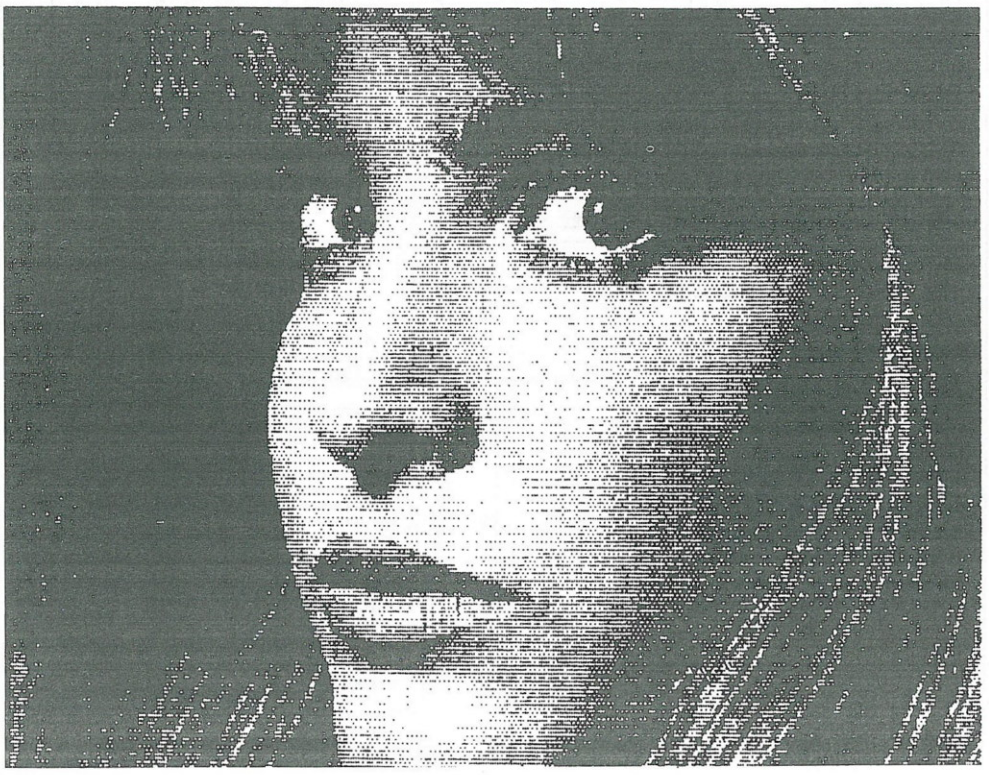
Az Amigához kapcsolható teletext dekóder nagy előnye, hogy vele a képújság oldalai feldolgozhatók, lemezen tárolhatók, kinyomtathatók.

Ahogy a C64-hez, úgy ehhez a géphez is léteznek EPROM programozókártyák, modemek, analóg/digitális és digitális/analóg konverterek, fényceruzák, rajzolótáblák.

A képdigitalizálók működhetnek videokameráról vagy digitalizálhatnak írásos dokumentumokat, fotókat. A Handy Scanner nevű eszköz például a nyomtatott írást grafikus formátumra digitalizálja, majd az egyes karaktereket azonosítja és a felismert szöveget szöveges fájllá alakítja. Gondoljuk csak el, mennyivel könnyebb vele például egy hosszú programlistát bevinni a számítógépbe! És a házi videózásba is bevonható az Amiga, ha van Genlock interfészünk vagy videokeverőnk. A digitizerrel rögzített képeket lézernyomtatóval színes fotó minőségben nyomtathatjuk ki. Innentől azonban kezd a dolog költségessé válni, jobb is, ha abbahagyom a felsorolást.

Befejezésként egy utolsó megjegyzés azokhoz, akik szeretik az Amigát a játékgépkategóriába beskatulyázni. Igen, tudom, hogy eredetileg játékgépnek indult, de szerencsére ennél jóval színvonalasabbra sikerült. Tudom, hogy kiválóan alkalmas szemet és fület elkápráztató játékprogramok futtatására — van is belőlük éppen elég —, de azt már nem hiszem, hogy komoly hardveres lehetőségeivel nem alkalmas magas szintű matematikai, műszaki vagy tervezői feladatok megoldására. Éppen ellenkezőleg! Amire egy IBM PC csak különféle bővítőkártyákkal telerakva képes, azt az alapkiépítésű Amiga úgyszólván fél kézzel megcsinálja.

Tóth Zoltán



OLVASTUNK..

és ha tetszettek előző kalandozásaink az elektronika, a villamosságtan alapjai körül, akkor tartsanak most is velünk — folytatjuk a témát.

Elektronikus szerkezeteink energiaellátása

Írtuk: (mai?) elektronikus szerkezeteink a működésükhöz a „külvilág"-tól *energiát* igényelnek. Ezt megtehetjük úgy, hogy a gépet „bedugjuk" az országos villamosenergia-hálózat közelünkben lévő végpontjába, a legközelebbi konnektorba, vagy például veszünk a boltban néhány elemet, feltölthető, újratölthető akkumulátort, és a gép működtetéséhez szükséges energiát — munkavégző képességet — egy megfogható tárgy által hordozott formában „tesszünk bele" a gépbe.

Nagy teljesítményű számítóközpontok gépeinek ún. szünetmentes energiaellátása természetesen ennél sokkal bonyolultabb: itt szinte követelmény a *folyamatos* ellátás akkor is, ha az országos villamosenergia-rendszerben valamilyen — katasztrófáktól eltekintve persze — mindig „rövid", átmeneti kiesés következik be. Ha nem így tennénk, a rövid, előbb-utóbb bekövetkező áramkimaradások miatt értékes adatok vesznének el a gépben. Ezért a nagy számítóközpontok gépeit a gondos és „pénzes" üzemeltetők sohasem kötik *közvetlenül* az országos villamos hálózatba, hanem közbeiktatnak egy viszonylag nagy **energiatárolót** (lásd az *1. ábrát*), amellyel a gép a hálózatkiesés alatt még legalább annyi ideig működtethető, amíg a nyitott állományok rendben lezárhatók és a gép szabályosan kikapcsolható. (Ez gyakran negyedórás procedúra, de a nagyobb bajmentes vészkikapcsoláshoz is legalább másodpercek kellenek.) Ez a tároló lehet egy szobányi akkumulátortelep, de kisebb kimaradások áthidalására megfelelhet egy helyi áramfejlesztő generátor tengelyére szerelt nagy, forgó tömeg — lendkerék — is. Ezek a megoldások arra is jók, hogy a gépet megvédjék az elektromos hálózatban mindig jelenlévő zavaroktól is. (A zavar lényege, hogy például a hálózat két vezetéke — pontja — közötti feszültség nem pontosan színuszos: tüskék, hirtelen ugrások lehetnek rajta, továbbá a feszültség értéke eltérhet a névlegestől stb.). A hálózatról működtetek egy villamos motort, ennek tengelyén elhelyezek egy lendkereket és egy áramfejlesztőt — a számítógépet erről táplálom: az eredeti hálózat kisebb zavarai ennélfogva nem jutnak el a számítógépig. Az egész együttes úgy működik, mint egy szűrő.

Az energiatárolók, energiaforrások

Ott, ahol nem tudunk energiát vételezni valamilyen erőműrendszerre épülő országos hálózatból, nekünk, magunknak kell helyben gondoskodnunk energiaforrásról, illetve az erőmű vagy az energiahálózat meghibásodásaira is nekünk kell felkészülnünk energiatárolók közbeiktatásával.

Az energiatároló annál jobb, minél olcsóbb, minél egyszerűbb, minél kisebb tömegű (méretű), minél veszélytelenebb stb.

Csak érzékeltetésül, hogy mekkora különbségek adódnak, hasonlítsunk össze — a teljesség igénye nélkül — néhány energiahordozót (tárolót) például az egységnyi tömegben tárolható energia (Wh/kg — emlékeztetőül 1 Wh = 3600 Ws = 3600 joule) alapján. Egy-egy energiatároló (hordozó) annál jobb, minél kisebb tömegben, minél több — általunk *szabályozottan* felszabadítható — energia tárolására alkalmas.

Kedvezőtlen Wh/kg mutatója van történetesen egy felettünk lévő dombon mondjuk 200 méter magasan lévő tónak, víztározónak, amelynek helyzeti energiáját egy áramfejlesztő generátorral alakíthatjuk át elektromos energiává. Ilyenkor 0,5 Wh/kg alatti értéket kapnánk. Vagy képzeljük el azt, hogy az országos villamosenergia-hálózatból nyert energiával vizet pumpálnánk fel a tóba és a tó vizével hajtanánk egy saját generátort, amelyre csak a számítógépünket kötnénk. (Tudomásunk szerint senki sem csinál ilyet!)

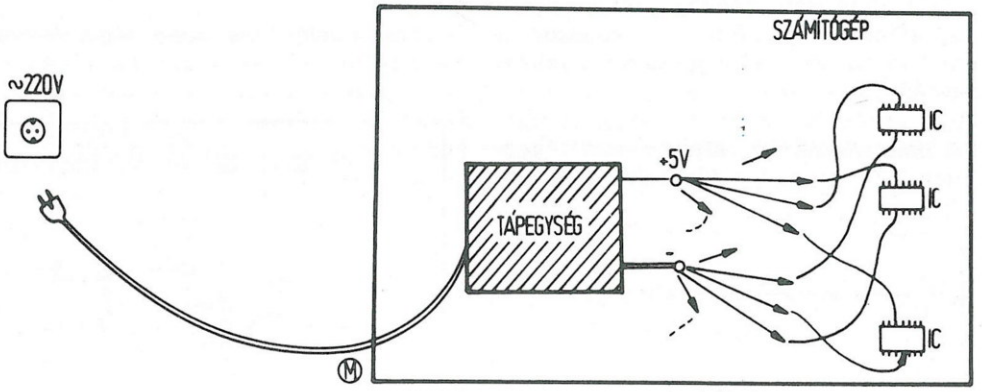
Nagy számítóközpontok számítógépeinél gyakran használt *ólomakkumulátoroknál* ez az érték valahol 30 Wh/kg körül, a lendkerekeknél majdnem 50 Wh/kg-nál van.

Vízmelegítős tárolóknál, ha a vizet 50 *kelvin* fokkal ($\Delta T = 50$ K) melegítenénk fel, már valamivel 50 Wh/kg feletti mutatót kapnánk, de nem tudunk ilyen számítógéphez használt megoldásról sem.

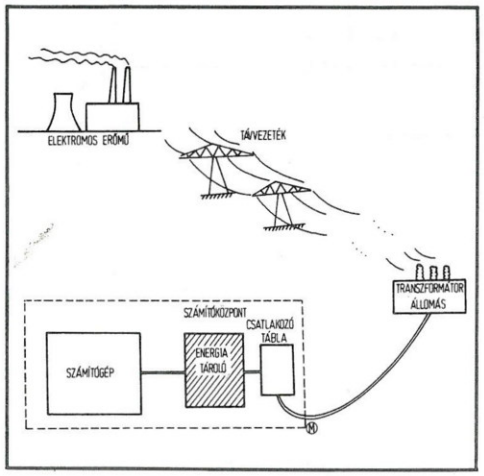
Csak az összehasonlítás kedvéért emlékeztetjük az olvasót néhány tüzelőanyag fajlagos energiájára. Tüzifa: <5000 Wh/kg; földgáz: >5000 Wh/kg; tüzelőolaj: $>10^4$ Wh/kg.

