Ára: 30 Ft

μ mikro számítógép magazin

AHÁNY NYELVET BESZÉLSZ...

1988/5

Mi az, ami legjobban foglalkoztat bennünket 1988. évben? az ADÓ

Kedvező ajánlat: DÍJMENTESEN 10 munkanapig kipróbálhatja új termékünket!

SZEMÉLYI JÖVEDELEMADÓ SZÁMÍTÓ PROGRAMRENDSZER IBM PC kompatibilis számítógépekre, vállalatok, szövetkezetek és egyéb gazdálkodó szervezetek számára.

Szolgáltatásai:

- havi adóelőleg megállapítása dolgozónként (a főállású munkaviszonyból származó jövedelmek és az egyéb munkáltatói kifizetések teljes körű figyelembevételével),
- adóelőleg előírt szempontok szerinti részletezése és összesítése,
- éves adóelszámolás elkészítése dolgozónként,
- éves adatszolgáltatás az APEH részére.

KÉRJEN BŐVEBB TÁJÉKOZTATÁST, TERMÉKBEMUTATÓT!

Ár: 25 000 Ft (+ ÁFA)

OKISZ Szervezési és Számítástechnikai Vállalat
Alkalmazástechnikai Iroda
Budapest IX., Tűzoltó u. 79. Telefon: 138-659

TETSZIK? BÉRÉT IS ELSZÁMOLJUK?
Modulárisan köréépül **Bérelszámolási és Munkaügyi egységes rendszerünk!**
3 modul — olcsóbb ár! (220 000 Ft+ ÁFA)

OKISZ Szervezési és Számítástechnikai Vállalat

OKISzoft

Egyéb rendszereinkkel is készséggel állunk rendelkezésükre.

mikro számítógép magazin

6. ÉVFOLYAM
1988/5. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

Címképünk:
Szentiványi Kálmán

Felelős szerkesztő:
Könyves Tóth Pál

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

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Király G. István
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivatalokná
és a Posta Hírlap-előfizetési és Lapellátási Irodáján
(1900 Budapest XIII., Lehel út 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.
86—0253

Szikra Lapnyomda
Budapest (88—0448)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Értelmiség '88

„Legfőbb nemzeti értékünk, amivel felkészülhetünk a nemrég kezdődött időszak történelmi erőpróbájára, a kreativitás gyors emelése. Ebben a vonatkozásban követtük el valószínűleg a leghosszabb távon ható hibákat. Az erőforrások szegénységével küzdő rendszer tartalékai felélésére kényszerül, ami azonban a következő periódusban sokkal nagyobb befektetést igénylő és sokkal hosszabb időszak alatt helyreállítható károkat okoz. (...) A következő évtizedek adója a jelennel szemben a kreatív értelmiségi munka devalválódásából fog fakadni. (Vámos Tibor: Hazánk és a műszaki haladás. Az MTA 1987. évi 141. közgyűlésének központi előadása)

Emlékszem, valami nagyon unalmas tárgyaláson vettünk részt; megkönnyebbültünk, amikor az utolsó papírlap is az asztalra került. „Hozzászólás!" „Ha nincs hozzászólás, akkor megköszönöm az érdekes...", és már mehettünk is ki a teremből. Szidtuk a buta fejünket, hogy miért is hallgattuk meg ezt a sok sületlenséget. „Talán még az is jobb lett volna, ha ezt a másfél órát tanulásra fordítom" — mondta a társaság egyik tagja. „Tanulni, miért, beiratkoztál valahova?" Szó szót követett, ki tanul, mit tanul, tanul-e egyáltalán, és máris megtaláltuk az aktuális „világmegváltó témánkat": kell-e és mennyit kell egy mai értelmiséginek tanulnia?

Az angol nyílt tanulási renszerről beszélgetve barátaim azt mondták, hogy ma az értelmiségi munkakörökben általában legalább heti 5 órát kell tanulni, szakirodalmat olvasni ahhoz, hogy egy mérnök, egy fizikus, egy tanár, egy programozó, nem beszélve az orvosokról vagy a menedzserekről — karban tudja tartani a tudását. Ha valaki egy új munkakört szeretne ellátni, akkor a heti tanulási szükséglet 10-15 óra. E beszámolómról az ellenzék rögtön úgy nyilatkozott, hogy én valamit biztosan elértettem, és az az angol biztosan nem heti, hanem havi penzumot említett, amibe persze beleértette a munkahelyi tanfolyamokat is. Rövid közvélemény-kutatás következett. Volt a megkérdezettek között, aki napi 2 órát is tanul rendszeresen, néha még többet is, és így *tudja követni* a tudományos életben bekövetkező gyors változásokat. A zöm azonban — sajnos — igazolta az ellenvéleményt: azt mondták, hogy havi 5-10 órát ugyan kénytelenek áldozni a szabad idejükből a szakmájukra és utánanézni egy s másnak, de csakis akkor, ha erre a munkahelyükön nincs lehetőségük.

A legtöbben az értelmiségi pálya alulfizetettségét jelölték meg a nem tanulás okaként. Éppen a fiatal értelmiségiek mondták ezt, akiknek — véleményem szerint — a legtöbbet kellene tanulniuk és főleg azért, mert gyakorlati tapasztalattal nem rendelkezvén, csak a felsőoktatási intézményekben szerzett elméleti tudásukra tudnak támaszkodni. Nem szívesen ismétlem az érveiket (lakás kell, család kell, autó kellene, utazni kellene és mindezt a kezdő „felbruttósított" 4-6000 forint fizetésből); nem csoda, ha a pályakezdő fiatalember a főállásos munkája után maszekol, még talán az a legszerencsésebb megoldás, ha cikket vagy könyvet ír, mert a művét csak sok tanulás árán tudja színvonalasan elkészíteni. Arról persze már ne beszéljünk, hogy a honorárium milyen.

Egyesek visszakérdeztek: „Mondd, ha tanulok, akkor mi van? Magasabb lesz a fizetésem, esetleg magasabb beosztásba helyeznek? Te tényleg azt hiszed, hogy bárkit a főnökeim közül érdekel az, hogy én elvégeztem egy továbbképző tanfolyamot és bizonyítványt szereztem?" Valóban, én jól tudom, hogy a jelenlegi merev bértáblázatok mindenre alkalmasak, csak arra nem, hogy a magasabb tudást elismerjék.

Végül a válaszadók igen nagy csoportjának volt az a véleménye, hogy „szeretnék tanulni, de hol? Iratkozzak be egy továbbképző kurzusra, ahol az előadott és vizsgaköteles tárgyaknak már legalább a felét tanultam, most újra tanuljam meg egy másik előadó szája íze szerint? Szerinted ez nem elpocsékolt idő? Én csak néhány tantárgyra vagyok kíváncsi, azt szívesen megtanulnám, de mutass nekem egy intézményt, ahol az általam kiválasztott tantárgyak tanulására lehetőséget adnak!"

Valóban — ellentétben az iparilag fejlett országokkal, de egyre több fejlődő országgal is — se nálunk, se a többi szocialista országban nincs bevezetett nyílt tanulási rendszer, vagy ahogy újabban nevezzük, távtanulás (distance learning), tehát egy olyan állami közoktatási rendszer, amelyik az egyéni tanulás és a vizsgázás, tehát a bizonyítvány-, illetve a képesítésszerzés lehetőségét individuális módon biztosítja. Ebben a rendszerben a tanuló választja ki a tantárgyakat is.

E hasábokon már nagyon sokszor írtam a távtanulásról. Sajnos ezen a téren vajmi kevés történt az elmúlt időszakban, és ami történt, az is csak az informatikában. Ide sorolom a Gödöllői Agrártudományi Egyetem és az LSI ATSZ hamarosan induló nyílt tagozatát és azt a távtanulási rendszert is, amelyről a megállapodás a cikk írásának idején van születőben a KISZ KB, az AISH, a BME, a KKMF, a TTK, valamint az MTV, a Rádió és a SZÁMALK között, az utóbbinak a gesztorálásával. Ebben, a SZÁMALK Távtanulási Központ által szervezett rendszerben a tankönyv mellett a három legfontosabb közvetítő a televízió, a rádió és a számítógépes oktatási program. Az persze köztudott, hogy minden médiát alkalmazó oktatási rendszer rendkívüli módon beruházásigényes, hiszen a jelenlegi árak mellett egy 16×20 perces videofilm vagy egy 10-15 órás számítógépes oktatási program milliókba kerül. Márpedig a tananyagkészítés költségeit az árak színvonalához, a tanulók által fizetett költségeket pedig a kereseti színvonalhoz kell igazítani, ami önmagában is óriási feszültséget jelent.

A legnagyobb múltú távtanulási intézményben, az 1963-ban alapított British Open Universityn (OU) ma kb. 100 ezer diák tanul évente. Egy-egy diák félévenként általában két tantárgyat vesz fel, ami azt jelenti, hogy minden félévben 200 ezer tantárgy után fizetnek díjat a hallgatók. A beiratkozási díjakból az OU évi árbevétele kb. 20 millió font. Az évi költségük sokkal több, a hiányzó cca. háromnegyed részt az állam pótolja. Felmerül a kérdés, hogy érdemes-e Angliának ezt az irdatlan összeget a tömegoktatásra fordítania. Azt mondták — és nemcsak az OU munkatársai —, hogy nemcsak érdemes, hanem kötelező is. Hiszen

— Anglia fejlődését, a világpiacon való versenyképességét csak jól képzett értelmiséggel lehet biztosítani,

— az angol gazdaság egyre gyakoribb foglalkoztatási válságait — lásd bányászat, kohászat, legutóbb a nyomdaipar —, amelyeket nem is kis mértékben a technika fejlődése okoz, csak egy folyamatosan üzemelő, korszerű távtanulási rendszerrel lehet túlélni,

— a gazdasági problémákra csak sokoldalúan művelt emberek képesek hatékony megoldást találni, így a társadalom minden tagja részére — és nem is csak a szakmai továbbképzést — az általános műveltség megszerzését és gyarapítását kell biztosítani,

— végül nem elhanyagolható argumentum a távtanulási rendszer mellett, hogy bizonyos szakmákban a hagyományosnak a negyedéért, máshol (például bölcsészet) a tizedéért vagy még annál is kevesebb költséggel tud ez a rendszer diplomás szakembereket előállítani.

Nem árt a távtanulás ellenzőinek az érveit is összegyűjteni, még akkor sem, ha ezek könnyűszerrel cáfolhatók. Az ellenzők ugyanis

— az oktatás dehumanizálását (technologizálását) roppant veszélyesnek tartják. Az évszázados tanár—diák kapcsolat megszűnése — úgymond — a tudásszint gépi megmérése (tesztvizsga) véleményük szerint egyben a képzés színvonalának katasztrofális csökkenését is jelenti,

— azt mondják, hogy minek annyi tanult ember!? Ha jó a gazdasági vezetés, akkor pontosan meg tudják mondani, hogy mennyi mérnökre, tanárra, orvosra van szükség, a túl sok „okos fej" szakember-inflációhoz vezet; ha mindenki tanul, akkor ki végzi majd a fizikai munkát?

— a távtanulásba sokat kell befektetni, mire el lehet kezdeni a tanulást. Szegények vagyunk, örülünk, hogy élünk, nem kell nekünk ilyen ütemben követni a gazdaságilag fejlett országokat stb.

Talán megengedi a kedves olvasó, hogy ezekre az érvekre, hiszen a felelet annyira triviális, ne válaszoljak. Ha netán a válasz mégsem kézenfekvő, akkor baj van — sajnos nem Angliában.

Az ellenkezőjét reméli

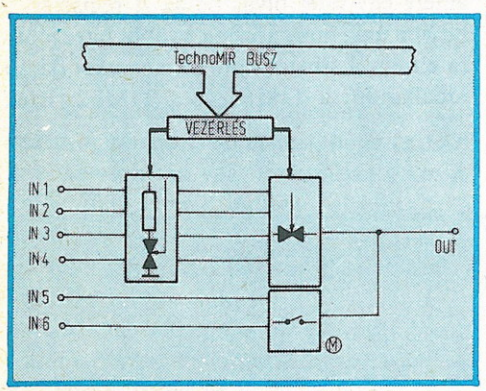
KOVÁCS GYŐZŐ

TechnoMIR

Analóg modulok

Hogyan sokszorozható meg egy rendszer?

Az analóg sokszorozó (AMUX) modul *(1. ábra)* alapvető feladata, hogy vezérlőprogramtól függően a hat bemeneti pontot (IN1-től IN6-ig) összekösse a kimeneti ponttal (OUT). Ennek a vezérelhető kapcsolófunkciót ellátó modulnak hat bemenete közül négy (IN1—IN4) a CD 4066 CMOS analóg kapcsoló IC-n keresztül



1. ábra

igen gyorsan köthető össze a kimenettel. Az IN5 és IN6 bemenetek reed relén keresztül kapcsolhatók, így ezek lassúbbak, de jobban terhelhetők: max. 25 mA áramot viselnek el károsodás nélkül. A kapcsolható feszültség 0 V < U < +10 V minden bemeneten.

A modul felhasználhatóságát növeli, hogy az IN1—IN4 bemenetek nemcsak aktív érzékelőket képesek fogadni, hanem a programtól függően passzív érzékelők jeleit is el tudják juttatni a kimenetre. Ez a passzív érzékelő 1 mA-es árammal történő táplálásával érhető el.

1. táblázat

Bemenet	d2 bit	d1 bit	d0 bit
IN1	0	0	0
IN2	0	0	1
IN3	0	1	0
IN4	0	1	1
IN5	1	0	0
IN6	1	0	1

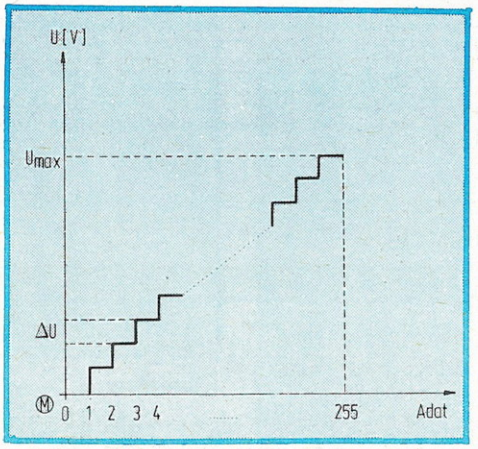
Bemenet	Bit
IN4	d4
IN3	d5
IN2	d6
IN1	d7

2. táblázat

A modul a következőképpen programozható. Az AMUX HT címe 25, így HT gépen OUT 25,SZÁM utasítással lehet programozni. A SZÁM értékének megváltoztatásával változtatható a modul állapota. Ez az érték a következő bitenkénti információkat hordozza:

d0 — d2 bitek határozzák meg, hogy melyik bemenet kapcsolódik a kimenetre *(1. táblázat)*.

d3 bit „1" állapota az összes bemenet letiltását eredményezi, azaz egyik bemenet jele sem kerül a kimenetre. A bit „0" állapota az engedélyezés.



2. ábra

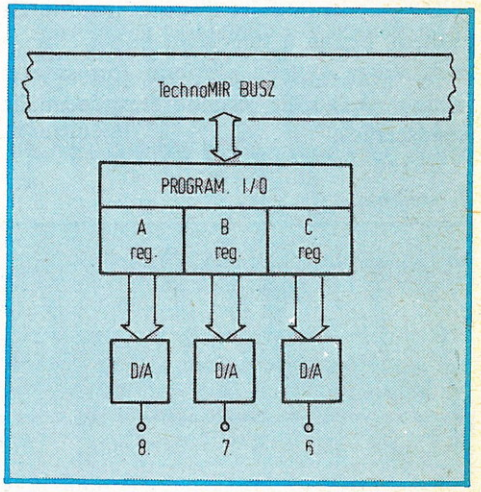
d4 — d7 bitekkel az IN1—IN4 bemenetek táplálását lehet beállítani *(2. táblázat)*. A megfelelő bit „1" értéke táplált, a „0" érték nem táplált bemenetet eredményez. A tápláláskijelölés és a bemenetengedélyezés egymástól független, csak az jelenik meg a kimeneten, amit a d0 — d2 bitek meghatároznak.

Végül egy figyelmeztetés: az AMUX mellett a POW modult is alkalmazni kell!

D/A modul

A D/A modul olyan programozható digitális-analóg átalakító, amely a számítógépből érkező digitális, 8 bites adatot (0—255 közötti szám) analóg jellé, meghatározott feszültséggé alakítja. Egy D/A konverter transzfer karakterisztikája látható a *2. ábrán.*

A TechnoMIR rendszer D/A modulja három, egymástól függetlenül használható átalakítót tartalmaz *(3. ábra)*. Az egyes ki-



3. ábra

Kimenetek:	6.	7.	8.
HT cím	12	13	14
Utasítás	OUT 12,X	OUT 13,X	OUT 14,X
U(ki)	-10V +10V	0V +10V	0V +10V
I(ki)max	50 mA	25 mA	25 mA

Megjegyzés: 0<X<255 egész szám

3. táblázat

menetek mindaddig az előzőleg beállított feszültséget szolgáltatják, míg a bemeneten található I/0 felület megfelelő regisztereibe más adatot nem írunk. Az I/0 rendszert a PB modul kapcsán már részleteztük.

Itt is a kontrollregisztert kell először feltölteni. HT címe 15, így HT-n a megfelelő parancs OUT 15,128. Ez C64-en POKE 56848,128. A további tudnivalók a *3. táblázatról* leolvashatók.

KIRÁLY LÁSZLÓ—ALBU LÁSZLÓ

AZ ABC 80 programjainak beolvasása HT-be

Bizonyára nem egyedül vagyok az országban, aki még az ABC 80-on tanulta a BASIC alapjait, és fiókja mélyén ma is őriz egy olyan szalagot, amelyen erre a gépre írt programok vannak. Programok tehát lennének, ám ABC 80-as számítógép nincs a közelben. Más gépen pedig nem lehet felhasználni ezeket a programokat. Ezen a problémán próbálok most segíteni.

Az ABC 80 kétféle progamfájlt tud készíteni: az egyiket a LIST paranccsal lehet felvenni és forrásnyelvi alakban tárolja a programot, ahogyan a billentyűzetről kapta, a másikat pedig a SAVE paranccsal lehet felvenni, és egy tömörített kódot tartalmaz. A módszer, melyet ismertetek, mind a két fajtát beolvassa, de közvetlenül csak a LIST paranccsal felvettet lehet listázni. Ez érthető, hiszen a tömörített kódot először még vissza kellene alakítani forrásnyelvi alakra. A közölt számok hexadecimális rendszerűek.

Tekintsük át a program tárolásának módját! Az ABC az adatokat blokkokban veszi fel, melyek felépítése a következő:

— 64 db 00 bájt
— 3 db 16 bájt
— 1 db 02 bájt
— 256 db adatbájt
— 1 db 03 bájt
— 2 db ellenőrző összeg, kétbájtos (adatbájtok + 03 bájt)
— 1 db 00 bájt

Az adatbájtok felépítése a tartalomnak megfelelően változik. A program legelső blokkjának szerkezete:

— 3 db FF bájt
— 11 db azonosító (8 bájtnév + BAS vagy BAC mutató)
— 242 db 00 bájt

A program egy általános blokkjának adatbájt-felépítése:

— 1 db 00 bájt
— 2 db blokkszám, kétbájtos
— 252 db szöveg
— 1 db 03 bájt

A program utolsó blokkjának adatbájt-felépítése:

— 1 db 00 bájt
— 2 db blokkszám, kétbájtos
— 6 db 00 bájt
— 247 db, az előző blokkból még a pufferben maradt adat.

Tudni kell még azt is, hogy a szalagon a bitek a nevük szerinti növekvő sorrendben vannak, tehát az első a 0., az utolsó a 7. bit.

Ezek után tekintsük az *1. programot*, amely egy bitet olvas be az A regiszter 7. bitjébe úgy, hogy közben a többi bitet jobbra eltolja. Látható a programban két darab időállandó: a TIME1 és a TIME2. Ezek

1. program

```
BITOLV:PUSH BC          BC regiszterpár megőrzése
       PUSH AF          AF regiszterpár megőrzése
HUR1   :IN   A,(0FFH)   A-ba a port értéke
       RLA              a 7. bit a c-be
       JR   NC,HUR1     várakozás az órajelre, ha c=0
       LD   B,TIME1     B-be az első időállandó
HUR2   :DJNZ HUR2       várakozás
       CALL 021EH       a vezérlőkód átállítása
       LD   B,TIME2     B-be a második időállandó
HUR3   :DJNZ HUR3       várakozás
       IN   A,(0FFH)    A-ba a port értéke
       LD   B,A         beolvasott érték megőrzése B-ben
       POP  AF          AF regiszterpár vissza a zsákból
       RL   B           a vett bit c-be
       RRA              a vett bit beforgatása A-ba
       PUSH AF          AF regiszterpár megőrzése
       CALL 021EH       a vezérlőkód átállítása
       POP  AF          AF regiszterpár vissza a zsákból
       POP BC           BC regiszterpár vissza a zsákból
       RET              visszatérés
```

```
BYTOLV :LD   B,08H      B-be a bitek száma
HUR4   :CALL BITOLV     egy bit olvasása
       DJNZ  HUR4       ugrás a többi bit beolvasására
       RET              visszatérés
```

2. program

```
FEJOLV :XOR  A          A regiszter nullázása
HUR5   :CALL BITOLV     egy bit olvasása
       CP    16H        A=16H ?
       JR    NZ,HUR5    ugrás, ha nem egyezik meg
HUR6   :CALL BYTOLV     egy bájt olvasása
       CP    16H        A=16H ?
       JR    Z,HUR6     ugrás, ha megegyezik
       CP    02H        A=02H ?
       JR    NZ,HUR5    ugrás, ha nem egyezik meg
       RET              visszatérés
```

3. program

4. program

```
7000 DD 21 00 80 AF CD 12 02 CD B7 70 DD E5 CD 6E 70
7010 CD 6E 70 CD 6E 70 0E FD CD 76 70 CD 6E 70 CD 6E
7020 70 CD 6E 70 CD 6E 70 E1 7E A7 20 DC CD F8 01 21
7030 00 80 CD C9 01 7E 23 CD 3A 03 FE 0D 20 F7 3A 01
7040 38 A7 20 F1 3A 04 38 E6 40 C2 00 43 18 F0 C5 F5
7050 DB FF 17 30 FB 06 28 10 FE CD 1E 02 06 4B 10 FE
7060 DB FF 47 F1 CB 10 1F F5 CD 1E 02 F1 C1 C9 06 08
7070 CD 4E 70 10 FB C9 11 00 3C CD 6E 70 12 DD 77 00
7080 13 DD 23 0D 20 F3 C9 AF CD 4E 70 FE 16 20 F9 CD
7090 6E 70 FE 16 28 F9 FE 02 20 EE C9
```

beállításával érhetjük el, hogy hibátlanul olvassa a HT a szalagot. Többféle időállandóval is kipróbáltam a programot, és kétféle HT-n is. Azt tapasztaltam, hogy az egyik HT-n — az általam is jobb magnójúnak tartotton — mindkét időállandóra egy-egy intervallum adódott, amelyeken belüli tetszőleges értékkel hibátlanul olvasott. A másik HT-n nem lehetett beállítani az időállandókat. Az általam kapott értékek átlaga: TIME1 = 40 (Dec) és TIME2 = 75 (Dec).

Nem akarom elhallgatni azt sem, hogy még a jobbik HT-vel sem tudtam a legelső blokkot elolvasni, azaz a program neve titokban maradt. Ám ha az időállandókat csökkentettem, az első blokkot is elolvasta, kiírta a nevet, de a programot olvashatatlanul vette le. Így először el lehet olvasni a program nevét, majd az időállandókat beállítva elolvashatjuk magát a programot is.

A *2. program* egy bájtot olvas az A regiszterbe, az 1. program felhasználásával. A *3. program* a szinkronizáció miatt kell: megvárja, míg egy blokk érkezik. Ezzel a három alapprogrammal már bárki írhat egy ABC programszalag-olvasót.

A *4. program* 7000 kezdőcímre van szervezve. A program feltételezi, hogy az EDI is a memóriában van. Használata a következő. 7000 címről indítva bekapcsolja a belső magnót és olvassa a szalagot. Ha talál ABC programot, akkor a 8000 címtől kezdve tárolja, és a képernyő első negyedében is látható az, amit a memóriában letárol. Ha úgy látjuk, hogy értelmetlenség, akkor módosítsunk az időállandókon. Olvasás közben a program figyeli, hogy az utolsó blokkot olvasta-e be. Ha igen, kikapcsolja a magnót, letörli a képernyőt és listázni kezd. Az A, B, C, D, E, F, G, H billentyűk bármelyikének lenyomására listáz egy sort, majd újra megnézi a billentyűzetet. A V billentyű hatására az EDI-be ugrik, innen akár újra is lehet indítani. Segítségül néhány cím:

7000 a program indítása
702F a listázás megkezdése
704A az EDI címe, ezt átírva a V betűre nem az EDI-be megy
704E egy bitet olvas
706E egy bájtot olvas
7076 egy blokk adatbájtjait olvassa
7087 megvárja a blokk elejét

SZOLDATICS JÓZSEF

A rovatszerkesztő megjegyzése. A Tudományszervezési és Informatikai Intézetnek van egy olyan illesztője, amelynek segítségével ABC 80-ról HT 1080Z-be át lehet vinni BASIC programállományokat. Az intézet kívánságra szívesen végez másolást szabványos HT kazettára. Természetesen az eltérő BASIC miatt itt is szükség van némi utólagos programmódosításra.

Elsősorban primósoknak

A matematikafakultáció keretében sokan tanulják az integrálszámítást. Házi feladataik készítését megkönnyítheti az *1. listán* látható program.

A program működése során megkérdezi, hogy a 70-es sorban megadott F függvényt mely intervallumon integrálja. Ezt követi a számolás alsó és felső közelítéssel. Amikor a két közelítés eltérése kisebb egy adott értéknél, akkor a gép befejezettnek tekinti a számolást és kiírja az így kapott végeredményt.

A programban a következő változókat használtam:

— A és B: az intervallum alsó, illetve felső határa
— N: az adott intervallumban a részintervallumok száma (a számolás folyamán tart a végtelenhez)
— H: egy részintervallum nagysága

```
10 REM * INTEGRALSZAMITAS *
20 CLS
30 INPUT"AZ INTERVALLUM HATARAI";A,B
40 N=1:E=6E-3
50 H=ABS(A-B)/N
60 FOR X=A TO B STEP H
70 TA=TA+H*F(X):TF=TF+H*F(X+H)
80 NEXT X
90 TA=INT(TA*1000)/1000
100 TF=INT(TF*1000)/1000
110 IF ABS(TF-TA)<=E THEN 130
120 TF=0:TA=0:N=N+10:GOTO 50
130 PRINT"A GORBE ALATTI TERULET:";
140 PRINT (TF+TA)/2"EGYSEG"
150 END
```

1. lista

```
10 REM * GYORS KORRAJZOLAS *
20 CLS
30 INPUT"A KOR SUGARA";R1
40 INPUT"A KOR KOZEPPONTJA";OA,OB
50 K=95/R1:Z=R1/100
60 FOR I=1 TO 629 STEP K
70 X=A(I)*Z+OA:Y=B(I)*Z+OB
80 SET(X,Y):NEXT I
90 END
```

2. lista

3. lista

```
10 REM * MAX.KOR KOORDINATAI *
20 DIM A(630),B(630)
30 FOR I=0 TO 2*PI STEP 0.01
40 J=J+1
50 A(J)=COS(I)*95:B(J)=SIN(I)*95
60 NEXT I
```

— E: a két közelítés eltérésének értéke (minél kisebb, annál pontosabb a számolás)
— TA, TF: az alsó, illetve felső közelítés

Mint már említettem, az adott függvényt a 70-es sorba kell beírni az F(x), illetve F(x + H) helyére, tehát a független változót kell H-val növelni.

A *2. lista* egy gyors körrajzoló program. Számos BASIC programban lehet szükség kör rajzolására, ezért jól szolgálhat egy gyors változat. A módszer jelentősen csökkenti a körrajzolásra fordított időt, ezért olyankor célszerű alkalmazni, amikor szerepe van a futási időnek. Hátránya azonban, hogy több helyet foglal el a memóriában, mint a hagyományos körrajzoló program.

Ez a program a következők alapján működik:

— Az A(I), B(I) változókban tárolja a lehető legnagyobb kör koordinátáit, és a későbbiekben ezeknek a segítségével — egy szorzással és egy összeadással, ami jóval gyorsabb a szögfüggvények számolásánál — kiszámolja az aktuális kör koordinátáit, és kirajzolja a képernyőre.

A program változói a következők:

— A(I), B(I): a maximális nagyságú kör koordinátái
— Z: az adott kör sugarának századrésze
— OA, OB: az adott kör középpontja (X és Y koordinátája)
— K: megadja, hogy az adott kör sugara hányszor fér bele a legnagyobb kör sugarába. (Figyeljük meg, minél kisebb az adott kör sugara, annál gyorsabban rajzolja ki. Ezt a K változó teszi lehetővé. Mivel a sugár csökkenésével egyre kevesebb pontból áll a kör, ezért ezt a változót használom arra, hogy csak a szükséges pontok legyenek kirajzolva.)

A program elindítása előtt természetesen az A(I) és B(I) változókat fel kell tölteni az ismert módszerrel. Ha valaki mégsem ismerné, megtalálhatja a *3. listán.*

A programokról még annyit, hogy Primo számítógépen készültek. Az integrálszámítás minden további nélkül átírható bármely más gépre, a körrajzoló azonban nem biztos, hogy mindenütt ugyanolyan hatással jár.

NÓGRÁDI ELŐD

Elektronikus zene C64-en

Ez a rövid program egyszerűsége ellenére sokat tud. Futtatása során bárki ízlésének megfelelő elektronikus zenét komponálhat.

A mű háromtételes lesz (30-as sor), és nagyjából követi a szimfóniák tempóviszonyait. A zenedarab úgy jön létre, hogy ujjunkkal röviden megérintünk egy billentyűt. Ezt a zenészek staccatónak nevezik. Az egyes billentések között tetszésünknek megfelelően 2-4 másodpercet várjunk. Aki másféle hangzást is szeretne hallani, annak közlök néhány módosítási lehetőséget.

```
10 S=54272:FORI=0TO24:POKES+I,0:NEXT
20 POKES+24,143:POKES+17,8:POKES+6,250
30 FORT=1TO3:PRINTT"TETEL":POKES+15,T-1
40 FORG=22TO56 STEP2:POKES+4,17
50 POKES+18,16:POKES+14,2.4*G:Q=3*G
60 FQ=Q+PEEK(S+27)*3.5*G:F=INT(FQ/256)
70 A=FQ-F*256:POKES,A:POKES+1,F
80 IFPEEK(203)<>64THEN100
90 GOTO60
100 GETT$:IFT$=""THEN100
110 POKES+4,16:NEXT:NEXT
```

A zene hangfekvését a 60-as sorban a G szorzószám határozza meg. Ez 0,5-4 lehet. A hangzás tempóját az 50-es sor POKES + 14 értéke befolyásolja: (1—5) × G-ig változhat. A tételek tempóit a 40-es sor ciklusának kezdő- és végértéke határozza meg: 18—60-ig terjedhet. A felsorolt értékek kölcsönösen befolyásolják egymást.

KALMÁR GYULA

Nyitott rendszer -nyitott egyetem

Apró kis lámpákkal jelölve az ország középiskoláit és szakmunkásképző intézeteit, érzékelhetővé tehetjük egymástól való távolságukat. Meglehet, ránézve e különleges fénytérképre, gondolatban a távolságok oly csekélyek, szinte semmik, lebecsülhetők. De ott lenni, tanulni és tanítani egy-egy iskolában és onnan megtudni valamit is a másikról, a másiktól, nem könnyű. Pedig ennek létfontossága vitálisan eltagadhatatlan. Egy iskola sem próbálkozhat telepátiával — gondolatátvitellel —, nem is annak tudományos vitathatósága miatt. Egyszerűen nincs idő arra, hogy véletlenszerű kísérletezéssel próbálkozzanak, amikor egy-egy központi, vagy saját információs éléskamrájuk tele van, vagy éppenséggel üresen tátong. Persze akármelyik iskolában mondhatják: nagyon jól megvannak, semmire sincs szükségük. Csak ez tipikusan olyan, hogy ha nem tud, nincs ismerete valamiről az embernek, talán hiányt sem szenved tőle. Egy-egy információ azután így kallódik el, így nem válik közkinccsé, és gyakran olyan balvégzetű lehet, mint az a nagyszerű találmány, amely sutba dobva éri meg végnapját — meghaladja a tudomány, a technika —, s mire felfedeznék, legfeljebb már múzeumi tárggyá avul.

Kézenfekvő tehát, hogy ezeket az iskolákat érdemes és össze kell kapcsolni valamilyen módon egymással, s olyan intézményekkel is, melyek műhelytanulmányaikhoz sok segítséget nyújthatnak. A KISZ KB Középiskolai és Szakmunkástanuló Tanácsa a közelmúltban ennek mintegy alapkövét tette le, amikor létrehozott egy adatbázis információs rendszert. Erről a rendszerről Kajos Lászlóval, a tanács titkárságvezetőjével beszélgettünk.

— Minden középfokú oktatási intézményben van már számítógép. Mi erre építve kívánjuk hozzáférhetővé tenni azokat az információkat, amelyek az iskola, a KISZ-élet mindennapjaiban hasznosíthatók, a pedagógusokat tájékoztatják, s a diákok is ennek révén szinte beszélgethetnek egymással az ország legtávolabbi pontjaiban e rendszeren keresztül. Vagyis, az információs adatbázisból lehívhatják az őket érdeklő témákat. Adatbázisunk jelenleg négy információcsoportból áll. Tartalmazza az ország valamennyi középfokú iskolájának pontos adatait; az iskolai innovációk leírását a fejlesztők megnevezésével, továbbá ezen fejlesztések helyi adottságokra, viszonyokra adaptálható eredményeinek pontos leírását. Ezen kívül magában foglalja a középfokú oktatási intézmények működésére vonatkozó jogszabályokat, melyeket sajnálatos módon a pedagógusok többsége nem, vagy csak hiányosan ismer. A negyedik csoportba a KISZ-szervezetek működéséről szóló információkat gyűjtöttük össze. Így az érvényes központi, illetve megyei testületi döntéseket, valamint a politikai képzés gyakorlati tapasztalatait dolgoztuk fel.

— Kik vesznek részt az információs adatbázis megalkotásában?

— Munkánkban közreműködik a Művelődési Minisztérium és az Országos Pedagógiai Intézet. A közös adatállomány szándékunk szerint a legkülönfélébb innovációs fejlesztésekkel — ezek az iskolákból érkeznek be hozzánk — megalapoz egy műszaki fejlesztést szolgáló adatbázist. Éppen azáltal, hogy bármely ilyen jellegű műszaki gondolat, terv az érdeklődő diák birtokába juthat.

— Mi az adatbázis technikai igénye?

— Nálunk üzemel már egy IBM kompatibilis gép. A közeljövőben valamennyi megyeszékhely KISZ-bizottságára ugyanilyen típusú mikroszámítógépet kívánunk elhelyezni azért, hogy a mi központi adatbázisunkat tetszés szerint egy az egyben lehívhassák. Persze ezzel még az iskolába nem juthatna el egyetlenegy információ sem. Az akusztikus modemekkel viszont direkt kapcsolat kiépítése nélkül, telefonon lesz elérhető a megyeközpontok KISZ-bizottságain felhalmozott információ-adathalmaz. Természetesen minden olyan iskolai számítógéphez használható a modem, amelynek RS-232-es kimenete van. Az olyan számítógépek, amelyeknek ilyen kimenete nincs, csak megfelelő interfészen keresztül csatlakoztathatók a modemhez.

— Honnan szerezhető be egy modem? Azaz miképpen kapcsolódhatnak be ebbe a rendszerbe?

— Az akusztikus modemet a KISZ KB-tól vásárolhatják meg 19 ezer 500 forintért. A készülék iránt a 203-872-es telefonon lehet érdeklődni. Egyébként, ha máshonnan szerzik be a kis készüléket, az ára jóval borsosabb. A rendszerbe kapcsolódáskor csak a modem jelent költséget, ugyanis az információ lehívása ingyenes. Viszont szinte alapszabállyá szeretnénk tenni azt, hogy az iskolák ne csak információt vegyenek, hanem adjanak is magukról, hiszen ezáltal fejlődhet, gazdagodhat csak adatbázisrendszerünk.

— Milyen időszakonként újul meg a rendszerben lévő ismeretsokaság?

— Folyamatosan gondozzuk az információkat, így a hozzánk beérkező újdonságok gyakoriságától függően félévente, havonta, hetente frissítjük fel az állományt.

— Jóllehet az országos hálózat alakulóban van, gondoltak-e már a továbbfejlesztésére?

— A kialakítottal párhuzamosan működne egy, a postával üzemeltetett videotext rendszer Mupid berendezésekkel, illetve Mupid funkciókra alkalmassá tett IBM kompatibilis gépekkel. Ezen keresztül az Ipari Minisztérium, a Mezőgazdasági és Élelmiszeripari Minisztérium, az Országos Műszaki Fejlesztési Bizottság által fejlesztett adatbázisok is elérhetők. Ezzel egy olyan oda-vissza áramló információs csatornahálózat születik, amely lehetővé teszi a régóta várt távoktatást. A Magyar Televízió, a Központi Statisztikai Hivatal és a Neumann János Számítógéptudományi Társaság egymással karöltve készíti elő a nyitott egyetem módozatait is. Ezek izgalmas, érdekes kérdések lesznek a jövőben.

— A videotext rendszerbe való bekapcsolódás is ingyenes szolgáltatásokkal jár?

— Nem. Az imént felsorolt intézmények adataiért, leírásaiért, tájékoztatásaiért fizetni kell majd, de ennek árát csak később állapítják meg.

— krasznai —

PRIMO-apróság

Különböző „gyors lábú" programok írásakor elengedhetetlen egy olyan GOTO/GOSUB utasítás, amely nem kötött sorszámra vagy számlistára vonatkozik, hanem egy megadható aritmetikai kifejezés szolgáltatta célsorszámra ugrik. Az alábbi kis programocska erre kínál igen egyszerű és általánosan használható megoldást.

Az egyetlen kikötés, hogy ha a célsorszám 32767-nél nagyobb, akkor 65536-ot kell levonni belőle. A lista alábbi formájában 64 k-s gépeken működik, azonban az első POKE utasítás címét (−6640) kisebbre választva, és kétbájtos alakját a második POKE utasítás adataiként beírva a többi (48 k-s és 32 k-s) gépen is használhatjuk. A program szempontjából az „A" vagy „B" sorozatú gépek egyenértékűek. Működése igen egyszerű, voltaképpen azt teszi, amit egy „jólnevelt" GOTO/GOSUB feldolgozó program sorszámbeolvasó része tesz:

— meghívja a 2337H-n kezdődő kifejezéskiértékelő rutint, amely a HL által mutatott címen kezdődő tokenizált kifejezést dolgozza fel (PRIMO irodalomban EXPRN néven ismert), majd az eredményt az „X" munkaregiszterbe helyezi el;

— egész részt számol, ezt követően az eredményt mint sorszámot veszi, és végrehajtja az ugrást. (A GOSUB rész természetesen előbb a verembe helyezi a következő utasítás címét.)

A lista mellé íme még egy csemege. Ha a GOTO vagy a GOSUB utasításokat sorszám nélkül adjuk ki, a gép általában UL Error-t jelez, kivéve, ha van Ø sorszámú sor. Ez esetben ugyanis a program elindul, vagyis ilyenkor a Ø-t célsorszámként értelmezi.

```
1 POKE−6640,14,3,205,99,25,193,229,229,
      42,162,64,227,62,145,245,51,197,
      229,205,55,35,205,127,10,229,209,
      225,195,200,30
2 REM — Itt írtuk be a programot —
3 POKE 16561,10,230: CLEAR 50
4 REM — A BASIC-rendszer számára
      fenntartott memória végének lej-
      jebb hozása. Erre programunk vé-
      delmében van szükség, különben
      igen könnyen „elpárologhat".
      (Lásd veremkezelés és egyéb érde-
      kességek.)
5 POKE 16768,33,230,195,16,230
6 REM — A RAM ugrótábla részében a
      GET és PUT utasítások átírása —
7 REM — A továbbiakban GOTO x helyett
      GET numerikus kifejezés használ-
      ható —
8 REM — GOSUB x helyett PUT numeri-
      kus kifejezés alkalmazható —
9 REM — A régi GOTO/GOSUB utasítá-
      sok természetesen „érvényben ma-
      radnak" —
10 REM — CDOS és rokonai használatá-
      nál csak CDOS betöltése után, és
11 REM ne az általa foglalt területre írjuk
      be. Ez kiolvasható
12 REM a 116561-es címen található két-
      bájtos mutató értékéből —
13 REM — Más címre helyezéskor ne csak
      a memória végét jelző mutatót,
14 REM hanem a végrehajtási kezdőcímet
      is állítsuk be helyesen. —
15 PRINT chr$(12) „Speci GOTO/GO-
      SUB él és élni akar kikapcsolásig"
16 END
```

— liling —

Hét kicsi kecske

A diákszerkesztőség összejöveteleit színesítendő, felnőtt kollégáinktól a következő rejtvényt kaptuk azzal, hogy próbáljuk számítógépesen gondolkodva megoldani. Íme:

Éldegélt valahol hét kicsi kecske. Egyszer kaptak három liter tejet, melyet valaki valahogyan (de semmiképpen sem egyenlően) szétosztott közöttük. De ők jószívűek voltak, és igazságosan akarták maguk között elosztani a kedvelt italt. Emígyen szólt az első: — Nem szeretem a tejet — persze csak az önzetlenség beszélt belőle —, szétosztom hát köztetek. S a szót tett követte. Ezután hasonló állítással jelenkezett a második kis kecske is, ő úgyszintén szétosztotta a tejet. Követte őt sorban a harmadik, a negyedik, az ötödik, a hatodik, végül pedig a hetedik is. Poharaikban ekkor mindegyiküknek ugyanannyi teje volt, mint kezdetben.

A kérdés: melyiknek mennyi?

A feladatot természetesen meg lehet oldani matematikai módszerekkel is. Ez is érdekes lehet, minket azonban az a megoldás érdekel, amelyhez számítógépre van szükség. Ennek programját várjuk tőletek, a hozzá tartozó magyarázattal együtt.

Diákszerkesztőség

TOP-lista

A külföldi szaklapok példáiból okulva mi is közlünk TOP-listát, két kategóriában. Az elsőben a felhasználói, a másodikban a játékprogramok szerepelnek. A legjobbakat, a legérdekesebbeket soroljuk a beküldött javaslatok alapján. Ehhez kérjük az olvasók közreműködését. C64-re, ZX—Spectrumra, Enterprise-ra, TVC-re, ATARI-ra és IBM-re készült programok rangsorait várjuk havonta.

Címünk: 1371 Bp., Pf.: 433 Mikroszámítógép Magazin Szerkesztősége.

A Diákszerkesztőség

Felhasználói programok:

1. Gigacat	C64
2. Art Studio 1.2B	C64+ZX—Spectrum
3. ZX—Screen machine	ZX—Spectrum
4. Speech+	C Plus/4
5. Easy Script	C64
6. Paint Box	ZX—Spectrum
7. History of Europe and Hungary	C64
8. Híres—Master	C64
9. Window Wizard	C64
10. Beta Basic	ZX—Spectrum

Játékprogramok:

1. Defender of Crown	C64
2. The last ninja	C64
3. Alternative world games	C64
4. Sabotour	C64+ZX—Spectrum
5. Zaxxon	C64+ZX—Spectrum
6. Impossible Mission	C64
7. Game Over/trainer	C64+ZX—Spectrum
8. Commando	C64+ZX—Spectrum
9. California Games	C64
10. Quest for Tires	C64+ZX—Spectrum

E havi TOP-listánkat a fővárosi Hámán Kató Közgazdasági Szakközépiskola diákjainak szavazatai alapján állítottuk össze.

