

# mikro számítógép magazin

Ára: 30 Ft

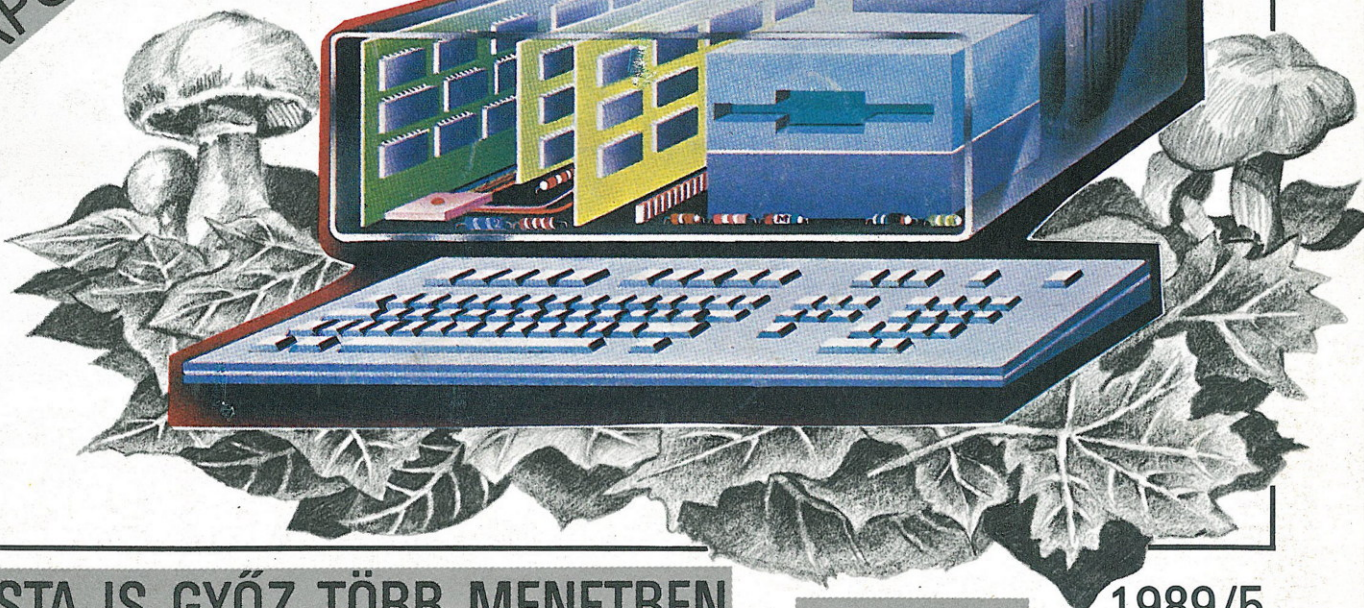
NEM SZÉGYEN A KÖRZŐ

ÉS VONALZÓ!

HOGYAN SZÖVEGELJÜNK?

VEGYEM?

EZ AZTÁN A  
KAPÓS MENÜ!



A LUSTA IS GYŐZ TÖBB MENETBEN

1989/5



*Rendszerei kiválasztásához, alkalmazói körülményei kialakításához segítséget nyújtunk, a szükséges gépeket biztosítjuk.*

*Részletes ismertető, szoftverbemutató, ár- és termékkatalógus!*

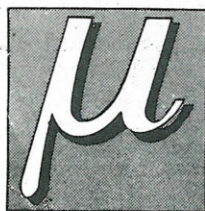


*Közösen megtaláljuk a megoldást...*



**SZOFTVERKERESKEDELMI ÉS  
FEJLESZTÉSI BETÉTI TÁRSULÁS**

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



## A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság  
vezetője:  
Kovács Győző

A szerkesztőség  
munkatársai:  
Bakos Tamás  
(programozástechnika)

Broczkó Péter  
(hírek)

Kovács Győző  
(levelezés)

Nagy Imre  
(tanuljunk együtt)

Petróczy Judit  
(könyvek)

Pinke György  
(NJSZT, alkalmazások)

Soltészné Vizi Zsuzsa  
(tervezőszerkesztő)

Simonyi Endre

Szebenszki Sándor

Szulyovszky Csaba

Tamásné Lakó Erika

Terebessy Ákosné

Varga János

Címképünk  
Velekey József Lajos munkája

**μ mikro számítógép  
magazin**



Felelős szerkesztő:  
Könyves Tóth Pál

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

Levél cím:  
1371 Budapest  
Pf. 433

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

Levél cím:  
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 (89-0551)  
Felelős vezető:  
Csöndes Zoltán vezérigazgató

INDEX: 25 629  
ISSN 0236-6088

### TARTALOM

2	Hittel és illúziókkal
9	A lusta is győz több menetben
10	Feladatok — megoldások
27	Rendszerfejlesztési eszközök
30	Adatvédelmi rendszer
32	Csak egy év, de micsoda fejlődés!
33	Merre tart a világ?
36	Bajorország — Hagyomány és haladás
38	Olvastunk...
41	Az RTST margójára
45	Programtermék — Vegyem?
46	Adok-veszek-cserélek

### TANULJUK EGYÜTT!

3

3	A Pascal rejtelsei
6	Nem szégyen a körző és vonalzó!

### CSIPEGETŐ

13

13	User port
13	Hess, te csúnya!
14	Grafikus monitor
14	A gyalogos
15	Örökélet
15	Egy, ami kettő
16	Extra nyomtatások
16	TOP-lista

### PROGRAMOZÁSTECHNIKA

17

17	A számítógépek motorja IV.
19	Ez aztán a kapós menü!
21	A szövegfeldolgozás elemei
22	Programozási fogások és melléfogások

### ENTERPRISE

23

23	A hang végül megszólal
24	Egy ismerős halmaz
25	Karakterek, karakterláncok
26	Mi a manó?

### μKLUB

42

42	Szenzoros, gyorstüzelő botkormány
43	Kérdés a lemezhez: FORMATTED?
43	Ki ad magyarázatot?

### SAKK

44

44	A figuratípusok jutalompontjai
----	--------------------------------

### KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

46

### AZ OLVASÓ ÍRJA

48

Néhány nappal ezelőtt az NJSZT ügyvezető elnöksége ülést tartott, meghallgatva Hanák Péter és Könyves Tóth Pál beszámolóját az NJSZT eddigi tevékenységéről, valamint az elkövetkezendő évekre vonatkozó javaslatokat.

Hát ami azt illeti, a felrajzolt múlt- és jövőkép nem volt valami szívderítő, mindjárt meg is jegyzem, hogy én egyáltalán nem így látom a világot, noha egyre több jel mutat arra, hogy ez a borús kép nemcsak borús, de sajnos reális is.

Nincs értelme annak, hogy a dolgozatot, ami nem a társaság, hanem két kiváló barátom véleményét tükrözi, teljes egészében ismertessem, erre valamilyen fórumon majd biztosan sor kerül. Nem szeretnék túl sokat a múlttal sem foglalkozni, ahhoz — korábbi főtitkári tisz-

végezni — mondják —, akkor megpróbálkozhat „szakmai, politikai szervezet”-té válni, esetleg a „szakmai elit klubjává”. Kevesebb esély van arra, hogy „tudományos fórum” legyen, még kevesebb, hogy szakmai etikai érdekvédelmi szervezetként működjön, és arra a legkevesebb az esély, hogy folytassa az ún. „társadalmi — a Társadalom infor-

„Az ember cselekedeteit többnyire nem az értelmi megfontolások, hanem vérmérséklete és érzelmei határozzák meg. Ez az oka annak, hogy az egyes ember életében azonos jellegű események és momentumok általában többször is ismétlődnek, és ezek határozzák meg az ember egyéni sorsát.”

(Károlyi Mihály: *Hit, illúziók nélkül.*)

sának 30 éves évfordulóját megünnepeleendő, Nyíregyházán létrehozzunk egy állandó számítástechnikai kiállítást, de a Művelődési Minisztérium is másra fordította azt a több mint 200 millió forintot (5 évre szóló büdzsé), amit pedig a lakossági elektronizálási programra szánt a kormány. És még azt sem mondták, hogy „sajnáljuk”.

Így azután tényleg nem lehet csodálkozni azon, hogy az emberek elvesztik a hitüket, és egyre kevesebben vállalják a „semminék dolgozzak?” helyzetet.

Én mégis csodálkozom, és megpróbálva a lehetetlent, szeretném, ha továbbvinnék a társadalmi programot. Hittel és nyilván egy jó adag illúzióval. Hittel, mert én hiszek abban, hogy vannak és lesznek olyan Neumann-tagok, akik továbbra is vállalják társadalmi munkában e program folytatását még akkor is, ha ezért semmiféle elismerést nem kapnak. Hiszek abban is, lehet, hogy illúzió, hogy a hazai számítástechnikáért felelős vezetők és főhatóságok is megértik, hogy társadalmi program nélkül nincsenek kiképzett alkalmazók, társadalmi igény nélkül nincs piac, továbbra nézve nincs iparilag fejlett gazdaság és a jelenlegi mélypontról nincs felemelkedés. A különböző fórumok által kritizált, elfuserált, óriási beruházások mellett jelentéktelen ráfordítással be lehetne vezetni például a nyílt tanulási rendszert, az Open University vagy Open College mintájára, egy nagyon kis támogatással fel lehetne lendíteni az amatőrmozgalmat, szélesíteni lehetne a klubhálózatot, újra lehetne, mert újra kell kezdeni az iskola-számítástechnikai projektet. Hiszem, hogy ezt a befektetést a kormányzat valóban fel fogja vállalni. Ha ez így történik, tehát a társaság megkapja a társadalmi tevékenységéhez szükséges központi támogatást, akkor egészen biztos vagyok abban, hogy nagyon sokan vállalják — még társadalmi munkákban is — a munkát, hittel és illúziókkal.

Kovács Győző

## Hittel és illúziókkal

tem miatt — túlságosan is kötődöm, és úgy vélem, hogy az akkori tevékenységemet nem is nekem, hanem a tagságnak kell megítélnie.

A dolgozat általános mondanivalója az, hogy a Neumann Társaság a jövőben képtelen lesz a különböző dokumentumokban rögzített és vállalt feladatokat ellátni, erre se ereje, se pedig tagsága nincs. A társaság *illúziókat* kerget, ami helytelen, ehelyett jobban tenné, ha számolna a realitásokkal, és csak azt tenné, amire erejéből, a tagok önkéntes tevékenységéből futja.

Az írás sok példát is felsorakoztat, például, hogy

- a társaság megpróbált és megpróbál állami feladatokat elvállalni, de ehhez se pénze, se apparátusa nincs,
- igyekszik szakmailag meghatározó szerepet vinni, de a szakmai ügyek máshol zajlanak,
- megpróbálja a társaság tagjait társadalmi munkára mozgósítani, de a tagok nem mozognak, kivéve egy-két lelkes embert, akik még csinálják, de meddig,
- a társaság elsősorban a fiataloknak nem eléggé vonzó, az amatőrök és a klubok elfordultak a társaságtól,
- a társaság tevékenységében, beleértve a vezetőséget is, nem a szakmai, hanem a szervezési és szervezeti kérdések dominálnak,
- a társaságnak megszűnt a szakmai, politikai szerepe,
- a társaság által vállalt társadalmi program pénz és vállalkozók hiánya miatt nem teljesíthető stb. stb. stb.

A tanulmány szerint a jelenlegi helyzetben a társaságnak mint rendezvény-, illetve szolgáltatóirodának kellene működnie, ezt a tevékenységét ma is végzi, erre a munkára a jelenlegi apparátus kiválóan megfelel.

Ha a társaság szakmai munkát is akar

matizálása — programot”, amit egyes fórumokon nem túl szerencsésen a „lakossági elektronizálási programnak” is neveznek.

A társaság életét nem túlságosan ismerő olvasónak szabad legyen elmondanom, hogy 1975-ben és 1980-ban is, a tisztújító közgyűléseken, de ha jól emlékszem, a rendes évi összejöveteleken is, sokat vitatkoztunk azon, hogy a Neumann Társaság a számítástechnika iránt érdeklődő (professzionális és nem professzionális) szakemberek, vagy pedig a számítástechnikai „elit” szervezete legyen. 1985-ig a társaság deklarálta az előbbi, és a tagok a tevékenységüket a „Társadalom informatizálása” program keretében végezték. Az 1985-ös tisztújítás ezt a programot nem vetette el, igaz, meg sem erősítette, a fordulat, ha a közgyűlés és az országos elnökség a javaslatot elfogadja, most várható.

Mondanom sem kell, hogy az ilyen változással, ami egyben azt is kell hogy jelentse, hogy a társaság átalakul tömegszervezetből a szűk szakmai kör szervezetévé, nem tudok egyetérteni, mert ez a döntés ellentétes azzal a számítástechnikai fejlődéssel, ami ma a világ fejlettebb részén látszik, és ami véleményem szerint az egyetlen követhető út.

Két kiváló barátomnak teljesen igaza van, a társadalmi programot szinte senki sem támogatja, se az állam, se a vállalatok. Az is igaz, hogy a tagok jó része is leállt, nem vállal társadalmi feladatot, hiszen az emberek többsége csak úgy szeret dolgozni, ha az ingyenmunkáért legalább megdicsérik, az már kevésbé lelkesíti őket, ha az a jutalom, hogy nem rúgnak beléjük.

A közelmúlt példái is azt mutatják, hogy a dolgozat íróinak van igazuk, hiszen például a Mikromagazin anyagi alapjainak megteremtésére indított mozgalom a vállalatok körében szinte teljesen visszhang nélkül maradt, gyakorlatilag nem volt jelentkező arra sem, hogy az első elektronikus számítógép átadá-

A békéscsabai MSZB MGTSZ megvételre följárni

1 db C—610-es számítógépet,  
8025-ös printert,  
120-as monitort.

Érdeklődni: **66/27-377-en.**



# A PASCAL REJTTELMEI

## 8. Új adattípusok definiálása

A Pascal sokak által leghatékonyabbnak ítélt eszköze a szinte korlátozás nélkül alkalmazható adatszerkezet-definiálás. A standard adattípusokból (**integer**, **real**, **boolean**, **char**, a Turbo Pascalban a **string** is) a programozó saját magának hozhatja létre azokat az adatszerkezeteket, amelyeket az adott feladat megoldásához a legjobban felhasználhatónak vél. A definiálható — a Pascal számára felismerhető — típusok választékát teljes egészében felsoroljuk, de közülük most és a következő folytatásokban csak néhánynak, a felsorolásban \*gal jelölteknek a pontosabb meghatározására és alkalmazására térünk ki.

A típusválaszték:

- tömbtípusok (\*),
- felsorolási típusok (\*),
- intervallum (subrange) típusok (\*),
- mutatótípusok,
- halmaztípusok,
- rekordtípusok,
- fájl- (állomány-) típusok (\*).

Ezek közül a tömbváltozókkal már találkoztunk **szamsor** nevű programunkban (6.3). Az itt használt **tomb** nevű változó azonban csak egy adattömb azonosítója (változóneve) volt, újabb változók típusának, szerkezetének leírására a **tomb** nevet nem alkalmazhattuk. Szintaxis szempontjából tehát hibás lett volna a következő deklaráció:

```
var tomb:array [1..10] of integer;
var sorozat:tomb;
ugyanis a Pascal fordító a tomb adattípust még nem ismervén, a
var sorozat:tomb;
deklarációval nem tud mit kezdeni. A helyes megoldás:
```

```
type tomb=array [1..10] of integer;
var sorozat:tomb;
A type kulcsszóval végzett típusdeklaráció után a tomb a Pascal számára egy adattípust jelent, amelyet képes felismerni és kezelni.
```

Például következő feladatunk, a névsorba rendezés programjában egy tíz, egyenként 20 karakteres névből álló típus definiálunk:

```
type tomb=array [1..10] of string
[20];
```

majd a programban három változót (**tiznev**, **regi**, **uj**) a változódeklarációban **tomb** típusként jelölünk meg.

Mivel a tömbök elemeinek típusára nincs előírás, egy tömb eleme lehet egy másik tömb is. Ezen az elven többdimenziós tömböket definiálhatunk, például:

```
type sor=array [1..10] of integer;
var matrix:array [1..20] of sor;
```

Ezzel ekvivalens a következő megoldás:  
var matrix:array [1..20] of array [1..10] of integer;

Hogy a két deklaráció közül melyiket válasszuk, azt a program tervezésekor mérlegelni kell. A második egyszerűbbnek tűnik, de az első megoldás a strukturált adatszerkezet révén lehetőséget ad — ha ez a programban szükséges — a **sor** típusú adat, vagyis az egydimenziós tizelemű **integer** tömb változódeklarációban történő alkalmazására is.

A tömbök elemeire hivatkozásnál az aktuális indexek értékét szögletes zárójelben, egymástól vesszővel elválasztva kell megadni. Például:

```
matrix[3,5]=12;
```

Kezdetben kényelmetlen, de megszokható korlátozása a Pascalnak, hogy a tömbök definiálásakor változót nem használhatunk, azaz a tömbök méretét a program megtervezésekor előre meg kell határozni. Ennek magyarázata, hogy a fordító minden változónak, így a tömböknek is, előre — még a futás előtt — lefoglalja a helyet a tárban (ez a helyfoglalás a lefordított, gépi kódú programban „már benne van”). Ha a tömbméreteket nem tudjuk előre meghatározni, ráterással kell dolgozni, azaz olyan méretű tömböt kell definiálni, hogy az a várható igényeknek biztosan megfeleljen.

```
program iskola;
type osztaly=array[1..20] of string[30];
{egy osztályban 20 tanuló, nevük max. 30 kar.}
osztalynev=(a1,b1,a2,b2,a3,b3,a4,b4);
{osztálynev típusú változó definiálása}
iskola=array[osztalynev] of osztaly;
{az iskola típusú változó definiálása}
{az iskolában osztálynev számú osztály}
var tanulo:iskola;
{a tanulo egy iskola típusú változó}
begin
tanulo[a1,7]:= 'kovacs';
{az a1 osztály 7. tanulójának neve}
clrscr;
writeln('A tanuló neve:');
writeln(tanulo[a1,7]);
end.
```

22. ábra

Nem esett még szó az indexek típusáról. Indexként csak **integer** vagy arra visszavezethető típusú adatokat alkalmazhatunk. Ilyenek a példánkban is használt egész számok és a következőkben ismertetendő felsorolási deklarációban szereplő adatok.

Bizonyos esetekben igen jól használhatók az ún. felsorolási (enumeratív) adattípusok. Ezek deklarálása egyszerű: a típusnevet követő egyenlőségjel után zárójelek között, egymástól vesszővel elválasztva fel kell sorolni a deklarált típus lehetséges értékeit. Például:

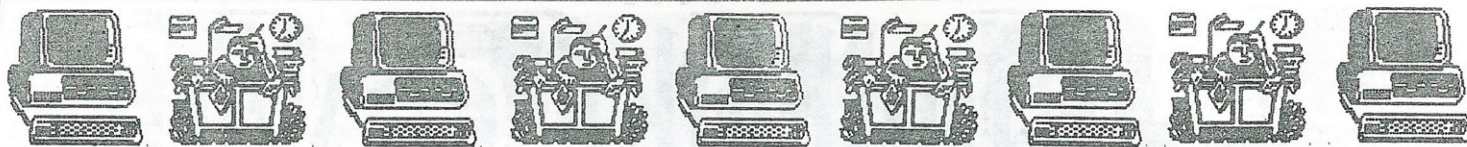
```
type irany=(észak,kelet,del,nyugat);
```

A felsorolás sorrendje meghatározza a lehetséges értékek relációját is, mivel a Pascal azokhoz 0-val kezdődően növekvő **integer** értékeket rendel hozzá. (Ezekhez az értékekhez a Pascal standard, például az **ord** vagy **integer** adatkonverziós függvényeivel egyébként hozzá is férhetünk. Ezzel a témával azonban — legalábbis jelenleg — nem foglalkozunk.)

Példánkra vonatkoztatva:

```
észak < kelet < dél < nyugat
```

Az előzőekben és már régebben is szó esett arról, hogy bizonyos adattípusokat a Pascal egészekre vezet vissza, illetve egészként tárol. Az ebből adódó lehetőséget már a Pascal definiálásakor is megragadták, és erre a tényre alapozva új változó-típusok is deklarálhatók. A típusok angol megnevezése **subrange** típus; a magyar nyelvű szakirodalom nem egységes: általában intervallum- vagy részhalmaztípusként emlegeti ezeket.



Az intervallumtípus fogalma talán úgy világítható meg legegyszerűbben, ha a felsorolási típusból vezetjük le. A felsorolási típus deklarációjában a típusba tartozó adatokat azok teljes felsorolásával kell megadni. Használjuk ehelyett a definícióra — ha a feladat jellege ezt lehetővé teszi — a felsorolás első és utolsó elemének (integer) értékét vagy egy-egy abból származtatott adatot. Például:

```
type ora = 0..23
      perc = 0..59
      ev = 0..99
      kisbetu = 'a'..'z';
```

A példák jól igazolják az intervallumtípus elnevezést: minden esetben a lehetséges integer vagy arra visszavezethető értékek egy tartományát adtuk meg az alsó és a felső határ megjelölésével. Ha a típusokat változódeklarációkban alkalmazzuk és a program futása során a változók értékadásokban szerepelnek, a Pascal — ha ezt az igényt a programban, a későbbiekben (a 13. részben) ismertetett módon kifejezzük — azt is ellenőrzi, hogy a változó a típusdeklarációban megjelölt értéktartományban van-e. Például:

```
var kezdet, veg: ora
    szuletesi_ev: ev;
```

Más programnyelveknél ez az ellenőrzés nem automatikus, ezt a programban külön el kell végezni (például a *kezdet* változó értékének helyességét értékadás után programmal kell ellenőrizni, tudniillik azt, hogy 0-nál nem kisebb és 23-nál nem nagyobb-e.)

Az új változótipusok létrehozásának elő-

nyeire bemutatunk egy igen egyszerű példát. Egy iskola négy évfolyamán két-két (a és b) osztály van. Minden osztályba legfeljebb húsz tanuló jár, nevüket maximálisan 30 karakterrel kívánjuk leírni. Hozzunk létre a feladat jellegéhez — egy tanuló meghatározásához — igazodó új adattípust! A program, kommentekkel ellátva, a 22. ábrán látható.

Első lépésként egy *osztaly* adattípust definiálunk. Ez nem más, mint egy húszelemű sztringtömb, ahol minden sztring hossza 30 karakter.

A következő lépés egy felsorolási típusdeklaráció, amely egy *osztalynev* típusú változót hoz létre. Ennek értékei rendre az egyes osztályok megjelölései lehetnek (1.a helyett a1-et írunk, mert a Pascal szintaxisa szerint változó számmal nem kezdődhet).

Az *osztalynev*, ami tulajdonképpen egy nyolcelemű tömb, lesz az egyik indexe az *iskola* típusú tömbnek — ennek minden eleme egy *osztaly* típusú tömb. Ezek után minden tanuló egy *iskola* típusú változóként deklarálhatunk.

Összefoglalva: létrehoztunk egy új adattípust, tulajdonképpen egy kétdimenziós sztringtömböt, amelyben a sztringek a tanulók nevei. Az egyes tömbelemekre (elnézést, hogy a tanulókat most így nevezzük) hivatkozni az osztály megjelölésével és a tanulók osztálybeli sorszámával lehet. Programunk erre is mutat példát: az a1 osztály 7. tanulójának Kovacs nevet ad. Hogy a program működik is, azt a 23. ábrán látható futtatási kép is bizonyítja.

tás történik, ha rövidebb, az adat minimális helyfoglalással, de csonkítás nélkül íródik ki.

**Real** értékek kiírásánál az egész és tört részek hosszának megadása hasonlóan egyszerű. Ehhez az adat (konstans, változó, kifejezés) után, attól kettősponttal elválasztva a teljes mező (előjel, számjegyek, tizedespont) szélességét, majd újabb kettőspont után a tört számjegyek számát kell megadni.

Például a 24. ábrán látható program a 25. ábrán bemutatott kiírási formátumot hozza létre. A 25. ábrán feltüntetett eredmények legfeljebb csak annyi magyarázatot igényelnek, hogy ha a megadott formátumba az adat „nem fér bele”, a kiírás a minimális jegyből álló lebegőpontos formátumba vált át.

## 9.2 A névsorba rendezés programja

A program három eljárásra épül. Szerkezetének meghatározásakor szándékosan nem a legkisebb terjedelemre, hanem a jó tagoltságra és eddigi ismereteink alkalmazására törekedtünk. Célunk volt egyrészt az eljárások függetlenítése a programkörnyezettől, másrészt a formális és globális paraméterek használatának érthető, jól dokumentált bemutatása. Ez a magyarázata annak is, hogy először egymástól függetlenül ismertetjük az egyes eljárásokat.

A *bevitel* eljárás (listája a 26. ábrán) tíz nevet olvas be a globális *tiznev* tömbváltozóba. A *write* eljárásban alkalmazott formázott kivitel célja az 1–10 számok helyérték szerinti egymás alá írása.

A *rendez* eljárás (listája a 27. ábrán látható) a buborék-algoritmussal rendezi a névsort, és a tizelemű tömb minden átvizsgálása után kiírja az elvégzett cserék számát is. Ha már nem történt csere, a rendezésnek vége.

Az eljárás formális paraméterei:

- **elem** (a névsorban lévő elemek száma),
- **regi** (a rendezetlen névsor, **tomb** típusú változó),
- **uj** (a rendezett névsor, szintén **tomb** típus).

A lokális változók:

- **temp** (átmeneti munkaterület két név cseréjéhez),
- **i** (ciklusváltozó),
- **csere** (számláló az elvégzett cserékhez).

Az eljárás a névsort nem önmagába rendezi, a *regi* adattömböt változatlanul hagy-

# 9. Program tíz név ábécé szerinti rendezésére

Mielőtt a feladat elvégzéséhez hozzálát-nánk, egy, a programban felhasznált Pascal-újdságot ismertetünk.

## 9.1 Formázott kivitel

Elsősorban táblázatok vagy hasonló kiírási formátumok készítésénél fontos, hogy a megjelenített adatok „külalakja” a táblázat

szerkezetének megfelelő legyen. Az ilyen, úgynevezett formázott kivitelre a Pascal is ad lehetőséget: a *write* és a *writeln* eljárásokban megadhatjuk az *integer* típusú adatok kiírásához rendelkezésre bocsátott mező szélességét az adat (konstans, változó, kifejezés) után, attól kettősponttal elválasztva, egy *integer* értékű kifejezéssel. Ha a definiált mező hosszabb a szükségesnél, balról szóközökkel töltődik fel (jobbra igazi-

A tanuló neve:  
kovacs

23. ábra

25. ábra

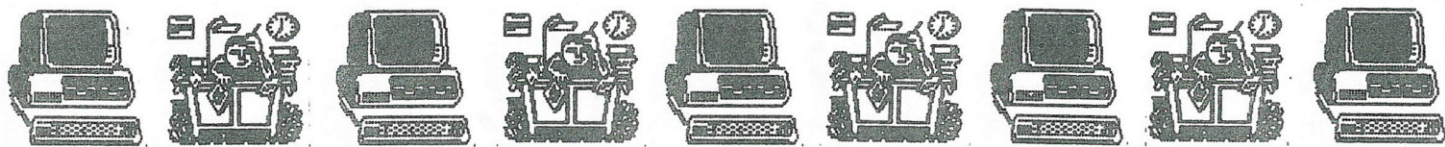
```
132.13
 132.13
132.1
1.3E+02
```

24. ábra

```
program formazott;
var a:real;
begin
  a:=132.13;
  clrscr;
  writeln(a:6:2);
  writeln(a:8:2);
  writeln(a:5:1);
  writeln(a:1)
end.
```

26. ábra

```
procedure bevitel;
var i:integer;
begin
  clrscr;writeln('Kérek 10 nevet');
  for i:=1 to 10 do
    begin
      write(i:2,'. név: ');
      readln(tiznev[i])
    end
end;
```



ja, a rendezést az új tömbön végzi el, amelynek kezdeti értékét az

```
uj:=regi;
értékadással biztosítjuk.
```

Ezt a talán körülményesnek tűnő módszert két okból választottuk. Egyrészt be akartuk mutatni az előbbi értékadás „szépségét” — ez ilyen formában más programnyelvekben nemigen valósítható meg —, másrészt szeretnénk szoktatni olvasóinkat ahhoz az elvhez, amely szerint adatállományokat önmagukba nem célszerű rendezni. Ezt a tanácsot egyébként jelen esetben nem szükséges megfogadni, mert hardverhibánál ugyanis mindennek vége; lemezes fájlok feldolgozásakor viszont — ha a rendezést nem az eredeti fájlban, hanem egy munkafájlban végezzük —, ha hiba keletke-

zik, legalább az eredeti, sokszor kincset érő adatállomány megmarad.

A kivitel eljárás a rendezett névsort írja ki a képernyőre. Listáját a 28. ábrán láthatjuk.

Formális paraméterek:

- *x, y* (a nevek kezdő pozíciójának koordinátái a képernyőn),
- *elem* (a névsorban lévő nevek száma),
- *nevsor* (a kiírandó névsor (**tomb**) típusú változó).

Lokális paraméter:

- *i* (ciklusváltozó)

A főprogram (a 29. ábrán látható teljes programlista végén) a három eljárás hívásán kívül csak említésre sem méltó „apróságokat” végez. Az aktuális paraméterek

megadása különösebb magyarázatot nem igényel.

A paraméter-átadási folyamatot a jobb megértés céljából mégis külön ismertetjük, hogy egyszerűbben írhattuk le Pascal-szintaxis szerinti értékadásokkal.

1. rendez eljárás:

```
elem:=10
regi:=tiznev
tiznev:=uj
```

2. kivitel eljárás:

```
x:=30
y:=1
elem:=10
nevsor:=tiznev
```

A program lefutása utáni eredményeket, illetve a futásra jellemző screent a 30. ábra szemlélteti.

A programlista átböngészése és az esetleges kipróbálás utáni időkre kiváló edzési lehetőséget ajánlunk: próbálkozzanak meg olvasóink a program rövidítésével, akár az áttekinthetőség, strukturáltság rovására is; például alkalmazzanak több globális változót, kevesebb változónevet, a rendez eljárásba kísérleljenek meg egy másfajta ciklusszervezést stb.

Nagy Imre

```
procedure rendez(elem:integer;var regi,uj:tomb);
var temp:string[20];
    i,csera:integer;
begin
    uj:=regi;
    repeat
        csera:=0;
        for i:=1 to elem-1 do
            if uj[i]>uj[i+1] then
                begin
                    temp:=uj[i];
                    uj[i]:=uj[i+1];
                    uj[i+1]:=temp;
                    csera:=csera+1
                end;
        write(csera,' ');
    until csera=0
end;
```

27. ábra

29. ábra

```
program nevsorbarendezes;
{formális és aktuális paraméterek,formázott kivitel}
{új változótipusok definiálása: type}
type tomb=array [1..10] of string[20];
var tiznev:tomb;
procedure bevitel;
var i:integer;
begin
    clrscr;writeln('Kérek 10 nevet');
    for i:=1 to 10 do
        begin
            write(i:2,'. név: ');
            readln(tiznev[i])
        end
    end;
end;
procedure rendez(elem:integer;var regi,uj:tomb);
var temp:string[20];
    i,csera:integer;
begin
    uj:=regi;
    writeln;writeln('A cserék száma:');
    repeat
        csera:=0;
        for i:=1 to elem-1 do
            if uj[i]>uj[i+1] then
                begin
                    temp:=uj[i];
                    uj[i]:=uj[i+1];
                    uj[i+1]:=temp;
                    csera:=csera+1
                end;
        write(csera,' ');
    until csera=0
    end;
end;
procedure kivitel(x,y,elem:integer;var nevsor:tomb);
var i:integer;
begin
    for i:=1 to elem do
        begin
            gotoxy(x,y+i);
            write(i:2,'. név: ');
            writeln(nevsor[i])
        end
    end;
end;
begin {főprogram}
    bevitel;
    rendez(10,tiznev,tiznev);
    gotoxy(30,1);writeln('A rendezett névsor');
    kivitel(30,1,10,tiznev);
    gotoxy(1,15)
end.
```

28. ábra

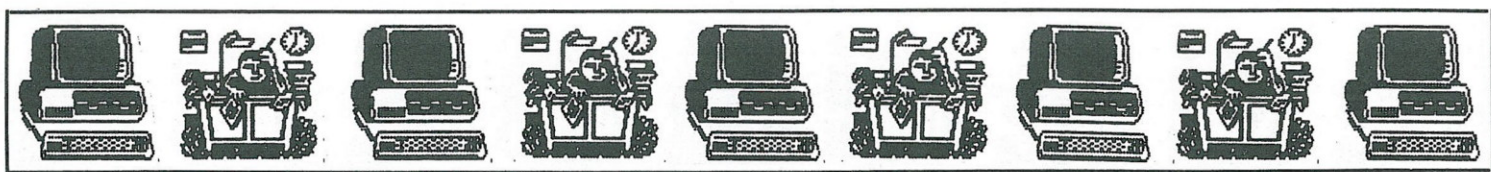
```
procedure kivitel(x,y,elem:integer;var nevsor:tomb);
var i:integer;
begin
    for i:=1 to elem do
        begin
            gotoxy(x,y+i);
            write(i:2,'. név: ');
            writeln(nevsor[i])
        end
    end;
end;
```

30. ábra

Kérek 10 nevet	A rendezett névsor
1. név: szabo	1. név: abafi
2. név: kiss	2. név: bakonyi
3. név: magyar	3. név: barna
4. név: orosz	4. név: csabai
5. név: abafi	5. név: jozsef
6. név: barna	6. név: kiss
7. név: bakonyi	7. név: magyar
8. név: nagy	8. név: nagy
9. név: jozsef	9. név: orosz
10. név: csabai	10. név: szabo

A cserék száma:  
9, 6, 5, 5, 3, 1, 0,

>



## Grafikus nyomtatás NLQ feliratokkal

# Nem szégyen a körző és vonalzó!

Sok szakmában elkerülhetetlen műszaki rajzok készítése. A számítógép speciális perifériákkal kiegészítve megkönnyíti ezt a munkát. Tökéletes rajzokat azonban csak jó minőségű, elsősorban vonalas rajzgépekkel vagy lézernyomtatókkal készíthetünk. A házi-iskolai mikrogépek használóinak rendszerint be kell érniük azzal a minőséggel, amire egy mátrixnyomtató képes. Gyakran alkalmazzák az interaktív rajzoló- (Paint Box, Botticelli) vagy tervező- (CAD) programokat, de ezekkel nem könnyű, vagy éppenséggel lehetetlen tetszőleges vonalból álló ábrát létrehozni.

Megtehetjük még, hogy egyedi programot írunk, és ezt a grafikus képernyő kinyomtatásával papírra visszük, de az így készült ábrák

„szabályos” vonalainak alakja a papíron egyhén szólna gyengécske.

Ha a műszaki rajz nívóját nem is érhetjük el, arra mégis van megoldás, hogy illusztrációk céljára jó minőségű ábrát készítsünk mátrixnyomtatóval. Ehhez olyan programot kell kidolgozni, amely nem rajzolja meg az ábra összes vonalát, hanem csak a legfontosabb szerkesztési pontokat, melyeket felhasználva, a kinyomtatott ábrán egyenes és görbe vonalzókkal, körzővel meghúzzhatjuk a szükséges vonalakat. Vagyis a szerkesztést a programmal, a kihúzást manuálisan végezzük.

A grafikus képernyő kinyomtatásával készülő ábrák másik hiányossága, hogy a feliratokban szereplő karakterek formája nem azonos a

nyomtatott szövegekben látható karakterekével. Ez a képernyőn és a nyomtatón használt eltérő sorfelbontás és karaktergenerátor használatának következménye. Az 1. ábrán láthatjuk a nagy A betű kétféle alakját. (Mindkét matrica gyártónként és géptípusonként változó.) Különbösen szembetűnő a különbség a grafikus és az NLQ-val nyomtatott karakterek között. Ezen viszont úgy segíthetünk, hogy az ábrába nem grafikus nyomtatással írjuk be a szövegeket. Az algoritmus elvileg is, gyakorlatilag is egyszerű: a grafikát a nyomtató soronként viszi papírra. Egy grafikus sor kinyomtatása után a nyomtatófejet visszaléptetjük a sor elejére, és karakteres üzemmódba váltva kiírjuk a feliratokat.

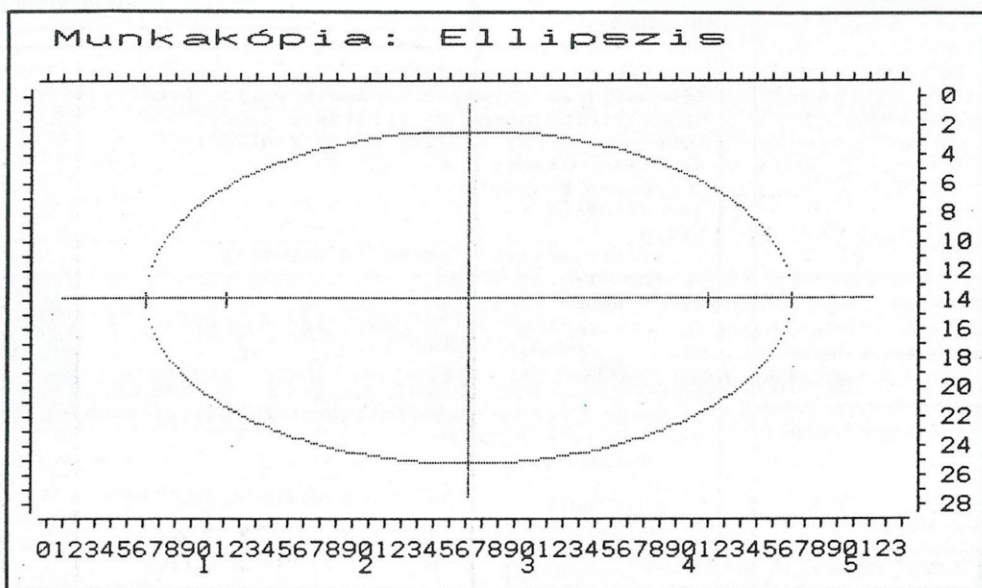
Mivel a gyakorlatban igen eltérő számítógépek és nyomtatók használatosak, a feladatot tökéletesen megoldó, de csupán egyetlen konfigurációra alkalmas program közlése helyett célszerűbbnek tartom az olvasót arra csábítani, hogy a program tervének elkészítésében tartson velem. A közölt programlistákkal az a célom, hogy bemutassak egy olyan variánst, melyet ki-kí saját gépére átformálhat.

### A feladat

Készítsünk tetszőleges grafikus pontokból álló ábrát, és az azt magába foglaló rajzfelületen elhelyezkedő szövegekkel együtt jelenítsük meg mátrixnyomtatón.

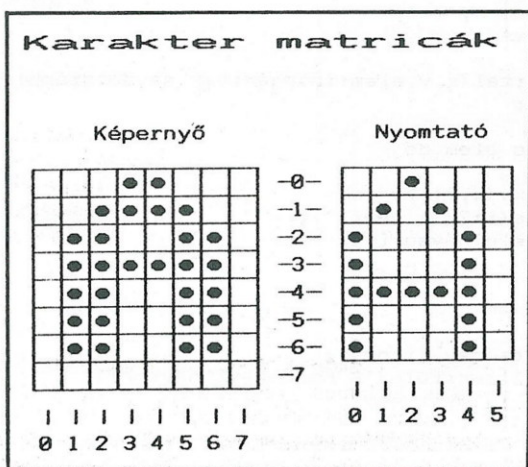
A feladat célszerűen a következő részekre bontható: rajzolás, munkakópia készítése, feliratozás, nyomtatás.

Első gondolatunk lehetne egy olyan menüvezérelt program, amely e funkciók kiválasztását teszi lehetővé. A grafikához szükséges munka-

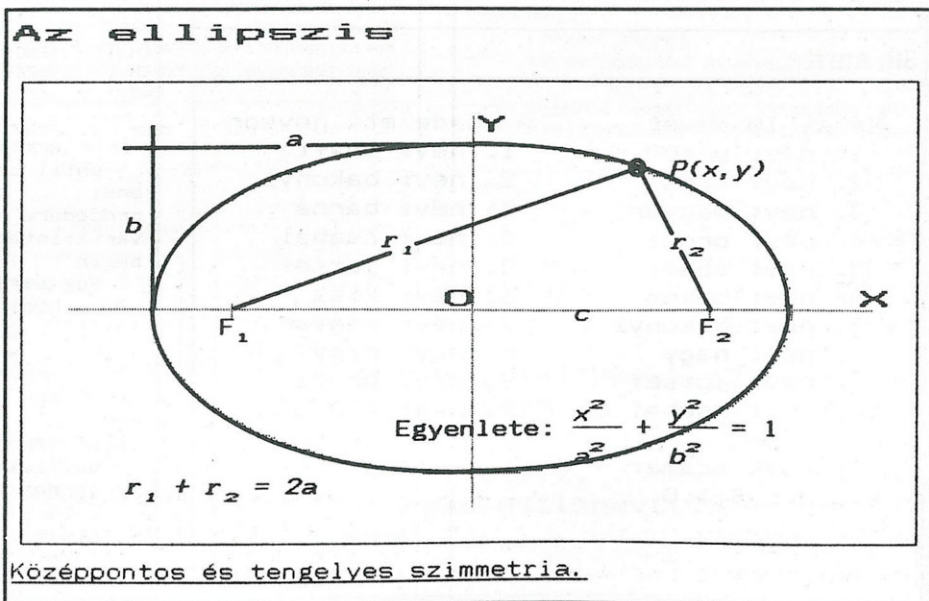


2. ábra

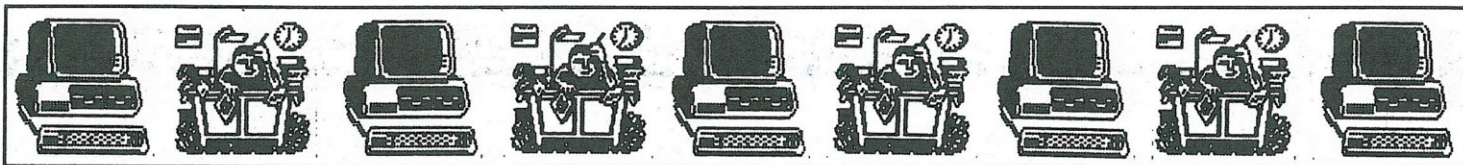
1. ábra



3. ábra







területek nagysága miatt azonban célszerűbb a négy részfeladathoz külön programokat tervezni. A programcsomag moduljai közötti adatszeréhez külső adathordozót használunk.

## Rajzolás

Minden ábra egyedi, ezért univerzális rajzólóprogram készítésére nem gondolhatunk. Az 1. modulban (1. lista) alkalmazott eljárás szemléltetésére egy egyszerű példát mutatunk. Ha a 3. ábrán látható rajzot a nagy felbontású képernyőn akarjuk létrehozni, akkor erre a grafikus utasításokat alkalmazó, a 2. listán látható programot használnánk. Ez a program az ellipszist, a koordinátatengelyeket, a fókuszokat és a csúcspontokat rajzolja meg. (Az ábrát szándékosan pontonként állítjuk elő, az egyenest és az ellipszist rajzoló rutinokat nem használjuk!)

Az ábra rajzfelülete most a képernyő, azaz vízszintesen 320, függőlegesen 200 pixel. Mivel a mátrixnyomtató grafikus nyomtatás közben egy grafikus jelet a nyomtatófejlben lévő tüoszlop egy kombinációjával hoz létre, a teljes rajzfelületet nem pixelsoronként nyomtatjuk.

