

# mikro

számítógép

# magazin

Ára: 30 Ft





Just for your  
TANDY  
COLOR COMPUTER

June 1986

Canada \$4.95

U.S. \$3.95

# The RAINBOW<sup>®</sup>

THE COLOR COMPUTER MONTHLY MAGAZINE

## Electronic Euphony

### Listen!

Use your CoCo as a synthesizer  
Add exciting sound effects for programs

### Compose!

CoCo Instant Music with PLAY strings  
Print your own sheet music paper

### Entertain!

Experience CoCoTV  
Play 'Name that Tune'

### And Marty!

MS-DOS to CoCo Conversion  
EPROM Erasure Technique  
Delphi Database Report



### Plus

Tracking down more disk space, swimming pool maintenance, from dancing telephones to the Castle of Doom, and 21 new product reviews.

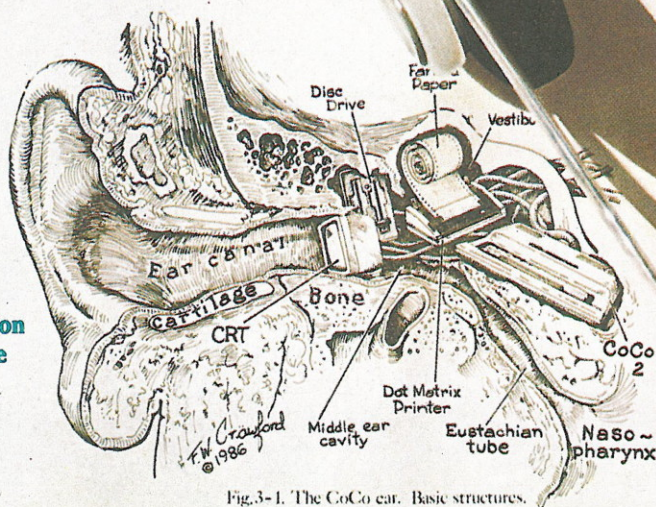


Fig.3-1. The CoCo ear. Basic structures.

állított gép lapja. Bár ez a gép — és a vele „rokon” Dragon — itthon méltatlanul alig ismert, a cikkcserének nemcsak az a jelentősége, hogy az első, számítástechnikai lapok közötti amerikai—magyar kapcsolatot. Az amerikai lapban sok gépfüggetlen anyag is megjelenik. Ezek átvétele természetesen már közvetlenül hasznot is jelent számunkra.

Az amerikai lap rövid történetét a The RAINBOW<sup>®</sup> 1986. július, ünnepi számában, a tulajdonos-főszerkesztő Lonnie Falk cikkében olvashatjuk. A lapot öt éve alapították. Kezdetben egy másfél négyzetméter területű asztallap volt a „szerkesztőség”. Az alapító egyetlen alkalmazottal dolgozott, aki jelenleg a lap üzleti vezetője, a cikkcsere fő támogatója: Patricia H. Hirsch. Az akkori lapterjedelm 2 oldal volt.

Jelenleg a szerkesztőség saját, 2000 m<sup>2</sup> alapterületű, háromszintes épületében működik, 73 alkalmazottal. A lap 244 oldalas, sokszínű. A tulajdonos számos már folyóiratot is kiad: például a 80 ezres példányszámú VCR Magazine-t a videokazetta-használóknak és a PCM címűt az IBM PC használóknak. A The RAINBOW<sup>®</sup>-nek kazettamelléklete is van, melyben a lap fontosabb programjait értékesítik, mintegy havi 8000 példányban.

A következőkben rendszeresen közölni fogjuk az átvett cikkeket.

# Előttünk a „Szivárvány”

Egyik munkatársunknak amerikai útja során alkalma nyílt az egyik szaklap szerkesztőségébe is ellátogatni. A baráti beszélgetés szakmai sikere, hogy a The RAINBOW (Szivárvány) c. amerikai szaklap és lapunk között cikkcsere indul.

A The RAINBOW<sup>®</sup> a Tandy (a HT—1080Z alapját képező TRS 80 Model I gyártója) által gyártott, TRS 80 Color elnevezésű, milliós példányban elő-



## A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

**A kiadvány  
a Tudományszervezési  
és Informatikai  
Intézettel  
együttműködve készül**

**A szerkesztőbizottság  
vezetője:  
Kovács Győző**

**E számunkat  
szerkesztették:**

**Bakos Tamás  
(programozástechnika)**

**Broczkó Péter  
(hírek)**

**Kovács Győző  
(levelezés)**

**Lindner László  
(sakkprogramozás)**

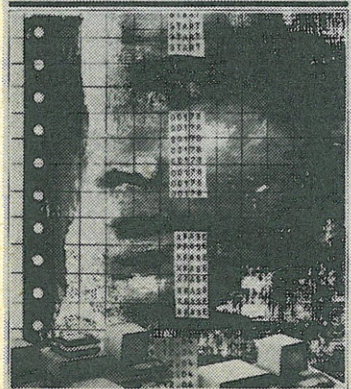
**Petrőczy Judit  
(könyvek)**

**Simonyi Endre  
(klub)**

**Varga András  
(iskola-számítógép)**

**Címképünk:  
Ramocsal Imri munkája**

 mikro számítógép  
magazin



**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. Petrus György  
igazgató**

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

**Terjeszti a Magyar Posta  
Előfizethető a hírlapkezelési  
hivataloknál  
és a Posta Hírlapelőfizetési  
és Lapellátási Irodáján  
(1900 Budapest V.,  
József nádor tér 1.)  
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.  
86-253**



**Szakra Lapnyomda  
Budapest (86-4622)  
Felelős vezető:  
Csöndes Zoltán vezérigazgató**

**INDEX: 25 629  
ISSN 0236-6088**

### Tartalom

Beszélgetések	2
Tájékoztató az 1987. évi	
Nemes Tihamér Tanulmányi Versenyről	6
Számítástechnika a fiataloknak	7
Mit tud a PROLOG?	9
Adok-veszek-cserélek	21
A PSION VU—3D rajzolóprogram	27
A kiszolgáltatott ember	35
Helyi hálózatok	36
Fantázia és tudomány?	38
Barátunk, a számítógép	40
A világ legnagyobb show-ja	48

### ISKOLA — SZÁMÍTÓGÉP

Az ismeretlen C16	3
Spectrum az asztalban	4
Térkép	4
Középiskolai Számítástechnikai Konferencia	5

### DIÁKROVAT

Credit	8
--------	---

### PROGRAMOZÁSTECHNIKA

Z80 programozási gyakorlatok	12
BASIC és gépi kód	14
Adalék a rendezéshez	15
Bináris keresőalgoritmus C64-re	16
Papírszélesztés	17
A VIC—20 tárcímei	19

### μPROGRAMOK

A Beta basic és alkalmazása	22
Manó a Spectrumon	24
Új karakterek	25

### SAKKPROGRAMOZÁS

Bitek és figurák	33
Gépi ellenfeleink	34

### FÓRUM

Rendellenességek és rendetlenkedések	43
--------------------------------------	----

### AZ OLVASÓ ÍRJA

PIAC	45
------	----

KÖNYVEK	46
---------	----

HÍREK-ÉRDEKESSEGEK	47
--------------------	----



„Kora este a padon  
Ülök künn a bástyán,  
Tünődöm a csillagok  
Néma fordulásán;  
Kaszáscsillag, az öreg  
A nyugati szélen  
Éppen nyugszik; eleget  
Ragyogott a télen.”

(Tóth Árpád: Kaszáscsillag)

Ülök az MTESZ közgyűlésén és hallgatom az előadókat, fel-felkapom a fejem, ha egy ismerős, de még inkább ha olyan hozzászóló lép a dobogóra, akinek mondánivalója is van, és azt érdekesen, a figyelmet felkeltve tudja előadni.

Nincs szándékomban a közgyűlésről késsői beszámolót írni, akit érdekelt, az a napilapokban, az MTESZ anyagokban elolvashatta. Egy öröközőld témáról írni, mert amióta csak az MTESZ-ben dolgozom, azóta állandóan visszatérő gondja minden egyesületnek a hazai és nemzetközi konferenciák rendezése. Erről ezen a közgyűlésen is legalább két-három hozzászóló beszélt.

Arról van szó, hogy egy 4-5 napos tanácskozást ma már 2-4000 Ft körüli részvételi díj alatt gyakorlatilag nem lehet megrendezni (és ehhez jön a szállás meg az étkezés), hiszen a terembérlési díjak, a tolmácsberendezések, a tolmácsok, a sokszorosítás és a nyomda, de még a postai díjak emelkedése is alacsonyabb részvételi költség mellett a konferenciákat biztonságosabbá teszi. A helyzetet csak súlyosbította, hogy a központi apparátus költségeinek fedezésére az MTESZ minden konferenciát meg is „adózta”, ráteve a költségekre bizonyos százalékot, amelyet azután az egyesülettől elvon. Egyáltalán nem kell csodálkozni, hogy ezt a gyakorlatot az egyesületek kritizálják, hiszen ez az „adó” a jól dolgozó, sok konferenciát rendező egyesületeket — így az NJSZT-t is — sújtja, míg a kevésbé aktívakat, a főleg dotációból élőket szinte támogatja.

De nem erről szeretnék beszélni, inkább az érem másik oldaláról, amelyről a közgyűlésen nem hivatalosan esett szó. Arról ugyanis, hogy a sok-sok nehézség és gond, költségvetési bakugrások árán megrendezett konferenciák hasznát milyen mértékben élvezzük, és kiaknázzuk-e azokat a lehetőségeket, amelyek egy-egy nemzetközi konferenciában rejlenek.

Külföldi konferenciára járni jó, sokkal jobb, mint Pestre, Szolnokra, Szombathelyre vagy Szegedre utazni egy-egy nemzetközi találkozóra, mert világot lehet látni, érdekes emberekkel lehet beszélgetni, egy-egy különleges színház vagy mozi színesítheti az utazást, és persze venni is lehet néhány olyan dolgot, ami itthon nem, vagy csak nagyon drágán kapható. A SICOB, a Hannoveri Vásár, a müncheni Systems, a Lipcsei Vásár vagy pedig a bécsi IFABO lassan már magyar házi rendezvénynek is számít, és ez így van jól, ha többet szerepelünk a világban, ha többet látunk a világból, ha többet hasznosítunk az ott tapasztaltakból, gyorsabban haladunk, előbbre jutunk.

Legalább ilyennek, ha nem hasznosabbnak tartom az itthon rendezett nemzetközi konferenciákat is, amelyek — ha sikerül elegendő és színvonalas előadást és előadót szervezni — a világot hozzák el hozzánk: azoknak is megteremtik a szakma „hírességeivel” a kapcsolat lehetőségét, akiknek

# Beszélgetések

nyelvi vagy sokszor anyagi problémák miatt a külföldi konferenciákon vagy kiállításokon való részvétel lehetősége nem adatik meg.

Mondom, erről is beszélgettünk az MTESZ-közgyűlés köreiben, azután egy hét múlva folytattuk a Teleteaching '86 nemzetközi konferencián. Azt nem értettük, ha idejön Budapestre — házhoz hozva a nemzetközi tapasztalatokat — 20, 50, 100 külföldi szakember, akiknek meg lehet hallgatni az előadását, akikkel a szünetben, az esti találkozókon, a kerekasztal-beszélgetéseken össze lehet ülni, akiket ki lehet kérdezni, akiket egy-egy intézetbe meg lehet hívni, hogy az ottani munkáról véleményt mondjanak, ezt miért nem használjuk ki eléggé. Ráadásul ez még pénzbe sem kerül, sőt ők fizetnek azért, hogy eljöhesse nek.

Félreértés ne essék, a Teleteaching '86 konferencia jó volt, és ami a résztvevőket illeti, optimálisnak volt mondható; az előadásokon, bemutatókon ugyanis cca 80-100, sőt az első napon 150 hallgató vett részt. Így nagyon kellemes és alkotó légkörben tudtunk beszélgetni az oktatástechnológiáról a körülbelül húsz országból érkezett szakemberekkel. Jók voltak az előadások, és többségükben jók az előadók is, ezért majdnem mindegyik után vita alakult ki. Az előadásokat magyarra fordítottuk, hogy azok a hazai szakemberek is eljöhessenek és megértsék, akik esetleg nyelvi problémákkal küzdenek. Ennek ellenére a hazai hallgatók gyakorlatilag nem vettek részt a vitában, alig kérdeztek, az embernek órákon keresztül azt volt a benyomása — miután inkább a külföldiek vitakoztak —, hogy nem egy budapesti, hanem egy párizsi vagy müncheni konferencián üldögél.

Amikor a konferencia rendezésébe kezdtem, azt hittem, hogy egy ilyen, nagyon sok újdonságot bemutató téma, mint a távoktatás, vagy — ahogyan a résztvevők a folytatásról szavaztak — távtanulás (distance learning) az oktatók, az oktatástechnikusok százait fogja csábítani, és a kb. 200 fős előadóterem kicsinek fog bizonyulni. Már a kezdet kezdetén úgy terveztük, hogy az anyagilag gyengén „eleresztett” oktatókat, illetve tanulókat kedvezményesen (1000 illetve 400 Ft), majd az utolsó előtti pillanatban meghirdetve, fizetés nélkül is beengedjük. Alig jöttek. Kiderült, hogy ebben néhány iskola vezetése is ludas, ahol pl. nem engedték el a tanárokat, mondván, hogy akkor ki tartja meg helyettük az órákat. Persze voltak olyan iskolák, ahonnan eljöhetek kikérésre vagy anélkül is. Én azt hiszem, hogy nem elegendő, pedig az előadások mellett egy — előre nem szervezett — kiállítás is összejött: néhány alig ismert hazai és a legmodernebb külföldi (videolemez) oktatástechnikai eszközökből állt a bemutató.

Ha nem beszéltem volna a Teleteaching '86 előtt más egyesületek vezetőivel, akiknek a nemzetközi konferenciákkal kapcsolatban hasonló tapasztalataik voltak, akkor azt hittem volna, hogy itt az informatikával és különösen az oktatással kapcsolatos jelenségről van szó.

Azt kell mondanom, hogy a probléma súlyosabb: valószínűleg nem tudjuk megbecsülni és kihasználni azokat a lehetőségeket, amelyeket nagyon sok munka és fáradság árán az egyesületek végül is biztosítanak. Ezek a konferenciák és találkozók néhány ember ügyévé szegényednek, pedig az egyesületek, így a Neumann Társaság is mindent megtesz annak érdekében, hogy az érdekelteknek időben tudomására jusson a tapasztalatcsere — a tanulás és egyben a tanítás — lehetősége.

A Teleteaching '86 előtt, amikor a hazai előadókat szerveztem, nem egy barátom azzal utasította el a részvételt, hogy „mit keresnek én ott a sok külföldi sztár-előadó között azzal a szerény kis eredménnyel, amit be tudnék mutatni...”. Szerencsére nem mindenki gondolkodott így, és a hazai „szerény” eredmények külföldi visszhangja azt bizonyította, hogy a mi előadásainkban is voltak olyan ötletek és eredmények, amelyekre bizony oda kellett figyelniük.

Kialakult az a véleményem is — egyoldalú információk alapján —, hogy a munkahelyek (és most itt az oktatási intézményekre gondolok elsősorban; ezért erdemes a jelenségre odafigyelni) a nem szervezett továbbképzést, tehát ahol nem „beiskolázásról” van szó, nem használják ki megfelelően. Én azt hittem, hogy az oktatásért felelős „főnökök” a részvételt az oktatástechnológiáért felelős szakembereknek kötelezővé teszik, majd a végén beszámoltatják őket, hogy az adott területen a hallottakból mit lehet hasznosítani.

Egy másik érdekes véleményt is hallottam, nevezetesen azt, hogy az embereknek általában nincs idejük konferenciára járni. A Teleteaching '86-on az előadások hét félnapon keresztül tartottak — gyakorlatilag reggel 9-től este 7-ig. „Aki a munkahelyén is helyt akar állni — mondják —, a gmkb-an vagy PJT-ben is keresni akar, annak a hét félnap megengedhetetlen kiesés.” A vélemény mögött sajnos több van, mint az az első pillanatban látszik, t. i. itt most arról van szó, hogy a napi munka mellett gyakorlatilag nincs, vagy egyre kevesebb idő jut az ismeretek felrészítésére, új ismeretek megszerzésére. Azt hiszem, felesleges lenne részleteznem az ebből a helyzetből keletkező problémákat, hogy hova jut az a szakember, akinek nem jut ideje a szakmai továbbképzésre. Ez már nemcsak az egyén, hanem a társadalom problémája is. Az innováció ma divatos fogalom, amelyről nem beszélni kell, hanem tenni is kell érte valamilyen körülményeket kell tehát teremteni, ami a szakembereket nemcsak arra kényszeríti, hogy dolgozzanak, hanem arra is, hogy fogékonyak legyenek az újra, tanuljanak. A munka eredményének elbírálásánál legyen követelmény annak újdonságtartalma, és hogy megfeleljen a mai minőségi igényeknek és technológiai előírásoknak. Ki kell alakítanunk a mai kor szakembermodelljét, aki rendszeresen tanul, képezi magát, minden újra fogékony, és a napi munkájában folyamatosan hasznosítani képes a tanultakat, természetesen a konferenciákon tanultakat is.

KOVÁCS GYÖZŐ



## Az ismeretlen C16 KERNAL rutinok 4.

A KERNAL rutinok ismertetésének befejező részét egy helyreigazítással kezdem. A VECTOR rutin leírásában, amely a szeptemberi számban jelent meg, kétszer is tévesen 16 bájtos területre hivatkoztam 32 bájtos helyett. Ezen a területen ugyanis 16 címet kell elhelyezni, ezek mindegyike két-bájtos. A hibáért elnézést kérek.

Az itt következő hét rutint eddig nem használtam, működésüket csak a leírás miatt tanulmányoztam. Sajnos a SETTMO esetében a rendelkezésemre álló irodalom ellentmondásai miatt csak sejtem a valóságot. Remélem, hogy akinek szüksége van ennek a rutinnak a használatára, hozzá tud jutni a szükséges pontos információhoz.

### CINT \$FF81 65409

Alaphelyzetbe állítja a képernyős szerkesztőt és a videoregisztereket. (A VC20-on ez a KERNAL rutin hiányzik.)

### IOINIT \$FF84 65412

Alaphelyzetbe állítja a beviteli/kiviteli rendszert. (A VC20-on ez a KERNAL rutin hiányzik.)

### SETMSG \$FF90 65424

Ezt a rutint legtöbbször az operációs rendszer rutinjai használják egy-egy képernyőre írandó üzenet kiadása előtt. Egy jelzőt állít be aszerint, hogy hibaüzenetről

vagy egyéb információról van szó. A kívánt értéket a hívás előtt az A regiszterbe kell tölteni, hibaüzenet esetén \$80-at, egyébként \$40-et. Az a rendszerváltó, amelybe a rutin az A regiszter tartalmát leteszi, egyúttal a futási mód jelzője is. Ennek negatív értéke esetén, vagyis hibaüzenet kiadása után, a BASIC interpreter közvetlen módba megy át.

### MEMTOP \$FF99 65433

Az átviteljelző (C) bit értékétől függően beállítja vagy kiolvassa a BASIC által használható RAM terület felső határára mutató változó tartalmát. C=1 esetén a mutató tartalma az X/Y regiszterpárba kerül, ha C=0, a regiszterek tartalma töltődik a mutatóba. Az X regiszterhez tartozik a változó alacsonyabb helyiértékű bájtja, Y-hoz a magasabb helyiértékű.

### MEMBOT \$FF9C 65436

Szerepe megegyezik a MEMTOP-éval, de a BASIC RAM alsó határára mutató változó értékét állítja, illetve olvassa. A BASIC interpreter inicializálásakor a MEMTOP és MEMBOT által kezelt változókba kerülnek az információk a megfelelő nullalapos mutatókba.

### SETTMO \$FFA2 65442

Ezzel a rutinnal kapcsolatosan a forrás-

munkámban teljes a káosz. (Zárójelben jegyzem meg, hogy ezek többsége a C64-re vonatkozik, de KERNAL rutinokról lévén szó, az erre vonatkozó adatoknak C16-ra is érvényeseknek kell lenniük.)

A rutin nem csinál mást, mint az A regiszter tartalmát elhelyezi egy rendszerváltóba, melynek neve: time-out flag, szabad fordításban időtúllépés-jelző. Hívása előtt a kívánt értéket az A regiszterbe kell tölteni. A kapcsoló állapotát a 7. (bal szélső) bit jelzi. Azt, hogy hogyan, csak sejtem, ugyanis az erre vonatkozó adatok is ellentmondanak egymásnak.

Dr. Úry László Commodore 64 című könyvének II. kötetében a következő olvasható: „Ha a rutin meghívásához az A regiszter 7. bitje magas, a time-out jelző beállítódik. Ha a bit alacsony, a jelző kikapcsolódik.” Ezzel szemben Erdős Iván Commodore 64 Assembly című könyvében a következőt találtam: „... a 7. bit magas — törlés ... alacsony — beállítás”. (Mindkét könyv az LSI ATSZ kiadványa, az utóbbinak a lektora dr. Úry László.) Ezekon kívül még egy — VC20-ról szóló — könyvben találtam a jelző beállításának mikéntjével, az szerint az alacsony érték jelenti a bekapcsolt állapotot. A leírások többsége a time-out kapcsolót a soros busszal hozza kapcsolatba, három helyen — a fentebb idézett két könyvben és egy német nyelvű folyóiratban — az IEEE buszra való utalást találtam. Tapasztalataim alapján én is az utóbira tippelek.

### IOBASE \$FFF3 65523

A beviteli/kiviteli modulok kezdőcímét tölti az X/Y regiszterpárba, alacsony/magas helyiértékű bájtrendszerben. Ez egy géptípustól függő állandó a C16-on \$FD00. (VC20-on: \$9110, C64-en: \$DC00.)

BARNA LÁSZLÓ

## C16 orgona

Ez az egyszerű kis zenei program orgona jellegű, folyamatos és változó időtartamú hanghatásokat vezérel billentyűzet segítségével.

Az adott billentyű lenyomásakor a megfelelő hang megszólal, és addig szól, amíg egy másik hangbillentyűt le nem nyomunk. Zenei szünetet a szóközbillentyűvel lehet létrehozni, de a hangot elhallgattatja a még fel nem használt billentyűk lenyomása is. A program tartalmaz egy védelmet: hibaüzenet vagy meggondolatlan BREAK esetén az újraindítás után a STOP? kérdésre adott I válasz inicializálja a hanggenerátort. Egyszerre két hangbillentyűt lenyomva érdekes hanghatások jönnek létre, bár a „hangszer” monofonikus.

A négyoktávnyi egész hangok kibővíthetők a félhangokkal is: erre a billentyűzet lehetőséget ad (4x12 db), csak az alsó sorban a SHIFT-et kell átlépni. Így azonban a kezelés lényegesen bonyolultabbá válik.

GULYÁS ANDRÁS

```

5 PRINT "XXXXXXXXXXXXXXXXXXXX ORGONA !!!"
10 PRINT "XXXXXXXX 1 - 8"
20 PRINT "XXXXXX - I"
30 PRINT "XXXXXX - K"
40 PRINT "XXXXXX - , "
50 PRINT "XXXXXXXXXXXXXXXXXXXX BILLYENTYUKKEL LEHET ZENELNI"
51 PRINT "XXXXXX SZOKOZ SZUNETET AD"
52 PRINT "XXXXXXXXXXXXXXXXXXXX BEFEJEZES: RUN/STOP"
53 PRINT "XXXXXX EZUTAN SZOL A HANG, UJRA"
54 PRINT "XXXXXXXXXXXXXXXXXXXX INDITASSAL A STOP?-RA ADOTT"
55 PRINT "XXXXXX I' ELHALLGATTATJA"
56 PRINT "XXXXXXXXXXXXXXXXXXXX SZORAKOZAST!"
60 VOL 6
70 DATA 596,0,0,0,0
80 DATA 917,929,939,944,953,961,967,5,971,5,0,0,0,0,0,0,0
90 DATA 596,453,345,685,854,704,739,770,917,798,810,0,571,516,0,0,810,864
100 DATA 643,881,911,383,834,262,897,169
110 DIM N(47): FOR I=1 TO 47: READ N(I): NEXT I
120 PRINT "XXXXXXXXXXXXXXXXXXXX STOP?": GETKEY W$: IF W$="I" THEN GOSUB 220
130 PRINT "XXXXXXXXXXXXXXXXXXXX LEHET ZENELNI!"
140 GETKEY A$: X=ASC(A$)
150 IF X<44 OR X>91 THEN GOTO 140
160 B=ASC(A$): K=B-43
170 IF N(K)<>0 THEN SOUND 1,N(K),60000
180 DO: GETKEY A$: X=ASC(A$)
190 IF X<44 OR X>91 THEN SOUND 1,N(K),0:GOTO 140
200 LOOP WHILE ASC(A$)=B
210 SOUND 1,N(K),0: B=ASC(A$): K=B-43: GOTO 170
220 FOR L=1 TO 47: SOUND 1,N(L),0:NEXT L
230 RETURN

```

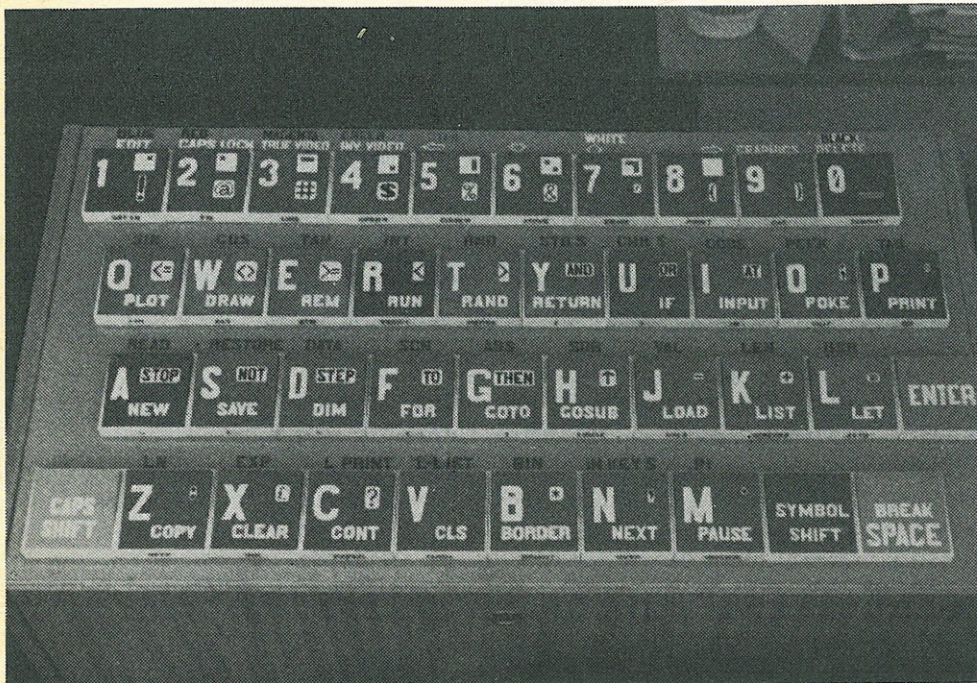


# Spectrum az asztalban

A Pécs város másik végén tanító fiatal gimnáziumi tanárt, Sebestyén Zoltánt a lakása szomszédságában lévő általános iskola hívta tanítani. Évekig gondolkozott, aztán megunva a közlekedésre pocskolt sok időt, komolyan foglalkozni kezdett az ajánlattal. Matematikatanár lévén úgy döntött, hogy ha vesz az iskola egy számítógépet, átjön. Az igazgató azonnal megvett egy ZX-Spectrumot, pedig akkor még 40 ezer forintba került. Így aztán Sebestyén Zoltán is általános iskolai tanár lett a szomszédban lévő iskolában. De ettől kezdve csak este 10 óra után látta a családját: minden este a számítógépet vigyázta a gyerekek kezében. Elzárni nem volt szíve, sőt inkább a sajátját is bevitte mindig, s közben egyfolytában azon töprengött: hogyan tudna kiszabadulni ebből a saját maga ásta csapdából, s visszanyerni az esti szabadságát.

Ekkor villant fel benne az ötlet, hogy a ZX-Spectrum billentyűzetét kivezeti egy asztallapra. Az ötletet tett követte, s egy gyermekíróasztal felületére szerelt ajtócsengő-kapcsolókkal elkészítette az új Spectrum-billentyűzetet. A gépet magát a biztonsági zárral ellátott fiókba rakta, a billentyűzet és a gép közötti kapcsolatot pedig a gép buszcsatlakozójára ráhúzható illetstőlvel biztosította. Ebben mérnök barátja nyújtott segítséget.

Az asztalba épített ZX-Spectrum billentyűzet



Azóta a Spectrum az iskola folyosóján áll. Reggel fél nyolckor bekapcsolják, s este, az iskolából utoljára távozó tanár kapcsolja csak ki. Délelőtt, a tanítási órák alatt a gép a szülőké; a gyerekeknek kihirdették, hogy küldjék be a számítástechnika iránt érdeklődő szüleit. A szünetekben persze, ha üres a gép, a gyerekek versenyt rohanak, hogy ki gépeljen be egy pár perc alatt is beírható programot. Délután már a különböző szakkörök használják a gépet, előre ütemezett módon. Este ismét kötetlenül a gyerekeké: előre feliratkozhatnak, hogy ki mikorra igényli.

A nagy billentyűzet újszerű géphasználatot vezetett be. Játékoknál hárman-négyen is hozzáférnek a géphez, s „munkamegosztás”-ban nyomogatják a billentyűket, miközben igazi csapatszellem: egy mindenkire — mindenki egyért alakul ki köztük. Ennek a megoldásnak a másik új alkalmazása a gép billentyűzetének falra akaszthatósága: a falon táblaként elhelyezkedő billentyűzet az egész osztály számára jól látható, és remekül segíti az oktatást.

A mintegy tízezer forintba kerülő billentyűzet iránt érdeklődők megkereshetik az alkotót, aki megfelelő igény esetén vállalkozik további billentyűzetek készítésére. Címe: Sebestyén Zoltán, 7632 Pécs, Testvérvárosok tere, általános iskola. Telefon: (72) 10-993.

## PRIMO

### Térkép

A program segítségével játékosan gyakorolhatjuk a földrajzot. A következő feladatokat menüből választhatjuk ki.

1. *A megjelölt pont felismerése.* A térképen egy földrajzi pontot takar el egy négyzet. Ennek a pontnak a nevét kell válaszként beírni. A gép számolja és kiírja a feladványokat, valamint a sikeres megfejtéseket.

2. *Utazás hibajelzéssel.* A gép a rendelkezésre álló adathalmazból kiválaszt egy úticélt, amelyhez az É, a D, a K és az N betűk segítségével el kell vezetni a képernyő bal alsó részén lévő pontot. A pont sebessége a SHIFT gombbal állítható. A képernyő jobb felső részén felirat jelzi, hogy közeledünk-e a célhoz vagy távolodunk tőle. A gép számolja és kiírja a távolodások számát.

```

1 REM TERKEP
9 U$=""
10 RESTORE : CLS : PRINT : PRINT
  "MI A FELADAT?":PRINT
12 PRINT"1 A MEGJELOLT PONT FELISMERESE"
13 PRINT"2 UTAZAS HIBAJELZESSEL"
14 PRINT"3 UTAZAS HIBAJELZES NELKUL"
15 PRINT"4 ELLENORZES"
19 PRINT"5 BEFEJEZTUK" : PRINT
20 INPUT L : CLS : IF L>5 OR L<1 THEN10
30 ON L GOTO 7300,7350,7350,7000,40
40 CLS : END
100 REM RAJZOLO .....
109 FOR K=1 TO 7
110 READY1,X1 : Y1=Y1/2 : X1=X1/2
112 READY2,X2 : Y2=Y2/2 : X2=X2/2
118 IF X2=0 THEN 130
119 GOSUB 990
122 X1=X2 : Y1=Y2
128 GOTO 112
130 NEXT K
132 RETURN
149 REM IDE KELL IRNI A HATAR, A DUNA,
290 REM A TISZA ES A BALATON KIRAJZOLA-
310 REM SAHOZ SZUKSEGES ADATOKAT
990 REM KET PONTON ATMEND EGYENES .....
1003 IF X2=X1 THEN LR=1 : GOTO 1020
1004 Z=(Y2-Y1)/(X2-X1)
1005 IF Y2=Y1 OR ABS(Z)<1 THEN LR=1
  ELSE LR=1/ABS(Z)
1007 IF X1>X2 THEN 1010
1008 LP=1 : GOTO 1012
1010 LP=-1
1012 Q=X1 : W=X2
1013 FOR X=Q TO W STEP LP*LR
1014 Y=Z*(X-X1)+Y1
1017 SET (X,Y)
1018 NEXT X
1019 GOTO 1032
1020 IF Y1>Y2 THEN 1024
1022 LP=1 : GOTO 1025
1024 LP=-1
1025 Q=Y1 : W=Y2
1026 FOR Y=Q TO W STEP LP*LR
1029 SET (X1,Y)
1030 NEXT Y
1032 RETURN
5000 REM A PONT VEZETESE.....
5002 QQ$=INKEY$ : IFQQ$<>"" QQ=ASC(QQ$)
5003 IF QQ$="" THEN 5002
5004 IF QQ$="E" THEN Y=Y+10
5005 IF QQ$="D" THEN Y=Y-10
5006 IF QQ$="N" THEN X=X+10
5007 IF QQ$="K" THEN X=X-10
5008 IF QQ$="E" THEN X=X-10
5009 IF QQ$="D" THEN X=X-10
5010 IF QQ$="N" THEN X=X-10
5011 IF QQ$="E" THEN Y=Y+10
5012 IF X>210 THEN X=210
5013 IF X<1 THEN X=1
    
```



## Középiskolai Számítástechnikai Konferencia

Fonyód,  
1986. augusztus

```

5014 IF Y>132 THEN Y=132
5015 IF Y<1 THEN Y=1
5016 IF (X=X9 AND Y=Y9) THEN 5002
5018 IE. POINT(X9,Y9)=0 AND X9>0 THEN
  SET(X9,Y9) ELSE RESET(X9,Y9)
5020 IF POINT(X,Y)=0 THEN
  SET(X,Y) ELSE RESET(X,Y)
5022 X9=X : Y9=Y : Q0=0
5113 RETURN
6000 REM ADATKEZELO .....
6005 K=RND(9) : V=RND(10)
6007 ON K GOTO 6011,6012,6013,6014,6015,
  6016,6017,6018,6019
6011 RESTORE 6100 : GOTO 6030
6012 RESTORE 6200 : GOTO 6030
6013 RESTORE 6300 : GOTO 6030
6014 RESTORE 6400 : GOTO 6030
6015 RESTORE 6500 : GOTO 6030
6016 RESTORE 6600 : GOTO 6030
6017 RESTORE 6700 : GOTO 6030
6018 RESTORE 6800 : GOTO 6030
6019 RESTORE 6900 : GOTO 6030
6030 FOR K=1 TO V
6032 READ Q0$,Y,X : Y=Y/2 : X=X/2
6040 NEXT K
6045 RETURN
6100 DATA GYOR,183,97
6200 DATA KOMLO,50,136
6300 DATA SZEKESFEHVAR,138,144
6400 DATA ESZTERGOM,191,164
6500 DATA PANNONHALMA,169,116
6600 DATA KOSZEG,158,32
6700 DATA BZD,230,259
6800 DATA EGER,201,266
6900 DATA BERETTQUJFALU,140,337
6910 DATA 0,0,0
7000 REM ELLENORZES .....
7001 GOSUB 7550
7002 GOSUB 100
7003 RESTORE 6100
7004 READ N$,Y,X
7005 X=X/2 : Y=Y/2
7006 IF N$="0" THEN GOTO 10
7007 PRINT$0,0,U$;U$;U$
7008 PRINT$0,0,N$
7009 GOSUB 7100
7030 Q0$=INKEY$ : IF Q0$="" THEN 7030
7038 GOSUB 7100
7044 GOTO 7004
7100 REM JELELO .....
7109 FOR I=X-3 TO X+3
7110 FOR K=Y-3 TO Y+3
7112 IF POINT(I,K)=0 THEN SET(I,K) ELSE
  RESET(I,K)
7120 NEXT K : NEXT I
7122 RETURN
7300 REM A MEGJELOLT PONT FELISMERESE .
7301 HE=0 : HI=0
7302 GOSUB 7500
7303 GOSUB 100
7304 GOSUB 6000
7306 GOSUB 7100
7308 PRINT$0,0,U$;U$;U$
7309 PRINT$0,0,"MI A NEGYZETTEL " ;
  "LETAKART HELY NEVE?"
7311 INPUT Q$ : PRINT$0,0,U$;U$;U$
7312 IF Q$=Q0$ THEN HE=HE+1:PRINT$1,0,
  "H E L Y E S ! " : GOTO 7316
7314 PRINT$1,0,"H I B A S ! HELYESEN: "
  ;Q0$ : HI=HI+1
7316 PRINT HE+HI; " JATEKBAN";HE;
  " TALALATOD VOLT."
7318 Q$="N" : INPUT "FOLYTATJUK";Q$
7319 GOSUB 7100
7320 FOR I=0 TO 3 : PRINT$I,0,U$;U$;U$
7321 NEXT I
7324 IF (Q$="I" OR ASC(Q$)=105) THEN
  7304 ELSE GOTO 10
7350 REM UTAZAS RAVEZETESSEL .....
7351 GOSUB 7520
7352 GOSUB 100
7354 GOSUB 6000
7355 X1=X:Y1=Y:D1=0:X=5:Y=5:HI=0
7357 SET(5,5)
7358 PRINT$0,0,U$;U$;U$
7359 PRINT$0,0,"AZ UTICEL: " ;Q0$
7360 GOSUB 5002
7361 RESET(5,5)
7362 D=SQR((X1-X)^2+(Y1-Y)^2)
7363 IF D1=0 THEN 7370
7364 IF D<D1 THEN PRINT$0,30,
  "KOZELEDIK"
7366 IF D>D1 THEN PRINT$0,30,
  "TAVOLODIK" : HI=HI+1
7370 D1=D : IF D>1 THEN 7360
7377 PRINT$0,0,"MEGERKEZTUNK!";U$;U$;U$
7379 GOSUB 7100
7380 IF L<>3 THEN PRINT$1,0,HI;
  " ALKALOMMAL TAVOLODTAL " ;
  "AZ UTICELTOL."
7388 Q$="N" : INPUT"FOLYTATJUK";Q$

```

```

7389 GOSUB 7100
7390 FOR I=0 TO 2:PRINT$I,0,U$;U$;U$
7392 NEXT I
7393 IF (Q$="I" OR ASC(Q$)=105) THEN
  7354 ELSE GOTO 10
7500 REM TAJEKOZTATO .....
7502 CLS
7504 PRINT :PRINT
7506 PRINT"A TERKEPEN EGY 'FOLDRAJZI " ;
  "PONTOT' TAKARUNKEL EGY NEGYZ" ;
  "ETTEL."
7509 PRINT"A NEGYZET KOZEPEEN LEVO 'PO" ;
  "NT' NEVET KELL"
7511 PRINT"BEIRNI."
7512 PRINT"(AZ ELSO BETU NAGY, A TOBB" ;
  "I KICSI.)
7514 GOTO 7600
7520 PRINT : PRINT
7521 PRINT"A KEPERNYO BAL ALSO RESZE" ;
  "SZEN LEVO PONTOT"
7523 PRINT"KELL AZ UTICELHOZ VEZETNI," ;
  " AZ 'E', A 'D', A 'K' ES AZ " ;
  " 'N' BETUK SEGITSEGEVEL. A BE-"
7528 PRINT"TUK AZ EGTAJAKAT JELOLIK."
7529 PRINT"A PONT SEBESSEGE A 'SHIFT' ;
  "-TEL ALLITHATO."
7531 PRINT"A KEPERNYO JOBB FELSO RESZ" ;
  "EN LEVO"
7533 PRINT"FELIRAT JELZI, HOGY KOZELE" ;
  "DUNK A CELHOZ, VAGY TAVOLOD" ;
  "UNK TOLE."
7536 PRINT : GOTO 7600
7550 PRINT : PRINT
7556 PRINT"A TERKEPEN EGY 'FOLDRAJZI " ;
  "PONTOT' TAKARUNKEL EGY NEGYZ" ;
  "ETTEL."
7559 PRINT"A NEGYZET KOZEPEEN LEVO 'PO" ;
  "NT' NEVET"
7561 PRINT"KIIRJUK."
7564 PRINT"AZ ELLENORZEST EGY KARAKTE" ;
  "R BEIRASAVAL"
7566 PRINT"FOLYTATHATJUK."
7600 PRINT : PRINT"A PROGRAM EGY KARA" ;
  "IKTER BEIRASAVAL FOLYTAT-HATD."
7604 Q0$=INKEY$ : IF Q0$="" THEN 7604
7605 CLS : PRINT CHR$(6)
7606 RETURN

```

3. Utazás hibajelzés nélkül. A program működése megegyezik az előzőével, de a gép nem írja ki a távolodások számát.

A programot Magyarország térképéhez készítettem, de más térképhez is használható. A működéséhez szükséges adatokat az érdeklődőknek levélben megküldöm. Ezeket DATA utasításokban a következő módon kell elhelyezni.

A 133-as sortól a 989-es sorig kell beírni a határ, a Duna, a Tisza és a Balaton vonala töréspontjainak koordinátáit, a 6100-tól a 6900-ig tartó sorokban lévő értékekkel összehangolva. Az értékek között 0 nem lehet, mert a „0,0” koordinátaérték az adott vonal rajzolásának befejezését jelzi a gépnek.

A 6000-es sortól egy gyors adatkezelő rész található, amely 9 csoportban tíz adategységet kezel. A földrajzi pontok adatait úgy kell kiegészíteni, hogy a következő sorok között tíz-tíz adategység (megnevezés és koordináták) legyen: 6100–6109, 6200–6209, 6300–6309, 6400–6409, 6500–6509, 6600–6609, 6700–6709, 6800–6809, 6900–6909. A 6910-es sorban levő értéket meg kell tartani.

A programot más térképre átirva, a metarányokat a 110-es, 112-es, 6032-es és 7005-ös sorokban kell módosítani.

SOMOGYI GYÖRGY

Már hagyományosnak számít, hiszen ezúttal negyedszer rendezték meg ezt a szakmai találkozót. A négynapos konferencia szervezői és lebonyolítói — a fonyódi Karikás Frigyes Gimnázium, Szakközépiskola és Kollégium, a TIT Budapesti Szervezete matematikai szakosztálya és az NJSZT Somogy Megyei Szervezete — munkáját dicséri, hogy az idén több mint százötvenen vettek részt közel 120 iskolából, s rajtuk kívül az előadók és a kiállítók.

A megnyitót dr. Komáromi József, a Fonyód Nagyközségi Tanács elnöke tartotta, aki nemcsak ízig-vérig lokálpatrióta, de a számítástechnikai oktatás lelkes támogatója is. A további előadók — tanárok, szakfelügyelők, ipari szakemberek — módszertani, oktatástechnikai, hardver- és szoftverkérdésekről, gépekről és rendszerekről tartottak előadásokat. Ízelítőként felsorolunk néhány érdekesebbet közülük.

Csóke Lajos (Eger): A számítástechnikai oktatás tapasztalatai

Kertész Ádám (Budapest): Programozási módszerek

Dr. Paronyai Gábor (Baranya m.): Középiskolai számítástechnikai szakkörök

Hajdú János (Csongrád m.): Z80 Assembler

Horváth István (Fonyód): Lokális hálózat HT gépre

Nyirati László (Székesfehérvár): FORTH programozás

Dr. Balla László (Zala m.): A középiskolai számítástechnika módszertana

Jelentősége miatt kiemelkedett a programból a számítástechnikai oktatás kérdéseiről folytatott vita, amely a Tudományos-szervezési és Informatikai Intézet igazgatójának beszámolóját követte. Páris György ugyanis a számítástechnikai oktatási program jelenlegi eredményeiről és jövőbeni folytatásáról beszélt, majd a számítástechnikát oktató pedagógusok ismertették sikerüket és elmondták gondjaikat.

A konferenciával egyidejű bemutatón a résztvevők „testközelben” vizsgálhatták meg a hazai iskolaszámítógép-gyártók termékeit, a HT-1080Z/64, a HT-3080C, a Primo, a TV-Computer, az Aircomp és a Homelab számítógépeket.

Legfontosabb talán, hogy a fonyódi találkozó a középiskolai számítástechnika olyan fórumává vált, ahol a pedagógusok szelektív kapcsolatot alakíthatnak ki számítástechnikusokkal és más intézményekben dolgozó kollégáikkal, így lehetőségük nyílik a közvetlen tapasztalatcserére. Ezért a rendezők és a résztvevők egyaránt a hagyomány folytatását szorgalmazzák.

ERDÉSZ ISTVÁN



# TÁJÉKOZTATÓ

## az 1987. évi Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Versenyről

A Művelődési Minisztérium — a Neumann János Számítógéptudományi Társaság és a Magyar Kommunista Ifjúsági Szövetség Központi Bizottsága szervezésében — az 1986/87-es tanévben is megrendezi a Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Versenyt.

A verseny tárgya, követelményei:

- számítástechnikai és programozási alapismeretek,
- számítástechnika-alkalmazási alapismeretek a középiskolai közismereti tárgyak, az egyszerű adatfeldolgozás stb. témaköréből,
- a feladatok rendszerszemléletű feldolgozása, algoritmusok célszerű kialakítása, a hibás adatok bevitelle elleni védelem, egyszerű és kényelmes kezelői felület kialakítása.

A versenyen a középfokú iskolák III. és IV. osztályos tanulói vehetnek részt. A versenyre legkésőbb

1987. január 9-ig

nevezhetik be az iskolák tanulóikat a következő címen:

**Neumann János Számítógéptudományi Társaság  
NEMES TIHAMÉR verseny  
1368 Budapest 5, Pf. 240**

A nevezésnek tartalmaznia kell

- az iskola nevét, címét és telefonszámát,
- a részt vevő tanulók számát évfolyamonként,
- a versenyért felelős tanár nevét.

A nevezést ajánlottan kell postázni.

Az első — az iskolában megtartandó — versenyforduló ideje

1987. február 10. kedd, 15.00 óra

Az első forduló dolgozatainak megíratásával és javításával kapcsolatos költségeket az iskolának kell viselnie.

A második, döntő fordulót Budapesten tartjuk meg

1987. április 11-én, szombaton.

Az időpontot és helyet később közöljük. A döntő résztvevőinek utazási költségét a küldő iskolának kell fedeznie.

A Nemes Tihamér OKSZTV első tíz helyezettje mentesül a felvételi vizsga alól és megkapja a maximális 30 pontot a matematika tárgyból, ha olyan felsőfokú oktatási intézménybe jelentkezett felvételre, amelyben a matematika felvételi tárgy. Ebben az esetben viszont matematikából érettségi vizsgát kell tenniük.

További kiemelkedő teljesítményért az NJSZT és a KISZ KB különjutalmakat ad.

A felkészülés megkönnyítésére a versenybizottság a következő könyveket ajánlja a versenyzők figyelmébe:

### 1. Alapvető számítástechnikai és programozási ismeretek

- 1.1 Szlávi—Zsakó: Módszeres programozás. Bp., 1986. MK.
- 1.2 Lócs: A BASIC és a Kíváncsi + Feladatgyűjtemény (Középiskolai szakköri füzetek). Bp., 1985., 1986. TK.
- 1.3 Wirth: Algoritmusok + adatstruktúrák = programok. Bp., 1982. MK.
- 1.4 Csépai: A számítástechnika alapjai. Bp., 1985. MK.

### 2. Nyelvek

- 2.1 Gordon—Körtvélyesi—Sós—Székely: Pascal programozási nyelv. Bp., 1982. SZÁMALK.
- 2.2 Bakos: Pascal PC-seknek. Bp., 1986. MK.
- 2.3 A LOGO programozási nyelv. Bp., 1986. MK.
- 2.4 Lásd 1.2

### 3. Alkalmazások

- 3.1 Newman—Sproull: Interaktív számítógépes grafika (első fele). Bp., 1985. MK.
- 3.2 Kaufmann: Az optimális programozás (Modellek és módszerek). Bp., 1968. MK.

A Nemes Tihamér OKSZTV versenybizottsága

# PÁLYÁZAT

A KISZ KB Középiskolai és Szakmunkástanuló Tanácsa és a Skála-Coop Számítógépes program pályázatot hirdet középiskolás diákok és pedagógusok számára.

A pályázat célja, hogy az ATARI típusú iskola-számítógép használatát megkönnyítse a tanórán és tanórán kívül, ezért a Skála a pályázaton elfogadott programokat MEGVÁSÁROLJA, és a gépek mellé egységcsomagban adja.

A pályázati szoftver vásárlási díja annak értékétől függ; néhány ezertől több tízezer forintig terjedhet. A pályázatra más géptípusra — Commodore 64, 16 — irt programot is beküldhetnek, ha az az ATARI számítógépre átirható. Átirásról a pályázat kiírói gondoskodnak. A programokat az alábbi témakörökben várjuk:

- oktatást segítő programok (demonstrációs, oktató, feleltető),
- mozzalmi munkát, szabadidős tevékenységet segítő programok (pl.: képűtségprogram, játékprogramok stb.),
- iskolai adminisztrációt segítő programok.

A programokat 1987. március 15-ig folyamatosan lehet beküldeni a következő címre:

**KISZ KB KSZT  
„ATARI PÁLYÁZAT”  
1388 Budapest Pf. 72.**

Beküldendő a program mágneses adathordozón (lemez vagy kazetta) és a program használatához szükséges leírás. A pályamunkákat szakemberekből álló zsűri bírálja el.

A megvásárlás mellett a SKÁLA által biztosított díjak:

1. díj ATARI 800 XL számítógép teljes kiépítéssel 1 éves használatra,
2. díj ATARI 800 XL számítógép teljes kiépítéssel fél éves használatra,
- 3—6. tárgyjutalom.