A generátorok

Azokat a szerkezeteket — generátorokat —, melyek nem elektromos energiát alakítanak át elektromos energiává, a vil-

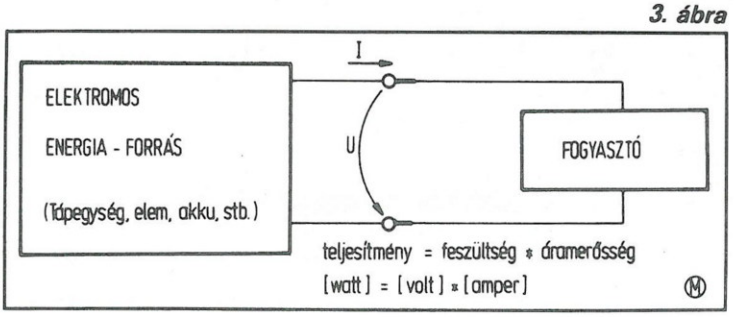


2. ábra. A számítógépet alkotó integrált áramkörök (IC-k) nem abban a formában igénylik az energiát, ahogy az a hálózatból vételezhető. Ezért van szükség tápegységre



1. ábra. „Igazi" számítóközpontok gépeit nem illik közvetlenül a villamos hálózatra kapcsolni

3. ábra



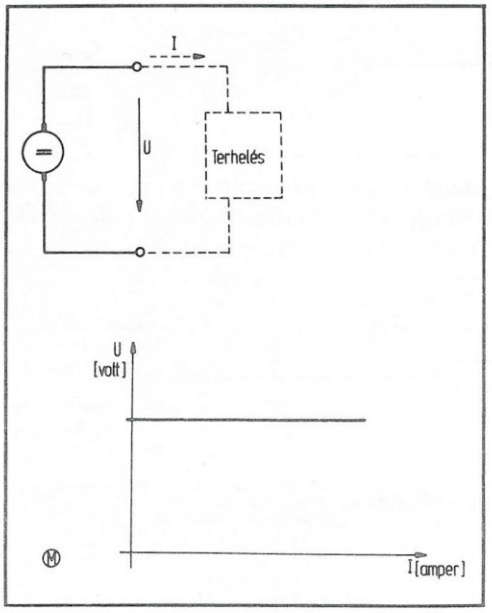
lamosságtanban úgy modellezhetjük, úgy képzelhetjük el, hogy bennük valamilyen *töltésszétválasztó mechanizmus* működik. Ezt a mechanizmust például azzal az előző cikkünkből megismert potenciállal, feszültségkülönbséggel jellemezhetjük, amelyre ez az áramforrás kapcsait fel tudja tölteni. A generátor — a galvánelem, az akkumulátor stb. — kapcsain létrejövő feszültség persze attól is függ, hogy terheljük-e árammal, és ha igen, mennyire. Másként fogalmazva: a telep „belső mechanizmusa" milyen mértékben, milyen ütemben képes pótolni a terhelésen keresztül a pólusokból elfolyt töltést.

Az akkumulátorok, a galvánelemek *belsejében* az áramot *nem* az elektromos tér, hanem valmilyen éppen ezzel *ellenkező irányba ható,* elektromos szempontból teljesen idegen, elektromos fogalmakkal le nem írható (például vegyi hatáson alapuló) jelenség — erő — hozza létre.

A tápegységek

Minden olyan számítógépnek, amelyet úgy konstruáltak, hogy a működéséhez szükséges energiát majd az országos villamos hálózatból fogja kapni, **tápegysége** van. Erre azért van szükség, mert a gépet alkotó **integrált áramkörök** — az IC-k — a működésükhöz szükséges energiát igen jól specifikált formában igénylik, és ez a forma jelentősen eltér attól a formától, ahogy az az országos (általánosan 220/380 voltos váltakozó feszültségű) hálózatból vételezhető. Az integrált áramkörök tipikusan néhány voltos stabil, egyenfeszültségű forrásból képesek a működésükhöz szükséges energia vételezésére.

4. ábra. Az ideális, stabil feszültséggenerátor feszültsége független az áramtól



Tápegységnek hívjuk számítógépeink és más, hálózatról működő elektronikus eszközeink — rádióink, televíziókészülékeink stb. — azon részét, amely a váltófeszültségű hálózatból vételezett energiát alakítja át az integrált áramkörök táplálásához szükséges stabil egyenfeszültségen vételezhető energiává (lásd a *2. ábrát*).

Egy elektromos energiaforrás által a környezetének leadott teljesítmény (az időegység alatt leadott energia) az áram és a feszültség szorzata (lásd a *3. ábrát*).

A gyakorlatban az energiaforrások kapcsain mérhető feszültség is és (terhelt esetben) az áram is változik az idő függvényében. A számítástechnikában — általában az elektronikában — használatos energiaforrások, a tápegységek, akkumulátorok stb. azonban közel állnak az ideális, állandó feszültségű forrásokhoz. Ennek az az oka, hogy az elektronikus eszközök építőelemei — az integrált áramkörök, illetőleg ezek építőelemei, a tranzisztorok — az energiát *stabil,* viszonylag szűk határok között specifikált feszültségű forrásból igénylik. Például az ún. bipoláris, **TTL** típusú áramkörök által igényelt tápfeszültség 4,75 és 5,25 V közé eshet. Az elektronikában használatos *tápegységek fő jellemzői* ezért

— a *névleges kimeneti feszültség* (például 5,0 V);

— a *maximális terhelhetőség* (például 10 A);

— a *hálózattal szemben támasztott követelmények* (például a bemenetre adható feszültség essen 220 V + 10%—15% érték közé, a hálózati frekvencia legyen 50 Hz ±1 Hz, az elviselhető rövid idei túlfeszültség értéke maximálisan 3 kV lehet);

— a *környezeti hőmérséklettel* szemben támasztott követelmények (például a környezeti hőmérséklet maradjon 0 °C és 75 °C között);

— a *stabilitás* (a kimeneti feszültség eltérése a névleges értéktől ±1%-nál nagyobb nem lehet);

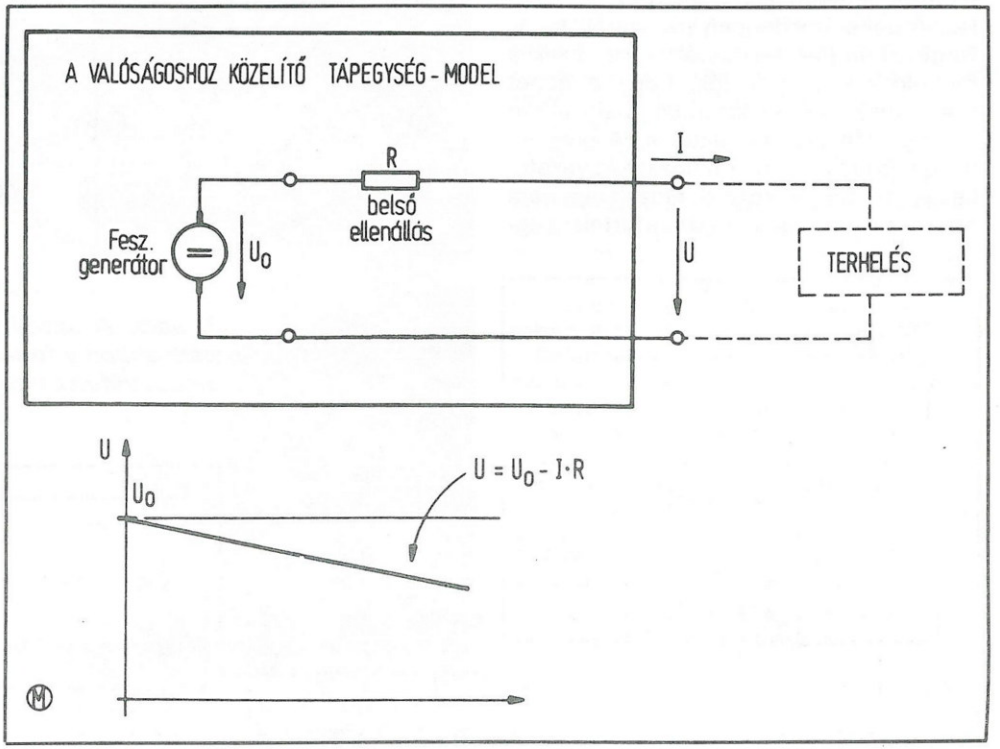
— *életvédelmi,* szigetelési követelmények stb.

Az elektronikai szakemberek a valóságos elektromos energiaforrásokat ideális elemekkel igyekeznek modellezni. Ilyen ideális — a valóságban nem létező — elem az ún. *feszültséggenerátor,* amely ugyanúgy absztrakció, mint például a matematikában a *pont* (lásd a *4. ábrát*).

A valóságos „állandó feszültségű" energiaforrások (tápegységek, galvánelemek stb.) persze nem ideálisak: kimeneti feszültségük egyaránt függ a terheléstől, másrészt feszültségük a terheléstől függetlenül is ingadozhat, függhet például a környezeti hőmérséklettől, a hálózati feszültségtől stb. A terheléstől való függést gyakran egy ideális, egyenfeszültséget szolgáltató *feszültséggenerátor* és egy *ellenállás* kombinációjával modellezhetjük (lásd az *5. ábrát*).

Előző cikkünkben bemutattuk az elektronika, a villamosságtan néhány alapvető fogalmát: az áramot, a feszültséget, a töltést, a kapacitást. Egészítsük ki ezt a sort most az elektromos *ellenállással.*

5. ábra. A valóságos tápegységek (elemek, akkumulátorok) pólusain terhelés hatására csökken a feszültség. Ezt gyakran ún. belső ellenállással modellezhetjük



Az elektromos ellenállás

Sok olyan jelenség van, melyben a két pont közötti feszültség és az egyik ponton **be**folyó és a másik ponton ezzel egyenlő, **ki**folyó áram között (legalább a számunkra érdekes feszültség/áram tartományban) jó közelítésben lineáris összefüggés van. (Lásd a *6. ábrát.) Azaz:*

$$\frac{\Delta U}{\Delta I} = \textbf{konstans}$$

Ezt a konstanst **ellenállásnak, rezisztenciának** nevezzük. Jele: **R,** mértékegysége az **ohm** (Ω), az alap mérékegységekkel kifejezve: $m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$.

Fémből készült vezetékeknél a vezetéken folyó áram nagysága igen széles feszültség/áram tartományban, a zérus pontból kiindulva, egyenesen arányos a feszültséggel. Ezért erre az esetre a „Δ" jeleket elhagyva írhatjuk:

$$I = \frac{U}{R}$$

Ez **Ohm** törvénye (lásd a *7. ábrát*).

Ohm törvénye a számítástechnika (a hardver) megértéséhez is nélkülözhetetlen, a mindennapi elektronika legegyszerűbb és egyik legfontosabb összefüggése. Segítségével az ellenállás egységét így definiáljuk: *az ellenállás egysége az* **ohm. Egy ohm** *ellenálása van annak a vezetéknek, amelyen* **1** *volt feszültség* **1** *amper áramerősséget hajt keresztül.*

A számítógépek „hőtermelése"

Már előző cikkünkben is említettük: vita folyik arról, lehet-e elvileg olyan számítógépeket tervezni, melyek működésükhöz egyáltalán nem igényelnek energiát. Engedjék meg, hogy itt befejezésül röviden még visszatérjünk a témára.