### 1. lista

```

0 REM"@MODUL-#1"
10 REM *****
20 REM * GRAFIKUS NYOMTATAS *
30 REM * NLQ FELIRATOKKAL *
40 REM *****
50 REM * RAJZ => LEMEZRE *
60 REM *****
70 INPUT "ABRA-NEV (-12 KAR.)";FL$
80 IF LEN(FL$)>12 THEN 70
90 :
100 GV=319 : GF=28 :REM RAJZFELULET
110 DIM G%(GV,GF) :REM GRAFIKON
120 DIM B%(6) :REM 2 HATVANYOK
130 B%(0)=1 :FOR K=1 TO 6
140 B%(K)=2*B%(K-1) :NEXT K
150 REM --- AZ ELLIPSZIS KONSTANSAI ---
160 A=120 : B=80 :REM - FELTENGELYEK
170 U=160 : V=100 :REM - - KOZEPPONT
180 E=B/A :REM - - LAPULTSAG
190 A2=A*2
200 C=SQR(A2-B*2) :REM FOKUSZTAVOLSAG
210 REM A 1/2-ELLIPSZIS EGYENLETE
220 DEF FNE(X)=E*SQR(A2-(X-U)*2)
230 REM >>>--- RAJZ KEZD --->>>
240 FOR X=U-A TO U+A
250 : FE=FNE(X)
260 : Y=V-FE :REM - - FELSO RESZ
270 : Q=INT(Y/7) :R=Y-7*Q
280 : G%(X,Q)=G%(X,Q) OR B%(R)
290 : Y=V+FE :REM - - ALSO RESZ
300 : Q=INT(Y/7) :R=Y-7*Q
310 : G%(X,Q)=G%(X,Q) OR B%(R)
320 NEXT X
330 REM - - - - - ABSZCISZA TENGYEL
340 Q=INT(V/7) :R=V-7*Q
350 FOR X=9 TO GV-9
360 : G%(X,Q)=G%(X,Q) OR B%(R)
370 NEXT X
380 FOR Y=1 TO GF-1 :REM ORDINATA TENG.
390 : G%(U,Y)=127
400 NEXT Y
410 G%(U-C,V/7)=127 :REM FOKUSZPONTOK
420 G%(U+C,V/7)=127
430 G%(U-A,V/7)=127 :REM - CSUCSPONTOK
440 G%(U+A,V/7)=127
450 REM <<<<--- RAJZ VEGE ---<<<<
460 :
1000 REM >>>>--- LEMEZRE --->>>>
1010 OPEN 8,8,8,FL$+". [G],S,W"
1020 PRINT#8,GV:PRINT#8,GF
1030 FOR S=0 TO GF : FOR O=0 TO GV
1040 : PRINT#8,CHR$(G%(O,S)+128);
1050 NEXT O,S : CLOSE 8
1060 REM <<<<--- V E G E ---<<<<
1070 END

```

A héttűs nyomtatónál a grafikus sorok száma  $GF = \text{int}(200/7)$ , a soron belül a grafikus jelek száma  $GV = 320$ . A teljes képernyőt kitöltő rajz pontjait egy  $G\%(GV,GF)$  tömbben tárolhatjuk. Ezt a képet a képernyő RAM-ban is megtalálhatjuk, tehát onnan minden adatot a  $G\%(...)$ -be tölthetünk. Ezt a módszert azonban a PEEK(X) lassúsága miatt sem érdemes alkalmazni, ezért más utat választunk: a  $G\%(...)$  közvetlen feltöltését.

Mivel a grafikus üzemmódban egy tüoszlop egy, a [128;255] intervallumba eső egybájtos adattal vezérelhetünk, először a rajzfelület (képernyő) egy (X,Y) pontjának állapotát tartalmazó  $G\%(O,S)$  elem indexeit kell meghatároznunk. Az oszlopindex vízszintesen nyilván  $O=X$ .

A sorindexet a  $Q = \text{INT}(Y/7)$  kifejezés adja, amely az Y koordináta hetedének egész része. Ugyanezen osztás maradéka  $R = Y - 7 \cdot Q$ , amely meghatározza, hogy az Y ordinátájú pont az S-edik soron belül, felülről számítva hányadik. Ezt a pontot úgy tudjuk kigyújtani, azaz a nyomtató megfelelő tujét úgy tudjuk vezérelni, hogy a kiírt grafikus jelek megfelelő egybájtos adatban a megfelelő bitet 1-re állítjuk.

Az (X,Y) pont rajzolásának eljárása, amely megfelel a RAJZOLD X,Y szerkezetű grafikus utasításnak:

$Q = \text{INT}(Y/7) : R = Y - 7 \cdot Q$   
 $G\%(X,Q) = G\%(X,Q) \text{ OR } 2^R$

A számolást gyorsíthatjuk azzal, hogy a rajzólóprogram elején kiszámítjuk a szóba jövő 2-hatványokat.

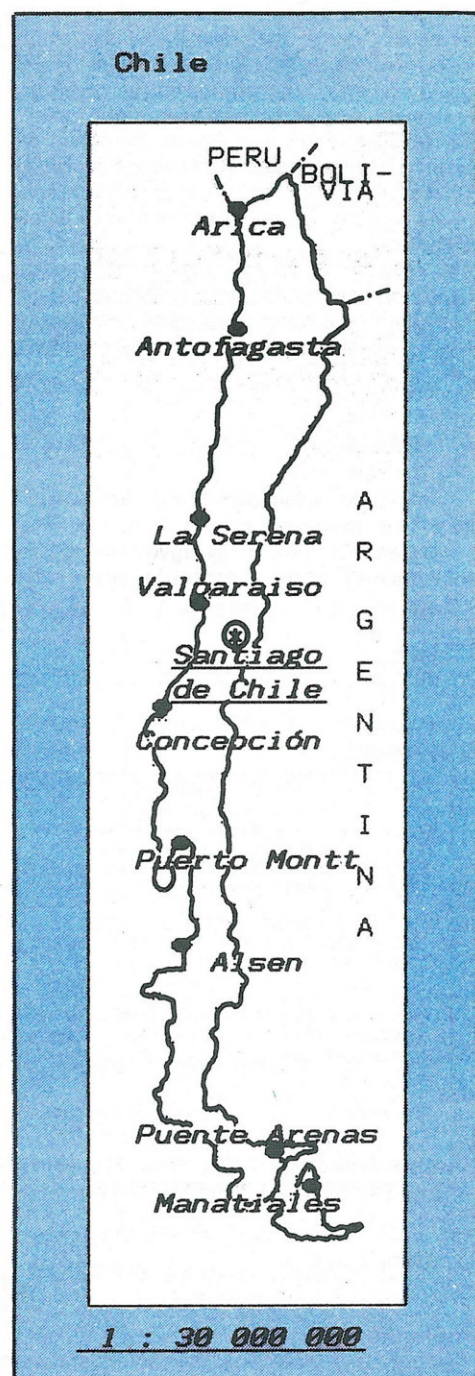
Az ellipszist rajzóló PÉLDA programot az 1. modulban a tervezett rajzolásnak megfelelően átalakítva láthatjuk. A program az elkészült rajzot lemeze menti. A lemezen a rajzfelület megadó GV, GF adatokat a  $G\%(O,S)$  tömb követi.

### 2. lista

```

0 REM"PELDA"
10 REM *****
20 REM * ELLIPSZIS RAJZOLAS *
30 REM * COMMODORE BASIC V3.5 *
40 REM *****
150 REM --- AZ ELLIPSZIS KONSTANSAI ---
160 A=120 : B=80 :REM - FELTENGELYEK
170 U=160 : V=100 :REM - - KOZEPPONT
180 E=B/A :REM - - LAPULTSAG
190 A2=A*2
200 C=SQR(A2-B*2) :REM FOKUSZTAVOLSAG
210 REM A 1/2-ELLIPSZIS EGYENLETE
220 DEF FNE(X)=E*SQR(A2-(X-U)*2)
230 REM >>>--- RAJZ KEZD --->>>
235 GRAPHIC 1,1 :REM GRAFIKA BE
240 FOR X=U-A TO U+A
250 : FE=FNE(X)
260 : Y1=V-FE :REM - - FELSO RESZ
280 : DRAW 1,X,Y1
290 : Y2=V+FE :REM - - ALSO RESZ
310 : DRAW 1,X,Y2
320 NEXT X
350 FOR X=9 TO 310 :REM X-TENGYEL
360 : DRAW 1,X,V
370 NEXT X
380 FOR Y=6 TO 193 :REM Y-TENGYEL
390 : DRAW 1,U,Y
400 NEXT Y
409 FOR I=-3 TO 3
410 DRAW 1,U-C,V+I :REM FOKUSZOK
420 DRAW 1,U+C,V+I
430 DRAW 1,U-A,V+I :REM CSUCSOK
440 DRAW 1,U+A,V+I
441 NEXT I
450 END

```

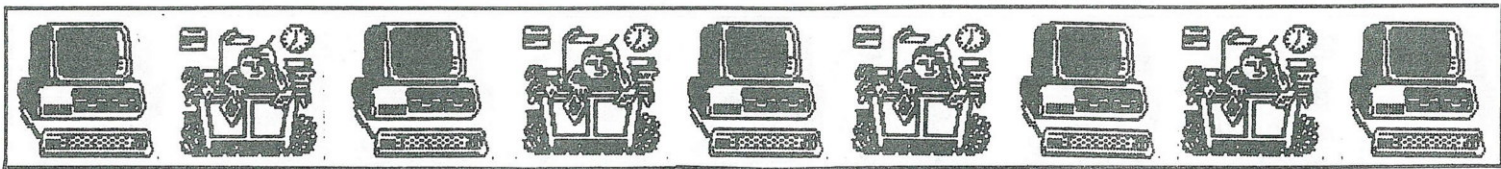


4. ábra

## Munkakópia és feliratozás

Az ábra feliratainak elhelyezéséhez szükségünk van egy olyan kópiára, amelyen ki tudjuk mérni a karakteres nyomtatásnak megfelelő sor- és oszlopkoordinátákat. Az 1. ábrán bemutatott eltérések miatt ez a hálózat nem egyezik meg a karakteres képernyő hálózatával. Az ellipszistről készített munkakópiát láthatjuk a 2. ábrán, melyet a programcsomag 2. moduljával készítettünk (3. lista).

A munkakópián bejelöljük a feliratokat, s ezeket a 3. modul segítségével lemeze visszük (4. lista). Ezzel a modullal egysoros címfeliratot és aláírást is készíthetünk. A rajzfelületre kerülő szövegek a sor, oszlop beírása után vihetők



be. A bevétel szerkezetéből következik, hogy hibás beütés esetén ugyanabba a sorba ismét beírva javíthatunk. A feliratok elhelyezésénél ügyelni kell arra, hogy egy sorba csak egyszer lehet beírni. Az egy soron belülré írandó különálló részeket szóközzel elválasztva lehet elhelyezni. Ha a szövegben határoló karaktereket (,) akarunk írni, a beírásakor idézőjelet kell használni. A beírás negatív sorszámmal lehet befejezni.

A 3. modul a szövegeket lemezre viszi. A grafikus és a feliratos állományokat a [G] és [F] toldalékok különböztetik meg. Ezért korlátozott 12 karakterre az ábra nevét az 1. modulban.

### Nyomtatás

A két lemezes állomány tartalmazza mindazt, amit a rajzpapíron meg akarunk jeleníteni. A 4. modul feladata a két állomány egybefűzése nyomtatás közben. A modul (5. lista) a rajz-

felület méretei után a feliratokat olvassa be. (A karakterenkénti beolvasásra a ; jelek miatt van szükség.) Ezután a címfeliratot, majd az ábra felső keretét iratjuk ki. A kép és a feliratok sorait felváltva visszük ki. Minden sorban, bal és jobb oldalon a keretet is karakteresen rajzoljuk meg. Végül az alsó keret és az ábra aláírása fejezi be a szerkesztési vázlatot. Az ábra kerete és a tényleges rajzfelület között egy-egy üres karakterhelyet is kihagytunk.

A kinyomtatott rajzon szükséges igazításokat filctollal, tuskihúzóval végezzük el.

A 4. ábrán szereplő térkép alakjából látható, hogy az alkalmazott megoldás a grafikus nyomtatásnál több lehetőséget ad a rajzfelület megválasztására.

### Fejlesztés

A négy modulból álló programcsomagot az érdeklődő olvasó tetszés szerint kibővítheti, javíthatja. Néhány triviális rutint szándékosan el-

hagytam a listákból: adatvédelem beolvasásánál, rajzolásnál, lemezhibák kivédeése stb. Ezeket az egyéni kivételben a szükséges mértékben pótolni kell. Az igazi továbblépést olyan kiegészítések jelentik, mint például a lemezes állományok módosítása, ábrák nagyítása, kicsinyítése, tükrözése, tetszőleges transzformációja, részekre bontása, ablakozása, több grafikus állomány összemácsolása. Az egész programcsomag egy átlapolási technikával dolgozó program vezérlése alá vonható. A modulokat az első kivételével érdemes lefordítani, bár a két nyomtatódokumentum futási idejének nagyobbik részét a nyomtató kiszolgálása viszi el.

Bármilyen konfigurációban és kiegészítésekkel készítjük el a programcsomagot, ne feledjük el, hogy a rajzok minden pontját meg kell tervezni. Az 1. modulnál a fejlesztés célja lehet néhány rutin kidolgozása is például egyenes, kör, egyes speciális alakzatok rajzolásához vagy sablon készítése kapcsolási rajzok, blokk-diagramok előállításához.

Dr. Hack Frigyes

### 3. lista

```

0 REM"@MODUL-#2"
10 REM *****
20 REM * GRAFIKUS NYOMTATAS *
30 REM * NLQ FELIRATOKKAL *
40 REM *****
50 REM * LEMEZROL => MUNKAKOPIA *
60 REM *** *****
70 INPUT " ABRA-NEV :";FL$
80 :
90 REM ' NYOMTATO VEZERLO JELEK '
100 BS$=CHR$(108)+CHR$(10) :REM MARGO
110 LP$=CHR$(51)+CHR$(21) :REM 8 LPI
120 ES$=CHR$(27) :REM ESCAPE
130 CR$=CHR$(13) :REM RETURN
140 BS$=CHR$(8) :REM GRAFIKA
150 SO$=CHR$(14) :REM DUPLA
160 SI$=CHR$(15) :REM D/G OFF
170 PS$=CHR$(16) :REM POZICIO
180 OPEN 8,8,8,FL$+ ". [G],S,R"
190 INPUT#8,GV,GF :REM RAJZFELULET
200 REM - - - - - FELSO/ALSO KERET
210 KV=INT(GV/6)+1
220 FOR P=1 TO KV
230 : FK$=FK$+" " : AK$=AK$+"T"
240 NEXT P
250 PS$=PS$+RIGHT$(STR$(KV+101),2)
260 REM - - - - - SP-180 VC ELOKESZITESE
270 OPEN 4,4,7
280 PRINT#4,ES$BMS;
290 PRINT#4,SO$ " \NKA K PIA : ";
300 PRINT#4,FL$SI$CR$
310 CLOSE 4
320 REM >>> --- NYOMTATAS --- >>>
330 OPEN 4,4
340 PRINT#4,ES$LP$
350 PRINT#4," ";FK$
360 FOR S=0 TO GF : REM 7*GF PIXEL-SOR
370 : PRINT#4,SI$+"I"BS$;
380 : FOR O=0 TO GV : REM GV PIXEL/SOR
390 REM - - - - - EGY GRAFIKUS JEL - - - - -
400 : GET#8,G$ : PRINT#4,G$;
410 : NEXT O
420 : PRINT#4,SI$PS$ "F";
430 : IF S/2=INT(S/2) THEN PRINT#4,S;
440 : PRINT#4,BS$
450 NEXT S
460 PRINT#4,SI$ " ";AK$CR$
470 PRINT#4," ";
480 FOR A=0 TO KV-1 : REM SKALAZAS
490 : PRINT#4,RIGHT$(STR$(A),1);
500 NEXT A : PRINT#4
510 PRINT#4," ";
520 FOR A=1 TO INT(KV/10)
530 : PRINT#4," " : PRINT#4,A;
540 NEXT A : PRINT#4,CR$CR$
550 CLOSE 4 : CLOSE 8
560 END
  
```

### 4. lista

```

0 REM"@MODUL-#3"
10 REM *****
20 REM * GRAFIKUS NYOMTATAS *
30 REM * NLQ FELIRATOKKAL *
40 REM *****
50 REM * FELIRATOK => LEMEZRE *
60 REM *****
70 INPUT " ABRA-NEV :";FL$
80 :
90 REM ' NYOMTATO VEZERLO JELEK '
100 ES$=CHR$(27) :REM ESCAPE
110 CR$=CHR$(13) :REM RETURN
120 PS$=CHR$(16) :REM POZICIO
130 SO$=CHR$(14) :REM DUPLA
140 SI$=CHR$(15) :REM SZIMPLA
150 DIM W$(6,1),U$(6) :REM - BETUTIPUSOK
160 W$(1,1)=ES$+"4" :U$(1)="ITA"
170 W$(1,0)=ES$+"5" :REM - DOLT BETU
180 W$(2,1)=ES$+"E" :U$(2)="FET"
190 W$(2,0)=ES$+"F" :REM - KOVER BETU
200 W$(3,1)=ES$+"-1" :U$(3)="SUB"
210 W$(3,0)=ES$+"-0" :REM - ALAHUZOTT
220 W$(4,1)=SO$ :U$(4)="DPL"
230 W$(4,0)=SI$ :REM - DUPLA SZELES
240 W$(5,1)=ES$+"S0" :U$(5)="EXP"
250 W$(5,0)=ES$+"T" :REM - FELSO INDEX
260 W$(6,1)=ES$+"S1" :U$(6)="IND"
270 W$(6,0)=ES$+"T" :REM - ALSO INDEX
280 OPEN 8,8,8,FL$+ ". [G],S,R"
290 INPUT#8,GV,GF :REM RAJZFELULET
300 CLOSE 8
310 DIM F$(GF) :REM - FELIRATOK
320 KV=INT(GV/6)+1 :REM SOR HOSSZA
330 REM >>> --- FELIRATOZAS --->>>
340 INPUT " ABRA CIME :";FE$
350 GOSUB 540 : FC$=FE$
360 REM CIKLUS : A FELIRATOK BEIRASA ---
370 : INPUT " SOR,POZ. =";S,P
380 : IF S<0 THEN 440
390 : IF P<0 OR S>GF OR P>KV THEN 370
400 : INPUT " ZOVEG :";FE$
410 : FE$=PS$+RIGHT$(STR$(102+P),2)+FE$
420 : GOSUB 540 : F$(S)=FE$
430 GOTO 370 : REM CIKLUS VEGE ---
440 INPUT " ALAIRAS :";FE$
450 GOSUB 540 : AS$=FE$
460 REM <<< --- BEIRAS VEGE --- >>>
470 REM >>> --- LEMEZRE --- >>>
480 OPEN 8,8,8,FL$+ ". [F],S,W"
490 PRINT#8,FC$
500 FOR S=0 TO GF : PRINT#8,F$(S) : NEXT
510 PRINT#8,AS$ : CLOSE 8
520 END :REM <<<<<< V E G E ---<<<<
530 :
540 REM ***** ALPROGRAM: BETUTIPUS *****
550 PRINT " IIPUSOK (I/N) "
560 FOR T=1 TO 6 : PRINT " U$(T)";
570 GET T$:IF T$="" THEN 570
580 IF T$<>"I" THEN 600
590 FE$=W$(T,1)+FE$+W$(T,0) :PRINT"*";
600 NEXT T : PRINT
  
```

### 5. lista

```

0 REM"@MODUL-#4"
10 REM *****
20 REM * GRAFIKUS NYOMTATAS *
30 REM * NLQ FELIRATOKKAL *
40 REM *****
50 REM * N=Y=O=M=T=A=S *
60 REM *****
70 INPUT " ABRA-NEV :";FL$
80 :
90 REM ' NYOMTATO VEZERLO JELEK '
100 BS$=CHR$(108) :REM B.MARGO
110 ES$=CHR$(27) :REM - ESCAPE
120 CR$=CHR$(13) :REM - RETURN
130 CH$=CHR$(141) :REM - VISSZA
140 BS$=CHR$(8) :REM GRAFIKA
150 SO$=CHR$(14) :REM - DUPLA
160 SI$=CHR$(15) :REM G/D OFF
170 DN$=CHR$(17) :REM - SET 2.
180 PS$=CHR$(16) :REM POZICIO
190 OPEN 7,8,7,FL$+ ". [G],S,R"
200 INPUT#7,GV,GF :REM RAJZFELULET
210 KV=INT(GV/6)+1 :REM - SORHOSSZ
220 BM=INT((80-KV)/2) :REM - KOZEPRE
230 BMS=BMS+CHR$(BM)
240 REM --- KERETEK ---
250 FOR I=0 TO KV : K$=K$+"- " : NEXT I
260 KF$=" "+K$+"- " : KA$="L"+K$+"J"
270 K$="I" : PS$=PS$+RIGHT$(STR$(KV+2),2)
280 DIM F$(GF)
290 REM >>>--- SZOVEGEK BEOLVASASA --->>>
300 OPEN 8,8,8,FL$+ ". [F],S,R"
310 INPUT#8,FC$ :REM - CIMFELIRAT
320 FOR S=0 TO GF :REM - FELIRATOK
330 : F$(S)=" "
340 : GET#8,Q$ : IF Q$=CR$ THEN 360
350 : F$(S)=F$(S)+Q$ : GOTO 340
360 NEXT S
370 AS$=" " :REM - - - ALAIRAS
380 GET#8,Q$ : IF Q$=CR$ THEN 400
390 AS$=AS$+Q$ : GOTO 380
400 CLOSE 8 :REM <<< BEOLVASAS VEGE ---<<<
410 REM >>>--- NYOMTATAS --->>>
420 OPEN 4,4 : PRINT#4,ES$BMS;
430 PRINT#4,DN$FC$CR$ :REM - - - CIM
440 PRINT#4,KF$BS$ :REM - F.KERET
450 PRINT#4,SI$K$PS$K$BS$ :REM - URES
460 FOR S=0 TO GF :REM SORONKENT
470 : PRINT#4,SI$ "BS$;
480 : FOR O=0 TO GV :REM - GRAFIKA
490 : GET#7,G$ :PRINT#4,G$;
500 : NEXT O
510 : PRINT#4,SI$CH$; :REM - VISSZA
520 : PRINT#4,SI$K$; :REM - B.KERET
530 : PRINT#4,DN$F$(S); :REM - SZOVEG
540 : PRINT#4,PS$K$BS$ :REM - J.KERET
550 NEXT S
560 PRINT#4,SI$K$PS$K$BS$ :REM - URES
570 PRINT#4,SI$KA$ :REM - A.KERET
580 PRINT#4,DN$;A$ :REM - ALAIRAS
590 CLOSE 4 : CLOSE 7
600 END :REM <<<<<< V E G E ---<<<<
  
```

A C64-en a beépített BASIC interpreter miatt sokan használják a BASIC nyelvet. Ezáltal programjaink belövése gyors és közvetlen, de kész programjaink futása meglehetősen lassú. A sebesség növelésének több módja is lehet: gépi kódban programozunk, egy másik magas szintű programnyelvvél próbálkozunk, vagy kész programunkat fordítóval lefordítjuk. Ha lusták vagyunk, az utolsó lehetőséget választjuk.

## Mitől gyorsabb a lefordított program?

Ahhoz, hogy erre a kérdésre válaszolni tudjunk, nézzük meg az interpreter és a fordító működése közötti különbséget. Futtatáskor az interpreter a programot soronként értelmezi, és a tárolt utasításkód (token) alapján meghívja a megfelelő ROM-rutinokat. Ha egy sorra többször kerül rá a vezérlés, akkor ezt a sort mindig újra kell értelmezni. További lassulást okoz az a tény, hogy az egész típusú műveleteknél a számolás mindig lebegőpontos alakban történik, ezért előtte egy egész → lebegőpontos, utána egy lebegőpontos → egész konverzióra van szükség. A konverzió miatt a program lassabban fut, mint ha lebegőpontos számokat használnánk. Nagyobb programoknál a GOTO-GOSUB utasítások végrehajtása és a változók feldolgozásának sebessége sokkal lassabb.

A fordítóprogram a sorokat csak egyszer, a fordítás alatt értelmezi. A fordító a BASIC programot egy gyorsabban futtatható kóddá, pszeudokóddá (P-kód) alakítja át. Van olyan fordító, amely gépi kódot is képes generálni. A fordítás többmenetes, közben átmeneti állományok is keletkeznek, ezért a fordítók működéséhez lemezegységre van szükség.

A sebességnövekedésnek további okai is vannak:

- A P-kódban a változók címzése közvetlenné válik, azaz nem a nevek, hanem a címük azonosítja a változókat.
- A kifejezések már a fordítás alatt kiértékelődnek.
- A GOTO-GOSUB ugrások címei a fordítás alatt keletkeznek, ezért az ugrások közvetlen címzésűek.
- A konstansok átalakítása fordításkor megtörtént.
- Gyorsul a szemégyűjtés (garbage collection) (PETSPEED, BASIC 64).

— Egész típusú számbábrázolás valósul meg.

— A karakterfüzér-műveletek kiszámítása gyors.

A P-kódra fordított programok mérete csökken. A működéshez azonban szükség van egy rutingűjteményre (runtime modul), melyet minden programhoz hozzáilleszt a fordító. Így érthetővé válik az a jelenség, hogy rövidebb programjaink fordítás után hosszabbak lesznek. Nagyobb programoknál a P-kód méretcsökkenése kompenzálja a runtime modul mintegy 4-5 kb-ajos hosszát, így a fordított program rövidebb lesz.

A fordítók képességeik szerint két csoportba sorolhatók.

Az egyikbe tartozik az AUSTRO-COMP, az AUSTRO-SPEED, a BLITZ és a SIMON'S COMP (ez utóbbi a Simon's BASIC utasításait is fordítja), a második csoportba pedig a PETSPEED és a BASIC 64. Az első csoportba tartozó fordítók az AUSTRO-COMP továbbfejlesztett változatainak tekinthetők. A többi már más működési elven dolgozik.

## Teszt

A fordítók határfokát az alábbi tesztprogrammal vizsgáltuk. A példa-program az első 20 000 számból kiválasztja a prímszámokat Eratoszteniész szitája segítségével. A prímszámokat nem írja ki, mert a számítási sebességek összehasonlítása volt a cél.

```
100 REM *** ERATOSZTENESZ SZITAJA ***
110 TI$ = "000000"
120 DIM P%(10000)
130 FOR C=1 TO 10000
140 P%(C)=1
150 NEXT
160 PRINT TI$
170 FOR C=1 TO 10000
180 IF P%(C) THEN 250
190 P%*C*2+1
200 K71
210 IF C+K*(C*2+1)>10000 THEN 250
220 P%(C+K*(C*2+1))=0
230 K=K+1
240 GOTO 210
250 NEXT
260 PRINT TI$
```

# A lusta is győz több menetben

## A TESZTEK EREDMÉNYEI

A fordító neve	Programhossz	Tömbfeltöltés	Futásidő
BASIC INTERPRETER	2 blokk	57 mp (100%)	12 p. 37 mp (100%)
AUSTRO-COMP	17 blokk	19 mp (33%)	3 p. 28 mp (27%)
AUSTRO-SPEED	25 blokk	15 mp (26%)	2 p. 23 mp (18%)
BLITZ	25 blokk	15 mp (26%)	2 p. 23 mp (18%)
SIMON'S COMP	17 blokk	19 mp (33%)	3 p. 28 mp (27%)
PETSPEED	112 blokk	6 mp (10%)	2 p. 09 mp (17%)
BASIC 64 (1-P)	23 blokk	19 mp (33%)	4 p. 37 mp (36%)
BASIC 64 (1-M)	24 blokk	16 mp (28%)	3 p. 45 mp (29%)
BASIC 64 (2-P)	22 blokk	8 mp (14%)	1 p. 30 mp (11%)
BASIC 64 (2-M)	24 blokk	6 mp (10%)	50 mp (6%)

A BASIC 64 következő üzemmódjait teszteltük:

- 1—P A számokat a megadott típusra (Integer, Real) fordítja, a lefordított program P-kódú.
- 1—M A számokat a megadott típusra (Integer, Real) fordítja, a lefordított program gépi kódú.
- 2—P Minden számot Integer típusra fordít, a lefordított program P-kódú.
- 2—M Minden számot Integer típusra fordít, a lefordított program gépi kódú.

## Az AUSTRO-COM, AUSTRO-SPEED, SIMON'S COMP használata

A fordító indítás után a lefordítandó program nevét kéri. Két menetben fordít, majd kiírja a hibák és a bővítések számát. A lefordított program a lemezen C/név formában található. Fordításkor keletkezik a Z/név állomány. Ha a lefordított programban futás közben hiba keletkezik, akkor a hibás BASIC-sor visszakereshető. Nézzük meg ezt egy konkrét példán:

```
10 INPUT A
20 PRINT 1/A
30 GOTO 10
```

A fordítás után keletkezett Z/PLD állomány:

```
6047=10
6050=20
6054=30
```

Futtatva a C/PLD programot, adatbevitelnél nullát megadva, a program a következő hibaüzenetet írja ki:

```
?DIVISION BY ZERO ERROR IN 6052
READY.
```

Ilyenkor töltsük be a Z/PLD állományt, és adjuk ki a következő parancsot: LIST —6052. Példánkban ez lesz az eredmény:

```
6047=10
6050=20
```

Ebből megállapítható, hogy a 20-as sorszámú BASIC-sorban van a hiba.

Mivel a BLITZ kétmeghajtós lemezegységgel vagy két külön meghajtóval is képes dolgozni, először ki kell választanunk a használni kívánt lemezegységet. A továbbiak megegyeznek az AUSTRO fordítók működésével.

## PETSPEED

Az egyetlen fordító, amely négy menetben dolgozik. A fordítandó programot a PETSPEED-et tartalmazó lemezre kell másolni. A lefordított program a lemezen név.WOW néven, a sorlista név.W néven keletkezik. Ha fordítás közben a lemez betelik, akkor a fordító megkérdezi, hogy törölheti-e a forrásprogramot. Ha a lefordított program futása közben hiba keletkezik, annak helyét az ERRORS programmal határozhatjuk meg. A tesztprogram 112 blokkos mérete azzal magyarázható, hogy a tömbterület a runtime modul és a P-kód közé kerül. A listából látszik, hogy a tesztprogram kb. 20 000 bájtost tömbbel dolgozik. A 80 blokknál nagyobb programok fordításához célszerű kétmeghajtós lemezegységet használni.

## BASIC 64

A tesztből látható, hogy a legjobb futásidőjű kódot fordítja, ezért részletesen is ismertetjük kezelését. Indítás után a képernyőn megjelenik a főmenü:

- 1 — Optimalizálás I.
- 2 — Optimalizálás II.
- 3 — Paraméter-beállítás
- 4 — Overlay

1. A változók kezelése a programban megadott típusnak megfelelően történik.

2. Minden numerikus változót egészként kezel! Minden szám egész, értéke (-32768,32767) között. Az osztás eredménye is egész!

3. A = az előállított kód típusa:

- P-kód,
- 6502/6510 kód,
- nem állít elő kódot.

B = szimbólumtábla betöltése. Egy másik program fordításakor előállított szimbólumtábla használata. Nem csak a BASIC 64 által előállított lista lehet, hanem például a PROFI-ASS 64 is előállíthatja, mondjuk overlay-alkalmazásoknál.

C = szimbólumtábla tárolása.

D = sorlista-generálás. Ez a többi fordítónál a Z/név állománynak felel meg.

E = az elérhető memória vége.

F = kódstart. A generált kód alapértelmezésben közvetlenül a runtime modul után kezdődik, itt tolható el a kezdőcím.

G = a runtime modul előállításának elhagyása: például overlay programok esetén csak az első programhoz kell ez az 5 kbájtos modul.

H = bővítések. A program a következő bővítéseket tudja kezelni: Supergrafik 64+, Simon's BASIC, EXBASIC Level II, BASIC 4.0, Supergrafik 64, valamint egyéb bővítéseket, melyek paramétereit az I,J pontokban kell beállítani.

I = a bővítésben használt tokenbájtok száma.

J = az ELSE utasítás tokenkódja.

K = hibakezelő sor. Ha a fordított program futásakor hiba keletkezik, akkor ha itt megadtunk egy sorszámot, a vezérlés arra a sorra kerül.

L = overlay. A karakterfüzerek kezelése overlay üzemmódban más lesz.

M = lemezparancsok kiadása, hibacsatorna lekérdezése.

4. Overlay programok fordítása. Egymást töltő és indító programok fordításakor használjuk. A fordítás kétmenetes: először az összes programot le kell fordítani az első menettel (PASS 1), majd a másodikkal (PASS 2).

Gyakran fordítás közben is szeretnénk megváltoztatni a fordító paramétereit. Ezt az alábbiak szerint tehetjük meg a programból:

REM @ M — A fordítás 6502/6510 kódra történik.

REM @ P — Fordítás P-kódra.

REM @ E sorszám — Hibakezelés a 3.E pontban leírtaknak megfelelően.

REM @ I = változó, ... — A felsorolt változók egész típusúak.

REM @ R = változó — A felsorolt változók valós típusúak.

REM @ O fokozatszám — Optimalizálási fokozat beállítása.

REM @ A változó = cím — A változó a megadott címre kerül. 768-nál alacsonyabb címek nem használhatók.

REM @ S cím — Alapértelmezésben a fordított program használja a kettapuffert. Ha erre a területre (828—1019) szükségünk van, akkor a programba egy R @ S1019 utasítást írunk be.

Reméljük, hogy a leírások alapján mindenki ki tudja választani a számára legalkalmasabb fordítóprogramot.

Bakos Imre—Kelemen Róbert

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

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

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

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

## 11. feladat:

### Egér a labirintusban 1.

Tegyük az egeret a labirintus bejáratához, a sajtot pedig a kijáratához. Megtalálja-e az egér a sajtot? Ön, mint az egér jó barátja, elhatározta, hogy segít neki. Hogyan? Írjon erre programot! Tegyük fel, hogy a labirintus térképe nem ismert, az egér csak az őt közvetlenül körülzáró falakat látja.

### Megoldás

Ennek a feladatnak az elvi megoldása olyannyira egyszerű, hogy bizonyára sok olvasónk meglepődik rajta.

Ha megvizsgálunk egy egybejáratú és egykijáratú labirintust, láthatjuk, hogy összefüggő falrendszerekből áll. A bejárat jobb oldali fala összefügg a kijárat jobb oldali falával és viszont. Ha a bejárat és a kijárat között több mint egy út vezet, akkor elképzelhetők olyan falrendszerek is, amelyek nem csatlakoznak sem a bejárat, sem a kijárat falaihoz. Minden esetben a bejáratától a ki-

járatig el lehet jutni végig a fal mellett haladva.

Most már könnyen adhatunk tanácsot az egerünknek: menj mindig a jobb oldali fal mentén, s biztos megleled a sajtot!

Az egér mozgását bemutató — Turbo—Pascal 4.0 nyelvű, IBM PC-n tetszőleges grafikus adapterrel működtethető — program megírásához először egy labirintust kellett rajzolni. Ez a Labirintus szubrutin feladata. A szubrutin az előző számban közölt program csekély átalakításával keletkezett. Nézzük a gondolatmenét és a működését:

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

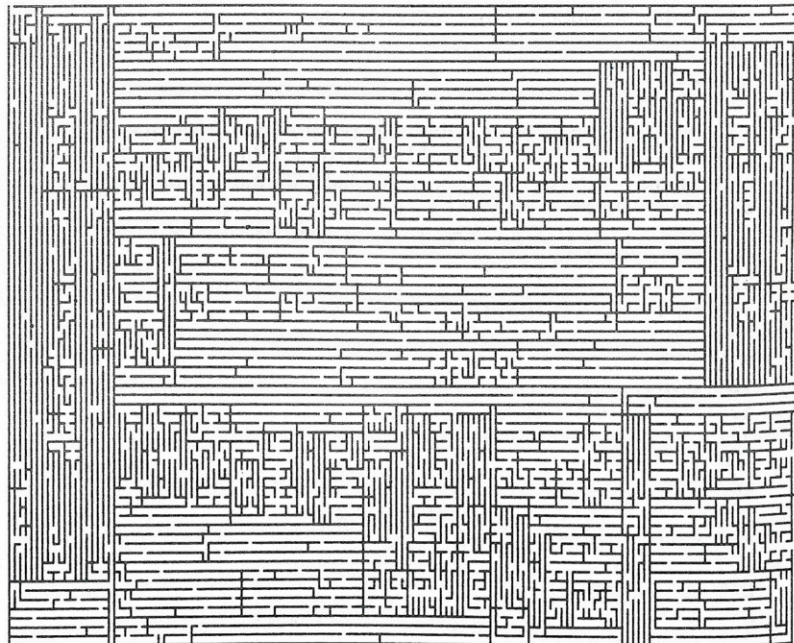
— Ismételjük meg ezt az eljárást a jobb és a bal oldali téglalpra!

— Folytassuk az eljárást mindaddig, míg a téglalapok tovább már értelmesen nem oszthatók!

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

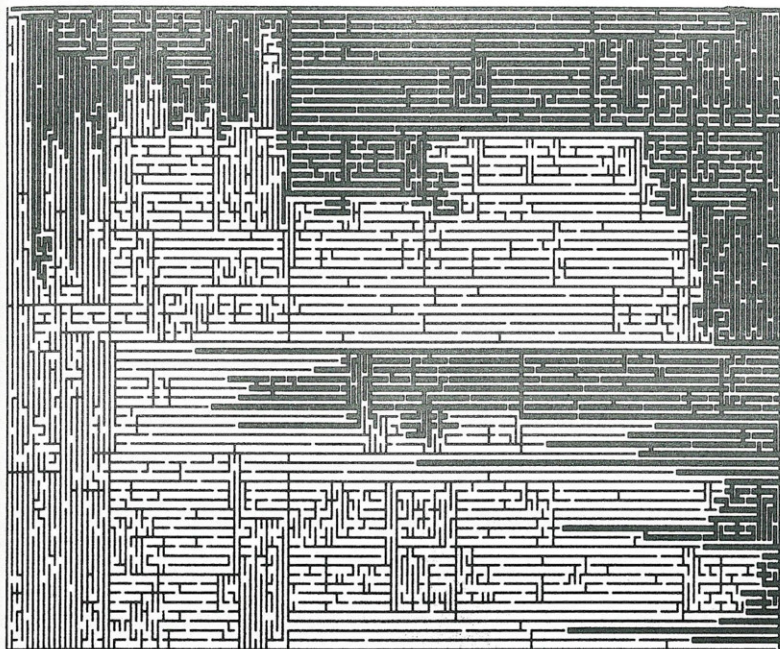
## A labirintus

### 1. ábra



# FELADATOK

## MEGOLDÁSOK



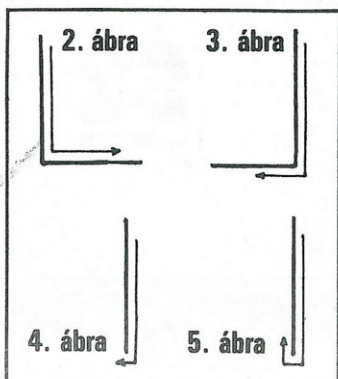
Az egér útja a labirintusban

6. ábra

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.

A fal két oldalán keletkezett téglalapokat az Xfelez vagy az Yfelez osztja fel tovább, a KOMPL konstans értékétől függően eltérő valószínűséggel. Ezzel a labirintus bonyolultságát lehet szabályozni.

A szubrutinnak nem marad más feladata, mint hogy a külső keretet, a be- és a kijáratot meg-



```

-----
{
  Eger a labirintusban
}
{
  Pinter Gabor
}
{
  1988.10.06.
}
-----

program eger;
uses
  Graph;

const
  SZ:=3; { Folyosó szélesség }
  KOMPL:=8; { Labirintus bonyolultság }

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

procedure Labirintus;

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

{ Fuggolegesen kettéosztja a tablát }
procedure Xfelez(
  x,y, { Bal felső sarok koordinátái }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztófal helyzete: }
  oszt : integer;
  { Az osztófalra a lyuk helye: }
  lyuk : integer;
begin
  { Kell e tovább osztani? }
  if (dx)>=2*SZ and (dy)>=2*SZ
  then begin
    { Fal rajzolása: }
    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 két oldalának levo
    tablak továbbosztása: }
    if Random(10)>KOMPL
    then
      Xfelez(x,y,oszt-x,dy)
    else
      Yfelez(x,y,oszt-x,dy);
  end;
end;

```

rajzolja. A labirintus rajzolását az innen meghívott Xfelez végzi el.

Ezután az egér elkezd a labirintus bejárását. A főprogram elején az egér kiindulási pontját úgy adjuk meg, hogy az a bejárat jobb oldalán legyen. Ettől kezdve az egérnek a fal mentén kell csak haladnia, hogy megtalálja a sajtot.

Az egér két dolog miatt kényeszerűlhet arra, hogy a haladási irányát megváltoztassa:

— Előtte fal van. Ekkor a 2. ábrán látható módon balra kell kanyarodnia.

— Jobbra tőle megszakad a fal, új folyosó bejárata van. Ekkor a falat követve jobbra kell fordulnia (3. ábra).

Nem szükséges megkülönböztetni a 4. ábrán látható esetet, mert ekkor a következő lépésben az egér el tud fordulni az 5. ábrán látható módon.

A programban az egér négy lehetséges haladási iránya miatt az itt leírtakat megvalósító programrészt case struktúrába van ágyazva. Az egér tényleges mozgását a második case látja el, majd ezt követi az egér útjának megjelölése. A 6. ábrán látható, hogy az egér tényleg kitalált a labirintusból.

A következő feladat a mostaniak látszólag csak kicsit módosított változata.

### 12. feladat: Egér a labirintusban 2.

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

Pintér Gábor

```

if Random(10)>KOMPL
then
  Xfelez(oszt,y,x+dx-oszt,dy)
else
  Yfelez(oszt,y,x+dx-oszt,dy)
end {if};
end {Xfelez};

{ Viszintesen kettéosztja a tablát }
procedure Yfelez(
  x,y, { Bal felső sarok koordinátái }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztófal helyzete: }
  oszt : integer;
  { Az osztófalra a lyuk helye: }
  lyuk : integer;
begin
  { Kell e tovább osztani? }
  if (dx)>=2*SZ and (dy)>=2*SZ
  then begin
    { Fal rajzolása: }
    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 két oldalának levo
    tablak továbbosztása: }
    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,y,dx,oszt-y)
    else
      Yfelez(x,y,dx,oszt-y);
  end {if};
end {Yfelez};

begin
  { Keret a be- és kijáratnál }
  MaxX:=SZ*(GetMaxX div SZ);
  MaxY:=SZ*(GetMaxY div SZ);
  Line(0,0,0,MaxY);
  Line(0,MaxX,MaxX-SZ,MaxY);
  Line(MaxX,MaxX,MaxX,0);

  Line(MaxX,0,SZ,0);

  { Labirintus rajzolása }
  Xfelez(0,0,MaxX,MaxY);
end {Labirintus};

```

```

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

  { Labirintus rajzolása }
  Labirintus; Fal:=GetMaxColor;

  SetColor(13);
  { Eger kiindulási helyzet }
  EgerX:=1; EgerY:=0;
  PutPixel(EgerX,EgerY,13);
  { Az eger kezdeti haladási iránya }
  EgerIrany:=LE;

  repeat
    { Milyen irányba mozdulhat? }
    case EgerIrany of
      LE :
        { Eger előtt fal van-e? }
        if GetPixel(EgerX-1,EgerY)<>Fal
        then EgerIrany:=BALRA
        else
          { Eger előtt jobbra fal van-e? }
          if GetPixel(EgerX,EgerY+1)=Fal
          then EgerIrany:=JOBBRA;
        BALRA :
          { Eger előtt fal van-e? }
          if GetPixel(EgerX,EgerY-1)<>Fal
          then EgerIrany:=FEL
          else
            { Eger előtt jobbra fal van-e? }
            if GetPixel(EgerX-1,EgerY)=Fal
            then EgerIrany:=LE;
        FEL :
          { Eger előtt fal van-e? }
          if GetPixel(EgerX+1,EgerY)<>Fal
          then EgerIrany:=JOBBRA
          else
            { Eger előtt jobbra fal van-e? }
            if GetPixel(EgerX,EgerY-1)=Fal
            then EgerIrany:=BALRA;
        JOBBRA :
          { Eger előtt fal van-e? }
          if GetPixel(EgerX,EgerY+1)<>Fal
          then EgerIrany:=LE
          else
            { Eger előtt jobbra fal van-e? }
            if GetPixel(EgerX+1,EgerY)=Fal
            then EgerIrany:=FEL;
        end {case};

      { A meghatározott irányba lép }
      case EgerIrany of
        LE : EgerY:=EgerY+1;
        BALRA : EgerX:=EgerX-1;
        FEL : EgerY:=EgerY-1;
        JOBBRA : EgerX:=EgerX+1;
      end {case};

      { Az eger utját jelezni kell! }
      PutPixel(EgerX,EgerY,13);
    until EgerY=MaxY;
  readln;
end.