Ezen felül a díjazottak meghívást kapnak a KISZ KB KSZT nyári számítástechnikai táborába, a közösségek részt vesznek a KSZT µ-klub pályázatán is.

**KISZ KB  
Középiskolai  
és Szakmunkástanuló  
Tanács  
SKÁLA-COOP**



# Számítástechnika a fiataloknak

Hol tart ma a számítástechnika terjesztése az ifjúság körében Mongóliában? E kérdés nyomába indultunk Ulánbátorban, és így jutottunk el a Mongol Forradalmi Ifjúsági Szövetség Központi Bizottságának székházába.

— A mongol fiatalok számára az 1986-os év jelzi a számítógépes korszak kezdetét. A dolog tulajdonképpen Todor Zsivkov elvtárs, a Bolgár Kommunista Párt főtitkára 1984. évi látogatásával kezdődött — mondja Purevacem elvtárs, a dolgozó fiatalok osztályának munkatársa. — Zsivkov elvtárs felfigyelt arra, hogy itt nagy szükség van mikroszámítógépekre, és intézkedett bolgár gyártmányú berendezések szállításáról. 1986 májusában megérkezett 10 darab Pravec—82 típusú, az Apple II-vel kompatibilis gép a bolgár Dimitrovi Kommunista Ifjúsági Szövetség ajándéka-ként. Mindjárt szerveztünk egy háromnapos bemutatót Ulánbátor ifjúsági műszaki házában, és ott a fiatalok ezrei próbálhatták ki a gépeket. Hatalmas sikerük volt.

— Jelenleg hol vannak ezek a gépek, és hogyan juthatnak hozzájuk a fiatalok?

— Itt, a Központi Bizottság székházában rendeztünk be a gépek számára egy szobát. Lehetővé tesszük, hogy bárki in-

gyen használhassa őket, ezért a helyiséget állandóan nyitva tartjuk. Megszerveztük az éjjel-nappali konzultációt is biztosító teremfelügyeletet.

— Ezek a mikroszámítógépek az elsők között érkeztek Mongóliába. Hogyan készítették fel a konzultációt biztosító fiatalokat?

— Bár e gépek beérkezésekor valóban még csak pár darab, néhány hónappal „idősebb” mikroszámítógép volt az országban, már készültünk ezek fogadására. Jömagam például a szófiai egyetemen végeztem, és ott természetesen a mikroszámítógépek alkalmazását is tanultam. Ezenkívül

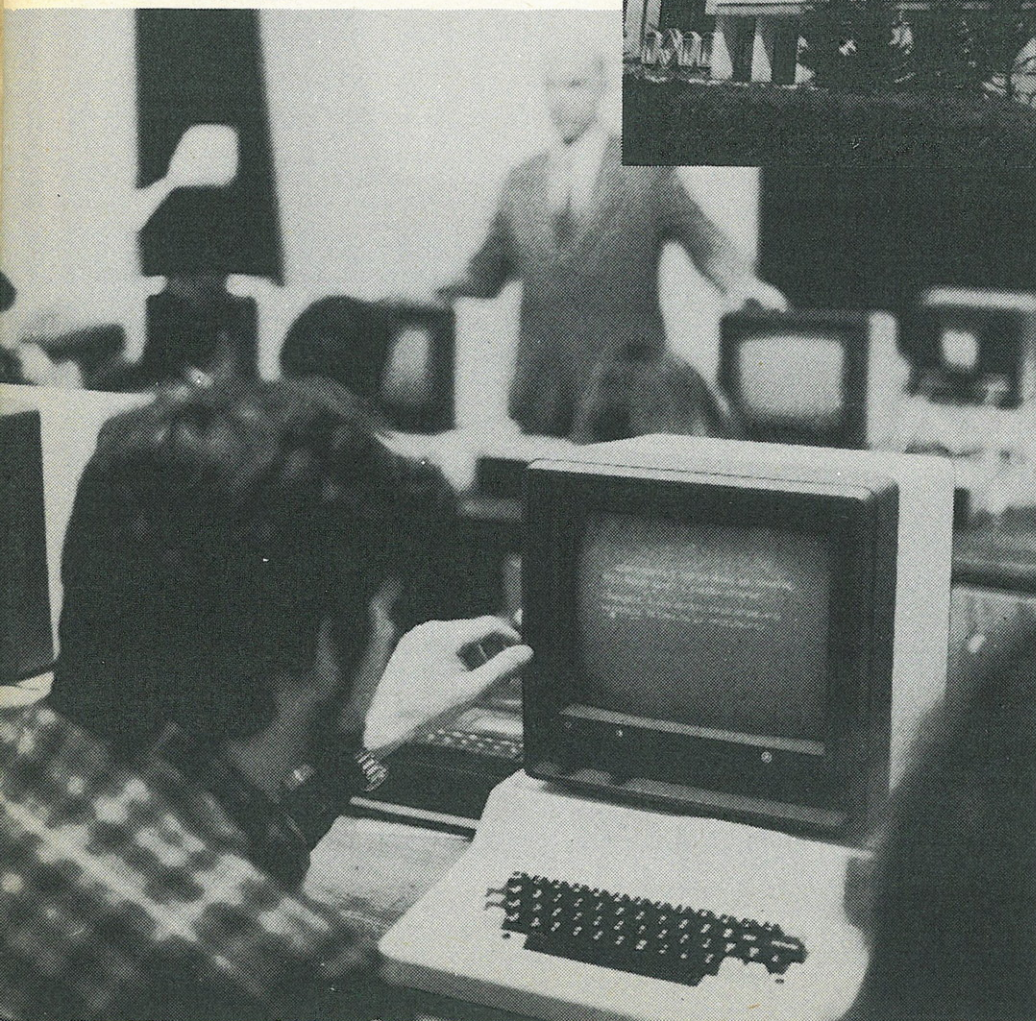
a dimitrovi komszomol a Pravec—82 gépekkel egy szakembert is küldött egy évre. Az ő feladata a gépközei információk átadása, az operatív konzultáció, valamint a számítástechnikai kultúrának az ifjúság körében való terjesztése.

— Van-e valamilyen szervezett formája az ismeretátadásnak?

— Folyamatosan indítunk tanfolyamokat, és szakkönyvek megjelentetésére is gondolunk. Kiadásunkban már meg is jelent az első füzet, amely a Pravec—82 kezelésének alapjait tartalmazza, természetesen mongol nyelven.



A Mongol Forradalmi Ifjúsági Szövetség székháza. Itt van a mikrokлуб



Pravec—82 típusú gépek — oktatókabinetben

— Milyen terveik vannak a számítástechnikának az ifjúság körében való terjesztésére?

— Ez két vonalon történik. Az állami irányvonal az iskolákon keresztül valósul meg, a művelődési minisztérium felügyelete alatt. Ennek keretében még az idén 30 szovjet mikroszámítógépet kapunk — valószínűleg az Apple II-vel kompatibilis Agat típusból. Az Állami Pedagógiai Intézet pedig már készíti elő a számítástechnika tantárgy bevezetését. Ez valószínűleg abba az iskolareformba épül bele, melynek keretében a jelenleg tízosztályos iskolai képzés 11 osztályosra alakul át. A másik vonal a számítástechnikai kultúra terjesztésének társadalmi irányvonala, melyet ifjúsági szövetségünk szervez. Itt jól kiépített bázisra támaszkodhatunk, mivel minden nagyobb városban vannak már ifjúsági műszaki házak. Ezekben szerelünk fel számítógéptermekeket, melyeket a szocialista országokból beszerzendő mikroszámítógépekkel akarunk felszerelni. Az ajándékba kapott 10 mikroszámítógép az első lépést jelentette. Ezzel ráléptünk egy hosszú útra, melyen a további métereket — vagy kilométereket — most készítjük elő.

DR. BROCKÓ PÉTER





A program a BASIC munkaterület alsó határa felett 2 k-val (4096 dec) kezdődik. Felhasználói karakterkészletek tervezésére és mágneslemezen való tárolására készült. Főbb funkciói tehát: adott kódú karakter képének megváltoztatása, lépkedés egyik karakterről a másikra, ugrás bárhova a karakterkészleten belül, karakterkészlet kimentése lemezre, karakterkészlet betöltése lemezről.

Apró és könnyen hibának vehető trükk a program 1000-es sora, melyre csak egyszer történik hivatkozás egy IF—THEN utasítással, a 210-es sorban. Ettől, illetve ennek a sornak a helyére, 1000-tól kerülhetnek esetleges bővítő programrészek, mivel e sorra csak akkor ugrik a program, ha olyan gombot nyomunk le, amit nem tud értelmezni.

A program a kurzorvezérlő gombokat használja a kurzornak adott karakteren belüli vezérlésére: a \* gombot a pont kigyújtására, a DEL-t a kikapcsolására, a CLR-t a karakter törlésére, a HOME gombot a kurzor visszapoziicionálására, a "+"-t és a "-"-t előre-hátra lépésre a karakterkészletben, az S-t és az L-t kimentésre, illetve betöltésre, a G-t pedig adott karakterre lépésre.

A program 9 blokk hosszú programfájlokba menti ki a karakterkészletet, így e kimentett sorozat bármilyen programhoz használható, feltéve, hogy a 2 k-tól 4 k-ig terjedő területet a felhasználói program nem használja.

A program a következő részekből áll: 100—117 A képernyő kirajzolása és a kezdő adatok beállítás

- 119—127 A karakter képének kirajzolása, várakozás egy gomb lenyomására  
128—159 Karakterkészlet-tervező funkciók  
160—173 Karakterkészlet-kimentő rutin  
180—193 Betöltő rutin  
200—204 A lemezegység hibacsatornáját lekérdező rutin  
210—220 Készletben ugró rutin  
1000— Szabad terület esetleges bővítések számára

A program beírása vagy betöltése előtt ne felejtjük el beírni a következőt:  
POKE 44,16:POKE 43,1:POKE 4096,0:NEW

OLÁH GERGELY

```

100 POKE650,128:PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
101 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
102 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
103 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
104 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
105 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
106 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
107 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
108 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
109 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
110 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
111 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
112 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND251:FORI=0TO2047
113 POKE2048+I,PEEK(53248+I):NEXT POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
114 POKE53272,(PEEK(53272)AND240)+2
115 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
116 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
117 C=1
118 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
119 FORI=0TO7:A=PEEK(2048+(C*8)+I):FORJ=7TO0STEP-1:POKE1024+40*I+(7-J),43
120 IF(A AND 217) <> 0 THEN POKE1024+40*I+(7-J),171
121 NEXT J
122 NEXT C
123 X=7:Y=0
124 POKE1024+(40*12)+11,C
125 POKE1024+9+(40*Y),31:POKE1024+(40*9)+(7-X),30
126 GETA$:IFA$="" THEN 126
127 POKE1024+9+(40*Y),32:POKE1024+(40*9)+(7-X),32
128 IFA$ <> "I" THEN 131
129 Y=Y+1:IFY < 7 THEN Y=0
130 GOTO124
131 IFA$ <> "I" THEN 134
132 Y=Y+1:IFY < 7 THEN Y=0
133 GOTO124
134 IFA$ <> "I" THEN 137
135 X=X-1:IFX < 0 THEN X=0
136 GOTO124
137 IFA$ <> "I" THEN 140
138 X=X+1:IFX > 7 THEN X=7
139 GOTO124
140 IFA$ <> "*" THEN 143
141 POKE2048+(C*8)+Y,PEEK(2048+(C*8)+Y)OR21X:POKE1024+(40*Y)+(7-X),171
142 GOTO124
143 IFA$ <> CHR$(20) THEN 146
144 POKE2048+(C*8)+Y,PEEK(2048+(C*8)+Y)ORND(255-21X):POKE1024+(40*Y)+(7-X),43
145 GOTO124
146 IFA$ <> "I" THEN 149
147 FORI=0TO7:POKE2048+(C*8)+I,0:NEXT I
148 GOTO119
149 IFA$ <> "I" THEN 152
150 X=7:Y=0
151 GOTO124
152 IFA$ <> "+" THEN 155
153 C=C+1:IFC=256 THEN C=255
154 GOTO119
155 IFA$ <> "-" THEN 158
156 C=C-1:IFC=-1 THEN C=0
157 GOTO119
158 IFA$ <> "G" THEN 160
159 GOSUB210:GOTO119
160 IFA$ <> "S" THEN 180
161 F$="":PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
162 GETA$:IFA$="" THEN 162
163 IFASC(A$) <> 31 AND ASC(A$) <> 96 AND LEN(F$) <> 16 THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
164 IFA$=CHR$(20) AND F$ <> "" THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
165 IFA$=CHR$(13) THEN 167
166 GOTO162
167 OPEN15,8,15,"IO":OPEN1,8,1,"O": "F$+"
168 GOSUB200
169 PRINT#1,CHR$(A1-INT(A1/256))*256):GOSUB200
170 PRINT#1,CHR$(INT(A1/256)):GOSUB200
171 FORI=A1TOA2:PRINT#1,CHR$(PEEK(I)):NEXT I:A1=1:CLOSE1:GOSUB200:CLOSE15
172 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
173 GOTO119
180 IFA$ <> "L" THEN 210
181 F$="":PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
182 GETA$:IFA$="" THEN 182
183 IFASC(A$) <> 31 AND ASC(A$) <> 96 AND LEN(F$) <> 16 THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
184 IFA$=CHR$(20) AND F$ <> "" THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
185 IFA$=CHR$(13) THEN 187
186 GOTO182
187 OPEN15,8,15,"IO":OPEN1,8,0,"O": "F$+"
188 GOSUB200
189 GET#1,A$:GOSUB200
190 GET#1,A$:GOSUB200
191 FORI=A1TOA2:GET#1,A$:POKEI,ASC(A$)+0: "NEXT:CLOSE1:GOSUB200:CLOSE15
192 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
193 GOTO119
200 INPUT#15,E1,E2$,E3,E4:IFE1=0 THEN RETURN
201 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
202 GETA$:IFA$="" THEN 202
203 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
204 RETURN
210 IFA$ <> "G" THEN 1000
211 F$="":PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
212 GETA$:IFA$="" THEN 212
213 IFA$=CHR$(47) AND ASC(A$) <> 58 AND LEN(F$) <> 3 THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
214 IFA$=CHR$(20) AND F$ <> "" THEN PRINTA$: "KARAKTERKESZLET TERVEZESRE KESZULT."
215 IFA$=CHR$(13) THEN 217
216 GOTO212
217 C=VAL(F$):IFC <> 0 THEN C=0
218 IFC > 255 THEN C=255
219 PRINT"KARAKTERKESZLET TERVEZESRE KESZULT."
220 RETURN
1000 RUN

```



# Mit tud

# a PROLOG ?



**Az alábbiakban egy olyan programozási nyelvről lesz szó, amelynek filozófiája alapvetően eltér a közismert programozási nyelvekétől.**

**A hagyományos programozási nyelvekben (FORTRAN, BASIC, FORTH, PASCAL, Ada stb.) a géppel való kapcsolat egyoldalú és egysíkú; a felhasználó mindig felszólító módban érintkezik a géppel: tedd ide, tedd oda, add össze, szorozd meg. E nyelvek ilyen, ún. imperatív vagy parancsotzó jellegével szemben a PROLOG egy egészen más, békésebb, erőszakmentes programozási stílust képvisel. A hagyományos nyelvek „parancs”, „utasítás”, „eljárás” stb. fogalmai helyett „tény”, „állítás”, „tudás” fogalmakban gondolkodik, és egy feladat megoldása a feladatot végrehajtó algoritmus elindítása helyett a feladatot leíró tudás bizonyos elemeire való rákérdezést jelent.**

## A PROLOG története

A mesterséges intelligencia kutatói általában az emberi gondolkodás egyik modelljének alapjául a matematikai logika tudományát tekintik. Ez irányú kutatások melléktermékeként fejlesztették ki a franciaországi Marseille-ben az első PROLOG interpretert, mint a matematikai logika egyik résznyelvében felírt matematikai tételek bizonyítására alkalmas eszközt. A francia kutatók ezt az első PROLOG interpretert a (francia) természetes nyelv feldolgozására (értelmezésére, fordítására) akarták használni.

Az eredeti interpreter egy példánya 1973-ban átkerült Angliába, az edinburghi egyetemre, ahonnan 1974-ben eljutott Magyarországra. Mivel a magyar kutatók ezt az interpretert nem tudták a hazai gépeken feléleszteni, úgy döntöttek: írnak egy újat. Így jött létre az első magyar PROLOG interpreter 1977–78-ban. Eközben Edinburg-ben elkészítették az első PROLOG fordítóprogramot is.

1981-ben lényeges változást hozott az, hogy a japánok bejelentették az 5. generációs számítógépek, vagyis az emberrel összehasonlítható intelligenciájú gépek fejlesztését. Noha a fejlesztés rendkívül hosszú távú célokat tűzött ki, azt már az elején

el döntötték, hogy ilyen színvonalú feladatok megfogalmazására csakis a PROLOG nyelv alkalmas. Elképzeléseik szerint az 5. generációs gépek a PROLOG-ot „helyből” fogják beszélni, úgy, mintha gépi kódjuk lenne.

Ezek az események nagy lökést adtak a nyelv fejlődésének. A világon sorra jelentek meg az újabb és újabb PROLOG rendszerek. Ezek közül most csak kettőt szeretnénk kiemelni.

Az egyik a hazai fejlesztésű MPROLOG. Nevében az M a modularitásra utal. A rendszer fő előnye az, hogy — éppen a modularitásból kifolyólag — rendkívül kifinomult eszközöket ad a felhasználó kezébe, olyanokat, amelyekkel több ember által megírt, igen nagy méretű és összetett programok is könnyen kezelhetők. A rendszer külön jellegzetessége a T/TS PROLOG kiegészítés, amely az MPROLOG-ot párhuzamosan megoldandó szimulációs feladatok megfogalmazására is alkalmassá teszi.

A másik az angliai fejlesztésű micro-prolog, amely jóval szűkebb felhasználói lehetőségeket nyújt, viszont elsősorban mikro-számítógépeken terjedt el. A legkisebb micro-prolog-változat a mindössze 48 kilobájtos Sinclair ZX-Spectrum számítógépen fut, és példányai Magyarországon is megtalálhatók. A micro-prolog a Logic Programming Associates, a Sinclair ZX-Spectrum a Sinclair Research cégek védjegye.

Minthogy az MPROLOG a világon legelterjedtebb és szabványként elfogadott PROLOG nyelvtan ajánlásait követi, és a cikk írója is a magyar fejlesztőcsoport egyik képviselője, ezért a cikkben felhozott programpéldák az MPROLOG nyelvhez kötődnek.

## A PROLOG elemei

Egy PROLOG program elindítása — mint már említettük — az előzőleg felvitt ismeretbázison alapuló kérdések feltevését jelenti, futása pedig a kérdésekre való válaszadás, illetve a kérdések, mint új állítások bebizonyítása.

Nézzük meg, hogy milyen eszközökkel adható meg az ismeretbázis. A PROLOG-világ két különböző osztályból áll:

— a tulajdonságok osztályából, amelyeket a programban különböző nevekkkel (pl. szép, piros stb.) azonosítunk, és

— a dolgok, tárgyak osztályából, amelyek szintén lehetnek nevek, de lehetnek számok, vagy egyéb speciális szimbólumok is. Külön jelentősége van az ún. „nem definiált tárgyak” osztályának, amelyeket megkülönböztettképpen a közönséges nevektől nagy kezdőbetűvel írunk, ellentétben a

többi nevekkkel, amelyeknek mindig kisbetűvel kell kezdődniük.

Ha egy tulajdonságnevet nulla vagy több objektummal, mint paraméterekkel képzelünk összekapcsolunk, akkor egy állításelemet kapunk. Az állításelem tulajdonképpen csak egy kifejezési forma, amelynek önmagában nincs külön értelme.

Az állításelemet írásban úgy jelöljük, hogy a tulajdonság neve után zárójelbe téve és vesszőkkel elválasztva leírjuk a paraméterobjektumokat. Például a szerelmes(csokonai,lilla)

kék(tenger)

zöld(Valami)

állításokban a „szerelmes”, „kék”, „zöld” tulajdonságneveket kapcsoltuk össze a „csokonai”, „lilla”, „tenger”, objektumokkal és a „Valami” nem definiált objektummal.

Ha egy állításelemnek azt az értelmezést tulajdonítjuk, hogy az öt létrehozó tulajdonság a vele összekapcsolt objektumokra mindig igaz, akkor elemi állítást vagy tényállítást kapunk.

Az egyetlen állításelemből álló feltételt egyszerű feltételnek nevezzük. Helyes feltételekből a VAGY és az ÉS logikai műveletek segítségével képezhetünk újabb, összetett feltételeket. Ha egy feltételt és egy állításelemet mint következményt a HA—AKKOR logikai művelet segítségével kapcsolunk össze, akkor összetett állításokat vagy következtetések leírására szolgáló szabályokat kapunk. Az azonos nevű és paraméterszámú szabályokat együttesen definíciónak, az egy definíciót felépítő állításokat, és az egy állításban VAGY kapcsolóval összekapcsolt részfeltételeket változatoknak vagy alternatíváknak nevezzük. Egy szabály feltételrészében szereplő elemi feltételeket hivatkozásnak vagy röviden hívásnak nevezzük.

A tényállításokat úgy írjuk le, hogy vesszük az őket felépítő állításelemet, a szabályokat pedig, hogy a következményt és a feltételeket a „:—” következtetést jelző szimbólummal választjuk el, majd mindkettőt egy „.” (pont) szimbólummal zárjuk le. A feltételek megfogalmazásánál vagy csak a megfelelő állításelemet vesszük (egyszerű feltételek esetén), vagy helyes feltételeket a „:” (vessző) ÉS szimbólummal, illetve „:” (pontosvessző) VAGY szimbólummal elválasztva hozhatunk létre összetett feltételt. Az egyértelműség biztosítása végett megengedett a tetszőleges mélységű zárójelzés is. Például:

zöld(mező)

mellékbolygója(nap,merkur)

vizes a járda:—esik az eső;locsoltak.

A tényállítások és a következtetési szabá-



lyok szolgálnak logikai programunk építőköveitül. Egy PROLOG program megírása a program logikai magját alkotó tényállítások és szabályok megírását jelenti. A teljes PROLOG programban nyer igazi értelmet a változók használata: ha a programot — az algebrai egyenletrendszerekhez hasonlóan — logikai állításrendszerként fogjuk fel, akkor a változók jelentése teljesen megegyezik az algebrai egyenletrendszerekben használt változók értelmezésével; míg ott az egyenlet megoldásának célja a változók az egyenlet által megfogalmazott megkötéseket kielégítő értékeinek megkeresése, addig a PROLOG program futtatásának célja a logikai változók azon értékeinek megkeresése, amelyek felvétele esetén a célállítást a programból logikailag levezethető.

Megjegyezzük, hogy a PROLOG nyelvdefiníció szabályainak értelmében az ugyanolyan nevű változószimbólumok csak egyetlen (alap- vagy összetett) állításon belül tekintendők azonosnak, és ha egy változó a program futása során értéket kap, akkor az azonos változók is egyidejűleg felveszik ugyanazt az értéket. Külön állításban előforduló azonos nevű változók értékértelme egymástól teljesen független. Egy PROLOG program végrehajtásának egy pillanatában azokat a változókat, amelyekhez a rendszer már talált behelyettesítési értéket, a talált értékkel lekötött, a többi lekötetlen vagy szabad változóknak nevezük.

A már megtervezett és megírt PROLOG programban az állításelemek felhasználásával tehetünk fel kérdéseket úgy, hogy őket ÉS kapcsolóval összekapcsoljuk. Az ilyen összekapcsolt kérdéseket célsorozatnak nevezük. Egy ilyen kérdés eredményeképpen a rendszer megpróbálja bebizonyítani, hogy az egyes állításelemek logikailag egyszerűen levezethetők-e az ismeretbázis állításaiából.

Noha más logikai műveletek segítségével is, de az adott műveletekkel is másféle szerkezeti állítások felépítése is elképzelhető, a PROLOG-ban alkalmazott megszorítások, vagyis hogy egy állítás következményoldala mindig egy egyszerű állításelem, valamint hogy a feltételoldal csak az ÉS és VAGY műveleteket alkalmazhatjuk, biztosítják azt, hogy egy ilyen bizonyítás a gép által elvégezhető, és a programozó által áttekinthető, jól követhető legyen.

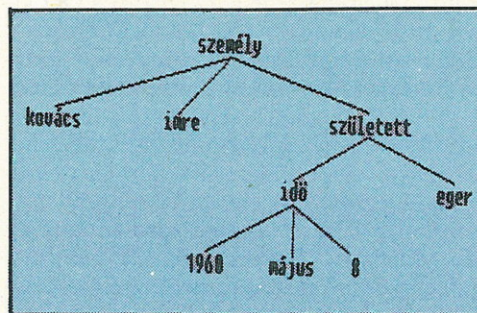
Az esetek legnagyobb részében az objektumok eddig elmondott osztályozása nem elegendő a megoldandó feladatok logikai alapú megfogalmazásához. Gyakran szükséges az objektumok szerkezetének finomabb ábrázolása, és hogy legyen lehetőség az objektumok egyes részeire is állításokat kimondani. Ezért az elmondott, ún. egyszerű objektumok fogalmát ki kell bővíteni, vagyis eszközöket kell adni összetett objektumok kezeléséhez is. A PROLOG-világban összetett objektumot képezhetünk úgy, hogy nulla vagy több egyszerű, vagy akár összetett objektumot egy felépítő (konstruktor) szimbólum segítségével egymással összekapcsolunk. Fontos megszorítás, hogy a felépítő szimbólum csakis egy egyszerű

név lehet, amelyet az objektumra való bármilyen hivatkozás esetén ismerni kell.

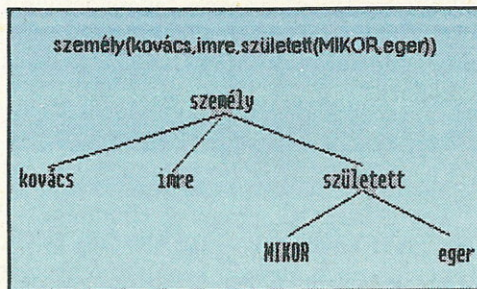
Az ily módon képzett objektumok olyan fákkal ábrázolhatók, amelyek csúcspontja a felépítő szimbólum, és amelyet a csúcspontból kiinduló ágak a paramétereket reprezentáló részfakkal kötnek össze. Természetesen az így képezhető faobjektumok is tartalmazhatnak változókat, értelemszerűen valamelyik levélelemük helyén. Egy változó is tartalmazó fa olyan félig meghatározott objektumot jelöl, amelynek a gyökér felé eső része teljesen meghatározott, és a teljes fa úgy válhat meghatározottabbá, hogy a változó helyére a programfutás során behelyettesíthető egy olyan — változó tartalmazó vagy nem tartalmazó — objektum, ami kielégíti a program és a célállítás által meghatározott logikai feltételeket.

Megjegyezzük, hogy nem magától értetődő programtervezői munka egy adott feladat követelményeihez legjobban illeszkedő objektumszerkezet megtervezése.

Az összetett objektumok írásmódja: a konstruktorszimbólum után zárójelek között, vesszőkkel elválasztva leírjuk az összetett szimbólumokat. Például a személy(kovács,imre,született(idő(1960,május,8),eger)) a következő faobjektumot jelenti:



A nem teljesen meghatározott objektumokra példa a következő, ahol a nem teljes meghatározottság az előzőhöz viszonyítva a születési időt jelző részben jelentkezik:



Az MPROLOG nyelv lehetőséget ad bizonyos szimbólumok ún. operátorként való deklarációjára. Az operátorszimbólumokat egy vagy két összetevős összetett objektumok leírásánál használjuk; ha egy operátort közékelődőnek (infix) nyilvánítunk, akkor amennyiben a szimbólum kétösszetevős kifejezést épít fel, úgy a megszokott zárójeles jelölésmód helyett a szimbólumot a zárójelek nélkül, a kétösszetevő objektummal közrefogva is írhatjuk, így is ugyanazt az összetett kifejezést jelenti. Azért, hogy a többszörösen összetett kifejezések használata is egyértelmű legyen, a

module árpád\_ház.

/\*\$ject\*/  
body.

operator(apja,lr,0).  
operator(nagyapja,lr,0).  
operator(király,st,0).

árpád apja álmós.  
sofi apja árpád.  
taksony apja sofi.  
mihály apja taksony.  
géza apja taksony.  
vazul apja mihály.  
1.endre apja vazul.  
1.béla apja vazul.  
salamon apja 1.endre.  
1.géza apja 1.béla.  
1.lászló apja 1.béla.  
1.kálmán apja 1.géza.  
1.istván apja géza.  
imre apja 1.istván.

1.endre király.  
1.béla király.  
1.géza király.  
1.lászló király.  
1.kálmán király.  
salamon király.  
1.istván király.

UNOKA nagyapja NAGYPAPA :-  
PAPA apja NAGYPAPA, UNOKA apja PAPA

:- VALAKI nagyapja taksony, VALAKI király.

endmod /\* árpád\_ház \*/.

deklarációban jelezni kell a zárójelezés irányát és az új operátor prioritását is.

Ha egy szimbólumot elő- (prefix) vagy utóképzős (suffix) operátornak deklarálunk, akkor egyösszetevős kifejezéseket zárójelezés nélkül, a felépítő szimbólumot az egyetlen összetevő elé, illetve után írva is kifejezhetjük magunkat.

Megjegyezzük, hogy egy MPROLOG állítás tulajdonképpen egy összetett kifejezés formájú, hiszen az egyes állításelemek felépítése megegyezik az összetett kifejezések ábrázolásmódjával, és a HA—AKKOR, a VAGY és az ÉS szimbólumok (":-" ";","";") szintén mint alapoperátorok vannak deklarálva.

Az operátorok használata jól megfigyelhető a fenti programban, amely az Árpád-házi királyok egy részének családfáját írja le.

## Végrehajtási algoritmus

Az algebrai egyenletekhez hasonlóan a PROLOG nyelven felírt logikai állításrendszereknek is van megoldási módszerük. Tegyük fel, hogy adott egy PROLOG állításrendszer és adott egy célállítás-sorozat, amelyet bizonyítani kívánunk. A megoldási módszer lényege: módszert adunk egy célsorozathoz olyan újabb célsorozatok keresésére, hogy ha az új célsorozat bizonyíthatóan teljesül, akkor az eredeti célsorozat



nak is teljesülnie kell. Ha ilyen műveletek alkalmazása során a mindig teljesülő, egyetlen célállításelemet sem tartalmazó üres célsorozathoz jutunk, akkor ebből következően a kiindulási célsorozatnak is teljesülnie kell.

A módszer leglényegesebb lépése a szabályok „visszafelé” alkalmazása. Definíció szerint ugyanis egy szabály akkor alkalmazható visszafelé, ha a következményrésze illeszkedik a hívásra, vagyis ha:

- a hívás neve és paramétereinek száma megegyezik a szabály következményrész nevével és paramétereinek számával,
- a paramétereik páronként vagy megegyeznek, vagy a bennük szereplő változóba behelyettesíthetők objektumok úgy, hogy azzal megegyezővé tehetők.

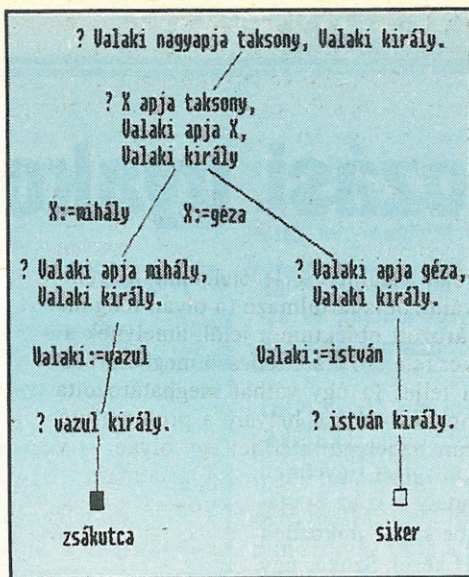
Ezt az illesztés-behelyettesítési lépést a PROLOG nyelvben egyesítésnek nevezik. Az egyesítés pontosabb definíciója: minden objektum egyesíthető önmagával; egy változó minden objektummal egyesíthető úgy, hogy a változó felveszi a másik objektumot értékül; egy összetett objektum akkor illeszkedik egy másik összetett objektumhoz, ha a nevük és a paraméterszámuk megegyezik, és a paramétereik páronként egyesíthetők; más esetben a két objektum nem egyesíthető.

Ezek figyelembevételével a végrehajtási módszer a következő. Ha egy következtetési szabályt „visszafelé” alkalmazunk, azaz a célsorozat valamelyik elemére illeszkedő következményszerű szabályt találunk, és egyesítés után a célállításelemet kicseréljük a szabály feltételrészére, akkor kapunk egy új célsorozatot, ami rendszerint hosszabb, de egyszerűbb állításelemekből áll, mint az eredeti. Ha illeszthető tényállítás is, az elfogható üres feltételrészű szabálynak is. Ez azt jelenti, hogy az adott állításelem közvetlenül teljesül, és ilyenkor ez az állításelem a még bizonyítandó célállítás-sorozatból elhagyható.

Ehhez az általános módszerhez a PROLOG nyelvdefiníció még a következő megszorításokat teszi hozzá:

- Mindig az aktuális célsorozat első elemével kell az illesztést megkísérelni.
- Ha az aktuális célsorozat első elemére több szabály vagy tényállítás is illeszkedik, akkor azt mondjuk, hogy választási ponthoz értünk. Ilyenkor mindig a programszöveg szerinti legelső választási lehetőség szerint kell a végrehajtást folytatni, és a többi ág végigjárására csak később kerülhet sor.

– Ha az aktuális célsorozat első elemére egyetlen szabály vagy tényállítás sem illeszkedik, akkor azt mondjuk, hogy a bizonyítás zsákutcába jutott. Ilyenkor meg kell kísérelni a legutoljára elhagyott választási pontból a következő választási lehetőség szerint folytatni a végrehajtást. Ezt a műveletet visszalépésnek (backtracking) nevezük. Fontos megjegyezni, hogy a visszalépés során a program teljesen visszalép korábbi állapotába, így például az időközben lekötött változók ismét felszabadulnak. Amennyiben az adott választási pontnál már az összes választási lehetőséget kimerítettük, akkor egy korábbi választási pont-



nál kell a további végrehajtást folytatni, és ha már a legkorábbi választási pontnál sincs szabad lehetőség, akkor azt mondjuk, hogy a bizonyítás kudarcra végződött, az eredeti célsorozat az adott ismeretbázis feltételei mellett nem teljesül.

## Levezetési fa

PROLOG programok végrehajtása jól szemléltethető az ún. levezetési fa segítségével. A fa gyökere a kiindulási célsorozat, és az egyes csúcsai a levezetés során előálló újabb célsorozatok. A levezetési fa élére a megfelelő egyesítési lépés által létrehozott változó megkötéseket szokás írni. Példaképpen tekintsük az Árpád-ház programhoz feltett

? Valaki nagyapja taksony, Valaki király.  
kérdés alapján előálló levezetési fát.

## Beépített állítások

Minden PROLOG rendszerben vannak olyan állítások, amelyeket a felhasználónak nem szükséges definiálnia, amelyeket a rendszer helyből ismer. Az ilyen állítások rendszerint valamilyen alapadattípuson értelmezett relációt valósítanak meg, bizonyos részük a rendszer egészét érintő (globális) mellékhatással is rendelkezik. Ezek az állítások általában érzékenyek arra is, hogy meghívásuk pillanatában mely paraméterük van lekötött, illetve szabad állapotban, és amennyiben a programozó nem ügyel ezekre a megszorításokra, akkor a rendszer hibaállapotba kerül, vagy egyéb nem várt módon reagál.

Az MPROLOG rendszerben megvalósított beépített állítások főbb csoportjai:

– Periféria- és fájlkezelő eljárások; betűk, jelek, szavak, illetve MPROLOG kifejezések kiírása, beolvasása, lemez, nyomtató, képernyő kezelése.

– Aritmetikai relációk: egész- és lebegőpontos tizedes számokból az alapműveletek segítségével felépülő számtani kifejezések értékének kiszámítása, összehasonlítások.

– Kifejezés- és füzérkezelés: MPRO-

LOG kifejezések futásidejű felépítése, illetve lebontása, szövegdaraboló, összefűző, összehasonlító eljárások.

- Végrehajtást befolyásoló eljárások.
- Műveletek a programok szerkezetén: programmodul, moduleljárás, relációk, állítások törlése, új állítások felvétele.
- Hibaállapot-lekezelő eljárások.

## A "!" (out) beépített eljárás

A "!" olyan beépített eljárás, amely mellékhatása miatt egyáltalán nem értelmezhető logikai állításként, és működése csak a PROLOG végrehajtási mechanizmus ismeretében érthető meg. Használatával a PROLOG nyelv holmi PASCAL vagy Ada színvonalára „fokozható le”, amennyiben a nyelv egyik fő erejét, a logikai működésmódot biztosító visszalépést tilthatjuk le vele. Használatát a levezetési fa ismeretében a legkönnyebb értelmezni.

Képzeld el, hogy a levezetési fán minden szabályegyesítést jelképező élhez megjegyezzük a felhasznált szabály valamilyen azonosítóját is. Nevezük ezt a pontot szabálybelépési pontnak. Amikor a "!" egyesítése kerül sorra, azaz amikor az aktuális célsorozatnak éppen egy "!" az első eleme, akkor általában már messze járunk a rá hivatkozó szabály belépési pontjától, és rendszerint a közöttük fekvő úton több nyitott elágazás, visszalépési pont is található. A "!" eljárás hatására a rá hivatkozó szabály belépési pontjaig az összes visszalépési pont megszűnik, mert a "!" mintegy „levágja” a levezetési fa nyitott ágait.

Program-végrehajtási szempontból a "!" hatására saját definíciójának további alternatíváira és a feltételsorban öt megelőző hívások által nyitva hagyott visszalépési pontokra már sohasem kerülhet a vezérlés.

## Összefoglalás

A PROLOG nyelv megjelenése hatalmas előrelépést jelentett elődeihez viszonyítva. A nyelv egzakt értelmezése következtében matematikai módszerekkel jól kezelhető és leírható. Ugyanezen okokból, valamint a programok és adatok egységes ábrázolásmódja miatt a nyelv maga is gépi eszközökkel is rendkívül könnyen, rugalmasan kezelhető.

A nyelvben alkalmazott főbb újszerű technikák, a visszalépés, a mintaillesztés, a futásidejű programkezelés stb. a nyelvnek olyan erőt adnak, amelyet elsősorban olyan feladatokhoz ajánlott alkalmazni, ahol különleges bonyolultságú döntéseket kell véghezvinni, vagy logikai következtetéseket kell levonni. A nyelv különösen alkalmas egyedi igényeket kielégítő alkalmazási programok gyors előállítására. Főbb alkalmazási területei: a természetes nyelvek feldolgozása, adatbázisok természetes nyelvű lekérdezése, a következtető rendszerek, például szakértői rendszerek következtető alrendszere és a számítógéppel segített tervezés (építészet, gyógyszertervezés, áramkörtervezés stb.).

KILIÁN IMRE



## Z80 programozási gyakorlatok 6.

Ahogy az előző részben megígértem, most megnézzük, hogy pontosan mi is az az interrupt (magyarul: megszakítás), és assembly programozás során milyen feladatok adódhatnak ebben a témakörben.

A Z80 processzornak két megszakításbemenete van. Ha az NMI (nem maszkolható megszakítás) bemenetre érkezik megszakításkérés, akkor a processzor a következő beolvasott utasítást figyelmen kívül hagyja. Először a programszámlálónak, amely már az új utasításra mutat, a felső, majd alsó bájta a verembe kerül, miközben az SP kétszer csökken eggyel, majd a PC-be 66H kerül. Szokás úgy fogalmazni, hogy a processzor egy újraindítást hajt végre a 66H címre. Ez a megszakításkérés nem maszkolható, azaz szoftver úton nem lehet tiltani az elfogadását.

A processzor másik megszakításbemenete az INT láb. Az erre a bemenetre érkező megszakításkérés elfogadása letiltható a DI utasítással, és engedélyezhető az EI utasítással. A valóságban két megszakításengedélyező bit van a központi egységben, szokásos jelölésük IFF1 és IFF2. Ha IFF1 értéke 1, akkor engedélyezve van a megszakítás, ha nulla, akkor a megszakításkérésről nem vesz tudomást a processzor. A DI utasítás mindkét bitbe nullát tesz, az EI pedig 1-et. Ez utóbbi végrehajtása után még a következő utasítást is végrehajtja a processzor, és csak ezután fogad el egy esetleges megszakításkérést. Ennek az az oka, hogy a megszakításrutin teljes lefutásáig általában nem fogadható el újabb megszakítás, márpedig az EI után kell hogy legyen még legalább egy RET utasítás.

Egy maszkolható megszakításkérés elfogadása után a processzor mindkét bitet törli, ezzel a további megszakításkérések elfogadását tiltja. IFF2 feladata az, hogy nem maszkolható megszakítás elfogadásakor megőrizze IFF1 tartalmát. Ilyenkor IFF1 értéke IFF2-be, IFF1-be pedig nulla kerül. Az NMI rutin végrehajtása végén a RETN utasítás hatására IFF1-be visszakerül IFF2-ből az eredeti érték, így ha tiltva volt, akkor tiltva, ha engedélyezve volt, akkor engedélyezve marad a maszkolható megszakítás az NMI rutin lefutása után is.

Foglalkozunk a továbbiakban a maszkolható megszakításokkal. A Z80-nak három megszakítás üzemmódja van.

### 6.1 program

```

6.1 program
ORG 65000 ;interrupt
SETINT DI ;
LD A,9 ; beállítás
LD I,A ; I:=9
IM 2 ; kettős interrupt mód
EI ;
RET ;
ORG 65129 ; a kiszámított cím
ORA PUSH AF ; regiszterek
PUSH HL ; elmentése
LD HL,CLOCK ; HL végig az óra megfelelő
; bájtyára mutat
LD A,50 ; 1 sec=50*20 ms
CALL STEP ;
LD A,60 ; 1 min=60 sec
CALL STEP ; (akku megmarad)
CALL STEP ; 1 h=60 min
LD A,24 ; 1 d=24 h
CALL STEP ;
RETIT POP HL ; regiszterek
POP AF ; vissza
EI ; engedélyezni kell
; az interruptot
STEP INC (HL) ; megfelelő bájtnövelése
CP (HL) ; elérte-e a limitet?
JR NZ,RETURN ; nem
LD (HL),0 ; törlés
INC HL ; mutató növelése
RET ; normális visszatérés
RETURN POP HL ; visszatérési cím elhagyása
JR RETIT ; vissza az interruptból
END ;
    
```

### 6.2 program

```

ORG 65129 ;interrupt cím
MAX EQU 10 ; a buffer max mérete
MASINT EQU $ ;
PUSH AF ; regiszterek
PUSH BC ; elmentése
PUSH DE ;
PUSH HL ;
LD HL,CLOCK ; az óra
DEC (HL) ; csökkentése
LD HL,FLAG ; van e
BIT 1,(HL) ; billentyűzet
JR Z,KHIT ; kérés
CALL RDB ; puffer olvasás
OR A ; ha
JR Z,KHIT ; üres a puffer
LD (LAST),A ; a billentyű érték
LD HL,FLAG ; érvényes billentyű
RES 1,(HL) ; jelzése
KHIT CALL READ ; a billentyűzet leolvasása
OR A ;
JR NZ,NOKEY ; nincs billentyű
LD HL,FLAG ;
BIT 0,(HL) ; első?
JR NZ,REPK ; ismételnél ugrás
SET 0,(HL) ; volt billentyű beállítása
CALL WRTB ; beírás a pufferbe
LD A,(ERT1) ; az óra beállítása
JR STEC ;
REPK LD B,A ; ismételt billentyű
LD A,(CLOCK) ; az óra
OR A ; nulla-e?
JR NZ,RETURN ; ha nem akkor visszatérés
LD A,B ; billentyű érték vissza A-ba
CALL WRTB ; beírás a pufferbe
LD A,(ERT2) ; ismételt várakozás
STEC LD (CLOCK),A ; óra beállítása
JR RETURN ; visszatérés
NOKEY LD HL,FLAG ; nem volt
RES 0,(HL) ; lenyomott billentyű
RETURN POP HL ; regiszterek
POP DE ; vissza
POP BC ;
POP AF ;
EI ;
RET ;
; olvasása a pufferből
RDB LD A,(BL) ; A-ba a puffer hossza
OR A ; zero bit állítása
RET Z ; ha üres a puffer
DEC A ; puffer hossza
LD (BL),A ; eggyel csökken
LD HL,(FO) ; a kiolvasandó karakter
LD A,(HL) ; A-ba
CALL STEPHL ; a mutató
LD (FO),HL ; léptetése
RET ;
; írás a pufferbe
WRTB LD B,A ; karakter kód B-be
LD A,(BL) ; a puffer hossza?
CP MAX ; tele
RET Z ; van a puffer
INC A ; puffer hossza
LD (BL),A ; eggyel nő
LD HL,(FI) ; a beírandó karakter
LD (HL),B ; a helyére
CALL STEPHL ; a mutató
LD (FI),HL ; léptetése
RET ;
; HL-t a puffer következő elemére állítja
STEPHL PUSH HL ; HL elmentése a vizsgálat miatt
LD DE,(BE) ; DE=puffer vége
OR A ; carry:=0
SBC HL,DE ; carry:=1 ha HL<DE
POP HL ; HL vissza
INC HL ; normális léptetés
LD C ; HL nem túl nagy
RET HL,(BS) ; ha HL túl nagy akkor a puffer elejére
RET ;
; V A L T O Z O K
FO DEFW BUFFER ; pufferből kiolvasandó karakter címe
FI DEFW BUFFER ; pufferbeli első szabad hely
BS DEFW BUFFER ; puffer kezdete
BE DEFW BUFFER+MAX ; puffer vége
LAST DEFS 1 ; a futó program számára
FLAG DEFB 0 ; első két bit
CLOCK DEFS 1 ; óra
BL DEFB 0 ; a puffer pillanatnyi hossza
ERT1 DEFB 35 ; első várakozás
ERT2 DEFB 5 ; ismételt várakozás
; billentyűzet leolvasása
; (rendszerfüggő)
READ EQU $ ; billentyűzet leolvasása
CALL KYSCAN ; key scan ROM rutin (spectrum)
LD A,D ; ha nincs
AND E ; billentyű akkor
INC A ; DE=0FFFFh
RET Z ; ha nincs billentyű
LD D,0 ; DE:=a billentyű kódja (nem ASCII!)
LD HL,TBL ; táblázat kezdete
ADD HL,DE ; az ASCII kód címe
    
```



0 mód. Ebben az üzemmódban a megszakítást kérő eszköz bármilyen utasításkódot tehet az adatbuszra, a központi egység végrehajtja azt.

1. mód. A megszakításkérés elfogadásakor a központi egység egy RST 38H utasítást hajt végre.

2. mód. A megszakítást kérő eszköz által az adatbuszon küldött bájt alkotja egy cím alsó bájtját, az I regiszter a cím felső bájtját. Az így kapott címről veszi a processzor a megszakításrutin kezdőcímét.

Ezeket a módokat az IM (interrupt mode) utasításokkal lehet kiválasztani. Az első két mód elég egyszerű. Nézzünk meg egy konkrét példát a harmadik, azaz kettes üzemmód használatára.

Tegyük fel, hogy az I regiszterbe 9-et töltöttünk, a megszakításkérő eszköz pedig 255-öt küld az adatbuszon. Ekkor a cím  $9 \times 256 + 255 = 2559$ . Ha most ezen a címen 105, a következőn, a 2560-as címen 254 található, akkor a megszakításrutin kezdőcíme  $254 \times 256 + 105 = 65129$ .

Mire való a megszakítás? Nos, ez eléggé gépfüggő. Tegyük fel, hogy 20 ms-onként történik megszakításkérés. Ennek segítségével megvalósíthatunk egy belső órát a gépben, készíthetünk BREAK vizsgálatot, amellyel minden, a megszakítást le nem tiltó gépi kódú program megállítható, továbbá ismétlő billentyűzet-leolvasást csinálhatunk.

## 6.1 feladat

Valósítsunk meg a gépben 2-es megszakítás üzemmódban egy órát, amely az ötvened másodperceket, másodperceket, perceket és órákat számlálja!

Mivel 20 ms-onként, azaz ötvened másodpercenként kerül a vezérlés a megszakításrutinra, ezért ha minden megszakítás alkalmával egy bájtot növelünk, akkor az az ötvened másodperceket fogja mutatni. Amikor eléri az ötvenet, azaz letelt egy másodperc, akkor nullázni kell, és egy másik bájtot, a másodpercszámlálót kell növelni. Amikor az eléri a hatvanat, akkor a következővel kell hasonló módon eljárni, és így tovább.

A 6.1 program két részből áll. Az első beállítja az I regisztert és az üzemmódot. Ez a rész bárhova kerülhet a tárba. A második rész viszont csak arra a bizonyos címre kerülhet, amelyet meghatározunk. A mai példánkban a 65129-es címre. Ez megfelel a Spectrum ROM-jának és hardverjének. A bájtok egymás utáni növelését úgy oldották meg, hogy ha már nem kell növelni a következő bájtot, akkor a STEP szubrutin a visszatérési címet kiveszi a veremből, így nem minden CALL után tér vissza a program.

Nagyon fontos, hogy meg kell őrizni minden regiszter értékét, hiszen a főprogram nem tud gondoskodni azok elmentéséről.

A második feladat, a BREAK gomb vizsgálata rendkívül egyszerű. Tulajdonképpen azt kell tenni, hogy ha a gomb meg van nyomva, akkor nem tér vissza a program oda, ahol a megszakítás előtt

volt, hanem elindítja a gép alaprendszerét. Arra kell ügyelni, hogy a BREAK gomb hosszan le lehet nyomva, és bár egy másodpercig tartó lenyomásnál a megszakításrutin ötvenszer fogja érzékelni, mégis, csak egyszer szabad rá a fent leírt módon reagálni.