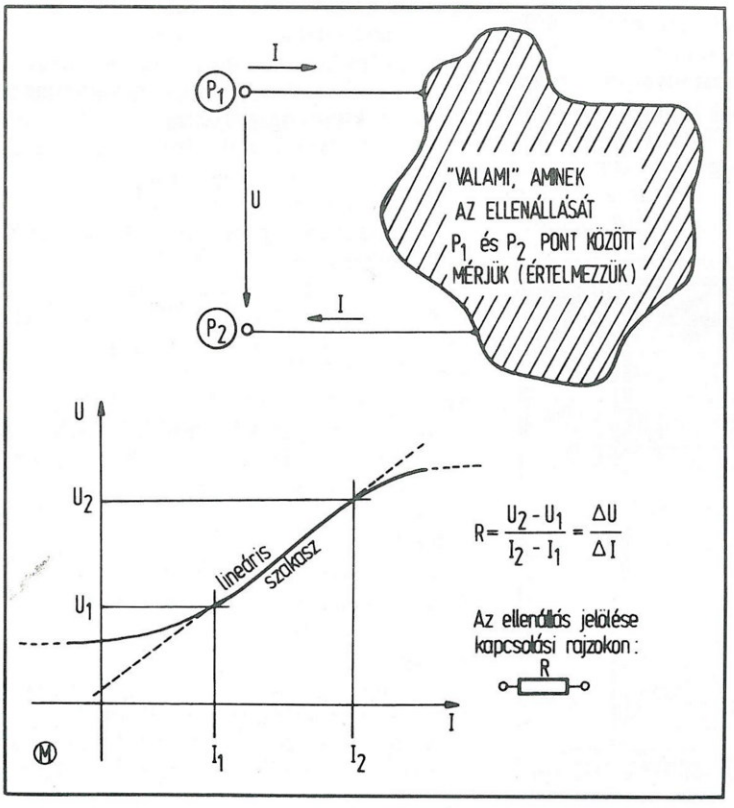
Mi lesz a sorsa annak a például kémiai energiának, melyet mai(?) számítógépeink működtetéséhez az erőművek „szabadítanak fel": ott égetnek el, ott alakítanak át hő-, majd elektromos energiává? Végső soron **hő.** Hő, amely *veszteségként* szétárad az erőmű környékén. Hő, amely ugyancsak veszteségként szétárad a távvezetékekből, az energiahálózat transzformátoraiból stb., és végül — ami ma a legnagyobb gond — hő, mely szétárad számítógépeinken (és -ből), miközben azok *belső állapota* **rendezettebbé** válik.

A számítógépek belseje tele van olyan *kapuáramköröknek* nevezett kapcsolókkal, melyeknek — mint egy ún. **ÉS** kapunak — egy kimenete és legalább két bemenete van, és a kimenet állapota (például az, hogy ott mekkora feszültség mérhető) attól függ, hogy a két bemenetére milyen jelet, mekkora feszültséget, mondjuk O voltot vagy 5 voltot adtunk-e. De már a kimenet állapotából (mai rendszereinknél!) nem tudnánk pontosan rekonstruálni, hogy ahhoz a lehetségesekből pontosan milyen bemeneti állapot (jel-) kombináció tartozott.

Külön magyarázat nélkül megemlítjük, hogy Neumann János annak idején ebből az *irreverzibilitásból* vonta le a termodinamika második törvénye alapján azt a következtetést, hogy számítógépeink az információfeldolgozáshoz *szükségszerűen energiát igényelnek (és emiatt hőt termelnek).*

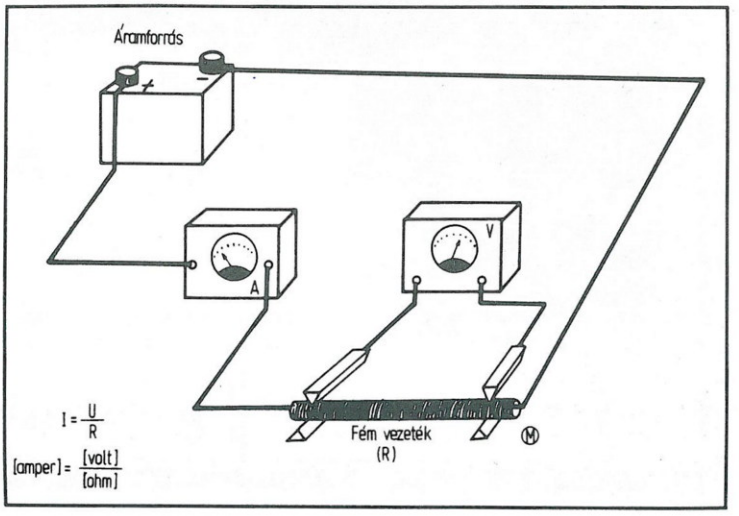
Mint már előző cikkünkben is írtuk, ez a *„termodinamikai szükségszerűségből"* termelődő hő ma sok nagyságrenddel kisebb annál, mint ami ténylegesen termelődik. De egyes előrejelzések szerint, ha a mikroelektronika olyan gyorsan fejlődik tovább, ahogy eddig, akkor a 2010-es évtized környékére eljuthatunk odáig, hogy ez már számítani fog. És akkor még elővehetik Charles Bennett (IBM) és Edward Fredkin (MIT) egymástól függetlenül felvetett ötletét azokról a kapuáramkörökről, melyek a kimenet generálása után *nem törlik* a bemeneti információt, hanem megőrzik (konzerválják) azt, és így lehetővé teszik az eredmény elérése után a folyamat *megfordítását.* (A téma közérdekűségét jelzi az angol Economist folyóirat 1989. február 11-i számában a Számítógépek hőhalála című cikk is.)

Egy ilyen *„konzerváló"* kétbemenetű ÉS-kapunak már nem egy kimenete **lesz,** hanem **három.** A két „többletkimenet" a bemeneti információt tartalmazza, őrzi meg. Lehet, hogy ma még megmosolyogjuk az ötletet, de mit fogunk csinálni a maiaknál sokkal több kaput tartalmazó morzsák (chipek) egységnyi felületén — *Neumann János szerint „törvényszerűen"* — termelődő hővel a harmadik évezred első harmadában és később? — *KE* —



6. ábra. Ha egy feszültség/áram tartományban a feszültség és az áram kapcsolatát lineárisnak tekinthetjük, akkor ezt a kapcsolatot egy konstanssal jellemezhetjük. Ezt a konstanst ellenállásnak (rezisztenciának) hívjuk

7. ábra. Ohm törvényének igazolása



Profi BASIC Borland módra

A számítógépek teljesítményének növekedése hardverük — áramköri kialakításuk — gyors fejlődésének és a jobb, hatékonyabb programok megszületésének köszönhető. A nyolcbites processzorral épített személyi számítógépek után ma már elterjedtek a 16, illetve 32 bites processzort használó számítógépek. Ezeknek a programozási nyelvei között ma is még a legnépszerűbb a hagyományosnak tekinthető BASIC nyelv. Természetesen a nagyobb teljesítményű gépeken futtatható BASIC legalább olyan fejlődésen ment át, mint a hardver.

Programozói körökben sokan szent borzadállyal tekintenek a BASIC-re, és óva intenek használatától, mert nem tesz eleget a modern programozástechnika néhány követelményének. Ennek ellenére két vezető szoftvergyártó cég, a Borland és a Microsoft kifejlesztettek, illetve továbbfejlesztettek olyan BASIC programozási nyelveket, amelyek hatékony felhasználást nyújtanak.

Mivel ezek a programok a felhasználók széles táborának készülnek, ezért a program megítélését és eladhatóságát alapvetően meghatározza az ember—gép kapcsolat kialakításának a minősége. Érdekes módon többé-kevésbé mindkét cég hasonló kialakítást tartott optimálisnak.

Melyek azok a szempontok, amelyeket a jelenlegi professzionális programok az ember—gép kapcsolatok kialakításánál figyelembe vesznek?

— Ablakos kialakítás: a képernyőn egyszerre több önálló funkciójú terület látható,
— hierarchikus menükialakítás,
— a szerkesztés a teljes ablakban történhet,
— a felhasználó választhatja meg az aktuális színeket,
— át- és visszalépés a DOS és a programrendszer között,
— környezettől függő segítő szövegek előhívhatósága,
— egér használata,
— aritmetikai processzor felhasználása,
— egyszerű és széles skálájú konfigurálhatóság (képernyő- és nyomtatótípus választhatósága),
— „barátságos" rendszer: azt csinálja, ami a legvalószínűbb,
— a menüből vagy a parancs kezdőbetűjének leütésével, vagy kurzormozgatással lehet választani,
— „forró billentyűk" használata: vannak a menü minden szintjén érvényes, a program alapvető funkcióit bárhonnan aktivizáló billentyűk.

Ezúttal a Borland cég Turbo BASIC 1.0 programját ismertetjük. Egy későbbi alkalommal visszatérünk a Microsoft QuickBasic programjának bemutatására is.

Bármely programozási nyelven programozunk is, az formailag többé-kevésbé hasonló lépésekből áll. Az első lépésben a programot egy szövegszerkesztővel meg kell írni, amit aztán le kell az adott gépi kódra fordítani, majd futtatható állapotra kell alakítani (linkelni). Végül jön a futtatás és a hibakeresés. A klasszikus, interpreter típusú BASIC nyelveknél a programírás és a futtatás, interpretálás a BASIC rendszerben ment végbe. S mivel fordítás nem volt, ezért külön fordító-szerkesztő program szolgált a helyes, már működő BASIC program lefordítására és linkelésére.

A Turbo BASIC egy összeintegrált szövegszerkesztő-, fordítóprogram és kipróbáló rendszer. (A szövegszerkesztő demói az ábrákon láthatók.) A rendszer és a felhasználó között menüvezérelt, a képernyőn kialakított, egy időben látható, több ablak teremti meg a kapcsolatot. Sajnálatos módon nem teszi lehetővé az egér használatát. Az ablakok közül mindig csak egy, az aktuálisan kiválasztott az aktív. Az ablaknak a mérete és helye a képernyőn változtatható.

A képernyő felső sorában láthatók az egyes kiválasztható almenük. Tekintsük át röviden a funkciókat.

File.
Ebben a menüpontban tölthetők be és menthetők el a programok, és itt írható hozzá valamely programrész egy másik programhoz. A tartalomjegyzék váltása, a DOS-ba való ideiglenes kilépés (Shell) is itt lehetséges.

Edit.
E menüpont választásával szerkeszthető a programszöveg.

Run.
A program futtatása.

Compile.
Programfordítás.

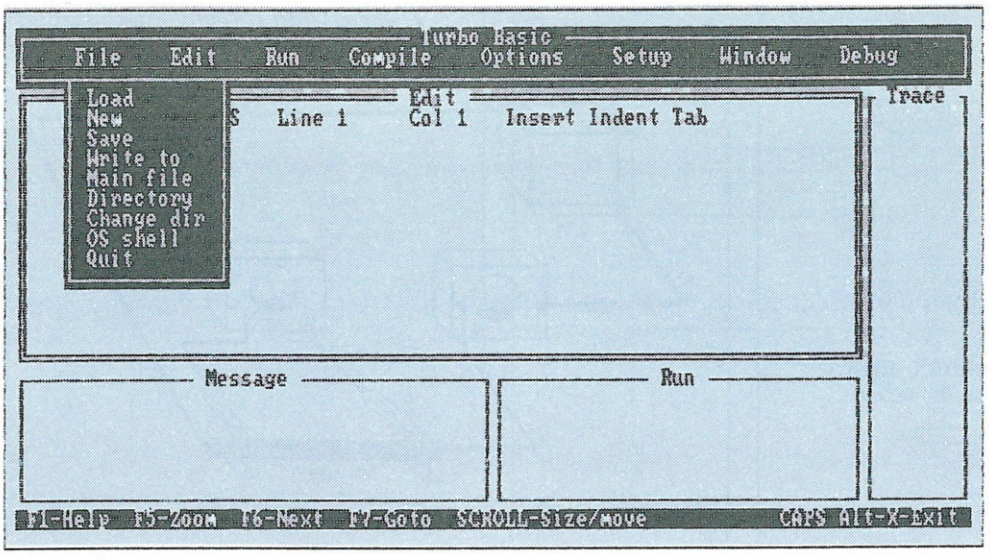
Options.
Különféle rendszerfeltételek beállítása:
— a programfordítási cél megadása; fordítani lehet egy .EXE fájlba, egy láncolható fájlba vagy a memóriába,
— a 8087-es aritmetikai processzor használata,
— a fordítás eredménye hová kerüljön,
— a billentyűzetről való programmegszakítás engedélyezése,
— a tömbhatár túllépésének figyelése,
— a túlcsordulás figyelése,
— a verem ellenőrzése,
— a fejlesztett program parancssorának megadása,
— a metautasítások megadása.

