MIKROSZÁMÍTÓGÉP MAGAZIN

A NEUMANN JÁNOS SZÁMÍTÓGÉP-TUDOMÁNYI TÁRSASÁG LAPJA

1986
március
Ára: 30 Ft

AJÁNLJUK A KÖVETKEZŐ MIKROSZÁMÍTÓGÉPEKRE KÉSZÍTETT RENDSZEREINKET:

főkönyvi könyvelés
folyószámla-vezetés
bér-, munkaügyi rendszer
állóeszköz-nyilvántartás
készletnyilvántartás
tanácsigazgatási alkalmazási szoftverek
kereskedelmi rendszer
pénztárgépes hálózat
nagyszámítógép-mikrogép kapcsolat kiépítése
adatrögzítési rendszerszoftver

A rendszerek az alábbi mikroszámítógépeken futtathatók:

IBM XT, AT
IBM kompatibilis mikrogépek
COMPUT—80
Z—80 alapú mikrogépek

Általános számítástechnikai szolgáltatásaink:

adatrögzítés
rendszerek üzemeltetése
rendszerszervezés
programozás
mikroszámítógépes rendszerek fejlesztése
többfelhasználós mikroszámítógépes rendszerek
(hardver + szoftver) bérbeadása

További felvilágosítással szívesen állunk rendelkezésükre a 131-072-es telefonszámon, illetve levélben.

Címünk: SZÁMSZÖV 1430 Bp. Pf. 16 Tel: 131-072 **1430. Budapest Pf. 16.**

MIKROSZÁMÍTÓGÉP MAGAZIN

4. ÉVFOLYAM
2. SZÁM
1986. MÁRCIUS

A NEUMANN JÁNOS SZÁMÍTÓGÉP-TUDOMÁNYI TÁRSASÁG LAPJA

Felelős szerkesztő:
Könyves Tóth Pál
Szerkesztőség:
1027 Budapest II., Fő u. 68.
Telefon: 154-250

Kiadja a Delta Szaklapkiadó és Műszaki Szolgáltató Leányvállalat
Felelős kiadó:
Faklen Pál igazgató
1142 Budapest VII., Garay u. 5.
Telefon: 415-583, 215-440

Terjeszti a Magyar Posta
Előfizethető
bármely postahivatalban,
a kézbesítőknél,
a Posta hírlapüzleteiben
és a Hírlapelőfizetési
és Lapellátási Irodánál
(Budapest V., József nádor tér 1.
Postacím: 1900 Budapest)
közvetlenül
vagy postautalványon,
valamint átutalással
a HELIR 215—96162
pénzforgalmi jelzőszámra.
Megjelenik havonta
Példányonkénti ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft

Szikra Lapnyomda
Budapest, (86-3004)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25629
ISSN 0236-6088

Címképünk:
A Számítástechnikai Kutató Intézet és Innovációs Központ PRONET lokális hálózata

Szki

Tartalom

μ '86

„A csillagokat egy virág szépíti meg, amelyet nem lehet látni."
(ANTOINE SAINT-EXUPERY: A KIS HERCEG)

A budapesti Tavaszi Fesztivál az idén új programmal bővül. A Neumann Társaság szervezésében egy héten keresztül kiállítást, bemutatókat, találkozókat szervezünk a számítástechnika iránt érdeklődő közönség számára. A μ '86 a BNV területén, az Utazás '86 kiállítás szomszédjaként reggeltől estig várja a látogatókat.

A μ '86 programjának csak egy részét szerveztük meg előzetesen, lehetőséget adtunk arra is, hogy a közönség kívánságának megfelelően aktuális előadásokat, találkozókat vagy más eseményeket is beiktassunk a programba.

A μFesztivál legnagyobb rendezvénye a „Számítástechnika mindenkiért, a számítástechnika mindenkié" kiállítás, amelyre a hazai gyártókat, fejlesztőket, tudományos intézeteket, szoftverházakat, gazdasági munkaközösségeket, szövetkezeteket és PJT-ket is meghívtuk. Azt kértük, hogy lehetőség szerint az olcsó számítástechnikai berendezéseiket, alkalmazói rendszereiket, programjaikat állítsák ki, hiszen a számítástechnika tömeges elterjedésének még ma is az egyik legnagyobb akadálya a magas ár.

Bemutatjuk a valamikori informatikai múzeum számára összegyűjtött öreg számítógépek megóvott darabjait. Néhány régi eszközt már a korábbi SZMSZM kiállításon is bemutattunk, most többé-kevésbé teljes képet szeretnénk adni a hazai számítástechnika történetéről.

Megragadom az alkalmat, hogy a rendezőség kérését tolmácsoljam. Valószínűleg nagyon sok olyan számítástechnika-történeti emlék van magántulajdonban, amelyekről még nem tudnak. Ezek lehetnek eszközök, alegységek, korabeli programleírások, pl. gépi kódú grafikák, könyvek, intézeti kiadványok, fényképek. Arra kérjük ezeknek az emlékeknek a tulajdonosait, hogy ezeket a becses darabokat ajánlják fel a kiállítás teljessé tétele érdekében; valamennyien gazdagabbak leszünk, ha teljesebb képet kapunk a magyar számítástechnika múltjáról.

A kiállítási terület egy részén folyamatosan találkozót rendezünk a μklubok tagjai részére. Úgy szeretnénk, ha ennek keretében tapasztalatcsere alakulna ki, gazdát cserélnének programok, módszerek, de még az eszközök is. Az NJSZT HCC klubjában nagyon sok amatőr számítógép készül. Szeretnénk ezeket a gépeket egy kiállítás keretében bemutatni, a legjobban sikerült, házi körülmények között épült gépet pedig díjazni.

A fesztivál ideje alatt olcsó alkatrészvásárt is rendezünk, kiárusítjuk a hazai számítástechnikai intézmények, vállalatok elfekvő anyagait.

Számítunk a pedagógusok részvételére, hiszen a következő évek egyik legnagyobb gondja, hogy milyen mértékben lehet a számítógépet, mint a tanítás eszközét használni. Természetesen nemcsak az informatika, hanem és főleg az egyéb tantárgyak tanítására. Bemutató órákat rendezünk, amelyekre a számítástechnika alkalmazásában tapasztalt és tapasztalatlan pedagógusokat is várjuk.

Összegyűjtöttük az elmúlt harminc évben készített számítástechnika tárgyú filmeket, videoanyagokat, amelyeket folyamatosan bemutatunk. Szeretnénk, ha egyszer majd egy informatikai filmarchívum is létrejönne, ahonnan az iskolák, μklubok filmanyagot tudnának kölcsönözni.

Meghívtuk — amikor ezt az írást nyomdába adom, még nem tudom, hogy a meghívást elfogadják-e — az NJSZT-vel együttműködő szervezeteknek, az Österreichische Computer Gesellschaftnak, a görög Computer Society-nek, a francia X2000 klubszervezetnek a képviselőit, hogy vegyenek részt a fesztiválon. Ha eljönnek, akkor találkozókat szervezünk velük, beszélgetéseket, tapasztalatcserét, hogy a μklub mozgalmat a külföldi tapasztalatok alapján még jobban meg tudjuk szervezni.

Végül remélem, hogy Herman H. Goldstine, aki Neumann Jánossal együtt alakította ki a mai modern számítógépek működésének elvét és Heinz Zemanek, aki az első osztrák számítógépet, a Mailflutter-t készítette, elfogadják meghívásunkat és részt vesznek a fesztiválon, találkoznak az érdeklődő szakemberekkel és amatőrökkel.

A fesztivál legfontosabb eseményeiről írtam, az alkalmi rendezvényeket, találkozókat, klubnapokat és más programokat a napi sajtóban közöljük.

Szeretnénk, ha a μ '86, amelyet ezután évenként megrendezünk, rangos és népszerű eseménye lenne a hazai számítástechnikai életnek, amelyhez valamennyi érdeklődő szakember és nem szakember támogatását, ötleteit és részvételét várjuk.

KOVÁCS GYŐZŐ

Gép a gépen I.

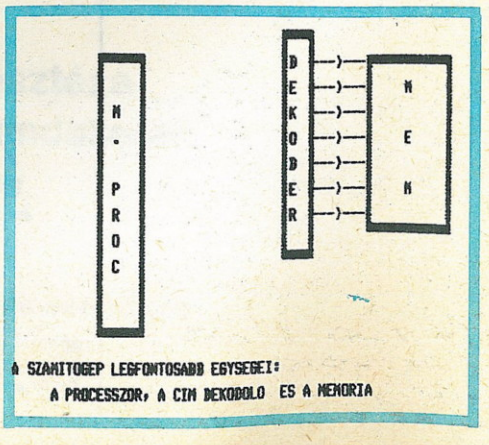
Ebben a sorozatban egy olyan BASIC programot közlünk, amelyet a fonyódi számítástechnikai tanári konferencián mutattak be. A program közlésére az a fogadtatás buzdított fel, amelyet a résztvevők mutattak. A szerző ezt a programját benyújtotta az Ötlet című lap HT-NYERŐ pályázatára, díjat azonban nem nyert vele.

Programunk, mely a HT iskolaszámítógépen futtatható, egy Z80 processzorral felépített mikrogép működését filmszerűen mutatja be. A programban semmilyen trükköt nem alkalmaztunk, tehát sem gépi kódú programot, sem szoftvervédelmet nem építettünk be.

A program moduláris szerkezetű. Olyan, mintha FORTH nyelven íródott volna. Egy-egy részfeladatot előre megoldottunk, hogy később tetszőlegesen sokszor felhasználhassuk. Az egyes blokkok szubrutinként szerepelnek. Egy nagyobb egység felépítéséhez az előbb megírt szubrutinokat használjuk, majd a következő még nagyobb egységeket ezekből rakjuk össze. Az egyes blokkok elején megjegyzésként felsoroljuk azokat a változókat, amelyek szerepelnek bennük. Ezek a szubrutin bemenő adatai.

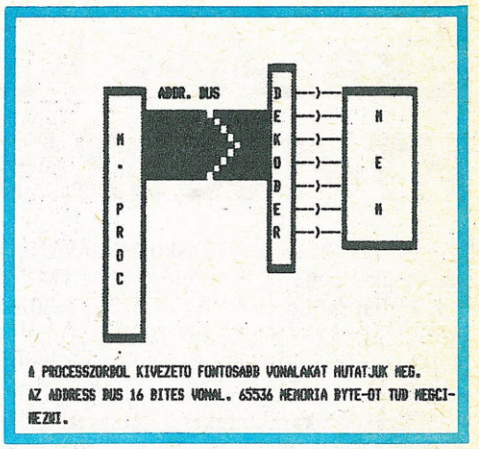
Az *1., 2. programlista* D100-as nyomtatóval kiíratva négy és fél oldalon fér el. A végeredmény azzal kecsegtet, hogy némi szöveges kiegészítéssel könnyen megértethető a processzor cím-, adat- és vezérlővonalainak működése. Ezt kívánja érzékeltetni az a néhány ábra, amelyet most és a későbbiekben közölni fogunk.

Az első néhány sor az inicializálást szolgálja, majd a legkisebb részleteket megva-

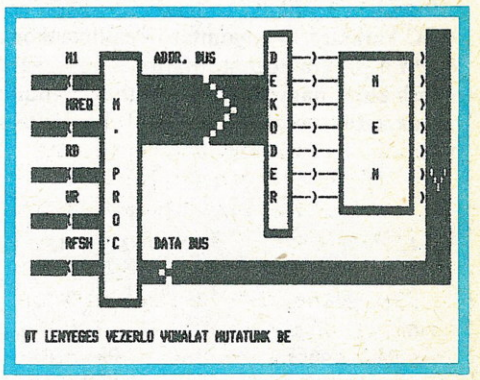


```
10 CLEAR100:DEFINTS,O,M,H,K,I,J
20 DIMA(15):A(0)=0:A(2)=0:A(4)=0:A(7)=128+64+32+16+8:A(8)=15:A(9)=15:A(10)=15:A(
11)=0:A(14)=0:A(15)=0
30 I1$=CHR$(151)+STRING$(2,131):I2$=CHR$(181)+STRING$(2,176)
40 GOTO3000
50 REM VIZSZINTES VONAL SV OV HV KV(1/2/3/4/5/6/7)
60 P=SV*64+OV:ONKVGOTO70,80,90,100,110,120,130
70 PRINTEP,STRING$(HV,131);:RETURN
80 PRINTEP,STRING$(HV,140);:RETURN
90 PRINTEP,STRING$(HV,176);:RETURN
100 PRINTEP,STRING$(HV,143);:RETURN
110 PRINTEP,STRING$(HV,188);:RETURN
120 PRINTEP,STRING$(HV,191);:RETURN
130 PRINTEP,STRING$(HV,32);:RETURN
140 REM*****************************************************
150 REM FUGGOLEGES VONAL SF OF HF KF(1/2)
160 A=128+21*KF:IFTTHENA=32
170 FOR I=SFTOSF+HF:PRINTEI*64+OF,CHR$(A);:NEXT:RETURN
180 REM*****************************************************
190 REM DOBOZ SD OD HD SZ
200 SV=SD:OV=OD+1:HV=SZ-1:PRINTESD*64+OD,CHR$(151);:KV=1:GOSUB60:PRINTESD*64+OD+
SZ,CHR$(171);
210 SV=SD+HD:KV=3:PRINTE64*SV+OD,CHR$(181);:GOSUB60:PRINTE64*SV+OD+SZ,CHR$(186);
220 SF=SD+1:OF=OD:HF=HD-2:KF=1:GOSUB160
230 OF=OD+SZ:KF=2:GOSUB160:RETURN
240 REM*****************************************************
250 REM DOBOZ TORLES SD OD HD SZ
260 FORI=SDTOSD+HD:PRINTE64*I+OD-1,STRING$(SZ+2,32);:NEXT:RETURN
270 REM*****************************************************
280 KF=3:GOSUB160:REM FUGGOLEGES VASTAG VONAL
290 REM*****************************************************
300 REM TELITES TORLES BALRA ES JOBBRA SD OD HD SZ ST G T
310 IFNOT(ST)THENK=1ELSESZ=-SZ:K=2
320 SF=SD:HF=HD:IFNOT(G)THEN360
330 FORJ=ODTOOD+SZSTEPST
340 OF=J:KF=K:GOSUB160:KF=3:GOSUB160
350 NEXT:RETURN
360 FORJ=ODTOOD+SZSTEPST
370 OF=J:KF=3:GOSUB160
380 NEXT:RETURN
3000 REM BEVEZETO
3010 CLS:PRINT"            A SZAMITOGEP MUKODESET MUTATJUK MEG!"
3020 PRINT:PRINT:PRINT"    MEGRAJZOLJUK A FOBB EGYSEGEKET,ES A LENYEGES CIKLUSOK
AT AZ  IDOBELI LEFOLYASUKNAK MEGFELELOEN ABRAZOLJUK."
3030 PRINT:PRINT:PRINT"  AZ ABRAZOLAS FOLYAMATOS LEGYEN, VAGY AZ EGYES RESZEK KO
ZOTT   MEGALLJAK? (F FOLYAMATOS M MEGALL)"
3040 A$=INKEY$
3050 IFA$="F"THENF=-1:GOTO3070ELSEF=0
3060 IFA$()"M"THEN3040
3070 CLS:GOTO1600
```

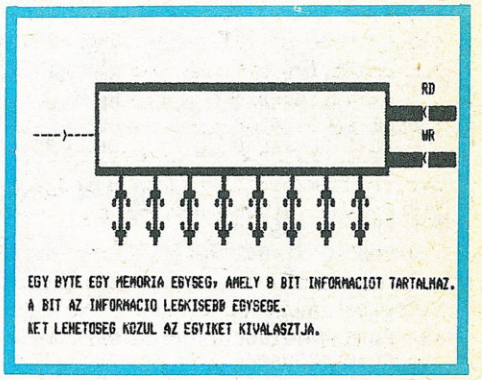
1. lista



A PROCESSZORBOL KIVEZETO FONTOSABB VONALAKAT MUTATJUK MEG. AZ ADDRESS BUS 16 BITES VONAL. 65536 MEMORIA BYTE-OT TUD MEGCIMEZNI.



8T LENYEGES VEZERLO VONALAT MUTATUNK BE



EGY BYTE EGY MEMORIA EGYSEG, AMELY 8 BIT INFORMACIOT TARTALMAZ. A BIT AZ INFORMACIO LEGKISEBB EGYSEGE. KET LEHETOSEG KOZUL AZ EGYIKET KIVALASZTJA.

2. lista

```
5000 REM PROBA1
5010 PRINT"VONALAK":INPUT"MELYIK VONAL 1-7";KV
5020 PRINT"HOVA TEGYEM":INPUT"FUGGOLEGES";SV
5030 INPUT "VIZSZINTES";OV
5040 INPUT"A VONAL HOSSZA";HV
5050 CLS:GOSUB50
6000 REM KULONBOZO VASTAGSAGU ES HELYZETU VIZSZINTES VONALAKAT KAPUNK
6010 REM KV=1-7 SV=10 OV=10 HV=15 ERTEKEKKEL CELSZERU KIPROBALNI

5000 REM PROBA2
5010 PRINT"VONALAK":INPUT" VALASZTAS 1/2";KF
5020 PRINT"HOVA TEGYEM":INPUT"FUGGOLEGES";SF
5030 INPUT "VIZSZINTES";OF
5040 INPUT"A VONAL HOSSZA";HF
5050 CLS:GOSUB140
6000 REM KET FUGGOLEGES VONALAT KAPUNK MIDKET ESETBEN
6010 REM KF=1-2 SF=3 OF=10 HF=10 ERTEKEKKEL CELSZERU KIPROBALNI

5000 REM PROBA3
5010 PRINT"DOBOZOK"
5020 PRINT"HOVA TEGYEM":INPUT"FUGGOLEGES";SD
5030 INPUT "VIZSZINTES";OD
5040 INPUT"A DOBOZ HOSSZA";HD
5050 INPUT"SZELESSEGE";SZ
5060 CLS:GOSUB190
6000 REM EGY TEGLALAPOT KAPUNK A KEPERNYON
6010 REM SD=3 OD=10 HD=8 SZ=4  ERTEKEKKEL CELSZERU KIPROBALNI

5000 REM PROBA4
5010 PRINT"TELITES TORLES ..."
5020 PRINT"HOVA TEGYEM":INPUT"FUGGOLEGES";SD
5030 INPUT "VIZSZINTES";OD
5040 INPUT"A DOBOZ HOSSZA";HD
5050 INPUT"SZELESSEGE";SZ
5052 ST=1:G=1:T=0
5060 CLS:GOSUB300
5062 ST=1:G=1:T=-1:GOSUB300
6000 REM EGY SATIROZOTT TEGLALAPOT KAPUNK, BALROL JOBBRA TELITODIK, MAJD FOGY
6010 REM SD=3 OD=10 HD=8 SZ=4  ERTEKEKKEL CELSZERU KIPROBALNI
7000 REM ST=-1-EL, ES 5062-BEN SZ=-SZ BEIRASAVAL JOBBROL BALRA TELIT ES TOROL
```

lósító szubrutinok következnek. Ezek csak vonalat húznak. Később a vonalakból doboz lesz, azaz egy téglalap, amelyet gépünk a képernyőre rajzol. Ezek a dobozok a számítógép egy-egy funkcionális egységét jelentik.

Tudjuk, hogy a kész programlista mindenkit arra csábít, hogy elejétől a végéig begépelje, és csak azután kezdje a hibakeresést. Így azonban egy idegen program elindítása nagyon nehéz feladat. Mindjárt szeretnénk elejét venni az ilyesfajta kínlódásoknak, úgyhogy az első részletekhez közlünk próbaprogramokat is. A 40 GOTO 3000 helyett 40 GOTO 5000-et írva, a tesztelés egyszerű.

NYIRATI LÁSZLÓ
Ybl Miklós Szakközépiskola
8000 Székesfehérvár,
Szabadságharcos út 57.

A szerző felajánlja, hogy iskoláknak saját kazettájukra felveszi, és némi kezelési költség fejében visszaküldi a programot. (A szerk.)

Ékezetes karakterek nyomtatása HT-1080Z/64-en

Az első ékezetes HT iskolaszámítógép még csak a rövid ékezetes betűket ismerte, ezért elég volt a TMT nyomtatón levő karakterek kódját módosítani. Az új, HT—1080Z/64 számítógépek már a hosszú ékezetes betűket is ismerik, amelyek kinyomtatásához fel kell használni a TMT nyomtatók karakterdefiniálási lehetőségét is (HDC karakterek).

Ezeket a karaktereket a nyomtató csak 10 kar/inch nyomtatási sűrűséggel tudja kiírni. Ha valamely nyomtatott sorban megváltoztatjuk a karakter szélességét, akkor ez a sor ne tartalmazzon HDC karaktert! A nyomtató beállításakor a helyes működés érdekében a programozható paraméterek gyári alapértékei közül csak a formátum nagyságát és az áthúzott nulla üzemmódot változtathatjuk a kapcsolók segítségével. ESCAPE sorozatokat ne használjunk!

```
                    ;   KÉSZITETTE: KISPAL ISTVAN
                    ;   DUNAÚJVÁROS
                    ;   BÁNKI DONÁT SZAKKÖZÉPISKOLA
                        ORG  4026H  ;PRINTER DCB
                    ;   LOAD 4026H   MODOSITASA
4026 8AFE               DB   138,254
                        ORG  41E2H  ;ÖNINDITAS
                    ;   LOAD 41E2H
41E2 E9                 JP   (HL)
                        ORG  64876  ;ADATOK
                    ;   LOAD 64876
FD6C 1B5B3164           DB   27,91,49,100;HDC KI
FD70 1B504440           DB   27,80,68,64;$
FD74 4A414040           DB   74,65,64,64
FD78 5F434040           DB   95,67,64,64
FD7C 4A415040           DB   74,65,80,64
FD80 40404040           DB   64,64,64,64
FD84 40401B5C           DB   64,64,27,92
FD88 1B505E40           DB   27,80,94,64;@
FD8C 41414042           DB   65,65,64,66
FD90 49405442           DB   73,64,84,66
FD94 41405E42           DB   65,64,94,66
FD98 40404040           DB   64,64,64,64
FD9C 40401B5C           DB   64,64,27,92
FDA0 1B504040           DB   27,80,64,64;Í
FDA4 41425F43           DB   65,66,95,67
FDA8 41424040           DB   65,66,64,64
FDAC 42404140           DB   66,64,65,64
FDB0 40404040           DB   64,64,64,64
FDB4 40401B5C           DB   64,64,27,92
FDB8 1B505C41           DB   27,80,92,65;Ö
FDBC 42424040           DB   66,66,64,64
FDC0 47424040           DB   71,66,64,64
FDC4 42425C41           DB   66,66,92,65
FDC8 40404040           DB   64,64,64,64
FDCC 40401B5C           DB   64,64,27,92
FDD0 1B505C41           DB   27,80,92,65;Ő
FDD4 42424642           DB   66,66,70,66
FDD8 42424342           DB   66,66,67,66
FDDC 46424242           DB   70,66,66,66
FDE0 5D414040           DB   93,65,64,64
FDE4 40401B5C           DB   64,64,27,92
FDE8 1B505C41           DB   27,80,92,65;Ö
FDEC 40424242           DB   64,66,66,66
FDF0 41424042           DB   65,66,64,66
FDF4 42424142           DB   66,66,65,66
FDF8 5C414040           DB   92,65,64,64
FDFC 40401B5C           DB   64,64,27,92
FE00 1B505841           DB   27,80,88,65;Ő
FE04 40424240           DB   64,66,66,64
FE08 41424040           DB   65,66,64,64
FE0C 42425941           DB   66,66,89,65
FE10 40404040           DB   64,64,64,64
FE14 40401B5C           DB   64,64,27,92
FE18 1B505841           DB   27,80,88,65;Ő
FE1C 44424240           DB   68,66,66,64
FE20 45424040           DB   69,66,64,64
FE24 46425941           DB   70,66,89,65
FE28 40404040           DB   64,64,64,64
FE2C 40401B5C           DB   64,64,27,92
FE30 1B505C41           DB   27,80,92,65;Ö
FE34 40424040           DB   64,66,64,64
FE38 42424140           DB   66,66,65,64
FE3C 40425C41           DB   64,66,92,65
FE40 40404040           DB   64,64,64,64
FE44 40401B5C           DB   64,64,27,92
FE48 1B505841           DB   27,80,88,65;Ő
FE4C 40424040           DB   64,66,64,64
FE50 42424140           DB   66,66,65,64
FE54 40425841           DB   64,66,88,65
FE58 40404040           DB   64,64,64,64
FE5C 40401B5C           DB   64,64,27,92
FE60 1B504040           DB   27,80,64,64;í
FE64 40404040           DB   64,64,64,64
FE68 5A434140           DB   90,67,65,64
FE6C 40404040           DB   64,64,64,64
FE70 40404040           DB   64,64,64,64
FE74 40401B5C           DB   64,64,27,92
                        ORG  65150
                    ;   LOAD 65150
FE7E 3EC9               LD   A,201
FE80 32E241             LD   (41E2H),A
FE83 C36600             JP   0066H
FE86 00500000           DB   00,80,00,00;' MS AS DSZ
FE8A E5                 PUSH HL      ;FŐPROGRAM
FE8B C5                 PUSH BC
FE8C 79                 LD   A,C
FE8D FE0A               CP   0AH     ;LF
FE8F 200C               JR   NZ,TOVA0
FE91 AF             VEZER: XOR  A
FE92 3288FE             LD   (65160),A
FE95 3289FE         UGR0: LD   (65161),A
FE98 C1                 POP  BC
FE99 E1                 POP  HL
FE9A C38D05             JP   58DH
FE9D FE0C           TOVA0: CP   0CH     ;FF
FE9F 28F0               JR   Z,VEZER
FEA1 FE0D               CP   0DH     ;CR
FEA3 2811               JR   Z,VEZKAR
FEA5 FE0E               CP   0EH     ;DSZ
FEA7 201E               JR   NZ,TOVA1
FEA9 3A89FE             LD   A,(65161)
FEAC B7                 OR   A
FEAD 2004               JR   NZ,UGR1
FEAF 3E0E               LD   A,0EH
FEB1 18E2               JR   UGR0
FEB3 AF             UGR1: XOR  A
FEB4 18DF               JR   UGR0
FEB6 AF           VEZKAR: XOR  A
FEB7 3288FE             LD   (65160),A
FEBA 3289FE             LD   (65161),A
FEBD C1                 POP  BC
FEBE E1                 POP  HL
FEBF CD8D05             CALL 58DH
FEC2 3E0A               LD   A,0AH
FEC4 C38E05             JP   58EH
FEC7 FE40           TOVA1: CP   64      ;@
FEC9 2008               JR   NZ,TOVA2
FECB 3286FE             LD   (65158),A
FECE 2188FD             LD   HL,64904
FED1 1876               JR   IRAT
FED3 FE24           TOVA2: CP   36      ;$
FED5 2010               JR   NZ,TOVA3
FED7 3A86FE             LD   A,(65158)
FEDA FE60               CP   96
FEDC 79                 LD   A,C
FEDD 2808               JR   Z,TOVA3
FEDF 3286FE             LD   (65158),A
FEE2 2170FD             LD   HL,64880
FEE5 1862               JR   IRAT
FEE7 3A86FE         TOVA3: LD   A,(65158)
FEEA FE60               CP   96
FEEC 79                 LD   A,C
FEED 3286FE             LD   (65158),A
FEF0 C2DEFF             JP   NZ,VÉGE
FEF3 FE60               CP   96
FEF5 2003               JR   NZ,TOVA4
```

```
138 FEF7 C1                POP  BC
139 FEF8 E1                POP  HL
140 FEF9 C9                RET
141 FEFA FE34     TOVA4:   CP   52      ;ő
142 FEFC 2005              JR   NZ,TOVA5
143 FEFE 21D0FD            LD   HL,64976
144 FF01 1846              JR   IRAT
145 FF03 FE36     TOVA5:   CP   54      ;ö
146 FF05 2005              JR   NZ,TOVA5A
147 FF07 21E8FD            LD   HL,65000
148 FF0A 183D              JR   IRAT
149 FF0C FE26     TOVA5A:  CP   38      ;ű
150 FF0E 2005              JR   NZ,TOVA5B
151 FF10 2100FE            LD   HL,65024
152 FF13 1834              JR   IRAT
153 FF15 FE24     TOVA5B:  CP   36      ;ő
154 FF17 2005              JR   NZ,TOVA6
155 FF19 2118FE            LD   HL,65048
156 FF1C 1828              JR   IRAT
157 FF1E FE55     TOVA6:   CP   85      ;ú
158 FF20 2005              JR   NZ,TOVA7
159 FF22 2130FE            LD   HL,65072
160 FF25 1822              JR   IRAT
161 FF27 FE75     TOVA7:   CP   117     ;ú
162 FF29 2005              JR   NZ,TOVA8
163 FF2B 2148FE            LD   HL,65096
164 FF2E 1819              JR   IRAT
165 FF30 FE69     TOVA8:   CP   105     ;í
166 FF32 2005              JR   NZ,TOVA9
167 FF34 2160FE            LD   HL,65120
168 FF37 1810              JR   IRAT
169 FF39 FE4F     TOVA9:   CP   79      ;Ő
170 FF3B 2005              JR   NZ,TOVA10
171 FF3D 21B8FD            LD   HL,64952
172 FF40 1807              JR   IRAT
173 FF42 FE49     TOVA10:  CP   73      ;Í
174 FF44 2060              JR   NZ,TOVA11
175 FF46 21A0FD            LD   HL,64928
176 FF49 E5       IRAT:    PUSH HL
177 FF4A CD78FF            CALL SORV?
178 FF4D DC88FF            CALL C,CR
179 FF50 E1                POP  HL
180 FF51 CD94FF            CALL ÍR
181 FF54 3E0D              LD   A,0DH  ;CR
182 FF56 CD8E05            CALL 058EH
183 FF59 3A89FE            LD   A,(65161)
184 FF5C FE0E              CP   0EH
185 FF5E 2004              JR   NZ,UGR2
186 FF60 AF                XOR  A
187 FF61 3289FE            LD   (65161),A
188 FF64 3A88FE   UGR2:    LD   A,(65160)
189 FF67 47                LD   B,A
190 FF68 2187FE            LD   HL,65159
191 FF6B BE                CP   (HL)
192 FF6C 3007              JR   NC,UGR4
193 FF6E 3E20     LÉP:     LD   A,20H
194 FF70 CD8E05            CALL 058EH
195 FF73 10F9              DJNZ LÉP
196 FF75 C1       UGR4:    POP  BC
197 FF76 E1                POP  HL
198 FF77 C9                RET
199 FF78 3A89FE   SORV?:   LD   A,(65161)
200 FF7B B7                OR   A
201 FF7C 2188FE            LD   HL,65160
202 FF7F 2801              JR   Z,UGR3
203 FF81 34                INC  (HL)
204 FF82 34       UGR3:    INC  (HL)
205 FF83 3A87FE            LD   A,(65159)
206 FF86 BE                CP   (HL)
207 FF87 C9                RET
208 FF88 AF       CR:      XOR  A
209 FF89 3288FE            LD   (65160),A
210 FF8C 3289FE            LD   (65161),A
211 FF8F 2188FE            LD   HL,65160
212 FF92 34                INC  (HL)
213 FF93 C9                RET
214 FF94 0618     ÍR:      LD   B,24   ;ALAKJA
215 FF96 CD9EFF            CALL KIÍR
216 FF99 0604              LD   B,4    ;KIRAK
217 FF9B 216CFD            LD   HL,64876
218 FF9E 7E       KIÍR:    LD   A,(HL)
219 FF9F CD8E05            CALL 58EH
220 FFA2 23                INC  HL
221 FFA3 10F9              DJNZ KIÍR
222 FFA5 C9                RET
223 FFA6 FE41     TOVA11:  CP   65      ;Á
224 FFA8 2002              JR   NZ,TOVA12
225 FFAA 3E40              LD   A,64
226 FFAC FE45     TOVA12:  CP   69      ;É
227 FFAE 2002              JR   NZ,TOVA13
228 FFB0 3E5B              LD   A,91
229 FFB2 FE33     TOVA13:  CP   51      ;Ű
230 FFB4 2002              JR   NZ,TOVA14
231 FFB6 3E5C              LD   A,92
232 FFB8 FE35     TOVA14:  CP   53      ;Ú
233 FFBA 2002              JR   NZ,TOVA15
234 FFBC 3E5D              LD   A,93
235 FFBE FE25     TOVA15:  CP   37      ;Ü
236 FFC0 2002              JR   NZ,TOVA16
237 FFC2 3E7D              LD   A,125
238 FFC4 FE23     TOVA16:  CP   35      ;ő
239 FFC6 2002              JR   NZ,TOVA17
240 FFC8 3E7C              LD   A,124
241 FFCA FE61     TOVA17:  CP   97      ;á
242 FFCC 2002              JR   NZ,TOVA18
243 FFCE 3E60              LD   A,96
244 FFD0 FE65     TOVA18:  CP   101     ;é
245 FFD2 2002              JR   NZ,TOVA19
246 FFD4 3E7B              LD   A,123
247 FFD6 FE6F     TOVA19:  CP   111     ;ó
248 FFD8 2008              JR   NZ,TOVA20
249 FFDA 3E7E              LD   A,126
250 FFDC 1804              JR   TOVA20
251 FFDE FE60     VÉGE:    CP   96
252 FFE0 2811              JR   Z,VISSZA
253 FFE2 4F       TOVA20:  LD   C,A
254 FFE3 CD78FF            CALL SORV?
255 FFE6 3008              JR   NC,SORBAN
256 FFE8 AF                XOR  A
257 FFE9 3289FE            LD   (65161),A
258 FFEC 3C                INC  A
259 FFED 3288FE            LD   (65160),A
260 FFF0 CD8D05   SORBAN:  CALL 058DH
261 FFF3 C1       VISSZA:  POP  BC
262 FFF4 E1                POP  HL
263 FFF5 C9                RET
264                        END
```

```
A NYOMTATÓ KÜLÖNBÖZŐ TÍPUSVÁLTOZATAI CS
BEN TÉRNEK EL EGYMÁSTÓL. AZ EGYES VÁLTOZATOK MEGK
Ő SZÁMJEGYÉVEL TÖRTÉNIK, ÁLTALÁNOS ESETBEN EZT JE
ülönböző típusváltozatai csak a papír kezelési le
l. Az egyes változatok megkülönböztetése a típuss
k, általános esetben ezt jelöljük 'X' jellel
A NYOMTATÓ KÜLÖNBÖZŐ TÍPUSVÁLTOZATAI C
BEN TÉRNEK EL EGYMÁSTÓL. AZ EGYES VÁLTOZATOK MEGK
Ő SZÁMJEGYÉVEL TÖRTÉNIK, ÁLTALÁNOS ESETBEN EZT JE
ülönböző típusváltozatai csak a papír kezelési lel
l. Az egyes változatok megkülönböztetése a típus
```

```
éHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóó
ő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@
DFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGH
éHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóó
ő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N
DFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGH
éHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóó
ő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@
DFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGH
éHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóó
ő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@SDFGHHéHóóGő$N@
```

A forrásprogramban szereplő LOAD utasítások előtt a ; jel csak a program kinyomtatása miatt szerepel, ezeket természetesen nem kell begépelni. Az assemblálást most is kazettára végeztessük, belépési címként a 65150 (FE7EH) értéket adjuk meg. A gépi kódú program használatánál a számítógép bekapcsolásakor védenünk kell a memóriát a 64 870-es címtől kezdve, és csak ezután töltsük be a programot, amely önindulós, tehát a betöltés után a számítógép visszatér a BASIC-hez.

Ettől kezdve a számítógép kész az összes ékezetes betű nyomtatására, sőt a @ és $ jeleket is képes kinyomtatni, pedig a magyar jelkészletnél eddig ezek helyett az A' és Ft jel jelent meg. A program hatására természetesen a nyomtató a gyári alapértékkel ellentétben CR=LF üzemmódban fog működni.

KISPÁL ISTVÁN

Gépi kódú rajzolás

Az iskolaszámítógépnél két lehetőség van grafika készítésére: a videoterület (3CØØH—3FFFH) megfelelő címére töltjük valamely grafikus karakter kódját, vagy ROM-rutin segítségével az x és y koordinátákat megadva készítjük el a grafikát. Most foglalkozzunk a második lehetőséggel.

A Ø15ØH címtől a ROM grafikai rutinjai (SET, RESET, POINT) kezdődnek, de ehhez a vermet megfelelően fel kell tölteni. Ezt a rutint nem CALL-lal kell meghívni, hanem JP-vel a Ø15ØH címre kell ugorni. A verem feltöltését az *1. program* végzi.

Az 5. sorban az akkumulátorba háromféle értéket tölthetünk: ØØH — POINT; Ø1H — RESET; 8ØH — SET.

Ezt a receptet szigorúan be kell tartani, egyébként hibaüzenetet kapunk. A használati leírásból látszik, hogy a rutin a szükséges adatokat a veremből olvassa ki, ezért az adatok vermelésének sorrendjére ügyelni kell. A visszatérési címre a bezáró zárójel elhelyezésével a rutin szintaktikai hibaellenőrzését védhetjük ki. Ha a képernyő méreténél nagyobb x és y koordinátákat adunk meg, akkor a képernyő hengerpalástszerűen működik, mivel túlcsordulás történik.

A *2. program* egy egyenest rajzol a képernyőre. A *3. program* négyzetet rajzol, és az egyenesrajzoló szubrutin használatát illusztrálja. A program érdekessége, hogy önmagát „átírja".

A példák megértése után érdemes megpróbálkozni hasonló feladatokkal, mint például képernyő-satírozás vagy négyzetháló-készítés.

SIEGLER GÁBOR

```
 1                          ORG  7000H
 2                          LOAD 7000H
 3                  :PONT A KEPERNYON
 4 7000 210F70              LD   HL,CIM
 5 7003 E5                  PUSH HL       : VISSZATERESI CIM A VEREMBEN
 6 7004 3E80                LD   A,80H    : MOST SET LESZ
 7 7006 F5                  PUSH AF       : AKKUMLATOR A VEREMBE KERUL
 8 7007 3E0D                LD   A,13     : X KOORDINATA = 13
 9 7009 F5                  PUSH AF       : VEREMBE KERUL
10 700A 3E14                LD   A,20     : Y KOORDINATA = 20
11 700C C35001              JP   0150H    : ROM RUTIN MEGHIVASA
12 700F 29          CIM:    DB   ')'
13 7010 CD4900              CALL 49H      : VARAKOZAS
14                          END
```

1. program

```
 1                          ORG  7000H
 2                          LOAD 7000H
 3 7000 CDC901              CALL 1C9H     : KEPERNYO TORLES
 4 7003 0628                LD   B,40     : AZ EGYENES HOSSZA
 5 7005 C5           CIKL:  PUSH BC       :SZAMLALO ELMENTESE
 6 7006 211270              LD   HL,CIM
 7 7009 E5                  PUSH HL       : VISSZATERESI CIM A VEREMBEN
 8 700A 3E80                LD   A,80H    : RAJZ
 9 700C F5                  PUSH AF       : AKKUMLATOR A VEREMBE KERUL
10 700D 78                  LD   A,B      : B REGISZTERT A-BA TOLTJUK
11 700E F5                  PUSH AF       : X=Y EGYENEST RAJZOLJUK
12 700F C35001              JP   0150H    : ROM RUTIN MEGHIVASA
13 7012 29          CIM:    DB   ')'
14 7013 C1                  POP  BC
15 7014 10EF                DJNZ CIKL     : ISMETLES AMIG B=0 LESZ
16 7016 CD4900              CALL 49H      : VARAKOZAS
17                          END
```

2. program

3. program

```
 1                  :NEGYZET RAJZOLASA ROM RUTINNAL
 2                          ORG  7000H
 3                          LOAD 7000H
 4 7000 CDC901              CALL 1C9H     : KEPERNYO TORLES
 5 7003 DD215070            LD   IX,ADAT  : IX AZ ADATOK CIMERE MUTAT
 6 7007 DD4600              LD   B,(IX+0) : B-BE KERUL AZ ELSO ADAT
 7 700A DD4E01              LD   C,(IX+1) : C-BE TOLTJUK A MASODIKAT
 8 700D CD3970              CALL LOOP     : EGYENES RAJZOLO RUTIN
 9 7010 DD4600              LD   B,(IX+0) : UJ ADATOK
10 7013 DD4E00              LD   C,(IX+0)
11 7016 CD3970              CALL LOOP     : VIZSZINTES VONALAK KESZEN
12 7019 214970              LD   HL,ZIZI  : A 34. SORT LD A,B-VE IRJA
13 701C 35                  DEC  (HL)     : 79 KOD 78-CA
14 701D 214B70              LD   HL,ZAZA  : A 36. SORT LD A,C-VE IRJA
15 7020 34                  INC  (HL)     : 78 KOD 79-CE
16 7021 DD4600              LD   B,(IX+0) : A PROGRAM ONMAGAT IRJA AT
17 7024 DD4E01              LD   C,(IX+1)
18 7027 CD3970              CALL LOOP     : ELSO FUGGOLEGES VONAL
19 702A DD4600              LD   B,(IX+0)
20 702D DD4E00              LD   C,(IX+0)
21 7030 CD3970              CALL LOOP     : MASODIK FUGGOLEGES VONAL
22 7033 CD4900              CALL 49H      : VARAKOZAS
23 7036 C35370              JP   VEGE
24 7039 C5          LOOP:   PUSH BC      :VONAL RAJZOLO RUTIN
25 703A CD4170              CALL CIKL
26 703D C1                  POP  BC       : SZAMLALO
27 703E 10F9                DJNZ LOOP     : ISMETLES B=0-IG
28 7040 C9                  RET
29 7041 00          CIKL:   NOP           : EGY PONT KITEVES
30 7042 214F70              LD   HL,CIM
31 7045 E5                  PUSH HL       : VISSZATERESI CIM
32 7046 3E80                LD   A,80H    : SET
33 7048 F5                  PUSH AF
34 7049 79          ZIZI:   LD   A,C
35 704A F5                  PUSH AF
36 704B 78          ZAZA:   LD   A,B
37 704C C35001              JP   0150H    : ROM RUTIN MEGHIVASA
38 704F 29          CIM:    DB   ')'
39 7050 2800        ADAT:   DB   40,0     : ADATOK ITT CSERELHETOK
40 7052 C9                  RET
41                  VEGE:   END
```

Kell-e nekünk számítógép-múzeum?

A címet a μM 1984/5. számában Csontos Tibor tollából megjelent kitűnő cikktől kölcsönöztük, remélve, hogy a szerző nem haragszik meg érte.

A kérdésre habozás nélkül pozitív választ kellene adnunk, hiszen köztudott, hogy a számítástechnikának az utóbbi két évtizedben mutatott fejlődése olyan viharos sebességű, amilyet eddig nem ismert a technikatörténet.

A húsz évvel ezelőtt hatalmas, klimatizált teremben működő, több tonna súlyú, csaknem 100 kW elektromos energiát igénylő számítógépek teljesítménye eltörpül egy mai közepes kiépítettségű mikrogép teljesítménye mellett. Ha általában a technikatörténetben mérföldkövekről beszélünk, akkor a számítástechnika fejlődésében a „százméterenkénti jelzőkövek" sem lennének elegendőek a kimagasló újdonságok megjelölésére.

A „korszakalkotó" jelző alkalmazásával is egyre óvatosabban kell bánni a számítástechnikában, hiszen a legkülönbözőbb szempontok szerint megkülönböztetett korszakok soha nem látott mértékben rövidülnek, olyannyira, hogy a „korszak" megjelölés egyre inkább értelmét veszti.

Ebben a szinte fantasztikusnak tűnő fejlődésben a hazai számítástechnikai szakemberek elsősorban az új és még újabb gépek megszerzésére és alkalmazására összpontosítottak. A régi gépek olyan felhasználókhoz kerültek, akiknek azok is újaknak számítottak. Később ezeken a helyeken is érvényesült a még újabb iránti általános törekvés, és a láncolat végén a régi gépek kényelmetlen nyűggé váltak.

Szintén érthető módon kicsit késve, de még szerencsére idejében támadt Kovács Győzőnek, az NJSZT főtitkárának az az ötlete, hogy gyűjtsük össze a hazai számítástechnika fejlődésének tárgyi emlékeit, létesítsünk számítástechnikai múzeumot. Az ötlet megvalósításának első lépése — a régi gépek összegyűjtésének és tárolásának lehetősége — 1978-ban valóra vált. A Nagyalföldi Kőolaj- és Földgáztermelő Vállalat algyői telepén két nagyméretű barakképületet kapott bérleti szerződés keretében az NJSZT.

Megállapodás jött létre az Országos Műszaki Múzeum (OMM) és az NJSZT között abban, hogy a társaság felkutatja és összegyűjti a hazai számítástechnika fejlődését demonstráló, technikatörténeti értékű hardvereszközöket, és ezeket átadja az OMM-nek, ahol a megfelelő dokumentálásukról és nyilvántartásukról gondoskodnak.

Az algyői raktár 1978-ban működni kezdett. Itt található az első hazai előállítású, elektroncsöves számítógép, a M—3 két dobtárolója. Sajnos maga a gép kiselejtezés után veszendőbe ment. Itt van az elegáns ICT 1905 teljes konfigurációja; a csodálnivalóan elnyűhetetlen MINSZK 22 és korszerűbb „öccse", a MINSZK 32; a raktár „ékessége", a jereváni RAZDAN 3; a maga idejében csúcsteljesítményű IBM 360/20; a nagyon egyéni megoldásokat mutató ELLIOT 803; a komor Siemens 4004-as rendszer egyes darabjai; a hazai gyártásúak közül a TPA 1001 és az egyik első R10.

A felsorolás nem teljes és nem is a gyártási évszámok sorrendjét tükrözi, de talán mégis alkalmas annak érzékeltetésére, hogy a számítástechnika-történeti értékek megmentésére irányuló törekvéseink hoztak bizonyos sikereket. Eddigi munkánk azonban nem volt problémamentes. 1982-ben az egyik barakkot az olajipar fejlesztési céljaira visszakellett adnunk a vállalatnak. Így az addig jól áttekinthetően, szinte kiállításszerűen tárolt eszközöket nyomasztó módon össze kellett zsúfolni. E kényszerű költözés utáni helyzet tette világossá azt, hogy gyűjtőtevékenységünk módszerén változtatni kell.