## 6.2 feladat

Valósítsunk meg a gépben 2-es megszakítás üzemmóddal egy billentyűzetleolvasást, amely fogadja a billentyűt, amikor a felhasználó azt lenyomja, majd ha lenyomva tartja, akkor 0,7 másodperc múlva ismét fogadja, majd ezután minden 0,1 másodpercben mindaddig, amíg a felhasználó fel nem engedi a billentyűt. Ha az éppen futó programnak nincs szüksége bemenetre, akkor tárolja a billentyűértékeket.

Ha minden másodpercben 50 megszakítás történik, akkor minden megszakítás esetén csökkentve egy számlálót, egy belső órát kapunk, amelyet ha egy billentyű első érzékelésekor 35-re állítunk, akkor 0,7 másodperc múlva lesz nulla, ha 5-re, akkor 0,1 másodperc múlva. Azt, hogy egy billentyűt először érzékel-e a megszakítás, úgy tudjuk vizsgálni, hogy ha nincs billentyű lenyomva, akkor egy bitet, a programban a flag változó nullás bitjét, nullára állítjuk, és ha van lenyomott billentyű, akkor 1-re. Így ha ez a bit nulla, de van lenyomott billentyű, akkor azt először érzékeli a megszakításrutin. Ekkor a billentyű értékét eltesszük egy pufferbe, az órát pedig 35-re állítjuk. Ha a bit nem nulla és van lenyomott billentyű, akkor azt csak akkor tesszük be a pufferbe, ha az óra nulla, ellenkező esetben ugyanis nem járt le a 0,7 vagy 0,1 másodperces várakozás.

A megszakításrutin másik feladata az, hogy kiszolgálja az éppen futó programot, ha az jelzi, hogy szüksége van egy billentyűértékre. Ha a gépben futó programnak egy értékre van szüksége, akkor a flag változó egyes bitjét 1-re állítja. A megszakításrutin innen tudja, hogy a pufferből egy értéket át kell tennie a megfelelő helyre (LAST változó), majd amikor az áthelyezés megtörtént, akkor a bitet nullázza, jelezve, hogy érvényes adat van a változóban. A futó program számára ez azt jelenti, hogy SET utasítással 1-re állítja a bitet, majd addig vár, amíg a bit nem törlődik.

Nézzük meg részletesen a puffert! Két műveletet kell megvalósítanunk.

1. Egy bájt beírása a pufferbe,
2. Egy bájt kiolvasása a pufferből.

Mindezt úgy, hogy mindig azt a bájtot kell kiolvasnunk, amit legrégebben írtunk be. Az ilyen szervezésű táratokat, puffereket FIFO-nak (first in, first out = első be, első ki) hívják. Ellentétben a vermekkel, amelyekből azt vesszük ki először, amit utoljára tettünk be. Ezeket LIFO-nak (last in, first out = utolsó be, első ki) hívják.

Egy FIFO puffer megvalósítása kétféle lehet. Az egyik, hogy mindig ugyanoda helyezzük a beírandó bájtot, és a többi, pufferben lévő bájtot LDIR vagy LDDR utasítással arrébb helyezzük. Ez a megoldás lassú a felesleges adatmozgatás miatt.

Másik megoldásként két mutatót használva írjuk és olvassuk a puffert. Ennek a megoldásnak az a hátránya, hogy ekkor a puffer mozog a tárban, sőt egy rosszul megírt program esetén végigvándorol az egész táron, ami menthetetlenül a rendszer lefagyását okozza. Ennek elkerülésére módosítjuk a két mutató mozgását, olyan módon, hogy nem növeljük, illetve csökkentjük őket vég nélkül, hanem amikor bármelyik eléri a pufferre szánt terület végét, akkor a puffer elejére mutató értéket töltjük bele. Regisztrálni kell azt is, hogy van-e még hely a pufferben, mert így az egyik mutató utolérheti a másikat, illetve olvasáskor azt, hogy van-e a pufferben kiolvasható adat.

```

LD A, (HL) ;A:=ASCII kód
RET ;
;T A B L A Z A T
TBL DEFN 'BMY65TGV' ;
DEFN 'NJU74RFC' ;
DEFN 'MKI83EDX' ;
DEFB SSH ;symbol shift
DEFM 'LQ92WSZ' ;
DEFM ;szóköz
DEFB CR ;carrige return
DEFM 'PO10A' ;
DEFB CSH ;caps shift
BUFFER DEFS MAX ;puffer
KYSCN EQU 2BEH ;spectrum ROM rutin
CSH EQU 0FFH ;
SSH EQU 0FEH ;
CR EQU 0DH ;
END
    
```



## BASIC és gépi kód

Legutóbb a gépi kódú programokat mentő rutin működését néztük meg közelebbről, és szó volt egy meglepő felfedezésről. Most néhány újabb gépi kódú utasítás ismertetése után egy újabb — számomra kellemetlen — felfedezésről számolok be.

### A számláló utasítások

Az ebbe a csoportba tartozó utasítások az indexregiszterek vagy valamelyik memóriabájt tartalmát növelik, illetve csökkentik egyel. Nevük az angol increment (növekedés), illetve decrement (fogyás) szó első betűiből és indexregiszterekre vonatkozó utasítások esetén a regiszter nevéből állnak. Az INC, INX és INY a növelő, a DEC, DEX és DEY a csökkentő utasítások. Az A regiszter tartalmát sajnos nem lehet hasonló utasítással egygel növelni vagy csök-

kenteni, de viszonylag egyszerűen megoldható.

A számláló utasítások az állapotregiszternek az N és a Z bitjét állítják be, a korábban ismertetett módon. A C (átvitel) bit állapotát nem változtatják, tehát ennek a figyelése nem tájékoztat arról, hogy például növelésnél elértük-e a \$FF után a \$00 értéket. (A C bitről eddig még nem volt szó, az eddig ismertetett utasítások egyike sem változtatja meg az állapotát.)

### A közvetett JMP utasítás

A JMP utasítás ismertetésekor csak az abszolút címzési módú változatról írtam. Nem kevésbé fontos a közvetett (indirekt) címzési módú JMP utasítás sem. Működése egy kicsit bonyolultabb az előbbiéénél. Az utasítás operandusa egy tárcím, de az ugrás

nem erre a címre történik. Az operandus által megadott helyen egy további kétbájtos címinformáció található, a már többször említett alsó bájt/felső bájt sorrendben. Az ugrás erre a címre történik.

A közvetett JMP utasítás gépi kódja \$6C, az assembly formájú leírásban úgy különböztetjük meg az egyszerű abszolút címzésűtől, hogy az operandust zárójelbe tesszük. Például: JMP (\$0300).

### Az újabb felfedezés

Legutóbb írtam arról a felfedezéséről, hogy kazettás tárolóról történő programbetöltés esetén a gépi kódú program egy egyszerű LOAD parancs hatására az eredeti helyére töltődött. Hozzátettem, hogy ilyesmiről eddig még sehol sem olvastam. Az újabb felfedezésem az, hogy olvashattam volna, csak egy kicsit jobban oda kellett volna figyelnem. A Commodore 64-es belső felépítése című könyv 107. oldalán — melyet korábban többször is átolvastam — a következőt találtam: »LOAD „NÉV”, 1 — BASIC program relatív betöltése, ill. a 3-as típusú gépi kódú program abszolút betöltése«.

Ez magyarázatot ad a múltkori „csodálatos felfedezésemre”. Elnézést kérek. A VC20-ra változatlanul a korábban leírtak érvényesek.

BARNA LÁSZLÓ

A 6.2 programban WRTB szubrutin az akkumulátorban megadott bájtot írja be a pufferbe, ha van hely. Az RDB szubrutin egy bájtot olvas ki a pufferből, illetve nulla akkumulátortartalommal tér vissza, ha üres a puffer, ugyanis nulla ASCII kódú karakter nincs.

Az IT rutin először csökkenti az órát, majd megvizsgálja, hogy a futó program számára kell-e érvényes adat. Ha kell, akkor megpróbálja a pufferből kiolvasni, és ha a puffer nem üres, akkor a kiolvasott értéket elteszi a LAST változóba, majd törli a flag 1-es bitjét.

Ezután, illetve ha a puffer üres, vagy ha nem kellett LAST változót állítani, elkezdődik a rutin második része. A READ rutin leolvassa a billentyűzetet, és a lenyomott billentyű kódjával tér vissza. Ez a rész meglehetősen rendszerfüggő. Ha nincs lenyomott billentyű, akkor a visszaadott érték nulla, és ekkor a futás a NOKEY címkénél folytatódik. Ha van lenyomott billentyű, akkor megnézi, hogy az előző megszakításnál észlelte-e a program billentyűt, azaz új billentyű lenyomásáról van-e szó, vagy ismételni kell a billentyűt. A program elteszi az új billentyűértéket a pufferbe, jelzi a flag változó nullás bitjében, hogy volt billentyű ebben a megszakításban, majd a CLOCK változóba átmásolja ERT1 változó értékét, amely alaphelyzetben 35.

Ha nem új billentyűről van szó, akkor megvizsgálja, hogy az óra nulla-e. Ha nem, akkor a billentyűértéket figyelmen kívül kell hagyni. Ha nulla, akkor elteszi az értéket a pufferbe, az órát pedig ERT2-nek megfelelően állítja be. Ha nem volt billentyű lenyomva, akkor a flag változó nullás bitjének törlése után visszatér.

Az RDB (read buffer) szubrutin először megnézi a puffer hosszát (BL=buffer length), ugyanis ha ez nulla, akkor üres a puffer. Ha nem, akkor BL-t csökkenti, a FO (first out) által mutatott címről kivesz egy bájt, majd lépteti FO-t a puffer következő elemére.

A felhasználói program kiszolgálását úgy is meg lehet oldani, hogy rendelkezésére bocsátunk egy szubrutint, amely a pufferből kiolvas egy értéket. Az út járható, de nagyon vigyázni kell, mert ekkor a megszakításrutin és az éppen futó program ugyanazon az adathalmazon végez változtatásokat.

Elképzelhető, hogy a puffer tele van, amikor olvasni akarunk belőle. FO-t léptetjük, BL-t csökkentjük, és kiolvasunk a pufferből egy kódot. Ha nem tiltjuk az olvasás idejére a megszakítást, meg-

történhet, hogy FO léptetése és BL csökkentése után egy megszakítás a még ki nem olvasott bájttra teszi az új billentyűértéket, hiszen FO és BL értéke alapján ez a hely már szabad, és a megszakítás lefutása után ezt az értéket fogja a programunk kiolvasni.

A WRTB (write buffer) szubrutin is BL vizsgálatával kezdődik, de most azt nézzük, hogy nem telt-e meg a puffer. Ha nem, akkor BL-t növeljük, a FI (first in) által mutatott címre letesszük az értéket, és léptetjük FI-t.

Ennek a billentyűzetleolvasásnak van egy hátránya. Ha gyorsan gépelünk, és egy billentyűt úgy nyomunk le, hogy az előző felengedése óta még nem volt megszakítás, akkor az így lenyomott billentyűt a rutin ismételt billentyűnek tekinti.

Próbáljuk meg átírni a rutint úgy, hogy egy billentyűt akkor tekintsen újnak, ha az előző megszakításban nem volt billentyű lenyomva, vagy ha más billentyű volt lenyomva!

A megszakításrutin változói:

ERT1	1 bájt	billentyűismétlésnél az első várakozás hossza
ERT2	1 bájt	billentyűismétlésnél a második, harmadik, ... várakozás hossza
CLOCK	1 bájt	óra
FLAG	1 bájt	alsó két bite használt 0. bit ismételt billentyű jelzése 1. bit billentyűre várakozás, illetve érvényes billentyű jelzése
BL	1 bájt	a pufferben lévő adatok száma
FI	2 bájt	a puffer első szabad bájtjára mutat
FO	2 bájt	a puffer első kiolvasandó bájtjára mutat
BE	2 bájt	a puffer végére mutat
BS	2 bájt	a puffer elejére mutat

Ha bármilyen formában átírjuk a megszakításrutinokat, de használni akarjuk a rendszer szolgáltatásait, akkor vigyázni kell arra, hogy amit az eredeti rutin elvégez, azt elvégezze a mi megszakításrutinunk is. Ezt nagyon egyszerűen megtehetjük, ha ez EI és RET utasítások helyére egy JP utasítást írunk, amely az eredeti megszakításrutint indítja.

VERHÁS PÉTER



## COMMODORE 64

## Adalék a rendezéshez

A rendezőprogramok nagy része úgy dolgozik, hogy a szomszédos elemeket kicseréli a feltételnek megfelelően. Amikor már sokalltam az időt a cserélgetésre, elgondoltam azon, hogy én ezt hogy csinálnám. Végignéztem tehát a számokon és megkerestem a legkisebbiket, azt kicseréltem az első helyre, és tovább kerestem a maradék számok között. Így a hasonló bonyolultságú programoknál — jelen esetben a buborékmódszernél — négyszer gyorsabban értem el az eredményhez. Remélem, hogy el-

járásom közreadásával sok rendezésre fecserélt időt takarított meg hasonló gondokkal küszködő társaimnak.

"B" mezőből

"A" számú rekordot tartalmazó

"F" tömb növekvő sorba rendezése

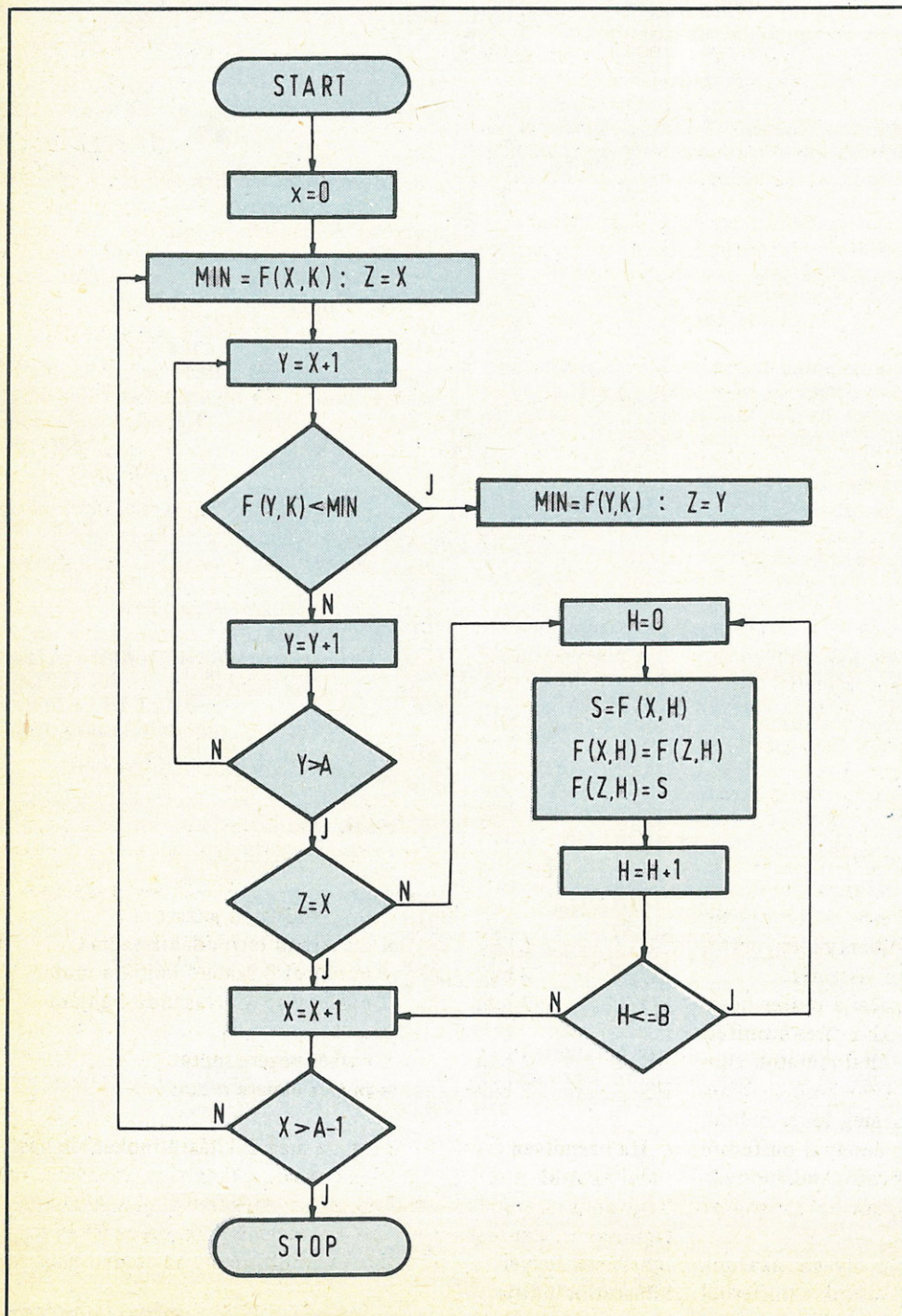
"K" kulcsmező szerint

Program Commodore 64-re

```
1 REM
2 REM RENDEZÉS
3 REM
```

```
10 INPUT "KULCSMEZO";K
20 FOR X=0 TO A-1
30 MIN=F(X,K):Z=X
40 FOR Y=X+1 TO A : IF F(Y,K)
< MIN THEN MIN=F(Y,K):Z=Y
50 NEXT Y
60 IF ZZ=X THEN 80
70 GOSUB 100
80 NEXT X
90 RETURN
100 FOR H=0 TO B:S=F(X,H):
F(X,H)=F(Z,H):F(Z,H)=S:NEXT:
RETURN
```

A rendezés folyamatábrája



Változók

A a rekordok száma

B mezőszám a rekordon belül

K a kulcsmező indexe a rekordban

X futó index, a rendezettség alsó határa

Y a vizsgált rekord indexe

Z a legkisebb kulcsértékű rekord indexe

A program magyarázata

10 Megkérdezi, melyik mező szerint rendezzen  
 20 Ciklus 0-tól az A-1 rekordig  
 30 Minimumértéknek felveszi az első rekordot Z=X rekordszámmal  
 40 A rekordok között megkeresi a legkisebb kulcsértékűt, Z=Y rekordot  
 60 Ha Z=X, akkor az első rekord kulcsértéke a legkisebb, és nem kell kicserélni  
 70-100 Ellenkező esetben Z <> X, akkor a Z számú rekordot ki kell cserélni az X számú rekorddal és az X értékét eggyel növelni.

GYÓRI IMRE

## SYMPOSIUM ON MOLECULAR ELECTRONICS AND BIOCOMPUTERS

Molekuláris elektronika  
és bioszámítógép szimpózium  
Budapest,  
1987. augusztus 24-27.

Felvilágosítást ad  
az Eötvös Loránd  
Fizikai Társulat  
1061 Budapest,  
Anker köz 1-3.  
Telefon: 227-040



# Bináris keresőalgorithmus C64-re

Különösen adatbázis-kezelési problémák megoldása során gyakran van szükség adathalmazok valamely adathalmazból való visszakeresésére. A kínáló leggyorsabb megoldás, hogy az adathalmaz minden elemét, az elsőtől az utolsóig megvizsgáljuk, összehasonlítva azokat a keresett elemmel.

Ez azonban különösen BASIC-ben, de más magas szintű programnyelven is meglehetősen lassú. Például egy ezer elemet tartalmazó adatbázis utolsó egyedét néhány perces türelmetlen várakozás után találjuk csak meg.

Természetesen van ennél jobb keresési

algorithmus is, nem is egy. A bináris keresési eljárás például még BASIC-ben megírva is az előbbinek töredékére csökkenti a hozzáférési időt. Ilyen algoritmus BASIC programja található például English—Szczepanowski: A VC—1541-es lemezegység programozása (DATA BECKER — NOVOTRADE 1985.) c. könyve 61. oldalán. (Sajnos a könyvnek ez a programja is hibás!) Helyette egy egyszerű kis BASIC programot ismertetek a probléma megoldására (1. lista).

Az M% változóban az adatfájl mérete

```
10 INPUT "A KERESETT SZO";N$(0)
20 S=INT(LOG(M%)/LOG(2)+1)
30 AHZ=1
40 FHZ=MZ+1
50 KZ=(AHZ+FHZ)/2
60 IF N$(0)=N$(KZ) THEN PRINT N$(KZ):GOTO 10
70 IF N$(0)<N$(KZ) THEN FHZ=KZ:GOTO 90
80 AHZ=KZ
90 S=S-1:IF S=0 THEN PRINT "NINCS ILYEN":GOTO 10
100 GOTO 50
```

1. lista

2. lista

```
100 MERET = $FB
110 FH = $FD
120 AH = $A9
130 CIKL = $FF
140 AT = $57
150 FEL = $59
160 SZ01 = $5B
170 SZ02 = $5D
180 HOSSZ = $5F
190 KITEVO = $4E
200 KOZEP = $4F
210 CHRGET = $73
220 CHROUT = $FFD2
230 FRMEVL = $AD9E
240 ADDRESS = $B08B
250 FACADR = $B7F7
260 LINOUT = $AB1E
270 CIM = $47
280 M = $14
290 N0 = $26
300 N1 = $28
310 NN = $A7
320 MAS = $CFF1
330 * = 49152
340 ,
350 JSR CHRGET ;ELSO VESSZO ATUGRASA
360 JSR ADDRESS ;A HIVASBAN MEGJELOLT
370 LDA CIM ;TOMBELEM CIMENEK
380 STA N0 ;N0-BA OLVASASA
390 LDA CIM+1
400 STA N0+1
410 LDA N0+1
420 STA N1+1
430 CLC ;EBBOL A KOVETKEZO ELEM
440 LDA N0 ;CIMENEK KISZAMITASA
450 ADC #3
460 STA N1
470 BCC 0
480 INC N1+1
490 JSR CHRGET ;2. VESSZO ATUGRASA
500 JSR FRMEVL ;A TOMBNERET BEOLVASASA
510 JSR FACADR
520 LDA M
530 STA MERET
540 LDA M+1
550 STA MERET+1
560 LDA N0 ;AT=N0
570 STA AT
580 LDA N0+1
590 STA AT+1
600 LDY #1
610 LDA <AT>,Y ;SZ01-BEN A KERESETT
620 STA SZ01 ;SZ0 LEIROBOL KIEMELT
630 INY ;ABSZOLUT CIME
640 LDA <AT>,Y
650 STA SZ01+1
660 LDA #15 ;A STRINGEK
670 STA HOSSZ ;HOSSZA -> HOSSZ-BA
680 LDA #1
690 STA KOZEP ;KOZEP=1
700 LDA #0
710 STA KITEVO ;KITEVO=0
720 STA KOZEP+1
730 HATV INC KITEVO ;KITEVO=KITEVO+1
740 ASL KOZEP
750 ROL KOZEP+1 ;KOZEP = KOZEP * 2
760 SEC
770 LDA KOZEP ;IF KOZEP < MERET
780 SBC MERET ;THEN MEG EGY CIKLUS
790 LDA KOZEP+1
800 SBC MERET+1
810 BCC HATV ;UGRAS HATV-RA
820 LDA MERET ;FH=MERET
830 STA FH
840 LDA MERET+1
850 STA FH+1
860 LDA #0
870 STA AH ;AH=0
880 STA AH+1
```

```
890 STA CIKL ;CIKL=0
900 HUOK INC CIKL ;CIKL=CIKL+1
910 LDA KITEVO
920 CMP CIKL
930 BMI HID
940 CLC ;KOZEP=AH+FH
950 LDA FH
960 ADC AH
970 STA KOZEP
980 LDA FH+1
990 ADC AH+1
1000 STA KOZEP+1
1010 LSR KOZEP+1 ;KOZEP=KOZEP/2
1020 ROR KOZEP ;KOZEP=FELEZO SORSZAM
1030 LDA KOZEP+1 ;NN=KOZEP - 1
1040 STA NN+1
1050 SEC
1060 LDA KOZEP
1070 SBC #1
1080 STA NN
1090 BCS FELEZ
1100 DEC NN+1
1110 LDA N1 ;AT=N1
1120 STA AT
1130 LDA N1+1
1140 STA AT+1
1150 FELEZ LDA NN
1160 STA FEL ;FEL=NN-HEZ TARTOZO CIM
1170 LDA NN+1
1180 STA FEL+1
1190 ASL FEL
1200 ROL FEL+1
1210 ASL FEL
1220 ROL FEL+1
1230 SEC
1240 LDA FEL
1250 SBC NN ;FEL=3*NN+AT
1260 STA FEL ;A KOZEPSO ELEM
1270 LDA FEL+1 ;LEIROJA CIMENEK
1280 SBC NN+1 ;KISZAMITASA
1290 STA FEL+1
1300 CLC
1310 LDA FEL
1320 ADC AT
1330 STA FEL
1340 LDA FEL+1
1350 ADC AT+1
1360 STA FEL+1
1370 LDY #1 ;A MINDENKORI KOZEPSO SZ
1380 LDA <FEL>,Y ;ABSZOLUT CIME SZ02-BE
1390 STA SZ02
1400 INY
1410 LDA <FEL>,Y
1420 STA SZ02+1
1430 LDY #0 ;OSSZEHASONLITAS
1440 HSNL LDA <SZ01>,Y
1450 CMP <SZ02>,Y
1460 BCC KISEBB ;HA A KERESETT KISEBB
1470 BEQ TOVABB ;HA EGYENLO
1480 JMP NAGYBB ;HA NAGYDABB
1490 HID BMI SEHOL
1500 TOVABB INY
1510 CPY HOSSZ ;IF Y=HOSSZ
1520 BEQ KIIR ;THEN KIIRATAS
1530 JMP HSNL ;KULONBEN VISSZAUGRAS
1540 KISEBB LDA KOZEP ;FH=KOZEP
1550 STA FH
1560 LDA KOZEP+1
1570 STA FH+1
1580 JMP HUOK
1590 NAGYBB LDA KOZEP ;AH=KOZEP
1600 STA AH
1610 LDA KOZEP+1
1620 STA AH+1
1630 JMP HUOK
1640 SEHOL LDX #00 ;A "NINCS ILYEN"
1650 LDA TXT,X ;UZENET KIIRATASA
1660 JSR CHROUT
1670 INX
1680 CPX #0B
1690 BNE T
1700 RTS
1710 TXT .TEXT "NINCS ILYEN"
1720 KIIR LDY #0 ;A MEGTALALT SZO
1730 KIR1 LDA <SZ02>,Y ;MAS-BA MASOLASA
1740 STA MAS,Y
1750 INY
1760 CPY HOSSZ
1770 BNE KIR1
1780 LDA #MAS<
1790 LDY #MAS>
1800 JSR LINOUT ;KIIRATASA
1810 RTS
1820 .END
```

ZEILEN: 173 SYMBOLE: 37 FEHLER: 0



van. Az N\$ tömbbe kerül beolvasásra az adathalmaz.

A bináris keresés alkalmazhatóságának előfeltétele, hogy rendezett adathalmazunk legyen. Ez azt jelenti, hogy például nagyság szerint, vagy éppen ábécébe kell rendeznünk az adathalmaz elemeit. A rendezés hatékony megvalósítása viszont legalább akkora probléma, mint a keresés. Erre viszont Erdős Iván: COMMODORE 64 ASSEMBLY c. könyvében találunk megbízható algoritmusokat.

A program gondolatmenete a következő.

1. Felezzük meg rendezett adathalmazunkat.

2. A keresett elemet hasonlítsuk össze a felező elemmel.

3. Ha egyezik, akkor a keresésnek vége.

4. Ha kisebb, akkor az adathalmaz alsó felébe, különben a felsőbe esik a keresett elem.

5. Ismételjük meg az eljárást az 1. lépéstől úgy, hogy a felezés mindig a maradék halmazra vonatkozzon.

Ezzel az eljárással egy M elemű tömb bármely eleme legfeljebb  $S = \text{INT}(\text{LOG}^2 M + 1)$ , azaz  $S = \text{INT}(\text{LOG}(M)/\text{LOG}^2 + 1)$  lépésben elérhető. Így például az 1000. elem ezer kereséslépés helyett tizenegy lépésben elérhető. (S a lépésszám.)

A következőkben egy assembly nyelvű bináris keresőalgoritmust szeretnék bemutatni, amely nagy adatbázisok hatékony kezelésében nyújthat segítséget. A program (2. lista) logikái megfelelője a közölt BASIC programnak, s így egy karakteres tömb valamely elemének megkeresésére szolgál. Hívása BASIC-ből: SYS 49152, N\$(0), M + 1, ahol N\$(0) a karakteres tömbnek az az eleme, amelybe a keresett szót kell elhelyezni. M + 1 a tömb elemeinek száma, ha a 0. elemet is számításba vesszük.

Az egyszerűség kedvéért a tömbelemek első 15 karakterét használjuk fel azonosításra. Ez azonban tetszőlegesen választható. Természetesen a tömböt előzetesen e szerint kell feltölteni. A tömbelemek egységes hosszát BASIC-ből az

$$N$(I) = \text{LEFT}\$(N$(I) + " ", 15)$$

értékadással érhetjük el. A kereső szót is ilyen alakra kell hozni:

```
INPUT K$
N$(0) = LEFT$(K$ + " ", 15)
A program működése
```

350—510 A SYS paraméterek átvétele: a 0. és az 1. tömbelem címének kiszámítása az ADRESS interpreter rutin segítségével, és a tömbméret beolvasása.

520—590 Munkaváltozók beállítása.

600—650 N\$(0) leírójából a keresett szó abszolút címét adó 2. és 3. bájtot SZO1-be töltjük. (A leírók hárombájtosak, az első, 0. bájtt a sztring hosszát, a második és harmadik, azaz az 1. és 2. sorozámú bájtt a sztring szövegének a címét mutatja.)

660 Itt kell beállítani a szóhosszt.

690—810 Megkeressük 2-nek M-nél kisebb, legnagyobb hatványát.

820—1220 A tömb felezése, részhalmozokra bontása.

1230—1360 A felező elem leírója címének kiszámítása.

1370—1420 A felező elem abszolút címének kiemelése a leíró 1. és 2. bájtból és SZO2-be tétele.

1430—1530 A keresett szó: N\$(0), és a

mindenkori középső szó összehasonlítása.

1540—1630 Alprogramok, melyek a részhalmozok határait állítják be.

1640—1710 A "NINCS ILYEN" üzenet kiírása a CHROUT KERNAL rutin segítségével.

1720—1810 A megtalált szót \$CFF1-től \$CFFF-ig terjedő területre másoljuk, és a LINOUT interpreter rutin segítségével a képernyőre írjuk. (Erre más lehetőség is van.)

A 930—1490—1640 sorok ugrási hidat képeznek. A 930-as sorban kijelölt ugrási cím (SEHOL címkével az 1640-es sorban) 128 bájtnál távolabb van. Ezért a közvetlen áthidalás nem lehetséges.

A megtalált szó képernyőre íratása itt természetesen egyszerűsítés a program kedvéért. Gyakorlatban általában nem ez a feladat.

Tételezzünk fel például egy 1300—1500 rekordot tartalmazó relatív fájlt. Tartalmazza a rekordok például dolgozók személyi adatait. Legyen az első mező a dolgozó neve. Tekintsük a név azonosításához elegendő első 15 karaktert keresési kulcsnak.

A relatív fájl egészének rendezettségét biztosítani körülményes, de a kulcsokból egy állandóan rendezett indexállományt hozhatunk létre egy tömbváltozóban. Ez a nevek mellett a rekord sorszámát tartalmazza.

A keresés erre a rendezett halmazra vonatkozik. Ha megtaláltuk a keresett nevet, kiolvassuk belőle a relatív rekord sorszámát, és az ADRESS rutin segítségével például egy R\$ nevű változóba másoljuk. Az R\$-ből BASIC-ben átvesszük a sorszámot, s a relatív fájlkezelésben ismert eljárás szerint meghívjuk a kívánt rekordot.

Ezzel kapcsolatban azonban ki kell elégtennünk két követelményt:

— R\$-et a BASIC programban előzetesen definiálni kell. Például: R\$ = "". Majd egy látszólagos műveletet kell vele végezni, így: R\$ = R\$ + " (legalább 20 szóköz) ".

— A rutin meghívásakor a SYS-ben nem elegendő csak az indexállomány nevét és méretét megadni, hanem R\$-re is hivatkozni kell.

Végezetül néhány mérési adat a futásidők összehasonlítására. Az időmérést egy 1300 elemet tartalmazó tömbben való keresésnél végeztem.

Soros keresésnél az elérési idő a keresett elem sorszámával nagyjából arányosan nő. Az ezrediken túli elemek elérése memóriában tárolt adatoknál 1 és 2 perc között van. Relatív fájlban való soros keresés esetén az ezrediken túli elemek megtalálása 6 percnél is hosszabb időt vesz igénybe.

A bemutatott BASIC programmal történő kerestetés a memóriában levő tömbben 18 és 25 másodperc közötti elérési időt ad.

Az assembly rutin használata esetén a meghívás és a sikeres visszatérés közötti idő 0,025 és 0,033 másodperc közé esik.

Ha az assembly rutint egy relatív fájlhoz kapcsolódó indexfájlban való kerestetésre használom, a meghívás és a relatív fájl rekord formázott képernyőre írása közötti idő 6—8 másodperc.

DR. JÁNOSA ANDRÁS

# Programozástechnika

## Papírszélesítés

Seikosha GP—50 S és ZX-printer tulajdonosok spectrumosok használhatják jól ezt a kis programot.

Az említett printerek elég keskeny papírra dolgoznak, általában csak 32 betűt tudnak egy sorba kiírni, 256 pixel szélességben. A Tasword típusú szövegszerkesztők a másik végletbe esnek: épp hogy felismerhető, de már nehezen olvasható sovány betűket használnak, így soronként 64 karaktert értek el.

Megpróbáltam átmenetet találni a 32 és a 64 között. A legcélszerűbbnek és a legérdekesebbnek a változó karakterszélesség látszott — gondoljunk csak az m és az i közötti különbségre. Így ugyan nehezebb szépen formázott szövegeket készíteni, de minden betű jól olvasható, és soronként átlagosan 50—60 karakter fér el.

A program a szöveget a grafikus képernyőre írja ki, az egyszerűség érdekében PLOT-tal, de még így is meglepően gyors. Ha a képernyő megtelik, COPY-val kimásolja a papírra, majd törli a képernyőt, és így tovább. Torzítás nélkül nyomtatja a lelógó írásjeleket és betűket is. Ennek módszerét a Primótól kölcsönöztem.

A programot eredetileg a SPECTRE-MAC nevű makrofeldolgozó fájljainak kiírásához készítettem, ezért csupán annak karaktereit ismeri fel, a b7 bitet nem veszi figyelembe:

0CH : kurzor home  
0DH : kurzor az aktuális sor elejére (Old-Line-CR)

1CH : 32 pixelenkénti tabulátor

1FH : kurzor le (soremelés)

20H—7FH : a Spectrum ASCII karakterei

A többi 00H és 1EH közötti karakterrel semmit nem csinál, még "?"-t sem nyomtat. A többi paramétert az 1. táblázat tartalmazza.

A program természetesen módosítható, például újabb vezérlőkarakterek felismerésére, a betűkészlet bővítésére. A már említett változó betűszélesség miatt a karakterkészlet azonban nem túl könnyen variálható. Az adatszerkezetet a 2. táblázat tartalmazza. Aki komolyabban akar bővítéssel foglalkozni, eljátszhat a disassemblerrel is, hisz maga a program alig száz sor, de — korlátozott számban — levélben szívesen elküldöm a forráslistát.

A szubrutin BASIC-ből is kényelmesen használható a Spectrum megfelelő csatornacímeinek átírásával:

```
LET a = PEEK 23631 + PEEK 23632 * 256 + 3 * 5
```

```
POKE a, 188: REM kezdőcím alsó bájtt
```

```
POKE a + 1, 252: REM felső bájtt
```

```
Ezután PRINT #3-mal és LPRINT-tel a keskenyített betűkkel írhatunk a nyomtatóra. A printer eredeti üzemmódját POKE a, 244: POKE a + 1, 9 segítségével állíthatjuk vissza. Az a változó értéke általában 23749, de csak ha nincs Interface—1 a géphez kapcsolva.
```

A program beírása a következőképpen történik. Az 1. listán látható segédprogrammal begépeljük a 2. listán látható hexadecimális számokat (az ellenőrző összegekkel együtt), az esetleges hibákat kijavítjuk,







# A VIC-20 tárcímei

A VIC-20-as géppel rendelkező olvasóink részére közreadjuk a VIC REVEALED, NICK HAMPSIRE, 1981. c. könyv alapján a legfontosabb tárcímeket, valamint a BASIC és a KERNAL ROM rutinokat.

POKE címek	Magyarázat
0-2	USR funkció helye
3-4	lebegőpontosból egész számmá konvertálás
5-6	egészről lebegőpontos számmá konvertálás
13	változó jelző: 0 = numerikus, 255 = sztringváltó
14	egész jelző: 0 = lebegőpontos, 128 = egész szám
69-70	aktuális változónév
71-72	aktuális változó kezdőcíme
147	load vagy verify jelző
152	nyitva levő fájlok száma
157	RUN (=0), vagy direkt (=80) módban dolgozik a gép
160-162	óra (TI\$)
183	az aktuális fájlnev hossza
198	a billentyűpufferben levő karakterek száma
199	inverzmód jelző: 0 = ki, 18 = be
201-202	kurzorhely (sor, oszlop sorrendben)
203	az éppen lenyomott billentyű kódja (=64, ha nincs billentyű)
204	kurzorvillogás-engedélyező jelző (0 = be, 1 = ki)
215	az utoljára lenyomott billentyű ASCII kódja
251-255	szabad zérólapos bájtok
256-511	processzor veremterülete
512-600	BASIC input puffer
601-610	logikai fájlszám-táblázat
611-620	egység számtáblázat
621-630	másodlagos cím vagy R/W táblázat
631-640	billentyűzetpuffer
650	billentyűismétlés jelző: 0 = csak a kurzorirányítók
651	128 = mindegyik billentyű ismétlés megtörténik
652	ismétlések közötti szünet
653	shift, CTRL, valamint C = jelzőbájt
657	shift + C = váltást engedélyező (0 = engedélyezve, 128 = tiltva)
673-767	szabad terület
<b>INDIREKT UGRÁSI CÍMEK</b>	
768-769	hibakiíró rutin (x regiszterbe a hibakódja)
770-771	BASIC melegindítás
772-773	tokenné alakítás
774-775	a memóriában levő soroknak lista formájú alakítása
776-777	következő utasítás végrehajtása
778-779	numerikus változó bevétele FAC-ba
780	akku átadása a SYS utasítás alatt
781	x regiszter átadása a SYS utasítás alatt
782	y regiszter átadása a SYS utasítás alatt
783	állapotregiszter átadása a SYS utasítás alatt

KERNAL MUTATÓK CÍMEI	
788-789	IRQ RAM vektora
790-791	BRK RAM vektora
792-793	NMI RAM vektora (RESTORE billentyű hatására)
794-795	OPEN logikai fájl vektora
796-797	CLOSE logikai fájl vektora
798-799	CHKIN KERNAL rutin vektora
800-801	CHKOUT KERNAL rutin vektora
802-803	CLRCHN KERNAL rutin vektora
804-805	CHRIN KERNAL rutin vektora
806-807	CHROUT KERNAL rutin vektora
808-809	STOP KERNAL rutin vektora
810-811	GETIN KERNAL rutin vektora
812-813	CLALL KERNAL rutin vektora
814-815	BASIC USR vektora
816-817	LOAD vektora
818-810	SAVE vektora
820-827	szabad terület
828-1020	kazettapuffer
1024-4095	3 k RAM bővítés helye
4096-7679	felhasználói BASIC terület képernyőmemória
7680-8191	8 k RAM/ROM bővítés 1
8192-16383	8 k RAM/ROM bővítés 2
16384-24575	8 k RAM/ROM bővítés 3
24576-32767	

**MEGJEGYZÉS:** Ha valamely bővítéssel dolgozik a gép, akkor a felhasználói RAM-terület elhelyezkedése a következőképpen módosul:  
 4096-4607 képernyőmemória  
 4608-?? felhasználói BASIC terület  
 37888-38399 színmemória

32768-36863	karaktergenerátor (ROM)
36864-37877	I/O BLOCK 1
36864	0-6 bitek állítják a képernyő vízszintes helyzetét
36865	a képernyő függőleges helyzetét állítja
36866	0-6 bitek a képernyő oszlopainak számát tartalmazzák
36867	0-6 bitek a képernyő sorainak számát tartalmazzák
36868	TV RASTER sorát adja
36869	0-3 bitek a karaktermemória helyzetét adják
	4-7 bitek a video eredeti kezdőcímét adják
36870	a fényceruza vízszintes pozícióját adja
36871	a fényceruza függőleges pozícióját adja
36874-36878	zenei bájtok
36879	0-2 bitek a keret színét állítják
	a 3. bit választ inverz vagy normál mód között
	4-7 bitek a háttér színét állítják
37136-37151	6522 chip PIA 1.
37152-37167	6522 chip PIA 2.

37888-38399	színmemória, ha van valamely bővítés
38400-38911	színmemória, ha bővítés nélküli a gép
38912-39935	I/O BLOCK 2
39936-40959	I/O BLOCK 3
40960-49151	8 k ROM bővítés
49152-57343	8 k BASIC ROM
57344-65535	8 k KERNAL ROM

## A BASIC INTERPRETER FONTOSABB RUTINJAI

Cím (hexadecimálisan)	Magyarázat
C000-C045	az elsődleges kulcsszavak ugrási címei
C046-C073	egyes funkciók ugrási címei
C074-C091	az operációs rendszer szerveződési és ugrási címei
C092-C192	BASIC kulcsszavak táblázata
C193-C2A9	BASIC üzenetek, legtöbbjük hibaüzenet
C38A-C3B7	FOR vagy GOSUB részére helyet készít a veremben
C3B8-C3FA	úr felnyitása a memóriában
C3FB-C407	teszt: a verem nem túl mély-e
C408-C434	a rendelkezésre álló memória méretének ellenőrzése
C435-	a megadott hibaüzenet kiírása, majd:
C474-C482	PRINT READY
C483-C532	új BASIC programsor a billentyűzetről
C533-C55F	programsorok átépítése a memóriában
C560-C57B	egy sor elfogadása a billentyűzetről
C57C-C612	kulcsszavak tokenjükre cserélése
C613-C641	megkeresi az adott sorszám-nak megfelelő sort
C642-	NEW végrehajtása, majd:
C660-C68D	CIR végrehajtása
C68E-C69B	BASIC mutatók visszaállítása programkezdetre
C69C-C741	LIST végrehajtása
C742-C7EC	FOR végrehajtása
C7ED-C819	BASIC kijelentés végrehajtása
C81D-C82B	RESTORE végrehajtása
C82C-C856	STOP és END végrehajtása
C857-C870	CONT végrehajtása
C871-C882	RUN végrehajtása
C883-C89F	GOSUB végrehajtása
C8A0-C8D1	GOTO végrehajtása
C8D2-C8EA	RETURN végrehajtása
C8EB-C905	DATA végrehajtása, azaz a következő utasításra ugrás
C906-C908	következő BASIC utasítás keresése
C909-C927	következő BASIC sor keresése
C928-C93A	IF végrehajtása
C93B-C94A	REM végrehajtása, azaz a következő BASIC sorra ugrás
C94B-C96A	ON végrehajtása
C96B-C9A4	fixpontos szám átvétele BASIC-ből
C9A5-CA1C	LET végrehajtása



CA1D—CA2B	egy ASCII szám hozzáadása akkumulátor 1-hez	D487—D4F3	ellenőrzi és felállítja a sztringet	DCCB	sá konvertálás
CA2C—CA7F	LET végrehajtásának folytatása	D4F4—D525	a sztringmutatókat felépítő szubrutin	DCCC—	INT végrehajtása
CA80—CA85	PRINT # végrehajtása	D526—D5BC	hulladékgyűjtő szubrutin (sztringeknél)	DCF2	
CA86—CA99	CMD végrehajtása	D5BD—D605	ellenőrzése annak, hogy jó-e a memóriában levő sztring	DCF3—DD7-	sztringből lebegőpontossá konvertálás
CA9A—CB1D	PRINT végrehajtása	D606—D63C	sztring összeállítása	D	új ASCII számjegy elrakása
CB1E—CB3A	a memóriából egy sztring kiírása	D63D—D679	sztringek összeláncolása	DD7E—	
CB3B—CB4C	HOME, CURSOR FEL, CURSOR LE, ... stb. karakterek kiírása	D67A—D6A2	sztring felépítése a memóriában	DDB2	
CB4D—CB7A	rossz input adatok kezelése	D6A3—D6DA	a nemkívánatos sztring törlése	DDB3—DDC-	sztringkonverziós állandó: 99999999,99999999
CB7B—CBA4	GET végrehajtása	D6DB—D6EB	a verem törlése	1	kiírás: IN; követi: BASIC sorszám
CBA5—CBBE	INPUT # végrehajtása	D6EC—D6FF	CHR\$ végrehajtása	DDC2—	
CBBF—CBF8	INPUT végrehajtása	D700—D72B	LEFT\$ végrehajtása	DDCD—	
CBF9—CC05	input elfogadása	D72C—D736	RIGHT\$ végrehajtása	DDDC	
CC06—CCFB	READ végrehajtása, egyes rutinjait a GET és INPUT is használja	D737—D760	MID\$ végrehajtása	DDDD—	TI\$ számmá konvertálása
CCFD—	üzenetek: EXTRA IGNORED, REDO FROM START	D761—D77B	sztringparamétereket kiveszi a veremből	DF10	
CC1D		D77C—D781	LEN végrehajtása	DF11—DF70	állandók a numerikus konverziók részére
CD1E—CD77	NEXT végrehajtása	D782—D78A	sztringmódról numerikus módra áll át	DF71—DF77	SQR végrehajtása
CD78—CD9D	adattípus ellenőrzése, REDO FROM START kiírása, ha kell	D78B—D79A	ASC végrehajtása	DF78—DFB3	energiafunkció ellátása
CD9E—CEF0	valamilyen numerikus vagy sztring kifejezés kiértékelése	D79B—D7AC	input bájt paraméter	DFB4—DFBE	NOT végrehajtása
CEF1—CEF6	zárójelen belüli kifejezés kiértékelése	D7AD—	VAL végrehajtása	DFBF—	állandók, sztringértékeléshez
CEF7—CEF9	a jobb (csukó) zárójel ellenőrzése	D7EA		DFEC	
CEFA—	a bal (nyitó) zárójel ellenőrzése	D7EB—D7F6	2 paraméter elrakása POKE vagy WAIT számára	DFED—E03F	EXP végrehajtása
CEFC		D7F7—D80C	lebegőpontossá fixpontos-sá konvertálás	E040—E089	EXPCONV sorozat-kiértékelő szubrutin
CEFD—CF07	a vessző ellenőrzése	D80D—D823	PEEK végrehajtása	E08A—E093	manipulációs állandó RND részére
CF08—CF0C	kiírás: SYNTAX ERROR és kilépés	D824—D82C	POKE végrehajtása	E094—E0F5	RND végrehajtása
CF0D—CF13	a leendő kiértékelésekhez szükséges funkciók felállítása	D82D—D848	WAIT végrehajtása	E0F6—E260	KERNAL ROM toldás a BASIC ROM-ban
CF14—CFA6	változónév keresése	D849—D84F	0.5 hozzáadása akkumulátor 1-hez	E261—E267	COS végrehajtása
CFA7—CFE5	azonosít és utalások felállítása	D850—D861	kivonás végrehajtása	E268—E2B0	SIN végrehajtása
CFE6—CFE8	OR végrehajtása	D862—D946	összeadás végrehajtása	E2B1—E2DC	TAN végrehajtása
CFE9—D015	AND végrehajtása	D947—D97D	akkumulátor 1. kiegészítése	E2DD—E30A	állandók a trigonometriai számításokhoz (pi, 2*pi, 0.25, ...)
D016—D07D	összehasonlítás: sztring vagy numerikus	D97E—D982	kiírás: OVERFLOW, és kilépés	E30B—E33A	ATN végrehajtása
D07E—D08A	DIM végrehajtása	D983—D9BB	egy bájt sokszorosítása	E33B—E377	állandók ATN számításokhoz
D08B—D112	változó elhelyezkedésének keresése	D9BC—D9E9	állandók: 1, SQR (0.5), SQR (2) stb.	E378—E386	RAM vektorok inicializálása
D113—D11C	vizsgálat: betű-e az ASCII karakter	D9EA—DA2F	LOG végrehajtása	E387—E3A3	CHARGET rutin másolata
D11D—D193	új BASIC változó elhelyezkedése a memóriában	DA30—DA58	szorzás végrehajtása	E429—E44E	üzenet: BYTES FREE, ****CBM BASIC V2 ****
D194—D1A4	a mutatókat elrendező szubrutin	DA59—DA8B	bitenkénti szorzás szubrutinja	E44F—E47B	vektorok inicializálása
D1A5—D1A9	32768 lebegőpontos alakban	DA8C—DAB6	akkumulátor 2-be tölt a memóriából	E47C—E4FF	üres terület
D1AA—D1D0	egy pozitív egész kifejezés kiértékelése	DAB7—DAD-	akkumulátor 1. és 2. tesztelése	KERNAL RUTINOK	
D1D1—D34B	egy elrendezés létrehozása vagy keresése	DAD4—DAE-	túlsordulás és alulcsordulás kezelése	E500—E504	A 6522 chip címével tér vissza
D34C—D37C	index alapján a tömb méretét számolja ki	DAE2—DAF8	szorzás 10-zel	E505—E509	a képernyő oszlop- és sor-számmal tér vissza
D37D—D390	FRE végrehajtása, majd:	DAF9—DAF-	10 lebegőpontos alakban	E50A—E517	olvassa/állítja a kurzor pozícióját
D391—D39D	konvertálás fixpontosból lebegőpontossá	DAFE—DB06	osztás 10-zel	E518—E580	I/O eszközök inicializálása
D39E—D3A5	POS végrehajtása	DB07—DB11	osztás végrehajtása	E581—E586	HOME funkció
D3A6—D3B2	ellenőrzi, hogy direkt módban van-e a gép, print ILLEGAL DIRECT	DB12—DBA1	osztás valamely számmal	E587—E5B4	a kurzor elmozdítása az aktuális sorban
D3B3—D3E0	DEF végrehajtása	DBA2—DBC6	akkumulátor 1-be tölt a memóriából	E5B5—E5C2	NMI rutin kezdete (RESTORE billentyű megnyomása esetén)
D3E1—D3F3	FN x szintaxisának ellenőrzése	DBC7—DBFB	akkumulátor 1-ből a memóriába tölt	E5C3—E5CE	6560 CHIP IN inicializálása
D3F4—D464	FN x kiértékelése	DBFC—DC/	akku 2-t akku 1-be másol	E5CF—E6C4	egy karakter elmozdítása a sorban
D465—D474	STR\$ végrehajtása	0B		E64F—E64E	egy sor bevitele, „RETURN”-ig
D475—D486	sztring vektor kiszámítása	DC0C—	akku 1-et akku 2-be másol	E742—E8E7	PRINT rutin
		DC1A		E8E8—E8F9	sormutató csökkentése és ellenőrzése
		DC1B—DC2A	akku 1. kerekítése	E8FA—E911	sormutató növelése és ellenőrzése
		DC2B—DC38	kiszámolja akku 1. előjelét	E912—E928	szín ellenőrzése
		DC39—DC57	SGN végrehajtása	E929—E974	képernyőkódról ASCII kódra konvertáláshoz táblázat
		DC58—DC5A	ABS végrehajtása	E975—EAA0	SCROLL rutinok
		DC5B—DC9A	összehasonlítja akku 1-et a memória tartalmával	EAA1—EB1D	IRQ rutinok
		DC9B—	lebegőpontossá fixpontos-	EB1E—EC45	billentyűzetfigyelés generálása
				EC46—EE13	billentyűzetmátrix



# ADOK-VESZEK-CSERÉLEK

- EE14—EEBF üzenet a soros buszon, figyelő állapotban levő egységnek
- EEC0—EEC4 másodlagos cím elküldése
- EEC5—EECD az egység figyelésének ki-kapcsolása, figyelő állapotba hozása után
- EECE—EEE3 beszélő állapotban levő egység másodlagos címe
- EEE4—EEF5 a puffer tartalma a soros buszra
- EEF6—EF03 üzenet kiküldése a soros buszra (nem beszélő egység)
- EF04—EF18 üzenet kiküldése a soros buszra (nem figyelő egységre)
- EF19—EFA2 egy bájt behívása a soros buszról
- EFA3—EFED NMI rutin folytatása
- EFEE—F035 átküldendő bájt
- F036—F173 NMI rutin, mely adatokat összegyűjt RS232 részére
- F174—F1E1 KERNAL üzenetek
- F1E2—F1F4 üzenet kiírása a képernyőre
- F1F5—F20D karakter bevétele egy csatornáról (GET módon)
- F20E—F279 karakter bevétele egy csatornáról (INPUT módon)
- F27A—F2CA kimenő karakter egy csatornára
- F2C7—F308 csatorna megnyitása input részére
- F309—F349 csatorna megnyitása output részére
- F34A—F3EE logikai fájl lezárása
- F3EF—F3F2 az összes fájl lezárása
- F3F3—F409 csatorna törlése
- F40A—F541 OPEN funkció
- F542—F674 LOAD RAM funkció
- F675—F733 SAVE funkció
- F734—F76F idő funkció
- F770—F77D STOP billentyű lenyomásának ellenőrzése
- F77E—F7AE hibakezelő rutin
- F7AF—F889 kazettafejléc keresése és olvasása
- F88A—F98D kazettaellenőrző rutin
- F98E—FABC kazettáról olvasó rutin
- FABD—FBE9 bájt kezelése kazettáról olvasás közben
- FBEA—FD21 kazettára író rutin
- FD22—FE90 hidegindítás
- FE91—FEA8 memóriaellenőrző rutin
- FEA9—FF5B NMI kezelése
- FF5C—FF71 BAUD sebesség táblázat (RS232)
- FF72—FF85 IRQ kezelése
- FF86—FFFF KERNAL ugrási címek (ugyanolyan az elhelyezkedésük, mint a C64-nél)

## KIEGÉSZÍTÉSEK A ROM RUTINOK JOBB MEGÉRTÉSE ÉRDEKÉBEN (POKE címek)

POKE cím	Magyarázat
211	kurzorpozíció az aktuális sorban
214	az éppen aktuális képernyősorszám
97—102	akkumulátor 1.
105—110	akkumulátor 2. a számításhoz
115—138	CHARGOT rutin
139—143	RND munkaterület
256—266	lebegőpontosból ASCII-vé konvertálási terület

(td)

Ebben a rovatunkban 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.

