MIKROSZÁMÍTÓGÉP MAGAZIN

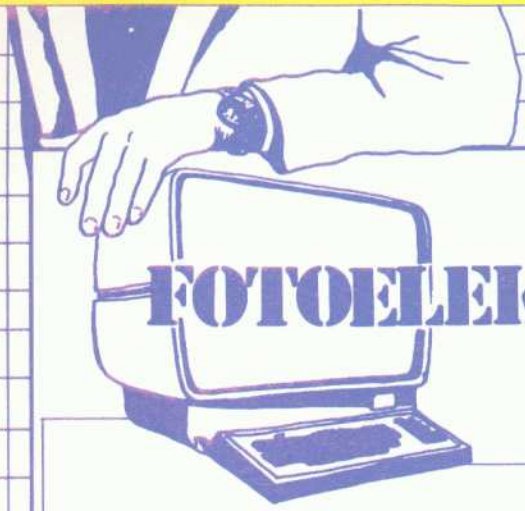
1986
10
Ára 30 Ft

PROGRAMTITKOSÍTÁS

COMMODORE PLUS/4 ISMERTETŐ

FORTH

MEPHISTO SZTORI



FOTOELEKTRONIK ISZ

Örömmel értesítjük Tisztelt Ügyfeleinket, hogy
Budapest VII., Dohány u. 5. szám alatt
új számítástechnikai szaküzletet nyitottunk.

VÉTEL — ELADÁS

Commodore 64-től . . .
. . . IBM PC AT-ig számítógépek garanciával,
Hi-Fi, video és egyéb műszaki cikkek
nagy választékával várjuk Önöket.

Telefon: 422-507

FOTO. ELEKTRONIK

4. ÉVFOLYAM
1986. 10. SZÁM

MIKROSZÁMÍTÓGÉP MAGAZIN

A NEUMANN JÁNOS SZÁMÍTÓGÉP-TUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

PR menedzser:
Pálhalmi Vali

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

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat
Felelős kiadó:
dr. Petrus György igazgató
Kiadóhivatal:
1065 Budapest VI., Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivatalokná
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.
Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.
86—253

Szikra Lapnyomda
Budapest (86-4121)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Címképünk:
A KODEX 2000 szövegszerkesztő

Tartalom

Teleteaching '86 nemzetközi távoktatási konferencia Budapesten

„Érik a szőlő, a kénszagú pincék
ajtaja tárva.
Izzik üvegre-csapódva a napfény,
röpköd a kárva.
Szárnyal a kedv, de a város a vár-
ra a mélyben hallgat.
Kong a harang a toronyban, őrli
a lassú unalmat."
(Fodor András: Kivárás)

Érik a szőlő, ősz van. Szeretem az őszt és szeretem Fodor András barátom verseit, mottót keresve a szerkesztőségi cikkhez, ezeket a Szekszárdról, szülővárosomról vagy harmincegy éve írt strófákat olvasgatom.

Szőlő, ősz, iskola, tanítás, október és egy nagy ugrás: távoktatás — így játszhatnánk az ismert asszociációs játékot.

A teleteaching szó a konferencia tiszteletére született, hiába keresné bárki is a szótárban. Angol barátaim is erősen kritizálták, meg is kértek, írjam mellé, hogy „remote education", így legalább az angolok is megértik. Odaírtam, így valóban jobb!

Van a szónak nagyon szép magyar változata is, távoktatás, de követni kell a nemzetközi szokásokat, ha már nemzetközi konferenciát rendezünk, akkor nevezzük el angolosan.

Teleteaching '86 — távoktatás és informatika, így teljes a cím, talán magyarázni sem kellene, hogy miről is szól ez a konferencia.

Az IFIP TC 3 (a nemzetközi információfeldolgozással foglalkozó Szövetség 3. sz. oktatási műszaki bizottsága) múlt évi norfolki ülésén fogadta el javaslatunkat, hogy rendezzünk konferenciát az informatikai tömegoktatásról, ezzel a problémával ugyanis a szövetség eddig nem foglalkozott. Miután tömegoktatásról van szó, a feladatot csak a hagyományostól eltérő módon, távoktatással lehet megoldani, ezért fontos tehát, hogy megismerjük a nemzetközi tapasztalatokat, azokat az új módszereket, tananyagokat és nem utolsósorban eszközöket, amelyekkel a távoktatás hatékonysága növelhető.

A Handelsblatt februári számában terjedelmes cikk foglalkozik a társadalom informatizálásával, amelyet a fejlődés egyik legfontosabb tényezőjének tart. A cikkíró szerint — nem tartom túlzásnak — 1990-re, tehát cca 5 év alatt az NSZK dolgozó lakosságának 70%-át az informatika különféle alkalmazására ki kell képezni.

Ha a képletet hazánkra alkalmazzuk, és figyelembe vesszük, hogy az ország műszaki színvonala, szervezettsége még nem éri el az NSZK-ét, akkor alacsonyabb faktorral kell számolnunk, úgy becsülhetjük, hogy nálunk az elkövetkező 5 évben a dolgozóknak „csak" az 50%-át kell informatikára kiképezni. Ez pedig azt jelenti, hogy 5 év alatt kb. 2,5 millió ember „beiskolázására" van szükség. Ezenkívül — természetesen — folyamatosan tanítani kell az ifjúságot, ők ugyanis ebben az évi cca félmilliós oktatási létszámban nincsenek benne.

Még egy másik tényezővel is számolni kell. Nevezetesen az informatikai ismereteket cca 3 évenként meg kell újítani, így az először kiképzetteket a 3 év után ismét tovább kell képezni, tehát 1990-ig valójában 3,5—4 millió fő informatikai kiképzése a feladat.

Ezt a gondolatsort — ami véleményem szerint valós — nem is érdemes folytatni, hiszen irreális volna elképzelni a probléma megoldását pl. tanfolyami alapon, hagyományos körülmények között, iskolapadban, tanárokkal. Egyáltalán honnan lenne ennyi tanár, hiszen a tanárokat is ki kell képezni!

A megoldás — a távoktatás. Amikor a konferenciát szervezni kezdtük, egyrészt a televíziós oktatásra gondoltunk, másrészt pedig a számítógéppel segített oktatásra. Azóta a fogalom kibővült, mert — a beküldött előadások és javaslatok nyomán

távoktatásnak nevezzük azt az oktatási formát, amikor a tananyagot a tanár és a diák között gép vagy adathordozó közvetíti.

Így a távoktatás körébe tartozik pl. az oktatófilm és a video, a hangszalag és a számítógéppel készített adathordozó (kazetta, floppy, videolemez stb.), maga a számítógép, a telefonhálózat, az adathálózat (videotex) és minden, ami a tananyagnak a széles körű terjesztését segíti. A távoktatásban fontos médium a könyv is, gondoljunk csak a programozott tankönyvekre, de nyilvánvalóan kitüntetett szerepe van a távoktatásban a rádiónak és a televíziónak.

A konferenciára 38 előadást fogadtunk el, a résztvevők 13 országból érkeznek (USA, Japán, Dánia, NSZK, Hollandia, Franciaország, Svájc, Spanyolország, Olaszország, Ausztria, Csehszlovákia, Bulgária és Thaiföld), és természetesen vannak hazai előadók is.

Az előadók egy része bemutat oktatóberendezéseket is, amelyekből kiállítást szervezünk. Eddig 15, ebből 5 külföldi szervezet jelentette be kiállítási szándékát. Nem tévedünk, ha úgy becsüljük, hogy a legérdekesebb és legújabb oktatási eszköz az interaktív videolemez lesz, de komoly érdeklődésre tarthat számot a folyamatos on-line kapcsolatban működő videotex terminál is. Az előbbit a Hollandiából érkező Quadramt és az Open University, valamint az NSZK-beli Telemedia cég, míg az utóbbit a Graz-i Egyetem (IIG) és a Mupid Computer Gesellschaft (MCG) mutatja be. A hazai szervezetek közül az OOK, néhány egyetem (BME, Agrártudományi Egyetem) és az SZKI jelezte kiállítási szándékát. Bemutatunk oktatófilmeket és videoanyagokat is, a Magyar Televízió, az MTA Műszerügyi Szolgálat és a Népszerű Tudományos Filmstúdió alkotásait.

Reméljük, sok és aktív résztvevője lesz a két kerekasztal-megbeszélésnek, az egyiket október 21-én rendezzük, a távoktatás kulcsproblémájáról: a tananyag (courseware) készítés technológiájáról, míg a másikat 23-án tartjuk az interaktív videolemezek, mint a legújabb oktatási eszköznek a szerepéről.

Nincs messze az az idő, amikor a jelenleg működő számítógépeket az ún. ötödik generációs gépek váltják fel, ami azt jelenti, hogy a mesterséges intelligencia bevonul az oktatásba is. A Számítástechnikai Kutató Intézet és Innovációs Központ (SZKI) a Teleteaching '86-hoz kapcsolódva a konferencia résztvevői részére október 24-én délután és 25-én egész nap MProlog tanfolyamot rendez, amelyen a résztvevők az SZKI számítógépen a gyakorlatban is kipróbálhatják ezt az intézetben fejlesztett korszerű logikai programozási eszközt. A tanfolyam ingyenes, a részvételre előzetesen, írásban kell jelentkezni.

Szeretettel várjuk az érdeklődőket, a pedagógusokat, az informatika szakembereit és a diákokat is, hogy elegendő ismerettel fogjunk hozzá a hazai informatikai távoktatási rendszer megvalósításához. Ne feledjük:

Szőlő, ősz, iskola, tanítás, október és TELETEACHING '86.

KOVÁCS GYŐZŐ

Részletes felvilágosítás az NJSZT Titkárságán (Bp. V., Báthori u. 16., tel.: 329-349; 329-390). A konferencia helye: TOT Mezőgazdasági Szövetkezetek Háza (Bp. XII., Normafa u. 54. Elérhető a 90-es számú autóbusszal).

A vezérprobléma – Algoritmizálás

Hányféleképpen helyezhetünk el egy rögzített, n×n-es sakktáblán n számú vezért úgy, hogy semelyik kettő ne álljon ütésben egymással (azaz minden sorban, oszlopban és átlós vonalban legfeljebb egy figura álljon)?
E feladat speciális eseteivel a Mikroszámítógép Magazin már foglalkozott. A probléma szépsége és megoldásának tanulságai miatt érdemes azonban még egyszer visszatérni rá.

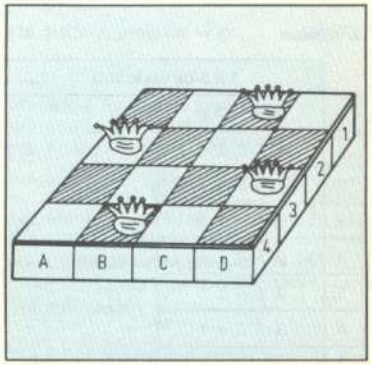
Ha n=1, akkor nyilván egyféleképpen helyezhetjük el az egy vezért, n=2 és n=3 esetén pedig könnyű meggyőződni arról, hogy a feltételeknek megfelelő elhelyezés nincs. Nem nehéz megtalálni n=4 esetén sem a lehetséges két megoldást. Ezek közül az egyik az *1. ábrán* látható, és jelölhetjük a következőképpen:
A2, B4, C1, D3 (vagy rövidebben: 2 4 1 3),
ami tehát azt jelenti, hogy az első vezér az A oszlop 2. sorában, a második a B oszlop 4. sorában, a harmadik a C oszlop 1. és végül a negyedik a D oszlop 3. sorában van. A másik lehetséges megoldás pedig: A3, B1, C4, D2.

Az 5x5-ös sakktáblához már több türelemre és kitartásra van szükségünk. Az öt vezért az öt oszlopba $5^5=3125$-féleképpen rakhatjuk fel. Ez már inkább a számítógépnek való feladat, erre már érdemes programot írni. Így nemcsak gyorsabban tudjuk megoldani a problémát, de annak az ellenőrzése is egyszerűbb, hogy valóban megkaptuk-e az összes megoldást, hiszen „csak" azt kell megvizsgálni, hogy programunk jól működik-e, ami jelen esetben könnyen belátható. Ahhoz, hogy a számítógépet használni tudjuk a probléma megoldásához, meg kell fogalmazni azt a megoldási algoritmust, amit a gép nyelvére lefordíthatunk. Érdemes ezt mindjárt általánosan leírni.

A V(1), V(2), . . ., V(n) tömbelemek értékei mutassák azt, hogy az 1., 2., . . ., n. oszlopban levő vezér hányadik sorban van. Ez azt is mutatja, hogy minden oszlopba pontosan egy vezért rakunk. Így ha a V tömb V(1), V(2), . . ., V(n) elemei egymástól függetlenül befutják az 1, 2, . . ., n értékeket, akkor ezzel az összes lehetséges elhelyezést modelleztük. Minden egyes elhelyezésről meg kell vizsgálni, hogy jó vagy rossz. Ha egy elhelyezésen belül az i és a k oszlopban álló vezér egy sorban van, akkor
V(i) = V(k),
ha pedig egy átlóban van, akkor
ABS(i—k) = ABS (V(i) — V(k))
teljesül. Így számunkra csak azok az elhelyezések lesznek megfelelőek, amelyekben semelyik két különböző oszlopban álló vezérre sem teljesül a most felírt két egyenlőség egyike sem. Ha valamelyik egyenlőség fennáll, akkor valamelyik két vezér ütésben áll, tehát az elhelyezés rossz.