```

# Könnyen választhat a SZÁMALK MENÜ-jéből

A SZÁMALK értesíti az érdeklődőket, hogy a MENEZSER-üzletág — a SZÁMALK-MENÜ — forgalmazza az alábbi magyar termékeket

## SEGÍT szoftver

IBM PC XT/AT kompatibilis gépekre  
Tárrezidens DOS HELP  
Minden DOS felhasználónak magyar nyelvű tájékoztatást ad  
KÉPERNYŐN DOS-használat közben  
Gyűjtött és szerkeszthető DOS-utasítások  
A felhasználó bővítheti, programozhatja saját SEGÍT-jét  
4500 Ft

## Szünetmentes tápegység

600 VA-es  
3 óras áramkimaradás mellett is tovább dolgozhat  
IBM PC XT/AT kompatibilis gépen  
AKKU-párolgás elleni védelemmel  
Mintadarab a BIT-BOLT-ban megtekinthető  
1 db AT 1 db XT és FX1000 együttesen áramvédelem  
125 ezer Ft

## SECRET

Védi dokumentumait IBM PC/86-286-386 computerén az illetéktelenektől a SECRET adat- és programvédelmi rendszer szoftver-hardver eszközökkel

## Kit, mit:

- a felhasználót
- a directoryt
- a programfájlt
- az adatfájlt is

## Mivel:

- passworddel
- kóddal
- hardverkártával (opció)
- EPROM-kulccsal (opció)

## Hogyan:

- dekódolhatatlan passwordkód bekevert kombinációja
- hozzáférés előtt illetékesnek nyit, rögtön utána zár a védelem (sebességcsökkenés maximum 10%)
- hardveres opció esetén floppy-boot letiltás
- törölhetetlen fájlok
- átnevezhetetlen fájlok
- a lemásolt védett fájlokat illetéktelenek nem használhatják

## Kitől:

- más illetéktelen felhasználótól
- szükség esetén a systemmenedzsertől is

## Hol:

- egy gépen
- hálózatba épített több gépen is

## Meghibásodáskor:

- a systemmenedzser hardver úton felélesztheti a rendszert

## MEDANIN

Az orvos, klinikus és farmakológus gyakorlati munkáját segíti az adattárolásban és az analízisben. Válaszok biológiai, diagnosztikus és terápiás kérdésekben. MEDikusok ANAlízise és INformációs szoftverje a MEDANIN. Készítette prof. dr. Berentei és prof. dr. Kékesi.  
100 ezer Ft

## KIR

Winchesterén és floppyjain tárolt fájljaiból directory csoportosításban saját katalógusfájlt készíthet, megjegyzésekkel elláthatja, csoportosíthatja, dBASE típusú katalógusfájlt tovább feldolgozhatja  
15 ezer Ft

Rendelésre — rövid határidővel — szállítjuk a fenti szoftvereket, illetve hardver kiegészítőket. Felhívjuk szíves figyelmüket, hogy megadott áraink nettó — áfa nélküli — árak

**Kérésére árkatálogust küldünk több mint 1500 legális szoftverre**

**Szükség esetén tanácsadás**

**Kérjen levélben ajánlatot, árkatálogust!**

**Válaszunk után írásban rendeljen! Rövid szállítási határidő!**

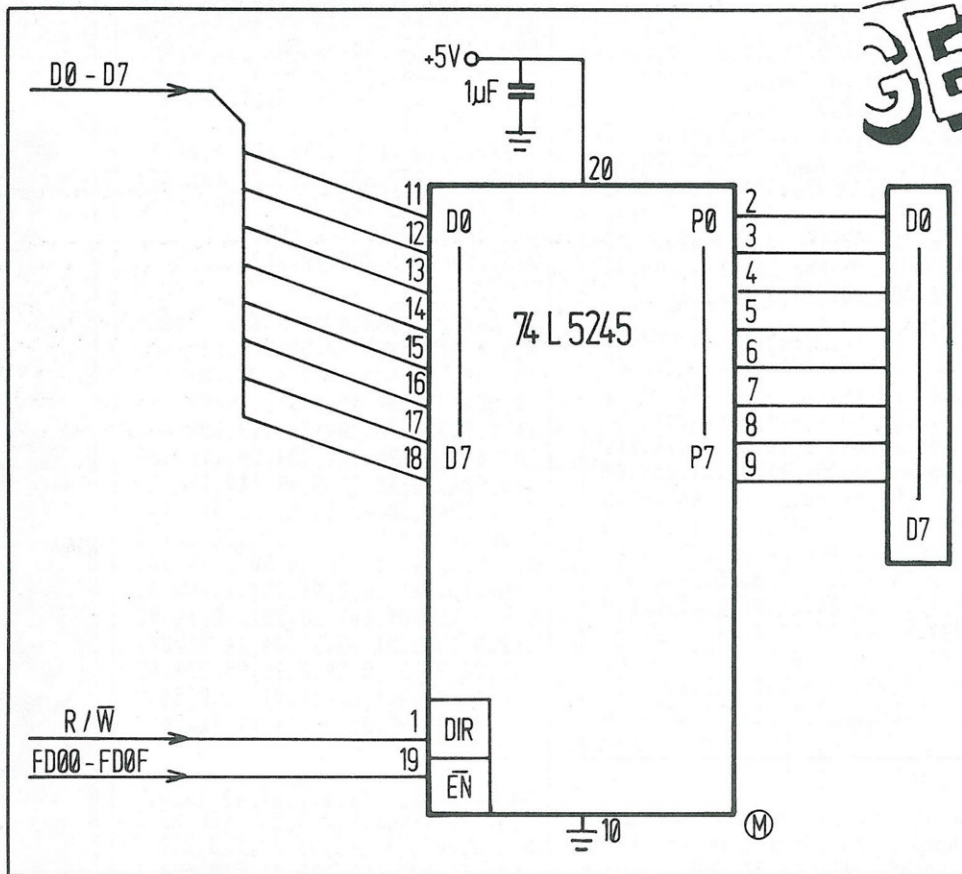
**A MENÜ iroda rendelési címe:**  
1123 Budapest, Kapitány u. 6. I/1.  
Telefonszám: 110-983

**A BIT-BOLT címe:**  
1136 Budapest, Raoul Wallenberg u. 5.

Tisztelettel a  
SZÁMALK-MENÜ  
ÜZLETÁG



**Könnyen választhat a SZÁMALK MENÜ-jéből**



1. ábra

Bizonyára sok C16-tulajdonos elgondolkozott már azon, hogy vajon miért nincs a gépén felhasználói csatlakozó. E kérdést én is megrágtam, de választ nem találva, magam kezdtem hozzá egy user port építéséhez. Az áramkört úgy terveztem, hogy a gép memóriabővítőjéről működjön.

Az 1. ábrán látható IC kétirányú sínmeghajtó erősítő áramkör. Az egész IC-t a 19-es lábára adott engedélyező jel vezérli. Az írás-olvasás irányát az 1-es lábára adott jellel vezérelhetjük meg.

A user port kiválasztásához — a Plus/4-től eltérően — én azt a megoldást választottam, hogy a 7700-10 IC 15., FD00-FD0F címzésű, bekötetlen lábával engedélyezem a portot.

**Hess,  
te csúnya!**

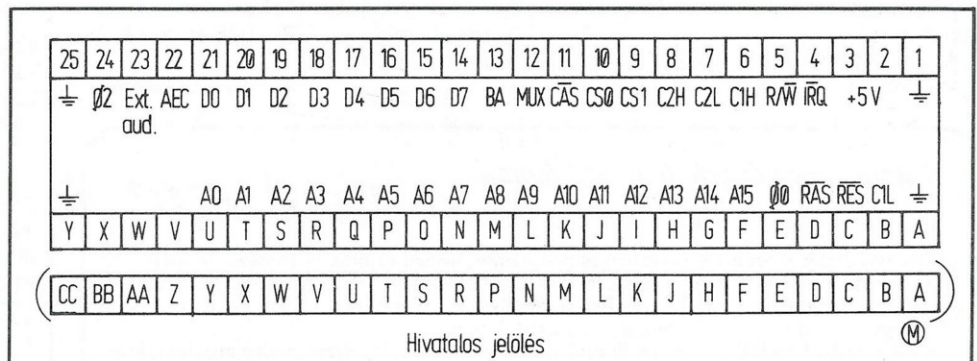
Ezután jön az építés neheze, mivel a 15-ös láb sehová sincs kivezetve. Ezt saját magunknak kell megoldanunk. A memóriabővítő (2. ábra) Z, AA, BB lábai bekötetlenek, ezért ide lehet vezetékkel a 15-ös lábat kivezetni. Ehhez a megoldáshoz egy csavarhúzó, forrasztópáka (nem pillanatforrasztó!) és mintegy 15-20 cm huzal szükséges.

A fedelet és a panelt borító hűtőlapot a csavarok kicsavarásával eltávolítjuk, majd a panel alsó felét borító fémlapot „operáljuk” le. Már csak a 7700-10 IC 15-ös lábát és a memóriabővítő bekötetlen kivezetését kell megtalálnunk, és máris forraszthatunk.

A user port panelt a memóriabővítő kivezetéseinek figyelembevételével alakítsuk ki.

Horváth Gábor

2. ábra



Sokan szeretnék megvédeni programjukat, listázását megakadályozni. Ehhez nyújt hathatós segítséget a következő program, amely a CTRL+ESC használatát hatástalanítja:

**TVC**

```

1! .....
..
- 30 db pont -
FOR A=6643 TO 6670: INPUT B:
POKE A, B: NEXT A
A beírandó értékek:
243,33,70,11,54,195,35,54,
1,35,
54,26,251,201,197,213,229,
33,22,
11,54,0,225,209,193,241,251,
201
Indítás: X=USR 6643
    
```

A program felhasználja azt, hogy az operációs rendszer, az editor kezelése folyamán a 2851 dec címre ugrik. Ezen a címen egy, az editor számára nélkülözhetetlen gépi nyelvű szubrutin található. Azt, hogy az operációs rendszer megszakításkor melyik címre ugorjon, a 49 dec címen kezdődő két bajt határozza meg. Az ezt kihasználó, gépi nyelvű szubrutinoknak egy JP 2851 utasítással kell befejeződniük.

Szalontai Béla

# Grafikus monitor

Bizonyára sokan szerettek volna már megírni grafikai megoldásokat beépíteni saját programjukba, esetleg ellenőrizni sajátjukat. Ez a monitorprogram éppen ezt teszi lehetővé. Segítségével teljes képet, ATTR-okat, grafikus karaktereket, különböző méretű sprite-okat kereshetünk meg.

A gépi kódú programot BASIC-ben is beírhatjuk, és a RANDOMIZE USR 30000-rel indíthatjuk el. Az elindítást követően megjelenik a menü. A 0 billentyű megnyomásával BASIC-be léphetünk ki. Ha 1-et nyomunk meg, akkor képernyőt kereshetünk ATTR-okkal vagy anélkül. A kívánt memóriacím beállítása után ezeknek a billentyűknek a segítségével végezhetjük a keresést: Q=memória+32; A=memória-32; O=memória+1; P=memória-1; L=ATTR-ok be; V=ATTR-ok ki.

Ha be akarjuk fejezni a keresést, akkor mindig a 0 billentyűt nyomjuk meg. Ekkor újra megjelenik a menü.

A karakterek keresését a 2-es gomb lenyomásával indíthatjuk el. A következő funkciók állnak rendelkezésünkre: Q=memória+8; A=memória-8; O=memória+192; P=memória-192; K=memória+1. Itt ugyancsak 0-val léphetünk ki.

Sprite keresésekor a program automatikusan 31 szélességű sprite-okat keres. A szélesség megváltoztatására az N és az M gombokat használjuk (N=szélesség-1; M=szélesség+1). A memóriacím a Q=memória+szélességgel, az A=memória-szélességgel, az O=memória+1-gyel, a P=memória-1-gyel változtatható. Mind a három üzemmódban a jelenlegi memória-címet a képernyő bal alsó sarkában olvashatjuk le.

Akinek nincs szüksége mind a három keresési módra, az egyenként is beírhatja a

```
10 DATA 62,7,50,72,92,50,141,92,
205,107,13,62,2,205,1,22,17,246,
119,1,141,0,205,60,32,1,254,247,
237,120,230,7,254,7,56,12,1,254,
239,237,120,230,1,254,1,40,234,2
01,203,71,40,12,203,79,202,20,11
8,203,87,204,175,118,24,200,205,
120,117,205,127,117,24,27,17,131
,120,1,10,0,201,213,177,62,1,205
,1,22,193,209,205,236,119,235,34
,14,91,205,107,13,201,42,14
```

```
20 DATA 91,17,0,64,1,0,24,237,17
6,205,163,117,24,17,1,1,0,205,10
,32,237,75,14,91,205,43,45,205,2
27,45,201,205,224,119,40,251,254
,76,40,69,254,86,40,26,42,14,91,
254,65,40,34,254,81,40,37,254,80
,40,39,254,79,40,38,254,48,202,5
6,117,24,217,33,0,88,17,1,88,1,0
,3,54,56,237,176,24,202,17,32,0,
237,82,24,12,17,32,0,25,24,6,43,
24,3,35,24,0,34,14,91,24,145,1,0
,3,237,91,14,91,33,0,24,25,17,0,
88,237,176,24,160,205,107,13,205
,120,117,205,236,119,123,178,202
,56,117,213,237,83,14,91,205,110
```

részprogramokat. A program felépítése a következő: a 10-es és a 20-as sor a kezdő memóriacímet állítja be. 30-290-ig terjed a billentyűk és a képernyő kezelése (a menü kiírása), 300-1010-ig pedig a képernyő kereshető. Az 1020-1790-es sorok a karaktereket keresik, az 1800-3150-es sorok pedig a sprite-okat.

Puskás László

```
30 DATA 13,209,33,0,64,6,32,197,
229,6,24,197,229,6,8,26,119,19,3
6,16,250,225,62,32,133,111,56,4,
24,6,24,226,62,8,132,103,193,16,
228,225,35,193,16,219,205,163,11
7,205,224,119,40,251,254,81,40,2
7,254,65,40,28,254,79,40,14,254,
80,40,25,254,40,40,50,254,75,40,
49,24,225,33,192,0,24,18,33,8,0,
24,13,17,0,0,24,20,17,192,0,24,1
5,195,56,117,237,91,14,91,25,235
,237,83,14,91,24,173,42,14,91,23
7,82,235,237,83,14,91,24,161,33,
1,0,24,227,0,195,56,117,205,120,
117,205,127,117,62,190,50,3,91,6
2,31,50,2,91,205,99,119,205,205
```

```
40 DATA 118,58,8,92,254,48,200,2
4,245,175,50,8,92,50,8,92,254,0,
40,249,254,48,200,254,79,204,6,1
19,254,77,204,22,119,254,80,204,
14,119,254,78,204,36,119,254,65,
204,64,119,254,81,204,50,119,254
,70,204,79,119,205,99,119,205,16
3,117,24,188,42,14,91,35,34,14,9
1,201,42,14,91,43,34,14,91,201,5
8,2,91,254,31,200,60,50,2,91,205
,107,13,201,58,2,91,254,1,200,61
,50,2,91,205,107,13,201,42,14,91
,22,0,58,2,91,95,25,34,14,91,201
,42,14,91,22,0,58,2,91,95,237,82
,34,14,91,201,42,14,91,22,0,58,2
,91,95,25,25,25,25,34,14,91,201
```

```
50 DATA 58,2,91,1,0,64,42,14,91,
235,42,2,91,229,213,229,120,95,2
03,63,203,63,203,63,87,123,230,7
,198,56,103,122,230,24,132,103,1
22,230,7,15,15,15,111,121,133,11
1,193,209,229,121,50,153,119,229
,235,120,1,30,0,237,176,235,71,2
25,36,124,230,7,32,10,125,198,32
,111,56,4,124,214,8,103,16,227,2
25,193,201,255,0,255,42,2,91,235
,42,4,91,26,189,40,9,19,19,122,2
54,255,200,32,244,201,19,26,188,
40,3,27,24,239,27,213,225,34,14,
91,35,35,34,2,91,201,253,203,1,1
74,205,191,2,253,203,1,110,201,2
05,60,32,205,250,32,205,241,43,2
01,22,2,6,19,1,32,71,82,65,70,73
,75,85,83,32,32,77,79,78,73,84,7
9,82,22,4,5,80,85,83,72,73,32,83
,79,70,84,87,65,82,69,32
```

```
60 DATA 49,57,56,55,32,127,22,8,
5,19,0,49,32,45,32,170,32,40,81,
44,65,44,79,44,80,44,76,44,86,41
,22,10,5,50,32,45,32,75,65,82,65
,75,84,69,82,32,32,32,40,81,44,6
5,44,79,44,80,44,75,41,22,12,5,5
1,32,45,32,83,80,82,73,84,69,32,
40,81,44,65,44,79,44,80,44,78,44
,77,44,70,41,22,14,5,48,32,45,32
,66,65,83,73,67,77,69,77,79,82,7
3,65,32,62,32,-1
```

```
100 LET M=30000
105 READ A: IF A<0 THEN STOP
110 POKE M,A: LET M=M+1
115 GO TO 105
```

## Kaland egy képzeletbeli országban A gyalogos

Könnyű dicsérni ezt a képes-szöveges programot, hiszen elsőrangú fantázia terméke. Az ármánykodó játéktól az egyéni elemzésig a GYALOGOS (The pawn) viszi el a pálmát. Azonnal feltűnik nagyfokú kimunkáltsága, amelyet a legtöbb hasonló játékban egyáltalán nem vagy csak alacsony színvonalon találhatunk meg.

A programban erőteljes az elemzőmunka. Leírható például: „Szerezz meg mindent, kivéve a hegedűt, azután öld meg az eget evő embert! Tépd le az egér farkát, és kösd össze vele a rudat meg a lasszót!” Természetesen áldozni kell egy cseppnyi időt arra, hogy begépeljük a NORTH vagy a GET KEY kulcsszavakat, de mindez megéri, mert sok olyasmit kínál a program, ami a többiben egyáltalán nincs meg.

A GYALOGOS-nak kiváló a grafikája — szöveges játékoknál eddig nem láttam hasonlót. A képek akár egy funkcióbillentyű megnyomásával, akár a CAMEO funkció alkalmazásával mozgathatók és feloszthatók.

Amikor először lépünk a játékban egy helyre, a képernyőn a teljes terület látható. A következő ott járaskor már a CAMEO vezérel vagy az illusztráció kicsinyített változata, ami a kép jobb felső sarkában jelenik meg. Ha CAMEO módban játszunk, valahányszor új területre érünk, mindig látjuk a kisebb kivágást. A „súgások” gyorsabban érzékelhetők, és ezek fel is pörgetik a játékot.

A GYALOGOS programnál egy meghatározott karaktersort gépelünk be, hogy erre választ kapjunk. Bizonyos feleleteket azonban csak akkor kaphatunk meg, ha azokhoz már elegendő pontunk gyűlt össze. Valamennyi kérdésnek 2-8-ig terjedő kódja van, mindegyikkel egyre részletesebb, pontosabb válaszhoz juthatunk.

A szöveges segédlet a GYALOGOS másik igen nagy erőssége. Az alfanumerikus bemenő jelek sorozata segít átjutni a játék nehezebb részein. Ez nem olyan „puskázó” rész, amelyet néhány hasonló játéknál megtalálhatunk.

Két másik segítség az önmásoló program és egy második lemez, ami a kimentett játéklások tárolására szolgál. Valószínűleg a játék csúcsa a KEROVINA MESÉJE című regény bájos története, amelyben a mesebeli király a háttérből segíti a GYALOGOS-t. Ha ezt elolvassuk, értékes információkat kaphatunk arról, hogy mi történik a játékban.

A RUN nyomán írta: Németh Attila





# Extra nyomtatások

Bizonyára sokaknak van a Commodore számítógépükhöz olyan nyomtatójuk, amely sokkal tágabb lehetőségeket nyújt, mint az MPS-803-as, de ezek kihasználhatatlanok, mert a programok nagy része nem tudja kezelni ezeket az extrákat.

Nos, körülményesen ugyan, de az Easy Script szövegszerkesztővel is válóra válthatók a kiaknázatlan lehetőségek. Hogy mit hogyan, arra a táblázat ad magyarázatot, és az eredményt a 2. ábra mutatja.

Első lépésként azonban az 1. ábrán látható prgramsort gépeljük be szövegünk elejére. Ezzel definiálhatunk a nyomtatóra kimenő egyéb olyan karaktereket is, amelyek eddig az Easy nem ismert. Ezek vezérlőkarakterek lesznek, amelyekkel a nyomtató írási paramétereit módosíthatók. Feladatunk csupán az, hogy a kívánt helyre ilyen jeleket tegyünk.

Egyetlen dolog, amire vigyáznunk kell, hogy a szövegszerkesztő ezeket is megjelenítendő betűként értelmezi, a nyomtaton azonban ez vezérlőkód, így a sorkiigazításnál ez problémát okozhat. Célszerű ezeket a sorokat manuálisan beszerkeszteni, elkerülve ezzel a csúnya margóbelógást.

Természetesen nem csupán az itt használt paraméterekkel lehet dolgozni. Ha a sortávolságot például nem 42-re, hanem 50-re szeretnénk beállítani, akkor az első sorban a 3=42 helyett 3=50-et kell írunk (a sortávolság paramétere n/216 collban van megadva).

Még egyszer megjegyez-

dő azonban, hogy ezek a lehetőségek csak a nagyobb kapacitású, általában az NLQ-t ismerő nyomtatókra érvényesek, mint például a

Seikosha SP1200VC-re, a Seikosha SP180VC-re, a Citizen 120D-re, az MPS 1000-re, az MPS 1200-ra stb.

Bárfai Barnabás

f1 2 4	☐4	dőlt betű
f1 2 5	☐5	álló betű
f1 2 - 1	☐-1	aláhúzás be
f1 2 - 0	☐-0	aláhúzás ki
f1 2 s 0	☐s0	felmás index
f1 2 s 1	☐s1	alás index
f1 2 t	☐t	nincs index
f1 2 e	☐e	vissz. duplázás be
f1 2 f	☐f	vissz. duplázás ki
f1 2 g	☐g	függ. duplázás be
f1 2 h	☐h	függ. duplázás ki
f1 2 f1 4 f1 1	☐f14f11	írógép betűminőség
f1 2 f1 4 f1 0	☐f14f10	normál betűminőség
f1 2 m	☐m	sűrített betű
f1 2 p	☐p	normál betű
f3 s a 1 3	☐sa13	condensed
f3 s a 1 0	☐sa10	normál
f1 2 3 f1 3	☐3f13	sorsűrűség

1. ábra

☐0=0: 1=1: 2=27: 3=42: 4=120: M

2. ábra

Ez a szöveg vízszintesen, függőlegesen, és mindkét módon vastagított. Lehetőség van ~~☐f14f11~~ és ~~☐s1~~ index készítésére is. Természetesen a dőlt betűs és aláhúzott nyomtatás is megoldható.

A sortávolság állítása szintén lehetséges. Lehetőség van továbbá mindezek keverésére, valamint az írógép betűminőség menet közbeni váltására, de a különböző írásszélesség beállítása sem jelent gondot.

		TOP LISTA										
játék programok		IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE	felhasználói programok	
1.	Rocket R.				*						1.	Power disk
2.	UGHlympics				*						2.	Giga Paint
3.	Zak McKrack	*			*						3.	Amica Paint
4.	Ocean R.				*						4.	Rockmon 5.3
5.	Skyfox II.				*						5.	Pagefox
6.	Last Ninja 2				*						6.	Printmaster
7.	Rizikó Triskit				*						7.	News room
8.	Pentis	*			*						8.	Windows
9.	Slap flight				*						9.	Game Maker
10.	Test Drive 2	*			*						10.	Cheese

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

Címünk:  
Mikroszámítógép Magazin  
Szerkesztősége  
1371 Budapest, Pf. 433  
A diákszerkesztőség

0900800007

## A számítógépek motorja IV.

### Processzorkezelés

A központi egység — a talán legfontosabb hardver erőforrás — idejének a folyamatok közötti elosztása a multiprogramozott operációs rendszerek alapja. Mint az erőforrások ütemezésénél általában, a cél most is a számítógéprendszer termelékenységének fokozása.

#### ALAPFOGALMAK, AZ ÜTEMEZÉS ESZKÖZEI

A multiprogramozástól azt várjuk, hogy javítsa a központi egység kihasználtságát és növelje a rendszer átbocsátóképességét (az adott idő alatt végrehajtható feladatok mennyiségét). A központi egység kihasználtsága azért nő, mert a B/K műveletek alatt újabb folyamatokat lehet aktivizálni. Az átbocsátóképesség növekedését egy egyszerű példával szemléltetjük.

Az A és B feladatok egyformán viselkednek: mindegyik fut egy másodpercig, majd B/K műveletet végez egy másodpercig, és ez ismétlődik százszor.

Ha a két feladatot egymás után futtatjuk, 400 másodpercre van szükség. A központi egység pedig csak az idő felében dolgozik.

Ha az A feladat várakozási idejében a központi egység a B feladatot hajtáná végre, a B várakozási idejében pedig újra az A-t, a teljes átfutási idő megfelelődné, a központi egység kihasználtsága pedig megduplázódna, azaz 100 százalék lenne. Figyeljük meg, hogy az A feladat ugyanakkorra fejeződik be, mint az előző esetben; a javulás abból ered, hogy ezalatt B is lefut.

A gyakorlatban természetesen ilyen jól illeszkedő feladatok nincsenek, de a multiprogramozás akkor is sokat segíthet, ha csak közelíti a fenti példát.

A központi egység (CPU) ütemezése azon a felismerésen alapul, hogy minden folyamat két műveletnek, a CPU-használatnak és a B/K használatnak a ciklikus ismétléséből áll. A ciklus végén központiegység-használat van. Statisztikai mérések azt mutatták, hogy — bár a központiegység-használati szakaszok erősen függenek a folyamatoktól és a hardvertől — általában minden folyamat során túlnyomó többségben vannak az igen rövid CPU-használati szakaszok és csak igen kevés, a központi egységet használó hosszú szakasz fordul elő. Az úgynevezett CPU-orientált programokra jellemző néhány igen hosszú ilyen szakasz.

A megszakíthatóság és ütemezhetőség érdekében minden folyamathoz többféle információt kell tárolni. Az információ tárolási helyét folyamatvezérlő blokknak (angolul: Process Control Block=PCB) nevezik, és szerkezete a következő:

— a folyamat állapota: új, kész, futó, váró vagy befejezett,

— az utasításszámláló tartalma a folyamat következőként végrehajtandó utasításának címe,

— a CPU regisztereinek tartalma,

— különféle tárkezelési információk: bázisok és limitek, laptáblák stb.,

— mindenféle elszámolási információ: CPU-idő, valósidő, időlimit, számlaszám, feladat- vagy folyamatazonosító stb.,

— B/K állapotinformáció: ki nem elégített B/K igények, a folyamatokhoz rendelt B/K egységek, nyitott fájlok stb.,

— CPU-ütemezési információ: prioritás, ütemezési sorban elfoglalt pozíció, egyéb paraméterek.

A PCB-eket az operációs rendszer területén kell tárolni vagy statikusan (a maximális folyamatszám rögzítése után) vagy dinamikusan, rendszerint lista formájában.

A PCB-vel a központi egység átkapcsolható egyik folyamatról a másikra és vissza. A cél, hogy valamelyik folyamat állandóan futó állapotban legyen. Egyprocesszoros rendszernél természetesen minden időpontban csak egy folyamat lehet futó, a többiek — kész állapotban — várnak, „készletli” sort alkotnak. A készletli sort — a B/K egységekre váró folyamatok soraihoz hasonlóan — láncolt listákkal, az ábra szerint szokták megvalósítani.

A készletli sor feldolgozása pedig — mint látni fogjuk — az egyes ütemezési algoritmusoktól függően változhat.

#### ÜTEMEZÉSI ALGORITMUSOK

A következő feladatnak a készletli sorból való kiválasztási algoritmus nagymértékben befolyásolhatja az egész rendszer hatékonyságát. A feladat súlyának érzékeltesére összefoglaljuk, hogy az ütemező algoritmustól milyen szempontok szerinti optimalizálást várnak el:

— CPU-kihasználtság, ami a gyakorlatban 40 és 95 százalék között szokott lenni;

— a rendszer átbocsátóképessége, amit többek között az óránként befejezett feladatok számával lehet mérni;

— fordulási idő, ami egy feladat leadása és befejezése között telik el. Ez különösen a felhasználó szempontjából fontos, de nagymértékben függ a B/K sebességétől;

— várakozási idő, ami a fordulási idő egyik komponense, amit a feladat a készletli sorban tölt. Az ütemező algoritmusok a fordulási időt tulajdonképpen csak a várakozási időn keresztül befolyásolják;

— válaszidő, ami különösen interaktív rendszereknél fontos, ahol a fordulási idő nem jó kritérium. Megtörténhet, hogy ha egy folyamat az eredmények egy részét már előállította, a többi eredmény meghatározását át lehet lapolni a meglévők közlésével. Így a felhasználót tulajdonképpen csak az első eredmény megjelenéséig eltelt idő — a válaszidő — érdekli.

Egy-egy algoritmus persze a fentiek közül egyszerre csak egyet vagy néhányat optimalizálhat. Jó esetben a többire sem ad átlagosnál rosszabb értéket. Egy interaktív rendszerrel például inkább a válaszidő szórását, mint az átlagát célszerű minimalizálni, mert ekkor a rendszer viselkedése jobban tervezhető.

#### Előbb jött — előbb fut

A legegyszerűbb ütemezés, melyet angol rövidítéssel FCFS-nek (First-Come-First-Served) is neveznek. A központi egységet mindig az először igénylő folyamat kapja meg. Megvalósítása csupán egy FIFO sort igényel, amelynek kezelése igen gyors lehet.

Az algoritmus hatékonysága ugyanakkor meglehetősen rossz. A fordulási idő például erősen függ a feladatok megjelenésének sorrendjétől. Tekintsünk egy három feladatból álló rendszert:

Feladat	CPU-idő
1	30
2	5
3	5

Ebben a sorrendben a három feladat fordulási ideje rendre 30, 35 és 40, tehát átlagosan 35 egység. Ha a feladatok sorrendje 2, 3, 1 lett volna, az átlagos fordulási idő  $(5+10+40)/3=18,3$ , alig több, mint az előző fele.

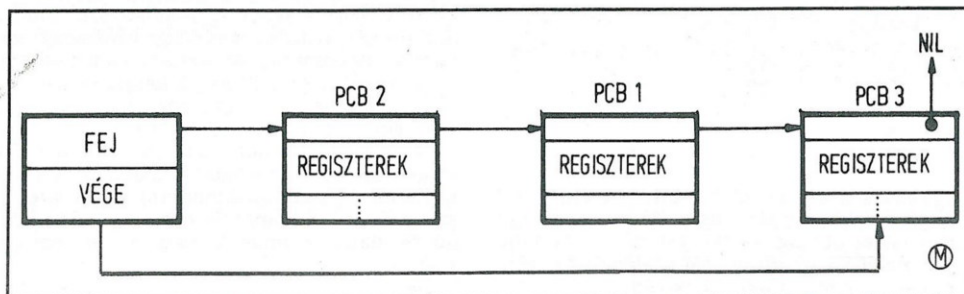
Az FCFS másik gyenge pontja az úgynevezett csordahatás. Ha a feladatok között egy CPU-orientált van, ez kiszajátja a központi egységet, a többi B/K orientált feladat pedig egy idő múlva mind a készletli sorban vár, és a perifériák kihasználatlanok. Majd a B/K orientált feladatok végül megkapják a központi egységet és gyorsan visszatérnek a B/K sorokba, amikor is a CPU lesz tétlen. A rendszer működését tehát az egyetlen nagy, CPU-orientált feladat el tudja rontani. Úgy tűnik, érdemes a rövidebb feladatokat előre venni.

#### Legrövidebb előnyben

Az algoritmus (angol rövidítése SJF-Shortest-Job-First) lényege, hogy amikor a központi egység felszabadul, azt ahhoz a feladathoz rendeljük, amelynek következő CPU-igénye a legkisebb. Ha több ilyen lenne, az FCFS algoritmus dönt.

Az előbbi háromfeladatos példán is tulajdonképpen az SJF algoritmus eredményét vetettük össze az FCFS eredményével és láttuk, mennyivel jobb fordulási időt adott. Ezen túlmenően kimutatható, hogy a „legrövidebb előnyben” elv optimális az átlagos fordulási idő szempontjából, mert ha egy rövidebb feladatot egy hosszabb elé viszünk a sorban, akkor többel csökken a rövid feladat várakozási ideje, mint amennyivel nő a hosszúé, tehát az átlag csökken.

Az SJF igazi problémája, hogy a készletli sorban minden feladat mellett tárolni kell a következő CPU-igény hosszát is. A kötegetelt rendszereknél a felhasználóktól kapott időkorlátok használhatók, a programozók érdekeltté válnak



a program futási idejének jó becslésében, mert alacsonyabb időkorláttal gyorsabban jutnak az eredményekhez. Ha viszont túl optimistán becsülnék, a rendszer a feladatot időtűlés miatt visszautasítja. Ezért az SJF kötegelt rendszereknél gyakori.

Mivel azonban az algoritmus különben egyszerű és optimális, nem szívesen mondanak le róla ott sem, ahol nincs igazi magas szintű ütemezés, mint például az időosztásos rendszereknél. Ilyenkor a matematikai statisztikával becslést lehet adni a következő CPU-igény hosszára, ha ismerjük az előző igényt és az erre adott becslést. A becslésre az exponenciális átlagot szokták használni, ami egy alkalmasan választott  $w < 1$  esetén a következő:

$$b_{n+1} = w \cdot t_n + (1-w) \cdot b_n$$

ahol  $t_n$  az utolsó CPU-igény hossza,  $b_n$  az utolsó,  $b_{n+1}$  pedig a következő igény becslése. A  $w$  súly tulajdonképpen a jelen és a múlt hatásának figyelembevételét szabályozza.

Az egyszerűség kedvéért  $w$ -t általában  $1/2$ -nek választják, ami azt fejezi ki, hogy a jövő szempontjából a múlt és a jelen egyformán fontos.

## Prioritás

A prioritásos ütemezés az SJF algoritmus általánosítása. A készenléti sorban minden feladathoz egy egész számmal kifejezett, a feladat fontosságát jelző prioritást is tárolunk. A CPU-t mindig a legmagasabb prioritású feladat kapja, ha pedig több ilyen van, akkor az FCFS elv dönt.

Az SJF-nél a prioritás speciálisan a következő CPU-igény reciprok értéke volt. Megegyezik még abban sincs, hogy a nagyobb számok jelentik a magasabb prioritást — mint nálunk —, vagy fordítva. A prioritás megadható belülről vagy kívülről. A belső prioritást a feladat különböző paramétereiből (SJF vagy időhatárok, fájlok száma, B/K igényesség, tárigény stb.) határozza meg a rendszer. A külső prioritást viszont a rendszeren kívül álló szempontok alapján adják meg: ilyen lehet a felhasználó privilégiuma, a magasabb díjtétel, a feladat objektív fontossága, üzletpolitika stb.

Minden prioritásos ütemezés közös problémája az úgynevezett kiéheztetés, ami azt jelenti, hogy egy kész állapotban lévő folyamat várakozási ideje nem korlátos. A folyamatosan érkező, nagyobb prioritású folyamatok egy alacsony prioritású munkát tetszőlegesen hosszú várakozásra kényszeríthetnek.

A probléma megoldása nem bonyolult. Be lehet vezetni mondjuk az „öregedés” fogalmát, a készenléti sorban eltöltött idő alapján. Ha minden öt perc várakozás után a feladatok prioritását eggyel növeljük, a legalacsonyabb prioritású feladat is belátható időn belül sorra kerül.

A prioritásos algoritmusok egy további finomítása lehet a kizárás. Ez azt jelenti, hogy ha a várakozási sorba az éppen aktív folyamatnál nagyobb prioritású folyamat érkezik, az kiszorítja az aktív folyamatot és megkapja a CPU-t. Az időosztásos rendszerekben, ha prioritásos ütemezést használnak, kizárásra feltétlenül szükség van, mert csak így biztosítható, hogy többé-kevésbé szabályos időközönként minden felhasználó hozzájusson a központi egységhez. Könnyen belátható, hogy az SJF — szintén prioritásos — algoritmus kizárással együtt rövidebb átlagos fordulási időt ad, mint anélkül.

## Körbenjárás

Az időosztásos rendszerek számára tervezett körbenjárás (angol rövidítése RR = Round Robin) lényege, hogy definiálnak egy kis időtartamot, amit időszletnek szoktak hívni. Értéke általában 10 és 100 ezredmásodperc között van. A készenléti sort zárt körnek tekintik, az alacsony szintű ütemező körben jár, és minden

folyamat rendre megkapja a CPU-t az időszlet meg nem haladó időre.

Az RR algoritmust legegyszerűbben egy FIFO sorral valósítják meg. Az érkező folyamatok a sor végére kerülnek, az ütemező veszi az első folyamatot, beállítja a megszakítóórát az időszlet szerint, és a folyamat indulhat. Ezután a folyamat visszaadhatja a CPU-t önként (mert befejeződik vagy B/K műveletbe kezd), vagy az időszlet lejártával bekövetkező megszakítás hatására, amikor a visszakérül a sor végére.

Az RR algoritmus hatékonysága erősen függ az időszlet nagyságától. Ha kicsi az időszlet, processzorosztásról beszélünk: minden felhasználó úgy érzi, hogy saját processzora van, ami az igazi processzor sebességének  $1/N$ -szeresével működik.

Az RR szoftvermegoldásaival azonban több probléma is van. Az éppen aktív folyamat megszakításakor menteni kell a regisztereket, az új folyamat indításakor pedig be kell azokat tölteni. Ezt a két tevékenységet együttesen környezetváltásnak nevezzük; időtartama gépenként változik, de általában eléri az időszlet tíz százalékát. A környezetváltás teljes egészében kieső idő, és jelentősen befolyásolja az ütemező hatékonyságát, hiszen az időszlet tizedével számolva, a CPU idejének tíz százalékát kell környezetváltásra fordítani. Ezért az látszik célszerűnek, hogy az időszlet jóval nagyobb legyen, mint a környezetváltás ideje, mert akkor egy adott feladathalmazra nézve csökken a környezetváltások száma és esetleg az átlagos fordulási idő is. Az időszlet növelésével az RR algoritmus FCFS-sé degradálódik, ami nyilván nem kívánatos.

Általánosan bevált becslés szerint az időszletet úgy kell megválasztani, hogy a CPU-igények nagy része, mintegy 80 százaléká egy időszlet alatt kielégíthető legyen.

## ÉRTÉKELÉSI MÓDSZEREK

A megfelelő ütemezési algoritmus kiválasztása összetett feladat. Mint az előzőekből láttuk, a választék elég nagy, sokféle szempont alapján lehet dönteni, ezért célszerű tennivalóinkat rendszerezni.

Első dolgunk az algoritmussal szemben támasztott igények rögzítése. A fentiek alapján az igényeket három fogalom köré csoportosítják: a CPU kihasználtsága, a válaszido és a rendszer átbocsátóképessége. Egy algoritmus értékelése előtt el kell döntenünk ezek egymáshoz viszonyított fontosságát. Megfogalmazhatók például a következő igények:

— maximális CPU-kihasználás adott válaszido-korlát mellett,

— maximális átbocsátóképesség azzal a megszorítással, hogy az átlagos fordulási idő a feladatok futási idejével egyenes arányban álljon.

Ha a választási kritériumokat már tisztáztuk, a szóba jöhető algoritmusokról el lehet dönteni, hogy megfelelőek-e. Az értékelés analitikusan, szimulációval vagy gyakorlatilag hajtható végre.

Az analitikus értékelés legegyszerűbb formája a determinisztikus modellezés. Rögzítünk egy feladathalmazt, és minden algoritmust ennek segítségével értékelünk. Tegyük fel, hogy a következő öt feladatot kell végrehajtani, melyek egyszerre érkeztek a rendszerbe:

Feladat	CPU-igény
1	10
2	29
3	3
4	7
5	12

Vessük össze az *Előbb jött — előbb fut*, a *Legrövidebb előnyben* és a *Körbenjárás* algoritmust az átlagos várakozási idő szempontjából! Az FCFS algoritmusnál a feladatok a felsorolási sorrendben hajtódnak végre:

Feladat	Várakozás
1	0
2	10
3	39
4	42
5	49
	140

Az átlagos várakozási idő tehát  $140/5 = 28$ . Az egyszerű (nem kizárásos) SJF algoritmus szerint a feladatok sorrendje és a várakozási idők a következőképpen alakulnak:

Feladat	Várakozás
3	0
4	3
1	10
5	20
2	32
	65

Most az átlagos várakozási idő csak  $65/5 = 13$  egység.

Az RR algoritmusnál válasszuk az időszletet tíz egységnek. A hosszabb feladatok több időszletet igényelnek. A végrehajtás sorrendje és a várakozási idők az alábbiak:

Feladat	Várakozás
1	0
2	10
3	20
4	23
5	30
2	20
5	10
2	2
	115

Az átlagos várakozás itt  $115/5 = 23$  egység.

A determinisztikus modellezés szerint tehát az SJF algoritmus az átlagos várakozási idő szempontjából ebben az esetben messze a legjobb. Az értékelési módszer gyors és egyszerű, bár ha megbízható eredményt akarunk, nem elegendő csak egy modellt felállítani, hanem több esetből kell általános következtetésekre jutni. Kimutatható például, hogy a fenti környezetben — amikor az összes feladat és azok CPU-igénye induláskor rendelkezésre áll — az SJF algoritmus minimalizálja az átlagos várakozási időt.

A determinisztikus modellek közös hibája, hogy a valóságban állandóan változó a terhelés, amit nem lehet statikus mintákkal jól közelíteni. Vannak azonban az ütemezésben más, meghatározható paraméterek is, például a CPU és a B/K igények eloszlása. Ezek megfelelő mintavétel után különböző eljárásokkal kezelhetők a matematikai statisztikával. Egyik alapfeladat lehet egy adott hosszúságú CPU-igény valószínűségének meghatározása, ami általános esetben exponenciális eloszlású. Ha emellett még sikerül az érkező feladatok eloszlását is megbecsülni, akkor a legtöbb algoritmushoz meghatározható az átlagos átbocsátóképesség, a CPU kihasználtsága, az átlagos várakozási idő stb.

Maga a számítógéprendszer felfogható szolgáltatók hálózatoként is. Minden szolgáltatóhoz (CPU, perifériák stb.) egy várakozási sor tartozik. Ha ismerjük az érkezési és a kiszolgálási ütemet, kiszámítható a kihasználtság, az átlagos sorhossz és várakozási idő. Ezt a módszert sorhálózat-analízisnek nevezik.

Ha például a sor átlagos hossza  $L$ , az átlagos várakozási idő  $T$  és a feladatok átlagos érkezési üteme  $C$  (feladat/másodperc), akkor a rendszer egyensúlyi állapotához az érkező és a kilépő feladatok számának meg kell egyeznie, ezért:

$$L = C \cdot T$$

Ez az összefüggés az úgynevezett Little-formula, melynek nagy előnye, hogy független az ütemezési algoritmusoktól és a feladatok érkezésének eloszlásától, és vele bármely, benne szereplő két mennyiség ismeretében a harmadik meghatározható.

A sorbanállási modell elemzése a néha bonyolult matematikai műveletigénye miatt jelenleg csak néhány algoritmusra és eloszlásra szorítkozik. Az egyszerűsítés érdekében bevezetett közelítések viszont sok esetben a megbízhatóság rovására mennek.

Az ütemező algoritmusok kiértékelésében pontosabb eredmény várható a szimulációtól, ami a számítógéprendszer egy programozott modellje. Szoftver úton reprezentálják a fontosabb komponenseket, az órát, a rendszer állapotát, a B/K egységek munkáját, a feladatokat és az ütemezőt. A szimuláció folyamán az algoritmus teljesítményét kifejező statisztikák ké-

szülnek. A szimulációhoz szükséges adatokat rendszerint véletlenszám-generátorokkal állítják elő, amelyek a feladatokat, a CPU-igényeket, az érkezési és kilépési ütemet stb. előírt eloszlásoknak megfelelően produkálják.

Az eloszlásokat nemcsak matematikailag, hanem empirikusan is meg lehet adni. A legjobb szimulációs adatanyag a valós rendszerek működésének részletes naplózásából (például mágnesszalagon való rögzítéséből) nyerhető. A napló valóságon alapuló adataival vizsgálható algoritmusok tényleg reális alapon vethetők össze. A szimuláció igen megbízható lehet, de nagyon költséges is: jelentős szoftverfejlesztés és sok gépidőt igényel.

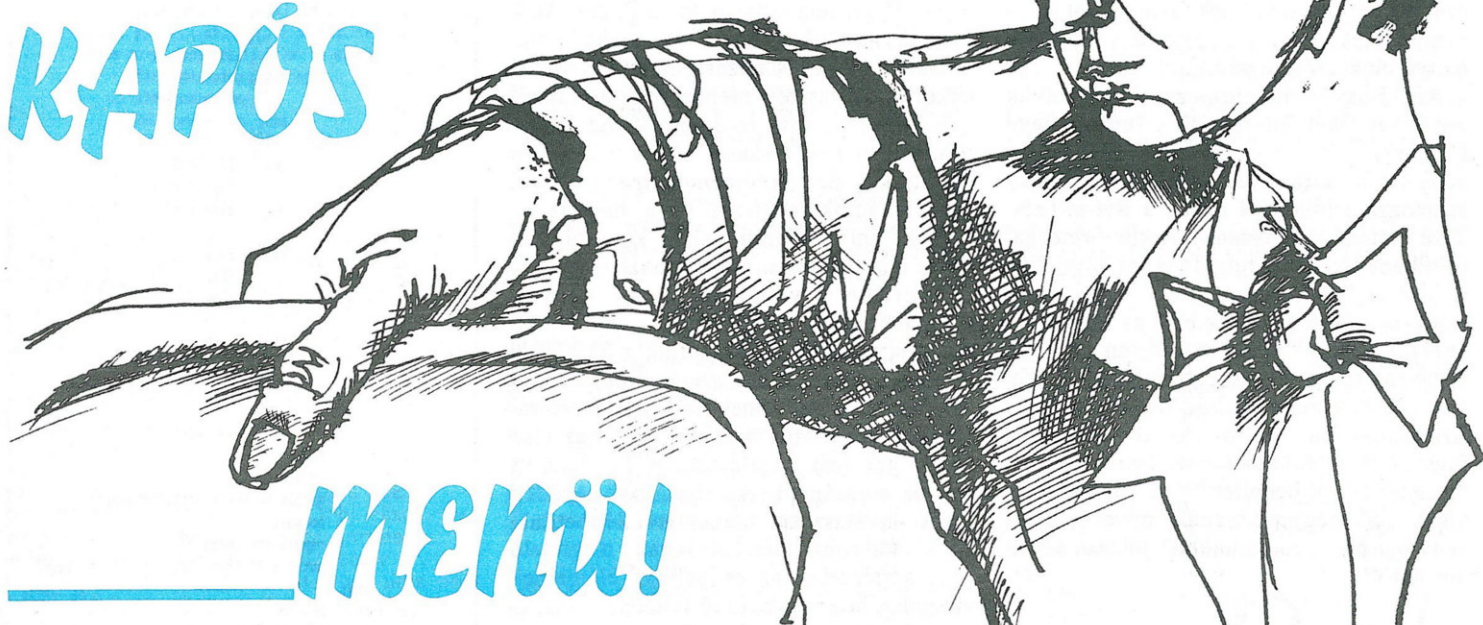
Teljesen megbízható eredmény tulajdonképpen csak a gyakorlattól várható. Meg kell írni az ütemező algoritmust megvalósító programot, és élesben kipróbálva végezni az értékelést. Az ilyen megközelítés persze igen költsé-

ges; az ára a programozási munkán kívül az, hogy a teljes számítógéprendszert ki kell vonni a folyamatos termelésből, hiszen a felhasználóktól nem várható el, hogy egy állandóan változó operációs rendszerrel dolgozzanak.