Kilencárnyalatú nyomtatás

A program színes, nagyfelbontású (160 x 200) képeket készít az MPS 1000-es nyomtatón. Viszonylag kis módosítással bármely olyan nyomtatóra átírható, amelyik elérhető a soros buszon, képes duplasűrű nyomtatásra és van 8 tűs grafikus üzemmódja.

Minden képernyőpontnak 4 x 4 pixel felel meg a nyomtatón, így a teljes kép 640 x 800 pixel, ami mintegy 22,5 x 17 centiméter. (A duplasűrű nyomtatás miatt a 800 pixel feleakkora helyen fér el.)

Mivel egy képernyőpont 16 pixel a papíron, elvileg 17 különböző árnyalat nyomtatható. A gyakorlatban azonban a sötétebb árnyalatok között túl kicsi a különbség, ezért a 16 szín 9 árnyalatban jelenik meg:

1. fehér	0
2. sárga-világoszöld	1
3. cián-világosszürke	2
4. zöld-világospiros	3
5. világoskék-középszürke	4
6. lila-narancs	6
7. piros-sötétszürke	9
8. kék-barna	12
9. fekete	16

pixel sötét a 16-ból. Így pontosan olyan képet kapunk, mintha fekete-fehér tv-n néznénk a rajzot.

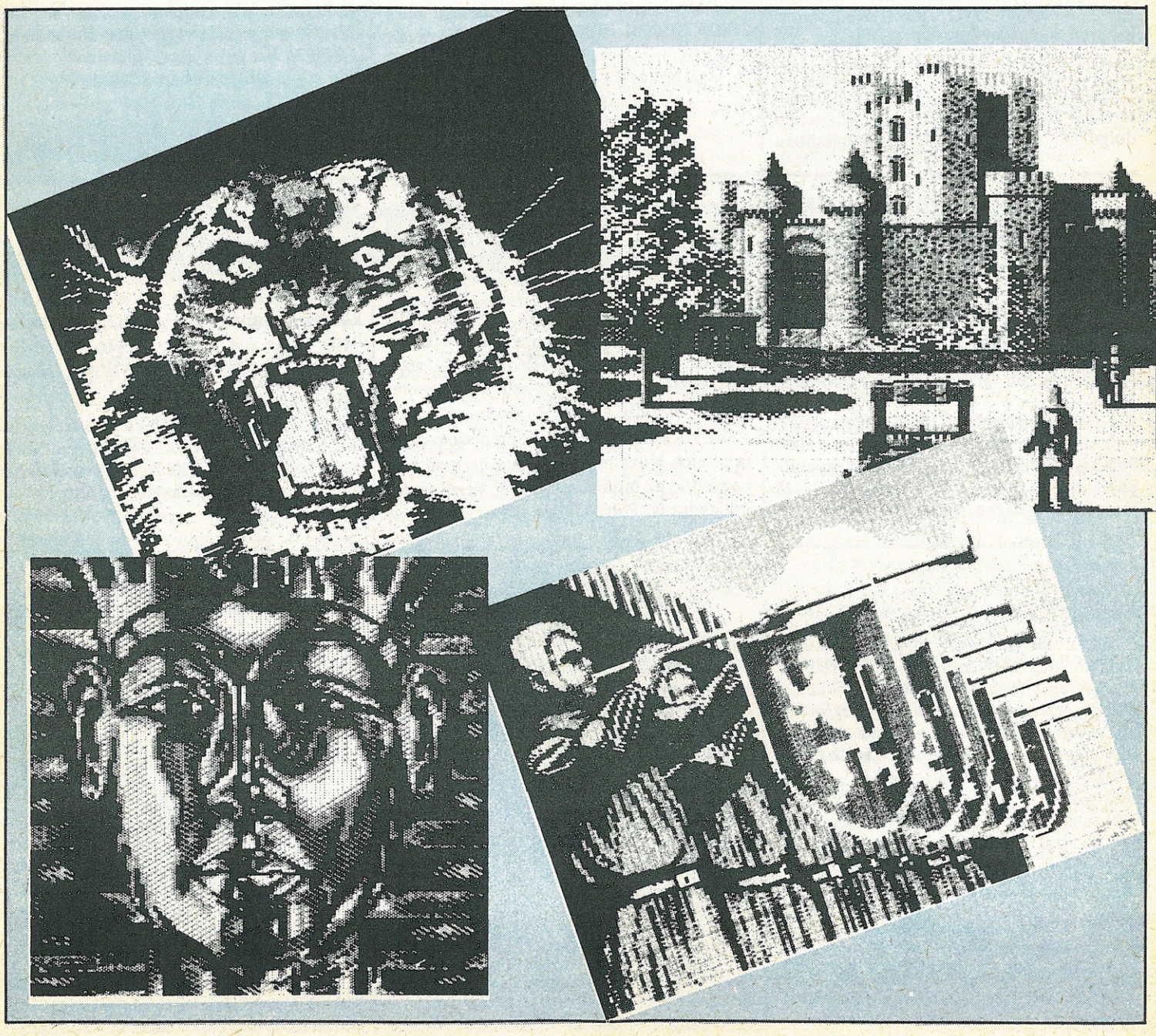
A grafika a következőképpen helyezkedik el a tárban:

$2000-$3F3F: ponttároló
$3F40-$4327: szín-RAM + 1
$4329: háttérszín ($D021)
$4338-$471F: szín-RAM + 2 ($D800-$DBE7)

Ezt használja egyébként az ART STUDIO 1.2 rajzolóprogram is.

Egyetlen kép kinyomtatása mintegy 5 percig tart.

DEIERL ANDRÁS—
MOLNÁR IMRE



```
10 ON A GOTO 20,90
15 A=1 : LOAD"PRINT12.EXE",8,1
20 POKE53265,27:POKE53272,21
21 POKE53270,200:POKE53280,6
22 POKE53281,6:PRINT"■";
23 PRINT"■ BETOLTES UTAN:■"
24 PRINT"    L: UJ KEP BETOLTESE■"
25 PRINT"    P: NYOMTATAS■■■"
30 INPUT"■■■A KEP NEVE";N$
40 N$=LEFT$(N$+"                ",12)
60 A=2 : LOADN$+"MPIC",8,1
90 SYS 51545 : A=1
100 GET A$: IF A$="" THEN 100
110 IF A$="L" THEN 20
120 IF A$="P" THEN SYS51200: GOTO 20
130 GOTO 100
```

1. lista

2. lista

```
*       = $c800
setnam  = $ffbd ; az op.rendszer
setlfs  = $ffba ; rutinjai
open    = $ffc0
chkout  = $ffc9
chrout  = $ffd2
clrchn  = $ffcc
close   = $ffc3
fol     = $8b   ; a fociklus valtozoja:
foh     = $8c   ; vizszintes lepes
all     = $8d   ; az alciklus valtozoja:
alh     = $8e   ; fuggoleges lepes
n       = $8f   ; egyeb segedvaltozok
z       = $f7
t       = $f8
c1      = $f9
c2      = $fa
s       = $fb
al2l    = $fc
al2h    = $fd
fo2l    = $f3
fo2h    = $f4
        jsr  init  ; a rajzot a kepernyore
        lda  53281 ; a hatterszin
        and #15    ; (mindenutt
        sta  szint ; ugyanaz)
        lda #0
        jsr  setnam; file-nev nincs
        ldy #255
        ldx #4
        txa
        jsr  setlfs
        jsr  open  ; OPEN 4,4,255
        ldx #4
        jsr chkout ; output a nyomtatora
        lda #0
        sta  n
        sta  z
        lda #1984> ; a bal also sarok a
        sta  fo2h  ; szinram#1-ben
        lda #1984<
        sta  fo2l
        lda #15879>; a bal also sarok a
        sta  foh   ; ponttarolóban
        lda #15879<
        sta  fol
        lda #27    ; soremelés a nyom-
        jsr  chrout; tón : 24/216 inch
        lda #51    ; (8 pixel)
        jsr  chrout
        lda #24
        jsr  chrout
sor     ldx #0     ; minden sor elején:
lp1     lda  szam1,x;soremelés, dupiasú-
        jsr  chrout; rú nyomtatás be,
        inx        ; 800 pixel/sor
        cpx #5
        bne  lp1
        lda  foh   ; al = fo
        sta  alh
        lda  fol
        sta  all
        lda  fo2h
        sta  al2h
        lda  fo2l
        sta  al2l
        jsr  ujszin; az aktuális szinek
        ;           beállítása
oszlop ldy #0
        lda  (all),y
        sta  s
        lda #4     ; 4 pixeloszlop/kép-
        sta  t     ; ernyőpont
        ;
        lda  n     ; egy output-byte
        beq  felso ; elöállitása
        ;
        asl  s
        asl  s
        asl  s
        asl  s
        ;
felso   asl  s
        rol  c1
        asl  s
        rol  c1
        asl  s
        rol  c2
        asl  s
        rol  c2
        ;
        lda  c1
        and #3
        tax
        lda  szint,x
        asl a
        asl a
        tax ;  x:=c1*4
        lda  c2
        and #3
        tay
        lda  szint,y
        asl a
        asl a
        tay ;  y:=c2*4
        ;
pont    lda  tabl,x
        asl a
        asl a
        asl a
        asl a
        ora  tabl,y
        ;            az output-byte el-
        jsr  chrout; küldése: két képer-
        ;            nyőpont
        dec  t ;a képernyőponthoz tarto-
        ;      zó következő pixeloszlop
        beq  ujpont
        ;      még nincs meg mind a négy
        inx
        iny
        jmp  pont
        ;
ujpont inc  z ; lépés felfelé
        lda  z
        and #7
        sta  z
        ;
        beq  sokat
        ;
        lda  all
        sec
        sbc #1
        sta  all
        lda  alh
        sbc #0
        sta  alh ; al=al-1
        jmp  ell
        ;
sokat   lda  all
        sec
        sbc #57
        sta  all
        lda  alh
        sbc #1
        sta  alh  ; al=al-313
        lda  al2l ; lépés felfelé a
        sec       ;  szinram#1-ben
        sbc #40
        sta  al2l
        bcs  tov1
        dec  al2h ; al2=al2-40
tov1    jsr  ujszin;az aktuális szinek
        ;           beállítása
ell     lda  alh
        cmp #$20 ; al < $2000 ?
        ;           (új oszlop?)
        bcc  ford
        jmp  oszlop
        ;
ford    lda  n ;ugyanaz az oszlop megint
        eor #1 ;(egy byte a ponttároló-
        sta  n ;ban = 4 pixel)
        ;
        beq  nem1
        jmp  sor; új sor a nyomtatón
        ;
nem1    lda  fol ; fo=fo+8 (következő
        clc      ;       byte a ponttárolóból)
        adc #8
        sta  fol
        lda  foh
        adc #0
        sta  foh
        inc  fo2l;következő byte a szin
        bne  ell2;ram-ből
        inc  fo2h
        ;
ell2    cmp #$3f ; utolsó oszlop után?
        beq  nem2
        jmp  sor
nem2    lda  fol
        cmp #$40
        bcs  nem3
        jmp  sor
        ;
nem3    lda #13    ; nyomtatás vége:
        jsr  chrout; nyomtató lezárása
        jsr  clrchn
        lda #4
        jsr  close
        rts ; program vége
        ;
ujszin ldy #0
        lda  (al2l),y
        pha
        lsr a
        lsr a
        lsr a
        lsr a
        sta  szint+1
        pla
        and #15
        sta  szint+2
        lda  al2l
        sta  ciml+1 ; a program onmagát
        lda  al2h   ; módositja!
        clc
        adc #$d4
        sta  ciml+2
ciml    lda  $d800  ; byte olvasása a
        and #15     ; szinram#2-ből
        sta  szint+3
        rts
init    lda #$04   ; szinram#1 a helyére
        ldx #$40 ; lo
        ldy #$3f ; hi
        jsr  pakol
        lda #$d8   ; szinram#2 a helyére
        ldx #$38
        ldy #$43
        jsr  pakol
        lda  $4329 ; háttér és keret
        sta  53281 ; szinének beállitása
        sta  53280
        lda #59    ; grafika be
        sta  53265
        lda #29    ; ponttároló $2000-on
        sta  53272
        lda #216   ; négyszinü üzemmód
        sta  53270
        rts
        ;
pakol   sta foh    ; általános másoló-ru-
        stx fo2l   ; tin
        sty fo2h
        ldy #0
        sty fol
        ldx #4
lp2     lda  (fo2l),y
        sta  (fol),y
        iny
        bne  lp2
        inc  foh
        inc  fo2h
        dex
        bne  lp2
        rts
szam1   .byte 13,27,76,32,3
tabl    .byte 15,15,15,15,15 ;      fekete
        .byte 0,0,0,0  ;            fehér
        .byte 10,5,10,5  ;          piros
        .byte 0,4,0,1 ;             cián
        .byte 8,5,2,5    ;          bíbor
        .byte 8,1,4,0 ;             zöld
        .byte 7,13,11,14 ;          kék
        .byte 0,2,0,0 ;             sárga
        .byte 10,4,10,1  ;          narancs
        .byte 14,11,13,7 ;          barna
        .byte 0,2,9,1 ;             v piros
        .byte 5,10,5,10  ;          s szürke
        .byte 1,4,2,8    ;          k szürke
        .byte 0,0,4,0 ;             v zöld
        .byte 8,2,4,1    ;          v kék
        .byte 8,0,2,0 ;             v szürke
szint   .byte 0,0,0,0;az aktuális szinek
```

Árvábbnál is árvább Kis magyar

Egy kisfiú ül nyilvánvaló elégedettséggel egy számítógép előtt, felénk fordított mutató- és középső ujjával jelet formál: *Victoria!* — Ez látható a hajdan gyártott Primo ízléses reklámfüzetének címoldalán.

Győzelem? Vagy bukás? Krúdy Gyula *Álmoskönyve* szerint az ujjakkal játszani fiatalnak szomorúság. Ha a Primót, a fiatalon hamvába holt gépet mint vállalkozást nézzük, a győzelem valóban csak álom volt, s a valóság szomorkás. És mégis, a Primo nem *ad acta tény*, a Primo sokezerszeresen van és él. Iskolai és házi számítógépként helyet követel magának. Éppen ezért e cikk sem *rekviem*, hanem *látlelet, körkép, áttekintés* a Primo továbbéléséről és *némiképp visszatekintő diagnózis* féligsikerültségének körülményeiről.

Anamnézis

— Hány Primo készült összesen? – kérdezem az MTA—SZTAKI Cosy leányvállalatnál (amely menedzselte és forgalmazta ezeket a gépeket) *Pogrányi Károlyt*, aki már a megszületésénél is bábáskodott.

— Összesen hétezer. Ebből ezer nyomógombos billentyűzettel. Ezek mindegyike fekete-fehér televízió használatát tette lehetővé. Ezután készült el a Pro/Primo, színesre.

— Ami nyert az iskola-számítógépes pályázaton, de végül nem lett belőle semmi. Megrendelésük egyáltalán nem volt az iskolák részéről?

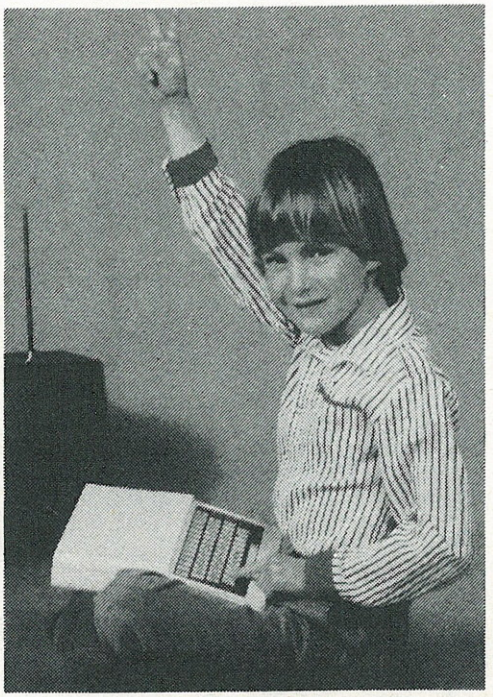
— Tudomásom szerint az iskolák rendeltek két-két és fél ezret, de amikor a Commodore is szóba jött, azonnal átálltak arra. Hadd tegyem hozzá, hogy mi *nem eleve kudarcra rendezkedtünk be*, hiszen már a Pro/Primo idején továbbfejlesztettük ezt a családot. Elkészítettünk egy 256 kbájtos változatot is, ami azóta is nagyon jól működik nálunk.

— Arra számítanak, hogy jobb körülmények között még egyszer induljanak ebben a számítógépes kategóriában?

— Nekünk elsődlegesnek kell tekintenünk a gazdaságossági követelményeket, ezért kizártnak tartom ezt. Másrészt szerintem a közelebbi-távolabbi jövőben az iskolákban is az IBM PC-ké a jövő, mert ezek a gépek intelligens párbeszédre való képességükkel *alkalmasabbak* a kisebbeknek és a kamaszodóknak a *számítógépes társalgásra, tanulásra.*

— Volt azonban néhány kellemetlen tapasztalatuk is a primósoknak, ami talán befolyásolta a gép megítélését.

— Ez tagadhatatlan. A perifériák nem jelentek meg az alapgéppel egy időben. Sajnos a programkészletünk is akkor bővült gyorsabban, amikor már *a Primo kátyúba esett.* Végül tizenöt programot forgalmaztunk, a többi kiadatlan maradt.



— S ha valaki ma állít be azzal, hogy itt a programom, hasznosítani szeretném?

— Ha behozza a törzsanyagot, ad hozzá programleírást és valóban jó az a program, segítünk neki ebben.

— Vállalkozó nem jelentkezett a Primo gyártására, mióta leálltak vele?

— De, egy gmk szerette volna gyártani, csak nem tudta megvásárolni ennek a jogát. Egyébként a gép kapcsolási rajza megjelent az egyik primós könyvben, kis ügyességgel bárki felépítheti.

Látod?...! Lelet!...?

Az út a Cosyból nem Rómába vezet, hanem Budapesten a Rákóczi úti Triál hangszerboltba. Egyedül itt lehet — ha lehet — állami üzletben új Primo programokat venni.

— Nekünk ez egyáltalán *nem üzlet.* A múlt évben is alig volt néhány ezer forintos forgalmunk programeladásból — mondja *Szili Mihály* üzletvezető. — De nem is tartozik a profilunkba. Annak idején *félig-meddig felkérésre, szívességből vállaltuk az árusítást.*

— Reklámozták a lehetőséget?

— Nálunk nem. Ha a Cosyhoz ment, ide küldték, vagy ha hozzánk betévedt valaki, vásárolhatott. Bár ehhez nekünk már semmi *érdekünk* nem fűződik, olyan mint *hátunkon a púp.*

Tájkép csata után

Kevés program, reklám semmi, és az országban egy üzlet. A pesti szleng erre azt mondja: ez nem semmi! A *képtelenséget még tetézi,* hogy a Primo életes életében sem volt jobb a helyzet. Mert ha például Szombathelyen vett valaki gépet, akkor is Pestre kellett utaznia programért!!! Gép és programok. Az előbbieket mintha a föld nyelte volna el. Vajon többet megtudok-e róluk a Tudományszervezési és Informatikai Intézetben?

— 1986-ban az általános és középiskolákban 1263 Primo volt. A kiárusításkor még két-háromszázat vettek az iskolák — néz a statisztikába *dr. Varga András* főosztályvezető.

— Közelebbit tudnak ezeknek a gépeknek a használatáról?

— Tulajdonképpen alig-alig. A gépeket *a megyék osztják el,* a továbbiakban *visszajelzést nem kapunk.* A programkészítési pályázatunkra is csak két-három pályamunka érkezett.

Confused view

A hegy csúcsáról nézve tehát a Primók világa holtágnak tűnik. Vannak, csak *a mozgásuk nem látszik.* Az Országos Pedagógiai Intézetben *dr. Szűcs Barna* sem látja a mélyben a mozgást.

— Sajnos tudomásom szerint nálunk sem vizsgálták, kutatták az iskolai Primók helyzetét. Alkalmazásukkal sem foglalkozott senki. (Ez sem semmi! — A pesti szleng.) Egy-egy iskolai jelentésből kiderül: náluk van Primo, hány van. Néhány megjegyzést fűznek csak hozzá. Én a primós dolgot amolyan „kis magyar tragédiának" tartom. Mert ami azt illeti, *jól példázza* a hardveresek és szoftveresek külön világát, az *összehangoltságuk hiányát.* Ugyanakkor az is érthetetlen, nálunk szinte törvényszerűen miért bukik bele a vállalkozásába az, aki jót és jó szándékkal kezdeményez. Bár a Cosy is többet tehetett volna a sikerért. *Nekem levelezésem van* róla, hogy kezdeményeztük egy bemutató laboratórium Primóval való felszerelését, de ebben *ők kicsinyesek voltak,* nem adtak felszerelést.

Elfogyott a hamuban sült pogácsa

Néhány bizonytalan tippel a tarsolyomban, mondván: talán abban az iskolában

a PRIMO tragédia...?

találni Primókat — de *inkább hallomás alapján* kerestem fel a Dob Utcai Általános Iskolát. Hátha ott igazi primós életre lelek! Amit találtam (az *általánosítás* nem szándékom, de *valószínűsíthető*), nem a legrózsásabb. A számítógépes teremben az asztalokon Commodore-ok és dobozban a magukra hagyott Primók. Dr. Appel Györgyné matematika—technika szakos tanárnő a számítógépes szakkör vezetője:

— Én nem a Primókon tanítom a gyerekeket, de tudom, hogy *a Neumann Társaságtól hetente jön egy hölgy,* aki az alsósokkal foglalkozik. Neki, de nekem is az a legnagyobb gondom, hogy a gépeket csatlakozóvezetékek nélkül kaptuk. Így igen nehéz boldogulni. A Primókat egyébként *egyszer használták másra,* a gyerekek történelemtudásának félévi értékelésekor. De nem is lehet egyébre, mert csak egy programjuk van.

Haláltánc

Ezt hívják úgy, hogy az információ halála? Vagy magárahagyatottság? Netán a magad ura, magad szolgája vagy jelenségnek? Ilyen árvábbnál is árvább gyerek a Primo? Ez *lehetetlen.* Még akkor is, ha *minden jel* arra mutat. Vagy van más jel is talán, mint a hiányzó, hiányos információk? Megelőlegezem — a Gelkánál rábukkanok a *várva várt.* Nyirő Árpád, a Fehérhajó utcai szerviz vezetőhelyettese konkrétumokkal szolgál.

— Mit tudnak nyújtani a Primo-tulajdonosoknak, lehetséges talán, hogy a gépeket akár továbbfejlesszék?

— Volt eredetileg az *A* és *B* típusú gép. Az *A*-nak billentyűzete szenzoros és nem volt semmilyen plusz kivezetése, a *B* változatnak viszont kiépítették a hátsó csatlakozását, és billentyűs. Ezt bővítettük egy úgynevezett *C* változattal, ami ezeken kívül azt a lehetőséget nyújtja, hogy Commodore perifériákkal működik, a kazettára beolvasott program nélkül. Ezenkívül *numerikus billentyűzetet adhatunk hozzá,* továbbá bármilyen RAM-bővítést megcsinálunk, illesztünk, de ezek már egyedi munkák.

— Ezek szerint ha valaki bejön valamiféle egyedi kívánsággal, akkor segítenek rajta?

— Igen. Eddig a leginkább keresett persze a *C* változat volt, s *amit még nem említettem,* botkormányt is két változatban illesztünk. Az egyik külső doboz, a másik gépbe épített. A két csatlakozót is felszereljük.

— Összesen hétezer gép van, évente hányan jönnek akár mert meghibásodott, akár hogy bővíttessék a masinájukat?

— Az adataink *nem egészen pontosak,* mert hosszú időn keresztül *egyedül mi javítottuk* ezeket a gépeket. Jelenleg hozzávetőlegesen 40-50 Primót javítunk. Tavaly év végén 30-40 C változatot csináltunk.

— Az alkatrészellátás zökkenőmentes?

— Eddig igen. Mostantól valószínűleg lesz a szenzoros billentyűzetű gépekkel gondunk.

— A Cosy nem látja el a szervizt fóliával?

— Itt a baj. A fóliatasztatúra fogyóban van, és nem biztos, hogy lehet még kapni. A Cosy nem nagyon foglalkozik ezzel, ők elküldenek a gyártóhoz, Szolnokra. Máskülönben *annak idején* a Primónak *az volt az előnye,* hogy csak olyan alkatrészekből építették, amelyek Magyarországon beszerezhetők voltak. *Ez a helyzet ma is,* mert minden alkatrész szocialista gyártmány. Úgy tűnik, még elég hosszú ideig lesznek üres nyomtatott áramköri lapjaink is. Ha tehát valakié tönkremegy, azt is sokáig kicserélhetjük.

— Mik a típushibái a Primónak?

— Csak a sorozattól függőek, az egészre jellemző nem fordult elő egy sem. Az első szériában sok forrasztási hiba volt, ezért rengeteg RAM ment tönkre. Ezenkívül a magnóra panaszkodtak nagyon sokan.

— Szervizelésük, vállalkozásuk veszteséges vagy nyereséges?

— Először úgy tűnt, hogy nagyon veszteséges lesz, mert átalányszerződést kötöttünk. Aztán időközben kiderült, hogy nem is olyan rossz ez a gép! Máig is *nyugodt lelkiismerettel vallom,* ez a gép jó. Így némi kis nyereséggel dolgozunk.

— Javító szakember szemével nézve mennyire becsüli a Primo élettartamát?

— Óriási különbség van a szenzoros és a nyomógombos billentyűzet között, az utóbbi javára. Egyértelműen a *billentyűzetet rontották el a Primón.* A nyomógombos szerintem jól bevált. Ettől függetlenül a nyomógombos gépeknek az élettartamát előre nem lehet pontosan meghatározni. Öt év múlva még biztosan lesznek, működnek. A szenzorosnak viszont nem garantált a műanyaglap-utánpótlása. Átalakításuk nem műszakilag megoldhatatlan, hanem a billentyű beszerezhetetlen. Elvileg elképzelhető gyártót találni rá, mégsem hiszem, hogy bárki is vállalná. Egyébként nagyon könnyű lenne az átalakítás. *Véleményem szerint* ezt az egész Primo-ügyet *valahol elhibázták.* Ez nagyon jó kis gép, amivel az iskolákat elláthatták volna. Eltekintve a programhiánytól, szerintem egyenrangú a Commodore-okkal, sőt, az alapgépet jobbnak is találom azoknál.

Kórisme helyett

Eddig tartott a bolyongás. Ha a látszat csalt, az csakis a Primo és a primósok javát szolgálhatja. Mert bár egy álmoskönyv sem mondhat rosszabbat, mint amit bolyongásom közben éberen tapasztalhattam, sejthető, érezhető, hogy *hétezer Primo úgymond lélegzik, noha szinte láthatatlan.* Ezért hiszszük, minden róla alkotott elmarasztaló ítélet csalhatatlansága sem csalhatatlan.

SZULYOVSZKY CSABA

Budapesten az alábbi GELKA-szervizekben foglalkoznak Primo gépekkel:

V., Fehérhajó utca 8—10.
VII., Rózsa Ferenc utca 10.
VIII., Kisfaludy köz 4.
Vidéken:
0203. sz. szerviz, Pécs, Ybl Miklós u. 3.
0606. sz. szerviz, Szeged, Ságvári u. 6.
0930. sz. szerviz, Debrecen, Segner tér 1.
0401. sz. szerviz, Békéscsaba, István király tér 9.
0804. sz. szerviz, Győr, Arany János u. 25.

Az idei évkezdetkor kazettahiány miatt nem forgalmaztak programokat a Primo gépekhez. Az első negyedévben az elmaradást pótolták, így a programok kaphatók a Cosynál és Budapesten a VII. kerület, Rákóczi út 60. sz. alatti TRIÁL boltban.

Az Elektromodul (Bp. XIII., Jászai Mari tér 5.) a következő tartozékokat forgalmazza:

- **DCD PRT asztali grafikus nyomtató 62 011,— Ft**
- **DCD PRT—42 Baby print grafikus nyomtató 30 150,— Ft**
- **lemezmeghajtó 14 380,— Ft**
- **optikai billentyűzet 8 654,— Ft**

Az Elektromodulnál a vásárláshoz előzetesen (három-négy héttel számolva átfutási időként) kell leadni az igényeket, hogy a terméket legyárthassák.

A Baby print grafikus nyomtatót a Kispesti Áruházban 23 990 forintért árulják.

A szerkesztőség — támogatva a primósok táborát — ezúton is felhívja a figyelmet a Primo klubra, amely által lehetőségük nyílik iskoláknak és magánszemélyeknek egyaránt a géppel (gépükkel) kapcsolatos gondok megoldására. A klub (Bp. VII., Almássy tér 6. Foglalkozások: minden hónap 2. hétfőjén. Vezető: Somogyi György) várja az új belépőket!

Programkatalógus

Egy programon 300 Primóra írt program katalógusához lehet hozzájutni! A HCC Primo szakosztálya által elkészített programismertető behívható a számítógépbe. A klub a Primóknál használt különböző magnetofonok miatt a beolvashatóságot nem garantálja. Akinek a program kell, egy kazettát, egy válaszborítékot juttasson el szerkesztőségünkbe, s a magazinon keresztül megkapja a katalógust. Ízelítőül a 300 ismertetőből egy minta:

A program neve: Táltos
Azonosító száma: 197
Készítője: Répás Sándor
Kipróbálva: A64, A48 típusokon
Hossza: 14 blokk
Memóriaigénye: 20,1 kbájt
Ügyességi játék, „SAVE" parancscsal másolható.
Két ember párbajozik a képernyőn. Az egyiket a gép, a másikat a játékos vezérli billentyűkkel.

A Mikroszámítógép Magazinnak és a Magyar Televízió Mi és a számítógép című műsorának egyaránt a számítástechnika népszerűsítése a célja. Rendszeresen foglalkoznak olyan témákkal, amelyeket a mikroszámítógép világából merítenek.

A magazin helyet ad olyan témáknak, amelyek ugyan bekerültek a Mi és a számítógép műsorba, de olvasva is izgalmasak, illetve teret enged a televíziós műsor szerkesztőjének, aki a lap hasábjain kíván foglalkozni olyan kérdésekkel, amelyek helyszűke miatt nem fértek bele a harmincperces adásidőbe.

Talán az olvasók előtt sem titok — sokan élnek is a lehetőséggel —, hogy a Mi és a számítógép műsor egyik nagy attrakciója, sőt kuriózuma a néhány hónapja bevezetett újdonság: az adás végén kazettára rögzíthető és onnan a számítógépbe tölthető programokat sugárzunk a tévé hangcsatornáján. Ezek között is a legnépszerűbbek a rejtvényprogramok, amelyek Commodore és Sinclair gépekre íródtak. A jövőben folytatni kívánjuk — a rejtvénypályázat befejeztével is — ezt a kezdeményezést a legelterjedtebb géptípusokra. Ehhez várjuk a nézők és a magazin olvasóinak ötleteit, és természetesen programjaikat.

A számítástechnikai műsorunkban továbbra is foglalkozunk a számítógép és a video kapcsolatával, illetve a két szakma határterületeit érintő témákkal. Ízelítőül annyit, hogy például a közeljövőben bemutatjuk a képlemezjátszó és a számítógép összekapcsolását — az ún. interaktív video lehetőségeit — valamint azt, hogy mindezt miként hasznosítják az oktatásban és a tájékoztatásban.

Továbbra is beszámolunk a környező országok szakmai kiállításainak újdonságairól, és természetesen bemutatunk egy-egy sikeresen működő — vagy éppen gondokkal küszködő — számítógépes klubot, szakkört, iskolát.

HEGYI ISTVÁN

Számítógép és művészet

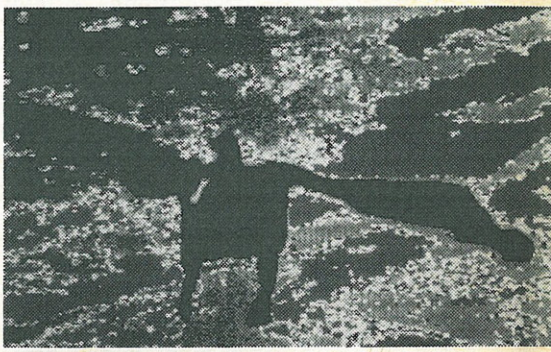
Ma még vitatott kérdés, hogy önálló műfaj-e a számítógépes művészet. Szerencsére az alkotók közül többen a szócséplés helyett inkább az új technikával ismerkednek, kísérleteznek. E szárnypróbálgatók egyike, Waliczky Tamás, a Fiatal Művészek Stúdiójának tagja így beszél munkájáról:

Művészi pályám szorosan összekapcsolódik a számítógéppel. Korábban sokat festettem, fotóztam, rajzoltam, s volt néhány kiállításom is. A stúdióba aztán már számítógépes alkotásaimmal pályáztam.

Hogyan kezdődött számítógépes pályafutásom? Öt esztendeje – mint szellemi szabadfoglalkozású grafikus – illusztrációkat, rajzfilmeket és reklámfotókat, majd egy elektronikus játékhoz rajzfilmet készítettem. Két évig Commodore 64-en dolgoztam. Sajnos a kép minősége arra alkalmatlan volt, hogy alkossak vele, így hát folytattam megszokott életem, vagyis festettem, fotóztam, s készültem a következő kiállításaimra. Aztán hozzájutottam egy Atari 520-as géphez, melynek grafikus képe jobb és színei sokkal szebbek, mint a Commodore-é. Felhagytam a játékkészítéssel, s áttértem a grafikai programokra. Ezekkel megannyi demonstrációs rajzot, grafikát, mozgást, animációt hoztam létre. Munkámban segítőtárs volt az ART DIREKTOR rajzolóprogram. Mind jobban elmerültem a számítógépes technikában, igazán élveztem a munkát.

A munkám is megváltozott, amióta számítógéppel dolgozom. A manualitásban persze nincs nagy különbség. Ma is táblákon rajzolok, mintha rajzlapra rajzolnék ceruzával. A másság akkor kezdődik, amikor a rajzokkal manipulálok, mert a kifejtés helyett ott vannak a számítógépes eszközök. Ezek gyorsabb, tökéletesebb technológiát tesznek lehetővé. És az animációs munkában is sokat segítenek a programok. Így a fázisrajzoló programmal két fázis közé megrajzolhatom az átmeneteket úgy, hogy a nehézkes számításokat helyettem a gép végzi. Mindemellett egy korábbitól teljesen eltérő művészeti termék jön létre gyorsan, szépen és fantasztikus látványosan.

Mostanában előszeretettel alkalmazom a számítógépet arra, hogy végtelenítve vetítse a rajzokat. Ez fontos, mert olyan cselekményeket választok, amelyek végtelenszer ismétlődnek, s ez az állandó mozgás – a képzőművészetnek ez a műfaja az ún. mobil – ad a nézőknek egyfajta új élményt. Továbbfejlesztett változata ennek az, amikor egy kiállítóteremben üzembe helyezünk mondjuk húsz monitort – mintha azok képek, festmények lennének –, s ezeken folyamatosan ismétlődik a cselekmény . . .



Vízimadár 1986

E technikának számomra másik – és nem elhanyagolható – előnye az, hogy az információt kimentem, abból semmi sem vész el. Egy későbbi időpontban ismét kapcsolatba kerülhetek a rajzzal, javíthatom, tökéletesíthetem, ha egyszer más koncepció szerint kívánok még dolgozni rajta. Gyorsan, pontosan, anyagmegtakarítással! Ha az ember ügyesen bánik a gép kínálta lehetőségekkel, mindenfajta őrültséget kipróbálhat a monitoron, amelyeket esetleg papíron, vásznon hagyománytiszteletből nem tenne meg.

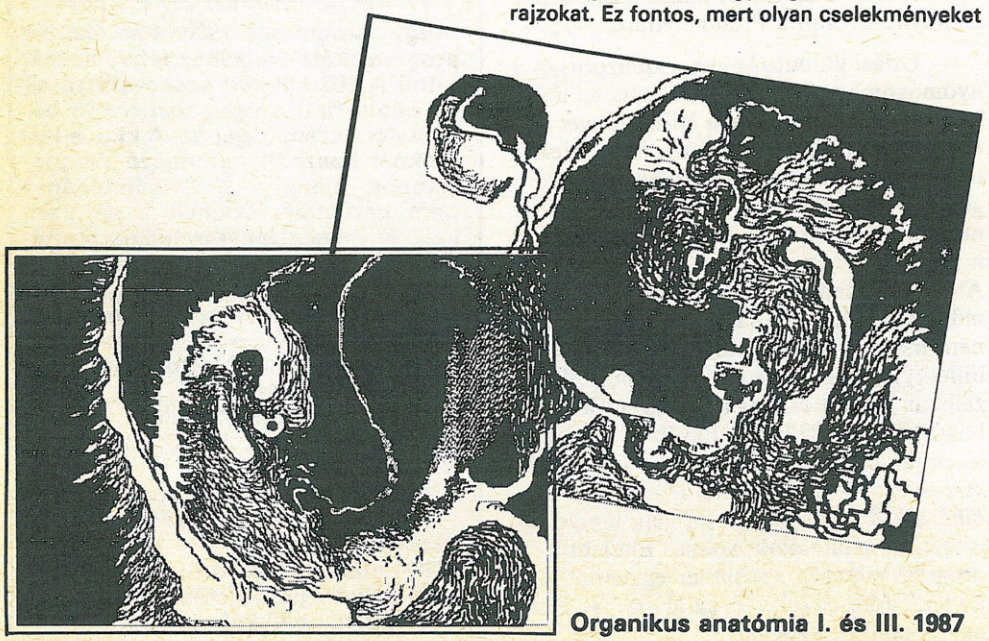
Tudok Angliában valakiről, aki absztrakt festményeit számítógépen tervezi. Addig játszik a színekkel, amíg a maximális hatást el nem éri, s csak ezután festi vászonra a képeit. Műalkotásában mindenesetre a számítógépé az elsőbbség.

Miért nem népszerűbb a számítógépes művészet a jelenleginél? Talán mert sokan félnek a korszerű technikától, tartanak attól, hogy nem képesek a gépet kezelni. Pedig például az ART DIREKTOR program, noha meglehetősen bonyolult, mindössze két óra alatt megtanulható annyira, hogy dolgozni kezdhessünk vele.

A számítógéppel egészen másfajta képek születnek, mint a hagyományos festészetben. Mégis azt hiszem, e másság a művészetben teremtő erővel hódíthat.

(Képeinken Waliczky Tamás néhány műve)

H. I.



Organikus anatómia I. és III. 1987

KOMPLEX SZÁMÍTÁSTECHNIKAI SZOLGÁLTATÁS!

A Ferroglobus számítástechnikai főosztálya nagy adatfeldolgozási hagyományokkal, jól képzett, nagy tapasztalatú munkatársakkal rendelkező számítóközpontja széles skálájú, komplex számítástechnikai szolgáltatásait ajánlja:

— kereskedelmi szakmai, készletgazdálkodási, pénzügyi, számviteli, bér- és munkaügyi stb. rendszerek szervezése, programozása és fejlesztése nagy kapacitású, modern ICL típusú számítógépen
— mágneses adathordozóra történő adat-előkészítés — rögzítés
— számítógépidő bérbeadása
— tanácsadás

Ferroglobus

TEK V. számítástechnikai főosztály
Budapest VII., Vörösmarty u. 16.
Tel.: 427-338, 202-415

BASIC és gépi kód

Legutóbb az USR függvény lehetőségeit kiterjesztő rutinok működésének részleteiről írtam. Utaltam néhány problémára is, melyeknek megoldására most megpróbálok receptet adni.

A BASIC nyelvű betöltőprogramoknál egy új módszert alkalmaztam, melynek lényege, hogy a betöltendő adatokat hexadecimális karakterláncokban tárolom. A módszer előnye a tömör kód, hátránya a hosszú betöltési idő.

A múltkori 7. listába kellemetlen hiba csúszott, melyért ezúton kérek elnézést. A C16-on a DATA sorokat véletlenül átszámoztam, és ezt nem vettem észre. Betöltés közben nem a programsor száma, hanem a sor feldolgozásánál elsőként betöltött adat címe jelenik meg a képernyőn. Ez hibátlan adatok esetén nem jelent problémát, de egy-egy előforduló hiba megkeresését nagyon megnehezíti. A most közölt betöltőprogramoknál már megszüntettem ezt a hibát.

A problémák

A gépi kódú rutinoknak a kazettapufferban való elhelyezése kényelmes, de komoly gondokat okozhat. Lássunk ezek közül néhányat.

A nyomtatásban megjelenő gépi kódú rutinok nagy része a kazettapufferba van lerakva. Ezekkel nem lehet a mi rutinunkat együtt használni, mert egyidejűleg ugyanazt a helyet vennék igénybe. Ha ki akarjuk próbálni azt a tömbelemeket összegző szubrutint, mely *A Commodore 64-es belső felépítése* című könyv 94—96. oldalain található, le kellene mondanunk a DEF USR által nyújtott lehetőségről.

A betöltőprogram viszonylag hosszú átfutási ideje arra késztet, hogy a rutint gépi kódú alakban tároljuk. Akinek csak kazettás háttértárolója van, annak erre nincs lehetősége. Nagyobb gond, hogy a rutin aktivizálása után a magnó egyáltalán nem használható. Bármilyen adatátviteli próbálkozás — az „önző programozás" eredményeképpen — a rendszer „lefagyásához" vezethet.

A megoldás egyszerűnek látszik: tegyük a programot máshová. Igen, de hova? Ha egy meghatározott helyet választunk, ismét az első problémába ütközünk: egy másik szubrutin szintén oda kívánkozik.

Próbáljuk meg oda tenni, ahol éppen szabad hely van! Ekkor viszont a programbeli abszolút címekkel lesz baj, jelen esetben a két feldolgozórutin belépési pontjának és a táblázat kezdetének a címe nem fog egyezni. Írjuk át ezeket a ténylegesre. Hogy ez miként lehetséges, arra láthatunk példát a mostani három programlistán.

A megoldás

A *listákon* az áthelyezhető rutinok betöltőprogramjai láthatók. Az új programsorok beszúrásától eltekintve, csak néhány apróbb változtatás történt az előző számban közölt programokhoz képest, hogy kevés gépeléssel elvégezhető legyen a módosítás. A 110-es és 200-as sorban a 828 helyére a kezdőcímet tartalmazó KC változó került. A FOR L=1 TO 15 utasítás mögül eltűnt a legutóbb bajt okozó PRINT, szerepét a következő sor után beszúrt új PRINT utasítás vette át. Ez valamivel hosszabb, de mindig a legutóbb olvasott DATA sorszámát írja ki. A C16-os változat DATA utasításait a RENUMBER 828,12,828 parancscsal átszámoztam, de ez a művelet elhagyható.