A most elmondott algoritmus szerint működik az *1. program*, amely a HT—1080Z iskolaszámítógép BASIC nyelvén íródott. Ez nyilvánvalóan elég „lassú" lesz, hiszen sok felesleges elhelyezést is megvizsgál, de a további programokkal való összehasonlítás miatt érdemes volt elkészíteni. A vázolt algoritmus megvalósítása a 2Ø—1ØØ-as sorokban látható, a program többi része csak kiegészítő tartozék. A 3ØØ-as sorban levő A$=INKEY$ utasítás arra szolgál, hogy törli az INKEY függvény tároló mezejét (a 16537-es című memóriarekeszt). Így a program csak az "U'jabb sakkta'bla?" kérdés megjelenése után reagál válaszunkra, és elfelejti, ha futás közben például a SPACE billentyűvel babráltunk. Ezenkívül a 31Ø-es sorban levő RUN utasítás érdemel említést. Látható tehát, hogy a RUN kulcsszó nemcsak parancsként, hanem utasításként is használható. Ez itt azért kényelmes, mert törli a változók értékeit, sőt a tömbök méreteit is, ami majd a 4. programnál lesz lényeges. Így az új sakktábla vizsgálatánál a program alapállapotból indul. Ha a programot 10-nél nagyobb n értékre is akarjuk futtatni, akkor a V tömböt is dimenzionálni kell, de látni fogjuk, hogy ennek nem sok értelme lenne.

Az 1. program n=4 esetén 23 másodperc alatt adja meg az összes elhelyezést, az n=5 értéknél pedig már majdnem 5 percre van szüksége *(1. táblázat)*. A 6×6-os sakktábla vizsgálata már reménytelennek látszik ezzel a módszerrel.



1. ábra

1. táblázat

Sakktábla	Program				
	1.	2.	3.	4.	5.
4×4	23 s	13 s	2 s	3 s	2 s
5×5	4 min 57 s	3 min 12 s	9 s	9 s	7 s
6×6			31 s	23 s	20 s
7×7			2 min 22 s	1 min 32 s	1 min 17 s
8×8			11 min 1 s	5 min 56 s	5 min 6 s

A *2. táblázat* tartalmazza az 5×5-ös sakktáblán lehetséges összes elhelyezést. Ha ezt figyelmesen tanulmányozzuk, akkor észrevehetjük, hogy ha például az 1. és 10. sorokból az egy oszlopban levő számokat összeadjuk, akkor eredményül mindig 6-ost kapunk. Ugyanez lesz az eredmény a 2. és 9., a 3. és 8., a 4. és 7., valamint az 5. és 6. sorban levő megfelelő számok összeadásakor is. Ez nem véletlen, hiszen sakktáblánk szimmetrikus a középső sorára, tehát ha találtunk egy jó elhelyezést, akkor azt tükröz-

ve a *2. ábrán* c-vel jelölt egyenesre, egy másik jó elhelyezést kapunk. Ez algoritmusunk szempontjából azt jelenti, hogy ha V(1), V(2), . . ., V(n) egy jó elhelyezése a vezéreknek, akkor n−V(1)+1, n−V(2)+1, . . ., n−V(n)+1 is egy jó elhelyezés, és az előzőtől különbözik. Ezzel a vizsgálandó esetek számát a felére csökkentettük *(2. program)*. A futási idő nem csökkent a felére, mert programunk valamivel hosszabb lett.

2. táblázat

	5×5-ös sakktábla	
	A B C D E	
1.	1 3 5 2 4	
2.	1 4 2 5 3	
3.	2 4 1 3 5	
4.	2 5 3 1 4	
5.	3 1 4 2 5	
6.	3 5 2 4 1	
7.	4 1 3 5 2	
8.	4 2 5 3 1	
9.	5 2 4 1 3	
10.	5 3 1 4 2	

A 6×6-os sakktábla vizsgálata még így sem látszik biztatónak, jó lenne valamilyen módon tovább csökkenteni a futási időt. Az elkezdett szimmetriavizsgálat folytatása sajnos nem látszik használhatónak. Igaz ugyan, hogy sakktáblánk még további szimmetriákkal is rendelkezik — például a *3. ábrán* látható a, b, c, d egyenesek mindegyikére szimmetrikus, sőt ezen egyenesek metszéspontja körül 90, 180 és 270°-kal elforgatva szintén önmagába megy át —, de ha ezeket is ki akarnánk használni, az aránytalanul sok vizsgálattal járna, kis n értékekre ez inkább még növelné a futási időt. A bonyodalmat az okozza, hogy például a 4 1 3 5 2 elhelyezést akár az a, akár a c egyenesre tükrözzük, mindig a 2 5 3 1 4 elhelyezést kapjuk, de a 2 4 1 3 5 elhelyezést a c egyenesre tükrözve a 4 2 5 3 1, az a egyenesre tükrözve pedig az 5 3 1 4 2 elhelyezést kapjuk (3. ábra, 2. táblázat). A további szimmetriákat tehát azért nehéz kihasználni, mert bonyolult lesz annak a vizsgálata, hogy a kapott megoldások közül melyek a különbözőek.

Eddigi programjainkat tanulmányozva feltűnhet, hogy sok felesleges vizsgálatot végeznek. Például ha az 1. és 2. oszlopban levő vezérek egy sorban vagy egy átlóban vannak, akkor teljesen felesleges a többi vezér összes lehetséges elhelyezését is megvizsgálni. Érdemesebb tehát a vezéreket egyesével felrakni a sakktáblára, és mindig csak azt vizsgálni, hogy a már fent levők között van-e olyan, amelyikkel ütésben van. Hogy ez az ötlet mennyire hatékonynak bizonyul, azt mutatja az 1. táblázat 3. oszlopa, ahol a *3. program* futási idői láthatók. A 3. program az ún. backtrack algoritmus alapján működik (lásd: Mikroszámítógép Magazin 1985/2. szám, 10—12. oldal, 6. program), és segítségével már gyorsan megkapjuk a feladat megoldásait 8-nál nem nagyobb n értékekre.

Algoritmusunkat tovább gyorsíthatjuk, ha észrevesszük a következőket. Egy, a j sor k oszlopában álló vezér ütéskörzetét egyértelműen tudjuk jellemezni a következő három számmal:
j, k+j, k−j.
Ez azért van így, mert ha egy másik vezér az i sor t oszlopában van, akkor ez a két figura pontosan akkor üti egymást, ha vagy j=i, vagy k+j=t+i, vagy k−j=t−i teljesül.

Tegyük tehát a következőt. Ha egy új vezért felteszünk a k oszlop j sorába, akkor az S(j), A(k+j), B(n+k−j) tömbelemek értékét eggyel megnöveljük, ha pedig levesszük a vezért erről a mezőről, akkor az említett tömbelemeket eggyel csökkentjük. Ekkor minden új vezér feltevése előtt csak azt kell ellenőrizni, hogy ha azt a t oszlop i sorába kívánjuk tenni, akkor az S(i), A(t+i), B(n+t−i) tömbelemek értéke 0-val egyenlő-e. Ha mindegyik értéke nulla, akkor ez a mező még szabad; ha nem, akkor már van olyan vezér a sakktáblán, amelynek ütéskörzetébe esik.

A most elmondottak figyelembevételével kapjuk a 3. program módosításaként a *4. programot*. (Ez hasonlóan működik, mint a Mikroszámítógép Magazin 1984/5. számának 15. oldalán található 3. program.) Ha most az 1. táblázatra tekintünk, akkor látható, hogy n=4 esetén növekedett a futási idő. Ennél az n értéknél az adminisztráció még jobban növekedett, mint amennyivel csökkent a vizsgálatok száma. Az n=5 értékre már körülbelül az előző szinten vagyunk, a 8×8-as sakktáblánál pedig már majdnem a felére csökkent a futási idő.

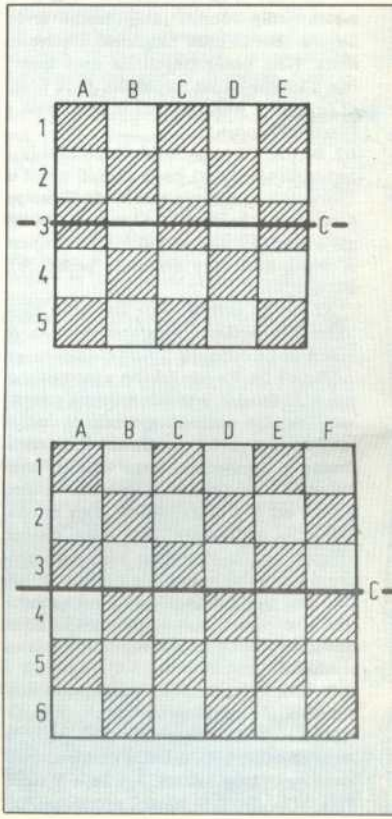
Tanulságos az *5. program* megtekintése is. Ezt a 4. programból úgy kaptuk, hogy abból minden, a feladat megoldása szempontjából lényegtelen utasítást, programsort töröltünk, és amit lehetett, tömörítettük. Ezzel sikerült elérni, hogy programunk listája lényegesen áttekinthetetlenebbé vált, a táblázatból viszont látható, hogy a futási idő csökkenése ezzel nincs összhangban. Ez a módosítás eredményezte a futási idő legrosszabb arányú csökkenését.

Az utolsó, a *6. program* a 3. program módosítása. Attól annyiban tér el, hogy az elhelyezéseket egy, a képernyőre kirajzolt sakktáblán szemlélteti *(4. ábra)*. A program kihasználja azt, hogy ha POKE utasítással a 15360—16383 című memóriarekeszek valamelyikébe egy karakter ASCII kódját írjuk, akkor a karakter a képernyő megfelelő helyén megjelenik. Ezt a programot futtatva nagyszerűen figyelemmel kísérhetjük a backtrack algoritmus működését is! Célszerű ehhez a számítógépet átkapcsolni félképernyőre. Ha futás közben például a SPACE billentyűt folyamatosan lenyomva tartjuk, akkor a program futása lelassul, az algoritmus működése követhetőbb.

Végezetül a *3. táblázatban* közöljük a feladat megoldását az n=4, 5, 6, 7, 8, 9, 10 értékekre.

MAJOR ZOLTÁN

2. ábra



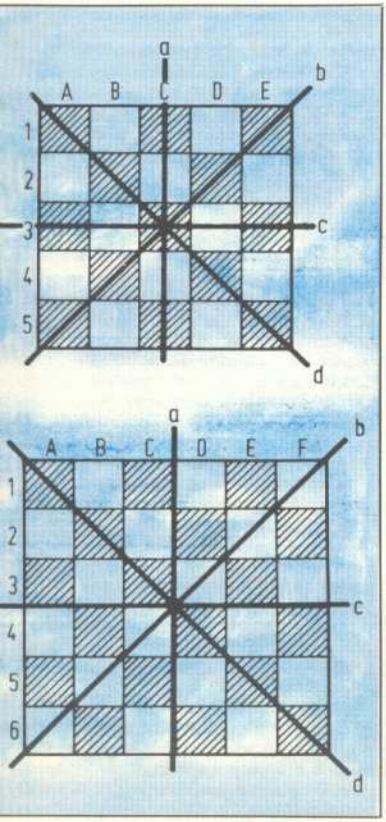
3. táblázat

Sakktábla	Elhelyezések száma
4×4	2
5×5	10
6×6	4
7×7	40
8×8	92
9×9	352
10×10	724

1. program

```
10 GOSUB 600
20 I=N
30 V(I)=V(I)+1
40 IF V(I)>N THEN V(I)=1 : I=I-1
   : IF I=0 THEN 300 ELSE 30
```

3. ábra




```
70 FOR J=1 TO N-1 : FOR K=J+1
   TO N
80 IF V(I)=V(K) OR
   K-J=ABS(V(J)-V(K)) THEN 20
90 NEXT K,J
100 GOSUB 500 : GOTO 20

300 PRINTÉ 975,"U'jabb sakkta'bla ?"; :
   A$=INKEY$
310 IF INKEY$="" THEN 310 ELSE
   RUN
330 REM A program ve'ge

499 REM Az eredme'nyek kiirata'sa
500 M=M+1 : PRINT M,
510 FOR J=1 TO N : PRINT
   CHR$(64+J); CHR$(48+V(J)) ;" ";
   : NEXT J: PRINT
540 RETURN

599 REM A kezdeti e'rte'kek bea'llita'sa
600 DEFINT A-Z
610 CLS : INPUT"N × N-es sakkta'bla,
   N=";N : PRINT : PRINT
620 M=0 : FOR J=1 TO N : V(J)=1 :
   NEXT J
630 RETURN
```

2. program
(Az 1. program módosítása)

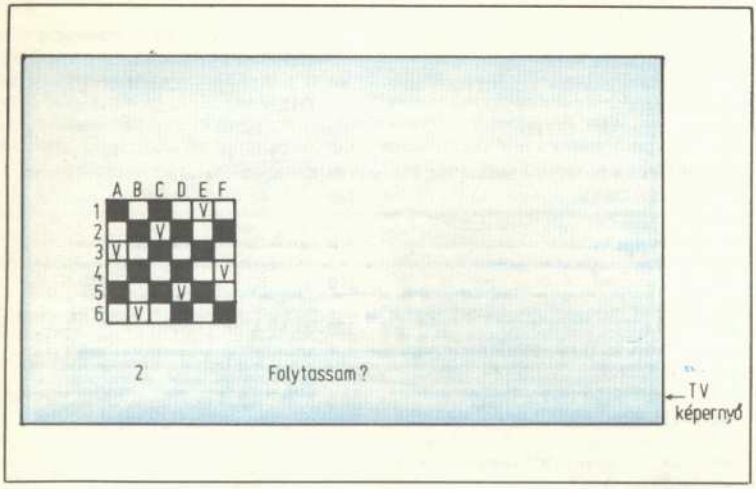
```
.
.
.
.
30 V(I)=V(I)+1 : IF I=1 THEN 50
40 IF V(I)>N THEN V(I)=1 : I=I-1
   : GOTO 30
50 IF 2*V(1)>N+1 THEN 300
60 IF 2*V(1)=N+1 AND 2*V(2)>N
   THEN 300

.
.
520  M=M+1 : PRINT M;
530 FOR J=1 TO N : PRINT
   CHR$(64+J); CHR$(49+N-V(J))
   ;" "; : NEXT J : PRINT
.
.
.
```