Ezekkel az utasításokkal adható meg a verem, a zene és a kommunikációs puffer mérete.

Setup.
Az ember—gép környezet beállítása, az ablakoknál használt színek kiválasztása. A rendszer által használt tartalomjegyzékek megadása.

Window.
Az ablakokkal kapcsolatos tevékenységek: ablakok megnyitása, zárása, méreteiknek megadása.

Debug.
Hibakeresés beállítása. Trace: nyomkövetés. Runtime error: .EXE programban való hibakeresés. A Turbo BASIC parancs üzemmódjában négy ablak látható:



EDIT.
A BASIC forrásprogram megszerkesztését: leírását, javítását, módosítását teszi lehetővé.

RUN.
Ebben az ablakban láthatók a futó program által létrehozott ábrák és szövegek, azaz eredmények.

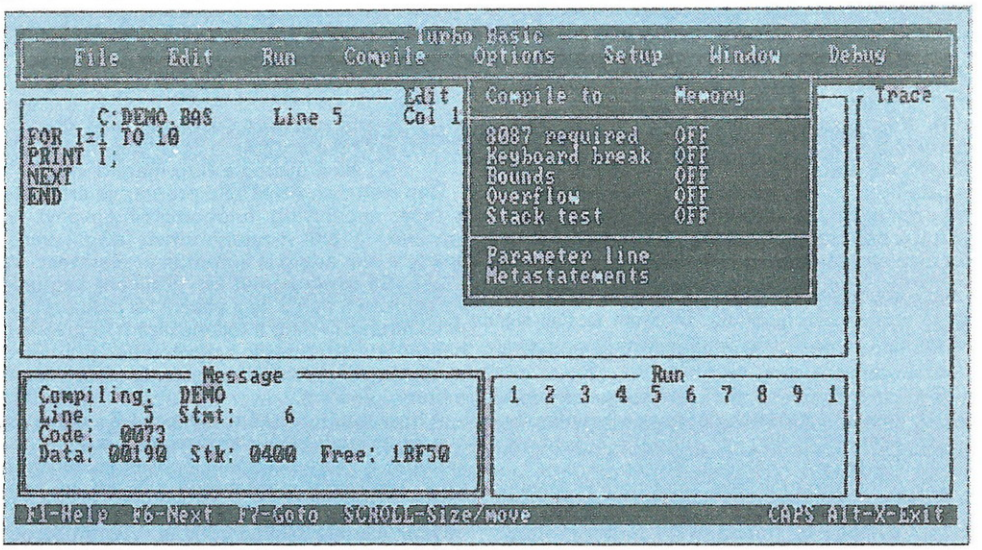
MESSAGE.
A rendszer által szolgáltatott információt mutatja, amit a fordításkor kapunk. Itt láthatjuk a lefordított program jellemzőit.

TRACE.
Ha engedélyezett, az éppen végrehajtott (aktuális) utasítás sorszámát vagy címkéjét mutatja.

Nézzük a Turbo BASIC programmal való programfejlesztés lépéseit. A File menü felhasználásával betöltjük a forrásprogramot vagy az Edit menüben megszerkesztünk egy új forrásprogramot. (Természetesen más szövegszerkesztővel is megírhatjuk a programot.) A következő lépés a program fordítása, mégpedig a Compile menü aktivizálásával (C billentyű megnyomása). A fordítás végén megjelenik a program neve, a programot alkotó sorok száma, a lefordított program kód-, adat- és veremszegmensének mérete. Az R (Run) billentyű megnyomásának hatására a program futni kezd. Ha a billentyű megnyomása előtt nem végeztünk fordítást, a program automatikusan végrehajtja azt.

A Turbo BASIC meglepő módon nem használ .OBJ fájlokat, hanem azonnal végrehajtható kódot generál. Ennek a megoldásnak az előnye az egyszerűségében rejlik. Nem szükséges a program és a BASIC segédkönyvtárak összelinkelése, hanem automatikusan futtatható .EXE fájlokat hoz létre. A nyilvánvaló hátránya viszont az, hogy a programokban használt közös részek ismételten a tárgyprogramba kerülnek, valamint hogy egyszerű módon nem szerkeszthetők hozzá külső .OBJ fájlok a programjainkhoz.

Ha egy hiba bekövetkezik a fordításkor, az Edit ablak lesz aktív, és a kurzor a hibára mutat a szövegben. Ha a programfutás alatt fordul elő hiba, szintén az Edit ablak lesz aktív, megmutatva a hiba helyét és a hiba kódját. Természetesen ezek csak akkor igazak, ha a Turbo BASIC fejlesztői környezetében dolgozunk. Ha egy már lefordított és önállóan elindított .EXE fájl futása közben lép fel hiba, a rendszer a hibaszámmal és a programszámláló értékének megadásával válaszol. Az ilyen információk felhasználásával a forrásprogramban is megkereshető a hiba. A forrásprogramot a Turbo BASIC rendszerbe töltve, a Debug menüt aktivizálva és a Runtime error almenüt kiválasztva, betölthetjük a hiba esetén megjelent értékeket. A fordító újrafordítja a forrásprogramot és megáll, ahol a forráskódhoz tartozó programszámláló értéke megegyezik a begépelt értékkel. Ilyképpen a hiba helye azonosítható.



A Debug menü másik opciója a Trace parancs. Aktivizálásakor a Trace ablakban megjelennek a forrásprogram sorszámai és címkéi. A Trace módban a program lépésenként futtatható.

A Turbo BASIC szövegszerkesztője nagyon hasonlít a Turbo Pascalnál használatoshoz, mindkettő a WorldStar szövegszerkesztő fortumátuma szerint dolgozik. Teljes képernyős szerkesztést tesz lehetővé, és sok olyan tulajdonsága van, amely túlmutat egy tipikus programszerkesztő szolgáltatásain. Lehetséges karakter, szó, sor és képernyőnyi tartalom mozgatása, másolása és törlése. Blokkokra vonatkozó parancsokkal kijelölt szövegterületek manipulálhatók.

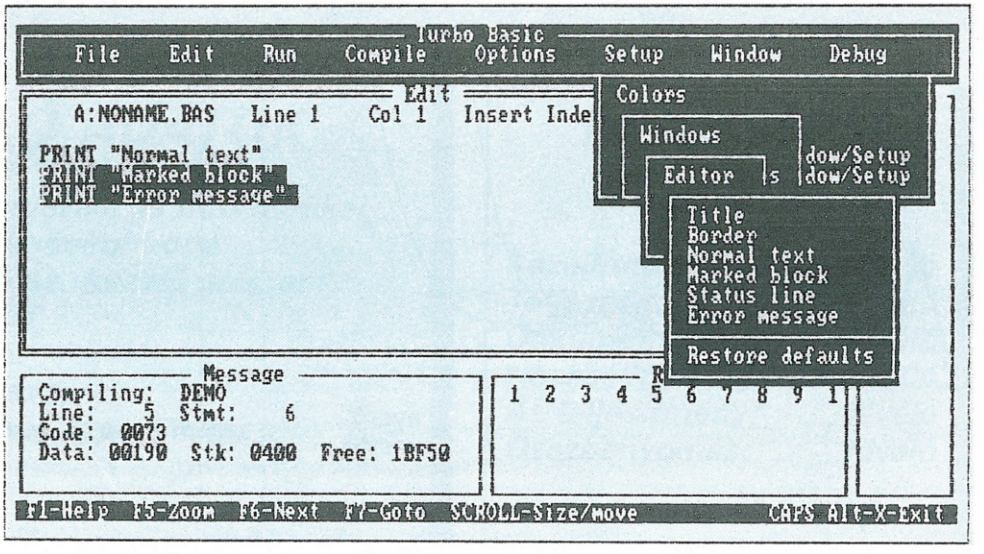
A Turbo BASIC szövegszerkesztője minimálisan 64 kbájtnyi szövegeket képes kezelni. Az $INCLUDE direktíva felhasználásával azonban nagyobb méretű programok is fordíthatók. A főprogramban lévő $INCLUDE direktíva utáni fájlnév adja meg a fordítónak azt a fájlmodult, amit a fordításkor a főprogramba be kell fordítani. A $INCLUDE modulok maguk is újabb $INCLUDE modulokat tartalmazhatnak, hatos mélységig.

A Turbo BASIC nyelv az IBM PC szabványos BASICA nyelvével kompatibilis, de annak jelentősen továbbfejlesztett változata. Specialitásai:

— a programsorokat nem kötelező sorszámozni, a sorokra való hivatkozásokat címkékkel valósíthatjuk meg;
— a számításoknál és a számábrázolásoknál az IBM PC-kben elhelyezhető Intel 8087-es aritmetikai koprocesszor által nyújtott előnyöket figyelembe vették. Ha a gépben van koprocesszor, akkor a Turbo BASIC ezt használja a számításoknál, ha nincs, szoftverben szimulálja a működését. A létrehozott .EXE programoknál is figyelembe vehetjük a koprocesszor felhasználását;
— használhatók a strukturált programozásra jellemző DO/LOOP, illetve SELECT CASE blokkok;
— a programban használt tömbök alsó határa megadható;
— kezelhetők tetszőleges bináris és szöveges adathalmazok (fájlok).

A Turbo BASIC tehát sok előnyös tulajdonságot kínál azoknak, akik a BASIC nyelvet használják. Önállóan futtatható programok készíthetők vele, sokoldalú, jól konfigurálható felületet nyújt a felhasználóknak. Jó szövegszerkesztője van, gyors fordítást tesz lehetővé, jelentősen kibővíti a klasszikus BASIC nyelv lehetőségeit.