Ha mégis megoldjuk ezt a kérdést, a következő problémát a működési környezet állandó változása jelenti. A programozók nemcsak új programokat írnak, hanem alkalmazkodnak is az ütemező sajátosságaihoz. Ha például a rendszer automatikusan osztályozza az interaktív és egyéb feladatokat, mondjuk úgy, hogy ha egy folyamat nem ír vagy olvas a terminálról egy másodpercig, akkor az nem interaktív, akkor érdemes lesz a programokba betenni egy felesleges képernyőre írást, hogy a program a magas prioritású, interaktív csoportban maradjon.

Bakos Tamás

## EZ AZJÁN A KAPÓS



## MENÜ!

A korszerű, sikeres programok szerzői igyekeznek a felhasználók kedvében járni. Egy program használati értékét nagymértékben növeli, ha a külvilággal való kapcsolata egyszerű, egyszerű és könnyen követhető. Nem véletlen, hogy ma már az IBM PC felhasználók nagy része nem közvetlenül az MS-DOS-t kezeli, hanem közbeiktat valamilyen barátságosabb vezérlőprogramot, például a PathMindert vagy a Norton Commandert. Az MS-DOS és a vezérlőprogramok között az az alapvető eltérés, hogy az MS-DOS parancsokra hallgat, a PathMinder pedig úgynevezett **menükkel** irányítható. A programozásban a menü — az étlapokhoz hasonlóan — választékot kínál, persze nem egy étterem, hanem egy program szolgáltatásaiból. A felhasználónak nincs más dolga, mint közölni a választását.

A menüvezérlés egyebek között azért népszerűbb a parancsok használatánál, mert megkíméli a felhasználót a parancsok formai szabályainak megjegyzésétől vagy a dokumentáció folyamatos lapozgatásától.

A menüből kétféleképpen választhatunk: a sorszámozott menütétel bevitelével vagy a kiválasztott tételre mutatóval.

Az első megoldás a régi típusú képernyők idejéből maradt, amikor a kurzor csak egy-egy soron belül le lehetett mozgatni, és visszalépéskor a karakterek törlődtek. Ma már általánossá vált a rámutatással, például átszínezéssel, mozgó nyíllal, inverz kiírással való választás.

Az alábbiakban — kapcsolódva a Magazin Turbo Pascal sorozatához — bemutatunk egy általános menükezelő eljárást, amely sok programban használható. Az el-

járási neve **menu**, szövege az 1. listán látható, működése pedig a következő.

Híváskor a képernyőn — annak előzetes törlése nélkül — megjelennek a menütételek. A szöveget egy karakterlánc (sztring) tömb tartalmazza, a vízszintes irányú elhelyezkedés szabályozható, függőleges irányban a menü a képernyő közepére íródik. A menüben a *függőleges irányú* kurzorozgató, valamint a Home és az End billentyűvel lehet navigálni. Az éppen választható tétel inverz formában jelenik meg, a válasz-

táshoz az ENTER-t kell leütni, ami egyben az eljárásból való kilépést is kiváltja. Az eredmény egy egynél kisebb, egész értékű kód, a kiválasztott tétel sorszáma, A Home az első, az End az utolsó tételre viszi a kurzort, lépésként haladva, az utolsó tétel után újra az első, az első előtt pedig az utolsó következik.

Ennek megfelelően az eljárás paramétereit az alábbiak:

- **kod**: a kiválasztott menüsor sorszáma (1,2, ... 10),
- **n**: a menütételek száma (legfeljebb 10)
- **ko**: a menü sorainak kezdőpozíciója a képernyőn,
- **honnan**: a menüszoveget tartalmazó, **strtomb** típusú, tizelemű tömb neve,
- **marad**: logikai érték, amely a menüszoveg kilépés utáni törlését vezérli. TRUE esetén a menü a képernyőn marad, FALSE esetén törlődik.

Az eljárás lényeges része a kurzormozgató billentyűk kezelése. Mivel csak ezek, valamint az ENTER érdekes számunkra, az 1. lista elején szereplő **fun** felsorolt típus ezeknek a billentyűknek a neveiből és a **Semmi** értékből áll, amit az összes többi billentyű elnevezésére vezetünk be.

Azt, hogy a kurzormozgató billentyűk közül melyiket ütötték le, a **funkod** nevű függvény értéke mondja meg. Ez a függvény tízféle értéket adhat vissza: a 8 kurzormozgató billentyű nevét, a **Ret**-et (ENTER esetén) és a **Semmi**-t. A kurzormozgató billentyűk közös tulajdonsága, hogy leütéskor két kódot juttatnak a billentyűzet-pufferbe. Ezek közül az első az **Esc** kódja (#27), a második pedig az 1. listán a **funkod** függvény **case** utasításában látható valamelyik érték. A második kód felderítésére jól használható a Turbo Pascal **keypressed** függvénye. A **funkod**-ba az összes kurzormozgató kódot beépítettük (az eljárás ezek közül csak négyet használ), mivel a függvény így más programokban jobban alkalmazható.

Maga az eljárás törzse tulajdonképpen egy — az ENTER leütéséig futó — ciklus, amelynek magja a négy kurzormozgató billentyűnek megfelelő tevékenységeket tartalmazó **case** utasítás. A ciklus előtt a képernyő megfelelő helyére kiírjuk a menü szövegét és „ráállunk” az első tételre. A ráállást a tétel inverz formában való kiírása jelenti, amit két egyszerű eljárással (**inverz** és **normal**) vezérünk. Figyeljük meg, hogy a leütött billentyűt mindig olyan **repeat** ciklusban elemezzük, amely csak a számunkra érdekes öt érték valamelyikére ér véget. Ezzel elérhetjük, hogy a programot más billentyű leütése nem zavarja meg. Az eljárás végén, a megfelelő paraméter alapján a menüt letöröljük vagy meghagyjuk a képernyőn.

A **menu** eljárás alkalmas minden olyan menü kezelésére, ahol a felhasználó **pontosan egy** opciót választhat. Az eljárás itt közölt formájában a menü legfeljebb tíz opciót tartalmazhat, ez azonban a **strtomb** típusdefiníciójának módosításával egyszerűen változtatható.

Kézenfekvő az eljárás olyan általánosítása, hogy több opció kijelölését is lehetővé tegye. Ilyen feladattal például akkor találkozunk, ha több jellemző alapján kell valamit kiválasztani. Célszerű, hogy az eljárás ekkor a kiválasztott menüsorok sorszámainak halmazát adja eredményül. Az általánosított eljárást **menuset** néven a 2. listán láthatjuk. Az eredményhalmaz elemei egész számok, a listán ez a halmaz az 1...10 intervallumtípushoz kapcsolódik, de ez természetesen módosítható.

A **menuset** úgy működik, hogy a kurzormozgató gombokkal bejárva a menü sorait, a kiválasztott soroknál leütjük a **RETURN** billentyűt. A menüben szerepelnie kell egy speciális „kilépő” opciónak is. Ha ezt választjuk, az eljárás befejeződik, és az első paraméter (**es**) tartalmazza a kiválasztott opciók sorszámaikat. Az eljárásból egyetlen opció kiválasztása nélkül is kiléphetünk. A kiválasztott menüsoroknak megfelelő programrészeket az **es** halmaz elemeinek vizsgálatához kapcsolódó feltételes utasításokkal programozhatjuk. Például:

```
if e1 in es
then ...
else if e2 in es
then ...
else ...
formában.
```

A kilépő opció sorszáma (**ki**) az eljárás utolsó paramétere. Az egyszerűség kedvéért elhagytuk a menü kilépés utáni törlésre vonatkozó paramétert, a törlés mindig megtörténik.

A **menuset** ugyanazokat a segédeljárásokat használja, mint a **menu**, ezeket a 2. lista nem tartalmazza. Az általánosabb feladatok megfelelően az eljárás törzse valamivel bonyolultabb. A **menuset** eljárás jól mutatja a Pascal halmaz típusának alkalmazhatóságát.

B.T.

```
type tema = 1..10;
      tenak = set of tema;

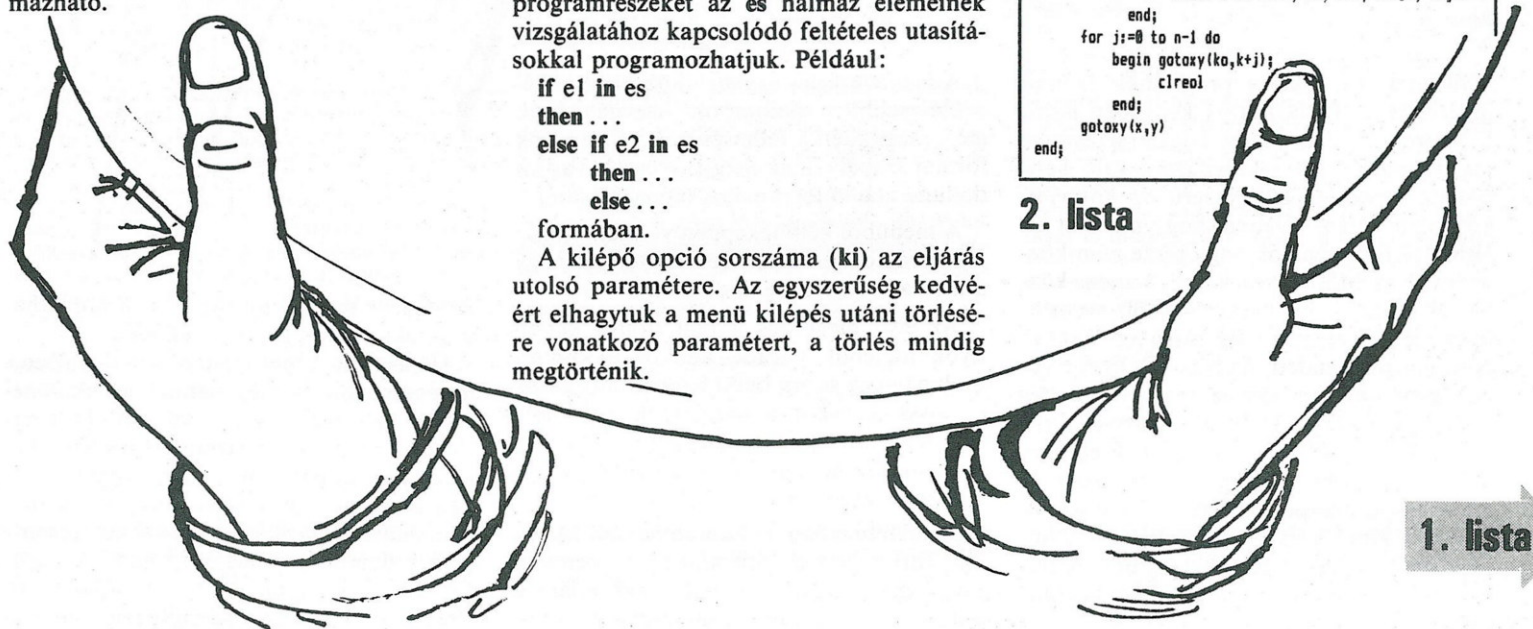
procedure menuset(var es: tenak; n, ko: integer;
                  honnan: strtomb; ki: integer);

      (Egy halmaz kiválasztása a menüből)
(es: Kiválasztott menüsorok halmaza)
(n: Menütételek száma)
(ko: Menüszoveg kezdő pozíciója a képernyőn)
(honnan: A menüszoveget tartalmazó strtomb neve)
(ki: A kilépés kódja)

(menuset törzse:)

begin ki:=(24-n) div 2; i:=0; x:=wherex; y:=wherey; es:=[];
for j:=0 to n-1 do
begin gotoxy(ko,k+j);
write(honnan[j+1])
end;
inverz;
gotoxy(ko,k+i); write(honnan[i+1]);
normal;
repeat c:=funktod
until c in [Ret, Le, Fel, Eleje, Vege];
while (i<>ki) or (c<>Ret) do
begin if (c<>Ret) and (not (i in es))
then begin gotoxy(ko,k+i);
write(honnan[i+1])
end;
case c of
Le: if i<n-1
then i:=i+1
else i:=0;
Fel: if i<1
then i:=n-1
else i:=i-1;
Eleje: i:=0;
Vege: i:=n-1;
Ret: es:=es+[i]
end;
inverz;
gotoxy(ko,k+i); write(honnan[i+1]);
normal;
repeat c:=funktod
until c in [Ret, Le, Fel, Eleje, Vege]
end;
for j:=0 to n-1 do
begin gotoxy(ko,k+j);
clrrow
end;
gotoxy(x,y)
end;
```

2. lista



1. lista

## Hogyan szövegeljünk?

# A szövegfeldolgozás elemei

```

type fun = (Eleje,Fel,PgUp,
            Bal,Jobb,
            Vege,Le,PgDn,
            Ret,Semmi);
str80 = string(80);
strtoab = array[1..10] of str80;
procedure menu(var kod: integer; n, ko: integer;
              honnan: strtoab; marad: boolean);
    (Általános menükezelés)
(kod: Kiválasztott menüsor indexe (1, 2,...)
(n: Menütelek száma)
(ko: Menüsöveg kezdő pozíciója a képernyőn)
(honnan: A menü szövegét tartalmazó strtoab neve)
(marad: A menü szöveg választás után marad (TRUE) vagy törlődik (FALSE))

    (A menü függőleges irányban a képernyő közepére kerül)

procedure inverz;
begin textbackground(white);
    textcolor(black)
end;

procedure normal;
begin textbackground(black);
    textcolor(white)
end;

function funkod: fun;

var c: char;
begin read(kbd,c);
    if c=#13
    then funkod:=Ret
    else if (c=#27) and keypressed
    then begin read(kbd,c);
        case c of
            #71: funkod:=Eleje;
            #72: funkod:=Fel;
            #73: funkod:=PgUp;
            #75: funkod:=Bal;
            #77: funkod:=Jobb;
            #79: funkod:=Vege;
            #80: funkod:=Le;
            #81: funkod:=PgDn;
        else funkod:=Semmi
        end
    end
    else funkod:=Semmi
end;

var i, j, k, x, y: integer;
    c: fun;

(aenu törzse:)

begin k:=(24-n) div 2; i:=0; x:=wherex; y:=wherey;
    for j:=0 to n-1 do
        begin gotoxy(ko,k+j);
            write(honnan[j+1])
        end;
        inverz;
        gotoxy(ko,k+i); write(honnan[i+1]);
        normal;
        repeat c:=funkod
        until c in [Ret, Le, Fel, Eleje, Vege];
        while c<>Ret do
            begin gotoxy(ko,k+i); write(honnan[i+1]);
                case c of
                    Le: if i<n-1
                        then i:=i+1
                        else i:=0;
                    Fel: if i<1
                        then i:=n-1
                        else i:=i-1;
                    Eleje: i:=0;
                    Vege: i:=n-1
                end;
                inverz;
                gotoxy(ko,k+i); write(honnan[i+1]);
                normal;
                repeat c:=funkod
                until c in [Ret, Le, Fel, Eleje, Vege]
            end;
            kod:=i+1;
            if not marad
            then for j:=0 to n-1 do
                begin gotoxy(ko,k+j);
                    clrscr
                end;
                gotoxy(x,y)
            end;
end;

```

A számítástechnika alkalmazása hosszú utat tett meg a hagyományos „számgyártástól” napjainkig. Ezen a fejlődésen belül külön említést érdemel az a mintegy tizenöt éves múltra visszatekintő terület, amelyet *szövegfeldolgozás*nak (angolul: word processing) neveznek. A szövegfeldolgozás eredeti célja az volt, hogy a számítástechnika eszköztárával támogassa írott anyagok, röviden *iratok* előállítását és további kezelését. Ma már a szövegfeldolgozás az általános irodaautomatizálás és a kiadói tevékenység alrendszerévé vált, és az alkalmazások igen jelentős részét — egyes becslések szerint 30-40 százalékát — képviseli.

Tekintettel arra, hogy a modern szövegfeldolgozó rendszerek IBM PC gépeken is jól használhatók, a már könnyen beszerezhető, nagyobb teljesítményű (például MicroVAX kompatibilis) eszközök pedig a teljes irodaautomatizálást is lehetővé teszik, megpróbálunk egy természetesen nem teljes, de reméljük, reprezentatív áttekintést adni a szövegfeldolgozás fogalmairól, eszköztáráról és módszereiről.

Az alapfeladat tehát iratok létrehozása és általános kezelése, ami igen sokféle műveletet jelenthet, a módosítástól a tároláson keresztül a helyesírás ellenőrzéséig, tartalomjegyzék- és indexkészítésig vagy a nyomtatásig.

A munka tárgya az *irat*, valamilyen meghatározott szerkezetű, általában szöveges információt tartalmazó állomány. Az iratok szerkezeti felosztása a legtöbb esetben a következő.

**Rész.** Ez a legnagyobb, közös tulajdonságokkal rendelkező egység, amelyet egy darabban akarunk nyomtatni. Gyakran a teljes irat egyetlen részből áll. Egy iratot akkor célszerű több részre osztani, ha az egyes részeknek önálló formát akarunk adni.

**Bekezdés.** Az iratnak az a szakasza, melyet az ENTER billentyű két leütése között írunk. Általában megtehetjük, hogy az eredeti bekezdésben maradvá kezdjük az új sort, de ilyenkor a sor végét valamilyen más billentyűvel kell jelezni.

**Sor.** A képernyő méretéhez, az irat formai jegyeihez igazodó szövegrész. A hagyományos gépirással ellentétben, a sor végét az irat nyers (szerkesztés előtti) változatában nem a szerző (a rendszer használója), hanem a szövegfeldolgozó rendszer jelöli ki úgy, hogy minden sor csak teljes szavakból álljon. A szerző a sorok hosszát adhatja meg, ami nemcsak rövidebb, hanem hosszabb is lehet, mint a képernyő szélessége. Ez utóbbi esetben a rendszer a sorok vízszintes mozgatójáról automatikusan gondoskodik.

**Szó.** A szöveg hagyományos, két szóköz közötti egysége. Egy szó nem lehet hosszabb, mint a megadott sorhossz. Sok rendszerben a szóköz mellett néhány speciális karakter (., / \* stb.) is a szó végét jelzi.

**Karakter.** A legkisebb, önálló szövegrész. Betűk, írásjelek és különböző speciális jelek — például rész vége, bekezdés vége — gyűjtőneve. A világnyelvek speciális betűi rendszerint rendelkezésre állnak, a magyar ékezetes betűket azonban külön kell előállítani. Ezt, a szövegfeldolgozásban létfontosságú, de

más alkalmazásoknál sem elhanyagolható problémát olyan programmal lehet megoldani, amely módosítja a gép belső karakterkészlete és a billentyűzet közötti megfeleltetést. Egyes szövegfeldolgozó rendszerekben a karaktereket nem a gép készletéből veszik, hanem grafikusán állítják elő, ami tetszőleges karakterkészlet definiálását teszi lehetővé, de ez csak nagyobb teljesítményű rendszereknél járható út. A szokásos, olvasható karaktereken kívül igen elterjedten használják a rendszer működését vezérlő karaktereket is. Ilyenek jelzik például a bekezdések vagy lapok végét, a szöveg valamilyen szempontból kitüntetett részeit.

Az iratok más szempontok szerint is darabolhatók: nyomtatáskor oldalakra, esetleg oszlopokra törölhetőek stb.

A megjelenítés eszköze — különösen a létrehozás és szerkesztés folyamán — a *képernyő*, amit ezért elsődleges fontosságúnak tekintünk. Más alkalmazásokkal ellentétben, a szövegfeldolgozásban a nálunk elterjedt képernyők közül az olcsóbb, monokrom típus jobban megfelel, mint a színes, mert élesebb képet ad és kevésbé veszi igénybe a szemet.

A szövegfeldolgozó rendszerek a képernyőt ablakként mozgatják, melyen keresztül egyszerre az iratnak csak egy része látható. Az ablaknak az irat fölötti mozgatójáról vezérlő parancsokkal, speciális billentyűkkel vagy egyéb eszközökkel gondoskodnak. Az írás során a képernyő aktuális pozícióját egy speciális jel, a *kurzor* mutatja. A kurzor legtöbbször egyetlen karakterre mutat, de bizonyos szerkesztési műveletek hatékony elvégzése érdekében „kiterjeszhető”, és ilyenkor egy vagy több karaktert, szót, sort, bekezdést vagy akár részt is lefedhet. A lefedett szövegdarabbal minden művelet elvégezhető, amit egy karakteren végre tudunk hajtani: törlés, átvitel, aláhúzás stb. A kiterjesztett kurzor tulajdonképpen a régebbi szerkesztők blokk-kijelölő funkciójának modern változata.

A képernyőhöz szorosan kapcsolódik a billentyűzet, amely magyar karakterkészlet használatakor általában eltér a szabványos-tól, ezért szokatlan a hagyományos gépiráshoz szokott felhasználóknak. Az ilyen billentyűzethez gyakran valamilyen speciális hardverelem is tartozik, és így a gép végső soron szövegfeldolgozó célrendszerrel válsat.

Mivel a képernyőn a szöveg könnyen manipulálható, kialakult a fogalmazásnak egy új, speciális módja, amikor is a szerző a szöveget közvetlenül a képernyőn komponálja, esetleg valamilyen előre összeállított vázlat alapján. Az iratok ilyen előállítására strukturált programozáshoz hasonló megoldás, amelyben a vázlat egyes pontjait többlépcsős finomítás során dolgozzuk ki. A módszer előnye, hogy a szerző folyamatosan látja az egész iratot, a szöveg már félkész állapotban is megvitatható, a társszerzők párhuzamos és összehangolt munkája egyszerűsödik.

Más fontos alkalmazási területekhez hasonlóan a szövegfeldolgozásban is kialakultak az „alpműveletek”, melyeket minden rendszer többé-kevésbé azonos módon tartalmaz, bár szabványosításról még korai lenne beszélni.

A következő részben áttekintjük a fontosabb szövegkezelő műveleteket.

## Programozási

## fogások és



## melléfogások

Legutóbb arra mutattam néhány példát, hogyan lehet a GOTO utasítás átgondolatlan használata révén megnövelni egy program terjedelmét a hatékonyság növelése nélkül. Most külföldi termésből mutatok be hasonlót, Heift *CAD* című könyvéből, mely magyarul 1987-ben jelent meg Data Becker — Novotrade kiadásban. A szaggatott vonalakat rajzoló szubrutinból (21—23. oldal) emeltem ki két részletet. A helykimélés érdekében nem idézem a teljes szubrutint, melyben — éppúgy, mint a könyvben közölt többi programban — hemzsegnek az 1/a és 2/a listákon láthatóhoz hasonló badarságok.

A könyv programjai *SIMON's BASIC* nyelven íródtak. Feltételezhető, hogy eredetileg minden utasítás külön sort foglalt el, ezzel is növelve a program és ezzel együtt a kiadó asztalára letett kézirat terjedelmét. A sorok összevonását a magyar változat készítőinek tulajdonítom, akik ezáltal megbontották a sorszámozás folytonosságát, megnehezítve a begépelésnél a *SIMON's BASIC* automatikus sorszámozási (AUTO) funkciójának használatát. Ráadásul a most tárgyalt szubrutinok az itteni listákon nem szereplő részébe egy sajnálatos gépelési hiba is becsúszott, bizonyítván, hogy a listázott programokat nem tesztelték előzőleg. (A könyv olvasói részére itt a megoldás: a lista 10136, 10157 és 10177 számú soraiban „GOTO 10186” helyett „GOTO 10185” irandó.)

Az 1/a listán látható részlet a szaggatott vonal első szakaszának végpontkoordinátáit számolja ki. Tekintsünk el az utasítások háttérétől, összpontosítsunk a megoldás módjára. Első pillantásra feltűnik, hogy a 10124-es sor a benne lévő GOTO-val együtt felesleges, ugyanis a GOTO nélkül is a 10125-ös sorral folytatódna a program. Ugyanígy szembevetendő a 10120-as sorban levő értékek feleslegessége, tehát ez a rész is elhagyható. Ha jobban belemélyedünk a lista elemzésébe, azt is észrevehetjük, hogy CG előjelét akkor kell ellenkezőre változtatni, ha CA kisebb CC-nél. Ugyanígy függ CH előjelváltása CB és CD viszonyától. Az IF utasításokban szereplő összetett feltétel csak felesleges bonyodalmakat okoz. Mindezek ismeretében készült el az 1/b listán látható egyszerűsített programrész, mely az eredetinel 10 sorral rövidebb.

A 2/a listán egy belső szubrutin látható, amely kapásból egyszerűsíthető a 2/b listán látható módon. Annak ismeretében, hogy CJ értéke csak 1 vagy -1 lehet, további egyszerűsítésre van mód az ELSE utasítás használatával. Most nem ezt az utat választjuk, hanem — ellentétben az előző esettel — megvizsgáljuk a CJ és CP változóknak a teljes szubrutinban betöltött szerepét. CP a vonalszakaszt megrajzoló LINE utasításnak az a paramétere, melynek értéke a rajzolás módját határozza meg: 0 esetén töröl, 1 esetén rajzol. Ez a változó csak a listán látható belső rutinban kap értéket, CJ aktuális értékének megfelelően. CJ-nek semmi más szerepe nincs, mint CP értékének meghatározása. Ha a program itt nem látható részein CJ = -1 helyett CP = 0 értékadó utasítást írunk, a GOSUB 10178 helyére pedig CP = 1 - CP kerül, a 2/a listán levő teljes programrész elhagyható, a program rövidebbé és gyorsabbá válik.

A következő példának csupán annyi köze van az előbbihez, hogy ez is műszaki tervezéssel kapcsolatos programból való. Itt a hibakezelésre látunk egy különleges megoldást. A *Technika* 1989. februári számában jelent meg az a C Plus/4-re írt program, melynek egy részlete a 3. listán látható. Hagyjuk itt is figyelmen kívül a programszöveg furcsa tagolását, összpontosítsunk a tartalomra. Az idézett programrész bonyolult számítások előzik meg, melyek — többek között — az A változó értékét is meghatározzák. Ha a 31-es sorra érve A-ban nulla van, a képernyőre kiíródik az „EZ BAJ” felirat, aztán mintha semmi baj nem lenne, tovább fut a program. Nem sokáig, mert a 34-es sor első utasításánál „DIVISION BY ZERO” hibaüzenettel befejezi futását. Az ilyesfajta hibakezelés láttán én sem tudok mást mondani, mint azt, hogy „Ez baj!”.

Barna László

```

...
10111 IF CA<CC AND CD<CB THEN 10115
10112 IF CA<CC AND CB<CD THEN 10117
10113 IF CC<CA AND CD<CB THEN 10120
10114 IF CC<CA AND CB<CD THEN 10123
10115 CG=-CG
10116 GOTO 10125
10117 CG=-CG:CH=-CH
10119 GOTO 10125
10120 CG= CG:CH= CH
10122 GOTO 10125
10123 CH=-CH
10124 GOTO 10125
10125 CK=CA+CG:CL=CB+CH
...

```

1/a lista

```

...
10111 IF CA<CC THEN CG=-CG
10114 IF CB<CD THEN CH=-CH
10125 CK=CA+CG:CL=CB+CH
...

```

1/b lista

```

...
10178 REM CIKLUS
10179 CJ=CJ*(-1)
10180 IF CJ= 1 THEN 10182
10181 IF CJ=-1 THEN 10184
10182 CP=1
10183 GOTO 10185
10184 CP=0
10185 RETURN
...

```

2/a lista

```

...
10178 REM CIKLUS
10179 CJ=-CJ
10180 IF CJ= 1 THEN CP=1
10181 IF CJ=-1 THEN CP=0
10185 RETURN
...

```

2/b lista

3. lista

```

...
31 IF A =0 THEN PRINT"EZ BAJ. "
32 A=A/2 :M1=M1/6:IFABS(M1)<HTHEN M1=0
33 M2=M2/6: :IFABS(M2)<HTHEN M2=0
34 X0=M2/A:IFABS(X0)<H THEN X0=0
...

```



## Gondolatok zeneszerkesztés közben

# A hang végül megszólal

Amatőr zeneszerkesztő programot kezdtem írni az átlagosnál jobb C64, de elég kevés Enterprise 128 ismeret birtokában. Az bátorított, hogy az utóbbi gépnek a C64-hez viszonyítva jóval egyszerűbb karakterdefiniálási és hanggenerálási lehetőségei vannak. Cikkem a munkámban eddig szerzett tapasztalataimról szól.

A program felépítése a karakterek átdefiniálásából, a funkciók gombok átprogramozásából, a megszólaltatandó kotta képernyőn való megszerkesztéséből, a képernyő kiolvasásából és a hangok megszólaltatásából áll. A teljes program 17 kb-át, ezért részletesen nem ismertethetem, de a jellemző részeket bemutatom.

### A KARAKTEREK ÁTDEFINIÁLÁSA

A kotta jeleit átdefiniált karakterekkel jelenítem meg. Például egy, a kotta egyik vonalán lévő 1/8 hang átdefiniált karakterekből az 1. lista szerint néz ki.

A DATA sorokban lévő 1-esek a kigyújtott, a 0-k pedig a sötét képpontok karakterekben. A CH változó azt mutatja meg a kardef függvénynek, hogy melyik ASCII kódú karakter alakját kell megváltoztatni. Természetesen a CH változó értékadási helyett paraméterátadást is alkalmazhattam volna.

#### 1. lista

```
713 let ch=144
715 data 00000110
717 data 00000111 <-- a hangj. zászlója
719 data 00000111
721 data 00000100
723 data 00000100
725 data 00000100 <-- a hangjegy szára
727 data 00000100
729 data 00000100
731 data 11111111 <-- kotta vonal
733 call kardef
521 let ch=136
523 data 00000100
525 data 00000100
527 data 00000100
529 data 00000100
531 data 00000100 <-- a szár folytatása
533 data 00000100
535 data 00000100
537 data 00000100
539 data 11111111 <-- kotta vonal
541 call kardef
329 let ch=128
331 data 00000100
333 data 00000100
335 data 00000100 <-- a szár folytatása
337 data 00000100
339 data 00000100
341 data 00000100
343 data 00011100
345 data 00111100 <-- a fej felső része
347 data 11111111 <-- kotta vonal
349 call kardef
353 let ch=129
355 data 01111100 <-- a fej alsó része
357 data 00111000
359 data 00000000
361 data 00000000
363 data 00000000
365 data 00000000
367 data 00000000
369 data 00000000
371 data 11111111 <-- kotta vonal
373 call kardef
```

### A FUNKCIÓS GOMBOK ÁTPROGRAMOZÁSA

A kotta jelei a megfelelő funkciók gombok használatával jelennek meg a képernyőn. Ennél az egyszerű problémánál torpantam meg először. A Felhasználói kézikönyv 44. oldalán szereplő állítás ugyanis egy kicsit általános, mert csak az 1–16 funkciók gombok programozhatók, ellenben 17–32-ig nem.

SET FKEY 11 CHR\$(204) programsor végrehajtása után például a Shift F3 gomb lenyomása a 204-es ASCII kódú karaktert adja.

### A KOTTA MEGSZERKESZTÉSE A KÉPERNYŐN

#### Néhány dolog a kurzorról

A szerkesztéshez kell egy kurzor. A kurzort az EXOS műszaki leírás 150. oldalán lévő utasítás Enterprise BASIC-re való értelmezése után így lehet bekapcsolni:

```
PRINT #102,CHR$(27);"O";
kikapcsolni azonban majd nem ezzel — mint ahogy a könyv írja —, hanem a
PRINT #102,CHR$(27);"o";
utasítással lehet.
```

Itt, a kurzor bekapcsolásánál jegyzendő meg, hogy a kurzor és a 142 ASCII kódú karakter formája megegyezik. A 142 kódú ka-

rakter alakjának változtatásával együtt változik a kurzoré is.

#### A kotta jeleinek „felépítése”

A kottajelek átdefiniált karaktereit tekintjük olyan építőköveknak, melyeknek különböző variációjú egymásra rakásával más és más „házat”, azaz kottajeleket, történetesen hangjegyet lehet „felépíteni”. Arról kell tehát gondoskodni, hogy ha a kezelő jelzi a gépnek a funkciók gombok valamelyikével a szándékát — adott esetben egy vonalon álló 1/8 hangot akar a képernyőn megjeleníteni —, akkor azok az „építőkövek” (karakterek) nyomtatódjanak egymás alá, amelyekből a kurzor aktuális helyén az 1/8 hang fog felépülni.

Az „építkezés” egyik részletét a 2. lista mutatja. A 2325–3440-ig terjedő ciklusban a program a kezelőtől billentyűk lenyomását várja. Ha a lenyomott billentyű ASCII kódja <201 (2475–2480), a képernyőn nem változik semmi. (A funkciók gombokat 201-től programoztam át.)

Ha a kód=204 (Shift F3), akkor a program hívja a LENT nevű függvényt, ahol az átadott paraméterek a megjelenítendő karakterek ASCII kódjai.

A LENT függvényben az Y és X változók tárolják a kurzor aktuális pozícióját. Láthatjuk tehát, hogy az aktuális pozícióban megjelenik

#### 2. lista

```
2325 do until inkey$="s"
.
.
2460 get x$
2465 if x$="" then 3435
2470 select ord(x$)
2475 case is<201
2480 goto 3435
.
.
2805 case 204
.
.
2845 call lent(129,128,136,144)
2847 let elt=-1:gosub 8000
.
.
3415 end select
.
.
3440 loop
.
.
3555 def lent(c1,c2,c3,c4)
3560 print #102,at y,x:chr$(c1)
3565 print #102,at y-1,x:chr$(c2)
3570 print #102,at y-2,x:chr$(c3)
3575 print #102,at y-3,x:chr$(c4)
3580 end def
```

#### 3. lista

```
35 ! exolv -> EXOS rutinnal olvassa a
képernyőt
40 code
exolv=hex$("3e,66,f7,05,32,db,12,ed,43,dc
,12,c9")
```

#### 4. lista

```
LD A,102 ;csatorna szám A-ba
RST 30,5 ;5-ös EXOS hívás
LD(4827),A ;STATUS az A-ból 4827
; -be
LD(4828),BC ;a kiolvasott karak-
; ter B-ből 4829-be
RET ;vissza BASIC-be
```

#### 5. lista

```
2265 let du1=240
.
.
3635 for x=1 to db-1
3640 call olvas(x)
3650 next x
4000 def olvas(x)
.
.
4010 print #102,at k(x),x-xkorr:;
4020 call usr(exolv,0)
.
.
4090 select peek(4829)
.
.
4205 case 128-128
4210 let du=du1/4:call
zaszlo(-2,144-128):call pit(73)
.
.
4300 end select
.
.
4315 end def
```

```

6080 def zaszló(y2,za)
6090 ! van-e zászló (1/8 hang-e) ?
6100 print #102,at k(x)+y2,x-xkorr;
6110 call usr(exolv,0)
.
.
6140 if peek(4829)=za then let du=du/8
6150 end def
    
```

## 6. lista

```

4320 def pit(base)
.
5520 let pt=base-4*(k(x)-ykorr)
5530 select pt
5532 ! case 37 to 41
5534 ! alsó c,d,e
5536 ! let pt=pt
5538 case 43 to 49
5540 ! alsó f,g,a,h
5542 let pt=pt-1
5544 case 51 to 55
5546 ! felső c,d,e
5548 let pt=pt-2
5550 case 57 to 63
5552 ! felső f,g,a,h
5554 let pt=pt-3
5556 end select
5570 sound pitch pt,left 255,right 255
,duration du
.
.
3620 end def
    
```

## 7. lista

a 129-es karakter, fölötté a 128-as, efölötté a 136-os és afölötté pedig a 144-es. (Érdemes összevetni a karakterek átdefiníálásával.) Ily módon a kívánt hangjegyet a program összerakta.

## A KÉPERNYŐ KIOLVASÁSA ÉS A HANGOK MEGSZÓLALTATÁSA

### A megszólaltatás elve

Egy hangnak — nagyon leegyszerűsítve és csak ennek a programnak a szempontjából vizsgálva — két alapvető tulajdonsága van: a frekvenciája (magassága) és az időtartama. A program a hang magasságát a hangjegy Y koordinátája, időtartamát pedig a kiolvasott karakter ASCII kódja alapján azonosítja.

### Képernyőolvasás

Eredeti tervem az volt, hogy az említett két információt a képernyő-memória olvasásából fogom nyerni. A képernyő-memóriában azonban túl hosszú időt vett igénybe az egyes hangjegyek BASIC programmal való megkeresése, ezért a hangjegyek koordinátáit a program egy tömbben tartja nyilván (ezt csinálja a 2847-es sor). Tehát nem kell a hangjegyeket a memóriából való kiolvasás előtt megkeresni. A kiolvasó függvény a tömbből nyert koordináta alapján a képernyő-memória adott helyére ugrik, és az ott talált értéket kiolvasa.

### Képernyőmemória-olvasó függvény

A képernyő-memória olvasása a C64-hez szokott programozó számára egy kicsit szokatlanul tűnhet. Egyik érdekessége az, hogy BASIC-ből csak nagyon komplikált úton, mégpedig a line paraméterblokkokon keresztül lehet olvasni. Egyszerűbb végrehajtani egy 5-ös EXOS funkcióhívást. A másik az, hogy 128 darabos karakterkészlet használatá-

nál az operációs rendszer az ASCII kód > 127 esetén a karakterkódból automatikusan kivon 128-at. Ez azt jelenti, hogy például a program kiír a képernyőre egy 130-as kódú karaktert, és ez a karakterkiolvasáskor már 2-es kódú lesz.

A 3. lista tartalmazza az EXOLV képernyő-memória-olvasó függvényt, amelyet olvashatóbb formában a 4. listán láthatunk.

### A hangok megszólaltatása

Az 5. listában a DU1 változó (2265) adja meg a megszólaló hang időtartamának kiinduló értékét 1/50 másodperc mértékegységben. A DB—1 kifejezés (3635) jelzi, hogy a hangjegyek koordinátáit tartalmazó tömb hány adatot tárol, hány hangjegyet kell az OLVAS függvénnyel kiolvasatni.

### A hang időtartama

A hang két tulajdonsága közül vegyük először az időtartamot. Mint már tudjuk, ezt az információt a karakter kódja hordozza.

Az 5. lista 4010-es sora a tömbből kiolvasott koordinátára állítja a kurzort, majd a 4020-as sor kiolvasa a karaktert. Ezután a kiolvasott karakter kódja szerint kell a tevékenységet folytatni (4090). Az EXOLV függvény a kiolvasott karakter kódját a 4829-es címre rakja le, így a BASIC onnan veszi át (4090).

Ha a kiolvasott karakter kódja 128—128=0, akkor a program azonosítja, hogy itt egy 1/4 hangról van szó, és ennek megfelelően a hang időtartamát osztja négygel (4210). Egyelőre csak 1/4 hang, mert 1/8 hangként csak akkor ismeri fel a program, ha megtalálta már a hangjegy szárán lévő „zászlót”; ez jelzi a kottában az 1/8 hangot.

A ZÁSZLÓ függvény a 6. listán látható. Hivási paraméterei (4210) azt jelzik, hogy a függvénynek az aktuális kurzorpozíciótól két-tővel feljebb kell keresni a 144—128=16-os kódú karaktert. Ez a „zászló” karakter.

Működése talán már ismerős. Kurzorpozícionálás (6100), karakterkiolvasás (6110). Ha a már előbb említett „zászló” karaktert megtalálta a függvény, akkor 1/8 hangról van szó, tehát a hang időtartamának kiinduló értékét nyolccal kell osztani (6140).

### A hang magassága

A ZÁSZLÓ függvény után térjünk vissza a főprogramra. A 4210-es sornál tartottunk. A hang időtartamának kiszámítása után a másik fő tulajdonságot, a hang magasságát is ki kell számítani. Ezt (is) végzi a PIT függvény, amit a 7. lista tartalmaz.

A PIT függvény úgy működik, hogy megvizsgálja a hangjegyeket a koordinátatömbben tárolt koordinátája és a paraméteradással kapott BASE változó közötti különbséget, majd ebből kiszámolja a PT hangmagasságot, mely a kiinduló érték lesz.

A program eredetileg nem tartalmazta az 5530—5556 sorokat, ami azzal a következménnyel járt, hogy nem tudta megkülönböztetni — a problémát egy zongorától kölcsönzött példán bemutatva — a fehér billentyűket a feketéktől. A „zenei” hatás siralmas volt. A hang végül is az 5570-es sorban szólal meg.

A program jelenlegi állapotában nem él igazán az Enterprise 128 által felkínált lehetőségekkel, a BASIC lassúsága miatt azonban — fordító hiányában — nem érdemes továbbfejleszteni. BASIC fordító birtokában majd a többszámolásokat és a videolapokat veszem célba.

Losonczy János

## Egy ismerős halmaz

A képernyő közepére helyezett keretbe rajzolja meg a program a két-, négy- vagy tizenhat színű Mandelbrot-halmazt. Kényelmesen vizsgálható vele akármekkora nagyításban is a halmaz bármely része. Az induláskor beállított nagyítást az <n> billentyű lenyomására tízszeresére növeli a program. A lenyomás pillanatában ábrázolt pontot a kép bal alsó sarkába teszi. A javasolt induló értékek tizenhat színű nyomásnál:

```

a = -1,4
b = -3
s = 4
    
```

Fehérvári Ferenc

```

100 PROGRAM "MANDEL"
110 SET STATUS OFF
120 TEXT
130 PRINT AT 2,2:"MANY SZINT HASZNALSZ?(2,4,16)
140 INPUT SZIN
150 IF SZIN<>2 OR SZIN<>4 OR SZIN<>16 THEN 130
160 PRINT AT 3,2:"KOMPLEX SIK(a,b) ES NAGYITAS(s)"
170 INPUT PROMPT"a=,b=,s=":A1,B1,S
180 GRAPHICS HIRES SZIN
190 SET PALETTE 0,RED,CYAN,YELLOW,MAGENTA,
RGB(.2,.1,.7),RGB(.6,.2,.2)
200 SET BIAS 43
210 LEP=SZIN/4+4
220 CLEAR SCREEN
230 SET LINE STYLE SZIN
240 PLOT 190,0;910,0;910,620;190,620;190,0
250 L=S/100
260 A=A1
270 PRINT AT 1,6:A1,B1
280 PRINT AT 2,16:"a=";INT(1/L);"s="
290 PRINT AT 3,16:"NAGYITAS=<n>"
300 PLOT 320,718,
310 PRINT #101:SZIN;"-SZIN"
320 FOR J=0 TO 700 STEP LEP
330 A=A+L
340 B=B1
350 FOR K=0 TO 600 STEP 4
360 B=B+L
370 X,Y,Q,W,N=0
380 DO WHILE N<32 AND Q+W<4
390 Q=X*X
400 W=Y*Y
410 XX=Q-W+A
420 Y=2*X*Y+B
430 X=XX
440 N=N+1
450 GET C$
460 IF LCASE$(C$)="n" THEN
470 A1=A : B1=B : S=S/10 : GOTO 220
480 END IF
490 LOOP
500 IF N=32 AND Q+W<=4 THEN
510 SET INK 1
520 PLOT 200+J,10+K
530 ELSE
540 SET INK MOD(N,SZIN)
550 PLOT 200+J,10+K
560 END IF
570 NEXT
580 NEXT
590 SET STATUS ON
600 END
    
```

**Fedezzük fel együtt!**

## Karakterek, karakterláncok

Amikor a billentyűzeten begépelünk egy programot vagy egy programnak adatokat adunk meg, karakterekkel — betűkkel, számjegyekkel és írásjelekkel — kódoljuk a számítógépbe jutó információkat, mint ahogy a gép is így írja vissza azokat. A képernyőn karaktereket, karakterláncokat látunk. Természetesen az, hogy milyen az információ a képernyőn, és az, hogyan dolgozhatunk tovább, a programtól függ, a program pedig a programozótól. Attól, hogy milyen ügyesen kezeli a ka-

rakterekre, karakterláncokra vonatkozó utasításokat, függvényeket.