■ Sinclair Spectrum 48 k, 80 k-ra keresek angol—magyar fordítóprogramot (szótárt), széles körű felhasználói programokat, használati útmutatóval, általános iskolai 4., 5., 6. osztályos tananyaggal kapcsolatos programot. Várom azok segítségét, akik megmutatnák, hogyan lehet a játékokba „belenyúlni”, hogy a feliratokat magyarra átírassuk, a billentyű helyett pedig botkormányt használhassunk. Kékesi Lajos, Bp., Havanna u. 46. IV. em. 21. 1181. Telefon: 585-163.

■ 16 k-s ZX—Spectrumot vennék. Vastag László, Bp., Dévényi u. 18. fszt. 6. 1135. Telefon: 496-370/1708.

■ ZX—Spectrumhoz programokat cserélek. Válaszokat a programok listájával együtt erre a címre kérek: Monori Zsolt, Leninváros, Malinovszkij út 42. 3580.

■ Commodore 16-ra küldjön egy játékprogramot kazettán, és én kettőt küldök cserébe. Ha egyet kölcsönöz néhány napra, kettőt küldök vissza. Majoros Zoltán, Miskolc, Tizeshonvéd u. 18. 3525.

■ Spectrumhoz Dk'tronics fényceruza eladó. Vennék angol „Your Computer” magazinokat, 1984—85-ös számokat. Telefon: 156-411.

■ VC20 számítógép programokkal, könyvekkel együtt 8000 forintért eladó.

Elek, Táncsics út 30. 5742.

■ C16, C116, Plus/4 számítógépre angol és orosz oktatóprogramok eladók. Kálmán Albert, Eger, Rákóczi út 31. III. 11. 3300. Telefonüzenet: 143-031, 330-345.

■ C64-es programokat cserélek. Kozák Zoltán, Sopron, Várfal u. 8/A, 9400.

■ C64-re készült játék- és felhasználói programokat, programleírásokat cserélek lemezen és kazettán. Pesti László, Bp., Klapka u. 3. 1193. Telefon: 780-224.

■ Eladó 3 db japán 1,8°|8,1 V|0,3 A léptetőmotor, 5 db 27 128 EPROM, gyári C16|+4 programkazetták. Vennék 74 LS 688-at. Bp. XIX., Petőfi út 1. VI. 18. Kakuk.

■ ZX—Spectrum szimulátor (repülőgép, helikopter) programokat kérek egyéb programokért. Varga István, Eger, Csapajev s. 1. III. 13. 3300.

■ C64-re készült játék- és felhasználói programot cserélek. Vukman Szabolcs, Pécs, Budai Nagy Antal u. 10. 7624.

■ ZX81 eladó. Ára 5000 forint. Telefon: 651-608.

■ ZX81 16 k-s bővítővel eladó. Árpádi János, Bp., Magyar u. 66. fszt. 6. 1212. Telefon: 486-297.

■ Lézer 310 típusú számítógép tulajdonosa vagyok. Ehhez a géphez keresek játékprogramokat. Babos Levente, Bp., Haris köz 1/A, II. 2. 1052.

■ A Mikroszámítógép Magazin 1983—85. évi számai eladók. Árajánlatot: Szeged, Postafiók 673.

■ EPROM-programozó készülékhez 28 lábás IC-foglalatok eladók (390,— Ft/db). Csányi Gábor, Mezőtúr, Ifjúsági ltp. 13. III. 16.



PRO-KONTRA GM.  
1074 Budapest.  
Csengery u. 7. fszt. 1/a  
☎ 417-893

## Cégünk ajánlataiból:

- 64 k-s memóriabővítő Commodore C16-hoz, 2300 Ft/db
- FAST—VC-1541 kommunikáció-gyorsító rendszer a C64-hez (minden gép-floppy közötti és floppyn belül műveletet 4—12-szeresre gyorsít)

### Konfiguráció beszereléssel együtt 7000 Ft

- Abszolút programvédelem C64-hez: cartridge-ben, műgyantával klónított EPROM-ba égetéssel, MÁSOLHATATLANNÁ teszi programjait.
- IEC buszról vezérelhető mérőpontváltó egység 2 x 30 vagy 1 x 60 bemeneti, 2, III. 4 kimeneti csatornával.
- Digitális kijelzésű elektronikus óra 8 x 14 cm-es digitmérettel, különféle színekben.
- Egyéb, közepes sorozatú fejlesztések igény szerint.

Pro-Kontra Automatizálási, Műszaki Tanácsadó és Közvetítő GM.

Budapest, Csengery u. 7. fsz. 1/a 1074

Telefon: 417-893

Levél cím: Budapest, Pf. 72. 1581

Telex: 22-7770





ZX—SPECTRUM

# A Beta BASIC és alkalmazása II.

## PROGRAMOK

### A program használata

**Betöltés:** LOAD `""`. A program betöltése előtt NEM KELL BETÖLTENI a Beta basic 1.8-at, mert rajta van a szalagon és magától betöltődik. Tehát LOAD `""` és betöltődik a loader, a Beta, a program és a szóállomány. Emiatt a töltési idő eltart egy pár percig.

Betöltődés után megjelenik az „Indítsd el a programot. (RUN + ENTER)” felirat. Ekkor a program még minden nehézség nélkül listázható, és RUN-ra elindul. RUN után megjelenik a menü.

### Beírás

A beírás az alsó képernyőre történik; ENTER hatására a szöveg a memóriába és a felső képernyőre kerül. Ha a beírni kívánt szöveg már nem férne el a tárban, a felső képernyőn inverz „A szótár megtelt” felirat jelenik meg, és a program kereső üzemmódra vált. Az elválasztó karakter a szorzásjel (CHR\$ 42), ezért ezt ne használjuk a szövegben. Visszatérés a menühöz: 0, ENTER.

A program 27060 bájt területen tárolhat feljegyzéseket. Ezek hossza tetszőleges, de max. 255 karakter. A beírás POKE utasítással történik adott címre; ezt a program végzi el.

### Keresés

A keresett szót az alsó képernyőre írva (+ ENTER) megjelenik a felső képernyőn inverzben, és alatta láthatók a megtalált szópárok. Egy kérdésre több válasz is lehet, például air = affair air, aircraft, airliner, air mail stb.

A kiírás késleltetett, így ha kiírás közben megtaláljuk a kívánt szót, nyomjuk meg a BREAK-et, és visszaáll a kereső üzemmód, várva a következő keresendő szót. Ugyanez a helyzet, ha a képernyő megtelt, és megjelenik a „scroll?” felirat. Ilyenkor „n” vagy BREAK válasz esetén szintén megjelenik alul a „KERESÉS” felirat, de a felső két sor elvész.

A nagybetűvel írt, de kisbetűvel keresett szót a szótár nem találja meg. Ilyenkor, vagy ha a szó nincs a szótárban, a gép a „Nem ismerem ilyen szót” választ adja.

A keresés PAUSE 20-szal lassított, de ez elhagyható. Keresésnél a program minden olyan rekordot kiír, amelyikben a keresett szó szerepel, mindegy, hogy az idegen nyelvű vagy a magyar részén.

### Kimentés

A szóállomány tetszőleges néven magnóra menthető. A kimentés után kívánságra ellenőrizhető a felvétel (VERIFY). Magnótöltési hiba esetén a kimentéshez, egyébként a menühöz jutunk vissza.

Ha tévedésből nyomtuk meg a 3. billentyűt, az idézőjelből kilépve STOP utasításra visszatér a menü.

### Betöltés

Ezt a pontot választva a program előbb ellenőrzi az adatmezőt, és ha az nem üres, felirat figyelmeztet annak törlésére. A fájl törlése után már elvégezhető a szóállomány betöltése. Hiba esetén a törlésre figyelmeztető felirathoz, egyébként a menühöz jutunk vissza.

### Szótörlés

Ugyanúgy működik, mint a keresés, de az utoljára kiírt szópár az „l” billentyű lenyomására kitörlődik a memóriából. Más billentyűt lenyomva visszatér a szótörlés üzemmód.

### Fájltörlés

A 6. billentyűt lenyomva megjelenik a „Törléshez nyomd le az ENTER-t” felirat. Az ENTER-t lenyomva törlődik az egész szóállomány, visszatér a menü alul inverz „FÁJL TÖRÖLVE” kiegészítéssel. Más billentyűt lenyomva visszajutunk a menühöz.

### Információ

Megjelenik a tár mérete, a felhasznált és a szabad bájtok száma, a beírt szópárok száma és diagramon a tár telítettsége. Egy billentyűt lenyomva visszatér a menü.

### A program megállítása

A sok ON ERROR utasítás hatására a program a szokott módon nem mindig állítható meg, de azért akad néhány megoldás.

1. Menünel a BREAK-et legalább 5-6 másodpercig nyomva megáll a program. Ekkor azonban, mivel az ON ERROR szubrutinként működik, de nem érkezett RETURN vagy POP utasítás, a GOSUB veremben felhalmozódnak a pr. 11. sorának címei és scroll? – „n” válasz esetén visszaugorhat a menü.

2. Meghívjuk az Információt (7 opció), ha megjelent, megnyomjuk a BREAK-et. Ekkor az alsó képernyőn megjelenik a keresés üzemmód. Most Caps shift + 6, az eredmény: H STOP in INPUT, 200: 1. Ezzel a módszerrel mindig megállítható a program.

3. Az előzőből következik. Szókeresés üzemmódban minden olyan esetben, ha a vezérlés a 215 ON ERROR 200 sorról kerül vissza a 200-as sorra, a program Caps. s. + 6-tal megállítható. Ez két esetben lehetséges: a scrollra BREAK (D BREAK üzenet), vagy ha a GOSUB verem üres, a 220 POP sor hatására (V No POP data üzenet).

Mivel az ON ERROR utasítást nem előírás szerint alkalmaztam (vagyis hibakezelő szubrutin vagy POP utasítás), előfordulhat, hogy a program megállításakor a GOSUB verem nem üres. Így megeshet, hogy listázásnál egy scroll? „n” válasz esetén visszajutunk a program egyik ON ERROR sorára. Ezért ha a program futása megállt, először CLEAR utasítást adjunk.

A program a Beta ON ERROR-ja miatt nem használ ékezetes betűket. Szintén emiatt futását az elején STOP-pal egyszer meg kellett állítani.

### A program leírása

#### Loader

A RAMTOP-ot 27699-re állítja, kirajzolja a címlapot. Betölti a Beta basicet és meghívja (RAND. USR 58419). Eltünteteti a Beta copyright feliratot (CLS), ismét kirajzolja a címlapot. Más billentyűhangot állít be (POKE 23609,5), és betölti a BASIC programot. A copyright felirat eltüntetése egyszerűbb lett volna a POKE 23624,63 (BORDER 7, INK 7) megoldással, de ekkor egy esetleges magnótöltési hiba sem lett volna olvasható.

#### A BASIC program (SAVE "" LINE 2)

1—5 Betölti a szóállományt (2 sor), kiírja a „RUN + ENTER” feliratot, megállítja a programot. Erre sajnos a Beta miatt van szükség, mivel az még nem jelentkezett be. Most megetheti. RUN után az első 5 sor kitörlődik (5 DELETE 1 TO 5).

10—50 Menü. 15 ON ERROR 15: BREAK esetén egy végtelen ciklus, ami miatt csak nehezen áll meg a program. Erre a sorra jutunk vissza a 300-as és a 405-ös INPUT sorról is STOP esetén. A GOSUB vermet minden szá-



bályos visszatérésnél a 10 CLEAR sor törli.  
 100—120 Adatbeírás, a beíró eljárás meghívása.  
 200—220 Keresés, a kereső eljárás meghívása. 215 ON ERR.  
 200: Scroll?-n válasz esetén visszatér a 200-as sorra.  
 GOSUB verem törlése: 220 POP.  
 300—335 Kimentés. 305—310: a fájl hosszának kiszámítása.  
 315-ös sor: R hiba esetén újra a kimentéshez tér vissza.  
 400—420 Szóállomány betöltése. 400-as sor: ellenőrzi, üres-e a fájl. 410-es: R hiba esetére.  
 500—505 Fájl törlés. Gépi kódú rutin.  
 600—605 Az adatsor végét keresi. (Első 0 cím.)  
 700—735 Információ. 725: Szavakat számoló rutin. 730: BREAK esetén a 200-as sorra ugrik.  
 800—815 Szótörlés, a szótörlő eljárás meghívása. 810-es sor Scroll?-n, GOTO 800.  
 1000—1025 Beíró eljárás.  
 1005 Keresi az első szabad helyet.  
 1010 Ellenőrzi, van-e elég hely a beírásra. Ha nincs, figyelmeztet, és keresésre ugrik.  
 1015 Elválasztó karakter beírása (CHR\$ 42).  
 1020 Beírja a szót (kiterjesztett POKE).  
 1100—1175 Kereső eljárás.  
 1105 C=a keresett szó első előfordulási helye 28700—55760 között.  
 1110 Ha C=0, nincs ilyen szó, az eljárás végére ugrik.  
 1115 h: a szótörlő eljárás miatt felvett változó. Értéke 1, ha a program megtalálta a keresett szót, 0, ha nem.  
 1120—1140 C-től visszafelé keresi az első elválasztó karaktert. Értéke C1 lesz.  
 1145 C1-től előre keresi az első elválasztó karaktert. Értéke C2.  
 1150 Ha nem találja, a keresett szó az utolsó a fájlban.  
 1155—1160 Kiírja a két elválasztó karakter közötti teljes szöveget. 1160 PAUSE 20: a kiírás késleltetése. Ez elhagyható, de így barátságosabb a keresés.  
 1165 C2-től keresi a következő szót.  
 1170 Ha C=0, nincs több ilyen szó, eljárás vége. Ha nem, visszatér az elválasztó karakterek kereséséhez.  
 1200—1245 Szótörlő eljárás.  
 1205 A kereső eljárás meghívása. Ha nincs ilyen szó (h=0), az eljárás végére ugrik.  
 1210—1215 Ha nem "1"-et nyomsz, az eljárás végére ugrik.  
 1225 Megkeresi a fájl végét (GOSUB 600), C1—C2 között kitorli a mezőt.  
 1230 C1-re tölti a törölt szó mögötti részt.  
 1235—1240 A fájl végéről kitorli a maradékot, és tájékoztat, hogy a szótörlés megtörtént.

## Gépi kódú rutinok

### Karakterszámláló rutin

			ORG	55760	
D9D0	01B469		LD	BC,27060	; ciklusszám
D9D3	211C70		LD	HL,28700	; kezdőcím
D9D6	110000		LD	DE,0	; itt lesz az eredmény
D9D9	3E2A	L1	LD	A,42	; CHR\$ 42
D9DB	EDA1		CPI		; összehasonlít
D9DD	2001		JR	NZ,L2	; ha nem egyezik, ugrás L2-re
D9DF	13		INC	DE	; ha igen, DE-t növeli
D9E0	78	L2	LD	A,B	; B-t A-ba tölti
D9E1	B1		OR	C	; és összehasonlítja C-vel
D9E2	20F5		JR	NZ,L1	; ha nem egyezik, ugrás L1-re
D9E4	D5		PUSH	DE	
D9E5	C1		POP	BC	; eredményt a BC-be
D9E6	C9		RET		

Vagy DATA 1,180,105,33,28,112,17,0,0,62,42,237,161,32,1,19,120,177,32,245,213,193,201

### Fájltörlő rutin

			ORG	55783	
D9E7	1600		LD	D,00	; 0-t tölt, majd az adott c.
D9E9	211C70		LD	HL,28700	; kezdőcím
D9EC	01B469		LD	BC,27060	; ciklusszám

D9EF	72	CIKL	LD	(HL),D	; beírja a 0-t
D9FO	23	INC	HL		; növeli a HL-t
D9F1	0B	DEC	BC		; csökkenti BC-t
D9F2	78	LD	A,B		; A-ba tölti B-t
D9F3	B1	OR	C		; és összehasonlítja C-vel
D9F4	20F9	JR	NZ,CIKL		; ha nem egyezik, ugrás CIKL-ra

D9F6 C9 RET ;ha igen, vége  
 Vagy DATA 22,0,33,28,112,1,180,105,114,35,11,120,177,32,249,201.

Az első rutin elhagyható, a második Beta basic-kel helyettesíthető. Például CLEAR 34799. A fájl 21000 bajtra csökkentjük. A program 500-as sorát kicseréljük a következőre:

```
500 FOR n = 34800 TO 52300 STEP 3500: POKE n,
STRING$(3500,CHR$(0)):NEXT n.
```

Ez a sor a 34800-as címtől kezdve 6 db 3500 karakter hosszúságú, 0-ból álló sztringet ír a memóriába, egészen az 55800-as címig. Ezenkívül még a következő javításokra van szükség: ha a Betát külön szalagról töltjük be, elmarad a loader, a program eleje pedig így módosul:

```
1 LOAD "" CODE : REM szóállomány betöltése
2 DELETE 1 TO 2 . Ekkor a program autostarttal indulhat.
```

At kell javítani még a következő sorokat:

```
310 SAVE m$ CODE 28700, cím - 28700
400 IF cím ≠ 28700
```

```
600 LET cím = INSTRING/28700, MEMORY$..
```

ezekben a sorokban a 28700-at 34800-ra kell javítani.

```
700 sor a 27060-at 21000-re,
a 28700-at 34800-ra,
az 55760-at 55800-ra
```

```
705—720 sor elmaradhat (ez a diagram)
```

```
725 sor törölve (karaktersz. rutin hívása)
```

```
1010 sor 55759-et 55799-re
```

```
1105 sor 28700-at 34800-ra
```

```
55760-at 55800-ra
```

1145, 1165 sorok

55760-at 55800-ra kell javítani

Természetesen más értékek is kipróbálhatók. A lényeg az, hogy a fájl törlése még elfogadható sebességgel történjen, és legyen elég hely a RAMTOP alatt a STRING\$ függvény számára.

### A program elhelyezkedése a memóriában

RAMTOP:	28699
Szóállomány:	28700—55759 27060
Karaktersz. rutin:	55760—55782
Fájltörlő rutin:	55783—55798
Beta basic:	55801—65367
UDG:	üres

A program és a RAMTOP közötti szabad terület CLEAR után

Az 1—5. sor kitörlése előtt: 556 bajt

Az 1—5. sor kitörlése után: 724 bajt

### A szóállomány

Jelenleg angol—magyar szavak vannak a szótárban, számuk 789. Ez 16949 bajt területet foglal le; szabad még 10111 bajt. Ha a program futását megállítjuk és kiadjuk a következő parancsokat: GOSUB 600: PRINT MEMORY\$ // (28700 TO cím), láthatjuk, hogy ez 24 sűrűn teleírt képernyőt jelent. Ennek ellenére a program egy oxfordi szótárral nem tudja felvenni a versenyt. Egy angol—magyar miniszótárban kb. 4000, az Ország Hírlap-féle kiszótárban közel 22 000 szó található. Mint már említettem, a rendelkezésre álló 27 060 bajt területre a hosszúságtól függően nagyjából 1000—1200 szópár írható be. Ezért ajánlatos csak a nem ismert szavakat beírni, és megtanulás után kitorolni.

A mostani szóállomány a következő képet mutatja. Először elkezdtem bemásolni az angol—magyar miniszótár szavait. Eljuttam a c betű elejéig, és akkor rájöttem, hogy elvettem a súlykot: ez kb. 7000 bajt. Idáig a szavak rendezettek. Ezek után egy angol könyvet olvasva csak a nem ismert szavakat szótáraztam ki. Innen kezdve rendezetlen az állomány, de a visszakeresés szempontjából szerintem ennek nincs jelentősége. Minden szót kisbetűvel írtam be, így a gép „C” kurzorral semmit nem talál meg.

Még egy szépséghiba: a szótár nem használja az ékezetes betűket. Megoldható lett volna grafikus karakterekkel, ekkor azonban a kiírás idejére ki kellett volna kapcsolni a Beta basicet KEYWORDS 0 utasítással. Ilyenkor előfordulhatott volna, hogy a kiírás közben BREAK-et nyomva leáll a program. A Beta kulcsszavak ugyanis grafikus üzemmódban érhetők el KEYWORDS 1 esetén.

LITAUSZKY GYÖRGY



# Manó a Spectrumon

A Commodore-hívők egyik érve a Spectrummal szemben: a Sinclair-gép nem tud sprite-okat kezelni. A Spectrumot a hardver oldalról valóban elég szegényen látta el Sir Clive. Programmal azonban — ha mindent nem is —, sok érdekes dolgot el lehet érni! Az alábbiakban egy  $16 \times 16$ -os, hátteret nem takaró, nem nagyítható, nem színes, szóval elég buta manót szimuláló programot mutatok be.

## Az alapelv

A Spectrumban levő ULA másodpercenként 50 alkalommal megszakításjelet küld a Z80-nak. A processzor erre — az IM 1 módban — a  $0038H$  címre ugrik; továbbírja a Spectrum óráját és elvégzi a billentyűzet dekódolását, majd folytatja, amit félbehagyott. Saját megszakításrutinokat, amelyek 20 ms-onként lefutnak, a Z80 IM 2 módjának felhasználásával készíthetünk. Az elég bonyolult IM 2 megértését is segíti, ha most végigkövetjük, hogyan működik a manó.

A megszakítás pillanatában a Z80 beolvassa az adatbuszon levő bájtot egy munkaregiszterbe. Ez a regiszter (alsó báj) az I regiszterrel (felső báj) együtt egy 16 bites memóriacímre ad. Erről a memóriacímre olvassa ki a megszakításrutin kezdőcímét a központi egység. A Spectrum sajnos nem használja ki az ebben az ugrótáblázatban rejlő lehetőségeket, terheletlen adatbuszról FFH-t olvas be alsó bájtnak.

A manórutin is egy megszakításprogram. Megvizsgálja, változott-e a manó pozíciója az elmúlt 0,02 másodperc során. Ha igen, törli a manót régi helyéről és kirakja az újra. Egyszerű feladat, csak hogy igen gyorsan kell megcsinálni: amíg az elektronsugár a keret felső szélét rajzolja. Mert ha nem készülünk el időben, és a manó épp a képernyő tetején van, csúnya villogást keltünk.

A minél nagyobb sebesség érdekében korlátoztam a manó tulajdonságait a fentebb felsoroltak szerint.

A program sajnos még így sem elég gyors: ha a manót a képernyő felső szélé közelében 3, 4 vagy 5 pixelenként mozgatjuk jobbra vagy balra, a manó alsó részé villog! Ez ellen sajnos csak megfelelő programozással védekezhetünk.

## Használat

A Spectrum igen nehezen engedi meg új utasítások definiálását, ezért a manót csak a C64 alapgép módszerével: POKE-okkal vezérelhetjük.

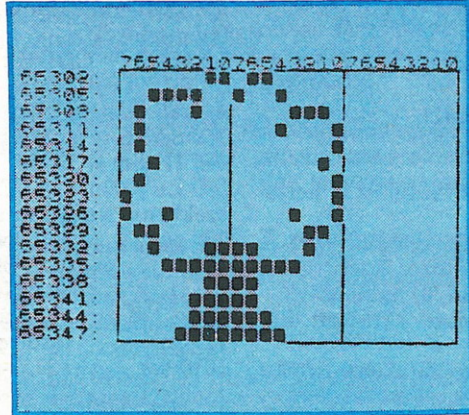
65 297: x koordináta  
0 .. 255  
65 298: y koordináta  
0 .. 255

a bal  
felső  
pixel  
helye

Ezek a koordináták nem egyeznek meg a Spectrum grafikáival, mert a  $0,0$  pont a

képernyő bal felső sarkát jelöli. Megfelelő koordináták megadásával elérhetjük, hogy a manó fent, jobbszélen vagy lent „kiúszson” a képből. Bal szélen erre a mutatóvanyra nem képes.

Mozgatásnál vegyük figyelembe, hogy bármilyen gyorsan is írjuk az új koordinátákat, a manó csak 50-ed másodpercenként ugrol. Gépi kódú programból azt is megvizsgálhatjuk: ugrott-e már a manó. A 65 295 (x0) és 65 296 (y0) változók a manó tényleges pozícióját mutatják.



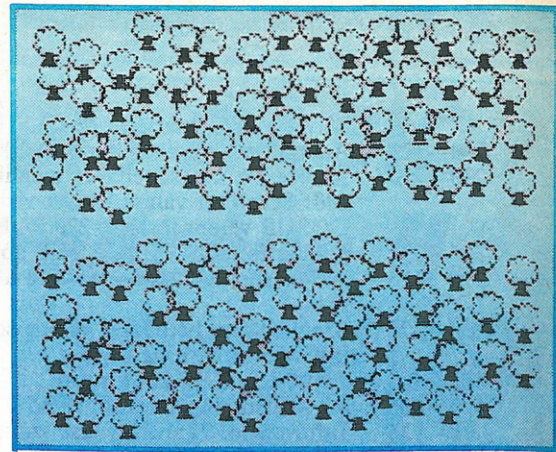
1. ábra. A manó térbeli képe

### 1. lista

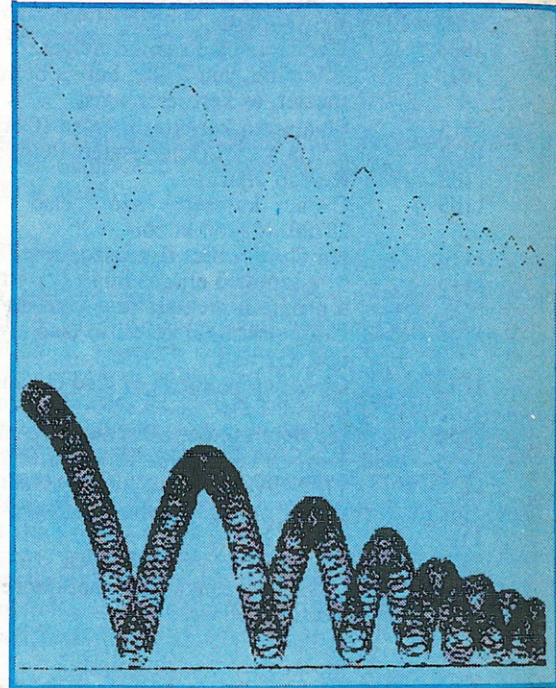
```

100 REM ** erdoraajzolas **
110 REM
120 REM
130 LET xx=65297: LET yy=xx+1
140 LET mod=65226
150 LET sp=65302
160 REM
170 RESTORE 600: GO SUB 400: CL
S
175 POKE mod,132
180 FOR x=5 TO 240 STEP 16
185 LET n=1
190 FOR y=5+INT (12*RND) TO 160
STEP 16
195 LET n=n+1: IF n=6 THEN GO T
O 230
200 LET i=x+INT (3-7*RND)
210 LET j=y+INT (3-7*RND)
220 PAUSE 1: POKE xx,i: POKE yy
j
230 NEXT y
240 NEXT x
300 POKE yy,192: POKE xx,0
309 STOP
400 POKE xx,0: POKE yy,192: PAU
SE 1
410 FOR i=0 TO 47
420 READ byte: POKE sp+i,byte
430 POKE mod,174
435 NEXT i
440 RETURN
600 REM ** kerak fa **
610 DATA 3, 95,0
611 DATA 60,144,0
612 DATA 66, 14,0
613 DATA 64, 17,0
614 DATA 64, 1,0
615 DATA 32, 2,0
616 DATA 64, 1,0
617 DATA 128, 1,0
618 DATA 144, 9,0
619 DATA 96, 6,0
620 DATA 35,196,0
6201 DATA 31,248,0
6202 DATA 3,192,0
6203 DATA 7,192,0
6204 DATA 7,224,0
6205 DATA 15,240,0
600 BORDER 0: PAPER 0: INK 9
610 CLEAR 65022
620 LOAD "sp_code"CODE
630 RANDOMIZE USR 65351
640 GO TO 100
900 SAVE "sp_erd0" LINE 800
910 SAVE "sp_code"CODE 65023,33
S
920 VERIFY "sp_erd0"
930 VERIFY "sp_code"CODE
940 STOP

```



2. ábra. Az sp-erdő futási eredménye



3. ábra. A labdapattogatás két ábrája

### 2. lista

```

100 REM ** d0m0 **
105 CLEAR 65022: LOAD ""CODE
110 LET xx=65297: LET yy=xx+1:
REM koordinaták
120 LET sp=65302: REM sprite me
zo kezdete
130 LET mod=65226: REM XOR/OR m
od
138 REM
140 REM ** palya kirajzolasa **
145 REM
150 DIM p(256): FOR i=1 TO 256
155 LET f=(i-1)/(350-i)*10
170 LET p(i)=160-ABS INT (100*C
OS I*EXP (-f/10))
180 PLOT i-1,160-p(i): NEXT i
190 GO SUB 600: CLS
200 RESTORE 500: GO SUB 400: RE
M labda definialasa
210 PLOT 0,0: DRAW 255,0
220 FOR i=1 TO 255: POKE xx,i:
POKE yy,p(i): NEXT i
230 POKE yy,192: POKE xx,0
240 RESTORE 550: GO SUB 400: RE
M kor definialasa
250 GO SUB 600
255 POKE mod,132: REM OR mod
270 FOR i=1 TO 255: POKE xx,i:
POKE yy,p(i): NEXT i
280 POKE mod,174: REM XOR mod
300 GO SUB 600: CLS
300 GO TO 200
400 REM ** definialo rutin **
410 POKE xx,0: POKE yy,192: PAU
SE 1
420 FOR i=0 TO 47
430 READ byte: POKE sp+i,byte
440 NEXT i
450 RETURN
600 REM ** lab4a **
610 DATA 7,224,0
611 DATA 24, 24,0
612 DATA 33,132,0
613 DATA 65,130,0
614 DATA 88, 6,0
615 DATA 152, 68,0
616 DATA 128, 49,0

```







DISPLAY KARAKTEREK

FF	70	14	12	14	70	30	40	40	
4300	>	70	14	12	14	70	30	40	40
4300	>	30	40	7E	52	52	52	42	30
4310	>	54	57	54	18	3C	42	47	42
4310	>	3C	30	44	47	44	30	3D	42
4320	>	42	42	3D	30	45	44	45	30
4320	>	3C	43	42	43	3C	40	47	44
4330	>	47	30	3E	40	43	40	3E	3C
4330	>	43	40	3C	40	3E	41	40	41
4340	>	3E	3D	40	41	3C	40	3E	43
4340	>	40	43	3E	3F	40	43	3C	40
4350	>	20	40	3E	01	02	20	40	30
4350	>	09	02	04	02	7F	02	04	10
4360	>	20	7F	20	10	00	1C	2A	00
4360	>	00	00	00	2A	1C	00	14	22
4370	>	7F	22	14	12	09	12	24	12
4370	>	00	04	03	00	00	04	03	04
4380	>	03	00	14	00	14	14	14	03
4380	>	3B	44	44	20	40	40	7F	40
4390	>	40	7F	40	40	70	40	20	30
4390	>	20	34	22	14	34	1C	10	14
43A0	>	60	50	40	44	7E	3E	41	5D
43A0	>	41	3E	63	55	45	41	63	07
43B0	>	40	7F	40	07	5C	62	02	62
43B0	>	5C	30	40	30	2E	41	7F	01
43C0	>	25	2A	10	22	51	3F	04	02
43C0	>	30	40	4A	40	39	00	15	52
43D0	>	55	22	01	06	01	01	7E	44
43D0	>	30	10	30	44	40	22	11	3E
43E0	>	40	4E	55	25	09	06	7F	00
43E0	>	00	00	00	04	4A	3F	0A	04
43F0	>	7F	49	49	49	32	7F	01	01
43F0	>	01	02	70	2E	21	3F	00	63
4400	>	1C	7F	1C	63	7F	20	1C	02
4400	>	7F	7E	21	1A	05	7E	40	3E
4410	>	01	01	7F	3F	20	20	3F	40
4410	>	07	00	00	04	7F	7F	40	7F
4420	>	40	7F	3F	20	3F	20	7F	7F
4420	>	40	40	30	7F	02	7E	40	40
4430	>	30	41	49	45	2A	1C	7F	00
4430	>	3E	41	3E	46	29	19	09	7F

PRINTER CÍMTÁBLA

FF	02	02	02	00	02	17	02	20	
4440	>	02	02	02	00	02	17	02	20
4440	>	02	2C	02	30	02	44	02	51
4450	>	02	5E	02	60	02	70	02	83
4450	>	02	0E	02	90	02	A0	02	B1
4460	>	02	0E	02	04	02	D3	02	D0
4460	>	02	DD	02	E2	02	E7	02	EF
4470	>	02	F7	02	F0	03	01	03	09
4470	>	03	15	03	18	03	22	03	28
4480	>	03	31	03	35	03	44	03	4D
4480	>	03	57	03	62	03	60	03	75
4490	>	03	7E	03	80	03	96	03	9E
4490	>	03	A5	03	A0	03	B7	03	BF
44A0	>	03	CC	03	D5	03	D0	03	E1
44A0	>	03	F2	03	F0	04	01	04	06
44B0	>	04	11	04	10	04	1E	04	27
44B0	>	04	30	04	30	04	42	04	4D

GÉPI KODÚ PROGRAMBETÉT

FF	FD	50	FD	5A	FD	52	4A	40	
44C0	>	FD	50	FD	5A	FD	52	4A	40
44C0	>	0A	3F	34	F9	03	41	44	00
44D0	>	07	FD	40	4A	00	34	F9	03
44D0	>	41	34	AE	70	5E	E9	70	5D
44E0	>	00	FD	50	5A	C6	85	30	6A
44E0	>	12	51	00	03	E9	70	75	00
44F0	>	A4	DD	DD	DD	20	80	10	0A
44F0	>	55	CD	92	60	00	CD	AC	9A

PRINTER CÍMMUTATÓ

FF	01	40
4500	>	01 40

PRINTER KARAKTEREK

FF	54	4A	7A	13	72	31	74	12	
4502	>	54	4A	7A	13	72	31	74	12
450A	>	E4	02	61	59	52	49	42	73
4512	>	79	19	69	17	F2	56	44	30
451A	>	64	13	43	14	21	F2	03	62
4522	>	59	52	49	42	79	71	64	15
452A	>	02	F2	02	61	59	54	49	42
4532	>	79	74	69	61	17	F2	02	61
453A	>	59	52	49	42	79	72	69	61
4542	>	17	F2	17	41	02	41	32	74
454A	>	69	62	59	54	49	42	F9	10
4552	>	41	02	41	33	72	69	62	59
455A	>	52	49	42	F9	01	59	54	49
4562	>	42	79	74	69	62	15	52	02
456A	>	F2	01	59	52	49	42	79	72
4572	>	69	62	15	52	02	F2	01	59
457A	>	55	02	11	72	02	11	75	69
4582	>	E2	02	61	59	53	03	73	79
458A	>	19	69	17	F2	11	55	11	41
4592	>	02	41	31	75	69	62	D9	11
459A	>	53	12	41	01	41	32	73	79
45A2	>	19	69	61	D9	01	55	55	04
45AA	>	75	69	62	17	72	00	F2	11
45B2	>	53	11	52	02	72	39	73	79
45BA	>	19	69	61	D9	11	79	49	54
45C2	>	49	F9	11	79	49	51	61	59
45CA	>	49	42	79	69	61	52	51	49
45D2	>	F9	02	50	7A	1A	EA	02	5A
45DA	>	04	0A	06	13	4A	3A	64	FA
45E2	>	13	44	5A	3A	EA	02	50	7A
45EA	>	32	6A	5A	12	CA	12	49	70
45F2	>	49	12	69	5A	E9	10	02	72
45FA	>	E9	13	49	52	02	72	E9	11
4602	>	41	01	42	12	02	21	E1	14
460A	>	42	52	62	72	03	62	65	72
4612	>	79	42	C9	44	22	D6	44	21
461A	>	52	59	62	39	79	1A	13	F6
4622	>	11	44	14	0C	31	03	52	59
462A	>	E1	0C	64	32	44	13	EC	44
4632	>	55	6C	F1	01	59	54	49	42
463A	>	79	74	69	62	12	52	02	72
4642	>	11	E2	44	51	14	51	04	71
464A	>	7A	0A	F1	16	72	79	42	49
4652	>	52	22	76	21	C2	41	51	59
465A	>	52	49	42	79	72	69	71	C1
4662	>	01	59	51	49	70	23	49	53
466A	>	C9	56	42	79	69	61	42	79
4672	>	71	69	E2	01	41	49	55	61
467A	>	69	35	59	CC	00	5A	49	41
4682	>	33	72	69	61	59	51	49	41
468A	>	F9	02	41	49	59	62	59	51
4692	>	4A	79	69	DA	04	56	61	69
469A	>	71	51	59	E9	44	24	79	72
46A2	>	0A	72	F9	4A	7A	1A	53	59
46AA	>	E9	41	49	5A	51	79	41	49
46B2	>	51	59	61	69	F1	50	03	72
46BA	>	69	61	59	03	F9	11	79	49
46C2	>	55	29	69	71	79	42	49	51
46CA	>	59	E2	43	49	51	59	63	33
46D2	>	56	43	F9	56	43	F9	52	44
46DA	>	72	12	59	53	62	73	E9	51
46E2	>	49	52	59	51	04	71	69	72
46EA	>	79	71	22	53	61	02	61	D3
46F2	>	4A	52	4A	76	24	D6	4A	52
46FA	>	4A	76	24	56	09	79	C9	49
4702	>	54	49	42	F0	02	61	53	55
470A	>	03	75	69	09	79	71	D1	04
4712	>	56	32	69	62	59	02	44	56
471A	>	22	76	22	D6	44	41	71	19
4722	>	56	22	76	22	D6	55	04	75
472A	>	24	42	49	51	59	E2	15	41
4732	>	75	42	49	51	59	E2	11	79
473A	>	42	49	54	59	62	69	3A	C2
4742	>	56	03	79	74	69	59	52	22
474A	>	42	52	C9	48	41	63	59	51
4752	>	49	43	F0					

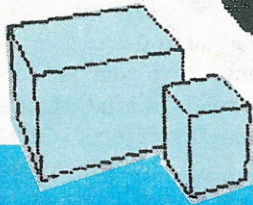
BEJELENTKEZŐ SZÖVEG

FF	42	09	70	69	74	03	73	20	
4755	>	42	09	70	69	74	03	73	20
475D	>	62	65	74	07	6C	74	76	65

A bővítéssel kapott új karakterek és ASC kódjaik:

CHR	ASC	CHR	ASC
Á	128	á	129
É	130	é	131
Ó	132	ó	133
Ő	134	ő	135
Ö	136	ö	137
Û	138	ű	139
Ü	140	ü	141
Û	142	ü	143
Š	144	š	145
↑	146	↓	147
←	148	→	149
⊕	150	≈	151
'	152	”	153
==	154	℄	155
⊥	156	℄	157
4	158	z	159
Δ	160	⊙	161
Σ	162	Ψ	163
Ω	164	α	165
β	166	β	167
δ	168	ε	169
γ	170	κ	171
λ	172	ρ	173</





## rajzolóprogram

**A VU-3D-t 1982-ben készítette a PSION szoftvercég. A program 48 k-s Spectrum gépen fut, háromdimenziós tárgyak rajzolására és ábrázolására alkalmas.**

Az ábrázolt testek teljesen körbeforgathatók a képernyőn, így minden irányból megvizsgálhatjuk azokat. A képet nagyíthatjuk és kicsinyíthetjük, közelíthetjük és távolíthatjuk. A közelítés addig fokozható, hogy a képernyővel, mint egy szeletelő síkkal, metszeteket is készíthetünk a rajzról. A pillanatnyi képet bármikor ki lehet merevíteni. Ilyenkor a test tetszőleges irányból

beábrázolható, vagy a nem látszó élek kitörölthetők. Az így kapott ábrát kinyomtathatjuk vagy screenként magnószalagra menthetjük.

Figyelembe véve a gép lehetőségeit, azt nem állíthatom, hogy a programmal alkatrészek is tervezhetők, de mértan, ábrázoló geometria stb. tanulásához hasznos segítséget nyújthat, és a térlátást is fejleszti. Azonkívül szórakozásnak sem utolsó egy-egy bonyolultabb alakzatot több irányból is megnézni.

A program elég elterjedt, de leírás hiányában kevesen vannak tisztában minden funkciójával. Mivel nekem már sikerült a

VU-3D-t megismernem, közlésem eddigi tapasztalataimat.

### A program használata

A VU-3D lényegesen több billentyűt használ, mint egy átlagos játékprogram, de ennek ellenére a program használata nem bonyolult. Használatát elmagyarázni annál inkább. Hogy ezen könnyítsek, összeállítottam a *táblázatot*. Ebből kiolvasható, hogy egy-egy almenünek milyen funkciói vannak, és ezek milyen billentyűvel aktivizálhatók. Ezért, ha leírás közben eljutunk egy új almenühöz, nem írom le és magyarázom

#### A VU-3D billentyűkiosztása

Menü Bill.	1. Modify a figure (Rajzmódosítás)	4. Create a new figure (Új rajz készítés)		6. Display (Kijelzés)	
		Open (alaplajz)	Create (testrajz)	Display (kijelzés)	Picture (kép)
"A"					Above (felülről) Sh
"B"					Below (alulról) Sh
"C"			Close (testlezárás)		Centre (középről) Sh Colour (szín vált.)
"D"		Delete (vonal töröl.) →			
"E"		End (alaprjz bef.) →			
"F"	Figure (figuraváltás)		Figure (figuraváltás)	Far (képtávolítás)	
"H"					Hidden line (nem látszó él töröl.)
"K"					Keep (kép magnóra m.)
"L"		Line (vonalhúzás)			Left (balról) Sh
"M"	Magnify (nagyítás)		Magnify (nagyítás)	Magnify (nagyítás)	
"N"	Next z (z koor. növ.)		Next z (z koor. növ.)	Near (képkitöltés)	
"O"			← Open (kurzor előhív.)		
"P"				Picture (képkimerev.) →	Print (nyomtatóra)
"Q"	Quit (kilépés) ↑		Quit (kilépés) ↑	Quit (kilépés) ↑ ←	Quit (kilépés)
"R"	Reduce (kicsinyítés)		Reduce (kicsinyítés)	Reduce (kicsinyítés)	Right (jobbról) Sh
"S"		Start (kezdő pont ki)			Shade (árnyékolás)
Kurz. bill.	Figuramozgatás	Kurzormozgatás	Figuramozgatás	Képforgatás	
CAPS + kurz.	Figuram. lassan	Kurzorm. lassan	Figuram. lassan	Képforg. lassan	



el annak minden funkcióját, hanem csak azt említem meg zárójelben, amelyekre éppen szükség van. Például (Magnify). Így fokozatosan, lépésről lépésre magyarázhatom el a program működését.

Aki nagyjából ismeri a VU-3D működését, az a táblázat segítségével mindjárt használni is tudja. A billentyűk használatát segíti az is, hogy az almenüknél megjelenő feliratok kezdőbetűi az adott billentyűkre utalnak (open=„O”, close=„C”, delete=„D”).

## Betöltés

A program betöltése a LOAD "" parancs kiadásával történik. Ekkor betöltődik a loader (EXAMPLE), a BASIC program (example), a címlap (1. ábra) és a gépi kódú rész (c).

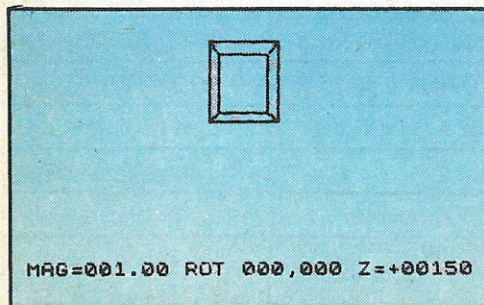
Betöltés után megjelenik a képernyőn egy pohár és egy kocka térbeli képe, valamint a DISPLAY menü. A képet a kurzorbillentyűkkel körbeforgathatjuk, nagyíthatjuk (Magnify), kicsinyíthetjük (Reduce), közelíthetjük és távolíthatjuk (Near és Far).

A PICTURE üzemmódról később lesz szó, de aki kedvet érez hozzá, a táblázat segítségével már most is kipróbálhatja.