3. program

```
10 GOSUB 600
20 K=2 : V(1)=V(1)+1 : IF
   2*V(1)>N+1 THEN 300
30 J=1
40 FOR I=1 TO K-1
50 IF J=V(I) OR K-I=ABS(J-V(I))
   THEN 100
60 NEXT I
70 V(K)=J : K=K+1 : IF K<=N
   THEN 30 ELSE K=N
80 If 2*V(1)=N+1 AND 2*V(2)> N
   THEN 300
90 GOSUB 500
100 J=J+1 : IF J<=N THEN 40
110 K=K-1 : IF K=1 THEN 20
120 J=V(K) : GOTO 100

300 PRINTÉ 975,"U'jabb sakkta'bla ?", :
   A$=INKEY$
310 IF INKEY$="" THEN 310 ELSE
   RUN
320 REM A program ve'ge

499 REM Az eredme'nyek kiirata'sa
500 M=M+1 : PRINT M,
510 FOR J=1 TO N : PRINT
   CHR$(64+J); CHR$(48+V(J)) ;" ";
   : NEXT J : PRINT
520 M=M+1 : PRINT M,
530 FOR J=1 TO N : PRINT
   CHR$(64+J); CHR$(48+N-V(J))
   ;" "; : NEXT J : PRINT
540 RETURN

599 REM A kezdeti e'rte'kek bea'llita'sa
600 DEFINT A-Z
610 CLS : INPUT"N × N-es sakkta'bla,
   N=";N : PRINT : PRINT
620 M=0
630 RETURN
```

4. program
(A 3. program módosítása)

```
10 GOSUB 600
20 V(1)= V(1)+1 : IF 2*V(1)>N+1
   THEN 300
30 J=V(1) : S=1 : GOSUB 200 : K=2
40 J=1
50 IF S(J)>0 OR A(K+J)>0 OR
   B(N+K-J)>0 THEN 90
60 S=1 : GOSUB 200 : V(K)=J :
   K=K+1 : IF K<=N THEN 40
   ELSE K=N
70 IF 2*V(1)=N+1 AND 2*V(2)>N
   THEN 300
80 GOSUB 500
90 J=J+1 : IF J<=N THEN 50
100 K=K-1 : J=V(K) : S=-1 :
   GOSUB 200 : IF K=1 THEN 20
   ELSE 90

199 REM A mutato'k bea'llita'sa
200 S(J)=S(J)+S : A(K+J)=
   A(K+J)+S :
   B(N+K-J)=B(N+K-J)+S
```



4. ábra

```
210 RETURN
.
.
.
.
620 M=0 : K=1 : DIM A(2*N),B(2*N)
630 RETURN
```

5. program
(A 4. program módosítása)

```
1 CLS:DEFINTA—Z:INPUTN:
  DIMA(2*N),B(2*N):M=0:K=1
2 V(1)=V(1)+1:IF2*V(1)>N+1:
  THENSTOPELSEJ=V(1);
  S(J)=S(J)+1:A(J+1)=A(J+1)+1:
  B(N+1-J)=B(N+1-J)+1:K=2
3 J=1
4 IFS(J)>0THEN9ELSEIFA(K+J)>
  0THEN9ELSEIFB(N+K-J)>
  0THEN9
5 S(J)=S(J)+1:A(J+K)=A(J+K)+1:
  B(N+K-J)=B(N+K-J)+1:V(K)=J:
  K=K+1:IFK<=NTHEN3EL
  SEK=N
6 IF2*V(1)=N+1AND2*V(2)>
  NTHENSTOPELSEM=M+1:
  PRINTÉ9,M
7 J=J+1:IFJ<=NTHEN4ELSEK=
  K-1:J=V(K):S(J)=S(J)-1:
  A(J+K)=A(J+K)-1:
  B(N+K-J)=B(N+K-J)-1:
  IFK=1THEN2ELSE7
```

6. program
(A 3. program módosítása)

```
10 GOSUB 600
20 K=2 : V(1)=V(1)+1 : IF V(1)>N
   THEN 300
25 POKE T-63+V(1)*64,M(1) :
   M(1)=PEEK(T+1+V(1)*64) :
   POKE T+1+V(1)*64,86
.
.
.
65 POKE T+V(K)*64+K,M(K) :
   M(K)= PEEK(T+J*64+K): POKE
   T+J*64+K,86
70 V(K)=J : K=K+1 : IF K<=N
   THEN 30 ELSE K=N
80 M=M+1 : PRINTÉ960,M; :
   PRINTÉ975,"Folytassam ?";
90 IF INKEY$="" THEN 90 ELSE
   PRINTÉ975,CHR$(205);
100 J=J+1 : IF J<=N THEN40
110 POKE T+V(K)*64+K,M(K):
   K=K-1 : IF K=1 THEN 20
120 J=V(K): GOTO 100
.
.
.
620 M=0 : T=15425 : CLS
630 FOR J= 1 TO N : M(J)=128 :
   POKE T-64+J,64+J : POKE
   T-1+J*64,48+J
640 FOR K=1 TO N : L=(K+J)/2 : IF
   K+J=2*L THEN POKE
   T+J*64+K,191
650 NEXT K,J
660 RETURN
```

A Mikromagazin ez évi 7. számában már megjelent egy, a gépi kódú grafika készítését segítő program az iskolaszámítógépre. A program egyszerűsége ellenére sokoldalúan használható, de a képek tárolásának megoldása nem eléggé helytakarékos: ugyanis a VIDEORAM megadott címétől kezdve megadott számú bájtot tárol sorfolytonosan, így például egy 10 × 10-es képmátrix 10 teljes sornyi helyet, azaz 640 bájtot foglal el a tárban. Másik hátránya, hogy használatához „térkép"-re van szükség, amely nyilvántartja a képek által elfoglalt területeket. Nem rendelkezik továbbá védelemmel sem, ezért alkalmazása nagy figyelmet kíván, mert a program felülírhatja önmagát, a rendszerterületet stb.

Most egy lényegesen „komfortosabb" programot mutatok be *(1. lista)*, amely elkerüli az amúgy is kis tárterület üres képmezőkkel való feltöltését, gondoskodik a képek tárban való elhelyezéséről, így nincs szükség térképre, és a hibák ellen sokoldalú védelemmel rendelkezik.

A program a BASIC interpreter lemezkezelő utasításait használja, BASIC-ből tehát kényelmesen kezelhető. 16 kbájtos gépre készült, de majdnemhogy relokálható. 64 kbájtos gép esetén — ha a programot a tár végére kívánjuk elhelyezni — csak a KEZD címkével jelölt direktívát kell átírni (például KEZD:EQU 0E000H).

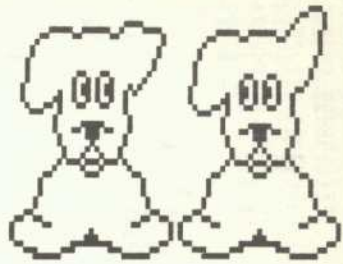
A mozgatni kívánt képeket vagy képrészleteket először BASIC-ben kell összeállítani, majd a program segítségével lehet a tárban elhelyezni. A tárolt max. 16 db kép jellemzőit (tárcím: 2 bájt, VIDEORAM cím: 2 bájt, az egy sorban lévő karakterek száma: 1 bájt, a sorok száma: 1 bájt) a program után, a 7170H címen kezdődő 16 × 6 bájtos tábla tárolja. A VIDEORAM cím tárolása elkerülhető lenne, de a program esetleges átalakításánál, fejlesztésénél hasznos lehet. A 16 × 6 (4CH) bájtos tábla utáni terület a képek tárolására szolgál.

A programot az OPEN7 vagy az OPENF utasítással kell aktivizálni: OPEN7 a 16 kbájtos, OPENF a 64 kbájtos gépeknél. Az OPEN beállítja a tárméretre utaló TARVEG rekesz tartalmát (a kezdeti érték 7F, ez a 16 kbájtos gépekre jellemző), és betölti a 16 elemes tábla első elemének első 2 bájtjába az első, 0. számú kép tárolására szolgáló terület kezdőcímét.

A kép tárolására a SAVEXX XX,XX,XX (VIDEORAM kezdőcím, soronkénti bájtszám, sorok száma) utasítás szolgál, ahol X egy hexadecimális számjegy. A SAVE hatására a kép az OPEN által kijelölt tárterületre kerül, és a következő hatbájtos táblaelem első két bájtja a következő kép tárolására szolgáló tármező kezdőcímére mutat.

A következő kép tárolása előtt a táblában hat bájtot „lépni" kell. Erre használatos a NAME utasítás. A NAME1 a táblaelemek első bájtjára mutató IX indexregiszter tartalmát hattal növeli, a NAME0 hattal csökkenti. A táblában az előre- vagy

Gépi kódú grafikát segítő program



visszalépést a COUNTR (képszámláló) rekesz számlálja. A táblából való kilépés — a 0. képnél kisebb vagy a 15. képnél nagyobb sorszámú kép — esetén FC ERROR hibajelzés keletkezik.

Ha túl sok a kép és kicsi a fenntartott tárterület, akkor előfordulhat, hogy a tárban a kép már nem fér el. Ebben az esetben a „tárvég" elérésekor a SAVE nem kerül végrehajtásra, és a program OM ERROR hibajelzéssel leáll.

A tárolt képek a LOADXXXX (VIDEORAM cím) utasítással jeleníthetők meg. Ha még nem tárolt képet próbálunk megjeleníteni, az hibát okoz. A különböző képek megjelenítéséhez a SAVE-nél leírtakhoz hasonló módon a tárban a NAMEØ vagy a NAME1 utasításokkal kell „lapozni".

A SAVE és a LOAD közös jellemzője, hogy csak a VIDEORAM területére (3CØØH—3FFFH) érvényesek. Ha az utasítások ezen kívüli területet érintenek, a program FC ERROR-ral leáll.

További szolgáltatás a CLOSE utasítás. Ez a IX indexregisztert az első képre és a képszámlálót Ø-ra állítja vissza. Használata akkor célszerű, ha a visszafelé lapozás túl sok NAMEØ utasítást igényelne. Például a 13. képről a visszatérés az 1. képre egy CLOSE és NAME1 utasításpárral egyszerűbb, mint 12 NAMEØ alkalmazása.

Az utasítások parancsként is használhatók az alábbiak szerint:

OPEN7, OPENF — megnyitás és tárméret-beállítás, 7: 16 kbájtos gép, F: 64 kbájtos gép (egy programban csak egyszer használandó).

NAMEØ, NAME1 — lapozás a képeket tároló tármezőben (a táblaelemekre mutató IX indexregiszter és a COUNTR változtatása)
Ø: vissza, 1: előre egy képpel.

SAVEXXXX,XX,XX — képtárolás; az argumentumban a VIDEORAM mező kezdőcíme, a soronkénti karakterszám, a sorok száma (a számok hexadecimálisak).