Eddig teljes konfigurációkat vettünk át a selejtező felhasználóktól. Így sok olyan, főként perifériális eszköz került a raktárba, amelyek technikatörténeti értéke nem indokolja megőrzésüket, és jelentős helyet foglalnak el. Ezért az utóbbi években felkutatott vagy felajánlott konfigurációkból csak az illető gépre, gyártóra legjellemzőbb részeket, egységeket veszünk át. Valamikor bizonyára sajnálni fogjuk – vagy unokáink sajnálni fogják – e korlátozást, amelyet a mai objektív kényszer, a helyszűke hozott létre. Az algyői raktár pillanatnyi feltöltöttsége 80 százalékos. Hogyan lehetne bővíteni, újabb számítástechnika-történeti értékek befogadását lehetővé tenni?

E kérdés szorosan összefügg a gyűjtemény alapvető és legnagyobb problémájával, nevezetesen azzal, hogy a raktározott eszközök a külvilágtól elzártak, mint technikatörténeti értékek és érdekességek eddig nem válhattak az érdeklődők számára hozzáférhetővé, megtekinthetővé. Algyőn csak egy — a túlcsordulás határán lévő — raktár van, de nincs kiállítóterem. Az Országos Műszaki Múzeum maga is rendkívüli helyhiánnyal küzd. Az állandó kiállítás létesítésére szolgáló termek, épület megszerzésére irányuló eddigi kísérleteink nem jártak eredménnyel. A reményt természetesen nem adtuk fel, de addig is, amíg véglegesnek tekinthető megoldást nem találunk, valamit tenni kellene.

Az első, „A számítástechnika mindenkié, a számítástechnika mindenkiért" című rendezvényen kiállítottuk a raktár néhány darabját, nyugodtan mondhatjuk: igen nagy sikerrel. Az itt tapasztaltak csak megerősítették azt a meggyőződésünket, hogy a számítástechnika-történeti emlékek nyilvános bemutatása nemcsak az idősebb, hanem a fiatalabb szakemberek, sőt az örvendetesen szaporodó számú, a számítástechnikát kedvelő és tanuló gyerekek körében is nagy érdeklődést váltott ki. Hogyan lehetne e problémát megoldani? Ötleteinket azzal a szándékkal tárjuk a szakma nagy nyilvánossága elé, hogy a kollektív bölcsesség segítségét kérjük.

A számítástechnikának ma már sok vidéki „fellegvára" van. Ezeken a helyeken viszonylag kis alapterületen, nagyon szép anyagból szerkesztett, számítástechnika-történeti kamarakiállításokat lehetne létrehozni, amelyeken a kiállított tárgyakat időnként cserélni lehetne. Kultúrházakra és művelődési központokra gondolunk elsősorban, de szóba jöhetnének a megyeszékhelyeken a technika házai is.

A számítástechnikai fő profilú vagy a számítástechnikát széles körben alkalmazó intézmények, vállalatok székházának például az előterében bizonyára vonzó látványt nyújtana egy esetleg lassan forgó alapra helyezett régi központi egység vagy vezérlőpult.

Mindkét forma kettős célt szolgálna: egyrészt – s ez a fontosabb – nagy nyilvánosságot adna a raktárba összegyűjtött technikatörténeti értékeknek, másrészt tehermentesítené, újabb eszközök befogadására tenné képessé az algyői raktárat. Bizonyára elképzelhetők további megoldások, további ötletek is – ezeket várjuk e cikk olvasóitól.

Az említett nehézségek ellenére itt, e helyen is kérjük a számítógép-felhasználókat, hogy mielőtt selejtezésre kerülő gépeiket megsemmisítenék, ajánlják fel az NJSZT titkárságán. Nagyon szeretnénk elérni, hogy a számítástechnika-történeti értékek mentésére irányuló munkánk folyamatossá váljon, és gyűjteményünk közelítőleg hűen kövesse a számítástechnika hazánkban végbement és végbemenő fejlődését.

DR. MUSZKA DÁNIEL

Alapozás XV.

Az információtárolás jelenségének (és műveletének) sikerült egy jól működő matematikai modelljét megalkotni. A modell kétségtelenül jól működik — önmagában. A valósághoz való hasonlóságának kérdését azonban további alapos vizsgálat hivatott eldönteni. Ehhez szükség van hatékony működésmód-leíró eszközökre. Ezekkel foglalkozunk a következőkben.

Az állapottér

Rendszerek viselkedését állapotjellemzőik alakulási folyamatával is leírhatjuk. Egy digitális berendezésben levő pont állapotát legegyszerűbben a {0;1} (kételemű) halmaz elemeivel írhatjuk le, jelölhetjük.

Ez a leírás nyilvánvalóan egyszerű, de az is nyilvánvaló, hogy elnagyolt, hiszen nem kezel csak két állapotot, a többi olyan számára, mintha nem is lenne. (Az elhanyagolások egy-egy modell használhatóságát nem biztos, hogy rontják. Modellünk is, durva elhanyagolásai ellenére, egy — gyakorlatilag fontos — körben jó, sőt legjobb modell. A túl keveset elhanyagoló modellek szinte sohasem lehetnek jók, legfeljebb csak nagyon pontosak. E nagyon pontos modelleket aztán legtöbbször nagyon nehéz kezelni. A jó modell a pontosság, a kezelhetőség és más igények együttesét kielégítő kompromisszumként szokott létrejönni.) Egyszerű modellünkben a pont állapotát leíró (megadó) jellemző teljes mozgástere, teljes élettere e kételemű halmaz. Mivel állapotjellemző mozgásteréről van szó, ezt a halmazt *állapottérnek* nevezzük *(1. ábra)*.

Ha pontosabb leírást használnánk, és az állapotjellemző összes 0 és 1 közötti értékét is figyelembe vennénk, akkor az állapottér a teljes [0;1] zárt intervallum lenne *(2. ábra)*.

Az állapottér pontjainak a rendszer (lehetséges) állapotai felelnek meg. És a rendszer (lehetséges) állapotainak az állapottér pontjait feleltetik meg. E megfeleltetéstől meg szokták követelni a kölcsönös egyértelműséget. Ez azonban sokszor ésszerűtlen, és csak felesleges munkát igényel, esetleg még a felhasználhatóságot is nehezíti. Ezért az állapottértől a következőkben a kölcsönösen egyértelmű megfelelést nem fogjuk megkövetelni. (Ezt még az is menthetővé teszi, hogy az állapotterek vetületei „alterei", és altereinek vetületei szintén alkalmasan definiált rendszerek — az egy-egy értelmű megfeleltetési követelményt teljesítő — állapotterei.) Az olyan jellemzőket, amelyeknek lehetséges értékei egy kételemű halmaz elemei, bináris változóknak — jellemzőknek — nevezzük. Ha egy rendszer minden jellemzője bináris, akkor a rendszert (tiszta) bináris rendszernek fogjuk hívni.

Vizsgáljuk meg most azt a fontos esetet, hogy mi lesz az állapottere egy olyan rendszernek, amelynek a modellezésnél csak két bináris jellemzőjét vesszük figyelembe.

A pontos válaszhoz ez az információ nem elég. Lehet pl., hogy a jellemzők nem függetlenek. Előfordulhat pl. olyan rendszer, amelyben, ha az első változó értéke 0, akkor a másodiké is az kell, hogy legyen. A lehetséges állapotokat a koordinátarendszer olyan pontjaival szemléltettük, amelyeknek a koordinátái a jellemzők értékeivel egyeznek meg *(3. ábra)*.

A *4. ábrán* a 3. ábra állapottere (3 elemű ponthalmaz) szerepel egy más elrendezésben. (Van, amikor célszerű az állapotteret átrendezni, vagy az állapotokat képviselő pontokat a szokványostól eltérően kijelölni.)

Ha rendszerünk bináris jellemzői egymástól függetlenül vehetik fel a 0 és az 1 értéket, az állapottér lehet pl. a sík egy négyzetének négy csúcsa *(5. ábra)*.

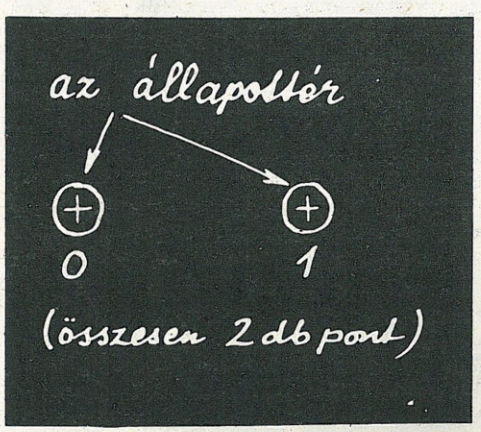
Minden végessok pontból álló állapottér (állapothalmaz) akár az egyenesen, akár a síkon, sőt akárhány dimenziós térben tetszésünk szerint elhelyezhető. Ilyenkor azonban a pontok koordinátái és az állapotjellemzők értékei nem mindig fognak egymással megegyezni *(6. ábra)*.

Ha három, egymástól független bináris jellemzővel dolgozó modellt használunk, az állapottér már „valódi" tér, a közönséges háromdimenziós tér is lehet *(7. ábra)*. A lehetséges állapotokat egy egységnyi élhosszúságú kocka, az „egységkocka" csúcsainak feleltettük meg.

„Terítsük ki" e pontokat is egy koordináta-tengelyre! *(8. ábra)*. A kiterítés nyilvánvalóan többféleképp is elvégezhető. Az itteni kiterítések (az 1., a 6. és a 8. ábra) olyanok, hogy az állapotleíró pontot megadó számsorozatot kettes számrendszerben értelmezve, a pontnak az origótól vett távolságát kapjuk meg.

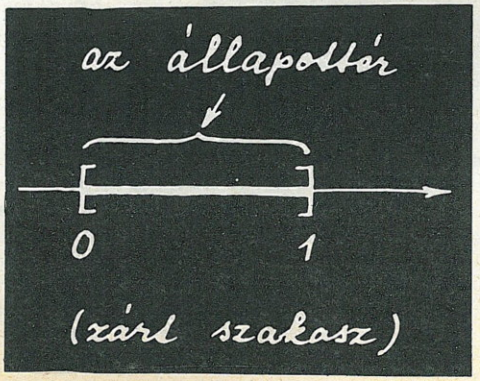
Ha egy kétváltozós rendszer egyik változójának értékei az [a;b], a másiké pedig a [c;d] intervallumból, egymástól függetlenül minden értéket felvehetnek, a rendszer állapotai egy téglalap belső és határpontjainak feleltethetők meg, kölcsönösen egyértelmű módon *(9. ábra)*.

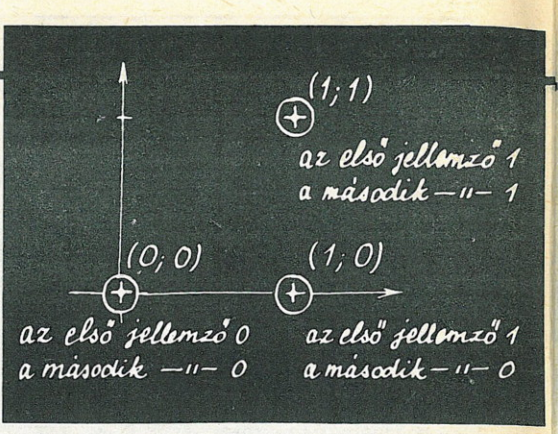
Az állapottér nagyon hasznos segédeszköz, hosszan lehetne még vele foglalkozni. Nekünk azonban egyelőre elég annyi, amennyit e rövid vázlat keretén belül meg-



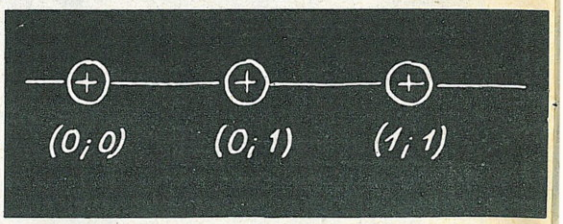
1. ábra

2. ábra



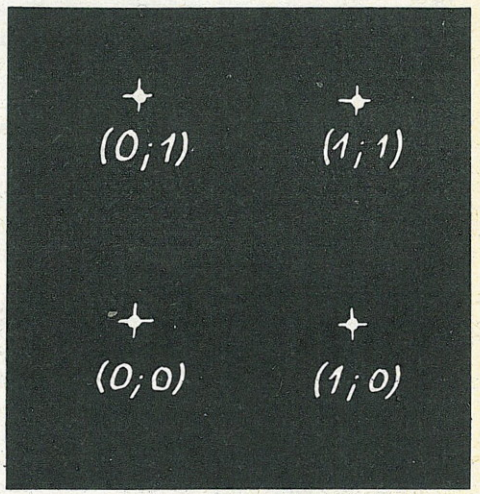


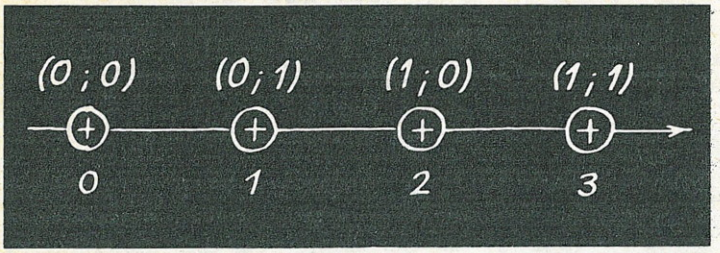
3. ábra



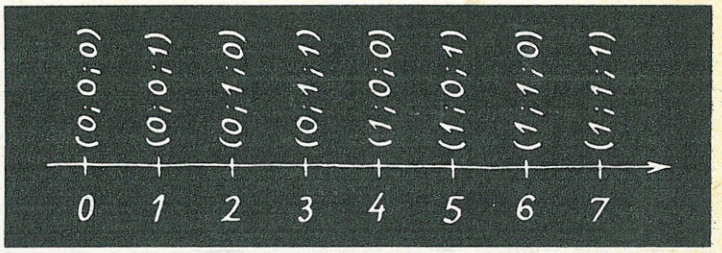
4. ábra

5. ábra

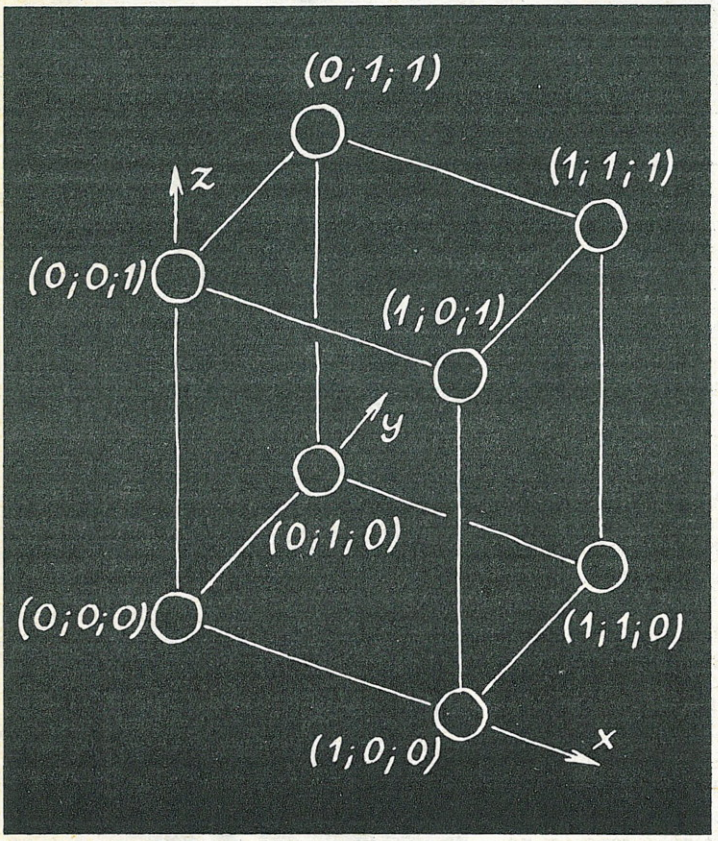




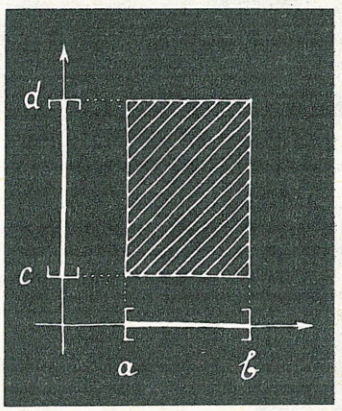
6. ábra



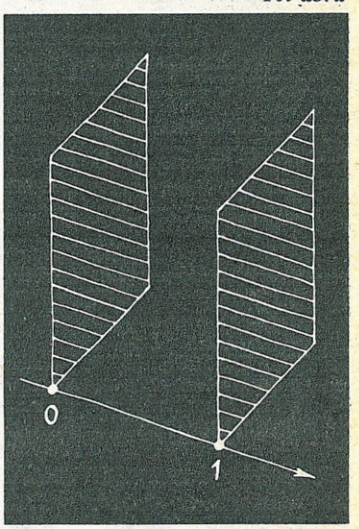
8. ábra



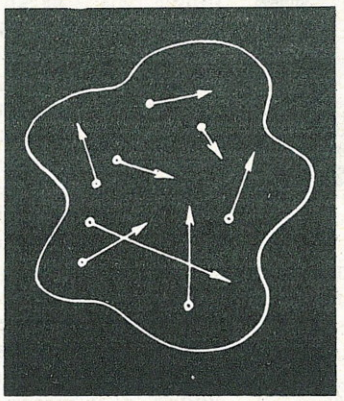
7. ábra



9. ábra



10. ábra



11. ábra

ismertünk. Mielőtt más téma tárgyalásába kezdenénk, egy gyakorlati és egy elméleti megjegyzést kell még tennünk az állapottérrel kapcsolatban. A hivatalos definíció szerint az állapottérben a rendszerállapotnak pontok felelnek meg. Célszerű, ha az állapotreprezentációra nemcsak pontokat, hanem más objektumokat (pl. rendszereket, halmazokat, formulákat) is használunk, amelyeket céljainknak megfelelően értelmezünk (dekódolunk) állapotjellemző értékekké. Így pl. a 7. ábra úgy is értelmezhető, hogy az állapottérben (állapotleíró területen) nem 8 pont, hanem 2 négyzet, illetve paralelogramma van *(10. ábra)*. Az állapottér egy másik egyszerű általánosítása a vektormező. Az előbb pontokhoz négyzetek tartoztak, itt pontokhoz vektorok *(11. ábra)*.

Az állapottér elméleti jelentőségére vonatkozik a másik megjegyzésünk. Az állapottér a matematika (és a formalizálható tudományok) olyan egységes tárgyalását lehetővé tevő eszköz, amelyhez hasonló hatékonyságú eddig még nem volt birtokunkban. Ugyanis egyszerre szemléletes is és egzakt is. Minden lényegeset meg lehet fogalmazni, és szemléltetni lehet állapotterek nyelvén. Sajnos, ma még az állapottér módszer egy alig ismert eszköz az irányításelméleten belül. Reméljük, eljön az az idő, amikor az általános iskolai tankönyvek is ezen az alapon tárgyalják és tanítják a matematikát, a fizikát és a többi formalizálható tudományt.

Rendszerek „életjelenségeinek" leírása

Rendszerek „életjelenségei" — modellezési szempontból — olyan pontok viselkedésével egyenértékűek, amelyeknek koordinátái a rendszerjellemzők értékei. Természetesen mindig tudnunk kell, hogy egy rendszerjellemző-érték a pontoknak mikor melyik koordinátáját adja. (A legegyszerűbb esetben egy rendszerjellemző a pontnak mindig ugyanannyiadik koordináta-értékét szolgáltatja.)

Figyeljünk meg egy rendszert, és vele egyidőben az állapotát képviselő pontot az állapottérben! A rendszer állapotváltozási folyamatainak az állapotleíró pont állapottérbeli mozgása felel meg. Ha a rendszer állapota nem változik, az állapottérbeli pont sem fog mozogni. Ha a pont mozog, a rendszer állapota is változik.

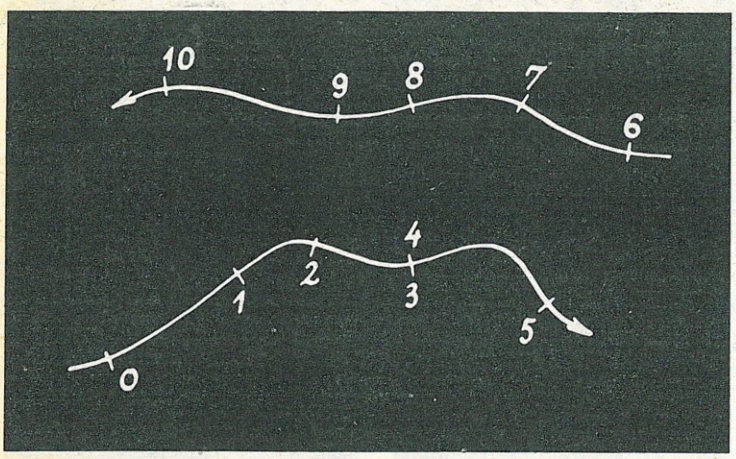
Ha az egyes állapotokhoz, illetve az egyes helyzetekhez az időpontokat is hozzákapcsoljuk, a rendszer viselkedéséről működéstörténetet, viselkedésnaplót készíthetünk.

Ennek két módját használjuk általánosabb körben.

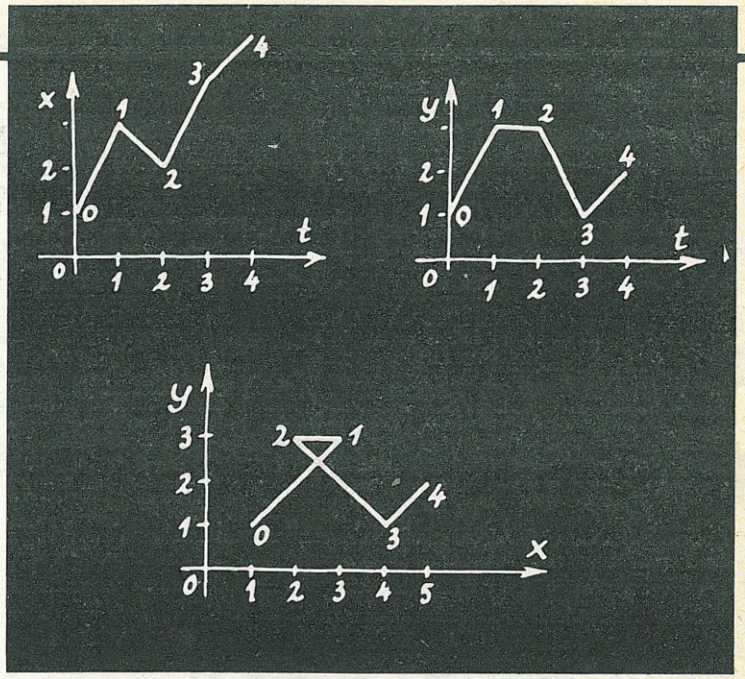
Az egyik módszer a következő. A rendszer viselkedését állapotai egymásutánja írja le. Ezeknek az állapotoknak pontok felelnek meg. A pontok mellé odaírjuk az időpontot, amelyben a szóban forgó állapot fennállt. Ez azonban a gyakorlatban nem mindig valósítható meg. A legtöbbször csak néhány (pl. egész) időpontot tudunk megjelölni *(12. ábra)*. Példánkban a rendszer az 5-ös és 6-os időpont között valamikor „ugrott".

Gyakran használható a pályára valahogyan (pl. merőlegesen) felvitt időfüggvény is *(13. ábra)*. Példánk esetében az ábráról leolvasható, hogy a rendszer a D állapotban 0 ideig, a C-ben viszont pozitív hosszúságú ideig tartózkodott. (A 0 ideig tartózkodás természetesen nem jelenti azt, hogy a rendszer nem volt a szóban forgó állapotban. Fordítva viszont igaz, hogy ha egy rendszer egy állapotot nem vesz fel, akkor abban való tartózkodásának időtartama 0.)

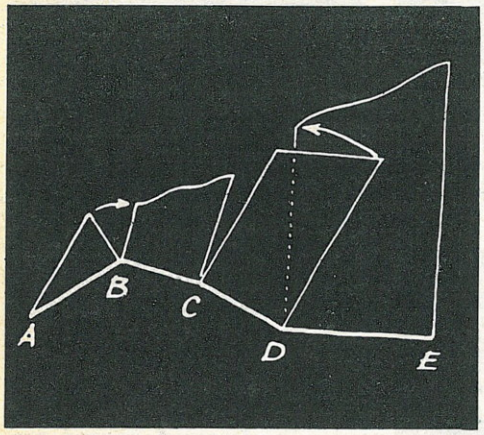
A másik működéstörténet-készítő, viselkedésnapló-író módszer az állapotteret több állapottérre bontja fel. A rendszer



12. ábra



14. ábra



13. ábra

minden jellemzőjének viselkedését ilyenkor általában külön-külön koordináta-rendszerekben ábrázoljuk, amelyek mindegyikében a független változó az idő. (A *14. ábrán* bemutatott példához megrajzoltuk az időpontokkal jelölt állapotalakulási pályát is.) Az elmélet főleg ezt a módszert használja működéstörténet-készítésre.

Visszatérve egy előző példánkra, az 5. ábra esetében a rendszer 4 állapotban lehet. Ezeket a *15. ábrán* nagyobb körökkel jelöltük.

A rendszer „életjelenségei", állapotváltozásai e körök egyikéről a másikára való átmenetek, átugrások révén figyelhetők meg, írhatók le. Ezeket az átugrásokat irányított vonalakkal szokták reprezentálni. Jól látszanak a rokon vonások a véges automaták körében használt „állapotátmenet" gráf és ábránk között.

A 15. b) ábra azonban, ha működésnaplónak, viselkedéstörténetnek szánjuk, egyértelműen hiányos (és ezért hibás). Az ábrán ugyanis nem szerepelnek pl. sem az időmúlásra utaló információk, sem pedig — ez a súlyosabb hiba — a rendszernek az „ugrásban levési" állapotai, azaz azok, amikor az egyik állapotból a másikba jut. (A szakirodalom automataábráit és automatamodelljeit ebből a szempontból megvizsgálva meglehetősen szomorú kép alakulhat ki a tárgyilagos szemlélőben.)

Ha a rendszernek „n" darab állapota van, akkor n^2 különböző állapotátmenet lehetséges, ha egy állapotból ugyanoda visszatérést vagy a helyben maradást (e kettő nem ugyanaz) is egy átmenetként számításba veszik. Ha a helyben maradást és az ugyanoda visszatérést meg kell különböztetni, akkor az állapotátmenetek száma nyilvánvalóan több is lehet. Gyakran nem fontos az összes különböző fajta átmenet egymástól való megkülönböztetése. Gyakori eset, amikor a különböző átmeneteket csoportokba foglalva, egy-egy csoporton belülieket azonosnak lehet tekinteni. Ha az átmeneteli állapotokat ugyanúgy körökkel jelöljük mint a többi állapotot, nyilvánvalóan tovább lehet az állapotokat szaporítani, úgy, hogy külön jelöljük az átmeneteli állapotokba való átmeneteleket is. Ezt a tevékenységet azonban nyilvánvalóan nem szabad nyakló nélkül és öncélúan űzni.

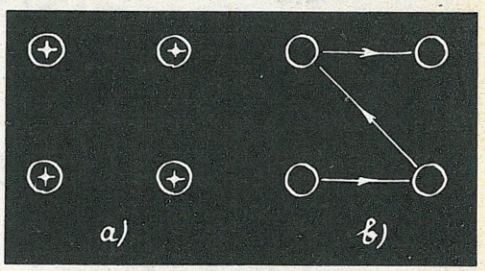
Ha az összes átmeneteli állapotot egyetlen állapotként vesszük csak számításba, hasznos működéstörténet-leíró eszközhöz jutunk. A *16. ábrán* a vonal a rendszer pályája az állapottérben. A rendszert egy járműnek tekinthetjük, amely a pályán (pl. egységnyi sebességgel) mozog. Az egyes állapotokban addig tartózkodik, ameddig az állapotoknak megfelelő körön belül halad. Az — egyetlen — átmeneti állapot a körökön kívüli síkrész.

A 16. ábrán szemléltetett módszert természetesen a pályához csatolt időjelölés már megismert módszereivel is hasznosan alkalmazhatjuk.

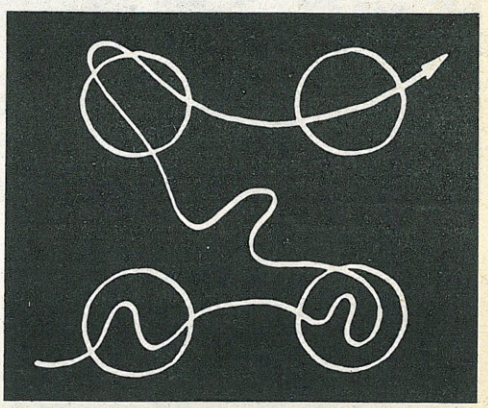
E módszercsoport rendkívüli hasznossága ott is megmutatkozik, amikor több rendszer egymástól független vagy egymással összehangolt működését akarjuk leírni. Ezen túlmenően természetesen az állapotkijelölő tartományoknak sem kell egymástól idegen köröknek lenniök *(17. ábra)*.

A *18. ábrán* egy bináris számlálás történetét szemléltetjük, ØØ-tól 11-ig. (Hol vannak az ábrán a külön nem jelölt átmeneti állapotok?)

Ha több összehangolt rendszer működését kell állapottérben ábrázolni, akkor azok összes jellemzőjét külön-külön számításba véve, azokat egyetlen (összetett) rendszer jellemzőinek is felfoghatjuk. Ezt pedig egy megfelelő kiterjedésű (dimenziószámú) térben ábrázolhatjuk, mint egyetlen pontot. Fontos lehet emellett (nem ehelyett!) a rendszerek külön-külön, saját állapotte-

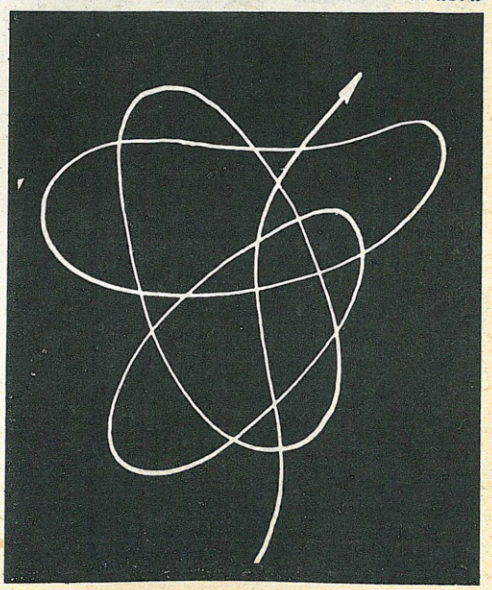


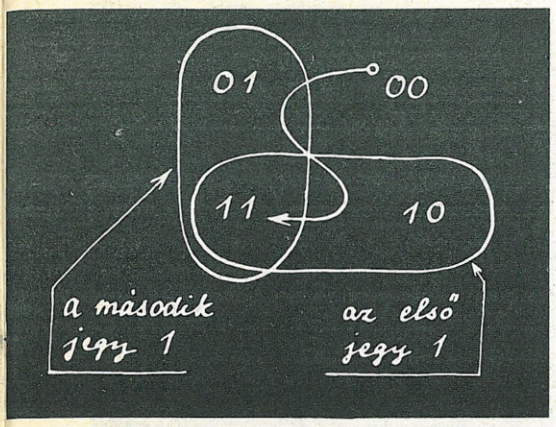
15. ábra



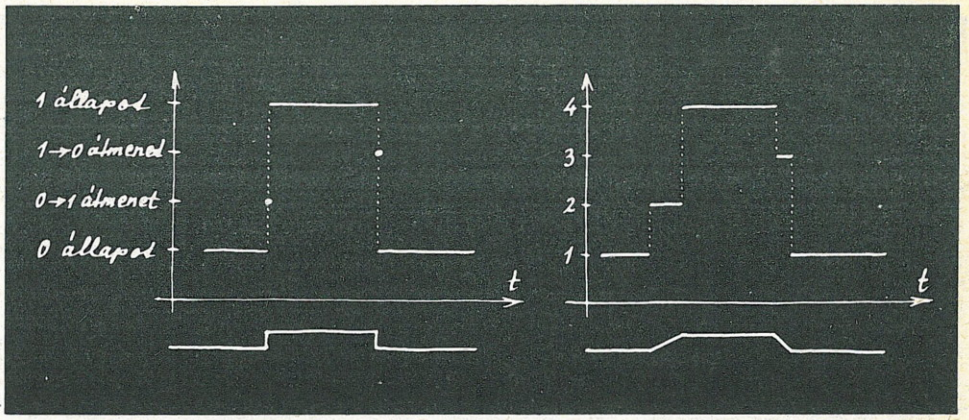
16. ábra

17. ábra

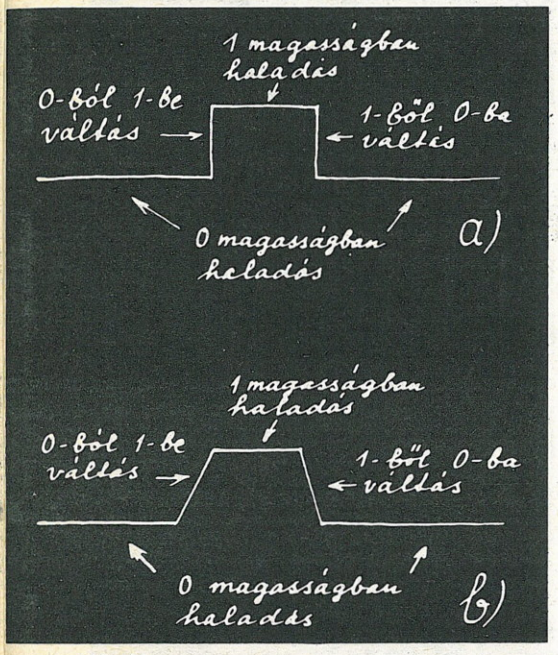




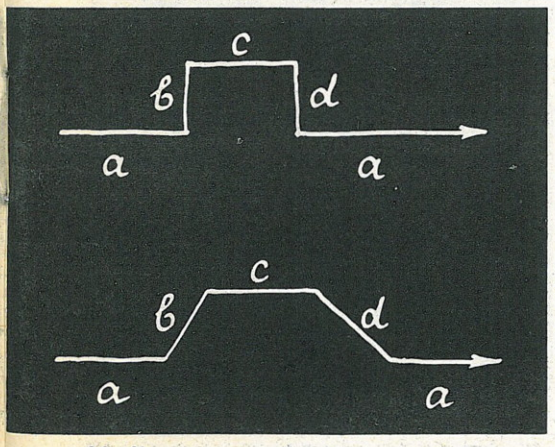
18. ábra



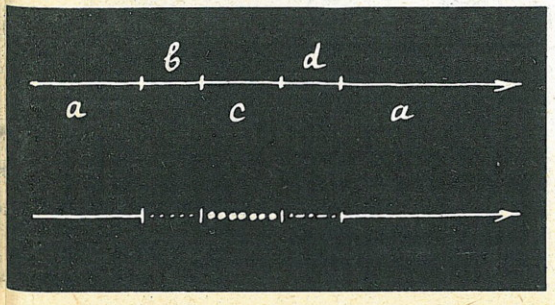
22. ábra



19. ábra



20. ábra



21. ábra

reikben való figyelemmel kísérése is. Például úgy, hogy az összes rendszert egyetlen állapottérben, pl. a legnagyobb kiterjedésűben ábrázoljuk. (Az ilyenkor alkalmazható összehangolási módszerekkel majd máskor foglalkozunk.)

Rendszerünk élete egy-egy működéstörténetben lényegében pálya volt, ugyanabban az absztrakt értelemben, ahogyan az emberi életet életútnak, életpályának szokták nevezni. Pályát mozgó pont is leírhat. A pont egyszerűbb esetekben vonalat rajzol, vonalat hagy maga után a síkon, a térben stb. A pálya sajátságait e vonal tulajdonságai adják. Nézzük meg tehát a működéstörténet-készítésnek (rendszerleírásnak) néhány vonaltulajdonságokra épített módszerét.

Vonaltulajdonságokkal leírt működés

Rögtön az elején meg kell mondanunk, hogy nemcsak vonalak, hanem ponthalmazok, felületdarabok, és más geometriai alakzatok is jól használhatók leírási célokra, a vonalakat egyszerűségük és széles körű használhatóságuk emelik ki az említettek közül. Másik bevezető megjegyzésünk az, hogy ha csak lehet, ugyanannak a feladatnak a reprezentálását *minél több módon végezzük el,* mert a lényegmegragadás és minél többoldalú megismerés mellett az *ellenőrzés*nek alig van ennél jobb módszere.

Egy bináris jellemzővel leírható rendszer működéstörténetének egy „ideális" és egy „realistább" leírását mutatja a *19. ábra.* A 19. a) ábrán a vonalnak 4 jellegzetes tulajdonságú részét figyelhetjük meg. Ezek: 0 magasságban halad, 0-ból 1-be vált, 1 magasságban halad, 1-ből 0-ba vált. A 19. b) ábrán a helyzet ugyanaz, csak az átmenetek valósághűbbek. Ha nem akarjuk az időtengelyt használni, a vonalak színezésével, grafikus vagy pl. betűkkel való megkülönböztetésével dolgozhatunk *(20. ábra).*

A vonalakat akár egyenesre ki is nyújthatjuk *(21. ábra).* Az idő kezelésére a módszereket már megismertük, ezzel itt az egyszerűség kedvéért nem foglalkozunk.

Nincs akadálya a koordináta-rendszerbe való visszatérésnek sem *(22. ábra).* Most is — mint máskor — a tengelyen állapotjellemző értékek vannak. Legfeljebb a szavakkal való megjelölésük szokatlan. Használhatunk azonban szavak helyett számokat is, lehet pl. az 1, 4, 2, 3 rendre a 0, az 1, a 0→1 átmeneti és az 1→0 átmeneti állapotot jelölő szám.

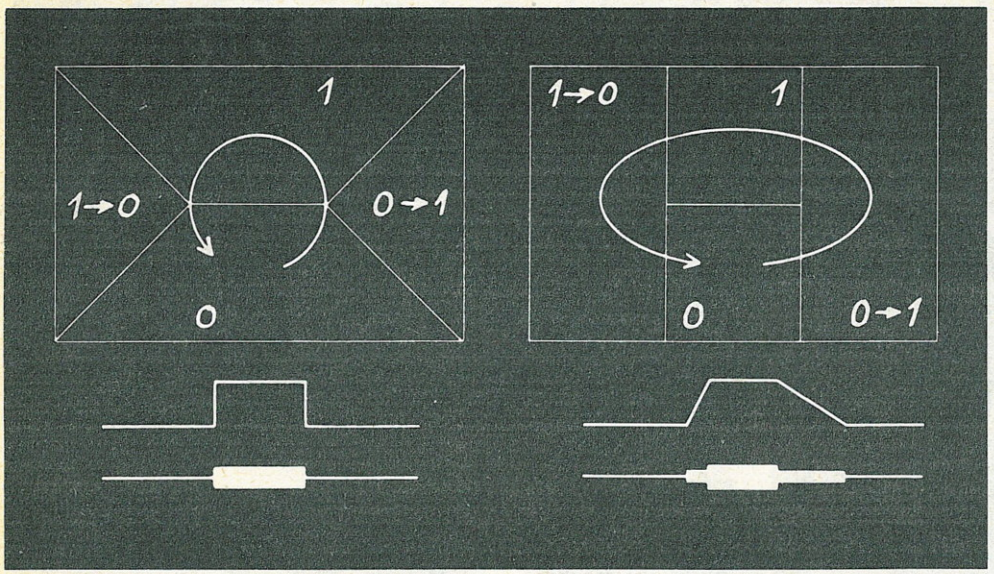
A *23. ábrán* a vonal valamilyen tartományban való tartózkodása adja a tulajdonságot. Ha pl. csak egy átmeneti állapotot különböztetünk meg, akkor a vonalnak (amit ki is teríthetünk egy egyenesre) pl. vékony, közepes és vastag kihúzása különböztetheti meg a 0, az átmeneti és az 1 állapotban levést (lásd: 23. ábra alján levő vázlat).

Ha a vonalnak az irányát figyeljük, megállapodhatunk abban, hogy bizonyos irányok vagy iránytartományok állapotoknak feleljenek meg, és fordítva. A töréspontoknak természetesen külön lehet értelmet tulajdonítani (lehet!). A *24. ábrán* bemutatott 2 vonaldarab értelmezését ráírtuk a vonalszakaszokra. (A vonalak irányítására — természetesen — szükség van.) A 24. ábrán az egyetlen átmeneti állapothoz 0 irányszög, az 1 állapothoz +60°, a 0 állapothoz −60° tartozik, és fordítva. Aztán pedig csak 2 iránynak van szerepe. A 0 szögű, átmeneti állapothoz tartozik, a 90°-os pedig vagy a 0 vagy az 1 állapothoz. Az egyértelműséghez egy 90°-os szögű szakasz értelmét meg kell adni, és meg kell állapodni abban, hogy minden átmeneti állapot szomszédai különböző állapotok. Ha nem tesszük e külön megállapodást, a vonalvastagság, a vonaltípus, vagy pl. számráírás alkalmas lehet az azonos irányú vonalszakaszok megkülönböztetésére.

A *25. ábrán* 4 irányt használtunk, amelyek jelentése 0, 1, átmeneti, közömbös. (A *közömbös* vonalon haladásnak nincs időigénye. Lehet a közömbös vonal pl. pontozott, vagy szaggatott görbe is.)

Nyilvánvalóan tetszőlegesen értelmezhetünk véges vagy végtelen „szélrózsákat", és nem csak egy közömbös (azaz csak segédvonalként használt) irány definiálható. Azt is megtehetjük természetesen, hogy egy-egy állapotnak több irányt is megfeleltetünk *(26. ábra).*

A színek, a vonalvastagságok, a különböző vonalkihúzás (szaggatott, pontozott stb.) szerepét már említettük. Ezeken kívül hasznos lehet az a megállapodás is, amely érdemi vonalszakaszokként csak egyenes szakaszokat enged meg, minden más vonal segéd (csak összekötő) feladatot lát el. *(27. ábra).*



23. ábra

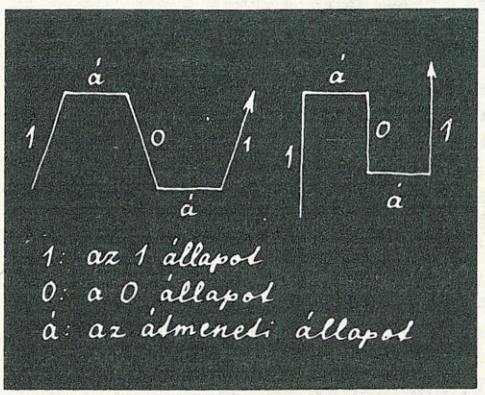
Az összes megkülönböztető jegy az összessel kombinálható. A vonalvastagság, a kihúzás módja, a szín, a szög mellett egymáshoz és tartományokhoz való viszony, koordináta-rendszerekben való elhelyezkedés mind informálhat valamiről. Noha mi ezekkel a lehetőségekkel alig éltünk, reméljük „fekete-fehér" korlátok között sikerült a módszer rendkívüli színességét és hasznosságát kellőképp szemléltetnünk. Ennek az eszköznek e sorok írója ugyanolyan fontos szerepet tulajdonít, mint az állapottérmódszernek.Természetesen a kettő nem független egymástól, tudatos és alkalmas kombinálásaik különleges hatékonyságú ábrázolási, leírási eszközök kialakítását teszik lehetővé.

A vonalak értelmezése és különböző értelmezések közötti összefüggések elmélete (és gyakorlata) egy *új matematikaalkalmazási fejezet,* melynek terjedelmét ma még csak sejthetjük. A vonaltulajdonságokkal való működésleírás hatékonyságáról — reméljük — már minden olvasó meggyőződött. Befejezésül egy, az értelmezési lehetőségek gazdagságát szemléltető példát mutatunk be.

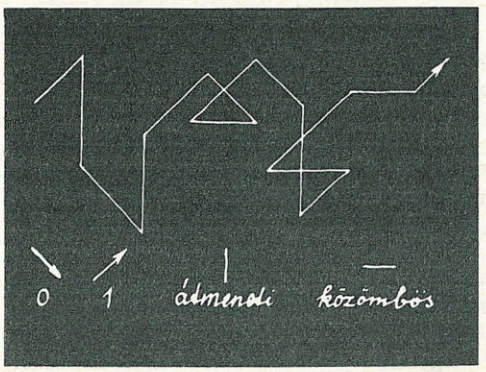
A működéstörténetet (vagy működéstörvényt) egy pályából származtatjuk. Az idő múlása az ívhosszal arányos, az egy-egy időpontbeli rendszerállapotot a vonal görbületi sugara határozza meg. E meghatározás módja azonban sokféle lehet. Függhet például a rendszer eddigi viselkedésétől is. A pillanatnyi állapot lehet például rekurzív, azaz előző értékekre, előző állapotokra támaszkodó módon is kiszámítható.

Hogy erre a nehezebb, de a számítástechnika „haladottabb" alkalmazásaiban már *nem nélkülözhető* értelmezésfajtára is lássunk egy példát, a *28. ábra* pályáját választjuk a szemléltetéshez.