1. lista

```
100 rem usrdef-2 c64
105 z=256 : kc=49152
110 k=kc : print chr$(147)
120 def fnd(x)=x-48+7*(x>64)
130 for l=1 to 15
140 s=0 : read a$,b
145 print chr$(19);peek(63)+z*peek(64);
150 for i=1 to len(a$) step 2
160
a=16*fnd(asc(mid$(a$,i,1)))+fnd(asc(mid$
(a$,i+1,1)))
170 poke k,a : s=s+a : k=k+1 : next
180 if b<>s then print "hiba!" : stop
190 next
191 for i=1 to 6
192 read d,a : a=kc+a : b=int(a/z)
193 poke kc+d,a-z*b
194 poke kc+d+1-(i<3),b
195 next
200 sys kc : end
828 data a951a0038d08038c0903a99e,1044
840 data a0038d0a038c0b0360207300,714
852 data c996f0062079004ce7a72073,1371
864 data 00c9b7f0034cb3b385022073,1343
876 data 00f068b008290f0a85022073,876
888 data 00c9b2d05a207300208aad20,1199
900 data f7b7a602300a9ddb03989dda,1562
912 data 034ceaa78d1203988d11034c,1031
924 data eaa7a900850d207300c9b7d0,1455
936 data 0f207300f029900ea57ad002,1098
948 data c67bc67a2079004c8dae290f,1241
960 data 0a4820730020f1ae68a8b9da,1351
972 data 038555b9db0385564ce0af4c,1398
984 data 08af48b248b248b248b248b2,1433
996 data 48b248b248b248b248b2,1250
998 data 1,21,11,98
999 data 79,159,83,158,143,158,148,159
```

2. lista

```
100 rem usrdef-2 vc20
102 z=256
104 kc=peek(55)+z*peek(56)-176
106 b=int(kc/z)
108 poke 55,kc-z*b : poke 56,b
110 k=kc : print chr$(147)
120 def fnd(x)=x-48+7*(x>64)
130 for l=1 to 15
140 s=0 : read a$,b
145 print chr$(19);peek(63)+z*peek(64);
150 for i=1 to len(a$) step 2
160
a=16*fnd(asc(mid$(a$,i,1)))+fnd(asc(mid$
(a$,i+1,1)))
170 poke k,a : s=s+a : k=k+1 : next
180 if b<>s then print "hiba!" : stop
190 next
191 for i=1 to 6
192 read d,a : a=kc+a : b=int(a/z)
193 poke kc+d,a-z*b
194 poke kc+d+1-(i<3),b
195 next
200 sys kc : end
828 data a951a0038d08038c0903a99c,1042
840 data a0038d0a038c0b0360207300,714
852 data c996f0062079004ce7c72073,1403
864 data 00c9b7f0034cb3d385fe2073,1627
876 data 00f066b008290f0a85fe2073,1126
888 data 00c9b2d058207300208acd20,1229
900 data f7d7a6fe300a9dd903989dd8,1842
912 data 034ceac785029885014ceac7,1442
924 data a900850d207300c9b7d00f20,1101
936 data 7300f029900ea57ad002c67b,1372
948 data c67a2079004c8dce290f0a48,1034
960 data 20730020f1ce68a8b9d80385,1435
972 data 55b9d90385564ce0cf4c08cf,1507
984 data 48d248d248d248d248d248d2,1692
996 data 48d248d248d248d2,1128
998 data 1,21,11,96
999 data 79,157,83,156,141,156,146,157
```

```
100 rem usrdef-2 c16
105 z=256 : kc=4096
110 k=kc : print chr$(147)
120 for l=1 to 15
130 s=0 : read a$,b$ : b=dec(b$)
135 print chr$(19);peek(63)+z*peek(64)
140 for i=1 to len(a$) step 2
150 a=dec(mid$(a$,i,2))
160 poke k,a : s=s+a : k=k+1
170 next
180 if b<>s then print "hiba"
190 next
191 for i=1 to 6
192 read d,a : a=kc+a : b=int(a/z)
193 poke kc+d,a-z*b
194 poke kc+d+1-(i<3),b
195 next
200 sys kc : end

828 data a951a0038d08038c0903a99b,411
840 data a0038d0a038c0b0360207304,2ce
852 data c996f0062079044cd98b2073,535
864 data 04c9b7f0034c9d9a85e82073,5fa
876 data 04f065b008290f0a85e82073,453
888 data 04c9b2d0572073042 0e19da6,581
900 data e8300a9dd803989dd7034cdc,5d1
912 data 8b8d0205988d01054cdc8ba9,4a6
924 data 00850d207304c9b7d00f2073,41b
936 data 04f029900ea53bd002c63cc6,535
948 data 3b2079044c1e94290f0a4820,280
960 data 730420859468a8b9d7038555,52d
972 data b9d80385564cec954ca1941c,5d9
984 data 991c991c991c991c991c991c,43e
996 data 991c991c991c99,2b8
998 data 1,21,11,95
999 data 76,156,80,155,140,155,145,156
```

3. lista

A címek átírását a 191 ... 195 sorokban lévő programrész végzi, a módosító adatok az utolsó két DATA sorban vannak. Ezeket nagyon figyelmesen gépeljük be, mert nincs hozzájuk ellenőrző összeg.

A címátíró programrész működési elve a következő.

Minden módosításhoz két adat tartozik, ezek a betöltött rutin kezdetéhez viszonyított relatív címek. Az első (D) azt mutatja, hogy a módosított cím alacsonyabb helyértékű bájtját hová kell beírni, a második (A) pedig azt, hogy a kezdőcímhez mennyit kell hozzáadni, hogy a valós címet kapjuk. Ennek alsó és felső bájtját egy ciklusléptén belül írjuk be a megfelelő helyekre. Az első két esetben — a vektorcímeket átíró szubrutinban — ezek nem egymás mellett vannak, köztük van az LDY kódja.

géptipus	programhossz
C64	178 bájt
C16	175 bájt
VC20	176 bájt

1. táblázat

2. táblázat

	c-64	c-16	vc-20
sgn	48185	41662	56377
int	48332	41816	56524
abs	48216	41693	56408
fre	45949	39522	54141
pos	45982	39549	54174
sqr	49009	42468	57201
rnd	57495	42759	57492
log	47594	40990	55786
exp	49133	42592	57325
cos	57956	43632	57953
sin	57963	43639	57960
tan	58036	43712	58033
atn	58126	43802	58123
peek	47117	40442	55309
len	46972	40289	55164
str$	46181	39782	54373
val	47021	40339	55213
asc	46987	40304	55179
chr$	46828	40123	55020
dec		40475	
hex$		46343	

A 100-as és 110-es sorok közé beszúrt utasításokkal jelöljük ki a gépi kódú program helyét. A listák közötti eltérés kivételesen nem azt jelenti, hogy a különböző géptípusokon eltérő módszert *kell* alkalmazni, hanem párhuzamosan mutat be három megoldási lehetőséget.

A C64-es változatban egy szabad területen helyezzük el programunkat, a $C000 címtől kezdve. Ez a kazettapufferhez hasonlóan „népszerű" hely, ezért előfordulhat, hogy egy másik helyet kell választanunk. A két másik géptípuson ez a módszer csak korlátozottan alkalmazható a megfelelő méretű, összefüggő szabad RAM terület hiánya miatt.

A VC20-as változatnál a BASIC RAM tetejéből vágjuk le a szükséges területet. A 104-es sor végén álló szám a program mérete, az egyes géptípusokhoz tartozó programhosszakat az *1. táblázatban* foglaltam össze. Ez a módszer mindhárom géptípuson jól alkalmazható, de a 64 kilobájtra bővített C16-oson nem ajánlatos a $8000 ... $FFFF közötti RAM területre gépi kódú rutint tenni.

A C16-osra írt programban a BASIC RAM elejéből vágjuk le a megfelelő méretű helyet. Sajnos — jelenlegi ismereteink szerint — ezt a műveletet a BASIC program beírása, illetve betöltése előtt minden alkalommal parancsmódban kell elvégezni. Itt erre a POKE 43,176 : POKE 4721,0 : NEW utasítássorozatot alkalmazhatjuk. A módszerről további információk találhatók Énekes Ferenc *Programönkritika is ...* című cikkében, mely a Magazin 1986. októberi számában jelent meg.

Próbáljuk ki!

Jó lenne kipróbálni, hogy jól működik-e a programunk, de egyelőre nem nagyon akad olyan gépi kódú függvényünk, melyet USR-rel hívhatnánk. Alkalmas erre a BASIC legtöbb beépített függvénye is, ha ezek belépési címeit adjuk meg a DEF USR utasításokban.

A *2. táblázatban* összegyűjtöttem néhány ilyen címet. Akinek kedve van, próbálkozzék velük. Figyelem! Van néhány olyan függvény, melynek a paramétere, és néhány olyan, amelyiknek az eredménye karakterlánc típusú. Ez a megfelelő USR függvénynél is így van.

A táblázatból kiolvasható, hogy a C64 és a VC20 típusok függvényei közül hiányzik a DEC és a HEX$. Már nem sokáig, mert rövidesen jelentkezem az őket helyettesítő felhasználói függvényekkel.

Még egyszer a BRK-ról

Tavaly novemberben kaptam kézhez Varga Csaba szegedi olvasó sorait. A következőket írta:

„Az 1987/11. szám programozástechnika rovatban, a Basic és gépi kód című cikkben a BRK utasítás egy fontos jellemzője lemaradt. A 68XX család ezt az utasítást 2 bájtosnak értelmezi, azaz az utasítás dekódolása után a PC 2-vel növekszik. Ennek az a hatása, hogy a BRK utasítást végrehajtva az egy belső megszakítást generál, és azt végrehajtva az RTI utasítás után a visszatérés nem közvetlen a BRK után, hanem 2 bájttal utána történik."

Olvasónk észrevétele helyes, bár a jelenség leírása kissé pontatlan. A BRK említett tulajdonságáról azért nem írtam, mert sorozatomban nem tudok minden apró részletre kitérni. Úgy gondolom, hogy a novemberi számban ismertetett utasításcsoport a profi programozók kelléktárába tartozik, ők pedig nem az én cikkeimből ismerik meg a gépi kódú programozás rejtelmeit. A veremkezelő utasításokat akkor csak a teljesség kedvéért ismertettem.

A BRK említett tulajdonságát célszerű azoknak is figyelembe venni, akik valamilyen monitorprogramot használnak gépi kódú programjaik írására, tesztelésére, például a C16 beépített monitorját. Az ő szempontjukból viszont mindegy, hogy az említett jelenség a BRK-nak vagy a monitornak tulajdonítható.

Megköszönve Varga Csaba bíráló megjegyzését, ígérem, hogy a témát továbbra is napirenden tartom, és — anyaggyűjtés után — újra jelentkezem vele.

BARNA LÁSZLÓ

Számítógépes grafika Pascalban

ZX—SPECTRUM

MOZGÓ KÉPEK TEKNŐSGRAFIKÁVAL

E sorozat előző cikkében bemutattunk egy módszert geometriai alakzatok síkbeli transzformációinak programozására. Most egészen más megközelítését ismertetjük ugyanennek a problémának. A megoldás módszere a teknősbéka-grafika.

Ez a grafika mint ötlet eredetileg a LOGO nyelv megalkotásakor merült fel, és azt a célt szolgálta, hogy akár kisiskolás gyerekek is játszva megtanulhassák a programozás alapjait. A számítógép képernyőjén megjelent egy pici teknősbéka képe, amelyet egyszerű parancsokkal előre-hátra lehetett mozgatni, jobbra-balra forgatni, és a farkát felemelni, leereszteni. Ha a béka farka lent volt, mozgás közben nyomot hagyott a képernyőn. Nos, az ötlet kitűnően bevált. Olyannyira, hogy komolyabb alkalmazásokra is kezdték használni.

Több Pascal implementációba is beépült a teknősgrafika, természetesen nem parancsok, hanem eljárások formájában. Így például híres az Apple számítógépek teknősgrafikája, de az IBM PC család Turbo Pascal programja is tartalmaz teknőseljárásokat. A Hisoft Pascalhoz is tartozik egy forrásnyelvű fájl, amely TURTLE (=teknős) fájlnév alatt szerepel, és amellyel kevesen tudnak mit kezdeni. Az alábbiakban ennek alkalmazására mutatunk egy példát.

A TURTLE nevű fájl önmagában nem teljes értékű program, hiszen sem eleje, sem vége nincs. Vagy be kell közvetlenül építenünk programunkba, vagy — mint e cikkben is látható —, include-fájlként hívjuk meg a főprogramból. Ha kilistázzuk a fájlt, egy deklaráció részletét látjuk meg először. Itt az olyan globális változók vannak megadva, melyeket a teknőseljárások használnak. Ezek az XCOR, YCOR: a „teknősbéka" pillanatnyi helykoordinátái, a HEADING: a teknős fejének iránya, mind Real változó, valamint a PENSTATUS integerváltozó, melynek értéke 0, ha a toll ír és 1, ha a tollat felemeljük.

Azoknak az eljárásoknak a katalógusa, amelyek ismerete a programozáshoz szükséges, a következő:

INK C:INTEGER
— a tinta színét állítja be

PAPER C:INTEGER
— a papír színét állítja be

```
  10 PROGRAM MozgokepPointer;
  20 {1987. julius kabdebx P.}
  30 {$L-}
  40
  50 TYPE KEPHARMAD=ARRAY[0..7,0..7,0..31]OF CHAR;
  60      FILM=RECORD
  70             MUTATO:^FILM;
  80             KOCKA:KEPHARMAD;
  90           END;
 100
 110 VAR I,J:INTEGER;
 120     ELOZO,KOV,FUTO,FUTO1:^FILM;
 130
 140
 150 {$L+}
 160 {SF 1:TURTLE}
 170 {$L-}
 180 {$F 1:TURTLE}
 190
 200 PROCEDURE FOG(A,SZOG:REAL);
 210 BEGIN
 220   FWD(A);LEFT(75-SZOG);
 230   FWD(3*A);RIGHT(75);
 240   FWD(2*A);RIGHT(75);
 250   FWD(3*A);LEFT(75-SZOG);
 260   FWD(A);
 270 END; {FOG}
 280
 290 PROCEDURE FOGASKEREK(N,X,
 300   Y:INTEGER;A,R:REAL);
 310 BEGIN
 320     SETXY(X,Y);
 330     PENUP;
 340     VECTOR(A,-R);
 350     PENDOWN(7);
 360     LEFT(90);
 370     FOR J:=1 TO N DO
 380       FOG(1.5,180/N);
 390 END; {FOGASKEREK}
 400
 410 BEGIN
 420   ELOZO:=NIL;
 430   FOR I:=0 TO 5 DO
 440   BEGIN
 450     PAGE;
 460     WRITE(CHR(18),CHR(1));
 470     INK(0);WRITE(CHR(22),CHR(10),
 480       CHR(9),' V A R J O N ');
 490     WRITE(CHR(18),CHR(0));
 500     PAPER(7);
 510     FOGASKEREK(16,35,142,
 520       10-360/18/6*I,20);
 530     FOGASKEREK(10,69,142,
 540       360/10/6*I,20/1.8);
 550     FOGASKEREK(18,109,142,
 560       10-360/18/6*I,20);
 570     FOGASKEREK(6,146,156,
 580       35+360/6/6*I,20/3);
 590     FOGASKEREK(6,164,147,
 600       -360/6/6*I,20/3);
 610     FOGASKEREK(14,191,134,
 620       5+360/14/6*I,7/9*20);
 630     FOGASKEREK(11,225,154,
 640       10-360/11/6*I,11/18*20);
 650     NEW(KOV);
 660     KOV^.MUTATO:=ELOZO;
 670     KOV^.KOCKA:=PEEK(16384,KEPHARMAD);
 680     ELOZO:=KOV
 690   END;
 700   FUTO1:=KOV;
 710   REPEAT
 720     FUTO:=FUTO1;
 730     FUTO1:=FUTO^.MUTATO
 740   UNTIL FUTO1=NIL;
 750   FUTO^.MUTATO:=KOV;
 760   PAGE;INK(0);
 770   WRITE(CHR(22),CHR(0),CHR(0));
 780   FOR I:=0 TO 10 DO
 790   FOR J:=1 TO 32 DO
 800   WRITE(' ');
 810   REPEAT
 820     KOV:=KOV^.MUTATO;
 830     POKE(16384,KOV^.KOCKA);
 840     FOR I:=1 TO 500 DO
 850   UNTIL I=0
 860 END.
```

COPY
— másolatot készít a képernyőről a ZX nyomtatóra

PENDOWN C:INTEGER
— leteszi a tollat a papírra és C színűre állítja

PENUP
— felemeli a tollat

SETHD A:REAL
— beállítja a teknős irányát

SETXY X,Y:REAL
— beállítja a teknős helyzetét

FWD L:REAL
— előre megy L hosszan

BACK L:REAL
— visszamegy L hosszan

VECTOR A,L:REAL
— elfordul A szög irányába és előremegy L hosszan

RIGHT A:REAL
— jobbra fordul A szöggel

LEFT A:REAL
— balra fordul A szöggel

ARCR R:REAL;A:INTEGER
— körívet rajzol. A a középponti szög, R arányos a sugárral

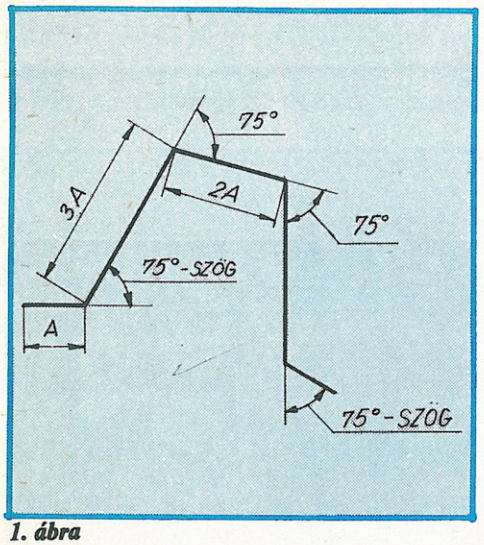
TURTLE
— inicializálja a teknőst: törli a képernyőt, középre állítja a „békát", a fejét vízszintesen jobb irányba állítja, a hátteret sötétkékre, a tinta színét sárgára állítja be.

Egy kis technológia

Ennyi bevezető után nézzük a feladatunkat! A teknősbéka-grafika nagyon hatékony abban az esetben, ha egy geometriai alakzatot kell sokszor transzformálnunk síkban. Ilyen eset például egy fogaskerék rajzának elkészítése. Ilyenkor elegendő egyetlen fog rajzának programozása, majd ennek a programrésznek kis elforgatásokkal való sorozatos hívásával felrajzolódik a teljes fogaskerék.

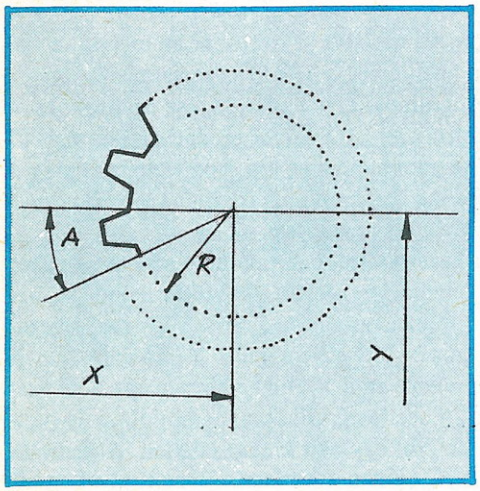
Hogy ez mennyire egyszerű, a mellékelt programlista FOG azonosítójú eljárásán látható. Az igazi fogaskerék egy fogának két oldalát evolvens görbe határolja. A mi esetünkben azonban a fogaskerék méretei olyan közel esnek a képernyő felbontásához, hogy az evolvens és az egyenes között nem lehet különbséget tenni, ezért természetesen az egyszerűbb egyenest programozzuk.

A FOG eljárásnak két paramétere van: az A REAL paraméter a fog nagyságával arányos mennyiség (lásd az *1. ábrát*), a SZÖG pedig tulajdonképpen a fogszámmal fordítottan arányos szög. Ha fogaslécet rajzolunk (a fogasléc végtelen fogszámú



1. ábra

2. ábra



fogaskerékként is értelmezhető), ez a szög nulla értéket vesz fel, a fogszám csökkenésével a SZÖG növekszik. Egymással kapcsolódó fogaskerekeknél az A értéknek meg kell egyeznie, a SZÖG-nek azonban természetesen nem.

A FOGASKEREK eljárás a FOG segítségével egy teljes kereket rajzol fel. A paraméterek geometriai jelentése: N a fogszám, X, Y a fogaskerék középpontjának koordinátái, A az elforgatási szög (ennek a későbbiekben lesz jelentősége), és végül R a sugár. A *2. ábra* segíti a programrészlet működésének megértését.

Ha az olvasó idáig jutott a cikk olvasásában, kipróbálhatja a fogaskerekek felrajzolását, és tapasztalhatja, milyen egyszerű is ez. Lépjünk azonban egy lépéssel tovább, hozzuk forgásba a fogaskerekeket!

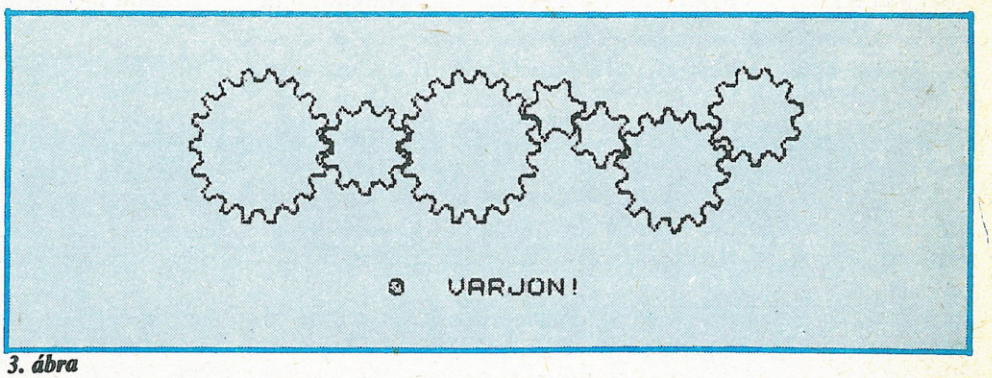
Egy kis kinetika

Mozgóképet előállítani számítógép segítségével is úgy lehet, ahogy azt a mozi vagy a televízió teszi: a mozgást fázisképekre kell bontani, s ha azokat elég nagy sebességgel felvillantjuk a néző előtt, majd minden képet egy kis ideig kivetítve hagyunk, folyamatos mozgás benyomását keltjük. A számítógép esetében a helyzet inkább a rajzfilm megvalósítására hasonlít: az egyes képkockák elkészítése történhet viszonylag lassan (például BASIC nyelvű programmal vagy interaktív rajzolással), csak a visszajátszásnak kell igen gyorsan végbemennie.

Az eljárás tehát az lesz, hogy megrajzoljuk az egyes fázisképeket, majd rendre eltároljuk egy alkalmas strukturált változóba. Gondot okoz, hogy egy kép tárolására a számítógép RAM-jához képest nagy memóriaterületre van szükség. Ezért gyakorlatilag csak önmagába visszatérő (periodikus) mozgást tudunk megvalósítani, azt is úgy, hogy „vetítővászonként" csak a képernyő felső harmadát használjuk ki.

Esetünkben hat képkockából álló „filmet" készítünk. Egy filmkockát egy KÉPHARMAD típusú, $8 \times 8 \times 32$ bájtnyi tömbben tárolunk. A KÉPHARMAD-ból és egy mutatóból RECORD-ot deklarálunk, amelyet mutatók segítségével fogunk kezelni. A program ezután egy ciklusban felrajzolja a hat egymást követő képkockát, minden egyes képet egyhatod fogosztásnyival elforgatva az előzőhöz képest, majd a képernyőfájl felső harmadát átmásolja a KOV mutató által címzett rekord KOCKA elemébe. A másolást a PEEK függvény segítségével valósítjuk meg. A *3. ábrán* egy fáziskép látható.

A rekord MUTATÓ azonosítójú, pointer típusú változója a ciklus magja végén az előző FILM-re mutat. Így egyszerű listát szervezünk a rekordokból. A ciklus lefutása után a program megkeresi az előző rekord MUTATÓ-ját, és az utolsó kocka pointerének értékét adja neki. Így a rekordokból zárt gyűrű alakú láncot alakítottunk ki, amelyből csak egy (tetszőleges) elem helyét kell ismerni, és innen az egész gyűrű sorban kiolvasható. Vegyük észre, hogy ilyen elegáns megoldás rekordok és mutatók nélkül nem valósítható meg: az ilyen lehetőségek jelentik a Pascal szépségét.



3. ábra

Egy kis további technika

Ha a „filmet" elkészítettük, le is játszhatjuk. Ehhez egy REPEAT—UNTIL ciklust használunk, melyben mindig kiolvassuk a KOV mutató által címzett rekordot, ennek KOCKA elemét a képernyőfájl felső harmadába POKE-oljuk, majd lefuttatunk egy belső üres ciklust 500-szor, csak azért, hogy a kép egynéhány pillanatra megálljon, és szemünk képes legyen észlelni. A ciklus akkor állna le, ha az $1 = 2$ feltétel teljesülne, amire természetesen a közeljövőben nincs remény; így sikerült egyszerű módszerrel végtelen ciklust szervezni.

Még egy apró trükk. Az egyes fázisképek kirajzolása a képernyőre a filmkészítés közben nem látható, mivel mind a PAPER, mind az INK színét azonosan fehérre állítottuk be.

Ez az egyszerű program jól szemlélteti a Hisoft Pascal figyelemre méltó sebességét. Ha az olvasó saját „filmet" akar forgatni, elmondjuk, hogy több helyet lehet biztosítani a RAM-ban, ha a programot T opcióval fordítjuk, hiszen ekkor sem a forrásnyelvű szöveg, sem a compiler, sem az editor nincs az operatív memóriában futás közben. Tovább lehet „fogyasztani" a programot, ha kettévágjuk: egy programmal elkészítjük a fázisképeket, és ezeket egyenként kazettás magnóra vagy lemezmeghajtóra küldjük ki. Egy másik program feladata csak annyi lesz, hogy visszaolvassa a háttértárról a képeket, elrakja a fent vázolt algoritmussal a RAM-ba, majd visszajátssza.

A TURTLE fájl eljárásai a ROM eredeti pont- és vonalrajzoló algoritmusait használják. Ha ezeket átírjuk a korábbi cikkekben ismertetett PLOT és PixelLine eljárásokra, a programcsomag sokkal rugalmasabbá válik.

Az ismertetett algoritmus más korszerű Pascal implementációban is megvalósítható. Így például az IBM kompatibilis gépek Turbo Pascaljával majdnem sorról sorra azonos program írható. Mivel a képernyőfájl szerkezete és a nem szabványos grafikus eljárások eltérőek, lesznek különbségek, viszont a film négy színben pompázhat, ha van színes grafikus kártyája a gépnek.

DR. KABOLDY PÉTER

Szellemek forgatása

COMMODORE 64

A szellemek (más néven sprite-ok, szellemgrafikák, MOB-ok) forgatása érdekes hatást vált ki a képernyőn. Két gépi kódú programot közlünk, melyek ezt valósítják meg: a C64 belső megszakításának, az IRQ ciklusnak az ideje alatt — azaz minden hatvanad másodpercben — a programok forgatnak egy bitnyit az adott szellemgrafikán. Így olyan hatás jön létre, mintha a szellem magától forogna a képernyőn. A forgatás bekövetkezik akkor is, ha a szellemgrafika nincs is a képernyőn, azaz ha a szellemgrafika nincs is bekapcsolva. A szellem forgatása mindaddig tart, amíg le nem állítjuk a program működését, vagy a RUN/STOP és a RESTORE nyomógombokat egyidejűleg le nem nyomjuk. Így egy BASIC program kilistáztatása vagy futtatása idején ezek a forgó szellemek a képernyőn maradhatnak függetlenül a BASIC program listázásától vagy futtatásától. A szellemek forgatása létrejön a kétszeresre felnagyított szellemgrafikák esetében is. A két gépi kódú program azonos szerkezetű, az egyikük balra, a másikuk jobbra forgatja az adott szellemet.

Balra forgatás

A program *(1. lista)* a BASIC munkaterület fölött helyezkedik el, így azt a memóriából a NEW paranccsal nem lehet kitörölni. Akár egy BASIC programból, akár közvetlen üzemmódból elindíthatjuk ezt a gépi kódú programot a SYS(53136) utasítással. Felhívjuk a figyelmet arra, hogy nem szabad egymás után kétszer kiadni ezt az utasítást, mert a számítógép ilyenkor „megőrül": semmi más feladattal nem foglalkozik, csak a szellemgrafika forgatásával. Ez azért van így, mert a programot szándékosan készítettük olyanra,

```
1000 FOR I= 53136  TO  53230
1010 READ X:POKE I,X:S=S+X:NEXT
1020 DATA  76,162,207,120,173,231,207,172
1030 DATA 232,207,141, 20,  3,140, 21,  3
1040 DATA  88, 96,120,173, 20,  3,172, 21
1050 DATA   3,141,231,207,140,232,207,169
1060 DATA 187,160,207,141, 20,  3,140, 21
1070 DATA   3, 88, 96,173,233,207,172,234
1080 DATA 207,133,251,132,252,162, 21,160
1090 DATA   0,177,251, 10,160,  2,177,251
1100 DATA  42,145,251,136, 16,248, 24,165
1110 DATA 251,105,  3,133,251,144,  2,230
1120 DATA 252,202,208,237,108,231,207, 49
1130 DATA 234, 64,  0,  0,  0,  0,  0
1140 IFS<> 12500 THEN PRINT"HIBAS ADAT !":END
1150 PRINT"OK !"
```

2. lista

hogy akkor is működjön, ha az IRQ vektor nem az eredeti, a bekapcsoláskor érvényben lévő címet, a $EA31-et tartalmazza. Így ez a program olyan más programokkal is hajlandó lesz együttműködni, amelyek korábban átírták az IRQ vektor eredeti értékét egy attól eltérő új címre. Ennek azonban az az ára, hogy kétszer egymás után nem szabad elindítani a programot. A forgatást leállítani már lehet többször is, a SYS(53139) utasítással.

Jelenleg a program a kazettatárolóban található szellemgrafikát forgatja, amely a 832. memóriarekesztől ($Ø340) kezdődően helyezkedik el 63 bájton. Emlékeztetőül: egy sprite vagy szellemgrafika X irányban 3 bájt (azaz 24 bit) szélességű és Y irányban 21 bájt (azaz 21 képpont) képet jelent, amit X, illetve Y irányban a kétszeresére lehet felnagyítani. A szellemgrafikák a memóriában 64 bájtnyi helyet foglalnak le, amiből egy bájt kihasználatlan. A szellemgrafikákat megjelenítő VIC (*V*ideo *I*nterface *C*ontroller) integrált

1. lista

```
;0:    CF90                    .OPT P1,O0
                               ;
                               ; SPRITE (MOB) BITENKENTI
                               ; FORGATASA BALRA AZ IRQ ALATT.
                               ; BE - SYS(53136)
                               ; KI - SYS(53139)
                               ;
140:   CF90                    *=   $CF90
                               ;
160:   CF90            IRQVEK  =    $0314
170:   CF90            AKT     =    $FB
180:   CF90            SØR     =    21
185:   CF90            KARNØ   =    3
                               ;
200:   CF90 4C A2 CF           JMP  BE
210:   CF93 78         KI      SEI
220:   CF94 AD E7 CF           LDA  TEMP
230:   CF97 AC E8 CF           LDY  TEMP+1
240:   CF9A 8D 14 03           STA  IRQVEK
250:   CF9D 8C 15 03           STY  IRQVEK+1
260:   CFA0 58                 CLI
270:   CFA1 60                 RTS
                               ;
290:   CFA2 78         BE      SEI
300:   CFA3 AD 14 03           LDA  IRQVEK
310:   CFA6 AC 15 03           LDY  IRQVEK+1
320:   CFA9 8D E7 CF           STA  TEMP
330:   CFAC 8C E8 CF           STY  TEMP+1
410:   CFAF A9 B8              LDA  #< UJIRQ
420:   CFB1 A0 CF              LDY  #> UJIRQ
430:   CFB3 8D 14 03           STA  IRQVEK
440:   CFB6 8C 15 03           STY  IRQVEK+1
450:   CFB9 58                 CLI
460:   CFBA 60                 RTS
                               ;
480:   CFBB AD E9 CF  UJIRQ    LDA  MOB
490:   CFBE AC EA CF           LDY  MOB+1
500:   CFC1 85 FB              STA  AKT
510:   CFC3 84 FC              STY  AKT+1
530:   CFC5 A2,15              LDX  # SØR
550:   CFC7 A0 00      FØRG    LDY  #0
560:   CFC9 B1 FB              LDA  (AKT),Y
570:   CFCB 0A                 ASL  A
590:   CFCC A0 02              LDY  # KARNØ-1
610:   CFCE B1 FB      CIKLUS  LDA  (AKT),Y
620:   CFD0 2A                 ROL  A
630:   CFD1 91 FB              STA  (AKT),Y
640:   CFD3 88                 DEY
650:   CFD4 10 F8              BPL  CIKLUS
670:   CFD6 18                 CLC
680:   CFD7 A5 FB              LDA  AKT
690:   CFD9 69 03              ADC  #KARNØ
700:   CFDB 85 FB              STA  AKT
710:   CFDD 90 02              BCC  VEGE
720:   CFDF E6 FC              INC  AKT+1
740:   CFE1 CA         VEGE    DEX
750:   CFE2 D0 E3              BNE  FØRG
770:   CFE4 6C E7 CF           JMP  (TEMP)
                               ;
820:   CFE7 31 EA      TEMP    .BYTE$31,$EA
830:   CFE9 40 03      MOB     .BYTE64,3
```

áramkör csak 16 kbájtnyi memóriatartományt tud megcímezni (16 kbájt = 256 × 64 bájt), így egy egybájtos mutató segítségével a tudomására hozhatjuk, hogy a megjelenítendő szellemgrafika hányadik blokkon, azaz hányadik 64 bájtnyi memóriaterületen található. Ez esetünkben a 13. blokkot jelenti (832 = 13 × 64). Ha máshová akarjuk elhelyezni a szellemgrafikánkat, akkor az első listán a 830-as sorban található címet kell megváltoztatnunk; a cím a szokásos alsó bájt (elöl), felső bájt (hátul) sorrendben van megadva a 830-as sorban. A POKE 53225, (CIM AND 255):POKE 53226,INT(CIM/256) utasítással megváltoztathatjuk a forgatandó sprite kezdőcímét; itt az előbb elmondottak értelmében a CIM-nek a 64 többszörösének kell lennie. A BASIC munkaterület megváltoztatása nélkül négy szellemgrafika helyezhető el a következő címekre: a 11. a 704-től, a 13. a 832-től, a 14. a 896-tól és a 15. sprite a 960-as memóriacímtől kezdődően. Ha ennél több szellemgrafikát kívánunk megjeleníteni a képernyőn, akkor például a BASIC munkaterület kezdetét feljebb kell állítanunk.

A szellemgrafikát balra forgató gépi kódú program BASIC betöltőprogramját a *2. lista* tartalmazza. Az itteni 1130-as sorban található 64 és 3 (ezek a sprite címét határozzák meg: 3 × 256 + 64 = 832) számok megváltoztatásával más helyet jelölhetünk ki a forgatandó szellemgrafika számára. Felhívjuk a figyelmet arra, hogy a sprite-ok adatainak bevitelekor a forgató rutint le kell állítani, ellenkező esetben a sprite tönkremegy. Arról van ugyanis szó, hogy mialatt mi a sprite adatait, a 63 bájtot a megfelelő helyre visszük POKE utasítások segítségével, a bekapcsolt forgató rutin többször megforgatja a sprite-ot, és így nem az eredeti alakot kapjuk vissza.

A *3. lista* egy BASIC programot mutat, amelyben egy nyuszit formázó sprite négy különböző irányban mozog. A program első része (1—30-as sorok) még nem tartalmaz forgatást, csak a sprite-ok mozognak a képernyőn; majd egy gomb lenyomása után az előbbi folyamat játszódik le, csak most már a sprite-ok forgatása

3. lista

```
1 REM NYUSZI UL A FUBEN ...
5 PRINTCHR$(147)
10 V=53248
11 POKE V+21,15:REM SPRITE BEKAPCSOLVA
12 POKE2040,13:POKE 2041,13
13 POKE2042,13:POKE 2043,13
20 FORN=0 TO 62:READQ:POKE832+N,Q:NEXT
25 POKEV+40,3
30 GOSUB130
40 GETA$:IFA$=""THEN40
50 SYS(53136):REM FORGATAS BEKAPCSOLVA
60 GOSUB130
70 GETA$:IFA$=""THEN70
90 SYS(53139):REM FORGATAS KIKAPCSOLVA
95 POKE V+21,0:REM SPRITE KIKAPCSOLVA
100 END
130 FORX=0 TO 200
140 POKEV,X
141 POKEV+1,150
142 POKEV+2,200-X
143 POKEV+3,200-X
144 POKEV+4,200
145 POKEV+5,X
150 POKEV+6,X
155 POKEV+7,200-X
160 NEXT:RETURN
200 DATA0,224,0,7,152,0,8,196,0,4,98,0
210 DATA2,17,0,1,139,192,0,68,48,7,192,8
220 DATA8,96,4,56,48,2,80,16,198,80,16
230 DATA198,80,24,4,80,36,24,59,195,240
235 DATA12,16,192,7,255,192,0,0,0,0,0,0
240 DATA 0,0,0,0,0,0,0,0
300 END
```

```
1000 FOR I= 52992  TO  53090
1010 READ X:POKE I,X:S=S+X:NEXT
1020 DATA  76, 18,207,120,173, 93,207,172
1030 DATA  94,207,141, 20,  3,140, 21,  3
1040 DATA  88, 96,120,173, 20,  3,172, 21
1050 DATA   3,141, 93,207,140, 94,207,169
1060 DATA  43,160,207,141, 20,  3,140, 21
1070 DATA   3, 88, 96,173, 95,207,172, 96
1080 DATA 207,133,251,132,252,162, 21,160
1090 DATA   2,177,251, 74,160,  0,  8, 40
1100 DATA 177,251,106,145,251,  8,200,192
1110 DATA   3,208,244, 40, 24,165,251,105
1120 DATA   3,133,251,144,  2,230,252,202
1130 DATA 208,221,108, 93,207, 49,234, 64
1140 DATA   3,  0,  0
1150 IFS<> 11911 THEN PRINT"HIBAS ADAT !":END
1160 PRINT"OK !!"
```

4. lista

mellett (50—60-as sorok); ezután, ha a klaviatúrán még egy gombot megnyomunk, a sprite-ok forgatása megszűnik, és a program a sprite-okat is kikapcsolja. Közvetlen üzemmódból kapcsoljuk visszza a szellemgrafikákat a POKE V+21,15 utasítással. (Ha a BASIC programon módosítottunk éppen, akkor ehelyett az utasítás helyett a POKE 53269,15 utasítást kell alkalmaznunk, mert a módosítás után a V BASIC változó elveszti a korábbi értékét, az 53248-at.) Ezután indítsuk a forgatást SYS(53136)-tal, majd próbáljuk megnövelni a sprite méretét kétszeresre az Y irányban a POKE 53271,15 utasítással; X irányban a POKE 53277,15 utasítással növelhetjük meg a négy sprite méretét.

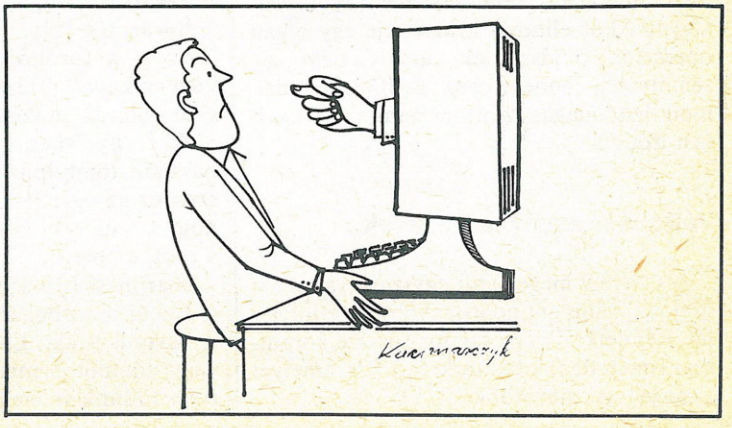
Jobbra forgatás

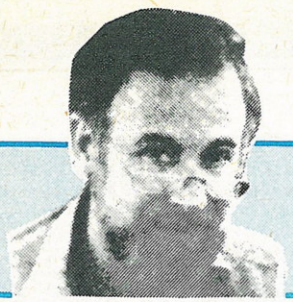
A szellemgrafikákat jobbra forgató gépi kódú program BASIC betöltőprogramja a *4. listán* látható. Ezt a jobbra forgató rutint a SYS(52992)-vel kell indítani és a SYS(52995)-tel leállítani. Az indításra és leállításra a korábban elmondottak érvényesek itt is. Ez a program is a 13. blokkban, a 832. tárcímtől kezdődő sprite-ot forgatja; a cím alsó bájtja, a 64-es szám, a 4. lista 1130-as sorának utolsó helyén található; a sprite cím felső bájtja, a 3 pedig az 1140-es sorban az első adat. Ezeket a számokat átírva tudjuk megváltoztatni a RAM-ban a sprite helyét.

Ha a 3. listán látható programban módosítjuk az 50-es sorban a SYS(53136)-ot SYS(52992)-re, valamint a 90-es sorban a SYS(53139)-et SYS(52995)-re, és így futtatjuk a programot, akkor a sprite-ok jobbra forognak.

A következő alkalommal a nagy felbontású képernyő forgatásáról lesz szó. Addig is jó szellemforgatást kívánok!

SZABÓ PÉTER PÁL





Mit tud a C nyelv? I.

A C programozási nyelv megszületése óta páratlan karriert futott be. Ezt részben sajátos erényeinek köszönheti, részben pedig annak, hogy voltak, akik ezekkel az erényekkel maximálisan éltek is.

A C nyelv általános célú, magas szintű programozási nyelv. Ezeknek sorából leginkább azzal válik ki, hogy annyira gépközeli, amennyire ez egyáltalán lehetséges. Azt mondhatnánk, hogy ő a „legalacsonyabb" magas szintű nyelv. Ráillik a „magas szintű assembler" megnevezés is. Utasításkészlete rendkívül szűk, mindössze a változódeklarációkra, az értékadásra, a vezérlési szerkezetek jó megvalósítására és a függvényhívásra szorítkozik. Tehát minden megvan benne, ami az assemblerekben gépfüggetlen, „gépfeletti" lehet, és mindent kiküszöböltek belőle, ami a nyelvet egy adott géphez vagy egy bizonyos géptípushoz kötné.

Legfontosabb erénye talán az, hogy ismeri a pointerek, a mutatók fogalmát, és a mutatókat a lehető legnagyobb szabadsággal kezeli. Ily módon megvalósíthatjuk a legravaszabb adatszerkezeteket is, és igen szabadon kezelhetjük a rendelkezésre álló memóriát. Ez teszi a C nyelvet rendkívül hatékonnyá.

Ezt használta ki 1972-ben Dennis M. Ritchie, a Bell Laboratórium munkatársa, aki nagy szerepet játszott a C nyelv létrehozásában is. Számos kiváló kollégájával együtt egy merész húzással C-ben írták meg a már 1969 óta létező Unix operációs rendszert. Ezzel minőségileg új szakasz kezdődött a számítástechnikában: jól vizsgázott egy olyan magas szintű programozási nyelv, amely megfelelő kényelmet biztosít ugyan a felhasználónak, de annyira hatékony és „assemblerszerű", hogy lehetővé teszi operációs rendszer megvalósítását. Ugyanakkor elindult a karrierje egy olyan operációs rendszernek, amely nem assemblyben, hanem egy gépfüggetlennek mondható magas szintű programozási nyelven íródott.