1. ábra



2. ábra



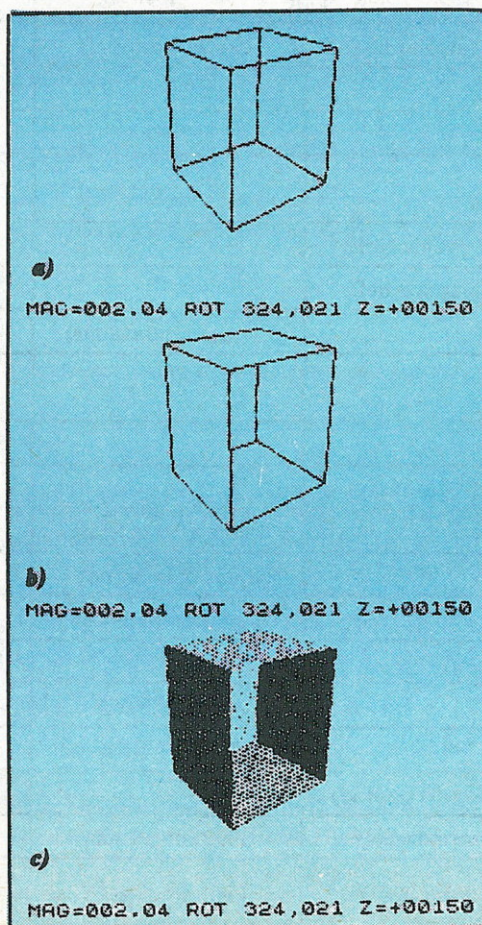
## Főmenü

A DISPLAY üzemmódból a Q billentyű segítségével juthatunk el a főmenühöz (Quit). Ez a következő pontokból áll:

1. Modify a figure = egy kész rajz módosítása
2. Abadon = az elkészült rajz kitörlése (új ábra rajzolása esetén)
3. Load a data file = egy régebbi rajz betöltése magnóról
4. Create a new figure = új rajz készítése
5. Save a data file = a kész rajz kimentése magnóra
6. Display = a rajz térbeli ábrázolása
7. Change colours = új papír-, tinta-, keretszínnek beállítása

Próbáljunk meg most egy új rajzot készíteni. Rajzoljunk egy 80 pixel élhosszúságú kockát. Válasszuk a főmenü 4. pontját

3. ábra

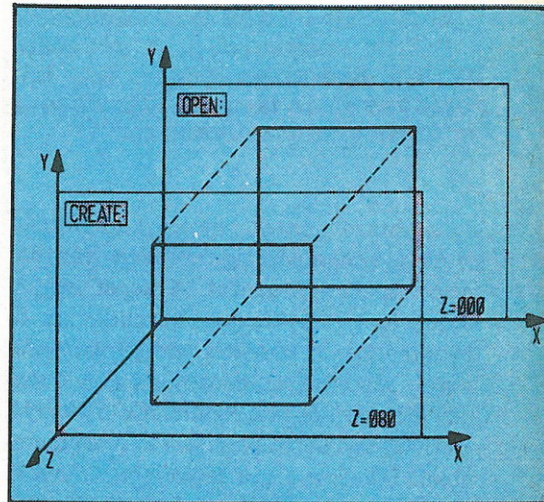


(Create...). Az ENTER-t lenyomva a következő felirat jelenik meg a képernyőn:

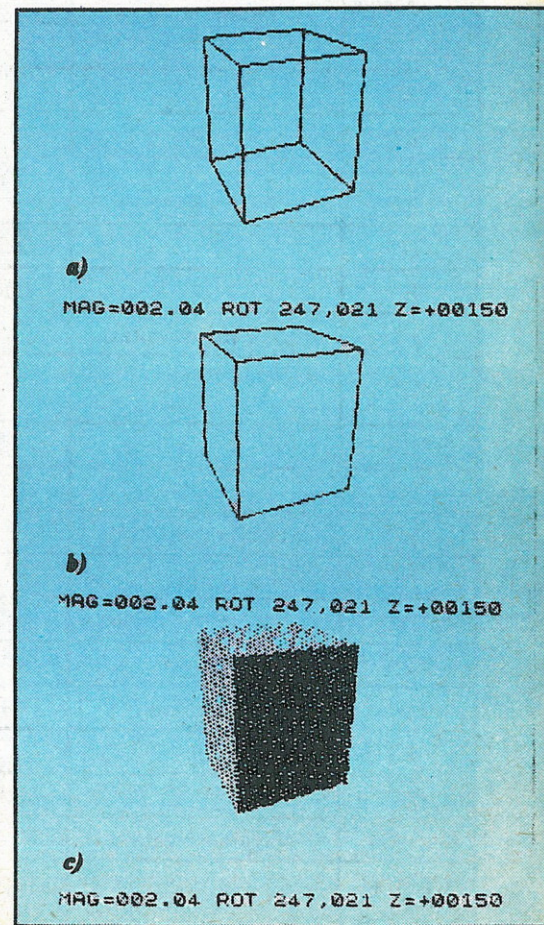
File already in existence  
Use Abadon to delete it  
Press any key to continue

Ez arról tájékoztat bennünket, hogy a pohár rajzát még nem töröltük ki a programból. Egy billentyű lenyomása után visszajutunk a főmenühöz és a 2. (Abadon) pontot választva elvégezhetjük a törlést.

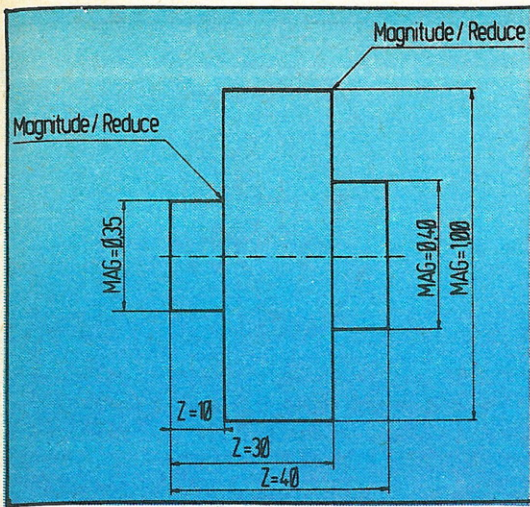
I. ábra. A VU-3D X-Y-Z koordináta-rendszere



4. ábra

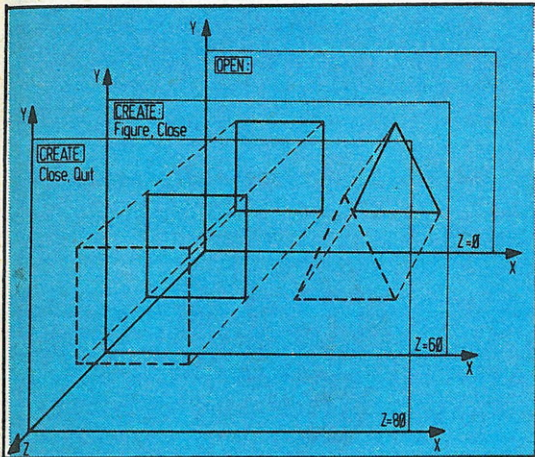




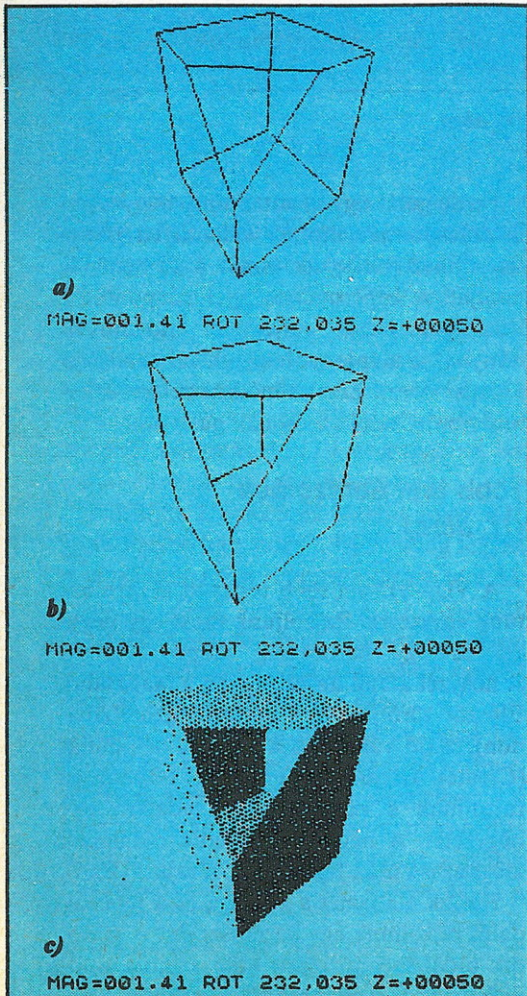


II. ábra. Két hasáb rajzolása

III. ábra. Lépcsős forgástest rajzolása



5. ábra



A törlés megtörténtéről a menü alatt megjelenő FILE ABANDONED felirat tájékoztat.

Ebből azt is észrevehetjük, hogy egy test rajzolását nem szakíthatjuk meg csak azért, hogy megnézzük, hogyan mutat a térben. Hiszen ha egyszer kiléptünk a „Create . . .” üzemmódból, oda már csak a rajz kitörlése után juthatunk vissza.

Ezek után már nincs akadálya, hogy elkezdjük a rajzolás.

### Kocka rajzolása

A 4. pontot választva a CREATE menühöz és egy síkbeli X—Y koordináta-rendszerhez jutunk. A test ábrázolása viszont egy térbeli X—Y—Z koordináta-rendszerben történik. Ezt a program írói a következő módon oldották meg.

Az X—Y koordináta-rendszerben megrajzoljuk a test alaprajzát. Ekkor még a Z koordináta értéke 0, ami a képernyő jobb alsó sarkában olvasható is. Az alaplap megrajzolása után az „N” (Next z=következő z) billentyűt nyomogatva állíthatjuk be a Z értékét, és ezzel a test vastagságát. (I. ábra).

Hívjuk elő a kurzort (Open). Ekkor az OPEN almenühöz jutunk. A kurzor a kurzorbillentyűkkel mozgatható X=0—240,

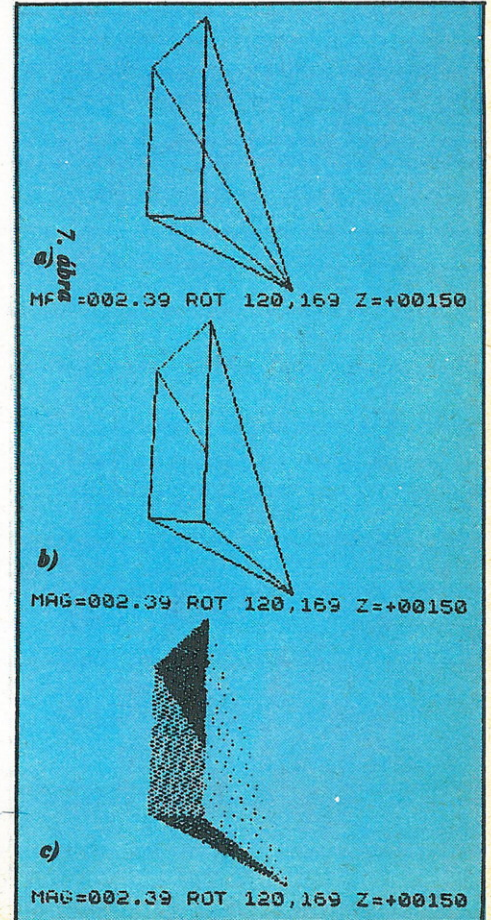
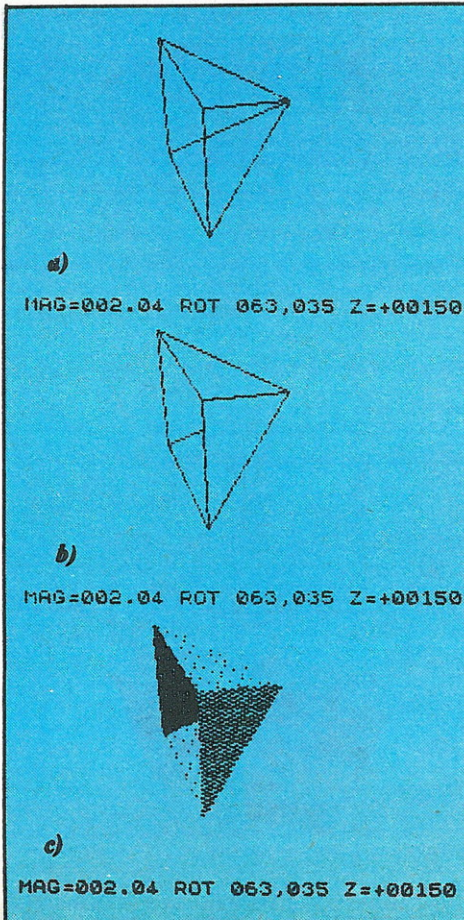
Y=0—144 értékek között. A finombeállítás a CAPS SHIFT billentyű és a kurzorbillentyűk egyidejű lenyomásával történik. A mindenkor X—Y érték a képernyő alján olvasható. Állítsuk a kurzort megfelelő helyzetbe, és tegyük ki a kezdőpontot (Start). Rajzoljuk meg egy 80×80-as négyzet három oldalát (vonalhúzás=Line, ha elrontottuk, akkor törlés=Delete). Ha a harmadik oldallal is készen vagyunk, az E billentyű lenyomásával fejezzük be a rajzot (End).

Visszajutunk a CREATE almenühöz, és megjelenik előttünk a teljes négyzet, de nem folyamatos, hanem szaggatott vonalakkal ábrázolva. Vagyis láthatjuk, hogy nem szükséges az egész síkidomot megrajzolnunk. A rajzot a gép — a kezdő- és végpontot értelemszerűen összekötve — befejezi.

Most állítsuk a Z koordináta értékét 80 pixelre. Fejezzük be a kocka rajzolását (Close). Ekkor megjelenik a DO YOU WANT THE TOP OPEN? Y OR N felirat. Vagyis nyitva legyen-e a kocka felénk néző része? Legyen nyitva. (Y).

Térjünk vissza a főmenühöz (Quit), és nézzük meg a kocka térbeli képét (6. Display), (2. ábra).

6. ábra





A kép alján lévő jelölések:  
 MAG = nagyítás = 1.00  
 ROT = elforgatás = Y tengely körül = 0; X tengely körül = 0  
 Z = most a képernyőtől való távolság = 150  
 Nagyítsuk ki a képet (Magnify) és forgassuk el a kurzorgombokkal (3. ábra), majd merevítsük ki (Picture).

Töröljük ki a nem látszó éleket. A H billentyű lenyomása után a fejléc alján megjelenik a villogó HIDDEN LINE felirat, és elvégzi a műveletet. Látható, hogy a kocka egyik lapja tényleg hiányzik (3.b ábra).

Árnyékoljuk be a képet (Shade). A most megjelenő feliratok a fényforrás irányáról érdeklődnek: Above = felülről; Centre = középről; Below = alulról; Left = balról; Centre = középről; Right = jobbról.

Válaszoljunk a megfelelő billentyűk lenyomásával, és a gép elvégzi az árnyékolást (3.c ábra). Árnyékolás közben a fejléc alján villogó SHADING felirat olvasható.

Térjünk vissza a DISPLAY menühez (Quit). Forgassuk tovább a kockát úgy, hogy nyitott részét előlről ne láthassuk. Készítsünk így is képet róla (4.a,b,c ábra).

Most állítsuk be úgy a kockát, hogy egyik csúcspontja felénk nézzen, és készítsünk így egy metszetet. Közelítsük a kockát

a képernyőhöz (Near), és közben kicsinyítsük a képet úgy, hogy az egész kocka elférjen a képernyőn (Reduce), (5.a,b,c ábra). A kép mutatja, hogy a kivágott nyíláson keresztül belátunk a kocka belsejébe.

Mivel rajzunkat alaposan körbenéztük, próbáljunk meg valamit módosítani rajta.

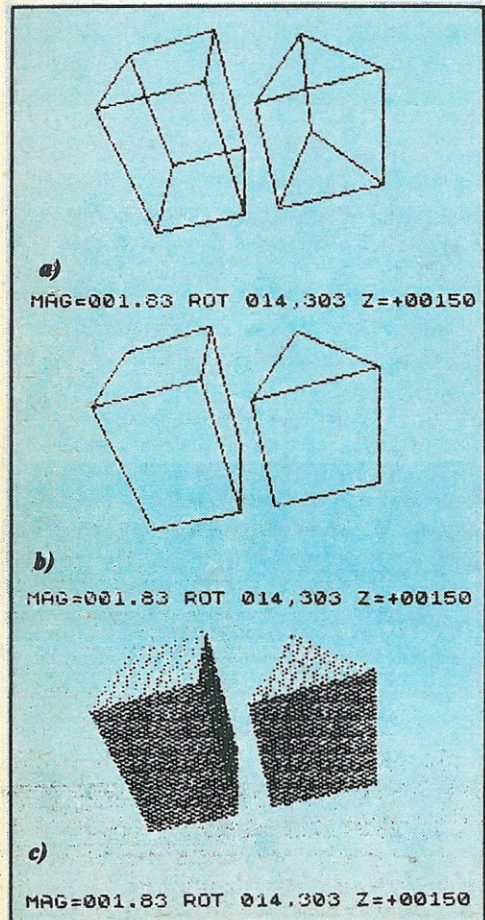
### Gúla rajzolása

Készítsünk most kockánkból egy négyzet alapú gúlát. Ezt úgy érhetjük el, ha a kocka egyik lapját ponttá zsugorítjuk.

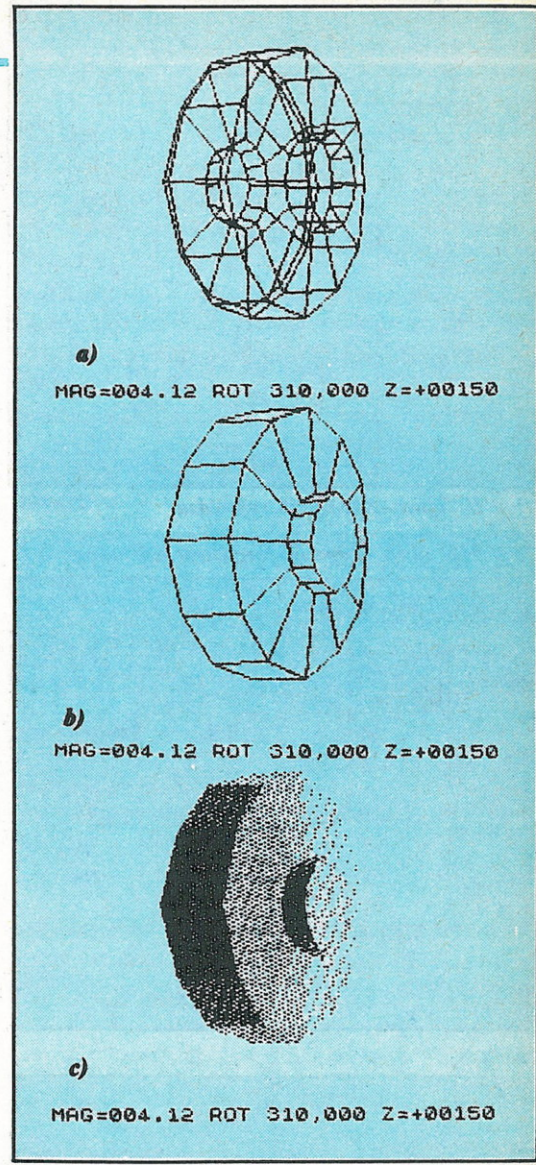
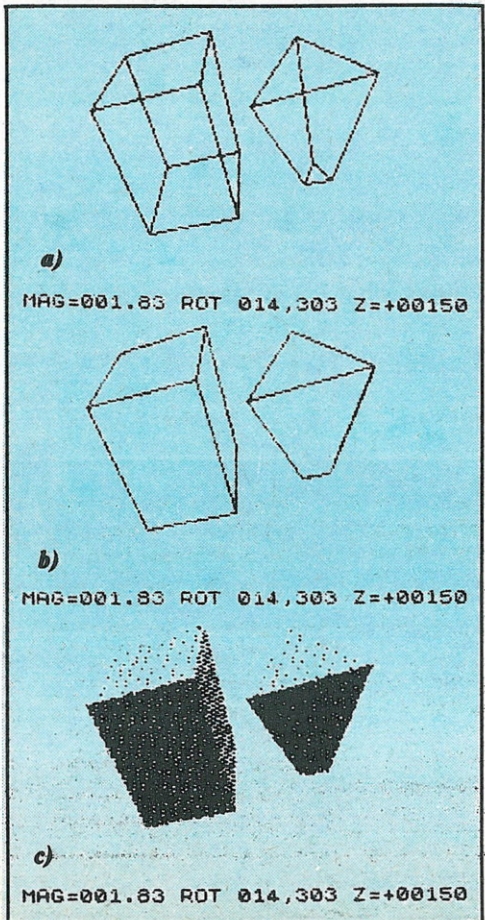
Térjünk vissza a főmenühez és válasszuk annak első pontját (Modify...). Ennél az üzemmódnál előbb a kocka alaplapját látjuk (Z=0), majd a Z koordináta növelése (Next z) után a fedőlapját (Z=80). Mind a két síkidom a kurzorbillentyűkkel mozgatható, nagyítható (Magnify), kicsinyíthető (Reduce). A kép alján olvasható a nagyítás mértéke, a síkidom középpontjának X és Y koordinátája és a Z koordináta értéke.

Zsugorítsuk a kocka alaplapját (Z=0) egy ponttá (Reduce). Hívjuk elő a fedőlapot (Next z), majd térjünk vissza a főmenühez (Quit). Ha megnézzük térben ábrázolva rajzunkat, láthatjuk, hogy egy gúlához jutottunk (6.a,b,c ábra).

8. ábra



9. ábra



10. ábra

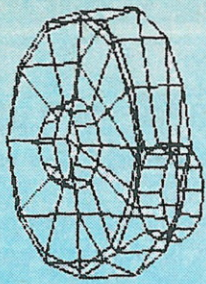
Ferde gúlát úgy készíthetünk, hogy a gúla csúcspontját eltoljuk valamilyen irányba. Ehhez válasszuk ismét a főmenü 1. pontját. A képernyőn megjelenő pontot a kurzorgombokkal toljuk el valamilyen irányba, az alaplapot hagyjuk változatlanul (7. a,b,c ábra). Ezzel a módszerrel készíthetünk ferde hasábot, csonka gúlát stb.

### Több test ábrázolása (II. ábra)

A VU-3D-vel több testet is ábrázolhatunk egyszerre. Rajzoljunk most egy négyzet és egy háromszög alapú hasábot. A négyzet alapú hasáb legyen a magasabb. Először töröljük ki az előző rajzot (Abandon), majd válasszuk a főmenü 4. pontját (Create). Hívjuk elő a kurzort (Open), és rajzoljunk a képernyő bal oldalára egy négyzetet. A négyzet harmadik oldalának meghúzása után fejezzük be a rajzot (End).

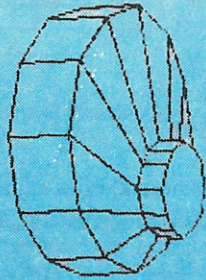
Hívjuk elő ismét a kurzort, és a jobb oldalra rajzoljunk egy háromszöget. A második oldal megrajzolása után szintén End-del zárjuk le a síkidomot.





a)

MAG=004.12 ROT 306,000 Z=+00150



b)

MAG=004.12 ROT 306,000 Z=+00150



c)

MAG=004.12 ROT 306,000 Z=+00150

11. ábra

Azt látjuk, hogy a négyzet szaggatott, a háromszög viszont folyamatos vonalakkól áll. Nyomjuk meg az F billentyűt (Figure). Most a négyzet rajza lesz folyamatos és a háromszögé szaggatott. Újabb figuraváltás után ismét a négyzet lesz szaggatott vonalú. CREATE üzemmódban a Close, Magnify, Reduce parancsok és a kurzorgombokkal történő mozgatás mindig a szaggatott vonallal ábrázolt figurára érvényesek.

Növeljük Z értékét 60 pixelre. Most váltunk figurát úgy, hogy a háromszög legyen a szaggatott rajzú (Figure). Zárjuk le a háromszög alapú hasábot (Close). Növeljük Z értékét 80-ig, és zárjuk le a négyzet alapú hasábot is (Close). Kész is vagyunk, nézzük meg az eredményt (8.a,b,c ábra).

Most készítsünk a háromszög alapú hasázból egy csonka gúlát. Válasszuk a főmenü 1. pontját (Modify). A megjelenő két alaplap közül az előbb már látottakhoz hasonlóan az egyik ismét szaggatott, a másik folyamatos vonalakkól áll. Válasszuk ki a háromszöget (Figure) és kicsinyítsük le MAG=0.20 értékre (Reduce). Növeljük Z értékét 80-ig anélkül, hogy bármi más vál-

toztatnánk. Nézzük meg az eredményt (9.a,b,c ábra).

### Forgástestek rajzolása

A VU-3D csak sík felületekkel tud dolgozni, ezért a programmal körök és körívek nem rajzolhatók. Így a kört kénytelenek vagyunk sokszöggel helyettesíteni.

A 10.a,b,c ábrán látható forgástest egy tizenkét oldalú sokszögből készült. Az ábrán azt érdemes megfigyelni, hogyan változtatható rajzolás közben egy henger átmérője.

Töröljük ki az előző rajzot és rajzoljunk fel a képernyő közepére egy 80 pixel csúcs-távolságú, 12 oldalú sokszöget. A 11. oldal megrajzolása után fejezzük be a sokszöget (End), majd kicsinyítsük le MAG=0.35 értékre (Reduce). Z értékét növeljük 10 pixelre (Next z). Most valahogy jelezni kell a gépnek, hogy itt egy hengeres test vége következik. Ha egyből felnagyítanánk a sokszöget MAG=1.00-re, akkor egy kúpos felületet kapnánk. Nagyítsuk fel tehát MAG=0.40-re, és kicsinyítsük is rögtön

vissza MAG=0.35-re. Ebből megérti a gép, hogy itt a henger vége, és hogy az alaplap méretén nem változtattunk (III. ábra).

Növeljük a Z-t 1 pixellel és nagyítsuk a sokszöget MAG=1.00-re. Most növeljük Z értékét 30 pixelre. Ismét Magnitude/Reduce, majd legyen Z 31 pixel, és csökkentsük a sokszöget MAG=0.40 értékre. Z értékét 40 pixelre növeljük, és végre befejezzük a rajzot (Close).

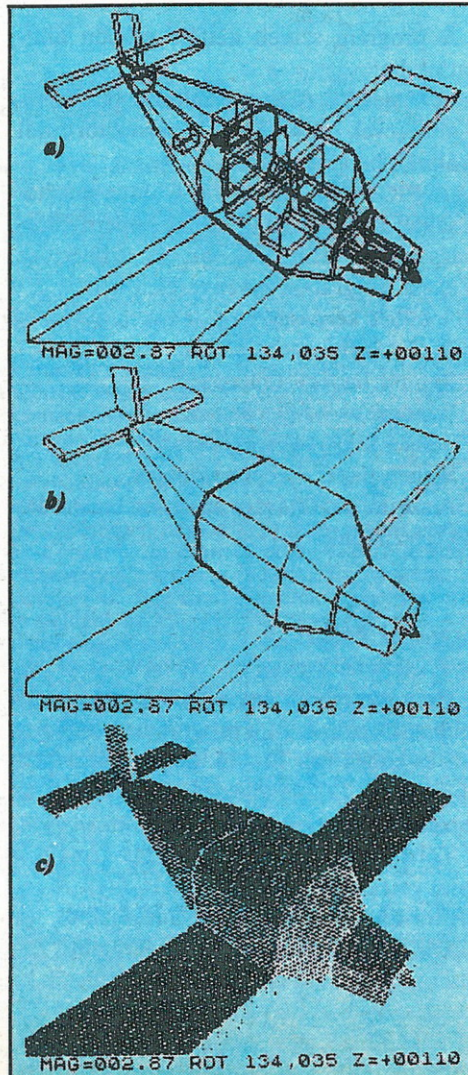
Ezzel a módszerrel készült a program működését bemutató pohár rajza is.

A 11.a,b,c ábrán látható excenter az előző forgástestből készült úgy, hogy az 1. (Modify...) opció segítségével az utolsó csapot a kurzorgombokkal excentrikusan eltoltam.

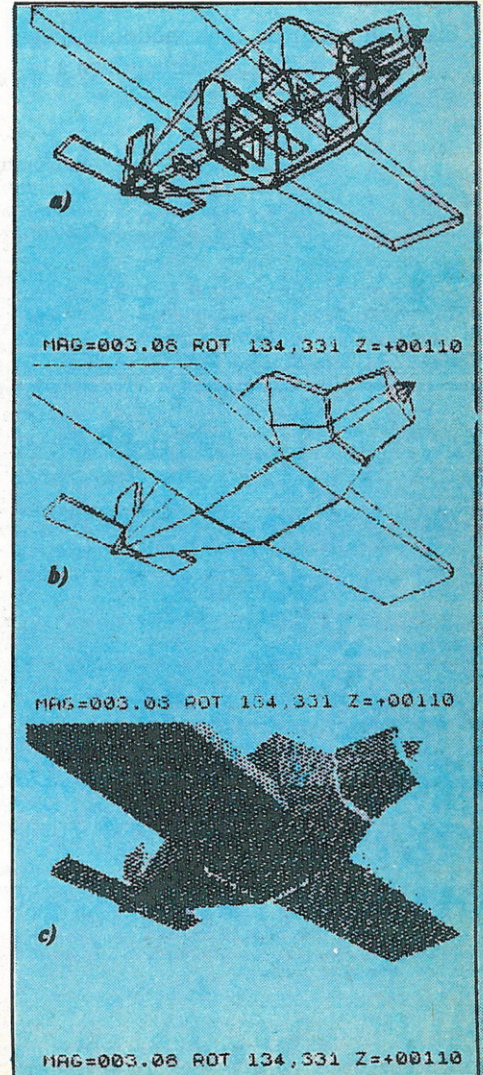
### Bonyolultabb testek ábrázolása

A fenti módszereket összekapcsolva bonyolultabb testeket is ábrázolhatunk. Ennek bemutatására készítettem el egy repülőgép röntgenrajzát (12-15. ábra).

12. ábra



13. ábra





Ha csak mint röntgenrajzot nézzük, nem túl bonyolult, de ne felejtjük el, hogy minden irányból megnézhető (12–13. ábra), és metszetek is készíthetők belőle (14–15. ábra). A Spectrumot mindenesetre alaposan megizzasztja. A kép forgatásánál egy-egy képváltás előtt is a nagyítástól függően 4–6 másodpercig számol. Az árnyékolást két, a nem látható élek kitörlését kb. három percig végzi.

Ha a repülőgép törzséről hosszmetsetet akarok készíteni, könnyen megkaphatom a program „Memory Full” üzenetét. Ilyenkor a program nem áll le. A kép is megmarad elforgatott helyzetében, csak éppen  $MAG=1.00$ -es nagyítással, és  $Z=+256$  pixel távolságra, hogy a számítógép végre egy kis levegőhöz juthasson.

Egy tanács: Mint már említettem, egy test rajzolását a programban nem lehet megszakítani, hiszen ha kilépünk a 4. opcióból, oda már képtörlés nélkül nem juthatunk vissza. Ezért egy bonyolultabb testet jó papíron megtervezni, mielőtt leülnénk a géphez.

## A rajz kimentése

Elkészült rajzunkat több módon is megörökíthetjük. PICTURE üzemmódban a ki-

merített kép screenként magnószalagra menthető (Keep). Ilyenkor a gép felteszi a „Title (név)” kérdést. Rajzunknak egy nevet adva, nyomjuk meg az ENTER-t. Megjelenik a „Start tape . . .” üzenet, és a kép a szalagra kerül a pillanatnyi nagyítás, elforgatás és Z távolság értékével. Az így kapott képernyőkép az üres gépbe is betölthető (LOAD””SCREEN\$), felhasználható saját program címlapjaként stb.

Szintén PICTURE üzemmódban a kimerített képet ki is nyomtathatjuk (Print). A mozgatható képet a főmenü 5. (Save a data file) pontja segítségével menthetjük magnóra. Forgassuk a képet a nekünk legjobban tetsző helyzetbe, mert így kerül a szalagra. Ezután válasszuk az 5. pontot és végezzük el a kimentést.

Az így kitöltött rajz csak a VU–3D programba tölthető vissza a főmenü 2. pontja segítségével. Magnótöltési hiba esetén a program hibaüzenettel leáll. Ilyenkor GOTO 2000 parancsral a főmenühöz jutunk vissza, RUN parancsral a DISPLAY üzemmódbhoz.

## A színek beállítása

A program színeit kétféle módon állíthatjuk be:

1. A főmenü 7. (Change colours) pontja segítségével. Ilyenkor képernyőtörléssel állnak be az új színek.
2. A PICTURE üzemmód Colour utasításával. Ilyenkor az új színek képernyőtörlés nélkül állíthatók be, és szintén az egész programra érvényesek.

A feltett kérdések:

- Border colour = keretszín
- Banner paper = fejlécpapír
- Banner ink = fejlécceruza
- Screen ink = képernyőceruza
- Screen paper = képernyőpapír

A színek beállítására az ismert színekódot használjuk.

## Egy-két ötlet

Ha készítettünk egy olyan rajzot, amely jobban tetszik, mint a programban lévő pohár és kocka (pl. 12–15. ábra), elérhetjük, hogy a VU–3D betöltése után egyből a mi rajzunk jelenjen meg a képernyőn.

Töltsük be a gépbe a VU–3D-t, majd a saját rajzunkat (2. Load a data file). A főmenünél lépünk ki az idézőjelből és STOP-pal állítsuk meg a programot. Ezután vegyünk elő egy kazettát, és töltsük ki az egész gépi kódú részt: SAVE „c” CODE 31 501, 32 183. Így lesz egy VU–3D gépi kódú részünk, ami ragyogóan működik, és

a mi rajzunk van benne, még hozzá a 32 812-es címtől kezdve, 12 330 bajt hosszsan.

Ha még a program kezdőszíneit is át akarjuk írni, ezt is megtehetjük a BASIC program 7. sorában:

```
7 LET b=5: LET pb=1: LET ib=7: LET ps=0: LET is=7
```

Hála a beszédes változóneveknek, könnyen megfejthetjük, hogy a  
pb = banner paper  
ib = banner ink  
ps = screen paper  
is = screen ink

A színek ismertek, rajta vannak a bilentyűkön. Javítsuk még át az 5120-as sort is:

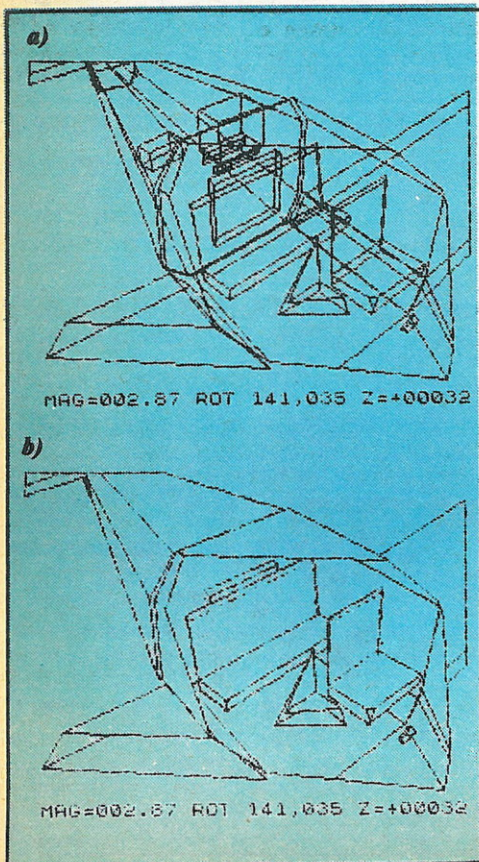
```
5120 LOAD „c” CODE: BORDER x: RUN  
ahol x szintén egy tetszőleges színekód 0-tól 7-ig.
```

Ezután másolóprogram segítségével szalagra tölthetjük az egész programot a következő sorrendben: EXAMPLE (eredeti), example (módosított), címlap (eredeti), c (módosított).

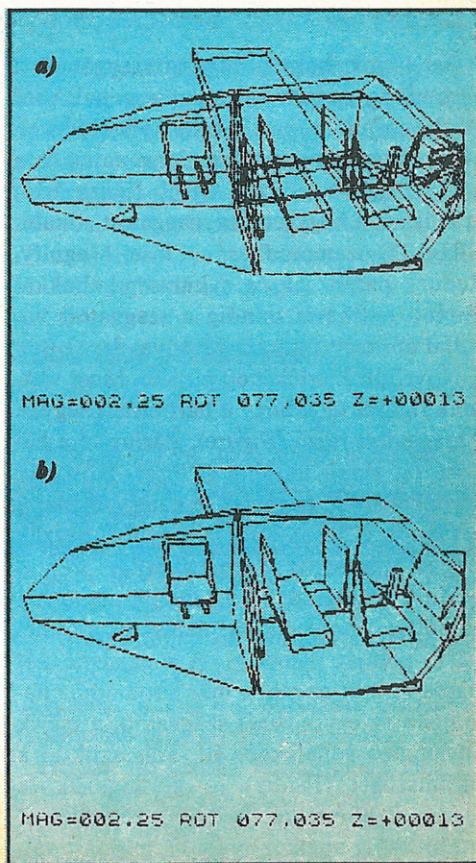
Ezzel leírásom végére értem. Remélem, sikerült újabb ötletet nyújtani a Spectrum-hívőknek ahhoz, hogy számítógépüket minél sokoldalúbban használhassák.

LITAUSZKY GYÖRGY

14. ábra



15. ábra





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

1. a) ábra. Támadó súlyozás világossal

**A**már említettük alapján tudjuk, hogy egy támadó stílusban játszó sakkprogramnak nagyobb esélye van a nyeresre, mint egy védekező stratégiájúnak. Ugyanis a jelenlegi programok gyengéje a középjáték és a végjáték közötti átmenet. Ezt elkerülhetjük éles támadó játékkal, melyben felborítjuk a pozíciós egyensúlyt, és ezzel a parti végső kimenetele gyakran már a középjátékban eldőlt.

Ezt a támadó játékot a megfelelő időpillanatban kell elkezdni. Ennek a pillanatnak a meghatározása nehéz feladat, mivel nagyon sok tényezőt kell figyelembe venni. Külön kell kezelni a zárt és a nyílt állásokat, mindkettőnél más-más szempontokat kell nagyobb súllyal számba venni. Zárt állásnál, ahol a centrumban a gyalogok rögzítik egymást, a szárnyi támadásokat kell előkészíteni, amelyek általában gyalogrohammal kezdődnek. A gyalog-előrenyomulás a tiszték támogatásával a leghatékonyabb. Majd miután lecserélődtek a gyalogok, akkor veszik fel egymással a küzdelmet a tiszték a nyílt vonalakon.

Nyílt állásoknál a gyalogroham veszélyes, mert lassú, és akadályt képezve elzárja a nyílt vonalakon mozgó gyors tiszték útját. Ezzel a támadás lelassul, és így lehetőséget ad az ellenfélnek, hogy átvegye a kezdeményezést.

Mozgó, labilis centrum esetében a legnehezebb meghatározni a támadás stratégiáját, mivel egy ilyen nem tisztázott pozíció átmehet az

1. b) ábra. Támadó súlyozás sötéttel

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

## BITEK ÉS FIGURÁK

### Állásértékelés VIII.

#### Támadó és védekező stratégiájú programok

8	8	10	11	9	8	8	8	
7	10	12	11	8	8	8	8	
6	8	12	13	10	8	8	8	
5	10	13	12	8	8	8	8	
4	7	11	12	9	8	8	8	
3	7	10	9	5	8	8	8	
2	3	6	7	5	8	8	8	
1	2	4	3	1	8	8	8	
	a	b	c	d	e	f	g	h

2. a) ábra. A támadó és a centrum. Súlyozás világossal

előbb említett egyik állástípusba, ahol más-más módszerrel kell támadni. Ezért a jelenlegi sakkprogramok az ilyen hadállásokat ütésekkel vagy a gyalogok rögzítésével igyekeznek megszüntetni. Ez nem helyes, mert ezzel sokszor hátrányosabb pozíciót vállalnak magukra.

Most viszont nézzük a támadó játék motiválásának legegyszerűbb esetét! A centrumellenőrzés mértékének megállapításánál már megismerkedtünk a sakktabla súlyozásának egyes módszereivel. Jelenleg is egy hasonló fogunk használni, csak azzal a különbséggel, hogy a súlyozó értékek nem a centrum felé fognak növekedni, hanem a támadás célpontja felé. Egy ilyen célpont lehet például az ellenfél térfele egyenes eloszlással. Ilyen súlyozást látunk az 1. a,b ábrán.

Az ilyen egyenes eloszlású táblának hátránya, hogy a figurákat az ellenfél tábora felé irányítja, különösebb célpont nélkül. Ezáltal a különböző szárnyakon lévő figurák az ellenfél térfele felé igyekeznek anélkül, hogy bármiféle összehangoltság lenne közöttük. Így csak azt érzük el, hogy a saját térfélen lévő mezőket túlságosan meggyengítjük, és ezzel lehetőséget adunk az ellenfélnek, hogy ezeken a gyenge mezőkön megvesse a lábát és behatoljon állásunkba. Tehát ha mégis ezt a módszert alkalmazzuk, akkor még más tényezőket is figyelembe kell venni, ami

8	1	3	4	2	8	8	8	
7	5	7	6	3	8	8	8	
6	5	9	10	7	8	8	8	
5	7	9	12	11	7	8	8	
4	8	12	13	10	8	8	8	
3	10	13	12	8	8	8	8	
2	8	11	12	10	8	8	8	
1	9	11	10	8	8	8	8	
	a	b	c	d	e	f	g	h

2. b) ábra. A támadó és centrum. Súlyozás sötéttel

biztosítja a figurák együttműködését: jutalompontot adhatunk például akkor, ha az ellenfél térfélen lévő egyik mezőt két vagy több különböző bábuval támadjuk. Hasonlóan akkor is, ha az egyes figurák által megtámadott mezők „közel” esnek egymáshoz. (A „közel” mennyiségi megállapítása a programbelövés feladata.) Ilyen módszerekkel megakadályozhatjuk haderőnk szétűzését, szétesését. Igaz, több olyan kombináció is előfordul a sakkban, aminek lényege, hogy egyetlen figurával megtámadjuk az ellenfél két figuráját, és ezáltal anyagi értéket nyerünk, vagy matthelyzetet hozunk létre. Ez azonban változatos számítás igényel, ezért a program ezt nem hagyja ki, annak ellenére, hogy tisztjei kis időre eltávolodtak egymástól, és így harmóniájuk felbomlott.

Precízebb táblasúlyozást kapunk, ha a centrumtáblát összevonnuk ezzel a támadó súlyozású táblával, vagyis a kettőt összeadjuk (2. a,b ábra). Ezzel időt takarítunk meg, mert nem kell külön-külön számítani az értékeket.

Ugyancsak időt takaríthatunk meg, ha ezt még a királytámadási táblával is összevonnuk (3. ábra), csak ennek hátránya, hogy minden királylépés esetén újra kell a súlyozó értékeket számítani.

A tapasztalat mégis azt mutatja, hogy megéri ez a megoldás, mert a király egy játszma folyamán keve-

set lép. Az algoritmusba, mely a lépésváriációkat vizsgálja, viszont beépíthetjük, hogy csak akkor módosítson ezen a súlyozáson, ha a király az eredeti helyétől legalább két lépést megtett. Ez nagyon ritka eset, mert egymás után kétszeri királylépésre általában csak a végjátékban kerül sor, ahol viszont a királytámadási súlyozás már felesleges. Nincs anyag a mattadáshoz; ilyenkor a gyalogbevétel a cél. Erre viszont a sima támadótáblát célszerű használni, mert ezzel a programot a gyalogok előretolására készítjük, így gyorsítjuk a gyalogbevételt.

Az előbbieken alapján el tudjuk képzelni, hogy ezen az alapon védekező stratégiájú programot is lehet készíteni. Ebben az esetben viszont magasabb értékkel a saját térfélen lévő mezőket látjuk el. Ezáltal a program kerülni fogja a gyalogokkal való előrenyomulást, és a tisztjeit is a saját alapsora közelében helyezi el.

A megfelelő táblasúlyozás kikísérletezése sok időbe kerül, míg megtaláljuk a leghatásosabbat.

A statikus állásértékelés általában egy szimmetrikus függvény. Ez azt jelenti, hogy amilyen szempontot az egyik színnel számításba vetünk, azt a másik színnel is számításba kell vennünk, csak ellentétes előjellel. Tehát egy értékelésnél mindkét színnel figyelembe kell

8	8	11	15	17	8	8	8	
7	10	14	19	16	8	8	8	
6	8	13	17	14	8	8	8	
5	10	15	14	10	8	8	8	
4	7	12	13	10	8	8	8	
3	7	10	9	5	8	8	8	
2	3	6	7	5	8	8	8	
1	1	2	4	3	1	8	8	
	a	b	c	d	e	f	g	h

3. ábra. Támadó és királytámadási súlyozás összevonnása

venni a centrumértéket ugyanúgy, mint a támadóértéket. A világos és sötét pontszámának különbsége adja magának az állásnak az ilyen komponensekből álló pozíciós értéket.

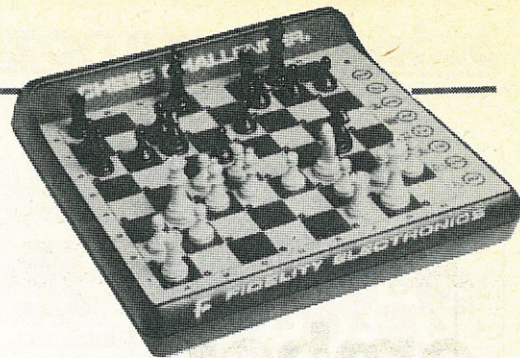
Ezt a szimmetriát viszont megbonthatjuk a támadótábla súlyozásával. Egy játszma alatt mind a két színnel külön-külön, másfajta súlyozást is használhatunk. Például a program figyelheti az ellenfél játékát, és ennek mértékében súlyozza a tábláját. Vagyis, figyelemmel kíséri az egész játszmát, és az előző lépések alapján kielemezni az ellenfél játékstílusát, amely lehet védekező, támadó, közömbös vagy várakozó. Itt a közömbös vagy várakozó játéktíluson azt értjük, hogy a játékos mindenféle konkrét terv nélkül lépked figuráival, és az ellenfél akciójára vár.

KOVÁCS P. ATTILA



## GÉPI ELLENFELEINK

### Chess Challenger



A Chess Challenger Sensory 9

Legutóbbi cikkeinkben az NSZK-bei Hegener+Glaser népszerű Mephistóit ismertettük, amelyek jelenleg a világ élvonalát képviselik. Utolérték és el is hagyták az USA-beli Fidelity készítményeit, amelyek éveken át világségek voltak. Fidelity azonban ma is a legerősebb versenytárs.

#### Történelmi érdem

Anélkül, hogy a számítógépes sakknak vagy akár csak a mikroszámítógépek sakk tudásának fejlődését ismertetni kívánnánk, szólnunk kell a legelső sakkmikró megjelenéséről. Hiszen azóta bő tíz esztendő telt el, és az első készülék neve ma is közzsájon forog.

E sorok írójától sokan érdeklődnek bizonyos típusú sakkszámítógépek képességei, ára felől, mert el szeretnének adni egy régi készüléket, vagy vásárolni akarnak egy újat. De ha a kérdező csak azt mondja, hogy Chess Challenger típusú készüléke van, ne csodálkozzon a kitérő válaszon. Ilyenkor ugyanis csak azt mondhatjuk: ez körülbelül úgy hangzik, mintha arra a kérdésre, hogy milyen autója van, azt mondaná: Volkswagen. A Chess Challengereknek is tucatnyi fajtája került ugyanis az évek során forgalomba, és az elsőtől az utolsóig akkora volt a változás, mint mondjuk a legöregebb bogárhátú Volkswagentől a Golfig.

Az első gyár, amely mikroprocesszorral működő sakkszámítógéppel megjelent a piacon, a Mi-amiban székelő Fidelity Electronics (ma Fidelity International) volt. 1976-ot írtak: a készüléket Ronald C. Nelson építette. A számítógépes sakk történelmi fordulója volt ez! Csakhamar megjelentek a továbbfejlesztett típusok: a Chess Challenger 3 és 10, majd a 7.

Már itt felhívjuk az olvasó figyelmét, hogy a típust kísérő számokból semmire sem lehet következtetni. Ezeket teljesen ötletszerűen bigyesztették a név mögé, és amint a továbbiakból is kiderül, a magasabb szám nem jelent fejlettebb vagy akár újabb típust. Az eddig felsoroltak — a 7 is, amelyből még néhol megjelenik egy-egy példány —, mind elavultak. Ugyanezt mondhatjuk a szám nélküli, Voice vagy Sensory melléknevvel ellátott készülékekre, valamint a Decoratorra, melyek itt-ott még szabálytalan lépéseket is tesznek.

A Chess Challenger Champion Sensory, nyomtatóval és Gedeon Béla fehér dobozban elrejtett gyorsítóval



A technikai fejlődést azonban tükrözi, hogy a Voice hangot ad, bemondja a megtett lépést, néha megjegyzést is fűz hozzá, tréfásat, szórakoztatót; a Sensory pedig, amint nevéből is következik, az első szenzoros típusú Chess Challenger volt. Ehhez nem kellett külön sakkasztalt használni a lépések megtételére. A számítógép maga, akárcsak a régi Mephistók, nyomógombokkal működik, és kijelzője mutatja a megtett lépéseket. A Decorator nem más, mint a régi Voice, csak díszesebb kivitelben: fából készült, nem műanyagból, mint a többi, olcsóbb készülék.

Ha az olvasónak ezek közül kínálnak egyet megvételre, azt ajánljuk, ne foglalkozzék vele. Az első Fidelity gyártmányú sakkszámítógép, amely ma is versenyképes, és jó sakköző számára is partner, a Chess Challenger Champion Sensory nevet viseli.

#### Az első sakkvilágbajnok

1980-ban, Londonban rendezték a mikroszámítógépek első sakkvilágbajnokságát, amelyet a győzelem után Champion melléknevvel ellátott Chess Challenger nyert meg. Ez azonban nem továbbfejlesztett, hanem merőben új készülék volt. Megszületésének előzménye: a Fidelity alkalmazásába lépett Kathe és Dan Spracklen, akik a Sargon programokkal elért kiemelkedő sikereikkel a mikroszámítógépes sakknak ekkor már koronázatlan királyai voltak. A Sargonokkal itt nem kívánunk foglalkozni, mert ezek — a Fidelity részére továbbfejlesztett programokkal párhuzamosan — továbbra is a piacon maradtak, és megjelentek újabb verziók is. A Sargonokra tehát visszatérünk.

A Spracklen házaspár teljesen új programot készített a Champion számára. A megnyitási könyvtárat, a lépésgenerátort és az értékelőfüggvényt — tehát lényegében a programnak minden részletét — újraírták. Kathe Spracklennek saját, annak idején hozzánk intézett soraiból idézünk: „A program a korábbinál gyorsabb, taktikai készsége jobb, könnyebben továbbfejleszthető újabb sakkbeli kritériumokkal, egyszerűbb a hadállások felállítása.” A hardver sebességét is a korábbi 1-1,5-ről 2 MHz-re emelték.