A rendszer egyetlen számmal jellemezhető állapotát a pályán mozgó pont határozza meg. A pont (út)sebessége a pálya kezdőponttól számított ívhosszának 1-gyel növelt értéke. A rendszer pillanatnyi állapotát úgy kell kiszámítani, hogy a pálya görbületi sugara abszolút értéke és görbülete abszolút értéke minimumát megszorozzuk a mozgó



24. ábra

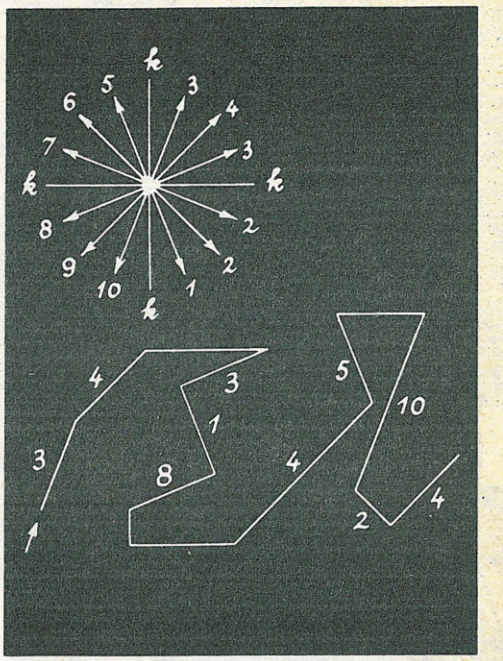


25. ábra

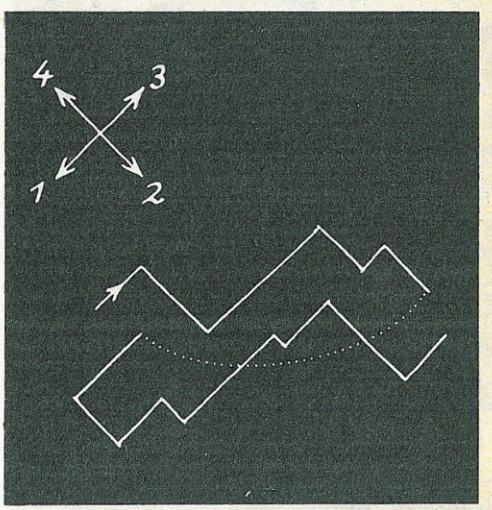
pontnak a kezdőponttól vett összes eddig előfordult távolságai közül a maximálissal.

E feladathoz azonban már gép kell. Ahhoz pedig még inkább, amelyben a sebesség az ívhossz 1-gyel növelt értéke, plusz a pillanatnyi állapotjelző szám abszolút értéke. Ez a definíció nemcsak rekurzív, hanem impilcit is. És egyben példa olyan feladattípusra, amelynek csak egyszerűbb elemeit vagyunk képesek — még a legnagyobb teljesítőképességű gépekkel is — megoldani. A közepesekkel pedig már teljesen tehetetlenek vagyunk . . .

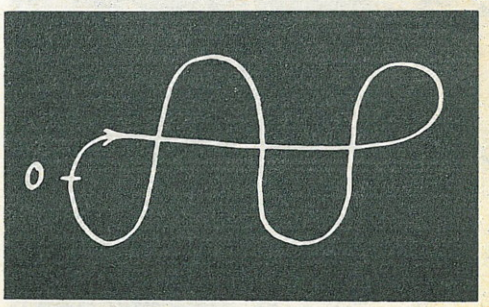
A mi saját állapotaink terében tehát vannak még állapothalmazok, amelyek tehetetlenségi állapotainkat reprezentálják. Azonban ezek visszaszorításában is hasznosak lehetnek az állapottérmódszerek, különösen úgy, ha általánosabban értelmezzük és



26. ábra



27. ábra



28. ábra

használjuk ezeket, mint ahogyan szokásos, azaz az állapottér egyszerűen csak az állapotok, az állapotegyüttesek ábrázolása valamilyen (szemléletes) módon, természetesen nemcsak a térben és nemcsak pontokkal.

Miután a leíróeszközöket tekintve alaposan felkészültünk, az *utánzás,* a *követés alapjelenségét* fogjuk elemezni, ami által a (gépen belüli) *terjedési, továbbítási* jelenségek egzakt megismeréséhez szerezzük majd meg az alapokat.

P. CS.

FORTH rendszerek összehasonlítása

Az alábbiakban különböző mikroszámítógépeken futó rendszereket hasonlítok össze. Valamennyit legalább egyszer kipróbáltam. CP/M alatt futó FORTH rendszert azonban nem volt alkalmam kipróbálni.

Sinclair Spectrumra sok felhasználónak megvan a sforthv48 nevű,

V 2. Ø March 83 M. HAMPSON
CP SOFTWARE

felirattal bejelentkező és az FP 50 nevű,

MIKE HAMPSON (May 1983)
SPECTRUM FLOATING-POINT
FORTH

felirattal bejelentkező rendszer. Ezekhez nem találtam szerkesztőt (editort), és egyébként hátrányosan eltértek a FORTH—79-től, ezért érdemi munkára alkalmatlannak találtam őket. Itt nem is foglalkozom velük.

Az összehasonlítás azért érdekes, mert a FORTH, mint rendkívül rugalmas rendszer, könnyen bővíthető. A hardverkötöttségek korlátain belül a más rendszereknél látott speciális szavak, ügyes megoldások saját rendszerünkbe is beilleszthetők.

Az összehasonlítás fő szempontjai a forrás, a dokumentáció, az alapszavak, a duplapontos szavak, a grafikus képességek, a speciális szavak, a szerkesztő, az assembler és a hardver reset.

A *táblázatban* megadott, µs-ban értendő végrehajtási idők a tényleges bruttó értékek, tehát azokat a belső megszakításokra fordított átlagidővel együtt mértem. A rövid időknél a pontosság kb. ±5 µs. A mérést hosszú ciklusokkal végeztem, ezért az EMIT és a számkinyomtatás futási idejébe a képernyőgörgetést (scroll) is belemértem.

A dokumentációként többször hivatkozott könyv teljes címe: Lipovszki Gy.—Subai L.—Beszeda T.: FORTH programozási rendszer és nyelv. Bp. 1985. LSI ATSZ.

Futási idő összehasonlító táblázat

MŰVELET	48 k-s Spectrum		Commodore 64		HT-1080Z	Primo A-48 és A-64
	Spec FORTH	Abersoft fig-FORTH	DATA-TRONIC FORTH	Performance Micro Dev FORTH	zFORTH	Primo FORTH
üres : definíció	130	90	140	160	185	110
SP!	60	50	70	110	90	70
üres DO...LOOP ≈	150	95	160	140	580	170
AND, OR	50	50	90	95	120	40
+	60	40	90	95	90	40
–	60	40	90	95	130	40
*	390	1 660	910	920	1 150	1 190
U*	390	360	710	710		660
/	1 750	2 890	4 630	5 300	1 480	2 020
U/	890		1 090	1 500		900
U/MOD		770				
*/	4 800	4 400	6 950	7 700	2 580	2 980
2*, 2/			70	70	110	40
DROP	50	30	50	65	90	40
2DROP	220	160	60	290		50
Konstans	55	40	85	95	105	45
LIT (szám)	50	40	95	105	105	45
EMIT	2 700	2 720	1 360	1 330	810	4 080
Egyjegyű szám kiíratása	14 400	12 700	16 000	13 500	11 700	15 000
Ötjegyű szám kiíratása	41 000	37 000	43 000	45 000	27 000	49 000
Elemi grafikus pont felrajzolása		290			290	1 620

Spec FORTH 48 k-s Spectrumra

Bejelentkezés:

SPECT ZX—FORTH 1.1
(c) 1982 ARTIC COMPUTING LTD.
ALL RIGHTS RESERVED
32470 BYTES FREE

Programcsere útján, kazettán terjedt el, eredeti leírásának fénymásolatával együtt. Kiegészítő leírása a FORTH-könyvben található.

Az alapszavak és a duplapontos szavak gyakorlatilag megegyeznek a „szabvány" fig—FORTH-szal. Grafikus szavaival pont, egyenes és kör rajzolható, a képernyő grafikus tartalma kinyomtatható. Külön lehetőség van karakteres nyomtatásra. Speciális szavai jól illeszkednek a hardverlehetőségekhez, de sajnos a portkezelő szavak hiányoznak.

A szabványos, de nehezen kezelhető sorszerkesztőt külön kell betölteni.

Assemblere nincs, de megírható. A futó program a BREAK-kel mindig üzembiztosan megszakítható.

Abersoft FORTH 48 k-s Spectrumra

Bejelentkezés:

48K SPECTRUM FIG—FORTH 1.1A
(c) Abersoft: 1983

Programcsere útján terjedt el. Kiegészítő leírása — melyben a CASE-re hibásan hivatkoznak — a FORTH-könyvben található, és jól használható. Eredeti leírással nem találkoztam.

Az alapszavak és a duplapontos szavak megegyeznek a „szabvány" fig—FORTH-szal. Grafikus szavaival pont és egyenes rajzolható, de kör nem. Speciális szavai jól illeszkednek a hardverlehetőségekhez, bár a hang előállításához kísérletezni kell, amiben a leírás sem segít. Sajnos a grafikus nyomtatás hiányzik, de szöveg nyomtatható. Portot tud kezelni. Beépített szabványos sorszerkesztője van, amit azonban elég nehéz használni.

A screenek betöltésekor kellemetlen meglepetések érhetnek, ha nem vigyázunk:

ha egy szó az adott sorba nem fér el, nem folytathatjuk a következőben; a magyarázat nyitó és csukó zárójelének ugyanabba a sorba kell kerülnie.

Rendkívül hasznos specialitás a memóriában tárolt RAM-DISC, amelyben 10 screent lehet tárolni. Ezt a területet egészben lehet kazettára menteni és onnan visszatölteni. Miután 10 screent meghaladó programot ritkán írunk, a RAM-DISC nagyon jól kihasználható. Az indexsorok listázhatók.

A CASE struktúrában vigyázni kell az utolsó ENDOF és az ENDCASE közötti szavaknál, mert az ENDCASE törli a verem (stack) legfelső elemét, ami eltér a szokásostól. A paraméterverem mérete több kbájt lehet, gyakorlatilag túltölthetetlen.

Assemblere nincs, de a FORTH-könyv alapján megírható.

Sajnos a futó program a Spectrum ROM program hibája miatt nem szakítható meg, RESET után nem indítható újra. Ezért a végtelen ciklusok vagy egyéb „eltévelyedések" nem korrigálhatók, és ha nem mentjük ki rendszeresen a RAM-DISC-et, forrásnyelvi programunk is elvész.

A futás nyomtatási, magnó- stb. hiba esetén BASIC hibaüzenettel áll le. Ilyenkor ENTER után RUN 3-mal melegstartot hajthatunk végre, tehát a program menthető.

Hasznos lehetőség, hogy programjainkat nemcsak forrásnyelven, hanem lefordított formában, komplett rendszerként is kazettára vehetjük. Ehhez magunknak kell módosítani a rendszert, és a BASIC behúzórészt is át kell írni!

DATATRONIC FORTH C64-re

Bejelentkezés:

```
C64 FORTH COMMODORE 64, V1
COPYRIGHT DATATRONIC AB,
1983
```

Programcsere útján mágneslemezen terjedt el, a ROM változat eredeti leírásának fénymásolatával együtt. Ez a rendszer a gépbe dugaszolható ROM-ban és mágneslemezen is megtalálható, azonos működésmóddal. Kiegészítő leírása a FORTH-könyvben szerepel.

Az alapszavak megegyeznek a fig—FORTH-szal, de jóval több duplapontos és néhány triplapontos szava is van. Nagy felbontású grafikát egyáltalán nem tud kezelni; ezt megírni is elég körülményes. Nincs lehetőség a belső hanggenerátorok közvetlen vezérlésére, ami elég nagy hátrány, éppen a szép hangú és változatos rajzkészségű C64-esnél. A be-kiviteli portokat tárként kell megcímezni, kezelésük nem okoz gondot. Speciális szavainak egy része csak a mágneslemezes rendszer fájlkezelését végzi. A nyomtató használata egyszerű. CASE struktúra nincs, de megírható, van viszont PICK szó.

Beépített szerkesztője a soronkénti szerkesztés mellett egy szerény képességű, teljes képernyős szerkesztést is lehetővé tesz. Ez utóbbit érdemes használni, de a törlés és beszúrás néha meglepetést okoz. A mágneslemez miatt a screenek használata nagyon kényelmes. A RAM-ban egyszerre 3 screen lehet.

Beépített assemblere van.

A program futása hardver resettel (STOP+RESTORE) bármikor megszakítható, a vezérlés üzembiztosan a FORTH rendszer melegstartjára adódik.

A rendszer jellegzetessége, hogy betöltés és elindítás után a BOOT—UP szóval inicializálni kell, enélkül a lemezműveletek és a szerkesztő nem működik.

Performance FORTH C64-re

Bejelentkezés:

```
FORTH C64
COPYRIGHT 1983
PERFORMANCE MICRO PROD.
SER# A100280
```

Jellegzetessége a fehér alapon fekete írás, ami elég zavaró, de átváltható. Programcsere útján terjedt el, mágneslemezen. Eddig semmiféle leírását nem láttam, az ismeretlen szavakat magam derítettem fel.

Az alapszavak és a duplapontos szavak megegyeznek a fig—FORTH-szal. Sajnos, nincs finom felbontású grafika, és a hanggenerátorokat sem lehet közvetlenül kezelni, ami a C64-nél elég nagy hátrány. A be-kiviteli portok kezelése nem gond, RAM-ként viselkednek. Speciális szavai, amelyek főleg a lemez és a fájlok kezelését végzik, elég zavaróan hatnak, kezelésük körülményes.

A nyomtató használatát külön kell megírni. PICK, ROLL szó és CASE struktúra nincs, de megírható.

Teljes képernyős szerkesztőjével élmény dolgozni: a célszerű szerkesztési-mozgatási műveletek funkciógombokkal vezérelhetők. A szerkesztőnek egyetlen, a C64-nél szokásos hibája van: páratlan számú karakter után nem lehet törölni.

A RAM-ban egyszerre 4 screen lehet. Mozgatásuk a RAM és a lemez között automatikus, rendkívül kényelmes.

Assemblere nincs, és minta hiányában elég nehéz megírni.

A futás hardver resettel megszakítható, de sajnos ilyenkor a rendszer néha „elszáll". A paraméterverem elég kicsi: 44 szimplapontos elem fér bele. Ennél is nagyobb baj, hogy a ciklusok belsejében nincs veremellenőrzés. Ha a verem túlcsordul, egy kritikus részbe történik a beírás, és ilyenkor a rendszer megmerevedik, hardver resettel sem indítható újra.

Hasznos lehetőség, hogy a programok lefordított formában, komlett rendszerként is lemezre vihetők.

TINY—FORTH HT—1080Z-re

Bejelentkezés:

```
TINY—FORTH 2.1
(c) 1979 BY THE SOFTWARE FARM
```

Programcsere útján, kazettán terjedt el, leírásával még nem találkoztam.

Alap- és duplapontos szavai megegyeznek a fig—FORTH-szal. Grafikus szavai jók, de a gép gyenge grafikus képessége korlátozza használatukat. Speciális szó végzi a véletlenszám-generálást. Portot tud kezelni. A rendszernek beépített sorszerkesztője van.

A program nem szakítható meg; a RESET-gomb megnyomásával a BASIC-be jutunk vissza, de innen nem indítható újra a FORTH.

Primo FORTH Primo A—48-ra és A—64-re

Bejelentkezés:

```
PRIMO FORTH 1.0
(c) BME—KSZK 1985.
25 520 byte szabad memória
```

Az Elektromodul forgalmazza kazettán, leírás külön kapható hozzá, amelyben sajnálatos hibák és hiányosságok fordulnak elő. Az összesítő szójegyzékben több szónál előjel nélküli műveletet jeleznek, holott a szavak a szokásos előjeles módon működnek. A SYSTEM szótár szavairól és használatukról semmilyen leírás nincs, pedig olyan szó is van, amely a többi fig—FORTH-ban nem található.

Az összehasonlított rendszerek közül ez az egyetlen, amelyik az 1983-as szabvány szerinti, emiatt néhány szónál (U*, U/ stb.) vigyázni kell.

Az alapszavak egy részét, amelyeket az átlagos programozó nem használ, külön szótárba, a SYSTEM-be tették. Ez előnyös abból a szempontból, hogy a kezdőket nem zavarja fölöslegesen. Nagyon sok duplapontos szó van. Grafikus képességei jók, nemcsak rajzolni, hanem törölni is részletenként lehet. Portot tud kezelni. Speciális szó inkább csak a szokásos szavak variálásából származott.

Sajnálatos, hogy a rendszer nem tudja kezelni a csatlakoztatható nyomtatót és a belső hangjelzőt sem. A rendszerrel való munkát nehezíti, hogy még mindig nem jelent meg a Primo alapszoftver leírása. Egyes tulajdonságokat csak ennek ismeretében lehetne megítélni.

Ez az egyetlen FORTH rendszer, amely magyar nyelvű üzeneteket küld. A VLIST és a screenek listázása nagyon kényelmesen szemlélhető, nem szalad el a kép. A program futása a BREAK billentyűvel vagy a hátsó RESET gombbal bármikor üzembiztosan megszakítható. Több, → szóval egymáshoz láncolt screen beolvasása nem folyamatos, minden egyes screen után le kell nyomni egy billentyűt.

Az egyszerű sorszerkesztő 32 karakter hosszúságú sorokra tördeli a screent. Használata nagyon nehézkes, amihez az érintős billentyűzet is hozzájárul. Ez a rendszer megérdemelt volna egy képernyős szerkesztőt. Egyszerre 4 screen lehet a memóriában.

A rendszernek assemblere nincs.

zFORTH HT—1080Z-re

Bejelentkezés:

```
(c) Z—FORTH V1.2
B. Markus A. Nagy — 1984.
```

A rendszer kazettán van, jó minőségű eredeti leírással. A fig—FORTH egyes, önmagukban nem használatos alaprutinjai a zFORTH-ban nem találhatók meg. A rendszer belső működésében is eltér a szokásostól, aminek nyilvánvaló oka, hogy 16 kbájtba kellett belezsugorítani mindent.

A minimumra csökkentett helyfoglalás ellenére a rendszer teljes értékű, sőt többet tud néhány jóval hosszabb társánál. A felhasználói programok a szokásosnál tömörebbek, de sok szöveges változót tartalmazó nagyobb programoknál könnyen kifuthatnak a szabad területből.

Az alapszavak néhány kivételével megegyeznek a fig—FORTH-szal. Duplapontos szavak nincsenek. Grafikus pont felrajzolása, törlése és tesztelés elvégezhető. Kár, hogy a számítógép grafikája gyenge felbontású. Speciális szavai portkezelést, hang- és véletlenszám-generálást végeznek. CASE struktúrája sokkal nagyobb, mint a többi rendszeré. PICK, ROLL szó nincs, nyomtató sem működtethető.

Ügyes megoldású beépített képernyőszerkesztője van. Kezelését az átlagosnál nehezebb megtanulni, de azután nagyon jól használható. Egyszerre csak 1 screen lehet a RAM-ban. Az EDIT és LOAD szavak egészen speciálisak: Ø argumentummal az éppen a RAM-ban levő screen kezelhető, negatív argumentummal pedig a szalagon éppen következő screen beolvasása kezdődik meg.

Külön betölthető assemblere van.

A program a hátoldali RESET gombbal bármikor megszakítható, ekkor üzembiztos melegstart történik. A rendszer minden szó végrehajtása után ellenőrzi a veremtárat, túlcsordulás és kiürülés szempontjából.

Amint a részletezésből kiderült, egyik rendszer sem ideális. Miután a rendelkezésre álló számítógép általában adott, nincs nagy választási lehetőségünk. Viszont ha egy rendszert komolyan és hosszabb távon is használni akarunk, érdemes egyszer nagyobb munkát befektetni, hogy saját céljainknak megfelelő szavakat írjunk, amelyek később sokszor felhasználhatók.

Az egyes rendszerekben rejlő lehetőségeket jól mutatják a magam készítette „alapbővítések".

A HT—1080Z gépen a zFORTH rendszert használtam, és olyan screennel egészítettem ki, amely sok apró ötletet tartalmazott: a gyakran használt, hosszú szavak helyettesítése rövid névvel; késleltetések; adott billentyűk lenyomását váró szó; végtelen ciklus könnyű szervezését megvalósító szó stb. A zFORTH megbízható és jó rendszernek bizonyult. Elsősorban saját építésű perifériák vezérlésére használtam.

A C64-es számítógéphez a Performance FORTH-t használom, amelyet alaposan kibővítettem. Az apró ötleteken túl nagyobb feladatot jelentett a nagy felbontású grafika alapszavainak, az egyszerűbb nyomtatókezelés és a grafikus nyomtatás szavainak megírása. Ezzel a munkámmal nem vagyok megelégedve, mert a futási idő rövidítése érdekében egyes részeket gépi kódú betétszavakkal kell helyettesíteni. E rendszerrel online mérőrendszert vezérlek.

Spectrum számítógépen az Abersoft fig—FORTH rendszert „javítottam fel". Megtanítottam a programot kört rajzolni, finom felbontású grafikát nyomtatni, sőt egy közepes képességű képernyőszerkesztőt is írtam. Megoldottam a program futás közbeni megszakítását. Még hátra van a LOAD szó rendellenességeinek megszüntetése.

GARAY LÁSZLÓ

A MAX-FORTH programozási rendszer

A Commodore 64 személyi számítógép BASIC V2 nyelvével bizonyos feladatokat még a forgalomban lévő BASIC bővítések és BASIC fordítók alkalmazásával is csak körülményesen és lassan lehet megoldani.

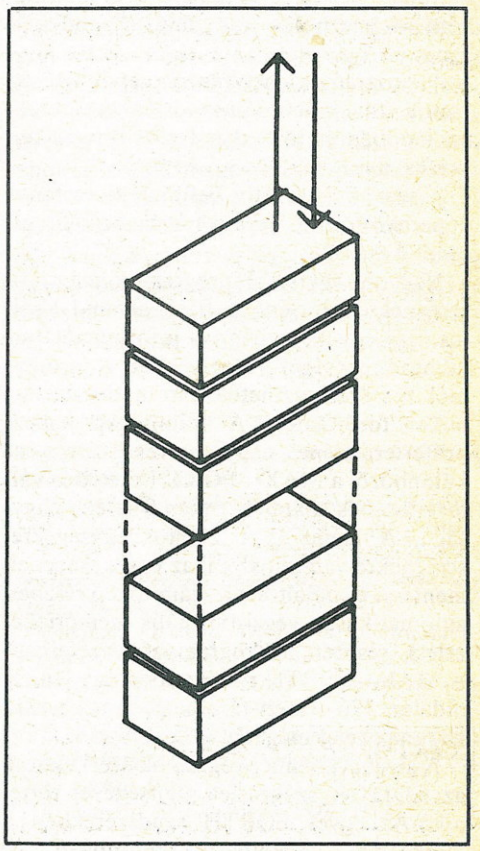
A BASIC nyelv alternatívájaként a programozók számára alacsony és magas szintű programnyelvek egyaránt rendelkezésre állnak. Ezek között sajátos helyet foglal el a FORTH.

A FORTH olyan negyedik generációs programozási nyelv és egyben operációs rendszer, amellyel a strukturális programozás legfejlettebb változata, a moduláris programozás valósítható meg. Compiler nyelv, mellyel interaktív (párbeszédes) módon lehet dolgozni. Egyesíti az Assembler nyelv gyorsaságát és rugalmasságát, a PASCAL strukturált programozhatóságát és a BASIC interaktivitását.

A FORTH láncolt kódú nyelv, ahol az új szavakat — melyek a BASIC parancsainak, függvényeinek FORTH megfelelői — a meglévő szavakkal lehet definiálni. A lefordított szavak az alapszókészlettől meg nem különböztethető módon beépülnek a szótárba, így minden FORTH program lényegében egy-egy új FORTH nyelv létrehozását jelenti. Egy nagyobb program megírása esetleg több száz új szó definiálásával jár. Például az ismert Calc Result program mintegy 1600 FORTH szóból tevődik össze.

A nyelvnek e rendkívüli rugalmassága követelte meg, hogy fig—FORTH, FORTH—79 és más, egymáshoz közel álló standardoknak megfelelően szabványosítsák, és ezzel elkerüljék a bábeli zűrzavart. A Magyarországon ez ideig beszerezhető Commodore FORTH nyelvek (M—FORTH, FORTH—64) megfelelnek e szabványok előírásainak, nem tartalmaztak rendszer- és gépspecifikus szavakat (grafika, zene), a felhasználóknak le kellett mondaniuk a BASIC-ben megszokott kényelmes sztringkezelésről, matematikai függvényekről, lebegőpontos műveletekről. Valószínűleg ez az oka, hogy a FORTH nyelv nem vált olyan széles körben használttá, mint ahogy értékei indokolnák.

Alapvetően megváltozott a helyzet a Mikrosystem által kifejlesztett MAX—FORTH 85 fejlesztőrendszer megjelenésével, mely a standard FORTH minden előnyének megtartása mellett, a rendszerben definiált mintegy 800 szó segítségével biztosítja a kényelmes és hatékony programozás lehetőségét.



1. ábra. A FORTH adatverem működésének elve. A FORTH a bemenő adatokat az adatveremből várja, a kimenő adatokat az adatveremben helyezi el. A hasábok a veremben tárolt adatokat jelképezik

A MAX—FORTH programozási rendszerben a fig—FORTH és a FORTH—79 standard szókészletét programfejlesztést segítő szavakkal (nyomkövetés, visszafordítás), speciális alkalmazási területet felölelő szavakkal (sztringkezelés) és gépspecifikus szavakkal (grafika, zene) bővítették.

A következőkben a MAX—FORTH rendszernek azokról a bővítéseiről lesz szó, amelyek a standard FORTH nyelvekben nem fordulnak elő.

A programok szerkesztése és tárolása

A FORTH számos tekintetben különbözik a BASIC-től. Ezek egyike az a mód, ahogy a programok tárolása és végrehajtá-

sa történik. A BASIC használata során a program forráskód alakban kerül tárolásra, a program futása során az interpreternek minden utasítást végrehajtható gépi formátumba kell interpretálni. Mivel a Commodore 64 BASIC nyelve a szintaktikai vizsgálatot is futás közben végzi, érthető az így megírt programok viszonylagos lassúsága. A FORTH rendszerek a forráskódot előre lefordítják, fordításkor végzik a minimális szintaktikai vizsgálatot arra vonatkozóan, hogy a felhasznált elemek korábban már definiált szóként, számként vagy sztringként értelmezhetők-e, így futásuk rendkívül gyors.

A programok megírására két lehetőség van: a szavak definiálása történhet interaktív módban és történhet forrás-screeneken szerkesztve.

A szavak interaktív definiálása esetén a képernyő törlése után a forráskód nem áll rendelkezésre.

Nagyobb FORTH programok megírása screenekben történhet. Ezek olyan 1 kbájtos, egy-egy képernyőnyi információt tartalmazó egységek, amelyekben a programok megszerkeszthetők, tárolhatók és bármikor fordíthatók. A számítógép lemezpufferterületének nagysága rendszerenként különböző, a MAX—FORTH esetében változtatható is, alaphelyzetben 8 screen nagyságú. A lemez is 1 kbájtos egységekre, screenekre van felosztva, az egyes screenek mentését és betöltését a számítógép részben automatikusan végzi (virtuális memóriakezelés), részben a programozó vezérelheti. A MAX—FORTH rendszerben egy lemezoldalon 170 screen tárolható, ami kb. 170 kbájtnak felel meg.

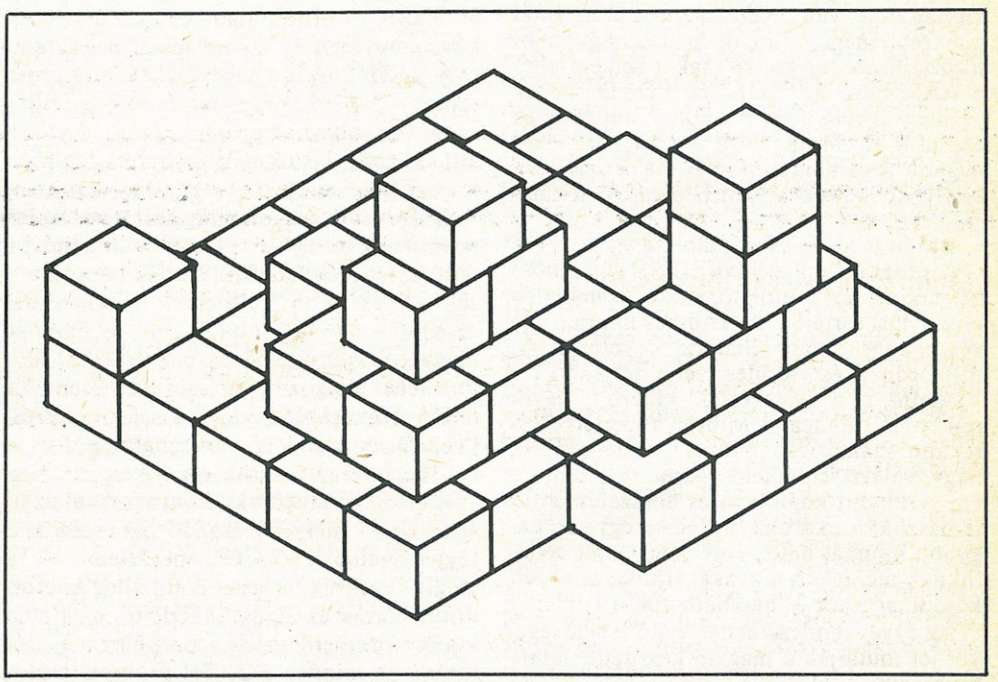
A screenekben a programok szerkesztése az EDITOR szókészlet segítségével történik. Az egyes FORTH rendszerekben a szerkesztés megoldása különböző, a MAX—FORTH az általam ismertek közül a legszellemesebb. A rendszer a Commodore képernyőszerkesztő programot használja, ugyanazt, mint a Commodore BASIC. Ez lehetővé teszi, hogy a felhasználó a képernyő tetszőleges pontján vihessen be adatokat, és azt a sort szerkessze, ahol a kurzor található. Az egyes sorok sorszámai [1), 2), 3) stb.] FORTH szavakként vannak definiálva, melyek a RETURN billentyű lenyomásakor az aktuális sort a bemeneti pufferbe viszik. A definiált editáló szavak segítségével a screenek szerkesztése, másolása, törlése, listázása könnyűszerrel megoldható. Az egyes screeneken belüli editálás, vagyis az egyes programsorok törlése, másolása, széthúzása ugyancsak kitűnően megoldott.

A megírt programok hibáinak megkeresését a MAX—FORTH rendszerben több hasznos segédprogram könnyíti. A rendelkezésre álló decompilerrel a felhasználó a rendszerben előforduló bármelyik magas szintű szót, a rendszer magját képező FORTH Kernel szavait is vissza tudja fordítani.

A nyomkövetési szolgáltatás igénybevételével a fejlesztők az újonnan definiált FORTH szavakban elkövetett hibákat kereshetik meg.

A MAX—FORTH rendszer a standardnak nevezhető forrás-screenek formájában való tároláson kívül a Commodore forrásfájlként való programtárolást is megoldotta, így forrásprogramok lemezen és szalagon egyaránt tárolhatók.

A rendszerben definiált szavak lehetővé teszik a FORTH alkalmazási programok tervezőjének, hogy a MAX—FORTH segítségével kifejlessze saját FORTH rendszerét, majd azt speciális „alkalmazási" programfájlként elmentse. Ha ezt a programfájlt betöltik és futtatják, az alkalmazási szó azonnal végrehajtódik. A felhasználó számára a FORTH rendszer nem elérhető, nem „listázható", így az alkalmazási program a FORTH rendszer jogvédelmét nem sértve árusítható, és megoldódik egyben a program titkosítása is.



2. ábra. A FORTH programok felépítése. A hasábok a programban definiált szavakat jelképezik. A program szavai párbeszédes üzemmódban is hívhatók, az egyes szavak átírhatók, kicserélhetők

Assembler, gépi kód

A felhasználók új szavakat a meglevő szavak segítségével, a gépi kódok szótárba történő közvetlen beírásával és a MAX—FORTH rendszerbe beépített, rendkívül hatékony, és a csak a magas szintű nyelvekhez hasonlíthatóan könnyen kezelhető Assembler segítségével végezhetik. A nyelv rendkívüli rugalmasságára jellemző, hogy a magas szintű FORTH nyelv és az Assembler nemcsak egy programon, hanem egy soron belül is keverhető!

A FORTH moduláris felépítése lehetővé teszi, hogy a felhasználók programjaikat ezen a nyelven fejlesszék ki, majd ha kell, a program felgyorsítása érdekében az Assemblert használva írják át azokat a szavakat, melyek gépi kódú futtatása szükséges, és mindezt a program struktúrájának megváltoztatása nélkül!

Az Assembler programozás FORTH környezetben igen egyszerű. Az Assembler szavainak beírása közvetlenül, interaktív módon és screenek segítségével egyaránt történhet. Fordításkor a gépi kód beszerkesztődik a szótárba, így a program futtatásakor nincs szükség az Assembler szótárra, amely a MAX—FORTH rendszerből törölhető is.

A FORTH—Assembler az adattárolásra ugyanazt az adatvermet használja, mint a FORTH, így a BASIC esetében szokásos nehézkes adatátadás elmarad.

Az Assembler programok a FORTH-hoz hasonlóan strukturáltak, az IF . . . ELSE . . . . . THEN, BEGIN . . . UNTIL, BEGIN . . . WHILE . . . REPEAT, BEGIN . . . AGAIN programstruktúrák állnak rendelkezésre.

A MAX—FORTH Assembler gépi kódú szubrutinok és makrók létrehozását is lehetővé teszi.

A rendszerben definiált szavak segítségével külső gépi rutinok is hívhatók. Az adatok itt is átadhatók vermen keresztül.

Vezérlési struktúrák

Egy programozási nyelv intelligenciája többek között vezérlési struktúráinak fejlettségén keresztül mérhető.

A FORTH nyelvek lehetőségeit minden BASIC-programozó irigyelheti, hiszen az IF . . . ELSE . . . THEN feltételes elágaztatáson kívül a BEGIN . . . UNTIL, a BE-

GIN... AGAIN, a BEGIN... WHILE... REPEAT határozatlan ciklusszervezési lehetőségek éppúgy megtalálhatók, mint a DO...LOOP határozott ciklusszervezés. A MAX—FORTH rendszerben a standard vezérlési szavakon kívül a CASE... ... OF... ENDCASE programvezérlő struktúra is definiálva van, amely — hasonlóan a BASIC ON GOTO, ON GOSUB utasításához — egy érték alapján műveletek egy csoportjának kiválasztását teszi lehetővé.

Hozzáférés magas RAM címekhez

A MAX—FORTH rendszerben speciális kiterjesztő szavak állnak rendelkezésre a RAM azon 12 kbájtos részének eléréséhez, amelyek a Kernel ROM és a B/K memórialeképzési területek alatt találhatók. Ezzel a felhasználó a számítógép tártartalmának ⅓-ét meghaladó, a BASIC-ből körülményesen hozzáférhető memóriaterülethez jut, melyben változókat vagy tömböket tárolhat.

Megszakítás, sztringkezelés

A magas szintű nyelvek általában nem teszik lehetővé a hardver megszakítások formájában történő vezérlését. A MAX—FORTH-ban ez mind alacsonyabb szinten (gépi nyelven), mind magas szinten (FORTH-ból) lehetővé válik.

A standard FORTH nyelvekkel ellentétben a MAX—FORTH-ban a sztringkezeléssel kapcsolatos szavak is definiálva vannak, melyek a felhasználó számára lehetővé teszik a BASIC-ben ismert sztringoperációk végrehajtását.

Képernyőkezelés, grafika

Ellentétben a C64 BASIC-kel és a standard FORTH nyelvekkel, a MAX—FORTH rendszer mind a képernyőkezelést, mind a nagy felbontású grafikák készítését magas szinten támogatja.

A képernyőkimenet vezérlőszavai segítségével lehetővé válik a kurzor pozicionálása, helyének meghatározása, a képernyő teljes és részleges törlése. A képernyő nagy felbontású grafikus és normál szöveges részekre osztható, a felhasználó határozhatja meg, hogy a képernyővágás melyik sornál történjen.

A definiált szavak mentesítik a programozót a VIC grafikai chip kezelésével járó fáradság nagy részétől, a felhasználónak nem kell foglalkoznia a chip-címek részleteivel, a hardverregiszterekhez való illesztéssel.

A rendszer magasabb szintű grafikai szavai segítségével a felhasználó nagy felbontású vonalakat, íveket, ellipsziseket, köröket, egyszeresen és többszörösen tükrözött képeket állíthat elő.

Hasonlóképpen megoldott a bittérképterület beállítása, a karaktermemória helyének megadása, illetve változtatása is.

A nagy felbontású grafikai szavak végrehajtási sebessége a Simon's BASIC hasonló grafikai parancsainál általában nem lényegesen kisebb, annak ellenére, hogy ezeket részben FORTH nyelven írták meg. Professzionális grafikai felhasználás esetén természetesen a szükséges szavak FORTH—Assemblerben átírhatók.

A MAX—FORTH rendszer a sprite-okkal kapcsolatos szavakat is tartalmazza. Egyszerre 8 sprite megjelenítésére van lehetőség. Ugyancsak a felhasználók rendelkezésére áll egy egyszerűen használható sprite-editor.

Zenei alkalmazás

A kitűnő zenei adottságokkal rendelkező Commodore 64 zenei programozása is rendkívül körülményes a BASIC nyelvben. A standard FORTH nyelvek egyáltalán nem tartalmaznak zenei szavakat.

A MAX—FORTH hang alrendszerének szavai nagymértékben megkönnyítik és leegyszerűsítik a Commodore 64 SID zenei chipjének programozását, sőt a rendszerhez kapcsolt külső SID chipek vezérlését is.

A Commodore 64 zenei képességének és a MAX—FORTH hang alrendszerének teljes bemutatására e cikk keretében nincs mód. A felhasználó által egyszerűen állítható frekvencia, hangerősség, szűrőfrekvencia, rezonancia, szűrő, felhangzás, elhalkulás, hangimpulzus-szélesség, kitartás, elengedés, hullámforma stb. értékek lehetővé teszik a számítógép professzionális zenei felhasználását. Zenei alkalmazásra minden valószínűség szerint az eredetileg is vezérlésre készült FORTH-nál megfelelőbb programozási nyelvet nem találhat a felhasználó.

A rendszer egy zenei editort is tartalmaz, mely lehetővé teszi, hogy a felhasználó háromszólamú zenét írjon akár interaktív módon, akár lefordított FORTH definíciók formájában.

Matematikai rutinok — lebegőpontos számítások

A standard FORTH nyelv a 8, 16 és 32 bites előjeles és előjel nélküli egész számokkal végzett alapműveleteket, veremoperációkat és tárműveleteket, az egész számokkal végzett aritmetikát támogatja.

A számítási feladatok döntő többsége egész számokkal megoldható. A mértékegységet úgy kell megválasztani, hogy tizedesjegy használatára ne legyen szükség. Ebben az esetben az adatbevitel és az adatmegjelenítés tetszőleges mértékegységben, lebegőpontos formában is történhet, az adatok egész értékké konvertálását a program végzi.

A MAX—FORTH rendszer lebegőpontos számítások végzésére alkalmas szókészlettel is rendelkezik, melynek segítségével a felhasználó lebegőpontos konverziókat, matematikai alapműveleteket, szögfüggvényszámításokat végezhet.

A lebegőpontos számításokat a FORTH nyelvtől és a FORTH programozóktól is idegenek, ez érezhető a MAX—FORTH-ban definiált lebegőpontos rutinok gyorsaságán és minőségén is. Amennyiben a felhasználónak pontos és gyors lebegőpontos számításokra, matematikai függvényekre van szüksége, ezeket a rutinokat FORTH vagy FORTH—Assembler nyelven definiálnia kell.

A MAX—FORTH alkalmazási lehetősége

A MAX—FORTH a ma Magyarországon a Commodore 64 számítógépre beszerezhető legjobb FORTH nyelvnek tekinthető. A vezérléstechnika, automatika ideális nyelve, de mindenütt jól felhasználható, ahol az eddigi magas szintű programozási nyelvek nehézkesnek és lassúnak bizonyultak. Gyors lebegőpontos számítások végzésére csak a rendszer ez irányú fejlesztése esetén javasolható. Hála a MAX—FORTH teljes nyitottságának, ez minden valószínűség szerint meg is oldható.

FEKETE ZOLTÁN

A téma iránt érdeklődők számára a következő szakirodalmat ajánljuk:

Lipovszki Gy.—Subai L.—Beszeda T.: FORTH programozási rendszer és nyelv. LSI ATSZ, 1985.

Pataki E.: Mit tud a FORTH programozási rendszer? Mikroszámítógép Magazin, 1983. évi szám, 12—14. oldal.

Forgács T.: Ismét a FORTH-ról. Mikroszámítógép Magazin, 1984. 4. szám, 12—14. oldal.

MAX—FORTH 85. FORTH fejlesztőrendszer. Microsystem Softinvest, 1985.

E. Floegel: FORTH Handbuch. Hofacker, 1982.

Monadjemi: Das Trainingsbuch zu FORTH. Data Becker, 1984.

K. Knecht: Einführung in FORTH. M. and T. 1984.

P. M. Chirlion: Der Einstieg in FORTH. M. and T. 1984.

Strukturált programtervezés

Előző cikkünkben ismertettük azokat a vezérlőszerkezeteket (szekvencia, szelekció, iteráció), amelyek alkalmazásával egy strukturált programszerkezet felépíthető. A tervezés első lépése a program szerkezetének, vázának meghatározása; a programban szereplő utasításokat és feltételeket erre a vázra illesztjük a tervezés későbbi lépéseiben. Most M. A. Jackson nyomán bemutatjuk, hogy a program által feldolgozandó, illetve feldolgozott adatok szerkezetének elemzése hogyan segíthet a program szerkezetének előkészítésénél.

Az adatok szerkezetének elemzése azt jelenti, hogy meghatározzuk, hogy az adat hogyan épül fel más, elemibb adatokból. Összetettebb adatot elemibb adatokból háromféleképpen építhetünk fel (az itt használt „felépítő" operátorok megfelelnek a strukturált programozás három vezérlőszerkezetének).

Az A adat, a B, C és D adatok *szekvenciája,* ha az A adat a B, C és D adatokból áll, ebben a sorrendben. Az adatszerkezeti szekvencia ábrázolása megegyezik a programszerkezeti szekvencia ábrázolásával *(1. ábra).*

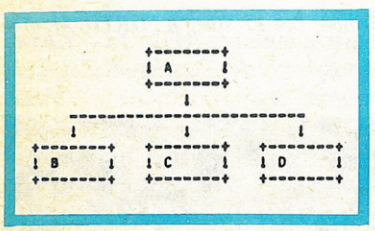
Az A adat a B, C és D adatok *szelekciója,* ha az A adat vagy a B adat vagy a C adat vagy a D adat. A „vagy" szót itt kizáró értelemben használjuk, azaz az A adat a B, C és D adatoknak csak egyike lehet. Az adatszerkezeti szelekció ábrázolása megegyezik a programszerkezeti szelekció ábrázolásával *(2. ábra).*

Az A adat B típusú adatok *iterációja,* ha az A adat B típusú adatok (esetleg üres) halmaza. Az adatszerkezeti iteráció ábrázolása megegyezik a programszerkezeti iteráció ábrázolásával *(3. ábra).*

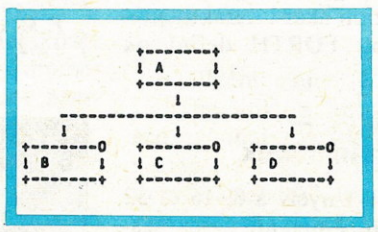
Egy adat szerkezetének meghatározásánál addig kell az adatot elemibb adatokra bontani, amíg olyan adatokhoz jutunk, amelyek feladatunk szempontjából már elemiek, azaz nem rendelkeznek további szerkezettel. Példaképpen a *4. ábrán* bemutatjuk egy olyan adatállomány egyszerűsített szerkezetét, amelyben a tételek egy könyvtárban tárolt egyedre (könyv, folyóirat) vonatkoznak.

A Jackson-féle programtervezési módszer szerint a programok szerkezetei megfelelnek a feldolgozandó, illetve az e program által készített adatok szerkezeteinek. Ez a megfelelés sok esetben megegyezés, azaz a programszerkezet megegyezik valamilyen adat szerkezetével. Egy ilyen esetet mutatunk be a következő feladat megoldásával.

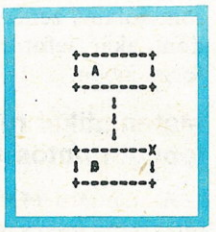
1. ábra



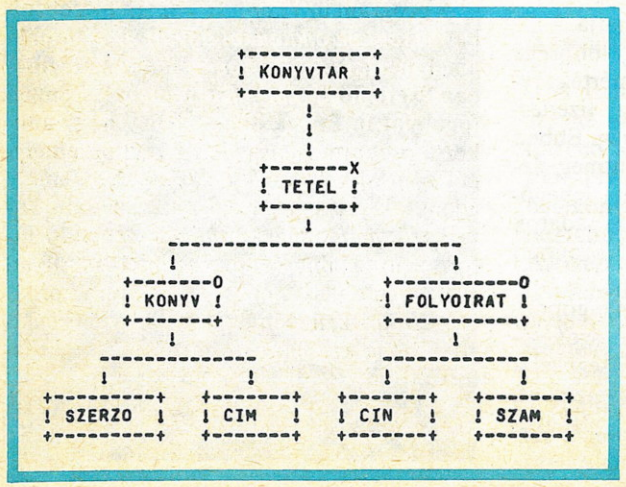
2. ábra



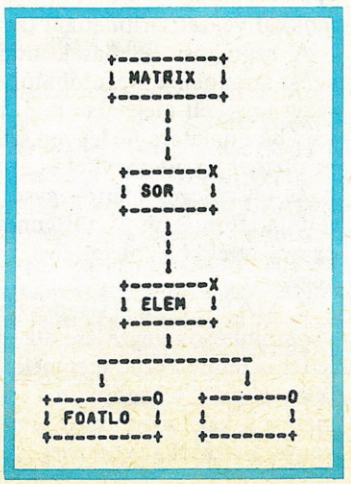
3. ábra



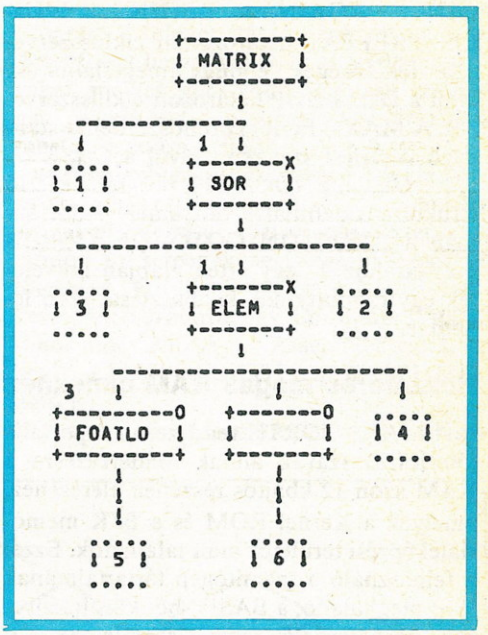
4. ábra



5. ábra



6. ábra



7. ábra

```
      FOR i=1. TO 10
      FOR j=1  TO 10
      IF  i=j THEN a (i,j)= 1:GOTO 100
      LET a(i,j)= i*j
100   NEXT j
      NEXT i
```

2. feladat. Írjunk egy olyan programrészt, amely az *a* 10×10-es mátrix elemeinek értéket ad úgy, hogy az i-edik oszlop j-edik eleme az i*j értéket tartalmazza, kivéve a főátlóban levő elemeket, amelyeknek nullát kell tartalmazniuk.