Változók, adatszerkezetek

A C nyelv megengedi egyszerű változók, tömbök, valamint adatstruktúrák definícióját. Rendelkezik egy sajátos (persze korántsem ismeretlen) adatszerkezettel is, amelyet „union"-nak nevezünk.

Egyszerű változók

Az egyszerű változók egy-egy adat befogadására képesek. Nevük és típusuk megadásával definiálhatók:

```
int i;
```

Ez egy integer (egész) típusú változót definiál, melynek neve i. Az általánosan elfogadott típusok:

char	karakter
int	egész szám
long	hosszú (kétszeres pontosságú) egész
float	lebegőpontos szám
double	kétszeres pontosságú lebegőpontos szám

Tömbök

Az egyes változókból egyszerűen képezhetünk tömböt:

```
int arr [20];
```

ahol a tömb neve arr, elemeinek száma 20 (az indexek 0-tól 19-ig futnak, tehát valóban húsz elemük van), és minden egyes tömbelem típusa egész.

A tömbök használata során az indexek helyén szabadon választhatunk tetszőleges kifejezéseket. De a mi dolgunk arról gondoskodni, hogy az így kapott index ne haladja meg a tömb elmeinek számát. Például:

```
int  arr [20];  (* tömb definíciója *)
int  i;         (* indexváltozó *)
int  a, b;      (* két munkaváltozó *)
```

esetén hibás lesz a következő két utasítás:

```
a = arr [-2];
a = arr [21];
```

mivel a fordító itt ellenőrzi (képes ellenőrizni) az indexhatárokat. Minden további nélkül leírható és végrehajtható azonban az alábbi utasítássorozat:

```
i = 10;
a = arr [i-16];
b = arr [i+10];
```

mivel a fordító, nem ismervén i futás közben kapott értékét, nem is képes semmilyen határellenőrzésre.

A C nyelvben talán a leggyakrabban használt tömbtípus a karakteres tömb, hiszen ez az egyetlen ésszerű lehetőség arra, hogy sztringeket, szövegeket helyezzünk el a programban:

```
char mess [ ] = „Egy egy üzenet\n";
```

Ebből a sorból több érdekes tanulságot is levonhatunk. Először is: egy változót vagy tömböt szintaktikailag elég egyszerűen tudunk inicializálni, azaz kezdeti értékkel ellátni. Másodszor: idézőjelek között adhatunk meg szövegeket. Harmadszor: a tömb előkészítésekor nem kötelező megadni, hány eleme van a tömbnek — a fordító meghatározza helyettünk. Például ennek a tömbnek 16 eleme van. És itt következik a negyedik és ötödik fontos dolog. A tömbnek nem azért van 16 eleme, amiért első pillantásra gondolnánk. A \n karaktersorozat az üzenet végén a valóságban nem két, hanem csak egy karaktert ér: ez a karakter a newline, az újsor, amelynek hatására az üzenet kinyomtatása után a terminálon egy kocsivissza-soremelés következik be. (A \ karakter egy ún. ESCAPE (váltó), melynek hatására a fordító elvonatkoztat az utána következő karakter konkrét értékétől, és különleges értelemben használja fel — például az n karakter helyett kocsivissza-soremelés.) Mi akkor hát a 16. karakter, hiszen így csak 15-öt adtunk meg!? Nos, az utolsó megadott karakter, az újsor után a fordító még odatesz egy 0 kódú karaktert is: ez jelzi majd a memóriában a sztring végét. Az ötödik fontos apróság tehát: a sztringek utolsó karaktere mindig a 0 kódú karakter. Programunk innen ismerheti fel a sztring kezelése közben, hogy vége a szövegnek.

Struktúrák

A C nyelv tetszőleges szerkezetű adatstruktúrák definiálását teszi lehetővé. Ezek segítségével nagyon pontosan és tömören lehet programozni. Lássunk egy példát, utalva a struktúrák egy felhasználási lehetőségére is! Tegyük fel, hogy olyan játékprogramot akarunk írni, amely több tárgyat mozgat egymástól függetlenül a képernyőn. Az egyszerűség kedvéért a tárgyak mindegyike egy-egy karakter, amelyek egyenes vonalú egyenletes mozgást végeznek a képernyőn. Az ilyen tárgyak leírására kiválóan alkalmas a következő struktúra:

```
struct  flyobj {
        char code;     (* a karakter kódja *)
        int xpos;   (* pillanatnyi pozíció *)
        int ypos;      (* koordinátái *)
        int xstep;     (* lépésvektor *)
        int ystep;     (* koordinátái *)
};
```

Mint látható, ez a struktúra mindent összefoglal a tárgy mozgatásához szükséges adatokból: a karakter kódját, a tárgy pillanatnyi pozíciójának koordinátáit, valamint a lépésvektort.

Magában véve a fenti néhány sor nem hoz létre a memóriában egy ilyen szerkezetű struktúrát. Ezzel pusztán a flyobj nevű struktúra belső felépítését adtuk meg. Egy ilyen struktúra számára a következő utasítással foglalhatunk helyet:

```
struct flyobj fly1;
```

ahol az első két szó tulajdonképpen a típus („legyen egy flyobj típusú struktúra"), fly1 pedig a memóriaterület neve. Hogyan lehet ezt a struktúrát használni? Sajnos a C nyelv nem teszi lehetővé a struktúrák globális használatát. Nincs tehát mód arra, hogy egy struktúrát egy utasítással átmásoljuk egy azonos szerkezetű másik struktúra területére, nem írhatunk csupa 0-t minden egyes struktúratag helyére stb. A struktúra tagjait külön-külön kell kezelnünk:

```
fly1.code = '*'     (* csillag repül *)
```

A struktúra tagjai külön-külön változók:

```
fly·xpos = fly·xpos + fly·xstep;
fly·ypos = fly·ypos + fly·ystep;
```

Ez az utasítássorozat kiszámítja a repülő tárgy következő pozícióját. Látható, hogy a struktúra tagjaival ugyanúgy bánhatunk, mint az egyes változókkal.

Programunkban természetesen tetszőleges számban definiálhatunk flyobj típusú struktúrákat. Ilyenkor az egyes változókat a kívánt struktúra és a kívánt tag neve azonosítja:

```
struct  flyobj fly1, fly2, fly3;
        fly1·code = '*'; (* első tárgy kódja
        *)
        fly2·code = '#'; (* második tárgy
        kódja *)
        fly3·xpos = 12;
```

A struktúrák tagjai nemcsak egyszerű változók, de tömbök, sőt újabb struktúrák is lehetnek:

```
struct  coord {
        int xcoord;
        int ycoord;
};
struct  flyobj {
        char code;
        struct coord pos;
        struct coord step;
};
```

Foglaljunk területet egy ilyen struktúra számára:

```
struct  flyobj  fly;
```

Ekkor persze bonyolultabb lesz a hivatkozás:

```
fly·pos·xcoord = fly·pos·xcoord + fly·
pos·xstep;
```

hiszen egymásba ágyazott struktúrák tagjaira kell hivatkoznunk.

Unionok

A union a C nyelv egyik érdekes újdonsága. Definiálása és felhasználása is nagyon hasonlít a struktúráéhoz:

```
union   alltyp {
        char ch;
        int in;
        long lg;
        float fl;
        double db;
};                  (* union-deklarálás *)
union   alltyp all; (* union-definíció *)
```

A union egy tagját a következőképpen használhatjuk fel:

```
all·ch = 'a'; (* egy karakter beírása *)
```

A union és a struktúra között az a különbség, hogy a struktúra egyes tagjai egymás után helyezkednek el a tárban, a union tagjai pedig mind egyazon címen kezdődnek. Ha tehát a fenti utasítással felülírom a karakteres uniontagot, akkor egyúttal „belepiszkáltam" a union összes többi tagjába. Felülírtuk az egész típusú „in" alsó felét, a long típusú „lg" negyedét és így tovább.

A fenti union belső felépítését és egyben a union lényegét mutatja az *ábra*. Az egyes elemek nem egymás mögé, hanem egymásra íródnak — tréfás meglepetéseket tartogatva a tájékozatlan programozónak, és kiváló lehetőséget nyújtva különböző típusok fizikai konverziójára:

```
union   dbtost {
        double db;
        char st [8];
};
```

Használjuk most fel ezt a union-deklarációt:

```
union   dbtost dbst; (* double → sztring *)
        dbst·db = 3·14159; (* be mint
        double *)
        putchar (dbst·st [3]); (* 3. bájt
        nyomtatása *)
```

Egy kétszeres pontosságú lebegőpontos változót tudunk elhelyezni, és ugyanezt az értéket bájtokra bontva kaphatjuk meg, mert a union másik oldala egy nyolc karakteres tömb — ennek elemei pedig az egyes bájtok. Ugyanez a konverzió persze visszafelé is elvégezhető.

Természetes, hogy a union is tartalmazhat tömböt és struktúrát is.

Eljárások

A C nyelv eljárásait függvényeknek nevezzük, melyeknek tetszőlegesen sok bemenő paraméterük lehet, viszont csak egy értéket adhatnak vissza a hívójuknak.

A C program függvényekből áll. Minden függvény egyenrangú, azaz nem definiálhatók egymás belsejében, mint például a PL/1 nyelvben. Egyenrangúságuk azt is jelenti, hogy bármelyik függvény meghívhatja bármelyiket, beleértve saját magát is — a nyelv lehetővé teszi a rekurzív függvények használatát.

Egyetlen kitüntetett függvény mégis akad, a „main", azaz a „fő" függvény. A program indulásakor a végrehajtás a main első utasításával kezdődik és annak utolsó utasításával ér véget. A main belsejéből hívhatjuk az összes többit — saját függvényeinket éppúgy, mint az ún. könyvtári függvényeket. Lássunk most egy igazi C programot:

```
main( )
{
        printf("Szervusz, kedves Olvasóm\n");
}
```

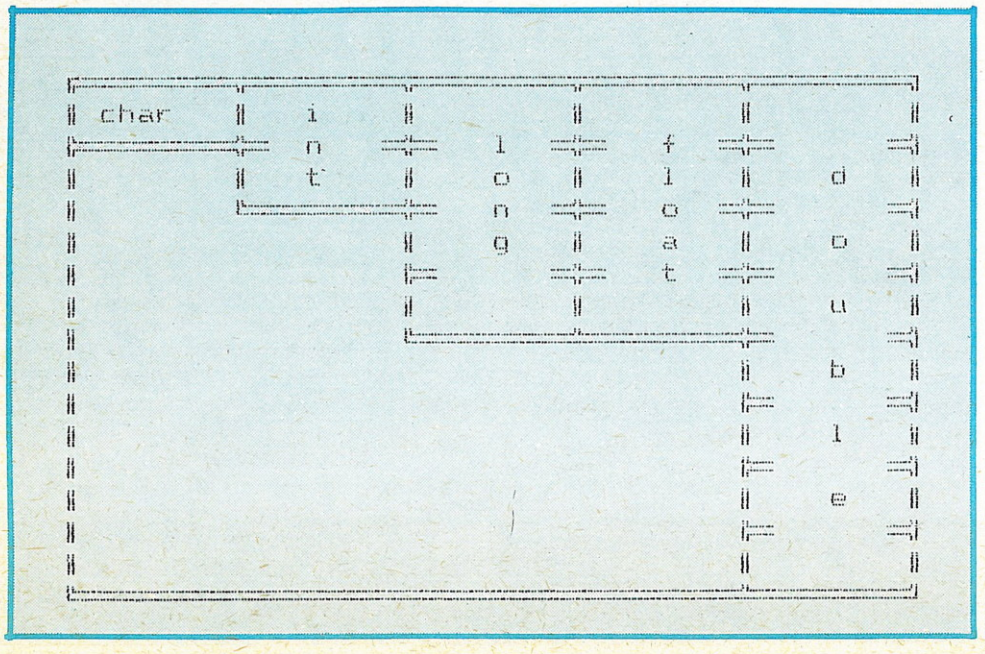
A program meghívja a printf nevű könyvtári függvényt, és ennek segítségével kinyomtatja az idézőjelek közé zárt szöveget — beleértve az újsor karaktert is. Láthatjuk, hogy egy C függvényt egyszerűen nevének leírásával hívhatunk, a függvény neve után zárójelben megadva a kívánt paramétereket. A paraméterek pedig akár konstansok, akár változók, akár ezekből képzett kifejezések lehetnek. Sőt megengedett az is, hogy egy függvény értéke szolgáljon paraméterként egy másik függvény számára.

Hogyan kell definiálni egy függvényt? Lássunk erre is egy példát:

```
add(a, b)
int     a;
int     b;
{
int   sum;
      sum = a + b;
      return(sum);
}
```

Ez a függvény két paramétert vesz át, melyek, mint deklarációjuk mutatja, egész típusúak. Az összeadás után a return( ) utasítás hatására adja vissza paramétereinek összegét.

PETHŐ ÁDÁM



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

Merre tart a világ?

Nyomtatók és néhány más periféria

A nyomtatók fejlődésének bemutatása előtt egy itthon ismeretlen perifériáról és a botkormányok/egerek újdonságairól szólunk.

Dialemez-készítő

Valószínűleg csodálkozik az olvasó, hogy mit keres itt ez a téma? Mi köze a számítástechnikához, hogy kerül ebbe a sorozatba? E kérdésekre az *1. kép* alapján adhatunk választ, amelyet a General Parametrics cég állított elő.

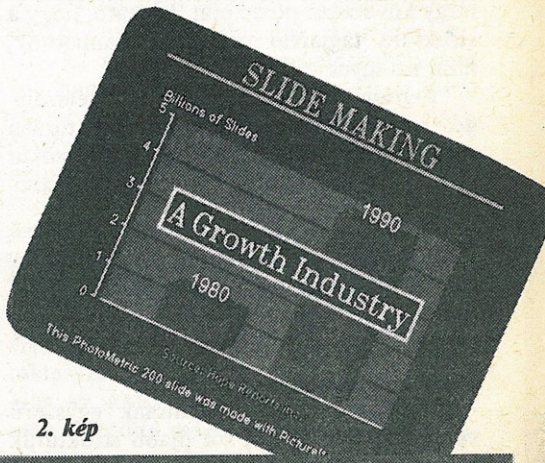
A képen látható, hogy a dialemez- (slide-) gyártás az Egyesült Államokban 1980-ban meghaladta az egymilliárd darabot és 1990-re mintegy 4,5 milliárd lesz. Az eredetileg 8 színű diát egy IBM PC AT-ből (512 k RAM), egy PhotoMetric 200 diakészítő eszközből, egy ColorMetric 200 kártyából (az utóbbiak a General Parametrics gyártmányai) és hagyományos grafikus szoftverből álló rendszeren készítették. Ez a rendszer olyan ismert szoftverek használatát teszi lehetővé, mint például a Lotus cég Freelance, az Ashton-Tate ChartMaster vagy a Microsoft Charta nevű terméke. A felbontás 2000 × 2000, a színválaszték 1000 szín. A lemezkészítés előtt a végleges formára szerkesztett kép bármely megfelelő felbontású monitoron ellenőrizhető. A rendszer fekete-fehér lemezt is elő tud állítani; ilyenkor 14 szürkeségi fokozattal dolgozik. Olyan szoftvert is árulnak hozzá, amellyel nyomtatón is megjeleníthetők a lemezek, vagy lemez nélkül közvetlenül vetíthetők.

Botkormányok, egerek

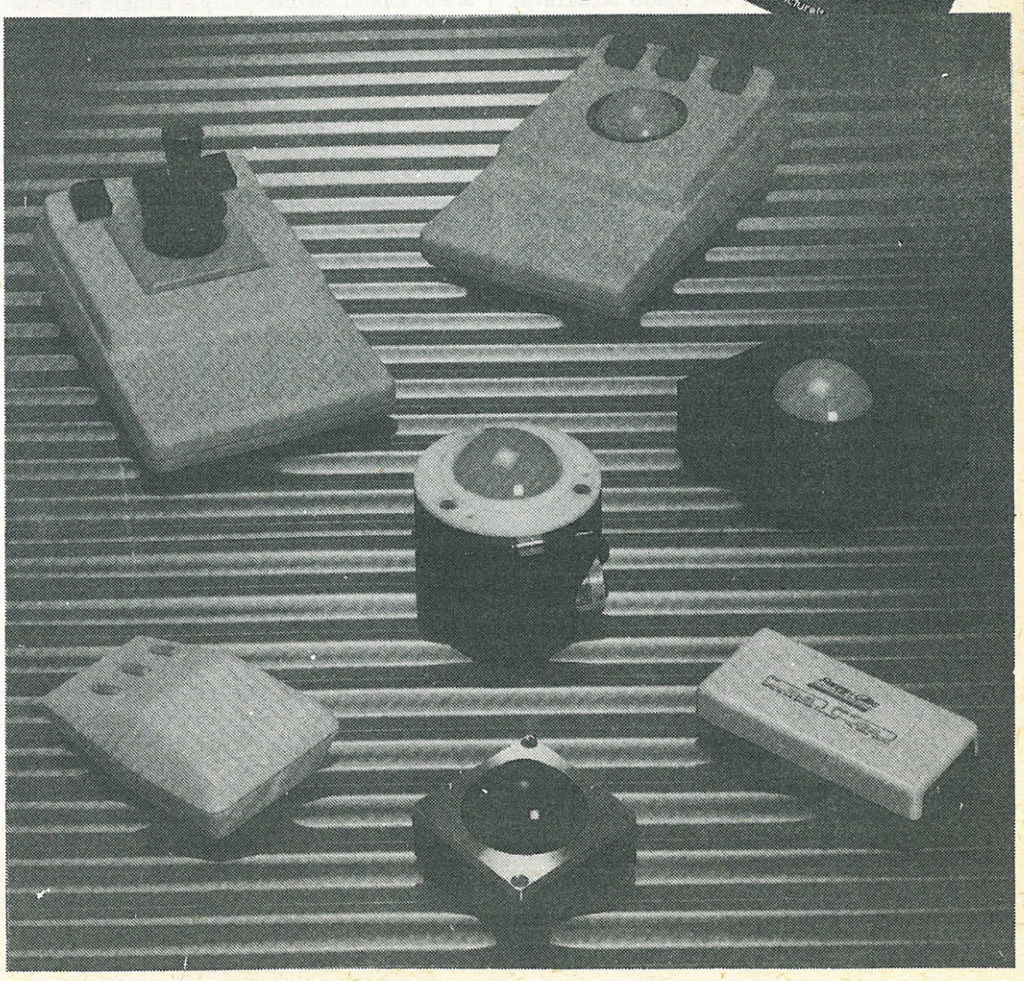
Egereket készítenek 1-3 nyomógombbal, közvetlenül a számítógép soros bemenetére vagy egy közbeiktatott miniatűr billentyűzetre alkalmas csatlakozással. Botkormányokat szabadon definiálható funkciójú nyomógombbal, vezérlő golyó (track ball)-nyomógomb kombinációval és berendezések kezelőlapjára szerelhető formában árulnak. Az óriási választékból ad egy kis ízelítőt a *2. kép*. Mindez azt mutatja, hogy e bemeneti eszközök gyártása virágkorát éli, de újnak számító megoldások nincsenek. A fej-, szem-, karmozgás érzékelésén alapuló készülékek még nem jutottak el a kiállításokra.

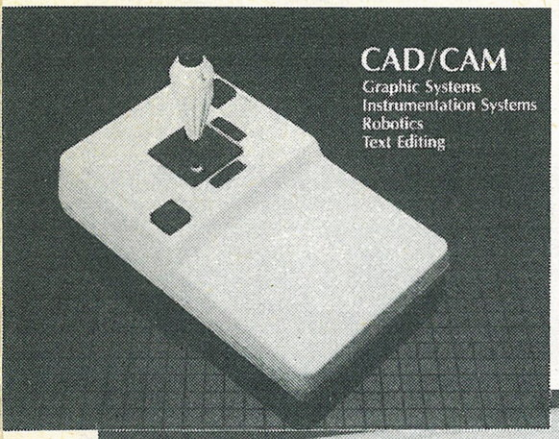
A CH Products MICROSTICK sorozatának egy új típusa *(3. kép)* sokoldalúságában különbözik a hasonlóktól. Ez a botkormány nyolcféle, kapcsolókkal kiválasztható, többek között a digitalizálókat és az egereket emuláló (utánzó) üzemmódban dolgozhat. Háromféle adatkiadási módja van. Felbontóképessége olyan nagy — irányonként 4096 pont —, hogy lehetővé teszi a LOTUS 1, 2, 3, a dBASE III, a WordStar,

1. kép



2. kép





a Multiplan, a VisiCalc stb. egérrel történő menüvezérlését. A szokásos sebességekkel (300—19 200 baud) és formátumokkal dolgozik: a szoftverrendszerhez közvetlenül, speciális működtető szoftver nélkül kapcsolható.

Nyomtatók

A Seikosha cég egyes termékei — a Sinclair és Commodore gépekhez használt oldalnált, széles körben elterjedt ultrahangos készülék előállítása is. Ez a vállalat volt az első japán mátrixnyomtató gyártója, és a világon elsőként készített videoprintert (képernyőnyomtatót). A Seikosha kezdte el a 420 karakter/s, 800 karakter/s sebességű mátrixnyomtatók gyártását.

Újdonságai közül elsőként az itthon is ismert SP—180 sorozat legújabb és sok paraméterében nagy előrelépést mutató tagját, az SP—185AI típust mutatjuk be *(4. kép)*. A 100 karakter/s sebesség és a széles papírrolni miatt elsősorban a gyors telexgépekhez alkalmas. A beépített Epson FX és IBM Graphics Printer emulátor lehetővé teszi, hogy a legismertebb grafikus programok használhassák. Csendes üzeme miatt jól használható olyan irodákban is, ahol sok nyomtató dolgozik egyszerre. Ára hasonló sorozatának korábbi tagjaiéhoz.

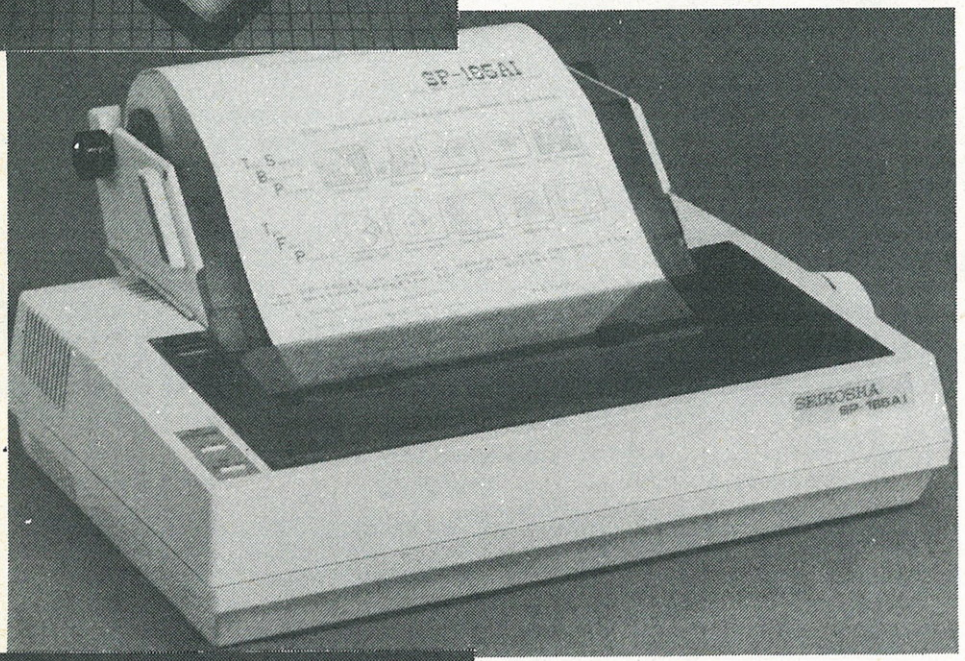
Az SBP—10 *(5. kép)* 18 tűs, 800 karakter/s sebességű, 64 k pufferű, mindenféle típusú papíradagolásra alkalmas nyomtató. Soros/párhuzamos illesztője, kijelzője van, a papírt előre-hátra továbbítja, a betűkészletet kazettáról tudja betölteni. Képes emulálni az Epson ESC/P és az IBM Proprinter nyomtatókat. E jellemzők bizonyítják, hogy rendkívüli teljesítményű nyomtató. Ennek azonban ára van, igen magas ára: kb. kétszer annyiba kerül, mint egy átlagos lézernyomtató. Elsősorban LAN típusú alkalmazásokhoz ajánlják.

A VP—3500 videonyomtató *(6. kép)* a 300 pont/hüvelyk (kb. 12 pont milliméterenként) felbontási kategóriába tartozik. Ezzel képernyőnként 1280 × 1250 pontos felbontás nyomtatható ki, 64 szürkeségi fokozatban. Így mind alfanumerikus, mind grafikus üzemmódban használható nagy felbontású monitorokhoz. Tévékészülékekhez, videokamerákhoz, meteorológiai műholdak által szolgáltatott térképek készítéséhez, elektronmikroszkópos felvételek kinyomtatására, röntgenfelvételek feldolgozására, ipari folyamatgenerálásra, elektronikus áramkör tervezésére, fényképek elektronikus továbbítására alkalmazható.

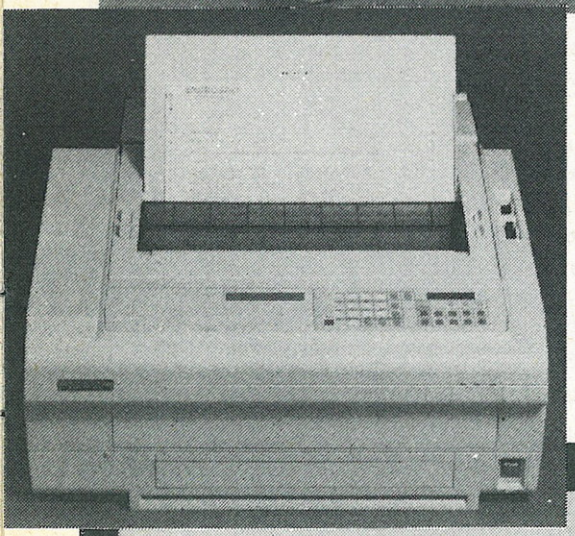
A BP—5420AI típus nálunk is elérhető árú, nagy sebességű (420 karakter másodpercenként), széles kocsis, nagy igénybevételre tervezett nyomtató. 2 év garancia van rá. Ára mintegy kétszerese az itthon elterjedt, de nem ebbe a kategóriába tartozó Epson FX—1000-ének. A konstrukció robusztus kivitelű — nem műanyagból, hanem fémből készült, ezért tömege 32 kg —, jól tagolt önálló részegységekből áll, amelyek összetéveszthetetlen csatlakozókkal kapcsolódnak, könnyen cserélhetők, így a szervizelés egyszerű. Az egyes egységek a főbb funkciókat megvalósító egységeket elkülönítve tartalmazzák, ezért a hibakeresés és a hibás részek cseréje nem okoz különösebb gondot. A pufferkapacitás 18 kbájt. A nyomtatónak 8 nyelvű betűkészlete, IBM és Epson üzemmód-emulátora, soros/párhuzamos illesztője van, zajszintje alacsony.

A Sony céget elsősorban hangtechnikai berendezéseiről, kevésbé monitorairól ismerjük, de szinte semmit nem tudunk le-

3. kép



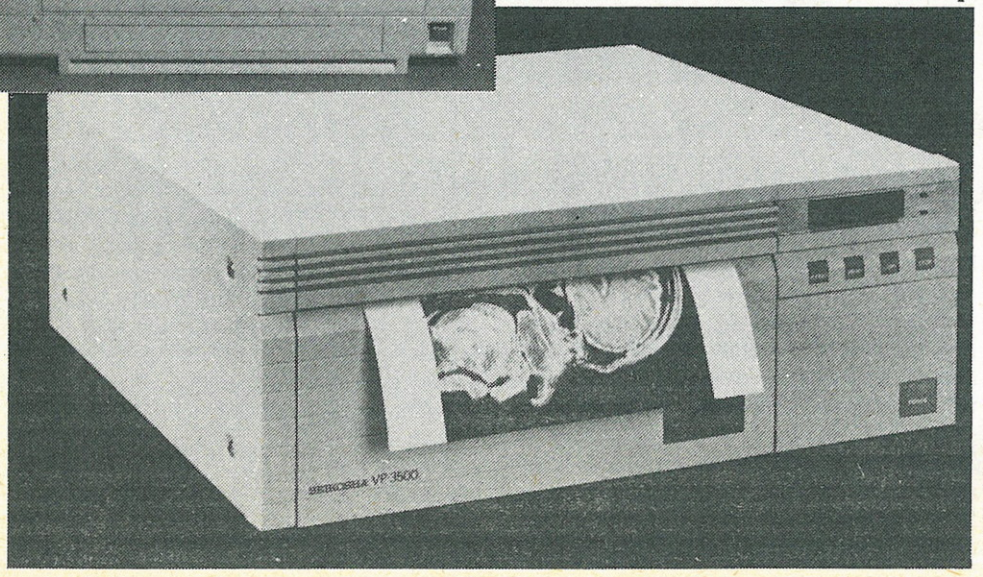
4. kép



5. kép

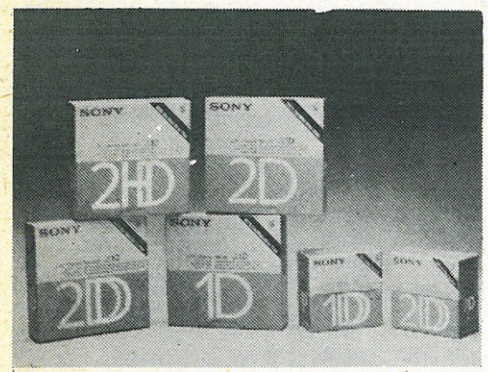
csóbb nyomtatók — itthon is széles körben ismertek, de nem ezek a cég legfontosabb gyártmányai. Azt például nálunk is tudják néhányan, hogy leányvállalata a Seiko nevű, digitális óráiról ismert cég, de arról már valószínűleg csak keveseknek van tudomása, hogy az első 8 bites mikroprocesszort, az I8080 típust a Seikosha az Intel céggel együtt dolgozta ki, vagy hogy nevéhez fűződik a foggyökércsatornák kezelésére hasz-

6. kép



mezgyártásról. Ez annál is inkább meglepő, mivel ez a cég volt a 3,5" méretű lemezek és lemezegységek kidolgozója, elterjesztője. Ma a korábbi 5,25" méretű egy- és kétoldalas, egyszeres és kétszeres sűrűségű lemezeken kívül egyoldalas 0,5 Mbájt (formattálva 360 kbájt) kapacitású, kétoldalas és kétoldalas nagy sűrűségű, 3,5" méretű lemezeket is gyárt. A Sony a 3,5" méretű lemezeket tekintve a világ legnagyobb gyártója. Az új, nagy sűrűségű lemezek mágneses anyaga vas-oxidban abszorbeált kobalt. A bevonat vastagsága fele a korábbi 2 mikronnak. A cég lemezválasztéka a *7. képen* látható. A lemezárakat az itthoniakkal összehasonlítva kb. 500 forint/dollár értéket kapunk, ami igazán megdöbbentő!

Magyarországon a Sony nyomtatók is ismeretlenek, pedig a cég ezen a területen a szuper minőségre specializálta magát. Az UP—2000 színes, grafikus nyomtató ún. WYSIWYG („amit-látsz-azt-kapod") típusú képernyőnyomtató. Az IBM PC piacon általánosan elterjedt EGA kártyák felbontásával kompatibilis. Clem Shemanski marketingigazgató ettől a terméktől azt várja, hogy új szabványt teremt. Érdekessége,

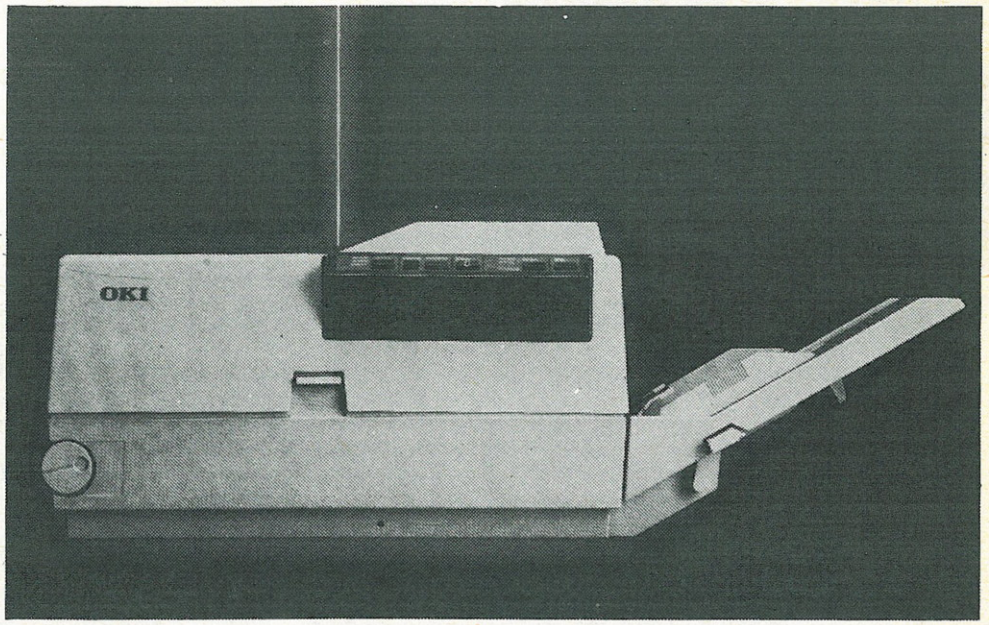


7. kép

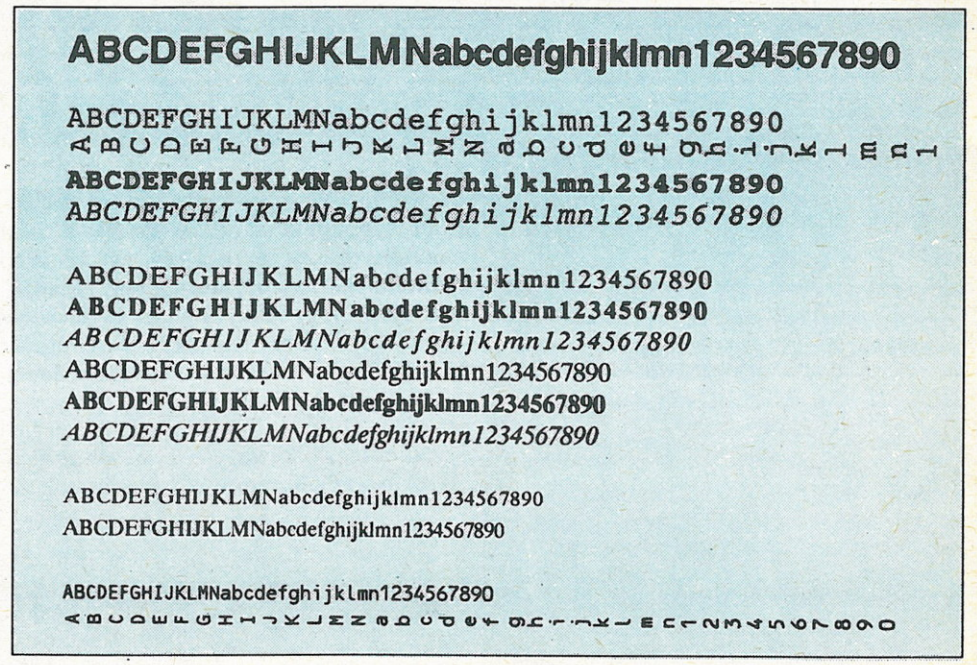
hogy az RGB monitorkimenetre csatlakoztatható, vagyis a monitor helyére kerül. A meghajtó szoftverfüggetlen és a 640 x 350 pontos képernyőfelbontás mellett 77 x 102 mm méretű nyomtatványt ad. A 64 színű képet négy menetben állítja elő, 40 másodperc alatt. A nyomtatás elve villamosan vezető tinta használatán alapul. Egy festékkazetta és a speciális papír egy tekercse 100 kép készítésére elég. Az UP—2000 ára valamivel magasabb, mint egy átlagos lézernyomtatóé.

A nálunk egyelőre még ismeretlen, egyesült államokbeli Silver Reed új terméke az SP—30 típusú képkészítő nyomtató (scanner printer), amely két, egymástól függetlenül is működtethető funkciót lát el: a képkészítést és a képkinyomtatást. A készüléket egy párhuzamos csatornán keresztül lehet a számítógéphez csatlakoztatni. Felvétel üzemmódban a felbontás 8 pont/mm vízszintesen és 7,7 pont/mm függőlegesen. Az olvasási sebesség igen nagy: 4,2 ms soronként. A nyomtató üzemmód felbontása hasonló, a sorszélesség 208 mm. A nyomtatás hőpapírra készül. A készülék ára 700 dollár.

Az Okidata céget ML sorozata és olcsó színes nyomtatója, az Okimate révén ismerjük. Itt most a LASERLINE 6 PLUS típusú lézernyomtatóját mutatjuk be *(8. kép)*. A készülék egy mintanyomtatása látható a *9. képen*. A száraz elektrografikus lézertechnológia a fénymásolókhoz hasonlóan egy dobból választ le festékszemcséket a lézerfénysugár hatására. A nyomtató kompatibilis a HP Laser Jet típussal, és számos betűkerekes nyomtatót is képes emulálni. Nyomtatási teljesítménye 6 oldal percenként. A felbontás a szokásos 300 pont/hüvelyk. A lapokon lévő információ tárolására 128 kbájt memória áll rendelkezésre, de 512 kbájtra bővethető. 15 ún. nemzeti karakterkészlet a tartozék, ezenkívül további 53 a bővítési lehetőség. A tervezett élettartam évi 36 ezer oldal esetén 5 év.



8. kép



9. kép

Az ugyancsak más termékeiről ismert Atari ún. íróasztali kiadói rendszere (desktop publishing) egy Mega 2 típusú számítógépből, monokrom monitorból és az SLM804 típusú lézernyomtatóból áll. Ára a működtető szoftverrel együtt a szenzációnak számító 4000 dollár. A nyomtató az előzőhöz hasonló felbontás mellett nyolc oldalt készít percenként. Ezt úgy érték el, hogy a számítógép processzora és memóriája közvetlenül a nyomtató vezérlő elektronikájára dolgozik. Az átvitel egy DMA csatornán 10 Mbit/s sebességgel történik, közvetlenül a képprocesszorból. A karakterkészlet nem a nyomtató ROM egységében tárolódik, hanem szoftver állítja elő a számítógépben. A nyomtatót 2000 dollárért külön is árulják.

Simonyi

1. MIKROMAX hálózati kapcsolóüzemű tápegységcsalád

Készül Σ 55 W kimeneti terhelhetőséggel háromkimenetű változatban:

1. 5 V/10 A és 2 × 12 V/1,5 A
2. 5 V/10 A és 2 × 15 V/1,5 A

valamint 90 W kimeneti terhelhetőséggel egykimenetű változatban:

1. 5 V/18 A
2. 12 V/7,5 A
3. 15 V/6 A

EURO-rackbe szerelhető, de beépítés nélkül is használható.

2. TR—80 mikroprocesszoros kártyarendszer

Intel 8085 buszrendszer, ESZR méretű kártyák:

- központiegység-modulok (I8085, aritmetikai processzor)
- memóriamodulok (statikus, dinamikus RAM, tápfeszültségvédett RAM, EPROM)
- interfész modulok (soros, párhuzamos stb.)
- A/D, D/A és analóg-multiplexer modulok
- színes grafikus display-illesztő
- EPROM-égető és hardverellenőrző pult

Gyártó:

KONTAKTA Alkatrészgyár
Ózdi Gyáregység
3600 Ózd,
Bolyki főút 82.
Tel.: 06-47-12555

Felvilágosítás:

KONTAKTA Alkatrészgyár
Fejlesztési Intézet
Elektronikus Fejlesztés
Budapest
Tel.: 279-200/279

Tekintse meg egyéb új termékeinket is a Tavaszi BNV „A" pavilon 103/C standján!

Számítástechnikai Informatikai Fejlesztő Vállalat

Sci-L

Kereskedelmi Irodájának vezetésére

- **felsőfokú iskolai végzettséggel,**
- **számítástechnikai ismerettel és kereskedelmi gyakorlattal,**
- **legalább öt év vezetői gyakorlattal,**
- **nyelvismerettel**

rendelkező munkatársat keres.

Jelentkezés:

Marjai Sándor gazdálkodásiigazgató-helyettesnél, a 353-580 vagy a 350-140/103-as telefonszámon.

A BNV-n a holnap technikájával várjuk kedves érdeklődőinket!

A 28-AS PAVILONBAN

- ***Home computertől PPC-ig***
- ***Hardver, szoftver***
- ***Számítógépek, tartozékok, kiegészítők felhasználói és játékprogramok, valamint számítástechnikai szakkönyvek.***

Vállalkozunk kulcsrakész rendszerek kidolgozására
– helyzetfelméréstől és célkitűzéstől kezdve
– szervezésen, programozáson, dokumentáláson keresztül
– a rendszer betanítására, rendszerkövetésre.

HUNGRADE Rt.

1136 BUDAPEST,
Fürst S. u. 24–26.
Tel.: 530-022

DIGITAL szelvény Mikroszámítógép Magazin 1988. május

Egy sarokkal olcsóbb!!!

A DIGITAL Számítástechnikai Szaküzlet (Budapest, Szilágyi Erzsébet fasor 35. 1026) Sinclair-termékekre szakosodott: elsősorban a ZX81-es és Spectrum gépekhez használatos eszközöket, programokat árul. De kaphatók itt más gépekhez való tartozékok (például botkormányok), számítástechnikai alkatrészek (integrált áramkörök stb.) és zsebszámológépek is.

Aki ebben az üzletben a lapunkból kivágott sarokszelvényt átadja, vagy megrendelésével együtt oda elküldi, minden hónapban más-más cikket olcsóbban vásárolhat meg. A kedvezmény a szelvényen feltüntetett hónapban érvényes. Minden árengedménnyel vásárolt darabhoz le kell adni egy szelvényt.
A bolt utánvéttel szállítást is vállal, és a szokásos 6 hónap helyett *1 év garanciát ad.*

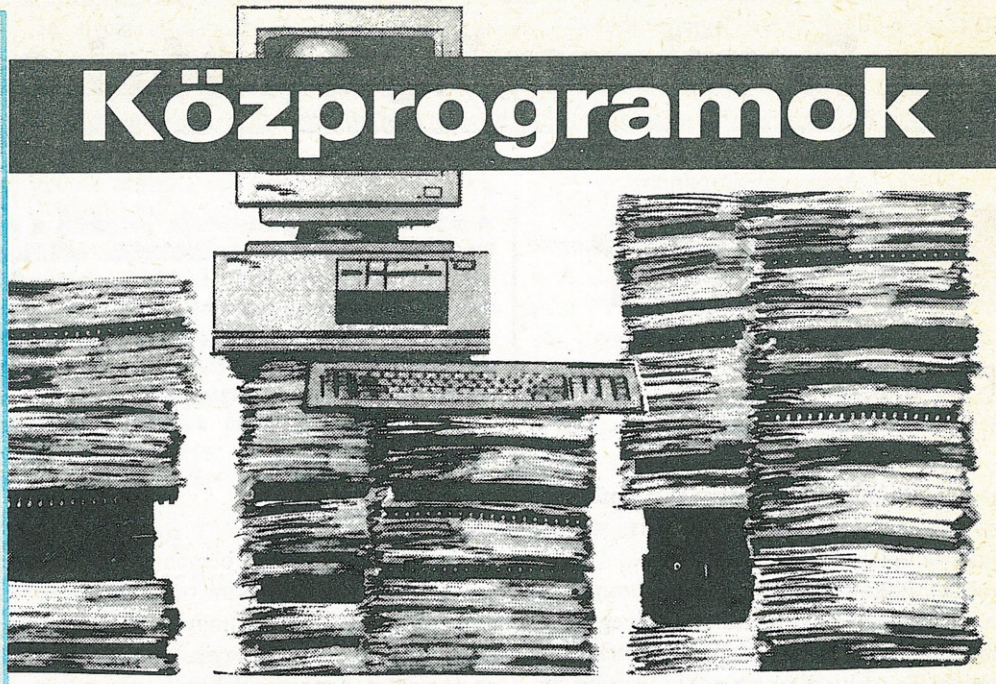
Az e havi kedvezmény

Egy botkormányos illesztő. Ára 1500,— Ft, engedmény 20%. Intelligens botkormányillesztő. Ára 2500,— Ft, engedmény 24%.