LOADXXXX — képmegjelenítés; az argumentumban a VIDEORAM mező

1. lista

```
                        ;GRAPHA: GEPI KODU GRAFIKAT SEGITO PROGRAM
                        ;
                        ;A BASIC BOVITES
                                ORG  4179H
 5 4179 C30470                  JP   OPEN
                                ORG  4185H
 7 4185 C32F70                  JP   CLOSE
 8 4188 C39470                  JP   XLOAD
                                ORG  418EH
10 418E C33670                  JP   NAME
                                ORG  41A0H
12 41A0 C37470                  JP   SAVE
                                ORG  41E2H
14 41E2 E9                      JP   (HL)
15                      KEZD:   EQU  7000H
16                              ORG  KEZD
17 7000 0000            POINTR: DW   0          ;MUTATO A TABLAK 1. BYTE-JARA
18 7002 00              COUNTR: DB   0          ;KEPSZAMLALO
19 7003 7F              TARVEG: DB   7FH        ;KEZDETBEN 16 KBYTE RAM
20                      ;
21                      ;OPEN?.OPENF
22 7004 117071          OPEN:   LD   DE.KEZD+170H
23 7007 ED530070                LD   (POINTR).DE
24 700B DD2A0070                LD   IX.(POINTR)
25 700F 7E                      LD   A.(HL)
26 7010 FE37                    CP   37H    ;A 7 KODJA?
27 7012 280A                    JR   Z.TOVABB:TARVEG 7FFFH
28 7014 FE46                    CP   46H    ;AZ F KODJA?
29 7016 C24A1E                  JP   NZ.1E4AH:FC ERROR
30 7019 3EFF                    LD   A.0FFH
31 701B 320370                  LD   (TARVEG).A:TARVEG FFFFH
32 701E 23              TOVABB: INC  HL
33 701F E5                      PUSH HL
34 7020 2A0070                  LD   HL.(POINTR)
35 7023 114C00                  LD   DE.4CH :16 KEP HASZNALHATO
36 7026 19                      ADD  HL.DE  :ELSO KEP ELSO BYTE CIME
37 7027 DD7500                  LD   (IX+0).L
38 702A DD7401                  LD   (IX+1).H
39 702D E1                      POP  HL
40 702E C9                      RET
41                      ;
42                      ;CLOSE
43 702F DD2A0070        CLOSE:  LD   IX.(POINTR):VISSZA AZ ELSO KEPRE
44 7033 AF                      XOR  A
45 7034 320270                  LD   (COUNTR).A:0 --) KEPSZAMLALO
46 7037 C9                      RET
47                      ;
48                      ;NAMEØ.NAME1
49 7038 FD210270        NAME:   LD   IY.COUNTR:IY A KEPSZAMLALORA MUTAT
50 703C 7E                      LD   A.(HL)
51 703D FE30                    CP   30H    ;A 0 KODJA?
52 703F 281C                    JR   Z.XØ
53 7041 FE31            X1:     CP   31H    ;AZ 1 KODJA?
54 7043 C29719                  JP   NZ.1997H:NEM 0,1. SN ERROR
55 7046 FD3400                  INC  (IY+0) :KEPSZAMLALAS (+1)
56 7049 FD7E00                  LD   A.(IY+0)
57 704C FE10                    CP   10H    ;16. KEP?
58 704E 2807                    JR   Z.FC1
59 7050 110600                  LD   DE.6   ;EGY KEPPEL ELORE
60 7053 DD19                    ADD  IX.DE
61 7055 23                      INC  HL
62 7056 C9                      RET
63 7057 FD3500          FC1:    DEC  (IY+0) :HIBA. VISSZAALLITAS
64 705A C34A1E                  JP   1E4AH  :FC ERROR
65 705D FD3500          XØ:     DEC  (IY+0) :KEPSZAMLALAS (-1)
66 7060 FD7E00                  LD   A.(IY+0)
67 7063 FEFF                    CP   0FFH   ;-1. KEP?
68 7065 2807                    JR   Z.FCØ
69 7067 11FAFF                  LD   DE.0FFFAH:-6: EGY KEPPEL VISSZA
70 706A DD19                    ADD  IX.DE
71 706C 23                      INC  HL
72 706D C9                      RET
73 706E FD3400          FCØ:    INC  (IY+0) :HIBA. VISSZAALLITAS
74 7071 C34A1E                  JP   1E4AH  :FC ERROR
75                      ;
76                      ;SAVEXXXX,XX,XX (VIDEOCIM, SORHOSSZ, SOROK SZAMA)
77 7074 CD3871          SAVE:   CALL ADDR   :VIDEOCIM BE
78 7077 DD7302                  LD   (IX+2),E:VIDEOCIM LSB --) TAR
79 707A DD7203                  LD   (IX+3),D:VIDEOCIM MSB --) TAR
80 707D 23                      INC  HL     :VESSZO ATLEPESE
81 707E CD4B71                  CALL EADDR  :SORHOSSZ BE
82 7081 DD7304                  LD   (IX+4),E:SORHOSSZ --) TAR
83 7084 23                      INC  HL     :VESSZO ATLEPESE
84 7085 CD4B71                  CALL EADDR  :SOROK SZAMA BE
85 7088 DD7305                  LD   (IX+5),E:SORSZAM --) TAR
86                      ;TABLAELEM 6 BYTE-JA KESZ
87 708B E5                      PUSH HL
88 708C CDA770                  CALL PMOVE  :ELOKESZULET AZ ATVITELHEZ
89 708F CDC070                  CALL SMOVE  :ATVITEL: VIDEO --) TAR
90 7092 E1                      POP  HL
91 7093 C9                      RET         :KEPTAROLAS KESZ
92                      ;
93                      ;LOADXXXX (VIDEOCIM)
94 7094 CD3871          XLOAD:  CALL ADDR
95 7097 DD7302                  LD   (IX+2),E
96 709A DD7203                  LD   (IX+3),D
97 709D E5                      PUSH HL
98 709E CDA770                  CALL PMOVE  :ELOKESZULET AZ ATVITELHEZ
99 70A1 EB                      EX   DE,HL
100 70A2 CD0971                 CALL LMOVE  :ATVITEL: TAR --) VIDEO
101 70A5 E1                     POP  HL
102 70A6 C9                     RET         :KEPBETOLTES A VIDEORA KESZ
103                     ;
104                     ;PMOVE SZUBR:TEMP, HL, DE BEALLITASA
105 70A7 3E40           PMOVE:  LD   A.40H
106 70A9 DD9604                 SUB  (IX+4) :VIDEOCIM NOVEKMENY (KOV. SOR)
107 70AC 32BF70                 LD   (TEMP).A
108 70AF DD7E05                 LD   A.(IX+5)
109 70B2 DD6E02                 LD   L.(IX+2):EGY SOR ATVITEL KEZDET
110 70B5 DD6603                 LD   H.(IX+3)
111 70B8 DD5E00                 LD   E.(IX+0)
112 70BB DD5601                 LD   D.(IX+1)
113 70BE C9                     RET
114 70BF 00             TEMP:   DB   0
115                     ;
116                     ;SMOVE SZUBR: ATVITEL: VIDEO --) TAR
117 70C0 CD2471         SMOVE:  CALL VIDEO
118 70C3 0600                   LD   B.0    ;BC BEALL.. MAJD 1 SOR ATVITELE
119 70C5 DD4E04                 LD   C.(IX+4)
120 70C8 EDB0                   LDIR        :EGY SOR ATVITEL VEGE
```

```
121 70CA 3D             DEC  A       ;UJ SOR KEZDESE
122 70CB 2808           JR   Z,SVEGE;NINCS TOBB SOR
123 70CD 0600           LD   B,0     ;(TEMP) --> BC
124 70CF F5             PUSH AF
125 70D0 3A8F70         LD   A,(TEMP)
126 70D3 4F             LD   C,A
127 70D4 F1             POP  AF
128 70D5 09             ADD  HL,BC  ;UJ KEPSOR TARCIME A HL-BEN
129 70D6 18EB           JR   SMOVE
130 70D8 DD4605  SVEGE: LD   B,(IX+5);SOROK SZAMA --> B
131 70DB DD6E00         LD   L,(IX+0);UJ TARCIM KISZAMITASA --> TABLA
132 70DE DD6601         LD   H,(IX+1)
133 70E1 1600           LD   D,0
134 70E3 DD5E04         LD   E,(IX+4)
135 70E6 19      SOR:   ADD  HL,DE
136 70E7 10FD           DJNZ SOR
137             ;TARTERULET-TULLEPES VIZSGALATA
138 70E9 3A0370         LD   A,(TARVEG)
139 70EC FE7F           CP   7FH
140 70EE 280A           JR   Z,T7
141 70F0 7C      TF:    LD   A,H
142 70F1 E6F0           AND  0F0H   ;ELSO 4 BIT
143 70F3 FE00           CP   0      ;FFFFH?
144 70F5 CA7A19         JP   Z,197AH;IGEN. OM ERROR
145 70F8 1808           JR   UJCIM
146 70FA 7C      T7:    LD   A,H
147 70FB E6F0           AND  0F0H   ;ELSO 4 BIT
148 70FD FE80           CP   80H    ;7FFF?
149 70FF CA7A19         JP   Z,197AH;IGEN. OM ERROR
150 7102 DD7506  UJCIM: LD   (IX+6),L
151 7105 DD7407         LD   (IX+7),H;UJ CIM A HELYEN VAN
152 7108 C9             RET
153             ;
154             ;LMOVE SZUBR: ATVITEL: TAR --> VIDEO
155 7109 EB      LMOVE: EX   DE,HL
156 710A CD2471         CALL VIDEO
157 710D EB             EX   DE,HL
158 710E 0600           LD   B,0    ;BC BEALL.. MAJD 1 SOR ATVITELE
159 7110 0D4E04         LD   C,(IX+4)
160 7113 EDB0           LDIR
161 7115 3D             DEC  A
162 7116 C8             RET  Z
163 7117 0600           LD   B,0    ;(TEMP) --> BC
164 7119 F5             PUSH AF
165 711A 3A8F70         LD   A,(TEMP)
166 711D 4F             LD   C,A
167 711E F1             POP  AF
168 711F EB             EX   DE,HL
169 7120 09             ADD  HL,BC
170 7121 EB             EX   DE,HL  ;UJ VIDEOCIM A DE-BEN
171 7122 18E5           JR   LMOVE
172             ;
173             ;A VIDEO RAM-ON BELUL VAN-E?
174 7124 F5      VIDEO: PUSH AF
175 7125 AF             XOR  A      ;0 --> C
176 7126 7C             LD   A,H
177 7127 D63C           SUB  3CH
178 7129 DA4A1E         JP   C,1E4AH;FC ERROR
179 712C AF             XOR  A      ;0 --> C
180 712D 3E40           LD   A,40H
181 712F 94             SUB  H
182 7130 DA4A1E         JP   C,1E4AH
183 7133 CA4A1E         JP   Z,1E4AH;FC ERROR
184 7136 F1             POP  AF
185 7137 C9             RET
186             ;
187             ;A CIMEK TOMORITESE 2 BYTE-BA
188 7138 CD5F71  ADDR:  CALL CHARIN
189 713B CB27           SLA  A      ;BITEK A 4 MSB HELYRE
190 713D CB27           SLA  A
191 713F CB27           SLA  A
192 7141 CB27           SLA  A
193 7143 57             LD   D,A
194 7144 23             INC  HL
195 7145 CD5F71         CALL CHARIN
196 7148 B2             OR   D
197 7149 57             LD   D,A    ;"H" BYTE KESZ
198 714A 23             INC  HL
199 714B CD5F71  EADDR: CALL CHARIN ;1 BYTE-OS CIM BE
200 714E CB27           SLA  A      ;BITEK A 4 MSB HELYRE
201 7150 CB27           SLA  A
202 7152 CB27           SLA  A
203 7154 CB27           SLA  A
204 7156 5F             LD   E,A
205 7157 23             INC  HL
206 7158 CD5F71         CALL CHARIN
207 715B B3             OR   E
208 715C 5F             LD   E,A    ;"L" BYTE KESZ
209 715D 23             INC  HL
210 715E C9             RET
211 715F AF      CHARIN: XOR  A      ;CIMKARAKTER BE
212 7160 86             ADD  A,(HL)
213 7161 FE40           CP   40H    ;BETU?
214 7163 DA6B71         JP   C,NUM  ;NEM
215 7166 D637           SUB  37H    ;IGEN
216 7168 E60F    NUM:   AND  0FH    ;MASZKOLAS 4 LSB-RE
217 716A C9             RET
218             END
```

2. lista

```
10 REM GEPI KODU GRAFIKA DEMO PROGRAM (16 KBYTE-OS GEPRE)
20 CLS:OPEN7
30 DATA0,0,0,0,0,0,0,0,0,0,0,0,160,140,131,169,0,0
40 DATA0,0,0,176,140,140,164,152,140,140,164,152,129,0,0,150,0,0
50 DATA0,0,150,0,0,0,0,0,0,0,0,0,0,160,133,0,0
60 DATA0,170,0,0,0,0,0,0,176,0,160,144,0,0,0,150,0,0,0
70 DATA0,150,0,0,0,0,144,170,148,149,189,170,0,152,134,0,0,0,0
80 DATA0,137,176,152,134,149,130,140,129,137,134,0,149,0,0,0,0,0
90 DATA0,0,0,0,0,170,0,160,152,188,156,176,0,170,0,0,0,0,0
100 DATA0,0,0,0,0,165,0,0,155,145,0,160,133,0,0,0,0,0
110 DATA0,0,0,0,160,134,140,166,140,166,140,134,164,0,0,0,0,0
120 DATA0,0,0,150,129,0,0,0,131,129,0,0,0,131,148,0,0,0
130 DATA0,0,168,129,0,0,0,0,0,0,0,0,0,0,169,0,0,0
140 DATA152,134,137,164,144,0,0,0,0,0,0,0,0,0,176,140,131,140,144
150 DATA165,144,0,0,0,0,160,176,190,180,176,0,0,0,0,0,176,133
160 DATA0,130,139,140,140,134,131,0,0,0,130,131,140,140,140,131,0,0
170 FORI=1T014:FORJ=1T018
180 READA:IFA=0THENA=128
190 PRINTCHR$(A);
200 NEXTJ:PRINT:NEXTI
210 SAVE3C00,18,0E
220 PRINT@12," ";
230 PRINT@75,CHR$(152);CHR$(131);CHR$(131);CHR$(140);CHR$(140);CHR$(144);
240 PRINT@263,CHR$(168);:PRINT@265,CHR$(149);CHR$(190);
250 PRINT@141," ";CHR$(168);CHR$(129);
260 PRINT@205," ";CHR$(160);CHR$(133);
270 PRINT@269,CHR$(152);CHR$(129);
280 NAME1:SAVE3C00,18,0E
290 FORI=0TO70:NEXT:NAME0:LOAD3C00
300 FORI=0TO70:NEXT:NAME1:LOAD3C00
310 GOTO290
```

Ez a program egy kutyát rajzol

CLOSE — visszaállás a Ø. képre.

A bemutatott formátumok betartása kötelező!

Hibajelzések:

SN ERROR — általános szintaxishiba vagy rossz argumentum a NAME utasításban.

FC ERROR — rossz argumentum az OPEN utasításban; kilépés a 16×6 bájtos táblából (túl sok NAMEØ vagy NAME1); a SAVE és a LOAD a VIDEORAM-on kívüli területet érint.

OM ERROR — A legnagyobb létező tárcímnél nagyobb címre történő tárolási kísérlet (a SAVE ilyenkor nem kerül végrehajtásra).

A program az EDI assembler editorral készült. A fordítást OPTION C-vel kazettára kell végezni, az EXEC. ADDR. 6CCH. kezdőcíme (hexadecimális).

A gépi kódú program betöltése előtt BASIC hideginditást kell végezni: ki-, majd bekapcsolás vagy SYSTEM és /Ø, és az ORG-nak megfelelően kell a RAMTOP-ot megadni (ez 7ØØØH esetén 28672). Aki a gépi kódú programozásban járatos, a program némi átalakításával a RAMTOP-ot a rendszerváltozó területre a programból is betöltheti; ilyenkor a BASIC programot feltétlenül CLEAR utasítással kell kezdeni.

A különböző képek megjelenítése között szükség lehet a képernyő törlésére. Mivel a CLS erre megítélésem szerint nem elég gyors, igényesebb felhasználó számára az a célszerű, ha a képernyő törlésére új gépi kódú rutint ír.

A program lehetőségeinek bemutatásához érdemes a *2. listán* közölt BASIC DEMO programot lefuttatni. Ekkor megjelenik az itt is bemutatott két kutyaalak. Célszerű a DEMO program „alakitgatásával" a különböző képmozgatási lehetőségeket és a program hibaellenőrző rendszerét is tanulmányozni.

A program használatakor a SYSTEM paranccsal óvatosan kell bánni. Az esetleges hibák elkerülésére SYSTEM/ formában alkalmazzuk, vagy a 41E2H rekesz tartalmát állítsuk vissza C9H-ra (BASIC-ből POKE16866,195)!

Bár a program hosszas nyúzáson ment keresztül, a biztonság kedvéért a hosszabb BASIC programokat a gépi kódú program alkalmazása előtt vegyük fel kazettára!

A program esetleges bővítésekor a 22-es sorszámú utasításban a 17ØH értékét a program mindenkori hosszának megfelelően írjuk át, hogy a program mögé mutasson!

NAGY IMRE

Tévedések nyomában

Néhány nappal azután, hogy e cikksorozat második részének a kéziratát leadtam, kezembe került egy szerény külsejű, de értékes tartalmú füzet. Erdős Zoltán *Rendszerváltozók és I/O címek* című összeállítását a NOVOTRADE—OCTASOFT jelentette meg, meglepően rövid átfutási idővel. (Az előszó dátuma és a megjelenés között eltelt idő kevesebb egy hónapnál.) Nagyon sok hasznos információt találtam benne a C16 és testvéreinek lelkivilágával kapcsolatban.