Készítsünk először egy számítógépes órát (33. lista)! A PRINT DATE\$,TIME\$ parancs kiadásával megtudhatjuk, milyen formában kell megadni a dátumot és az időt definiáló karakterláncot a DATE és a TIME utasításokkal. Nézzük végig azokat a műveleteket, amelyekkel kialakítjuk a bemenő adatokból a 300 és 310-es sorokban szereplő karakterláncokat!

```

100 REM --- 33. program ---
110 REM --- ora ---
120 TEXT
130 STRING X$
140 INPUT AT 6,14,PROMPT "Ev : ";E$
150 INPUT AT 8,14,PROMPT "Ho : ";H$
160 INPUT AT 10,14,PROMPT "Nap : ";N$
170 INPUT AT 12,14,PROMPT "Ora : ";O$
180 INPUT AT 14,14,PROMPT "Perc : ";P$
190 INPUT AT 16,14,PROMPT "Masodperc : ";M$
200 REM --- potlas ---
210 CALL POT(H$,H$)
220 CALL POT(N$,N$)
230 CALL POT(O$,O$)
240 CALL POT(P$,P$)
250 CALL POT(M$,M$)
260 DEF POT(X$,REF X$)
270 LET X$="00"&X$:LET X$=X$(LEN(X$)-1)
280 END DEF
290 REM --- datum,ido beallitas ---
300 DATE E$&H$&N$
310 TIME O$&": "&P$&": "&M$
320 REM --- ido kijelzes ---
330 TEXT :LET R$=CHR$(161)
340 LET T$=TIME$
350 PRINT AT 6,10:"Ora : ";R$;VA
L(T$(1:2))
360 PRINT AT 8,10:"Perc : ";R$;VA
L(T$(4:5))
370 PRINT AT 10,10:"Masodperc : ";R$;VA
L(T$(7:))
380 GET Q$
390 IF Q$="" THEN 340
400 IF Q$=CHR$(13) THEN 480
410 REM --- datum kijelzes ---
420 TEXT :LET D$=DATE$
430 PRINT AT 6,16:"Ev : ";R$;VAL(D$(1:4))
440 PRINT AT 8,16:"Ho : ";R$;VAL(D$(5:6))
450 PRINT AT 10,16:"Nap : ";R$;VAL(D$(7:))
460 WAIT DELAY 5
470 GOTO 340
480 END
    
```

**33. lista**

```

100 REM --- 34. program ---
105 REM --- oroknaptar ---
110 NUMERIC A,B,C,T,L
120 DIM NAP$(7)
130 LET NAP$(1)="vasarnap":LET NAP$(2)="hetfo":LET NAP$(3)="kedd":LET NAP$(4)="szerda"
140 LET NAP$(5)="csutortok":LET NAP$(6)="pentek":LET NAP$(7)="szombat"
150 TEXT
160 INPUT AT 4,6,PROMPT "Ev(1980-2079) : ";E
170 INPUT AT 6,6,PROMPT "Ho : ";H
180 TEXT
190 PRINT AT 2,8:"Ev: ";E; " Ho: ";H
200 LET E$=STR$(E):LET H$="0"&STR$(H)
210 LET D$=E$&H$(LEN(H$)-1)
220 LET N=1
230 LET N$="0"&STR$(N):LET N$=N$(LEN(N$)-1)
240 DATE D$&N$
250 LET N$=" "&STR$(VAL(N$)):LET N$=N$(LEN(N$)-1)
260 CALL NAPOK(E,H,N,L)
270 PRINT AT MOD(N,16)+4+INT(N/16),4+17*INT(N/16):N$; " ";NAP$(L)
280 LET N=N+1
290 !
300 REM --- napok szama? ---
310 TIME "23:59:59"
320 WAIT DELAY 1
330 LET H0$=DATE$(5:6)
340 IF VAL(H0$)=H THEN 230
350 PRINT AT 20,1:
360 !
370 REM --- het napjai ---
380 DEF NAPOK(A,B,C,REF T)
390 IF B<3 THEN
400 LET B=B+12:LET A=A-1
410 END IF
420 LET T=A+B+C+INT(B*(B+1)/5)
430 LET T=T+INT(A/4)-INT(A/100)+INT(A/400)
440 LET T=MOD(T,7)+1
450 END DEF
    
```

**34. lista**

```

100 REM --- 35. program ---
110 REM --- fenyujsg ---
120 TEXT
130 LET S$=""
140 INPUT AT 3,4,PROMPT "Szoveg: ";X$
150 INPUT AT 5,4,PROMPT "Ablak hossza (max.40): ";N
160 TEXT
170 IF LEN(X$)<N THEN
180 LET X$=X$&S$(N-LEN(X$))
190 END IF
200 PRINT AT 10,20-INT(N/2):X$(N)
210 LET X$=X$(2:LEN(X$))&X$(1)
220 GET Q$
230 FOR T=1 TO 100
240 NEXT
250 IF Q$="" THEN 200
260 CLEAR SCREEN
    
```

**35. lista**

```

100 REM --- 36. program ---
110 TEXT :PRINT AT 12,8:"Nyomj le egy billentyut!"
120 WAIT DELAY 2
130 TEXT
140 LET Q$=INKEY$
150 IF Q$="" THEN 140
160 PRINT AT 10,16:"Karakter: ";CHR$(161)
170 PRINT AT 12,16:"Kod: " CHR$(161);ORD(Q$)
180 IF ORD(Q$)>13 THEN 140
    
```

**36. lista**

```

100 REM --- 37. program ---
110 TEXT :LET I=32
120 PRINT AT 22,6:"A botkormanyt le-fe mozgasd!"
130 GET Q$
140 IF Q$="" THEN 130
150 LET I=I-(ORD(Q$)=180 AND I<160)
160 LET I=I+(ORD(Q$)=176 AND I>32)
170 PRINT AT 12,8:CHR$(161); "Kod: " I; TAB(20); "Karakter : ";CHR$(I)
180 GOTO 130
    
```

**37. lista**

```

100 REM --- 38. program ---
110 REM --- versenyfutas ---
120 SET CHARACTER 146,0,8+4,8+4,8+4,8+4+2,16+8+4,32+16+8+4,128+64+32+8+4+2,0
130 TEXT :RANDOMIZE
140 DIM V(10)
150 FOR I=1 TO 10
160 PRINT AT 2*I,2:CHR$(145+I)
170 LET V(I)=2
180 NEXT
190 LET X=RND(10)+1
200 LET V(X)=V(X)+2
210 PRINT AT 2*X,V(X)-2: " ";CHR$(145+X)
220 IF V(X)<38 THEN 190
230 PRINT AT 21,6:"A gy";CHR$(147); "zt es: ";CHR$(X+145)
    
```

**38. lista**

A beolvasott adatok helyességének vizsgálatával most nem foglalkozunk. Ezt az olvasóra bizzuk. A POT függvényblokkban (260-as sor) az egyjegyű számok miatt teszünk a számok elé nullákat, az & művelettel. Összefűzzük a 00 és az X\$ karakterláncokat. Végül az  $X\$ = X\$ \text{ LEN } X\$ - 1$ : utasítással az X\$-ba az utolsó két számjegy kerül. A LEN függvény a karakterlánc hosszát adja meg, ezért az utolsó két karakter lesz az X\$ új értéke; egy szeletet kivettünk a karakterláncból.

A karakterláncok szeletelését legegyszerűbben a következő parancsokkal vizsgálhatjuk:

```
A$ = "ABCDEFGG"
```

```
PRINT A$(2:4),A$(3),A$(2:),A$(5)
```

Az R\$ karakterlánc-változóba egy vezérlőkaraktert teszünk, amelyiknek a kódszáma 161. Kiírásakor törlődik a sor, így nem marad „szemét” az új értékek megjelenésekor.

Az idő és dátum kijelzésekor a T\$ és D\$ szeleteit írjuk ki, de úgy, hogy a számjegyekből álló karakterláncokat átalakítjuk számokká a VAL függvénnyel. Erre azért van szükség, hogy az elől álló nullák ne jelenjenek meg a képernyőn.

A programból az ENTER billentyű lenyomásával léphetünk ki. Ez is egy vezérlőkarakter, aminek kódszáma 13. Ezt vizsgáljuk a 400-as sorban.

A Felhasználói kézikönyvben olvashatjuk, hogy a DATE\$ nemcsak a hónapok napjainak számát, de a szökőévet is figyelembe veszi. A programmal könnyen megnézhetjük, hogy ez igaz-e. A program érvényességi körének kiterjesztését (az 1980 előtti évekre is legyen igaz) az olvasóra bizzuk.

Naptárprogramunkban (34. lista) egy képletet használunk a hét napjainak meghatározására. Az algoritmust a NAPOK függvényblokkból kiolvashatjuk. A hónapok napjainak számát a DATES függvénnyel kapjuk meg (300–340-es sorok). Az időt beállítjuk éjjél előtt egy másodpercre, a WAIT DELAY utasítással várakozunk egy másodpercet, és megvizsgáljuk, hogy átléptünk-e a következő hónapba.

Az előző programból az is kiderül, hogy a kiírás formátumának kialakításához jól használhatók a függvények. A 270-es sorban a numerikus függvényekkel a helyet, a karakterlánc-függvényekkel és műveletekkel — láncolás, szeletelés — a kiírás formáját adjuk meg (a számjegyek elején álló nullákat hagyjuk el).

Kis helyen hosszabb szöveg megjelenítésére szolgál az úgynevezett ablak, ami ugyanúgy működik, mint a fényűjság (35. lista). Ennek alapötlete az, hogy a karakterlánc első n karakterét kiírjuk, ezután az első karaktert a karakterlánc végére tesszük. Amennyiben a karakterlánc rövidebb, mint az ablak, a szóköz karakterekkel kiegészítjük a karakterláncot.

A 36. listán látható programmal az ORD függvényt mutatjuk be, amivel a billentyűzettel és a botkormánnyal megadható karakterek kódját határozhatjuk meg. Gondoljunk azokra a karakterekre is, amelyeket két billentyű lenyomásával adhatunk meg (SHIFT, CTRL, ALT)!

A kódokhoz tartozó karaktereket a 37. lista programjával írathatjuk ki, a CHR\$ függvény alkalmazásával. A 150-es és a 170-es sorokkal IF utasításokat váltottunk ki. Ha a kifejezésben szereplő logikai kifejezés igaz, akkor annak értéke -1, egyébként 0.

A karakterdefiniálást egy játékprogrammal mutatjuk be (38. lista). A 146-os kódszámú karakter egy kengurut ábrázol, ami lehet persze, hogy nem a legtökéletesebb. A karakter kilenc pontsorból áll, minden sorban nyolc pont van. Ezt a nyolc pontot egy bináris számnak tekintjük. Amelyik pont világít, az az 1, a sötét a 0 számjegy. A SET CHARACTER utasításban a világító pontok helyiértékeinek összegeként adtuk meg a sorok decimális értékeit.

A program persze tökéletesíthető: indítsanak más ugráló állatot is, vagy tegyék élethűbbé az ugrálást!

Dusza Árpád

## Mi a manó?

### Elveszett program

Kövér Gábor, a miskolci Enterprise klub tagja a következőkre hívja fel a géptulajdonosok figyelmét: A SIMON vagy ASMON használatkor a RESET gomb megnyomása esetén az Editorban lévő forrásszöveg elveszni látszik ugyan, de nem törlődik ki. Igaz, az assembler változói átíródnak. A 0271h–72h címeiken tárolódik az Editor-puffer kezdőcíme, ami általában 0801h. Innen D paranccsal kilistázva a memória tartalmát, megállapíthatjuk az elveszettnek vélt szöveg végcímét. Ezután az utolsó bájt tartalmát be kell írni a 0273h, illetve a 0289h–8Ah címekre, ezenkívül egy ennél egygyel nagyobb értéket a 0279h–7Ah címre. Mivel az assembler a szövegünk első báját kinulázza, ennek eredeti érték-

két is vissza kell állítani. Így visszakaphatjuk kényszerűségből elvesztett assembly programunkat.

### Nyomtatás

Racskó Tamás, aki a szegedi Enterprise klub tagja, az RPR 210–01 típusú nyomtatónál szerzett tapasztalatát osztja meg olvasóinkkal: bár az EXOS-leírás szerint a gép a nyomtató betöltésénél a printer aljzaton '/ready' jelölést alkalmaz, azaz egy negatív szintű jelet, a nyomtató nem működik. A ROM listát tanulmányozva láthatjuk, hogy ezt a jelet az Enterprise az adat kiküldése után vizsgálja. Ezek szerint '/ack' (acknowledge) jelet vár a nyomtatótól, de az én gépem csak a 'busy' jel betöltése után működött helyesen.



Gaetsch Günterné rajza

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

## Hardver

rendelkeznie kell alapfokú áramkört 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 szoftverismere-tekre.

## A kódolás és hibái

A kódolás során a megtervezett programot a mikroprocesszor által végrehajtható utasításkódnak megfelelő, szimbolikusan megfogalmazott nyelven írjuk le. Ez processzortól függő feladat, ami az utasításkészlet részletes ismeretét és az adott utasításkészlettel való programozási gyakorlatot igényel. A program megírása pontosan azt jelenti, hogy megvalósítjuk azt az algoritmust, ami a bemeneti jelekből létrehozza a kimeneti jeleket. A program megírásakor már figyelemmel kell lenni a konkrét hardver kialakításra, nevezetesen arra, hogy milyen címen van a RAM/EPROM memória és melyek a be/kimenetek címei.

A Magazin sok cikke éppen ilyen kódolási kérdésekkel foglalkozik és különböző megoldásokat mutat be. A következőkben a Z80 mikroprocesszorra vonatkoztatva inkább azt vizsgáljuk meg, hogy milyen hibákat követhetünk el a kódolás során.

### Kódolási hibák

**Hibás vagy hiányzó inicializálás.** A program változóinak hibásan vagy egyáltalán nem adunk kezdeti értéket.

**A feltételes ugrások logikájának megfordítása.** Például JR C, utasítás felcserélése JR NC, utasítással. Felidézve a kivonás vagy az összehasonlítás hatását (A az akkumulátor, M egy regiszter vagy memóriahely tartalma):

Nulla (zero) jelzőbit = 1 ha  $A = M$  és 0 ha  $A < M$ .

Átvitel (carry) jelzőbit = 1 ha  $A < M$  és 0 ha  $A > M$ , így a JR C, azt jelen-

ti, hogy akkor lesz ugrás, ha  $A < M$ , a JR NC, pedig, hogy akkor, ha  $A > M$ . Ha azt szeretnénk, hogy az egyenlőség a másik esethez tartozzon, cseréljük meg A és M tartalmát, vagy adjunk 1-et M-hez. Ha mondjuk akkor akarunk ugrani, ha  $A > M$ :

CP 6

JR NC, ADDR

vagy ha  $A > 6$  esetén akarunk ugrani:

CP 7

JR NC, ADDR

**A számlálók vagy mutatók rossz helyen való változtatása.**

**A triviális eset hibás kezelése.** Például mi a helyzet akkor, ha egy puffertben nincs adat vagy hibás billentyűt ütünk le stb.

**Az operandusok sorrendjének megcserélése.** LD A,B a B tartalmát tölti az A-ba, az LD B,A utasítás az A tartalmát tölti B-be.

**A feltételjelző bitek megváltoztatása,** mielőtt felhasználnánk értéküket. A 8 bites INC és DEC utasítások a carry jelzőbit kivételével mindegyik jelzőbitre hatással vannak, a POP AF és EX AF,AF" utasítások minden jelzőbitet befolyásolnak, a logikai utasítások a carryt törlik.

**A címek és tartalmak összetévesztése.** LD HL,2000H utasítás HL-be 2000H-t tölt, az LD HL,(2000H) utasítás H-ba a 2000H című L-be a 2001H című memóriarekesz tartalmát tölti!

**A számok és karakterek összetévesztése.** ASCII 7 kód értéke 37H, a 07 kód a BELL karakter.

**Decimális és hexadecimális számok összetévesztése.** A BCD 37 értéke például 55, ha hexadecimális számként kezeljük.

**A kivonás sorrendjének összetévesztése.** A SUB és CP utasítások az  $A - M$  műveletet végzik és nem az  $M - A$ -t!

**Szubrutin hatásainak figyelembevétele.** Pontosán dokumentálni kell, hogy egy rutin milyen regisztereket használ és milyen jelzőbiteket ront.

**A shiftelő és rotáló utasítások nem megfelelő használata.** Ezeket célszerű mindig ellenőrizni.

**Egy tömb hosszának helytelen megállapítása.** Például az 50H-tól 54H-ig terjedő memóriaterület öt bajtot foglal el!

**A regiszterek és regiszterpárok összetévesztése.** A H és L regiszter együtt alkotja a HL regiszterpárt, és az INC HL utasítás a H tartalmát is megváltoztatja, ha L tartalma FFH.

**A 16 bites számok vagy címek két szomszédos memóriahelyet foglalnak el.** Emlékezzünk, hogy a Z80 minden kétbájtos mennyiséget alacsony/magas helyiérték formájában tárol. Például az LD (10H),HL utasítás végrehajtás után az L regiszter tartalmazza a 10H című rekesz, a H regiszter a 11H című rekesz tartalmát.

**A verem és a veremmutató összekeverése.** A DEC, INC és LD a veremmutató értékét és nem a verem tartalmát befolyásolhatja. A PUSH és POP utasításokkal lehet a verem tetejére írni, illetve onnan olvasni. A CALL, RET, RETI, RETN és RST utasítások szintén a vermet használják a programszámláló el- és visszamentésére. A megszakításra adott válasza is eltárolódik a veremben a régi programszámláló értéke. Az EX (SP),HL utasítás a veremmutatót nem változtatja meg.

**Elfelejtjük a veremmutatót beállítani.** Egy program indításakor érvényes RAM memóriacímeket kell a veremmutató kezdőértékének adni — és a verem lefelé „terjeszkedik”.

Természetesen ezek a hibák sokak számára szinte elkövethetetlenek

tűnnek, de mi van, ha mégis megtörténnek?

## Hibakeresés

A programok hibáinak felderítése és kiküszöbölése a programozók munkájának tekintélyes részét teszi ki. Az előbbieken ismertetett programtervezési módszerek áttekinthetőbbé teszik a programokat és csökkenthetik a hibák előfordulásának gyakoriságát, de a hibát nem zárják ki. A következőkben először az egyszerűbb hibakeresési eszközöket mutatjuk be.

A lépésenkénti programvégrehajtás (single-step) során a mikroprocesszor az utasításokat egyenként, egymás után hajtja végre, egy jelre mindig csak egyet. Két megoldás lehet: hardveres és szoftveres. Hardvermegoldásnál minden lépés után WAIT állapotba vezéreljük a processzort (a WAIT vonalára nulla szintet kapcsolunk), és ilyenkor a processzor cím-, adat- és vezérlővonalain az aktuális pillanatbeli állapot marad fenn. E vonalak vizsgálatával (leolvasásával) a programvégrehajtás minden lépése végigkövethető. Az ábrán egy ilyen megoldást mutatunk be.

A kapcsolásban a processzor jelei két 16 bemenetű multiplexeren keresztül olvashatók le. A CLEAR bemenetre kapcsolt jellel töröljük a multiplexer bemenetére kapcsolt U2 jelű számlálót. Ezután a multiplexer bemenetén megjelenő — a Z80-as tok lábain meglévő — jeleket sorban egymás után beolvassuk a BEOLVASÁS bemeneten keresztül. A 32. léptetőjel után beolvasódik az utolsó jel is, és csupán egy kis programra van szükség, amely az így beolvasott jeleket CÍM—ADAT—VEZÉRLŐJEL formájában összerakja. Összetettebb, bonyolultabb programmal akár a végrehajtott utasítások disassemblált formáját is megjeleníthetjük. Azt, hogy a programvégrehajtás valóban utasításciklusonként (lépésenként) haladjon, a kettős D tárolóból kialakított áramkör teszi lehetővé. Ennek részletes elemzésétől itt eltekintünk. A RUN/STEP jellel vezérelhetjük a folyamatos, illetve lépésenkénti programvégrehajtást.

A kapcsolat előnye, hogy mindössze négy — három bemenő és egy kimenő — jellel vezérelhető. A konkrét, megvalósított esetben egy Primo botkormány-csatlakozójára kapcsolódott ez az áramkör, amit a szaggatott vonallal bekeretezett rész jelöl.

Mechanikailag az áramkör egy meghosszabbított lábú, Z80-as processzort tartalmazó foglalattal dugható a vizsgálandó mikroszámítógép processzorának foglalatába, átvéve annak funkcióját, de biztosítva a lépésenkénti futtatást. Ez az áramkör nem használható, ha a vizsgált rendszer dinamikus memóriát tartalmaz, mert WAIT állapotban nem adja az RFSH frissítőjelet.

Szoftvermegoldásként a program a rendszerben lévő monitorprogram felügyelete alatt fut úgy, hogy minden programlépés után a vezérlés visszaadódik a monitornak. Így a regiszterek tartalmát folyamatosan figyelemmel kísérhetjük.

A lépésenkénti futtatásnak vannak azonban hátrányai is:

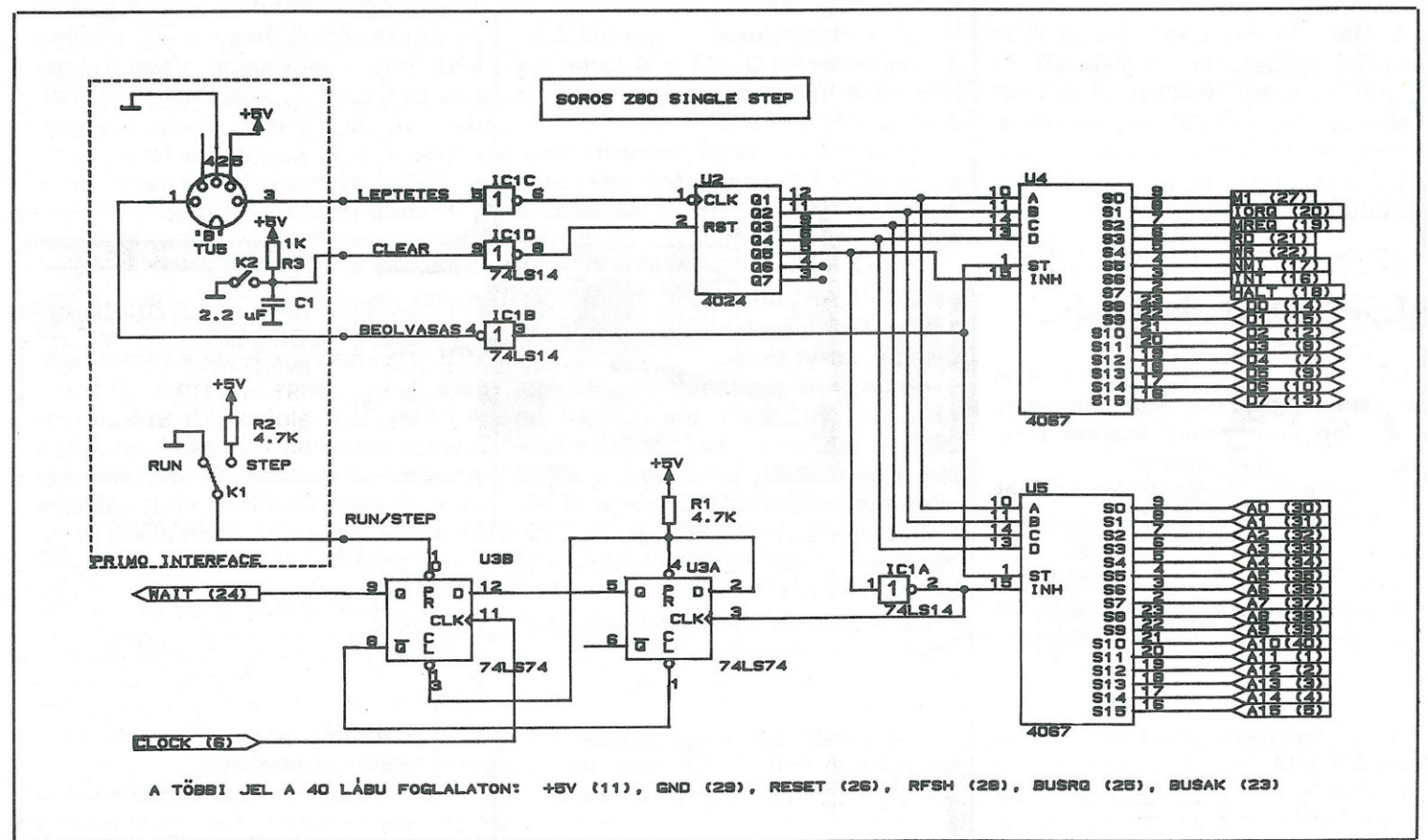
- a programvégrehajtás nagyon lelassul, nehéz kivárni egy ciklus lefutását,

- a módszer megszakításos rendszerekben csak korlátozottan használható, a rendszer időbeli viselkedése teljesen eltorzul.

Ezért ezt a megoldást csupán rövidebb programok vagy programrészek logikájának ellenőrzésére célszerű használni.

Dr. Kónya László

## Z80-as, hardver megoldású, lépésenkénti programfuttató rendszer



# BÖRZE



## COBRA COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET  
1067 Budapest IX., Hlataa út 7. Telefon: 476-100/300

SZÁMLAKÉSZÍTÉSTŐL  
A KÖNYVELÉSIG

### COBRACONTO

Ügyviteli programrendszer moduljai:

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

KÉRJEN RÉSZLETES TÁJÉKOZTATÓTI

## TUTTI

### ELECTROCOOP KISSZÖVETKEZET

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

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



## ENET

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



- Komplex  
számítástechnikai  
szolgáltatás!
- Szünetmentes  
áramforrás
- Új árkatálogos

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

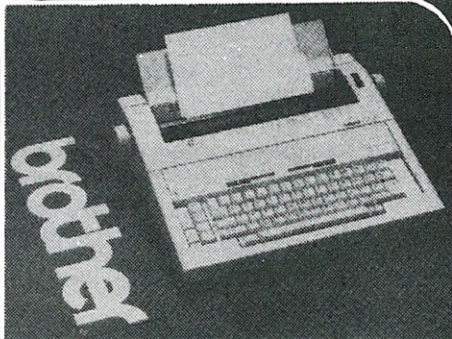


Kisszövetkezet  
megnyitotta

### COMPUTER SZAKÜZLETÉT

- hardver, szoftver
- elektronikai elemek
- integrált áramkörök
- számítógépek, perifériák
- blokkolóórák
- vonalkódeszközök, biztonsági  
rendszerek

SZEGED  
Kazinczy u. 8. Tel.: 62/12-259  
Berzsenyi u. 2. Tx.: 82-726



BROTHER AX 15 típusú,  
margarétafejes,  
elektronikus írógép

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



1146 Bp., AJTÓSI DÜRER  
SOR 10.  
Levél cím:  
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

# Adatvédelmi rendszer

## IBM PC XT/AT-re

A nagy- és minigépes operációs rendszerekkel ellentétben az MS-DOS-nak nincs olyan lehetősége, amellyel szabályozni lehetne a felhasználók hozzáféréseit a programokhoz vagy adatokhoz. Bár a legtöbb hálózati szoftvernél már meg lehet adni felhasználói jogokat, hiányzik a fájlok encrypt/decrypt védelme. Két probléma adódik. Egyrészt az esetek többségében a központi gép nincs annyira elzárva, hogy ne férhessenek fizikailag hozzá, másrészt a hálózat összekötő elemein végigfutnak az adatok, így ott is — minimális hardversegítséggel — elérhetők, ahol nem jogosultak a használatukra. A jelenleg forgalomban levő titkosító programok a gyakorlatban nem használhatók, mivel munka előtt és után a védendő fájlokat manuálisan kell kódolni, a védelem pedig fájlokhoz és nem személyekhez kötött.

A Secret nevű program olyan adatvédelmi rendszer, amely a felhasználói programok számára transzparens (azaz a programokat nem kell módosítani az adatvédelemhez), ugyanakkor hierarchikus védelmet biztosít — azaz a felhasználók a fájlok és alkönyvtárak bizonyos csoportjaihoz jogosultak hozzáférni. Mindezeket túl valódi fájltitkosítást ad, tehát nem lehet egyszerűbb vagy bonyolultabb trükkökkel — például nyomkövetéssel — visszafejteni a kódot vagy megtudni a jelszavakat.

Természetesen — amennyiben a közvetlen hozzáférést nem tiltjuk le — a rendszer nem tudja megakadályozni, hogy nyomkövetéssel vagy kódmódosítással a felhasználó hozzáférjen a fájlokhoz (a teljes elzárás nem oldható meg, hiszen például floppyról saját rendszert betöltve, a védelmi rendszer nem indul el, a fájlokhoz hozzá lehet férni), azonban a fájlok tartalmát valóban csak a jogosult felhasználók ismerhetik meg, más hozzáférési kísérlet csak értelmetlen kódsorozat eredményez. Ha a rendszert hardverkiegészítővel használják, megakadályozható a floppyról történő rendszerindítás is.

Az adatvédelmi rendszernek ez a verziója 64 hierarchiaszintet enged meg. Ezeket összesen 4500 fájlnev adható meg — ezek bármelyike tartalmazhat globális fájlnevkaraktereket is, azaz a védhető fájlok száma nincs korlátozva —, 800 kijelölt alkönyvtár hozzáférést lehet szabályozni, és legfeljebb 2500 egyedi felhasználó neve kerülhet a rendszerbe.

A rendszer lényege egy olyan tárrezidens program, amely az összes, MS-DOS-on keresztülműködő fájl be- és kivitelt „elkapja”, és speciálisan encrypt módszerrel reverzibilisen kódolja a ki-, illetve bemenő adatokat. A rendszer működhet önmagában és kiegészíthető kódoló/dekódoló hardverberendezéssel.

A rendszer a következő programokat tartalmazza: tárrezidens be/kivitel kódoló program, menüprogram, LOGON/LOGOFF programok, adminisztrátorprogram és segédprogramok.

### Be/kivitel kódoló program

A rendszer lényege egy tárrezidens program. Ez kódolja a védendő fájlokat kivitelnél, illetve dekódolja bevitelnél. A program „ráül” az INT-21-re.

Feladata, hogy „elkapjon” minden fájlmegegyezést. Ellenőrzi, hogy a megnyitni kívánt fájl szerepel-e a védett fájlok listáján. Ha nem, semmit nem csinál: transzparens az ilyen fájlokra. Ha igen, megnézi, hogy az adott felhasználó jogosult-e a fájl írására/olvasására. Ha nem, ACCESS DENIED hibakódot ad vissza. Ha igen, az OPEN-t végrehajtja a DOS-szal, és a visszatéréskor kapott FCB-t, illetve fájlazonosítót megjegyzi a későbbi be/kiviteli műveletekhez.

Minden más be/kiviteli művelet esetén csak akkor tesz valamit, ha a fájlhoz tartozó azonosítót elraktározta.

Beviteli műveletnél először beolvastatja a DOS-szal a szükséges blokkot, majd ezt dekódolja (decrypt).

Kivitelnél először a puffert kódolja (encrypt), majd a DOS-szal kiírta a már kódolt tartalmat a fájlba. Ha a felhasználó csak olvasásra jogosult, hibajelzéssel visszatér a programhoz. A puffert eredeti állapotára állítja vissza.

CLOSE esetén a fájl lezárása után az FCB-t, illetve a fájlazonosítót törli a listáról. Egyéb fájlműveleteket (például RENAME, DELETE) csak írásra jogosult felhasználóknak enged meg.

A floppyt csak a kijelölt felhasználók használhatják.

### Menüprogram

E programmal szabályozható a felhasználók munkája. Az adatadminisztrátor adja meg, hogy az egyes személyek milyen parancsokat, milyen paraméterekkel adhatnak ki, és hogy kiléphet-e a felhasználó a menüvezérlésből a rendszerbe vagy sem (természetesen a kilépéskor is működik a védelmi rendszer). Ilyenkor a menüprogramban van a bejelentkezés (LOGON) és a kijelentkezés (LOGOFF) is. Ha a hardverkiegészítő letiltja a floppy használatát, a rendszerindításkor pedig a rezidens program után a menüprogram aktivizálódik, az adott számítógépes konfiguráció teljesen védett az illegális használat ellen.

### LOGON program

A LOGON programmal a felhasználó azonosítja magát. Nevének beírása után a program bekéri egyéni jelszavát. Ezután a program ellenőrzi a beírt jelszó kontrollösszegét. Ha nem egyezik, a hozzáférési kísérlet jogosulatlan volt. Ha egyezik, az adott felhasználó tábláját bemásolja a rezidens területre, és beírt valódi jelszavával — egy bonyolult algoritmussal — konvertálja (decrypt) a megengedett szintek melletti jelszavakat. Ezzel az eljárással, ha jó egyéni jelszót adott meg, a szintek valódi jelszavai alakulnak ki — persze csak azoké, amelyekre jogosult. Más felhasználóról hiába állapítja meg — például nyomkövetéssel —, hogy mi az illető jelszó ellenőrzőösszege, a teljes egyéni jelszó ismerete nélkül nem tudja megállapítani a szintjelszavakat, és így a fájl tartalmát sem kaphatja meg. (Megjegyzés: a hardverelemet is tartalmazó védelem esetén a szintjelszavak szükségesek, de nem elégségesek a

fájltartalom olvasásához vagy írásához: szükség van még a később tárgyalandó hardvereszközre is, amelybe ilyenkor a felhasználói tábla bekerül. Ezzel a kettős védelemmel elérhető, hogy sem az egyéni jelszó véletlen megismerése esetén, sem külön, az eszköz ellopásakor nem kapható meg a fájl valódi tartalma.)

### LOGOFF program

Törli a rezidens program tábláit és adatpuffereit. A következő LOGON-ig nem lehet a floppyt használni, és a védett fájlok egyike sem érhető el. LOGOFF-nál a felhasználói tárterület is törölődik.

### Adminisztrátor- és menügeneráló program

Az adminisztrátorprogramot egyetlen felhatalmazott személy, az úgynevezett adatadminisztrátor kezelheti. Ez a program készíti el és módosítja a kódolt hozzáférési jogosultságot tartalmazó táblákat. Hardverkiegészítő esetén az eszközökbe ez írja be a felhasználói táblarészeket. (Ilyenkor a felhasználók módosíthatják saját jelszavaikat.) Az egyes felhasználókhoz rendelt menük — kiadható parancsok — elkészítése is az adatadminisztrátor feladata. A menük elkészítése után a generáló program a menürendszer számára kódolva tárolja az információkat és a HELP-eket.

### Segédprogramok

A rendszer különféle segédprogramokat tartalmaz. Közülük néhányat a felhasználók kezelhetnek (például fájlok átkódolása, másolása kódolással), a többi az adatadminisztrátor munkájának könnyítésére szolgál.

### Hardverleírás

Az adatvédelmi rendszer működhet csak szoftveresen is, de két okból is kiegészíthető hardverelemekkel. Egyrészt a bejelentkezést lehet valamilyen fizikai eszközhöz — az úgynevezett személyi azonosítóhoz — kötni, másrészt le lehet tiltani a rendszerindítást floppyról, hogy az adott számítógépes rendszert csak a merevlemezzel, az adatvédelmi rendszert is elindítva lehessen betölteni.

### Személyi azonosító hardver

Erre a hardverre elsősorban ott van szükség, ahol megkövetelik, hogy a titkosság valamilyen fizikai eszközhöz kötődjön, amivel a felhasználókat el lehet számoltatni. Lényegében az adott felhasználó paramétereit tartalmazza olyan formában, ahogyan az a szoftvernek a LOGON program által használt fájljában lenne. Persze ebben az esetben a fájl nincs a rendszerben! Ilyenkor a felhasználói bejelentkezéshez (a LOGON-hoz) mind a hardverelemet, mind a jelszót ismerni kell.

Bíró András



# Bajor gazdasági napok Budapesten: műszaki szimpózium

## 1989. 05. 15.

- 9.00 Megnyitó (külön meghívóval)
- (P) 11.00 **Bajor Idegenforgalmi Hivatal:**  
Turizmus Bajorországban — a jövő piaca
- (B) **Siemens AG:** Energiahelyzet 2000-ben
- (P) 13.30 **Poligrat GmbH:** Elektrokémiai úton polírozott felületek a gyógyszeripari készülékeknél
- (B) **Siemens AG:** Innováció a nyilvános telekommunikációban
- (P) 15.30 **Dr. Ing. Jürgen Gutmann:**  
Noninvazív érdiagnosztika ultrahang — Doppler készülékkel, fényreflex- és vénaelzáró-pletizmográfiával
- (B) **Dynacord:** PA-rendszer processzor-végfokozattal

## 1989. 05. 16.

- (P) 9.00 **Elau GmbH:** Elektronikus vezérlések gyakorlati alkalmazási lehetőségei a mobil technikában
- (B) **Friedrich Deckel AG:** Gépek és módszerek a szerszám- és formagyártáshoz
- (P) 11.00 **Elau GmbH:** CNC pozicionáló rendszerek és CNC vezérlések — Funkció, alkalmazás, gyakorlat
- (B) **Maschinenfabrik Niehoff GmbH Co KG:** A rézdróthúzás alapjai
- 13.30 **Fritschi GmbH:** Modern keménységvizsgáló módszerek gumi alkatrészek, „O” gyűrűk, tömítőgyűrűk keménység mérésére pontosan az eddig érvényes előírások szerint
- (B) **Chetra GmbH:** A müncheni Chetra Dichtungstechnik GmbH-től a budapesti Chetra Kft.-ig

- (P) 15.30 **Drei S—Werk:** Túléces nyújtógépek tüellátása a fejlődés tükrében
- (B) **Feodor Burgmann:** Modern tömítéstechnika a füstgáz-kéntelenítő berendezéseknél

## 1989. 05. 17.

- (P) 9.00 **JV Kunststoffwerk GmbH:**  
Szűrés polipropilén szűrőelemekkel
- (B) **ABN Apparatebau Nittenau GmbH:** Hordozható és vontatható kompresszorok: gazdaságos megoldások sűrített levegő alkalmazására külső- és belső munkahelyeken
- (B) 11.00 **Handwerkskammer OB:** A kézműipar, mint a polgári társadalom része. A kézműipar és az új technológiák
- (P) **Lütges Datensysteme:**  
Elektronikus adatfeldolgozási megoldások a polgári társadalomban
- (P) 13.30 **ID Design:** A formatervezés menete egy esettanulmányon bemutatva
- (B) **Weiler Werkzeugmaschinen:**  
Gazdaságos esztergálás hagyományos és CNC-esztergákkal
- (P) 15.30 **VDID Verbandes Deutscher Industrie-Designer:** A formatervezés terminológiája az alkalmazás során
- (B) **Wolland Telemetry:** A telemetria a múltban és a jelenben. Telemetriai rendszerek és alkalmazások

B = Bartók terem  
P = Palma terem

az IBM PC/XT és PC/AT között?

# Csak egy év, de micsoda fejlődés!



Magyarországon a professzionális személyi számítógép kategóriában szinte egyeduralkodóan terjedtek el az IBM PC/XT és AT gépek, valamint másolataik, a klónok. Vajon melyek az alapvető különbségek a két típus között?

A számítástechnikában a hardver fejlődése a gyorsabb műveletvégrehajtást jelentő nagyobb sebesség, a nagyobb, közvetlenül megcímezhető memóriataromány, az adatbusz szélességének növelése, az újabb és gyorsabb perifériák alkalmazása felé mutat.

A közös ős, az IBM által 1981-ben forgalomba hozott IBM PC (IBM Personal Computer = IBM személyi számítógép) volt. A PC-t főleg a 16 bites mikroprocesszora különböztette meg a korábbi generációkhoz tartozó 8 bites mikroszámítógépektől. Bár a külvilág felé az adatvonal-szélesség itt is csak 8 bites, de a belső felépítése valódi 16 bites kialakítású volt. Az IBM PC öt kártyacsatlakozót, magnetofonillesztőt és ROM-ban lévő BASIC-interpretert tartalmazott.

A PC nagy piaci sikere után, 1983-ban az IBM számos fejlesztést jelentetett be a gépen, és a modellt elnevezte IBM PC/XT-nek. Itt az XT az „eXtended Technology = kiterjesztett technológia” kifejezésben szereplő két betűre utal.

Tulajdonképpen az XT nem más, mint egy nagyobb teljesítményű tápegységgel és merevlemezegységgel — eleinte 10 Mbájtos Tandon gyártmányú meghajtóval — felszerelt, régi PC. Az XT alapkártyája a PC kissé áttervezett, módosított változata. A legfontosabb változás, hogy rajta 256 kbájt (újabb fejlesztésekben 640 kbájt) helyezhető el, a PC 64 kbájtjával szemben. Eltűnt a kazettaillesztő egység, és az eredeti öt kártyacsatlakozó helyett nyolcat építettek be. Az XT állandó részévé vált az aszinkron kommunikációs (soros RS232) kártya, amelyen keresztül más számítógépekkel vagy rendszerekkel lehet kapcsolatot teremteni.

Az XT alapkonfigurációja tehát 128 kbájt memóriát, egy merevlemez és egy hajlékonylemez meghajtót tartalmazott. A rendszerszoftver is fejlődött: megjelent az új, már a mai DOS-ok generációja, a PC-DOS 2.00-ás verziója.

A chipgyártási technológia eredményei azok a 8088-as mikroprocesszorok, amelyek a PC eredeti 4,77 MHz-es órajel-frekvenciájánál nagyobb sebességgel, 8 vagy akár 10 MHz-cel is működnek. A programkompatibilitás megtartása — például időzítés szoftverrel — és ezeknek a tokoknak az XT-ben való maximális sebességű alkalmazása egy, a gyakorlatban jól használható megoldást hozott: a turbósítást. Ennek az a lényege, hogy a rendszer működtető órajel-frekvenciáját egy kapcsolóval (hardver), vagy adott billentyűkombinációval (szoftver) átkapcsolhatjuk.

A sorozatos fejlesztések egy adott fázisban ismét egy új mikroszámítógéphez vezettek: ez volt az 1984 októberében bejelentett IBM PC/AT. Itt az AT az „Advanced Technology = haladó (fejlett) technológia” kifejezésben szereplő két betűre utal.

Az IBM PC/AT típus legfontosabb tulajdonsága, hogy egy valódi 16 bites mikroprocesszort, az Intel 80286-os mikroprocesszort tartalmazza. Ennél az adatvonal-szélesség 16 bites, és 24 bitre megnövelt címbusszal a 16 Mbájtnyi memória közvetlenül címezhető. Az XT-vel való kompatibilitás megtartása érdekében az eredeti bővítő

csatlakozósor és jelei változatlanul megmaradtak; az újonnan megjelent cím- és adatvonalak az eredeti csatlakozókkal párhuzamosan meghosszabbított 2 x 18 pólusú csatlakozósorra vannak kivezetve.

A 80286-os mikroprocesszornak két üzemmódja van. Az MSDOS alatt a tényleges címzésű üzemmódban fut, mintegy emulálva az IBM PC-ben lévő processzor működését. Ilyenkor a processzor a rendszermemóriából közvetlenül 1 Mbájtot tud megcímezni. A másik módban a processzor 16 Mbájtos fizikai memóriát címez. A minimális rendszermemória 256 kbájt, a maximális kiépítésben memóriabővítő kártyákkal akár 8 Mbájt is lehet.

Az AT 1,2 Mbájtos hajlékonylemez-meghajtót tartalmaz. Ezt a kapacitást egy nagy írássűrűségű (high density) lemez felhasználásával éri el. Ilyen lemezek a szokásos 360 kbájtos lemezegységeken nem használhatók. A nagy kapacitású meghajtók tudják olvasni az XT szabványos, 360 kbájtos lemezeit, de a rajtuk írt 360 kbájtos normál lemezek olvasása az XT 360 kbájtos meghajtóján nem garantálható.

Az AT opcionálisan kezdetben 20 Mbájtos, ma ennél jóval nagyobb kapacitású lemez meghajtót tartalmazhat.

A billentyűzet is megváltozott, az új más elrendezésű, és kétirányú kommunikációt végez. Ezért az XT és az AT billentyűzete nem cserélhető fel!

A gép előlapján elhelyezett kulcsos kapcsoló lezárásával a billentyűzet ki lehet kapcsolni. Így hosszabb feldolgozásoknál a gépet ott lehet hagyni, nem piszkál bele senki. A kulcs a személyi, kizárólagos használatot is biztosítja — ha csak másnak is nincs kulcsa hozzá. A kulcsra zárás ezenkívül mechanikusan megakadályozza a gép fedelének a levételét is.

Az AT-nek soros és párhuzamos interfész-kártyája van. A párhuzamos (Centronics) interfész ugyanaz, mint a PC/XT-n. A soros interfész az XT szabványos, 25 pólusú csatlakozója helyett egy 9 pólusú csatlakozót használ, ezért soros berendezéseknek a géphez csatlakoztatásához DB9—DB25 csatlakozóátalakítót kell használni.

Mivel az XT-kben hasznosnak bizonyult az aktuális idő és dátum jelzésére szolgáló külső óráramkör, az AT ezt már beépítve tartalmazza.

Az AT-ben lévő óra/naptár IC-t egy kis, 64 bájtos CMOS RAM-területtel is kiegészítették. Az óra tovább működik, és a RAM megőrzi a tartalmát a gép kikapcsolása után. Ez a RAM-terület tárolja azokat az információkat, hogy mennyi memória van a rendszerben installálva, és milyen lemez meghajtókat és perifériákat tartalmaz a rendszer. Így nincs már szükség a PC és XT alaplapon eddig alkalmazott, a konfigurációt beállító kapcsolósorra. Rendszergeneráláskor az ún. SETUP programot kell lefuttatni, amely egy menürendszer segítségével kérdezi le az aktuális konfigurációt, ezt a CMOS RAM-ba írja, és a további rendszerindítások során már innen a rendszerprogramok kiolvashatják. A CMOS RAM és az óra tápellátásáról a gép kikapcsolása után egy kis NiCd telep gondoskodik, melyet időnként cserélni kell.