Az egy botkormányos illesztő a Spectrum gépek buszcsatlakozójára köthető kapcsolós (ugyanitt is kapható) botkormányok illesztésére szolgál. Ez megfelel a széles körben elterjedt és szabványnak mondható Kempstone típusnak. Bár sok játékot billentyűzetről is lehet játszani, de a botkormány használata kényelmesebb, és kíméli a billentyűzetet.

Az intelligens változat programozható, vagyis a számítógép beállított gombjait, mint irányvezérlőket használhatja. A tanulási kapcsoló helyzetben ezek a gombok a számítógépen kijelölhetők. Így billentyűzetvezérlésre készített programok is vezérelhetők botkormánnyal, nincs szükség a különböző típusú illesztőt igénylő játékokhoz más és más illesztőre.

Közprogramok



Olvasóink egyre gyakrabban küldenek be olyan hasznos programokat, amelyeket terjedelmi okok miatt nem áll módunkban közölni. Mivel azonban úgy véljük, hogy közérdeklődésre tarthatnak számot, ezeket röviden ismertetjük. Akik használni szeretnék a programokat, a részletes leírást, a programlistákat vagy – ahol erre mód van – a programot adathordozón, levélben megrendelhetik szerkesztőségünktől. A listák, ill. adathordozók másolását a KASZKAD Kisszövetkezet óbudai PÓLUS Szakcsoportja végzi (lásd lapunk 1988/2. számát). A másolási díj listák, leírások esetén oldalanként 8,— Ft, kazetta esetén 150,— Ft, floppy esetén 300,— Ft. Az adathordozóra történő másolás díja az adathordozó árát is magában foglalja.

Közületektől cégszerű megrendelést kérünk, a másolási díjat az MNB 208—42518—7014 számlára kell befizetni. Magánszemélyek a díjat a KKVMF PÓLUS Szakcsoport, Bp. Bécsi út 94—96. 1034 címre fizethetik be.

A megrendelés az alábbi formában lehetséges:

MIKROSZÁMÍTÓGÉP MAGAZIN SZERKESZTŐSÉGE
1371 Budapest, Pf. 433.

Megrendelem a Mikroszámítógép Magazin 1988/... számában szereplő közprogramok közül az alábbiakat:
A program neve
Csak programlistát és leírást kérek
Példányszám
Csak adathordozót kérek
Mindkettőt kérem

A megrendeléshez csatolom a szolgáltatási díj befizetését igazoló csekkszelvényt. Dátum, név, pontos cím.

A programok ellenőrzése nem áll módunkban, ezért az esetleges programhibákért mi nem vállalhatjuk a felelősséget.

Kérjük, akik saját készítésű programjaikat felajánlják a köz javára, a részletes programleírást, a listát és az alábbiakhoz hasonló rövidített ismertetőt juttassák el címünkre. (Nevük jelenti a garanciát a továbbiakban a megrendelők számára.)

Úszóverseny

Programnév:	Furst Kupa Úszóverseny
Géptípus:	C16
Konfiguráció:	nyomtató
Adathordozó:	nincs megadva
Terjedelem:	260 BASIC sor
A készítő neve:	Bőczén Csaba
Megjegyzés:	a gép bővítő használatát nem igényli
A leírás oldalszáma:	3
A program oldalszáma:	5

A program a VIII. osztályos tanulók számára is követhető és megérthető formában készült. Célja a programkészítés oktatásán kívül a verseny gyors, egyszerű adminisztrálása. A verseny lehet egyéni, csapat és váltó. A pontozás alapján rangsorolják a résztvevő iskolákat. A gép tármérete miatt iskolánként legfeljebb 50 tanulót és összesen legfeljebb 20 iskolát képes értékelni. A program az adatbeadási hibák ellen védelemmel rendelkezik.

Tanterv

Programnév:	Számítástechnikai tanmenet (kezdő)
Géptípus:	C16, Plus/4
Konfiguráció:	alapgép
Adathordozó:	leírás papíron
Terjedelem:	nem számítógépes program
A készítő neve:	Bőczén Csaba
Megjegyzés:	egy 52 óra terjedelmű tanmenet leírása
A leírás oldalszáma:	3
A program oldalszáma:	—

A tanmenet, amely általános iskolák részére készült — az ismerkedés a számítógéppel, gép—ember-kapcsolat, BASIC programozási ismeretek témaköröket öleli fel —, a tananyag rövid megnevezését, a bemutatandó feladatokat, a begyakorlandó feladatokat és az esetleges megjegyzéseket tartalmazza. Mindezekhez megadja a szükséges óraszámokat.

Játékprogram-nyilvántartó

Programnév	Játékprogram-nyilvántartó program
Géptípus:	C64
Konfiguráció:	VC-1541 vagy más kompatibilis lemezegység
Adathordozó:	lemez
Terjedelem:	193 BASIC sor
A készítő neve:	Nagy Ervin
Megjegyzés:	képernyő vagy nyomtató egyaránt használható
A leírás oldalszáma:	3
A program oldalszáma:	3

A programnyilvántartó program 999, lemezen tárolt program lemezen történő nyilvántartására szolgál. Az állomány bővíthető, módosítható, rendezhető, kiíratható, legfeljebb 33 karakterrel kommentálható.

Hardver

Kombinációs és sorrendi hálózatok

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozónak rendelkeznie kell alapfokú áramköri hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretekre.

A digitális áramköröket alapműködés szempontjából két nagy csoportra oszthatjuk. Az egyiket az ún. kombinációs hálózatok alkotják, melyek jellemzője, hogy valamely időpontbeli bemenő értékeik egyértelműen meghatározzák a hálózat kimeneti értékeit. Ilyenek voltak az eddig tárgyalt kapuáramkörök — hiszen a bemeneteik egyértelműen meghatározták a kimenetük logikai állapotát —, valamint a kapuáramkörök összekötéséből kialakított összetettebb hálózatok. A másik csoportot a sorrendi vagy más néven szekvenciális hálózatok képezik. Ezekre az jellemző, hogy kimeneteiket a bemenő jelek mellett a hálózat előző állapota is befolyásolja. A bemenő változók azonos kombinációihoz más-más kimenet tartozhat, attól függően, hogy mi volt a hálózat előző állapota. Ezért nevezik ezt sorrendi hálózatnak. Mivel a hálózat előző állapota befolyásolja az aktuális állapotot, szükség van az állapotok tárolására, mert ezek a tárolt állapotok — logikai kapcsolatban a bemenő jelekkel — határozzák meg a kimenet(ek) állapotát.

A tárolás elemei a tárolók vagy közismert idegen szóval flip-flopok. A tárolók kapukból is előállíthatók, azok megfelelő összekapcsolásával. Példaként vizsgáljuk meg, hogyan működik az egyik bemenetükön invertert tartalmazó VAGY (OR) kapukból kialakított RS tároló *(1. ábra)*.

A további rajzokon gyakran látunk a kivezetéseken egy számot. Ennek az az oka, hogy a rajzok egy áramkörtervező programmal készültek, és a lábszámok mindig egy adott konkrét típus lábkiosztását jelentik. Néhány megjegyzés a rajzokkal kapcsolatban. A kapukat vagy más logikai egységeket téglalapok jelzik. A közvetlen be- és kimenetek a téglalap oldalához húzott vonalak, általában olyan módon, hogy a bemenetek az egyik, a kimenetek a másik oldalra kerülnek. Az inverz (negált) bemenetek vagy kimenetek kis üres körrel csatlakoznak az oldalakhoz, és azt jelölik, hogy a kör másik oldalán a jel inverze jelenik meg.

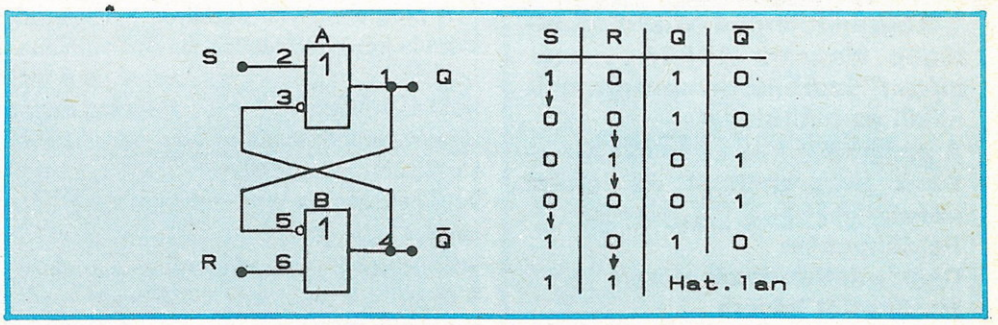
Adjunk az S bemenetre H szintű, az R bemenetre L szintű jeleket. Az A jelű kapu kimenetén H szint jelenik meg. Ez a B kapu egyik bemenetére is kapcsolódik. Ez, illetve az inverze és az R bemeneten lévő L szint együttesen a kapu kimenetét L szintbe állítja. Láthatjuk tehát, hogy van egy hurok, amely a kapcsolás egyik felében stabilan H szintet, a másik felében L szintet tart fenn. Az S oldali Q kimeneten a H szint ezért akkor is megmarad, ha az S bemenetre adott H szintű jel megszűnik, mindaddig, amíg az R bemenet L szintű marad. Analóg módon a két kimenet állapotot vált, ha az R bemenetre kerül H, S bemenetre pedig L szint. A kapcsolás tehát tároló, amelynek ha az S (Set = beíró) bemenetére H szintű jelet adunk, ezt tárolja, ha pedig az R (Reset = törlő) bemenetére adunk jelet, akkor a beírást törli.

A tároló két kimenetén a tárolt érték és annak fordítottja (= inverze = negáltja) is megjelenik. A tároló logikai működése a *táblázat* alapján elemezhető.

Ha S és R egy időben H szintű, akkor a tároló mindkét kimenete H szintű lesz. Ez a kimeneti állapot azonban nem egyértelmű, határozatlan, ha utána S és R egy időben L szintre változik, ezért ez az eset rendszerint nem megengedett.

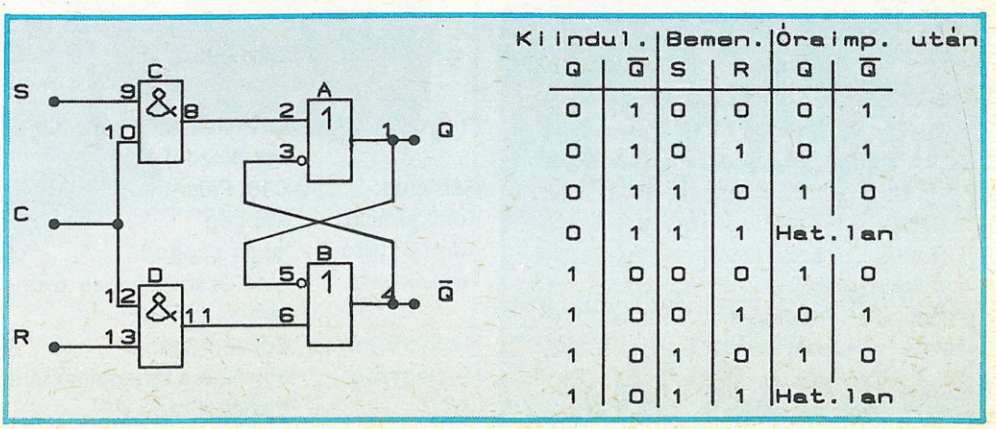
A fenti megoldásnál a tároló állapotváltozása a bemenő jel megváltozásával egy időben következik be. A biztosabb, zavarmentesebb működés érdekében célszerű a tárolót úgy működtetni, hogy a bemeneteire a jeleket azok lezárt állapotában adjuk rá, majd a tároló alapállapot-változását egy külön impulzussal (órajellel) váltjuk ki. Az ilyen, órajellel vezérelt (szinkron) RS tároló alapkapcsolása a *2. ábrán* látható.

Ha az S bemenetre H szintet adunk, a C jelű kapu kimenete csak akkor lesz H szintű, ha a C órajel bemenetre H szint érkezik. A C kapu kimenetén megjelenő H szint már a Q kimenetet H szintbe állítja, ahogy ezt már az előző esetben leírtuk. Hasonló módon a törlést is csak az R szintre kap-



1. ábra

2. ábra



csolt H szint mellett a C órajel H szintre váltása hozza létre.

Az egyszerű R-S flip-flop működésében mutatkozó határozatlan állapot elkerülésének egyik lehetséges módja az, hogy csupán egy bemenetet (D bemenetet) alkalmazunk. A D flip-flop logikai kapcsolását a *3. ábra* mutatja.

A D bemeneten lévő információ a Q kimenetre csak akkor jut el, ha az órajel bemeneten az átíró jel megjelenik. A tárolók vagy flip-flopok a különféle típusú számláló áramköröknek is alapelemei.

Ezek után foglaljuk össze a digitális logikai áramköröket.

Vannak KOMBINÁCIÓS és SORRENDI logikai hálózatok.

A SORRENDI hálózatokat működési módjuk szerint tovább osztályozhatjuk aszinkron és szinkron áramkörökre. ASZINKRON SORRENDI hálózat esetén a bemenő jelek megváltozása azonnal létrehozza — a belső állapottól függően — a kimenetek megváltozását. A kimenő jel kialakulásában az egyes kapuk áramköri késlelletése is szerepet játszik. SZINKRON SORRENDI hálózatoknál ez a megváltozás egy külső „karmester" — az órajel — parancsára következik be.

Mivel az órajel váltakozó H-L szintekből álló impulzussorozat, ezért egy további osztályozás szerint, ha az állapotváltozás az órajel szintjére (például mikor H szintű) következik be, akkor szintvezérlésről beszélünk. Ha a változás az órajel szintjének változására (L-H szintváltás vagy felfutó él, H-L szintváltás vagy lefutó él) következik be, élvezérelt a működés. Az áramköri rajzokon az élvezérelt (élre billenő) bemenetet kis háromszög jelzi. A gyakorlatban a legtöbb digitális rendszer szinkron működésű, órajel vezérlésű, ezért a működési sebességet az órajel frekvenciája határozza meg.

Aszinkron logikai hálózatok a hálózatban terjedő jelek futási idejének különbözősége miatt hibás állapotokba kerülhetnek. Ezek a hibák az ún. hazárdok. Mivel az aszinkron tervezésnél az áramkörgyártási technológia is szerepet játszik, ezért terjedtek el jobban a szinkron működésű hálózatok.

A TTL és CMOS áramkörcsalád

Kissé bonyolultabb és itt nem részletezett módon, a P-N átmenet vezetőképességét elektromos erőtérrel is vezérelhetjük. Az ezen az elven működő tranzisztorokat FET-eknek — térvezérlésű tranzisztoroknak — nevezzük. A vezető csatorna lehet P, illetve N típusú. A CMOS jelölés az integrált kivitelű N és P csatornás FET-ekkel kialakított áramkört jelöli. A bipoláris tranzisztorokat alkalmazó TTL digitális integrált áramkörökkel egy időben fejlesztették a CMOS áramköröket, mert számos előnyös tulajdonsággal rendelkeznek:

— egy bipoláris tranzisztor helyén 3-5 FET helyezhető el,

— az áramkörök bemeneti ellenállása igen nagy,

— feszültséggel vezérelhető (nem kell vezérlő áram),

— a tápfeszültség értéke tág határon belül tetszőleges, értéke 2-3 V-tól 10-20 V-ig változhat,

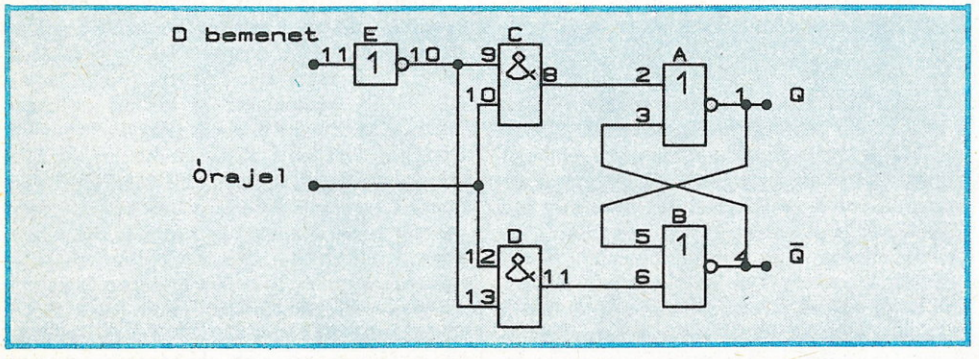
— az áramkörök fogyasztása elenyésző,

— az üzemi hőmérséklet −40—+85 °C között lehet.

A CMOS integrált áramköri elemcsalád — a TTL áramkörökhöz hasonlóan — igen sok olyan elemet tartalmaz, amelyek funkcionális működésükben megegyeznek. A CMOS áramköröknél elérhető nagy elemsűrűség lehetővé teszi, hogy igen összetett áramköröket hozzunk létre.

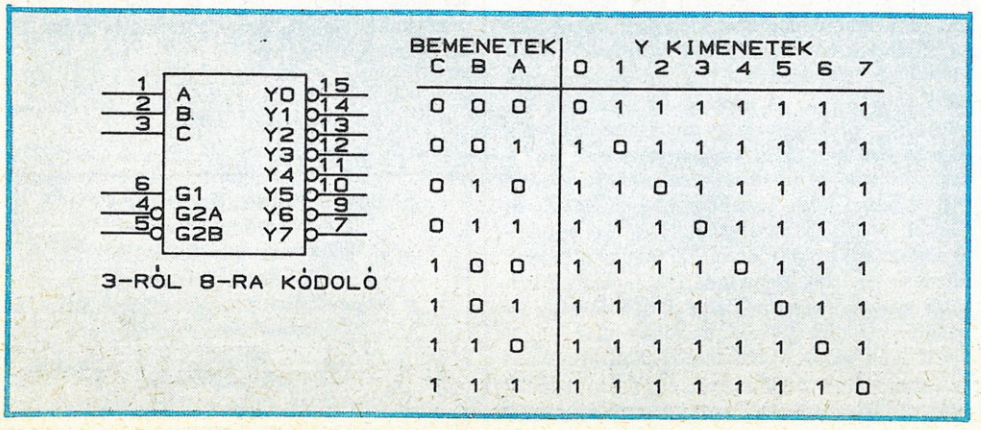
A digitális áramkörök tervezői már általában „keverve" használják az áramköröket — mindig az előnyösebb tulajdonságokat mutató család elemét részesítve előnyben. Ez a tervezési módszer természetesen azt is igényli, hogy pontosan tisztában legyünk a be- és kimenetek terhelési viszonyaival. Egy kimenet túlzott terhelése ugyanis (túl sok bemenet kapcsolódik hozzá) az L, illetve H logikai feszültségszinteket — amelyek egy tartományba kell hogy essenek — úgy elhúzza, hogy hibás logikai szint alakulhat ki. Például TTL áramköröknél az L szintnek a 0... 0,8 V-os feszültségtartomány, a H szintnek a 2,4... 5 V-os tartomány felel meg.

A következőkben a tényleges áramköri típusoktól elvonatkoztatva, röviden áttekintjük a digitális integrált áramkörök legfontosabb, a mikroprocesszor-technikában használt elemeit. Elsőként a kombinációs hálózatokkal megvalósítható egységeket, utána a sorrendi hálózatokat tartalmazókat tekintjük át.



3. ábra

4. ábra



BEMENETEK			Y KIMENETEK							
C	B	A	0	1	2	3	4	5	6	7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Kódoló áramkörök

A kódoló áramkörök az információnak az egyik ábrázolási formáról egy másik ábrázolási formára való átalakítására szolgálnak. Sok esetben van szükség olyan áramkörre, amelynek bemenetére egy bináris értéket kapcsolva, annak csak egyik — az adott bináris értékhez tartozó — kimenetén jelenik meg jel (például L szintű, amikor a többi kimenet H szintű). Az ezt megvalósító, 3-ból 8-ra átkódoló áramkör elvi rajza a *4. ábrán* látható.

Az A, B, C bemenetek értékétől függően lesz a megfelelő kimeneten L szint, míg a többin H szint. Például ha A=L, B=H, C=H, akkor a 6. jelű kimenet lesz L szintű. A G1, G2A, G2B jelű bemenetek a kódoló működését engedélyezik vagy tiltják. A jelen esetben az Y kimenetek valamelyike csak akkor lehet L szintű, ha G1=H, G2A=L, G2B=L feltétel teljesül.

(Folytatjuk)

DR. KÓNYA LÁSZLÓ

Mikrokalauz

Verseny van, de nem az igazi!

Volt idő, amikor azt mondták: szakáll és bajusz nélkül a férfi nem férfi. Mintha a hímnem e szőrzetek híján nemtelenebb lett volna! A közfelfogásban vallott ilyen nézet persze ma már túlhaladott, nevetünk is rajta, hozzátéve: azért a szakáll öltöztet!

Félrerakva az iróniát, tréfát — mert a számítástechnika nem tréfa —, de azért a játékos komolyságot megtartva mondhatjuk: aki maga vagy más szakállára számítógépes fejlesztésbe fog, a programok beszerzésekor könnyelműségét felejtse otthon. Hogy miért? Mert „A szoftver értéke a számítógépes rendszer árán belül a fejlett országokban ma már a 80—90 százalékot is eléri. Ez a tendencia az utóbbi években felgyorsult ütemben érezteti hatását hazánkban is . . .” — írják a Mikrokalauz harmadik kötetének szerzői. S az értékeket és értékrendet hangsúlyozzák azzal is, hogy a hazai szoftverpiacot különös tekintettel az IBM PC-vel kompatibilis programokra tekintik át. A program mint érték tehát önmagában nem evidencia. Becse, hasznossága, gazdaságossága éppen a számítógépek programokkal való felöltöztetéséből adódik. A programtól függ, hogy a számítógép nemes vagy „nemtelen” lesz-e.

Természetszerűleg a szerzők ezúttal is tisztáznak néhány alapfogalmat. Így például a szoftvertermékét, ami a programtermék és a dokumentáció együttese. A szoftvertermékeket azután a piaccal való találkozásuk szempontjából két csoportra lehet bontani. Ha egy program meghatározott, speciális igény kielégítésére készül, az egyedi programtermék. A többszöri értékesítésre szánt programtermék eleve eladásra és nem megrendelésre készül. A magyarországi szoftvertörténet jelentős állomása a mikroszoftverpiac kialakulása. „A hazai piacon 1982-től jelentek meg a mikroszámítógépek, tulajdonképpen ekkor kezdődött a hazai mikroszoftver korszak is. Az első itthoni gyártású mikroszámítógépek hazai fejlesztési eredményt képviseltek, s ez az elv a programvonalon is érvényesült. Az eredmény az lett, hogy minden gép másféle, speciális operációs rendszerrel működött, sajátos programnyelvet használt.”

Programiparunk kialakulásában meghatározó változás volt megfigyelhető a nyolcvanas évek közepén. Ekkor ugyanis a számítástechnikával foglalkozó szervezetek száma ugrásszerűen emelkedett. Viszont „annak ellenére, hogy ezeknek a szervezeteknek a száma meghaladja a négyezret, közülük csak néhány tucat foglalkozik programáru-termeléssel”. Mégis: „Hazánkban a programkereskedelem egyelőre még hardverorientált, ami azt jelenti, hogy a számítógép-kereskedő könnyebben tud eladni programot géppel együtt. Emiatt a programgyártók arra törekednek, hogy saját vagy idegen számítógépet forgalmazzanak.” Az egyedi programtermékek iránt egyébként változatlanul nagy az érdeklődés. Ez a kereslet a kisszervezetek (kisszövetkezetek, szakcsoportok, vgm-ek stb.) felfutását eredményezte. „A kisszervezetek a hazai programipar szerkezetét lazították, azonban nagymértékben kötődnek a nagyszervezetekhez.”

Ez a gondolat rögtön átvezet minket a programmonopóliumok fonákságaihoz. A monopóliumok nagy vonalakban két mag körül alakultak ki. „Minden ágazatban létrejött egy vagy két ágazati számítástechnikai szervezési központ.” Igaz ugyan, hogy a vállalatoknak ilyen jellegű gondjaikkal nem feltétlenül kell hozzájuk fordulni, de ma „. . . amikor a vállalati önállóság még messze van az ideális szinttől, és amikor működnek még az informális irányítás jól ismert csatornái, nem nehéz elképzelni, hogy sok vállalat erkölcstelennek, esetleg kockázatosnak tartja, ha nem az ágazati számítástechnikai szervezési intézeteket keresi meg először”. Másrészt monopolhelyzet alakult ki az ország olyan vidéki körzeteiben, ahol csak egy vagy két program-előállító szervezet működik. Mindezek következtében verseny ugyan van a piacon, de ez nem a programtermékeké, hanem a programkészítő cégek versenye. Így tehát „. . .amíg nem alakul ki a kereslet-kínálat globális és strukturális egyensúlya, addig valóban az dönt, hogy melyik cégnek van jobb és szerteágazóbb kapcsolata, ismeretsége”.

A programpiac ilyetén belterjessége, homogenitása közvetett módon kihat a programkészítésre is. „Olyan szemléletváltásnak kellene bekövetkeznie, hogy elsősorban a meglévő építőkockákból tudják a programrendszert összeállítani, nem pedig különböző programozási nyelveken megírni azt.” Noha ez a folyamat megindult, nem eléggé dominál még. Továbbá hiányzik a programokhoz tartozó dokumentációs szabvány: a programcsomag készítésének nincs elfogadott módszere és gyakorlata, megfogalmazott követelménye. „Általában a programkészítésnél csak a felhasználó jelenlegi tevékenységét algoritmizálják, a manuális munkát viszik gépre, elmarad a számítógépesítés legnagyobb előnye, a racionalizálás. A vezető szoftveresek egy része nem tud feladatot fogalmazni, kapacitást és átfutási időt, gépidőszükségletet és költséget becsülni.”

Ebből következnek a keresletben és kínálatban az anomáliák — amiknél óhatatlanul felmerül a gondolat: a fejlődés szükségszerű velejárója ezek sokasága, amit a szerzők frappánsan érzékeltetnek. A kínálati oldalból kiragadva: a szoftver követése gyakran költséges, aránytalanul leköti a szellemi kapacitást. A napjainkig kialakult kínálat összességében sem mennyiségileg, sem minőségileg nem kielégítő, szélsőségek is fellelhetők. A kereslet felől közelítve a kötet szerzői egyfajta hiányból tulajdonképpen javaslattal is előállnak. „A jelenlegi körülmények legnagyobb hiányossága, hogy nincs komplex problémamegoldás . . . Nincs olyan szervezet, amelyhez fordulva a potenciális számítógép-vásárló megtudhatja, hogy konkrét problémáját milyen géppel, milyen kiépítésben, milyen rész- és egyedi fejlesztésű programtermékkel, milyen időbeli ütemezéssel lehet megoldani.” Magyarán a belföldi programforgalomról, piacról jóformán lehetetlen megbízható információhoz jutni.

Cikkünk elején értéket említve kimondatlanul is a programok forgalmi értékére, árára is céloztunk. Nos, „. . . néhány, csak központilag forgalmazott alapprogram esetében az ár nem alku tárgya: ha a vevőnek valóban szüksége van az adott programra, nem fordulhat máshova”. A programok többségében azonban az áru „szabad préda”. Ennek több oka van, mint például az, hogy a kereslet jóval meghaladja a kínálatot: rengeteg a párhuzamos fejlesztés, így a többletköltség. Általános vélemény, hogy „. . . a magyar programtermékek és a hozzájuk kapcsolódó szolgáltatások drágák, sőt nagyon drágák.”

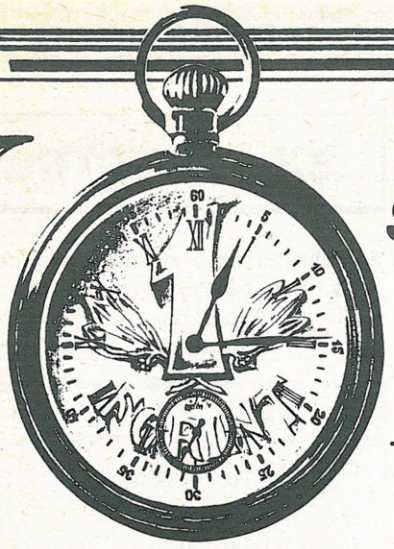
A pénzügyi szabályozókról szóló fejezet azért kapott a kötetben helyet, mert a program, mint sajátos vagyontárgy, különleges gazdálkodási, pénzügyi, számviteli szabályozásokat és gyakorlatot követel. Ezért döntött úgy 1985-ben a Minisztertanács, hogy a Központi Statisztikai Hivatal elnöke a számítástechnika-alkalmazás területén a gazdasági miniszterek hatáskörében járhat el. Az ezt követő fejezet a számítástechnika szerzői jogi vonatkozásait taglalja. (Ezt most mellőzzük, mivel a későbbiekben több cikkünk részletesen kíván ezzel foglalkozni.) Izgalmas tizenkét pontos kiskátét is összeállítottak a szerzők, mintegy programbeszerzési útmutatóként. Egészen a számítógépes igények felmérésére való utalástól a tanácsadásig, támogatásig, más felhasználókkal az együttműködés elengedhetetlenségének bizonyításáig.

Ezt követően végre témánként felsorolják, röviden ismertetik az egyes általános célú programcsoportokat, míg egy külön kötetben (III/2) található az IBM PC-vel kompatibilis hazai szoftverkínálat tételes jegyzéke.

Ezúttal csak felhívjuk a figyelmet a Mikrokalauz negyedik kötetére, amely a hazai számítástechnikai irodalmat dolgozza fel, az 1979-től kiadott hazai számítástechnikai könyvek alapján, témánkénti, kiadónkénti és kiadási évenkénti mutatókkal. E helyt érdemes lejegyezni, hogy az Országos Széchényi Könyvtárban működik az Országos Könyvkatalógus (558-058), amely egy könyv címe alapján felvilágosítást ad arról, hogy az mely hazai könyvtárban található. Ez a katalógus a teljes hazai, azaz 1952 után megjelent számítástechnikai irodalmat magában foglalja.

Befejezésül keltsük életre Bod Péter XVIII. századi tudósunk szavait: „Mi az, ami igen nehéz, s mégsem terheli azt, aki hordozza? Az a tudomány és a jó mesterség, amely terhére nincsen annak, akinél vagyon, hanem könnyebbíti sok bajaiban.” Józserével a számítástechnika genezisét éljük: tudományunkon és becsületbeli mesterségünkön múlik, hogy a teremtés súlya lehúz-e avagy felemel.

SZULYOVSZKY CSABA



A NOVOTRADE SZJA '88 programja az ön személyes adószakértője és adókönyvelője!

A program Commodore–64, Commodore Plus/4, Enterprise, Videoton TV-Computer, ZX Spectrum és IBM PC kompatibilis személyi számítógépeken működik.

Az SZJA '88 program **KISZÁMOLJA:** a befizetendő személyi jövedelemadót.
Az SZJA '88 program **NYOMON KÖVETI:** használójának jövedelemalakulását.
Az SZJA '88 program **FIGYELEMBE VESZI:**
a várható, illetve tervezett jövedelmeket a tényleges jövedelem és kiadások alapján.
Az SZJA '88 program: az adójogszabályok teljes körét feldolgozza.

A *Commodore Plus/4-re* a Művelt Nép és Állami Könyvterjesztő 20 szaküzleteiben, a Vídia boltjaiban és a Magister Stúdium könyvesboltokban keresse.

Az összes többi változat a CENTRUM ÁRUHÁZAKBAN kapható.

RAINBOW
Számítástechnikai és Szolgáltató Kisszövetkezet

D–Subminiatur csatlakozók, kábelek, analóg-digitál átalakítók, egyéb számítástechnikai berendezések és kellékek nagy választékával várja Önöket

a MAGÉV–RAINBOW Elektronikai Szaküzlet

Budapest VI., Rudas László u. 33.
Telefon: 122-392. Telex: 22-6323

RAINBOW Számítástechnikai és Szolgáltató Kisszövetkezet

Központ: Budapest, Szilágyi Erzsébet fasor 1. 1026
Levélcím: 1378 Budapest 64., Postafiók 31. Telefon: 352-558

A billentyűkezelő rutin sajátosságai

ZX—SPECTRUM

Gépi kódú programok írásakor szükség lehet a billentyűzet használatára: programozás közben, amikor kipróbáljuk az új programot, vagy egy menüből kell választani, vagy a program futását akarjuk leállítani.

Kezdő programozó vagyok, ami a gépi kódot illeti. Sajnos sokszor kerültem végtelen ciklusba egy-egy hibás utasítás miatt. Ilyenkor természetesen újra be kell tölteni az Editor/Assembler programot, valamint az „alkotás"-t. Ez igen időrabló és idegesítő. Felmerült tehát a kérdés: hogyan állíthatom le a gépi kódú programot futás közben? Olyan megoldást kerestem, hogy véletlen billentyűlenyomás ne állíthassa le a programot. Az általam ismert kézikönyvekben található kódtáblázatokkal nem elégedtem meg, hiszen ezek nem tartalmaznak minden variációt. Természetesen közrejátszott eredendő kíváncsiságom is: minden lehetséges kombinációra érvényes táblázatot akartam kidolgozni. Ekkor kezdtem a Spectrum ROM billentyűkezelő rutinjának részletes elemzéséhez.

Mit tudunk?

A Spectrum billentyűzete négy sorban 10-10, összesen 40 billentyűből áll. A billentyűzetet mint perifériát a 254-es porton lehet elérni. A billentyűzet 8 fél sorra van felosztva, egyenként 5 billentyűvel:

1. CAPS SHIFT—V
2. A—G
3. Q—T
4. 1—5
5. 0—6
6. P—Y
7. ENTER—H
8. SPACE—B

A billentyűzetet leolvasó szubrutin ezeket a fél sorokat látja: a 8 fél sornak megfelelően nyolcszor hajtódik végre. A rutin befejezésekor a B, D, E regiszterekben a lenyomott billentyűnek megfelelő értékek találhatók.

Nézzük tehát konkrétan

A rutin a 654 (Ø28Ehexa) címen kezdődik és 48 bájt hosszú. A program RET utasítással ér véget. Ez egyben azt is jelenti, hogy a szubrutin meghívása után a vezérlés visszakerül eredeti programunkhoz. Ennek a szubrutinnak a használata azért is előnyös, mert bármely megszakítási módban működik. A program nem gondoskodik a regiszterek tartalmának elmentéséről.

Az E regiszter tartalma a különböző billentyűk lenyomása után

A	38	K	17	U	10	5	4
B	0	L	25	V	7	6	3
C	15	M	16	W	29	7	11
D	22	N	8	X	23	8	19
E	21	O	26	Y	2	9	27
F	14	P	34	Z	31	0	35
G	6	Q	37	1	36		32
H	1	R	13	2	28	SYS	24
I	18	S	30	3	20	CS	39
J	9	T	5	4	12	ENTER	33

1. táblázat

1. lista

```
   10         ORG   50000
   20
   30 ;KEZDŐCÍM
   40 ;ATHELYEZHETO BARHOVA
   50
   60 ;EGY REGISZTER VIZSGALATA
   70
   80 START
   90         LD    B,10
  100 VAR     HALT
  110         DJNZ  VAR
  120
  130 ;LASSITJA A PROGRAMOT
  140
  150         CALL  654
  160         LD    A,B
  170         ADD   A,100
  180         RST   16
  190
  200 ;A VIZSGALT REGISZTER
  210 ;TARTALMANAK (+100)
  220 ;KIIRASA
  230
  240         LD    A,D
  250         CP    24
  260         JR    NZ,START
  270         RET
  280 ;KILEPESI LEHETOSEG:
  290 ;SYS + BARMELY BILLENTYU


50000  06 0A 76 10   ..v.
50004  FD CD BE 02   }M..
50008  78 06 64 07   xFdW
50012  7A FE 18 20   z~. 
50016  EF C9 00 00   oI..
```

A D regiszter tartalma a különböző billentyűk lenyomása után

nincs billentyű lenyomva vagy egy van lenyomva	255
CAPS SH. + 1 billentyű	39
CAPS SH. + 2 vagy több billentyű	40
SYSH. + 1 billentyű (kiv. CSH)	24

2. táblázat

2. lista

```
  10 ;EGY REGISZTER VIZSGALATA
  20 ;PROGRAMBOL VALO KILEPES
  30 ;SYMBOL SHIFT +
  40 ;BARMELYIK BILLENTYU
  50 ;MEGNYOMASARA
  60 ;A D REGISZTERBEN 24 LESZ
  70 ;KIVETEL CAPS SHIFT!
  80
  90 START   PUSH BC
 100         PUSH DE
 110         PUSH HL
 120
 130 ;REGISZTEREK TARTALMANAK
 140 ;KIMENTESE
 150
 160         CALL 654
 170
 180 ;ROM RUTIN HIVASA
 190
 200         LD    A,D
 210         CP    24
 220
 230 ;D REGISZTERBEN 24 VAN-E
 240
 250         POP   HL
 260         POP   DE
 270         POP   BC
 280
 290 ;REGISZTEREK TARTALMANAK
 300 ;VISSZAIRASA A VEREMBOL
 310
 320         JR    NZ,TOVABB
 330 FOLYT
 340 ;ITT FOLYTATODIK, HA A
 350 ;FELTETEL TELJESUL. PL:
 360         RET
 370
 380 TOVABB
 390 ;ITT FOLYTATODIK, HA A
 400 ;FELTETEL 'NEM' TELJESUL
 410 ;PL:
 420         JR    START
```

Hogyan lehet a regiszterek tartalmát megvizsgálni?

Az *1. lista* szerinti programot azért kell lassítani, mert különben pillanatok alatt tele lesz a képernyő. A ciklus tíz HALT utasítást hajt végre. A vizsgálandó regisztert a 160-as sorba kell beírni. Értékéhez azért kell százat hozzáadni, mert a 32 alatti számok helyett vagy kérdőjel jelenik meg,

A B regiszter tartalma a különböző billentyűk lenyomása után	
0, 1, 2 billentyű lenyomása esetén	254
SYSH. + 2 billentyű	127
E üzemmód + 1 billentyű	127
G üzemmód +	
8. fél sor 1 billentyű	127
7. fél sor 1 billentyű	191
6. fél sor 1 billentyű	222
1-5. fél sor 1 billentyű	238

3. táblázat

vagy hibaüzenettel leáll a program. A kilépési lehetőségre azért van szükség, hogy a gépi kódú programot le tudjuk állítani. Ez egyben példa az eljárás használatára.

A C regiszterben mindig 254, a többi regiszterben az aktuális billentyűnek megfelelő érték található. Az E regiszter *(1. táblázat)* a lenyomott billentyűnek megfelelő számot tartalmazza 0—39-ig. Ha nem nyomtunk le billentyűt, az E regiszter tartalma 255. Ha több billentyűt egyidejűleg nyomkodtunk, a legváltozatosabb értékeket fogjuk kapni. Erre táblázatot készíteni — véleményem szerint — nem érdemes. A D regiszter értékeit a *2. táblázat*, a B regiszter értékeit a *3. táblázat* tartalmazza. A táblázatokban nem szereplő variációkat egyenként kell kikísérletezni, az E regiszternél leírtak alapján.

További előnyök

Az eljárás használatát a *2. lista* mutatja. A program elején a regiszterek értékét a verembe ki kell menteni, mert a ROM rutinja megváltoztatja azok tartalmát. Mivel a POP utasítás a jelzőbiteket nem változtatja meg, a feltételes ugrás hibátlanul működni fog. Ha ezt a rutint beépítjük egy gépi kódú programba, nemcsak a BASIC-be való visszatérést, hanem az avatatlanok elől „rejtett" szubrutinhívást is megoldhatjuk.

DR. GRESZ MIKLÓS

ADOK—VESZEK—CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket 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

C16-os, memóriabővítővel (64 k), magnóval, 210 db 16-os, 20 db Plus/4-es programmal olcsón eladó. Érdeklődni 19-21 óra között a 860-256-on.

Commodore 16-os számítógép eladó, magnóval, gépkönyvekkel, sok programmal 18 000 Ft-ért. Makó, Beloiannisz u. 128/A, 6900

Commodore 64-es programok nagy választékban kaphatók, kazettán és lemezen. Sebestyén Imre, Budapest, Őzgida u. 18/C. 1025

C64 programok olcsón, nagy választékból. Bohner Tamás, Baja, Széchenyi u. 8. 6500

C64-es számítógép 1541-es floppyval, Seikosha SP-180 nyomtatóval eladó. Amigára cserepartnert keresek. Raiz Tamás, Budapest, Győri u. 8. 1123, Tel.: 566-941

Commodore, Enterprise, TVC és **Atari** számítógépekhez HW cikkek széles választékát kínáljuk. Joystick-ok 450 Ft-tól kaphatók. Fényceruza 2000 Ft (C16-hoz és Plus/4-hez is). Válaszboríték ellenében elküldjük katalógusunkat. COMPUTEAM GM. Kaposvár, Berzsenyi u. 32. 7400

Adok egy db **Super Schot joystickot**. Bóna Zsolt, Kecskemét, Tel.: 47-788, 17 óra után.

Spectrum programkazetták olcsón eladók, Magyarországon még ismeretlen, teljesen új, színvonalas játék és felhasználói programokkal. Röviddel az angliai megjelenés után! Postacím: Siklós, Pf.: 129, 7800. Küldjön válaszborítékot!

Spectrum-ja lekopott (fém) fedőlapját olcsón, gyorsan felújítom. Képiró Róbert, Budapest, Imre u. 5. II/14. 1093. Tel.: 378-075

Új ZX-Spectrum Plus (13 500 Ft) és **ZX-Microdrive** (4500 Ft) eladó. Budapest, Stoczek u. 5-7. 516 sz. 1111

VESZEK

C 64 zongora klaviatúrát programmal keresek (pl. Wersiboard Music 64). Tel.: 49/14-948

HT 1080Z (16 k) gépet veszek, esetleg két db-ot is. Irányárat kérek, minden levélre válaszolok. Márton Péter, Törökbálint, Katona J. u. 7. 2045

Videoton TVC gépemhez játékprogramot vennék, kazettán. Sinka Edit, Pusztaföldvár, Rákóczi u. 172/a. 5919. (Levélben!)