A piacon a CC Champion Sensory számítót hosszú időn át a legjobb sakk-mikroszámítógépnek. A következő, Travemündében rendezett világbajnokság „kereskedelmi” csoportjában a Scisys cég Mark V-je megelőzte ugyan, de ennek hardverproblémái miatt változatlanul a Champion volt a világszerte legelterjedtebb sakkszámítógép. Technikai szempontból is megelőzte vetélytársait, hiszen amint legutóbb írtuk — a Mephistó készülékek, s a NOVAG számítógépei is később tértek át a szenzoros módszerre. Igen könnyen kezelhető; az indulási és érkezési mező könnyed megnyomásával kell a lépéseket megtenni, s ezeket — csakúgy, mint a gép válaszlépéseit — a mezők bal alsó sarkában levő dióda kigyulladására jelzi. A bábszimbólumok megnyomása útján lehet hadállásokat is felállítani. Lépéseket vissza lehet venni (legfeljebb 19-et), a sakkóra akár a felhasználó, akár az időkontrollig még rendelkezésre álló időt jelzi. Számol azalatt is, amíg az ellenfél a lépés sora. Kilenc játékfokozata van, ami különböző gondolkodási időket jelent a lépésenkénti 10 másodperctől a 3 perces versenyfokozaton át a végtelenig, vagyis az időhatár nélküli elemzésig, és van mattkereső játékmódja, amely feladványok megfigyeltetésére szolgál. Vagyis rendelkezik mindazokkal a legfontosabb tulajdonságokkal, amelyeket egy korszerű számi-

tógéptől elvárunk. A tábla és a készlet fából készült. Eredeti ára 1000 DM körül volt. Ma már nem árúsítják, alkalmi vételben 3-400 DM körüli árat kérnek érte, amit feltétlenül megér.

Mint érdekességet említettük meg, hogy Gedeon Béla, a Mikroelektronikai Vállalat gyöngyösi gyáregységének osztályvezető mérnöke készített a készülékhez egy kiegészítőt, gyorsító berendezést, amely a mélyebb számításot segíti elő. Készülékével — a Fidelity gyár hozzájárulásával — részt vett az 1983-ban Budapesten rendezett 3. mikroszámítógép világbajnokságon, és 4 pontjával az élmezőnyben végzett, pedig valamennyi résztvevő közül a legrégebbi típust képviselte.

#### Sensory 9

1983-ban jelent meg az üzletekben a Chess Challenger Sensory 9 típusú sakkszámítógép, amelyet az 1983. évi vb-n ismerhettünk meg. A számos nagyobb teljesítményű kísérleti készülék között 3 pontot gyűjtött, ezzel elnyerte a legjobb, kereskedelmi forgalomban lévő készülék részére kitűzött különdíjat. (Gedeon Béla készüléke kísérletinek volt minősítve.) Azóta számos teszt, versenyeredmény és egyéb tapasztalat bizonyította, hogy ez a készülék ugrásszerű fejlődést jelentett az ember gépi ellenfeleinek sorában. Már megjelenések között a Spracklen házaspár, hogy kísérleti párosmérkőzésen a Sensory 9 a Campionnál 15:5 arányban bizonyult jobbnak. Egyszerű műanyag kivitele és a fejlettebb program által lehetővé vált technikai egyszerűsítésének következtében a Championnál lényegesen olcsóbban, 4-500 DM körüli áron került forgalomba. Meg kell említenünk, hogy az amerikai készülékek NSZK-beli ára — amely mellett a könnyebb összehasonlítás kedvéért, s mivel honfitársaink a leggyakrabban ott vásárolnak, megmaradunk —, a dollár árfolyamának ingadozása következtében gyakran változott.

A Sensory 9 16 k ROM és 2 k RAM méretű tárolóval rendelkezik, szemben a Champion 20 k ROM és 3 k RAM-jával. 1,5 MHz-en fut; a Champion, amint említettük, 2 MHz-en. Nem ad hangot, s — ami nem előny —, nem jelzi az időt. Belső órája természetesen van, hiszen szintén kilenc fokozatra állítható. Új képessége, hogy jelzi a döntetlent minden olyan esetben, amikor a sakkszabályok azt előírják. Tehát felismeri a patot, a háromszori hadállás-ismétlést, és ismeri az 50 lépés szabályt is. Egyenlő állás esetén döntetlent ajánl, illetve ilyen ajánlat fölött dönt. Reménytelen állásban a játszmat feladja. Mindezeket a diódák kigyulladására jelzi, megfelelő alakzatokban. Akkor is jelez, ha hét lépésen belül — bármilyen fél részére — védhetetlen mattot lát. Mindezek mellett jelentősen nőtt a korábbi típusokhoz képest a készülék játéktudása, főként taktikai téren. Mintegy 3000 hadállást tartalmaz megnyitására, s korlátozottan moduláris szerkezete lehetővé teszi kiegészítő — például megnyitási — modulok beiktatását.

Újonnan ezt a készüléket sem árúsítják már. De feltétlenül megfelel egy közepes sakköző igényeinek, és a 2-300 DM körüli alkalmi vétel érdemes meggondolni. A Sensory 9-et leginkább a gyár saját újabb típusai szorították ki a piacról, elsősorban a hatalmas karriert befutott Elite-sorozat, amelynek ismertetésére legközelebb kerül sor.



Sokszor beszélünk arról, hogy az „informatikai forradalom” mennyire megkönnyíti majd munkánkat a nem is olyan távoli jövőben. De erre fel kell készülni, mert nincs annál veszélyesebb, mint amikor valaki olyan eszközt kap a kezébe, amelynek használatára még éretlen.

# A kiszolgáltatott ember

A tokiói utca képe megváltozott a május eleji csúcstalálkozó közéletével. A szokásosnál több rendőr teljesített szolgálatot. A rendőrség felszerelése korszerűsödött, elsősorban a számítástechnika lehetőségeinek felhasználásával, ami a mikroelektronika hazájában nem meglepő. A japán hatóságok több mint harmincezer, szélsőséges nézeteket valló potenciális terrorista adatait tartják nyilván, akiknek jelenléte a csúcstalálkozó biztonsága miatt nem volt kívánatos. Tokióban is mindennaposá vált a Svédországban megszokott kép: a személyi igazolvány adatait vagy egyéb ismertető jegyeket beillentyűzik a rendőrségi autó terminálján vagy a rendőrök rádiókészülékének billentyűzetén, és máris az intézkedő közegek rendelkezésére áll mindaz, amit a gyanúsak tűnő emberről a hatóság adatbankjainban nyilvántart.

A technika azonban ennél is érdekesebb ellenőrzési rendszerek kidolgozására is lehetőséget nyújt. Angliában és az NSZK-ban sikeres kísérletek folynak olyan tévékamerás komplex rendszerek kifejlesztésére, amelyek képesek felismerni az előttük elhaladó gépjárművek rendszámait és összevetni a körözött, valamint a megfigyelt járművek jegyzékével. Egyezés esetén riasztják a mozgó őrszemeket.

Az elektronizálás nemcsak az üzemek védelmét könnyíti meg a jogtalan behatolástól elektronikus kártyával, hanggal, ujjlenyomattal vagy mondjuk a szemfenék érinthetőségét felhasználó azonosító berendezésekkel. Egyes országok elektronikus olvasható útlevél készítenek vagy éppen géppel feldolgozható vízumot ütnek be a különben hagyományos útlevélbe. Így amikor a bevándorlási hatóság útlevélvizsgálója látszólag hanyagol néhány pillanatra szöveggel lefelé kinyitva a pultra teszi az útlevélet, az azonosítás automatikusan végbemegy, minimálisra csökkentve a tévedés esélyét, és egyúttal megnehezítve az útlevél-hamisítást is. Nem elég ugyanis tökéletes, mindenben megfelelő kópiát készíteni, a hamis adatokat valamilyen módon be kell vinni a számítógépes rendszerbe is.

Amint az állampolgár egy informatikai társadalomban kapcsolatba kerül az állammal vagy vállal-

attal, kitörhetetlen azonosítóval, amely egész életét végigkíséri, és legbensőbb magánéletének titkai is feltáruznak, ha valamely hatóságnak szüksége van rá. Ehhez senkinek nem kell külön hatékony apparátust üzemeltetnie, csak megszereznie kell a szétszórtan meglévő részinformációkat.

Pár esztendeje azt mondta valaki az egyik számítástechnikai kongresszuson: „A hitelkártya és a bankcsekk a legtökéletesebb eszköz, amely nyomon követheti az egyén mozgását a világban.”

Mr. Smiss évek óta az American Express hitelkártyájával rendelkezik. Különmunkáiból származó jövedelme és fizetése bankszámlájára érkezik, ahonnan rendszeresen automatikusan átutalják hitelkártyájának fedezetét. Mr. Smiss számára mindez nagyon kényelmes. Nem kell magánál készpénzt hordania, fizetéseit a kártya számának telefonba történő bemondásával vagy a kártya felmutatásával egyszerűen végzi. A kártya használatát különleges kódjai és a legújabbakba már beépített mikroprocesszor teszi egyszerűvé és biztonságossá. Smiss úr élvezeti a kényelmet. És ahol megfordul, nyomot hagy maga után, amely belekerül a számítógépek emlékező egységeibe. Egy idő után a hitelkártya-intézet elkészíti mozgásprofilját, és ha például a különben városából alig ki-mozduló Smiss kártyáját egyik nap Párizsban, másnap mondjuk a Bahama-szigeteken használják fel, a számítógépes rendszer jelzi az elfogadónak, hogy valami nincs rendjén, igazoltatni kell a kártya felmutatóját.

Smiss úr közben észre sem veszi, hogy bevételeit ellenőrizheti az adóhivatal fürkész szeme, vásárlásaiból megállapíthatják jellemét, családi körülményeit, de még egészségi állapotát is. A bankszámla forgalma nemcsak arról ad eligazítást a hatóságoknak, hogy milyen jövedelmétől kopaszthatják meg az állampolgárt, hanem a bankoknak támpontot adhat arra, mennyire fegyelmezett fizető, mennyire hitelképes az ügyfélük.

A telefonon vagy csomagküldő áruházakból történő rendelések szintén nyomot hagynak a számítógépek memóriájában. Értő kezekben ezek a különben jelentéktelen dolgok meghatározzák, milyen címlistára kerül fel, konzervatív



vagy liberális beállítottságú-e (mert akkor ez vagy amaz a párt szólítja fel reklámlevélben, hogy szavazzon rá), vagy egyáltalán érdemes-e a biztosítóintézeteknek betegbiztosítási vagy életbiztosítási ajánlattal megkeresni őt. Ilyen információk a kórházak és a betegbiztosítók adatbankjából is beszerezhetők.

Mr. Smissről halmozódnak az adatok. Anélkül, hogy valaki személy szerint foglalkozna vele, minden lépését szemmel tartják. Munkahelyének számítógéprendszere nyilvántartja ellankadásait, késéseit, pihenőidejét. Esetleg automatikusan adja a fegyelmet és az elbocsátást. De mindez, amíg az információk nem kerülnek egy helyre, nem nagy baj. Az adatok ugyanis csak összességükben képeznek olyan titkot, amely a magánélethez való jogot sérti. Ide azonban már nem kell más, mint hogy valahol valaki kiadja a parancsot, hogy a még (szerencsére) külön működő adatbázisokat össze kell kapcsolni ezen a területen is. Így az állampolgár ideálisan ellenőrzötté és irányítottá válhat. Remélhetően elég érett lesz az emberiség ahhoz, hogy ezt megakadályozza.

Az információkkal való kereskedés a fejlett tőkés országokban jelentős üzletgá vált. Cégek adnak és vesznek különböző szempontok szerint összeállított címlistákat. A magánélet helyenként annyira összeszűkült, hogy már nem magánügy: ki, mit, mennyiért és hol vásárolt, vagy milyen az egészségi állapota.

A beszéd felismerő rendszerek, az automata telefon- és telekommunikációs központok fejlődése is kétoldalú. Egyrészt megkönnyítik az ember és a technika, valamint az ember-ember kommunikációt, de egyúttal lehetőséget nyújtanak egy ország teljes hírközlésének ellenőrzésére is. Már korábban is fi-

gyni tudták egyes gyanús személyek telefonbeszélgetését, de ez nem volt általános. Később megteremtődött annak a lehetősége, hogy olyan ellenőrző berendezéseket készítsenek a mikroelektronika eszközeivel, amelyek egy-egy „hívószó” alapján rögzítik azt a beszélgetést, amelyben az a szó elhangzott, és felhívják rá az illetékesek figyelmét, hogy alaposan, emberi munkával ellenőrizzék le a hívót.

A távközlésbe való technológiai beavatkozás ezen túlmegy. Mi szerencsére ennek csak egy olyan primitív formáját ismerjük, amikor a vállalat ügyintézőinek telefonjairól a „takarékoskodás” örve alatt ki-kötetik a távhívás lehetőségét. De mit szóljon az az angol munkavállaló, akinek minden munkahelyi telefonját dokumentálja egy újonnan kifejlesztett elektronikus központ, és egyben megakadályozza a gyakran hívott magántelefonszámok felhívását is. Mint a gyártó cég önelégülten hangoztatta: rendszerük tízezer számot képes nyilvántartani.

A pénz is „szagot kapott” az elektronika fejlődésével. Korábban, ha útját ellenőrizni akarták, ezt a számok és sorozatok hosszadalmas kézi listázásával, majd e „tengeri kígyó” számjegyzékek figyelésével teheték meg. Így érthető, hogy egy-egy átadott váltásdíj vagy bankrablásból származó pénz felszívódhatott. Rövidesen ez is múlttá válik. Egy-két éven belül a dollár bankjegyek magasabb címleteinek felső részén fém szürke sáv fog megjelenni: a pénzre géppel olvasható formában holografikusan rögzített sorozat- és sor-szám. Ezután már csak elektronika kérdése, hogy akár minden bankjegy forgalma külön-külön is követhetővé válik.

A pénzáttalási rendszereknek kényelmünket, anyagi biztonságunkat, a kommunikációs rendszereknek egymás jobb megértését kell(ene) szolgálniuk. Felöltünk egy új informatikai kor küszöbére, de a felnöttség önmagában nem jelenti azt, hogy tudunk is bánni magával az eszközzel. Ezt meg kell tanulnunk úgy, ahogy a vadász megtanul bánni a puskával. Ahhoz azonban már kellő etikai érzék és morális felkészültség kell, hogy az új eszközt pillanatnyi haragunkban, félelmünkben, bizonytalanságunkban ne fordítsuk embertársaink ellen.

Ha a számítógéprendszerek nem hibáink feltétlen dokumentálását — mint némelyik rugalmas munkaidőt nyilvántartó rendszer teszi —, hanem munkánk segítését, az összefüggések megkeresését, a mechanikus munka alól való megszabadulást jelentik, akkor képesek leszünk valóban emberhez méltóan belépni abba az elektronikus korbba, amely nemsokára nálunk is megkezdődik.



A személyi számítógépes helyi hálózatok gyártói egyre újabb rendszereket jelentenek be, és növekszik a régebbi rendszerek modernizált vagy javított változatainak száma is. A jelenlegi irányzat egyértelműen az IBM vonal erősödése, ezért sorozatunk e befejező része is elsősorban az IBM kompatibilis helyi hálózatokkal foglalkozik. (A sorozat az 1986. szeptemberi számban indult. — A szerk.)

Néhány „nem igazi” helyi hálózatról is szó lesz, amelyek bizonyos mértékig eltérnek az előző részekben ismertetett klasszikus helyi hálózati megoldásoktól, de ezeket kedvező árú és szolgáltatásai miatt ma is sok helyen alkalmazzák.

## A Net One hálózat

A Net One hálózat az Ungermann-Bass amerikai cég terméke, alapsávú Ethernet típusú helyi hálózat. Újabb változata a Net One Personal Connection, mellyel IBM PC-k is rendszerbe kapcsolhatók. A hálózat egy vagy több szerverállomást (lehet IBM PC XT) és munkaállomásokat tartalmaz, hálózati csatlakoztatását a PC-be dugaszolható hálózati csatlóegység valósítja meg.

A Net One hálózatban élő szerverfunkciók: az osztott fájl- és nyomtatókezelés, valamint egy másik helyi hálózathoz való csatlakoztatást biztosító gateway (kapu) funkció. Nem szükséges, hogy a szerverek „dedikáltak” legyenek; dedikált szerver ugyanis nem tölthet be munkaállomás-funkciókat is a fájlkezelés mellett.

Az alapsávú hálózat koaxiális vagy optikai kábelen 10 Mbps sebességgel működik, a hálózatba kapcsolható állomások száma 64, az áthidalható távolság max. 2,5 km. Létezik egy széles sávú (frekvenciaáthelyezéses) Net One hálózat is, amely sodrott érpárú kábelen, 5 Mbps sebességgel, CSMA/CD protokollal működik; az áthidalható távolság max. 9 km. Mindkét változat MS-DOS operációs rendszer alatt működik. A széles sávú hálózat csatlóegységén kívül a rendszerhez rádiófekvenciás modem is tartozik.

## 3 Com Ethernet hálózat

A 3 Com Ethernet típusú helyi hálózatban az adatátviteli közeg szintén koaxiális vagy optikai kábel, az átvitel sebessége 10 Mbps, a a bekapcsolható legnagyobb állomátszám 64, a max. kábelhossz 2,5 km és a működőtétő operációs rendszer ugyancsak MS-DOS. A rendszer Ether Share, illetve Ether Print programjai biztosítják a hálózat működését, valamint az osztott fájl-, illetve nyomtatáskezelést, soros vagy párhuzamos nyomtatóval. Az Ether Share program a hálózat egyik munkaállomásáról tölthető

be rendszerindításkor. A hálózati szerverállomás IBM PC XT vagy AT alkalmazása esetén munkaállomásként is használható (nem dedikált szerver), ami a tapasztalatok szerint nem lassítja számottevően a rendszer működését.

A hálózatba archiváló tárként (backup-ként) 36—252 Mbájt kapacitású szalagegység is bekapcsolható. A rendszer hozzáférési jogosultságát jelszavakkal védik; a lemezterületek vagy közös vagy kizárólagos hozzáférésűek. Az egyidejű írás ellen rekordzár véd.

A 3 Com Ether Mail programja segítségével a hálózatban elektronikus levelezés is folytatható: a névre szólóan elküldött üzeneteket ki-ki lehívhatja saját elektronikus levélládájából.

## Omninet hálózat

A gyártó Corvus cég első „helyi hálózata” a Multiplexer volt, amely azonban még nem igazi osztott rendszer, hiszen a többfelhasználós konkurens lemezkezelést a lemezterület előzetes felosztásával (patricionálással) oldotta meg.

Az Omninet max. 2,5 km hosszúságú sodrott érpáron 1 Mbps sebességgel, CSMA/PA hálózateléréssel működő, 64 munkaállomásos rendszer. (A CSMA/PA hálózatelérési módszer hasonló az 1986. októberi számban ismertetett CSMA/CD-re, azzal az eltéréssel, hogy az adatsomagot akkor tekinti sikeresen célba jutottnak, ha az ellenállomásról visszakapott egy pozitív nyugtát. Innen száramzik a PA, azaz Positive Acknowledgement megnevezés is.) Az Omninet hálózathoz IBM PC-k, Apple II és III sorozatú gépek, a Texas Instr. professzionális gépei, valamint a DEC Rainbow gépei tartozhatnak, akár vegyesen is. A hálózat előnye az alacsony ár, továbbá hogy különböző operációs rendszerű (DOS 3.1, CP/M/Apple, CP/M—86, UCSD—P) gépek kapcsolhatók vele rendszerbe.

Az Omninet hálózatvezérlő programja a Constellation, melynek I-es változata csak is Apple gépekhez készült, a II-es változat viszont már az IBM PC-ket is kezeli. Nem azonos típusú gépekhez mindenképpen a Constellation II-re van szükség.

A hálózati lemezkapacitás 5—126 Mbájt között változhat, archiválási célokra a szalagegység (200 Mbájt) áll rendelkezésre. A hálózat megbízhatósága igen jó, amit a nyugtázásos hálózat-hozzáférési rendszerrel valósítottak meg (CSMA/PA).

A hálózati szerverállomás dolga az osztott lemezegység- és nyomtatásvezérlés, egykártyás Z80-as vezérlőegységével. Az Omninethez max. 8 szerver kapcsolható, a rendszer hozzáférési jogosultságát a felhasználói név, illetve a jelszó biztosítja.

A központi számítógéppel való kapcsolatot SNA gateway (például IBM PC AT) teremti meg, így az egyes munkaállomások a

rajtuk futó IBM terminálemulációs programok — például IBM 3270 — segítségével virtuális terminálkapcsolatot létesíthetnek a központi géppel, és annak termináljaként adatokat vihetnek át, hívhatnak le, illetve programokat futtathatnak rajta. A gateway biztosítja, hogy a fenti szolgáltatást egyidőben több felhasználó is igénybe veheti.

## Point 32

A hálózatot három változatban az angol Apricot készíti. A hálózat az Omninet rendszert követi, sintopológiájú, az átviteli sebesség 1 Mbps, a kábel sodrott érpárú, a hálózatelérés CSMA/CD rendszerű, a telepítési távolság 600 m, illetve vonali erősítőegység beiktatásával 1200 m lehet. A rendszerbe kapcsolható munkaállomások száma 32/10 hálózat esetén max. 16 (optimálisan 4-6), a 32/20, illetve 32/80 hálózatban 32 (optimálisan 8-12). A hálózatba max. 10 szerverállomás kapcsolható be. A hálózati szerverek dedikáltak, azaz csak osztott lemez- és nyomtatókezelést végeznek, munkaállomásként nem használhatók (nincs se képernyőjük, se billentyűzetük). A szerver memóriája 512 kbájt, a rendszerhez 10, 20, illetve 80 Mbájtos lemezegység, valamint 3,5"-os floppy tartozik. A kisebb teljesítményű hálózatok háttérkapacitása utólag is bővíthető, 80 Mbájt-ra.

A hálózatba osztott hozzáféréssel V24, illetve Centronics interfészű nyomtatók (például Epson) kapcsolhatók. A rendszerben tárolt fájllokhoz a felhasználók olvasáskor konkurens módon férhetnek hozzá, az egyidejű írás ellen pedig rekord szintű a védetség. A rendszer egyszerűen bővíthető: új hardver csatlakoztatása esetén új rendszert kell generálni. A hálózatba az Apricot F1, F2, F10, Fp, X<sub>1</sub>10, X<sub>2</sub>20 gépek kapcsolhatók be.

A hálózatot az MS-NET hálózatvezérlő szoftver — a PC-be dugaszolt csatlókártyán beépítve —, illetve az MS-DOS 3.1 operációs rendszer működteti. A hálózat-hoz archiválási céllal szalagegység is csatlakoztatható.

## PC-net hálózat

Az Orchid Technologies személyi számítógépes helyi hálózata max. 2,5 km hosszúságú, 75 ohmos koaxiális kábelen 880 kbps átviteli sebességgel, CSMA/CD protokollal működik, és max. 64 munkaállomást kapcsolhat rendszerbe. Kisebb eltérésekkel több néven is forgalmazták: AST's PC-net, Santa Clara System's PC-net, IDE Associates IDEA-net. A hálózati csatló viszonylag egyszerű felépítésű, kevés hálózati funkciót lát el, ezért a hálózatvezérlés jelentős feldolgozó teljesítményt köt le a PC-ben. Számos opcionális egységet ajánlanak a vevőknek: többfunkciós memóriabővítés, különböző portok, hálózati programbetöltő



# LÓZATOK

kártya stb., melyekkel a felhasználó munkáját könnyítik.

A hálózati szerver egyúttal munkaállomás is, más is. Munkaállomás minden olyan felhasználói PC, melynek perifériái (például a lemezegysége) elérhetetlenek a hálózat többi állomása számára, de lehet osztott hozzáférést biztosító PC is, melynek perifériái „közösíthetők”.

A rendszer segítségével megvalósítható a funkciók távoli végrehajtása, például egy programfordítás helyi kezdeményezéssel, de a hálózat egy másik gépén.

## A Plan 4000 rendszer

Az amerikai Nestar cég első helyi hálózata a Cluster One volt, amely Apple gépeket szervezett hálózattá. A Plan 4000 koaxiális kábelen működik, 500 m hosszon 254 állomást kapcsolhat rendszerbe, az átviteli sebesség 2,5 Mbps. A hálózathoz fájlszerver, IBM 3270 és 3278 típusú terminálemulációs program, telexkezelő, valamint két Plan hálózatot összekapcsoló — gateway funkciókat ellátó — szerver is kapható.

A fájlszerver Motorola 68000 mikroprocesszor alapú, lemezkapacitása 60 Mbájt. A rendszerhez 500 Mbájtos szalagegység (archiváló tár) is kapcsolható. A tárolt fájlokat jelszó védi. A hálózat munkaállomásai és az őket működtető operációs rendszerek az alábbiak lehetnek: Apple II és III gépek, operációs rendszerük DOS 3.1 vagy CP/M, valamint IBM PC-k, operációs rendszerük MS-DOS.

A Plan 4000 hálózat meglehetősen drága, de van egy olcsóbb változat is, a Plan 3000, amely csak egyetlen — 60 Mbájtos — lemezegységet tud kezelni. A Plan 2000 változat csatolóegysége hordozza az IBM PC XT alapú szervert működtető szoftvert. Mindhárom hálózat összekapcsolható egymással is a gateway szerver segítségével. Két összekapcsolt Plan 4000 hálózat között megvalósítható a virtuális lemezkezelés, azaz az egyik hálózat munkaállomása magához kapcsolhatja a másik hálózat lemezegységét, mintha az fizikailag is az ő gépéhez tartozna.

## Net Ware

Az amerikai Novell cég két helyi hálózatot forgalmaz: Net Ware/X-et és a Net Ware/S-et.

A Net Ware/X hálózat 1,43 Mbps sebességű; max. 2,5 km hosszúságú koaxiális kábelt használ, a hálózatalérési protokoll CSMA/CD rendszerű, a hálózatba kapcsolható max. állomásszám 256.

A Net Ware/S 500 kbps sebességgel, 500 m hosszúságú sodrott érpárral dolgozik, a

hálózatalérési protokoll egyedi megoldású, a hálózatba kapcsolható max. állomásszám 24.

A Novell hálózatvezérlő szoftvert is szállít különböző gyártók rendszereihez, többek között a Quadnet VI-hoz, a proNET-hez, a Personal Mini-hez, az Omninet-hez, az Ethernet-hez, a PC-net-hez, a Multilink-hez, a Plan 3000-hez, az SMC-net-hez, valamint saját rendszereihez. Az egyes rendszerek közötti egyedüli különbséget a csatolóegység fizikai szintű vezérlése jelenti. A fenti rendszerek majdnem mindegyikében IBM PC XT a hálózati szerverállomás, kivételt képez a saját rendszer, a Net Ware/S, amely Motorola 68000 alapú fájlszervert alkalmaz.

A Net Ware egyedi megoldású rendszer, amelyben különböző trükkökkel sikerült megnövelni a hálózat teljesítményét; például műveletgyorsító előtár (cache tár) beépítésével, illetve a lemezen való ún. elevátoros keresési módszer alkalmazásával, amikor az egyszerre jelentkező fájlhoz fordulásokat úgy sorolják, hogy azokat minimális fejmozgatással lehessen elvégezni.

A Net Ware rendszerhez 15 szerverállomás kapcsolható, amely lehet akár dedikált, akár nem dedikált. A Net Ware operációs rendszere valódi fájlszerver funkciót valósít meg: a fájlok megnyitásakor automatikusan kizárja más állomásoknak ugyanezen fájlhoz fordulását (tehát egyszerre két helyről nem nyitható meg ugyanaz a fájl), továbbá lehetőség van a munkaállomások közötti üzenetváltásra, valamint az osztott nyomtatásvezérlésre is.

A Net Ware előnyét éppen az adja, hogy a szabványos, és más IBM alapú rendszerekben alkalmazott PC-DOS fájlstruktúrától kismértékben eltérő fájlstruktúrával rendelkezik. Az adatok sértetlenségét és titkoságát igen fejlett adatvédelmi rendszer garantálja: egyrészt a DOS fájlstruktúrájánál több „zár” iktatható be, másrészt a hozzáférési jogosultságot négy szinten ellenőrzi. Az első szint a logon/password (jelszó), amely a rendszerbe való belépést engedélyezi; a második a trustee (meghatalmazási) szint, amely közli, hogy a felhasználó milyen művelettel fordulhat valamely fájlhoz (írás, olvasás, fájlleltetés, törlés stb.); a harmadik szint a directory right (könyvtári betekintési jog), amely vigyázza, hogy a felhasználó mely könyvtárak fájljait hívhatja le; végül a negyedik szint a file right (fájlműveleti jog), amely megadja, hogy a felhasználó mely fájlokat módosíthatja az adott könyvtárban. Ezeket a jogokat a hálózatalérési rendeli hozzá a fájlokhoz, illetve az egyes felhasználókhöz.

A Net Ware támogatja a dinamikus lemezterület-hozzárendelést, vagyis lemeztérület(ek) felhasználó(k)hoz rendelését oly

módon, hogy ezek a területek a fájlműveletek (például törlés) függvényében módosulnak.

A szerverhez max. három nyomtató — két soros és egy párhuzamos — kapcsolható, a nyomtatás a fájlkezelés mellett háttérműveletként hajtódik végre. A Net Ware nyomtatási rendszere az egyik legjobb; olyan opciói vannak, mint a többpéldányos nyomtatás, nyomtatás közbeni papírváltás, nyommon követhető a függőben lévő nyomtatások, átrendezhető a nyomtatás várakozási sora, a várakozó nyomtatások egyik nyomtatóról a másikra átirányíthatók. Mindezek a speciális nyomtatásvezérlő utasítások segítségével hajthatók végre. Meg kell említeni azonban, hogy a Net Ware — számos előnye mellett — nem képes néhány DOS parancs végrehajtására. Ilyenek például: PATH, CADIR, BACKUP.

## Nem „igazi” helyi hálózatok

A tapasztalat szerint meglehetősen gyakoriak a szerényebb felhasználói igények, különösen kisebb intézmények részéről. Az olcsóbb hálózatok rendszerint csillag topológiájúak, és egy központi gép köré települnek, amely időosztásos rendszerben működik, és szabványos, például V24 interfészen keresztül tart kapcsolatot 4-8 munkaállomással. Munkaállomásként egyszerű terminál is megfelel, ami jelentős árcsökkenést tesz lehetővé.

Az ilyen típusú rendszerek központi gépe akár egy IBM PC XT vagy AT is lehet, és például az Anex cég által készített hálózati interfészártyával a rendszer viszonylag olcsón négyfelhasználóssá tehető. Ezt a rendszert az Anex Multi DOS operációs rendszere működteti oly módon, hogy a műveletvégző teljesítményt megosztja a központi gép és a munkaállomások között.

Ezek a rendszerek alkalmasak olyan közismert szoftvertermékek futtatására is, mint a Multimate vagy a Lotus 1-2-3, és képesek lebonyolítani felhasználók elektronikus levelezését. Tapasztalatok szerint a 3-4 felhasználóval működő hálózat gépi válaszidője alig hosszabbak a hagyományos helyi hálózatok válaszidőinél. Nagyobb távolságok áthidalásához természetesen modemre és adatátvitel-vezérlő szoftverre is szükség van.

Ezeket a rendszereket elsősorban olyan alkalmazásokhoz ajánlják, ahol az adatállományt központilag tárolják, de több helyről, nem feltétlenül egyidejűleg kérdezik le, illetve aktualizálják. Az ilyen rendszerekben tehát nem alapkövetelmény, hogy minden állomás egyszerre férhessen hozzá a teljes adatállományhoz. Két ilyen megoldást mutatunk be.



## a) Multilink Advanced

A Davong cég által fejlesztett rendszer a Multilink Advanced hálózat, amely kilenc állomás számára biztosít többmunkahelyes (multi user) és többfelhasználós (multi tasking) környezetet. A hálózat gépeit MS-DOS operációs rendszer működteti; a rendszer nagy előnye, hogy működéséhez nincs szükség drága hálózati csatolóegységre. A Multilink rendszer központi gépe IBM PC AT. Ennek feldolgozó teljesítményén osztoznak a munkaállomások.

Megfelelő memóriabővítéssel nagyobb tárigényű programok futtatása is megoldható. Ez esetben egy-egy felhasználóra 448 kb-ot központi tárkapacitás jut. A rendszerhez mind hardcopy nyomtató, mind osztott hozzáférésű „központi” nyomtató kapcsolható (soros vagy párhuzamos interfészen keresztül), melyet a hálózat egyes állomásai bejelentkezésük sorrendjében használhatnak. A fájlkezelési rendszer osztott hozzáférést biztosít a központilag tárolt programokhoz, illetve adatállományokhoz. A fájlvédelem jelszavas.

A Multilink hálózaton olyan közismert programtermékek is futtathatók, mint a Word Star, a dBase III, a Multimate és a Lotus 1-2-3. A munkahelyi interfész V24, és olcsó terminálokkal a hálózat teljes ára nem éri el egyetlen AT gép árát. Az alacsony ár a rendszer egyik legnagyobb előnye.

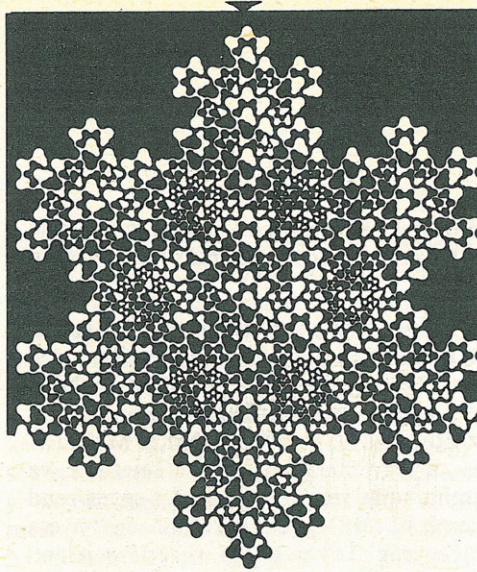
A hálózat munkaállomásai többek között DEC VT 52, VT 100, illetve IBM 3101 gépek lehetnek. A rendszerhez több cég szállít interfészegységet, például az IBM (Communication Adapter), a Software Link (Multiuser Communication Port Board) vagy a Quadron (Dual Assynchron Card). A terminálhálózat prioritásos rendszerűre is kiépíthető.

## b) A LAN Link hálózat

A Multilinkhez hasonló egyszerű hálózat a LAN Link, amely egy központi telepítésű IBM PC-ből (célszerűen AT) és a hozzá csillag elrendezésben, V24 interfészen keresztül kapcsolódó négy munkaállomásból áll. Az olcsón megvalósítható hálózat központi számítógépét, a szervert, valamint munkaállomásait PC-DOS 2.0, MS-DOS 2.0 verziószámú vagy valamely fejlettebb operációs rendszer működteti. A hálózat átviteli sebessége 112 kbps. A hálózat munkaállomásai adott esetben közönséges képernyős terminálok is lehetnek, de természetesen IBM PC-k is.

A hálózatba soros, illetve párhuzamos interfészű nyomtató kapcsolható, amely a hálózati munkaállomások számára biztosítja az osztott nyomtatást. A munkaállomások közvetlenül vagy modemén keresztül csatlakoztathatók a szerverre. A fájlkezelési rendszer jelszóval védett, és megengedi a fájlokhoz való egyidejű hozzáférést olvasásra, de az egyidejű írás ellen rekord szintű védelmet ad.

CSEH KÁLMÁN



A Mandelbrot halmaz egy rendkívül bonyolult alakzat, melyet a  $Z_0=0$ ,  $Z_{i+1}=Z_i^2+C$  rekurzió definiál. A Mandelbrot halmaz elemei azok a  $C$  komplex számok, melyekhez tartozó  $|Z_n|$  sorozatnak van véges felső korlátja. Bizonyítható, hogy ha egy  $k$ -adik iterációs lépésnél  $|Z_k| \geq 2$  értéket kapunk, ennek a  $|Z_n|$  sorozatnak nem lesz vé-

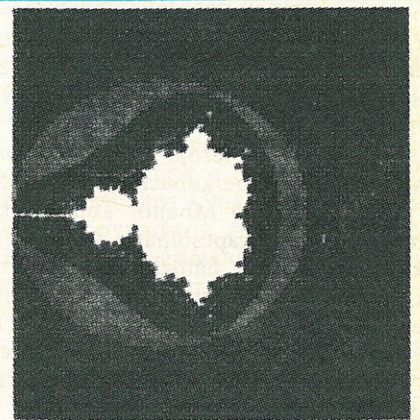
### 1. program

```

MANDELBROT 1
10:INPUT "XB=";X,"XI=";I,"YB=";Y,"Y
14:JU
20:INPUT "ITERACIOK (1-15)";I;N
30:IF N<156010 20
40:INPUT "COLOR MOD (0-2)";I;I;POKE
17410,N;POKE 17470,56+157*(N/10),
56+157*(N/1)
50:F=(X-X)/100;G=Y-F;H=1;J=(U-Y)/F
60:FOR I=0 TO J:CALL 17310:CALL 1740
>NEXT I
80:TEXT :CSIZE 1:COLOR B:LF 3:
LPRINT "REAL PART :";X;:LCURSOR
17:LPRINT " ";:J;0
90:LPRINT "IMAG.PART :";Y;:LCURSOR
17:LPRINT " ";:J;U
100:LPRINT "COLOR MOD :";I;N
110:LPRINT "ITERACIOK :";N
120:LF 4:END
    
```

ges felső korlátja, vagyis az ehhez tartozó  $C$  komplex szám nem eleme a Mandelbrot halmaznak.

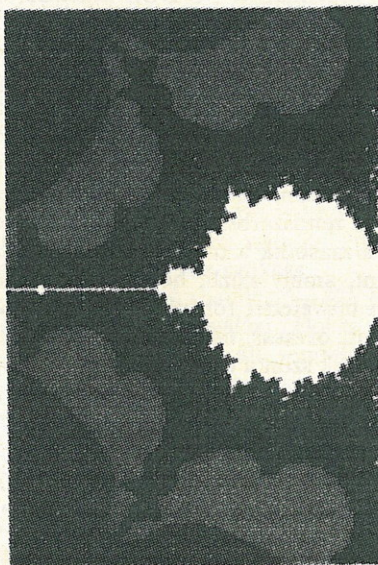
A PTA-4000 típusú gépre készített 1. programmal a komplex számsík egy téglalap alakú részét vizsgáljuk, hogy ennek pontjai eleme-e a Mandelbrot halmaznak, vagyis  $C$  értékei e tartomány pontjai lesznek. A program egy  $C$  számot akkor tekint a (közelítő!) Mandelbrot halmaz elemének, ha az általunk megadott maximális iterációs számig a rekurzív sorozatban egyetlen  $|Z_i| \geq 2$  értéket sem kap (az ábrázolásnál ezekhez a pontokhoz nem rendel színt). A tartomány pontos pontjaihoz aszerint a  $K$



```

REAL PART :-2 , 2
IMAG.PART :-2 , 2
COLOR MOD : 0
ITERACIOK : 30
    
```

### 2. program



```

REAL PART :-1.5 , -1.25
IMAG.PART :-0.18 , 0.18
COLOR MOD : 2
ITERACIOK : 80
    
```



```

REAL PART :-0.864 , 0.864
IMAG.PART :-0.95 , -0.76
COLOR MOD : 1
ITERACIOK : 60
    
```



# A ÉS TUDOMÁNY?

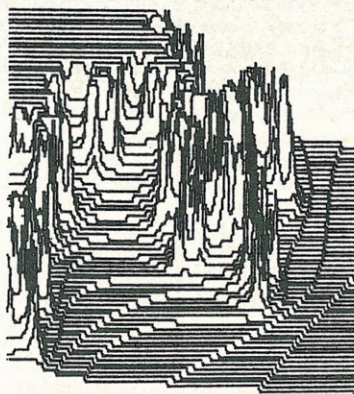
## almaz előállításakor keletkező „színes” ábrák

(Hátsó borítónkon)

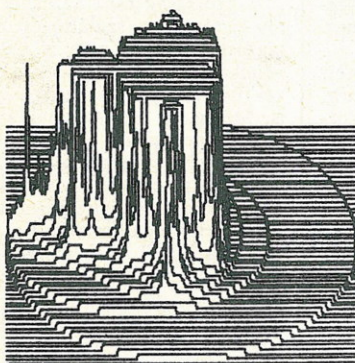
### MANDELBROT 2

```

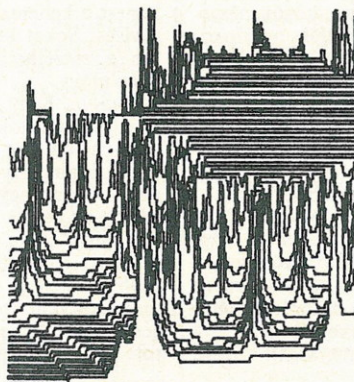
10: INPUT "X0="; X, "X1="; Q, "Y0="; Y
15: INPUT "N="; N; E: POKE 17418, E
20: GRAPH : F=(Q-X)/210: M=4: W=717256+
254: U=210*F+Y
25: Q=(X+Q)/2: U=(Y+U)/2: POKE 17476, 1
54
30: INPUT "PHI="; A, "THETA="; B: M=SIN
A: K=SIN B: D=COS B: N=COS A
40: GDCURSOR (100, -255): SOR6N
50: A=E*YD+100*(K*Y+M*X): S=-100*(K*Y+
M*X): U=255/(A-S)
60: FOR I=-100 TO 100 STEP 4: G=U+I*F
80: FOR J=-100 TO 100: T=(J+I*F)/N: Z=Q
+T*F
90: CALL 17340: Z=INT (D*PEEK W-J*F*
K+I*F*M*K-S)*U+.5: R=PEEK (W-109+J
)
100: IF J=-100 GDCURSOR (J, Z): E=(Z)*R
105: IF I=-100 GOTO 150
110: IF R)=Z AND E=0 GOTO 160
120: IF R)=Z LET E=B: Z=R: ON (X<=R)+1
GOTO 150, 160
130: IF E=1 GOTO 150
140: E=1: GDCURSOR (J, R-(R-Z)*X(X)=Z)
150: LINE -(J, Z): POKE W-109+J, Z: R=Z
160: X=R: NEXT J: NEXT I: POKE 17476, 165
: GDCURSOR (-100, -10): TEXT
    
```



X0, X1 = 0.0640, 0.0640  
Y0 = 0.9100  
N = 30



X0, X1 = -2.0000, 2.0000  
Y0 = -2.0052  
N = 30



X0, X1 = -1.5000, -1.2500  
Y0 = 0.1255  
N = 30

GÉPI KÓDÚ RÉSZ : 4310 h-4490 h

```

4316 > BE 43 50 BE EF BA BA 43
431E > 7E BE 43 50 BE EF BA BA
4320 > 43 7E BE 43 50 BE F0 1A
432E > BA 43 7E 50 79 40 79 D9
4330 > D9 D9 1A 84 D9 D9 D9 0A
433E > 0A 07 E5 00 83 9A 5A 00
4340 > 0E 02 5A 10 50 7A 40 79
434E > D9 D9 D9 0A 0A 07 F5 00
4350 > 03 9A AE 47 FF 14 D9 D9
435E > D9 1A 50 79 84 D9 D9 D9
4360 > 0A 40 79 FD 90 50 7A 5A
436E > 10 0A 07 F5 00 83 FD 0A
4370 > 5A 00 0A 07 F5 00 83 9A
437E > A5 47 FF 50 79 D9 D9 D9
4380 > 1A 40 7A 05 01 FD C3 BE
438E > 00 83 9A 05 01 FD C3 BE
4390 > 43 80 FD 0A 40 47 0A A5
439E > 47 FE 41 84 07 05 09 13
43A0 > 9A 05 06 5A 06 4A 05 BE
43A6 > 43 16 05 19 4A 17 BE 43
43AB > 31 BE 43 91 0A 05 19 5A
43B0 > 19 4A 05 BE 43 16 05 01
43B6 > 4A 06 BE 43 31 05 00 4A
43BC > 19 BE 43 31 E9 47 FE 00
43C0 > 05 0E 4A 00 5A 00 BE 43
43C6 > 20 05 0F 5A 01 4A 01 BE
43CC > 43 20 05 00 5A 0E 4A 0F
43D0 > BE 43 16 05 00 BE 43 44
43D6 > 05 07 BE 43 40 05 03 BE
43DC > D0 D2 00 42 EF 47 FE 01
43E0 > A5 47 FE 07 1E 09 01 9A
43E6 > 05 02 5A 0E 4A 0F BE 43
43EC > 1F 05 02 5A 02 4A 18 BE
43F0 > 43 16 05 00 5A 00 4A 00
43F6 > BE 43 16 05 01 5A 00 4A
4400 > 01 BE 43 20 05 01 5A 01
4406 > 4A 06 BE 43 16 05 00 4A
4410 > 02 BE 43 31 0E 0F A5 47
4416 > FE 30 30 00 83 AE 47 FE
4420 > 9A 4A 02 BE A2 C4 0A 00
4426 > FD 00 BE A9 E4 BE A5 19
4430 > BE 47 09 E9 47 FC 00 40
4436 > 47 4A 01 FD 2A 2A 07 09
4440 > 19 FD 00 FD A0 A5 47 FC
4446 > 0A BE A2 BE 05 02 BE AC
4450 > 2F BE A7 09 E9 47 FC 00
4456 > FD 2A FD 0A EF 47 FC 00
4460 > 40 4E 05 09 20 00 0E 04
4466 > 99 42 9A
    
```

Beírás : POKE-val

KAZETTARA RÖGZÍTÉS :  
CSAUEN "név": 17174, 17500

BETÖLTÉS KAZETTARÓL :  
CLOADN

— közelítő — Mandelbrot halmaz pontjaihoz az általunk megadott maximális iterációs számot rendeljük.) A szemléletesebb és esztétikusabb ábrázolás kedvéért lehetőség van a vizsgált tartomány középpontja körül phi szöggel az YX síkot, theta szöggel az XZ síkot elforgatni.

A bemenő adatok:

$X_0, X_1$  Az ábrázolandó tartomány valós részének kezdő- és végpontja;

$Y_0$  Az imaginárius rész kezdőpontja (a végpont mindig  $Y_1 = Y_0 + X_1 - X_0$  lesz, vagyis mindig egy négyzet alakú tartománnyal dolgozunk);

$N$  Az általunk megadott maximális iterációs szám;

phi XY sík forgatásának szöge (a példákban 0);

theta XZ sík forgatásának szöge (a példákban 0,18-0,22 radián).

A gépi kódú részt mindkét program használja. Bővítő nélküli gépeknél vigyázzunk, nehogy javítás, beszurás, ill. más ok miatt a BASIC program a gépi kódú program által lefoglalt memóriaterületet megváltoztassa. Ajánlatos ezért a gépi kódú rész néhány első bajtját ellenőrizni és esetleg javítani.

MÉSZÁROS CSABA

Olvasóink figyelmébe ajánljuk Peitgen H.O. — Richter P.H. The Beauty of Fractals Images of Complex Dynamical Systems (Springer 1986) c. könyvét, amely ebben a témában elméleti és rendkívül gazdag képanyagával kiemelkedő összefoglaló munkának számít.

index (ezt „kiesési index”-nek nevezzük) szerint rendel egy-egy szint, amelyiknél először kap  $|Z_k| \geq 2$  értéket. Természetesen minél nagyobb maximális iterációs számot adunk meg, annál pontosabb képet kapunk a Mandelbrot halmazról.

A bemenő adatok (baloldalt a változó neve, jobboldalt az értelmezése):

$X_0, X_1$  A vizsgált tartomány valós részének kezdő- és végpontja;

$Y_0, Y_1$  A vizsgált tartomány imaginárius részének kezdő- és végpontja;

ITERÁCIÓK Az általunk megadott maximális iterációs szám (maximális értéke 255 lehet), ameddig a vizsgálat folyik;

COLOR MOD Értéke 0, 1, 2 lehet. A Mandelbrot halmazhoz nem tartozó pontokhoz rendelt szín a következőképpen lesz meghatározva:

$$\text{INT} \left( \frac{K \text{ kiesési index}}{2^{\text{color mod}}} \right) \pmod{4}$$

ahol INT jelöli az egészrész függvényt.

0 választásnál más-más kiesési indexhez más-más szín jelenik meg, 1 választásnál minden második, 2 választásnál minden negyedik kiesési indexnél jelenik meg más szín, így elkerülhetjük az ábrák esetleges túlzott tarkaságát.

A program az ábra elkészítése után nyomtatja az általunk megadott adatokat is. Az ábra a legfinomabb felbontás négyeszeresével készül (X és Y irányban is kétszeres felbontás), a sebesség növelése érdekében.

A 2. program a komplex értelmezési tartományt az XY sík részeként, a kiesési indexet pedig ezekre merőleges térbeli Z — nem komplex! — tengelyen ábrázolja. (A



Motó: Az igazság mindig egyszerű.  
(Egy középiskolai tanártól)

# Barátunk, a számítógép

## Együtt dolgozunk

Alig két éve majdnem írtam egy cikket *Mikrogép és aerobic* címmel. Akkor azt hittem, hogy azonos szintű, múltó divatokról van szó. Tévedtem. Az aerobic ment, a mikro maradt. Ne feledjük azonban az aerobic szorgalmazójának, Jane Fondának az infarktuszát! Példájak mutatja, hogy egészséges dolgokkal is lehet visszaélni, mint például a mikrokkal! Azt is tartuk vigyázó szemünk előtt, hogy a fejlett számítástechnikájú országokban a mikro nem kíván az lenni, amilyen szerepet betölt nálunk a számítástechnika-éhség miatt. Ott a mikro nem pótolja, hanem kiegészíti a nagygépes lehetőségeket. Már számtalanszor előfordult velünk, magyarokkal, hogy a más által kitalált lehetőséggel sokáig nem élünk. Majd ezután mi megmutatunk, túlhajtva az eredeti célon. Ettől talán óvakodnunk kellene!