A mátrixnak mint adatnak a szerkezetét az *5. ábra* mutatja. A programnak ugyanez a szerkezete. A *6. ábrán* látható az elemi tevékenységekkel és feltételekkel kiegészített programszerkezet. Az elemi tevékenység BASIC nyelven:

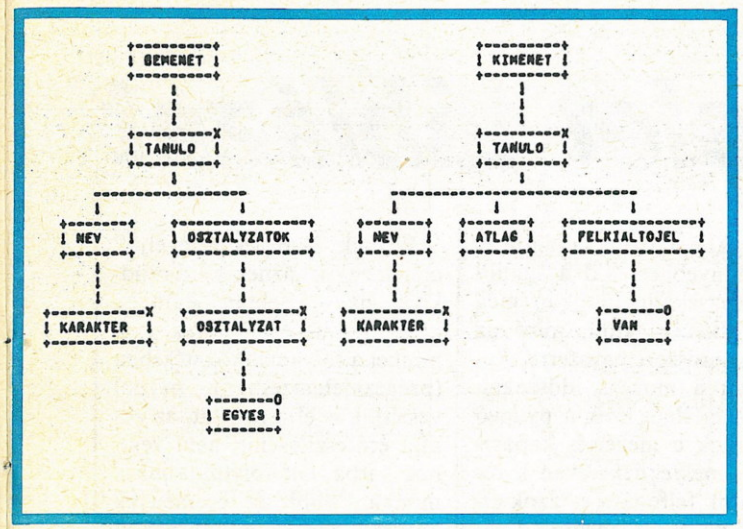
1 LET i=1
2 LET i=i+1
3 LET j=1
4 LET j=j+1
5 LET a(i,j)=Ø
6 LET a(i,j)=i*j

A feltételek:

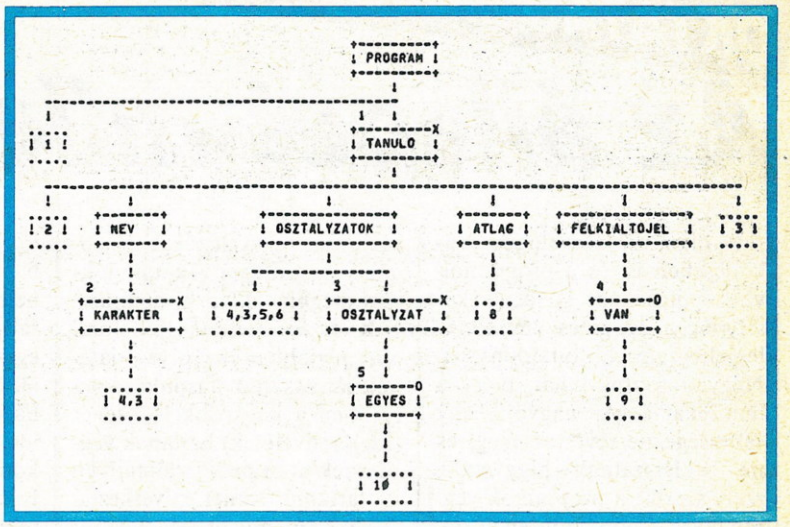
1 i=<1Ø
2 j=<1Ø
3 i=j

A programrészlet BASIC kódját a *7. ábra* tartalmazza. A kód írásakor felhasználtuk a FOR utasítás által nyújtott egyszerűsítési lehetőségeket.

Az előző cikkben ismertetett feladat megoldásában a bemenő adat szerkezete



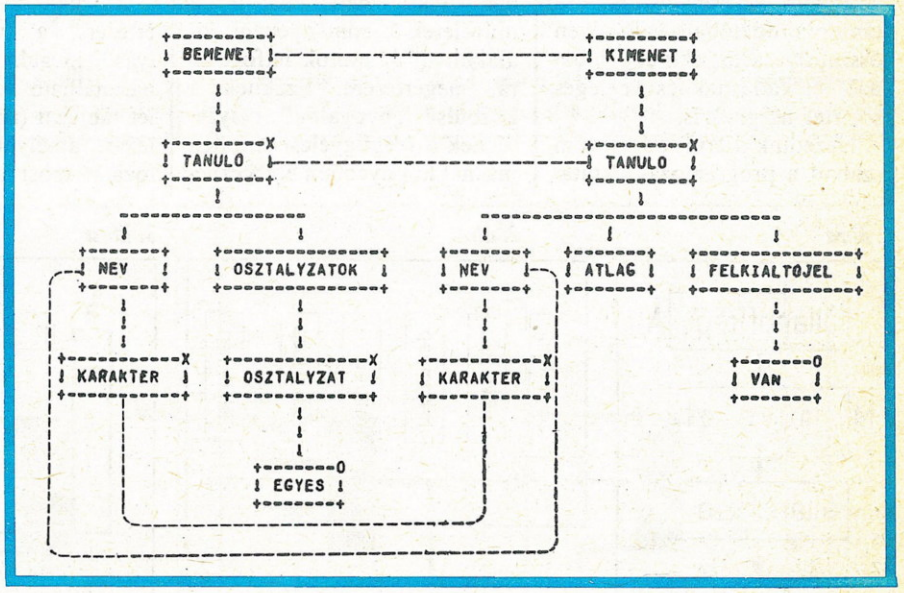
8. ábra



9. ábra

```
100 REM átlagok kiiratása
110 LET i=1
120 IF i>LENx$ THEN STOP
130 LET n$=""
140 IF x$(i TO i)=":" THEN GOTO 160
150 LET n$=n$+x$(i TO i): LET i=i+1:
    GOTO 140
160 LET n$=n$+x$(i TO i) LET i=i+1:
    PRINT 'n$;: LET ossz = Ø
    LET s=Ø:LET egyes = Ø
170 IF x$(i TO i)="." THEN GOTO 200
180 IF x$(i TO i)="1" THEN LEST egyes=egyes+1
190 LET ossz=ossz+CODEx$(i TO i)- CODE "Ø":
    LET s=s+1:LET i=i+1: GOTO 170
200 PRINT TAB 22; ossz/s;
210 IF egyes > Ø THEN PRINT "!";
220 LET i=i+1:GOTO 120
```

10. ábra



11. ábra

volt a program szerkezete is. Előfordulhat azonban az is, hogy a program egyik adat szerkezetével sem egyezik meg, hanem egy olyan „bővebb" szerkezet, amely részként minden adat szerkezetét magába foglalja: a programszerkezet az adatszerkezetek „burkolója". A következő feladattal egy ilyen esetet mutatunk be.

3. feladat. Az x$ karakterváltozó tanulók neveit és osztályzatait tartalmazza. A tanulók nevét az osztályzatoktól kettőspont választja el. Az egy tanulóra vonatkozó sorozatot pont zárja le. Készítsünk olyan programot, amely leírja a tanulók nevét és az osztályzatok átlagát, és ha a tanuló osztályzatai között egyes osztályzat is előfordult, az átlag után egy felkiáltójelet is ír. Feltehetjük, hogy minden tanuló legalább egy osztályzatot kapott, és hogy az x$ változó tartalma hibátlan.

A feladat be- és kimenő adatainak szerkezetét a *8. ábra* mutatja. Az elemi tevékenységek és feltételek számaival kiegészített programszerkezetet a *9. ábrán* láthatjuk. Az elemi tevékenységek a ZX-Spectrum BASIC nyelvén:

1 LET i=1
2 LET n$=""
3 LET i=i+1
4 LET n$=n$+x$(iTOi)
5 PRINT 'n$;
6 LET ossz=Ø:LET s=Ø:LET egyes=Ø
7 LET ossz=ossz+CODEx$
iTOi—CODE "Ø": LET s=s+1
8 PRINT TAB22; ossz/s;
9 PRINT "!";

A feltételek:
1 i=<LENx$
2 x$(iTOi)<>":"
3 x$ iTOi<>"."
4 egyes>Ø
5 a(iTOi)="1"

A BASIC kódot a *10. ábra* tartalmazza.

Amikor a programszerkezetet adatszerkezetekből származtatjuk, a következő szabályok megtartására kell ügyelnünk.

1. A programszerkezetnek minden olyan „dobozt" tartalmaznia kell, amely valamelyik adatszerkezetben előfordult.

2. A programszerkezet nem tartalmazhat olyan „dobozt", amelynek nincs megfelelője valamelyik adatszerkezetben.

3. Az adatszerkezetekben egymásnak megfelelő „dobozokat" a programszerkezet csak egy példányban tartalmazhatja. Két „doboz" megfelel egymásnak az adatszerkezetekben, ha együtt kell őket feldolgozni, vagy ha az egyiknek megfelelő adatból készült a másiknak megfelelő adat. Ezeket a megfeleltetéseket feladatunk esetében a *11. ábra* mutatja.

A program tervezése folyamán előfordulhat, hogy olyan szerkezethez jutunk, amelynél a szelekcióban, illetve iterációkban szereplő feltételeket nem tudjuk megfogalmazni. Ennek oka az lehet, hogy amikor a program futása eléri e pontot, amelynél döntést kell hozni a szelekció valamely ágának kiválasztására vagy az iteráció következő végrehajtására, nincs elegendő információ a döntés meghozatalára. Ilyen esetekben a visszalépéses (backtrack) programozást alkalmazhatjuk. A sorozat következő része ezt a programozói technikát ismerteti.

MOLNÁR MÁTÉ

(Folytatjuk)

Az állapot-művelet táblázat

A digitális számológép működésében is, és a programok végrehajtásában is a szakaszosság, a lépegetés a működés legjellegzetesebb tulajdonsága. Nagyon fontos tehát, hogy a tanulókat a gép nagyon nagy sebessége ne tévessze meg, és ne a látszatnak higgyenek, mely szerint a programok végrehajtása során minden történés sima, fokozat nélküli. Ez az illúzió, az eseményeknek a tanulók tudatában való ilyen összefolyása az oka sok hibának, elakadásnak és végleges sikertelenségnek is.

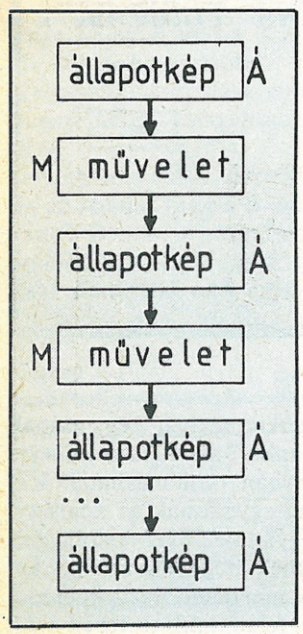
Nézetünk szerint addig nem szabad a programozás oktatásában az első lépéseknél tovább menni, amíg a tanulókban megbízhatóan ki nem alakult az a működésfogalom, ami legjobban az olyan játékok játszásához hasonló, amelyekben a játékosok lépéseket (ezek műveletek) hajtanak végre, ezek után pedig valamilyen időtartamú szünet következik.

Ezek a műveletek közötti szünetek alkalmasak (és egyben nélkülözhetetlenek is) a műveletek eredményeként kialakult új állapotok felfogására, megértésére. Ezeknek a közbülső „nyugalmi" helyzeteknek a megfigyelése minden másnál könnyebben elvégezhető. Ezekbe a köztes állapotokba könnyebben tud a tanuló belekapaszkodni, jobban meg tudja őket ragadni, mint az egész rendszert egyszerre, különösen a mozgás időszakaiban. Általában ezek a nyugvó időszakok a megértés kapaszkodói, melyeknek révén a folyamatok felfogása és azok értelmi követése megindul és megvalósul az elmében.

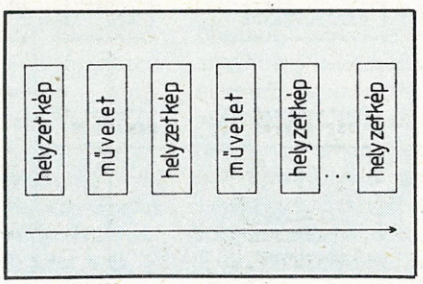
A szakaszos működés megértésére, a tudati „egybefolyás" megakadályozására jól használható az állapot-művelet táblázat (röviden Á-M táblázat), amelynek egyszerű változatát most itt szemléltetjük.

Vegyük észre és értékeljük ennek az eszköznek ezt a rendkívül nagy lélektani előnyét, hogy bárhol is zavarnak meg minket a folyamat követésében (programelemzésben), bárhol veszítjük is el a fonalat, az eddigi erőfeszítéseink nem vesznek kárba. Ott folytathatjuk a munkát, ahol az események követése kiesett a kezünkből, nem kell mindent elölről kezdeni, mert eddigi eredményeink nem ugranak össze, folynak szét, nem tűnnek el, hanem változatlanul megmaradnak, sőt azt a helyet sem kell megkeresni, hogy hol hagytuk abba a munkát, mert ez is elté-

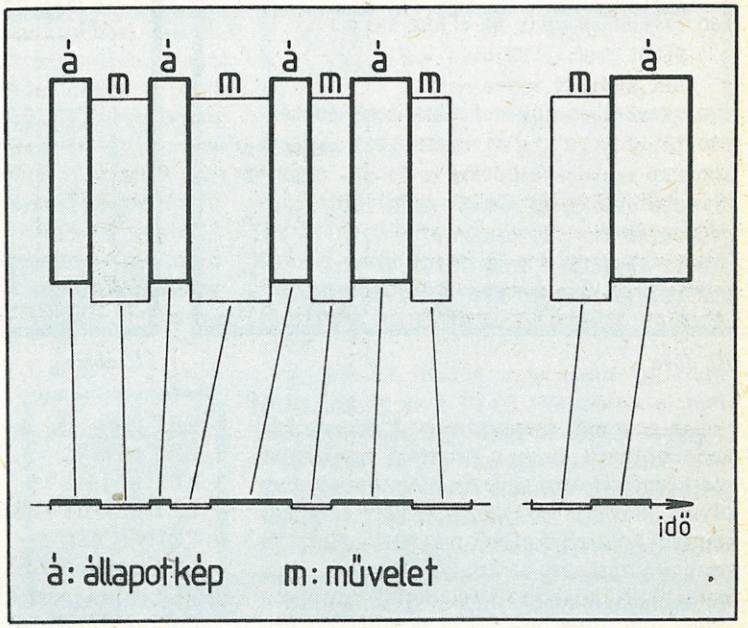
1. ábra



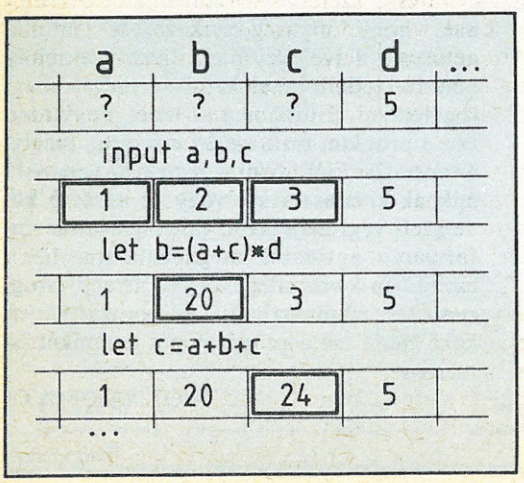
2. ábra



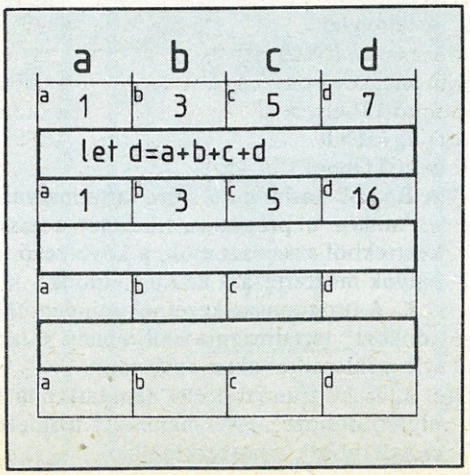
3. ábra



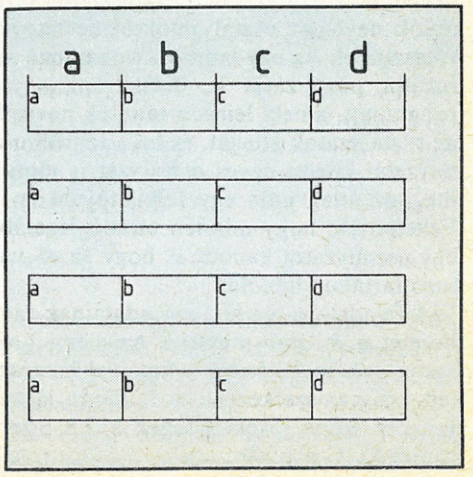
4. ábra



5. ábra



6. ábra



a	b	c	d
1	2	3	4

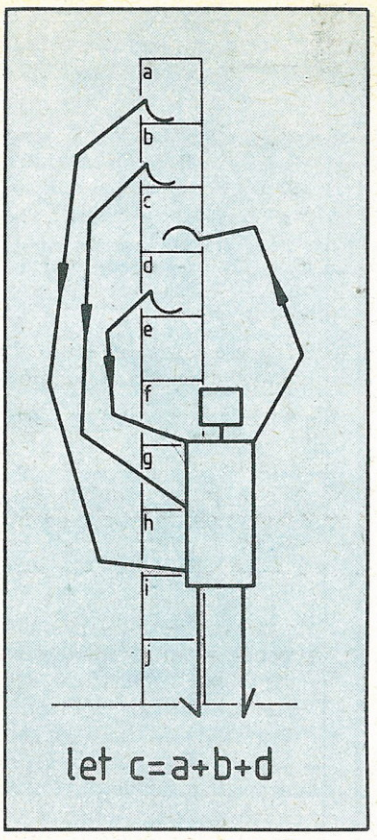
let c=(a+b)*c

a	b	c	d
1	2	9	4

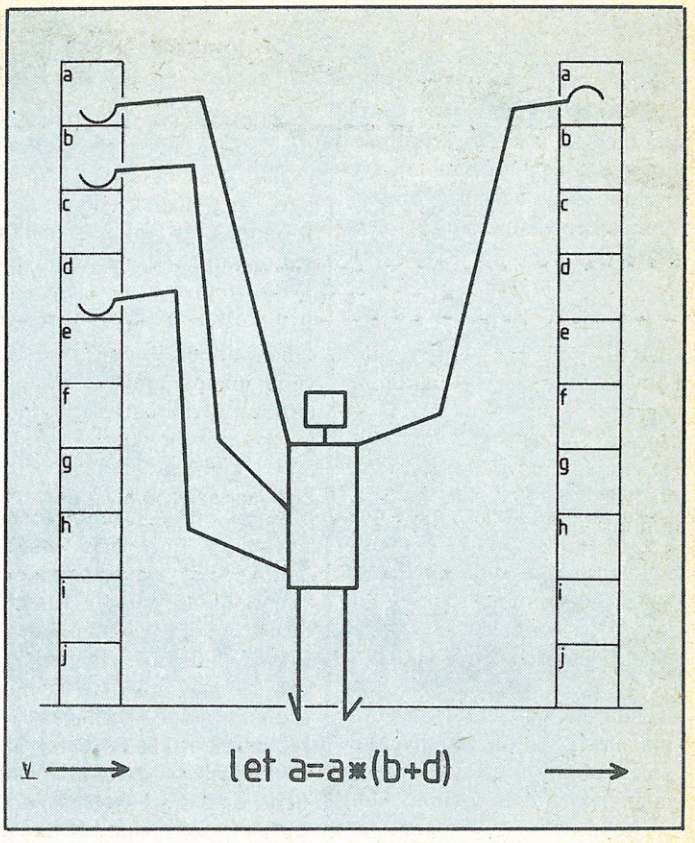
let a=(c-d)/b

a	b	c	d
2,5	2	9	4

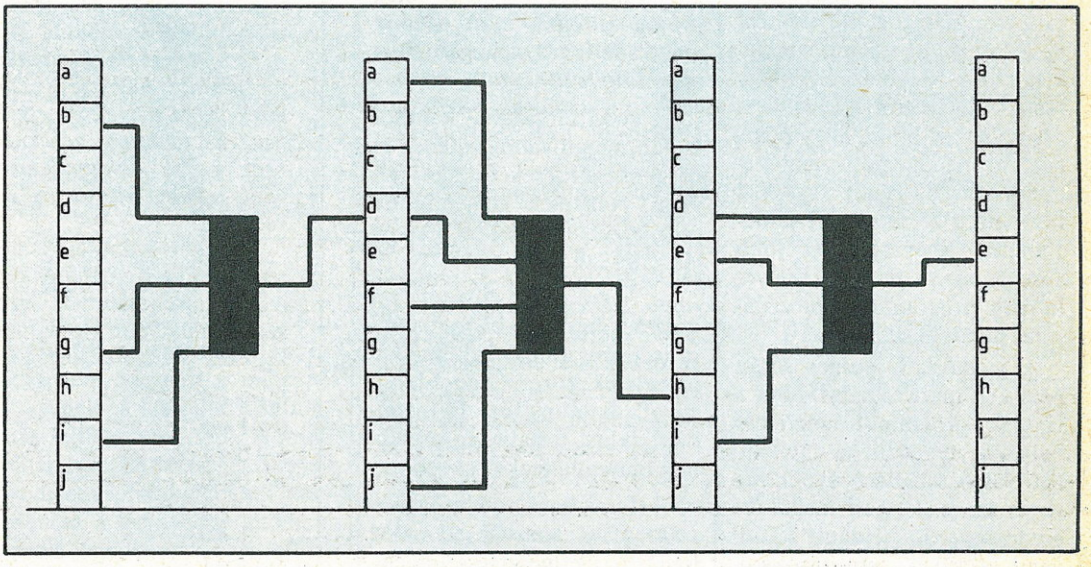
7. ábra



8. ábra



9. ábra



10. ábra

veszthetetlen és azonnal magától látható. Ez a tudat nagyon megnyugtató, csökkenti a tanuló izgalmát, kiküszöböli a feszültségeket, a szorongást. Nem kell reszketnie, hogy jaj mi lesz, ha elveszti a fonalat. Egyszerűen nem lehet semmit sem elveszteni, eltéveszteni is alig.

Az Á-M táblázat váltakozó rendben tartalmaz állapotokat és műveleteket. Többféle elrendezés, megjelenítésforma lehetséges, és így többféle haladási irány is. Lehet a táblázatot fentről lefelé, balról jobbra, de többféle iránnyal is szerkeszteni. Az áttekinthetőség követelményét azonban mindig a lehető legnagyobb mértékben célszerű kielégíteni.

Az *1. ábra* egy fentről lefelé haladó, a *2. ábra* pedig egy balról jobbra haladó Á-M táblázat szerkezetét szemlélteti. A haladási irányt mindkét esetben másképp jeleztük. Irányjelzésre a számozás is megfelel. Ha így az időbeliséget is hűen tudjuk leírni — például időtengelyre vetítéssel —, az Á-M táblázatot működéstörténetnek is felfoghatjuk *(3. ábra)*. Az ábrán az állapotok rendelkezésre állási időszakaszainak és a műveletek zajlási időszakaszainak egy nagyon egyszerű esetét szemléltettük. Természetesen előfordulhat, hogy egyes intervallumok átfedik egymást, és az is, hogy közöttük egy harmadik fajta (nem állapotrendelkezésre állási és nem is műveletvégzési) intervallum foglal helyet.

Az állapotkép és a helyzetkép kifejezések közül mindig az éppen kifejezőbbet kell használni. Mi itt nem teszünk köztük különbséget.

A *4.*, az *5.*, a *6.* és a *7. ábrán* egy-egy tömörebb táblázattípust mutatunk be, üresen és néhány szemléltető bejegyzéssel. A megváltozó tartalmú, helyesebben az új értéket kapó rekeszeket keretezéssel emeltük ki. A *8. ábrával* egy-egy műveleti lépés — tapasztalatunk szerint nagyon hasznos — antropomorf (helyesebben robotomorf) szemléltetésére adunk ötletet. A robot azt az utasítást kapta, hogy gondoskodjon arról, hogy c-ben a, b és d tartalmának összege legyen. A *9. ábrán* érzékelhető igazán az Á-M táblázat előnye, különösen az olyan értékadásoknál, ahol van olyan rekesz, amely előbb operandust, majd pedig operátumot tartalmaz.

Az egy sorba több utasítás írását az első időszakban célszerű mellőzni, később azonban ez is szemléltethető egymás után dolgozó robotokkal. Mivel a rajzi megjelenítés ilyen esetekben már kissé nehézkes, és az áttekinthetőség sem éri el az elvárható mértéket, e rajzok didaktikailag már nem kifizetődők.

A 9. ábra ékesszólóan szemlélteti azt, hogy nem mindig érdemes az oktatási célú modelleknél az oktatott rendszert szolgaian másolni. A számológépben egyetlen a, egyetlen b, egyetlen c stb. rekesz van. Mi eggyel több példányban ábrázoltuk ezeket, mint ahány utasítást ábrázolni kívántunk *(10. ábra)*.

Meg kell fontolni, hogy a pedagógiai cél érdekében mikor célszerű a gép fizikai valóságához illeszkedő, 8. ábrán szereplő modellt, és mikor az időbeliséget érzékeltető 9., illetve 10. ábrán használt modellt alkalmazni.

Hasznos gyakorló feladatokat lehet szerkeszteni, ha megadjuk egy Á-M táblázat M részét, és a tanulókkal töltetjük ki az Á részét és megfordítva, egy táblázat adott A részéhez kerestetünk műveleteket (utasításokat).

Egyszerű példáinkban minden sorban és minden oszlopban ugyanazok az állapotjellemzők (rekeszek) szerepeltek. Az ideális az lenne, hogy az

összes rekesz, regiszter helyet kapna. Ezt természetesen több okból is lehetetlen megvalósítani. Például a helyhiány gyakran még azt is kikényszerítheti, hogy egy-egy sorban, illetve oszlopban ne az előzővel azonos jellemzősorozatot szerepeltessük. Ilyenkor előnyös a 7. ábra utolsó Á sorában látható, szabadon kitölthető rekesznevekkel használható Á-M táblázatűrlap.

Megjegyezzük, hogy Á-M táblázatot magával a számológéppel is tudunk készíttetni. A formai megjelenés természetesen aszerint alakul, hogy mik a gépi adottságok, és ki menynyi energiát fordít a külalakra. Van, amikor nincs szükség teljes Á-M táblázatra, hanem elég annak csak Á része; ilyenkor csak az állapotképek egymásutánját jelenítjük meg. (A műveletek külön megjelenítésére a program listája kiválóan megfelel.) Ez az a pont, ahol fel kell hívnunk a figyelmet arra, hogy noha eddig az Á-M táblázat pedagógiai alkalmazásairól és pedagógiai értékeiről beszéltünk, a számológéppel készített Á-M táblázatról (vagy csak Á táblázatról) súlyos tévedés volna azt hinni, hogy pusztán egy pedagógiai szemléltetőeszköz a sok közül. A géppel készített Á-M, illetve Á táblázatoknál egyszerűbb és jobb eszközt nem ismerünk bonyolultabb programok kipróbálására, hibák megkeresésére.

A gyakorlati munka során nem kell mást csinálni, mint a vizsgált programot megfelelő helyeken teletűzdelni „feltételes" vagy „feltétlen" nyomtatási, és ha szükséges, megállást eredményező, például STOP vagy INPUT utasításokkal. Természetesen a ciklusváltozókra is ki kell terjednie figyelmünknek. A műveletek (műveletcsoportok) kiíratása nehezebben oldható meg, de a gyakorlatban ez nem szokott nélkülözhetetlen lenni. Általában elegendő, ha a műveletcsoportokba (elejére, végére, közepére, ahová kell) „nyomjelző" nyomtató utasításokat helyezünk. Így, házilag, nagyon könnyen, kiváló minőségű programfutás-követéseket lehet végezni, különleges „követő" programok igénybevétele nélkül. A sikeres próba, illetve a hibák kijavítása után az Á-M táblázatot nyomtató utasításokat természetesen el kell távolítani a programból, ez a munka azonban gondosságon, figyelmességen kívül (nehogy mást távolítsunk el) nem szokott mást igényelni.

E rövid kis ismertetés fő mondanivalóját egy példával foglalhatjuk össze legtömörebben. A mozivásznon, noha folytonos és fokozat nélküli változásokat látunk rajta, valójában nem mozog a kép, hanem képtovábbítási műveletek közti állóképek egymásutánja kelti ezt az illúziót, mert a képeket a nagy sebesség miatt nem tudjuk szétválasztani, azok egybefolynak. És a filmszínház esetében ez jó, ez a hasznos. A számológépben ugyanúgy műveletek közötti „állóképek" egymásutánjával van dolgunk, mint a filmvetítésnél. Az egymás utáni állapotokról kialakult elmebeli képek — még ha nem is mind tud kialakulni — a nagy sebesség miatt itt is egybefolynak. Ez azonban nemcsak hogy nem jó, hanem egyenesen káros is. Le kell tehát lassítanunk a „futást", hogy a különbözőket valóban különbözőknek tudjuk felfogni, mert enélkül maga a felfogás, a megértés válik illúzióvá.

Az Á-M táblázattal kapcsolatban még további érdekes és hasznos kérdéseket lehet felvetni. Néhányat megemlítünk ezek közül, amikre később még visszatérünk.

— A számolási folyamatgeneráló közismert kapcsolatábra is kiegészíthető úgy, hogy a műveletek közé „állapotdobozokat" iktatunk be. Ez azonban ciklust tartalmazó esetekben bonyodalmakat okozhat.

— Ismétlések, ciklusok követésére is hasznos az Á-M táblázat. Természetesen a ciklusváltozókat is explicite szerepeltetni kell.

— Mi a helyzet egyidejű folyamatok esetében? (Az Á-M táblázatnak ilyenkor is sok hasznát vehetjük.)

— A „lépegetős" viselkedés tanulmányozására — mint láttuk — az Á-M táblázat megfelelt. Folytonos állapotváltozás, folytonos műveletvégzés és ilyenek párhuzamos zajlása esetében is lehet valamilyen szemléltető módszert adni? Meglepő, de lehet.

— Van-e köze az Á-M táblázatnak az ún. állapottérhez? Van, mégpedig olyan, ami az oktatásban is kiválóan hasznosítható.

POGÁNY CSABA

Zöldségeskert

„Mit nevezünk a számítógépnél akkumulátornak? A: Olyan tárregisztert, ahol az aritmetikai és logikai műveletek eredményeit tartják. B: Nagy számítógépek áramellátását végzi. C: Az adatok sínjei. A helyes válasz: A."

(ÉLET ÉS TUDOMÁNY, 1985. 33. szám, 1050. oldal, Zsebiskola 14.)

És mi lenne az akkumulátor például többszörös szóhosszúságú műveletek esetében? És többcímű gépeknek nem volna akkumulátora, vagy minden tárrekesze az volna?

„S a gép számítási ismeretei mindössze néhány kettes számrendszerbeli egyszerű műveletre terjednek ki; ezek a következők:

$0+0=0$
$0+1=1$
$1+0=1$
$0=0$
$0\neq 1$
$1\neq 0$
$1=1$

A gép minden aritmetikai és logikai műveletet ezekre az alaplépésekre és a számjegyeknek egy helyértékkel jobbra vagy balra való eltolására vezet vissza." (ÉLET ÉS TUDOMÁNY, 1985. 33. szám, 1055. oldal, A „buta" számítógép)

A felsoroltak nem műveletek, hanem relációk. Ez csak hiba, a „buta" számítógép hibája. Biztosan azért buta. De hogy egy reláció hogyan lehet egy „buta" vagy akár okos számítógép alaplépése, az már rejtély.

„S a hetvenes évek elején már egy számítógép feldolgozó (processzáló) egységét egyetlen szilíciumlapra tudták integrálni . . ." (ÉLET ÉS TUDOMÁNY, 1985. 38. szám, 1215 oldal, Emberközeli számítógép)

Az emberközeli számítógép már processzálni is tud. A nyelvrontáshoz még nem ért, ezt egyelőre még mások végzik helyette. A processzálásával kapcsolatban kommentárolónkban több megjegyzésünk nincs, csak az, hogy ha valaki szétszed egy emberközeli gépet (embertávolit úgysem tud szétszedni), legyen óvatos a feldolgozóegység közelében, nehogy az esetleg úgy megprocesszálja, mint újságíró a szaknyelvet.

Végül egy háztáji termelésű zöldség. A µMagazin 1985/4. száma mellékleteként közölt Z80-as kódtáblázatban néhány, a szokásostól eltérő jelölés szerepelt. Az F8, FA, FC mezőkben levő feltételes utasítások feltételkódja a szokásos M (mínusz) betűjelöléstől eltérően N (negatív) betűjelöléssel szerepel. A 80-tól BF-ig terjedő mezőkben levő aritmetikai, logikai utasítások „egyszerűsített" írásmódban szerepelnek. Hiányzik az egyik operandusként és eredményként szereplő A regiszter. A szokásos jelölés sem következetes azonban. Az ADD, az ADC, az SBC utasításnál kiírják, az AND, az XOR, a SUB, az OR, a CP utasításnál pedig elhagyják a célregiszter jelölését. A különböző gépeken megvalósított assemblerek kisebb részletekben különbözhetnek, ezért mindig gondosan győződjünk meg a konkrét megvalósítások sajátságairól, és legyünk óvatosak a kódtáblázatokkal is.

A MINŐSÉGÜGY KÖZÜGY

SZÁMÍTÓGÉPEK MINŐSÍTÉSE

A számítástechnika kezdetén egy-egy ősgépre még nem lehetett azt mondani, hogy áru, abban az értelemben, ahogyan egy mai gép az. Mihelyt azonban az 1 elemű sorozatokból valódi (akár csak néhány) elemű sorozatok lettek, és több géptípust is gyártani kezdtek, a gép áruvá vált, és megjelent a piac is, az összes, más piachoz hasonló jellemzőjével.

A piacok életjelenségeiben az egyik főszereplő kétségtelenül a *minőség*, amely más tényezőktől függő fontossággal, de mindenben szerepet játszik. A gépek (most már termékek, áruk) minőségének kérdése, minősítésének módszerei ezért egyidősek magával a piaccal. Ez logikus, hiszen ha van piac, akkor lehet választani. És e választást lehet jól vagy kevésbé jól végezni, e választásban pedig a minősítésnek döntő szerepe van.

A számítástechnika rövid története során nagyszámú minősítő eljárást dolgoztak ki, amelyeket ha összegyűjtenénk, alkalmunk volna derülni a rengeteg furcsaságon, egyoldalúságon, mai szemmel nézve primitivitáson. Ezeket a régi piaci iránytűket azonban nincs jogunk mindenestül a középkori alkímia téves és zavaros tényekedései közé utalni. Inkább az alkímiának az értékes ágaihoz hasonlók, azokhoz, amelyekből a modern kémia kinőtt.

Nehogy azonban valaki is tévedés áldozata legyen, fel kell világosítani mindenkit, hogy a számítástechnikai minősítés tudománya ma még nagyon messze van attól, hogy a modern kémiához egzaktságban és sikerekben hasonlítható legyen. A minősítési módszerek „kultúrtörténete" nem túl gazdag, egy-két érdekes mozzanatot érdemes azonban ismertetni belőle.

A kapitalista piac mozgatója a verseny. Verseny pedig nincs minősítés nélkül. Minden gyártó igyekezett minél versenyképesebb terméket előállítani, és igyekezett versenyképességet fokozó minősítést szerezni árujának. Megszülettek tehát az összehasonlítások. Mivel már akkor nagyon összetett, sokjellemzős rendszerekről volt szó, egyes jellemzők fontosságát mások rovására észrevétlenül ki lehetett domborítani, úgy is például, ahogyan a gyártó igénye kívánta. A reklámtevékenység erősödése révén a piacon való — sosem volt — biztonságos eligazodás végképp illúzióvá vált.

És — megint csak a piac igényei — létrehozták a gépkiválasztással foglalkozó, tanácsadásból élő szolgáltatók rétegét. A különféle, jobbára már akkor is vitatható minősítési módszerek publikálása ezzel lelassult. Addig az alkimisták azt állították, hogy tudnak aranyat csinálni, és meg is mondták, hogy hogyan. Az új tanácsadói foglalkozás kialakulása után továbbra is azt állította mindegyik, hogy birtokában van az aranycsináló receptnek, de bolond lenne azt másnak elárulni. A minősítési szakértők saját üzletüket rontották volna, ha közzéteszik, hogy hogyan végzik munkájukat. Mivel ez a titkolódzás egyaránt jó volt a tudásnak is és a hozzá nem értésnek is az elleplezésére, megjelentek és hamarosan elszaporodtak e fontos területen a sarlatánok. Ez pedig nem használt a minősítés ügyének és közvetve a minősítés fejlődésének sem. A minősítés megbecsültségét tovább rontotta a szakemberek körében az a tény, hogy általában azok foglalkoztak vele, akik a szakma többi érdemi ágában nem váltak be. Az alkalmatlan, akinek a versenyen semmi esélye sem volt, egyszer csak feltűnt, mint versenybíró vagy zsűrielnök.

Érdekes a piac reagálása erre a dekadens jelenségre. Mivel a kezdeti időben a gépvásárlás mindig komoly beruházás volt, a helyes választáshoz az alapos minősítést mindenképpen el kellett végezni. Mivel a „szakértőkben" az említetteken kívül még azért sem lehetett megbízni, mert időközben majdnem mindegyik valamelyik cég (esetleg több cég) fizetett embere lett, a vásárlók a következő szellemes módon kerülték el a szakértő vámpírok csapdáit. A szóbajöhető összes cégtől ugyanolyan tartalmú ajánlatot kértek, felszólítva mindet a tényszerű adatközlésen kívül arra is, hogy emeljék ki termékük előnyeit a piacon levő más termékekhez képest. Így — mivel az árajánlatokat mindig megbízható profik készítették —, megbízható adatokhoz jutott a vásárolni szándékozó. Ha még ezek után versenytárgyalásra is meghívta a reménybeli eladókat, lényegében minden számára fontos adathoz hozzájuthatott.

A nagygépek részesedése a mai világban a darabszámot tekintve elenyészőre zsugorodott. Az imént említett, tipikus beruházói módszer nagyobb gépek vásárlásánál napjainkban is jó, és a számítástechnikában is egyeduralkodóvá vált. A beruházások rentabilitásával foglalkozó általános tudományoskodással „alátámasztott" döntési módszerek azonban a számítástechnikában sohasem tudtak gyökeret verni, úgy tűnik, másutt is egyre inkább kezdik hitelüket veszteni. A személyi gépeknek a számítástechnikában eddig példa nélkül álló óriási darabszáma a piacot lényegesen átalakította. A személyi gépek vásárlója vásárol, és nem beruház. Tehát kockázata kisebb. A kínálat viszont olyan nagy, olyan gazdag, hogy szinte lehetetlen benne eligazodni. A propaganda harsog, a reklámok vakítanak, úgyhogy a vásárló, ha megkérdeznénk, hogy mit óhajt, nem azt mondaná, hogy nagy teljesítményű, megbízható és olcsó gépet, hanem tiszta képet a piacról, hogy nyugodtan választhasson.

E tiszta kép azonban csak óhaj marad. A piac ugyanis forrongva alakul, változik. Az újdonságok naponta jelentkeznek, s helyezik más beállításba a néhány nappal előbb általánosan elfogadott ítéleteket. Élni viszont kell. Vásárolni is kell. Tehát választani is kell. Igen, de hogyan? A választás a viselkedés egyik fő komponense. Nem lehet hibázni csak helytelen választással. Egy bűn lehetséges csak: az adott körülmények között nem a legjobb viselkedésmód választása.

Aki él, nem tehet mást, mint pillanatról pillanatra újra és újra választ. Az értelmes lény elméjével minősít, és utána választ. De minden más is minősít, és aszerint választ. Valamilyen minősítési módszere tehát minden létezőnek van.

Ismerjük meg ezek közül a legegyszerűbbet!

A minősítendők számunkra fontos jellemzőit kiemeljük, a többit elhanyagoljuk. A jellemzők értékeit számokkal adjuk meg. E számokat pedig összeadjuk, összeszorozzuk, vagy valamilyen eljárás szerint feldolgozzuk. A feldolgozás eredményét — amely egy szám —, a minősítendőhöz csatoljuk, minőségét jellemző számként. És két minősítendő közül a nagyobb minősítőszámút nyilvánítjuk jobbnak.

Lássunk egy nagyon egyszerűsített példát!

A gép közvetlen elérésű tárkapacitását (T), lebegőpontos összeadási (Ö), szorzási (S) és osztási (O) műveleteinek sebességét és árát (Á) vesszük számításba, mást nem. T értékét számítsuk (például 8 bites) bájtokban, Ö, S és O értékét pedig úgy, hogy 1 másodperc alatt, ha csak ezeket csinálná a gép, hány műveletet végezne el belőlük. Á értékét forintban adjuk meg.

Az M számmal a gépet minősítjük. Legyen először

$$M = \frac{T + Ö + S + O}{Á}.$$

Ha egy gép esetében $T \approx 32\,000$ bájt, $Ö \approx 25\,000$ db, $S \approx 10\,000$ db, $O \approx 5000$ db, $Á = 30\,000$ Ft, akkor

$$M \approx \frac{72\,000}{30\,000} = 2{,}4.$$

Ha egy másik gépre $T \approx 64\,000$ bájt, $Ö \approx 50\,000$ db, $S \approx 20\,000$ db, $O \approx 5000$ db, $Á = 100\,000$ Ft, akkor

$$M \approx \frac{139\,000}{100\,000} = 1{,}39.$$

Melyiket vegyük meg? Ne siessük el a döntést! Olvassuk el a következő folytatást is!

µPROGRAMOK

VC20

Négy ötlet

1. Ha olyan programot akarunk futtatni a VC20-on, amely csak 3 kbájt memóriabővítést igényel, de nekünk 16 kbájt vagy ennél nagyobb bővítőnk van, a bővítő kihúzása, illetve a gép kikapcsolása nélkül a következőket írjuk be még a program betöltése előtt:

POKE 44,4:POKE1024,0
:POKE5630:POKE 648,30:NEW
(RETURN)

Ilyenkor eltűnik a kurzor. A RUN/STOP—RESTORE együttes megnyomása után adjuk be: SYS58238(RETURN). Most újból bejelentkezik a gép 6655 Byte Free felirattal, és máris használhatók a 3 kbájt bővítőt igénylő programok.

2. Ha bármilyen bővítőegység van a VC20-ba bedugva, és a gépet az alapkiépítésben szereplő memóriatartalomra akarjuk visszakapcsolni, akkor adjuk be a következő utasítássorozatot:

POKE642,16:POKE644,30
:POKE648,30:SYS64824

Ezután nyomjuk meg a RETURN gombot. Rögtön bejelentkezik a gép 3583 Byte Free felirattal.

3. Az ún. védett programokat rendkívül nehéz, szinte lehetetlen lemásolni. A másolás problémája legtöbbször azonban megoldható, ha beadjuk a POKE55,30 :SAVE"NÉV" utasításokat.

4. BASIC programok mentése. Ha tévedésből NEW vagy RESET hatására BASIC programunk elveszett, és meg akarjuk magunkat kímélni az újragépelés fáradságaitól, adjuk be a gépnek a következő utasítássorozatot a szigorúan megadott sorrendben:

POKE46,PEEK(56)—1:POKE45,
PEEK(55)+247:CLR
RETURN-t nyomni
POKE PEEK(44)*256+PEEK(43)+1,
PEEK(44) RETURN-t nyomni
63999 RETURN-t nyomni
FORI=PEEK(44)*256+PEEK(43)TO
PEEK(46)*256+PEEK(45):IF
PEEK(I)OR PEEK (I+1)OR
PEEK(I+2)THEN NEXT:
RETURN-t nyomni
POKE45,(I+3)AND255:POKE46,
(I+3)/256:CLR RETURN-t nyomni

Egyes esetekben, amikor már az utolsó sort is begépeltük és RETURN-t adtunk, hibajelzést kaphatunk. A hibajelzés azonban ne akadályozzon meg senkit abban, hogy újból kilistázza elveszettnek hitt BASIC programját.

JÓKAI ANDRÁS

COMMODORE 64

Floppykezelés Kernal-rutinok segítségével

Fenti címmel igen hasznos és színvonalas cikk jelent meg a µMagazin 1985/4. számában, Frey Judit tollából. A közölt programok, rutinok némelyike azonban olyan operandusokat tartalmaz, amelyeket a birtokomban levő assemblerek (HELP+, PROFIMAT, ASM 64) nem tudnak értelmezni. Az

LDX# < NEV
LDY# > NEV

típusú utasítások operandusairól van szó.

A cikk nem tesz említést arról, hogy ezek a programok milyen assemblerrel fordíthatók le. Nem világos, hogy egyáltalán létezik-e ilyen assembler, vagy a fenti operandusok csupán a programozónak szólnak. Hivatkozik a cikk a The Anatomy of the Commodore 64 (Abacus Software) című könyvre. Sem ez a könyv, sem a The Anatomy of the VIC 1541 Disk Drive nem árulja el az asembler mibenlétét. Csupán az Advanced Machine Code Programming for the Commodore 64 (Granada Technical Books) című könyvben bukkantam rá a MIKRO 64 assemblerre, amely valószínűleg megengedi a fenti típusú operandusok használatát.

Akik nem rendelkeznek ilyen vagy ehhez hasonló assemblerrel, kénytelenek a NEV címke LO és HI bájtját kiszámolni, és azt a programba beírni. Azon túl, hogy ez igen kényelmetlen feladat, ha a NEV nem fix helyen van, a program relokálhatósága is elvész.

Az alábbi lista az említett cikk 7. példájának átfogalmazásával mutatja a probléma egy megoldását. Ebben a példában a NEV hosszának megállapításával sem kell bajlódnunk.