Böngészés közben egy olyan információra bukkantam, amely ellentmondott korábbi ismereteimnek. Utánanéztem, hogy mi az igazság: ez adta az ötletet ehhez a cikkhez.

Mi van a $13 címen?

A VC20 és C64 típusú gépeken ezen a címen az „aktív I/O egység" található, és nagyon sokáig úgy tudtam, hogy a C16-on sincs ez másképp. Éppen ezért meglepett, amikor a fentebb említett füzetben a következő szöveget olvastam: „CHANNL 0013 flag: input promt (1 = input promt elnyomás)". Ehhez hasonló információt találtam az időközben külföldről beszerzett forrásokban is. Mi az igazság?

Utánanéztem. Megtudtam, hogy a $13 címen lévő rendszerváltozót azok a BASIC utasítások használják, melyek egy OPEN BASIC utasítással megnyitott adatállományon végeznek beviteli/kiviteli műveleteket. Az utasítás végrehajtásának kezdetén ide kerül annak az állománynak a logikai száma, melyen a művelet végrehajtódik. Az interpreter végrehajtó modulja innen tölti a megfelelő regiszterbe a CHKIN és CHKOUT, valamint a CLRCHN KERNAL rutinok hívása előtt a logikai fájlszámot. A beviteli/kiviteli művelet végén a $13 címre ismét Ø kerül, jelezve, hogy megint az alapértelmezés szerinti fájlok (a billentyűzet és a képernyő) vannak érvényben. Így van ez a C16-on is.

Honnan van hát a félreértés? Elterjedt programozási fogás az INPUT utasítás által kiírt kérdőjel elnyomására az alábbi BASIC utasítássorozat használata:

```
...
POKE 19,1
INPUT A$
POKE 19,Ø
...
```

Az INPUT és az INPUT# utasítások feldolgozását a bevezető és a lezáró néhány gépi utasítástól eltekintve ugyanaz a ROM-rutin végzi. Az eltérő részben dolgozik az INPUT# rutin a $13 címen levő változóval. A közös végrehajtó részben csak egy vizsgálat történik ennek a változónak a tartalmára vonatkozóan. Ha nullát tartalmaz, akkor kiíródik az input promt (egy kérdőjel és egy szóköz), nullától különböző tartalom esetén az interpreter kihagyja a kiíró utasításokat. A fenti BASIC utasítással tehát félrevezetik az interpretert, amely „azt hiszi", hogy nem az alapértelmezés szerinti állománnyal dolgozik, ezért nem írja ki a kérdőjelet.

A fentiekből látható, hogy a $13-on levő rendszerváltozó szerepe nem változott meg, csak valamelyik dokumentáció írója nevezte ki a — mondhatni illegális — mellékteveékenységet fő feladattá, a többi összeállítás készítője pedig ellenőrzés nélkül átvette.

Szövegváltozók tárolása C16-on

Ha dr. Ury Lászlónak az LSI ATSZ kiadásában megjelent Commodore-tárgyú könyveiben csak ilyen lényegtelen tévedések lennének, és csak ilyen arányban, mint az előbb említett kiadványban, nem elégedetlenkednék. De sajnos . . .

A COMMODORE C16 fedélcímű könyvéből idézek néhány, a szövegváltozók tárolásával kapcsolatos részletet. A LET utasítás leírásánál olvashatjuk a 118—119. oldalon: „A C—16 BASIC a sztringeket kétféleképpen tárolja. A két tárolási mód közti különbség néha fontos lehet. . . . A változók területén a sztring neve, hossza, illetve a sztring első elemére mutató 2 byte kerül tárolásra. Amennyiben egy sztringkonstans kerül a változóba, a mutató magába a **programba mutat vissza** (helykímélés céljából). A sztringkifejezések eredményeiként előálló sztringek a BASIC munkaterületen kerülnek tárolásra." Néhány oldallal később, a LOAD utasításnál a overlay technika használatával kapcsolatosan a következő szöveg olvasható: „A betöltés a program eredeti változóit nem törli, feltéve, hogy a másodszorra betöltött program rövidebb az elsőnél. A sztringkonstansokkal végrehajtott értékadások eredményei és a függvénydefiníciók azonban mindenképpen elvesznek."

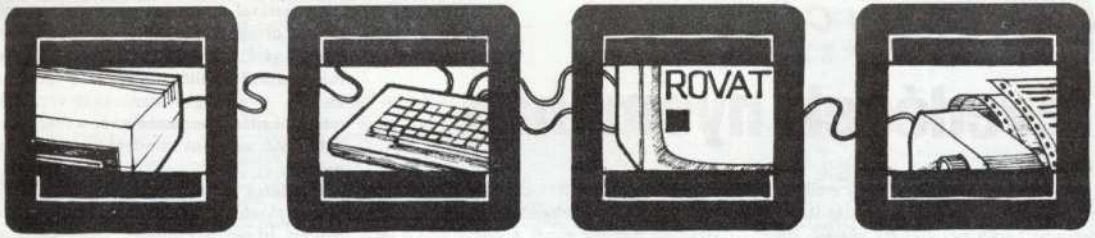
Mindezek az állítások a VC20 és C64 gépekre igazak, de a C16-on egészen más a helyzet. Itt ugyanis minden, szövegváltozónak történő értékadás hatására a változóba kerülő szöveg a BASIC munkaterületre kerül, függetlenül attól, hogy az értékadó utasítás jobb oldalán konstans vagy kifejezés áll.

Van még egy különbség a korábbi géptípusokhoz képest. A C16 minden, a sztringterületen lévő szöveg után tárol egy kétbájtos toldalékot. Ennek tartalma attól függ, hogy a szöveg egy változóhoz tartozik vagy nem. Az első esetben a két bájton egy mutató van, mely a változóban levő leíróra mutat, a hosszat tartalmazó bájtra. A második esetben, vagyis amikor a szöveg már nem tartozik sehová, mert a hozzá tartozó változó új értéket kapott, a toldalék első bájtjában a szöveg — toldalék nélküli — hossza, a másodikban $FF érték van. Ez a módszer az úgynevezett szemétgyűjtő (garbage collection) algoritmust egyszerűbbé és gyorsabbá teszi. Méréseim szerint a közel 12 kilobájtnyi munkaterület „kitakarítása" kb. 1/20 másodpercet vesz igénybe. Ezáltal feleslegessé válik az a C64-nél használt fogás, hogy sok karakterlánc-műveletet tartalmazó BASIC programokban időnként egy X = FRE(Ø) vagy hasonló utasítással kiprovokáljuk a szemétgyűjtés végrehajtását, mielőtt a „hulladék" mennyisége és az „összeszedésére" fordítandó idő kritikussá válna.

Az előbbiekből kiderül az is, hogy programátfedés (overlay) esetén a szövegkonstans nem vész el. Így feleslegessé válik a korábbi géptípusok használatánál alkalmazott trükk, melynél a szövegváltozónak való értékadásnál egy szövegkonstansból egy üres sztring hozzáadásával kifejezést csinálunk e minta szerint:

A$ = "SZOVEG" + ""

BARNA LÁSZLÓ



SZELLEMGRAFIKA

A BASIC és a gépi kódú, assembly szintű megvalósítások gyorsaságának összehasonlítására bemutatunk egy programot, melyben azonos kerületű, oldalhosszúságú két téglalap mentén mozog két, mindkét irányban kétszeresre nagyított, a teljes mezőt betöltő szellem. Az egyik színe fehér, a másiké fekete. Mozgásuk „párhuzamosan" történik, pontosabban a fekete BASIC-ből, a fehér gépi kódból vezérelt. A kezdeti, beállított értékek kb. azonos sebességet eredményeznek, szemre nem dönthető el, hogy melyiket vezérli BASIC, melyiket gépi kódú utasítás.

Ajánljuk, hogy begépelés és indítás után mindenki várja ki türelmesen a programfutás végét. A fehér szellem már futás közben megmutatja gyorsaságának egy részét. Aki ezzel nem elégszik meg, az tovább játszhat a programmal. A BASIC program 18-as sorában a 49152+14 címre írt érték növelésével fehér sebessége lassítható, az 56325 címre írt érték csökkentésével viszont gyorsítható. A 49152+14 címre írt 0 érték megváltoztatható a 104-es DATA sor bekeretezett 0 értékének átírásával is. A gépi kódú program tulajdonképpen azt a feladatot látja el, amit a BASIC program 21—28-as sorai, vagyis mozgatja a fehér szellemet a pályán. A gépi kódú DATA sorok mellett *(1. lista)* közöljük az assembler listát is *(2. lista)*.

```
0 PRINT"□■";:GOSUB100
4 REM ********************************
5 REM * A FEKETE & FEHER VERSENYE... *
6 REM * SPRITE GRAFIKAI DEMO-PROGRAM *
7 REM * (C) BY FÖLDI ENDRE 1986.4.10 *
8 REM ********************************
9 :
10 V=53248
11 FORC=0TO63:POKE11*64+C,255:NEXT
12 POKEV+32,12:POKEV+33,11
13 POKEV,24:POKEV+1,50
14 POKEV+2,65:POKEV+3,50
15 POKEV+23,3:POKEV+29,3
16 POKE2040,11:POKE2041,11
17 POKEV+21,3:POKEV+40,0
18 POKE49152+14,00:POKE56325,23
19 SYS49152
20 :
21 FORX=65TO255:POKEV+2,X:NEXTX
22 POKEV+16,2
23 FORX=0TO40:POKEV+2,X:NEXTX
24 FORY=50TO208:POKEV+3,Y:NEXTY
25 FORX=40TO0STEP-1:POKEV+2,X:NEXTX
26 POKEV+16,0
27 FORX=255TO65STEP-1:POKEV+2,X:NEXTX
28 FORY=208TO50STEP-1:POKEV+3,Y:NEXTY
29 IFS<20THENS=S+1:GOTO21
30 :
31 FORC=20TO5STEP-2:POKE56325,C
32 GOSUB35:NEXTC:END
33 FORC=CTO60STEP2:POKE56325,C
34 GOSUB35:NEXTC:END
35 FORL=0TO500:NEXT:RETURN
36 :
100 FORC=49152TO49257:READA
101 POKEC,A:NEXT:RETURN
102 :
103 DATA120,169, 13,141, 20,  3,169,192
104 DATA141, 21,  3, 88, 96,169,  0,205
105 DATA105,192,240,  6,238,105,192, 76
106 DATA 49,234,169,  0,141,105,192, 32
107 DATA 37,192, 76, 49,234,162,255,236
108 DATA  0,208,240,  4,238,  0,208, 96
109 DATA169, 54,141, 32,192, 96,160,208
110 DATA204,  1,208,240,  4,238,  1,208
111 DATA 96,169, 71,141, 32,192, 96,162
112 DATA 24,236,  0,208,240,  4,206,  0
113 DATA208, 96,169, 88,141, 32,192, 96
114 DATA160, 50,204,  1,208,240,  4,206
115 DATA  1,208, 96,169, 37,141, 32,192
116 DATA 96,  0
```

1. lista

```
C000 78          SEI
C001 A9 0D       LDA ##0D
C003 8D 14 03    STA $0314
C006 A9 C0       LDA ##C0
C008 8D 15 03    STA $0315
C00B 58          CLI
C00C 60          RTS
C00D A9 00       LDA ##00
C00F CD 69 C0    CMP $C069
C012 F0 06       BEQ $C01A
C014 EE 69 C0    INC $C069
C017 4C 31 EA    JMP $EA31
C01A A9 00       LDA ##00
C01C 8D 69 C0    STA $C069
C01F 20 25 C0    JSR $C025
C022 4C 31 EA    JMP $EA31
C025 A2 FF       LDX ##FF
C027 EC 00 D0    CPX $D000
C02A F0 04       BEQ $C030
C02C EE 00 D0    INC $D000
C02F 60          RTS
C030 A9 36       LDA ##36
C032 8D 20 C0    STA $C020
C035 60          RTS
C036 A0 D0       LDY ##D0
C038 CC 01 D0    CPY $D001
C03B F0 04       BEQ $C041
C03D EE 01 D0    INC $D001
C040 60          RTS
C041 A9 47       LDA ##47
C043 8D 20 C0    STA $C020
C046 60          RTS
C047 A2 18       LDX ##18
C049 EC 00 D0    CPX $D000
C04C F0 04       BEQ $C052
C04E CE 00 D0    DEC $D000
C051 60          RTS
C052 A9 58       LDA ##58
C054 8D 20 C0    STA $C020
C057 60          RTS
C058 A0 32       LDY ##32
C05A CC 01 D0    CPY $D001
C05D F0 04       BEQ $C063
C05F CE 01 D0    DEC $D001
C062 60          RTS
C063 A9 25       LDA ##25
C065 8D 20 C0    STA $C020
C068 60          RTS
C069 00          BRK
C06A EA          NOP
C06B EA          NOP
C06C EA          NOP
```

2. lista

A program leírása

0	Gépi kód betöltése
10	VIC kezdőcím
11	Szellem a 11-es helyre
12	Képernyőszínek
13—14	0. és 1. szellem kezdőkoordinátái
15	Mindkét szellem kétszeresre nagyítása mindkét irányban
16	Blokkmutató a 0. és 1. szellemre
17	Bekapcsolás, fekete szellem színe
18	Minden megszakítást arrébb visz, 23 a hardvermegszakítás sűrűsége
19	Új megszakítási cím beállítása fehér szellemnél
21	Fekete szellem jobbra mozgatása
22	MSB átkapcsolás
23	MSB=1 utáni mozgatás jobbra
24	Fekete szellem mozgatása le
25—27	Fekete szellem mozgatása balra
28	Fekete szellem mozgatása fel
29	Ciklus 20-szor
31	Fehér szellem felgyorsul
32	Lassítás
33—34	Fehér szellem lelassul a kezdeti értékre
35	Várakozó ciklus
100—101	Gépi betöltő rutin
103—116	Gépi kód

FÖLDI ENDRE—ÉNEKES FERENC

A PIXEL

A PIXEL rutin akkor használható eredményesen, ha különálló pontokat vagy vonalakat akarunk rajzolni. A rutin részei a ROM-ban is megtalálhatók. A PIXEL rutin ezt is használja, továbbá lehetővé teszi az alsó két (szerkesztő) sor kezelését, valamint a színek beállítását.