—KL—



A figuratípusok jutalompontjai

A Northwestern Egyetemen David Slate, Larry Atkin és Keit Gorlan által kifejlesztett Chess 4.5 sakkprogram jutalompontot ad az egyes figurák, figuratípusok elhelyezkedésére, ha azok bizonyos kritériumoknak megfelelnek. A futók bónuszát a Chess 4.5 a következő képlet alapján számítja:

Fb = P + EM + OM + D + OB + OW

ahol

- P = a futó pszeudolegális lépéseinek a száma
- EM = a futóátlón lévő ellenséges figurák anyagi értékének összege a következők szerint: gyalog = 0, huszár = 3, futó = 3, bástya = 5, vezér = 9, király = 10
- OM = a futóátlón lévő saját figurák anyagi értékének összege a következők szerint: gyalog = 0, huszár = 3, futó = 3, bástya = 5, vezér = 9, király = 0
- D = annak az átlónak a hosszánál eggyel kisebb érték, amelyiken a futó áll. Tehát ha a futó c3-on áll, akkor 7
- OB = gyalogakadály-értékelés. Az előző alkalommal részletesen ismertettük
- OW = minden ellenséges gyalog esetében, ha az a futó előtt az átlón helyezkedik el, a következőképpen számítjuk:

  −5 ha a sor < 4,
  +1 ha a sor ≥ 4,
  +1 ha a gyalog a futó mögött van

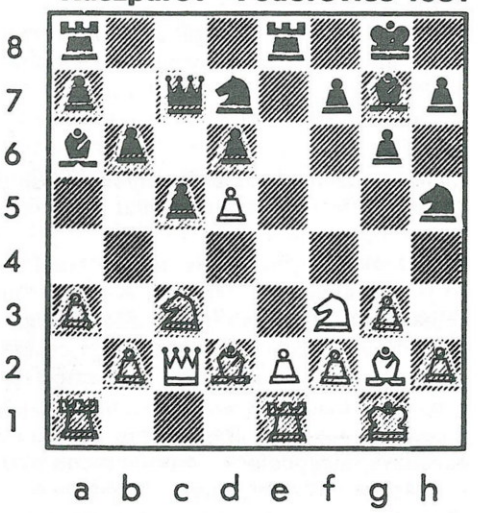
Dap Hartman ANALYSE programja értékelte a most ismertetett futóbónusz-függvényt is, ugyanazt a 832 mesterjátszmát alapul véve, amelyre már eddig is építettük az elemzést. Az ANALYSE programmal két grafikont kaptunk. Az egyiken a nyerő és a vesztő fél bónuszpontjának értéke látható a féllépésszám függvényében *(1. ábra)*, a másik függvény pedig a nyerő és a vesztő játékos bónuszának különbségét mutatja *(2. ábra)*.

A huszár bónuszaként a Chess 4.5 program a huszár pszeudolegális lépéseinek a számát tekintette. A pszeudolegális lépések esetünkben azok, amelyeket megtehetne a figura, ha nem vennénk figyelembe, hogy ellépve helyéről, királyunk esetleg sakkba kerülne. Viszont az eddigiekben használt definícióval ellentétben, nem számít a program készítői szerint pszeudolegális lépésnek az, ha a huszár olyan mezőre lépne, ahol saját figurája áll - tehát védi azt. Példaként vizsgáljuk meg a sakktáblán látható hadállást, amely a Kaszparov—Fedorovics parti 28. féllépése után jött létre, az 1981-ben játszott grazi mérkőzésen. A világos huszárok mobilitása 10, a sötéteké 7.

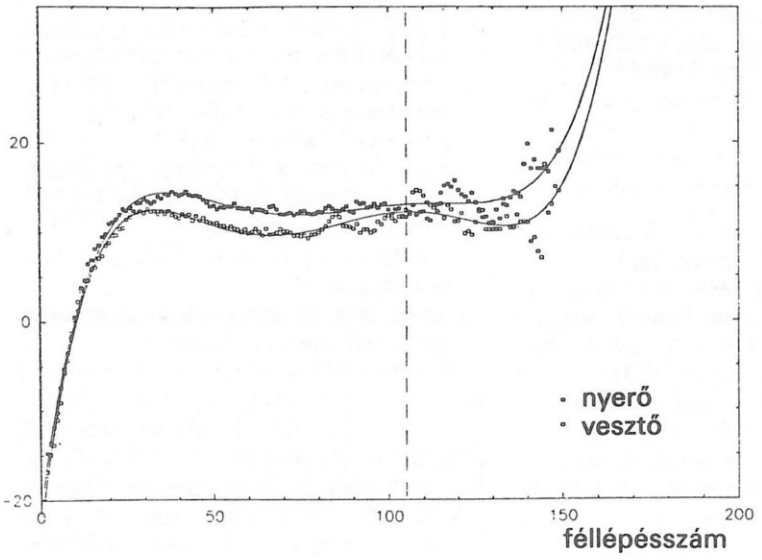
A *3.* és *4. ábra* a huszárok mobilitásának értékét szemlélteti a nyerő és vesztő fél esetében.

Kovács P. Attila

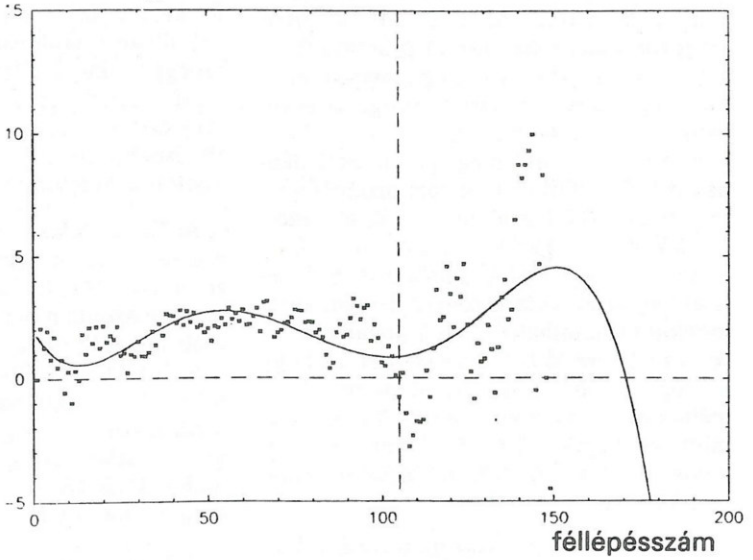
Kaszparov—Fedorovics 1981



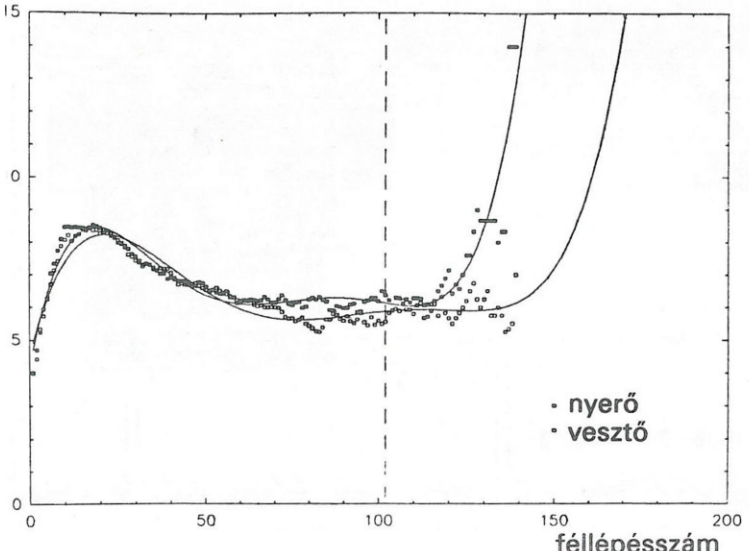
1. ábra. A nyerő és a vesztő fél futóbónuszának értéke



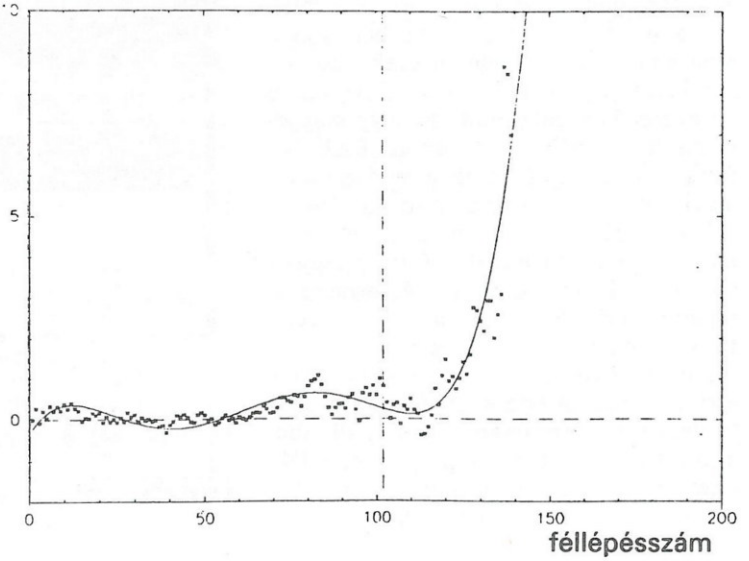
2. ábra. A nyerő és a vesztő futóbónuszának különbsége



3. ábra. A huszár bónusza



4. ábra. A nyerő és a vesztő játékos huszárbónuszának különbsége



PROGRAMTERMÉK

Most már lehet szellemeskedni!

Régóta nélkülözött szoftvereszközhöz juthatnak hozzá a TV-Computer tulajdonosok. Megszületett egy testre szabott Sprite Editor, magyarul szellemgrafikai programcsomag, ami más gépekre már tömegméretekben létezik. Ebben a hónapban ezt a szoftvert faggatjuk.

A közönség a szellemgrafikát elsősorban a játékszoftveriparból ismeri. Ez a technikai lehetőség azonban nemcsak a játékprogramokban használható, általában támogathatja az animációt. Az animátor úgy dolgozhat, mint egy rajzfilmtervező, aki a képet több, átlátszó fóliára rajzolt ábrából, mozzanatokból rakja össze. A szellemgrafikai programcsomagokkal éppen ilyen jellegű mozgásmozzanatok hozhatók létre. A gépek eredeti grafikai lehetőségeivel ugyanis rendszerint nem könnyű megoldani egy képecske mozgatását akár csak egy kicsit is bonyolult háttér előtt. Néhány számítógépnek, mint például a népszerű Commodore 64-nek vannak szellemgrafikai eszközei. Tulajdonosaik látszólag könnyű helyzetben vannak. A C64 sprite-jainak használata a valóságban mégsem olyan egyszerű. Még az ilyen gépek sem nélkülözhetik tehát a jó szellemgrafikai szoftvert. Megjegyezzük, hogy az első olyan hardverchip, amelyik szellemgrafikát valósított meg, a Texas Instruments cégé volt.

Mi a helyzet a TV-Computerrel a szellemgrafikát illetően? Sajnos a gépbe hardvertámogatást nem építettek be. Ezért mindent szoftverrel lehet csak megvalósítani. Az elv igen egyszerű: a mozgó ábra alatti területeket ki kell menteni a képernyőtárból, és a képecske adatait kell odamásolni. Amikor a képet el akarjuk mozdítani, akkor előbb vissza kell állítani a háttéradatokat, majd az elmozdulás utáni helyről kell kimenteni a hátteret, és újra feltenni a képre a szellemet. A gyakorlatban vigyázni kell arra, hogy a kép adatain nem lehet akármikor büntetlenül babrálni, mert ha a képernyőfrissítő hardver éppen akkor jár azon a területen, akkor nem garantált, hogy a képernyőn az általunk elképzelt kép jelenik majd meg. A jelenséget villogásnak hívják. A szellemgrafikai szoftvereknek az itt említett problémákat mind meg kell oldaniuk egyetemben a kényelmes használhatósággal, ami mint mondottuk, az olyan gépek eredeti szoftverénél sem teljesül, amelyeknek hardver sprite-jai vannak. Az alapelvek egyszerűsége persze nem jelenti azt, hogy olyan könnyű volna szellemgrafikai programcsomagokat írni.

Mielőtt rátérünk a vizsgált termék adataira, elmondjuk , hogy a terjesztőt, a Tudományszervezési és Informatikai Intézetet is elkapta az átszervezési járvány. Az oktatási szoftver forgalmazását vállalati gazdálkodási rendben működő szervezeti egységük, a kereskedelmi iroda vette át. Munkájukhoz sok sikert kívánva, nézzük legújabban forgalmazott terméküket.