A mikro fogalma rendkívül eltérő kapacitást, teljesítményt és általános lehetőségű gépek széles skáláját takarja. Jól hasonlítható egymással egy régebbi típusú IBM nagygép és egy újabb. Ugyanakkor egy „zseb”-Sinclair képességeit aligha lehet összevetni egy IBM AT lehetőségeivel, nem beszélve a 32 bites, szerényebb IBM RT-ről. Ezért a további mondanivalót kényszerűen a közepes kategóriájú (IBM PC XT, AT és hasonló) mikroszámítógépekre kell korlátoznom. Megjegyzendő, hogy e kategória alatt (például Commodore 64) a lehetőségek drasztikusan csökkennek, míg a magasabb kategóriájú gépek hazai elterjedésére még várunk kell.

Ha a barátunk használatával kapcsolatos tapasztalataimat és észzéseimet összegezmem kellene, akkor mondanivalómat három pontba illene összefoglalnom. A korábbi számítógépes lehetőségekkel és azok hiányával szemben a mikro világát most jelent a végső felhasználónak és mást a fejlesztőnek, a számítástechnikai szakembernek. Ugyanakkor a mikro — mikro. Olyan általános jellemzőkkel is leírható — a nagygépekkel szemben — amelyek mind a felhasználó, mind a fejlesztő lehetőségeit meghatározzák. Logikusnak tűnik, hogy mondanivalómat ezen általános jellemzőkkel — korlátokra és lehetőségekre bontva — kezdjem. Ezt a rövid összefoglalót a végső felhasználók figyelmébe ajánlom. Annak ellenére, hogy itt bizony néhány technikai rész kérdésre is ki kell térnem. Kezdjük tehát az általános lényeggel!

### Gyermekbetegségek

Előttek egy IBM PC XT, 640 kb-át központi egységgel és 20 Mb-át háttérkapacitással. Jobbra egy IBM PC AT 3 Mb-át központi egységgel és 80 Mb-át háttérkapacitással. Mennyi ez?

Nem kevés! Korábban a jó öreg IBM 370-en dolgoztam, amely 750 kb-át központi egysége mellé hiába rendelkezett igen bő háttérkapacitással, abból én csak 52-100 Mb-ajnyit használhattam ténylegesen. (Csak a kevésbé tájékozottak kedvéért jegyzem meg, hogy egy

szabványos gépelt oldal (A4) 1800 karaktert (bájtot) jelent. Egy 500 oldalas könyv tehát bőven elfér 1 Mb-aj területen.) Közepes végső felhasználóként. Mit jelent a „közepes”? Nos, volt vagy száz programom és mintegy 150 ezer nyilvántartási egységem, rekordom.

Mindez azt jelenti, hogy a mikro adott kategóriája a hozzám hasonló „közepes” (magyarországi viszonylatban!) felhasználók számára akár meg is felelhet. Azonban azt a 20 Mb-aj háttértárat „pilanat alatt” megeszik a programok, és a 80 Mb-ajtból is csak fatnyi marad, ha elkezdjük az adatok tényleges betöltögetését.

Nem, kis barátunk határozottan nem való komoly, összetett vállalati rendszerek támogatására. Legfeljebb integrálatlan részrendszerek kezelését biztosítja számunkra, ha helyesen választottuk meg a gép egységeit, az ominózus konfigurációt.

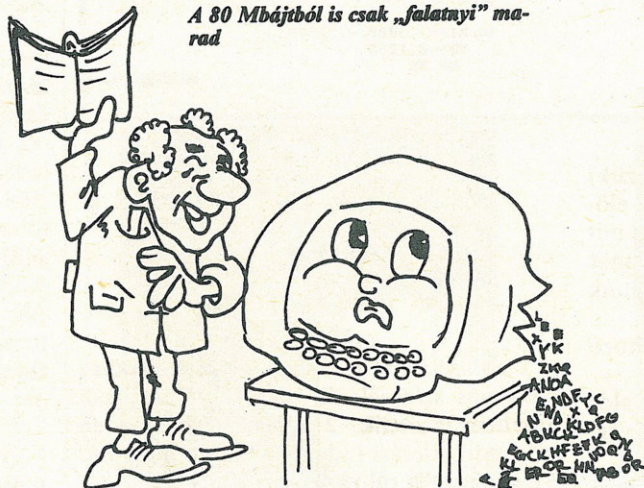
Kiváló kollégáim közül némeylek most ráncolhatják homlokukat: ők találtak megoldást. Nos, a találmány így hangzik: a rendszert részekre kell szabdalni. Ha az egyik ikszedével végeztünk, akkor kimentünk, betöltjük a második ikszedet és minden megy a maga sorjában. A szegény ember főztje... A 60-as években dolgoztunk így, lekapkodva-felrakosgatva a mágnesszalagokat. A helyzet végesen hasonló. Akkor ezt számunkra az operátor végezte, tiszteletreméltó számú kilométereket téve meg a konzol és a szalagegységek között.

teljesítménycsökkenésekkel is kell számolnunk.

Oda se neki! Magyarországon még egyetlen vállalat sem ment tönkre abban — másutt már százával —, hogy egy információ perceket késett. Valóban: nekünk rendkívüli élmény a feldolgozási lánc lerövidülése, az említett gyalog módszer kellemetlenségei dacára is online elérhető információ. Nem kell formanyomtatványra írnom az adatot. Talán még kódolhatnom sem kell. Nem lyukasztja félre információim x százalékát az adatrögzítő. A hibát azonnal látom és azonnal javítom. Nem kell gép-íróért könyörögnöm. Nem kell futás után a „kimenőben” keresgélnem az anyagot. Mindez remek!

Csak néhányan már láttak nagygépes online rendszert is! Nem érzékeltük, hogy adatunk mentődik-betöltődik-e vagy sem. Nem kapkodunk diszkettcserék után. Szóval a mikrogépes rózsza nem csak illatozik, hanem nemesítettekhez képest a kelleténél kicsivel jobban tövises. Sokat lendítene a helyzetben a cserélhető rögzített lemez. Még nincs.

Tudom, hogy ej, ráérünk. Ezért csak közbevetőleg jegyzem meg, hogy hiába van nekem szép központi tárom. Egy komolyabb nagygépes lemezegység gépárdjához képest az én kis rögzített lemezem egy csiga. Húsz adatnál, ötvennél, százánál? Kitérdekel mindez? Egy több ezres vagy több tízezes állományból való szelektív visszakeresésnél már kaparom a kezelőpultot: no, gyere már!



Ritkán, de zavarba jött; mit is kellene most fel és mit le. Most ezt magunk élvezhetjük. A kimentés és visszatöltés pedig nem egy pillanat. Ne adj Isten, hogy közben át-szervezésre is igény legyen!

Nyilvánvaló, hogy ez a módszer adott esetekben — összefüggő állományok és programok esetében — nem is használható. Az esetek másik részében jön a sakkozás: mit, mikor és hová. Mindezzel pedig nemcsak a kimentés-betöltés idejét kell eltűnünk, hanem egyéb

Mert itt van, ugyebár, az adatkezelés nem elhanyagolható kérdése. A háttértárak technikai adottságain kívül a sebességet nagyjából két dolog határozza meg: a tölem függő adattömeg és az adatkezelő. No, itt mutatkoznak a gyermekbetegségek!

Az adott kategóriába tartozó gépeken rendelkezésünkre állnak a — nagyon régen — magas szintűnek nevezett nyelvek (COBOL, FORTRAN, PL/1 stb.). Ezek azonban csak igen korlátozott álló-

mánykezelési képességekkel bírnak. Ha ravasz megoldásokkal magunk láncolunk, indexelünk, mutatunk, akkor elérjük a kívánt gyorsaságot. Az első változásnál beleőrülünk a módosításba. Ha pedig valóban magas szintű adatkezelőt alkalmazunk (dBase III, dACCESS, rBase stb.), akkor el kell tűrnünk minden általánosított rendszer hátrányát: a viszonylagos lassúságot. Mi tagadás, egyelőre még a mikro nem olyan jól ellátottak hatékony adatkezelési lehetőségekkel, mint nagyobb rokonaik.

A relatív lassúsággal meg lehet alkuodni, és ezt könnyen megteszi a magyar felhasználó. A végső felhasználó, aki nem igényel valóban percre pontos információt vagy — eddigi tapasztalat hiányában — nem tudja, hogy ilyet igényelhetne. A fejlesztő, aki eddig a lyukasztóban üldögélt órákat és most percek alatt ütheti be a programját. Melikük fog a gyorsaságra panaszkodni?

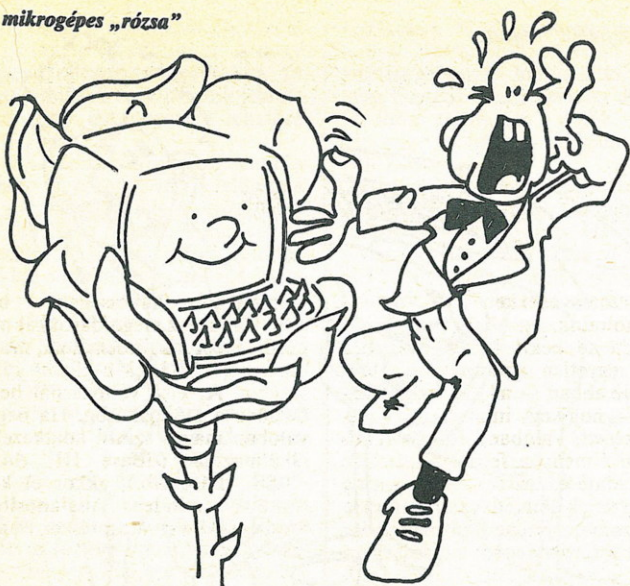
A gyorsaság csak az egyik fontos kritérium a számítógép alkalmazásánál, talán nem is a legkiemelkedőbb. Voltaképpen az idővel függ össze a biztonság kérdése is. Mégpedig hűsbavágóan komolyan. Éppen csak a minap „állt feje” egy nem általunk üzemeltetett rendszer, amelynek pontos időben igen fontos pénzügyi információkat kellett volna biztosítania. Nem volt „back-up”, azaz háttér; azért az adatok tíz százaléka megmaradt.

Mi szükséges a valódi biztonság-hoz? Két dolog. Egyrészt a háttér. Programok esetében ez nem ügy. Készítünk róluk másolatot és szükség esetén visszatöltjük azokat. A program nem változik, ha nem szándékosan módosítjuk. Más az adat, amelynek tartalma naponta módosulhat, hiszen éppen ez az online rendszerek értelme. Mit naponta? Óránként, sőt percenként! Ha már itt van egy közvetlenül elérhető számítógép, akkor feltétlenül cél, hogy mielőbb bevigyük azt az ismeretet, amit megszerezünk.

Adott mértékű változások, adatbevitel, törlések után — hogy hol a mérték, azt a rendszer jellege mutatja a tervezőnek — másolatokat készítünk állományainkról. Ez idő, és egyéb feldolgozásoktól von el teljesítményt. A mikrón érzékelhető módon. Nagygépen mindezt nem is észleljük. Ha nem feledkezünk el erről a követelményről, akkor félig rendben is vagyunk.

Félig. Mert pillanatonként, minden feladatunk után nem készíthetünk háttérmásolatot. Végeztünk, mondjuk harminc módosítást — ötvennél akartunk másolatot készíteni — és puff, észlált a rendszer. Mi a teendő? Akár végső felhasználó, akár fejlesztő vagyok, be fogom tölteni a korábbi, nem sérült változatot. Majd eszméletlen nyomozásba fogok, hogy a korábbi változat óta miket is csináltam?





Most mindazt újra elvégezhetem, ha még emlékszem a dolgokra. Vezessek talán róluk kézi naplót?

Nos, tegyük fel, hogy úgy döntök: gépi naplót vezetek. Rögzítsek csak a gép, hogy mikor és mit végeztem. Majd egyszerre csak azt észlelem, hogy így az átlagos válaszidő megkétszereződött. Nem írhatom kis barátom rovására ezt: kétszer jegyezte fel mondanivalómat. És mindentől nem leszek boldogabb, mert elő kell keresnem, amit addig végeztem, és újra végre kell hajtanom. Egy nagyszámítógépes rendszer esetén ilyen bajaim nincsenek. Csak elmaradt környezetben nincs automatikus, a feldolgozási időt gyakorlatilag nem befolyásoló naplózás. Csak közepesen elmaradott környezetekben nincs automatikus újraindítás és a korábbi tranzakciók automatikus újra végrehajtása. Szóval: feladatától függ, hogy szabad-e mikróban gondolkodnom, ha megfelelő biztonsági szintet és reális válaszidőt együtt kívánok tartani.

És végül itt van egy lényeges momentum. Te ülsz a gép mellé vagy én? Tudomásul illene venni, hogy a személyi számítógépek nem véletlenül viselik ezt a nevet! Jelenleg két alternatíva mutatkozik számunkra, a fenti kérdés — ki ül a gép mellé — megvitatásán kívül.

Az egyik lehetőség: egyikünk se. Végső felhasználók vagyunk, és van, aki elvégzi a technikai feladatot. Kezeli a gépet. Ekkor azonban következnek a formanyomtatvány, a kódolás, a hibalista, a forduló, a gépidőbeosztás problémája. Kis gépen imitáljuk a nagy gondjait. Nem egy kiváló megoldás.

A másik lehetőség: használunk valamilyen osztott feldolgozási módszert. Ilyesmivel az operációs rendszerek szintjén igen korlátozottan megoldható. Ha azonban egyidejűleg ugyanazokat az adatokat akarjuk kezelni, akkor venniük kell egy erre a célra alkalmas adatkezelőt. És akkor látjuk, hogy ilyen nincs. Ha, korlátozott felhasználót kiszolgálva, mégis találunk valamiféle szoftverszerkeztűt, akkor egyszerű képletbe botlunk: já, kettőn vagyunk felhasználók; és mindegyikünk feldolgozási ideje a háromszorosára(!) nő!

Mi tagadás, kis barátunk még nincs ellátva az állományok osztott használatát biztosító képessé-

gekkel. Óva intenek bárkit attól, hogy egy ilyen „többfonalú” rendszert maga fejlesszen ki, hacsak nincsen pár tízmillió dollárja, vagy nem hagyja hidegen az esetleg be nem fejezett és vissza nem fejtett tranzakciók sorsa. Vagyis ez az osztott használat szorosan összefügg a korábban feszegetett idő és biztonság kérdésével. Imitáljunk tehát egy régi vágású magyar nagyszámítógépes környezetet, sorban álló felhasználókkal, egy kis gépecskén?

A gondok nem elhanyagolhatóak. És még nem is szóltunk a valóban „távfeldolgozási” feladatokról. Ilyesmik már megvalósíthatók mikrókon is, ha valaki az idő és a biztonság említett korlátait eltűri.

A BMW vállalatnál egy tranzakcióbevitelünk — az adatok megfogalmazásától a végeredmény elkönyveléséig — mintegy 20 másodpercet vett igénybe. Ezalatt a számítógép több mint ezer(!) egyéb feladatot hajtott végre. Miközben teljes biztonságban éreztük magunkat, és adatainkat mások is kezelték. Mire megyünk egymással, kis barátom? Hiszen teljesen világos, hogy ha még osztott használatú, kényelmesen biztonságos, türelmemet próbára nem tevő sebességű feladatok elvégzésére is képtelen vagy! Miért és miként lehetsz mégis barát?

Ne haragudj, de most le kell szögezmem, hogy semmi olyasmit nem tudsz, amit egy komoly nagygépes online rendszer ne tudna! Lásd be, hogy nekünk kevesebb az anyagi forrásunk, és ezért melegen bízunk a te barátságodban. Lásunk csak, hogy szerény áron mennyi újjal örvendeztetél meg bennünket.

## Most végsőfelhasználók

Tegyük fel, hogy most én vagyok a legvégsőbb felhasználó vállalatunkban! Ide került kis barátunk, a mikrogép, és én éppen lelkesedni fogok érte, vagy pedig legyintek. E két, szélsőséges magatartás között is ingadozhatok. Lásunk példákat!

### 1. példa

Jártak nálunk — ismered őket —, amolyan szervezők vagy mik. Szóval programozók, tudod. Ké-

szítettek egy valamit, amit készletnyilvántartási rendszernek neveznek. Ott van azon a vacakon, azon a dobozon, azaz a számítógépen. A főnök behívott és a kezembe nyomott egy olvashatatlansokszorosítottmányt. Tanulmányoztam, majd mindent megérték. Utána a szakemberek elő is fognak adni nekem arról a rendszerről. És január 1-jétől — ezek a dinnye szervezők miért mindig pont ezt a lehetetlen napot jelölik ki átadásra — ezzel a géppel fogok dolgozni, állítólag.

Hát, tudod, én elolvastam azt a valamit. Kódoljak így és használjak olyan betűket. Nehogy véletlenül az x után y-t csináljak, mert az baj, azt a gép nem tűri. Legjobb, ha előre papíron leírom az eseményeket, a kézikönyv előírásainak megfelelően, majd beviszem. A gépbe. A piros hármast gombot kell lenyomnom, ha a zöldségeket, a sárga ötöst, ha a karalábét akarom megismerni. Bal kisujjammal tartsam lenyomva az Alt és Ctrl — érted Te ezt? — billentyűket egyidejűleg, miközben jobb mutatóujjammal Z-t ütök és orrommal lazán érintem a felső állású F-et. Majd RETURN — tudod, ilyen a balatoni vonatjegy is — nyomandó, ha... És PgÜp, ha felfelé akarok „péjdzselni”. (Te, szerinted az mi?)

Hát tudod mit fogok én nyomni, ha a főnök tényleg bevezeti ezt a rendszert? Az ágyat, sokáig. Remélem, majd csak elviszi innen az ördög el a frászkarikát. Addig is ölél szerető barátod. (Ui.: Hallottam, utálad a Kovácsot. Ajánld neki ezt a rendszert!)

### 2. példa

Cicuskám! Ez itt egészen frankó! Az a mókás a SZÁM...-ból egy fél délelőtt itt magyarul. Bekapcsolt egy olyan kis masinát, amit a tévében is mutattak. Pár vacak billentyűről — talán ötről — elmondta, hogy mit csinálnak. Kicsit túlmagyarázta a dolgot. Betegnek néz ez bennünket? Ne tudnék megjegyezni francos öt billentyűt?

Tudod, egy készletrendszerről van szó. Ez a kis vacak mindent megcsinál. Nem kell papírra írógatni, csak egyszerűen beütini olyan írógépen. Mellettem van egy csinos doksi — ezt Te, drágám, nem érted, a dokumentáció, amely röviden mondja el nekünk, hogy mit és hogyan csinálunk —, amit bármikor fellapozhatok. Tudom,

hogy abba a fényes dobozba csak A-t, B-t és C-t üthetek — írja a doksi. Ha elnézem, a gép szelíden és okosan figyelmeztet. Egy pillanats a doksiba, egy a képernyőre, és minden megy.

Kár, hogy kicsit szűkszavú. Azt mondja, hogy rossz adat. De melyik és miért? No, látod, Parányom, ezért van a doksi. Fél perc alatt megtudhatom, hogy mit toltam el.

Szóval így, drágám. Persze azért Rád is gondolok. Ha visszajössz, Neked is megmutatom, olyan izgi. A hétvégét, ugye együtt töltjük? Addig is csókollak.

### 3. példa

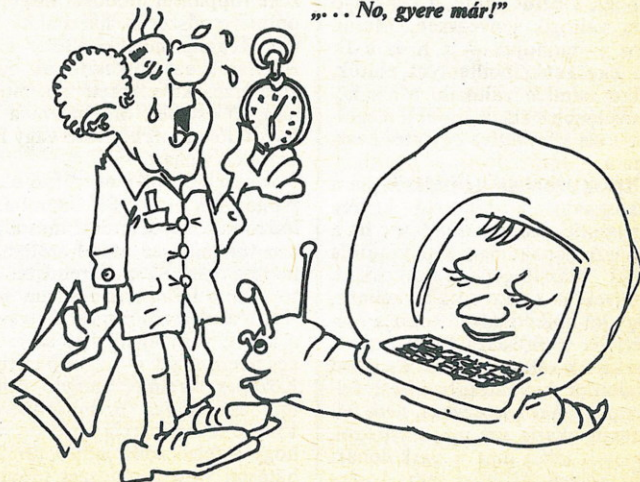
Kedves Bátyám! Ismered a Szabót, aki nálunk számítógépekkel foglalkozik. Képzeld, azt állította, hogy készített egy készletnyilvántartási rendszert. Egészen egyszerű — állította. Így kapcsold be a gépet, és utána csak annyit mondj neki, hogy KÉSZLET. Onnan minden magától megy. Nem sokat beszélt róla, olyan szűkszavú. Lehet használni, ha akarjuk — így ő.

Nem akartam ezt elhinni. Mikor a többiek elmentek, bementem abba a szobába, ahol az a kis gép van. Úgy tettem, ahogy Szabó mondta: bekapcsoltam a gépet. Minden simán ment. Már láttam ilyen gépeket, tudod. Azt is, hogy egy kis angoltól nem esem kétségbe. Itt azonban nem azt mondta nekem a gép, hogy „enter date”, hanem azt, hogy jelezzem a dátumot. És elfogadta a magyar dátumot. Majd nem a szokásos — másutt tapasztalt és érthetetlen — speciális jel villant fel, hanem ez a kérdés: Melyik rendszert kívánja használni?

Gondoltam, megpróbálom. Hangsúlyoznom kell, hogy semmit sem tudtam az egésztől. Mondd neki: KÉSZLET — így említette Szabó. Hát beírtam: KÉSZLET.

Tömör és udvarias módon kérdezte tőlem a gép: miről kívánok tájékozódni? Akarok-e azonnal programot futtatni? Nem, azt nem, nem is tudom, miről van szó. Akarom-e megismerni a rendszer tartalmi lényegét, vagy inkább technikai használatának módjával kívánok ismerkedni? Nem árt egyik sem, másik sem — gondoltam, és az egyiket választottam. A gép mindent elmagyarázott a rendszerről és megkérdézhettem, hogy mit is ért „átlagos ár”-on. Megmondta. Majd kedvem támadt átváltani a

... No, gyere már!?”





technikai kérdésekre. Erre a gép elmagyarázta, hogy így adok be adatot, úgy kapok eredményt, ha hiba van, mit kell csinálnom. Ha valamit nem értettem, mindig lenyomhattam a „H” vagy az „F1” billentyűt — ezt sugallta a gép —, hogy magyarázatot kapjak.

Nem akartam mások adatait bántani, ezért először csak lekérdőzöprogram végrehajtására merészkedtem. A gép kérdezte: akarok-e programot végrehajtani. Mondtam, hogy igen. Jelezte, hogy ehhez a program kódjára-nevére lenne szükség. Ismerem? Mondtam, hogy nem. Erre kiírta, hogy mire képes.

Innen minden egyszerűen ment. A havi készlet átlagértékéről érdeklődtem, és a gép elmondta, hogy milyen adat alapján képes válaszolni. Kódokat is alkalmazott, de alul mindig mutatta, hogy melyik esetben mit kell írnom. Ha eltoltam, ismét jelezte: abba a dobozba azt kellett volna betűtnöm, hogy ... Majd közölte az eredményt. A változtatást nem engedte meg: sajnos, nem ismer, ilyesmire én nem vagyok jogosult.

Majd leszek. Még egy-két órát játszottam ott, de jelentkeztek a főnöknél, hogy ez a rendszer nekem kell. Kell, érted? Ajánlom a vállalatod figyelmébe is. Addig is szeretettel öllelek.

### Váltsuk ismét komolyabbra a szót!

Essen szó most a fejlesztők felelősségéről! Valóban olyanoknak kell lenniük a számítógépes rendszereknek, amilyenek?

A magyar fejlesztő könnyen elhithető az addig teljesen tapasztalatlan felhasználóval és a régi gépek környezetben felnőtten egyaránt, hogy bizonyos megoldásoknak úgy kell lenniük. A fele sem igaz. ... Ha valóban komoly gondról van szó, akkor pedig a fejlesztő azért felelős, mert nem közölte a végző felhasználóval: ez a rendszer pedig NEM mikrogépre való! Nézzük csak meg, hogy mitől jó vagy rossz egy mikrogépes rendszer?

A rossz fejlesztő összetéveszti a végző felhasználót magával. Egyik rendszeremnél szememre vetették, hogy a következő lapot miért K betűvel, a dokumentációt miért D betűvel lehet kérni a „megszokott” funkcióbillentyűk helyett. Megszokott? Kinek? Ha az én rendszeremben a betűt elütök, a képernyő nem változik lényegesen. Marad rajta — mondjuk — a K és a D. Ha egy funkcióbillentyűt elütök, akkor minden változik. Mi a kényelmesebb? Talán legjobb megoldás a két alternatíva együttes használata.

Rossz azonban a rendszer, ha a felhasználót a fejlesztő képére kényszeríti. Rossz a rendszer, ha a felhasználónak meg kell tanulnia olyan számítógépes funkciókat, amelyek a tudta nélkül kellene, hogy jól működjenek. Rossz a rendszer, ha a felhasználóra bizza a diszkettes váltogatását és összetett módon a kimentési-betöltési feladatokat. Aki autót vezet, nem feltétlenül akarja azt maga zsirozni, pucolni, talán még a parkolóházi szűk helyen leállítani sem. A gépkezelés olyan egyszerű, mint az

autóvezetés. Ettől még nem szabad másodlagos feladatokra is kényeszerítőnek lennie.

Valljuk be, hogy kevésbé tudunk online rendszereket tervezni. Fantáziánk csak az ikszedik — másnak! — tervezett rendszer után kezd kibontakozni. Nem szégyellendő tapasztalatlanságunk első sorban a felhasználó-gép kommunikáció tervezésében mutatkozik meg. (Itt nem szabad felednünk a korlátokat sem, sokszor 132-es, 160-as nyomtató helyett 80 pozícióra kell szorítkoznunk.)

Kellemetlen megoldás, ha a végző felhasználónak újra csak formanyomtatványokat kell előre kitöltenie — szerencsére ilyen torz rendszer egyre kevesebb látható. Ezzel szemben igen kényelmes, ha a felhasználó a képernyőn megszokott formátumú, elrendezésű bizonylatot látja viszont — akár a „természetesnek” megfelelő színében. Egy jól tervezett hagyományos bizonylat is tartalmazza — hátul, esetleg alul — a kitöltési utasítást. Egyre több bizonylatot látunk, amelyen a megfelelő rovat mellett — ha az kódolt — felsorolják a lehetséges értékeket is.

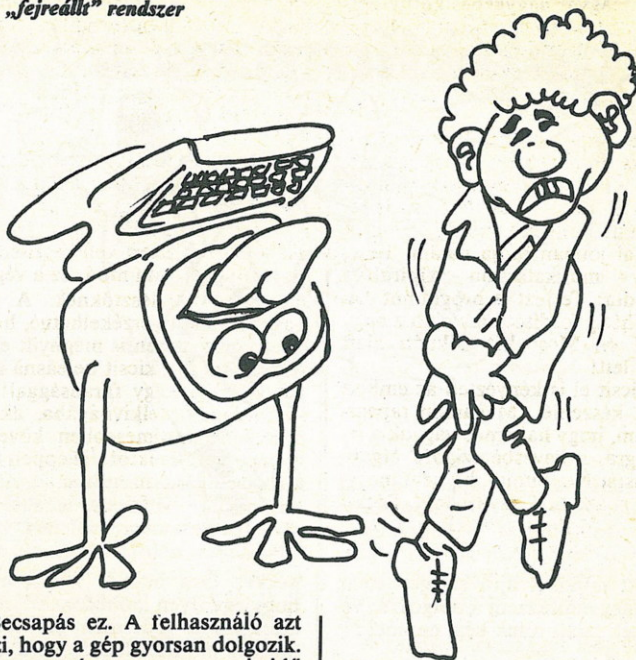
A többnyire elérhető 24×80-as képernyő persze kicsi. Távol esik a bizonylatzabványoktól, sajnos, még őrizve a lyukkártyák emlékét. Ha lassít is rendszerünkön, célszerű vagy lapozási lehetőséget adni a felhasználónak a kitöltési utasításhoz, vagy több képernyőből összehozni egy „bizonylatot”, rajta a szükséges tudnivalóval. Ne kelljen a felhasználónak emlékeznie — az veszélyes és a hibák miatt időrabló —, és ne kelljen fél kézzel doksit forgatnia. Általában, különösen az azonosítók tekintetében, online rendszernél kicsit jobban takarékoskodhatunk a kódok alkalmazásával. Egyéb jellegű, osztályozó kódoknál ennek az igénynek elmentmondhat a szűk méret, és éppen hogy sarkallhat a kódok alkalmazására.

A felhasználó megszokta, hogy számára jelentős számú esetben az a bizonyos bemenet és kimenet azonos. Magyarul: egy papíron dolgozik. Kedvező tehát, ha a nem számítási, összegzési feladatoknál, az adatbevitelnél és változtatásoknál a kimeneti kép a bemenetivel azonos. A felhasználó szeme önkéntelenül is a „szokott helyen” keresi az adatot. Adjuk meg neki ezt a minimális lehetőséget!

A jó tervezőnek arra is gondolnia kell, hogy a felhasználó időérzéke roppantul módosul kisgép — online rendszer — használata esetén. Tegnap még kitöltötte a bizonylatot, azt lyukasztották, hibaellenőrzés következett stb. stb. X nap, jó esetben óra múlva a felhasználó kapott kedvező vagy kedvezőtlen választ.

A gép előtt ülve ez az idő összezugorodik. Az ember bámul a villogzó képernyőbe, és x másodpercen túl rágia az asztal szélét: no, mi lesz már? Az okos rendszer előre közli a felhasználóval, ha hosszabb a válaszidő: most felteheted a kávé, vagy ejtsd meg az esedékes telefonokat — inthet barátunk. Közepes türelmet igénylő válaszidő esetén (ez 5 másodperctől 15–20 másodpercig terjedhet úgy, hogy a felső zóna már a türelmet határon van) a rendszer időnként közölheti, hogy most ott tart...

### A „fejreállt” rendszer



Becsapás ez. A felhasználó azt hiszi, hogy a gép gyorsan dolgozik. Egy nagygépen ugyanennyi idő alatt tízszer annyit végzett volna. Mi tagadás, még járnunk kell az online rendszerek pszichológiai iskolájába.

Minderről még sokat lehetne mondani. Egyetlen lényeges momentumba foglalható össze a tervezés alapproblémája. RÉGEN élesen szétvált a számítógépes feldolgozás és a végző felhasználó megszokott tevékenysége. RÉGEN a számítógépek közutálatának tárgya volt az úgy lenézett ügyvitel. Hogyan és mit csinál a végző felhasználó? Kit érdekel! Adja csak az adatokat úgy, ahogy „a gép megköveteli!” MA ez a képlet megfordult. MA illenek úgy megtervezni azt a rendszert, hogy az tökéletesen beilleszkedjen a végző felhasználó rutinos munkavégzésébe.

RÉGEN a számítógép felhasználója akkor volt boldog, ha némi nehézségek árán megkapta az igényelt kimutatásokat, elemzéseket, adatsorokat. MA, kis barátunk mellett akkor fogja jól érezni magát, ha egyszerű kezelési módot nyújtunk számára — a gépen valamit előkeresni talán könnyebb is, mint a megfelelő dossziét megtalálni —, és életét a minimális mértékben változtatjuk. Ez van.

Meg még egy fontos dolog: fantázia az alkalmazási területek fellelésében, amik mikrókkal támogathatók. Amelyek „gépesítésére” akkoriban nem is gondoltunk, mert ugye a forduló, a kényelmetlenség-

gek... E téren nincs is gond a magyar fejlesztőkkel. Inkább csak a buzdító, értő odafigyeléssel: már ne csak én sugalljam, drága végző felhasználó, hogy mit is lehet. Persze, vállalati érdekeltségre is szükség lenne ahhoz, hogy annyira fantáziadús legyen a jól jutalmazott tervező, mint otthon.

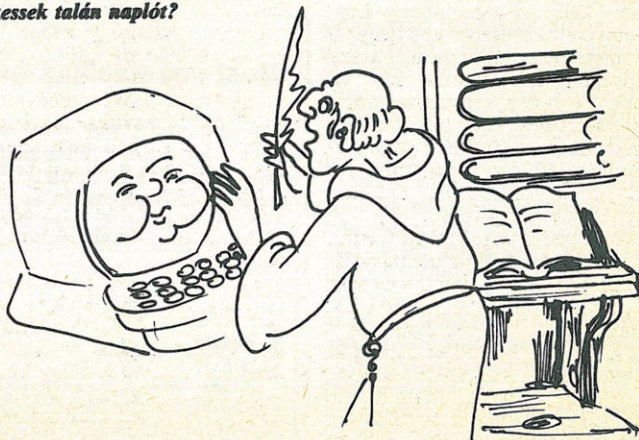
Valahogy így. Felhívásnál beadom: SZULINAP. A gép megkérdezi: legközelebbi vagy a múlt? A múltat választom. Kié? — kérdezi a gép. Mondom: NEJ. Erre kiírja, hogy három éve, tavalyelőtt és tavaly nejem egy csomag női cigarettát és egy blúz kapott szülinapjára. Holnap valami mást kell vennem...

### Most fejleszték

Marika? Nem emlékszel, hogy milyen kódok vihetők abba a mezőbe? Jé, ez a programrészt ugyanaz, mint amott. Másoljuk át! Legálább egyforma lesz. Mikor is változtattam utoljára ezt a programot? Két hete?

Kis barátunk hallatlan előnye a rugalmas fejlesztési lehetőség. Ha kellőképpen felkészítjük, akkor már a feladat elején rögzíthetjük adott ismereteinket, fokozatosan bővítve és módosítva azokat igény szerint. Felállíthatjuk saját „mintaállományainkat”, és a program azonnal tesztelhető. Hány „fordulót” kellett várni erre a hagyomá-

### Vezessék talán naplót?





## Rendellenességek és rendetlenkedések

Az 1986. szeptemberi számban a 8. oldalon leírt lemezoldaj-jelölés „ügyes” megoldás, és a legjobb módszer arra, hogy a lemez állományait tönkretegyük. A lemezen levő ID ugyanis nem a felhasználók tájékoztatása céljából van a lemezen, hanem azért, hogy a lemezegység operációs rendszerre értesülhessen a lemezcsere-ről. De ha két különböző lemezen, vagy ugyanannak a lemeznek a két oldalán ugyanazt az azonosítót találja, nem veszi észre a cserét, és a korábban használt lemez foglaltsági térképe (BAM-ja) alapján ír a mostani lemezre is, esetleg felülírva programjainkat, de mindenképpen felülírva az új lemezoldaj BAM-ját. Ne sajnáljuk tehát a fáradságot, és jelöljük a különböző lemezoldalak különböző ID-vel!

Tetszett a 29. oldalon olvasható „Egy új BASIC utasítás”, az INPUT+. A középső hasábon található assembly listába azonban hiba csúszott. A 270-es program sorban a műveleti kód nem STA, hanem STY. A 2. listán levő BASIC betöltőprogramban már a hibátlan kód szerepel.

Értetlenül olvastam a 29. oldalon kezdődő „GET# helyett INPUT#” című cikket. A problémának egy sokkal egyszerűbb és jobb megoldása található „A VC-1541-es lemezegység programozása” című könyv 251. oldalán kezdődő fejezetében. Az INPUT\* rutinnal szinte korlátozás nélkül, maximálisan 254 karakter hosszúságú rekordok olvashatók be a lemezről. A beolvasott rekord tartalmazhat vesszőt, kettőspontot, pontosvesszőt, sőt 13-as ASCII értékű karaktereket is. Egyedül a CHR\$(0) karakterek olvasása okozott — a feladat szempontjából áthidalható — problémát.

Ki ad magyarázatot a 38. oldalon olvasható „Ki ad magyarázatot?” című írással kapcsolatban arra a problémámra, hogy 1400 ill. 1900 hogyan lehet 300-1000 közötti szám? Lehet, hogy csak egy jel hiányzik valahonnan? Talán 10 000 volt a felső határ? (Igen, sajtóhiba volt; a helyes intervallum 300-10 000. — A szerk.)

BARNA LÁSZLÓ

nyos magyar nagygépes környezetben? (Itt se tévedjünk el! A nagygépes fejlesztőrendszerre képest a mikrók csak nekünk jelentenek üdítő és nagyszerű lehetőséget!)

Nem akarok minderről sokat lamentálni, mivel a végső felhasználó — elsőre! — úgy érezheti, hogy ez neki nem fontos. A kicsit is tapasztalt fejlesztő pedig mindezt sokkal jobban tudja nálam. Tény, hogy munkatársam kipirultan mondta: Te, ezt a programot három hétig fordítottam volna a nagy IBM-en. Most két délután alatt kész lett!

Kicsit el is kényeztetni az embert ez a készlet. Magamon tapasztaltam, hogy hajlamos vagyok a lazaságra, a kevésbé szigorú átgondolásra. No, bumm, ha nem megy, akkor azonnal javíthatok. Régen ezért a fordulósámokban győztesek székelyablájára kerültem volna egy komoly cégnél.

Ne tévesse meg sem a fejlesztőt, sem a végső felhasználót ez a kétségtelenül rugalmas fejlesztési lehetőség! Nehéz vele nem visszaélni. A gép villogva figyelmeztet, hogy válaszra vár: hát mondd nekem valamit. Régen, a ceruzát rágya kicsit jobban megfontoltam a választ. Mindehhez sok köze van a végső felhasználónak is. Talán nem a várt gyorsasággal készül el a rendszer és talán kicsit tetűlassú lesz. Mert minden alkalmazásfejlesztési készség, prototípus-alkotási lehetőség ellenére kis barátunk nem kényszerít a JÓ tervezésre, az OPTIMÁLIS terv kialakítására. Sőt! Másokat úgy sem zavarunk, majd csak elkettyeg a masina alapon lazábbnak érezzük a hatékonysági követelményeket.

### Kétpályás játék

A régi, nagygépes környezethez hasonlóan kezdtem fejleszteni új rendszeremet. Nem, én nem vagyok programozó, „csak” rendszerszervező. Szóval diktáltam — jól-rosszul — a feladatot a programozónak, aki kellemetlenül visszakerdezett ostobaságaimra. Majd egy kis irkálás a papíron, és kegyetette a gépet.

Ha már van, megpróbálom — gondoltam. Tanultam valamikor programozni — 1971-ben. Vagyis nem tudok. Itt valóban egy magas szintű nyelvel dolgozunk, amely roppant hasonlít az angolra. Nem kell megjedni, negyven-ötven szóval megél az ember. És utána minden mennyire más!

Tudom, hogy nevetséges, de nekem kell kiszámolnom, hogy mennyivel kevesebb ebédjegy és kilométerpénz jár nekem, ha szabadságon, kiküldetésen stb. vagyok. Elvégre a kétszázézer adminisztrátort nem ezért tartja vállaltunk! Ez van. De nem fogok én mindig plajbászt fogni, mert minek. Nekiveselkedtem, és írtam egy tíz soros programot.

Most pedig angolra kell fordítanom a rendszer dokumentációját. Eddig Zsó gépelte, miután én szintén gépeltem. Ő adott neki külsőcsint. Amíg volt türelme. Mert most itt ezt a bekezdést cseréld ki, amott hagyd ki azt a részt, itt pedig szúrjuk be — változtattam a véleményemet. Valamint az oldalszámot, a tartalomjegyzéket, a hivat-

kozásokat. Most, most megtanultam a „funkcióbillentyűket” és a szövegszerkesztő használatát. Bár nem használ a szememnek, direkt fordítom az anyagot a gépbe. A végén majd — talán — szépen legépelem Zsóka.

Nem folytatom. Csak annyit akarok mondani, hogy valóban barátunk ez a számítógép. Ha olyan a végső felhasználó, akkor végtelenül barát rendszert kell készíteni a számára. Erre van mód; ez a végső tanulság a fejlesztőknek. A felhasználó pedig érzékelhetné, hogy kettős értelemben is megnyílt előtte a pálya. Ha kicsit beleátná magát — nem nagy fáradsággal! — kis barátunk lelki világába, akkor többet és értelmesebben követelhetne a fejlesztőtől. Ez éppen úgy a saját érdeke, mint a saját játék most nem kell fejlesztő, most ez megcsinálom magamnak. Nem nehez.

### A búcsú

Sokat szövegeltem eddig barátunkról, a számítógépről. Ezt nem is lehet befejezni, csak abbahagyni. Adósságot is leróttam: a mikrogép mégsem aerobic.

Zárasképpen szeretném összegezni néhány „üzenetemet”, ha nem hangzik nagyképpűnek ez a kifejezés. Nézzék el nekem, mert a jó szándék vezetett. És a ritka helyzet. Fejlesztő is vagyok, végső felhasználó is. Érték is a számítógépekhez, meg — olyan igazán — nem is. Meg tudnék fejteni egy magas szintű programot, de nem mindegyiket tudnám jól megírni. Szóval picit a két véglet között állok, és céloom e kettő érdekeinek és lehetőségeinek egyesítése. Ezért írtam.

Remélem, hogy az ifjú tehetségek egy picit már látják, hogy a mikrogép nem pusztán játék, nemcsak szórakozás. Ugyanakkor — ha netán nem szeretett kötelező tárgyról lenne szó (remélem, hogy nem) —, a programozás nemcsak kín. Tehetek, amit akarok, ameddig magam vagyok a fejlesztő és a felhasználó. Ha azonban hivatásból végzem ezt a vonzó munkát, akkor el kell felednem fejlesztő mivoltomat, és át kell kapcsolnom a végső felhasználó hullámhosszára. Nem a BASIC nyelv a megváltó; nem ennek tudása a végcél. A géptől, a nyelvtől, a számítástechnikától nem tanulok feladatmegoldó készséget. Legfeljebb adott, szűk struktúrákban segít eligazodni a gép. Ha nem így lenne, akkor ugyebár, csak jó rendszerek szülnének. Teszik? Ha majd ifjú kollégám, ugyanúgy képes lesz figyelni a raktáros, a főkönyvelő, a személyzeti osztályvezető igényeire, mint a gép követelményeire, ha majd az ő szavukat ugyanolyan pontos kinnal kell betanodni — és ezt örömmel, a cél érdekében tesszed —, mint a DO CASE—END CASE párost, akkor megértél: szakemberré váltál. Addig csak amatőr vagy.

A kevésbé fiatal korosztály végre megpróbálhat kilépni eddigi szűk kereteiből. Kicsi odafigyeléssel naponta tudna érvelni. Nem igaz, hogy a gép azt úgy követeli. Nem valós, hogy amit kívánok, az lehetetlen. Téged csak akkor fizetlek meg, fejlesztő, ha valóban ízlé-

semnek megfelelően alakítod a rendszert. Ehhez azonban az kell, hogy ez a korosztály megismerkedjen a gép közvetlen természetével. Netán még az általam nem mindig helyeselt funkcióbillentyűkkel is. Talán a magasabb szintű nyelvek általános képességeivel és legvégül magával a nyelvvél is. Ez a könnyebb feladat. Megváltozni: ez a legnehezebb. Mert, ne feledjük, a fejlesztő is értékeli a felhasználót!

Nálunk számítógépes bérszámfejtés van, de nekem kell megadnom a kajajegyek mennyiségét: mikor voltam szabadságon. Nos, ott van a szabadságpapírom. Miért nem vonja le a gép? Talán a felhasználó volt tehetetlen a rendszer kijelölésekor? Talán nincs fantáziája, hogy az általam korábban említett feldolgozási láncban megtalálja a helyes arányokat? Talán a fiatalok mégsem tanulhatnak tőlünk feladatkielölést és megoldási készséget? Nagy kópé ez a mikró: hamar hű képet ad rólunk, alkalmazókról!

A mikró hamar visszarúg, szemben a lehetetlen és értelmetlen feladatokat is zavartalanul — bár drágán —, emésztgető nagygépekkel. Hamar kiderül, hogy az információs rendszer működtetésének költségei kettősek: emberi és technikai költségtényezőkből tevődnek össze. Nagygepnél főleg az utóbbi volt drága. Most, látszólag, olcsó. Mennyi emberi ráfordítás árán? Könnyen fejleszünk alkalmazásokat, de vigyázat! Sokat vagy kidobunk, vagy rabszolgájává válunk. Előny: legalább nem akarunk többpéldányos, több száz oldalas tablókat készíteni, amire a mikró nem való. Testre szabottabb lehet az információ. Hátrány: a mikró egy ideig tűri, hogy vele imitálják a nagygépet. Majd váratlanul áll fejre, mármint az a rendszer, amit csak nagygépre lett volna szabad tervezni.

Végül: újra kell együtt tanulnunk ezt a szakmát fejlesztőknek és végső felhasználóknak. Ami a nagygépek esetén kívánatos, de lehetetlen volt, az most elérkezett. Többet kell tudnia a sikerre vagyó fejlesztőnek a felhasználói mindennapokból. És mód van arra, hogy többet ismerjen a jó rendszer kívánó végső felhasználó a számítástechnika érintkezési felületéből. Ez barátunk legfőbb érdeme: szűkül a szakadék.

Most pedig nézem tizenhárom éves lányomat, aki bekapcsolta a kis IBM PC-t és akinek — egyelőre mint „végső felhasználónak” —, szeretettel magyarázom, mit hogyan. Legelső alkalom; még nem sokat ért mindebből. Mégis megdöbben, hogy pillanat alatt lekérdezheti társasági programjait, mert már vannak ilyenek. Öt percet bíbelődünk a géppel, felárazva magántárgyait, és máris meglepődik azon, hogy mennyi mindene van. Gyerekekre szabott készletnyilván-tartás.

Talán majd számítástechnikus lesz, talán nem. Szeretni fogja ezt a kis masinát. Előbb még hosszán kell nevelnem, hogy értesse meg a mások igényeit. Hogy tudja pontosan megfogalmazni saját feladatát. Ehhez már csak kiegészítés az a csöppnyi technikai tudás, amit barátunk megkövetel.

DR. HALASSY BÉLA



**A  $\mu$ M szerkesztői nagyon sok levélre személyes választ küldenek, ha a probléma nem bírja el a több hónapos válaszadást. Itt most megpróbáltunk néhány közérdekű levelet összegyűjteni és ezekre válaszolni.**

*Esztérgár Pál, Budapest,*

*Fehérvári út 35. 1117*

Eleget téve a — néhány hónappal ezelőtti — felhívásnak, ezúton keresem a MIKROSZÁMÍTÓGÉP MAGAZIN olvasói között a CPC (Schneider, Amstrad) számítógépek tulajdonosait. Én ugyanis ezzel a — Nyugat-Európában az egyik legelterjedtebb és nagyon sokat tudó — géppel, továbbá hozzá elég bőséges irodalommal, programokkal rendelkezem, és mivel jól átllok tapasztalatok dolgában, szívesen adok tanácsot mind a „LOCOMOTIVE BASIC”, mind a gépi nyelv vonatkozásában, de én is örömmel venném mások tapasztalatait.

Úgy hiszem, hogy az a kérésem, hogy a MIKROSZÁMÍTÓGÉP MAGAZIN tegyen közzé egy — CPC tulajdonosokat kereső — felhívást és regisztrálja a jelentkezőket, nem ellentétes a lap célkitűzéseivel, és elősegítheti ezen tényleg sokat tudó és sokoldalú géptípus néhány hazai példányának célszerű hasznosítását.

*A turistaforgalom növekedésével egyre színesebb lesz az itthoni házi-számítógép tábor. Ezért adunk teret szívesen a hasonló felhívásoknak; találjanak egymásra az azonos gépeket használók! Szeretném ezzel a  $\mu$  Klub mozgalmat is egy kissé felélelteni.*

*Menyhárt Tibor, Debrecen,*

*Gyepűsor u. 12. 4031*

A Mikroszámítógép Magazinban és egybeült közölt cikkek felkeltették az érdeklődésemet a FORTH programnyelv iránt. Bár sikerült hozzájutnom a ZXFORTH 1.1 programhoz, a ZX81-re írt ZXFORTH leírásához sajnos a mai napig sem.

Nem lehetne levelemnek ezt a részét „AZ OLVASÓ ÍRJA” rovatban lehozni? Talán valamilyik olvasó tudna nekem segíteni. Cserébe COMPILER-t, ASSEMBLER-t, DISASSEMBLER-t, TOOLKIT-ot, GRAPHICS TOOLKIT-ot, valamint játékprogramokat tudnék felajánlani. Köszönettel.

*Gerliás Péter, Budapest,*

*II., Kelemen L. u. 2. 10. ép. I. 2. 1026*

Segítségüket szeretném kérni a lapjuk ez év júniusi számában megjelent „Ékezetes ábécé C64-re” programmal kapcsolatban.

A programot többszöri próbálkozás ellenére sem sikerült megvalósítani. Rögtön az elején a POKE 44,16: POKE 45,61: POKE 4096+64,0 NEW parancs után a gépem syntax hibát jelez. Ez utasítássor nélkül legépel program pedig indítás után befagy. Lehet, hogy bennem van a hiba, de ha esetleg a sajtóhíba ördögbe incselkedett, kérem, írják meg a helyes megoldással együtt, mert nagyon szeretném, ha a gépem végre tudna magyarul.

*A program hibáinak javítását a  $\mu$  M októberi számában közöltük.*

*Balogh Gábor, Sopron,*

*Madách u. 8. 9400*

A szeptemberi Mikroszámítógép Magazinban olvastuk Pivarnyik Attila javaslatát, amellyel teljesen egyetértünk.

Ezzel a levéllel jelezni szeretnénk jelentkezésünket annak a két oldalnak a szerkesztésére — szerkesztésének segítése — amely a SPECTRUM számítógéppel foglalkozna.

Amennyiben igényt tartanak segítségünkre, kérjük jelezzék a fenti címen.

Ötleteinkből néhány:

- a gép néhány trükkje;
- rövid felhasználói programok (BASIC, gépi kód);
- rövid játékprogramok (BASIC, gépi kód);
- programozástechnikai problémák;
- játékvivat: hír, riportok, egyéb tanácsok;
- hosszabb gépi kódú vagy BASIC programok közlése több számon keresztül;
- SPECTRUM TOP lista (játékprogramok).