LINDEISZ LÁSZLÓ

```
A9 03          LDA #3          ;LOGIKAI FILE SZAMA
A2 08          LDX #8          ;FLOPPY EGYSEG SZAMA
A0 03          LDY #3          ;CSATORNA SZAM
20 BA FF       JSR $FFBA       ;SETLFS RUTIN
20 .. ..       JSR SNEV        ;A 'NEV' CIMKE CIME-1 A STACK-BE KERUL
46 49 4C NEV   .TEXT "FILENEV"
68       SNEV  PLA             ;LO BYTE A STACK-BOL
AA             TAX
68             PLA             ;HI BYTE A STACK-BOL
A8             TAY
A9 07          LDA #SNEV-NEV   ;A FILENEV HOSSZA
E8             INX
D0 01          BNE NO          ;LAPHATAR UGRAS?
C8             INY
20 BD FF       JSR $FFBD       ;SETNAM RUTIN
20 C0 FF       JSR $FFC0       ;OPEN RUTIN
  .
  .
  .
```

TISZTELT SZERKESZTŐSÉG!

A Mikroszámítógép Magazin 1985/4. számában Richter Jörg által írt „Lemeznévváltoztatás" című programmal kapcsolatban kisebb kiegészítést szeretnék tenni. A programot begépeltem és kipróbáltam. Ha csak a lemez nevét változtatjuk meg, semmi probléma nincs, de ha az ID-t is megváltoztatjuk (ami egyébként szerény véleményem szerint nem éppen egészséges), akkor produkálja a program a lehet, hogy csak nekem nem tetsző szépséghibát. A DOS Verzió és Formátum jelét listázáskor (képernyőre is és printerre is) a név és ID utáni sorba írja *(1. lista)*. Kisebb kiegészítéssel ez kiküszöbölhető. Ez az általam küldött első verzió *(2. lista)*. A második verzióval *(3. lista)* az eredeti programmal megváltoztatott „fejléc" hozható eredeti állapotba. Ha ezt elvégeztük, az első variációt véglegesíthetjük, így a DOS Verzió és Formátum jele név és ID változtatás után a helyén marad. (A második verzió CHRS kódjai csak C 64 és VC 1541 esetén érvényesek.)

```
0 "[illegible]"
2A
37   "LEGHAJO"          PRG
599 BLOCKS FREE.
```

1. lista

2. lista

```
NEV*

1142 PRINT#1,"B-P:";8;165
1144 GET#8,D1$,D2$:PRINTD1$;D2$
1262 PRINT#1,"B-P:";8;164
1263 PRINT#8,CHR$(160)
1264 PRINT#1,"B-P:";8;165
1266 PRINT#8,D1$;D2$
```

```
1260 PRINT#1,"B-P:";8;164
1262 PRINT#8,CHR$(160);CHR$(50)
     ;CHR$(65)
```

3. lista

KOVÁCS TIBOR
1212 Budapest,
Hardy Miklós u. 18.

COMMODORE 64

BASIC adatíró program

Haladó programozók gyakran írnak kisebb assembler rutinokat, amelyeket BASIC programból hívnak SYS utasítással. Ezek a rutinok kétféleképpen kapcsolhatók a főprogramhoz:

1. Az elkészült assembler rutint monitorprogram segítségével programfájl formájában lemezen tároljuk, majd azt a BASIC program indításakor a memóriába töltjük.

2. Az assembler programból BASIC DATA sorokat készítünk, majd az adatokat a főprogramba építve, egy ciklusban POKE utasításokkal töltjük be a megfelelő memóriaterületre.

Azoknak, akik a 2. módszert választják, segít az adatíró program. A művelet során ugyanis segédeszköz nélkül több hibát vethetünk, amelyek felderítése esetenként igen fáradságos s hosszadalmas munka lehet. Előfordulhat, hogy a monitorral vagy más módon kinyomtatott tártartalom újragépelésekor bizonyos adatokat elírunk, kihagyunk. Az adatokat betöltő ciklus ciklusváltozója kezdő- és végértékének meghatározásakor hibát vethetünk, mivel assembler rutinunk kezdő- és végcímét többnyire hexadecimális formában ismerjük, és ezt számoljuk át decimálisra. Ha végcímet nem adunk meg a ciklusban, hanem utolsó adatként –1-et, vagy más, a rutinban nem előforduló értéket adunk, hibát ugyan nem vétünk, de időt veszítünk a beolvasott adat ismételt ellenőrzése során.

Az adatíró program egy tetszőleges tárterületről képes lemezen azonnal létrehozni egy BASIC programot, amely tartalmazza egyrészt a betöltő FOR ciklust REM utasítás formájában, másrészt a tár tartalmát DATA utasításokban.

Az assembler rutin memóriába töltése után adjunk NEW parancsot, majd LOAD"DATAIRO", 8 és RUN. A feltett kérdésre a kezdő- és végcímet tetszőlegesen adhatjuk meg hexadecimális (5 karakter, az első $) vagy decimális formában. Szükség van továbbá a létrehozandó DATA program első sorszámának és a lépésköz értékének meghatározására, majd a program nevének megadására. A futás befejeztével a gép által készített programot a szokásos módon a tárba tölthetjük, kilistáztathatjuk.

Az összeszerkesztéshez saját BASIC programunk betöltése után a HELP+ program append utasításának (A"programnév", 8) segítségével vagy a BASIC mutatók módosítása után az elkészített DATA sorokat tartalmazó programot a bent levő programhoz fűzzük. A kezdő sorszám meghatározásakor ügyeljünk a sorszámozás növekvő sorrendjére! Végül a REM utasításban szereplő FOR betöltő ciklust a programban tetszőleges sorszámmal ellátva áthelyezzük, majd a kész programot újra eltároljuk a lemezen.

VASS ANDREA

```
100 REM         **********************
105 REM         *                    *
110 REM         *  DATAIRO PROGRAM   *
115 REM         *                    *
120 REM         *    VASS ANDREA     *
125 REM         *                    *
130 REM         *  1985.SZEPTEMBER   *
135 REM         *                    *
140 REM         **********************
145 REM
150 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
155 DIM SZ$(15)
160 FOR I=0 TO 15:READ SZ$(I):NEXT I
165 POKE53280,11:POKE53281,11
170 PRINT"■■■■                                                    ■";
175 PRINT"■     TETSZOLEGES  MEMORIATERULETROL      ■";
180 PRINT"■                                        ■";
185 PRINT"■    BASIC DATA PROGRAMFILE KESZITESE    ■";
190 PRINT"■                                        ■■■■";
195 INPUT"■KEZDOCIM                ";KC$
200 INPUT"VEGCIM                  ";VC$
205 INPUT"■ELSO SORSZAM            ";SO
210 INPUT"LEPESKOZ                ";L
215 INPUT"■DATA PROGRAMFILE NEVE ";A$
220 :
225 REM *** CIMKISZAMITAS
230 :
235 IF LEFT$(KC$,1)<>"$" THEN KC=VAL(KC$):GOTO 245
240 C$=KC$:GOSUB 440:KC=C
245 IF LEFT$(VC$,1)<>"$" THEN VC=VAL(VC$):GOTO 260
250 C$=VC$:GOSUB 440:VC=C
255 :
260 REM *** OUTPUT FILE MEGNYITASA
265 :
270 A$="@:"+A$+",P,W"
275 OPEN2,8,2,A$
280 PRINT#2,CHR$(1);CHR$(8);
285 :
290 REM *** REM UTASITAS
295 :
300 KC$=STR$(KC):KC$=RIGHT$(KC$,LEN(KC$)-1):KC$=RIGHT$("00000"+KC$,5)
305 VC$=STR$(VC):VC$=RIGHT$(VC$,LEN(VC$)-1):VC$=RIGHT$("00000"+VC$,5)
310 LL=0:LH=SO:IF SO>255 THEN LL=INT(SO/256):LH=SO-256*LL
315 PRINT#2,CHR$(50);CHR$(8);CHR$(LH);CHR$(LL);CHR$(143);
320 PRINT#2," FOR I=";KC$;" TO ";VC$;":READ A:POKE I,A:NEXT I";
325 PRINT#2,CHR$(0);
330 :
335 REM *** DATAK
340 :
345 MH=88:ML=8
350 LH=LH+L:IF LH>255 THEN LH=LH-256:LL=LL+1
355 FOR I=KC TO VC STEP 8
360 PRINT#2,CHR$(MH);CHR$(ML);CHR$(LH);CHR$(LL);CHR$(131);CHR$(32);
365 J=I
370 DA$=STR$(PEEK(J))
375 DA$=RIGHT$(DA$,LEN(DA$)-1):DA$=RIGHT$("00"+DA$,3)
380 PRINT#2,DA$;
385 J=J+1
390 IF J=I+8 OR J>VC THEN GOTO 400
395 PRINT#2,CHR$(44);:GOTO 370
400 LH=LH+L :IF LH>255 THEN LH=LH-256:LL=LL+1
405 MH=MH+38:IF MH>255 THEN MH=MH-256:ML=ML+1
410 PRINT#2,CHR$(0);
415 NEXT I
420 PRINT#2,CHR$(0);CHR$(0)
425 CLOSE2
430 END
435 :
440 REM *** CIM : HEXA->DECIMALIS
445 :
450 C1$=MID$(C$,2,1)
455 C2$=MID$(C$,3,1)
460 C3$=MID$(C$,4,1)
465 C4$=MID$(C$,5,1)
470 FOR I=0 TO 15
475 IF SZ$(I)=C1$ THEN C=4096*I :I=15
480 NEXTI
485 FOR I=0 TO 15
490 IF SZ$(I)=C2$ THEN C=C+256*I:I=15
495 NEXTI
500 FOR I=0 TO 15
505 IF SZ$(I)=C3$ THEN C=C+16*I :I=15
510 NEXTI
515 FOR I=0 TO 15
520 IF SZ$(I)=C4$ THEN C=C+I    :I=15
525 NEXTI
530 RETURN
```

Innováció a számítástechnikában

Még tavaly ősszel történt, hogy a Neumann Társaság Kritikusok Klubjának rendezésében ifj. Marosán György előadást tartott Innováció a számítástechnikában címmel. A Mikroszámítógép Magazin szerkesztősége felkért, hogy írjak cikket az ankétról.

Kihegyezett ceruzákkal ültem le, ám hamarosan becsuktam jegyzetfüzetemet. Az előadás nagyon érdekes volt, engem különösen érdekelt. A fő téma a kisvállalkozásoknak az innovációban betöltött szerepe volt, de — mivel a Mikroszámítógép Magazin, nem pedig a mikrovállalkozás magazin számára kellett volna írni — úgy gondoltam, ebből nem lesz cikk.

Annál nagyobb volt a meglepetésem, amikor néhány nap múlva levelet kaptam a szerkesztőségtől: információkat kérnek a SZUBRUTIN GMK tevékenységéről, melynek képviselője vagyok. A levél tartalmának lényege, hogy a magazin bővíteni kívánja profilját, fórumot szeretne biztosítani a számítástechnikai kisvállalkozásoknak, arra törekszik, hogy koordinálja ezek tevékenységét, vagy ha úgy tetszik (és ezt már én fogalmazom így), szeretné elősegíteni a kisvállalkozások innovatív tevékenységét.

Ezt a kifejezést, hogy innovatív, kissé bátortalamul írnám le, ha nem vettem volna részt az ankéton. Az előadó azonban a bevezetőben nagyvonalúan eltekintett az innováció fogalmának meghatározásától. Azt hiszem, ez bölcs dolog volt, mert manapság ezt a fogalmat eléggé tágan értelmezik: a tudományos alapkutatások műszaki fejlesztésbe kerülése, technologizálása és megvalósítása folyamatától kezdve a haveri alapon történő inkorrekt üzletecskék megkötéséig minden belefér.

Az ankétot Dobó Andor, a klub vezetője nyitotta meg. Bemutatta a főszereplőket, és a klub hagyományaihoz híven meghirdette a „nap hőse" pályázatot, amelyet az a vitázó nyer meg, aki megtépázva a padlóra kerül.

Kertész Ádám bevezetőjében humorosan, de meglehetősen szkeptikusan indított. Csak egy költői kérdést idézek: felnőttek vagyunk-e eléggé ahhoz, hogy ilyen kérdésekkel zaklassanak? A kérdésre választ nem kapott, az előadás utáni vitát végighallgatva a válaszom: nem tudom.

Ifj. Marosán György vette a lapot, illetve kiderült, hogy kapta. Újítás és kockázat című könyvének megírására a Móra Kiadó kérte fel. (Hátha jó lesz a gyerekeknek is, kommentálta maliciózusan.) Előadásának tárgya: ami a könyvből kimaradt.

A jó hangulatot keltő bevezetés után fontos kérdések következtek. Az előadó a csereárányok romlását elemezte egy egyszerű grafikonon szemléltetve. A konklúzió: elsősorban a tudományos eredményeket tartalmazó termékek értékesítése biztosíthatja népgazdaságunk fejlődését, természeti kincseink ugyanis korlátozottak.

Az előadó a fejlett ipari országokban megfigyelhető jelenségek alapján érzékeltette, hogy korábban a nagy vállalatok voltak az innováció központjai, de egyre nagyobb szerepet kaptak a kisvállalkozások, illetve az azokból kinövő vállalatok (például Apple story). Különös hangsúlyt kapott az ötlet, azaz tehetséges, sokszor „balmenetes" emberek kezdeményezőképességének és kockázatvállalásának eredményeképpen születő technikai — egyúttal piaci — robbanások szerepe és jelentősége. Az alkalmazott módszer: néhány tehetséges, ötletdús szakember kiválik a nagy cég kötelékéből, és kisvállalkozási keretben megvalósítja zseniális elképzelését, betör a piacra, alkot valami nagyszerűt, és világcéggé válik.

Ennek a látszólag egyszerű képletnek feltételei is vannak: tőke, megfelelő infrastruktúra (például helyiség), támogatás. Ezek a tárgyi feltételek, van azonban más is. A kisvállalkozásokat menedzselni kell, és ez nem egyszerű, megfelelő emberre van szükség.

Egy másik, a szó szoros értelmében életbevágó szempont: kockázatot kell vállalni. A fejlett tőkés országokban kisvállalkozásokat támogató bankok vannak. A bank tisztában van azzal, hogy az általa tőkével támogatott kezdeményezések közül nagyon kevés fog befutni, de ha egy befut, megtérülhet tíz másik elsorvadó vállalkozásba fektetett tőke. Ez viszont csak úgy érhető el, ha közös a kockázatvállalás. A bank részéről nem az adott tőke kamata a nyereség, hanem az eredmény meghatározott része. A másik oldalról: a bank — mondhatni magándetektívek segítségével — kideríti, hogy a vállalkozó mekkora tőkével képes rendelkezni (beleszámítva, hogy az anyós is kölcsön ad), ezt a „belépti díjat" kéri. Ő adja a többit, ez lehet nagyságrendekkel magasabb összeg, mint amit a vállalkozó nyújt.

Nyugaton az ilyen kisvállalkozásokat támogató központok beváltak; egy reprezentáns példa, hogy a Szilíciumvölgy egy egyetem tudományos — ha úgy tetszik — innovációs parkjából nőtt ki.

Mi a helyzet hazánkban? Az előadó biztatott bennünket: kicsit jobb, mint gondolnánk. Nézzük a feltételeket. Tőke van, a nagyvállalatok rendelkeznek elegendő pénzzel, hogy támogassák a kicsiket. Gond azonban, hogy félnek a kockázattól. Infrastruktúra, magyarul helyiség, eszközök: itthon is kellene innovációs parkokat kialakítani, tehát biztosítani, hogy a kockázatvállalók hosszan elnyúló beruházások helyett gyorsan jussanak a szükséges munkahelyhez, eszközökhöz. Csak egy lényegtelennek látszó, mégis meghatározó tényező: mennyi idő kell ahhoz, hogy egy néhány emberből álló csoportosulás telefonnal ellátott irodához jusson.

Tegyük fel, hogy az objektív feltételek rendelkezésre állnak. (Az előadó szólt arról, hogy Magyarországon is létezik innovációs bank.) Milyen irányba léphetünk, hogyan „innováljunk"? A válasz — és ez talán az alapkutatásokkal foglalkozókat nem lelkesíti: a piac dönti el, hogy a szellemi munka eredménye, a kezdeményezőkészség, a kockázatvállalás megél-e vagy sem.

A piacon mindig vannak kis rések, ám egy kisvállalkozás számára a kis rés is nagy lehet. A kis rések betöltése nagy eredményeket is hozhat, és ez nemcsak a népgazdaság számára, hanem a kockázatvállalók személyes jövedelme szempontjából is hasznos.

A hozzászólók nagy része ecsetelte a hazai nehézségeket. Nincs piaci lehetőség, Magyarország kicsi, nincs infrastruktúra, nincs támogatás. Ifj. Marosán György parírozott: megfelelő, menedzser típusú vállalkozó kell, ügyesnek is kell lenni, még a PM-et is meg lehet győzni, ha érzékelhető az eredményesség és érezhető az elszánt kockázatvállalás.

Volt, aki elmesélte egy jó és gazdaságos ötlet megvalósítására tett kísérleteinek kudarcát. Az előadó feltételezte, hogy nem az alkalmas piaci résen próbáltak behatolni.

Egy számítástechnikai intézet képviselője rámutatott arra, hogy hazánkban a szellemi koncentráció az intézetekben van, tehát ne a kisvállalkozásokat tekintsük az innováció fellegvárainak. Ezt az előadó nem vitatta, de feltette a kérdést, miért válnak ki sok esetben az ifjú tehetségek az intézetekből, és miért kisvállalkozásokban valósítják meg elképzeléseiket. Volt, aki erre azt mondta, hogy a kisvállalkozások a nagyok támogatásából élnek, és innovatív szerepük kicsinyes, de jövedelmező üzletté degradálódik. Ez a megjegyzés sem maradt válasz nélkül: biztosítsák a „nagyok" is a tehetségek anyagi elismerését.

Egy hozzászóló kifejtette, hogy a hazai kisvállalkozások nagy problémája, hogy nem ismerik őket, és ők sem ismerik a piacot. Ifj. Marosán György bedobott egy ötletet: az innováció szempontjából is, és egyénileg is hasznos lenne egy olyan vállalkozás, amely ezt a szellemi piacon levő rést betöltené.

Ennek az ötletnek és a magazin fentebb említett kezdeményezésének véletlenszerű egybeesése inspirált arra, hogy néhány szóban említést tegyek az ankétról, valamint az, hogy ebben a klubban csupa olyan téma kerül napirendre, amelyek érdekesek és bármilyen képzettségű érdeklődő számára érthetők. És a vita! Az mindent megér. Itt el szabad mondani még az eretneknek tűnő véleményeket is.

A klubdélutánon a „nap hőse" címet senki sem nyerte el; az előadó is és a hozzászólók is adtak, de kaptak is. A végső benyomás azonban egyértelmű, és ez elhangzott az ankét berekesztésekor: „Innovare necesse est".

SZABÓ SZABOLCS

Számítástechnika az Alma Materokban

Kissé hitetlenkedtem, amikor megtudtam, hogy a budapesti Berzsenyi Dániel Gimnáziumban 23 éve tanítják a számítástechnikát, hiszen a szélesebb közvélemény számára e tudomány nem régen vált divatos fogalommá, és középiskolai oktatása is ugyancsak az első lépéseket tette meg hazánkban. Kálmán Ferenc igazgató:

— Kezdetben speciális matematikai tagozatunk tantervébe illesztettük a számítástechnika oktatását, külön óraszámmal. Elsősorban elméletileg, mert gépeink nem voltak. E közel negyedszázados hagyomány, valamint legjobb diákjaink sok nagyszerű eredménye a különböző pályázatokon és versenyeken jelentős helyzeti előnyt teremtett számunkra az iskolaszámítógépes program beindulásakor, 1983-ban.

— Miben nyilvánult meg konkrétan ez az előnyös helyzet?

— Például abban, hogy elsőnek nem egy, hanem mindjárt három — HT — gépet kaptunk. Eredményeinkre a kerületi tanács is felfigyelt, innen is kaptunk gépet.

Koltai Márta matematika—fizika szakos tanárnő, az egyik haladó szakkört vezetője:

— Attól kezdve, hogy az első gépeket 1983 áprilisában megkaptuk, a tanév végéig szinte benn laktunk a gyerekekkel az iskolában.

A szűk teremállomány ellenére igazgatónk „kiszorított" egy helyiséget, ezzel lehetővé vált, hogy a gépeken bárki bármikor dolgozhasson. Megszerveztük a munka állandó tanári felügyeletét, így a gépek kihasználtsága már maximálisnak mondható.

— Tanítás, szakkör, aztán gépidő, ami ugyan kötetlenebb, de mégis tanári felügyelet, szinte mindennap. Ne haragudjon a kérdésért, de kaptak mindezért valami külön juttatást? Vagy kötelező volt?...

— Nem, nem volt kötelező. Kezdetben társadalmi munkában végeztük, majd egy ideje úgynevezett irányított labormunka címén óránként negyven forintot „kasszírozunk"...

— Ez sem valami sok. Miért csinálják?

— Miért sakkozik az ember? — kérdez vissza Siegler Gábor, a másik haladó szakkör vezetője.

— A gyerekek miatt csináljuk — folytatja a tanárnő. — Lelkesen és komolyan dolgoznak. Több olyan oktatóprogramot készítettek már, amelyet a Tudományszervezési és Informatikai Intézet megvett, pályázataikkal számítógépet nyertek, különböző országos versenyeken szereztek győzelmeket. Ezek a sikerek is magyarázzák talán, miért csináljuk.

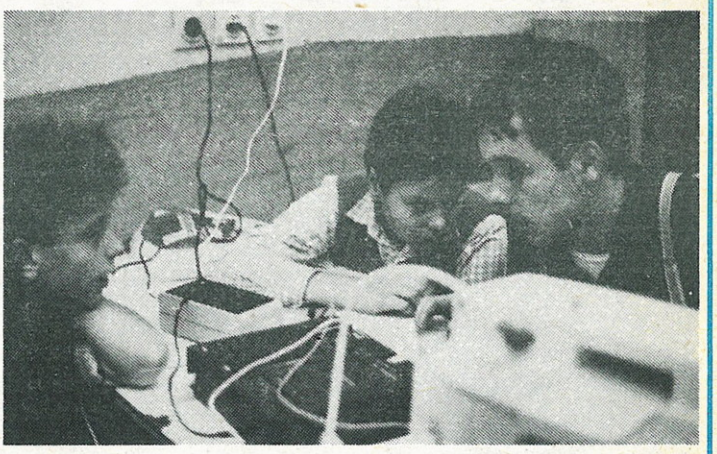
— Milyen nehézségi szintet jelentenek a pályázatokra beadott programok?

— Az ötlet mostani pályázatukra a térbeli malom önfejlesztő változatát, egy szövegszerkesztő programot és egy növényhatározót, illetve kőzetfelismerő programot kellett készíteni. Ezek komoly szellemi erőfeszítést, profi felkészültséget igénylő feladatok.

— Igen, de akik ezeket csinálják, kiugró tehetségű diákok. Már egyetemen vannak, vagy tanulmányi versenyeredményeik alapján biztosan oda kerülnek, mint például a korábbi beszélgetésünkben többször említett Kós Géza, aki matematikai diákolimpiát nyert. A többiekkel mi lesz?

— Valóban, Kós Géza, Molnár Attila és még néhány diákunk tudása olykor minket is zavarba hoz. Ne feledjük azonban, hogy a számítógépekkel túlnyomórészt ők is itt ismerkedtek meg! Ami a többieket illeti, a technika tantárgy lehetőséget ad, hogy mindenki legalább alapfokon elsajátítsa a számítástechnikát, megtanuljon egyszerűbb programokat írni. Használhatják a gépeket, az érdeklődőbbek járhatnak a hat szakkör valamelyikébe. Szeretnénk, ha minden tárgyban, ahol csak lehet, felhasználnák a számítástechnikát. A tanárok megadják az ötletet, amit modellezni lehet, a gyerekek aztán oktatóprogramot írnak.

— Azt elhiszem és tudom is, hogy ez a matematikában és a fizikában beválik, de a többihez az kell, hogy a szaktanárok legalább valamennyire ismerjék a számítástechnika lehetőségeit.



— A tavalyi BASIC tanfolyamon biológia-, nyelv- és történelemtanárok is lelkesen részt vettek. Számukat a jövőben tovább szeretnénk növelni.

— A szakkörvezető tanároktól hallottam, hogy nehéz a könyvek beszerzése és egy-egy drágább példányból legfeljebb egyet tudnak venni — fordulok az igazgatóhoz.

— Az évi 5—6000 forintos keretből valóban nem könnyű gazdálkodni, de ennek többszörösét fordítjuk 30 ezer kötetes könyvtárunk bővítésére. Lehetőség szerint, apránként és súlyozva. Ami a példányszámot illeti, hát abban sajnos van igazság...

— A túlterhelt tanároknak elég kevés idejük lehet a fejlődéssel lépést tartani. Lehet ezen segíteni?

— Ez általános gond, de itt különösen érezhető. Az iskola vezetése a mostani tanévtől óraszámcsökkentéssel próbál meg a gondon enyhíteni.

— Visszatérve a középiskolai számítástechnikai oktatás egészére, véleménye szerint a szakkörök létrehozása országos méretekben megoldást jelent?

— Mi tantárgyként szerettük volna bevezetni, didaktikusabban tanítani — erre más szocialista országokban vannak példák, így az NDK-ban is a mi elképzelésünkhöz hasonló módszerrel oktatják a számítástechnikát. Az oktatási irányítás azonban nem járult hozzá ehhez, mondván, hogy ez valaminek kárára történne, vagy a diákok óraszámát növelné. Mi nem mondtunk le erről és akár kísérletképpen is vállalnánk a tantárgyszerű oktatás.

— Úgy tudom, a Berzsenyi nemsokára a Duna-partra költözik. Mit vár ettől a változástól?

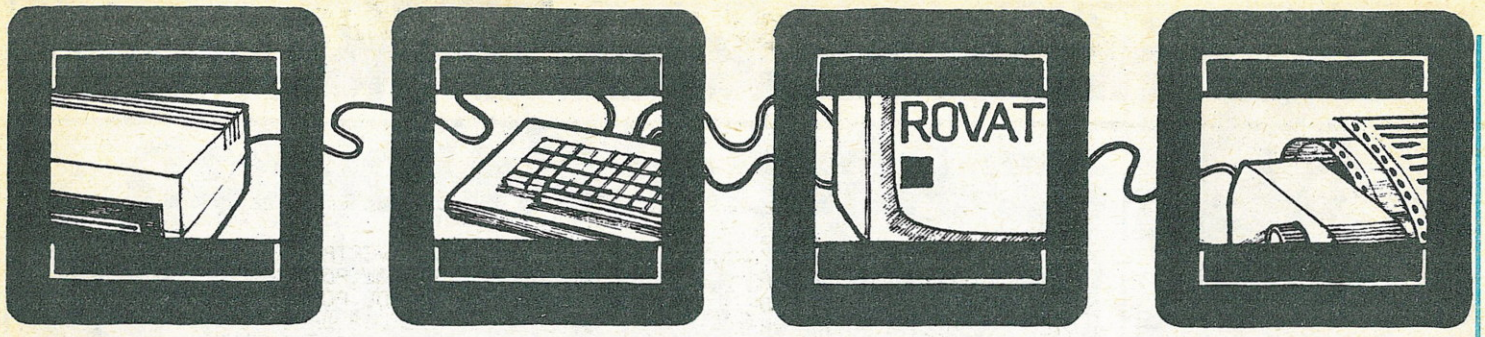
— Nagy terveink vannak. Egy számítástechnikai labort szeretnénk létesíteni egy komplex információs bázis részeként. Itt a diákok feldolgoznák az oktatási-technikai apparátus minél nagyobb részét, és ezeket a programokat és eszközöket más hasonló intézményekkel lehetne cserélni, egy széles körű rendszerré bővíteni. A hangtárral és könyvtárral együtt a kerület komoly kulturális bázisává fejlődhetnénk, a gyakorlatban megvalósítva a nyitott iskola elvét. Ehhez persze többszörösére kellene gépállományunkat növelni, a megvalósításhoz azonban a jelenleginél jobban kellene éreznünk a központi segítséget.

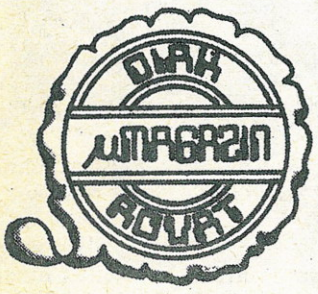
— Milyen segítséget?

— A középiskolai számítástechnikai program nagyszerű kezdeményezés volt, de a gépeket például ma még mindig nem lehet intézményesen beszerezni. Ha egy Spectrumot akarunk venni, akkor a Bizományihoz kell fordulnunk...

Az iskolának tehát nagy tervei vannak. A szakkör utáni gépidőben megkérdeztem néhány diákot, ők mire készülnek. A negyedikes Szilvási László szoftver programozó szeretne lenni. A harmadikos Hajnal Zoltán még nem döntött, hogy az ELTE-re, vagy a Műszaki Egyetemre fog jelentkezni, de az biztos, hogy későbbi pályáján használni fogja a számítógépet. A szakkörben töltött időről úgy beszél, hogy ezalatt „érzelmi kapcsolat" jött létre közte és a számítógépek között...

LACZKA MIKLÓS





DIÁKOK, SZAKKÖRÖK!

Folytatjuk hagyományainkat! Továbbra is közlünk hasznos programokat, szellemes áramköröket, érdekes felfedezéseket, újdonságokat, eredeti ötleteket, ínyencségeket. Bemutatjuk a számítástechnika oktatásában és alkalmazásában élenjáró szakkörök eredményeit és módszereit. Minden pénteken 14—17 óráig találkozhattok a diákszerkesztőkkel a budapesti Berzsenyi Dániel Gimnáziumban (XIII. ker., Huba utca 7. szám alatt), ahol a bemutatott ötleteiteket ki is próbálhatjátok. Továbbra is szeretettel várunk minden érdeklődő diákot és szakkört.

A rovat anyagát a budapesti Berzsenyi Dániel Gimnázium tanulói írták.
A grafikákat Földi Endre, a Képző- és Iparművészeti Szakközépiskola IV/B osztályos tanulója készítette.

A RÉGI BOTKORMÁNY ÚJ LEHETŐSÉGEI

Igényesebb ZX—Spectrum-tulajdonosok gépeik billentyűzetének védelme érdekében a különböző játékokhoz botkormányt használnak. Külföldön és itthon is igen sokféle gyártmányú és típusú botkormány kapható.

Az itt közölt kiegészítő áramkörrel azoknak próbálok segítséget nyújtani, akiknek olyan botkormányuk van, amelybe gyárilag be van építve egy úgynevezett gyorstüzelő áramkör. Eddig a különböző lapokban közölt kapcsolások egyike sem vette ezt figyelembe. A gyorstüzelő áramkört tápfeszültséggel kell ellátni, mégpedig úgy, hogy a botkormány karjával közös pontja 0 volton legyen. Ez az itt látható kapcsolás segítségével meg is valósítható.

A megépített áramkörrel Kempstone és Datel I. R. interfészre programozott játékok vezérelhetők. Az áramkör ki van egészítve egy RESET gombbal és egy VIDEO kimenettel, amelyet minden nehézség nélkül beépíthetünk interfészek dobozába, ugyanis ezek ki vannak vezetve a számítógép külső csatlakozósávjára. A csatlakozósáv bekötése a számítógép gépkönyvében mindenki számára elérhető. A botkormány kapcsolóinak állapota például az IN223 utasítással olvasható be (A5=Ø).

A kapcsolóknak és az alsó öt adatbitnek a megfeleltetése a következő:

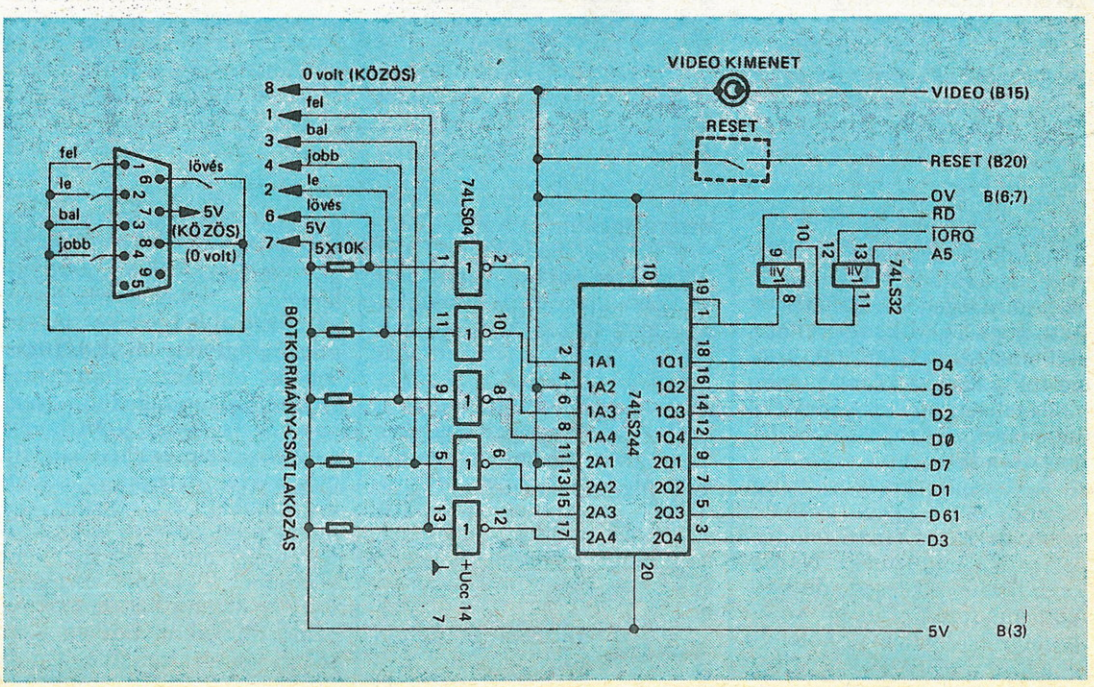
„jobb" — DØ
„bal" — D1
„le" — D2
„fel" — D3
„lövés" — D4

A botkormány alaphelyzetében és elengedett lövésgombbal az IN223 utasításra az összes adatbiten Ø-t olvas be a processzor.

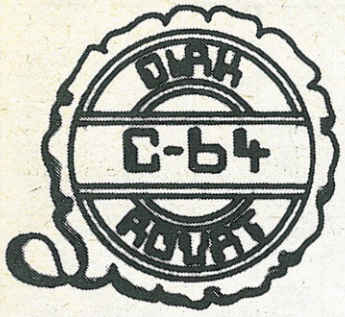
Egy vagy több kapcsoló zárása esetén a beolvasott bájtban a zárt kapcsoló(k)hoz tartozó bit 1 lesz.

Az áramkör megépítése nagy körültekintést és odafigyelést igényel, mivel egy rossz bekötéssel könnyen kárt okozhatunk gépünkben. Az interfészt csak kikapcsolt gépre tegyük fel, vagy vegyük le. Ne mozgassuk bekapcsolt állapotban.

GYEBROVSZKY ANDRÁS



LEFOGLALT BASIC VÁLTOZÓK



A BASIC V2-verzióból jól ismerhetjük a BASIC interpreter által lefoglalt változókat. Ilyenek az ST, TI és a TI$. A Simon's BASIC ezek tárát bővíti ki a PENX, PENY, POT és a JOY változóval. Az alábbi program egy módszert mutat az ilyen változók definiálására.

A program előállít két változót, a £1-et és a £2-t. Ezek funkciója hasonló a Simon's BASIC-féle JOY változóéhoz. A £1 az 56321-es bájt, a £2 az 56320-as bájt alsó 5 bitjét tartalmazza minden olvasáskor. Tehát ez a két változó a két botkormány vizsgálatát leegyszerűsíti és meggyorsítja.

A feladatot lényegében az könnyíti meg, hogy az interpreter kifejezés kiértékelő rutinja átfut a RAM-on. A 778—779-es vektor átírásával saját programunkra irányíthatjuk a BASIC-program fordítását. Ha rutinunk lefutott, a fordítást vissza kell adni a BASIC interpreternek.

Amikor BASIC-program futásánál az interpreter talál egy változót, mielőtt azt azonosítaná, kiad egy JMP (778) utasítást. Ilyenkor kezdhet futni a felhasználó által írt program. Példánkban a 100—140 sorokban levő utasítások végzik a RAM-ban levő vektor átírását. Tehát a programot a SYS49152 BASIC utasítással aktivizálhatjuk. Ezek után semmi dolgunk, csak arra kell ügyelnünk, hogy ne bántsuk a programot.

A rutin megvizsgálja a soron következő BASIC bájtokat. Ha azok nem felelnek meg a £1 vagy a £2 karaktersorozat ASCII kódjainak, akkor visszaállítja az aktuális BASIC bájt mutatót (300—370 sorok), és visszatér a ROM-ba. Ellenkező esetben az arra való rutin segítségével a FAC-ba (lebegőpontos akkumulátor) tölti a megfelelő adatot (280). Ezután betölti az akkumulátorba a következő BASIC bájt értékét, és a program fordítását visszaadja az interpreternek (290).

A 3. lapon levő vektorok sok egyéb lehetőséget nyújtanak a BASIC interpreter — ehhez hasonló — módosításához.

TASS ZOLTÁN

```
2
                        ;       JOYSTICK FIGYELO
                        ;
10:     C000                    .OPT OO,P,P4
20:     C000            RAMVECT =    778      ; ATIRANDO RAM-VECTOR
30:     C000            CHRGET  =    $73
40:     C000            YTOFAC  =    $B3A2    ; Y KONVERTALASA A FAC-BA
50:     C000            PORTA   =    $DC00    ; CONTROL PORT 2
60:     C000            PORTB   =    $DC01    ; CONTROL PORT 1
70:     C000            BASPOINT =   $7A      ; AKTUALIS BASIC BYTE MUTATO
80:     C000            RETTOROM =   $AE86    ; VISSZATERES A ROM-BA
90:     C000                    *=   49152
100:    C000 A9 0B              LDA  #<FOPRG      ; A VECTOR ATIRASA
110:    C002 8D 0A 03           STA  RAMVECT
120:    C005 A9 C0              LDA  #>FOPRG
130:    C007 8D 0B 03           STA  RAMVECT+1
140:    C00A 60                 RTS
150:    C00B 20 73 00 FOPRG     JSR  CHRGET
160:    C00E C9 5C              CMP  #"£"         ; TALAN FONTJEL
170:    C010 D0 25              BNE  NEMFONT
180:    C012 20 73 00           JSR  CHRGET
190:    C015 C9 31              CMP  #"1"         ; UTANA 1-ES
200:    C017 D0 06              BNE  NEM1
210:    C019 AD 01 DC           LDA  PORTB
220:    C01C 4C 26 C0           JMP  AT
230:    C01F C9 32      NEM1    CMP  #"2"     ; UTANA 2-ES
240:    C021 D0 0C              BNE  NEM1NEM2
250:    C023 AD 00 DC           LDA  PORTA
260:    C026 29 1F      AT      AND  #%00011111    ; FOLOSLEGES BITEK LEVAGASA
270:    C028 A8                 TAY
280:    C029 20 A2 B3           JSR  YTOFAC
290:    C02C 4C 73 00           JMP  CHRGET       ; VISSZA
300:    C02F A6 7A      NEM1NEM2 LDX BASPOINT  ; BASPOINT=BASPOINT-2
310:    C031 D0 02              BNE  *+4
320:    C033 C6 7B              DEC  BASPOINT+1
330:    C035 C6 7A              DEC  BASPOINT
340:    C037 A6 7A      NEMFONT  LDX BASPOINT   ;BASPOINT=BASPOINT-1
350:    C039 D0 02              BNE  *+4
360:    C03B C6 7B              DEC  BASPOINT+1
370:    C03D C6 7A              DEC  BASPOINT
380:    C03F 4C 86 AE           JMP  RETTOROM
```

PRÓBÁLD MEG!

Írd át a Primo karakterkészletét!

Tudjuk, hogy képernyőnk 256 × 192 pontból álló négyzethálóból. Amikor egy normál karakter megjelenik, egy 6 × 12-es mátrixot foglal le. Ebből max. 5 × 8 ponton helyezkedik el a karakter. A maradék 1 pontnyi rés választja el a következő karaktertől, fölül egy-, alul hárompontnyi sáv tartja a sortávolságot.

A memóriában egy CHR$ karakter 8 bájtot foglal el. Minden bájt 8 bitjéből csak a középső 6 kerül a képernyőre.

A legmagasabb helyiértékű bit a süllyesztett karaktereket (y, g, j stb.), a legkisebb a karakter végét jelöli. A kirajzoló rutin visszatér a főprogramhoz. Persze, ha a bit 1, akkor érvényesül a funkció, ha 0, akkor nem. Ebből látszik, hogy egész képernyő magasságú karaktereket is készíthetünk, föláldozva néhány CHR$ kód helyét egy karakter számára.

Ahhoz, hogy kis Primónkkal közöljük, hogy új karakterkészletre van szükségünk, át kell írnunk a 16459, 16460 memóriacímeket. Ezekben az eredeti érték: 3507H. Az itt található címtől kezdődően 1 kbájtnyi terület (8 bájt/kar. × 128 kar.) szolgál a 128—255-ig terjedő karakterek meghatározására. Célszerű ezt a területet a BASIC-től megvédeni;

10 POKE 16561,255,99
:REM — RAMTOP átírása
20 POKE 16459,0,100
:REM —az új karaktertábla kezdete.

Ezután az új karakterek kódjait POKE utasítással beteszszük a helyükre. Legyen K a karakter kódja:

(K = 128—255).

30 POKE 25600 + (K—128),
A, B, C, D, E, F, G, H

Vegyük példának a görög „fi" betűt:

K = 230
A = 128 — 10000000
süllyesztett karakter
B = 40 — 00101000
C = 84 — 01010100
D = 84 — 01010100
E = 84 — 01010100
F = 56 — 00111000
G = 16 — 00010000
H = 17 — 00010001
karakter vége

Megjegyzések:

1. Ne feledkezzünk meg egy Cirill nevű betűtípusról!

2. A RESET visszaírja a 16459—16460 bájtot.

3. A Primo *teljes* karakterkészlete 31F7H—35C6H-ig terjed (ROM-ban).

Pályázat: Írjuk át a 0-tól 127-ig terjedő karakterkészletet! A megoldásokat a szerkesztőség címére kérjük, PRIMO felirattal. Ha valaki tudja, hogy miért *nem* megoldható a feladat, ő is írjon!

HAJNAL ZOLTÁN

SPECTRUM A PRIMÓBAN ?

Nincs benne, de egy hasznos tulajdonságban megegyeznek. A ZX—Spectrum utasításait nem kell betűnként begépelni, elég egy-két billentyűt lenyomni. Ezt a Primo is tudja. Az *1. táblázat* mutatja, hogyan. Ha például a RUN utasítást akarjuk elérni, egyszerre le kell nyomnunk a SHIFT, a CONTROLL, a ↓ és az N billentyűket. GOTO-hoz elég a ↓ és a RETURN billentyű. A CALL utasításhoz a SHIFT-et, a ↓-t és az A-t kell lenyomnunk.

Most jön a meglepetés: a képernyőn megjelenő karakter nem alkalmas az utasítások felismerésére. De ez nem baj. Ha az így beírt programsort kilistázzuk, a kívánt utasítás már a helyén lesz.

Amint látható, a Primo—BASIC összes utasítása és függvénye elérhető egy karakter beírásával. Ezeket képtelenség megjegyezni. De a fontosabbakat hamar megtanulja a programozó.

A figyelmes olvasó találhat olyan utasításokat a táblázatban, amelyeket a gép értelmez ugyan, de végrehajtásuk ERROR jelzést eredményez. Ezek a későbbi Primo—DISK-rendszereket kezelő utasítások lesznek. Hagyjuk őket nyugodtan.

(NE) NYÚLJ BELE!