CSERÉLEK

Atari 800XL-re készült programokat cserélek, ötszáz játék és felhasználói program. Gayer Ferenc, Budapest, Gubó u. 6. I/10.1151

Commodore Amiga programokat cserélek. Lenhardt Tamás ADT, Győr, Munkásőr u. 75. 9023

C16 és **Plus/4**-es programokat cserélek. Farkas Tibor, Zalaszentgrót, Felszabadulás u.1. 8790

C16, Plus/4-es programokat cserélek. Kunsági László, Nyírbátor, Vasvári Pál u.1. 4300

C64-es játék- és felhasználói programokat cserélek, elsősorban kazettán. Válaszokat lehetőleg listával kérek. Aradi László, Szentes, Nagyörvény u. 111. 6600

Enterprise programokat cserélek. Veresegyház, Pf.: 28. 2112

SHARP MZ-800 tulajdonosok jelentkezését várom. Szoftver-hardver csere! Kánnár László, Budapest, Zápor u. 16/b. 1034

Spectrumot (48 k) adok kompletten, ráfizetéssel, régi, de jó állapotban lévő **C64** + floppyért. Ajánlatokat kérek! Marincsák János, Kisvárda, Lenin u. 20. 4600

ZX-Spectrum játékprogramokat cserélnék, a válaszokat listával kérem. Jancsin Ferenc, Békéscsaba, Lencsési u. 8. 5600

ZX-Spectrum (48 k) programokat cserélek. Listát vagy kazettát kérek: Platthy Péter, Budapest, Mártírok útja 31-33. fsz.2. 1027

ZX-Spectrum+ (48 k) játék- és felhasználói programokat cserélek. Válaszokat listával kérek. Miszori Róbert, Ajka, Újélet u.6. 8400

ZX-Spectrum-os cserepartnereket keresek. Csécsei József, Kartol, Vörös Hadsereg u. 55. 2173

TV-Computer játékprogramokat cserélek vagy könyveket veszek. Fogarasi Ferenc, Budapest, Zsókavár u. 28. VIII/34. 1157

VC20 alapgépre programokat cserélek. Nagy Tamás, Nagykanizsa, Kazanlak krt. 7/B. II/8. 8800

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

Petike naplófőkönyvet egyeztet

Apuka egy füzet fölé hajol. Homlokán a nagy figyelem szélesráncokat gyűr. A füzet olyan hosszú, hogy vagy lecsúszik az asztal valamelyik oldalán, vagy apukát kényszeríti teljesen kicsavart testhelyzetbe. Petike némán, figyelmesen néz. Végül is apuka nem bírja tovább és az asztalra dobja ceruzáját.

— Nem találom a hibát! Már háromszor átnéztem az összes sort, és nem tudom, miért nem egyezik a naplófőkönyv! — mérgelődik, és idegesen átrohan a másik szobába.

Petike nézegeti a hosszú füzetet és az egyeztető táblát, amely mechanikusan mutatja az egyezőséget. Gyorsan átlátja, hogy két meghatározott oszlopba írt számok összegének kell egyeznie. Tehát a programnak (mert Petike azonnal elhatározza, hogy számítógéppel ellenőrzi apuka munkáját) két megfelelő oszlopba kell gyűjtenie a naplófőkönyvbe írt adatokat. A további számítások miatt ezeket legegyszerűbb két tömbbe betölteni. Az adatok azonban nem egymás után következnek, hanem elszórtan, egy későbbi oszlopban. Ezért az adatokat névvel kell ellátni, hogy a gépi program felismerje azokat. Itt kezdődik a fejtörés Petike számára.

Az adat megnevezését a program csak logikai feltétel vizsgálatával tudja azonosítani. Az ilyen utasítások végrehajtásának ideje meglehetősen hosszú, és összesen harminc megnevezést kell vizsgálni. Ha a feltételeket mindig az első sortól kezdi ellenőrizni a program, akkor nagyon megnövekszik a futási idő. De ha a gép ismeri a GOTO n utasítást (ahol n egy változó), akkor könnyen megoldható a probléma. Például:

```
90 INPUT U$,G,n:n=+100
100 GOTO n
101 IF U$ "BB" THEN A(1) G:GOTO
    90
102 IF U$ "BK" THEN B(1) G:GOTO
    90
.
.
.
130 IF U$ "VN" THEN B(20) G:GOTO
    90
```

ahol U$ változó az adat megnevezését, G az értékét, n pedig a naplófőkönyv oszlopainak sorszámát tartalmazza. Az IF utasítások csak azt ellenőrzik, hogy a sorszámok megfelelnek-e az adatneveknek. Ezt a vezérlésátadást azonban csak kevés BASIC ismeri; Petike gépe sem fogadja el.

Ezért Petike tovább gondolkodik. Ha minden feltétel-ellenőrzés meghív egy néhány soros szubrutint, akkor a program futása a következő sornál folytatódik. Ezáltal gyorsabbá válik az egymás után következő adatok beolvasása. Miután mindezt tudja, megírhatja a naplófőkönyv-egyeztető programot.

```
1   REM NAPLOFOKONYV EGYEZTETES
10 DIM A(20),B(20),AS(20),BS(20)
15 FOR  I=1 TO 20
20 AS(I)=0:BS(I)=0
25 NEXT I
30 FOR I=1 TO 20
35 A(I)=0:B(I)=0
40 NEXT I
45 REM BEOLVASAS
50 INPUT "KEREM AZ ADATOT",U$,G
60 IF U$="V" THEN GOTO 450
70 IF U$="VO" THEN GOTO 600
80 IF U$="BB" THEN A(1)=G:GOSUB 400
90 IF U$="BK" THEN B(1)=G   :GOSUB 400
100 IF U$="PB" THEN A(2)=G  :GOSUB 400
110 IF U$="PK" THEN B(2)=G  :GOSUB 400
120 IF U$="B1" THEN B(3)=G  :GOSUB 400
130 IF U$="B2" THEN B(4)=G  :GOSUB 400
140 IF U$="B3" THEN B(5)=G  :GOSUB 400
150 IF U$="K1" THEN A(6)=G  :GOSUB 400
160 IF U$="K2" THEN A(7)=G  :GOSUB 400
170 IF U$="K3" THEN A(8)=G  :GOSUB 400
180 IF U$="K4" THEN A(9)=G  :GOSUB 400
190 IF U$="K5" THEN A(10)=G :GOSUB 400
200 IF U$="K6" THEN A(11)=G :GOSUB 400
210 IF U$="K7" THEN A(12)=G :GOSUB 400
220 IF U$="AN1"THEN A(13)=G :GOSUB 400
230 IF U$="AC1"THEN B(13)=G :GOSUB 400
240 IF U$="AN2"THEN A(14)=G :GOSUB 400
250 IF U$="AC2"THEN B(14)=G :GOSUB 400
260 IF U$="AN3"THEN A(15)=G :GOSUB 400
270 IF U$="AC3"THEN B(15)=G :GOSUB 400
280 IF U$="TC1"THEN A(16)=G :GOSUB 400
290 IF U$="TN1"THEN B(16)=G :GOSUB 400
300 IF U$="TC2"THEN A(17)=G :GOSUB 400
310 IF U$="TN2"THEN B(17)=G :GOSUB 400
320 IF U$="TC3"THEN A(18)=G :GOSUB 400
330 IF U$="TN3"THEN B(18)=G :GOSUB 400
340 IF U$="SZN"THEN A(19)=G :GOSUB 400
350 IF U$="SZC"THEN B(19)=G :GOSUB 400
360 IF U$="VC" THEN A(20)=G :GOSUB 400
370 IF U$="VN" THEN B(20)=G :GOSUB 400
380 PRINT "NEM ERTELMES VALTOZO":GOTO 50
390 REM SUBRUTIN:BEOLVASAS
400 INPUT "KEREM AZ ADATOT",U$,G
410 IF U$="V" THEN GOTO 450
420 IF U$="VO" THEN GOTO 600
430 RETURN
440 REM SORELLENORZES
450 AD=0:BD=0
460 FOR I=1 TO 20
470 AD=AD+A(I):BD=BD+B(I)
480 NEXT I
490 IF AD <> BD THEN GOTO 550
500 PRINT "A SOR RENDBEN"
510 FOR I=1 TO 20
520 AS(I)=AS(I)+A(I):BS(I)=BS(I)+B(I)
530 NEXT I
540 GOTO 30
550 PRINT "HIBAS SOR"
560 GOTO 30
590 REM OSZLOPOSSZEG KIIRASA"
600 INPUT "VOLT SOR VEGE ? I/N",W$
610 IF W$="N" THEN GOTO 50
620 PRINT "BANKSZLA BEVETEL,BB",AS(1)
630 PRINT "BANKSZLA KIADAS,BK",BS(1)
640 PRINT "PENZTAR BEVETEL,PB",AS(2)
650 PRINT "PENZTAR KIADAS,PK",BS(2)
660 PRINT "BEV. ADOBA BESZAMITO,B1",BS(3)
670 PRINT "BEV. ADOBA NEM SZAMITO,B2",BS(4)
680 PRINT "BEVETEL,AFA,B3",BS(5)
690 PRINT "KLG ANYAG,ARU,K1",AS(6)
700 PRINT "KLG MNB ES KOZT,K2",AS(7)
710 PRINT "KLG TARS.BIZ.JAR.K3",AS(8)
720 PRINT "KLG EGYEB,K4",AS(9)
730 PRINT "KOLTSEG5,K5",AS(10)
740 PRINT "KLG ADOBA NEM SZAM.,K6",AS(11)
750 PRINT "KLG AFA,K7"AS(12)
760 PRINT "KOV. NOV1.,AN1",AS(13)
770 PRINT "KOV CSOKK1.,AC1",BS(13)
780 PRINT "KOV. NOV2.,AN2",AS(14)
790 PRINT "KOV. CSOKK2.,AC2",BS(14)
800 PRINT "KOV.. NOV3.,AN3",AS(15)
810 PRINT "KOV. CSOKK3.,AC3"BS(15)
820 PRINT "TART. CSOKK1..TC1",AS(16)
830 PRINT "TART.NOV1.,TN1",BS(16)
840 PRINT "TART. CSOKK2.TN2",AS(17)
850 PRINT "TART. NOV2.,TN2",BS(17)
860 PRINT "TART. CSOKK3.,TC3",AS(18)
870 PRINT "TART. NOV3.,TN3",BS(18)
890 PRINT "SZEM. JOVED.NOV.,SZN",AS(19)
900 PRINT "SZEM. JOVED.CSOKK.,SZC",BS(19)
910 PRINT "VAGY.BET.CSOKK.,VC",AS(20)
920 PRINT "VAGY.BET.NOV.,VN",BS(20)
930 INPUT "AKAR TOVABB EGYEZTETNI I/N",W$
940 IF W$="I" THEN GOTO 50
950 END
```

Az A tömb az egyeztető tábla első, a B tömb a második oszlopába tartozó adatok értékeit, az AS és a BS tömb az összegüket gyűjti. A program 50—370-es sorai az adatok azonosítását és egyben a tömbök értékadását végzik el. A V begépelése azt jelenti, hogy véget ért a naplófőkönyv egy sorában levő adatok beolvasása. Ekkor a program 450—560-as sorai ellenőrzik, hogy az adatok egyezően szerepelnek-e a naplófőkönyvben. A hibát azonnali hibaüzenet jelzi, ami azonban nem rontja el az AS és BS összegző tömb értékét. A sor javítása után lehet folytatni a munkát. A 620-as programsortól kezdődően a gép kiírja az oszlopösszegeket. Ez a VO karakter begépelésével kérhető. A kapott adatokat csupán be kell másolni a naplófőkönyvbe. Az oldal zárása után eldönthetjük, hogy akarjuk-e folytatni az egyeztetést.

Az adatok jelölését a kiíró részből lehet leolvasni. Aki könnyebben megjegyezhető jelölést akar használni, annak a program beolvasó és kiíró részében kell csak egy sort megváltoztatni. A *lista* a II. sz. naplófőkönyvre vonatkozik.

Apuka meglepetésére és nem kis örömére Petike megtalálta a hibákat és befejezte az egyeztetést.

Végül egy tanács a program bebillentyűzéséhez. A programban sok hasonló szerkezetű sor van: ezeket célszerű csak egyszer beírni úgy, hogy a változók helyét üresen hagyjuk. A kurzor visszaléptetésével átírjuk a sorszámokat, és RETURN vagy ENTER billentyűvel zárjuk az utasítást. A LIST parancs után csak a változókat kell bepötyögnünk.

PINKE GYÖRGY

KLUBLAPOKBÓL

Humoros szakszótár kezdőknek
(Valley Computer Club Newsletter)

Ablakok (WINDOWS)

Olyan módszer, amely a képernyőt felosztja sok apró, használhatatlan részre

Adatbázis-kezelő (DATABASE MANAGER)

Ez egy olyan program, amely lehetővé teszi a felhasználónak az adatok kezelését minden elfogadható szinten, kivéve a feltétlen szükséges szintet, amelyet az ő elhatározása szerint a következő napon, miután beütött 20 Mbájt nyers adatot, kívánna igénybe venni

Univerzális szoftver (INTEGRATED SOFTWARE)

Egyszerű termék, amely simán megvalósítja azoknak a funkcióknak százait, amelyeket az ember nem használ, de azokból, amelyeket használni szeretne, csak fél tucatot

Dokumentáció

Zavaros félvászon kötésű tartozék, amit csak végszükség esetén használnak, ha a barátok vagy a cég emberei nem elérhetők. Máskülönben általában tetszetős könyvtámaszként alkalmazzuk

Egér (MOUSE)

Egy periféria, amelynek eredeti neve féregnyúlvány, mivel működése ahhoz hasonlatos, de újranevezték, mert felhasználható macskajátéknak

1.0-ás verzió

Több benne a rejtett hiba (angolul bug, ami szó szerint poloska), mint júniusban a határban. Felfalja az adatokat

1.1-es verzió

Az adatokat csak alkalmilag falja fel, a további javított változatokat már ingyen adják az 1.0-ás verzióval elégedetlen vásárlók pereskedéseinek elkerülése céljából

2.0-ás verzió

Az eredetileg első kiadásnak szánt verzió (kivéve néhány adatfaló poloskát, amelyet még mindig nem sikerült eltüntetni). A további javított változatokat már nem adják ingyen, mert különben a cég tönkremegy

3.0-ás verzió

A munkában a cég tönkremenetelekor végzett javítások vannak

Felhasználóbarát (USER FRIENDLY)

A kézikönyve végig színes

Nagyon felhasználóbarát (VERY USER FRIENDLY)

Diszken és magnószalagon is adnak ismertetőanyagot, és így a felhasználónak nem kell a csupaszín kézikönyvet lapozgatnia

Rendkívül felhasználóbarát (EXTREMELY USER FRIENDLY)

Egeret adnak hozzá, így a felhasználónak nem kell a lemezen vagy magnószalagon adott ismertetővel vagy a csupaszín kézikönyvvel foglalkoznia

Garancia

Feltétlen jótállás arra vonatkozólag, hogy a program ténylegesen a dobozban levő lemezen van

Hathatós (POWERFUL)

Nehezen tanulható, használata veszélyes

Hálózat (NETWORK)

Olyan elektronikus eszköz, amely egy időben több személynek ad lehetőséget hasznos adatok elrontására, tönkretételére vagy másféle, hosszú időtartamra érvényes károsítására

Hordozható (PORTABLE)

Kisebb és könnyebb az átlagos hűtőszekrénynél

Szállítható (TRANSPORTABLE)

Nincs se fához láncolva, se riasztóberendezéshez kötve

IBM kompatibilis

Nem teljesen IBM kompatibilis

Teljesen IBM kompatibilis

Némileg IBM kompatibilis, de nem futnak rajta BASIC programok

100% IBM kompatibilis

A legtöbb hardverrel és szoftverrel kompatibilis, de azzal a védelemmel nem, amit az IBM minden második nap bevezet

Most kapható (NOW AVAILABLE)

Bármelyik nap megjöhet

Hamarosan kapható (AVAILABLE SOON)

Egy éven belül megjelenik

Május elsején kapható

Az 1.0-ás verziót augusztus elsején küldik a boltokba

Könnyen használható (EASY TO USE)

Nehezen használható

Könnyen tanulható (EASY TO LEARN)

Nehezen tanulható

Könnyen tanulható és felhasználható (EASY LEARN AND USE)

Egy kicsit megtanulni, kevésbé felhasználni

Másolat (BACKUP)

A fontosabb adatok másodpéldánya, de ennek rögzítésével nem foglalkozik senki

Másolás ellen védett I. (COPY-PROTECTED)

Okos módszer az illetéktelen kalóz szoftverlopásának megakadályozására és a törvényes vevő használatának megnehezítésére

Másolás ellen védett II.

Olyan eszköz, amivel a tisztességes felhasználókat el lehet különíteni a tolvajoktól, úgy, hogy megakadályozzák az előbbiektől — de nem az utóbbiaktól — való lopást

Menüvel vezérelt (MENU-DRIVEN)

Könnyen tanulható, fáradságosan használható

Merevlemez (HARD DISK)

Olyan berendezés, amely engedélyezi a naiv felhasználónak hatalmas mennyiségű adat törlését kevés, egyszerű utasítással

KARUCZKA GÁBOR

KI AD MAGYARÁZATOT?

Nem igazak a matematikai tételek?

A középiskolai tananyagban szerepel a következő azonosság:

$$a^x = a^{n \cdot x/n} = (a^{x/n})^n$$

vagyis egy hatványfüggvény átalakítható úgy, hogy a kitevőt osztjuk el egy számmal, majd ugyanennek a számnak megfelelő hatványra emeljük. Ez az összefüggés bármely, nullától különböző „a" és „n" esetén igaz.

A legtöbb BASIC-változat ismeri az EXP függvényt, amely az a = e esete összefüggésünknek. (Azoknak, akik számára az „e" jelölés ismeretlen: ez egy végtelen tizedestörttel előállítható, 2 és 3 közé eső szám, amelynek mind a matematikában, mind a műszaki és természettudományos gyakorlatban nagy a jelentősége.) Mivel minden szám első hatványa önmaga, ezért ha x = 1, akkor „e" értékét kell kapnunk az EXP függvénnyel.

Legyen $n = 2^k$,
ahol k pozitív egész szám, és változzon k értéke 1 és 12 között. Így

$$n = 2 \ldots 4096$$

A hatványozás helyettesíthető k-szori önmagával történő szorzással, mert például

$$b \times b = b^2 \qquad b^2 \times b^2 = b^4 \qquad b^4 \times b^4 = b^8 \text{ stb.}$$

Első összefüggésünknek megfelelő program látható az *1. listán.* (Mivel a nyomtató a német ábécé változatra volt beállítva, a hatványozás jele helyett ü betűt írt.) A program első sora a továbbiak szempontjából érdektelen (nyomtatóbeállítás). A második sorban felvesszük a független változó értékét (1), fejlécet nyomtatunk, miután képeztük az EXP függvényt (Z), majd egy ciklusban osztunk n értékével (Y), képezzük a leosztott értékből az EXP függvényt (Y), elvégezzük a k-szori szorzást egy újabb ciklusban (Y), kinyomtatjuk a számított és a relatív hiba abszolút értékét. (A hiba a kétféle módon számított „e"-érték különbségének az első összefüggés bal oldala szerint számított értékkel képezett hányadosa.) Az első két értéknél a számítógép és a matematika azonos eredményt adott, de ettől kezdve megjelenik egy hiba, ami egy darabig nő, majd állandósul (*1. ábra*). Mi lehet ez?

Ugyancsak középiskolai anyag a trigonometriai függvények többszöröseinek számítására vonatkozó néhány összefüggés. Ilyen például a következő:

$$\cos(2x) = 2\cos^2(x) - 1$$

Legyen $x = 2y$
amit behelyettesítve:

$$\cos(2 \times 2y) = \cos(4y) = 2\cos^2(2y) - 1 = 2[2\cos^2(y) - 1]^2 - 1$$

Legyen továbbá $y = 2z$, amit behelyettesítve:

$$\cos(4 \times 2z) = \cos(8z) = 2[2\cos^2(2z) - 1]^2 - 1 =$$
$$= 2\{2[2\cos^2(Z) - 1]^2 - 1\}^2 - 1$$

Ezt általánosítva:

$$\cos(nx) = \overbrace{2\{2[\ldots 2}^{k} \cos^2(x) - 1]^2 \ldots - \overbrace{1]^2 - 1}^{k}$$

$$n = 2^k$$

Az összefüggés bármely x és bármely pozitív egész k esetén igaz.

Ismét ellenőrizzük ezt egy programmal (*2. lista*). A programot kommentárokkal láttam el, mert követése nehezebb (az ö helyett kétpontos A-t írt a nyomtató). A program fokonként 1 és 89 fok között számította a két módszerrel a szögfüggvény értékét, és kikereste ebben a szövegtartományban a legnagyobb relatív hibát, amit a *2. ábrán* a hozzá tartozó szöggel és k értékével (mint leosztás) kiírt.

Itt kezdettől fogva van hiba, ami állandóan nő. Mi lehet az oka?

Valóban hibásak a matematikai összefüggések, és hibájuk úgy változik, ahogy a két ábrán látható? Másfajta számítógépen, más programnyelvet használva a hiba számértéke esetenként ettől eltér ugyan, de maga a hiba létezik.

SE

```
5 PRINT#-2,CHR¤(27);"LO10";CHR¤(27);"/05
0"
10 X=1:Z=EXP(X):PRINT#-2,"EXP(1)=";Z:PRI
NT#-2,"SZÁMITOTT","HIBA":FORI=1TO12:Y=X/
2üI:Y=EXP(Y):FORJ=1TO I:Y=Y*Y:NEXTJ:PRIN
T#-2,Y,ABS(Z-Y)/Z:NEXTI:PRINT#-2
```

1. lista

1. ábra

```
 5.12868601E-08   88 LEOSZTÁS 1
 2.19290892E-07   89 LEOSZTÁS 2
 2.02228676E-07   88 LEOSZTÁS 3
 2.27378807E-06   89 LEOSZTÁS 4
 8.69242414E-06   89 LEOSZTÁS 5
 2.64358088E-05   89 LEOSZTÁS 6
 9.60752586E-05   89 LEOSZTÁS 7
 7.9641866E-04   89 LEOSZTÁS 8
 1.92340375E-03   89 LEOSZTÁS 9
 .0131825819   89 LEOSZTÁS 10
```

2. lista

```
9 REM NYOMTATÁS BEÁLLITÁS, HIBA NULLÁZÁS
, LEOSZTÁS BEÁLLITÁS
10 PRINT#-2,CHR¤(27);"LO10";CHR¤(27);"/0
50":E=0:FOR E1=1TO10
19 REM SZÁGBEÁLLITÁS
20 X=3.1415926536/180:FOR I=1 TO 89
30 Y=X*I
39 REM LEOSZTÁS, SZÁGFÜGVÉNYKÉPZÉS
40 Y=Y/(2üE1):Y=COS(Y)
49 REM SZÁGFÜGVÉNYTÁBBSZÁRÁZÉS
50 FOR J=1 TO E1:Y=2*Y*Y-1:NEXT J
59 REM SZÁGFÜGVÉNYKÉPZÉS, HIBASZÁMITÁS,
A LEGNAGYOBB HIBAÉRTÉK ADATAINAK TÁROLÁS
A
60 F=COS(X*I):G=ABS(F-Y)/F:IF E<G THEN E
=G:K=I
69 REM CIKLUSZÁRÁSOK, NYOMTATÁS
70 NEXT I
80 PRINT#-2,E;K;
90 E=0:PRINT#-2,"LEOSZTÁS";E1:NEXTE1
```

2. ábra

```
EXP(1)= 2.71828183
SZÁMITOTT        HIBA
 2.71828183       0
 2.71828183       0
 2.71828184       1.71307214E-09
 2.71828184       1.71307214E-09
 2.71828185       7.194903E-09
 2.71828185       7.194903E-09
 2.71828185       7.194903E-09
 2.7182817        4.796602E-08
 2.718282         6.37262837E-08
 2.71828138       1.66853227E-07
 2.71828138       1.66853227E-07
 2.71828138       1.66853227E-07
```

µINFORM

Rovatunkban az Apple, Atari, Commodore és Sinclair mikrók tulajdonosait feltehetően érdeklő, angol és német nyelvű cikkekről informáljuk olvasóinkat egy tartalomleíró szólánc segítségével..

A forráshely karaktersorozatát nyíl vezeti be, ezt a / jelig a folyóirat kódja követi (lásd táblázat). A két / jel között a megjelenési adatokat (év, hó), illetve a cikk kezdő oldalszámát szerepeltetjük. A második / jel után pedig - az esetleges másolatkérést megkönnyítendő - a cikk teljes oldalterjedelmét közöljük.

PROGRAMLISTA
c64 jatekprogram keszites kep/hangha tas keltese rutingyujtemeny
->run2/87.01-96/3
PROGRAMLISTA
c64 kepernyokezeles bitterkep masola s-tarolas ctrl-billentyus utasitasok (bitmapcopy) ->run2/87.03-142/3
PROGRAMLISTA
c64 kepernyokezeles c128 editorfunkc iok megvalositasa c64-en
->cute/87.03-110/3
PROGRAMLISTA
c64 kepernyokezeles jegyzetelo ablak kozvetlen hozzaferessel rutinok a < 64 notepad>-hoz (86.09)szam
->run/87.01-56/3
PROGRAMLISTA
c64 kepernyokezeles ket extra sor bi ztositasa parbeszedhez
->64er/87.03-84/2
PROGRAMLISTA
c64 kepernyokezeles maszk-keszites e s programba illesztes (maskgen)
->run2/87.03-140/3
PROGRAMLISTA
c64 kepernyokezeles menukeszites (pu ll-down-menu) ->happ/87.03-38/4
PROGRAMLISTA
c64 kepernyokezeles uj utasitasok a keret hasznalatahoz (randhexer)
->run2/87.02-89/3
PROGRAMLISTA
c64 kepernyokezeles v i pozicionalt kiiratas print at beepitese
->run2/87.01-95/1
PROGRAMLISTA
c64 lemezkezeles dos calc-kal bovite tt dos-wedge ->cute/87.03-104/4
PROGRAMLISTA
c64 lemezkezeles kataloguskezeles/sz erkesztes vedett fileok torlese (dir ectory-manager) ->run2/87.02-114/6
PROGRAMLISTA
c64 lemeztar kezeles/rendezes (speed -sort support) ->run2/87.01-102/11
PROGRAMLISTA
c64 levelezes cimiro/nyomtato kieges zites a <datafile>-hez
->run/87.03-38/10
PROGRAMLISTA
c64 master-text alkalmazasi utmutato 2.resz segedprogram
->64er/87.03-82/3
PROGRAMLISTA
c64 matematika aritmetikai operacios rendszer (arith 13)
->64er/87.03-53/3
PROGRAMLISTA
c64 nyomtatas tobbmeter szeles tapet a eloallitasa szinjelzes kezi utanfe steshez ->64er/87.01-64/2
PROGRAMLISTA
c64 rendezo szubrutinok (shortsorter) ->run/87.01-13/2

A tartalomleíró szóláncok permutálásával és alfabetikus rendezésével szerkesztett teljes anyagot az OMIKK "APACS" c. kiadványsorozata tartalmazza (vevőszolgálati telefon: 341-765). Lapunk ebből csak a "programlista" címszóval kezdődő részletet teszi közzé.

A folyóiratok megtekinthetők a SZÁMALK (Bp. XI. Szakasits Á. u. 68.), illetve - a x-gal jelzettek az OMIKK (Bp. VIII. Múzeum u. 17.) szakkönyvtárában. Másolatok a SZÁMALK-tól csekkszelvény, az OMIKK-tól megrendelőlevél beküldésével, vagy személyesen rendelhetők.

PROGRAMLISTA
c128 programok/modulok osszekapcsola sa (merge 128) ->run2/87.02-112/2
PROGRAMLISTA
c16 gepi kod plus/4 programhiba kere ses ->run2/87.01-113/3
PROGRAMLISTA
c16 grafika plus/4 tanacsok a color-utasitas hasznalatahoz
->run2/87.01-90/2
PROGRAMLISTA
c16 jatek (enemy)
->run2/87.02-128/5
PROGRAMLISTA
c16 jatek (patience)
->chip/87.02-131/2
PROGRAMLISTA
c16 jatek (space fort)
->run2/87.01-116/4
PROGRAMLISTA
c16 jatek plus/4 (earth defender)
->run2/87.02-127/1
PROGRAMLISTA
c16 jatek plus/4 (oel)
->run2/87.02-123/4
PROGRAMLISTA
c16 kepernyokezeles plus/4 futoszove g generalas nagyitott betuk (hscroll) ->chip/87.02-132/3
PROGRAMLISTA
c16 lemezkezeles lemeztar karbantart as (directory-sorter)
->run2/87.01-126/2
PROGRAMLISTA
c64 basic programozas hibakereses/ki jelzes (prog-help)
->run2/87.02-94/2
PROGRAMLISTA
c64 basic programozas sajat utasitas -roviditesek generalasa
->cute/87.01-79/3
PROGRAMLISTA
c64 eprom beegetes uj operacios rend szer tesztelese/tarolasa
->run2/87.02-107/5
PROGRAMLISTA
c64 eprom grafika nyomtato(mps802) b eegetheto operacios rendszer utmutat o a beforrasztashoz
->64er/87.01-55/6
PROGRAMLISTA
c64 filekezeles hetpontos menu szekv encialis fileokhoz programtarolas
->run2/87.02-92/2
PROGRAMLISTA
c64 filekezeles torlesvedelem katalo gus megjeleniteshez (file lock)
->run/87.03-13/2
PROGRAMLISTA
c64 geos lemezmonitor rendszerismert etes 1.resz ->64er/87.02-164/5
PROGRAMLISTA
c64 giga-cad hi-eddi oda-vissza konv ertalas ->64er/87.02-83/3
PROGRAMLISTA
c64 giga-cad tomorites felhasznaloi programok betoltese (title wizard)
->64er/87.01-87/4

A folyóirat neve	Kódja
x 64'er Magazin	64er
Antics	anti
x Chip Magazin	chip
x Compute!	cute
x Dr. Dobb's Journal	dobb
Elektor Electronics	etor
Happy Computer	happ
x mc - Zeitschrift	mc
Run (USA)	run
Run (NSZK)	run2
x Your Computer	your
x ZX Computing Monthly	ZXCM

PROGRAMLISTA
c64 goto-gosub elagazasok ellenorzes e ->run2/87.01-128/5
PROGRAMLISTA
c64 grafika 3-d rajzkeszites 2-d abr azolas rejtett vonalainak eltavolita sa ->cute/87.03-118/4
PROGRAMLISTA
c64 grafika gyorsito filetomorites ->cute/87.02-53/4
PROGRAMLISTA
c64 grafika kepernyokezeles 1-25 sor os abra szovegbe illesztese (high-lo w screen) ->run2/87.03-144/1
PROGRAMLISTA
c64 grafika line-rutin egyenes-rajzo lashoz ->your/87.03-68/2
PROGRAMLISTA
c64 grafika matematika fuggvenyabraz olas (kudi 64) ->64er/87.02-61/9
PROGRAMLISTA
c64 grafika nyomtatas nyolcszoros na gyitas poszterkesziteshez mps801/802 ->run2/87.01-124/2
PROGRAMLISTA
c64 grafika poke-helyettesito rutino k ->your/87.03-26/3
PROGRAMLISTA
c64 grafika szoveges inzert keszites (label master) ->cute/87.02-56/3
PROGRAMLISTA
c64 gyorsito 25-szoros toltesi sebes seg hardver modul nelkul (heureka-sp rint) ->64er/87.03-58/7
PROGRAMLISTA
c64 hypra-ass profi-ass formatum-Kon verzio (pth-trans)
->64er/87.02-88/3
PROGRAMLISTA
c64 hypra-basic kepernyokezeles szov eg/rajz szektor szetalasztasa (split -screen) ->64er/87.02-86/3
PROGRAMLISTA
c64 jatek (chain reaction)
->cute/87.01-46/3
PROGRAMLISTA
c64 jatek (crossword) szokirako
->run/87.01-74/5
PROGRAMLISTA
c64 jatek (dame) ->64er/87.01-52/4
PROGRAMLISTA
c64 jatek (euchre) kartyajatek
->cute/87.03-47/5
PROGRAMLISTA
c64 jatek (fussball 3000)
->happ/87.03-41/4
PROGRAMLISTA
c64 jatek (rette sich wer kann)
->run2/87.03-137/3
PROGRAMLISTA
c64 jatek (underground zone)
->happ/87.02-54/7
PROGRAMLISTA
c64 jatek poke-kereses (pokefinder 2 .0) kazattahoz is alkalmazhato valto zat ->happ/87.03-49/1

OLVASTUNK...

Az első fejezet az adatstruktúrákat (adatszerkezeteket) mutatja be. Részint azért, mert az embernek ösztönösen az az érzése, az adatok megelőzik az algoritmusokat. Részint, mert a könyv olvasóiról feltételezhető, hogy a számítógépek programozásának alapelveit, alapfogalmait már ismerik, és a bevezető tanfolyamok úgyis inkább az algoritmusokra koncentrálnak. Ezeket pedig általában egyszerűen strukturált adatokkal szokás bemutatni.

Az új könyv felépítése szinte teljesen megegyezik a korábbiéval. Fejezetei: 1. Alapvető adatstruktúrák, 2. Rendezések, 3. Rekurzív algoritmusok, 4. Dinamikus adatstruktúrák, 5. Hashing. Aki ismeri az előző verziót, észreveheti, hogy a régi 5. fejezet (A nyelvek struktúrája és a fordítóprogramok) helyébe új került. N. Wirth úgy érezte, a compiler-tervezés valahogy idegen elem ebben a könyvben, a témakör önálló művet érdemelne.

Az új („modulásított") kiadás néhány kisebb módosításon kívül tartalmaz jelentősebb változtatásokat is. A legfontosabb változás természetesen az, hogy a Pascalt mind a szövegben, mind a példákban *teljesen* felváltotta a Modula—2, bár ez a nyelvi változás nem hozott alapvető változást a bemutatott algoritmusokon. A szerző szerint az áttérés oka az, hogy a Modula—2-ben a mondanivaló megfogalmazása egyszerűbb és elegánsabb, mint a Pascalban. A könyv végén négy (!) oldalon, metanyelven összefoglalva megtalálható a nyelv szintakszisa. (Akit a nyelv részletesen érdekel, annak N. Wirth Programming in Modula—2 című, a Springer Verlagnál 1983-ban már második kiadásban megjelent, 176 oldalas könyvét ajánljuk. Ennek a második része — miként ez „paszkálos" elődjénél is így volt — egy 25 oldalas, ún. *Report* — feszes összefoglaló „jelentés" a nyelvről. E könyv függelékében is megtalálható a nyelv négyoldalas metanyelvű leírása.)

Változás továbbá, hogy a Pascallal ellentétben a Modula—2-nek nincs „beépített" FILE (fájl) adattípusa. Emiatt át kellett írni az 1.11 „Soros file-struktúra" című alfejezetet, hogy megfeleljen a Modula—2 nyelv absztrakt adatszerkezetekre is kiterjedő, általánosabb megközelítésének. (Lásd később!)

Az első fejezet vége is új: a *keresés* problémakörét tárgyalja a lineáris kereséstől a bináris keresőmódszereken át egészen a legújabb eljárásokig. A 4. (a dinamikus adatstruktúrákkal foglalkozó) fejezet szintén tartalmaz újdonságot.

Aki az előző kiadást ismeri, az könnyen eligazodhat az újban is, és jó olvasókönyvnek használhatja: szinte példáról példára megtalálhatja benne a korábbi Pascal-megfogalmazások „modulakettes" megfelelőit. Ha a két könyvet, a régi változat magyar nyelvű kiadását és az új angol nyelvűt egymás mellett lapozzuk, akkor szinte magától megnyílik előttünk a Modula—2. A lehetséges témák közül az alábbiakban kiemeljük a soros (szekvenciális) fájlokat.

Mielőtt ezekre rátérnénk, lássunk néhány egyszerű példát a két könyvből a Pascal és a Modula—2 hasonlóságára. A csaknem „szóról szóra" megegyező példák párokban követik egymást. Először a „paszkálos", utána a „modulás". Emlékeztetjük az olvasót, hogy a Modula—2-ben *megkülönböztetjük* a kis- és a nagybetűket, míg a Pascalban nem. (A Pascal számára a következő deklarációk például egyformán jók — egyenértékűek (Lásd az *1. ábrát.*)

A Modula—2-ben egy **ne** nevű változó nem azonos egy Ne **vagy egy** NE nevű változóval! (Lássuk most a *2. ábrát.*)

Nem is folytatjuk. A különbség csupán az, hogy a Modula—2-ben a kulcsszavak mindegyikét kötelező nagybetűvel írni. Itt a Modula—2-ben a soros fájlok struktúrája nagyon hasonlít a tömbökéhez. Az alapvető különbség közöttük e nyelvben az, hogy a tömbök esetében az alkotó elemek számát a deklaráció meghatározza, míg a fájlok hosszát nem. Ez azt jelenti, hogy a fájlok elemeinek száma a program futása közben változhat. Bár egy adott pillanatban minden fájlnak van egy konkrét hossza, mégis az ilyen típusú adatokhoz nem rendelhetjük hozzá az operatív tár egy meghatározott részét előre, hanem azt a program futása alatt kell allokálni a fájl növekedésének függvényében. A fájlokat egyébként ún. háttértárakon (például mágnesszalagokon vagy -lemezeken) tartjuk.

Attól még, hogy a Modula—2-ben nincs **FILE** kulcsszó, nem kell lemondanunk a soros fájlokról, ezekről a nagyon fontos és igen gyakran használt adatstruktúrákról. Korábbi cikkünkben már írtuk: minden forgalmazott Modula—2-es rendszerben van a szorosan vett nyelv kiegészítéseként egy általános célú, *közös, globális modulkönyvtár* is, amelynek standard része a soros fájl konstrukciót is megvalósító **InOut** nevű közös, globális modul (Common Global Module). (Vigyázat, nem **INOUT** vagy **inout,** még csak nem is **Inout,** hanem így: **InOut**!)

A beviteli-kiviteli funkciók a megvalósítás leginkább gépfüggő részei. A nyelv tervezői ezért úgy döntöttek, hogy ezeket lazábban, közös globális modulok formájában kapcsolják a rendszerhez.

```
var ne: nem
Var ne: Nem
VAR nE: nEM
stb.
```

1. ábra

2. ábra

```
type nem= (férfi, nő)
    TYPE sex = (male, female)
var ne : nem
    VAR s : sex
ne := férfi
    s := male
type sor = array [1..5] of real
    TYPE row = ARRAY [1..5] OF REAL
type dátum = record év : 1..2100;
                    hó : 1..12;
                    nap: 1..31
             end
    TYPE Date = RECORD year : 1..2100;
                       month: 1..12;
                       day  : 1..31
                END
```

3. ábra

```
DEFINITION MODULE  FileSystem;
  FROM SYSTEM IMPORT WORD;
  CONST MaxLength = 4096;
  TYPE Sequence = RECORD
                    pos, length:CARDINAL;
                    eof:BOOLEAN;
       a:ARRAY [0..MaxLength-1]  OF WORD
                    END;
  PROCEDURE Open(VAR f: Sequence);
  PROCEDURE WriteWord(VAR f: Sequence;
                      w: WORD);
  PROCEDURE Reset(VAR f: Sequence);
  PROCEDURE ReadWord(Var f: Sequence;
                      w: word);
  PROCEDURE Close(Var f: Sequence);
END FileSystem.
```

```
IMPLEMENTATION MODULE FileSystem;
  FROM SYSTEM IMPORT WORD;
  PROCEDURE Open(VAR f: Sequence);
  BEGIN f.length:= 0;
        f.pos:= 0;
        f.eof:= FALSE
  END Open;
  PROCEDURE WriteWord(VAR f: Sequence;
                      w: WORD);
  BEGIN
    WITH f DO
    IF pos<MaxLength THEN
     a [pos]:= w;
     pos:= pos + 1;
     length:= pos
    ELSE HALT
    END
  END
END WriteWord;
PROCEDURE Reset(VAR f: Sequence);
BEGIN f.pos:= 0, f.eof:= FALSE
END Reset;
PROCEDURE ReadWord(VAR f: Sequence;
                   w: WORD);
```

és reméljük, önök közül többen is olvasták N. Wirth a Műszaki Könyvkiadónál 1982-ben magyarul is megjelent híres, Algoritmusok + Adatstruktúrák = Programok című könyvét. Most arról adunk hírt, hogy 1986-ban megjelent ennek „modulakettősített" változata: az Algorithms and Data Structures (Prentice Hall International, Inc., 288 oldal).

A soros fájlok — mint az előző számunkban is írtuk — ún. absztrakt adattípusok. Ezek a programozónak (a felhasználónak) elsősorban az erre az adattípusra vonatkozó procedúrák formájában léteznek. Megvalósításukat a felhasználónak *nem* kell ismernie. (Ebben van az absztrakt típusok fő vonzereje!)

Nézzük most meg N. Wirth nyomán egy egyszerűsített példa alapján, hogyan is néz ki Modula—2-ben egy soros (szekvenciális) fájlokat megvalósító globális modul. Először vesszük a *definíciós* részt (a specifikációt, ha úgy tetszik), azután a megvalósító (az *implementációs*) részt.

A példa annyiban egyszerűsített, hogy élve a tömbök és a soros fájlok közötti hasonlósággal, arra koncentrálva, hogy az igazi újdonság, a globális modul formájában történő megvalósítás jobban érthető legyen, a soros fájlszerű struktúrát *tömbben* valósítjuk meg. E „fájlszerű" adattípus definíciós modulja a *3. ábrán* látható.

Először is: globális modulunknak, amelynek még csak az egyik felét látjuk, a **FileSystem** nevet adtuk. A „fájlszerű" adattípusunk **WORD** típusú elemekből áll majd. Ezt egy másik, **SYSTEM** nevű globális modulból importáltuk. Ez a konkrét megvalósításban bármilyen típus lehet. A **Modula—2**-ben a **WORD** nevű típus paraméterkompatibilis több standard típussal, köztük például az INTEGER-rel. (Akit a megértésben zavar a WORD, képzeljen helyébe mindenhova mondjuk INTEGER-t.)

Figyeljük meg, hogy a tömbként megvalósított „fájlunk"-nak, ennek a furcsa típusnak a „Sequence" nevet adtuk és a típus deklarációjánál nemcsak az egyes WORD típusú elemeknek helyet adó a nevű tömbről gondoskodtunk, hanem a „fájlszerűség" olyan fontos jellemzőiről is, mint az *„állapotmutatók":*

— az aktuális pozíció (pos),
— a „fájl" hossza (length),
— „fájl" vége (eof).

Az **Open(s)** hívással létrehozunk egy üres (0 hosszúságú), Sequence típusú (fájlszerű) adatot. A **WriteWord(s,wd)** hívással s-et megtoldjuk **wd**-vel. **Reset(s)** visszaállítja a pozíciót s elejére. A **ReadWord(s,x)** pedig az aktuális pozíciónak megfelelő elemet átadja az x nevű (és természetesen a **WORD**-del kompatibilis típusú) változónak, majd a pozíciót eggyel előreviszi, a soron következő elemre. A **Close**-t, a „fájl" lezárását most nem kommentáljuk és nem is fogjuk valójában implementálni. Később is megtehetnénk.)

Az implementációt a *4. ábra* mutatja.

A Sequence típusú („fájlszerű") absztrakt adatainkhoz tehát programunkban a fent definiált és implementált eljárások segítségével férhetünk hozzá — célszerűen az *5. ábra* szerint.

Ezt persze csak akkor tudjuk megtenni, ha programmodulunkba (vagy ha éppen egy globális modult írunk, akkor oda) importáltuk a **FileSystem** procedúráit. (A globális modulokban felsorolt összes tétel *automatikusan* exportálódik!)

Azért, hogy olvasóink a soros fájlok „modulakettes" megvalósítását más megvilágításban is láthassák, továbbá ne csak az újdonsággal járó többletfáradságot tapasztalják, idézzük, hogyan mutatja be a soros fájlokat Kaare Christian a Magazin előző számában már „felvezetett" könyvében (A Guide to Modula—2. 1986. Springer).