A programcsomagot szerencsére mágneslemezen lehetett kipróbálni, ami elég kedvező, főleg ha a sok C Plus/4-en elszenvedett kazettaolvasási zűrre emlékszem. Kezelése elég egyszerű. A program betöltése után rögtön kapunk egy kis ízelítőt a lehetőségekből. A képen egy szemüveges ürge animált ábrája jelenik meg. Az animáció nem villog, tehát mindenben teljesíti a szellemgrafikáról elmondott szakmai igényeket. Kérdés, hogy a használatot is elég kényelmessé teszi-e. Az a véleményünk, hogy igen. A termékkel kapcsolatban biztosan lesznek majd, akik még többet várnának tőle, de a pillanatnyi helyzetben annak is örülni kell, hogy egyáltalán van már ilyen szoftver a TVC-re. Azonkívül valószínű, hogy a szerző, Canjavec Attila sem ül majd sokáig a babérjain, és — saját és mások elégedetlenkedéseire reagálva — még tovább fejleszti a terméket. A programcsomag késői megjelenése annyi előnnyel járt, hogy a piacon előtte megjelent hasonló szoftverek kitaposták az utakat. Ezért nem csoda, hogy a program az ikonvezérelt modern szoftverkategóriába sorolható.

A program használatakor némi lelki válságot okozott a TVC lassúságának betudható időigényesség. Ezen sajnos aligha lehet segíteni. A program egyébként kevés kezelési fogyatékosságot mutat, de volt olyan érzésem, hogy azért még lehetne javítani rajta.

A szerző a képi megjelenítésre eléggé ügyelt. A képfelület beosztása nem agyonnyomorított. Az eredeti színválasztás illeszkedik a TVC lehetőségeihez, de a felhasználóknak alkalmuk nyílik a kísérletezésre. (Rosszabb színösszeállítást hamar el lehet érni!)

A program főbb funkciói: rajzolás, festés, forgatás (csak 90 fok), tükrözés, visszatöltés, másolás, radírozás, kimentés/betöltés, animáció(!), méretválasztás (elég dinamikus!), NOT/AND/XOR/OR grafikai választás, „manó" azonosítószám-módosítás, tárba helyezés és visszamásolás onnan. Különösen fontos eleme a programnak az animátor. A képsorozatot ekképpen már készítés közben kipróbálhatjuk, amiből idejekorán kiderülhet, hogy elég mozzanatból tettük-e össze a cselekményt, nem túl „darabos-e" a mozgás. A mozgás darabosságát persze olykorolykor éppen erőltetni is akarjuk. Fontos a programokba való beépítéshez adott eszköz, a sprite-kezelő segédprogram, ami a hétköznapi BASIC programozóknak jó tippeket ad a „manók" programokba való beépítésének lehetőségeire.

A programcsomag szolgáltatásait a hétköznapi használatban nehéz kimeríteni, ha — mint már emlegettük — lesz is még az alkalmazóknak vele kapcsolatos hiányérzetük. Sajnos nem volt elég idő a játszadozásra, pedig nagyon kedvemre való volt a programcsomag vizsgálata.

A program nemcsak játékszoftverként használható, hanem segítheti az oktatóprogramok mozgalmassá tételét anélkül, hogy a pedagógusoknak grafikai szoftveressé kellene kinőniük magukat.

Jó lenne, ha e programcsomag minden iskolába eljuthatna, ahol TVC található. A termék által nyújtott teljesítményhez képest alacsony árfekvése remélhetően lehetővé is teszi ennek az óhajnak a teljesítését.

Zsadányi Pál

ÖSSZEFOGLALÓ ADATOK

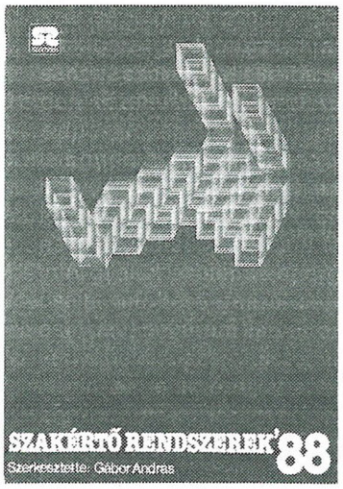
Forgalmazó:	**Tudományszervezési és Informatikai Intézet Kereskedelmi Iroda**
Terméknév:	**Sprite-készítő és -kezelő program**
Szerző:	**Canjavec Attila**
Géptípus:	**TV-Computer**
Hordozó:	**hajlékonylemez**
Dokumentáció:	**7 oldalnyi tömör, precíz leírás**
Ár:	**312 Ft**

MINŐSÍTŐ ADATOK

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

KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

Szakértő rendszerek '88
Ismeretalapú információfeldolgozás Magyarországon
Szerk.: Gábor András (Budapest, 1988. SZÁMALK, 500 oldal. Ára: 380,— Ft)



A mesterséges intelligencia kérdéseivel foglalkozó kutatók számának növelésével és a gyakorlati alkalmazások számának emelkedésével együtt növekszik a szűkebb és tágabb szakmai közönség érdeklődése. Ezt az igényt igyekszik ez a tanulmánykötet kielégíteni. A kötet a hazai kutatások és alkalmazások helyzetéről, színvonaláról nyújt tájékoztatást. Elsősorban annak az olvasóközönségnek szól, amelynek van némi számítástechnikai műveltsége, jártas valamely alkalmazási területen, és a mesterséges intelligencia eszközön alapuló fejlesztésekhez affinitása van.

A könyv első részében a mesterséges intelligencia áttekintése, ezen belül is hangsúlyosan a szakértő rendszerekkel kapcsolatos ismeretek kaptak helyet. Az áttekintés, összefoglalás a fogalmak, a terminológia tisztázásán túl a hazai törekvések, eredmények nemzetközi mércével való összevetését is tartalmazza. Az ismeretalapú információfeldolgozás mibenlétének bemutatása után a hazai fejlesztői eszközökkel ismerkedhet meg az olvasó.

A második részben konkrét alkalmazások, illetve azokhoz fűződő szűkebb körű áttekintések találhatók. Az alkalmazások jórészt a szakértő rendszerek körébe sorolhatók, mivel ezen a területen van viszonylag a legtöbb eredményünk. Ám, ha nem is a teljesség igényével, helyet kaptak a mesterséges intelligencia más területeit érintő — a mesterséges beszédfunkciót, a természetes nyelv és a számítástechnika kapcsolatát taglaló — tanulmányok is.

A szövegben előforduló fogalmak megmagyarázására, az összefüggések megértésének elősegítésére a kötetet záró kislexikon szolgál. A kislexikonban megtalálható az általánosabb, több helyen is előforduló kifejezések, fogalmak értelmezése.

Nagy, T. — Gault, D. — Nagy, M.: Így készül a szakértői rendszer (Budapest, 1988. Novotrade, 248 oldal. Ára: 388,— Ft)

A kötet általános betekintést nyújt a számítógépes szakértői rendszerek működési mechanizmusába, és konkrét példákon keresztül azt is bemutatja, hogy maga az olvasó hogyan készíthet ilyen rendszereket.

A könyv megértéséhez nincs szükség sem matematikai, sem számítástechnikai ismeretekre. A szakértői rendszerek működésének, felépítésének és elkészítésének lényege magának a könyvnek a tanulmányozásával is elsajátítható. A megértést nagymértékben segíti egy külön is megvásárolható mágneslemez, amely a könyvben ismertetett MICRO—PS rendszergenerátort és a segítségével elkészített kis szakértői rendszert tartalmazza.

Számítástechnikai feladatok 2000-ig. I. Szerk.: Heténvi Pálné (Budapest, 1988. OMIKK, 364 oldal. Ára: 130,— Ft)

Az egyetemi és középiskolai oktatók által összeállított kétkötetes feladatgyűjtemény első kötete olyan témaköröket tartalmaz, amelyek a középiskolai számítástechnikai tananyaghoz igazodnak, de közöttük néhánynak a megoldása jóval e szint feletti tudást igényel, néhány feladathoz pedig általános iskolai ismeretek is elégségesek.

A könyv fejezetei a tartalomra utaló kulcsszavak összefoglalójával kezdődnek, ismertetik a feladat megoldásához szükséges ismereteket, és bibliográfiát is közölnek a legfontosabb szakkönyvekről. Ábrák jelzik a feladatok nehézségi fokát.

A kötet egy általános programozási feladatgyűjtemény, és így egyetlen programozási nyelvhez sem kötődik. Mivel azonban a BASIC nyelv a legelterjedtebb és legnépszerűbb programozási nyelv a mikroszámítógépet használók körében, ezért a feladatok nagy része ehhez a nyelvhez kapcsolódik. Az olvasó a függelékben az egyes feladatokhoz megoldási javaslatokat is talál.

Röntgensugaras litográfia

A kutatók szerint már szinte teljesen kimerítették a jelenlegi áramkör-maratási technológiák lehetőségeit. A jövő: a röntgensugarak alkalmazása a chipgyártásban. Várhatóan az 1990-es évek közepére a jelenlegi verseny győztesei válnak az évi 100 milliárd dolláros forgalmat bonyolító chippiac legfejlettebb részének irányítóivá. A vesztesek viszont kénytelenek lesznek szembenézni ipari hatalmuk és katonai erejük csökkenésével.
Ismerve a tét nagyságát, az USA kormánya 25 millió dollárt állított be a jelenlegi költségvetésben a röntgentechnológia fejlesztésére. Ez azonban csak töredéke annak, amit a szakértők szerint Japán vagy akár a nyugat-európai országok költenek erre a célra. A röntgensugaras litográfia néven ismert technológia célja az óriási részecskegyorsítók, a szinkrotronok alkalmazása a minden eddiginél finomabb félvezető áramkörök kialakítására.
Míg a ma legkorszerűbb morzsák mintegy milliónyi áramkört tartalmaznak, a röntgensugarak segítségével előállított chipek akár egymilliárdos áramkörsűrűséggel is dicsekedhetnek majd. Ezzel a sűrűséggel a chip teljesítménye és sebessége hatalmas mértékben növekedhet, mivel lényegesen kevesebb idő kell a jelek áthaladására a szorosan elhelyezkedő komponensek között.

Winchester kártyán

40 Mbájtos merevlemezes tárolót kínál a Computer 2000 cég. A Hardcard 40 a többi kártyához hasonlóan dugaszolható a számítógépbe, a meghajtó és a hozzá tartozó elektronika is ugyanarra a lemezre van szerelve. Az új winchester használható az IBM PC, XT, AT és a PS/2 család modelljeiben, valamint a kompatibilis gépekben is. A két 3,5 hüvelykes merevlemezt tartalmazó meghajtó négy író-olvasó fejjel működik, hozzáférési ideje 35 ms. Áramkimaradásnál a fejek automatikusan parkolópályára kerülnek. A meghajtó kikapcsolt állapotban 100 g, üzem közben 10 g terhelést visel el, ezért a hordozható modellekben is alkalmazható.

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

ADOK

Amigások! A legújabb programok olcsón, garanciával eladók. Érdeklődni lehet: Keresztes Gábor, Budapest, Laky-köz 11. 1142. Tel.: 643-452

Atari 800 XL (64 k) + 1050 tip. Drive (180 k-ra bővítve) + TV + epromégető + 80 db lemez felhasználói- és játékprogramokkal + szakirodalom eladó. Irányár: 35000 Ft. Nagy Gábor, Jászberény, Gács L. u. 14. IV/12. 5100

Commodore 16-os számítógép eladó magnóval, gépkönyvekkel, sok programmal. Makó, Beloiannisz u. 128/A. 6900

C16-os gépemet 1 joystickkel, magnóval, kb. 70 játékprogrammal, szakirodalommal eladnám. Ár 12 000 Ft. Érdeklődni: 641-568-as telefonon lehet (Bődi Zoltán)

C Plus/4 számítógép magnóval, programokkal eladó. Tel.: 06/36/25-648 (kizárólag szombat-vasárnap 7 és 18 óra között.