A rutinban az x és y értékét a PLOT utasításhoz hasonlóan határozzuk meg. A COLOUR paraméter első három bitje az INK értéket, a második három bitje a PAPER értékét, a két utolsó bit az esetleges BRIGHT és FLASH üzemmódot szabályozza.

SCHMITT PÁL

A PIXEL RUTIN

```
10
20 X       EQU 100
30 Y       EQU 100
40 PIXADD  EQU 8876
50 COLOUR  EQU 222
60
0664  70 PIXEL   LD B,Y
0E64  80         LD C,X
3E8F  90         LD A,191
CDAC22 100       CALL PIXADD
47    110        LD B,A
04    120        INC B
3E81  130        LD A,1
0F    140 ROTATE RRCA
10FD  150        DJNZ ROTATE
47    160        LD B,A
7E    170        LD A, (HL)
B0    180        OR B
77    190        LD (HL),A
7C    200 ATTR   LD A,H
0F    210        RRCA
0F    220        RRCA
0F    230        RRCA
E603  240        AND 3
F658  250        OR 88
67    260        LD H,A
36DE  270        LD (HL),COLOUR
C9    280        RET
      290
```

Hozzászólás

A Magazin 1986/5. számában megjelent *Ékezetes ábécé C64-re* című cikkel kapcsolatban szeretnék néhány megjegyzést tenni.

A szerzők elképzelése, hogy a C64-felhasználók számára „saját" definiálású magyar ékezetes kiíratásokat tegyenek lehetővé — dicséretes, örömömre szolgáló vállalkozás. A megadott programlista, majd a futtatási, belövési eljárások azonban bosszúságot — elkerülhető bosszúságot — okoztak. Helytelen ugyanis a 28. oldalon közölt BASIC-terület eltolási parancsok kiadása. Erre egy lehetséges, kipróbált verzió:

```
HB=INT(8193/256):LB=
8193-256*HB:PO
KE43,LB:POKE44,HB:NEW
majd
LOAD "1. programnév", 8
RUN
```

A korrekciós eljárás lehetővé teszi az új, ékezetes betűkészlet, ill. a „KAR"; „PRINT.SEQ" fájlok létrehozását. A felhasználó hívhatja ezeket a fájlokat, beépíti programjába, és így teljesül a szerzők által kitűzött feladat.

A másik megjegyzésem: véleményem szerint a szerzők nem vették figyelembe a C64 géphez illeszthető valamennyi printer karakterkészletét. Ezek sajnos nem azonosak. A program által létrehozott ékezetes ábécé csupán az MPS—801 nyomtatót képes „magyarul" meghajtani.

A szerzők körültekintőbb eljárással osztatlan örömet szerezhettek volna a μMagazin olvasóinak.

KELEMEN ANTAL

Szerkesztői megjegyzés: Az olvasó által megadott összefüggés nem azonos a szerzők helyesbítésében közölttel.

A MÁTRIX–64

Ez a program a matematikai programok olyan típusát képviseli, amelynek tulajdonképpen még nincs kialakult magyar elnevezése. A táblázatos adatfeldolgozó program, rovatos adatkezelő, mátrixérték-számító stb. elnevezések nagyjából körülírják a tényleges funkciókat. A legelterjedtebb az angol SPREAD SHEET megnevezés, de reméljük hamar találunk rá magyar megfelelőt.

A programok jellemzője, hogy a feldolgozandó adatokat táblázatos formában jelenítik meg. Ezáltal kitűnően alkalmazhatók különféle bonyolult kalkulációk, gazdasági számítások, valamint egyszerűbb szimulációs feladatok megoldására, illetve azok eredményeinek formázott megjelenítésére.

A Commodore 64 mikroszámítógépen több ilyen jellegű program is elterjedt, az ismertebbek a következők:

— Multiplan (más gépeken is fut, elsősorban CP/M operációs rendszer alatt)
— Busicalc
— Practicalc
— Easy Calc Result
— Calc Result

A felsoroltak közül a Calc Result nálunk is elég elterjedt, és igen népszerű a felhasználók körében. Nemrégiben került a számítástechnikai üzletekbe a magyar nyelvű változata, a MÁTRIX—64, amely megjelenésében, felépítésében és funkcióiban is teljesen megegyezik az eredeti Calc Resulttal, sőt még annak angol nyelvű rövidítései is megtalálhatóak benne.

A táblázat felépítése

A MÁTRIX—64 úgynevezett „lap"-okat kezel, amelyek 63 oszlopból és 254 sorból állnak. Az oszlopokat betű jelöli (A, B, C, . . ., X, Y, Z, AA, AB, AC, . . ., BK), a sorok pedig sorszámmal vannak ellátva (1—254), így a táblázat minden egyes rovatát egy betűből és egy számból álló koordinátapárral tudjuk azonosítani. A program 32 darab 63 × 254-es nagyságú lapot tud kezelni, tehát egy 32 × 63 × 254-es méretű, háromdimenziós mátrix kezelésére van lehetőségünk.

Az egyes rovatokba tetszés szerint írható numerikus érték, szöveg vagy aritmetikai és logikai kifejezés is. A mezők szélessége alaphelyzetben 8 karakter, ez azonban maximum 18-ig változtatható. A megadott mezőszélesség csak a kijelzés formátumára vonatkozik, az adott mezőben tárolt információ hossza elérheti a 255 karaktert is. Az éppen aktuális mezőt egy teljes mezőszélességű kurzor jelöli, amely a kurzorvezérlő billentyűkkel tetszőleges irányba léptethető.

A képernyő felső három sorában a felhasználónak szóló információk olvashatók. Itt jelennek meg a választható parancsok kezdőbetűi, a kurzor aktuális pozíciója, a még szabad memóriaterület, a számítási formátumok, az aktuális mező teljes tartalma, valamint a mező típusa (LABEL = címke, vagyis szöveges; VALUE = érték, vagyis numerikus). A képernyő többi részét az adatmezők töltik ki.

Az *1. ábrán* egy jövedelmezőségi tábla egy része látható. A mezőszélesség 10-re van állítva, és jól látható, hogy a szövegek balra, a numerikus értékek pedig jobbra rendezettek. A kurzor a C2 mezőben található.

A lap méretéből adódóan elvileg 63 × 254 = 16 002 mező áll rendelkezésünkre, ebből azonban egyszerre csak 2063 lehet kitöltve. Ennek az az oka, hogy a program minden egyes mező adatait a mező szélességétől függetlenül 8 bájton tárolja, a kitölthető mezők száma tehát a memória szűkössége miatt korlátozott. Sajnos a lemez 170 kbájtos kapacitása is szab egy bizonyos határt a felhasználónak, mindössze 4 teljesen kitöltött lap adatait képes tárolni. Ha azonban laponként csak kb. 400-450 mezőt töltünk ki — ami azért elég tekintélyes táblázatot eredményez —, akkor mind a 32 lapot igénybe tudjuk venni. A 32-es lapnak egyébként kitüntetett szerepe van. Ha az egyes lapok között műveleteket végeztetünk, például összegezzük az azonos mezők tartalmát, akkor az eredmény ezen a lapon jelenik meg.

A memóriában egyszerre mindig két lap adatai találhatók, alaphelyzetben ezek az 1-es és a 32-es lapok. Azt, hogy közülük éppen melyik legyen látható a képernyőn, az F1 funkcióbillentyű megnyomásával választhatjuk ki.

A program minden használt lap tartalmát egy ún. munkafájlban tárolja a lemezen, amely minden lemezolvasó, -író művelet alkalmával felülíródik a lap aktuális tartalmával. A végleges, kész lapot a felhasználó által adott névvel rögzíti a lemezen.

Kifejezések használata

Mint korábban már említettük, a mezőkbe nemcsak adatok, hanem matematikai és logikai kifejezések is írhatók. Íme egy példa.

Írjunk az A1 mezőbe 25-öt, a B1-be pedig A1*2+8 kifejezést! A B1 mező tartalma 58 lesz, de ha most az A1 mezőt például 4-re módosítjuk, a B1 mező tartalma is meg fog változni, mégpedig a megadott kifejezés szerint, azaz 16-ra.

Logikai kifejezést például az alábbi módon használhatunk. Egy tetszőleges mezőbe írt IF A1>2 AND B1<C2 THEN 1 ELSE 0 kifejezés az adott mező értékét az A1, B1, C2 mezők értékétől függően 0-ra vagy 1-re állítja.

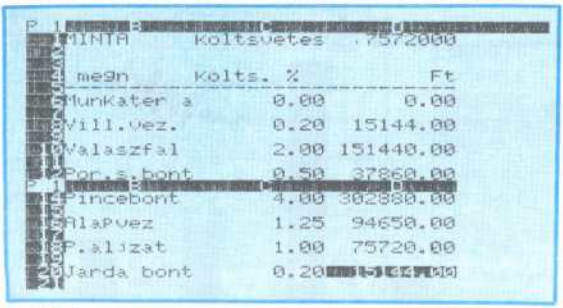
A program egyik leghasznosabb szolgáltatása, hogy lehetőség van megadott kifejezések más mezőkben való megismétlésére, abszolút és relatív értelemben egyaránt. Egyszerű módon megoldható például az, hogy a C oszlop 3-tól 25-ig terjedő összes eleme az A1+1 kifejezésnek megfelelő, ugyanazon értéket vegye fel. Ezt nevezzük abszolút áthelyezésnek. Ha azonban ugyanebben az intervallumban egyesével növekvő számsort szeretnénk kapni, azt is könnyen megoldhatjuk az A1+1 kifejezés C3-tól C25-ig tartó relatív megismétlésével.

1. ábra

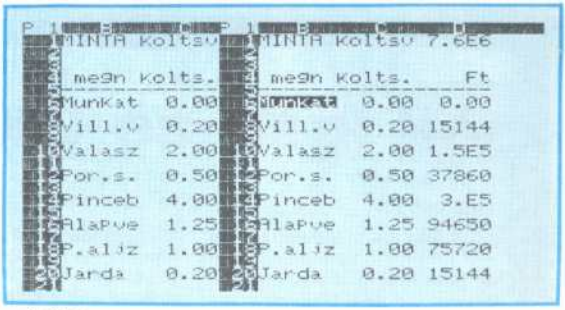
MINTA	Koltsvetes	7572000
meGn	Kolts. %	Ft
Munkater a	0.00	0.00
Vill.vez.	0.20	15144.00
Valaszfal	2.00	151440.00
Por.s.bont	0.50	37860.00
Pincebont	4.00	302880.00
AlaPvez	1.25	94650.00
P.aljzat	1.00	75720.00
Jarda bont	0.20	15144.00

2. ábra

MINTA	Koltsvetes	7572000
meGn	Kolts. %	Ft
Munkater atadas	0.00	0.00
Vill.vez. bontas	0.20	15144.00
Valaszfal bontas	2.00	151440.00
Por.s.bont	0.50	37860.00
Pincebont	4.00	302880.00
AlaPvez	1.25	94650.00
P.aljzat	1.00	75720.00
Jarda bontas	0.20	15144.00



3. ábra



4. ábra

Beépített függvények használata

A MÁTRIX—64 beépített függvényeit két csoportra oszthatjuk: az egyik csoportba a BASIC-ből megismert függvények egy része tartozik, a másik csoportot speciális függvények alkotják. Az argumentum az első esetben egy szám, egy kifejezés értéke vagy egy mező azonosítója lehet, az utóbbi esetben a függvény értéke az adott mező tartalmából számítódik. A speciális függvények argumentuma több számból, illetve mezőazonosítóból áll, ezek az adott függvény értelmezési tartományát jelölik ki. Például ATLG(A5:A10) az A5-től A10-ig található összes mező értékének átlagát számítja ki.

Matematikai függvények:

ABS( )	Abszolútérték-függvény
INT( )	Egészrészfüggvény
TORT( )	Törtrészfüggvény
EXP( )	Exponenciális függvény
LN( )	Természetes alapú logaritmus függvény
LOG10( )	Tizes alapú logaritmus függvény
GYOK( )	Négyzetgyökfüggvény
RND( )	Véletlenszám-generáló függvény
SIN( )	Szinuszfüggvény
COS( )	Koszinuszfüggvény
TAN( )	Tangensfüggvény
ARCSIN( )	Arcus szinuszfüggvény
ARCCOS( )	Arcus koszinuszfüggvény
ARCTAN( )	Arcus tangensfüggvény

Speciális függvények:

ATLG( )	Átlagfüggvény
MIN( )	Minimumérték-függvény
MAX( )	Maximumérték-függvény
SZORAS( )	Standard szórásfüggvény
SUM( )	Összeg- (summa-) függvény
BET( )	Bázisérték- (kamatoskamat-) függvény
SZAML( )	Számláló függvény (az értelmezési tartományban található VALUE típusú mezők száma

Speciális szolgáltatások

A program sok szolgáltatással segíti a felhasználót az adatmezők képernyőn és nyomtatón való megjelenítési formátumának megválasztásában. Az 1. ábrán látott nyomtatási formátum nem alkalmas a teljes táblázat nyomtatására, csak a képernyő aktuális tartalmát jeleníti meg, ezenkívül nem is szép, hiszen láthatók fölösleges információk is, mint a táblázat koordinátái, a kurzor stb. Lehetőség van olyan formázott nyomtatásra is, mint amilyen a *2. ábrán* látható. Tetszés szerint megadható a nyomtatandó oszlopok sorrendje, az egyes oszlopokon belül a mezők nagysága is, így akár a papír teljes, 80 karakter szélessége kihasználható.