Mármint a BASIC-programba. Ha akarod, megteheted, de óvatosan. A BASIC-program tárolásának helyét a

? 256*PEEK(16549)+PEEK-(16548)

begépelésekor a gép kiírja. Az A32-es változaton ez az érték 17386=43EAH. Ettől a címtől kezdődően a program a következőképpen tárolódik:

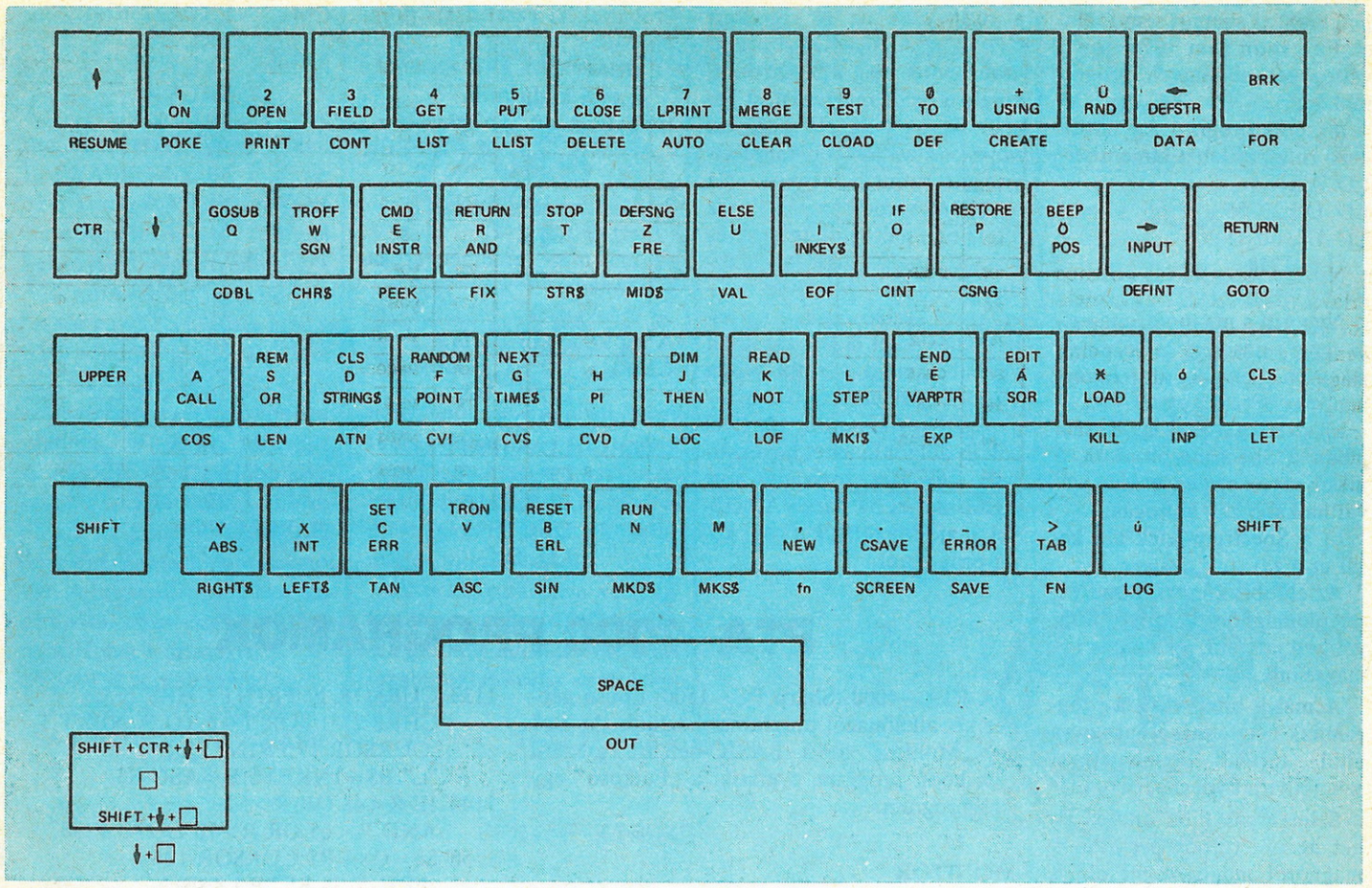
2 bájt → a következő BASIC sor helye a memóriában
2 bájt → az aktuális sor sorszáma
ezután a programsor

1. táblázat

KÓD	UTASÍTÁS	SHIFT	CTR	↓	BILLENTYŰ
217	ABS	x	—	x	Y
210	AND	x	—	x	R
246	ASC	—	—	x	V
228	ATN	—	—	x	D
183	AUTO	—	—	x	7
156	BEEP	x	x	x	Ö
193	CALL	x	—	x	A
241	CDBL	—	—	x	Q
247	CHR$	—	—	x	W
239	CINT	—	—	x	0
184	CLEAR	—	—	x	8
185	CLOAD	—	—	x	9
166	CLOSE	x	—	x	6
132	CLS	x	x	x	D
133	CMD	x	x	x	E
179	CONT	—	—	x	3
225	COS	—	—	x	A
171	CREATE	—	—	x	+
186	CSAVE	x	—	x	.
240	CSNG	—	—	x	P
232	CVD	—	—	x	H
230	CVI	—	—	x	F
231	CVS	—	—	x	G
136	DATA	—	—	x	←
176	DEF	—	—	x	0
155	DEFDBL	nem lehet kiíratni!			
153	DEFINT	—	—	x	→
154	DEFSNG	x	x	x	Z
152	DEFSTR	x	—	x	←
182	DELETE	—	—	x	6
138	DIM	x	x	x	J
157	EDIT	x	x	x	Á
149	ELSE	x	x	x	U
128	END	x	x	x	É
233	EOF	—	—	x	I
194	ERL	x	—	x	B
195	ERR	x	—	x	C
158	ERROR	x	—	x	—
224	EXP	—	—	x	É
163	FIELD	x	—	x	3
242	FIX	—	—	x	R
172	fn	—	—	x	,
190	FN	—	—	x	>
129	FOR	—	—	x	BREAK
218	FRE	x	—	x	Z
164	GET	x	—	x	4
145	GOSUB	x	x	x	Q
141	GOTO	—	—	x	RETURN
143	IF	x	x	x	0
201	INKEY$	x	—	x	I
219	INP	—	—	x	ó
137	INPUT	x	—	x	→
197	INSTR	x	—	x	E
216	INT	x	—	x	X
170	KILL	—	—	x	*
248	LEFT$	—	—	x	X
243	LEN	—	—	x	S

1. táblázat (folytatás)

KÓD	UTASÍTÁS	SHIFT	CTR	↓	BILLENTYŰ
140	LET	—	—	x	CLS
189	LIST	—	—	x	4
181	LLIST	—	—	x	5
167	LOAD	x	—	x	*
234	LOC	—	—	x	J
235	LOF	—	—	x	K
223	LOG	—	—	x	ú
175	LPRINT	x	—	x	7
168	MERGE	x	—	x	8
250	MID$	—	—	x	Z
238	MKD$	—	—	x	N
237	MKS$	—	—	x	M
187	NEW	x	—	x	'
135	NEXT	x	x	x	G
203	NOT	x	—	x	K
161	ON	x	—	x	1
162	OPEN	x	—	x	2
211	OR	x	—	x	S
160	OUT	—	—	x	SPACE
229	PEEK	—	—	x	E
200	PI	x	—	x	H
198	POINT	x	—	x	F
177	POKE	—	—	x	1
220	POS	x	—	x	8
178	PRINT	—	—	x	2
165	PUT	x	—	x	5
134	RANDOM	x	x	x	F
139	READ	x	x	x	K
147	REM	x	x	x	S
130	RESET	x	x	x	B
144	RESTORE	x	x	x	P
159	RESUME	—	—	x	↑
146	RETURN	x	x	x	R
249	RIGHT$	—	—	x	Y
222	RND	x	—	x	Ü
142	RUN	x	x	x	N
173	SAVE	—	—	x	—
174	SCREEN	—	—	x	.
131	SET	x	x	x	C
215	SGN	x	—	x	W
226	SIN	—	—	x	B
221	SQR	x	—	x	Á
204	STEP	x	—	x	L
148	STOP	x	x	x	T
244	STR$	—	—	x	T
196	STRING$	x	—	x	D
188	TAB(	x	—	x	>
227	TAN	—	—	x	C
169	TEST	x	—	x	9
202	THEN	x	—	x	J
199	TIME$	x	—	x	G
189	TO	x	—	x	0
151	TROFF	x	x	x	W
150	TRON	x	x	x	V
191	USING	x	—	x	+
245	VAL	—	—	x	U
192	VARPTR	x	—	x	É



2. táblázat

1 bájt → ØØ = sor vége jelzés
(az utolsó sornál 2 bájton van Ø, FILE vége)

A *2. táblázatban* a BASIC-utasítások előtt található kódszám helyettesíti az utasításokat, a többi paraméter karakterenként van tárolva, kivéve a GOTO, GOSUB, RUN utasítások sorszámhivatkozásait, ezek bináris formájúak.

Példa:
1Ø PRINT „A": GOTO 1Ø
A memóriában:

17 386—249—F9H } a következő
67—43H } BASIC sor címe
1Ø—ØAH } ennek a sornak
Ø—ØØH } a száma
178—B2H — PRINT
32—2ØH — SPACE
34—22H — SPACE
65—41H — A
34—22H — A
58—3AH — :
141—8DH — GOTO
32—2ØH — SPACE
1Ø—ØAH } sorszám
Ø—ØH }
Ø—ØH — sor vége jelzés

A program tárolásának helyét megváltoztathatjuk, akár például helyfoglalás céljából, gépi kód számára. Legyen az új cím C, akkor
POKE 16549, INT(C/256):
POKE 16548, C—INT (C/256)-*256

Ezt mindig a gép bekapcsolásakor célszerű elvégezni. Sok sikert!

HAJNAL ZOLTÁN

PROGRAMVÉDÉS, -MÁSOLÁS MIT, HOGYAN VÉDJÜNK?

Eddig is sokat feszegetett kérdéshez szólok elsőként hozzá: a programvédelem és a programmásolás problémájához, annak megoldásához Spectrumon. Természetesen nincs olyan program, amely ne lenne lemásolható; ha másképpen nem, hát két magnó segítségével, bár ez nem megbízható, és sok türelem kell, míg az ember a megfelelő felvételi szintet eltalálja. Ám a lemásolt programban levő programozási trükköket már lehet védeni, hogy illetéktelenek ne ismerhessék meg.

Legegyszerűbb mód erre a sorok „eltüntetése". Ez abból áll, hogy direkt módon a sorba vezérlőkaraktereket helyezünk el. Lehet ezenkívül a sorszámot is nullázni, de ez sem biztosít jobb védelmet.

Védelmet jelenthet az INPUT sorok számát mutató változó nullázása, de ebben az esetben nem használhatjuk a belső INPUT rutint. Ennél jobb a BASIC-be való visszatérési címet jelölő változó törlése, de ekkor a programban ki kell védeni az összes hibalehetőséget, nehogy egy téves adat vagy válasz beírásakor a program elszálljon. Ám az ilyen programokat MERGE utasítással be lehet hívni és ki lehet listáztatni.

Egy másik védekezési mód azt használja ki, hogy a Spectrum a BASIC programnál a változókat is kimenti. Így amikor a gép beolvasta a programot, megállítod, esetleg belenyúlsz vagy RUN-nal újraindítod, a program „2 variable not found" szöveggel leáll. Ilyenkor a legelső dolgunk a behívás után a bizonyos változó(k) értékének kiíratása legyen (PRINT), csak azután boncolgassuk a programot, majd ha el akarjuk indítani, tegyük vissza az értékeket a megfelelő változókba.

Rafináltabb megoldás, ha van egy hasonló programunk:
1 REM
itt a második program fej nélküli gépi betöltője van, az A regiszterben pedig 255 helyett Ø legyen
2 CLEAR
a második program RAMTOP-ja úgy, hogy plusz sor beírását a gép ne tudja elfogadni elegendő puffer hiányában
3 RANDOMREUSR 2376Ø
A gépi kódú betöltőrutin:
LD A,Ø
LD DE,23755
LD IX
hossz (CLEAR érték° 23755)
SCF
CALL 1316
LD HL,1
LD (23621), HL }
LD (23623), HL }
az 1. sor első utasításától folytatja a rutin a futást, de meghívhatjuk itt a RUN belső rutinját is
RET

Ha tehát ez behívja a főprogramot, és azt megállítod, utána akár ki is húzhatod a gé-

pet, mert az semmit sem fogad el. Ez a rutin nem használható változókat alkalmazó programokhoz.
A második programot a következő rutinnal lehet kimenteni:

```
LD A,Ø
LD DE,23755
LD IX, hossz
CALL 1218
RET
```

Nehezíti a magnóról magnóra való másolást a kattogós program. Ennek két fajtája van:

a) sztereofelvétel egyik sávjában a Spectrum-program, a másik sávon egy egyenletes, jóval halkabb zaj, kattogás;

b) a Spectrum-jelek alá kerül ez a zaj.

Az előbbit könnyű kiszűrni: sztereomagnón a programsávot kell játszani, a kattogás így megszűnik.

A másik hibát csak a programnak egy másolóprogram által történő kimentésével szüntethetjük meg.

Szintén sok hiba forrása lehet az ún. turbós programok magnóról magnóra másolása, hiszen itt még pontosabb jelbeállítás szükséges. Ezeket viszont saját behívóprogramunk kis átírásával szintén le lehet másolni. Ám helyes, ha megvizsgáljuk ezeket a behívórutinokat, hiszen a gyorsabb kitöltés és beolvasás sok Spectrum-programozó vágya. De az ilyen gyorsabban kitöltött programokat nem szabad gyenge minőségű kazettára, rossz állapotú magnóra (poros fej, gyenge motor stb.) kitölteni.

Még három rövid megállapítást tennék. Gépi kódú programok írásakor az IY regiszterre vigyázni kell, belső rutinok használják. Alapértéke 5C3AH.

A másik probléma, hogy például ha turbó beolvasó (LOADER) programot írunk, az OUT(254),N-nél nehogy a 4-es bitet (hangszóró) bekapcsoljuk, mert ha a szalagról egy jelet kap, és a hangszóró is szól, a gép „megfagy".

Harmadszor: gépi kódban programozóknak javaslom egy rutin elkészítését, amellyel gépi kódú programokat is leállíthatunk. IM2-es megszakítási módban kell egy megszakításkezelő rutint írni, amely az 1F54H-n kezdődő BREAK lenyomását vizsgáló rutint hívja. A CY értéke nulla, ha le van nyomva a Caps Shift + Space. Egyébként a CY értéke egy. Ez a rutin csak az A regisztert használja. A BREAK vizsgálat után hívjuk meg a 38H rutint, amely a belső megszakítást kezeli. Ez a tv-figyelő bájtokat növeli, és a belső billentyűzetvizsgáló rutint is meghívja. Ez a rutin a veremben tárolja az általa használt regisztereket, és visszatérés előtt visszatölti kezdeti értékeiket. Mivel ez megszakítást kezelő rutin, a regiszterek tartalmát el kell tárolni a verembe, majd a rutin végén vissza kell hívni tartalmukat.

Végül egy kis rutin, amellyel sok jó dolgot ki lehet hozni. Írjunk egy 1-es sorszámú sort REM-mel és 54 darab A betűvel, majd ide töltsük a gépi kódú programot.

Próbáljuk ki ezzel a kis programmal:

```
 1 REM
   a gépi kódú rutin
1Ø FOR A=Ø TO 7Ø3:
   POKE 22528+A,PEEK
   (A): NEXTA
2Ø POKE 23812,Ø:REM
   ezt változtatva
   más-más eredményt ka-
   punk
3Ø RANDOMIZE USR 2376Ø
```

A gépi kódú program:

```
CMK:  LD HL,22528
      LD BC,768
      LD D,Ø
CIKL: LD A,(PAPER)
      CP (HL)
      JR Z,CMK 1
      DEC (HL)
      LD, D,1
CMK1: L O A,(23624)
      SRL A
      SRL A
      SRLA A
      AND 7
      BIT O,(HL)
      JR Z,NULLA
EGY:  SET 4,A
      SET 3,A
NULLA:OUT (254),A
      INC HL
      DEC BC
      LD A,C_
      OR B
      JRZ, CIKL
      LD A,D
      OR A
      RET Z
      JR CMK
PAPER: NOP
```

SZILVÁSI LÁSZLÓ

PTA–4000 PROGRAMOK

A PTA—4000 (Sharp PC—1500) típusú géphez két alkalmazói programot közlünk. Az első egy „Monitor" nevű, BASIC-ben írt egyszerű gépi kódú program, a másik, a „Futkosó" egy játékprogram.

MONITOR

A program segíti a gépi kódban vagy haladó BASIC-ben való programozást. Segítségével egyszerűvé válik a bájtok olvasása, esetleg írása, vizsgálhatjuk például a programtárolást vagy a képernyőmemóriát.

A program DEF A-val indítható. A kezdőcím beírása után a következőt láthatjuk:

ØØ	16581	9	B9	ØF
Az előző bájt tartalma	A bájt címe	A bájt ASCII kódja	A bájt tartalma	A következő bájt tartalma

A középső bájt értékét átírhatjuk bármely ØØH-tól FFH-ig terjedő hexadecimális számmal. A címet növelni a →, csökkenteni a ← billentyűvel lehet. A programból való kiszálláshoz az OFF-ot használhatjuk.

```
1Ø1Ø "A":WAIT Ø:CLS:INPUT"
     KEZDŐCÍM:";K
1Ø2Ø A$="Ø123456789ABCDEF":CLS
1Ø3Ø B=PEEK(K—1):C=PEEK
     K:D=PEEK(K+1)
1Ø4Ø B$=MID$(A$,INT(B/16)+1,1)
     +MID$(A$,B—INT(B/16)*16+1,1)
1Ø5Ø C$=MID$(A$,INT(C/16)+1,1)
     +MID$(A$,C—INT(C/16)*16+1,1)
1Ø6Ø D$=MID$(A$,INT(D/16)+1,1)
     +MID$(A$,D—INT(D/16)*16+1,1)
1Ø7Ø PRINT B$;"␣␣␣";K;"␣␣␣␣"
     :CURSOR 13:PRINT CHR$ C:
     CURSOR 17:PRINT C$;
     "␣␣␣␣";D$:BEEP 1
1Ø8Ø Q$=INKEY$:Q=ASC Q$
1Ø9Ø IF Q=12 OR Q=8 LET
     K=K—(Q=8)+(Q=12):GOTO 1Ø3Ø
11ØØ IF Q=15 END
1111Ø IF Q<48 OR Q>57
     AND Q<65 OR Q>7Ø GOTO 1Ø8Ø
1120 BEEP 1
1130 CURSOR 17:PRINT CHR$ 255;
     CHR$ 255: FOR T=Ø TO 5: NEXT T
     :CURSOR 17:PRINT Q$;
     "␣":B$=INKEY$:R=ASC R$
1140 IF R<48 OR R>57
     AND R<65 OR R>7Ø GOTO 113Ø
1150 S$=Q$+R$:CURSOR 17
     :PRINT S$:BEEP 1
1160 Q=Q—(48+(Q>64)*7)
     :R=R—(48+(R>64)*7):S=Q*16+R
1170 POKE K,S:K=K+1:GOTO 1Ø3Ø
```

FUTKOSÓ

A játékprogram célja: a képernyőn végighaladni egy ponttal balról jobbra, kikerülve az útba eső pontokat. A felső sorokban gyorsabb a pont, ezért ott haladva a gép több pontot ad.
A program DEF Z-vel indítható. A pont irányításához két billentyűre van szükség, ezeket bekéri a gép.

```
 1Ø "Z":WAIT:
    PRINT" * * * FUTKOSÓ * * *"
 2Ø INPUT" Fel:";F$,"Le:";L$
 3Ø PRINT" ENTER-rel indul"
    :P=Ø:C=Ø:J=5:A=3
 4Ø WAIT Ø:C=C+1:PRINT"
    ";C;".pálya ";P;" pont ":PAUSE:CLS
 5Ø FOR X=15 TO 14Ø STEP
    J:M=2 ^(RND(8)—1)
    :MM=2 ^(RND(8)—1):
    M=M OR MM:GCURSOR X:
    GPRINT M:NEXT X
 6Ø FOR T=4 TO 152:A$=INKEY$
 7Ø A=A—(A$=F$)+(A$=L$)
 8Ø A=A—(A=7)+(A=—1)
 9Ø M=POINT T:MM=2 ^A:GCURSOR T
    :GPRINT M OR MM
1ØØ X=POINT(T+1):N=MM
    OF X:IF N=X GOTO 13Ø
11Ø P=P+8—A:NEXT T:J=J—1:IF
    J=1 LET J=5
12Ø GOTO 4Ø
13Ø WAIT:PRINT" Ütközés történt ":PRINT"
    Pontszám: ";P
14Ø INPUT "Új játék? (I)";A$
    :IF A$="I" GOTO 3Ø
15Ø END
```

BALÁZS GYÖRGY

SPRITE KIRAKÓ-BETÖLTŐ PROGRAM

Az alábbi kis assembly program azoknak segíthet, akik BASIC-ből kívánják a sprite-okat kezelni. A program funkciója: egy tetszőleges sprite elhelyezése a képernyőn.

A rutin három bemenő adatot vár egy SYS utasítás után, a SYS paraméteren kívül. Az első szám a kihelyezendő sprite száma. Ennek értéke értelemszerűen 0—7-ig terjedhet. A második adat az előbb kiválasztott sprite leendő x koordinátája. A program ide egy kilenc biten ábrázolható számot vár. A harmadik szám az y koordináta. Ez egy egybájtos adat. Tehát a szintaxis:

SYS49152,<mob száma>,
<x-koord>,<y-koord>

A program előnye a BASIC megfelelőjével szemben, hogy futási sebessége lényegében nem függ a bemenő paraméterektől, gyorsabb és megszabadít minket a sprite-ok kellemetlen villódzásaitól.

Ez a kis rutin nem versenghet a nagyobb BASIC-bővítések grafikus utasításaival, ezért inkább oktató célja miatt lehet érdekes.

TASS ZOLTÁN

```
5 REM  £££££££££££££££££££££££££
6 REM  £                       £
7 REM  £  SPRITE KIRAKO - BETOLTO  £
8 REM  £                       £
10 DATA32,241,183,138,10,72,32,253,174,3
2,235,183,104,168,138,153,1,208,165,20
20 DATA153,0,208,152,74,168,173,16,208,2
5,43,192,166,21,208,3,89,43,192,141,16
30 DATA208,96,1,2,4,8,16,32,64,128
50 FORT=0TO50:READD:S=S+D:POKE49152+T,D:
NEXT:IFS<>5392THENPRINT"ADAT HIBA !"
READY.
```

```
                    ;   SPRITE KIRAKO
                    ;
20:     C000                .OPT OO,P,P4
30:     C000                #=   49152
40:     C000        BYTEBE  =    $B7F1    ; EGYBYTE-OS ADAT BEVETELE
50:     C000        VESSZOIG =   $AEFD    ; VESSZO UTAN POZICIONALAS
60:     C000        CIMBYTBE =   $B7EB    ; EGY CIM ES EGY BYTE BEVETELE
70:     C000        XCOORD  =    $14      ; KETBYTE-OS ADAT IDE KERUL($14/$15)
80:     C000        MSB     =    $D010    ; X-COORDINATAK 8. BITJEI
90:     C000        MOBHELY =    $D000
100:    C000 20 F1 B7           JSR  BYTEBE  ; HANYAS SZAMU SPRITE
110:    C003 8A                 TXA          ;
120:    C004 0A                 ASL          ; KETSZERESE A ZSAKBA
130:    C005 48                 PHA          ;
140:    C006 20 FD AE           JSR  VESSZOIG  ; X,Y COORDINATAK BEVETELE
150:    C009 20 EB B7           JSR  CIMBYTBE  ; Y-COORDINATA AZ X REGISZTERBE
160:    C00C 68                 PLA
170:    C00D A8                 TAY
180:    C00E 8A                 TXA
190:    C00F 99 01 D0           STA  MOBHELY+1,Y  ; A SPRITE Y-COORDINATAJA
200:    C012 A5 14              LDA  XCOORD
210:    C014 99 00 D0           STA  MOBHELY,Y    ; X-COORDINATA ALSO 8 BITJE
220:    C017 98                 TYA
230:    C018 4A                 LSR
240:    C019 A8                 TAY
250:    C01A AD 10 D0           LDA  MSB
260:    C01D 19 2B C0           ORA  TABLA,Y      ; AKTUALIS BIT KIGYUJTASA
270:    C020 A6 15              LDX  XCOORD+1
280:    C022 D0 03              BNE  KINT         ; HA KELL UGYMARAD,
290:    C024 59 2B C0           EOR  TABLA,Y      ; KULONBEN EL KELL OLTANI
300:    C027 8D 10 D0 KINT      STA  MSB          ; IGY KERUL A HELYERE
310:    C02A 60                 RTS
320:    C02B 01         TABLA   .BYTE%00000001
330:    C02C 02                 .BYTE%00000010
340:    C02D 04                 .BYTE%00000100
350:    C02E 08                 .BYTE%00001000
360:    C02F 10                 .BYTE%00010000
370:    C030 20                 .BYTE%00100000
380:    C031 40                 .BYTE%01000000
390:    C032 80                 .BYTE%10000000
```

Assemblerek, cross-assemblerek

1. (Cross-)assemblerek és a rendszermodellezés

Nyolcrészes sorozatunk a mikroszámítógépek assemblereiről, cross-assemblereiről szól. Célja, hogy a cross-assemblerek példáján keresztül megismertesse az olvasót az assembler programok működésével. A bemutatáshoz a rendszermodellezési eszközöket használjuk fel, s így készítjük el az assemblerek működésének egy szabványos algoritmusát, modelljét.

A sorozat részei:

A mikroprocesszorok hatalmas arányú terjedése következtében, amelyben a teljesítményük növekedése és az áruk csökkenése játssza a fő szerepet, a különböző célú, teljesen vagy részben számítástechnikai eszközök mind számukat, mind típusukat tekintve rendkívül gazdag és egyre bővülő világot alkotnak. Igaz azonban az, hogy az újabb mikroprocesszorokkal épített berendezésekre érdemes és szükséges is az éppen elavuló mikroprocesszorokra írt szoftverkincs átmentése, hiszen maga a szoftver nem válik feltétlenül korszerűtlenné azáltal, hogy egy régebbi hardveren valósították meg.

Az egy-egy berendezésen bevált szoftvernek egy másik berendezésre való átvitele magasabb szintű nyelvek esetében többékevésbé megoldott. Nagyobb sebességkövetelménynek eleget tevő feladatoknál nélkülözhetetlen a szoftverátvitelnek a gépi nyelvhez közeli szinten való megoldása.

Így, annak ellenére, hogy a BASIC értelmező megléte szinte általános — sőt, a laikus felhasználó szinte kizárólag ezen keresztül használja a gépét —, s a magasabb szintű nyelvek alkalmazását lehetővé tevő, illetve többnyire ezeknek csak egy szűkített változatát megvalósító fordítóprogramokra is bőven van példa, a mikroszámítógépek legáltalánosabb programozási nyelve továbbra is az assembly szintű nyelv maradt.

A mikroszámítógépek programozási eszközein belül tehát meghatározó jelentősége van az assemblereknek, azaz a gépi kódhoz közel álló szimbolikus nyelven írt programok fordítására használt programoknak.

Az assembler és a ritkábban létező szerkesztő-áthelyező programok sokszor nem magán a mikroszámítógépen, hanem más, legtöbbször nagyobb teljesítményű számítógépeken futnak. Ebben az esetben beszélünk cross-assembler programról. A cross-assemblerek kialakítását az indokolja, hogy a mikroprocesszorokat — mikroszámítógépeket elsődlegesen egy-egy adott területre korlátozott feladatok megoldására alkalmazzák. Az ilyen rendszerek pedig többnyire nem is tudják azokat a feltételeket biztosítani, amelyek a programfejlesztés kényelmes felhasználói környezetéhez lennének szükségesek. Sőt előfordulhat, hogy még az önálló programozhatósághoz szükséges minimális feltételeket (megfelelő tárnagyság, háttértár létezése) sem elégítik ki. A nagyobb gépen végzett programfejlesztés előnye abban rejlik, hogy a fordítóprogram beilleszkedhet egy már meglévő operációs rendszerbe, felhasználhatja annak minden szolgáltatását (szerkesztőprogramokat, segédprogramokat stb.), s ez jelentősen megkönnyítheti a programfejlesztési munkát.

Nem szabad persze megfeledkezni arról sem, hogy az előbb említett „fogyatékosságokat" mutató mikroszámítógépek ellenkezőjére, az elegendően nagy központi tárral, viszonylag nagy kapacitású (hajlékony- vagy kemény-) mágneslemezes háttértárral, fejlett operációs rendszerrel rendelkező rendszerekre is egyre több példa akad. Sőt a fejlődés egyértelműen az ilyen rendszerek irányába mutat. Az ilyen berendezéseknél a tágabb értelemben vett keresztfordítók jelentősége fog megnövekedni, az olyan programoké tehát, amelyek az egyik gép assembly nyelvén írt programokat képesek egy másik gép assembly nyelvére átfordítani.

A különböző típusú mikroprocesszorokhoz írt assembler, illetve cross-assembler programok között igen nagyfokú hasonlóság állapítható meg, dacára a processzorok felépítésében és működésében meglévő különbségeknek. Hasonlítanak a program által elvégzendő feladatokban, a programok struktúrájában és működésében. Az egyezés alapja az egyes programozási nyelvek többségében minden meglévő szintaktikai vagy akár szemantikai eltérés ellenére megnyilvánuló logikai azonosság. Ha pedig ilyen azonosság létezik, önként adódik, hogy kíséreljük meg a fordítóprogramok általános sajátosságait megállapítani, a feladataikat és megvalósítási módjukat a lehetőség szerint „szabványosítani".

Célunk tehát az, hogy a közös tulajdonságok kiemelésével az assembler, illetve cross-assembler programok készítésének egy „szabványos" algoritmusát kialakítsuk és bemutassuk. Az algoritmus megfogalmazásánál rendszermodellezési eszközöket kívánunk igénybe venni, azaz megkíséreljük a fordítóprogramot önálló rendszernek tekintve, annak működését rögzített feladatokat végző operátorok segítségével leírni, valamint ezen operátorok között az adatmezőkön és a szekvenciális gerjesztésen keresztül megvalósuló kapcsolatokat is szabványosítani. Ezáltal olyan eszközt hozhatunk létre, amely segítséget nyújthat a mikrogép felhasználójának (cross-)assembler programja tervezésében és megvalósításában.

Miben áll ez a segítség, azaz milyen előnyöket várhatunk egy fordítóprogram — vagy általánosan egy tetszőleges programtermék — ilyen módon való modellezésétől?

Az előnyök közül hármat emelünk ki. Legfontosabbnak azt tartjuk, hogy az operátorokkal való szemléltetés hozzásegítheti a rendszerprogramozót az összetettebb programok logikájának, működésének, az adatmezők közötti kapcsolatoknak, az adatmezők időbeni változásának stb. nyomonkövetéséhez. Ahhoz, hogy maga is képes legyen egy fordítóprogramot vagy más hasonló, esetleg még nagyobb bonyolultságú programot elkészíteni, az szükséges, hogy néhány már kész programot végigtanulmányozva megértse azok belső összefüggéseit, az alkalmazott megoldási „trükköket". A programok ilyen mértékű megértéséhez (amit aztán az utasításszintű vizsgálat és értelmezés követhet) igen hasznos támogatást nyújt a rendszermodellezésből kölcsönzött eszközök alkalmazása.

Így válhat a programozó képessé arra, hogy ezeknek az eszközöknek, módszereknek az elsajátításával saját programjait gyorsabban és hatékonyabban tudja elkészíteni. Ha a tervezés során programjáról mint rendszerről gondolkodik, s előbb benne funkcionális részrendszereket (operátorokat) jelöl ki, s ezzel párhuzamosan és kölcsönhatásban az adatmezők, adatállományok struktúráját definiálja, ha rögzíti az egyes operátorok közötti aktivizálási sorrendet, a feltételeket, valamint a közösen használt adatokon keresztül megvalósuló kapcsolatokat, s ha mindezt kapcsolatábrában is megjeleníti, akkor dinamikusan tudja programját felépíteni és áttekinteni. A kapcsolatábrában az operátorok legalkalmasabb helyét próbálkozások útján tudja meghatározni. Mivel az ábrában igen könnyen végigkövethető, hogy tetszőleges operátor működése milyen hatással van az adatmezők változására és a többi operátor működésére, a tervezés során elkövetett logikai hibák jó része automatikusan napvilágra kerül, ha különböző bemenő adatokkal történő gerjesztés mellett a programozó beméri, teszteli a rendszer működését. Így például az olyan jellegű hibák kiszűrésére is alkalmas a módszer, amelyek azáltal keletkeznek, hogy a programozó elfeledkezik

valamilyen gerjesztés során előálló rendszerállapot(ok) kezeléséről.

A rendszer (program) működésének algoritmizálása tetszőleges finomságú felbontásban elvégezhető. Nyilván az út a kisszámú, összetett operátorokból felépített, egyszerű kapcsolatábrával rendelkező rendszertől a nagyobb számú, de egyszerűbb működésű operátorból álló, bonyolultabb kapcsolatábrához vezet. S miután a programozó elkészítette a végleges kapcsolatábrát, definiálta az összes résztvevő operátort (működésüket, valamint be- és kimeneti változóikat, azaz az adatmezőket), magának a programozási munkának (kódolásnak) a nehézsége és munkaigénye is jelentősen csökken, mert pontos útmutató van a kezében, hogy mit és milyen módon rendezve kell elvégeznie.

A kész programok megértésében, új programok tervezésében nyújtott segítség mellett nem kevésbé fontos a módszernek egy harmadik területen, a programok módosítása során való alkalmazása. Ha egy programot valamilyen okból meg kell változtatni, a kapcsolatábrában könnyebben megállapítható, hogy az előírt módosításokat melyik operátor(ok)ban kell véghezvinni. Az operátorok cserélhetők, funkcióik kiterjeszthetők; s a változtatások következményei a rendszer működésében azonnal ellenőrizhetők. Így nagyobb biztonsággal (megbízhatósággal!) hajthatók végre egy már működő, bevezetett programban a korántsem veszélytelen módosítások.

Feladatunkat rögzítettük, röviden ismertettük az alkalmazni kívánt módszert, s a tőle elvárható előnyöket. Néhány olyan feltételt — vagy feltételezést — kiindulópontként fel kell még sorolnunk, ami az általunk tárgyalt (cross-)assembler működését és algoritmusát erősen befolyásolja. A feltevések gyakorlatilag mégsem korlátozzák példánk általánosságát; sőt több esetben éppen az általánosíthatóság érdekében szigorúbb megkötéseket vettünk fel, mint az egy-egy egyedi esetben szükséges lenne.

Így mindjárt az első megkötés is olyan, hogy a megvalósítást általánosabb érvényűvé teszi, elhagyása egyszerűsítené helyzetünket. (Cross-)assembler programunkról ugyanis feltételezzük, hogy nem közvetlenül betölthető tárgyprogramot állít elő, hanem szükség van egy ún. betöltőprogramra, amely a tárgyprogram egyes bejegyzései alapján elkészíti és betölti a mikrogép tárába a futásra kész programot. A tárgyprogramot úgy állítjuk össze, hogy nem zárjuk ki egy több független programmodul összekapcsolására és az eredeti tárcímekről más tárcímekre való áthelyezés megvalósítására alkalmas ún. szerkesztő-áthelyező (linkage editor vagy relocatable loader) program használatát sem.

A (cross-)assembler programot elsősorban a kisgépek sajátosságainak figyelembevételével tervezzük, bár természetesen nincs elvi különbség a mini- és a nagyszámítógépen futó fordítóprogramok között. Ez esetünkben két dolgot jelent: egyrészt a központi tár méretéről feltételezzük, hogy nem elegendően nagy a fordítás során keletkező különféle táblák tárolására, és így ezeket háttértáron kell elhelyeznünk, másrészt úgy tekintjük, hogy lehetőségünk van a háttértárak fizikai szintű kezelésére, azaz például mindig rendelkezésünkre állnak a lemezen lévő adatállományok aktuális hozzáférési lemezcímei.

A következő részekben megvizsgáljuk a kiválasztott assembly szintű nyelv sajátosságait, foglalkozunk azzal, hogy milyen feladatokat ró a makróhasználat engedélyezése a fordítóprogramra; bemutatjuk, hogy milyen módon tekinthető a (cross-)assembler program egy modellezési eszközökkel vizsgálható rendszernek; definiáljuk az operátorokat és működési módjukat. Végül pedig kapcsolatábra segítségével ismertetjük a rendszer működését.

Hangsúlyozni kívánjuk, hogy a bemutatásra kerülő rendszer kidolgozásakor két legfontosabb szempontként a (cross-)assembler programok belső felépítésének, működésének megértését és a kiválasztott módszer alkalmazhatóságának, hasznosságának bizonyítását tartottuk szem előtt. A bemutatásra kerülő (cross-)assembler program csak egy a sokfajta lehetséges megoldás között, amely nem törekszik arra, hogy programozástechnikai szempontból a lehető leghatékonyabb legyen. Létezhetnek és léteznek is nála egyszerűbb, gyorsabb, valamely erőforrás kihasználása szempontjából kedvezőbb megoldások. A forma tehát önkényesnek tűnhet, a lényeg azonban a példának a fenti két szempontból értelmezett tartalma!

Az előzőekben feltűnhetett, hogy nem tettünk különbséget az assemblerek és a cross-assemblerek között, illetve az elmondottakat mindkettőre vonatkoztattuk. Ez így helyes, mert elvileg sem funkciójukban, sem működésükben nincs különbség a kétfajta fordítóprogram között. Amit az egyikről a rendszermodellezési eszközökkel megállapíthatunk, érvényes lesz a másikra is. A cross-assembler nem más, mint az assembler egy speciális megjelenési formája.

Eltérésük mindössze annyi, hogy az assembler arra a gépre készül, azon a gépen működik, amelyiknek assembly nyelvén írt programok fordítása a feladata, a cross-assembler esetében ezzel szemben különválik az a számítógép, amelyiken a fordító fut, attól a számítógéptől, amelyiken a lefordított programok működnek. Persze ebből a tényből a kétfajta fordítóprogram készítésére nézve egy nagyon lényeges gyakorlati különbség következik: a cross-assemblerek egy másik gép assembly nyelvén íródnak, és így ennek a gépnek a létező assembler fordítóprogramja segítségével fordítódnak le és válnak betölthető, futtatható programokká; az assemblerek készítésekor pedig nincs mivel lefordítanunk a saját assembly nyelven írt programot, mert nincs még fordítóprogramunk, hiszen éppen ezt akarjuk létrehozni.

Hogyan lehet tehát egy assemblert lefordítani, tesztelni és „belőni"? Egy módszer kézenfekvőnek látszik: készítsünk előbb az adott számítógéphez egy másik gépen működő cross-assembler programot, majd írjuk meg újból a fordítót, most már a saját assembly nyelven, és fordítsuk le a cross-assembler segítségével. A tesztelés szintén a „vendéglátó" számítógépen történhet, ha létezik még egy szimulátorprogramunk is, azaz egy olyan program, amely lehetővé teszi, hogy a mikroszámítógép gépi kódjára lefordított programot a „vendéglátó" gép is értelmezni és végrehajtani tudja.

Ez a gyakorlati akadály mondanivalónk lényegét nem érintheti. Így a következőkben is egészen addig, amíg csak lehet, együtt tárgyaljuk majd az assembler és a cross-assembler sajátosságait, és csak ott teszünk közöttük különbséget, ahol ez elengedhetetlen.

VÁRGEDŐ TAMÁS

PÁLYÁZAT

A számítástechnikai területen tevékenykedő egyéni alkotók támogatására pályázatot ír ki dr. Simonyi Endre (Budapest, Trencsényi u. 19. 1125). A pályázat díjait Simonyi dr. adja, a pályázatok bírálását a szükséges szakértők bevonásával végzi.

A pályázat feltételei

1. Pályázni lehet bármilyen, számítógépekkel kapcsolatos megoldással (hardver, szoftver egyaránt), ami kielégíti az alábbiakat:

1.1 Hardver esetén feleljen meg a szabadalmaztatás, szoftver esetén a szerzői jogi védelem tartalmi követelményeinek a pályázat.

1.2 A pályázó(k) legyen(ek) a pályázat tárgyának kizárólagos tulajdonosa(i).

1.3 A pályázat tárgya alkalmazható legyen (átdolgozás nélkül) nagy sorozatban gyárott (vagy jövőben gyártásra kerülő) külföldi számítógéprendszerben.

1.4 A megoldásról 3 példányban előnyleírást kell küldeni, amely feleljen meg a találmányi bejelentések megfelelő részeivel kapcsolatos követelményeknek, vagyis tartalmazza azt, hogy más ismert megoldásokhoz képest milyen előnyöket biztosít.

Ezzel egyidejűleg megküldendők még a pályázó(k) személyi adatai (név, lakcím, személyi szám), felbélyegzett, megcímzett válaszboríték, valamint a 200,— Ft összeg befizetését igazoló utalvány. Az összeg a pályázat lebonyolítására fordítódik.

1.5 Az 1.4 pont alapján kiválasztásra kerülnek a figyelemre méltó megoldások. Ezek pályázóit értesítjük. Az értesítettek 15 napon belül megküldik: a teljes részletességű leírást (amely alapján szakember megvalósíthatja) 3 példányban, a megoldást (hardver esetén: az eszközt; vagy, ha nincs még, úgy működő modelljét; vagy ha ez sincs, úgy a megoldás alkalmasságának igazolására szolgáló más bizonyítékot; szoftver esetén: a szükséges magyarázatokkal ellátott listát és futtatásra közvetlenül alkalmas hordozón a szoftvert; vagy, ha ez nincs, úgy a megoldás alkalmasságának más tárgyi bizonyítékát), további 300,— Ft összeg befizetését igazoló utalványt és a következő nyilatkozatot: Alulírott(ak) ... a ... megnevezésű megoldás alkotója(i) és kizárólagos jogú tulajdonosa(i) tudomásul veszem (vesszük), hogy amennyiben a pályázaton a megoldást díjazzák, úgy annak teljes jogú tulajdonosa dr. Simonyi Endre lesz minden további térítés nélkül. Amennyiben azonban a megoldást Simonyi dr. értékesíti, és az értékesítés részére évi 100 000,— Ft összeget meghaladó tiszta hasznos eredménnyel jár, úgy ennek 50%-a az alkotót illeti meg.

2. A pályázat határideje 1986. IV. 30. Az elbírálás első menete lezajlik 1986. V. 31-ig, második menete 1986. VII. 31-ig. A végeredményről az érdekeltek 1986. IX. 30-ig értesítést kapnak.

3. A pályázat öt fődíja egy-egy mikroszámítógép (Dragon 64, 32; Simon 68; Microstar típusúak). A díjak teljes listájának egy példányát 50,— Ft összeg befizetését igazoló utalvány és megcímzett, felbélyegzett válaszboríték ellenében megküldjük.

A pályázattal kapcsolatos mindennemű levelezést, befizetést, érdeklődést stb. a következő címre kérjük: dr. Simonyi Endre, Budapest, Trencsényi u. 19. 1125.

SZVT Szervezési Szakosztály
Számítógépalkalmazási Munkabizottság
SZVT Békés megyei Szervezete

PÁLYÁZATI FELHÍVÁS

1986. szeptember 16—19 között Szarvason rendezendő IV. Számítástechnikai Szervezési Akadémia előkészítése céljából — az érintett tárcák és a KSH támogatásával — pályázatot hirdetünk *kidolgozott és az Akadémia helyszínén bemutatható számítástechnikai szervezési megoldás* szerzői számára.

A pályázat témája: többmunkahelyes rendszerek mikroszámítógépek felhasználásával.

A pályázat értékelésénél a bíráló bizottság előnyben részesíti azokat a bevezetett és működő megoldásokat, amelyek

- szélesebb körben szoftveráruként értékesíthetők,
- valamely gazdasági folyamat hatékonyságának növelését segítik,
- referencia alkalmazójuk biztosított.

A pályázatnak tartalmaznia kell

- a feladatot vagy problémát, amelynek megoldása szervezési és számítástechnikai szempontból sikeres volt,
- a gyakorlati megoldás számítástechnikai módját,
- a megoldás bevezetéséhez szükséges szervezési feladatokat, alkalmazott módszereket,
- a megoldás alkalmazási tapasztalatait, gazdasági előnyeit, ill. szükséges feltételeit,
- a referenciahely megjelölését és korreferátor nyilatkozatát,
- legfeljebb 20 soros annotációt a sikeres megoldásról.

A pályázat terjedelme a 25 gépelt oldalt ne haladja meg!

Pályadíjak: I. díj 15 000,— Ft
II. díj 10 000,— Ft
III. díj 5 000,— Ft

A bíráló bizottságnak joga van a díjak átcsoportosítására. Egyes díjak kiadásától eltekinthet, ha arra érdemes javaslat nem érkezett, ugyanakkor érdemes javaslatok esetén a díjak számát és összegét emelheti is.

A pályázat jeligés. A pályázathoz jeligével ellátott lezárt borítékban mellékelni kell

- a szerző(k) nevét, pontos címét, munkahelyét, beosztását, telefonszámát,
- több szerző esetén a pályadíj megosztásának arányát,
- nyilatkozatot arról, hogy a pályázaton való részvételnek jogi akadálya nincs.

A pályamunkát három példányban kell beküldeni az SZVT Titkárságra (Budapest VI., Anker köz 1—3. félem. 4. 1061) „IV. Számítástechnikai Szervezési Akadémia" és a jelige megjelölésével a feladó megnevezése nélkül.

Beküldési határidő: 1986. április 30.

A pályamunkák értékelésére 1986. május hóban kerül sor. A IV. Számítástechnikai Szervezési Akadémián a díjazásra javasolt pályamunkák — a szokásos előadói díj mellett — előadásra és bemutatásra kerülnek. A díjak odaítéléséről és a különdíjak kiosztásáról a pályamű, az előadás és bemutató együttes értékelésével a bíráló bizottság az Akadémia utolsó napján a helyszínen dönt, a díjak kiosztására is ott kerül sor.

A díjazott pályaműveket az SZVT jogosult saját területén felhasználni, írásos publikáció esetén a MTESZ előírása szerinti szerzői díj térítése mellett. Ezek azonos tartalommal az SZVT más pályázatára nem nyújthatók be.

A nem díjazott és dicséretre nem javasolt pályaműveket, a beadási határidőn túl érkezett anyagokkal együtt a bíráló bizottság jegyzőkönyv felvétele mellett megsemmisíti.

Pályamű visszaadását nem vállaljuk!

További felvilágosítás az SZVT Titkárságon (222-093) szerezhető be.

AZ AKADÉMIA SZERVEZŐ BIZOTTSÁGA

PÁLYÁZATOT HIRDETÜNK
SZEMÉLYI SZÁMITÓGÉPEK
humánus és rentábilis alkalmazásának segitésére

I.Szoftverkészitők és -felhasználók

1.Szorgalmazzuk általános értelemben vett művelődési, a művelődéshez kapcsolódó szoftverek készitését:
- -művelődési intézmények ügyvitelének segitése,
- -művelődési, oktató és humán jellegű szakfeladatok segitése,
- -lakossági célú programkészlet bővitése.

2.A lakossági célú programokon belül külön szorgalmazzuk olyan szoftverek készitését, amelyekkel számitógépes szolgáltatások végezhetők az egyéni--családi életvitel kiegyensúlyozásához, esetleges életmódváltásához, egyfajta önmenedzselési eszközként.

Kalkuláló életviteli egyensúlyprogramok:
- -háztartási pénzügyek,
- -érzelmi egyensúly,
- -életstratégia tervezés, stb.

Ügyintézést segitő, tájékoztató programok:
- -igazgatási ügyintézést könnyitő tájékoztatások,
- -takarékpénztári betételhelyezés és hiteligénylés lehetőségei
- -lakossági szolgáltatók cimjegyzéke, szolgáltatószolgáltatásaik, stb.

Ezen programötletek felvetésével a célunk az önálló, személyes életvitelhez szükséges képességek, ismeretek rendszerezése úgy, hogy azok a rászorulók számára mindinkább áttekinthetőek, hozzáférhetőek legyenek a háztartásgazdálkodás, a mindennapos lakáskultúra, az egészség, a nevelés, a kiskert, stb témakörökben.

Jelentkezni lehet az ötlettől kezdve a felhasználásra kész programig bármely fázisban levő munkával, szolgáltatási tervekkel, stb.

Pályázati jelentkezésként az elképzelések irásos vázlatát kérjük.

A pályamunkák készitőinek konzultációs lehetőséget biztositunk, és szükség esetén az egyes részfeladatok elvégzéséhez segitséget, tanácsadókat ajánlunk. Az elfogadott pályamunkákat esetenként dijazzuk, de főleg hirdetési lehetőséget, forgalmazási csatornát biztositunk a számukra.

II.Információt szolgáltatók

Olyan cégek jelentkezését várjuk, amelyek információira a lakosságnak szüksége lehet, a számitógépek rentabilis üzemeltetésére vállalkozók ügyfeleiként. Keressük tehát a kapcsolatot általában a lakossági szolgáltatókkal:
- -művelődési programok kinálóival,
- -igazgatási szervekkel (ügyfélirodáikkal),
- -kereskedelmi hálózatokkal,
- -pénzintézetekkel, stb.