Az XT-ben alkalmazott képernyőmeghajtó kártyák az AT-kben is használhatók.

A gépbe opcionálisan a 80287 típusjelű, a számítási műveletek elvégzésére optimalizált arit-

metikai processzor is betehető; csupán be kell helyezni az alaplapon lévő foglalatba.

Az IBM PC/AT 16, az XT 8 hardver megszakításvonalat és az előbbi 8, az utóbbi 4 közvetlen memória-hozzáférést — DMA-t — segítő vonalat tartalmaz. A megnövelt DMA-lehetőség igen alkalmassá teszi a gépet adatgyűjtési célokra. Az adatátvitel a külvilág és a memória között közvetlenül, a processzor tevékenységének mellőzésével, igen nagy sebességgel mehet végbe. Hogy újabb egységekkel is bővíthető legyen, a tápegység teljesítményét is megnövelték.

A technológiai fejlesztések eredményeként az eredeti 6 MHz-es órajel-frekvenciával működő AT-k után megjelentek a nagyobb sebességű, turbósított változatok, 8, 10, 12 MHz-re gyorsítva a gépet. A nagy sorozatban történő gyártás következtében gazdaságossá vált a gépben lévő, több áramkörből álló egységek egy tokba integrálása. Ebben a CHIP cég járt az élen, ezért nagyon sok AT-ben láthatunk ilyen, CHIP felirattal IC-eket. Ezek általában már nem a szokásos DIP-tokozásúak, hanem a kisebb felületen több kivezetést lehetővé tevő négyzet alapú, ún. „grid array” (ejtsd: grid erréj) tokozással készülnek. A gépben lévő nyomtatott áramkört lapok már több vezetőrétegből állnak, és az alkatrészek felületszerelt technológiával kerülnek a panelekre.

Az IBM PC/AT fejlettebb hardverjét az újabb operációs rendszerek is támogatják. Az első ezek közül a PC-DOS 3.0 verziójú rendszer volt, majd sorban megjelent a 3.10, 3.20, 3.30 és az ablakos-grafikus kezelést nyújtó 4.0-ás változat is. A rendszer parancsait a régi verziókkal való kompatibilitás megtartása mellett olyanokkal egészítették ki, amelyek az AT fejlettebb hardverje adta lehetőséget kihasználják. Vegyesen használhatók a különféle kapacitású és méretű hajlékonylemez és az igen nagy kapacitású merevlemez egységek. E DOS-verziók mindegyike csak az 1 Mbájtos rendszermemóriát tudja kezelni, de kezelőprogramokkal az 1 Mbájt feletti memóriataromány is felhasználható memórialemez vagy adattárolás céljaira.

Az egymás között cserélhető hardverelemeknek köszönhetően az XT és AT közötti különbségek nem mindig olyan élesek, mint ahogy ez az előbbiekből következne. Összerakhatunk például olyan gépet is, amely XT alaplappal, de 1,2 Mbájtos meghajtókat tartalmaz, és aztán találgathatjuk: ez most XT vagy AT? Természetesen a józan eszünk azt súgja, hogy az alapvetően meghatározó alaplap miatt a gép inkább TX, de igazából egy XT—AT vegyesgép.

Bár ismertetésünket az AT gépekkel lezártuk, a fejlődés természetesen nem állt meg: megjelent a még gyorsabb, még több memóriát tartalmazó, igen hatékony perifériakezelésű PS/2 (Personal System 2) típusjelű számítógépcsalád. Lehetőségeinek kihasználásához már egy minőségileg más operációs rendszerre, az OS—2-re van szükség.

Dr. Kónya László

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

# Merre tart a világ?

## ... És még mindig kártyák

A sorozat eddigi részeiben kizárólag arról írtam, hogy milyen hardverújdonságok határozzák meg a fejlődés irányát. Nem foglalkoztam azzal, hogy a fejlődésben döntő szerepet játszó cégek milyen, nem hagyományos eszközökkel segítik saját fejlesztéseiket, erősítve egyúttal saját pozícióikat is. Ilyen akció volt, amikor az Apple cég a Macintosh gépek elterjesztése érdekében a legnagyobb harmincöt egyesült államokbeli egyetlen hallgatóinak hatvanszázalékos árengedmény-

nyel árulta gépét. Ezzel ugyan átmenetileg esetleg ráfizetett, de a hallgatókat ehhez a géphez szoktatva elérte, hogy ez a géptípus széles körben elterjedjen.

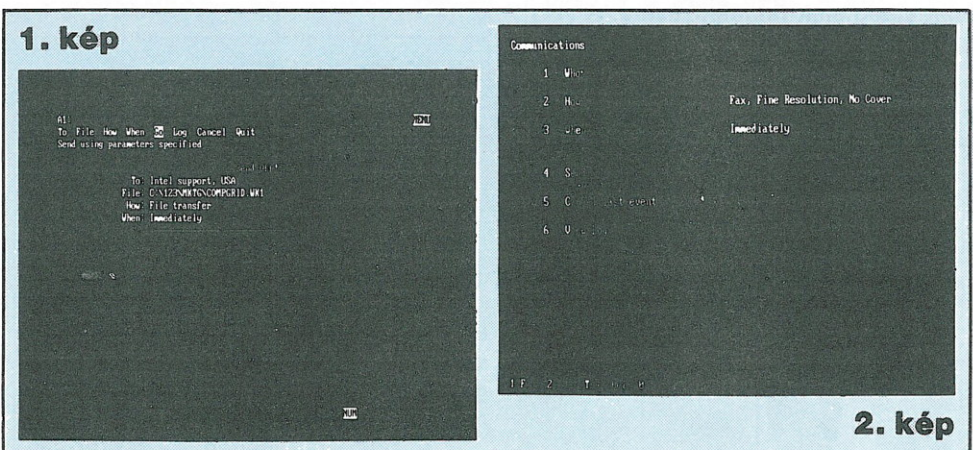
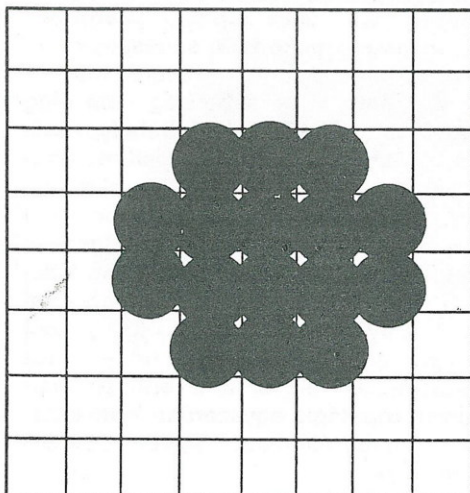
A másik igen sikeres tevékenység volt a klubok bevonása a fejlesztésekbe. Ennek két nagy sikerű példája az említett Macintosht kifejlesztő csoportban a HCC — a magyar klub névadójának — közreműködése és az IBM PC billentyűzetét kidolgozók között néhány más klub részvétele.

### Egyéb Intel-kártyák

A legújabb akció az Intel céghez fűződik. Mintegy 65 ezer dollár értékben ajándékba adtak Connection CoProcessor elnevezésű kártyákat a legnagyobb ötven PC-használó klubnak. E klubok szervezetének, az APCUG-nak az igazgatója, Jerry Schneider szerint ez lehetővé teszi számukra egy olyan világhálózat kiépítését, amelyen keresztül az ingyenes kiadványok, cikkek és más információ átvihető lesz. A kártyának saját kommunikációs szoftvere van, amellyel két irányban mozgatható szöveg, grafika és bináris fájl telefonvonalon keresztül (1. kép). Képes kommunikálni fax-berendezésekkel (2. kép). Tartozékként a klubok megkapták a Send-Off! elnevezésű szoftvert is, amivel kommunikálhatnak a Lotus 1-2-3 és WordPerfect 5.0 szoftvereken belül. További tartozéka egy 2400 bps sebességű modem is.

Mintegy öt évvel ezelőtt a „minőségi” nyomtatást a margarétakerekes nyomtatók képviselték a mikrogépes felhasználásokban. Jelenleg a lézernyomtatók a legelterjedtebbek. Ezek

1. ábra

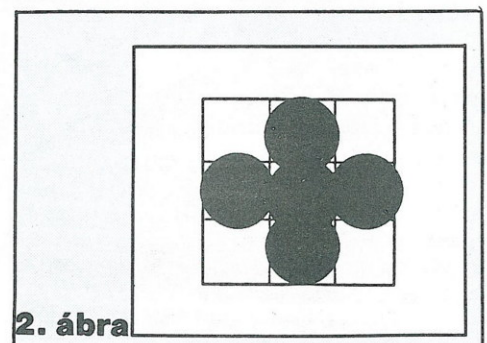


közül is a piac mintegy ötven százalékánál a Hewlett—Packard LaserJet Series II a meghatározó szerepű. Ezek az újabb eszközök lehetővé tették az ún. levélművészetet, és egyúttal a grafika nyomtatását is. Nem alkalmasak azonban a professzionális nyomtatási követelmények kielégítésére. Utóbbiak közül a legalacsonyabb minőségi követelményeknek az ún. desk top publishing (DTP) legmagasabb színvonalú része felel meg. A DTP-nél általában a PostScript lapleíró nyelvet használják és drága speciális nyomtatókat. A nyomtatványokba sokszor fényképeket is be kell építeni. A fényképek „beolvasására” léteznek már nem túl drága lapolvasók — scannerek —, de a lézernyomtatók 12 pont/mm felbontása ehhez nem elég.

Ezenkívül probléma az is, hogy egy kép mintegy 1 Mb-ot adtmennyiség. Ezt egy soros vagy párhuzamos csatornán át kell vinni, ami hozzávetőlegesen öt percig tart. Egy matematikailag egyszerűen leírható grafika ugyan a PostScript nyelvben kezelhető, de a fényképek nem ilyenek. A fekete-fehér fényképeken a szürkeség különböző fokozataitól lesz a kép felismerhető. Erre a nyomtatók nem képesek, mert csak vagy tesznek pöttyöt, vagy nem. Azaz vagy fekete, vagy fehér színnel dolgoznak. Ennek megoldásában az ún. féltó-

nus változat segít, a fekete színt nagy, a szürkét kis méretű pötty adja. Újságoknál ehhez 2,5-3,5, a fénymásolóknál 3,5-4 pontsor/mm felbontást használnak. A lézernyomtatók is használják ezt a módszert, de a nagy pontok képzése nem megfelelő. A 64 szürkeségi fokozat előállítására például csak 1,5 pontsört használnak. A négyponos felbontásnál már csak tíz szürkeségi fokozat lehetséges (1. és 2. ábra).

Az Intel cég új kártyarendszere, a Visual Edge megváltoztatta a pontformát (3. ábra). A négyponos felbontás mellett lehetővé teszi a 37 szürkeségi fokozatot (3. kép). A két kártya közül az egyik a szöveg átvitelére a számítógép soros vagy párhuzamos csatornáját használja, a másik a nyomtató video-csatornájára viszi a grafikát egy nagy



2. ábra

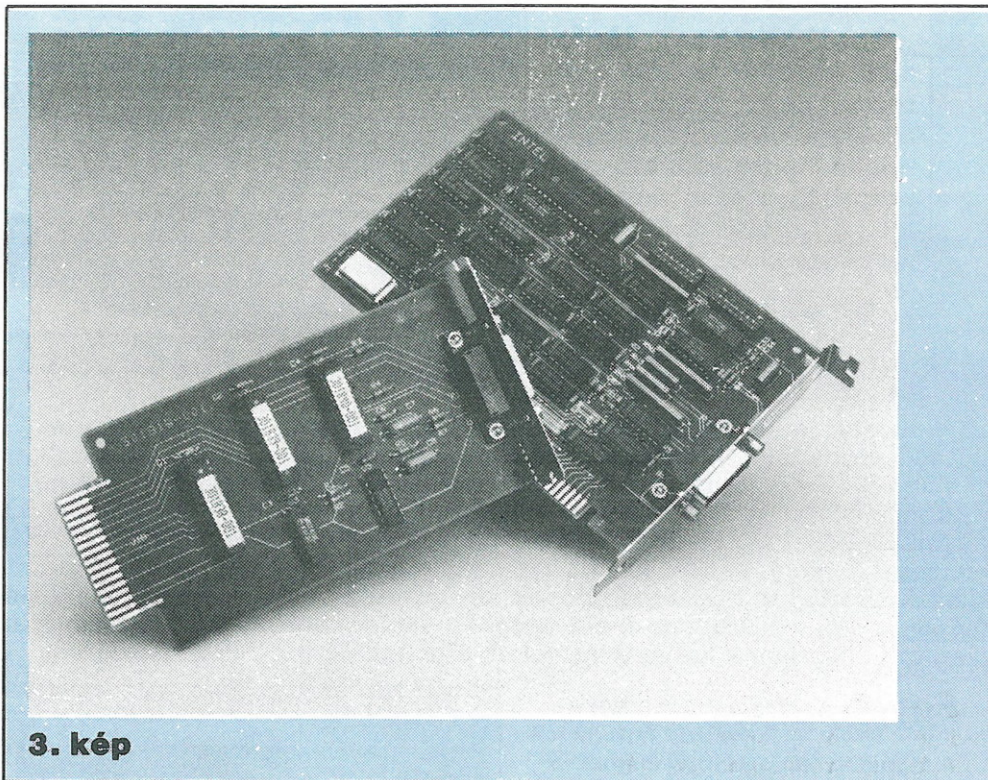
sebességű csatornán. A kártyákhoz a Digital Research elkészítette a GEM Artline szoftvert, amellyel a grafikatervezők kiválaszthatják a szűrkeségi fokozatot. A rendszert máris elfogadta a Xerox cég is.

## Commodore-kártyák

Az Amiga 2000 és 2500 (erről később írok) kiegészítésére készült az Amiga 2286D Bridgeboard. Ez a 80286-os kártya (4. kép), amellyel AT kompatibilitás érhető el. Mivel az Amiga több feladat együttes futtatására alkalmas, egyidejűleg futtathatók rajta az Amiga és MSDOS programok.

Ugyancsak az Amiga 2000-hez készült az A2350 PVA kártya (5. kép). Ez analóg és digitális videojeleket digitális feldolgozással RGB és kompozit videojelekké alakít. Két külső jelforrást kezelhet, átkapcsolhat Amiga video és kombinált videojelek feldolgozására. A színpalettát a felhasználó választhatja meg.

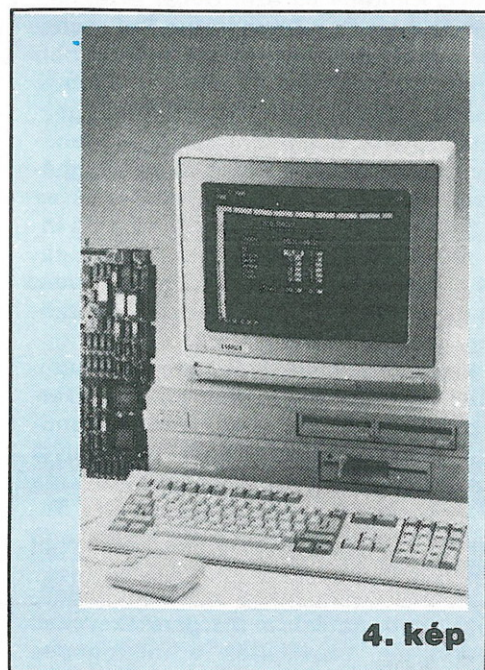
Egy másik processzorkártya az



3. kép

A2620. A kártyán egy Motorola 68020 mikroprocesszor van, ami 14,3 MHz frekvencián dolgozik, kiegészítve egy 68881 matematikai segédprocesszorral és egy 68851 virtuális memóriakezelővel (MMU). A kártyán 2 Mbájtos, 32 bites szervezésű tár is van, amely megkészszereshető. A kártya a nagyobb sebesség ellenére teljesen kompatibilis az Amiga-szoftverekkel.

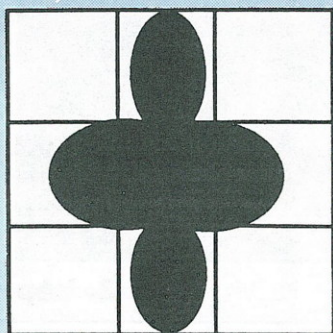
A harmadik segédprocesszor-kártya egy transzputer kártya (ilyenekről már volt szó a sorozatban). A T414 In-MOS processzorú kártya 15 MHz-en dolgozik. A kártyán 2 k 50 ns sebességű RAM van. A másik változatnál a processzor T800 típusú, a frekvencia 20



4. kép

MHz, a tár 4 k, és ez kiegészül egy, a 68882 típusúhoz hasonló lebegőpontos processzorral (FPU). Mindkét kártya négy soros, teljes-duplex csatornás. A második működési sebessége 1,2 Mflops (lebegőpontos művelet másodpercenként). Ha a sebesség nem elég nagy, akkor újabb kártyák behelyezésével tovább lehet párhuzamosítani, és a sebességet többszörözni. A párhuzamosság jobb kihasználására a régebbi operációs rendszert is kidolgozott, HELIOS néven. A rendszer többfeladatos, többfelhasználós és többfolyamatos. Az ilyen párhuzamosító kártyákból felépített hálózat előnye, hogy nincs adatütközés, felgyorsul a rendszer, bármilyen topológia egyszerűen kialakítható, és újabb állomások egyszerűen beépíthetnek.

Az asztali vetítőberendezések — a



3. ábra

5. kép





6. kép

félvezető-technika gyors fejlődésének eredményeként — egyszerű optikai készülékből számítógéppel is ellátott, bonyolult berendezéssé alakultak át.

### Kodak-berendezések

A cég itthon eddig kizárólag fényképezőgépeiről — de azokról már csaknem egy évszázada — és filmjeiről volt ismert. Mostanáig valóban ezek voltak az egyedüli termékei. Az utóbbi időben azonban a helyzet megváltozott. Nem hagyták ugyan abba a hagyományos termékek gyártását, fejlesztését, de tevékenységüket kiterjesztették az ezekhez kapcsolódó egyéb eszközök, többek között vetítőberendezések előállítására is.

Az LC500 (6. kép) „az első színes videovetítő, amely LCD (liquid crystal display = folyadékkristályos kijelző) technológiát használ” — mondta Charles F. Wilkinson, a cég kereskedelmi

igazgatója. Három LCD-elemmel mind analóg, mind digitális kompozit videó- és RGB-jeleket képes fogadni. Alkalmas CGA képek és dialemezek vetítésére. Tömege mindössze 6 kg, táskaméretű, és üzembe helyezése nem szakember számára is csak két percig tart.

A következő eszköz a professzionális felhasználóknak készült. A Datashow HR rendszer Macintosh és AT gépekhez csatlakoztatható (7. kép), és nagy képernyős, nagy felbontású vetítésekre alkalmas. Az AT változat CGA és EGA képekhez használható. A Macintosh-változat 512 x 342 pont felbontású. A kontrasztarány mindkét változatnál 20:1. Kezelése ennek is nagyon egyszerű, mindössze három gombja van: egy a kontraszt szabályozására, a másik kettő pedig a hálózati kapcsoló és a képfordító (pozitívból negatívba és viszont). A képeken mindenféle szoftvervezérelt módosítás is végezhető. A készüléket



7. kép

8. kép



infravörös távvezérlővel akár tíz méterről is lehet irányítani. A rendszer hardver része a más rendszerekben is használható Ektalite L5 típusú vetítő.

### Sharp-eszközök

A cég számára ez a terület az LCD-alkalmazások egyikét jelenti. A QA-25 LCD Computer Projector Panel (8. kép) a számítógépek által készített képeket vetíti ki, lágy szürke háttérrel és mélykék karakterekkel. Szobai fény mellett is használható. Mivel a cég ezt a berendezését elsősorban iskoláknak ajánlja, ezért az ára az előzőeknél lényegesen alacsonyabb. Ez is CGA kompatibilis, és IBM PC-hez vagy Apple II gépekhez kapcsolható.

Simonyi Endre

## Hagyomány és haladás

Bár a mezőgazdasági jellegű Bajorország a II. világháború után kialakult gazdasági helyzettel eleinte nehezen boldogult, a köztársaság gyorsan és tartósan modern ipari területté fejlődött. A „fényes gazdasági csoda” alapja és fejlődési állomásai a témája a Budapesten — 1989. május 7—16. között — megrendezésre kerülő „Bajorország — a hagyománytól a jövőbe” című kiállításnak. A Bajor Gazdasági és Közlekedési Minisztérium megbízásából az IMAG-München szervezi ezt a bajorországi kiállítást, mely válogatott kiállítási tárgyakkal és vállalati példákkal dokumentálja az agrárországból a magas színvonalú technika országává történő fejlődést.

„Bajorországban másként járnak az órák” — ez a szólás járta sokáig a köztársaságban, ami azt is jelentette, hogy lassabban, mint az NSZK iparvidékein. Bajorországban az agrárországból modern ipari területté való fejlődés lassabban ment végbe, mint más helyeken, de éppen ezért annál alaposabban. 1939-ben a bajor kereskedőknek még 40%-a a mezőgazdaságban volt foglalkoztatva. Ez 1957-ben 20% volt, ma már az összes bajor lakoságnak csak 4%-a keresi meg kenyerét saját földjén.

Bajorország ma már az európai gép- és járműjavítás, az elektro- és elektronikai ipar, a légi és távközlési ipar fontos központja. Ide telepítették azokat a technológiákat, melyek gazdasági jövőnket meg fogják határozni.

Egy „Bajorország — az ország és lakói” című kiállítás betekintést nyújt Bajorország földrajzába, kultúrájába, ősi kézműiparába, valamint állami intézeteinek életébe. A történelmi visszapillantás megmutatja, hogy a „fényes gazdasági csoda” semmiképpen sem a semmiből eredt.

A kiállítás fő vonala a jármű- és gépjáratás, a kémia és környezettechnika, az elektroipar és elektronika ágazataira tagozódik. Ezenfelül bemutatkoznak a köztársaság légi és távolsági közlekedési és egyéb magas színvonalú technikával dolgozó vállalatai is.

Egy külön részlegben „Magyarország és Bajorország” címszó alatt szó lesz arról a szoros kapcsolatáról, amely mindig is fennállt a két Duna menti terület között.

50 cég, köztük olyan nagyvállalatok mint a Siemens, a BMW és a MAN, ezenkívül kis- és közepes vállalatok széles skálája is bemutatkozik szöveges és képillesztrációkkal, számos termékmodellel és kiállítási tárggyal. Láthatók lesznek az iparosodás kezdeti idejéből származó régi gépek is, csakúgy mint a legkorszerűbb technológia, valamint a fogyasztási cikkek. Rövidfilmek adnak betekintést a bajor vállalatok színvonalába és termékskálájába, és a látogatóknak ismét lehetőségük lesz végigkövetni modelleken, gépeken és számítógépeken keresztül azt a változást, amelyen Bajorország az utóbbi években átment.

A Bajor Gazdasági Minisztérium hivatali főnöke, Hans-Martin Jepsen úr, miniszteri osztályigazgató, aki az országos kiállítást Magyarországon 1988 novemberében a sajtónak bemutatta, azt hangsúlyozta, hogy „ilyen nagyszabású rendezvényt a köztársaság eddig még egyetlen

más országban, sem Nyugaton, sem Keleten nem rendezett”.

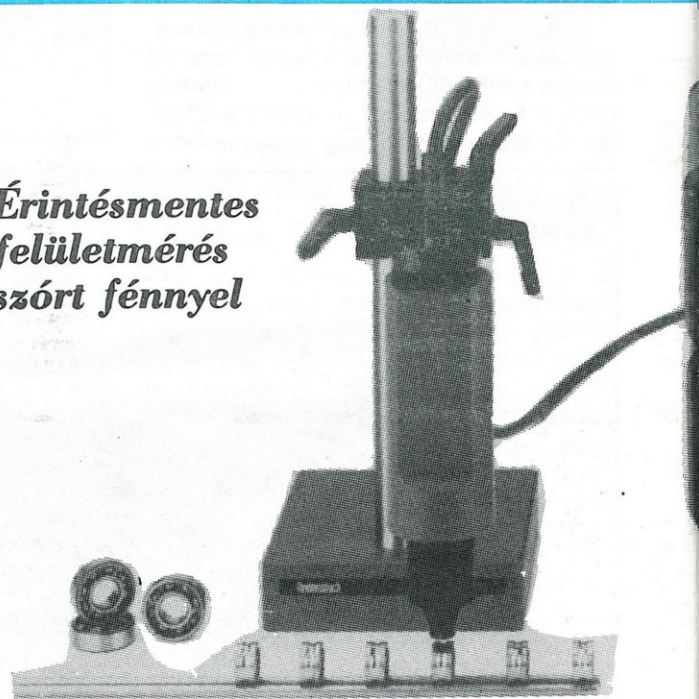
A program kiegészítéseképpen, széles körben szakosított szimpóziumrendezvényre is sor kerül, mely alkalmat ad a bajor és magyar üzletembereknek tapasztalat- és eszmecserére. Egy nagyszabású könyvkiállítás teszi teljessé a rendezvényt, mely bajor könyvkiadók műszaki, gazdasági és kulturális témákkal foglalkozó érdekes kiadványait mutatja be. Ráadásként pedig egy bajor specialitásokat felvonultató sörkertet állítanak fel a Budapesti Kongresszusi Központ területén, ahol a látogatók megismerkedhetnek a köztársaság konyhaművészetével is.

A magyar Társasági Törvény 1989. január 1-jei életbe lépésétől mindkét fél azt várja, hogy új impulzusokat ad a bajor—magyar gazdasági együttműködésnek. (A bajor vállalatoknál ez iránt igen nagy az érdeklődés, ez abban is tükröződik, hogy, mint már említettük, ötven cég vesz részt a rendezvényen.)

Ezzel a kiállítással új fejezet kezdődhet a bajor—magyar gazdaságok történetében. Bárcsak teljesülne minden elvárásunk!

### Mikrométer-tartományon belüli érintésmentes dimenziómérés

Érintésmentes felületmérés szórt fényvel



**Rodenstock**  
**RM 400 és RM 500**

# A nagyszabású budapesti rendezvény

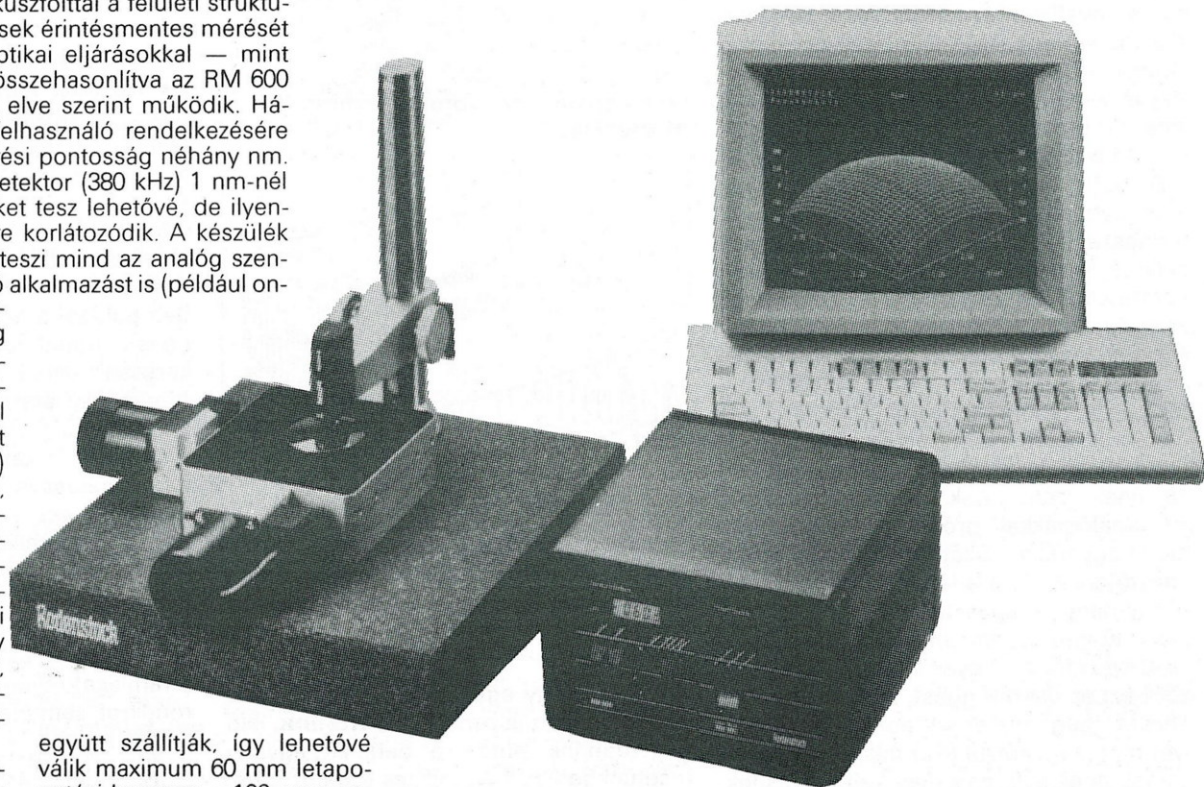
## átállítja a gazdasági váltókat

A Rodenstock RM 600 Laser Stylus készülékkel olyan egyszerű felületi profilmérő áll a mérés technikások rendelkezésére, ami  $1\ \mu\text{m}$  átmérőjű fókuszfóttal a felületi struktúra, kiemelkedések-bemélyedések érintésmentes mérését teszi lehetővé. Más ismert optikai eljárásokkal — mint például a háromszögelés — összehasonlítva az RM 600 szenzor dinamikus fókuszálás elve szerint működik. Három mérési tartomány áll a felhasználó rendelkezésére ( $600\ \mu\text{m}$ ,  $60\ \mu\text{m}$ ,  $6\ \mu\text{m}$ ), a mérési pontosság néhány nm. A különlegesen gyors fókuszdetektor ( $380\ \text{kHz}$ )  $1\ \text{nm}$ -nél nagyobb pontosságú méréseket tesz lehetővé, de ilyenkor a méréstartomány  $4\ \mu\text{m}$ -re korlátozódik. A készülék moduláris felépítése lehetővé teszi mind az analóg szenzoroként — kedvező áron — való alkalmazást is (például online érdességmérés gyémánt-esztergagépen), mind pedig komplett, számítógép-vezérelt, 3 dimenziós mérőállomással való fejlesztését is. Ezzel olyan mérési feladatok, mint lakkozott lemezek (narancshéj) topográfiai meghatározása, elektromos érintkezők síkságmérése, vastagréteg-technikában a frissen nyomott szerkezetek dimenziómérése gyorsan és különösebb beállítási igény nélkül elvégezhető. Egy  $4 \times 4\ \text{mm}$  nagyságú felületen,  $50\ 000$  mérési pont felvételéhez csak  $125$  másodperc szükséges.

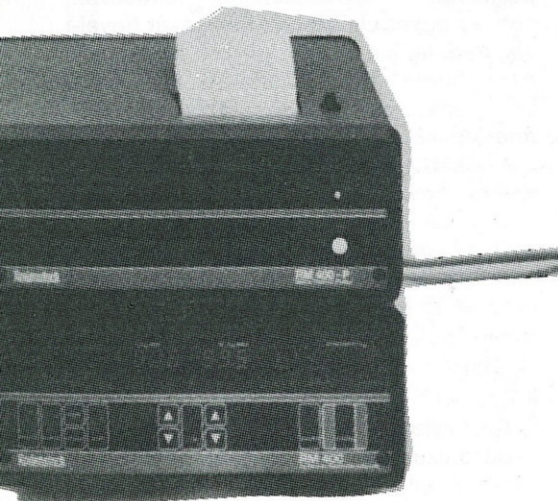
Egy nagy pontosságú előtöltő egységgel összeépítve az RM 600 Laser Stylus érdességmérő készülékként is alkalmazható. Ezt a kétdimenziós mérőegységet egy stabil, rezgés-csillapított gránitasztallal

együtt szállítják, így lehetővé válik maximum  $60\ \text{mm}$  letapogatási hossz  $\approx 100\ \text{nm}$  pontosságú profilmerések elvégzése.

(Referencia: Budapesti Műszaki Egyetem, Gépipari Technológiai Tanszék, dr. Főrián István vagy Faragó József.)



## Rodenstock RM 600



Finommegmunkált alkatrészek, mint forgattyús tengely, dugattyúcsapszeg, vagy finomlemezek érdességének és alakeltéréseinek meghatározása a korszerű gyártási folyamat számára kiemelkedő jelentőséggel bír. Növekvő mértékben mutatkozik igény olyan mérési eljárások iránt, amelyek elég gyorsan és megbízhatóan mérik a gyártás folyamán előforduló érdességválto-

zásokat vagy alakeltéréseket. A gyártási mérésekhez az eddig elsősorban alkalmazott mechanikus módszerek az üzemi feltételek között teljesen automatizált mérésre kevésbé alkalmasak.

A modern optoelektronika a gyors szenzorokkal napjainkban lehetőséget nyújt a felületek érdességének és alakváltozásainak optikai úton történő megállapításához. A Rodenstock cég által — az autó- és gördülőcsapágy-ipari felhasználókkal együttműködve — kifejlesztett RM 400 és RM 500 készülékek az ismert fényoszóródás elvén alapulnak; a felületről visszaszóródó fényt egy diódasorral érzékelik és mikro-számítógéppel értékelik ki. Mindkét készülékben van egy nagy pontosságú mérőobjektív a szórási szög detektorába történő leképezésre, ami által nagy mérési pontosság érhető el, és egyben a készülék az ipari környezet iránt érzéketlenné válik. Az optikai eljárás gyakran panaszolt szennyeződéserzékenységét egyszerűen sűrített levegős tisztítással küszöbölik ki.

Az RM 400 érdességellenőrzésre és többek között köszörülési és hideghengerlési eljárások ellenőrzésére alkalmas. Fizikai okokból szükséges egy új érdességi mérőszám, az  $S_N$  paraméter bevezetése.

Ennek az  $S_N$  értéknek igen magas az információ tartalma, mivel a mechanikus

módszerrel ellentétben egy teljes felületről ad értékelést. Ez azonban hátrányos is lehet ott, ahol szabványos mechanikus módszerrel mért paraméterek — amelyek csak vonalas kiértékelésre vonatkoznak — és az  $S_N$  közvetlen összehasonlítása fontos.

Az  $S_N$  érték viszont — összekötve folyamatellenőrzéssel — technológiai zavarokról, mint például a köszörűkorona vagy a henger kopása, közöl pontos adatokat.

Az RM 500 a visszaszóródó fény szöghelyzetét is kiértékeli, ezáltal — matematikai számítással — arra is képes, hogy egy kör alakú rész geometriai formaváltozását pontosan mérje. Már kipróbálták forgattyús tengely csapágyrezgésének vagy hajtómű-tengelyek pontos kör alakjának mérésénél.

Az elért pontosság (kisebb  $1\ \mu\text{m}$ -nél) hasonló a modern alakmérő készülékek pontosságához. Ez utóbbi csak speciális mérőszobákban alkalmazható, míg az RM 500 az üzemi környezetben is nagy pontosságú mérési eredményeket szolgáltat.

Az RM 500 készülék mellett a kör alak mérésére egy nagy pontosságú forgatókészülék is szükséges, ami a munkadarab megfelelő forgatását végzi. A köralakdiagram, a Fourier-spektrum, a P+V értékek és az excentricitás adatait vagy egy egyszerű termikus nyomtatón, vagy egy külső számítógépen keresztül adja ki a készülék.

## A villamosságtan alapfogalmai

A villamos alapfogalmakhoz, melyeken az elektronika is nyugszik, annak az elvnek a következetes betartásával célszerű közeledni, hogy egy mennyiség definíciója adja meg egyúttal a kérdéses mennyiség elvi mérési utasítását is, továbbá ez az elvi mérési utasítás álljon a lehető legközelebb a *gyakorlati* mérési utasításhoz.

A fogalmak kialakulása-kialakítása a tudományban nem esetleges: a tudomány a természet jelenségeinek leírásánál használatos fogalmait a *mindennapi élet* tapasztalataiból absztrahálta. Például a *mechanika* alapvető fogalmai (a hosszúság, a tömeg, az idő) már igen régen mindenki számára szemléletesek, érzékelhetők voltak, és a tudományban is ezek nyertek meghatározott, kvantitatív értelmet.

Addig, amíg a villamos jelenségek újszerűnek számítottak, gyakran mechanikai analógiákkal próbálták megérteni, megmagyarázni őket. (Megmagyarázni annyit jelent, mint a logika törvényei szerint néhány, a tapasztalat alapján is igaznak elfogadott *általános* összefüggésre visszavezetni az egyes mennyiségek közötti azt az összefüggést, melyet *egy-egy konkrét* megfigyelésnél tapasztalunk, melyet megmagyarázni akarunk. A magyarázatnak nemcsak logikailag kell helyesnek lennie, hanem fel kell keltenie a megértés *érzetét* is: az új jelenséget magától értetődően be kell tudnunk illeszteni világképünkbe. A magától értetődőség pedig kizárólag a *megszokásból* adódik.)

A villamosságtan alapfogalmai (mennyiségei)

- a **hosszúság** (l), mértékegysége a méter (m);
- a **tömeg** (m), mértékegysége a kilogramm (kg);
- az **idő** (t), mértékegysége a másodperc (s);
- az **áramerősség** (I), mértékegysége az amper (A).

A hosszúságot, a tömeget, az időt mindenki „ismeri”, az „új” mennyiség a villamosságtanban járatlanok számára az **áramerősség**. Ennek mértékegységét — az ampert — így definiáljuk:

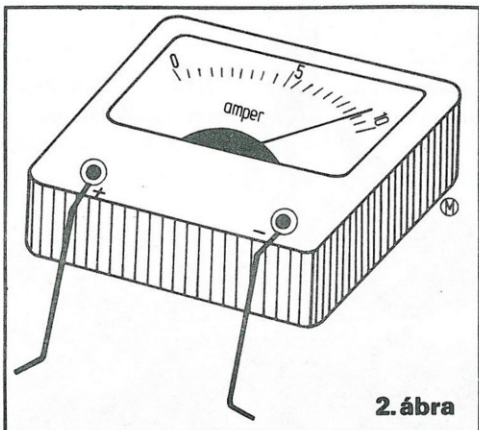
*1 amper erősségű áram akkor folyik egy vezetékben, ha az a vezető egy vele párhuzamosan futó, végtelennek tekinthető, tőle 1 méter távolságban elhelyezett másik vezető 1 méter hosszú darabjára — ha abban ugyanaz az áram folyik —  $2 \times 10^{-7}$  N erővel hat. (Lásd az 1. ábrát.)*

- a) Kapcsolási séma az áramerősség egységének megállapításához
- b) Az elvi elrendezés perspektivikus képe
- c) A kapcsolási séma a gyakorlatban

(Simonyi Károly Villamosságtan I. című könyvéből, a szerző engedélyével)

## Az ampermérő

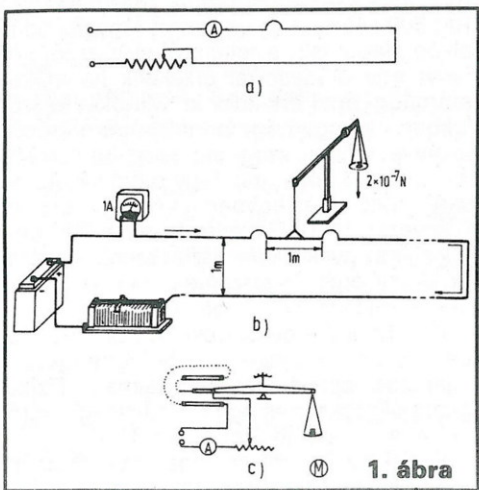
Biztosan sokan láttak már mindennapi gyakorlatban szokásos árammérő műszert (ampermérőt). Tudják, hogy nem hasonlít az 1. ábrán látható elrendezéshez, inkább a 2. ábrán láthatóhoz. A műszer fogalmához szorosan kapcsolódik a **hitelesítés**.



2. ábra

## Ampermérő

Ahhoz, hogy egy tetszés szerinti hatáson alapuló műszert **hitelesítsünk**, elő kell tudnunk állítani a definiált egység (esetünkben az 1 A) tetszés szerinti **többszörösét**, illetve **tört részét** is. Minthogy a definíció kapcsán csak az áram egységének, az *amperek* (1 A-nek) a hitelesítéséhez, előállításához készítettünk elrendezést, gondolatmenetünkben még nem jutottunk el az **árammérő műszerig**. Kérdés: például miből állapítható meg, hogy egy vezetéken az előző áram kétszerese vagy többszöröse folyik-e? Magából az erőhatásból nyilvánvalóan nem, mert éppen további kísérletek célja lesz (lenne) megállapítani, hogyan függ ez az erőhatás az áramerősség nagyságától. Azt azonban feltétlenül meg tudjuk állapítani, hogy mikor van egy áramkörben **ugyanakkora** áramerősség, mint a másikban: akkor tudniillik, ha mindkét áramkör környezetében **ugyanazon hatásokat** ész-



1. ábra

leljük. Így például két azonos áramerősséget vivő vezeték *egygyé fogva* olyan vezetékünk van, amelyben az egyikben folyó áram *kétszerese* folyik. Ez természetesen tetszés szerinti számú vezetékkel elvégezhető, és így tetszés szerinti pontossággal elő tudunk állítani különböző áramerősségeket, és **hitelesíteni** tudunk kényelmesen kezelhető — célszerűen kiválasztott hatásokon alapuló — árammérő műszereket is. Ezeknek a műszereknek csak olyan tulajdonsággal kell bírniuk, hogy a rajtuk keresztülhaladó áramtól egyértelmű módon térjenek ki. Mai mutató műszereinkben például a mérendő áram egy permanens mágnes pólusai között elforduló tekercsen folyik keresztül. A tekercs addig fordul el az áram hatására, amíg az áram elfordító hatása egyensúlyba nem kerül a műszerbe épített hajszálrugó ellentétes irányú hatásával.

Vegyük észre, hogy a villamosságtan a mechanikához képest egyetlen új alapfogalma (mennyisége, mértéke) definiálásánál, mérésénél teljesen elfeledkezhettünk arról, hogy az általunk mért áram valamilyen *töltéshordozók* mozgása (áramlása) révén valósul meg: sem elektronokról, sem más töltéshordozókról nem kellett közben beszélnünk. És itt nem valamilyen alkalmi módszerrel állunk szemben: a mai számítógépeink (a hardver) alapját képező elemi elektronikus folyamatokban (olyanokban, mint például egy kondenzátor feltöltődése — hogy pontosan mi a kondenzátor, arra még visszatérünk) olyan *nagyszámú* elektron vesz részt, hogy tipikusan megelégedhetünk a jelenségek *makroszkopikus* leírásával: nem az *egyes* elektronok mozgását figyeljük, írjuk le, hanem eltekintünk az események „finom” struktúrájától, és csak arról beszélünk, ami számunkra az adott esetben érdekes.

Másképpen fogalmazva: a *mikrorendszerek* egyes egyedeinek tulajdonsága vagy állapota nem jelentkezik közvetlenül egy *makromérésben*. Minthogy ilyenkor mindig a részecskék igen nagy *sokasága* vesz részt egy makrojelenség létrehozásában, azok tulajdonságainak valamilyen időbeli vagy térbeli *átlagértéke* szerepel a makroösszefüggésekben.

Aki valamennyit is foglalkozott már elektronikus szerkezetekkel, vagy járt már olyan munkahelyen, ahol ilyeneket mértek, javítottak, az tudja, hogy a munkaasztalokon az *árammérő*, az *időmérő* és a ritkábban előforduló *hosszúságmérő* mellett nem *tömegmérő*, hanem tipikusan *feszültségmérő* található. Hogy mi az elektromos feszültség, arra még visszatérünk. (Lásd a 3. ábrát.)