*Volt egy régi szerkesztőségi cikkem, amelynek az volt a címe — ha jól emlékszem — hogy „Nytott Szerkesztőség”, ami röviden azt jelenti, hogy szeretnénk a számítástechnika iránt érdeklődőket, főleg ha íráskészséggel is rendelkeznek, a szerkesztőségi munkába bekapcsolni. Nagyon sok ötletet kapunk, de sokkal kevesebb írást. Csak az elkészült cikkből tudjuk megítélni, hogy vállalkozunk-e hosszú távú együttműködésre vagy nem vállalkozunk, mert esetleg se a beküldött írás tartalma, se a stílusa nem üti meg azt a mértéket, amelyet munkatársainktól elvárunk.*

*Külső munkatársakkal az együttműködés úgy szokott elindulni, hogy küldenek egy írást, az megjelenik, azután jön a következő. Később az illetők eljárnak a szerkesztőségi megbeszélésekre, és észre sem vesszük, máris szoros a kapcsolat, számítunk egymásra. Sajnos, ennek az ellenkezője fordul elő legtöbbször. Jön egy ötlet, esetleg egy írás, utána pedig hiába várjuk a folytatást; rendszeresen írni egy havonta megjelenő magazinba nem kis vállalkozás.*

*Várjuk az írást másoktól is.*

*Fekete Sándor, Oradea 3700, Bihar, Romania*

*Breiner Béla u. 59/A*

Le kell szögezmem, hogy én már többször írtam önöknek. Tehát röviden rátérek a témára:

Nagyon szépen kérem önöket, ha lehet, adják közre a levelező rovatban e pár következő sort; persze lehet átdolgozva is, úgy, hogy a lényeg megmaradjon. Szeretném, ha közös erővel és elég kitartással létre lehetne hozni egy univerzális jellegű modern jelvényt, amely összekapcsolná a számítástechnikát szerető amatőröket; tehát ez valahogyan segítene abban, hogy a rokonlelkek egymásra találjanak. Nagy álmom, vagyis vágyam, hogy ez ne csak Magyarországon legyen érvényes, hanem ha nem is az egész világon, de Európában; sőt lejjebb is adom: vezessük be Európának ezen a részén. Lehetne jelvény helyett érme, esetleg plakett stb., stb. Van valakinek véleménye, hát az ötletemet hozzáfűzhet valami megjegyzést, észrevételt stb. stb.

Az igazság az, hogy én még nem terveztem meg a jelvény formáját és tartalmát, de ha gondolják, én megcsinálom.

*Az ötlet, amelyet hűséges levelezőnk küldött Erdélyből, érdekes. Azt hiszem, hogy nem vagyunk jelvény ügyben kompetensek, így nem merem biztatni, hogy kezdje el a tervezést. A  $\mu$  kluboknak van már egy jelvényük, talán alkalmas lehetne a szofiver és a hardver amatőrök összetartozásának kifejezésére is.*

*Török György, Gomba,*

*Rákóczi út 24. 2217*

Önök a Mikroszámítógép Magazin 1986. júliusi számában közölték egy cikket a fényceruza készítéséről ZX-SPECTRUM-hoz. Szeretném, ha közölnék, hogy hogyan lehetne fényceruzát készíteni C-64-höz.

*Talán egyik olvasónk tudna segíteni.*

*Várjuk a cikkét.*

*Juhász Sándor, Csánytelek,*

*Pusztaszeri út 59. 6647*

Örömmel olvasom az Önök lapját, mert minden számban van valami, ami érdekel. A közeljövőben én is szeretnék egy számítógépet. (Főiskolai hallgató vagyok a kecskeméti Gépipari és Automatizálási Műszaki Főiskolán, szüleim pedig kisiparosok.) A belföldi árak azonban igen magasak. Nemrégiben olvastam az Önök lapjában, hogy az ATARI 800XL számítógép, hozzá nyomtató plusz floppy, olcsó monitor és egy szövegszerkesztő program 1140 DM. Ezzel kapcsolatban lenne néhány kérdésem.

1. Bécsben mennyivel térnek el az árak az NSZK-tól (általában)?
2. Nem csalóka-e az alacsony ár, nem megy-e a gép rovására?
3. Önök szerint érdemes-e a fenti konfigurációt behozni, vagy valamilyen más típust javasolnának?

Más. Mi a véleményük a TV Computerről és a PRIMO-ról? Esetleg ezek valamelyikét vásároljam? Ezeket milyen a hang és a grafika?

*A levélből csak azokat a részleteket közöltük, amelyekre „időtálló” választ tudunk adni.*

*Elég sok hasonló levelet kapunk, amelyekre részben levélben válaszolunk, de sokszor ebben a rovatban is.*

*Általában az NSZK a „legolcsóbb” európai ország, ha valaki számítógépet, de általában elektronikus berendezést akar vásárolni. Ez nem zárja ki azokat az eseteket, amikor egy alkalmi kiadás alkalmával pl. Bécs is lehet nagyon olcsó. Az 1140 DM, ha akarom alacsony, ha akarom, magas ár. A nagyon alacsony ár mögött nagyon sokszor távol-keleti szállítót találunk, ilyenkor nem ritka a minőségi probléma, amelyre néhány olvasói levélben is találunk példát. Ami a típust illeti, én mindig olyan gépet vennék — ezt többször megírtam már —, amiből sok van az országban, és amihez szerviz is tudok találni. Legjobb tudomásom szerint ennek a követelménynek a külföldi gépek közül ma csak a Commodore felel meg.*

*Ami a hazai gépeket illeti, én pártolom a magyar ipart. Nem megvetendő előny, ha az ember a problémáit itthon el tudja intézni. A hazai választékból már csak a TV Computer maradt meg, a többi gyártó az olcsó gépekkel nem bírva a versenyt, „bedobta a törülközőt”.*



Papp László, József A. Fiúkollegium,

Székesfehérvár, Széchenyi u. 13. 8000

Én egy fehérvári diák vagyok, s szabad időmben programokat írok C-64-re. Írtam egy programot, egy új SAVE rutint, amelynek segítségével tetszőleges memóriarészt lehet elmenteni. Az Önök lapjában szeretném közzétenni.

Ha ez lehetséges, kérem írják meg, milyen leírást vagy listát kell küldennem a programról, és milyen feltételek mellett közlik.

Csak kipróbált és nyomtatón listázott programokat fogadunk el. Egy sorban 40 karakternél több ne legyen, különben a listát nem tudjuk jól törölni. A szerzők küldjenek jó programleírást. Ne feledjék, nem a padszomszédjuknak készül az írás, hanem egy újságnak, így legyen részletes, érthető, de ne legyen fecsegő, mert azt senki sem szereti.

Somfai Gábor, Budapest,

Rákóczi út 8/B. 1072

Lapjuk régi olvasójaként az alábbiakban összegyűjtöttem néhány olyan — technikai, illetve koncepcionális — ötletet, amelyek talán segíthetnek abban, hogy a lap színvonala még jobb legyen.

1. Kezdjük mindjárt a tartalomjegyzékkel. Sajnos a lapban szereplő legérdekesebb cikkek egy része a tartalomjegyzékben eleve nem szerepel. (Például a 86. júliusi számban a tartalomjegyzék-ből kimaradt a „Mibe kerül a 16 bit” című cikk, amely szerintem ebben a számban a legérdekesebb írás volt. (Javasolom azt is, hogy a tartalomjegyzék a címek mellett minden egyes esetben tüntesse fel azt is, hogy milyen géptípusra vonatkozik a cikk, ha ez lehetséges.) A lap jelenleg annyira eklektikus ugyanis, hogy nem ártana, ha már az első szemrevételezésnél mindenki látna, hogy egyáltalán foglalkoznak-e kedvenc gépével az adott számban.

2. A lapban közölt programoknál sok esetben teljesen olvashatatlanok a grafikus jelek. A legtöbb nyugati számítástechnikai lap ezen úgy segít — bár nyomdatechnikailag erre kevésbé vannak rászorulva —, hogy nem a képernyőn megjelenő jelet ábrázolják, hanem azt, hogy melyik billentyűt kell lenyomni a létrehozásához, ez általában jobban kislabilizálható. Arra lenne szükség, hogy a szöveg közben ne közöljenek programokat, ezeket a lap végén közöljék, géptípusonként csoportosítva, rövid magyarázattal arról, hogy a programot hogyan kell begépelni. Van egy másik segítség is: a jobb lapok ma már úgynevezett „proofreaderekkel” közlik a programokat, a profreader program betöltése után az új program minden egyes soránál valamilyen betű vagy számkombináció jelzi, hogy az adott sort jól gépelték-e be, nem a program végén derül ki, hogy az ötödik sorban hiba van. Javasolom, hogy a hibaigazításokat mindig a programrész végén közöljék, jelenleg ugyanis teljesen véletlenszerűen bukkanak rá korábbi számokban megjelent programok javítására.

3. Magyarországon ma a felhasználók nagy része csere útján jut a programjaihoz, illetve az azokra vonatkozó irodalomhoz. Közöljenek rendszeres összeállítású hazai klubhálózatról, bemutatóra a klubokat, címmel, az ott preferált géptípusok feltüntetésével stb., ez jelentősen megkönnyíthetné az ismeretek bővítését és az ehhez szükséges kapcsolatok létrehozását.

4. Fordítsanak nagyobb figyelmet a laikus felhasználókra, azokra, akik a gép működését, a

gépi kódú programozást stb. még nem ismerik, illetve sohasem fogják megismerni, mégis használni akarják gépüket anélkül, hogy számítástechnikai szakemberekkel válnának. Szerintem a géptulajdonosok közül sokan tartoznak ehhez a körhöz. Néhány tipp:

— Segítsenek eligazodni a géptípusok és programok tengerében. Olvasóik nagy része véletlenszerűen jut programokhoz, nincs áttekintésük arról, hogy milyen a kínálat, egy-egy célt milyen program szolgálna legjobban. Példamutatónak tartom e tekintetben a C-64 „Software alkalmazói segédlet” III. kötetében közölt összeállítást, amely hét adatfeldolgozó programot hasonlít össze, megkönnyítve az eligazodást. Javasolom, hogy a főbb géptípusokra készítsenek hasonló összefoglaló értékelést, legalább a szövegszerkesztő, a spreadsheet és a másoló programokról. — A nem szakértő felhasználó sokszor tengernyi irodalom átrágása után sem képes tisztába jönni a legelemibb trükkökkel, amelyek a gépe működtetéséhez szükségesek. Két példa:

1. A programok betöltése. Nem lenne érdektelen áttekinteni a betöltés feltételeit, azt a 10-20 trükköt, amellyel egy laikus rájöhet arra, hogy a kezében tartott mágneslemezről betöltött program miért nem indul el (pl. a programhoz bootot kell keresni, a program egy másik program része, gépi kódban íródott, és ezért más utasítással kell betölteni, nem a run parancs indítja, hanem valami más, pl. sys utasítás stb.). A géptulajdonosok nagy része ilyen elemi problémákkal küszködik.

2. Özöne létezik a másolóprogramoknak. Érdemes lenne összegezni azt, hogy mikor melyiket érdemes használni, hogyan kell lemásolni a lemásolhatatlan programokat. Saját példám alapján írom mindezt, én például heteket töltöttem azzal, hogy megpróbáltam lemásolni a „Master 64”-et, míg végül rátaláltam a B 52 című másolóprogramra, amellyel ez sikerült, és amelynek a létezéséről sejtelmem sem volt.

5. Közöljenek több ismertetést az egyes gépek alkalmazásáról, konkrétan ismertetve a megoldandó problémákat, a felhasználók számára is érthetően. Így például egy adott adatrendezési szoftvernél engem nem túlzottan érdekel, hogy a memóriában a program mit hova tett stb., érdekel viszont az, hogy a rendszer mekkora adathalmazt kezel, milyen típusú programmal stb. Mindezekből ugyanis rá tudok jönni arra, hogy a saját szakmai problémáimhoz milyen típusú megoldást kell keresnem, mielőtt még megtanulnék programozni. Így például adatrendezési problémáimnál én még nem tudom összehasonlítani a C-64-re készült „Superbase” és „Datamat” programok gépi kódját, számomra az a fontos információ, hogy a „Datamat” előnye a gyors, egyszerű kezelhetőség, de a „Superbase”-t kell használnom olyankor, ha bonyolultabb rendezési folyamatokat akarok elvégezni stb.

6. Sokkal több információt olvasnék szívesen a mikrogepek külföldi áráról, aktuális piaci helyzetükről. A hazai árak botrányosnak tartom, minősíthetetlen az, hogy a monopolhelyzetben levő hazai gyártók valamiféle kreált belső árral hasonlítják össze saját áraikat, így igazolva a nyugati árszintet akár tízszeresen is meghaladó áraikat. Ilyen témában foglalkozhatnának a magánimport nehézségeivel, a vámokkal stb. is, elég sok mondanivaló lenne ezen a területen is.

Somfai Gábor levelénél jobb befejezést nem tudtam volna találni, tanácsait köszönjük, megszívleljük.

További jó együttműködést kíván a szerkesztő:

KOVÁCS GYÓZÓ

A nyugatnémet TEDAS adatbank — amely a Mikrocomputer Zeitschrift folyóiratkiadó vállalat on-line szolgáltatása — egyik legutóbbi összeállításában értékelte a személyi számítógépek piacát. E szerint a személyi számítógépek gyártói eddig 13 millió darab PC-t értékesítettek. Az elkelt berendezések száma — amennyiben a jelenlegi tendencia folytatódik — 1990-re el fogja érni a 40 millió darabot.

Európában 1985-ben mintegy 4,6 milliárd dollár értékben adtak el személyi számítógépeket. A jelenlegi piaci trendek még egy ideig várhatóan nem fognak lényegesen megváltozni, csak az arányok tolódnak el. 1992-re a becslések szerint a másodgépek, valamint az elavult berendezések pótlására megvásárolt új gépek a forgalom több mint ötven százalékát fogják kitenni.

1987-ben várhatóan 1,7 millió eredeti IBM PC és ezzel szemben mintegy 2 millió (!) IBM PC kópia — a Távol-Keletről és egyéb országokból származó hamisítvány — kerül a felhasználókhoz a téma szakértőinek mértékadó véleménye szerint.

kj

**Az 1986. októberi szám 25. oldalának utolsó mondata szerint „az igen jónak bizonyuló gyors rendezőt... a szerencsések megtalálhatják a könyvben”.**

Olvasóinkat „szerencsés” tesszük, és közöljük mind a gyors rendező BASIC-programját (1. lista), mind a Shell—Metzner rendezőt (2. lista, mely egy utasítással rövidebb, mint a cikkben a könyvből idézett) a legtöbb gépen futó BASIC-változatban. Mindkét programban, ha a tétel számja több mint 100, a 10. sort módosítani kell.

```
5 REM GYORSRENDEZŐ
10 DIM A(100):F(15):L(15)
20 INPUT "TEETELSAAM":N
21 FOR I=1 TO N
22 A(I)=1+INT(RND(O)*N):PRINT A(I):NEXT I
30 S=0:F=1:L=N
40 M=INT((L+F)/2):I=F:J=L
50 IF A(I)<M THEN I=I+1:GOTO50
60 IF A(J)>M THEN J=J-1:GOTO60
70 IF I>J THEN 120
80 IF I=J THEN 100
90 T=A(I):A(I)=A(J):A(J)=T
100 I=I+1:J=J-1
110 IF I<=J THEN 50
120 IF I>=L THEN 140
130 F(S)=I:L(S)=L:S=S+1
140 L=J:IF F<L THEN 40
150 IF S=0 THEN 170
160 S=S-1:F=F(S):L=L(S):GOTO40
170 FOR I=1 TO N:PRINT A(I):NEXT I
```

```
5 REM SHELL-METZNER RENDEZŐ
10 DIM A(100)
20 INPUT "TEETELSAAM":N
21 FOR I=1 TO N
22 A(I)=1+INT(RND(O)*N):PRINT A(I):NEXT I
30 M=N
40 M=INT(M/2):IF M=0 THEN 120
50 K=N-M:J=1
60 I=J
70 L=I+M:IF A(I)<=A(L) THEN 100
80 T=A(I):A(I)=A(L):A(L)=T:I=I-M
85 IF I>=1 THEN 70
100 J=J+1:IF J>K THEN 40
110 GOTO60
120 FOR I=1 TO N:PRINT A(I):NEXT I
```



**Vadnai Szabolcs:**  
**Commodore 64 programozói zsebkönyv**  
 (Budapest, 1985.  
 Novotrade Rt. 68 oldal.  
 Ára: 149,— Ft.)

A zsebkönyv célja, hogy az olvasónak egy helyen összegyűjtve szolgáljon mindazokkal az információkkal, melyeket a C64 amatőr és szakember-felhasználói eddig csak fáradsággal, több könyvből tudtak összeszedni. A kötet főbb fejezetei: BASIC, Simon's BASIC, Hang és zene, Grafika, Gépi kódú programozás, Help kártya használata. A függelékben kódtáblázatokat és a mágneslemezekkel kapcsolatos tudnivalókat találja az olvasó.

**1001 játék és a GRAPHICS BASIC**  
**Commodore 64-re**  
 (Írták: Erdős Iván, Schmidt Endre, Németh István, Székely László)  
 (Budapest, 1985.  
 LSI ATSZ, 115 oldal.  
 Ára: 100,— Ft.)

A kötet gyerekeknek, felnőtteknek, mindazoknak szól, akik C64 számítógép tulajdonosok vagy kedvelők, akik szórakozni és játszani szeretnének a számítógéppel. Hazánkban igen sok számítógépprogram van forgalomban, melyek használatát nem mindenki ismeri. Ez a kötet tartalmazza a legelterjedtebb C64-re írt játékok ismertetését és egy 1000 címszóból álló minilexikon is, amely útmutatóul szolgál az ügyességi, logikai, stratégiai játékok közötti eligazodáshoz.

Azoknak a felhasználóknak, akik saját játékleveleiket kívánják megvalósítani, a könyv harmadik fejezete nyújt segítséget, melyben a GRAPHICS BASIC című BASIC-bővítőt ismertetik a szerzők. Ennek használatával igen egyszerűvé válik a C64 számítógép képernyőkezelése.

**Hofmanné Boskovitz Éva:**  
**A Z80 assembly programozása**  
 (Budapest, 1985.  
 LSI ATSZ, 368 oldal.  
 Ára: 249,— Ft.)

A könyv azok számára készült, akik a Z80-as mikroprocesszorral még nem foglalkoztak, és akiknek az assembly programozás még nem a szakterületük.

A kiadvány főbb témakörei: a programfejlesztés rendszertechnikai eszközei, gépi kód, assembler, magas szintű nyelv, adatmozgatás, Z80 utasítások, programok, bemeneti-kimeneti műveletek, adatkezelés.

A példaprogramok tematikusan épülnek egymásra, és sok programozási fogást, trükköt tartalmaznak. A könyv ismerteti a Magyarországon leginkább elterjedt fordítóprogramok kezelését is.

**Szűcs Pál:**  
**Video kézikönyv**  
 (Budapest, 1985.  
 OMIKK, 323 oldal.  
 Ára: 117,— Ft.)

A kézikönyv a hazai videofelhasználók számára foglalja össze azokat az információkat, melyek a videózás megkezdéséhez nyújtanak segítséget.

A kötet a televíziós technika alapjait, az NTSC, a SECAM és a PAL rendszert ismerteti. Leírja a videokamerák fajtáit, a képmagnetofonok, a monitorok, a tévé-vevőkészülékek és a videovevők legfontosabb jellemzőit. Ismerteti a video fejlődésének új irányait: a műholdas és a kábeltelevíziót, a képlemezeket és a videotex rendszereket. Végül röviden összefoglalja a video használatát az oktatásban.

**Bárdos Attila—**  
**Körtvélyesi Géznáné:**  
**Programozási alapfeladatok gyűjteménye**  
 (Budapest, 1985.  
 SZÁMALK, 209 oldal.  
 Ára: 101,— Ft.)

A 444 programozási alapfeladatból álló gyűjtemény célja a programozás tanulásának és tanításának elősegítése. A kötet szerzői olyan fogásokat, módszereket ismertetnek, melyek begyakorlása révén jártasságot szerezhet az olvasó a problémamegoldás területén. A feladatok nehézségi foka az egy-két utasítástól a 2-300 utasításnál is több eljárást tartalmazó bonyolultabb programokig terjed.

A feladatgyűjtemény témái: elemi algoritmusok, ciklusok, alapalgoritmusok, adatfeldolgozási algoritmusok, vegyes feladatok. A fejezeteken belül a nehézségi fok szerint csoportosítja a feladatokat, melyek megoldásához számos ötletet kínál. Ezekhez az önálló megoldások helyességének ellenőrzését segítő programterv, algoritmus és program kapcsolódik.

**Madarász László:**  
**Digitális CMOS kapcsolásgyűjtemény**  
 (Budapest, 1986.  
 Műszaki Könyvkiadó,  
 204 oldal. Ára: 93,— Ft.)



A könyv a CMOS technológiával készülő digitális integrált áramkörök felépítésével, típusaival és alkalmazásával foglalkozik. Közel száz áramkört részletesen ismerteti. A szerző közli a kapcsolásokban felhasznált integrált áramkörök bekötési rajzát és főbb adatait is, ez önmagában is nagy érdeklődésre tarthat számot. Mivel az eddig megjelent, digitális technikával foglalkozó könyvek a CMOS áramkörök gyakorlati alkalmazásával csak érintőlegesen foglalkoztak, ez a könyv mind az ipari felhasználók, mind pedig az amatőrök számára a legújabb ismereteket nyújtja.

**Mi micsoda magyarul a számítástechnikában?**  
**Mikroszámítógépes értelmező szótár**  
**Szerkesztette: Kis Ádám**  
 (Budapest, 1986.  
 Tömegkommunikációs Kutatóközpont,  
 171 oldal. Ára: 75,— Ft.)

A szótár készítői az utóbbi évek szakirodalmában mind gyakrabban felbukkanó számítógépes fogalmakat jól kezelhető szótárba gyűjtötték. Céljuk az volt, hogy a számítástechnika iránt érdeklődők, elsősorban a tanulóifjúság segítséget kapjon a folyóiratokban, gépismertetőkből, programnyelvekkel kapcsolatos leírásokban található fogalmak értelmezéséhez. A szerzők

ismeretanyagból elsősorban a mikroszámítógépek és a személyi számítógépek tanulmányozása során előkerülő szakkifejezéseket emelték ki. A magyar—angol számítástechnikai kisszótár (tévésen az Angol—magyar számítástechnikai kisszótár címet viseli) a fogalmak rövid magyarázatát is megadja. Az angol—magyar szójegyzék elsősorban az angol szakirodalom olvasóinak nyújt segítséget a legfontosabb kifejezések értelmezésében.

Az olvasó a kötet végén külön összegyűjtve találja az adatátvitellel kapcsolatos angol—magyar szöveget és a BASIC-nyelv utasításainak gyűjteményét.

**Szlávi Péter—Zsakó László:**  
**Módszeres programozás**  
 (Budapest, 1986.  
 Műszaki Könyvkiadó,  
 116 oldal. Ára: 50,— Ft.)

A szerzők, miután konkrét példák alapján megfogalmazzák a programkészítéssel kapcsolatos problémákat, közérthető stílusban szólnak azok megoldásáról, a korszerű strukturált programozási módszerekről. Az elveket több példán keresztül mutatják be. Módszert adnak a programok helyességének bizonyítására, a hatékonyság növelésére. A könyvben ismertetett elveket megfelelően egy elvont, magyar alapszavakból álló magas szintű programozási nyelvet definiálnak, ebben írják programpéldáikat is, de a függelékben megadják a különböző BASIC-változatokra vonatkozó átkódolási szabályokat is.

**Erdős Iván:**  
**A COMMODORE 64 assembly nyelvű programozása**  
 Budapest, 1986.  
 LSI ATSZ, 146 oldal.  
 Ára: 138,— Ft.)

A könyv közérthető módon ismerteti a C64 processzorának utasításkészletét és az assembly programozást segítő — hazánkban legelterjedtebb — programcsomagokat. Módszertani utasításokat is közöl, hogy az assembly nyelvet mikor és hogyan kell használni a C64 esetében.

## MIKROSZÁMÍTÓGÉP-FELHASZNÁLÓK

### \$ helyet

#### 1. ajánlatunk:

WINCHESTER DISC CONTROLLER  
 SCSI (SASI) interface  
 — max. 4 db tetszőleges típusú és kapacitású  
 — 5,25" Winchester disc drive vezérlése  
 — funkcionálisan kompatibilis a XEBEC S1410, III. WD 1002-SHD vezérlőkkel  
 Ár: tartozékokkal 59 900 Ft

#### 2. ajánlatunk:

IBM KOMPATIBILIS FLOPPY DISC DRIVE CONTROLLER  
 Ár: tartozékokkal 18 900 Ft  
 Szállítás raktárról  
 12 hónap garancia



## Centrumban

A Centrum Áruházak Vállalat egyszerre 30 áruházában kezdett forgalmazni mikroszámítógépet, mégpedig a Videoton által gyártott TV-Computert. Különösen jelentős az a tény, hogy az áruházak közül több mint 20 vidéken van, szét-szórva az egész országban. Így végső soron a Centrum is aktívan részt vesz a számítástechnikai eszközök terjesztésén túl a számítástechnikai kultúra országos terjesztésében.

## Spectrum – Commodore 64 kapcsolat

Kétféle ár–teljesítmény kapcsolatot épített ki a Házi-számítógép-építő Klub, a HCC a ZX-Spectrum és a Commodore 64 között. Az olcsóbb változat pár száz forintos áron 50 baud sebességű átvitelt biztosít a Spectrumból a C64-be. Képernyőtartalom és adatállomány egyaránt átküldhető közvetlen üzemmódban.

Mintegy tíz-hússzoros áron az átvitel sebessége is tíz-hússzorosára nő, és ez a megoldás már kétirányú kapcsolatot is biztosít.

## Lakossági számítógépszerviz

1984 januárjában alakította meg az Agroindusztria Innovációs Vállalat a Professional PC szervizt, elsősorban az SZKI által készített M08X és Proper típusú számítógépek javítására. Azóta profiljuk kibővült, és a házi-számítógépektől kezdve az IBM PC-ig mindennel foglalkoznak. Az igényeknek megfelelően, a közületi szolgáltatásokon kívül alig fél éve megalakult a lakossági szolgáltató részleg is, mely főképpen a Commodore 64 és a Sinclair Spectrum gépek javításával foglalkozik.

A Professional PC szerviz legfontosabb jellemzője a gyorsaság. Ennek érdekében a felhasználók közelébe kell települnie. Ezért hozta létre országos hálózatát: ma már Budapesten kívül hat másik városban is van 4-6 fős kirendeltsége. Ezek nem csupán rendelésvégző helyek, hanem maguk is javítanak. Nem utolsósorban e hálózatnak is köszönhető, hogy a szervizszerződést kötő intézményekhez Budapesten 24 órán, vidéken pedig 48 órán belüli kiszállást vállalnak.

## Angliai telefonok

Erőteljesen növekszik a kereslet Angliában a kommunikációs eszközök, így a korszerű telefonok iránt. Jól jellemzi a helyzetet az ország telefonellátottságának alakulása: 1982 közepén 28,5 millió telefon volt az országban, azaz ezer lakosra 500 telefon jutott; az előrejelzések szerint 1988-ban már 40 millió készülék lesz.

A telefon ugyanakkor egyre intelligensebbé válik, egyre többet tud. Ez elsősorban az üzleti célokat szolgáló 10 millió készülékre vonatkozik, melyek közül az újak már számítógéphez csatlakozva címtárakhoz, határidőnaplókhoz is kapcsolatot biztosítanak. További jelentős fejlődés várható, ha a nyilvános hálózatban a jelátvitel teljes egészében digitálissá válik.

## Keresztnevek

Az ifjú szülők jól tudják, milyen nehéz feladat az újszülött gyermek számára megfelelő nevet választani. Általában rengeteg fejtörés, tanácskozás, a nagyszülők, a rokonság, az ismerősök, a barátok széles körű bevonása szokta megelőzni a névadást.

Ezekon a gondokon kíván most segíteni Franciaországban egy újfajta szolgáltatás. A dél-franciaországi Annecyben megnyílt egy találékony fiatalember, a 28 éves Claude Bouvier irodája, amely számítógép bevonásával siet a szülők segítségére. Begyűjti az újszülöttre vonatkozó adatokat, a szülők kívánságait, a rokonság, a barátok véleményét, az általuk preferált neveket és mindenféle információt, többek között a gyermek nemét, a szülők családnévét, azt a kívánságukat, hogy milyen hosszú legyen a keresztnév, összhangban legyen-e a családnévvel, modern legyen-e vagy régi vágású stb. Mindezeknek az adatoknak a bevitel után a számítógép tíz keresztnévet terjeszt elő, ezek közül választhatják ki a szülők a nekik legmegfelelőbbet.



## Árakkal

1985 őszén fejlesztette ki a Besztercebányai Számítógépgyár Csehszlovákia első, az IBM PC-vel kompatibilis mikroszámítógépet, a PP-06A-t. Egy évvel később, 1986 őszén ugyanitt készült el Csehszlovákia első, az IBM PC XT-vel kompatibilis típusa, a PP-06B. Újdonságot jelent, hogy

az ezeket a típusokat hirdető prospektuson már is található: az A változat 100 ezer, a B változat 150 ezer koronába kerül (220 ezer és 330 ezer forint). Ez még sokáig eszmei ár marad, mert az idén csupán 50, 1988-ban pedig az A változathoz 100, a B változathoz pedig 300 darab gyártását tervezik.

A gépekhez kétféle operációs rendszert adnak: a PP-DOS az MS-DOS, a MIKROS-86 pedig a CP/M-86 megfelelője.



A PP-06 típusú mikroszámítógép

## Kardiológia

A szovjetunióbeli Kisinyovban a szívbetegséggel való küzdelem hatékonyságának javítására központi kardiológiai adatbázist hoztak létre, mely a betegek személyi adatain kívül kórtörténetüket és néhány legutolsó EKG-jüket is tartalmazza. Az utóbbiaknak a számítógéphez való továbbítására a városi telefonhálózatot használják. Az orvosokat ellátták egy hordozható, az EKG mérésére és továbbítására szolgáló, Volna nevű készülékkel.

Az EKG-felvétel a következőképpen történik. Az orvos akár a beteg lakásán is elkészíti az EKG-t, majd közvetlenül a beteg lakásából felhívja a kardiológiai központ számát. Bediktálja a diszpécsernek a beteg azonosító adatait, majd a telefonra egy tapadókorongot helyezve, a Volna típusú készülék továbbítja az EKG-mérés eredményét. A vételt a hierarchikus rendszer első szintjén lévő Elektronika-60 típusú mikroszámítógép végzi.

# ÉP-GYÁRTÓK, FIGYELMÉBE

t Ft!

**ÁRENGEDMÉNY**  
nagyobb darabszámú  
rendelés esetén  
Felvilágosítás:  
Gigor Györgyné  
Telefon: 693-253  
Levélcím:  
1369 Budapest Pf. 317.



**mikromatika**



Ez végrehajtja a zavarosúrést, a zavarok kiegyenlítését, továbbá kijelöli az EKG-görbén a jellemző szakaszokat, majd feladja az így előkészített görbét egy SZM-4 típusú minigépre. Ez behelyezi az új EKG-t a beteg már meglévő adatai közé, illetve új beteg esetén felviszi a személyi adatokat és elvégzi a „betegfelvételt”. Ezután a friss EKG, továbbá a rendelkezésre álló kórtörténet alapján egy relációs szervezésű tudásbázis felhasználásával az orvosszakértői rendszer diagnózist készít. Mindez néhány másodpercet igényel csupán. Ezért a utzpécs az EKG-vizsgálat után az előtte lévő képernyőről azonnal beolvassa a telefonba az orvos számára a diagnózist.

Az 1985 óta sikeresen üzemelő rendszerbe évente 6-7 ezer EKG-mérés érkezik be. A kezelési hatékonyság javulásának eredményeiről az eltelt időszak rövidsége miatt még korai beszélni, de mind a betegek, mind az orvosok ma már nélkülözhetetlennek tartják az új rendszert.

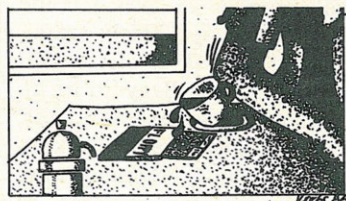
## Adatmentők

Kávét vagy üdítőt ömlik a hajlékonylemeze — tipikus esemény az irodai alkalmazásoknál. Ám a lemezen lévő információ ilyenkor elvész, ami sokszor pótolhatatlan kárt okoz. Ennek elhárítására készült fel a hajlékonylemezeket is gyártó svéd Kopparberg cég. Az ősszel Budapesten tartott bemutatójukon érdeklődők nagy tömege követte végig a mosásból, vegyszeres kezelésből és a borítékjából kivett hajlékonylemez felületi csiszolásából álló mentési technológiai folyamatot. Az eredmény: az IBM PC ismét el tudta olvasni a lemezen lévő információt, így az átmenthetővé vált egy másik, új lemezre.

A bemutató során elmondták, hogy felkészültek a „gyengén” felírt lemezek olvasására is. Ez a probléma általában a lemezek többéves tárolása után jelentkezik: váratlanul kiderül, hogy a mikroszámítógép már nem képes elolvasni a felírt információt. Ekkor a labo-

ratóriumukban egy rendkívül érzékeny géppel olvassák végig a lemezt, amely az igen „halk” jelet is képes olvasni. Ez azonban a „zajok” felerősödéséhez is vezet. A zajt a technológia következő lépcsőfokán kiszűrik.

A Kopparberg cég adatmentési szolgáltatása saját gyártmányú lemezei esetében Svédországban ingyenes. Már folytak a tárgyalások egy magyar-svéd vegyesvállalat létrehozására, mely előreláthatólag az év közepén hazánkban is megnyitja adatmentő szolgáltatását.



## Spectrum + CP/M

A lengyel Nevelés- és Tudományügyi Minisztérium tavaly iskolaszámítógép-pályázatot hirdetett. A megadott követelményrendszer első pontja volt, hogy a pályázó gépnek kompatibilisnek kell lennie a Lengyelországban jelenleg legjobban elterjedt ZX-Spectrum-mal. A többi követelmény között találjuk a szocialista gyártmányú alkatrészbázis igényét, a lokális hálózatba kötés lehetőségét a szaktantermek kialakíthatóságához, a jó kiépíthetőséget (hajlékonylemezes tároló, lengyel gyártmányú nyomtató csatlakoztathatósága), a maximum 100 ezer zlotys árát s az akár évi száz-ezer darab gyártására is képes kapacitást.

A pályázatot az Elwro 800 Junior nyerte, ami egyben az 1985-ben bemutatott Elwro 700 háziszámítógép gyártás-előkészítésének leállítását is jelentette. A Junior nem csupán a Spectrummal kompatibilis, hanem a CP/M operációs rendszer is fut rajta. Mindkét üzemmódjában képes kezelni a hozzá csatlakoztatott hajlékonylemezes tárolót és nyomtatót. Az új iskolaszámítógépet 1986 nyarán a pozsnai vásáron mutatták be elsőként a nagyközönségnek.

Az Egyesült Államokban minden évben rendeznek néhány nagyobb számítógép-tudományi konferenciát, kiállítással egybekötve, az ország különböző részein. Ezek egyikét, a Nemzeti Számítógép Konferenciát (National Computer Conference — NCC '86) 1986-ban a Nevada állambeli Las Vegasban tartották, június 16-tól 19-ig.

Az NCC-t ezúttal a legrangosabb szakmai társaságok védnöksége alatt tartották meg. A konferencia jelszava: „A számítástechnika dinamikus kiterjesztése a vezetők, a szakemberek és a felhasználók körében” volt. A szervezők azt mondták, hogy ez lesz a számítástechnika „rendezvényeinek rendezvénye” (Show of Shows).

A rendezőség igyekezett is a résztvevőknek mindent megadni; például a szállodák és a néhány kilométerre levő kiállítási csarnok között ingyenes autóbuszok álltak rendelkezésre, volt üzenetközvetítés számítógépes hirdetőtáblán, ezenkívül a Computer World és egy másik, számítógépeket forgalmazó szaklap ingyenes napi különkiadásban jelent meg. Mindezek ellenére ez a rendezvény nem volt annyira sikeres, mint az előzőek.

Az 1973-as, első NCC, melyet New Yorkban tartottak, a „mainframe”-ekkel foglalkozott, és közönsége inkább a tudományos kutatók közül került ki. Mostanára azonban megváltozott az eredeti cél, és az utóbbi konferenciának a nagyközönséget igyekeztek megnyerni. Az üzletemberektől a felhasználóig terjedő spektrumban a téma viszont túl általánossá vált. A sikertelenség fő oka mégis az volt, hogy az elmúlt hónapokban már több hasonló, de jobban specializált kiállítást rendeztek. Rövid idő alatt nem lehet sok újat látni, és mind a kiállítók, mind a látogatók kezdtek belefáradni a sok rendezvénybe.

A kiállítók száma ezúttal mindössze 400 volt a néhány évvel korábbi közel ezerrel szemben, a 25 ezer látogató pedig csupán az egy-egyede a négy évvel ezelőttinél. Más kérdés, hogy akkoriban a rendezvény túlméretezettségét panaszolták.

Az egyetlen szenzáció az volt, hogy az IBM ezt a kiállítást választotta új gyártmányainak bejelentésére. Néhány nagyobb amerikai hardver cég (Digital Equipment, Data General, Apple, Compaq) és mikrosoftver-vállalkozás (Ash-ton-Tate, Lotus, Microsoft) távol maradt az NCC-től. Ezzel szemben, főleg a japán cégeknek köszönhetően, a nemzetközi kiállítók és a nemzetközi látogatók aránya nőtt az előző évekhez képest.

Egy egész napot fordítottak a konferencián a szakemberek és az egyetemi hallgatók találkozására. A tapasztalatátadási lehetőség ab-

ban segítette a fiatalokat, hogy végiggondolják: hogyan kell megtervezni az előmenetelt az informatikai szakmában.

Hangsúlyozták, hogy csak azok a számítástechnikai vállalkozások lesznek sikeresek, melyek a megoldásra összpontosítanak, és csak azok a felhasználók, akik az információt a stratégiájukban képesek alkalmazni.

Az NCC-n első ízben osztottak ki elismerő okleveleket az elmúlt évben kiváló eredményt elért három szakembernek és három gyártmánynak. Ez utóbbi kategóriában a Plus Development Corporation jutalmazták a 10 megabajtos „Hardcard”-ért, a Telos Software Products céget a „Business Filevision” adatrendszerért és a Tera-data Corporation a DBC/1012 „Database Computer System”-ért.

Az NCC '86 fénypontjaként említett IBM-bejelentésben a több mint 125 új vagy továbbfejlesztett gyártmány közül a következők voltak a legjelentősebbek:

**System/36.** Új modell gyorsabb feldolgozóegységgel, nagyobb központi tárral és kiépíthetőséggel. Bővült az irodai alkalmazások lehetősége, és továbbfejlesztették az operációs rendszert is.

**System/38.** Újabb modell, nagyobb teljesítmény.

**Összekapcsolt System/36 — System/38.** A berendezések egymás erőforrásait képesek használni.

**Új megjelenítő rendszerek.** Display (IBM 3171, 3190, 3191 stb.), grafikus beviteli egységek (IBM 3117 és 3118).

**„InfoAblak” (InfoWindow).** IBM PC érintő képernyővel, amely videolemezzel összekapcsolva sokféle információ párbeszédessé megjelenítésére alkalmas.

**Displaywrite 4.** A szövegfeldolgozó szoftver feljavított változata, amely a PC és System/36 közötti „seamless” összekapcsolásra alkalmas.

Az IBM PC RT modellt is bemutatották, amelynek megnövelték a teljesítményét, megváltoztatva a hardvert és a szoftvert. A rendszer gyűrűs hálózatba kapcsolható.

A másik jelentős bejelentést az American Telephone and Telegraph (AT&T) tette; piacra hozta ugyanis a régóta várt Unix System V operációs rendszer 3.0 változatát. Ezzel különböző rendszerű számítógépeket lehet összekötni az adathálózaton. Megemlíthetjük még a Ricoh System cég két új, perccenként 40 oldal teljesítményű lézernyomtatóját, a Leading Edge új modemjét és PC/AT rendszerű mikroszámítógépet, a NEC Information System miniszámítógépeit, a Zenith Data Systems és a Toshiba cég öljebe vehető, hordozható számítógépeit, valamint a Fujitsu cég dokumentumolvasó egységét.

A tudományos kutatási és fej-



# NCC'86

## A világ legnagyobb show-ja

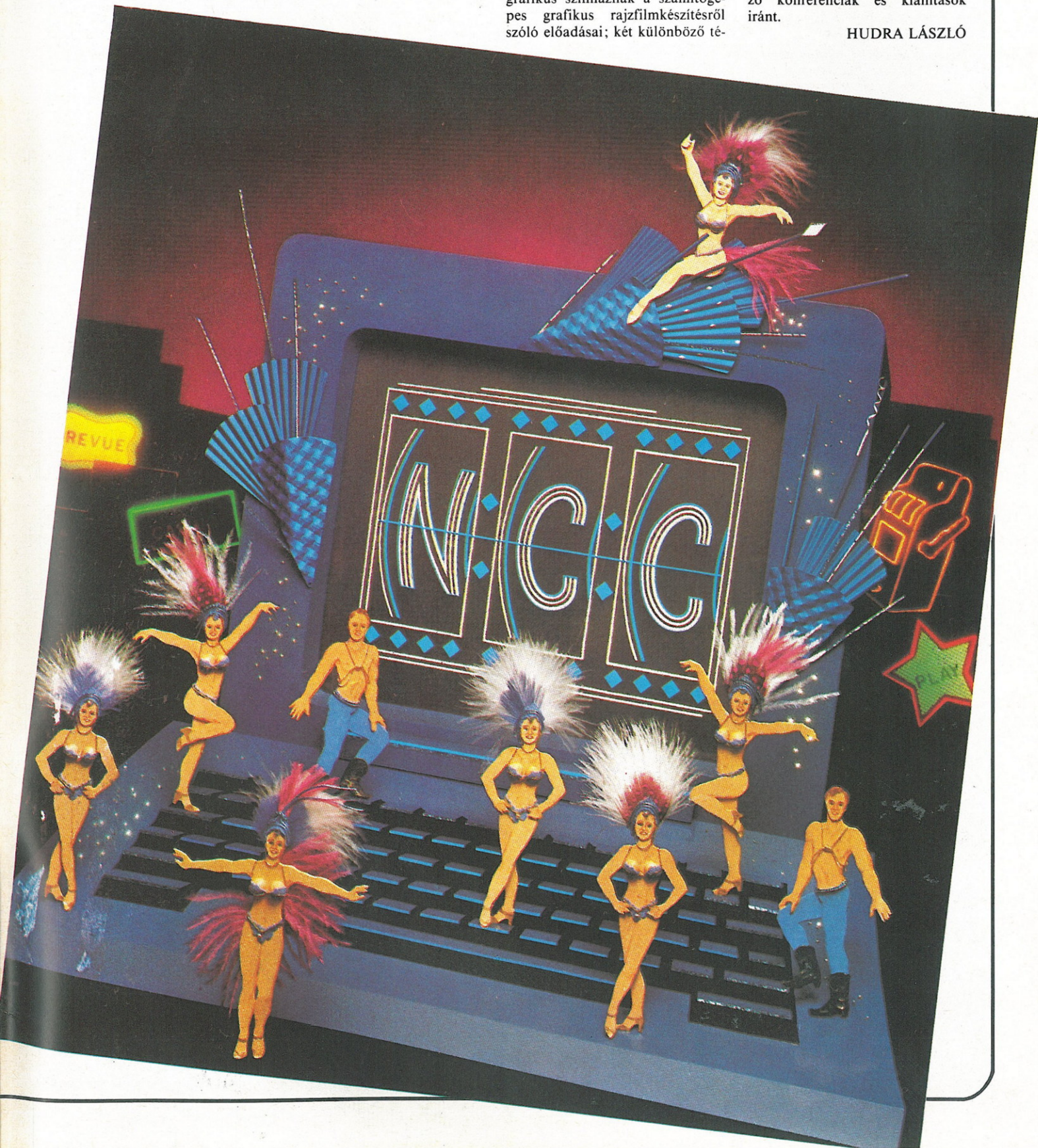
lesztési témák kiállítását külön tartották. Ezen több amerikai és egy kanadai egyetem, valamint kutatóintézetek vettek részt. Itt minden nap tájékoztatták az érdeklődőket a különböző számítástudományi és technológiai kutatási eredményekről. Az utolsó napon rendezték meg a kutatás és fejlesztés napját, továbbá a kisebb üzleti vállalkozások napját. Ez utóbbin a számítógépek alkalmazásának előnyeit és problémáit vitatták meg.

Igen érdekeseek voltak az ún. grafikus színháznak a számítógépes grafikus rajzfilmkészítésről szóló előadásai; két különböző té-

mát mutattak be, naponta kétszer. Látni lehetett egy háromdimenziós képet, amikor is a vetítövászaronról kinyúló kígyószerű tünemény szinte a nézők feje felett lebegett.

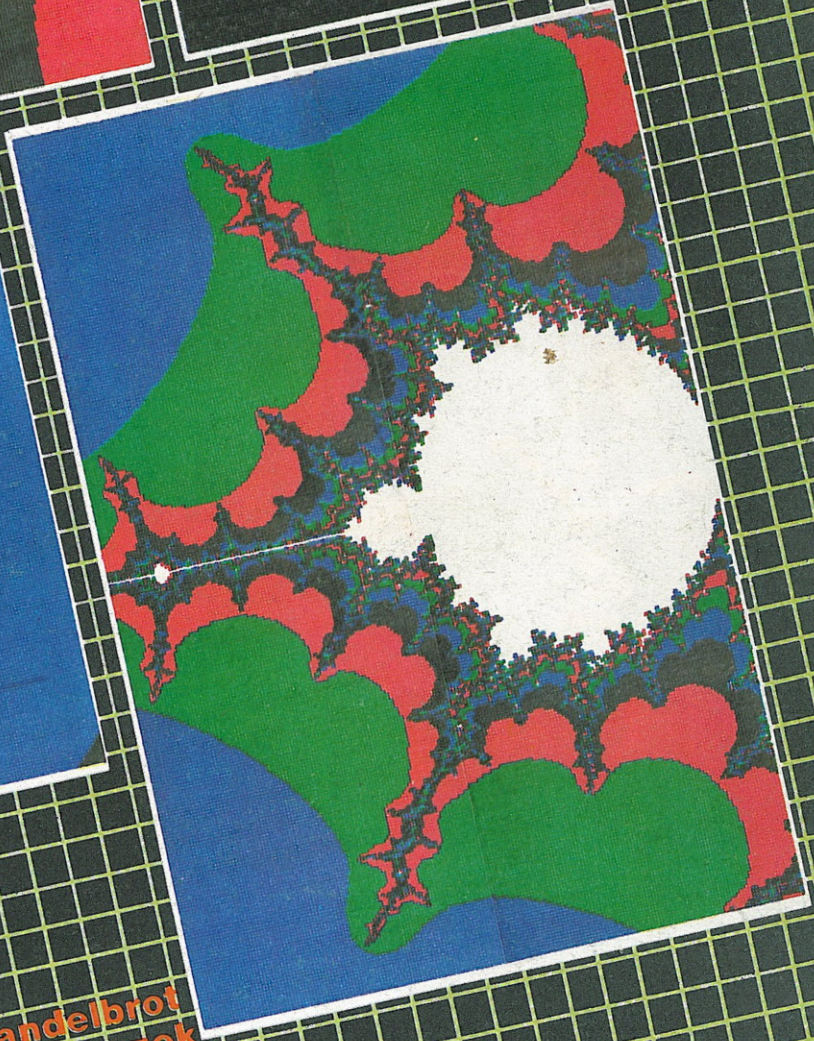
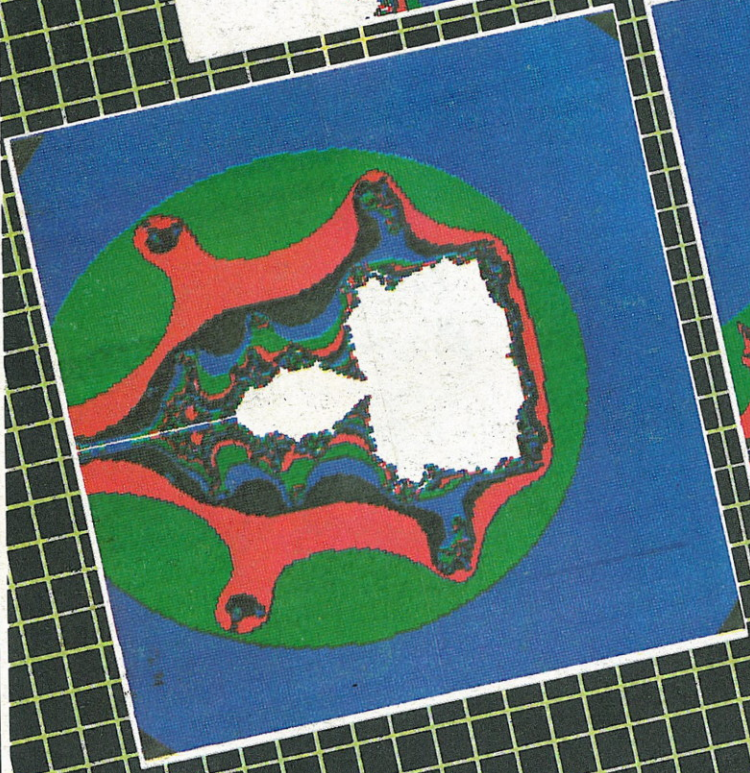
Erről a kevésbé sikeres NCC-ről is a legtöbb látogató valószínűleg meglepéssel távozott. A társaságok máris szervezik és hirdetik a további speciális kiállításokat és a Chicagóban rendezendő NCC '87-et. Az NCC '86 tapasztalatainak felhasználásával bizonyára fokozni tudják az érdeklődést a következő konferenciák és kiállítások iránt.

HUDRA LÁSZLÓ





# Tudomány és fantázia?



Mandelbrot  
halmazok  
PTA 4000-rel