III.Számitógépek rentábilis működtetése

1.A számitógépoktató központok és referenciahelyek rentábilis üzemeltetése pályázatra olyan művelődési házak (vagy más, a szolgáltatások jellegével, minőségével kapcsolatos kikötéseinket vállaló felek) jelentkezését várjuk, amelyek a vonzáskörzetükben:
- -képesek felvállalni a művelődési házak munkatársainak informatikai oktatását, képzését, központi tematikájú és állami vizsgabizonyitványt nyújtó tanfolyamokkal,
- -élenjárnak a saját ügyviteli feladataik számitógépes bonyolitásában, és ezt referenciaként más intézmények számára is megtekinthetővé, megismerhetővé teszik,
- -egyfajta menedzseri szerepkörben, esetleg idővel közvetlenül is, segitik a kisebb művelődési házak informatizálását egy önmagában is rentábilitásra törekvő szolgáltatásként.

2.A számitógépek rentábilis használata pályázatra olyan művelődési házak (és más, az 1-es pontban utalt) intézmények jelentkezését várjuk amelyekben fokozottan súlyt fektetnek a számitógépeknek a sokoldalú, idővel a beruházást is fedező, rentábilis hasznositására:
- -ügyviteli-számviteli teendők könnyitése,
- -művelődési szakfeladatok segitése,
- -lakossági szolgáltatások kialakitása, működtetése.

A jelentkező intézmények munkatársait kérjük, hogy pályázati jelentkezésükben röviden irják le jelenlegi helyzetüket, céljukat, számitógépes munkatervüket, és lehetőleg tényszámokban is kifejtve azt, hogy hogyan szeretnék javitani, gazdaságosabbá tenni a tevékenységüket a személyi számitógépekfelhasználásával.

Azon intézményeket, amelyek pályázatát elfogadjuk, bázishelyeinknek tekintjük,és valamennyi, a bázishelyeknek szánt szolgáltatást számukra biztositjuk. Egyebek között gazdálkodási és felszerelési tanácsadással segitjük munkájukat, közeget teremtünk az egymás közötti információcseréjükhöz, hirdetési lehetőséget biztositunk, és rendszeresen ellátjuk őket a számitógép felhasználók számára fontos aktuális információkkal.

IV.A pályázat támogatása

Végül: örömmel fogadnánk a kezdeményezésünkkel egyetértők anyagi vagy más természetű hozzájárulását (támogatás vagy akár visszafizetendő hitelkeret formájában):
- -a pályázat bonyolitásához,
- -az egyes témakörökhöz, stb.

V.Jelentkezés a pályázatra

A pályázatra folyamatosan lehet jelentkezni.
A pályázat nem kötődik sem géptipushoz sem gépméretthez.
A pályázatok első nyilvános vitája Székesfehérvárott lesz 1986.április 7-13. között.
Cimünk: Népművelési Intézet TTM Csoport Bp 1251 Pf 33 Corvin tér 8.
Telefenunk: 388-122/34,70; 388-176.

JÁTÉK A Betűkkel, vagy AMITAS

Szaklapjainkban, a Mikromagazinban, az Ötletben és a Számítástechnikában már sokszor írtam arról, hogy nincs olyan magyar gyártmányú vagy itthoni kereskedelemben kapható mikroszámítógép, amely képernyőjén, nyomtatóján a *teljes* — és ezt a szót hangsúlyozom — magyar betűkészletet képes lenne kijelezni. Természetesen itt nem arra gondolok, hogy a betűket, mint rajzokat nem képes előállítani. Ezt minden rajzmegjelenítésre alkalmas gép tudja.

Hiába!

Továbbra is jelennek meg cikkek, hirdetnek termékeket, tisztán szoftveres és hardvermódosítással kiegészítetteket egyaránt, amelyek elsősorban a C64, C16, VC20, Spectrum gépekre ígérik a „teljes magyar betűkészlet", „az összes ékezetes betű" stb. megvalósítását. Mindezt azután, hogy a Mikromagazinban és az Ötletben bemutattam, mi kell ahhoz, hogy *a valóban teljes, megfelelő méretű és elhelyezésű* betűkészlet megvalósulhasson. *Bizonyítottam, hogy ez egyiken sem lehetséges!*

Mivel ez eredménytelen maradt, megismétlem az Ötletben leírtakat, és felhívom a figyelmet arra, hogy ez a tevékenység megtévesztő, kimeríti a tisztességtelen verseny fogalmát.

A SIMON 68-nál vagy a Dragon 64-nél meg lehetett oldani ezt a problémát, de a többi gépnél nem. Ennek oka a hardverlehetőségek különbsége. Az említett két típusnál a karaktereket az MC 6845 képernyővezérlő, illetve a 6847 típusú videogenerátor és egy 2716 típusú EPROM állítja elő. Ezek viszont képesek 8 × 16 pontból álló, ún betűmátrixot használni. (Erről részletesen írtam a Mikromagazin 1984/5. számának 31. oldalán.) Ezzel szemben a Primo 6 × 12, a Commodore és a Sinclair gépek 8 × 8, a HT—1080Z teljes vagy fél sor kijelzéstől függően 4 × 12 vagy 8 × 12 pontos mátrixot használ, amiben a feltétlenül kihagyni szükséges betű és sorköz miatt nem fér el minden betű, és nem helyezhető el a magyar helyesírás szabályai szerint. (A bővített memóriájú HT változatnál — kismérvű torzítással — a fél sor kijelzésnél megvalósítható.)

Nézzük meg a cikkemben szereplő betűpárt: Ő és g. A betű és sorköz levonása után az én példámban 7 × 12, míg a Primónál 5 × 8, a Commodore és a Sinclair gépeknél 7 × 7, úgyhogy a sorközre a jól olvashatóságot biztosító 4 pont helyett 1 pont marad. A hivatkozott cikkben látható, hogy a SIMON 68-nál, a Dragon 64-nél a betűk előállítása és elhelyezése is könnyen megvalósítható. De a Primónál vagy a többinél ez lehetetlen! Legfeljebb kisméretű nagybetűkkel, feljebb tolt alsószáras, összenyomott mássalhangzókkal, egyes betűk elhagyásával, vagyis a magyar helyesírás sutba dobásával, úgy, ahogy az egyik reklámozott termékkel a cikk címét előállítottuk.

Természetesen lehetőség van a már említett rajzos betűelőállításra például úgy, hogy több betűmátrixot összekapcsolunk — már ahol ezt egyáltalán lehet —, de akkor vagy az egy sorba írható karakterszám 16-20 karakterre, vagy a sorok száma 8-12-re csökken. Ez tehát szintén nem megoldás!

Miért foglalkoztat újra és újra ez a probléma? Azért, mert *ilyen gépekkel nem lehet helyesírást tanítani*, sem helyesen és érthetően írni. Tessék tehát megtanítani az itthon forgalmazott gépeket magyarul!

Megvalósítható ez a követelmény? Igen, mert először is mindkét fajta IC olcsó, nagy tömegben gyártott (a 6845-öt Bulgáriában is gyártják), másodszor nemcsak 68XX vagy 65XX típusú mikroprocesszoros számítógépekben, hanem Z80-asokban is használhatók, így a legtöbb hazai gép átalakítható lenne.

Végül, hogy ne csak bíráljak, feladatot adjak: a HCC vállalja bármely itthon nagy mennyiségben kapható mikorszámítógéphez *a teljes magyar betűkészletet helyes elrendezésben kijelezni képes megoldás kidolgozását.* Ez a javaslat már 9 hónapja megjelent az Ötletben, de eddig sem a gyártók, sem a forgalmazók, sem az oktatás irányítói nem fordultak hozzánk. Talán azért, mert ez csak a használókat érdekli? Nem tudom. Mi ajánlatunkat mindenesetre továbbra is tartjuk!

DR. SIMONYI ENDRE

Ki ad magyarázatot?

A Mikromagazin 1984/6. számában, a Primo-teszt cikkben hiányosak a karakterek. Ehhez hasonlót a Spectrumon is elő lehet idézni.

A Spectrum karakterkészlete a 15360-as címen kezdődik. Ezt a címet a CHARS rendszerváltozó őrzi, és POKE utasítással átírható. A 236Ø6-os címre 1-et írunk, a karakterek első bájtja hiányzik, mert az új kezdőcím 15361 lesz.

2-es beírása esetén már a következő karakter első sora az előző karakter alsó részén lesz, ennek viszont hiányzik a teteje.

Ennek magyarázata, hogy a kiolvasást a harmadik bájtnál kezdi a gép, innen 8 bájtot olvas. Az utolsó két bájt már a következő karakter első két bájtja lesz. Természetesen nem a kiíratás, hanem a karakterkészlet sorrendje szerint.

Más értéket is beírhatunk. Például POKE 236Ø6,8 a esetén a következő karakter fog megjelenni, mondjuk a helyett b.

„KOMPUTER" HÍREK

A Novotrade Rt. megküldte számomra a fenti című, havonta megjelenő tájékoztatójának eddigi számait.

Nem azt kaptam, amit vártam! Meglepve tapasztaltam, hogy az új és általuk forgalmazandó termékeknek nemcsak előnyeit, hanem hátrányait is ismertetik.

A C16-ról például leírták azt, hogy a „gyorsabb órajel ellenére a BASIC interpreter lassúbb a C64 BASIC-jénél; ezt a típusú gépet kimondottan iskolagépnek szánták, bár úgy tűnik, a 16 kbájtot a nebulók hamar kinövik".

(Ez utóbbinál arról azért nem tettek említést, amit a Mikromagazin 1985/6. számában a Mit vegyek? című cikkben leírtam, hogy a 16 kbájt RAM területből mennyi marad a felhasználónak — de ez már tényleg sok lenne!)

Ugyanezt teszik a játékprogram-fejlesztő munkájuk ismertetésénél is, ahol a nagy sikerek mellett a sikertelen fejlesztésekről is beszámolnak. Az egyik ezzel kapcsolatos kijelentést a hazai ipar és tudomány is megszívlelhetné: „Úgy gondoljuk, még mindig jobb egy projektet idejében leállítani, mint létrehozni egy eladhatatlan terméket".

A Közlemény rovat egyik hírét különösen fontosnak tartom. Eszerint az egyetemi Commodore Klubban egyesek a forgalmazó engedélye nélkül, pénzért árulnak olyan programokat, amelyeket a Novotrade Rt. forgalmaz. „A programok másolása önmagában nem szabálytalan, azonban a programokat árusítani, sokszorosítani csak hatósági engedély birtokában lehet." És ennek oka: „A számítógépes programok szellemi alkotások, azok jogosulatlan felhasználását jogszabály tiltja".

Két megjegyzésem van. 1. Az idézet mondatokhoz hasonló ugyan már szerepelt lapunk egyik vezércikkében, de ez a szabály a klubok számára annyira fontos, hogy célszerűnek tartom ebben a rovatban is közölni. 2. Az említett klub *nem* a HCC Commodore Szekciója.

Nagyon érdekesek az úti beszámolók, amelyek megítélésem szerint reális képet nyújtanak a felkeresett országokról, cégekről, és ami a leglényegesebb, az itthoni helyzetről. Egy, a hazai felhasználók figyelmébe ajánlható, többször, de nem elégszer leírt idézet: „Az USA-ban személyi számítógépnek az IBM PC-t és az Apple-t tartják. Jó játéknak számít a Commodore 64." (Ezt egyébként egy amerikai barátom úgy írta levelében 1984 végén, hogy „az itteniek várják, hogy az IBM vagy az Apple nyeri-e meg a háborút; addig nem vesznek számítógépet, hanem Commodore-t".)

Jó lenne, ha ezt az információs anyagot minél többen megismernék.

—i —e

Építsünk számítógépet!

Értesítjük az Olvasókat, hogy ha április 1-jéig elküldik megrendeléseiket dr. Simonyi Endrének, akkor 5200 forintért vállaljuk a három kártya elkészítését, az alkatrészek beültetését, a kártyák élesztését, vagy 3200 forintért a kártyák elkészítését. A szállítási határidő május 31.

A szükséges tápegység készítését is vállaljuk, áráról később adunk értesítést.

A kártyák árában az alkatrészek ára természetesen nincs benne, a tápegységében azonban igen.

HALÁSZ AMANDA
HCC „Építsünk számítógépet" Szekció titkára

TUDÁSPRÓBA

1. Mi a hő szerepe a „hőnyomtatókban”?

2. Mi a triád?

3. Mi a tetrád?

4. Igaz-e, hogy egy $x_1, \ldots, x_n, \ldots$ sorozat akkor és csak akkor tart egy véges határértékhez, ha létezik $a_1, \ldots, a_n, \ldots$ szigorúan monoton növekvő és $f_1, \ldots, f_n, \ldots$ szigorúan monoton csökkenő sorozat úgy, hogy $a_1 \leqq x_1 \leqq f_1, \ldots, a_n \leqq x_n \leqq f_n, \ldots$ azaz minden x_i az a_i és az f_i által határolt intervallumban van, és az $\frac{f_k - a_k}{f_{k+m} - a_{k+m}}$ $(k = 1,2,\ldots;\ m = 1,2,\ldots)$ alakú számok között tetszőleges nagyok is előfordulnak? (Vagy ami ugyanaz, reciprokaik között tetszőleges kicsi pozitív számok is előfordulnak.)

5. Hány vége van a számítástechnikai mágnesszalagos tárakban használat alatt levő mágnesszalagoknak?

6. Általánosabb fogalom-e a robot, mint az automata?

7. Vannak-e olyan információhordozók, amelyekre több, különféle módon olvasható információt rögzítenek?

8. Igaz-e, hogy egy 32 bites mikroprocesszornak azért van 64 lába, mert kétszer 32 egyenlő 64-gyel?

9. Van-e olyan számábrázolási rendszer, amelyben két jegy sorrendjének felcserélése ellentettjére változtatja az egyik jegy értékét?

10. Mi a „művonal”?

11. Minek a címe volt a „Gépek és programok”?

12. Azonos-e a gépi kódú programozás az assembly szintű nyelven való kódolással?

13. Mi az autokód?

14. Mi a MIMIC?

15. Mi köze a BASIC EXAPT-nak a BASIC-hez?

16. Van-e olyan IC, amelynek nemcsak oldalt, hanem elöl és hátul is vannak lábai?

17. Lehet-e két irányban végtelen sorozatnak rekurzív definíciója?

18. Melyik nyelvet tették közzé előbb, az ALGOL 60-at vagy a PL/1-et?

19. Van-e kereskedelmi forgalomban számológép elektrolumineszcens megjelenítővel?

20. Lehet-e fénycső (neoncső) elven működő kijelzőt szerkeszteni és számológépekben használni?

Összeállította
TAKÁCSY ILDIKÓ

Piac

A RAMOVILL – önmagáról

Cégünk 3 éve foglalkozik számítástechnikai eszközök kereskedelmével. A kezdeti években bizományos jellegű tevékenységet folytattunk, mely kizárólag gépkereskedelemre korlátozódott. Rövid idő alatt jutottunk el oda, hogy ilyen módon jóllehet jövedelmező tevékenységet végzünk, azonban az országba áramló eszközök minőségére gyakorlatilag nincs ráhatásunk. Sokszor vérző szívvel hoztunk forgalomba olyan eszközöket, melyekről tudtuk, hazai bázisuk nincs és valószínű nem is lesz, fejlesztése rendkívüli nehézségeket fog okozni, szoftverrel való ellátása intézményesen nem biztosítható.

Nos ilyen előzmények után határoztunk úgy, hogy olyan uniformizálási tevékenységre vállalkozunk, mellyel egy magas színvonalú gépet reális áron tömegesen hozunk forgalomba úgy, hogy egyidőben biztosítjuk a szerviz- és szoftverfeltételeket. Választásunk kiindulási alapja az IBM kompatibilitás. Fél év alatt mintegy fél tucat gépet vizsgáltunk meg. Ezek: CAF, EVERGO, SUPER PC, ERICSON, TATUNG, COPAM, IZOT, KITT 16, VT 16.

Alapfeltétel a kompatibilitás mellett a hálózati üzemre való alkalmazhatóság és a távlati bővíthetőség. A kiválasztás első féltideje lezajlott. Ez idő alatt az árat figyelmen kívül hagyva kizárólag a műszaki szempontok vezéreltek választásunkban. A továbbjutott gépek: CAF, ERICSON, IZOT.

A következő kiválasztási forduló a gazdaságosságot és a beszerzési gyorsaságot hivatott vizsgálni. Egy azonban biztos: egy 256/640 KB RAM-mal, 10, ill. 20 MB-os winchesterrel, 1 MB floppyval, színes monitorral, soros és párhuzamos interfésszel,hálózati hardverrel és szoftverrel rendelkező gépet 500 ezer forint alatti áron kívánunk forgalmazni úgy, hogy üzembehelyezését, garanciális és garanciaidőn túli szervizelését vagy átalánydíjas karbantartását biztosítjuk.

Számítógéposztályunk csak nem régen alakult, addig is míg tevékenységének gyümölcse beérik, arra orientálja a kötelékébe tartozó Computer boltot, hogy a jelenleg kialakult irreálisan magas árak mérséklésére törekedjék. A legfrissebb technológiával készült berendezések beszerzése továbbra sem képzelhető el (az ismert devizális okok miatt) csak „magánimport” útján. A pillanatnyi fonák helyzetben azonban nehéz úgy befolyásolni annak irányát, hogy a külkereskedelmi törvénynek is egyidőben maradéktalanul eleget tegyünk. Hiszen végül is a „kint dolgoztam, van x dollárom, mit hozzak haza?” kérdésre nincs jogunk válaszolni. És ez vezet végül oda, hogy ebben a számítástechnikai kultúrával még alig rendelkező országban a ZX 81-től a VAX-ig minden megtalálható.

HANTZMANN FERENC
ELEKTROIMPEX—RAMOVILL
Video-Hi-Fi Szaküzlet

ZX-Spectrumra

ANDROID NIM

Lapunk 1984/1. számában ismertettük a NIM játékprogram alapját képező nyerő stratégiát. Közöltünk egy megoldást a CBM 64-es gépre. Felkértük olvasóinkat, hogy próbálják meg átírni, egyszerűsíteni a programot itthon ismertebb, könnyebben elérhető gépre. Felhívásunkra érkezett egy 4. osztályos olvasónk programja.

A játékban a NIM játékszabályai érvényesek. Bármelyik sorból tetszőleges számú androidot lehet kilőni. Az győz, aki a legutolsó androidot lövi le. A piros robotok hajtják végre a lövést. Először a sor számát kell beütni, majd a kívánt számú androidot. A kérdéshez le kell nyomni egy gombot.

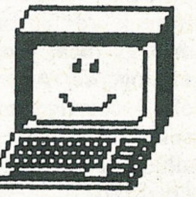
Ha a REM-eket nem írjuk be, 16 k-s gépen is fut a program.

RUSZNYÁK GÁBOR

```
    1 REM ####################
    2 REM #   ANDROID NIM    #
    3 REM #                  #
    4 REM #     PROGRAM:     #
    5 REM #                  #
    6 REM #  Rusznyak Gabor  #
    7 REM ####################
    8 REM
    9 REM
   10 RESTORE 9500: FOR I=0 TO 16
7: READ A: POKE USR "A"+I,A: NEX
T I: GO SUB 9000: REM [GRAFIKUS]
   20 BORDER 4: PAPER 0: INK 6: C
LS
   25 REM [PROGRAMSZERKEZET]
   30 PRINT AT 3,12;"[A] [A] [A] [A]
[A] [A] [A]"
   40 PRINT AT 4,12;"[B] [B] [B] [B]
[B] [B] [B]"
   50 PRINT AT 5,12;"[C] [C] [C] [C]
[C] [C] [C]"
   60 PRINT AT 6,12;"[D] [D] [D] [D]
[D] [D] [D]"
   70 INK 5: PRINT AT 9,15;"[A] [A]
[A] [A] [A]"
   80 PRINT AT 10,15;"[B] [B] [B] [B]
[B]"
   90 PRINT AT 11,15;"[C] [C] [C] [C]
[C]"
  100 PRINT AT 12,15;"[D] [D] [D] [D]
[D]"
  110 INK 4: PRINT AT 15,18;"[A] [A]
[A] [A]"
  120 PRINT AT 16,18;"[B] [B] [B]"
  130 PRINT AT 17,18;"[C] [C] [C]"
  140 PRINT AT 18,18;"[D] [D] [D]"
  150 INK 2: PRINT AT 3,0;"[A]";AT
4,0;"[B]";AT 5,0;"[C]";AT 6,0;"[D]
"
  160 PRINT AT 9,0;"[A]";AT 10,0;"
[B]";AT 11,0;"[C]";AT 12,0;"[D]"
  170 PRINT AT 15,0;"[A]";AT 16,0;
"[B]";AT 17,0;"[C]";AT 18,0;"[D]"
  180 LET A$="": DIM A(3): LET A(
1)=7: LET A(2)=5: LET A(3)=3
  190 PRINT AT 20,18; INK 7;"RG s
oftware ©"
  999 REM [ANDROIDOK MOZGATASA]
1000 PRINT AT 20,1;"                ": LET
A$="": GO TO 1130
1010 BEEP RND*.1,(RND*10)-30: IN
K 6: LET K=3: LET L=9: IF A(1) <>
0 THEN FOR I=8-A(1) TO 7: LET R=
INT (RND*4): GO SUB 1080: NEXT I
1020 BEEP RND*.1,(RND*10)-30: IN
K 5: LET K=9: LET L=12: IF A(2) <
>0 THEN FOR I=6-A(2) TO 5: LET R
=INT (RND*4): GO SUB 1080: NEXT
I
1030 BEEP RND*.1,(RND*10)-30: IN
K 4: LET K=15: LET L=15: IF A(3)
<>1 THEN FOR I=4-A(3) TO 3: LET
R=INT (RND*4): GO SUB 1080: NEXT
 I
1040 RETURN
1050 FOR J=1 TO 10: GO SUB 1010:
 PRINT INK 2;AT 6*X-3,0;"[A]": GO
 SUB 1010: PRINT INK 2;AT 6*X-3,
0;"[E]": NEXT J
1060 PRINT INK 2;AT 6*X-3,0;"[A]"
1070 RETURN
1080 IF R=0 THEN PRINT AT K,L+3*
I;"[A]"
1090 IF R=1 THEN PRINT AT K,L+3*
I;"[B]"
1100 IF R=2 THEN PRINT AT K,L+3*
I;"[C]"
1110 IF R=3 THEN PRINT AT K,L+3*
I;"[D]"
1120 LET A$=A$+INKEY$: RETURN
1125 REM [JATEKOS KERDEZES]
1130 INK 7: PRINT FLASH 1;AT 1,9
;"TE KOVETKEZEL!"
1140 LET A$=A$+" ": LET A$=A$(1)
: IF A$=" " THEN LET A$="": GO S
UB 1010: GO TO 1140

1150 IF CODE A$<49 OR CODE A$>51
 THEN LET A$="": GO TO 1140
1160 LET X=CODE A$-48: PRINT AT
20,1;X
1165 LET A$=""
1170 IF A$="" THEN GO SUB 1010:
GO TO 1170
1180 IF CODE A$<49 OR CODE A$>55
 THEN GO TO 1170
1190 LET Y=CODE A$-48: PRINT AT
20,5;Y
1191 IF Y>A(X) THEN GO SUB 1050:
 GO TO 1000
1195 LET A$="": GO SUB 1200: IF
A(1)<>0 OR A(2)<>0 OR A(3)<>0 TH
EN GO TO 1395
1196 GO TO 4000
1200 PRINT INK 2;AT 6*X-3,0;"[E]"
1210 GO SUB 1010
1220 IF Y>A(X) THEN PRINT AT 20,
1;"                ": GO TO 1050
1230 PRINT AT 1,9;"
 "
1240 FOR J=1 TO 5: GO SUB 1010:
PRINT INK 2;AT 6*X-3,0;"[A]": GO
SUB 1010: PRINT INK 2;AT 6*X-3,0
;"[A]": NEXT J
1250 PRINT INK 2;AT 6*X-3,0;"[E]"
1260 IF A(1)<>0 THEN LET K=3: LE
T L=9: INK 6: FOR I=8-A(1) TO 7:
 LET R=1*(X>1)+2*(X=1): GO SUB 1
080: NEXT I
1270 IF A(2)<>0 THEN LET K=9: LE
T L=12: INK 5: FOR I=6-A(2) TO 5
: LET R=1*(X=3)+2*(X=2): GO SUB
1080: NEXT I
1280 IF A(3)<>0 THEN LET K=15: L
ET L=15: INK 4: FOR I=4-A(3) TO
3: LET R=2*(X=3): GO SUB 1080: N
EXT I
1285 REM [LOVES]
1290 LET A(X)=A(X)-Y
1300 PRINT INK 2;AT 6*X-2,1;"[F]"
1310 FOR I=1 TO 5: FOR J=1 TO 10
: BEEP .05,I*J: NEXT J: NEXT I
1320 FOR I=1 TO 3: FOR J=-20 TO
-14: BEEP .01,J: NEXT J: NEXT I
1330 FOR K=1 TO Y: PRINT INK 2;A
T 6*X-2,1;"[F]": PLOT 24,190-48*I
NT X: DRAW INK 7;240+24*K-24*X-2
4*(A(X)+Y),0
1340 LET X1=6*X-3: LET Y1=33+3*K
-3*A(X)-3*Y-3*X
1350 PRINT INK 3;AT X1,Y1;"[A]";A
T X1+1,Y1;"[A]";AT X1+2,Y1;"[A]";A
T X1+3,Y1;"[A]"
1360 FOR I=1 TO 10: FOR J=-20 TO
 -10 STEP 3: BEEP .01,J: NEXT J:
 NEXT I
1370 PRINT AT X1,Y1;"   ";AT X1+1
,Y1;"   ";AT X1+2,Y1;"   ";AT X1+3
,Y1;"   "
1380 PLOT INVERSE 1;24,190-48*IN
T X: DRAW INVERSE 1;240+24*K-24*
X-24*(A(X)+Y),0
1390 PRINT INK 2;AT 6*X-2,1;"[F] "
: NEXT K
1391 PRINT INK 2;AT 6*X-3,0;"[A]"
: PRINT AT 20,1;"                ": RETURN
1395 GO TO 1460
1399 REM [SZAMITOGEP NYERO LOGIKA]
1400 LET A1=INT (A(1)/4): LET A2
=INT ((A(1)-A1*4)/2): LET A3=A(1
)-(A1*4+A2*2)
1410 LET A4=INT (A(2)/4): LET A5
=INT ((A(2)-A4*4)/2): LET A6=A(2
)-(A4*4+A5*2)
1420 LET A7=INT (A(3)/4): LET A8
=INT ((A(3)-A7*4)/2): LET A9=A(3
)-(A7*4+A8*2)
```

```
1430 LET S1=A1+A4+A7: LET S2=A2+
A5+A8: LET S3=A3+A6+A9
1440 IF S1=0 AND S2=0 AND S3=0 T
HEN LET FLAG=2: RETURN
1450 IF (S1=2 OR S1=0) AND (S2=2
 OR S2=0) AND (S3=2 OR S3=0) THE
N LET FLAG=1

1455 RETURN
1460 PRINT INK 7; FLASH 1;AT 1,9
;"EN KOVETKEZEM!"
1465 IF RND>.75 THEN GO TO 1510:
 REM A SZAMITOGEP TEVEDESE
1470 FOR I=1 TO 3: LET FLAG=0: L
ET C1=A(I): IF C1<>0 THEN FOR K=
1 TO C1: LET FLAG=0: LET A(I)=A(
I)-1: GO SUB 1400
1480 IF FLAG=1 THEN LET X=I: LET
 Y=C1-A(I): LET A(I)=C1: PRINT A
T 20,1;X;AT 20,5;Y: GO SUB 1200:
 GO TO 1000
1490 IF FLAG=2 THEN GO TO 3000
1500 NEXT K: LET A(I)=C1: NEXT I
1510 LET X=INT (RND*3)+1: LET Y=
INT (RND*A(X))+1: IF Y<=A(X) THE
N PRINT AT 20,1;X;AT 20,5;Y: GO
SUB 1200: GO TO 1000
1520 GO TO 1510
2999 REM A SZAMITOGEP NYERT
3000 LET X=I: LET Y=C1-A(I): LET

 A(I)=C1: PRINT AT 20,1;X;AT 20,
5;Y: GO SUB 1200: INK 2
3010 PRINT INK 2;AT 8,7;"  ";
AT 10,4;"          
";AT 11,4;"          
 ";AT 12,4;"          
 ";AT 13,4;"  "

3020 RESTORE 4030: FOR I=1 TO 15
: READ A,B: BEEP A,B
3030 IF INKEY$<>"" THEN RUN
3040 NEXT I: GO TO 3020
3999 REM A JATEKOS NYERT
4000 PRINT INK 6;AT 9,13;"  
 ";AT 10,9;"        ";AT
11,9;"        ";AT 12,9;"
  "
4010 RESTORE 4020: FOR I=1 TO 11
: READ A,B: BEEP A,B: IF INKEY$<
>"" THEN RUN
4015 NEXT I: GO TO 4010
4016 RUN

4020 DATA 1,0,1,0,.3,0,1,0,.7,4,
.3,2,.7,2,.3,0,.7,0,.3,-1,1.5,0
4030 DATA .5,-1,.5,-1,.5,0,.5,2,
.5,2,.5,0,.5,-1,.5,-3,.5,-5,.5,-
5,.5,-3,.5,-1,.7,-1,.3,-3,.5,-3
8999 REM HASZNALATI UTASITAS
9000 BORDER 2: PAPER 6: INK 0: C
LS
9010 PRINT AT 1,6;"A N D R O I D
  N I M";AT 1,6; OVER 1;"________
_____________"
9020 PRINT AT 3,0;"  A játékban
a NIM játékszabá-     lyai érvény
esek. Bármelyik       sorbol tets
zöleges számú and-    roidot lehe
t kilőni."
9025 PRINT "  Az győz, aki a leg
utolso and-    roidot lövi le. A
piros robo-    tok hajtják végre
a lövést."
9030 PRINT "  Először a sor szám
át kell be-    ütni, utána a kivá

nt számú       androidot."
9040 PRINT '"  A játékhoz sok si
kert kiván:"
9050 PRINT '"  Rusznyák Gábor"
9060 PRINT '"  A kezdéshez nyomj
 egy gombot!"
9070 PRINT AT 20,1;"RG software
©"
9080 PAUSE 0: RETURN
9500 REM UDG
9510 DATA 015,031,059,127,127,06
0,031,015: REM
9520 DATA 240,248,220,254,254,06
0,248,240: REM
9530 DATA 003,127,255,223,223,22
3,223,223: REM
9540 DATA 192,254,255,251,251,25
1,251,251: REM
9550 DATA 127,063,031,031,025,04
9,097,097: REM
9560 DATA 254,252,248,248,152,14
0,134,134: REM

9570 DATA 097,097,243,000,000,00
0,000,000: REM
9580 DATA 134,134,207,000,000,00
0,000,000: REM
9590 DATA 015,031,027,063,031,00
3,031,015: REM
9600 DATA 240,248,188,220,220,18
8,248,240: REM
9610 DATA 015,031,061,059,059,06
1,031,015: REM
9620 DATA 240,248,216,252,248,19
2,248,240: REM
9630 DATA 000,015,031,063,123,12
7,063,028: REM
9640 DATA 000,240,248,252,222,25
4,252,056: REM
9650 DATA 192,254,255,249,248,24
8,248,248: REM
9660 DATA 000,015,060,232,192,00
0,000,000: REM
9670 DATA 016,016,056,004,060,06
8,060,000: REM

9680 DATA 016,016,056,068,124,06
4,060,000: REM
9690 DATA 040,040,056,068,068,06
8,056,000: REM
9700 DATA 040,000,068,068,068,06
8,056,000: REM
9710 DATA 016,016,068,068,068,06
8,056,000: REM
9998 REM A PROGRAM KIMENTESE G
O TO 9999
9999 CLEAR : SAVE "NIM" LINE 1:
VERIFY ""
```

PROGRAMOZÓK MESÉLIK



Az Eureka című számítógépes játék programozása volt eddig a legnagyobb kaland a magyar játékprogramozók számára. Öngyilkos szerződés kötötte őket: a határidő lejárta után minden nap késés 666 font munkadíjlevonással fenyegetett. Előrebocsáthatjuk, hogy végül nem volt levonás; a játék határidőre elkészült. Hanem ami addig történt . . .

A Sinclair Spectrumra írt változat a Novotrade szoftverstúdiójában készült, a Commodore 64 változatot a Caesar Stúdióban fejlesztették, és aludni egyik helyen sem igen tudtak. A programozók az utolsó néhány nap minden percét a stúdióban töltötték, kis lábaskában hordták az ennivalót, és aki végképp nem bírta az iramot, az egy fotelben vagy összetolt asztalok tetején dőlt le egy-egy félórára, hogy azután újból a programba vesse magát.

Az emberi kimerültség legvégső határán szédelgő programozók a szó szoros értelmében drága időből egy ízben hosszú perceket vesztegettek el azzal, hogy összevitatkoztak: vacsoráztak-e aznap, vagy sem.

A játék végül piacra került, előkelő helyezéseket ért el a slágerlistákon, és a megrendelő, a Domark cég hatalmas összegű, huszonötezer angol fontos díjat tűzött ki annak, aki legelőször végigjátssza a hatalmas méretű játékot. A szoftverhez egy kis könyvecske is tartozott, amelyet úgyszintén alaposan végig kellett tanulmányoznia annak, aki a végső megfejtésre volt kíváncsi. A megfejtés egy londoni telefonszám volt: aki először felhívta ezt a számot, az nyerte meg a példátlanul nagy összeget.

Sok-sok hónap telt el a játék megjelenése után, és a kincsvadászok még mindig a kalandjátékot nyüstölték — eredménytelenül. A leleményesség persze nem mondott csődöt: számtalan telefon érkezett a Novotrade-hez: a hívók, akikhez eljutott egy-egy kalózmásolat, jelentékeny százalékot kínáltak a díjból annak a programozónak, aki hajlandó elárulni a telefonszámot.

Végül is egy tizenöt éves angol diákgyerek nyerte meg a huszonötezer fontot. És mit tesz a véletlen — vagy a szervezés: éppen akkor vehette át, amikor Londonban az őszi számítógép-világkiállítást tartották.

(R. T.)

KEDVES OLVASÓINK!

Ezúttal a hozzám érkezett leveleket ismertetem. Olyan sokat kaptam, hogy elnézést kérek szerkesztőtársaimtól, de az ő válaszaikat csak a következő számban fogom közölni.

Szabóné Kis Ildikó, Sopron,

Pázmány Péter u. 5. 9400

Valószínű, hogy nem emlékszik már az egy évvel ezelőtt stoppoló két lányra, akik Önnek köszönhették, hogy aznap este részt vehettek a soproni valétálás eseményein. De én mást is köszönhetek Önnek: a lelkesítő, élvezetes beszélgetést a számítógépről. Ennek hatására kezdtünk el férjemmel foglalkozni, ismerkedni a géppel.

Most eljutottunk addig, hogy közösen összeállítottunk egy tucat programot ZX-Spectrum számítógépre. A programok az általános iskolai 3. osztályos magyar nyelvtanhoz kapcsolódnak. Segítségükkel a tanuló gyakorolhatja a főnevek felismerését szövegből, szavak közül, megadott módokon csoportosíthatja azokat, emlékezetpróbát tehet, nyelvtani totót tölthet ki stb. A gép a feladatok megoldását ellenőrzi, értékeli. A program kezelése a tanuló számára egyszerű, érthető.

Ezeket a programokat bemutattam több iskolából összegyűlt kollégák előtt. Ott vetődött fel, hogy jó lenne terjeszteni, sokszorosítani azokat, de tanácsot nem tudtak adni, hogy hova forduljak ez ügyben. Ezért Önt terhelem kérdésemmel: milyen intézményeket ajánl, amelyek előzetes vizsgálatok után segítségemre lehetnének ezeknek a programoknak a forgalmazásában, elterjesztésében?

Kiválóan emlékezem találkozásunkra és beszélgetésünkre is, amelynek ilyen eredményére igazán nem számítottam. Némi büszkeséggel közlöm levelét, remélve, hogy a nyilvános válaszadással is egyetért. Azt hiszem nincs is egy éve, hogy találkoztunk, és ez a rövid idő elegendő volt arra, hogy férji közreműködéssel ugyan, de egy humán érdeklődésű pedagógus „összeállítson egy tucat programot", amellyel harmadik osztályos gyerekeket tanít az általános iskolában. Körülbelül ez a célja az NJSZT által meghirdetett, a kormányprogramban is megfogalmazott társadalmi akciónak: megérteni és megtanulni, hogy mire való a számítógép, és alkalmazni a napi munkában. Köszönöm Önnek, hogy bemutatta, így kell ezt csinálni a gyakorlatban.

Sajnos a Sinclair számítógépek mechanikai felépítésüknél fogva nem alkalmasak komoly iskolai igénybevételre, így az oktatásügy nem támogatja ezeket. A programokat szerkesztőségünk szívesen fogadja, jó oktatóprogramokat örömmel közlünk.

Vadas Tamás, Budapest,

Regöly u. 6. 1182

Tanuló vagyok, 1985 februárjában kezdtem el foglalkozni a számítástechnikával. Május 27-én ott voltam a TV-BASIC vizsgán, ami ugye alig négyhónapos tudással és gyakorlat nélkül elég merész vállalkozásnak tűnt. Az Ötletben megjelent helyes válaszok alapján ellenőriztem magam, és kiderült, hogy alig maradtam el az elfogadható alsó pontszámtól.

A Magazin egyik számában olvastam, hogy lesz még egy TV-BASIC vizsga előreláthatólag 1986 márciusában. A kérdéseim: Hasonló lesz-e az előző vizsgához? Tehát újra lehet-e választani négy géptípus között, és megint csak az „Alap BASIC"-ből lesznek a kérdések? Mennyi lesz a részvételi díj?

Olvastam a vizsgaanyagot tartalmazó füzetekről is. Ha lehetne, szíveskedjenek részemre elküldeni mind a 4 füzetből 1-1 példányt utánvéttel!

Közérdekű kérdésére a felelet akkor jelenik meg, amikor a TV-BASIC-et már ismét javában sugározza a televízió. A tanfolyam január első hetében kezdődött, a sorozat egy-egy adását a nézők egyszer délután, egyszer pedig a késő esti órákban láthatják, így a diákok is és a felnőttek is nyugodtan végignézhetik a műsort. A tanfolyam április végén fejeződik be, ezért a vizsgát május elejére tervezzük. A vizsgázás módja azonos lesz az előzővel, de a vizsgakérdések mások lesznek. Ismét négy géptípus között lehet választani. A részvételi díjban még nem állapodtunk meg, én azt szeretném, ha a diákok lényegesen alacsonyabb vizsgadíjat fizetnének, mint a felnőttek. A korábbi vizsgaanyagot valamennyi μklubnak megküldtük, ott lehet tanulmányozni.

Styt Sándor Gábor, Budakeszi,

Vörös Hadsereg u. 88—89. 2092

Az a kérdésem, lehet-e itthon Sinclair ULA-t kapni, és ha igen, hol?

Szeretnék egy gépet építeni, de nincs meg az Építsünk számítógépet rovat I., II., III. része. Ha lehet, küldjék el nekem.

Mellékelem a Kapstone, Spectrum botkormány-csatlakozó kapcsolási rajzát. Remélem, hasznát veszik.

Legjobb tudomásom szerint nem lehet kapni. Az Építsünk számítógépet hiányzó részeit másolatban elküldtem, a kapcsolási rajzot pedig Simonyi Endrének továbbítottam.

Mesterházi András, Budapest,

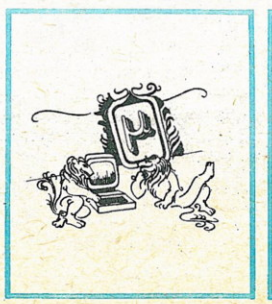
Attila u. 63. 1013

Lapjuk rendszeres olvasója vagyok, megjelenése óta. De valamit hiányolok a rovatok közül. Tudom, hogy van egy hírekkel, érdekességekkel foglalkozó rovatuk, de szerintem ez a néhány hasáb nem elég a szakmában történő sokféle aktualitás közlésére. Kérésemmel és javaslatommal sok számítógép-rajongó (programozó és játékgyűjtő) kérését tolmácsolom. Szükség lenne egy olyan 3 oldal terjedelmű rovatra, amely sok angol és nyugatnémet számítástechnikai folyóirathoz hasonlóan, olyan közérdekű cikkeket közölne, mint például: külföldi vagy magyar játékszoftverek, újdonságok bemutatása, képes leírása; az év játéka – a hónap játéka; hogyan dolgoznak a világ számítógépgyártó cégei és nagy játékgyártói; a tíz legkedveltebb játék közlése közönségszavazat alapján; számítógép- és számítógép-tartozék újdonságok képes ismertetői; a középoldalon színes poszter; sokkal több BASIC játékprogram, zenei és grafikai program; a személyi számítógépek külföldi és hazai árainak rendszeres közlése; hirdetések; programpályázatok és rejtvények (jutalmak).

Kaptam egy másik hasonló levelet is, más javaslatokkal. Az elmúlt két és fél év is bizonyítja, hogy amit lehetett, azt megvalósítottuk. El kell mondanom, hogy nagyon szeretjük a játékprogramokat, de az erre szánt ter-








jedelmet nem fogjuk növelni. Ezzel szemben egyre több oktatási célú vagy a napi életben alkalmazható programot szeretnénk kapni és közölni. A havi megjelenéssel valóban lehetőség nyílik arra, hogy a legfrissebb ajánlatokról és árakról tudósítsunk; erre már felkértük azokat a cégeket, amelyek a legtöbb számítógépet forgalmazzák.

Szendrei Péter, Kazincbarcika,

Szabó L. u. 8. 3700

Nagy számítógép-kedvelő vagyok, már 2 éve foglalkozom vele. Szeretnék venni magamnak egy Commodore 64-est, de több barátom azt tanácsolta, hogy inkább külföldön nézzek utána, mivel itthon nagyon drága. Azt szeretném konkrétan megtudni, hogy Ausztriában mennyibe kerül ez a masina, továbbá, hogy milyen vámrendelkezések vannak a behozatalára, és hogy tényleg jobban megéri-e ott vásárolni, mint idehaza.

Ami igaz, az igaz, külföldön a számítógép sokkal olcsóbb, mint nálunk, de remélem, nem sokáig. Számítógépet abból a devizából lehet külföldön vásárolni, amelyet a kiutazó hivatalosan vásárol, vagy hivatalos útjain igazoltan megtakarít. A számítógép behozatalához engedélyt kell kérni a Magyar Nemzeti Banktól, a belföldi forgalmi érték után 15% vámot kell fizetni. Azt javasolom, hogy utazása előtt kérdezze meg a Vámhivataltól az éppen aktuális előírásokat. Ami az árat illeti — változó. Attól függ, hogy melyik országba utazik. Itt Európában az NSZK-ban lehet a legolcsóbban és Franciaországban a legdrágábban gépet vásárolni. Az árak napról napra változnak. Ha nagy szerencséje van, akkor még egy kiárusítást is kifoghat, és a géphez akár az eredeti ár feléért is hozzájuthat.

Hodján István, Budapest,

Vezér u. 55. 1188

Elektronikai műszerész vagyok, és a sors úgy hozta, hogy munkám során közelebbi kapcsolatba fogok kerülni egy komolyabb számítógéppel, de úgy érzem, hogy programozói tudásomat ez a gép (M80) meghaladja, ezért szeretném komolyabban megismerni a programozás tudományát. Szeretném megkérni Önöket, hogy legyenek szívesek küldeni címet vagy címeket, ahol programozást lehet tanulni és vizsga is tehető.

A SZÁMALK tanfolyamokat ajánlom. Javasolom, hogy keresse fel a Vállalat oktatási osztályát (Budapest XI., Szakasits Árpád út 68.), ahol megkaphatja a tanfolyamok jegyzékét, és tájékozódhat a beiratkozás feltételeiről.

Krassalkovics Stefán, Miskolc

III., Rácz Á. u. 19. 3532

Többször is olvastam lapjukban, hogy valaki hozott külföldről valamilyen számítógépet, és nehézséget okoz neki, hogy a gépkönyv vagy az ismertetés idegen nyelvű. Én vállalnám a német nyelvű könyvek lefordítását.

Íme az ajánlat, amelyet örömmel közlünk. Javaslatát, hogy szívesen írna a lapba, köszönjük. Várjuk bárkinek az írását, rajzokat, fényképeket, tudósításokat, programokat. Rendelést általában senkinek sem adunk.

Remetei Ernő, Baja,

Engels u. 29/A 6500

Az olvasó írja részben – az Ötletben is – többen várnak valami programot a VC20-ra. Bár vizsgapapíromon alig száradt meg a tinta, néhány programocskát sikerült írnom e kedves kis masinára. A program talán nem haszontalan annak, aki a sajnos tényleg csak a géppel való ismerkedést segítő füzetkénél tovább kíván lépni. A többit is szívesen elkészítem, ha arra érdemesnek ítéli őket.

Nagyon megköszönném, ha tanácsot adna, milyen irányú programokkal célszerű hobbiként másokat is megfertőzni. Tanácsai abban is segítségemre lennének, hogy vállalatomnál egy kis BASIC alapfokú ismertetőt szervezhessek. A játékos oktatóprogramot úgy írogatom, hogy ha megjön a májusban a Quellénél befizetett C64-es, csak a POKE-olással azon is fusson. A „játékos"-t azért írom, mert az asztali és zsebszámológéppel nagyon sok kollégám már elválaszthatatlanul együtt él, de a számítógépet kínainak tartják. Így a bevezetőben mint számológépet mutatom be, majd a változókat bevonva, egy-egy összetettebb számítást, majd egy intelligens kérdésekre adott egyszerű válasszal igazolva, mily nagyszerű fárasztó manuális számolás helyett ugyanazt a jó és kisebb bizonytalanságú eredményt megkapni. Csak ezt követi a BASIC annak, aki nemcsak használni akarja a kész programot, de érdeklődési körének megfelelően, munkáját könnyítendő azt saját részére el is tudja készíteni. Nem vagyok pedagógus, így a jó logikai felépítettségű oktatóanyagokat eléggé sok INS/DEL billentyűzéssel formázom.