A *méterrudat* és a *stopperórát* — pontosabban a stopperóra helyett a mérendő idők rövidege miatt gyakran használatos ún. *oszcilloszkópot* — nyilván nem



és talán önök is olvasták előző számunkban az elektronikáról szóló eszmefuttatásunkat. Most ezt folytatjuk, segítségül hívva — sok helyen szó szerint vagy majdnem szó szerint átvéve a megfelelő részeket — Simonyi Károly professzor alapvető munkáit (az alább felsorolt művek közül az első kettőt), melyek egymást követő kiadásain villamosmérnökök generációi nőttek fel, melyekhez felvilágosításért, eligazításért pályafutásuk során később is biztonsággal fordulhattak. (Simonyi Károly: *Villamosságтан, 5. kiadás. Budapest, Akadémiai Kiadó, 1983. Elméleti villamosságтан, 10. kiadás. Tankönyvkiadó, 1987. Elektronfizika, 5. kiadás. Tankönyvkiadó, 1987. A fizika kultúrtörténete, 3. kiadás. Gondolat Kiadó, 1986.*) Simonyi professzor úr szíves jóváhagyásával az idézőjeleket az alábbi szövegből elhagytuk.

hagyhattuk el, mert az elektronikus jelenségek térben és időben játszódnak. A kevésbé gyors időbeli lefolyású jelenségek esetében, viszonylag „lassú” elektronikus szerkezetek vizsgálatánál a jelek gyors terjedése miatt gyakran élünk (élhetünk) azzal az egyszerűsítéssel, hogy elektronikus szerkezeteinkben a jelenségek gyakorlatilag „egyidejűleg” játszódnak le;

- a) A mechanika alaplászerei: az óra, a méterrúd és a mérleg (vagy az erőmérő rugó)
- b) A villamosságтан elvi alaplászerei: az óra, a méterrúd, a mérleg és az ampermérő
- c) A villamosságтан gyakorlati alaplászerei: az óra, a méterrúd, a voltmérő és az ampermérő (Simonyi Károly Villamosságтан I. c. című könyvéből, a szerző engedélyével)

a jelek terjedése a lejátszódo folyamatokhoz képest „végtelen gyors”, ezért a szerkezetek térbeli kiterjedése miatt a terjedési időktől eltekintünk, így készletünkben a „métrerúd” is gyakran elhagyhatjuk. Határesetekben úgy szoktunk eljárni, hogy előírjuk: az összekapcsolt és együtt vizsgált kritikus részek ne legyenek egymástól távolabb, mint mondjuk 0,15 méter (lásd például egyes ECL-típusú, gyors integrált áramkörti családok tervezési előírásait).

### Az elektromos feszültségről

A mai világszerte használatos SI-mértékegységrendszerben az elektromos feszültség mértékegysége, a volt (jele: V) ún. származtatott mértékegység. A tömeg (az erő) alapfogalmának kiváltásához emlékezzünk vissza fizikai tanulmányainkra: a villamos mennyiségek több egyenleten keresztül kapcsolódnak a mechanikai mennyiségekhez.

Anélkül, hogy már itt a feszültség definíciójára kitérnénk, emlékezetbe idézzük, hogy a teljesítményt — az időegység alatt végzett munkát — kiszámíthatjuk az elektromos feszültség és áram szorzataként is, és az időegység alatt megtett út és az erő szorzataként is. (Az olvasó szíves megértését kérjük, amiért a szoros fogalmi építkezéstől eltekintünk, és a teljesítmény és az erő fogalmát is ismertnek tételezzük fel.) Így tehát:

$$1 \text{ VA} = 1 \frac{\text{Nm}}{\text{s}}, \text{ amiből}$$

$$1 \text{ N} = 1 \frac{\text{VAs}}{\text{m}}$$

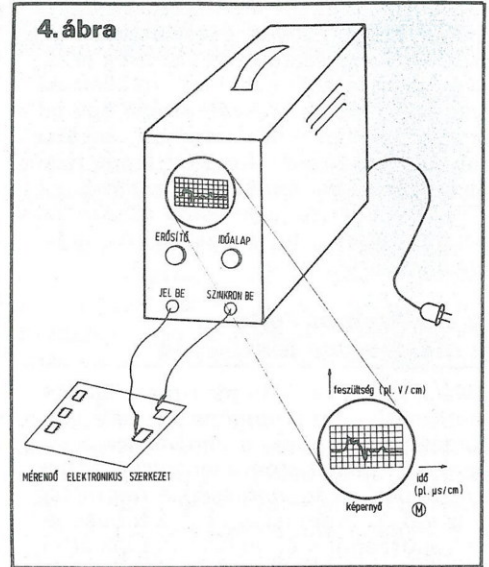
Vagyis az erő, melynek mértékegysége a newton (jele: N) és ezáltal — a mechanikai egyenleteken keresztül közvetve — a tömeg is kifejezhető a hosszúság, az idő, az áramerősség és az újonnan segítségül hívott feszültség alapegységeivel. A gyorsulás, a tömeg és az erő Newton óta ismert összefüggése alapján:

$$1 \text{ kg} = 1 \frac{\text{VAs}^3}{\text{m}^2}$$

### Az elektronika alaplászerei

Mit tettünk tehát? Először az óra, a mérleg (az erőmérő) és a méterrúd segítségével rögzítettük az áramerősség önkenyves alapegységét, majd módszert ta-

láltunk egy bármilyen célszerű hatáson alapuló gyakorlati ampermérő hitelesítésére, végül az elektronikus gyakorlat számára a tömegmérő „alaplászert” kiváltottuk egy feladatainkhoz alkalmasabb feszültségmérővel. Megemlítettük még, hogy a nem túl gyors működésű, tipikusan a másodperc százmilliomod részénél nagyobb kapcsolási idejű rendszerek vizsgálatánál, mérésénél fiókunk mélyén fe-



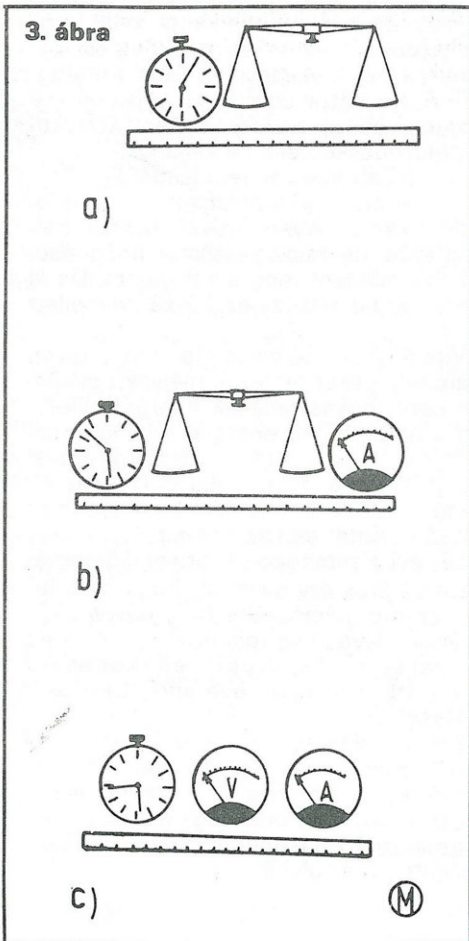
Oscilloszkóp

lejthetjük a „mérőrudat” is. Képzletbeli elektronikus munkapadunk tipikus alaplászerei tehát az árammérő, a feszültségmérő és az időmérő.

Az elektronikus szerkezetekben lejátszódo jelenségek olyan gyorsak, hogy feszültség- és időmérésre, de gyakran árammérésre is stopperóra helyett egy kombinált műszert, az oszcilloszkópot kell használnunk (lásd a 4. ábrát). Ha az oszcilloszkópot a vizsgált elektronikus szerkezet valamelyik pontjára kapcsoljuk (az oszcilloszkóp mérőszondáját odaérintjük), akkor az oszcilloszkóp képernyőjén megjelenik mondjuk az elektromos feszültség időbeli lefolyásának megfelelő rajzolat („görbe”, „hullámalak”): az oszcilloszkóp a feszültséget a képernyő függőleges tengelyére vetített távolsággá, az időt a vízszintes tengelyre vetített távolsággá transzformálja.

Azt, hogy mikortól kezdődjenek az idő mérése, az oszcilloszkópon külön be lehet állítani; az eszköznek van például egy külön szinkronjelbemenete. A mérés akkor indul, amikor az oszcilloszkóp az erre adott szinkronjelre érzékelt.

A gyakorló elektronikusok sokat erő tapasztalatból tudják, hogyan kell bánni az olyan, ma már nagyon sokféle fogásrútrútrákra alkalmas, kifinomult mérőeszközrel, mint az oszcilloszkóp.



Tegyünk itt egy kis kitérőt. Ma, amikor az elektronikus összeszerelő ipar szabványos, digitális integrált áramkört választékból (áramkör-saládokból) építkezik, melyekben a jelek nagysága (amplitúdója), felfutási ideje is szabványos — két értéket vehet fel megadott tűrészathatárokon belül —, gyakran elég csak egy-egy jel meglétét vagy hiányát érzékelni. A jelek konkrét hullámalakjának képi megjelenítése általában már nem is érdekes. Ezért szabványos, digitális áramkörök vizsgálatánál a hagyományos oszcilloszkópokat kiválthatják egyszerűbb *indikátorok* is, illetve beléphetnek helyettük (mellettük) olyan jelérzékelők, melyekkel egy-egy jel konkrét „hullámalakja” helyett inkább több jel logikai összefüggéseit lehet vizsgálni. Ezek az ún. *logikai analizátorok*.

E kitérő után térjünk vissza az alapokhoz, mert a feszültség fogalmát túl nagyvonalúan vezettük be.

### Az elektromos töltés, az elektromos feszültség

Már volt róla szó, hogy a mai digitális, elektronikus számítógépeink alapjául szolgáló szerkezetek, a hardver működésének leírásánál három mennyiség játszik fő szerepet: az *áramerősség*, a *feszültség* és az *idő*. Az áramnál az esetek többségében rendszerint megfeledezhetünk arról, hogy ezzel a fogalommal számunkra „lát-hatatlan” töltéshordozók mozgását írjuk le makroszkopikusan. Témánk szempontjából megelégedhetünk azzal, hogy az áram értéke annyi, amennyit a hitelesített árammérő műszerünk mutat.

Vezessük most be az elektromos *töltés* fogalmát. Azt az elektromos töltést, ami egységnyi (1 amperes) áramerősség esetén 1 másodperc alatt halad át egy adott felületen (például egy vezeték adott keresztmetszetén), *egységnyi töltés*-nek nevezzük. Az elektromos töltés egysége a *coulomb* (betűjele: **C**). Egy coulombnyi töltés mintegy  $6,25 \times 10^{18}$  darab — tehát elképzelhetetlenül sok — elektron töltésének felel meg.

A Coulomb-törvény szerint az ellentétes töltések vonzzák, az azonosak taszítják egymást. Az erőhatás nagyságát vákuumban (azaz úgy, hogy a két pontszerű töltés közötti tér vagy vákuum, vagy a teret levegő tölti ki), pontszerű töltések esetén az alábbi összefüggés írja le:

$$F = \frac{1}{4\pi\epsilon_0} \cdot \frac{Q_1 Q_2}{r^2}$$

ahol **F** az erő *newtonban*, **Q<sub>1</sub>** és **Q<sub>2</sub>** a két pontszerű töltés *coulombban*, **r** a távolság *méterben*,  $\epsilon_0$  pedig a két töltés közötti teret kitöltő anyagra jellemző állandó, a permittivitás, melynek dimenziója:  $m^{-3} \cdot kg^{-1} \cdot s^4 \cdot A^2$ . Ez vákuumra (levegőre)  $8,854 \times 10^{-12}$ .

Az elektromos *feszültséget* mindig két pont között értelmezzük. Két pont közötti feszültség mérőszáma annak az energiának (munkának) felel meg, melyet akkor kell kifejtenuünk, ha egységnyi (1 *coulombnyi*) töltést viszünk át az egyik pontról a másikra, vagy ami akkor szaba-

dul fel, amikor egységnyi töltés „megy” át egyik pontból a másikba.

Valószínűleg minden olvasónk látott már *feszültségmérőt* is. Sőt, tudja, hogy igen gyakran kombinált, feszültség és áram mérésére egyaránt alkalmas univerzális műszerek az általánosak a kisebb mérési pontosság-osztályokban, az 1 százalékos mérési pontosság alatt.

Az áramméréssel kapcsolatos kijelentésünkhöz hasonlóan itt is elmondjuk: két pont közötti elektromos feszültség annyi, amennyit e két pontra kötött, *hitelesített műszerünk* mutat.

### Kondenzátorok, kapacitás

Előző, a Magazin 1989/4. számában megjelent cikkünkben írtuk: mai digitális integrált áramkörök lényegüket tekintve sok tízezer, százezer, gyakran már akár több millió célszerűen összekötött mikroszkopikus méretű elektrontartályból (*kondenzátorból*) és vezérelhető kapcsolóból (tranzisztorból) állnak. Mik is ezek a kondenzátorok?

Egyszerűsítve: ha két vezetőből álló felületet úgy helyezünk el egymáshoz képest, hogy egyenlő nagyságú, de ellentétes előjelű töltést adva rájuk a közöttük lévő téren (a kondenzátor belsején) kívül a töltések vonzó-taszító hatása gyakorlatilag nem érzékelhető, akkor ezt az elrendezést *kondenzátornak* vagy *sűrítőnek* nevezhetjük. Ilyen elrendezésű, egymástól szigetelőlével — ami levegő is lehet — elválasztott két vezető (rendszerint fém) felületből álló kondenzátor két felületén így a töltés mennyisége abszolút mértékben azonos, de ellenkező előjelű.

Legyen az egyik elektródán **+Q** töltés, a másikon pedig **-Q**. Legyen a feszültség a két elektróda között **U**. Ha a kondenzátor mindkét elektródáján megnöveljük a töltést mondjuk kétszeresére, akkor a két felület közötti feszültség is kétszeresére nő. Általában **n**-szeresére növelve a töltést, a feszültség is **n**-szeresére nő. A kondenzátorra felvitt töltés és a kondenzátor két pólusa között mérhető feszültség *hányadosa állandó* és csak a két vezető felület geometriai méreteitől, elrendezésétől és a két vezető közötti teret kitöltő szigetelőanyag tulajdonságaitól függ. Ez az állandó a szobán forgó elektródaelrendezés *kapacitása*:

$$C = \frac{Q}{U}$$

Ezt a  $\left[ \frac{\text{coulomb}}{\text{volt}} \right]$  dimenziót Faraday tiszteletére *faradnak (F)* nevezzük. *1 farad kapacitása annak a kondenzátornak van, melyet 1 coulomb töltés 1 volt feszültségre tölt fel.*

Ha kiszámítanánk, mekkora kapacitása van két egymástól 1 mm távolságra elhelyezett, 1 m<sup>2</sup> felületű (tehát meglehetősen nagy) lemezpárból készített kondenzátornak, amelyek között levegő van, akkor nagyon kis számot kapnánk:  $8,86 \times 10^{-9}$  *faradot*. Ezért a gyakorlatban a mérnökök az egységnyi kapacitás tört részeivel, mikro-

faradokkal és pikofaradokkal számolnak:  $1 \mu F = 10^{-6} F$  és  $1 pF = 10^{-12} F$ .

Egy kondenzátorban tárolt energiát a kapacitás és a feszültség ismeretében így számítjuk ki:

$$\text{energia} = \text{töltés} \times \text{feszültség}$$

Említettük, hogy ha ma egy integrált áramkörön kialakított mikroszkopikus méretű kondenzátort legalább 1,5 volt feszültségre feltöltötenek, legalább mintegy ezer darab elektronnyi töltéssel, akkor ehhez

$$W = Q \times U = 10^3 \times 1,6 \times 10^{-19} \times 1,5$$

$$\text{joule} = 2,4 \times 10^{-16} \text{ joule}$$

energia szükséges. Ha egy digitális integrált áramkör mondjuk egymillió ilyen kicsi kondenzátort tartalmaz és ezek mindegyikét egymilliószor feltöltjük és úgy kisütjük másodpercenként, hogy kisüléskor az energia elvész (a kondenzátor környezetében hővé alakul), akkor egy ilyen integrált áramkörben

$$2,4 \times 10^{-16} \times 10^6 \times 10^6 = 2,4 \times 10^{-4} \text{ watt}$$

teljesítmény alakul át hővé. A gyakorlatban — mint előző cikkünkben írtuk — az egyéb veszteségek miatt ma még legalább ennek ezerszerese, azaz legalább tízed watt nagyságrendű villamos teljesítmény alakul át hővé.

### Energiafogyasztás

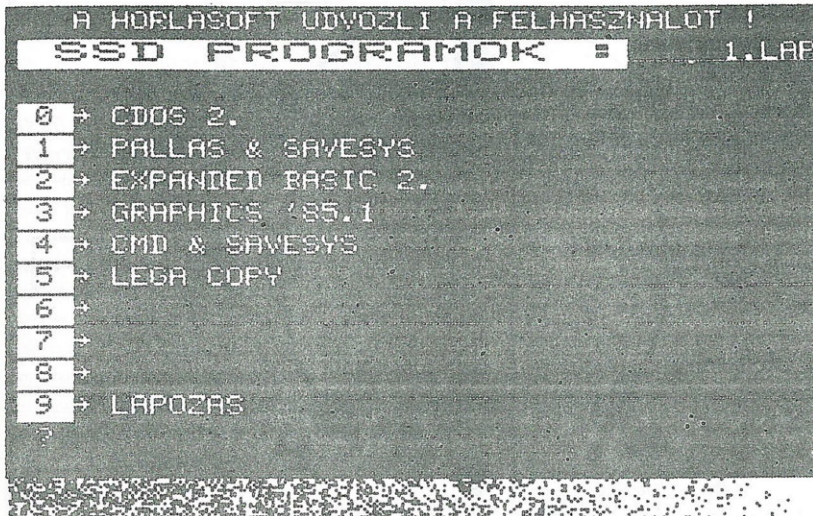
Mivel egy-egy professzionális számítógép igen sok ilyen és ehhez hasonló integrált áramkörből épülhet fel, ezek a hővé átalakuló tízed wattok összeadódva elérhetik akár a száz watt nagyságrendű teljesítményt is. (Próbált már valaki egy szokásos, 100 W teljesítményű izzót néhány perces bekapcsolás után megfogni?) Ezt a hőt a számítógépekből el kell vezetni, nehogy a hőre érzékeny áramkörök működésképtelenné váljanak.

Ez a *hőelvezetés* nem is mindig könnyű feladat, a tervezők hajlamosak elfeledkezni róla. „Nyers erővel” persze, például erős, de zajos ventilátor beépítésével sok mindent meg lehet oldani. De ki szeret egész nap zajos készülék mellett ülni?

Vita folyik azon, elvileg lehetne-e olyan számítógépeket tervezni, melyek *egyáltalán nem fogyasztanak energiát*, illetőleg, amelyekből az energiát a feldolgozás után, a folyamatok visszafordításával vissza lehetne nyerni. (Lásd Charles H. Bennet és Rolf Landauer cikkét, valamint Csákány Antal észrevételeit a Tudomány 1986. évi Számítógép-szoftver különszámában.) Erős érv amellett, hogy nem lehet: az információfeldolgozás *növeli* a körülöttünk lévő világ *rendezettségét*, és ez szükségszerűen *entrópiacsökkenéssel* és emiatt szükségszerűen *energiafelhasználással* jár.

Ennek a vitának ma nincs (még?) gyakorlati jelentősége, mert a mai számítógépekben elvesző energia — mondjuk így — „technikai tökéletlenségeink” miatt sok nagyságrenddel nagyobb annál, amiről elméletileg vita folyhat.

# Az RTST margójára



RTST  
menükép

A Primo él és élni akar. Újabb és újabb hardver- és szoftverfejlesztések készülnek hozzá. Ezek egyike az RTST (Rögzített Tartalmú Soros Tároló, lásd a Magazin 1989/4. számában), amely a maga nemében páratlan módon megkönnyíti és gyorsítja a primósok alkotó munkáját. Az alábbiakban mint a megvalósítás egy lehetséges, működő formáját, szeretném tapasztalataimat és saját RTST-met ismertetni.

A hardver elkészítésében különösebb probléma nincs, ha túlvagyunk a NYÁK-laprajzolás és -készítés nehézségein. Ha ez gondot okozna, a 6+1 IC még akár „fejlesztő” NYÁK-on is összehuzalozható. A pár darab alkatrészt az elektronikai kereskedésekben olcsón megvásárolhatjuk, az egyetlen nagyobb kiadás az EPROM. Alapos mérlegelés után a 27256-os, 32 kbájtos eszköz mellett döntöttem, mert kapacitáshoz viszonyítva legkedvezőbb az ára — 5–800 forint körül —, és már egyetlen darabnak a tárolóképesége is elegendő a legfontosabb programok elhelyezéséhez.

Az RTST EPROM-ba bármilyen programot beírhatunk, de néhány szempontra ügyelni kell. Először is az elhelyezendő programból célszerű eltávolítani a bejelentkező screent és az autostartot: a záróblokkot 177;BIH-ra kell cserélni. Ha a program távol eső címekre betöltődő, önálló bájtokat is tartalmaz, akkor ezeket a főprogramot követő címekre át kell írni, majd LDIR segédrutinok beillesztésével gondoskodjunk arról, hogy elindulás után eredeti helyükre kerüljenek. Lokalizálni kell továbbá azokat a bájtmekanismusokat — ha vannak —, amelyek egyébként meggátolják a kimentett program normális elindulását és/vagy futását. Végül nagyon pontosan meg kell határozni a kész program kezdőcímét, végcímét és indítási címét. Ezeket az operációkat azonban csak a Z80 gépi kódjában otthonos programozóknak tanácsolom.

Az RTST menüprogramját áttanulmányozva beláttam, hogy a menüszoveg részé-

re rendelkezésre álló 80 bájtnyi hely nem lesz elég a tíz programnév igényes megjelenítéséhez. Legcélszerűbbnek látszott — és ezt a gyakorlat később igazolta — a menüprogram méretét 512 bájtra növelni. Így még néhány további bővítést is el tudtam helyezni. A hosszabb menüprogram miatt azonban módosítani kellett a felléptető rutin elejét és a behúzóprogramot is úgy, hogy a menü betöltése a 65024;FE00H címre induljon.

A behúzó töltőrutinja előtt egy CALL 01C9H (képernyőtörölő) utasítást kell elhelyezni. A menüszoveget egyéni ízlésünknek megfelelően alakíthatjuk, a paramétertáblázatban azonban valamennyi cím megadásakor a legnagyobb pontossággal kell eljárunk, hiszen a 32768-ból egyetlen bájtt elszámolása is biztos kudarcra ítéli munkánkat!

Az ábrán bemutatom a magam készíttette RTST menü képét. Az esztétikai megoldáson kívül az sem mellékes, hogy a legfontosabb közkezen forgó programokat sikerült egy 32 k-s EPROM-ban elhelyezni.

A gyakorlati tesztek, nyüzögőpróbák során hamar kiderült, hogy a 16679;4127H RAM-területre betöltődő behúzóprogram gyakran felülíródik, mivel például a WRA 2 szoftver akkumulátort sok ROM-rutin használja adatok átmeneti tárolására. Újra és újra magnóról betölteni azonban nagyon kényelmetlen, hosszabb távon nem járható út. A Primo ROM-tokjait megvizsgálva úgy

találtam, hogy a 12110–12266;2F4EH–2F4EAH címek között 157 bájtt felhasználatlan terület rejlik, ahol kényelmesen elhelyezhető a behúzóprogram. A beírás természetesen kifogástalanul megvalósítható a 1659;067BH címtől kezdődő ROM 1 rész átégetésével is, de az általam ajánlott terület a ROM 3-as tokban eredetileg is üres, így a gyári rendszerprogram érintetlenül megmaradhat. A tok kiforrasztása azonban ebben az esetben sem hagyható el, cserébe viszont végleges, stabil és mindig kéznél lévő megoldást kapunk, megszabadulva a kazettás magnó gyötreimétől. Bármelyik variációt választjuk is, ügyeljünk arra, hogy a menüprogram végén lévő JP utasítás operandusa annak megfelelően legyen definiálva!

Kezdetben sokszor elfeledkeztem arról, hogy a behúzórutin meghívása előtt meg kell nyomni az RTST RESET gombot. Ilyenkor — mivel a számlálók nem nulláról indultak — mindenféle összevisszaság töltődött a képernyőre, és percekig tartott, míg rájöttem, hogy nem hardverhibával állok szemben, hanem csupán figyelmetlen voltam. Ugyanezt a jelenséget produkálja ugyanis az EPROM bizonytalan érintkezése a foglalatban. Ne felejtjük el tehát minden újraindítás előtt a számlálókat nullázni!

Az RTST előnyeit csak felsorolom, bizonygatnom nem kell: óriási tárolókapacitás, gyors, hibamentes működés, olcsóság, a programok könnyű elérhetősége a tárbán. Talán nem érdektelen, ha közreadom 32 kbájtt betöltési idejét három különböző perifériáról, 3,5 MHz órajellel mérve:

kazettás magnó: 290 másodperc,  
hajlékonylemez: 83 másodperc,  
RTST: 7 másodperc.

Az eredmények önmagukért beszélnek. Következő lépésként, még egy 27256-os tok felhasználásával, folyamatban van a 64 kbájtos változat készítése.

Befejezésül megemlítem, hogy a szerkesztőségen keresztül készséggel állok az RTST-t építők rendelkezésére tanácsokkal, a módosítások részleteivel, ROM-EPROM égetésekkel, ha problémájuk adódna. A továbbfejlesztéshez pedig szívesen fogadok minden ötletet, javaslatot.

Horváth László

## A TUDOMÁNSZERVEZÉSI ÉS INFORMATIKAI INTÉZET

előzetes megbeszélés szerint díjmentes programbemutatót tart (vidéken is) az általa forgalmazott oktatóprogramokból.

Horváth Zsuzsa 665-011/2663 mellék  
vagy 813-197

Budapest, Pf. 454, 1372

Minden kedden 17-től 20 óráig  
HCC ENTERPRISE klub  
a VSZM

Közösségi Házban  
(Bp. XI., Fehérvári út 120.)  
Klubvezető: Romvári Gábor  
Telefon: 810-950/473

# Szenzoros, gyorstüzelő botkormány

Ennek a kapcsolásnak a hagyományos botkormánnyal szemben az az előnye, hogy nincs kontakthiba, az átlós irányok pontosan kapcsolhatók, könnyű, elegáns a kezelése, és szinte korlátlan élettartamú lehet. A szenzor szerepét bármilyen fémtárgy betöltheti, szigetelőalakra szerelve. Célszerű díszcsavarokkal elkészíteni, az 1. ábra szerint. Ha a vezetéke 10-15 centiméternél hosszabb, akkor árnyékolni kell.

Ujjunkat a szenzorhoz érintve, a testünk által felvett hálózati bűgőfeszültség a kapcsolás bemenetére jut. A FET felerősíti, a germániumdióda és a kondenzátor szűri. Ekkor a tranzisztor emitterén magas szint lesz, tehát az analóg kapcsoló zár, és alacsonyra állítja a botkormány kivezetését. Ha két szomszédos szenzort érintünk meg egyszerre, akkor két analóg kapcsoló zár, így az átlós irány érvényesül.

A tűzgombot célszerű lábkapcsolóként megépíteni. Az 555-ös IC aszimmetrikus kitérésű tényezőjű — hosszú alacsony, rövid magas — négyszögjelet állít elő. Így nem kell erőltetni az ujjunkat: elég benyomni a tűzgombot, és máris löhetünk egy 1 Mohmos potival beállított sebességű sorozatot.

Praktikus kiegészítő a kapcsoláshoz a 3. ábrán látható hangjelző egység, amely minden beavatkozáskor rövid pityegő hangot ad. A 2. és 3. ábrán nyilakkal megjelölt pontokat kell összekötni. A 100 kohmos potival a hangmagasságot, az 1 kohmossal a hangerőt változtathatjuk. Ügyelni kell arra, hogy a maximális áramfelvétel 100 mA alatt legyen. A CD 4066-os IC-t tegyük foglalatba, és kössünk egy 47 nF-os kondenzátort a 14. és 7. lába közé.

A készülék érintésvédelmi szempontból ideális, hiszen 3,9 Mohmos soros ellenállás kapcsolódik az érlethető részekhez.

A kapcsolás C64-re készült, de megfelelő átalakítással más gépekhez is használható.

Végül egy rövid programmal kipróbálhatjuk botkormányunkat.

Az 1. Control portba dugva:

10 PRINT PEEK(56321)  
20 GOTO 10

Ekkor a következő bajtértékek adódnak:

É: 254, K: 247, D: 253, NY: 251  
ÉK: 246, DK: 245, DNY: 249, ÉNY: 250  
TÜZ: 239

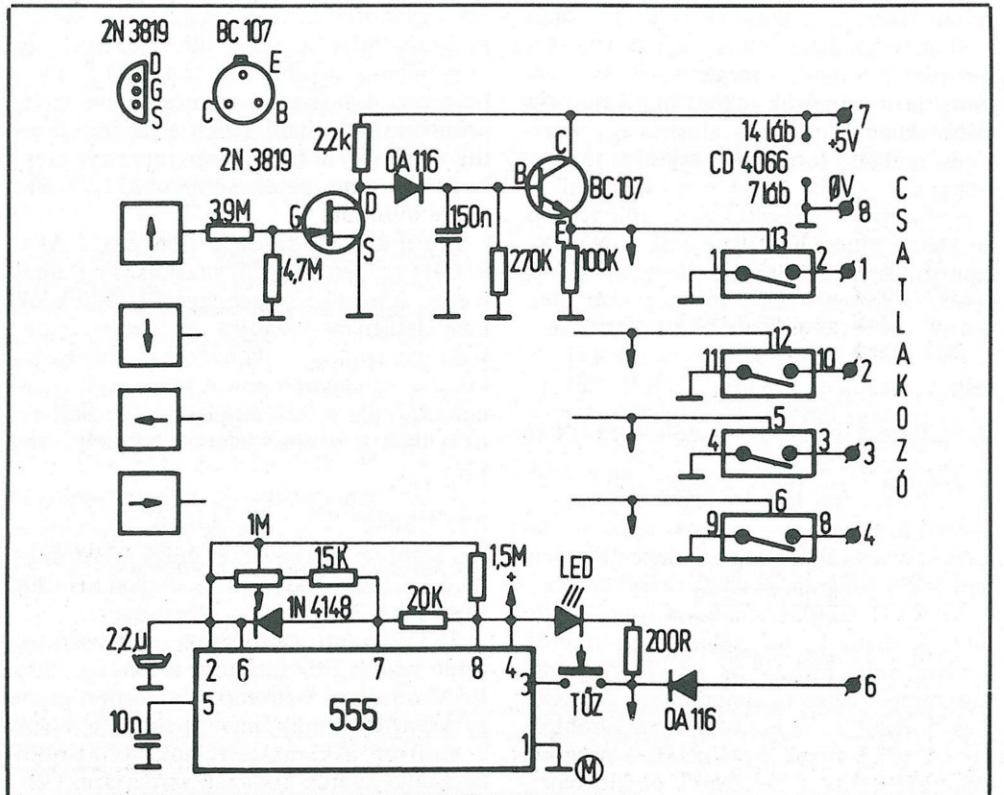
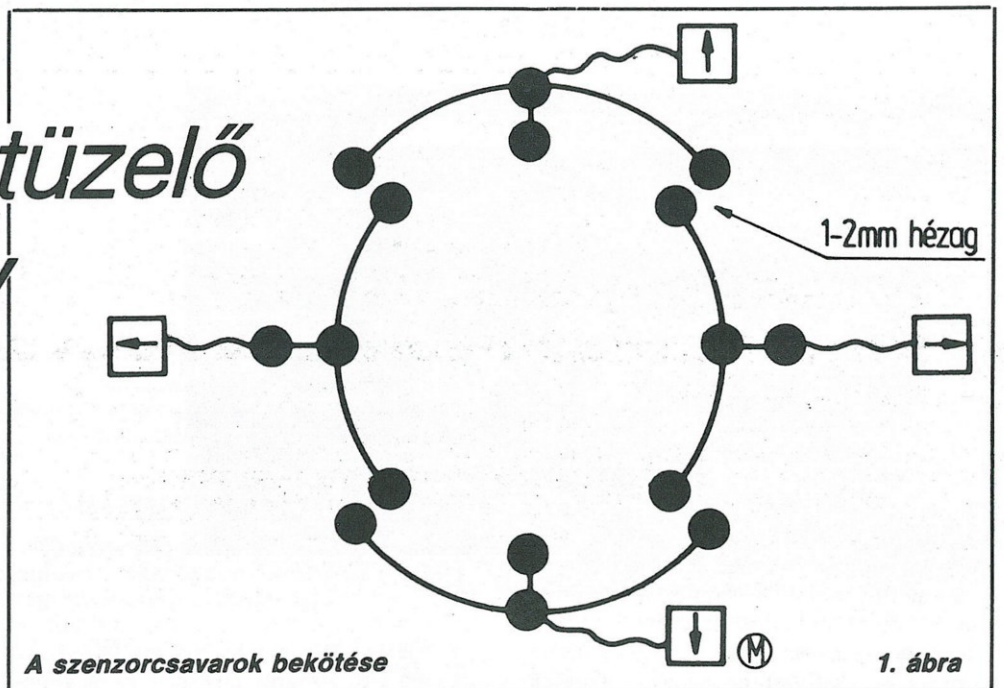
A 2. Control portba dugva:

10 PRINT PEEK(56320)  
20 GOTO 10

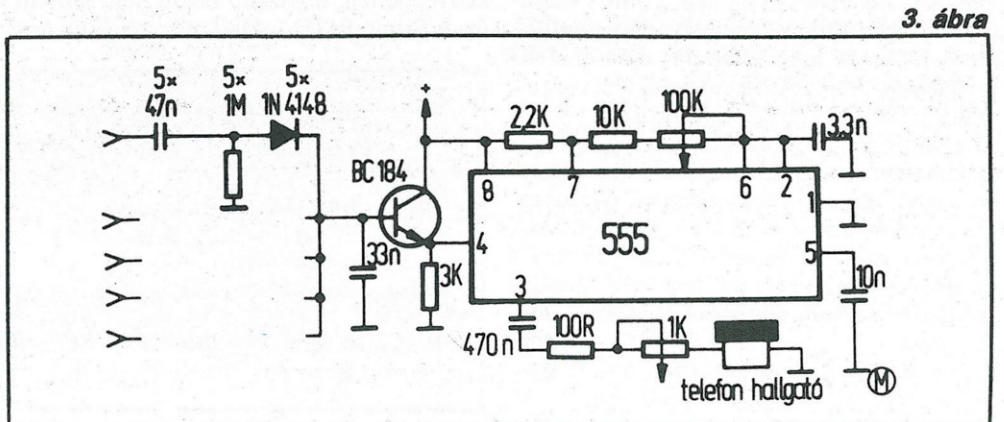
Itt a bajtértékek 128-cal kevesebbek az előzőeknél:

É: 126, K: 119, D: 125, NY: 123  
ÉK: 118, DK: 117, DNY: 121, ÉNY: 122  
TÜZ: 111.

Tóth Vilmos



2. ábra



3. ábra

# Kérdés a lemezhez: FORMATTED?

Aki sokszor gyötörte már lemezeit, jól tudja, mit jelent az ID MISMATCH ERROR. Egyes sávok másolása, félbehagyott és elfelejtett teljes lemezes másolások — megannyi lehetőség zűrös lemezek létrehozására. És ne felejtjük ki a D.B. COPY-t sem, amellyel elérhető, hogy a 37-es sáv a normál lemezterülettel eltérő azonosítót viseljen. Ilyenkor jó szolgálatot tehet az alábbi program, amely végigellenőrzi az egész lemezt, és minden egyes sávról megállapítja, hogy milyen kóddal formázták. A programmal felderíthető a programvédelemnek az a módja is, amikor sávonként eltérő ID kóddal védik a lemezt másolás és beavatatlankok által végzett módosítgatások ellen.

Magyarázatként tekintünk át röviden, mit is jelent a lemezvezérlőnek átadott kód, eredeti nevén: JOB.

Ez a kód átadható mind az öt puffer vezérlőrekeszébe. A vezérlőrekeszek a lemezes egység 0 ... 4-es címein található.

## A pufferek vezérlőrekeszében tárolt parancsok

JOB	JOB	Végrehajtandó művelet
< 128	< \$80	nincs művelet
128	\$80	blokkolvasás
144	\$90	blokkírás
176	\$B0	header olvasása
208	\$D0	programfuttatás a pufferben
224	\$E0	format

## A pufferek vezérlőrekeszében tárolt hibüzenetek

LDA # \$ JSR \$60A eredménye

1	00 OK
3	21 NO SYNC FOUND
6	24 READ ERROR
8	26 WRITE PROTECT ON
B	29 ID MISMATCH ERROR
F	74 DRIVE NOT READY

Dr. Zana János

### A FORMATTED? című Program listája

```

MD 5 OPEN15,8,15,"I"
SJ 6 INPUT#15,EN$,EM$,ET$,ES$
XR 7 IFEN$="00" THEN10
MM 8 STOP
SJ 10 DINT(41),H$(41),L$(41)
FJ 130
PRINT"IDOWN]INSERT[SPC]IRVSON]DISK[RVSOFF
]ISPC]IN[SPC]IDRIVE[DOWN]"
QP 140 GOSUB910 (Várakozás
billentyűre)

FJ 160 PRINT:S=1:D1=65:D2=65
(Az egyes szektort vizsgálja)
SJ 170 FORI=0 TO41
KH 180 T(I)=1 ( T(I)segitésével
vezérelhető, hogy melyik tracket olvassa)
RC 190 NEXTI
MG 200 JOB=176 (Floppy vezérlő
kód: HEADER olvasása)
SP 202 FOR T=1 TO0 STEP -1: IFT=1
THENI=0
SX 204 GOSUB270 (A nullás
track olvasása)
BQ 206 NEXTT: PRINT#15,"I" (A floppy
vezérlő visszaállítása)
EB 210 FORT=1 TO41
GS 220 GOSUB270 (Főprogram: a
lemez vizsgálata a 41-es trackig)
AK 230 NEXTT
MM 235 IFD1<32 ORD2<32 THEN
PRINT"IDOWN]FORMATTED[SPC]CHR$(D1);"[SP
C] CHR$(D2);"[ID]SPC]CODE"
(A Program végén
még egyszer kiírja az ID értékét)
SP 240 PRINT#15,"I":
PRINT"IDOWN]ONE[SPC]MORE[SPC]DISK?[SPC]<
/N)"
(Iniciál, hogy a fej
visszatérjen a 18-as track-re)
RQ 250 GETA$: IFA$="" THEN250
HJ 255 IFA$="N" THEN CLOSE15: END
DC 260 GOTO130

S z u b r u t i n
SB 270 TA=1+(T>9) (Tabulátor)
XM 280 IF T(T)=0 THEN460 (Itt lépne
ki, ha valamilyik tracket nem kérnénk)
SG 290 GOSUB970: IF NOTI THENI=-1:
GOTO460
(Megakadályozza, hogy kétszer
olvassuk az egyes tracket)
AA 295 IF T<>B THEN330
(Nem fogadja el a lemeztől beolvasott
téves track-számot, például, ha a
rosszul pozícionált fej a szomszéd track
számát olvasná)
QE 300 IF E=1 THEN350
(A lemezvezérlőben maradt kód
hibátlan olvasást jelent)
HR 310 H$(T)=CHR$(0)
PX 320 L$(T)=CHR$(0)
PH 330 PRINT
    
```

```

TAB(TA);T;"TRACK[SPC]IS[SPC]NOT[SPC]FORMA
TTED"
DM 340 GOTO460
KB 350 PRINT#15,"M-R" CHR$(22) CHR$(0)
(Az ID-kód első betűje)
AJ 360 GET#15,A$
XC 370 IFA$="" THENA$=CHR$(0)
FC 380 IF ASC(A$)>31 THEN400
FA 385 D1=ASC(A$)
FH 390 A=ASC(A$)+64:A$=CHR$(18)+
CHR$(A)
(Inverz betűt írunk, ha az ID-kód
nem nyomtatható karakter)
JE 400 H$(T)=A$
XQ 410 PRINT#15,"M-R" CHR$(23) CHR$(0)
(Az ID-kód második betűje)
HR 420 GET#15,A$
QH 422 IF A$="" THENA$=CHR$(0)
KP 424 IF ASC(A$)>31 THEN430
HC 425 D2=ASC(A$)
DK 426 A=ASC(A$)+64:A$=CHR$(18)+
CHR$(A) (Inverz betű)
EH 430 L$(T)=A$
QC 440 PRINT
TAB(TA);T;"TRACK[SPC]FORMATTED[SPC]WITH[SP
C]";H$(T);L$(T);
CHR$(146);"[SPC]ID[SPC]CODE"
(A CHR$(146) megszünteti az
esetleges inverz írásképet)
AJ 450 PRINT"UP]"
KD 460 RETURN
SS 890 END
JH 910
PRINT"IDOWN]PRESS[SPC]IRVSON]RETURN[RVSOFF
]ISPC]TO[SPC]CONTINUE"
XD 920 GETC$: IFC$="" THEN920
CD 930 RETURN
JOB végrehajtása a floppy memóriájában
BC 970 TRY=0 (A
Próbálkozások számának kezdőértéke)
XR 980 PRINT#15,"M-W" CHR$(8) CHR$(0)
CHR$(2) CHR$(T) CHR$(S)
GF 990 PRINT#15,"M-W" CHR$(1) CHR$(0)
CHR$(1) CHR$(JOB) (P a r a n c s)
AD 1000 TRY=TRY+1
AM 1010 PRINT#15,"M-R"; CHR$(1) CHR$(0)
AB 1020 GET#15,E$
(A Parancs végrehajtása után
visszamaradt kód kiolvasása)
XA 1030 IFE$="" THENE$=CHR$(0)
RR 1040 E=ASC(E$)
GQ 1050 IF TRY=500 THEN1070
FB 1060 IFE>127 THEN1000 (Még nem
fejeződött be a végrehajtás)
AC 1070 PRINT#15,"M-R"; CHR$(24) CHR$(0)
CHR$(1): GET#15,B$
(A lemezen levő headerből kiolvasuk
a track tárolt számát)
JA 1080 IF B$="" THENB$=CHR$(0)
QQ 1090 E=ASC(B$)
AP 1170 RETURN
    
```

## Ki ad magyarázatot?

Programjaimat általában a HELP támogatásával írom és működtetem. Egyik programom több szekenciális fájlt olvasott és írt. Működése során a gép egyszerre lemerevedett, és ebből az állapotból csak a RUN/STOP és RESTORE billentyűk egyidejű lenyomásával lehetett kihozni.

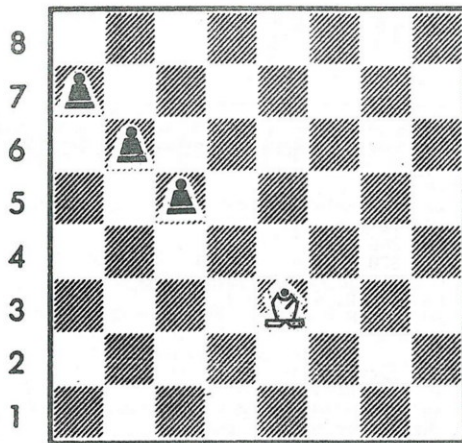
Amikor a hiba többször is előfordult, a segédprogram HELP utasításához fordultam (ez a # H). Megtudtam, hogy a gép egy INPUT utasítást nem tu-

dott végrehajtani; a feltöltendő változó szöveges típusú volt. Megpróbáltam kiírni az ASCII kódját. Az ASC függvényre INVALID QUANTITY ERROR üzenetet kaptam. Végül kiderült, hogy a változó hossza 0. Természetesen ellenőriztem: a keresett állomány létezett a lemezen, terjedelme 1 blokk volt.

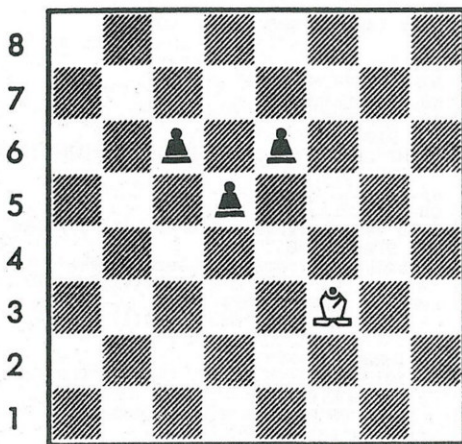
Vajon hogyan fordulhatott elő, hogy létező állományból képtelen volt a program adatot beolvasni?

Dr. Zana János

# A figuratípusok jutalompontjai



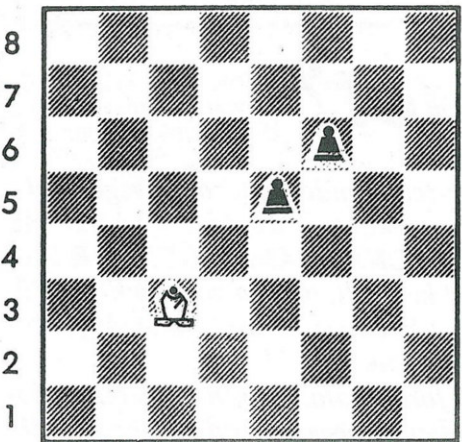
1/a ábra Büntetőpont: -25



1/b ábra Büntetőpont: -15



1/c ábra Büntetőpont: -10



1/d ábra Büntetőpont: -5

Az előző alkalommal bemutatuk, hogy a Northwestern Egyetemen David Slate, Larry Atkin és Keit Gorlan által kifejlesztett Chess 4.5 sakkprogram hogyan számítja a jutalompont értékét a vezér esetében. Most a futó bónuszértékével foglalkozunk.

A futó jutalompontjának kiszámításához használt függvény bonyolultabb, mint a vezéré. Ennek több oka van. Ez az egyetlen olyan figuratípus a táblán, amelynél a közép- és a végjátékban a királysárnyai és a vezérsárnyai báb nem helyettesíthető egymással, ugyanis az egyik csak világos, a másik csak sötét mezőre léphet. Ebből a speciális helyzetből adódóan tehát az egyik futó csak a világos mezőkön, a másik csak a sötét mezőkön lévő ellenséges figurákat támadhatja. Ezért ha az egyik futót már leütötték, akkor az egyensúly könnyen felborulhat.