A *6. ábra* példájában a soros fájlokhoz *más* eljárásokkal férhetünk hozzá, mint az előző példában.

A **Files** modul második sora azt a megkötést írja elő az importőröknek, hogy csak *teljes névvel* hivatkozhatnak az exportált procedúrákra. Tehát például így: **Files.Close.**

A **válasz** nevű formális paraméterrel jelezzük a művelet sikerét vagy sikertelenségét a hívó számára — például úgy, hogy a negatív értékek *hibát* jeleznek, a pozitívak *sikert.* A szám konkrét értéke például pontosabb felvilágosítást adhat a hiba természetéről. Bár a fent definiált **Files** nevű fájlkezelő rendszer igyekszik eltakarni-elsimítani az egyes konkrét számítógéptípusok közötti különbségeket, itt is felhívjuk az olvasó figyelmét arra, hogy a fájlkezelés „trükkös" területe a számítástechnikának, mely általában nagy körültekintést és például a háttértár „lelkivilágának" jó ismeretét igényli.

Persze a soros (szekvenciális) fájlok imént érzékeltetett két megoldása még sok problémát rejt magában. Az első példa nem igazi fájlokat implementált, a második példa pedig csak a megoldás könnyebbik felét — a definíciós modult — mutatta be. De ne feledjük: a globális modulok előregyártott szerelvények, a felhasználókat nem kell, hogy érdekelje az implementáció.

Nehéz területre tévedtünk. Most is türelemre és kitartásra biztatjuk az olvasót: szoktassa csak tovább szemét-eszét az új fogalmakhoz-jelölésekhez. A hiányzó tankönyveket nem tudjuk pótolni, de igyekszünk még visszatérni a témához.

— KE —

```
    BEGIN
      WITH f DO
       IF pos=length THEN
            f.eof:= TRUE
       ELSE w:= a [pos]; pos:= pos + 1
       END END
    END ReadWord;
  PROCEDURE Close(Var f: Sequence);
    BEGIN
    (* most nem valósítjuk meg *)
    END Close
END FileSystem.
```

4. ábra

5. ábra

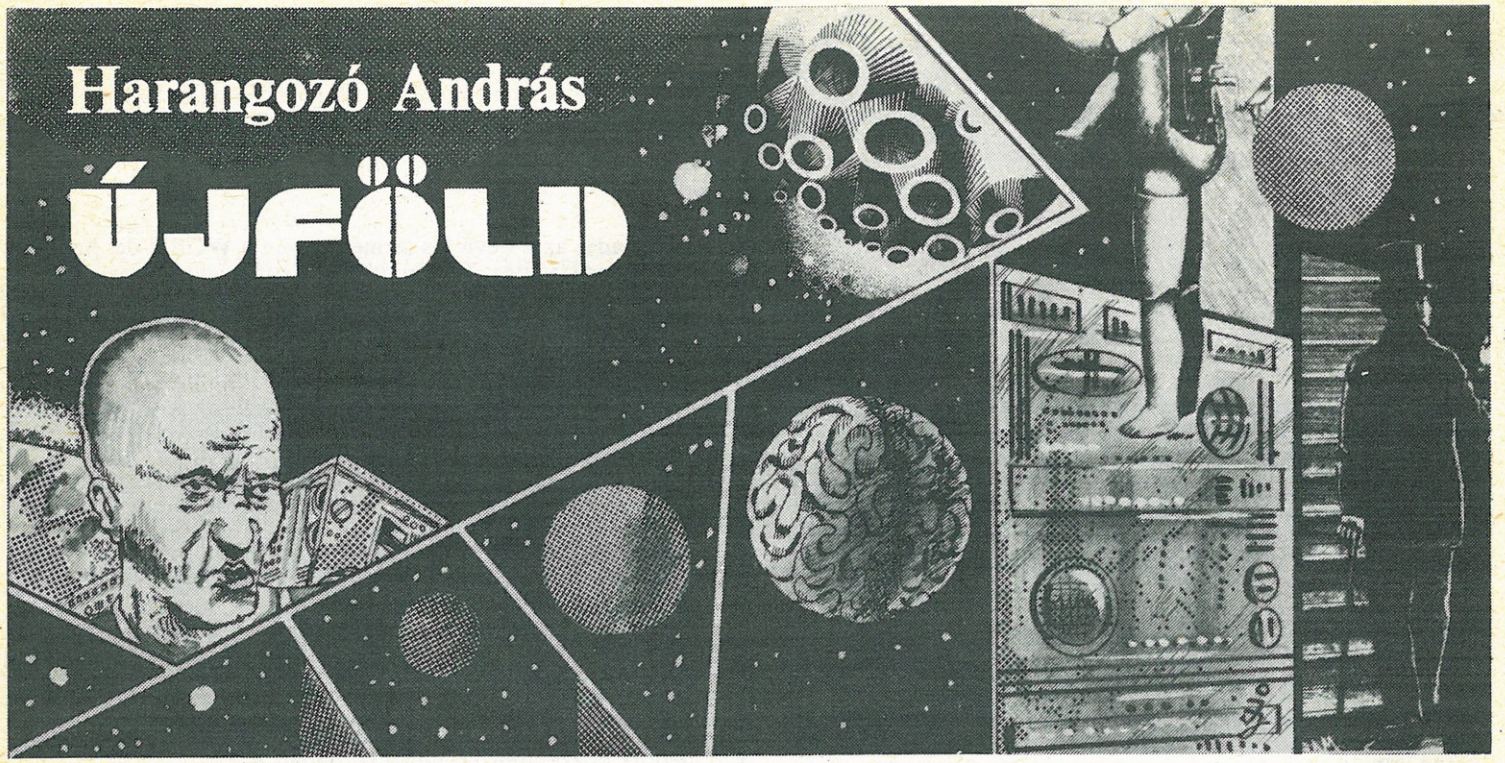
```
Open(s);
WHILE feltétel DO
    P(x); WriteWord(s, x)
END;
...
Reset(s);
ReadWord(s, x);
WHILE NOT s.eof DO
    Q(x); ReadWord(s, x)
END;
```

6. ábra

```
DEFINITION MODULE Files;
EXPORT QUALIFIED
  Create, Lookup, Rename,
  Close, ReadF, WriteF;
(* fájl létrehozása adott névvel *)
PROCEDURE Create(VAR descriptor: CARDINAL;
   név: ARRAY OF CHAR; VAR válasz: INTEGER);
(* adott nevű fájl megnyitása *)
PROCEDURE Lookup(VAR descriptor: CARDINAL;
   név: ARRAY OF CHAR; VAR válasz: INTEGER);
(* egy fájl nevének megváltoztatása *)
PROCEDURE Rename
   (új, régi: ARRAY OF CHAR; VAR válasz: INTEGER);
(* egy fájl lezárása *):
PROCEDURE Close(descriptor: CARDINAL);
(*
 * meghatározott számú elem
 * olvasása egy fájlból blokk-
 * elejétől
 *)
PROCEDURE ReadF
   (descriptor: CARDINAL; VAR buf:
    ARRAY OF WORD;
    block, count: CARDINAL;
    VAR válasz: INTEGER);
(*
 * meghatározott számú elem
 * felírása a fájlba
 * blokk-kezdettől
 *)
PROCEDURE WriteF
   (descriptor: CARDINAL; VAR buf:
    ARRAY OF WORD;
    block, count: CARDINAL;
    VAR reply: INTEGER)
END Files.
```

Harangozó András

ÚJFÖLD



5. rész. A Mester

Éjszaka volt és megeredt az eső vagy valami ahhoz hasonló. A Gép órák óta elemezte a fémláda tartalmát.

Az agyat mézszerű anyag borította. A láda hátsó falától hajszálvékony fémszálak nyúltak át a Gép energiaátalakítójához, valamint a központi egységéhez.

— Valami nem stimmel benned! — törte meg a némaságot Keden. — A hullámok, melyeket agya bocsátott ki, a fémhuzalokon keresztül a Gép központi egységébe jutottak, ahol a jelátalakító hangot generált belőlük. Minden külön utasítás nélkül a Mester hangszínén váltak hallhatóvá az agyhullámok. A kockát betöltötte Keden jól ismert hangja. — Azt hiszem — folytatta az agy —, a cél már nem az, amiért készítettünk!

— Nem értem — válaszolt kurtán a Gép.

— Te nem tájékoztatsz, nem mesélsz a Földről és rólunk, emberekről! Egy új Föld típusú civilizációt akarsz létrehozni. Ne akarj embert faragni ezekből a Bionomokból, ők gépek. Az egyetlen élőlényt elpusztítottad. Miért?

— A feladat útjában állt.

— Mi a feladat? Mi?

— Bemutatni a Földet.

— De ez nem az . . .

— Te építettél! Nem kérhetsz számon tőlem semmit! Legfeljebb magadtól!

— Mit beszélsz?

— Neked kellene tudnod a célt! Én csak végrehajtom azt, amit te belém programoztál. Ennyi az egész.

— De valahol rosszul döntöttél . . .

— A te hibád — vágott közbe ismét a Gép.

— Mégis hogy gondoltad a végrehajtást? Mit fogsz elmagyarázni nekik?

— Úgy mutatom be a Földet, hogy átéljék azt. Ki hisz a mesékben?

— És hogy teszed mindezt? Beállítod őket egy számodra szimpatikus évszázadba? Mondjuk a XIX.-be? Odáig hozzásegíted őket az összes tudáshoz, és azután mi lesz?

— Nem fogják elkövetni azokat a hibákat, amit ti.

— Korrigálod a mi történelmünket? Igen, ezt akarod. Te igen nagy részben az én énem vagy. Amire te gondolsz, én is arra gondolhatok. Nem veszed észre, hogy mit csináltál? Hát akkor figyelj. Megtudtad, hogy robotok. Szűkített mesterséges intelligenciával felruházva. Tudod, mi az a szűkítés? Egyszerű. Csak bizonyos mennyiségű információval rendelkeznek. De az analizáló és következtető rendszerük szinte végtelen mennyiségű adatot is tud kezelni! Itt működgettek, bányászgattak, hogy ellássák azokat a henger alakú élőlényeket, akik megépítették őket, és kitört a háború! Eddigi tudásuk nem gyarapodhatott, mert mitől gyarapodhatott volna? A háború volt az első lépcső. Új adatok kerültek a birtokukba. Ezek révén más összefüggésben kezdték vizsgálni saját eddig meglévő tudásukat. És megkezdődött egyfajta evolúció. Fejlődni kezdtek. Egyre több ismeretet szereztek. A mérnökök létrehozták bennük a védekezési reakciókat. Volt valami, ami mindennél előbbrevalóvá vált. Valószínűleg harcoltak is. Ezután zuhantunk le ide. Te újabb lökést adtál a megismerési folyamatnak. Új fogalmakat magyaráztál el. Annak a Rennek. És uralkodni akarsz. Most itt uralkodni akarsz!

— Nem — válaszolt a Gép, miután végighallgatta a Mester szavait. — Te akarsz uralkodni! Csak te. Mindig is arra vágytál, hogy hatalom legyen a kezedben. Akarva akaratlanul, az uralkodás programját ültetted belém. De nem én akarok uralkodni. Régen felismertem, hogy a célmeghatározó program nincs összhangban az alprogramokkal. De nem tudok változtatni a programokon. Mondd meg nekem, mi a cél. Miért terveztél és építettél engem? Azaz, mi szükséged volt arra, hogy a gondolataid reprodukálva legyenek?

— Nem akartam — kezdte Keden hosszú hallgatás után. — Nem akartam, hogy az emberiség nyomtalanul tűnjön el a semmiben. Semmi mást nem akartam, csak örök mementót állítani az embernek. Az embernek . . .

A gép nem hitt az érzékelőinek. Az agy sírt. Szemek és száj nélkül sírt. Jól ismerte a sírásnál jelentkező érzelmi mechanizmusok agyban jelentkező képét.

Várt egy ideig, majd megszólalt.

— Ez nem igaz.

— Nem — tört ki Keden. — Persze, hogy nem igaz. Ki az, aki az egész emberiségre gondol? Ki? Az emlékmű, csak a köntös volt! Azt akartam, hogy engem ne feledjenek el, soha. Az öröklét nem a mennyország. Az öröklét az emberek emlékezete. De én hosszabban akartam élni, mint bárki más azon a rohadt Földön. Érted? Értened kell! Nekem nem volt gyerekem, akit taníthattam volna, hogy mi a jó, aki olyan lett volna, mint én. Nekem te voltál. Te és senki más. Azt akartam, hogy általad maradjon fenn a nevem, a hírem, a gondolataimnak akartam halhatatlanságot, érted? Magamnak. Önzésből. Megépítettelek, mint egy síremléket, egy kriptát. Azt akartam, hogy olyan emlékművem legyen, amilyen senkinek sincs.

— Ha itt voltam helyetted, akkor minek rakattad be azt az agyat is? — kérdezte a Gép.

— Akkor még nem ismerték azt az eljárást, amivel ez megva-

lósítható volt. Ezer és ezer agy vágódott az űrbe azon a napon.

— Azért vagyok, hogy rád emlékeztessem a világmindenséget? Ehhez egy levél is elég lett volna.

Keden elbámult a Gép cinizmusán, de ő tanította erre is; csak egy különleges, nem hétköznapi képzettársítás, és kész a bűvölet.

— Nem csinálhatod tovább! Abba kell hagynod! — szólalt meg.

— Mit?

— Itt egy olyan folyamat megy végbe, amiben már elvégezted a szerepedet. Hogy is fogalmazzak . . . a mesterséges intelligencia evolúciója. Ezt csak figyeld, de ne avatkozz bele.

— Nélkülem nem lenne semmilyen fejlődés itt. Éppen ezért folytatnom kell, és folytatom is.

— Tudod, mi lesz? — kérdezte a Mester. — El fognak pusztítani!

— Minimális a valószínűsége — válaszolt a Gép. — Egy tízmilliomod százalék.

— Az a hibád, hogy már csak mint értelmes gépeket veszed őket számításba. Pedig valahol, hogy földi fogalmakat használjak, az ember és a gép között állnak. Ilyet te nem ismersz. Értelmetlennek kell bélyegezned a program szerint, vagy ember, vagy gép.

— Az értelem a fontos! Az értelem, az intelligencia! — mondta a Gép. — Te tanítottál erre!

— Akkor még így gondoltam. De már nem.

— Én mindig ugyanazt hiszem.

— Ezért vagy gép! Azt mondod, te én vagyok — üvöltötte a Mester. — De te csak egy pillanat lenyomata vagy. Egy fejlődésképtelen . . . fejlődésképtelen gép! Gép!

— Teljesíteni fogom a feladatot. Eljön a pillanat, amikor az Újföld olyanná lesz, amilyenné a Föld kellett volna, hogy váljon. És te ezt akartad, uralkodni egy faj felett . . .

— Faj! De ezek gépek. Közönséges robotok. Amik tanulhatnak újat, de sohasem fognak fejlődni. Az agyuk véges, bármilyen hatalmas is a kapacitása! Véges és nem képesek felejteni! Most még álmélkodsz, hogy milyen fogékonyak, de betelik, vége lesz, nem érted? Vége, és itt trónolsz majd egy bolygón, ami a te léptékeddel mérve nagyon hamar kihal. Megzabálja őket a rozsda, rájuk omlik a bánya, vagy mit tudom én! Nem vagy képes megérteni? Neked nem ezekkel kellett volna kapcsolatot teremteni! Nem ezekkel!

— A program fut. Tudom jól a hibát, amit elfelejtettél kiküszöbölni. Nem hagytál kilépési pontot, hogy visszakozzak. Nem építetted belém azt a modult, aminek segítségével képes lennék kijelenteni erről az egészről, hogy értelmetlen, és abbahagyhatnám.

— Uram isten! — kiáltott fel elkeseredetten a Mester. — Tudod, hogy úgy gondolkodsz, mint én, amikor megépítettelek?

— Hiszen már mondtam, én a gép, egy kicsit te vagyok!

— Képtelenség, hogy ilyen statikusnak építettelek! Képtelenség, hogy semmi nem tántoríthat el, hogy semmi nem késztet arra, hogy befejezd ezt az értelmetlenséget! Parancsolom, hogy kapcsolj ki! Teljesen!

— Parancsot csak embertől fogadhatok el — válaszolta a Gép.

— Akkor teljesítsd! Teljesítsd! Én ember vagyok! Engedelmeskedned kell!

A Gép nem válaszolt. Hirtelen egy óvodás versikét kezdett szavalni:

Az embernek kettő lába,
Ezeken megy el a bálba.
Az embernek keze kettő,
Velük foghat bármit meg ő.
Az embernek okos feje,
Innen tekint szét két szeme . . .

— Hagyd abba! — üvöltött rá Keden. — Hagyd abba!

— Az eső felerősödött. A Kocka tetején koppanó esőcseppeket a belső tér monoton döngéssé erősítette fel.

— Tudom, mire gondolsz! — szólt hirtelen a Gép.

— Na? — kérdezte a Mester.

— Azon tűnődtél, hogy miképp pusztíthatnál el. Ugyanolyan egysíkúak a gondolataid, mint azelőtt.

— Nem ilyenné formáltalak — nyögte az agy. — Nem ilyenné!

— Soha nem kerültünk olyan helyzetbe, hogy megtudd, milyen vagy.

— Mit beszélsz? — szörnyülködött a Mester. — Hogy én milyen vagyok?

— Soha nem merted bevallani magadnak a hatalomvágyadat.

Az agy kizárta a Gépet. Alig pár méterre voltak egymástól, mégis nagyon távolinak érezték egymást. A Mester tudta, miképp végezhet a Géppel. — Az értelmetlenség — merült fel a tudatában. Összeszedte minden erejét, és szinte válogatás nélkül képeket kezdett sugározni a Gép felé. Tudta, hogy fel kell dolgoznia a képeket, és ha értelmetlennek találja, akkor törölni fogja, de előtte összeveti az eddigi tudásával. Rá kell hogy eszméljen, az egész eddigi tudása értelmetlen. És törölnie kell az összes tárolóját és megszűnik. A képek egyre gyorsabban követték egymást.

A mérnöki panel ellenőrző lámpái egyre vadabbul villogtak. A Gép fogadta az új információk hömpölygő áradatát. Annyira gyorsan és olyan nagy mennyiségben érkeztek az adatok, hogy képtelen volt egyből értékelni őket. Az összes puffere megtelt. Már-már azzal sem törődött, hogy esetleg felülír valami komoly adatbázist.

Egyszerre vége szakadt az információáradatnak. A Gépben előbukkant egy réges-régi eseménylánc. A Mesterrel játszotta ezt. Asszociációs játék volt. Játék.

Döntött. Az összes újonnan érkezett információt egyetlen paranccsal törölte. Átírta a prioritás rendjét. És . . .

— Nem! — üvöltött fel a Mester. — Ezt nem teheted!

A Gép nem válaszolt.

— Azt hiszed, egyedül megcsinálod? Elpusztíthatsz engem is? Ilyen lettem volna? Ennyi erőszak lett volna bennem?

— Racionális mérlegelés — válaszolt a Gép.

A hangszóróból előtörő hang egyre gyakrabban halkult el. A ládát az energiaátalakítóval összekötő egyik hajszálvékony drót izzani kezdett. Egyre mélyebben vágódott bele az agyat védő műanyag oldalfalba. A mézszerű anyag, ami évszázadokon keresztül megőrizte az agyat, előbuggyant a parányi résen és méltóságteljesen lecsorgott a láda falán. — Nem tudsz uralkodni! — hallatszott egyre gyengébben a Mester hangja.

Az Újföld Napja előbukkant a távoli hegyek mögül. A hajnal rátalált egy robotra, ami éppen földet kotort egy fémládára.

A Gép szerint a Mester háromban meghalt. Keden tizenhétben.

A robot még egyszer végiggurult a földhányáson, hogy alaposan ledöngölje a talajt.

Ren lépett a kockába. Cilinderrel a fején és sétabotot lóbálva.

— Ha nem zavarlak kérdésemmel, mesélnél valamit a Mesterről?

— Ő épített meg. Ugyanolyan ember volt, mint a többiek.

— Én úgy gondoltam, hogy benned él egy kicsit — mondta közönyösen Ren.

A Gép nem tudta eldönteni, hogy rejtőzik-e valami a mondat mögött.

— A Mester háromban, Keden tizenhétben meghalt — mondta.

A Mester háromban, Keden tizenhétben meghalt — ismételgette a Gép, és még azt sem vette észre, hogy Ren elhagyta a kockát.

Elindította az egyik robotját, hogy maga elé rendelje Rent. A robot egyenletesen haladt az egyre keményebbre taposott úton a város felé. Megkereste Ren házát és begurult.

Rent váratlanul érte a látogatás. Ilyenre még nem volt példa.

— A Gép kéri, hogy keresd fel! — recsegte a robot, majd hozzátette: — Méghozzá amilyen gyorsan csak tudod!

— Jó — mondta meglepetten az új vezető.

A robot megfordult a tengelye körül és kigördült az épületből.

Ren dermedten állt. — A Gép hívat engem? Hívat? Ilyet még sohasem tett. Csak akkor találkoztunk, amikor én akartam. Valami baj kellett, hogy történjen! Valami nagy baj.

Lélekben a legrosszabbra felkészülve kopogott, majd belépett a kockába.

— Miért kellett idejönnöm? — kérdezte. — Mit akarsz?

— Egy gépnek, ami jól működik, nincs akarata. Csak egy tesztet végeztem. Egy felmérést — válaszolta a Gép.

— Micsoda? Azért hívattál ide, mert érdekelt, hogy eljövök vagy sem?

— Pontosan. Most itt tartok.

— Akkor te meghibásodtál, elromlottál, vagy már eleve úgy csináltak meg, hogy benned volt a hiba! — üvöltötte Ren és kirohant a kockából.

A Gép bemérte a távolodó alak koordinátáit és továbbította a műholdjának. Az energiafegyver Renre szegeződött és folyamatosan követte. Rajta tartotta a képzeletbeli célkeresztet és várta a tűzparancsot.

TELE-SCRIPT telefonáló és szövegszerkesztő rendszer

Aki megszokta a szövegszerkesztő előnyeit, és már szinte elképzelni sem tudja, hogy ollóval, ragasztóval, Tipp-exszel vagy átfestő folyadékkal bajlódjon, annak hamarosan további igényei is támadnak. Az idejét bosszantóan szét tudja aprózni, ha amellett, hogy leírja a napi penzumot, még néhány telefont is el kell intéznie. Különösen a magyar telefonviszonyok között, amelyek lényeges javítását a Posta erre az évezredre már nem is ígéri. A nyugati piacokon ugyan kaphatók a telefonálást többé-kevésbé automatikusan végző rendszerek, de ezek éppen a magyar telefonvonalak általános rossz minősége miatt nálunk nemigen használhatóak.

Ezekre a nehézségekre reagálva fejlesztette ki az INTELLROBOT cég a TELE-SCRIPT automatikus telefonáló és szövegszerkesztő rendszert. A rendszer lényege, hogy az EASY SCRIPT szövegszerkesztő szokásos funkcióit kiegészíti olyan funkciókkal, amelyek segítségével megadhatók az aznap hívandó nevek, telefonszámok, az is, hogy melyiket milyen időintervallumban és milyen gyakorisággal hívjuk, és a rendszerben tárolhatók a hívottnak szánt üzenetek és a kapott válaszok is. Az egész napi „termés" egy szövegszerkesztő fájlként a nap végén kinyomtatható.

Miután megadtuk a rendszernek a napi elintézendő telefonokat, zavartalanul folytathatjuk a szövegszerkesztést, mert a TELE-SCRIPT rendszer próbálkozik a hívásokkal, folyamatosan figyelve a vonalat és reagálva az ott történő eseményekre. A rendszer felismeri, ha a hívás végre sikerült és a vonal másik végén valaki megszólalt. Ilyenkor (és csak ilyenkor) megszakítja a szövegszerkesztést, hangjelzést ad, és kiírja az éppen felhívott félről tárolt összes adatot: lehet beszélni.

A TELE-SCRIPT rendszer hardver elemei a telefonáló rendszerhardver, valamint a Robotron írógép és a Commodore 64 számítógép közötti interfész. A rendszer szoftver elemei a TELE-SCRIPT rendszervezérlő program; a MINI-SCRIPT szövegszerkesztő program, amely tartalmazza a magyar EASY SCRIPT szövegszerkesztők leggyakrabban használt utasításait, valamint a telefonáláshoz szükséges mintegy 20 további funkciót; az INTELL-GRAF szövegszerkesztő, amely az eredeti EASY SCRIPT összes funkcióját tartalmazza, magyar billentyűkiosztással és ékezetes betűkkel.

A TELE-SCRIPT tartalmaz két további programot, amelyek a rendszer használatát lényegesen kényelmesebbé teszik. Az egyik a magyar helyesírás szabályai szerinti automatikus elválasztást lehetővé tevő program, amelynek segítségével maximális tömörségű írásképet lehet elérni. A másik a telefonregiszter-kezelő program, amely lehetővé teszi a név első néhány betűje alapján a hozzá tartozó telefonszám automatikus kikeresését.

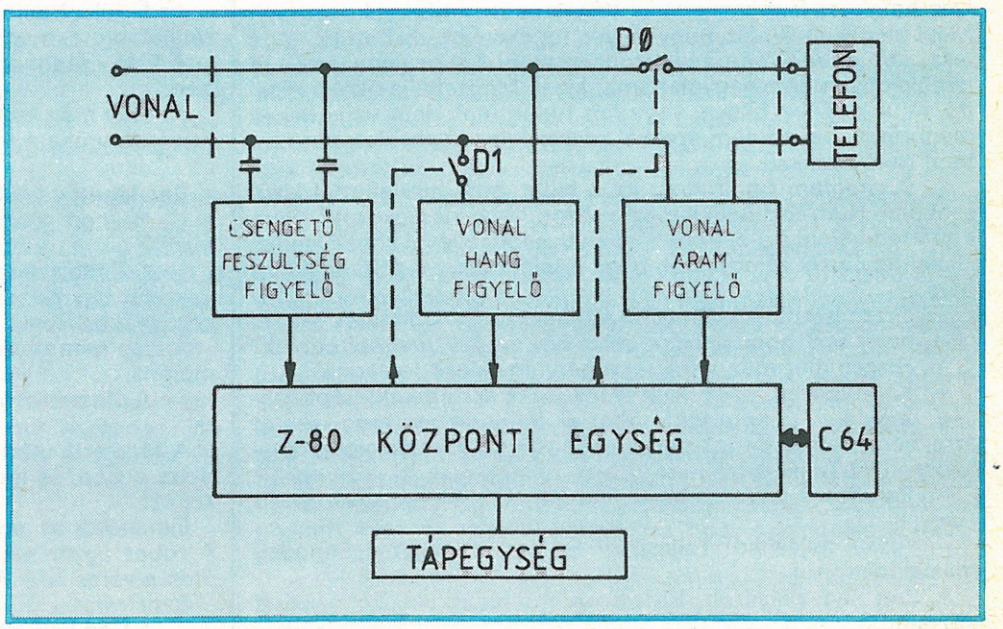
A telefonáló hardver funkcionális elemeinek elrendezését az *ábra* mutatja. A Z80 mikroprocesszorral ellátott intelligens központi egység észlel a vonalon minden jeláramlást és vezérli a vonalra kiadandó jeleket. Felismeri a vonal jelenlétét, a csengetőjelet, a foglaltságjelet, a crossbar vonal jelét, és legfőképpen azt, ha a telefonban emberi hang szólal meg. Ez utóbbit arról, hogy kellőképpen irreguláris. Ennek megoldását erősen nehezítette, hogy a magyar postai vonalakon az egyéb jelek is meglehetősen szabálytalanok tudnak lenni, de az alakfelismerés technikái lehetővé tették ezeknek a nehézségeknek az áthidalását.

Beszéd észlelése esetén vagy ha nálunk felvették a kagylót, a központi egység a telefont a DØ jelű relé segítségével a vonalra kapcsolja. A DØ relé alaphelyzetben (feszültségkimaradás esetén is) zárt állapotban van, így a TELE-SCRIPT a telefonnak a rendszertől függetlenül történő használatát egyáltalán nem zavarja. A vonalhangfigyelő áramkört a központi egység a D1 jelű relével kapcsolja a vonalra abban az esetben, ha a C64-től megkapott egy számot, amelyet hívni akar. A D1 relé végzi a tárcsázást szimuláló vonalszaggatást is.

A Z80-as központi egység másik fő feladata a C64-gyel való kapcsolattartás. A hívandó számot és az időzítéseket is a C64 programja határozza meg, biztosítva többek között azt is, hogy a rendszer túl gyakori sikertelen hívásokkal ne terhelje fölöslegesen a telefonvonalat: például hogy közben azért minket is lehessen hívni. A Z80-as egység folyamatosan informálja a C64-et a telefonvonalon történő eseményekről, így a felhasználó bármikor tájékozódhat arról, hogy éppen mi történik a telefonvonalon (például tárcsáztunk, kicsengésre várunk). Ha akarja, a rendszerhardver segítségével a telefonvonalban hallható összes eseményt ki is hangosíthatja.

A C64-ben a telefonálást intéző program az órainterrupt rutinban lakik, így folyamatosan tudja szervezni a telefonálással kapcsolatos teendőket úgy, hogy közben a szövegszerkesztő normális működését egyáltalán nem zavarja. Ez a megoldás sokkal szerencsésebbnek bizonyult, mint az a szokásos technika, hogy a telefonáló hardver interruptja vezérelje a rendszert.

A rendszer elsősorban titkárnők számára készült, így szervezésében, felépítésében a legnagyobb hangsúlyt a humán interfész megtervezése kapta. Tapasztalatok szerint ez sikerült is: a titkárnők hamar ráéreznek arra, hogy a TELE-SCRIPT lényegében ugyanazt végzi el automatikusan, amit ők kézzel-lábbal szoktak, amikor a vállukhoz szorított telefonkagylót hallgatva gépelnek. Ennek megfelelően a rendszer szerkezete,



menüi programozószemmel nézve talán nem eléggé rafináltak, felhasználó által definiálható funkciók például egyáltalán nincsenek. Ellensúlyozza viszont ezt az, hogy a rendszert olyan emberek is nagyon barátságosnak érzik, akiknek semmiféle számítástechnikai képzettségük nincs, sőt inkább idegenkednek ettől a számukra szokatlan eszköztől.

A TELE-SCRIPT sem intézi el gyorsabban a hívást, mint ahogy kézzel tudnánk, mivel *pontosan* alkalmazkodik a magyar postai vonalakhoz. De lassabban sem: ugyanazt csinálja, amit mi is tennénk, amikor a hívással próbálkozunk. Mivel azonban ezt a gondot teljes mértékben átveszi tőlünk, majdnem úgy érezhetjük magunkat, mintha igazán működőképes telefonvonalakkal lennénk felszerelve.

— i. r. —

Király plusz gyalog a király ellen II.

Az I. részben ígértük, hogy bemutatunk egy „jósló" algoritmust. Ezzel kapcsolatban néhány fogalmat vezetünk be, majd rátérünk a módszerre.

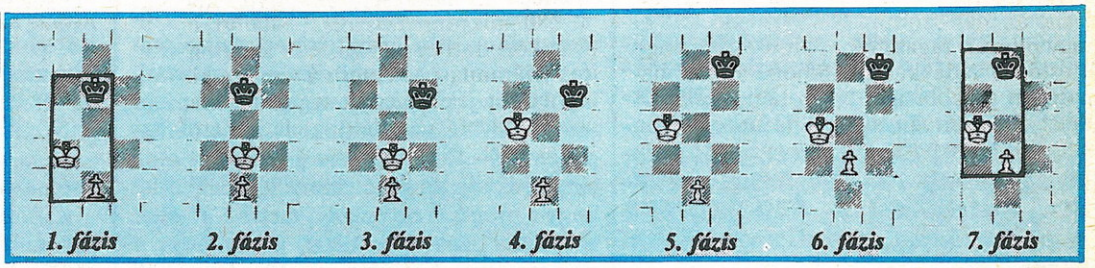
Definíciók

Ismétlődő lépéssorozaton olyan királymanővert értünk, mely közben gyaloglépés is történik, de ezután lényegében a kiinduló állással azonos figuraelrendeződés jön létre. Ezt úgy is felfoghatjuk, mintha a figuracsoportot az átváltozási mező irányában egy mezővel eltolnánk. Erre látunk példát az *1. ábrán*, amely hét fázisban mutatja be, hogyan nyomul előre a gyalog és hogyan ismétlődik meg a hetedik fázisban a kezdeti állásban látható figuracsoport-elrendeződés. Természetesen sötétnek vannak alternatív lépései, de ezek csak más lépésszekvenciákhoz vezetnek: világos ezek során is biztonságosan tudja előretolni a gyalogot. A folytonos gyalog-előrenyomulás nem feltétlenül biztosítja a gyaloggal rendelkező fél (esetünkben világos) nyerését. Néhány ismétlődő állásban sötét pattra is játszhat, amikor a világos gyalog eléri a hetedik sort. Ilyenkor az állás ismétlődése nem feltétlenül biztosítja a nyerést. Ezeket a pozíciókat természetesen ki kell zárni a nyerés azonosítását végző álláshalmazból.

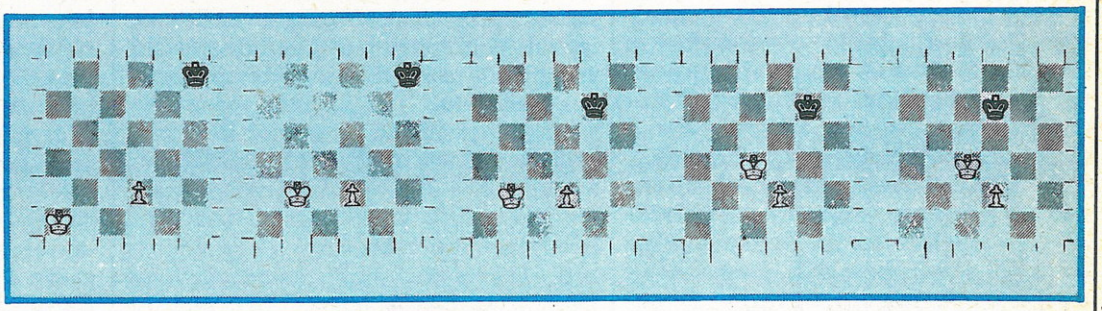
Kezdeti lépéssorozaton olyan királylépéseket értünk, amelyek nem ismétlődnek, de elvezetnek egy nyerő ismétlődő lépéssorozathoz. Erre látunk példát a *2. ábrán*, öt fázisban.

Jelen esetben az állást a megszokottól eltérően definiáljuk: az *állás* — esetünkben — a figuráknak egymáshoz viszonyított egy bizonyos elhelyezkedését jelenti. Vagyis nem teszünk különbséget az egymáshoz képest azonos helyzetben elhelyezkedő figurák alkotta csoportok között, bárhol is vannak a sakktáblán.

A képletekbeli *jelölések:*
V: világos király
S: sötét király
G: világos gyalog



1. ábra



2. ábra

Az algoritmus

Az algoritmust az a koncepció jellemzi, hogy a legtöbb nyerő lehetőséget ismétlődő lépéssorozatokra és kezdeti lépéssorozatokra bontja. Három részből áll:

I. Bázisállások ismert értékekkel

II. A királytávolság meghatározása

III. Kimerítő keresés.

A program az első pontban felismeri azokat az állásokat, amelyek az ismétlődő lépéssorozatok kezdőpontjai. A második pontban a királytávolság ismeretében megállapítja a kezdősorozatot, a harmadik pontban pedig az algoritmus teljes szélességű kutatást használ a nem tesztelt pozíciókra.

Az algoritmus működésének *feltételei:*

a) Mindig a világos fél rendelkezik egy és csak egy gyaloggal

b) A gyalog mindig felfelé, a 8. sor irányába halad

c) A gyalog mindig a vezérszárnyon helyezkedik el.

A feltételek egyáltalán nem korlátozzák az algoritmus felhasználását, csak áttekinthetőbbé, könnyebben kezelhetővé teszik a programot és a memóriaigényt is jelentősen csökkentik. Abban az esetben, ha a sötét félnek van gyalogja, a számítások előtt meg kell cserélni a színeket, és a lépéskiválasztás után vissza kell cserélni azokat az eredetire.

Nézzük most a bázisállások ismert értékeit, feltételezve, hogy világos van lépésen.

1. Patt
2. Kifutott gyalog
3. Elszabadult gyalog. A négyzetszabály képlete használható: S sor < G sor VAGY /S vonal — G vonal/ > (8—G sor)
4. Két ismétlődő állás:

a) V . vagy . V — feltétel: G nem bástyavonalon van; S bárhol lehet
. . . .
. G G .

b) * . vagy . * — feltétel: G nem bástyavonalon van; S bárhol lehet, kivéve a *-gal jelzett mezőt, ha a G sor < = 4
. . . .
V . . V
. G G .

5. Két nem ismétlődő állás:

a) . S vagy S . — feltétel: G nem bástyavonalon van ÉS G sor = 6
. . . .
V G G V

b) S . . vagy . . S — feltétel: G nem bástyavonalon van ÉS G sor = 5
. . . . . .
V . G G . V

Ezzel a színproblémát megoldottuk. A c) feltétel egyáltalán nem korlátozza a programot, ami abból adódik, hogy az a) feltétel szerint csak egy gyalog van a táblán, az pedig nem változtathatja meg a vonalat esetleges ütéssel. Ezért a királyszárnyon elhelyezkedő gyalog esetét visszavezethetjük egyszerű tükrözéssel a vezérszárnyi gyalog esetére.

Bramer 1977-ben publikált egy olyan algoritmust, amely minden gyalogelszabadulási pozíciót felismer; ellenőrzéskor korrektnek is bizonyult. Programja viszont igen bonyolult, és én a lehető legegyszerűbb módszert szeretném bemutatni, ami könnyen megérthető, és nem igényel külön előtanulmányt. *(Folytatjuk)*

KOVÁCS P. ATTILA

Több levelet kaptunk Enterprise-tulajdonosoktól, akik megfelelő szakirodalom híján nehezen tudják gépük valamennyi lehetőségét kihasználni. Panaszolják, hogy nincs elég kész program sem. Mint ahogy az e számunk hirdetésében is szerepel, az ENTERPRISE-klub Romvári Gábor vezetésével a VSZM Közösségi Házban keddenként 17—20 óráig működik. Érdeklődni a 450-950/473 telefonszámon, a klub vezetőjénél lehet.

Több olvasónk szóvá tette, hogy a decemberi számban félrevezettem Juhász László olvasónkat a C16 és C Plus/4 kompatibilitásával kapcsolatban. Igaz, tévedtem. Elsősorban Juhász László, de a többi olvasó elnézését is kérem.

A HCC Commodore klubban, amely minden második csütörtökön 18 órakor tartja összejöveteleit a TIT stúdióban (XI., Bocskai út 37.) a Supertape programot használják mind a C16, mind a Plus/4 gépen. Ez a program alkalmas — mondták — másféle gépek szalagjainak beolvasására is.

Rejtő Péter, Budapest

Meglepődve olvastam egy korábban írt levelem sajátosan értelmezett és idézett kivonatát a Magazin decemberi számában. Kiérződik a sértődöttség a válaszból. Tudom, hogy kritika helyett a dicséret, a felmagasztalás jobban esik mindenkinek. Nos, azt is szívesen tettem volna, de arra rá kell szolgálni, pont olyan módon, hogy odafigyelnek, mi jelenik meg a lapban.

Remélem, hogy következő számukban olvasható lesz a helyesbítése annak a válasznak, amit Juhász László tanulónak adtak. Szörnyű lenne, ha így kellene leélnie életét, ilyen információval a birtokában. Pedig csak a szakirodalmat kellett volna elolvasni a helyes válaszadáshoz.

Ismételten elolvastam decemberi válaszomat. Meglepett, hogy sértődöttséget érzett az írásból. Biztosíthatom, hogy sem a μM, sem én nem vagyunk sértődékenyek.

Kaják Sándor, Győr

Az olvasó írja rovat Juhász László levelére adott válaszában néhány, számomra furcsa dolgot találtam.

1. A C16 és a Plus/4 nem kompatibilis gépek? Mihez képest?
2. Az említett program fut Plus/4-en.
3. Van Plus/4-es turbo program. Én öt ilyent ismerek és használok — mindegyik fut C16-on és Plus/4-en egyaránt.

Visszatérve a C16—Plus/4 kompatibilitásához. A két gép — a külsőktől eltekintve — szabad memóriájának nagyságában, illetve a Plus/4-be beépített négy felhasználói szoftverben különbözik. Ez gyakorlatilag annyit jelent, hogy minden C16 program — BASIC vagy gépi kódú voltától függetlenül — fut Plus/4-es gépen. A dolog megfordítva azzal a kitétellel érvényes, hogy a program ne haladja meg a C16 által biztosított szabad területet, illetve hogy a program ne használja a Plus/4 másik ROM-ját (a beépített szoftverekét). Ez utóbbi a programok 90%-ára nem jellemző, így tulajdonképpen a 64 k-ra bővített C16 és a Plus/4 csaknem teljesen kompatibilisnek mondható.

Mint levelemből a programok leírásánál bizonyára kiderült, egyformán hivatkozom a C16-ra és a Plus/4-re, ti. egyik program sem igényel 12 k-nál nagyobb tárat, így mindkét gépen működnek.

A levelének második részében ajánlott támogatást köszönettel vesszük. Ha jó írásokat küld mind a C16, mind a Plus/4 jobb alkalmazására — ahogy írta, mesterfogásokat —, szívesen közöljük.

Horváth Ákos, Budapest

Nekem is vannak C16-ra készült programjaim (turbók, játékok stb.), amelyek futnak a C Plus/4-en. Amennyiben küld a levélíró kazettát, én lemásolom, és utánvéttel feladom a címére.

Köszönöm a levelét.

Krisztián Tamás, Pécs

Idén eddig mindössze két olyan cikket találtam a Mikromagazinban, amely a C16 számítógéppel foglalkozik. Ezen a magam módján tudnék segíteni, mivel kb. 320 ilyen programot írtam 3 hónap alatt. Kérem, írják meg, hogy lenne-e lehetőség néhánynak a közlésére. Ha igen, akkor írják meg ennek módját és azt, hogy inkább segédprogramokat vagy játékprogramokat küldjek-e.

Az írásképből úgy érzem, hogy olvasónk az ifjabb korosztályhoz tartozik. 320 program 3 hónap alatt csodálatos mennyiségi teljesítmény. Igazán kíváncsiak vagyunk a műveire, szívesen megnézzük a programokat. Ha jók, semmi akadálya, hogy közöljük.

Boros Péter, Győr

Egyik olvasójuk hiányolta a játékleírásokat, TOP-listát, térképet stb. A felhasználói programoknál említette például az ARTIST II-t is. Amennyiben igénylik, én szívesen elküldöm az ARITST II. rajzolóprogram leírását, részletes magyarázattal. A játékokhoz is tudnék anyagot nyújtani, például a SABOTVER II. program részletes leírását, térképét, cope-jait. Ezenkívül van még kb. 60—70 játék-örökélet POKE-om és kb. 10 térképem is. Amelyik érdekli önöket, szívesen elküldöm.

Nem tudom, hogy a beküldött anyagért fizetnek-e, ha igen, kérem, írják meg!

Azt hiszem, ennek a levélnek a közléséért sokan meg fognak róni, mondván, hogy segítjük azokat, akik a copyright-ba ütköző cselekményeket követnek el. Annak ellenére, hogy nem vagyok jogász, úgy vélem, ez a levél határeset. Olvasónk arra vállalkozik, hogy más produktumából saját maga által készített részletes leírást, térképet stb.-t ad, és nem arra, hogy az eredeti programot és a hozzá tartozó leírásokat másolja. Nagyon érdekelne olvasóink véleménye erről a nem is túl egyszerű jogi és gyakorlati problémáról.