C16, C Plus/4 programokat adok és cserélek, kazettán. Listát kérek és küldök. Agárdi Tibor, Kecskemét, Dankó P. u. 37. 6000

C64-re 1987/88-as programokat adok (20 Ft/db) és cserélek, kazettán és lemezen. Cím: ifj. Varga Péter, Nyíregyháza, Óvoda u. 49. 4400

C64-re játék-, demo-, felhasználói programokat adok - a régebbi nívós programoktól a legújabb, 1989-ben készültekig - lemezen és kazettán (20-25 Ft/db). Kérésre listát küldök! Csere is elképzelhető új, színvonalas játékok esetén. Valent Gábor, Nyíregyháza, Északi krt. 21. 4/17. 4400

C64-hez hangdigitalizáló készülék, RESET-SUPER Reset kapcsoló, C64-es kazettás egységhez hardveres másolókészülék eladó. Válaszborítékért ismertetőt küldök. Tárkány László, Szeged, Rózsa F. sgt. 128/a. 6726

C64-es programokat adok és cserélek kazettán (10 Ft/db). 600 db programról

Oktatási tervek a szomszédban

Csehszlovákiában is egyre nagyobb figyelmet szentelnek az iskolák korszerű technikai felszerelésére. Erre 1981—85 között például kétmilliárd koronát fordítottak. Ennek döntő részét — az NDK-beli gyakorlatnak megfelelően — az iskolák zsebszámológépekkel való felszerelésére használták fel. A program keretében az általános iskoláknak 32 ezer darab, a középiskoláknak pedig 23 500 darab számológépet juttattak. Azóta a korszerűbb számítástechnikai eszközök, mikroszámítógépek beszerzésére, oktatótermek kialakítására, oktatóprogramok fejlesztésére fordítanak egyre nagyobb figyelmet. A tervek szerint 1990 végéig az általános és középiskolákba negyvenezer 16 bites mikroszámítógépet szállítanak. Hivatalos statisztika nem lévén — szakértői becslések szerint — a 16 bites mikroszámítógépek állománya Csehszlovákiában 1988 végén még nem érte el tízezret, amiből az általános és középiskolákban alig néhány tucat üzemelt, elsősorban a tagozatos iskolák szaktantermeiben.

Nyugati vészjelzés, ami ránk is vonatkozik

Az NSZK szabadalmi hivatalának elnöke több évet összefoglaló jelentésében olyan tényekre hívja fel a figyelmet, amelyek rendkívül nyugtalanítók. A nagyszámítógépek terén a 12 851 japán szabadalmi bejelentéssel szemben 999 USA-beli és 357 NSZK-beli bejelentés szerepel. A mikroelektronikában a 17 964 japán bejelentés mellett 771 az USA-beli és 291 az NSZK-beli. A jövő információs társadalmának kiépítését megtakaríthatjuk, ha továbbra is csak beszélünk és semmit sem teszünk — jelentette ki ironikusan az elnök. Az adatok nagy részét már ma is amerikai adatbankokból szerezzük be, amiért fizetni kell, ahelyett, hogy kiépítenénk saját adatbankjainkat. Ehhez azonban sem a bonni minisztériumok, sem az ipar nem ad pénzt. A megfelelő adatbankok létesítése meggyorsítaná a szabadalmi ügyek intézését. Enélkül azonban az ügyek megfulladnak a papírhegyek alatt.

DOK—VESZEK —CSERÉLEK

listát küldök. Tóth Kornél, Nagykálló, Ady út 28. 4320. Tel.: 346.

C64-es (új) programokat adok-veszek-cserélek, kazettán. Varga Péter, Letenye, Farkas J. u. 1. 8868

C64, floppy, joystick, hangdigitalizáló, 40 lemez szoftver ára 31 200 Ft. Levélcím: Kiss Lajosné, Miskolc, Vörösmarty u. 68. 3530

C64-re közel 1000 db felhasználói- és játékprogram lemezen, kazettán átlag 16 Ft/db áron eladó. Kérésre listát küldök. Cím: Sasvári Gábor, Lenti, Petőfi út 33. 8960

Commodore 128-as számítógép 500 db jó programmal, magnóval, szakkönyvekkel együtt vagy külön olcsón eladó. Érdeklődni, vagy árajánlatot tenni a következő címre kérek: Cseh Róbert, Szentes, Nagygörgős u. 31. 6600

Commodore VC 1541 II. floppy vámkezelten, originál csomagolásban eladó. Jobbágy Erika, Érd, Intéző út 15-17/A. Fsz. 2. 2030

Datasettesek figyelem! IC-s másolókészülékek eladók! (2 Datasett közti adatcserét tesz lehetővé!) Kérjen tájékoztatót! Tóth Lajos, Székesfehérvár, Vörös-hadsereg út 82. 8000

Enterprise programokat adok, veszek, cserélek - kiválló minőségben. Listát kérek! Egy héten belül válaszolok. Több, mint 250 programom van. Kiss Ernő, Hódmezővásárhely, Éva u. 12. III/11. 6800

Fordításokat készítek a **64'er Magazinból**. Több mint 3000 oldalnyi kész anyag! Kedvező árak. Textomat+, GEOS 2.0, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Budapest, Pf.: 400. 1446. Válaszborítékot kérünk!

JOYSTICK SZERVIZ a Flórián Áruházhoz közel! Javítás, magnófej-beállítás. C64 játékprogramok kazettán és floppyn (10 Ft/db). Bp. III. Kerék u. 36. IV/24. Hétfőn és szerdán 17-től 19-ig.

Eladó egy alig használt **Commodore 1351-es MOUSE.** Érdeklődni 15 óra után telefonon: 688-558.

MO8X kompatibilis barkács számítógép eladó 1 db 8"-os 256 k meghajtó, 64 k memória, matematikai processzor, soros, párhuzamos kimenet, teljes szoftver és hardver dokumentáció - 20 db mágneslemez Filetóth Attila, Budapest, Sárfű út 13. VI/20. 1156

ZX-Spectrum (48 k) alig használt (nagy billentyűs) + Kempston illesztő + szakirodalom + sok játék- és felhasználói program eladó. Irányár 9000 Ft. Sztárcsevics Csaba, Lenti, Kossuth u. 47. 8960

VESZEK

Commodore 1541 vagy **1551**-es lemezegységet veszek. Lajtos Máté, Debrecen, Martonfalvi u. 14. Fsz. 2. 4032

ELITE I-II-t keresem C64-re. Rácz, Veszprém, Felszabadulás út 87/B. Fsz. 1. 8200

ZX81 vagy **Primo** számítógépet, valamint hozzákapcsolódó szakirodalmat vásárolnék. Ajánlatot levélben az alábbi címre: Tihanyi Sándor, Kisszállás, III. ker. 12. 6421 vagy telefonon a 77/21-166/234 számra kérek.

Vásárolnék üzemképtelen, bontásra szánt számítógépet, perifériákat, tartozékokat. Ajánlatokat levélben kérek. Cím: Simon Sándorné, Hódmezővásárhely, Tuhutum u. 9. 6800

Vennék **Atari 800 XL**-hez magnót és programokat kazettán, vagy cserébe kívánság szerint elektronikus készülékeket építek. Kovács Tibor, Kartal, Rákóczi út 38. 2173. Tel.: 225-026/2

CSERÉLEK

Amiga 500 tulajdonosokat keresek információ és programcsere céljából. Listát kérek, küldök. Bárdos Ferenc, Kalocsa, Miskei út 20. 6300

C16 magnóval, joystickkal, sok könyvvel és 100 db játékkal 15 000 Ft-ért eladó. Érdeklődni lehet levélben: Trencsényi Zoltán, Mátészalka, Hajdu köz 5. 4700

C16, C Plus/4 és C64-es programjaimat kínálom fel cserére, szalagon és lemezen! Főleg oktató- és játékprogramok érdekelnek, lehetőleg listát kérek! Várom a régi cserepartnereim jelentkezését is! programokkal + dokumentáció. Miron Gyula, Debrecen, Karácsony Gy. u. 9. II. 16. 4029

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

Spectrum (48/128 k-s) színvonalas játékok és felhasználói programok - küldött kazettára - 15 Ft/db áron eladók. Válaszborítékért katalógust küldök. Kiss Henrik, Budapest, Határ u. 103. 1213

Spectrumhoz (ZX81-hez is) Alphacom 32 hőnyomtató 4 tekercs eredeti papírral; ZX81 HW-SW irodalom; BIT LET 1-51; Mikrovilág 1985-86-87-88; Mikroszámítógép Magazin 1986-87-88 számai eladók. Veres Sándor, Miskolc, Oszip I. u. 14. I/1. 3529

Spectrum programokat eladok (3-75 Ft), megfelelő partner esetén cserélek. Régi és új egy helyen. Válaszborítékért ingyenes katalógust küldök. ifj. Hack József, Ajka, Pálma u. 19. 8400

Spectrum programok eladók, 5 Ft/db áron. 866-447

Spectrum megkopott (fém) fedőlapját olcsón, gyorsan felújítom. Képiró Róbert, Budapest, Imre u. 5. II/14. 1093. Tel.: Háromhatvennyolc - nullahetvenöt.

Spectrum 48/128 k programok eladók. Katalógusért küldjön válaszborítékot! Frim András, Pécs, Zrínyi u. 1. 7621

Szuper játék- és felhasználói programokat adok-veszek és cserélek kazettán. Hika György, Szeged, Sárosi u. 1/A. 6724. Telefon: 62/30-496 (17 óra után!) Csak cserélek!

C16, C Plus/4 programokat cserélek. Listát kérek. Polyák József, Örményes, Dózsa Gy. út 15. 5222

C16 és **C Plus/4** programokat cserélek kazetán és lemezen. ifj. Hajas Sándor, Szeged, Vedres u. 18/B. X. em. 29. 6726

C64-re játék és egyéb programokat cserélek és olcsón eladok. Listát kérek. Várhegyi István, Nyírbátor, Derzsi u. 31. 4300

C64-re programokat és leírásokat cserélek, adok. Cserealapom 2000 program és 200 leírás. $5^1/_4$-es lemezekért programokat adok, megegyezés szerint. Szabó Csaba, Ócsa, Üllői út 28. 2364

C64-es programokat cserélek kazettán. Listát kérek! Bernáth Csaba, Eger, Meder u. 3. 3300

Enterprise programcsere. Listát kérek! Rigó Balázs, Gödöllő, Kossuth L. u. 41. 2100

Enterprise programcsere, vétel, olcsó eladás, válaszborítékért lista. Magyar József, Budapest, Bánya u.21. I/7. 1214

Enterprise programcsere. Lakatos Péter, Nagykőrös, Zsemberi u. 44. 2750

Enterprise programcsere! Közel 400 program. Válaszboríték ellenében lista. Gyors csere, olcsó eladás. Cím: Sándor József, Bonyhád, Bezerédi u. 41. III/5. 7150

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

Az Olvasó írja

A levelezési rovatban most három levelet közlünk, olyan írásokat, amelyekhez hosszabb kommentárt fűzök, mert úgy vélem, hogy másoknak is hasonló problémáik vannak.

Varsányiné Bodó Katalin, Nagyatád

A µM, valamint a „Levelezési rovat" régi olvasója vagyok, ezért kérem, segítsen az alábbi problémám megoldásában.

Eddig családunknak Primo típusú számítógépe volt, azonban ettől megszabadultunk. Szeretnénk egy „komolyabb" számítógépet vásárolni, az alábbi szempontok figyelembevételével:

1. Gépi nyelven nem óhajtok programozni.
2. Magyarországon megfelelő programmennyiséget lehessen beszerezni.
3. Gyermekeim 8 és 5 évesek.
4. Floppyval szeretném használni.
5. Külföldi beszerzésre lehetőségem van, közel van az osztrák határ, de jelenleg érvényben van a 20 000 Ft-os vámrendelet is!
6. Nem tudom eldönteni a választást C64, C Plus/4, ZX—Spectrum, Enterprise között.
7. Az esetleges géphibákat férjem el tudja hárítani.