Több lehetőség van arra is, hogy a táblázat különböző részeit egyszerre láthassuk a képernyőn. A *3.* és *4. ábra* a táblázat megadott helyen történő vízszintes, illetve függőleges irányú kettémetszését mutatja. A két részben külön-külön és együttesen is tudunk mozogni. Egy másik lehetőség az ún. ablak létrehozása, amelyet az *5. ábrán* láthatunk. A szétvágás és az ablak egy képernyőn kombinálva is alkalmazható, például a *6. ábra* szerint.

Arra is van lehetőség, hogy a táblázat első oszlopa állandóan a képernyőn legyen, sőt az első oszlop mezőszélessége ebben az esetben külön is definiálható, az összes többi mező 8 karakter szélességű lesz. *(7. ábra.)*

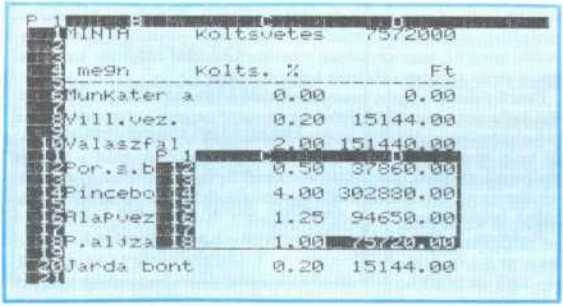
Végül még egy nagyon hasznos szolgáltatás: az adatmezők grafikus megjelenítése hisztogram formájában. Egy megadott sor vagy oszlop ábrázolható így. A *8. ábra* az 1. ábra D oszlopának egy részét jeleníti meg.

Parancsok

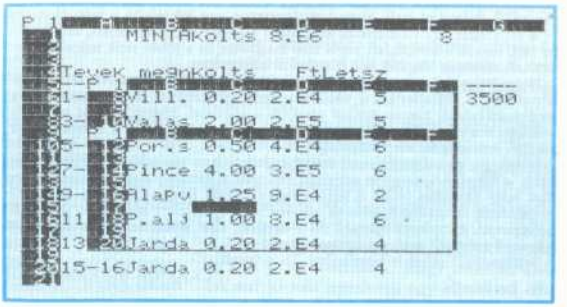
Parancs üzemmódba az F7 billentyű lenyomásával kerülhetünk. Ekkor az információsorban megjelennek a használható parancsok kezdőbetűi, valamint a funkcióbillentyűk jelentései. Ezek a következők:

F3	Adott mezőre ugrás
F5	Segítség (minden üzemmódban hívható, kilistázza a különböző parancsokat és a rövidítések jelentését)
F6	Hardcopy (másolatot készít a nyomtatóra az aktuális képernyőről)
CLEAR	A memória törlése
B	Törli az aktuális kurzorpozíció tartalmát
D	Lemezalmenü hívása
E	Editor- (szerkesztő-) almenü hívása
F	Formátumalmenü hívása
L	Osztott, ablakos vagy tabulált üzemmód törlése
O	Sor- vagy oszlopfolytonos újraszámítási sorrend kiválasztása
Q	Kilépés a MÁTRIX—64 programból (előtte automatikusan tárolja a felhasznált lapok tartalmát)
R	Automatikus vagy manuális újraszámítás kijelölése
—	Ismételt karakter (az utána leütött karakterrel feltölti a teljes mezőt)
G	Globális parancsok almenüjének hívása
P	Lapalmenü hívása

5. ábra

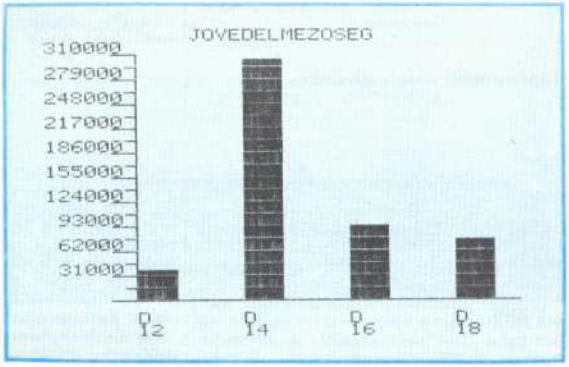


6. ábra



	MINTA	Koltsvetel	7572000
	megn	Kolts. %	Ft
1	Munkater atadas	0.00	0.00
2	Vill.vez. bontas	0.20	15144.00
10	Valaszfal bontas	2.00	151440.0
5	Por.s.bont	0.50	37860.00
40	Pincebont	4.00	302880.0
25	Alapvez	1.25	94650.00
20	P.alizat	1.00	75720.00
20	Jarda bontas	0.20	15144.00

7. ábra



8. ábra

Lemezalmenüből hívható parancsok

B	BACKUP (másolat készítése a munkalemezről)
C	CATALOG (lemez tartalomjegyzéke a képernyőre)
D	DIF-fájl létrehozása (ez egy speciális fájltípus, amely lehetővé teszi a létrehozott adatállomány más programokban — például DATAMAT-ban — való felhasználását). Jelentése: Data Interchange Format
E	ERASE (fájl törlése a lemezről)
I	Inicializálás
L	LOAD (fájl betöltése)
N	NEW (formattálás)
S	SAVE (fájl kimentése lemezre)

Editor- (szerkesztő-) almenüből hívható parancsok

C	COPY (adattartomány átmásolása másik területre)
D	DELETE (sor vagy oszlop törlése)
G	GRAPHIC (hisztogram felrajzolása)
I	INSERT (sor vagy oszlop beszúrása)
M	MOVE (adattartomány áthelyezése másik területre)
P	PRINT (adattartomány nyomtatóra küldése)
R	REPLICATE (kifejezés abszolút vagy relatív ismétlése más területen)
S	SPLIT (képernyő vízszintes vagy függőleges osztása)
T	TABULÁTOR (a bal szélső oszlop fix rögzítése)
W	WINDOW (ablak létrehozása)

Formátumalmenüből hívható parancsok

C	COLOR (kurzorszín beállítása, mezőként is lehet)
G	GLOBAL (globális formátum bekapcsolása)
M	MAXIMUM (adott mezőben maximális pontosságú számábrázolás)
I	INTEGER (ábrázolás egész számként)
$	Két tizedes pontosságú ábrázolás
L	LEFT (balra igazítás a mezőben)
R	RIGHT (jobbra igazítás a mezőben)
*	Számok helyettesítése csillaggal

Globális parancsok

C	COLUMN (oszlopszélesség beállítása)

Egy levél és egy válasz

Tisztelt Szerkesztőség!

A Mikroszámítógép Magazin április havi számában olvastam a Novotrade ügyvezető igazgatójával készített riportot. Az interjúban elmondottakkal majdnem mindenben egyetértek. Nagy örömömre szolgál, hogy a nem számítógépes szakember is úgy gondolkodik, ahogy minden adatfeldolgozó szakembernek gondolkodnia kellene.

Amikor azt olvastam a cikkben, hogy „A vállalatoknál alkalmazott felhasználói szoftvercsomagok ára nehezebben nyomható le, részben a magát makacsul tartó szemlélet miatt, hogy az olcsó szoftver nem is lehet jó", arra gondoltam, hogy ez nem teljesen így igaz. Hiszen én is felhasználó vagyok, és a programokat nem az áruk nagysága, hanem a használhatóságuk szerint válogatom, és csak azután nézem az árukat.

Néhány nap múlva, ahogy elolvastam a cikket, a következő eset történt. A SOFTWARE '86 kiállítás ideje alatt a Novotrade olcsóbban árusította a Commodore 610-es gépen futtatható MS—ADEL nevű programcsomagot. A kiállításon a Novotrade bemutató részénél megérdeklődtem a következőt. Ha egy vállalat megvásárolja az MS—ADEL-t és több Commodore 610-es géppel rendelkezik, akkor minden géphez meg kell venni a programcsomagot, vagy van-e lehetőség arra, hogy a Novotrade bizonyos összeg ellenében átmásolja annyi példányban, ahányban a vállalatunknál szükség van rá. A válasz az volt, hogy természetesen van lehetőség a másolásra és nem szükséges egy vállalatnak megvenni annyiszor egy szoftvert, ahány gépe van, hiszen a használati jogot már megvette, és a többi csak másolás kérdése. A másolás kb. 500 és 1500 forint között van.

Nagy örömmel mentem haza és dicsértem kollégáimnak a Novotrade üzletpolitikáját. Mindenkit a vállalatunknál lebeszéltem a programcsomag megvásárlásáról, hiszen sokkal olcsóbb a másolat. Ebben meg is nyugodtam egészen addig, amíg a kollégáim el nem mesélték, hogy a másolás csak a hibás lemezek kicserélésére szolgál.

Minthogy a mai napig nem értem az interjúban elmondottak ellentmondását a tényekkel, nagyon szívesen venném a Szerkesztőség segítségét abban, hogy lehetővé tenné a kérdés megvitatását Rényi Gáborral. Vagy ha helytelenek az információim és mindez nem igaz, mert kollégáim jártak el helytelenül, és rosszul vetették fel a kérdést, ahol vásárolni akartak, akkor legyenek szívesek segíteni útmutatással.

Segítségüket előre is köszönöm. Tisztelettel

KRUZSLÁK ERZSÉBET
rendszerszervező

Tisztelt Kruzslák Erzsébet!

Olvasói levelére válaszolva szeretném tájékoztatni Önt a következőkről.

A levelében említett lemezmásolási díj arra az esetre vonatkozik, amikor a megvásárolt szoftvert tartalmazó lemez vagy lemezek megsérülnek. Ilyen sérülés adódhat illegális másolási kísérlet következtében is. Egyes védelmi eljárások ugyanis másolási kísérletnél a törzslemez megsérülését is okozhatják. Természetesen ilyen esetekben a mellékelt dokumentáció felhívja erre a lehetőségre a felhasználó figyelmét.

A törzslemez tartalma sérülhet még helytelen használat, törlés, egységhiba, de még számos egyéb okból kifolyólag is.

Amennyiben valamelyik ügyfelünk ilyen problémával keres meg bennünket, részére 500,— Ft másolás díj ellenében a sérült lemezt kicseréljük. Természetesen akkor, ha a sérülés nem helytelen használatból adódik, a csere díjmentesen történik.

Egészen más a helyzet, ha egy ügyfél egy szoftvert több példányban szeretne, házon belüli használatra. Egyfelől szoftvereinket — mint a hazai és külföldi szoftverforgalmazók döntő többsége is — ún. egyprocesszoros felhasználásra forgalmazzuk. Másfelől a felhasználó oldaláról is nézve — indokolt igény az, hogy ne kelljen a szoftver vételárát annyiszor kifizetnie, ahány gépen szeretné vállalaton belül használni.

Ezen felhasználói igény ismeretében, többpéldányos vásárlásnál mennyiségi kedvezményt tudunk ügyfeleink részére biztosítani. Természetesen a kedvezmény nem sértheti a szoftver fejlesztőinek szerzői jogait, igy ezt saját bevételünk terhére biztosítjuk. Ennek mértéke függ mind a konkrét szoftvertől, mind az igényelt darabszámtól is. Így a kedvezmény összegét mindig egyedi elbírálás alapján, a felhasználóval folytatott tárgyalás után tudjuk megállapítani.

Remélem, sikerült kérdésére kielégítő választ adni. Tisztelettel

RÉNYI GÁBOR
ügyvezető igazgató
NOVOTRADE RT

F Formátum beállítása minden mezőben
R Rekalkuláció (újraszámítás)

Lapalmenüből hívható parancsok

A ADDITION (lapok összeadása a kifejezések figyelembevételével)
C COPY (lap átmásolása másik lapra)
D DELETE (lap törlése a memóriából és a lemezről)
E ERASE (lap törlése a memóriából)
G GET (lap beolvasása a lemezről)
N NEGATION (előjelek átváltoztatása a lapon)
P PUT (a háttérlap kimentése a lemezre)
R RENUMBER (lapok átszámozása)
\+ Lapok összeadása a kifejezések figyelmen kívül hagyásával.

Dokumentáció

A programhoz tartozik egy 46 oldalas, Easy Script szövegszerkesztővel írt leírás, amely összefoglalja a program használatával kapcsolatos tudnivalókat. A parancsok tényszerű felsorolásán kívül egyszerű példákat is tartalmaz, ezáltal tanítja is önmagát. Az ábrákkal, nyomtatási képekkel kiegészített dokumentácó meglévő hibái (néhány logikai és didaktikai hiba, valamint egy-két pongyola megfogalmazás) ellenére hasznos segítség lehet mindazoknak, akik szeretnének behatóbban is megismerkedni a MÁTRIX—64-gyel.

SZATMÁRI LÁSZLÓ

BASIC és gépi kód

Legutóbb a töltő- és a tárolóutasításokról volt szó, vagyis azokról, melyek a mikroprocesszor regiszterei és a memória egyes bájtjai közötti adatátvitelt végzik el. Most megismerkedünk az utasításoknak egy újabb csoportjával, néhány címzési móddal, egy hasznos gépi kódú rutinnal és ez utóbbival kapcsolatosan a BASIC betöltőprogram módosított változatával.

Adatátvitel az A regiszter és az indexregiszterek között

Négy utasítás tartozik ebbe a csoportba: a TAX, TAY, TXA és TYA. Nevük az angol transfer (átvitel) szó kezdőbetűjéből és az adatmozgatásban részt vevő regiszterek nevéből áll. A TAX és a TAY az A regiszterből viszik át annak tartalmát az indexregiszterekbe, a TXA és TYA pedig az idexregiszterek tartalmát viszik az A-ba. Annak a regiszternek a tartalma, amelyből az átvitel történik, nem változik.

Mind a négy utasítás a töltőutasításokkal megegyező módon állítja be az N és a Z feltételbiteket. Erről az előző részben volt szó.

A címzési módok

A címzési módok rövid leírása a segédletben megtalálható. Ebben a részben azokról a címzési módokról írok részletesebben, amelyek a mostani példaprogramban is előfordulnak.

Az egyes címzési módokat az assembly nyelvű forráslistán az operandus megjelenési formájáról ismerhetjük fel, a gépi kódnál pedig a művelet kódjából állapíthatjuk meg, hogy melyik címzési módról van szó.