Az értékelőfüggvénynek gondoskodnia kell az egyensúly megtartásáról. Ha az egyik félnek hiányzik már az egyik futója, akkor célszerű a gyalogjait olyan színű mezőkön elhelyezni, amelyeken a leütött futó mozgott. Ezzel egyrészt kompenzálja a leütött futó által létrejött mezők gyengeségét, másrészt a táblán lévő futónak nagyobb mozgásteret ad. Így a gyalogstruktúra és a futók értékelése szorosan összefügg egymással.

A futók bónuszát a Chess 4.5 a következő képlet alapján számította:

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

ahol  
P: a futó pszeudo-legális lépéseinek száma

EM: a futóátlón lévő ellenséges figurák anyagi értékének összege a következők szerint:

$$\text{gyalog} = 0 \text{ huszár} = 3 \text{ futó} = 3 \\ \text{bástya} = 5 \text{ vezér} = 9 \text{ király} = 10$$

OM: a futóátlón lévő saját figurák anyagi értékének összege a következők szerint:

$$\text{gyalog} = 0 \text{ huszár} = 3 \text{ futó} = 3 \\ \text{bástya} = 5 \text{ huszár} = 3 \text{ király} = 0$$

D: annak az átlónak a hosszánál eggyel kevesebb, amelyiken a futó áll. Tehát, ha a futó c3-on áll, akkor 7

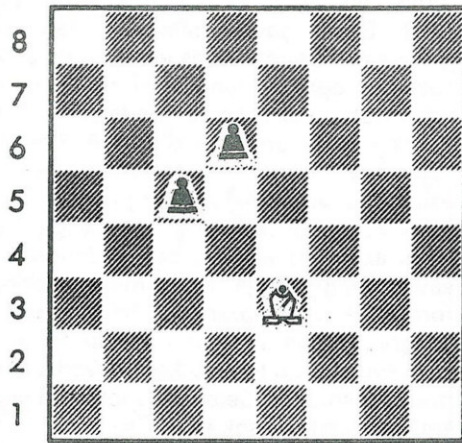
OB: gyalogakadály-értékelés az 1. ábra alapján. Az értékek: -25, -15, -10, -5, +1 a specifikus gyalogszerkezetnek megfelelően

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.

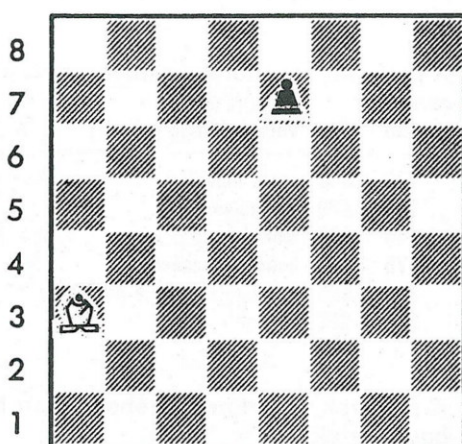
Az előbbi képlet alapján a 2. ábrán látható hadállás esetében, amely Kaszparovnak Mihajlsicsin elleni 1981-es játszmájában jött létre, a 31. féllépésben a világos és a sötét futókra a következő érték adódik:

$$Fb(\text{világos}) = 14, Fb(\text{sötét}) = 30.$$

Kovács P. Attila

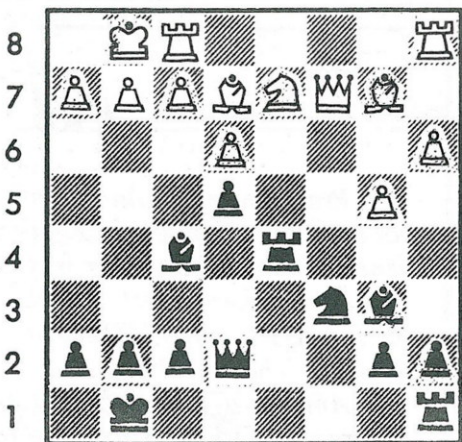


1/d ábra Büntetőpont: -5



1/e ábra Jutalompont: +1

## 2. ábra. Mihajlsicsin—Kaszparov. Szovjet bajnokság, 1981. 31. féllépés



2. ábra Jutalompont: +1

# PROGRAMTERMÉK

## VEGYEM?

## Gombászat számítógéppel

Vajon hányan szenvednek évente gombamérgezést? Gondoltam, telefonálok valahová, hátha tudnak mondani valami adatot, de a posta nem fogadott kegyeibe, sorra meghíúsultak a hívásaim. Akárhányan vannak is azonban az áldozatok, aki „hozzájut” egy ilyen élményhez és túléli, az bizony soha nem felejt el. Ezért fogadtam örömmel, amikor a Novotrade-ben egy gombászati ismeretterjesztő programot kezdtek nekem lelkesen ajánlgatni bevizsgálásra.

Amint korábban egy cikkemben az oktatóprogramok generációival kapcsolatban már fejtegettem, a Novotrade termésében elég sok olyan nagy fajsúlyú programcsomag van, amely harmadik generációsnak tekinthető: a meglévő oktatási tananyagok támogatására épül. Itt futólag megemlítem az „A nyuszi olvasni tanít” című sikeres programcsomagot, amelynek már talán nem is kell reklám, de újszerű, a kisgyerek lélektanához igen jól igazodó megoldásai miatt feltétlenül említésre érdemes. És nemcsak a sokat szidott szóképoltvasásra tanít, hanem a szótagolást is célba veszi, amit én személy szerint pártolok.

A Novotrade régebbi termékeiben szórványosan, újabb termékeiben erőteljesen jelentkezik a negyedik generációs jellemzők: az elsajátítandó ismeretek feldolgozását úgy tervezik meg, hogy eleve feltételezik a számítógép használatát, tehát a programok nem meglévő tananyagoknak alárendelve, azok korlátainak engedelmeskedve készülnek. Az elrugaskodást pozitívan segíti, hogy a Novotrade új stúdiója játékok készítésével is foglalkozik, ami nagy rutint ad egy téma élvezetes és művészi igényességű kibontásához.

A játékról jut eszembe, és már korábban is beszéltem róla, hogy a megfigyelések szerint a tanulási folyamat hatékonyságát növeli, ha a rögzítés során minél bonyolultabb emberi tevékenységet alkalmazunk. Mindenki tapasztalta, hogy könnyebben megjegyzi azt, amit felír valahova, mintha ugyanazt hallomásból kellene megjegyeznie. A szöveg rögzítése olyan plusztevékenység, amely erősíti a tanulást. A számítógépes játékok ugyanilyen jó hatásúak lehetnek, ha megfelelő célt támogatnak.

No, de nézzük a gombászprogramot. A program jellemzőit a szokásos összefoglaló táblázat mutatja.

A programcsomag kezelése egyszerű. A kazetta betöltése elég hosszú időt vesz igénybe a turbóbetöltő ellenére is. Eleve 60 perces kazettán van, több modult is tartalmaz, mert az év különböző hónapjaiban indulhatunk gombászatra. Ezért jó előre tájékozódni, hogy mikor is terem az a csodálatos őzlábgomba, amelynek megszerzéséért igen alaposan meg kell küzdeni erdei csatangolásunk során. A játék eredeti dokumentációja (a kazettaborítón) ugyan igen lakonikus, de a legfontosabb kiderül belőle: az egész program a nagy gombászkönyvön alapul. Így abban a családban, amelyik korábban nem szerezte be ezt a művet, emiatt külön kiadásra is kell számítani, ha igazán meg akarják ismerni a gombákat.

A szerzők azt tűzték ki célul, hogy játszva tanítsanak, amit a fenti gondolatok alapján csak üdvözölni lehet. A játék bekapcsolása a tanulási folyamatba ennél a témánál több, mint jó ötlet. Olyan erős késztetéssel serkenti a játékost, hogy kedve támad mindjárt a hétvégén kimenni az erdőbe, kipróbálni frissen szerzett tudását. A program szerzői így elérik céljukat: sikerül igen széles kört lázba hozni és megismertetni a gombákkal. Széles kört, mert ha a gyerek játszik, akkor hamarosan segítségért fordul a szülőhöz, és nemsokára az sem tud mekkülni a témától.

A program cselekménye meglepő fordulatokat rejteget. Eredetileg arról volt szó a borítón található szövegben, hogy az erdőben csatangolva minél több gombát kell gyűjteni, és el kell jutni a nagy őzlábgombához, amelynek megszerzése igen nagy teljesítmény. Hogy a szöveg utóbbi részét komolyan kell venni, az akkor derül ki, amikor egyszer csak mások is megjelennek az erdőben. Ilyenkor sokszor a szerencse segít, hogy ne haljunk meg a kosarunkba gyűjtött mérges gombáktól. Fontos felismerni ezeket, de meglepő módon az is hasznunkra válhat, ha tartunk magunknál egyet-kettőt belőlük. Az erőnkön kívül pénzzel is kell gazdálkodni (a szellemes ne-

vű „duplával”), ami a gombákkal kapcsolatos árusítási szabályok változásában is segít: például megjelenik egy turista, aki gombát kér, majd kiderül, hogy detektív. A program tervezői tehát igen sok jó ötletet beledolgoztak a játékba.

A szerzők a képi megjelenítésre legalább annyira ügyeltek, mint a cselekményre. Erdőbeli bolyongásunk animációja a C Plus/4 adta lehetőségeket maximálisan kihasználja, bár időnként zavaró, hogy túl sokat rohan a kép egyetlen mozdulatunkra. Az ikon rendszerű menüvezérlés ma teljesen korszerűnek mondható. Ez utóbbi, a néhány képernyőoldali útmutatóval együtt további olyan elem a szoftverben, amelynek fényében azt mondhatjuk, öndokumentált programról van szó. A képi megjelenítést emeli a madárcsiripelést utánzó, randomizált hangeffektus. (A családtagjainknak számítógépes papagájok teljesen lázba jöttek tőle.)

A tartalmi anyag megítélése a legnehezebb kérdés. A rendelkezésre álló rövid idő alatt elég keveset használtunk ki a játék lehetőségeiből, mert — a kívánt szakirodalom birtokában lévén — rögtön kipécéztük a legjobb hónapot és időjárási körülményeket. Persze a többi hónapban termék gomba is érdekes, nemcsak az a csodálatos őzlábgomba. Aki belelendül, az biztosan nem hagyja abba a cél megtalálása után sem a játékot, sőt, inkább mondhatjuk, a tanulást. A nagy kazettányi méret azt mutatja, hogy a teljes nagy gombászkönyvet talán nem lehetett belepasszírozni a C Plus/4-be, de a program bizonyosan igen nagy adathalmazra épül. Mi, csavargásaink során, több tucat gombával találkoztunk. A csodálatos nagy őzlábgombával is, csak még meghaltunk, mielőtt hazaérhettünk volna vele. Ilyen az élet — a gombák birodalmában.

A program az otthoni önképzésen kívül hasznos, értelmes játékként segíti az iskolai szabadidős programok színesítését vagy a kultúrintézmények klubjainak szolgáltatásait. Jó lenne, ha e program révén is csökkenne a haszontalan játékokkal elfecsérelt idő.

Zsadányi Pál

### ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Novotrade
Terméknév:	Csavargás a gombák birodalmában
Szerző:	Greensoft
Géptípus:	C Plus/4
Hordozó:	kazetta
Dokumentáció:	egy kazettaborítányi oldal
Ár:	399,- Ft

### MINŐSÍTŐ ADATOK

Kezelhetőség:	kiváló
Teljesség:	kiváló
Dokumentálhatóság:	jó
Használhatóság:	kiváló
Ár/teljesítmény:	jó
Összbenyomás:	kiváló

**VC 1541-es lemezmeghajtó egység**  
 Felhasználói kézikönyv  
 (Budapest, 1988.  
 Novotrade Rt., 96 oldal.  
 Ára: 110,— Ft.)

A VC 1540-es és a VC 1541-es közel azonos típusú lemezegységek, így a kézikönyvben csak az utóbbi típusmegjelölést használják. Az egység mind a VC20-as, mind a C64-es alapgéphez csatlakoztatható. A VC 1541-es lemezegység jelentősen növeli a VC-rendszer teljesítőképességét. A lehetőségekkel azonban csak az tud igazán élni, aki behatóan ismeri a BASIC programozási nyelvet és a C64-es alapgépet.

Mivel a VC 1541-es intelligens, egylemezes, nagy tömegű adathalmaz tárolására alkalmas lemezegység, kezelését és programozását célszerű alaposan megismerni. A könyv ehhez nyújt segítséget.

**Pethő Ádám:**  
**IBM PC/XT felhasználóknak és programozóknak 3.**  
**A ROM BIOS és ami mögötte van**  
 (Budapest, 1988.  
 SZÁMALK, 327 oldal.  
 Ára: 240,— Ft.)

Az IBM PC/XT felhasználásáról, programozásáról szóló könyvsorozat harmadik kötete a gép és a rajta futó operációs rendszer olyan kérdéseit tárgyalja, amelyek az első két kötet olvasásánál joggal merültek fel az olvasóban. A szerző válaszol azokra a kérdésekre, hogy milyen hardver- és szoftvereszközökkel valósítják meg az MS-DOS operációs rendszert, illetve milyen áramkörök és vezérlésükhez milyen rutinok állnak a programozó rendelkezésére.

A kötetet haszonnal forgathatják azok, akik mélyebbre akarnak látni a gép működésébe, operációs rendszerébe. Hasznos lehet minden programozó, program- és rendszer-szervező, sőt felhasználó számára is, akik a rendszert alaposan kell ismerjék és tisztában kell lenniük a driverek és a hardver lehetőségeivel is. Az egyes fejezetek a különböző hardverelemeket tárgyalják. Ezután következik a hardver és közvetlen programozásának ismertetése, a ROM BIOS megfelelő részének általános áttekintése és a rutinhívások részletes leírása. Több fejezet végén példaprogram is található. A könyv utolsó két fejezetében az olvasó olyan ismereteket talál, amelyek nélkül számos feladatot nem tudna megoldani.

**Traister, Robert J.:**  
**BASIC-ből a C-be**  
 Budapest, 1988.  
 Prentice Hall — Novotrade Rt.,  
 138 oldal. Ára: 298,— Ft.)

A kötet útmutató mindazok számára, akik addig BASIC nyelven programoztak, és a kö-

zelmúltban tértek át, vagy éppen most szándékoznak áttérni a C nyelvre. Elhatározásukat több tény igazolja. A C nyelvet rugalmassága, tömörsége és végrehajtási sebessége alapján sokan a mikrogepeken ma elérhető legjobb programnyelvnek tartják. A nyelv csaknem teljesen független a gépi környezettől, és így az elkészült programok közel minden, C fordítóval ellátott számítógépen változtatás nélkül futtathatók.

Úgy tűnik, hogy a C nyelv elsajátítása éppen a BASIC programozónak okozza a legtöbb gondot. A tanulási nehézségeket a két nyelv eltérő szerkezete önmagában még nem indokolná. A magyarázat inkább abban rejlik, hogy nincs olyan szakirodalom, amely a kezdők tudásának megfelelően kapcsolatot tudna teremteni a két nyelv között.

A BASIC-ből a C-be című könyv kifejezetten a BASIC nyelvben jártas és a C-ben kezdő programozóknak készült, és éppen ezért a két nyelv közötti kapcsolat tükrében igyekszik bemutatni a C nyelv alapjait. Ha az olvasó párhuzamosan tanulmányozza a BASIC programot és annak C nyelvű változatát, könnyebben fogja áthidalni a problémamegoldás különbözőségeiből eredő tanulási nehézségeket, mint ha teljesen el kellene szakadnia a BASIC-ben megszokott logikától. A könyv felépítése az előbbieket miatt olyan, hogy azt a programozók kézikönyvként is használhatják.

## Fényűjság

Felszerelték Miskolc első fényűjságját, melyet a nép által „villanyrendőr”-nek nevezett városközponti sarkon helyeztek el. A Videoton gyártmányú számítógép által vezérelt fényűjság központja a Városi Művelődési Központban van. Mivel a készülék a kapcsolóberendezéssel 600 ezer forintba került, ezért a Művelődési Központ nem csupán a saját rendezvényeit kívánja reklámozni, hanem „bérmunkát” is vállal. Megszerkesztették a tarifát is, betűnként öt forintot kérnek, s egyszerre 20-22 betűt képesek felvillantani. A tervek szerint a fényűjság mindennap, még vasárnap is — reggel nyolctól este tíz óráig — működik majd. S bár a készülék szombat—vasárnap felügyelet nélkül is üzemelhetne, mégsem hagyták magára. Nem szeretnék, ha áramkimaradás miatt vagy műszaki hiba következtében leállna.

## Fékpofák

A Rába Vagon- és Gépgyár szombathelyi Főtömműgyárában bevezették a számítógépes fékpofa-ellenőrzést. A berendezéseket az amerikai Eaton cégtől szereztek be. A három — egymástól elkülönített — készülék a kalibrálás után, a félfurat-megmunkálás utáni és a szegecselés utáni fékpofa-összeállítás ellenőrzi. Ezek mindegyike külön mérőkészülekből és a hozzákapcsolt számítógépből áll. A képernyők működtetése és az egyes funkciók kiválasztása kézi érintéssel történik, ami a műhelykörnyezetben lényeges előny. A számítógéppel összekapcsolt készülék — egy elméletileg megadott felülethez képest — 44 pontban méri a fékpofa palástjainak eltérését. Jelzi a magasságtéréseket, a hibás pontokat külön megjelöli, a túrérszárú kilógó értékeket pedig aláhúzza.

A modern berendezések nagymértékben megkönnyítik a fékpofák vizsgálatát, ám a kezelésük, a működtetésük, a karbantartásuk egyúttal magasabb szaktudást igényel, mint a hagyományos eljárások.

## Ajándék

Rövid háziünnepség keretében cserélt gazdát egy TPA típusú számítógép, mely eddig Miskolcon az Északterv tulajdonában teljesítette feladatát. A vállalat új, nagyobb teljesítményű gépet vásárolt, s a régebbi gépükkel a város középfokú oktatását kívánják támogatni. A választás a Kossuth Gimnáziumra esett, mely még nem rendelkezett ilyen, nagyobb teljesítményű géppel. Az ajándékot Varga Zoltán, az Északterv igazgatója adta át a Kossuth Gimnázium oktatóinak, akik természetesen nagy örömmel fogadták, hiszen hozzásegíti őket ahhoz, hogy hatékonyabban tanítsák diákjaikat az elektronikus számítógépek kezelésére, alkalmazására.

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

ADOK

**Amiga 500 programcsere!** Listát kérünk! HIGH VOLTAGE. Balassagyarmat, Pf.: 118. 2661. Mindenféle magyar demo és intro érdekel!!!

**Atari 800XL** magnóval, 2 db joystickkal, 100 db programmal, dokumentációval 15 000 Ft-ért eladó. Érdeklődni lehet az alábbi címen hétköznap 15 órától. Szalajka József, Budapest, Csontváry u. 27/B. 1181

**Comodore floppy** olcsón eladó! Esetleg az egész konfigurációt megegyezéssel Spectrum+ vagy Atari 800-ra cserélem. Árajánlatot kérek: Marincsák János, Kisvárdra, Lenin u. 20. 4600

**C Plus/4, C16** (első bővítésű) géphez szuper cartridge! Újrendszerű monitor! Fejbeállító, Hypra Load-Save kazetta és disk, BASIC kompresszor, Fájlmásoló, Fájltömörítő, Disk-monitor, Nyomatógépprogram. Kiss György, Budapest, Vízior-gona u. 10. VIII/77. 1039. Tel.: 801-869 (19 óra után).

**C16, C Plus/4, C64, C128, Amiga 500** számítógépekre 1988-89-es programok eladók lemezen és kazettán. Keresztfalvi János, Budapest, Doberdó út 4. 1034. Kérésre listát küldök!

**C16-os** számítógépet magnóval, több mint száz programmal, szakirodalommal együtt eladom. Irányár 12 000 Ft. Keller Bálint, Rakamaz, Béke u. 5. 4465

**C16** (64 k), magnó, 1541/11 lemezegység, kazettákkal, lemezekkel, bő szakirodalommal külön is eladó. Együtt olcsóbb! Ár megegyezés szerint. Gyarmati Ervin, Taksony, Rózsa Ferenc u. 41. 2335  
**Comodore Plus/4** (új) + RF 501C floppy eladó. Kiszely János, Nyíregyháza, Fazekas János tér 24. III/30. 4400

**C Plus/4, C16** programokat olcsón eladok kazettán. Válaszboríték ellenében listát küldök. Török József, Borsodnádásd, Eötvös u. 5. fsz. 3. 3671

**Comodore Plus/4** (új), magyar nyelvű leírás 12 000 Ft-ért eladó. Varga Szabolcs, Budapest, Tóalmás u. 97. 1172

**C64**, floppy, magnó és 2 db joystick együtt, 1986-88 évi programok lemezzel



### Digitális papír

A számítógépes adatok egy teljesen új hordozóját fejlesztették ki Angliában, az ICI Electronics nevű cégnél. A digitális papír poliészter alapú bázisra felvitt különleges festékpolymer. A rögzítés technikája a merev optikai lemezekéhez hasonló, vagyis lézerral készített bemélyedések jelzik a bináris nullákat. Am a hajlékony alapra felvitt érzékeny réteg használata sokkal kényelmesebbé teszi ezt a technikát, és különleges sokoldalúságával megnyitja az utat a példátlan tömegű adatot hordozó hajlékonylemezek, szalagok, csíkok, jelzések készítése felé.

Egy közönséges 267 milliméteres orsón elférő 730 méter hosszúságú, 13 milliméteres szalagon 600 gigabájt adatmennyiség tárolható.

A lézeres adatbeírás tökéletesítésével ez az adatmennyiség a jövőben tovább növelhető. A felhasználó szemszögéből nézve a 600 gigabájt szalagon ezer kompaktlemez vagy több, teljes hosszúságú játékfilm anyagát tárolhatják. Rendkívüli adatsűrűsége következtében a digitális papír kiválóan alkalmas meteorológiai, geofizikai vagy más műholdas adatok, orvosi felvételek és iratok tárolására.

Az ICI Electronics szakértői jelenleg a digitális papírral működő meghajtó kifejlesztésén dolgoznak. Az egy gigabájtos rendszer bármely adatához átlagosan 28 másodperc alatt lehet majd hozzáférni. Ez a mutató azt jelzi, hogy a digitális papír a kezdeti időkben gyakorlatilag csak archiválásra szolgálhat, melyről egész állományokat célszerű visszatölteni mágneses háttértárra. Maga a feldolgozás pedig már az utóbbi alkalmazásával valósítható meg.

Az USA-beli massachusettsi ABT Books of Cambridge hordozható elektronikus könyvtárát mutattott be: egy CD-ROM lemezjátszót épített be egy hordozható számítógépbe. Egy másik kis vállalat, a virginiai Pindar Computer Corporation két hasonló hordozható számítógépet árusít, szintén beépített CD-ROM lemezjátszókkal. Clark ABT, az ABT Könyvkiadó elnöke egyelőre kutató, miképpen tehetné termékeit kisebb méretűvé és olcsóbbá. Az ABT gépe 8 kg-ot nyom, a vállalat azonban már szabadalmi bejelentést tett egy kisebb konstrukcióra is.

A CD-ROM perifériával ellátott hordozható mikrógép azért jelentős, mert a CD-ROM tárolási kapacitása 600 Mbájt, s már több száz tematikus katalógus jelent meg ezen az új adathordozón. A CD-ROM lehetővé teszi, hogy tonnyai könyveknek megfelelő információt vigyünk magunkkal diplomatautazásunkban, s a hordozható gép segítségével azt bárhol, utcán, villamoson, vonaton ülve lapozgathatjuk, átdolgozhatjuk.

## ADOK—VESZÉK — CSERÉLEK

és kazettával külön is eladók. Tel.: 62/27-530

**C64 és ZX-Spectrum** színvonalas játécképeket adok-veszek. László Ferenc, Budapest, Közraktár u. 24. 1093

**Sürgősen eladnám** új C64-esem + 1541/11 fajtájú floppyt + 42 db lemezt szuper játékokkal + GEOS-t és néhány könyvet, újságot 43 000 Ft-ért! Érdeklődni lehet: Pálmai Róbert, Mád, Kossuth Lajos u. 52. 3909

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

**C64-es** szuperszínvonalas programokat adok-veszek-cserélek, kazettán és lemezen. Horváth László, Budapest, Bajza u. 63. 1046

**Commodore 64-es** számítógépe van és szeretném bővíteni programkészletét? Nálunk nagy választékban kaphatók játék-, zene-, felhasználói-, és egyéb érdekes programok kazettán és lemezen is. Írjon nekünk és kérje ingyenes tájékoztatókat! Dukán Zoltán, Sopron, Laktanya u. 20. 9400

**Commodore 128** számítógép floppyval, monochrome monitorral, nyomtatóval, sok könyvvel és kb. 150 programmal nagyon olcsón eladó! Ajánlatokat a következő címre kérek: Tatabánya, Ifjúmunkás u. 8. 3/2. 2800

**C128D** eladó (60 000 Ft) vagy C64 + 1541 + 25 000 Ft-ra cserélhető! Kecskemét, Széchenyi s. 4. VII/713. 6000

**Enterprise** számítógép jótállással, 2 db könyvvel és 40 db programmal sürgősen eladó. Irányár 22 000 Ft. Varga József, Kecskemét, Felsőcsalános 173. 6000

**Enterprise**-ra magyar ékezetes karakterkészlet eladó. A memóriából helyet nem foglal el. Bányai Csaba, Nyíregyháza, Tanácsköztársaság tér 3. 4400

**Ezt a lehetőséget nem szalaszthatja el!** Commodore 128D - 3 elsődleges üzemmód: C128, C64, CP/M, beépített 1571-es floppy - és 100 db diszk válogatott programokkal olcsón eladó. Ugye érdekli Önt?! Radics Róbert, Kiskunfélegyháza, Szőlő u. 10/4. 6100

**Fényceruza Plus/4**-hez, postai utánvételt 1300 Ft. Computeam GM. Kaposvár, Berzsenyi u. 32. 7400

**A 64'er Magazinból** fordításokat készítek. Több mint 3000 oldalnyi kész anyag! Kedvező árak. Textomat+, GEOS 2.0, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Buda-

pest, Pf.: 400. 1446. Válaszborítékot kérünk!

**Joystick** szervíz a Flórián Áruházhoz közel! Javítás, magnófej-beállítás. C64 játécképeket kazettán és floppy-n 10 Ft/db. Budapest, III. ker. Kerék u. 36. IV/24. Hétfőn és szerdán 17-től 19-ig.

**Mikroszámítógép Magazin** régi számai eladók. Tóth Gábor, Debrecen, István u. 17B. 4031. Tel.: 52/32-169

**MSX rendszerű számítógép** nagyon olcsón eladó vagy színes monitorra cserélhető. Bővebb információ az alábbi címen: Jancsin Ferenc, Békéscsaba, Lencsési u. 8. 5600. Tel.: 39-827

**Primo SSD**-hez 32 k-s komplett felhasználói programcsomag égetése. Vidékre utánvétellel. Előzetes megbeszélés: Horváth László 583-544 délelőtt.

**Seikosha SP 180 VC** teljes magyar ékezetes nyomtató C64-hez, magyar gépkönyvvel eladó. Gyöngyösi Attila, Tel.: 588-280 vagy 853-111/189

**Spectrum** programok (48 k) eladók. Magyarországon a legolcsóbban, 12 Ft/db. Válaszborítékért katalógust küldök. Megfelelő partner esetén cserével is foglalkozom. Eötvös Levente, Debrecen, Cserépek u. 22. 4026

**Spectrum** (48/128 k) és C64 (lemez/kazetta) programok olcsón eladók. Válaszborítékért tájékoztatót küldök. Régi és új programok egyaránt kaphatók. Kiss Gábor, Győr, Venyige u. 31. 9028

**Tudja Ön, hogy milyen egyszerű a FASTLOAD CARTRIDGE-el dolgozni?** Vállalom bővített karakterkészlet, Reset gomb, IRQ (pause) kapcsoló beépítését. Seikosha SP 1200VC - 180VC nyomtatók magyarosítását, FASTLOAD, SPEEDTAPE programsegítő modulok készítését. Szívesen küldök tájékoztatót! Bártfai Barnabás, Budapest, Móricz Zs. u. 22. 1193. Tel.: 668-411/64 (délelőtt).

**TVC** (64 k) programokat adok-veszek-cserélek. (600 db) Listát kérek! Molnár János, Szolnok, Jászi F. u. 10. VI/25. Tel.: 56/31-085

**VC 1541/II**-es floppy minőségi programokkal eladó! (21 db 3M lemez; TEST DRIVER; GAMES-EK stb.) Semjén Péter, Pécs, Sarohin T. u. 13B. II/7. 7632

**ZX-Spectrum** programok eladók. Válaszborítékért katalógust küldök. Bogyó Frigyes, Pécs, Bihari János u. 3/B. fsz. 1. 7633

**ZX-Spectrumhoz** új joystick interfész (2 bemenettel) olcsón eladó. Ezenkívül a Mikroszámítógép Magazin 84/6-tól 88/8-ig terjedő számai. Beregszászi Gábor, Budapest, Abos u. 13/A. 1122

**ZX-Spectrum 48 k** (14 000), printer (papírral 6000), interfész 2. (joystick-kal 3000), 10 kazetta játékprogram (4000) Ft-ért eladó! Az egész együtt 1 számítógéppel és könyvekkel 27 000 Ft. Érdeklődni e hónap végéig lehet. Molnár Ádám, Budapest, Muhar u. 57. 1028 vagy 169-884

### VESZÉK

**C64**-hez Newsroom és Gemaker nevű programot keresek. Választ levélben várok. Tóth Dávid, Dunakeszi, Rákóczi u. 13. 2120

**Commodore 1541-es** üzemképtelen drive-ot veszek. Árjajánlatos jelentkezést postán kérek. Remeczki István, Budapest, Attila u. 49. 1013

### CSERÉLEK

**C16, C Plus/4** programokat cserélek vagy eladok (10 Ft/db) kazettán. Listát kérek és adok! Szűcs Géza, Kapuvár, Szabadság u. 49. 9330

**C16, Plus/4** programokat cserélek kazettán. Cserealapom kb. 370 program. Keresem: Fortress Underground, Eliza, Super Tape, Rázós út, Reklámújság, Képjűség nevű programokat. Gyukity Péter, Budakalász, Sallai Imre u. 3. 2011

**Commodore 16, C Plus/4-es** márkatársak figyelme! Gépi kódban jártas cserepartnert keresek, program- és tapasztalat-csere céljából. Németh Tamás, Dunaújváros, Alkotás u. 10. II/1. 2400

**C Plus/4** programokat cserélek. Keresek: nyák tervezőt. Érdekelnek: harver ötletek, zene programok. Kóptaki Ákos, Bonyhád, Kossuth u. 57. 7150, Tel.: 74/51-585

**C64-re** szuper játék- és felhasználói programok állnak rendelkezésemre. Csereajánlatokat listával várok! Csak lemezen érdekel a csere. Komjáthy Ágnes, Siklós, Felszabados u. 96. 7800

**C64-es** programokat cserélek, másolok lemezen és kazettán. Listát kérek. Molnár József, Kőszeg, Gyöngyös u. 24. fsz. 2. 9730

**Enterprise** programokat cserélek. Listát kérek! Molnár Imre, Budapest, Havanana u. 7. V/27. 1181

**IBM PC XT-re** játék- és felhasználói programot cserélek. Borsodi Zoltán, Budapest, Kárász u. 4. II/1. 1131

**Sinclair Spectrum 48 k-s** programokat cserélek. Válaszokat csak levélben, listával kérek! Glattfelder György, Nyíregyháza, Legyező u. 21/6. 4400

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

# Az Olvasó írja

Lázár Károly, Budapest

Úgy adódott, hogy most volt szükségem egy olyan programra, amely ábécésorrendbe rendez ékezetes betűket is tartalmazó szavakat. Elővettem tehát lapjuk 1988. évi 7. számát, amelynek 6. oldalán „Ékezetes névsorok rendezése” cím alatt ilyen programot közöltek. Spectrum gépem van, így a 6. lista szerint készítettem el a programot, ami sajnos nem működött. Az egyik hibára a program első futtatásakor azonnal fény derült: a 330. sorban nem  $a\$(i, j)$ , hanem  $f\$(i, j)$  a helyes. A másik hiba felderítése már sokkal több munkát igényelt: a 450. sor utolsó előtti utasításában  $LET\ i = i - m$  a helyes, nem pedig  $i + m$ , ahogy a listában áll. Hibás listák megjelenése a számítástechnikai szakirodalomban sajnos nem ritkaság. A szóban forgó listát azonban láthatólag a számítógép nyomtatta ki, tehát nem sajtóhibáról van szó. A szerző úgy látszik, nem vette a fáradságot, hogy közlésre beküldött saját programját a kinyomtatás előtt lefuttassa, hiszen a hibaüzeneteket ő is megkapta volna a géptől és kijavíthatta volna a hibákat.

A program ismertetéséből nem derül ki egyértelműen, hogy a Spectrumra írt program csak akkor működik helyesen, ha az ékezetes betűket felhasználói grafikus jelek formájában állítjuk elő és használjuk, nem pedig valamely más karakterek cseréjével, ami szintén szokásos eljárás. Van ugyan egy rövid utalás a cikk végén az ötlet egyik cikkére, mint javasolt módszerre az ékezetes betűk előállításához, de nagyobb hangsúlyt kellett volna adni ennek a követelménynek (amely egyébként alapját képezi a 350. sorban közölt átkódolásnak is).

Annak érdekében, hogy másokat megkíméljenek az esetleges bosszúságtól, kérem, hogy tegyék közzé az említett hibajavításokat lapjukban.

*Mint azt a fenti levél is tanúsítja, többször előfordult, hogy lapunkban hibás lista jelent meg. Mivel a szerkesztőségnek nem áll módjában (gépek hiányában), hogy ellenőrizze ezeket a programokat, ismétlenül felhívjuk kedves állandó és leendő szerzőink figyelmét arra, hogy listáikat fokozott ellenőrzés után juttassák el címünkre. Megértésüket és segítségüket köszönjük.*

**THF, Veszprém („Egy fanatikus Amiga-rajongó”)**

Olvasom, olvasom a Mikroszámítógép Magazin, és már megint azon veszem észre magam, hogy dühöngök. Nagyon jól tudom, hogy egy-egy géptípus védelmében harcra szálló leveleket önkönként kaphatnak, én most megis ezt teszem. Ez a gép a Commodore új szupersztár gépe, az Amiga. Az önkönként újságjában — magazinjában — erről a gépről még egyetlen cikk sem jelent meg, elvéve csak néhány félmondatos utalás!! Es jó, ha volt egy egész sor!!!

Nem gondolják, hogy a magazinak együtt kellene haladnia a korrallal? Hiszen önkönként annyi géptípussal foglalkoztak már — köztük réges-rég bukott gépekkel is —, miért pont az A500-as marad ki? Tudom jól,

hogy az ország számítógép-állományának igen nagy része C16, C Plus/4, C64, Sinclair, Enterprise gépekből áll . . . de ez nem jelenti azt, hogy azok, akik komolyabb gépekkel foglalkoznak, azok nem léteznek. Azt is tudom, hogy önöknek a géptulajdonosokhoz kell igazodniuk, azonban szerintem a magazin színvonalát is emelné egy-egy amigás cikk, azonkívül óriási úrtöltene be, mert nemcsak a  $\mu$  Magazin nem jelent meg amigás cikkeket.

*Eddig valóban nem közzeltünk amigás írásokat, mivel nem találtunk megfelelő szerzőt, és olvasóinktól sem kaptunk használható anyagot. De szeretnénk megnyugtatni a fenti sorok íróját és minden Amiga-rajongót, hogy a közeljövőben sorozatot indítunk és előreláthatóan a 89/6-os számunkban már olvashatnak a Magazinban kedvenc gépükről.*

Lőrincz Gergely, Budapest

14 éves, lelkes olvasójuk vagyok. Commodore Plus/4-gyel rendelkezem. Progra-

mozni tanulok és gyűjtöm a programokat. Lenne egy tippem: Miért nem tesznek az újságba posztert. (Például: Commodore, Enterprise, ZX-Spectrum játékoknak a címkéjét, főszereplőjét vagy a kazetta borítóját felnagyítva.) En minden pénzt megadnék érte, ha még drágább is lenne.

*Fiatal olvasónk levele elgondolkodtatott, vajon hányan vennék meg lapunkat a felémelt áron?*

*Mindent megteszünk annak érdekében, hogy a Magazin továbbra is eljusson az érdeklődőkhöz, még olyan áron is, hogy az állandó költségemelkedések ellenére — a veszteségünk további növekedése mellett — az idén is 30 Ft-os áron juthatnak hozzá.*

*Végül pedig egy tiltakozás, amely kapcsolódik a februári számunkban megjelent felhíváshoz:*

**Fekete Katalin, rendszerszervező és számítógép-tulajdonos**

Mélységesen egyetértek a Mikroszámítógép Magazin 1989/2. számában közzétett Nem rejtjük véka alá című cikk felhívásában foglaltakkal.

*Szerkesztőségünk tagjai továbbra is várják hasznos észrevételeiket és érdeklődésüket.*  
Tamásné Lakó Erika

## Az AKADÉMIAI KIADÓ számítástechnikai könyvújdonságai

**Kotsis Domokos—Quittner Pál:**  
**Számítástechnika rendszerszervezőknek**  
**2., bővített kiadás**

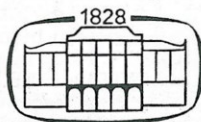
A mű szerzői a számítógépes témakörök szinte mindegyikével foglalkoznak, így az adatszervezési eljárásokkal, az adatbázis-kezelő és operációs rendszerekkel, az adat-előkészítés korszerű módjaival, valamint a döntési táblázatok készítésének és felhasználásának lehetőségeivel. Részletesen tárgyalják a személyi számítógépekkel kapcsolatos tudnivalókat, önálló fejezetet szentelnek a távadat-feldolgozásnak, a szoftver védelmének stb.  
Kb. 480 oldal — Ára kötve kb. 200,— Ft

**Álló Géza—Kelemen Dezső—Hegedűs Gy. Csaba—Szabó József:**  
**A digitális képfeldolgozás alapproblémái**  
**Az elektronika újabb eredményei 4.**

A szerzők a digitális képfeldolgozás hazánkban is művelt legfontosabb területeit tekintik át. Mivel az egységes elmélet hiánya nagyon megnehezítette a rendszerező munkát, a könyvnyebb áttekintés végett kidolgoztak egy modellt és az anyagot eszerint rendezték négy témakörbe: képjavítások, geometria, korrekciók, szegmentálás, alakfelismerés. Ezen belül a megoldási módszereket vették a csoportosítás alapjául, rámutatva az egyes eljárások előnyeire és hátrányaira.

Kb. 400 oldal — Ára kötve kb. 100,— Ft

A fenti kiadványok előjegyezhetők, ill. postai szállításra — a portókölség felszámításával — megrendelhetők az akadémiai könyvesboltokban:



**STÚDIUM Akadémiai Könyvesbolt**  
1052 Budapest V., Váci u. 22.  
**MAGISZTER Akadémiai Könyvesbolt**  
1052 Budapest V., Városház u. 1.

*Mindkét könyvesboltban számítástechnikai szakkönyvek, játék- és egyéb programok (köztük profi szoftverek) gazdag választéka!*

## A Rudas László Szakközépiskola ajánlata

**Egyhetes számítástechnikai nyári tábort szervez IBM PC gépekre, melynek keretében a számítástechnikai angol szavak helyes kiejtését is elsajátíthatják a tanulók.**

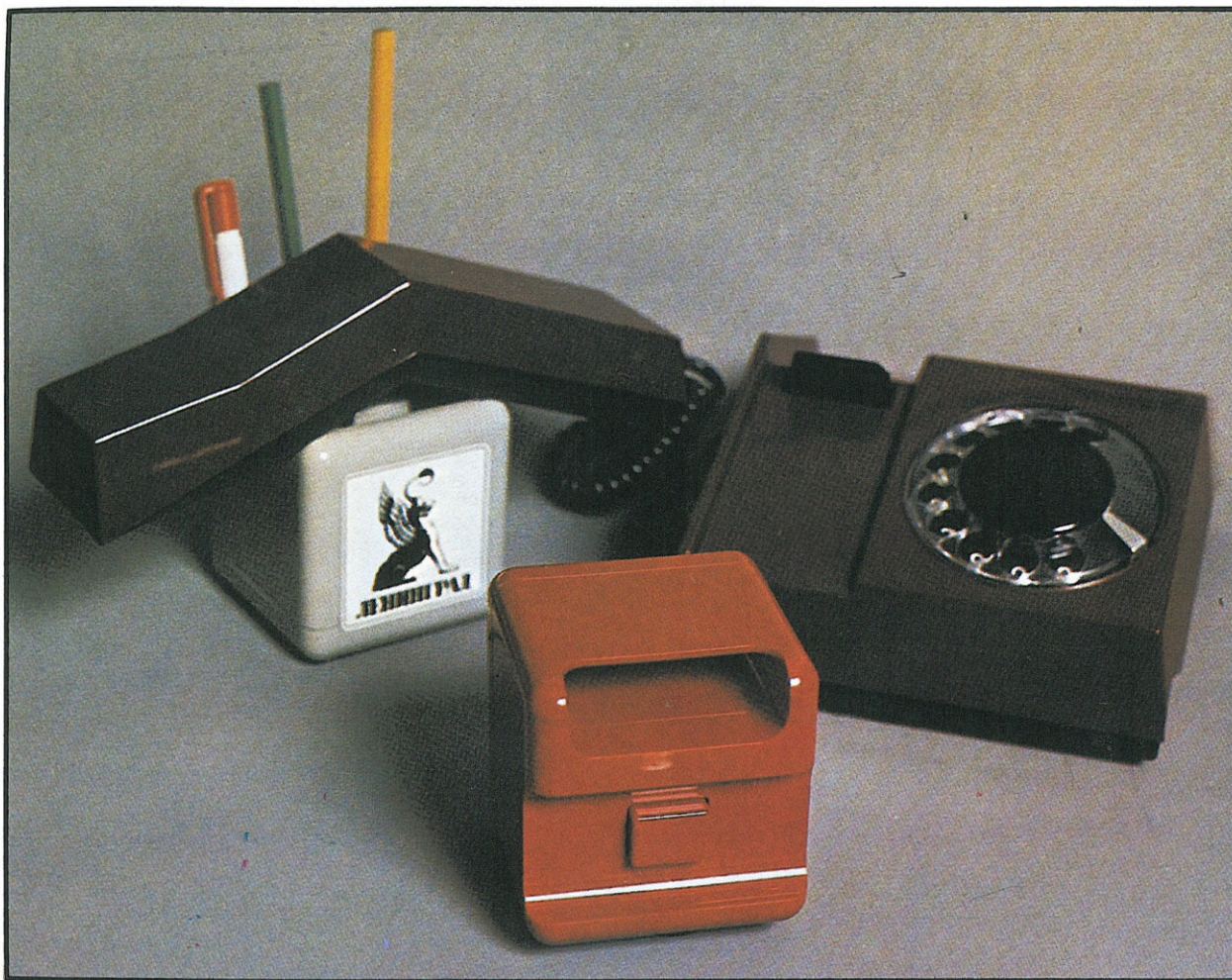
**Elsősorban tanulók jelentkezését várjuk 12 éves kortól 1989. május 20-ig turnusmegjelöléssel.**

**Elhelyezés az iskola kollégiumában.**

**Első turnus június 25-től.**

**Utolsó turnus augusztus 6-tól kezdődik (beérkezés vasárnap délután).**

**Levél cím: PC-tábor 2403 Dunaújváros Római körvasút 47. Pf. 8**



**Az elektronikus telefonalátét**

*Ceruzatartóval kombinált telefonalátétet készített a leningrádi Lenelektronmas Tudományos-Termelő Egyesülés. A telefonbeszélgetések szünetében erre helyezhetjük a telefonkagylót, amit érzékelve kellemes dallamot kezd játszani. Így egyrészt emlékeztet bennünket a függőben maradt beszélgetésre, másrészt pedig beszélgetőpartnerünket kellemes zenével szórakoztatja.*

**ZENÉLŐ ALÁTÉT**

# Bajor Szabadállam – Hagyomány és jövő

## Bajor kiállítás Magyarországon

1989. május 7-16, naponta 10<sup>00</sup> – től 18<sup>00</sup> óráig

## Könyvkiállítás

1989. május 7-17, naponta 10<sup>00</sup> – től 18<sup>00</sup> óráig

## Sörkert bajor zenével

1989. május 7-17, naponta 16<sup>00</sup> – től 24<sup>00</sup> óráig

## Tudományos szimpóziumprogram

1989. május 14-17,  
naponta 9<sup>00</sup> – től 16<sup>00</sup> óráig

**Budapest**  
**Kongresszusi Központ**

A müncheni Bajor Gazdasági és Közlekedési  
Minisztérium rendezvénye

A szimpózium rendezvényeinek részletes programját a  
Gépipari Tudományos Egyesülettől kaphatja meg,  
1055 Budapest, Kossuth L. tér 6/8. Telefon: 530-025