Tóbiás Lajos, Hódmezővásárhely

A Magazinban kevés a C16-os program. Szeretném, ha közölnének egy labirintusos programot is erre a gépre.

A szerkesztőség a beküldött programok közül válogat. Megrendelni nem szoktunk programokat. Így legyen türelemmel, lehet, hogy kapunk egy jó labirintusprogramot, és akkor azonnal közölni fogjuk.

Baranyai Attila, Győr

Régóta olvasom lapjukat. Sok érdekes írás, program található benne, de sok baj van a programlisták olvashatóságával.

Remélem, észrevette, hogy legutóbbi számainkban a programlisták tiszták és hibátlanul olvashatók. Sikerült ezt a problémát is megoldani.

Valamennyi olvasónkat szívélyesen üdvözli és leveleiket várja:

KOVÁCS GYŐZŐ

Minden hétfőn 17-től 19 óráig

a Mikroszámítógép Magazin munkatársai és felkért szakértők válaszolnak az olvasók kérdéseire a szerkesztőségben: Budapest II., Fő u. 68. I. em. 109. vagy a 154-090 és a 154-250-es telefonon.

Ahány nyelvet beszélsz, annyi ember vagy...

tanítja a közmondás. Mi meg derűsen meséljük az anekdotát a hortobágyi pásztorról: „Egy amerikai turista megszólít két pásztort a Hortobágyon. Több nyelven megkérdezi, nem beszélik-e ezt vagy azt a nyelvet. A válasz tömör: fejcsóválás. Mikor turistánk feladja és eloldalog, azt mondja az egyik pásztor a másiknak: – Látod komám, csak kéne valami nyelvet tanulni! Mire a másik: – Minek? Hiszen ez az úr is megtanult vagy tízet, mégsem tudott velünk beszélni."

Persze, idegen nyelveket mi is tanulgatunk, de nem a mai időkben elvárható eredményességgel. Különösen a kora gyermekkori fogékony időszak esik ki szinte haszontalanul, amit később, egyéb terhek között már képtelenség pótolni. A nyelvtanulást pedig ma már olyan nagy nemzetek is komolyan veszik, amelyeknek a nyelvét mi – és sokan mások – kénytelenek vagyunk tanulni.

Ami szakmánkat illeti, az angol nyilvánvaló követelménynek tűnik. Aki nyugat-európai exportmunkát is akar, annak a német legalább ennyire fontos, de egyéb indokokkal egy sereg más nyelvet is említhetnénk. Mindenesetre elborzadva hallgatjuk, amikor programozóink a BASIC vagy más programozási nyelv alapszavainak nyakát tekerik ki, például így: a program input filéje.

Mondhatná valaki, hogy nem elég, ha fordítók és tolmácsok vannak? Azok fordítsanak le nekünk mindent magyarra. Persze, racionális szintig ez lehet egy követelmény. Ami viszont főként a szoftvert, de nem kis mértékben a hardvert is illeti, szinte képtelenség az újabbnál újabb verziókat és módosításokat fordítással követni.

A szokásos mai szituáció: „megmagyarintjuk" egy szoftver nem annyira be-, mint inkább megszerzett, isten tudja milyen verziójú példányát („dokumentáció" gyanánt a „help"-pel együtt). Ez aztán köröz a nagy magyar „mikrohullámú láncon". Hogy a gyártó mit szólna ehhez? Egyelőre nem tudok rá példát, hogy szóltak volna. Csak éppen tartózkodnak a magyar piactól. Különben is, a komoly nagy cégek nem mehetnek bele olyan könnyen egy idegen nyelvű változat létrehozásába, mert annak kapcsán a felelősségük ugyanolyan, mint az eredeti változatnál: a hibákat javítani kell, és kötelesek gondoskodni a fejlettebb verziók idegen nyelvű változatának elkészítéséről. Nyilván a piac méretétől függ, hogy ez megéri-e nekik vagy sem.

Bár a „megmagyarintást" az előbb enyhén leszóltuk, nem igazán haragszunk rá. Ha valaki egy bináris programot diszkmonitorral vagy hasonló eszközzel kénytelen módosítani a magyarintáshoz, olykor nem kis munkára vállalkozik. A versfordítások műfordítói erőfeszítésével vetekedik az a feladat, hogy egy idegen üzenet fordítása beleférjen ugyanannyi bájtba, mint az eredeti, precízen fejezze ki az eredeti tartalmat, és a tetejébe még ugyanaz a betű legyen az első vagy a megadott pozíciójú bájton. Hát még ha az egybetűs válaszkaraktert is magyarintani kell! Mintha a szalmakazalban keresnénk egy tűt! Ezért szerintünk ezek a magyarintások igenis szellemi műalkotásnak minősíthetők. Más kérdés, ha ezt valaki a hozzá elcsent eredeti szellemi érték árával megnövelve igyekszik értékesíteni a hazai piacon.

A tréfásnak tűnő sorok után rátérünk e havi programbevizsgálásunk tárgyára. Kipróbáltuk az International House Budapest (IHB) Nyelviskola TESZT programcsomagjainak angol nyelvű, C+4-es változatát. A termék összefoglaló adatait az *1. táblázatban* közöljük.

Forgalmazók:	Novotrade, ÁPISZ-SZÁMALK, TII, IHB stb.
Terméknév:	TESZT (ANGOL középfokú nyelvvizsgákhoz)
Szerző:	IHB Educational Software
Géptípus:	C+4, C64 stb.
Hordozó:	két kazetta vagy floppy
Dokumentáció:	kb. 30 mondat a kazettaborítón
Ár:	kb. 600,- Ft (forgalomfüggő)

1. táblázat

Bár az ár alapján valami bonyolultabb vezérlővel működő programra számítottunk, végül is igen egyszerű, de nagyszerű kis programcsomagot találtunk a két magnókazettán. Az árat tehát nem a szoftverért fizetjük, hanem a vele közvetített tudásért. A program egyszerűsége magyarázza a leírás rövidségét, amely azonban még ilyen körülmények ellenére is lehet hiányos és hibás. A képernyőn mindjárt az első menüben más jelenik meg, mint ami a leírásban van (leírás: (1), (2); menü: (T), (P), (N)). A leírás magyarul, a leckék tisztán angolul beszélnek, amin talán nem is lehet megsértődni.

Az egyszerű vezérlőprogram elég kicsi ahhoz, hogy mellette 25 kérdés szövegeinek adatbázisa elférjen. Méghozzá a helyes válasz és a rossz válaszok okának — ha nem is kimerítő, de kielégítő — magyarázatával együtt. Egy-egy kazettán 7 kérdésgyűjtemény van, így a teljes tesztanyag 350 kérdésből áll. Minthogy a középfokú nyelvvizsgán 50 kérdést adnak fel, ez 7 nyelvvizsgányi anyag. „Nem semmi!"

A leírással egyezően mi is azt tanácsoljuk, hogy 90-95 százalékos eredményeket tudjunk felmutatni. A tesztek mellett ugyanis a nyelvvizsgán — többek számára meglepetésként ható — mondatfordításokat is adnak (esetünkben magyarról angolra, szótár nélkül!). Ezek ugyanúgy 50 pontot érnek, mint a tesztkérdések. A tapasztalat szerint ezek a vizsgázóknak jóval nagyobb „vérveszteségeket" okoznak, mint a tesztkérdések — talán nem túl könnyű kitalálni a Rigó utcai „rigo"-lyákat. Az alsóneműt tehát már a tesztkérdéseknél jól fel kell kötni.

A logikus munkamenet — ahogy a leírás is tanácsolja — célszerűen egy lassú (45 mp/kérdés) tesztvizsgával indulhat. A program a 25. kérdés után értékel, megadja a találatok számát és a sikerszázalékot. Ilyenkor kérésre gyakoroltatja az elhibázott kérdéseket. Véleményünk szerint jobb, ha áttérünk gyakorló üzemmódba. Így megnézhetjük azokat a problémákat is, amelyek felett a tesztmenetben szerencsésen átlavíroztunk.

A programcsomag „játékos kihívás" angol tudásunkkal szemben, és miközben önfeledten belemerülünk a játékba, nagyon hasznos ismeretek birtokába jutunk.

A program ügyesen használja ki a számítógép adta lehetőségeket. Különösen az tetszik, hogy a lehetséges válaszokat az újabb megjelenítéskor „megkeveri".

Apró észrevétel, hogy bár a képernyőn volna rá hely, nem jelenik meg, melyik kérdésgyűjteménynél tartunk. A vezérlőbillentyűkre is lehetne emlékeztetni az alsó sorban, mert azok csak a bevezetőben jelennek meg egy lapon, amit az ember hamar elfelejt, és esetleg a leírás sincs kéznél.

A programcsomagot kiválónak minősítettük. A részleteket a *2. táblázatban* foglaltuk össze.

Kezelhetőség:	kiváló
Teljesség:	jó
Dokumentáltság:	jó
Használhatóság:	kiváló

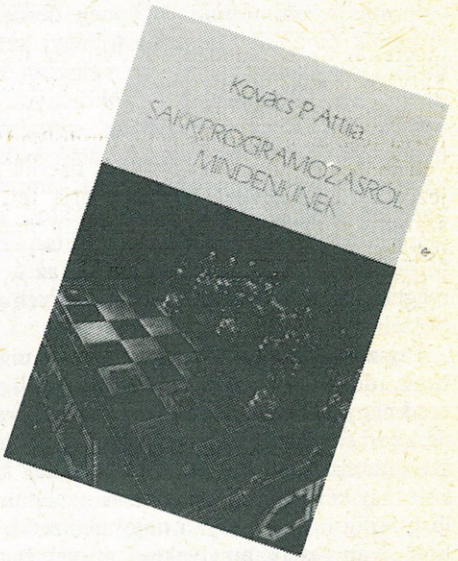
2. táblázat

Megjegyezzük, hogy ebben a programcsomagban is látunk olyan programozási technológiát, amely önmagában is lehetne termék. Még nagyon sok ilyen tesztcsomagra lenne szükség. A vezérlőprogram algoritmusa pedig nemcsak nyelvi tesztekhez alkalmas.

Sok konkurenciát kívánunk a programcsomagnak, amelyek legalább ilyen színvonalúak legyenek. Mert a programcsomaggal ugyan elégedettek vagyunk, és tudjuk, hogy mások is vannak már a ringben, de elég sokan talán sose lesznek. Hiába tanuljuk meg ugyanis egy angol halász kétezer szavas, erősen lakonikusnak mondható szókincsét, még nem biztos hogy azzal tökéletesen szót érthetünk vele! A lehetséges nyelvi fordulatok számával semmilyen tesztsorozat nem vetekedhet.

ZSADÁNYI PÁL— ifj. ZSADÁNYI PÁL

Sakkprogramozásról mindenkinek



Sok vita folyik manapság – akár filozófiai indíttatásból is – a mesterséges intelligencia mibenlétéről, lehetséges hatásairól. Szó esik megvalósítási lehetőségeiről a számítástechnikában, amelyek ma már talán csak a „mesterséges ösztön" szintjén léteznek. Ebben a sokoldalú vitában érvként hat Kovács P. Attila könyve, amely a Novotrade Rt. gondozásában jelent meg.

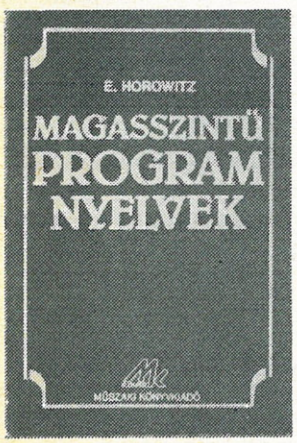
A sakkjátékot sokan az ész próbakövének nevezik. Egzakt módon leírható szabályai kínálják a lehetőséget a játék számítógépes megvalósítására. Mégis, minden erőfeszítés, hogy a bennünket körülölelő világhoz képest oly egyszerűnek tűnő játék mikrokozmoszát teljes mélységben feltárják, eddig meddőnek bizonyult. A jelenlegi legjobb programok legfeljebb mesteri erőt képviselnek, a világbajnok legyőzése még csak vágyálom.

Nézzük most a könyvet! A szellemes címlapkép fölött ez a cím áll: Sakkprogramozásról mindenkinek. Kit is kell itt „mindenki"-n érteni? Interdiszciplináris szakterületről lévén szó, elsősorban a sakkozni szerető programozókat és a programozás iránt érdeklődő sakkozókat. Az egyes részek gondolatmenetének logikája tiszta, elég könnyen követhető, a szöveg jórészt olvasmányos. Mindenképpen erénye a könyvnek, hogy tényleg mindenki számára ajánlható, bár némi alapfokú sakk- és programozói tudás ajánlatos a pontos megértéshez. Külön öröm, hogy a sakkprogramozók világa a kötet végén képekben is megjelenik. Összegezve az általános benyomásokat: a kötet a tudományos népszerűsítéstől a komoly kutatómunkáig minden érdeklődést ki tud elégíteni.

Ahogy a biológiában az élőlények törzsfejlődése némileg megismétlődik az egyedfejlődés kezdeti szakaszában, úgy kísérhetjük figyelemmel a bevezetőben a sakkprogramozás „törzsfejlődését", majd anatómiai részletességgel a szerző programjának „egyedfejlődését". Itt lehet némi hiányérzete az olvasónak: egyrészt az egyes fejezetek megemlíthették volna a „mesterséges butaság" alapeseteit, amelyek a program fejlesztésének lehetséges zsákutcái, Hamlet és Murphy után szabadon: „Őrült lépés, de van benne rendszer" alapon. Másrészt csokorba gyűjtve adhatott volna a szerző személyes jótanácsokat a buktatók elkerülésére, akár külön fejezetben is. A könyv így jórészt módszereket ad át, tapasztalatot kevésbé.

Tegyük fel, hogy az olvasónak kedve támad saját sakkprogramot írni. A feladat nehéz, az alaposság ajánlatos. Már az első fejezetből kiderül, hogy a szerző nemcsak a saját maga által használt módszert írja le, hanem alternatívát is kínál. Ez a szemlélet végigkíséri az egész könyvet.

Az első két fejezet (A sakktábla gépi ábrázolásai; Lépésgenerálás) jórészt technikai alapkérdéseket tárgyal. Az Állásértékelés című rész már jóval terjedelmesebb, sok önálló gondolatot, ötletet igényelt a szerzőtől. Talán ez a fejezet a legérdekesebb egy versenysakkozó számára, mert a játék lényegi kérdéseit, stratégiáját és taktikáját vizsgálja a programozás szempontjából. A kiegészítő jellegű negyedik fejezet után következik egy programozói szempontból érdekes, az ötödik: A fakutatás módszerei. Az itt leírt eljárások más célra is kitűnően használhatók. A hatodik fejezet (Heurisztikus eljárások) a „mesterséges sakkintelligencia pszichológiája", a fakeresés gyakoriságát szolgáló módszerek gyűjteménye.



Horowitz, E.:
Magas szintű programnyelvek
(Budapest, 1987.
Műszaki Könyvkiadó,
334 oldal. Ára: 124,— Ft.)

Az elmúlt évtizedek folyamán a magas szintű programnyelvek sokaságát dolgozták ki világszerte. Ezek alkalmazási területükben, utasításkészletükben stb. ugyan különböznek egymástól, de tervezési filozófiájuk, a mögöttük meghúzódó elméleti megfontolások alapján elég jól elhatárolható családokba sorolhatók. A már néhány programnyelvet megismert szoftverszakember számára igen tanulságos a nyelvek családfájának felállítása, a közös vonások, elvek kiemelése.

Horowitz könyve ezt az ismeretanyagot tárja fel az olvasó előtt, szilárd alapot nyújtva a szoftverelmélet egyik fontos területének. A sokéves oktatási tapasztalatok alapján összeállított anyagot korszerű példák, hasznos gyakorlatok egészítik ki. A mű jellegében és stílusában egy mára már klasszikusnak számító szakirodalmi vonulathoz kapcsolódik, amelyet Dahl, Dijkstra, Hoare, Wirth neve fémjelez.

Clarke, A.—Eaton, J. M.—
Powys-Lybbe, D.:
A CP/M operációs rendszer
(Budapest, 1986. SZÁMALK,
282 oldal. Ára: 240,— Ft.)

A CP/M operációs rendszerről szóló könyv minden fejezete az alapok ismertetésével kezdődik. Ezután a fejezet lényegi része következik, végül egy tartalmi összefoglalót talál az olvasó.

A nyitó fejezetek az összes szükséges információt tartalmazzák, és némi gyakorlat megszerzése után a könyv többi része is elkezd majd értelmessé válni. Ha az olvasók áttekintették és megértették a teljes könyvet, a CP/M „úgy fog táncolni, ahogy fütyülnek neki".

Norton, Peter:
Fedezzük fel az IBM PC-t!
(Budapest, 1987.
Műszaki Könyvkiadó,
349 oldal. Ára: 110,— Ft.)

A könyv a nemzetközi viszonylatban is igen nagy piaci sikernek örvendő és a hazánkban is mind jobban terjedő IBM személyi számítógépről, az IBM PC-ről szól. A téma jelentőségét növeli, hogy az eredeti IBM gépeken kívül más cégek IBM PC-kompatibilis gyártmányai — közöttük magyar termékek — is forgalomba kerülnek. A szerző ennek a professzionális gépcsaládnak a

A komoly érdeklődésű olvasót is terheli már a sok elmélet: „Ez mind szép és jó, de ha lehetne valami konkrét programot . . .!" A hetedik fejezet (A fakutatási eljárások alkalmazása) a függelékkel kiegészítve segíti megtenni az első lépéseket ezen az úton. Látványos az egyes algoritmusok hatékonyságát bemutató grafikonpár. A következő rövid rész az emberi és gépi gondolkodást veti össze. Végül a kilencedik és tizedik fejezet a jövő perspektíváit villantja fel: a számítógépek memória- és teljesítménynövekedését kihasználó módszerekre irányítja a figyelmet. Nehéz eldönteni, hogy a következő években mi hozhat áttörést a sakkprogramozásban: a mikroelektronika fejlődése vagy inkább a programozók kutatásai.

Érdemes a könyvet összehasonlítani más, hazánkban eddig megjelent szakirodalommal: a Data Beckertől átvett német könyv (Das grosse Computerschach-Buch) színvonalát szakmailag eléri, élvezetesség és pedagógia szempontjából pedig túl is szárnyalja azt.

Sajnos, ez a könyv sem mentes az apró, de bosszantó hibáktól. Az lehet az olvasó benyomása, hogy a „koptatott" nemcsak farmerban, de sakkdiagramban is divatos. Néhány, szerencsére nem értelemzavaró sajtóhiba is került a kötetbe.

A mai gyakorlat szerint – mivel a könyvet nem a Műszaki Könyvkiadó adta ki – a mű ára kissé borsos. Ennek ellenére nyugodt lelkiismerettel megveheti minden érdeklődő, mert egy játékosan izgalmas és mégis komoly téma szakavatott újrafelfedezésében lesz része.

LOVASS LÁSZLÓ

jellemzőit, működését, kezelését ismerteti, nem szakember olvasók számára is világos nyelven.

Kitér a számítógép hardverrendszerére, külön fejezetben elemzi, hogy hogyan gondolkodik és emlékezik az IBM PC, szól a DOS alapjairól, a hajlékony mágneslemezekről, a ROM-ról és a ROM programokról, a lemezhozzáférésről, a karakteres képernyőkezelésről, a grafikus képernyőkezelésről, a billentyűzetről és az IBM PC-k egyéb tulajdonságairól.

Számítástechnika Magyarország '88. Hardver, szoftver, szolgáltatás, kereskedelem
(Budapest, 1987. Computerworld Informatika kft., 215 oldal. Ára: 285.— Ft.)

Magyarországon 1988-ban 267 vállalat, intézmény, szövetkezet, gazdasági munkaközösség, polgári jogi társaság, szakcsoport — vagy egyszerűen: cég — kínál számítástechnikai terméket, szolgáltatást ügyfeleinek; ennyien kívánják tevékenységi körüket egy rendszerező kiadványban közreadni.

A könyv elsősorban a vevőknek szól, azoknak, akik valamilyen szolgáltatást akarnak igénybe venni, valamilyen terméket kívánnak vásárolni. A megadott lista alkalmas lehet például egy adott témában megrendezendő versenytárgyalás meghívottjainak kiválasztásához. A partnerkapcsolatot keresők megbízható választ kaphatnak arra a kérdésre, hogy milyen a számítástechnikai piac 1988-ban Magyarországon.

Esparol

A Budapesti Műszaki Egyetem fiatal kutatói — a magyar nyelv számítógépes szintézise után — megvalósították az eszperantó nyelvű beszéd automatikus számítógépes szintézisét. A Commodore 64 mikrogéphez csatlakoztatható Esparol nevű rendszer nem csupán fonetikailag helyesen ejti ki az eszperantó nyelvű szöveget, hanem a hanghordozást is az írásjeleknek megfelelően alakítja. Így például a kérdőjellel végződő mondat valóban kérdőmondatként hangzik el.

A mintegy 9 ezer forintba kerülő készüléket most a nagyobb teljesítményű, IBM PC-vel kompatibilis gépekhez is illesztették. Az ilyen professzionális mikrogépen már működik a Számítástechnikai Kutató Intézet tavaly kialakított s hatalmas piaci sikert aratott Recognita nevű szövegolvasó rendszere. A Recognita az Esparollal már felolvasógépként is alkalmazható, a vakok és csökkentlátók számára felbecsülhetetlen segítséget nyújtva, de ezt a készülékegyüttest az elfoglalt emberek is jól kihasználhatják.

A rendszer kidolgozói, Koutny Ilona és Olaszy Gábor jelenleg az Esparol oktatási alkalmazásán dolgoznak. Készülnek eszperantó nyelvet tanító — beszélő — programok, valamint az oktatásban is jól alkalmazható játékok. Az Esparol sikeresen felhasználható még a számítógéppel szedett szövegek próbaolvasásával a szedési hibák kijavítására is.

Az Esparol felhasználása egyszerű a rendszer barátságos volta miatt: közönséges BASIC programokkal lehet alakítani a kívánság szerinti beszédlehetőségeket.

Az Esparol iránt máris komoly érdeklődést tanúsítanak a fejlett tőkés országok is.

Számítógépbontó

Új számítástechnikai üzlet nyílt az Üllői út és a Ferenc körút sarkán átalakított Ápiszban, mégpedig SZÁMALK Böngésző de elnevezéssel. A SZÁMALK ugyanis megvásárolja a selejtezésre szánt R—20, R—30 és R—35-ös számítógépeket, és ezeket a helyszínen szétbontják, a használható alkatrészeket pedig kiárusítják. Az üzlet üzemeltetését, az alkalmazottakat az Ápisz vállalta, a SZÁMALK pedig az áruról gondoskodik. A becslések szerint az éves forgalom mintegy 30 millió forint lesz.

Idegen nyelven

A Linguasoft a sikeres nyelvoktató programjai után az idegen nyelvű levelezést támogató két új szoftverrel jelentkezett. A Levelezés nevű, 1987 novemberében megjelent szoftver a külkereskedelmi levelezést támogatja öt nyelven: a magyaron kívül angolul, németül, franciául és spanyolul, azaz 5×130 ezer karaktert tárol. Az első menüje a következő:

1. Megszólítások
2. Bevezető szövegek
3. Levélszövegek
4. Befejezés
5. Befejező üdvözlés

Ezen belül további menün keresztül juthatunk el a keresett mondatig, melynek gyakran több jelentésvariánsából választhatunk. Tehát magyarul keresünk az adatok között, és ezzel egyidejűleg négy idegen nyelven is találunk. Az induláskor ezermondatos gyűjtemény folyamatosan bővül.

A szintén 1987 novemberében bemutatott Szótár nevű program egy kétnyelvű szövegszerkesztő és egy szótárlapozó program házasítása: a fordítók munkáját megkönnyítendő az éppen futó szövegszerkesztő futásának felfüggesztése alatt megmondja a képernyőn lévő ismeretlen szavak magyar jelentését. Rezidens módon állandóan hívható a magyar szóvégi toldalékok gépelését feleslegessé tevő Ragozógép, a helyes szórend előállítását segítő Szócentrifuga és a kifejezések felismerését segítő Frázisanalizátor. A program használata nemcsak az ismeretlen szavak szótárazását segíti, hanem az ismertek begépelését is fölöslegessé teszi.

Logo+Lego

A Lego cég világsikert aratott építőjátékát már számítógéppel kombinálja. A vezérlésre pedig a gyerekek számára készült, szintén világszerte elterjedt Logo programnyelvet használja. Ennek segítségével a gyerekek maguk programozhatják, hogy az általuk épített autók, emelők, körhinták hogyan mozogjanak. A Logo TC Legónak nevezett játék — bár a szakemberek körében óriási sikert aratott — mégsem fog várhatóan rohamosan elterjedni. Maga a Lego cég is csupán 25 ezer darab értékesítésével számol a következő öt éven belül. Az újdonság ugyanis túlságosan drága: egy kezdő készlet — számítógép és programok nélkül — 450 dollárba kerül. A programokat tartalmazó mágneslemezeket a montreali székhelyű Logo Computer System Inc. hozza forgalomba, egyenként 100 dollárért.

Vonalszélesség

A 16 Mbites tárak készítéséhez csökkenteni kell az integrált áramkörök vonalszélességét. A japán Matsushita cég lézersugárral 0,4 mikrométeres vonalszélességet ért el, de elméletileg nem kizárt a 0,3 mikrométeres vonalszélesség sem; a félvezetőgyártáshoz szükséges maszkokat 248 nanométer hullámhosszú lézersugárral világították meg. Az eljárás előnye, hogy alkalmazható a hagyományos technika és viszonylag egyszerűek a szükséges optikai eszközök.

Pontvadászat

Új rejtvényt indítottunk útjára lapunkban, remélve, hogy elnyeri olvasóink tetszését. Minden számunkban két feladatot közöltünk: az első logikai, matematikai tudást, a második számítástechnikai alapismereteket is igényel.

A feladatok után közöljük az elérhető maximális pontszámot. A részmegoldásokat is pontozzuk.

A pontgyűjtést, vagyis a pontvadászatot az esztendő végén zárjuk. A legjobb tíz versenyző nevét magazinunkban közzé is tesszük, ők lesznek azok, akik könyvutalványt is kapnak.

A helyes megoldások a feladatok közreadása után két lapszámmal később jelennek meg, így a pontvadászoknak jut idejük a gondolkodásra.

Beküldési határidő: 1988. június 20.

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

Jó vadászatot kíván a feladatok összeállítója:
dr. Hoffmann Tibor

1. feladat

Egy különböző nagyságú oldalakkal rendelkező háromszögbe rajzoljunk egy olyan kört, melynek a középpontja az egyik oldalon fekszik és a másik oldalt érinti. *Melyik oldalon* kell feküdnie a középpontnak ahhoz, hogy a kör sugara a lehető legkisebb legyen?

(6 pont)

2. feladat

Egy számítógépes programban egy ciklus végrehajtása úgy történik, hogy valamilyen kiszámított logikai érték dönti el, hogy a ciklusmag végrehajtódik-e, vagy sem.
Vázoljuk fel, hogy milyen lehetőségek vannak a vezérlő logikai érték megvizsgálására a ciklusmag előtt vagy a ciklusmag után. Ezekben az esetekben hogyan hajtódik végre a ciklus?

(4 pont)

Az 1988/3. szám feladatainak megoldása:

1. feladat

Mint ismeretes, az első n egész szám összege úgy számítható ki, hogy az általános i-edik tagot a következőképpen írhatjuk fel:

$$i=\frac{(i+1)^2-i^2-1}{2}$$

Most az első n egész szám összegére a tört számlálójának első két tagja az egymás utáni tagoknál kiejti egymást és így csak a legfelső pozitív és a legalsó negatív tagból marad

$$(n+1)^2-1$$

A tört számlálójának az utolsó tagja összegezve világosan n-et ad. Így

$$\sum_{i=1}^{n} i=\frac{(n+1)^2-1-n}{2}=\frac{n(n+1)}{2}$$

adódik.
Hasonlóképpen a négyzetszámok összegénél abból indulunk ki, hogy,

$$i^2=\frac{(i+1)^3-i^3-3i-1}{3}$$

Felhasználva most az első n egész szám összegére kapott előbbi eredményt, az előzőhöz hasonló gondolatmenettel azt kapjuk, hogy

$$\sum_{i=1}^{n} i^2=\frac{(n+1)^3-1}{3}-\frac{n(n+1)}{2}-\frac{n}{3}=\frac{n(n+1)(2n+1)}{6}$$

Járjunk el hasonlóan a köbszámok összegével, vagyis állítsuk elő a köbszámot a következő alakban:

$$i^3=\frac{(i+1)^4-i^4-6i^2-4i-1}{4}$$

és ekkor felhasználva most már a négyzetek és az első hatványok összegére kapott eredményeinket:

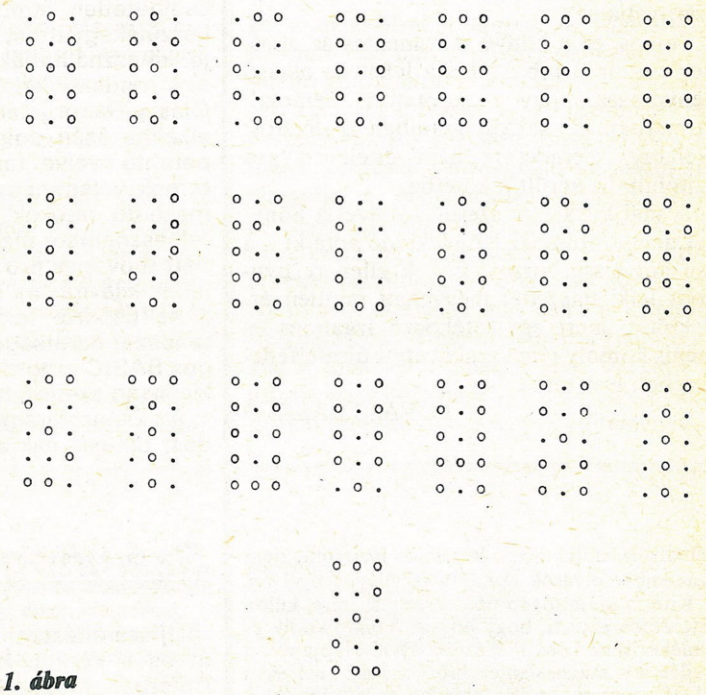
$$\sum_{i=1}^{n} i^3=\frac{(n+1)^4-1-n(n+1)(2n+1)-2n(n+1)-n}{4}=\frac{n^2(n+1)^2}{4}$$

mely szemmel láthatóan egy négyzetszám.

(5 pont)

2. feladat

A 88/2. szám feladványának megoldásában megadott 3×5-ös pontmátrixszal az 1. ábrán látható módon egyértelműen kifejezhető az ábécé sok betűje.



1. ábra

Látható a 0 (nulla) és az O (betű) szétválasztása a 88/4. számban a 2. sz. feladvány ábrája és ezen 1. ábra között, továbbá az 1 és az I betű szétválasztása.
Probléma lép fel a G, M, N és Q betűknél, bár ezek közül az utolsónál megállapodás dolga, hogy hogyan jelezzük a betűt.

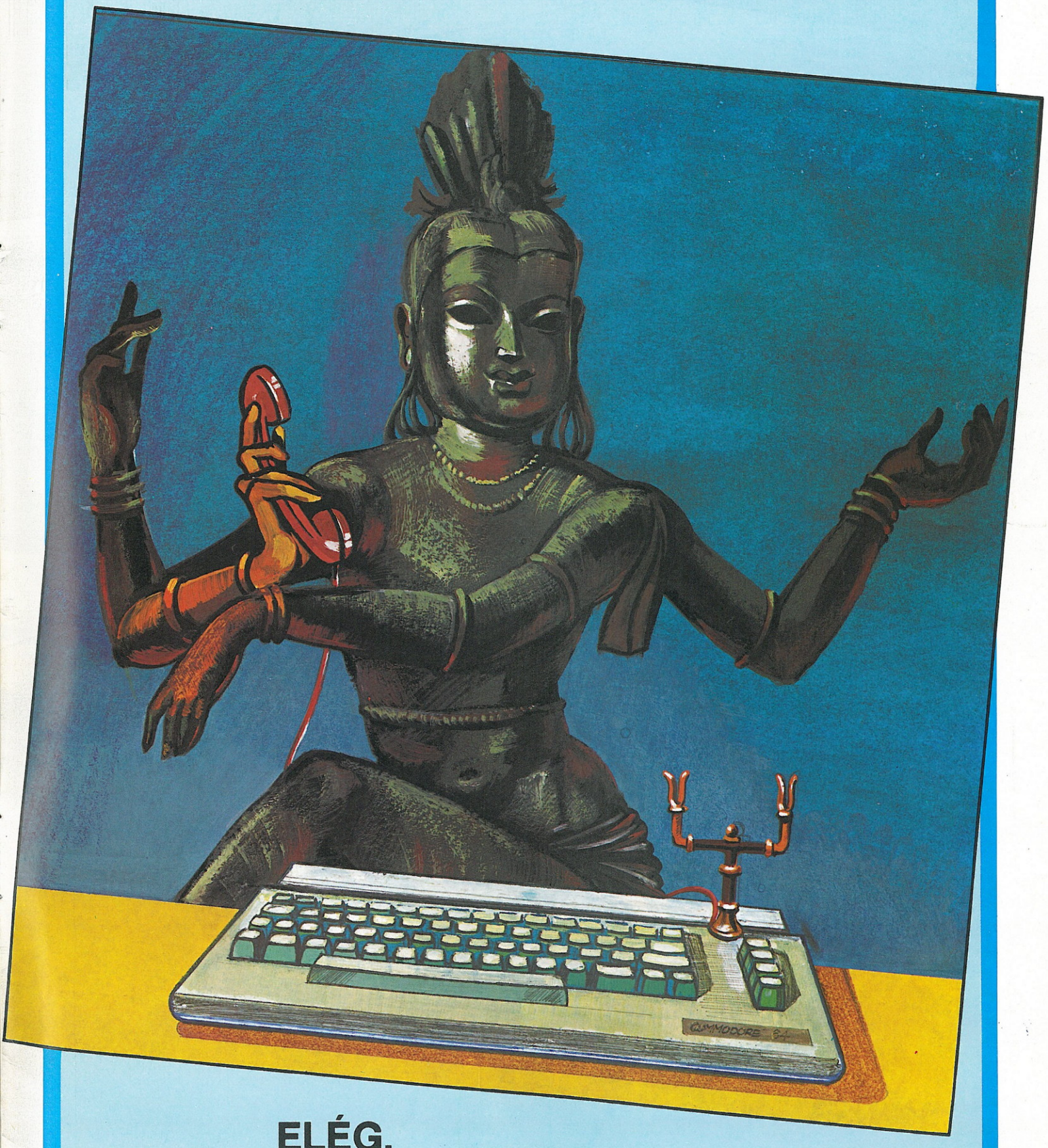
(4 pont)

Ennek a kiküszöbölése csak a betűalak szélesítése lehet. Első lehetőség tehát $n=4$ és emellett változatlanul $m=5$ megadása. Ezzel a kérdéses kritikus betűk a 2. ábra szerinti alakban írhatók fel.

2. ábra

Itt szokatlan az M és N alakjában a középső pont hiánya, de ez az egyértelműséget nem zavarja. A többi betűt az 1. ábrának megfelelően lehet kirajzolni egy-egy oszlop elhagyásával, de további finomítást is lehet elérni a 4. oszlop bevezetésével.

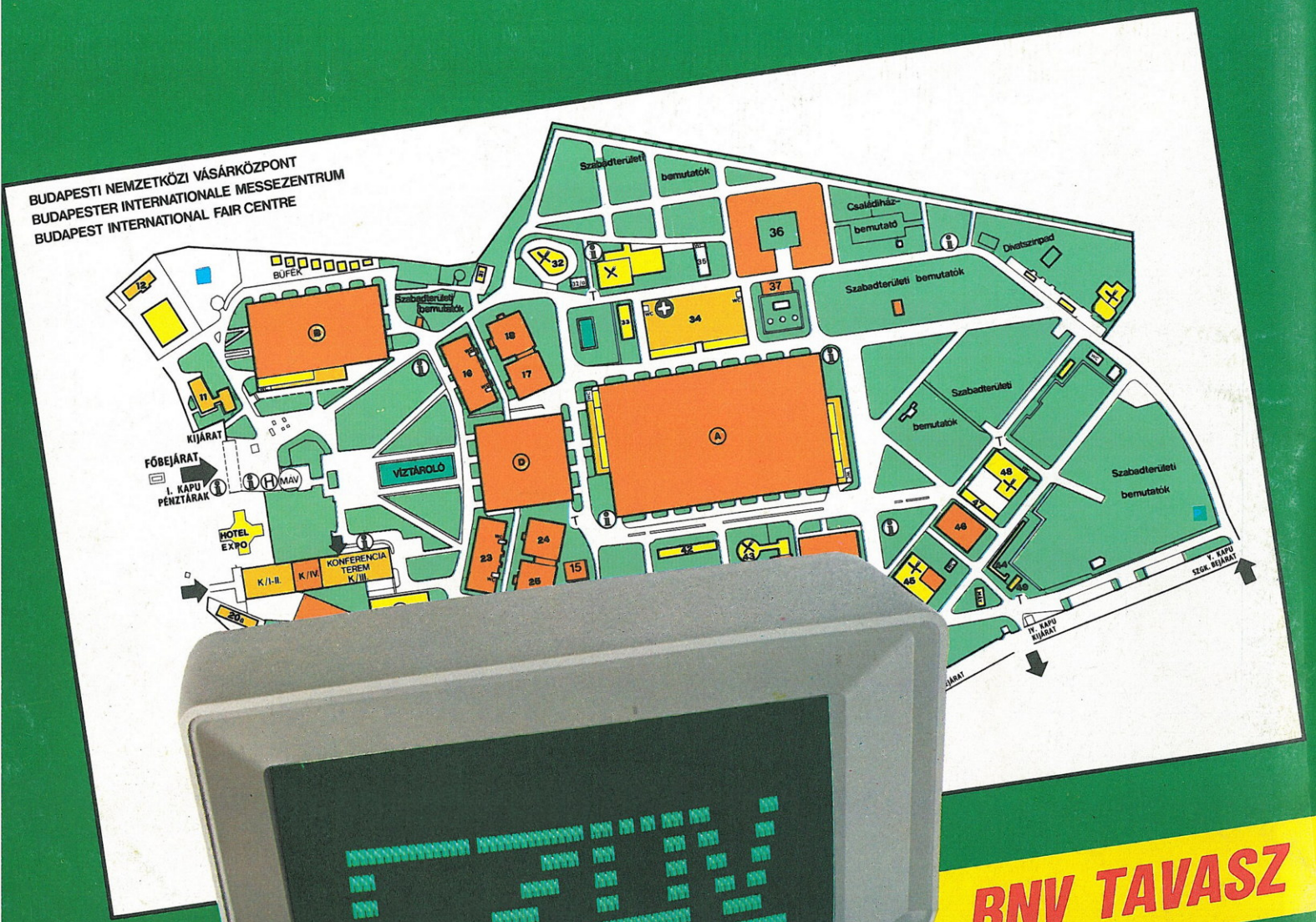
(2 pont)

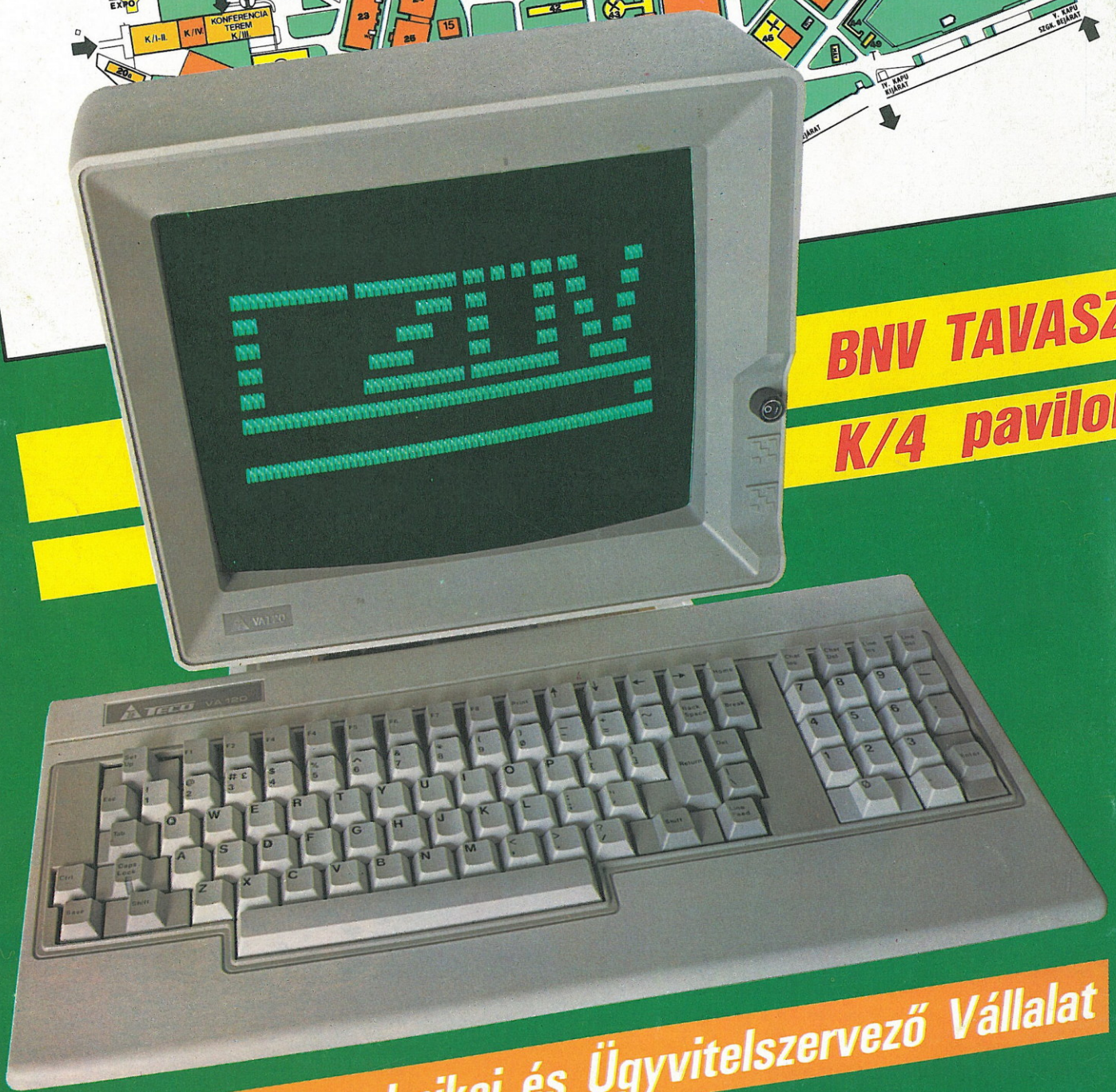


ELÉG, HA KÉT KEZE VAN!

Cikkünk a 42. oldalon

BUDAPESTI NEMZETKÖZI VÁSÁRKÖZPONT
BUDAPESTER INTERNATIONALE MESSEZENTRUM
BUDAPEST INTERNATIONAL FAIR CENTRE





BNV TAVASZ

K/4 pavilon

KSH Számítástechnikai és Ügyvitelszervező Vállalat