Nem hiszem, hogy tanácsommal nagyon sokan egyetértenének, ennek ellenére az a meggyőződésem, hogy ma — ha valakinek módjában áll választani, és az anyagi lehetőségei sem túlságosan korlátozottak — IBM PC kompatibilis számítógépet érdemes csak — hosszú távra — vásárolni. Mondom ezt annak ellenére, hogy — szerintem egy kezdetben helyes, most már hibás beszerzéspolitika következtében — a lakásokban, az üzemekben és az iskolákban is nagyon sok C64, CPlus/4, C 16, Sinclair ZX—Spectrum, QL, azután Enterprise, Primo és más gép van üzemben. Tehát aki ilyen gépekkel rendelkezik, az társakat talál magának, programokat cserélhet, van szerviz is, ahol a hibás gépet megjavíttathatja. Mindezeket figyelembe véve, ezeknek a gépeknek az ideje már lejárt, ha hiszek néhány külföldi barátomnak, már az IBM PC-é is, de nem annyira. Ha most 1989-ben az ember gépet vesz, akkor azt többnyire nem játékok futtatására veszi, ti. a mai, egyre rosszabb gazdasági viszonyok mellett ilyen nagy beruházást csak játék céljára józan ésszel már nem szabad tenni. Úgy gondolom, hogy a gép Önöknek azért kell, hogy a gyerekek tanuljanak rajta, pl. programozni, de tanulóprogramok segítségével más tárgyakból is készülhetnek, kártyakiegészítéssel VTX-terminálnak tudják használni, szövegszerkesztővel levelezzenek, tanulmányt írjanak, ezenkívül adattárolásra vagy számításokra használják. Én arra is gondolnák, hogy ezzel a géppel már pénzt is lehet keresni, vállalkozni lehet, mert elég gyors, elég nagy a tárolási kapacitása, és ráadásul jó alkalmazói programok is vannak hozzá. A gép továbbfejleszthető, pl. Winchester-memóriával, színes monitorral, rajzolóval, egérrel és így biztosan jó gép lesz, és így az értékét is kevésbé veszti el, mit a levelében felsorolt házi számítógépek. Itt hívom fel a figyelmét arra, hogy a µ 1989/9. számában részletesen megírtuk, hogyan lehet PC kompatibilis gépet olcsón beszerezni.

Klencsár Zoltán, Kaposvár

Kérem, hogy ha lehetősége van rá, a mellékelt válaszborítékban segítsen az alábbi problémámban:

Immár fél éve, hogy elkészítettem egy felhasználói-karakterszerkesztő programot. A program Commodore 16, Commodore 116 és Commodore +4 számítógépeken futtatható. Összehasonlítva a kereskedelemben kapható, hasonló feladatok elvégzését szolgáló programokkal, úgy találtam, hogy azok között is megállja a helyét. Van, amiben alul marad, de sok mindenben felülmúlja azokat. Azt hiszem, hogy számos — ilyen kis géppel rendelkező — programozó jó hasznát venné e programnak. Értékesíteni szeretném tehát valamilyen módon. De nem tudom, hogy ez lehetséges-e egyáltalán, és ha igen, akkor hogyan. Foglalkozik valamely kisszövetkezet még ezekkel a kis teljesítményű gépekre írt programokkal? Vagy esetleg az lenne a megoldás, ha a Mikro Magazin „Adok — Veszek — Cserélek" rovatában bocsátanám áruba? Nem tudom. De a segítségét előre is köszönöm.

Hosszan válaszolok, mert ismét mindenkit érdeklő kérdésről van szó: elkészült egy termék, el kellene adni.

Az első kérdés, hogy valóban kész-e az a bizonyos karakterszerkesztő program. Ti. egy ilyen program sok feltételnek meg kell, hogy feleljen. A legegyszerűbbek: van-e hozzá megfelelő dokumentáció, hibátlan-e a program; stb., vagy a bonyolultabbak: kell-e számítástechnikai ismeret a kezeléséhez, párbeszédes módon lehet-e használni, „bolondbiztos"-e a kezelése, védett-e másolás ellen, vagy nem, könnyen tovább lehet-e fejleszteni, tehát ki lehet-e egészíteni, megfelel-e a technológiai előírásoknak, és még lehetne sorolni a technikai kritériumokat vég nélkül. Arra a kérdésre, hogy egy ilyen vizsgálatot ki végezhet el, többféle válasz is lehetséges. Meg lehet próbálni, ha nagyon biztos a dolgában, az NJSZT-nél egy minőségi vizsgálatot kérni, vagy Commodore program esetében a forgalmazó, pl. a Novotrade, vagy a kiskereskedők, esetleg a kisvállalkozások, akiket ez a gépkategória érdekel, biztosan elmondják a feltételeiket, illetve ha nagyon érdekli őket a termék, elvégzik a fentiekre vonatkozó vizsgálatokat. Ha valaki PC programot készít, akkor nyugodtan fordulhat vagy szoftverházakhoz, hardvergyártókhoz, vagy pedig kereskedőházakhoz, mint pl. SZÁMALK, SZKI, Videoton, Novotrade, Metrimpex stb., mert ott azután szigorúan megvizsgálják a kínált árut, de gondoskodnak a szakszerű piacra viteléről is. Végül, ha az elkészült program — ne haragudjon meg a feltételezésért — amatőr gyártmány, akkor a legjobb, ha a µ „Adok — Veszek — Cserélek" rovatában próbálja meg az értékesítést, azután majd meglátja, hogy érdekli-e az eredmény az alkalmazókat, vagy nem. Sok sikert!

Krajnyák Levente, Győr

Feltűnt nekem, hogy a Mikroszámítógép Magazinban milyen sok figyelmet szentelnek az Enterprise gépeknek. Miért nem lehetne más gépekről is ennyit írni. Talán azért van ennyi cikk róla, mert sok beküldött anyag van? Mármint sok írás érkezik be ezzel a géppel kapcsolatban a szerkesztőségbe?

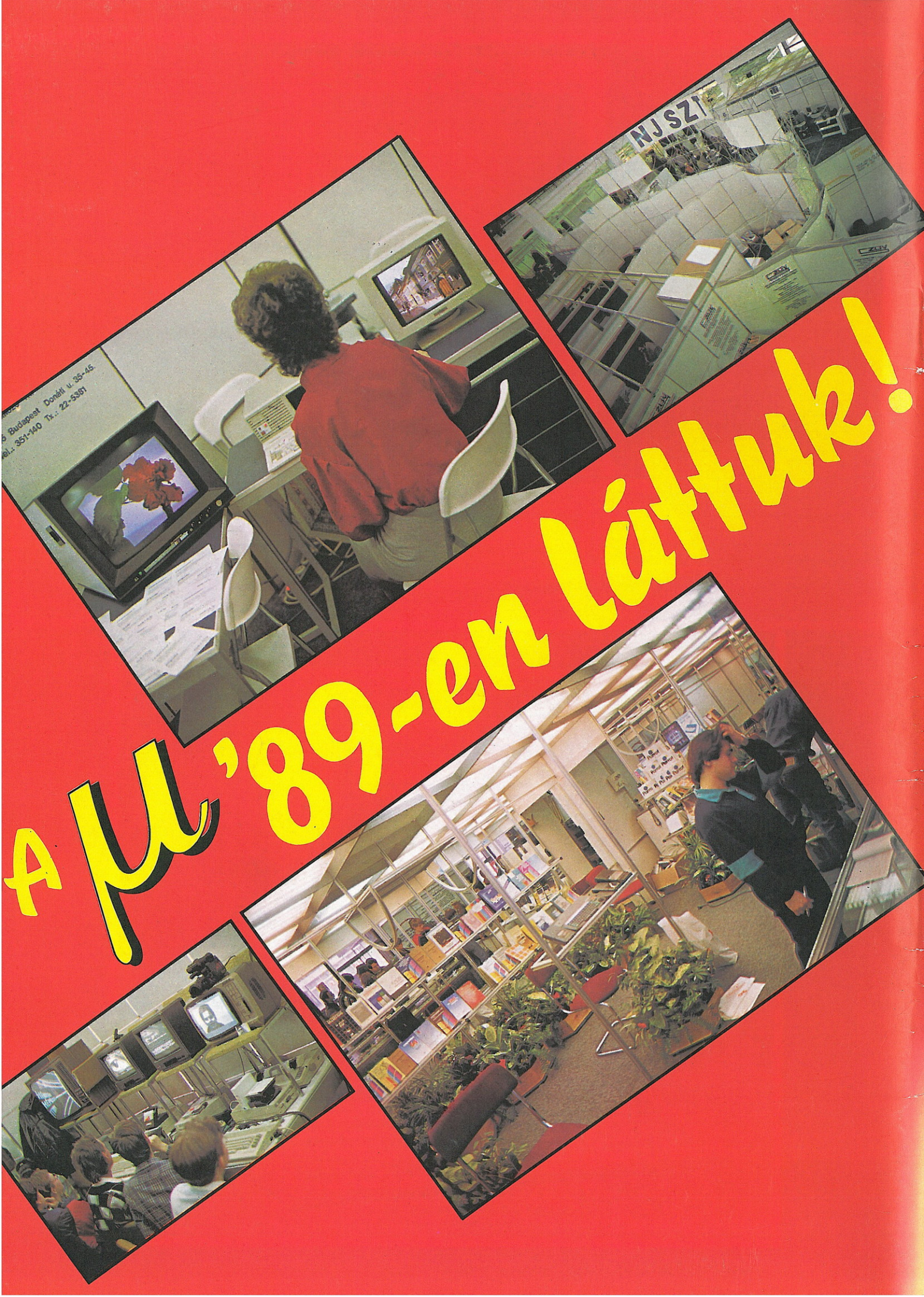
Az 1989/2-es számban olvastam Pákozdy Pál levelét. Szerintem az ötlet kitűnő. Ha minden hónapban kiadnának egy újságot, amelyben 4-5 géphez lenne 2-3 játékról leírás, akkor biztos, minden számítógép-tulajdonos megvenné ezt. Igaz, hogy ez többletmunkát jelent, de megéri.

1. Igaza van, az Enterprise ma az egyik legdivatosabb amatőr gép. Ráadásul kevés hozzá az alkalmazói program, ami nagy kihívás a programfejlesztőknek. Ha visszalapoz a µ korábbi számaira, akkor volt már nekünk HT, Primo, Commodore, Sinclair és mindenféle korszakunk is.

2. Ami a tematikus számokat illeti, én nem nagyon hiszek a sikerben. Egyrészt csökkent az érdeklődés a játékok iránt, másrészt csökkent a fizetőképes kereslet. Az ötletét azonban még nem vetettük el.

3. Nem szívesen foglalkozunk kézzel vagy géppel írott programokkal, ti. a „bepötyögést" senki sem vállalja. Ráadásul olvasóink ma már nem nyúlfarknyi programokat küldenek, hanem jó hosszúakat, amelyeknek a hibátlan bevitele bonyolult probléma. A jövőben persze (mikor?) a közölt programokat felírjuk egy lemezre, és a lemezt az újsággal együtt adjuk el, mint ahogyan azt más lapok is teszik (igaz, nem nálunk). Egy ilyen fejlődésben reménykedik, a mindig optimista:

Kovács Győző


NJSZT
Budapest Donáti u. 35-45.
Tel.: 351-140 Tx.: 22-5361
A μ'89-en láttuk!