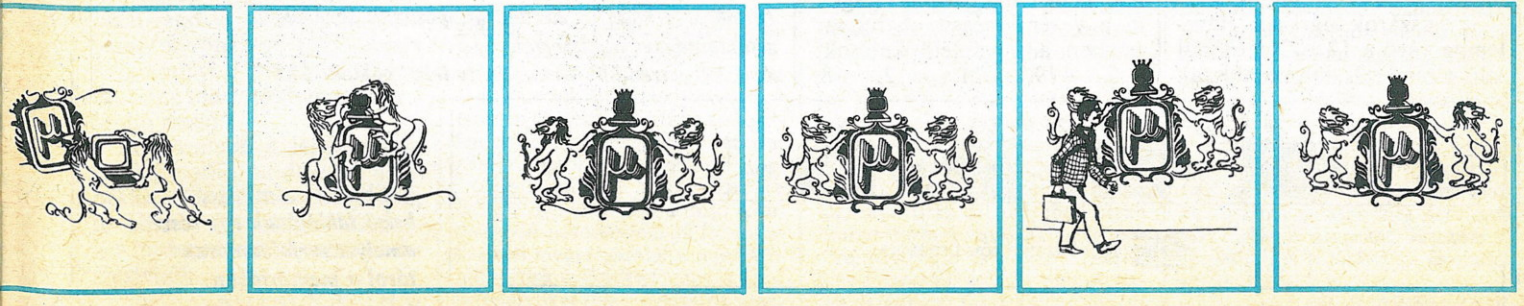
Először a kérdésére válaszolnék, bár levele második részében már válaszolt rá. Én nem hiszek abban, hogy a számítástechnika oktatására vannak csalhatatlan és egyértelmű módszerek. Találkoztam olyan diáktársasággal, amely szinte sohasem játszott számítógéppel, a játékprogram csak addig érdekelte őket, amíg az elkészült. Arra voltak büszkék, hogy milyen játékstratégiát találtak, illetve azt milyen programozási trükkökkel és grafikával tudták megvalósítani. Én azt hiszem, hogy értelmes embereknek elég azt megmutatni, hogy a számítógép hasznos „jószág", nem kell különleges tehetség a gép használatának megtanulásához, illetve mennyire könnyítheti meg a gép az ember munkáját. Emlékszem olyan bérelszámoló kollégára, aki kézzel-lábbal tiltakozott a számítástechnika tanulása ellen, amíg egyszer ki nem próbált egy bérelszámoló programot. Ettől kezdve játszva végezte a munkáját anélkül, hogy a számítástechnikát szerette volna — mondta ő —, mégis abszolút híve lett a számítástechnika alkalmazásának. Azt azonban nem hiszem, hogy hajlandó lenne a BASIC-et megtanulni. Ezzel a kis példával azt szeretném bemutatni, ha számítástechnikát oktat, ne markoljon sokat. Lesznek a munkatársai között, akiknek bőven elég lesz, ha egy-két statikai méretezőprogramot megmutat és a használatát megtanítja, vagy bemutatja, hogyan lehet adatokat keresni a vállalati adatbankban. Nem hiszem, hogy a társadalom informatizálása eredményeként mindenkinek programot kell írnia. Szörnyű lenne.

A C64 ASS. lista ügyben azt tanácsolom, forduljon a Novotrade-hez, segíteni fognak.

●

Nagyon sok olvasónk kér levélben választ a levelére. Ha ezek a válaszok késnek, annak számos oka lehet, például nem tudunk a kérdésre válaszolni, esetleg nem tudjuk elküldeni a µM régebbi számait. Előfordult már, hogy a levél elveszett, esetleg nem érkezett meg; olvasóink megértését kérjük, leveleiket pedig változatlan örömmel várjuk.

KOVÁCS GYŐZŐ



BITEK ÉS FIGURÁK

LÉPÉSGENERÁLÁS I.

Cikksorozatunk első részében bemutattuk a sakkhadállás gépi ábrázolásának különféle módszereit: a táblaindexes és a bittérképes ábrázolásmódot. A számítógép ezekkel könnyen meg tudja különböztetni a különféle figuratípusokat. Így érzékeli a pozícióit. Egy konkrét állásból azonban hogyan generálja a lépéseket?

Amíg eljutunk egy pozícióban lehetséges legális lépések sorozatához, két algoritmust használunk. Az egyik generálja a lépéseket és létrehozza a „félig legális" lépések listáját, de még nem veszi figyelembe például azt, ha egy figuránk kötésben áll, és ezzel ellépve, a királyunk sakkban maradna, vagy ha magával a királlyal lépnénk sakkba, vagy hogy olyan mezőre, ahol egy saját figuránk áll, nem léphetünk. Ezután kerül sor különféle módszerekkel a valóban legális lépések kiszűrésére a félig legális lépések listájából.

A lépésgenerálás különféle módszereinek taglalása előtt meg kell ismerkednünk a belső tábla kifejezéssel. A belső tábla számunkra nem látható, a számítógép ezen értékeli az állásokat. A lépésgenerálás különféle módszereinél a tábla nagysága nem egyforma. Például a táblaindexes eljárásoknál 10×12 mezőn tároljuk a pozíciót, a bittérképesnél pedig 8×15 bájton. A táblaindexes ábrázolás három legfontosabb lépésgenerálási módja: 1. a halmazmező módszer; 2. a táblavezérlésű módszer; 3. a lépéslista folyamatos felújítása.

Mielőtt ezekkel közelebbről megismerkednénk, nézzük meg, hogy a táblaindexes ábrázolási módban a számítógép belső táblája miért is áll 10×12 mezőből (*1/a* és *1/b ábra*).

A programnak be kell tudnia határolni a sakktábla sorainak és oszlopainak elejét és végét. Ezt a legegyszerűbben úgy tehetjük meg, ha a 8×8-as tábla köré egy extremális elemekből álló „gyűrűt" helyezünk, mely megakadályozza az egyes figurák kilépését a tábláról.

A huszárok ugrásait figyelembe véve a 12×12 mezőből álló elrendezés logikusabbnak látszana, de mivel a tábla sorait a gép sorfolytonosan tárolja a memóriában *(2. ábra)*, ezért gondolatban egy hengerpalástot képzelhetjük el, amelynek szélei találkoznak. Így kevesebb memória igénybevételével ugyanarra az eredményre jutunk. Most vizsgáljuk részleteiben.

1. A *halmazmező módszer* azon alapszik, hogy egy adott figura számára a lehetséges lépések a mezők címe közötti relációk alapján könnyen meghatározhatók. Rendeljük az egyes mezőkhöz az *1/a* ábra szerint a memóriacímeket. Ha a királyt a 36-os mezőre képzeljük, akkor félig legális lépései a következők: 25, 35, 45, 46, 47, 37, 27, 26. Általánosabb formában, ha a király a k-adik mezőn áll, akkor a „félig legális" lépések a következők:

k−11, k−01, k+09, k+10, k+11, k+01, k−09, k−10

Láthatjuk, hogy mennyire egyszerű a félig legális lépések listájának felírása. Ezek után meg kell vizsgálni az egyes lépésekhez és az érkezési mezőkhöz fűződő tényezőket. Amennyiben azon saját figuránk áll, akkor kivesszük a lépéslistából, majd megvizsgáljuk, hogy lépésünk következtében nem került-e sakkba. Ha igen, ezt is kiveszszük a listából; így végül megkapjuk a legális lépéseket. Hasonlóképpen nem nehéz kitalálni a kulcsokat a többi figurára. A hosszú lépésű figurák esetén (vezér, bástya, futó) ha az adott kulcs szerinti első mező üres, akkor rá is léphet és át is mehet rajta. Így tovább vizsgálhatjuk ugyanazzal a kulccsal a figura többi lépését, amíg saját vagy ellenséges bábura nem akadunk. Az azon túli lépéseket természetesen automatikusan kiiktathatjuk, mivel ezek a figurák nem ugorhatnak. Ha a 36-os mezőn álló király helyébe vezért képzelünk el, akkor a 11-es kulccsal nemcsak a 47-es mezőt vizsgálhatjuk meg, hanem a 47+11=58-ast is.

Az extremális elemeknek, melyekkel körülvettük a táblát, most látjuk a jelentőségét. Tudjuk, hogy ezzel a kulccsal már nem lehet tovább vizsgálni a vezér lépéseit. A huszár esetben a megfelelő kulcsok: −8, −19, −21, −12, +8,

	110	111	112	113	114	115	116	117	118	119
	100	101	102	103	104	105	106	107	108	109
8	90	91	92	93	94	95	96	97	98	99
7	80	81	82	83	84	85	86	87	88	89
6	70	71	72	73	74	75	76	77	78	79
5	60	61	62	63	64	65	66	67	68	69
4	50	51	52	53	54	55	56	57	58	59
3	40	41	42	43	44	45	46	47	48	49
2	30	31	32	33	34	35	36	37	38	39
1	20	21	22	23	24	25	26	27	28	29
	10	11	12	13	14	15	16	17	18	19
	00	01	02	03	04	05	06	07	08	09
		A	B	C	D	E	F	G	H	

A sakktábla sötét mezője

1/a ábra. A számítógép belső táblájának elhelyezkedése a táblaindexes ábrázolási módszernél

	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT
	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT
8	EXT	$84	$82	$83	$85	$86	$83	$82	$81	EXT
7	EXT	$81	$81	$81	$81	$81	$81	$81	$81	EXT
6	EXT	$00	$00	$00	$00	$00	$00	$00	$00	EXT
5	EXT	$00	$00	$00	$00	$00	$00	$00	$00	EXT
4	EXT	$00	$00	$00	$00	$00	$00	$00	$00	EXT
3	EXT	$00	$00	$00	$00	$00	$00	$00	$00	EXT
2	EXT	$01	$01	$01	$01	$01	$01	$01	$01	EXT
1	EXT	$04	$02	$03	$05	$06	$03	$02	$04	EXT
	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT
	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT	EXT
		A	B	C	D	E	F	G	H	

A sakktábla sötét mezője

1/b ábra. A hadállás kezdeti állapotának ábrázolása a táblaindexes módszerrel. Az EXT extremális elem, amely lehet például $FF

19	20	21	22	23	24	25	26	27	28	29	30	31
EXT	EXT	$04	$02	$03	$05	$06	$03	$02	$04	EXT	EXT	$01

A sakktábla sötét mezője

2. ábra. A számítógép belső táblájának egy része, amelyet sorfolytonosan tárol a memóriában

+19, +21, +12. Így a 43-as mezőn álló huszár lehetséges lépései:

43−8=35 megfelel az E2 mezőnek

43−19=24 megfelel a D1 mezőnek

43−21=22 megfelel a B1 mezőnek

43−12=31 megfelel az A2 mezőnek

43+8=51 megfelel az A4 mezőnek,

és így tovább.

Ha a huszár a tábla szélén van, akkor az extremális elemek jelzik, hogy lelépett a tábláról.

A bástya esetében az XY címet (pl.: 56-os mezőn x=5, y=6) a lehetséges irányoknak megfelelően módosítjuk: x,y+n; x,y−n; x+n,y; x−n,y ahol n=1,2,3 . . .

A futó lehetséges irányai: Z+n,Q+n, Z+n,Q−n; Z−n,Q+n; Z−n,Q−n.

A lépéslista előállításánál a fentieken kívül a különleges lépéseket is figyelembe kell venni, így például a gyalog en passant lépését és a király és bástya rövid, illetve hosszú sáncolási lehetőségét.

2. A *táblavezérlésű lépésgenerálás* módszerében gyorsabb az előzőnél. A lehetséges lépések sorozatát tároló táblát használ, hogy feleslegessé tegye azok újbóli kiszámítását. Így az algoritmus gyorsabbá válik, de a táblázatok letárolása memóriaigényesebb. A szemléletesség kedvéért bemutatunk egy példát, melyen a bástya a 43-as mezőn áll. A hozzá tartozó táblázat:

33 23 00 00 00 00 00

53 63 73 83 93 103 00

42 41 00 00 00 00 00

44 45 46 47 48 150 00

A program látja a táblázat alapján a megvizsgálandó mezőket, és észreveszi, ha ott figura áll. Ha a mező üres, akkor veszi a táblázat adott sorának következő elemét, és ugyanabban az irányban tovább vizsgál, amíg csak a táblázatban zérus elemre nem talál. Ez jelzi a program számára, hogy ebben az irányban elfogytak a mezők, és a táblázat következő sorára ugrik. A táblázat végét egy extremális elem jelzi (esetünkben 150), amely megmondja a program számára, hogy nincs több megvizsgálandó mező.

Legközelebb a lépésgenerálás harmadik módszerét, a lépéslista folyamatos felújítását, továbbá a bittérképes ábrázolás lépésgenerálási módjait ismertetjük.

KOVÁCS ATTILA

Tankönyv vagy ifjúság elleni bűntett?

A könyv, amely 1985 őszén jelent meg, középiskolai tankönyv, és olyanokat talál benne a tanuló, mint amiket a következő válogatással szemléltetünk. Megjegyzéseinket zárójelbe tettük. *(A recenzens terjedelmes felsorolásából — helyszűke miatt — csak részleteket tudunk közölni. — A szerk.)*

19. oldal: „A programokban szereplő matematikai formulák tulajdonképpen *értékadó utasítások*, de ez itt nem fontos." (Nagyon súlyos kijelentések. Eredetük és indoklásuk hiányzik.)

35. oldal: „Megegyezünk abban, hogy a megfelelő kis- és nagybetűk között sehol nem teszünk különbséget. Ebben nincs semmi elfogadhatatlan: eddig is, ha azt láttuk leírva, hogy Arany János vagy azt, hogy ARANY JÁNOS, ugyanarra a személyre gondoltunk." (Na és mi lesz az „arany János"-sal?)

64. oldal: „Mivel *minden folytonos függvény közelíthető polinomokkal* (9. ábra), így Babbage gépe elvileg alkalmas lett volna tetszőleges folytonos függvény táblázatának elkészítésére." A 9. ábra felirata pedig ugyanezen az oldalon: „Karl Weierstrass (1815—1897) tétele szerint minden folytonos görbe tetszőleges pontossággal közelíthető polinommal." (Ha függvény, milyenek pl. a végtelen függvénysorral definiált folytonos függvények, pl. a folytonos és sehol sem differenciálható függvények *adott pontosságú* polinomapproximációi Babbage gépével? Ha pedig görbe, a Peano görbe polinom approximációja se volna kutya Babbage gépétől, de még a maiaktól sem. De maradjunk egyszerűbb példáknál! Milyen pontosságú polinom közelítése van pl. az abs (x) vagy pl. a sin (x) függvénynek a valós számok halmazán? Semmiképpen sem tetszőleges.)

86. oldal: „Az AND-del összekapcsolt feltételek együttesen akkor teljesülnek, ha mindegyik külön-külön is teljesül." (Ugyanez igaz a VAGY-gyal vagy másképp összekapcsolt feltételekre is.)

277. oldal: „Vigyázzunk az INPUT W$ elején és végén levő zűrökre!" (Egy jellemző példa a szakmai pongyolaságot jópofasággal elütő nagyszámú megnyilatkozás közül.)

329. oldal: „Amikor *adatról* beszélünk, többnyire olyan feljegyzésre gondolunk, amelynek *egyik része ismert, a másikat keressük:* tudjuk a nevet, keressük a telefonszámot . . ." (Eredeti. Ha megtaláljuk a másik részt, megszűnik annak adatvolta. És ha egyik részét sem ismerjük a feljegyzésnek? Egy minden részében előttünk ismeretlen idegen telefonfüzet eszerint nem tartalmaz adatokat?)

362. oldal: „A feladatokat Te BASIC nyelvben oldod meg. A valóságban hasonló jellegű feladatokat kell megválaszolni a nagy számítógépek működtetésekor is, de a programokat 0—1 sorozatok alakjában írjuk meg! Az utolsó fél mondat nem pontos, de nem baj." (Igen, dörgölheti a szemét a tanuló, hogy jól lát-e. Ha ő az iskolai órán valamiben nem pontos, akkor az baj. Rossz jegyet kap érte. A tankönyvíró pedig honoráriumot.)

369. oldal: „A számítógépek életbenmaradásában döntő fontosságú volt a tranzisztorok megjelenése." (Vaskos tévedés. Jól mutatja a szerző járatlanságát szakmai kérdésekben.)

Aztán van még a könyvben több mottó, magyarázattal, néhány szó a mítoszokról, aztán „egy kis matematika" és „még egy kis matematika", aztán (nagyon félresikerült) játék a „játsszunk a ciklusképpzéssel", „egy kis filozofálgatás, egy kis történelem", „gyökkeresés oroszlánfogással", „egy kis hangtan", „egy kis csillagászat", egy „őrült autós", és labirintusépítés és töméntelen nyelvhelyességi hiba.

Nincs a könyvben munkára nevelés, tiszta didaktikai szerkezet, rendesen megalapozott fogalomkialakítás, gyakorlati szemlélet, szerzői vagy lektori lelkiismeretesség. De ezek fel sem tűnnek, annyira dominál a rengeteg hiba. Ki kell mondani, hogy e tankönyv minden szempontból tele van hiányosságokkal és hibákkal. Úgy mint egy régi híres könyv, Lietzmann Wo steckt der Fehler című műve. Csakhogy e könyv címe nem az, hogy „Hol a hiba?", hanem az, hogy „Számítástechnika a speciális matematikai osztályok részére". Írta Simonovits Miklós egyetemi tanár, a matematikai tudományok doktora, lektorálta Csirmaz László a matematikai tudományok kandidátusa, és Szeredi Péter tudományos munkatárs. A tankönyvet az Országos Pedagógiai Intézet részéről véleményezte dr. Urbán János főelőadó, a matematikai tudományok kandidátusa. A tankönyv a művelődési miniszter rendeletére készült, az Országos Pedagógiai Intézet közreműködésével. A 202/1976. (M.K.6.) OM—KM sz. együttes utasítás 22. pontja alapján a tankönyv kéziratát a Tankönyvkiadó igazgatója jóváhagyta, és ugyanezen utasítás 23. pontja alapján a Művelődési Minisztérium 21 935/84. sz. alatt bevezetését és használatát a 1985/86. tanévtől a gimnázium III—IV. osztálya számára engedélyezte.

POGÁNY CSABA

Hetedhét Commodore

A gép hetedhét országra szóló sikere, avagy a hetes beosztású füzetek indították a szerzőket a címadásra, nem tudom, de tetszik. Végre találkozott Pál Zsuzsanna programozó és Révbíró Tamás szöveg- (vagy mese-?) író személyében egy olyan szerzőpáros, amely eltalálta a hangot, a metódust, a számítógép-tanítási didaxist. Együttműködésük mégsem volna látványos, ha nem akadnak Dévényi Erika személyében olyan illusztrátorra és műszaki szerkesztőre, aki bájos és vidám rajzaival még csinossá is tette az amúgy szerény kiállítású füzeteket.

Kiadó: a Novotrade RT. A megcélzott olvasókör: felsős gyerekek. A cél: ismerkedés a géppel és a BASIC nyelvvel. Felosztása gépenként (ez eddig C64, C16) 3-3 füzet, 7-7 napra szánt fejezettel. Felhasználási módja: egyéni tanulás, olvasás; 3×7 órás tanfolyam. Megjelenési formája: az Easy Script szövegszerkesztő programja és egy vékony filctoll, mint ékezetkiíró segítségével előállított, fotókész kézirat sokszorosítása.

Nem tudom ugyan, hogy valóban felsős gyerekek olvassák-e a füzeteket, de az is jó, hogy kézbe kapják. Igazából pedig ott a legfőbb haszon, hogy az oktatók, szakkörvezetők olyan könyv alapján ismertetik meg a gyerekeket a számítástechnikával, amelytől idegen a szakemberes nagyképűség.

VOTISKY

Kovács Mihály: Számítógép a fizikatanításban

(Budapest, 1985. Országos Pedagógiai Intézet, 180 oldal. Ára: 47,— Ft.)

A könyv két, egymástól független tanulmányt ölel fel *Gépi kódú programozás a Z80-as mikroprocesszorral működő számítógéphez* és *Számítógép a fizikaórán* címmel.

Az első rész az alapfogalmak és a kezelési ismeretek rövid összefoglalása után a Z80-as mikroprocesszor utasításkészletét mutatja be. A programpéldák tárgyának és nehézségi fokának alkalmas megválasztásával a szerző fokozatosan készíti fel az olvasót a gépi kódú szubrutinok BASIC programokba való beépítésére, sőt megírására is. A téma kimerítő elemzésére azonban nem vállalkozik, de a továbbgondoláshoz vezérfonalként bőséges hazai irodalmat sorol fel.

A második részben áttekintést kapunk az iskolaszámítógép felhasználási lehetőségeiről a fizika és a technika oktatásában, majd a gép hátlapján található 20 pólusú csatlakozósáv jelvezetékeivel: a hanggenerátor és a két digitális beviteli kapu csatlakoztatási lehetőségeivel ismerkedhetünk meg. Az ún. felhasználói kapuhoz csatlakoztatott eszközökkel vezérlést és mérést is végezhetünk. Az eszközök vezérlésére és az idő mérésére gépi kódú program szolgál.

A számítógép iskolai felhasználásában kialakult tendenciák között újdonságnak számít az a megközelítés, amelyet a szerző bemutat: nem a régi szemléletet kell az új eszközökre vinnünk, hanem meg kell újítanunk szemléletmódunkat. Elterjedt balhiedelem, hogy az új eszközök működésének ismeretére semmi szükség, maradjanak azok csak fekete dobozok. Tévedés! A fekete dobozoknak egyre világosabbá kell válniuk!

A könyvet ajánljuk mindazoknak a középiskolai fizikatanároknak, akik bátran vállalják az újat, és azoknak is, akik még nem tudták rászánni magukat. Haszonnal forgathatják a technika- és matematikatanárok is.

DR. PAPP SÁNDOR

Programok C16-ra

A Novotrade a C16-os gépek megjelenésére oktatóprogram-fejlesztést kezdett. Most közreadjuk a megjelent, illetve a közeljövőben megjelenő programokat, hogy ezzel is segítsük az iskolák munkáját.

I. ÁLTALÁNOS ISKOLAI PROGRAMOK

	AJÁNLOTT ÉLETKOR	A MEGJELENÉS IDEJE
MATEMATIKA		
Oszthatóság	8—12 év	1986. január
Halmazok	8—12 év	1986. január
Axonometria (testek ábrázolása, transzformációk)	8—18 év	1985. dec.
Kombinatorika	9—14 év	1986. január
Valószínűségszámítás	10—14 év	1986. február
Statisztika	12—14 év	1986. március
FIZIKA		
Elektromos áram	12—13 év	1986. február
Fénytan	14 év	1986. január
Atomfizika	12—14 év	1986. január
KÉMIA		
Kémiai reakciók	12—13 év	1986. február
BIOLÓGIA		
Idegrendszer	12—13 év	1986. február
FÖLDRAJZ		
Topográfia	10—14 év	1986. február
A Szovjetunió gazdasági földrajza	14 év	1986. február
OROSZ		
Cirill betűk	8—9 év	1986. február
Szótár programcsomag (szókincsfejlesztés, ellenőrzés)	8—14 év	1986. március
ANGOL		
Nyelvtan, szókincsgyakorlás programcsomag	9—14 év	1986. II. félév
MAGYAR		
Olvasás	6—7 év	1986. február
Hangrendilleszkedés	8—10 év	1986. június
Időmértékes verselés	10—14 év	1986. május
Mondatelemzés	10—14 év	1986. június
TÖRTÉNELEM		
Adatbázis	10—14 év	1986. február —május
ÉNEK—ZENE		
Zenei alapismeretek	6—14 év	1986. április

II. OKTATÓJÁTÉKOK

	A megjelenés ideje
DIENES-JÁTÉKOK (logikai készséget fejlesztő programcsomag)	
Bűvös kör	1986. január
Abrakadabra	1986. január
Logikai utak	1986. március
Építsünk számokat!	1986. március
Térbeli amőba — térlátás, szituációfelismerés	1986. január
Tányértorony — kombinatív készség	1986. január
Bigyulabolt — vállalkozás, szimuláció, gazdasági ismeretek	1986. január
Sakk	1986. január

III. OKTATÓPROGRAMOK MINDENKINEK

Gyorsolvasás — 4 hétre tervezett, önállóan elvégezhető tanfolyam programcsomagja	1986. február
Gitároktatás — zenei alapfogalmak, akkordismeretek	1986. február
Planetárium — csillagászati programcsomag	1986. június

IV. ISKOLAI SEGÉDPROGRAMOK

Tanuló-nyilvántartás	1986. január
Tornaverseny-értékelés	1986. január

V. PROGRAMOZÓI SEGÉDPROGRAMOK

Bitlecke + dokumentáció Gépi kódú alapismeretek (2-es számrendszer, bit, bájt, PEEK, POKE, SYS) A dokumentáció részletes magyar nyelvű rendszerváltozó és I/O leírást tartalmaz	1986. január
Assembler oktatóprogram-csomag felhasználói segédprogramokkal kiegészítve	1986. március
Kiegészítő dokumentációk ROM-lista, hardverleírás	1986. március

Mikrogép a körúton

Az elmúlt egy-két év alatt gombamód szaporodnak a számítógépeket árusító üzletek a belvárosban, s ma már hozzászoktunk a kirakatokban elhelyezett, árcédulával ellátott számítógépek látványához. E boltok közös jellemzője azonban, hogy a fejlett országokból behozott mikroszámítógépeket kínálják.

Ezért jelent újdonságot a Múzeum körúti Keravill bolt kirakata, melyben a Primo számítógép látható. Szocialista országban ez az első bolt, ahol hazai gyártású számítógép vásárolható kiskereskedelmi forgalomban készpénzért, azonnal.

A Primo mindhárom változata (32, 48 és 64 k) megvásárolható 15—24 ezer forintért. A forgalom fokozatosan fut fel, jelenleg havonta mintegy tucatnyi gépet értékesítenek. Perifériákra is van igény, így Primóhoz illesztett nyomtatót is forgalmaznak, továbbá szoftvert is árusítanak, és nemcsak játékprogramokat, hanem van például öntanító BASIC tanfolyamuk, sőt FORTH fordítóprogramjuk is.

Winchester-mentés

Az utóbbi időben egyre több nehézséget jelentett, hogy a hazai mikrogépekhez alkalmazott Winchester-tárak mentése nem volt megoldva. 1985 során mind több mágnesszalag-kazettás tároló érkezett az országba. Ezek a videokazettáknál egy kicsit kisebb kazettán 10—60 Mbájt tárolási lehetőséget nyújtanak. Ilyen egység alkalmazása lehetővé teszi a nagyszámítógépeken rutinszerűen alkalmazott mentéseket, vagy bizonyos lemezállomány kiírását. Alkalmazása tehát többek között azt is biztosíthatja, hogy ugyanazon a Winchester-táron több olyan nagy állományt is elhelyezzenek, amelyekre nincs mindig szükség és egyidőben sohasem kellenek.

A mágnesszalag-kazetta másik gyakorlati előnye a Winchester-táron fixen elhelyezett állomány „hordozhatóvá" tétele. Egy Winchester-táron levő állományra például szükség lehet a város másik végében működő gyáregységben, vagy szüksége lehet rá a fejlesztőnek, aki otthoni gépén szeretné folytatni este a munkát. Ilyenkor mágnesszalag-kazettán lehet átvinni az állományt.

Előnye még, hogy a kiírás és visszatöltés rendkívül gyors: 60 Mbájt 8-10 perc alatt.

A mágnesszalag-kazettás tároló viszont nem olcsó. Egy 30 Mbájtos például gépbe építve (mérete megegyezik az 5,25-ös hajlékonylemezes tárolóéval, így annak rekeszébe, ahelyett szokták beépíteni) 110 ezer, külön dobozba helyezve pedig 150 ezer forint. Ez utóbbi verzió a gyakorlatban jobban alkalmazható, különösen több olyan autonom üzemeltetett gép esetén, melyek Winchester-tárolóval rendelkeznek. A mágnesszalag-kazettás egységnek ugyanis nem kell állandóan a konfigurációba illesztve üzemelni, hanem szükség esetén csatlakoztathatják arra a gépre, amelynél éppen szükség van rá. A hazai, IBM PC-vel kompatibilis gépekhez ma már hétköznapi dolognak számít a mágnesszalag-kazettás egység szállítása. Így a Proper—16, VT—16, MXT, MAT gépekhez a felhasználók már a gyakorlatban is alkalmazzák ezeket.

Hordozható mikrogép

A Mobi—X és a PTA 4000 után most új zsebméretű, hordozható mikroszámítógép jelent meg, a Norax 64.

Ez az adatgyűjtésre orientált gép jól alkalmazható gyors beviteli igény (például leltár) esetén, vagy folyamatosan, akár hetekig gyűjtött adatok gépre vitelére, például árammérő óra leolvasására, továbbá a mezőgazdaságban, erdőgazdaságban, állattenyésztésben.

A gép, amely magában foglalja az akkumulátoros tápforrást is, egy tenyérben elfér. Mikroprocesszora Z80 alapú. 4 k csak olvasható és 4 k operatív memória áll a vezérlőprogram, továbbá 64 k a gyűjtött adatok rendelkezésére. 2 × 16 pozíciós kijelzője és 16 nyomógombos billentyűzete van.

RS 232 illesztőjén keresztül az adattár feltölthető, kiolvasható, illetve tetszés szerinti kommunikációra alkalmazható, például másik mikroszámítógéppel is összekötthető.

Az akkumulátor 48 óránként igényel utántöltést. Ha az akkumulátor töltöttsége egy kritikus szint alá csökken, akkor a rögzítési funkciót automatikusan leállítja, de az adattár az információkat további 48 órán át megőrzi. Az utántöltést követően adatmentés nélkül tovább használható.

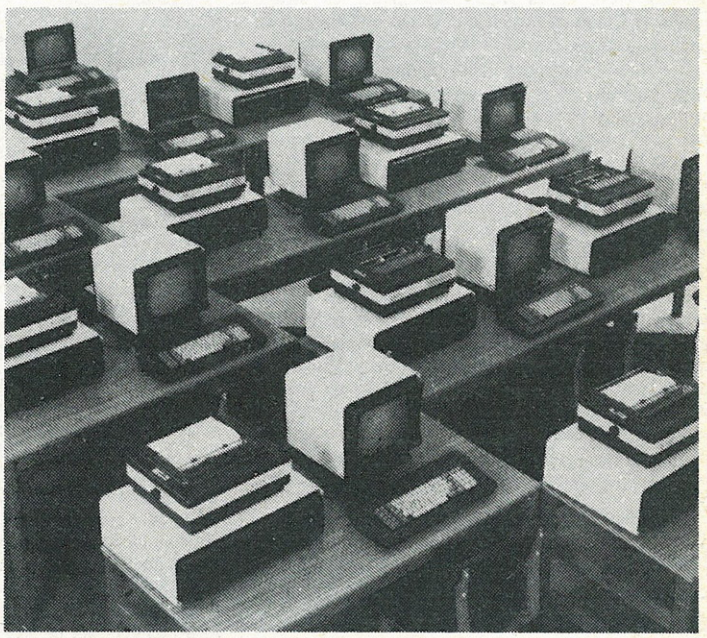
A Data Manager Kisszövetkezet által forgalmazott Norax 64 műszaki jellemzőit tekintve megközelíti az angol Mikrofinét, amelyet Bulgária forgalmaz a szocialista országokban.

Lokális hálózatok

Egyre több lokális hálózatba köthető mikroszámítógépet kínálnak a hazai piacon. Az úttörő a Műszertechnika Gmk-nak a még 1984-ben bemutatott Multi Center nevű többterminálos gépe volt, amely még 8 bites mikroprocesszort alkalmaz. A 16 bites gépek elterjedésével ugrásszerűen megnövekedett a kínálat, ami már nem csupán típus-, hanem elvi szintű választékot is jelent. Az SZKI-Scil például kínálja többterminálos rendszerét, mely — mint a neve is mutatja — olcsó terminálok telepítését jelenti egy Proper—16 mikrogép irányítása alá. A másik, kissé drágább változat az osztott intelligencia lehetőségeit biztosító hálózata, melynek minden eleme egy-egy önálló, saját perifériákkal is kiépíthető s azt más gépekről is elérhető, általa gyártott mikroszámítógép.

Megjelentek a hazai piacon nyugati importgépekből álló lokális hálózatok is. Ilyen például az 5 G Kisszövetkezet által forgalmazott, amerikai gyártmányú, az IBM PC-vel kompatibilis Rair gépekből álló hálózat. Ez utóbbiak közé számíthatjuk a Távol-Keletről behozott, IBM PC-vel kompatibilis gépeket s az azokból kialakított hálózatokat is. A lokális hálózatok működését biztosító hazai szoftvereket 1985/5. számunkban, a 46. oldalon tekintettük át.

Lokális hálózattá összekötött Proper—16 mikroszámítógépek



Sikerszoftver

Az 1985. év egyik hazai csúcsterméke a dACCESS III jelzésű adatbázis-kezelő program. Ez az első, szocialista országban kidolgozott program, mely a jelenleg legkorszerűbb, relációs adatmondellt valósítja meg. A dACCESS III bármely, az IBM PC-vel kompatibilis gépen futtatható, s a szinte nemzetközi szabványként elfogadott dBASE III adatszerkezetekkel dolgozik. A felhasználó számára azért is előnyös, mert az általa már korábban is alkalmazott, megszokott programnyelv keretei között használhatja. Jól hasznosítható tulajdonsága e szoftvertermékeknek, hogy párhuzamosan lehet vele használni 10 nyitott adatbázist és 10 indexállományt.

A dACCESS III kedvező fogadtatását jól jellemzi, hogy 1985. márciusi megjelenése után hat hónappal az NSZK-ban már több mint 200, hazánkban pedig csaknem 50 esetben értékesítették, 98 ezer forintos áron! A sikert látva, a forgalmazást végző SOFTINVEST már megbízást adott a fejlesztőknek a dACCESS III lokális hálózatban történő alkalmazhatóságának kialakítására. Ez a verzió várhatóan augusztusra készül el.

Kínai szupermini

Fél éve vizsgálta be az állami átvételi bizottság az első 32 bites kínai számítógépet Pekingben. A 2780 jelzéssel gyártott gépnek a neve is sugallja a VAX 11/780-nal való kompatibilitását. A gépet az Észak-Kínai Számítástechnikai Kutatóintézet fejlesztette ki. Jellemzői közül a sebessége említésre méltó: egymillió művelet másodpercenként.

VÁLASZOK A TUDÁSPRÓBA KÉRDÉSEIRE

1. Jelenleg kétféle hőnyomtató terjedt el nagyobb számban. Az egyikben speciális papírral dolgoznak, amely hő hatására visszafordíthatatlanul megváltoztatja a színét. Az írás apró hőkezelő fejek segítségével történik, amelyek kis hőtehetetlenségű pontokat melegítenek alkalmas (rövid) ideig.

A másik hőnyomtatónál a festék hasonlóan kerül a papírra mint a hagyományos festékszalagos nyomtatóknál, csak a festékszalag itt kétrétegű. Az írófej (tulajdonképp olvasztófej) a festékhordozó szalaggal érintkezik, a hordozószalag a festékréteggel, (rá van tapasztva) a festékréteg pedig a papírral. Az írófej meleg pontjai átmelegítik a hordozószalag velük érintkező pontjait, ezek pedig megolvasztják a velük érintkező részen a festékréteg pontjait. A megolvadt festékpontok tapadnak a papírra. A hideg pontokon levő festék a hordozószalaghoz tapadt állapotban marad. Az írófej itt nem mechanikus impulzussal présel rá a festékszalagból egyes pontokat a papírra, hanem mindenütt rápréseli a festéket a papírra, de az csak egyes pontokon (a felmelegítetteken) ragad hozzá ahhoz.

2. A *triád* hármast, három elemből összetettet, három részből állót jelent. Számítástechnikában „három számjegyből álló" jelentésben használják. E jegyeket azonban eleve kettes számrendszerbelieknek tartani nem helyes. Vannak ugyanis diadikus triádok mellett decimális triádok, hexadecimális triádok stb. is.

3. A *tetrád* négyest, négy elemből összetettet, négy részből állót jelent. A számítástechnikában négy (szám)jegyű alakzatot értünk rajta. A *monád* egy-, a *diád* kettő-, a *triád* három-, a *tetrád* négy-, a *pentád* öt-, a *hexád* hat-, a *heptád* hét-, az *oktád* nyolc-, a *nonád* kilenc-, a *dekád* tízelemű együttest jelent. Ha ezeknek számjegyek az építőelemei, mindig meg kell mondani, hogy azok milyen számrendszerbeliek. A triádikus triádnak pl. mindhárom jegye hármas alapú számrendszerbeli.

4. Igaz. Nyilvánvalóan igaz marad a tétel, ha a $\leq$ helyett tetszőleges helyeken $<$ jelet írunk. Ha pedig a törtek reciprokával dolgozunk (ez mindig megtehető), azt kell mondanunk, hogy a konvergencia szükséges és elégséges feltétele az, hogy a reciprok értékek között a zérushoz tetszőleges közeli számok is előfordulnak. (A nulla ezek torlódási pontja.)

Végül pedig úgy is fogalmazható a tétel, hogy a

$$\frac{f_r - a_r}{f_s - a_s} \qquad (r = 1,2,\ldots; s = 1,2,\ldots)$$

alakú számok között tetszőleges nagyok előfordulása, a 0-hoz tetszőleges közeliek előfordulásával, mindkettő pedig együttesen is és külön-külön is a konvergenciával egyenértékű.

Ez a konvergenciakritérium azon a tételen alapszik, hogy csupa pozitív elemből álló szigorúan monoton csökkenő, (tehát korlátos) sorozat akkor és csak akkor tart nullához, ha az elemei hányadosait képezve, az így kapott számok között a 0-hoz tetszőleges közeli számok is előfordulnak. (Ez egyenértékű a tetszőleges nagy számok előfordulásával.) Ha csak nagyobb indexű elemet osztunk kisebb indexűvel, ebben a részhalmazban a 0-hoz tetszőleges közeli értékek léte szükséges és elégséges feltétele a konvergenciának. Ha viszont csak kisebb indexűeket osztunk nagyobb indexűvel, az így kapott részhalmazban tetszőleges nagy számok előfordulása lesz a konvergencia szükséges és elégséges feltétele.

5. A legtöbbnek kettő. Régebben voltak olyanok amelyeknek egy végük sem, és olyanok is amelyeknek 10-nél is több végük volt. Az Ural—1 és az Ural—2 gép mágnesszalagja végtelenítve volt, az elejét és a végét össze kellett ragasztani. Így meg lehetett oldani — csak egyirányú mozgatás lehetősége mellett is — a tetszőleges sokszori olvasást és felírást is. Természetesen a maitól eltérő mozgatómechanizmust használtak.

A maga idejében hardver- és szoftverszempontból csúcsnak és egyben mérföldkőnek számító dán GIER gépnek olyan mágnesszalagkezelő mechanizmusa is volt, amelyben a szalagállomány sok kis tekercsen helyezkedett el, egy nagy korongra erősítve. A szalag végeinek száma a kis tekercsek számának kétszerese volt. Így az elérést gyorsították meg azáltal, hogy a korong — legfeljebb 180 fokos fordulattal — az éppen használni kívánt kis tekercset az író-olvasó mechanizmushoz vitte, az pedig a szokásos módon vette kezelésbe. Ha egy szalagot ilymódon pl. 10 kis tekercsre vágtak szét, az elérési idő az eredetinek, egy additív állandótól eltekintve, egytizedére csökkent.

6. Nem.

7. Igen. Vannak amelyekre szemmel olvasható szöveg mellé mágneses információ kerül. Vannak amelyeken a nyomtatással felírt szöveg mellett lyukasztással rögzített jelek találhatók. Lyukasztott információ — régebben — még mágnesszalagon is előfordult, amit fotódiódás leolvasással érzékelt a gép.

8. Nem. (Van 32 bites mikroprocesszor amelynek 64-nél kevesebb lába van.) Az információ be- és kiáramoltatás — aminek a módja nagymértékben befolyásolja a lábak számát — elvileg független a belső szóhosszúságtól.

9. Van. A római számábrázolási rendszerben a CX értéke 110 (+100+10), az XC értéke pedig 90 (−10+100).

10. Különösen az első időkben a számítástechnikában is használt tárolóegység. Lényege az információ mechanikai illetve akusztikai jel formájában való cirkuláltatása. Elve olyan, mintha tőlünk nagyobb távolságra levő mikrofonba mondanánk néhány szót, a mikrofon által vett hang pedig elektromos vezetéken keresztül elektromos formában hozzánk továbbítódna és egy erősítőt és hangszórót működtetne, amely újra leadná szavainkat a mikrofonba.

11. Az egyik hazai számítástechnikai ősfolyóiratnak. Ez a szakkörökben térítés nélkül terjesztett kiadvány a Központi Statisztikai Hivatal Ügyvitelgépesítési Főosztálya Elektronikus Számológép Részlegének számítástechnikai közleményeit tartalmazta. (Közvetlen előde volt az Információ—Elektronika és a Számítástechnika folyóiratnak.) Tudománytörténeti szempontból is érdekes füzetei ma már muzeális ritkaságok.

12. Nem. A gépi kódú programozás a lehető „legalacsonyabb" programnyelvi szinten zajló tevékenység. Az assembly szintű munka eredményét az assembler (lényegében fordító) program fordítja gépi kódra.

13. A program csak a gép számára közvetlenül feldolgozható jelek formájában képes a gép munkáját irányítani. A gépi tevékenységek közvetlen vezérlését ezeknek a tevékenységeknek a jeleivel, ún. kódjaival végeztetjük. A kódolás általánosan valaminek valamilyen jelölésrendszer jeleivel, szabályainak betartásával való leírása, speciálisan főleg egy programnak a gépi tevékenységet közvetlenül vezérelni képes jelek rendezett összessége formájában való előállítása.

Az első időkben a programozók a gépi nyelvnél kissé magasabb szintű, esetleg saját maguk által kialakított jelrendszert használó nyelveken leírt programjaikból maguk alakították ki a gépi utasítások kódjaiból álló, közvetlenül felhasználható programot. Ehhez a fordítási munkához nemritkán alacsonyabb képesítésű szakemberek segítségét is igénybe vették. Ez a tevékenység lényegében gépi utasítások felírásának, majd pedig egy másik munkamenetben, azok kódjainak felsorakoztatásából állt. (A címeket is a második menetben szokták konkretizálni.) Amikor az utasításnak megfelelő gépi jelet (kódot) írták be, akkor „kódoltak".

A kódolás munkáját a géppel elvégeztető rendszereket (nyelveket és hozzájuk tartozó fordítóprogramokat) hívták autokódnak, mert általuk a gép maga kódolt, önkódolóvá, autokódolóvá lett.

Helytelen azonban csak a gépi kódot tekinteni kódnak. Kód lényegében minden jel. Kódolás minden jelölés is. Az angol nyelvű szakirodalomban máig is él a kód szó ilyen, általános értelmű használata, amikor pl. egy eljárás kódján annak (nem szükségképpen gépi kódú) programját értik.

Hazánkban a leghasználtabb és méltán híres autokód az Elliott 803—B típusú gép A—103 jelű autokódja volt. Hazai kutatók is készítettek azonban különböző autokódokat. Többek között az Ural—2 gépre az A—103 és az UTRA jelűt. Az előbbi nagymértékben segítette az Elliott 803—B és az Ural—2 közötti programáramlást.

14. Egy 1965-ben közzétett, eredetileg IBM 7090 számológépcsalád tagjain, majd pedig sok más gépen, pl. a CDC 3000 és 6000 sorozatú gépeken használható, széles körben elterjedt, azóta továbbfejlesztett, de máig sem elavult, folytonos folyamatokkal leírható rendszerek viselkedésének modellezésére szolgáló rendszer és programozási nyelv.

15. Semmi. A BASIC főleg személyi számológépek a BASIC EXAPT pedig számvezérlésű szerszámgépek programozásának nyelve. A BASIC szó az első esetében a Beginner's All-purpose Symbolic Instruction Code kezdőbetűiből származik, a másik esetében azonban a szó eredeti jelentésében szerepel. A BASIC EXAPT így *ALAP EXAPT*-ot jelent, vannak ugyanis más EXAPT-nyelvváltozatok is.

16. Van. A lábak (a csatlakozók) számának állandó emelkedése miatt az IC-tokok nagyon megnyúltak. Ha az eddig kihasználatlan első és hátsó oldalra is lábak kerülnek, a tok bizonyos szempontból előnytelen arányait közelíteni lehet a négyzetéhez. Különösen nagy lábszámnál a „körüllábas" tokozást részesítik előnyben az „oldallábas"-sal szemben. Nem kell azonban félni, hogy túl közel vagyunk a lábak számának felső határához. Van még hely a tok alatt és felett is, és a „kétlábsoros„ és „többlábsoros" megoldások elterjedése sem lenne meglepetés.

17. Lehet, pl. 2 hatványainak két irányban végtelen sorozatára vonatkozik a következő definíció.

ha $n=0$, $a_n=1$,

ha $n\neq 0$,

$a_n = 2^{sg(n)}((1+sg(sg(n)-1))a_{n-1}+(1-sg(sg(n)+1))a_{n+1})$.

Ne feledkezzünk meg az explicit alakról sem! Nyilvánvaló, hogy

$a_n = 2^n$.

18. Az ALGOL 60-at.

19. Van. Pl. a *Sharp* cég *Grid Compass* nevű terméke. Ebben félvezető anyag világít elektromos potenciálkülönbség hatására.

20. Lehet. Az ilyen kijelzőben két egymáshoz közeli függőleges sík egyikében függőleges huzalok, a másikában vízszintes huzalok vannak. Ha egy függőleges huzal és egy vízszintes között (megfelelő) potenciálkülönbséget hoznak létre, akkor köztük kisülés jön létre, és a kialakult plazma világító pontként érzékelhető. Több mint egy éve piacon van az IBM egy majdnem háromnegyed millió képpontos, fénycső elven működő kijelzője.

INFORMÁCIÓTECHNIKAI VÁLLALAT

Központ:
Budapest V., Bécsi u. 8.
Levélcím:
Budapest, Postafiók 314.
1369
Telefon: 184-899
Telex: 22-4381, 22-6841

TELJES MŰSZAKI KISZOLGÁLÁSSAL
látjuk el a

ROBOTRON 1715 típusú PROFESSZIONÁLIS SZEMÉLYI SZÁMÍTÓGÉPEKET

- **ÜZEMBE HELYEZÉS**
- **GARANCIÁLIS ÉS GARANCIAIDŐN TÚLI JAVÍTÁS**
- **KARBANTARTÁS**
- **ALKALMAZÁSTECHNIKAI SZOLGÁLTATÁS**