Abszolút címzés

Ezzel a címzési móddal már a feltétel nélküli ugróutasításoknál is találkoztunk. Az operandus kétbájtos (négy hexadecimális számjegy), és a műveletben részt vevő memóriabájt címét tartalmazza. Ezen a módon 64 kilobájtos belső tár akármelyik bájtja megcímezhető vele.

Nullalapos címzés

Az abszolút címzéshez hasonló, de mivel operandusa csak egybájtos, a tárnak csupán 256 bájtnyi területét, a nulladik lapot (a $ØØ . . . $FF közötti tartományt) lehet vele elérni. Néha — például töltő- és tárolóutasítások esetében, indokolatlanul — abszolút címzéssel helyettesítik. Néhány utasításnak nincs is nullalapos címzési módja; a VC20 és C64 gépek BASIC ROM-jában gyakori JSR $ØØ73 sem helyettesíthető nullalapos változattal.

Használatát az abszolút címzéssel szemben a tömörebb kód és a gyorsabb végrehajtás indokolja.

Közvetlen címzés

Az előbbiektől eltérően az operandus nem a műveletben részt vevő adat címét tartalmazza, hanem magát azt az adatbájtot, mellyel a műveletvégzés történik. Az assembly listán az operandus első karaktere egy # jel.

Ez a címzési mód minden töltőutasításnál használható, viszont egyik tárolóutasításnál sem. Érdemes utánagondolni, hogy miért.

Implicit címzés

Ennél a címzési módnál az utasítás kódját nem követi operandus. Ennek két indoka lehet:

1. Az utasításokban nem vesz részt memóriabájt, mint például a regiszterek közötti adatátvitel esetében.

2. A processzor az utasítással kapcsolatos memóriacím(ek)et az utasítástól függetlenül kezeli. Például az RTS-nél a szubrutinból való visszatérés címét a veremmemóriának az SP regiszter által megcímzett bájtpárja tartalmazza. Itt két cím is szerepel — a veremcím és a veremben található visszatérítési cím —, de ezekkel az RTS utasítás programozásánál nem kell törődnünk.

A programokról

A most közölt BASIC betöltőprogramok egy olyan gépi kódú rutint töltenek a $Ø2C8 . . . $Ø2FF területre, mellyel a memória egy meghatározott részének a tartalmát lehet mágneslemezre vinni. Saját gépi kódú alprogramjainkat is ezzel menthetjük a háttértárolóra. Az *1. listán* a C64-re írt változat BASIC nyelvű betöltőprogramja és assembly listája látható, a *3.* és *4. listán* a VC20-as változaté. Az assembly listáról leolvasható, hogy a két program csak a hívott ROM-rutinok belépési pontjainak címében különbözik.

Aki mágnesszalagos háttértárral dolgozik, a BASIC program beírásakor a 728-as sorban lévő DATA utasítás két adatát módosítsa: a harmadik helyre 8 helyett 1-et, az utolsó helyen 982 helyett 975-öt kell bevinni.

A betöltőprogramok abban különböznek az eddig használtaktól, hogy minden DATA sor ki van egészítve egy ellenőrző ösz-

```
100 REM GEPI KODU MENTES (C64)
110 FOR I=712 TO 760 STEP 8
120 S=0
130 FOR J=0 TO 7
140 READ A : POKE I+J,A : S=S+A
150 NEXT
160 READ A : IF A=S THEN 190
170 PRINT "ADATHIBA A" I "SZAMU SORBAN"
180 STOP
190 NEXT
200 PRINT "SAVE RUTIN BETOLTVE" : NEW
712 DATA 32,253,174,32,158,173,32,130,984
720 DATA 183,166,34,164,35,32,189,255,1058
728 DATA 162,8,169,2,168,32,186,255,982
736 DATA 32,247,2,165,20,133,251,165,1015
744 DATA 21,133,252,32,247,2,166,20,873
752 DATA 164,21,169,251,76,216,255,32,1184
760 DATA 253,174,32,158,173,76,247,183,1296
READY.
```

1. lista

```
100 REM GEPI KODU MENTES (VC-20)
110 FOR I=712 TO 760 STEP 8
120 S=0
130 FOR J=0 TO 7
140 READ A : POKE I+J,A : S=S+A
150 NEXT
160 READ A : IF A=S THEN 190
170 PRINT "ADATHIBA A" I "SZAMU SORBAN"
180 STOP
190 NEXT
200 PRINT "SAVE RUTIN BETOLTVE" : NEW
712 DATA 32,253,206,32,158,205,32,130,1048
720 DATA 215,166,34,164,35,32,189,255,1090
728 DATA 162,8,169,2,168,32,186,255,982
736 DATA 32,247,2,165,20,133,251,165,1015
744 DATA 21,133,252,32,247,2,166,20,873
752 DATA 164,21,169,251,76,216,255,32,1184
760 DATA 253,206,32,158,205,76,247,215,1392
READY.
```

2. lista

```
02c8  20fdae    jsr $aefd
02cb  209ead    jsr $ad9e
02ce  2082b7    jsr $b782
02d1  a622      ldx $22
02d3  a423      ldy $23
02d5  20bdff    jsr $ffbd
02d8  a208      ldx #$08
02da  a902      lda #$02
02dc  a8        tay
02dd  20baff    jsr $ffba
02e0  20f702    jsr $02f7
02e3  a514      lda $14
02e5  85fb      sta $fb
02e7  a515      lda $15
02e9  85fc      sta $fc
02eb  20f702    jsr $02f7
02ee  a614      ldx $14
02f0  a415      ldy $15
02f2  a9fb      lda #$fb
02f4  4cd8ff    jmp $ffd8
02f7  20fdae    jsr $aefd
02fa  209ead    jsr $ad9e
02fd  4cf7b7    jmp $b7f7
```

3. lista

```
02c8  20fdce    jsr $cefd
02cb  209ecd    jsr $cd9e
02ce  2082d7    jsr $d782
02d1  a622      ldx $22
02d3  a423      ldy $23
02d5  20bdff    jsr $ffbd
02d8  a208      ldx #$08
02da  a902      lda #$02
02dc  a8        tay
02dd  20baff    jsr $ffba
02e0  20f702    jsr $02f7
02e3  a514      lda $14
02e5  85fb      sta $fb
02e7  a515      lda $15
02e9  85fc      sta $fc
02eb  20f702    jsr $02f7
02ee  a614      ldx $14
02f0  a415      ldy $15
02f2  a9fb      lda #$fb
02f4  4cd8ff    jmp $ffd8
02f7  20fdce    jsr $cefd
02fa  209ecd    jsr $cd9e
02fd  4cf7d7    jmp $d7f7
```

4. lista

szeggel. Hibás bevitel esetén nem kell az egész DATA részt átböngészni, mert a program automatikusan kiírja annak a sornak a számát, amelyikben az első hibát találta. A hiba javítása után a programot újra kell indítani. Futtatás előtt nem árt a betöltőprogramot a háttértárolóra menteni, mert sikeres futás után törli önmagát.

A rutin C16-os változata azért nem szerepel, mert a C16 beépített monitorának S funkciója kis eltéréssel ugyanezt a feladatot hajtja végre.

A gépi kódú rutin működését a következő részben fogjuk elemezni. Most azt nézzük meg, hogy hogyan kell használni és miként lehet az ezzel kimentett gépi kódú programokat a háttértárolóról visszatölteni.

A gépi kódú rutin használata

A betöltő sikeres lefutása után az új rutin máris használható. Hívásának formája: SYS 712, "programnév", kezdőcím, végcím+1. A következő utasítással magát a rutint menthetjük ki SAVE névvel: SYS 712, "SAVE", 712, 768.

Mágneslemezről mindkét géptípusnál á LOAD "programnév", 8,1 utasítással lehet a kimentett programrészt az eredeti helyére a tárba tölteni. Mágnesszalagos tároló használata esetén a C64-en a LOAD "programnév", 1,1 utasítás hatására töltődik a program helyére. VC20-nál a mágnesszalagról való betöltés egy kicsit bonyolultabb. A szalagot a program kezdete elé pozicionáljuk, és sorra kiadjuk a következő BASIC utasításokat:

POKE 183,0 : POKE 147,0 : SYS 62929

BARNA LÁSZLÓ

Azokhoz szólnak ezek a programok, akik a FORTH alapjait már értik, és szívesen látnak példákat, vagy éppen közvetlenül fel tudják őket használni. Néhány megoldást mutatok rekordszerkezetre, pontosabban: rekordszerkezetet létrehozó definiáló szóra. Az alkalmazott nyelvjárás: FIG—FORTH 1.1

FORTH rekordszerkezetek

A rekord különböző hosszúságú adatokból — mezőkből — álló adatszerkezet. A rekord mezőit néha együtt szeretjük kezelni — általában például rekordonként írunk, olvasunk —, ezért összefüggő memóriaterületen, egymás mögött helyezzük el őket. Viszont csak külön-külön tartalmaznak értelmezhető információt, így többnyire mégis az egyes mezőkre és nem a teljes rekordra hivatkozunk.

Egy FORTH szótól, mellyel rekordokat akarunk kezelni, azt várjuk, hogy megkaphassuk tőle a mezők címét (az ilyen szavakat rövidebben „mezőcímszó"-nak fogjuk becézni). Két különböző szerkezetű rekordhoz tartozó mezőcímszó működése egyforma lehet, csak a rekord felépítését leíró adatok — a rekord kezdőcíme, mezőinek száma, hossza — különböznek bennük. Ez azt sugallja, hogy ne bíbelődjünk külön minden mezőcímszóval, hanem írjuk meg azt a definiáló szót, amelynek csak megadjuk a rekordszerkezet adatait, és létrehozza a testre szabott mezőcímszót.

Legyen az első ilyen definiáló szó neve REK. A vele definiált szavak azonkívül, hogy előállítják a rekordmezők címét, tárterületként is szolgálnak a rekord számára. A rekord a rekordkezelő szó „hasában" — paramétermezőjében — lesz.

A rekord szerkezetét úgy adjuk meg a REK-nek, hogy a mezőhosszakat a veremre tesszük, mégpedig fordított sorrendben: az első mező hossza lesz felül, tehát azt adjuk meg utoljára. A verem tetejére a mezők számát tesszük. A REK így a következő elemeket fogyasztja el a veremről:

$mezőhossz_{msz}$ $mezőhossz_{msz-1}$ . . . $mezőhossz_1$ msz, ahol msz a mezők száma.

Akit nagyon zavar a mezőhosszok fordított sorrendje, adja meg őket az eredeti sorrendben (a mezők száma legfelül marad) a REK definíciójába, a <BUILDS utánra pedig szúrja be az alábbi programrészletet:

```
>R HERE R 2* + HERE DO I! 2 +LOOP
  HERE R 2* + HERE DO I 2 +LOOP R>
```

A definiált mezőcímszó a veremre tett mezősorszámból adja a mező címét. Ha a mezősorszám értelmezhetetlen, a program hibajelzéssel „elszáll". Különben igen vad eseményekre vezethetne egy hibásan megadott mezősorszám. A „rekord" veremhatása tehát: (mezősorszám — mezőcím).

Képzeljünk el például egy lelkes könyvbarátot, aki szívesen ad kölcsön könyveket. Egy-egy kölcsönzésről a következőket jegyzi fel:

a könyv írója, címe (60 karakter)
a kölcsönvevő neve, címe (80 karakter)
a kölcsönzés éve, hónapja, napja (3 × 2 azaz 6 karakter)

Az ő adatainak így definiálnánk a REK-kel rekordot:

6 80 60 (mezőhosszok) 3 (mezők száma) REK KOLCS. Az így létrehozott KOLCS belsejében lesznek a rekord adatai. A KOLCS működése:

1 KOLCS a rekord kezdőcímét,
2 KOLCS a kezdőcímnél 60-nal nagyobb számot,
3 KOLCS a kezdőcímnél 140-nel nagyobb számot teszi a veremre.
Bármi egyebet talál a KOLCS a vermen, hibajelzéssel elszáll.

A REK-nek megadottakat a KOLCS vagy más mezőcímszó használja fel címszámításra. Ez úgy lehet, hogy a szükséges adatokat a REK a definiálandó mezőcímszó paramétermezőjébe írja.

A paramétermező fontos fogalom, sokszor fogom leírni. Nekem is, az olvasónak is kényelmesebb, ha a nevét teljesen önkényesen megkurtítom. Ezentúl „paramző"-nek fogom hívni. Több, mint felével rövidebb!

Érdemes meggondolni, hogy a mezőcímszavak paramzője milyen adatokat tartalmazzon, és melyiket hol. Egy lehetséges elrendezés:

1 bájton a mezők száma,
mezőnként 2 bájton a mező relatív címe a rekord elejéhez képest, azaz a mezőcím és a rekord kezdőcímének különbségei. (Relatív cím a rekord elejéhez képest: ez lenne a cím, ha a rekord a 0 címen kezdődne. A tényleges [abszolút] cím a relatív cím és a rekord kezdőcímének összege.) Ezután a rekord adatainak fenntartott terület (hossza a mezőhosszok összege).

Így a paramző szinte minden szükségeset tartalmaz, sőt egy feleslegeset is. Az első mező relatív címe ugyanis valahogy mindig 0 lesz, ezt nem kell tárolnunk. Ha viszont a relatív címeket „előrecsúsztatnánk", azaz az első helyen a második mező relatív címét tartanánk, a második helyen a harmadikét stb., akkor a legutolsó helyre az „utolsó utáni" relatív cím kerülne, ami, ha kicsit utánagondolunk, nem más, mint a rekord hossza. Végül tehát a következőképpen építjük fel a paramzőt:

1 bájton a mezők száma,
mezőnként 2 bájton a következő mező relatív címe, a rekord adatainak fenntartott terület.

Példánkban ilyen lenne a KOLCS paramzője:

mezők száma | 2. relatív cím | 3. relatív cím | rekord-hossz

Írjuk meg a programnak azt a részét, amely a mezőhosszok és a mezők száma felhasználásával kialakítja a paramzőt! A szó neve LETESZ lesz, veremhatása: ($mezőhossz_{msz}$ . . . $mezőhossz_1$ msz — rekordhossz), ahol msz-szel a mezők számát jelöljük. A rekordhossz megegyezik az utolsó relatív címmel, amelyet a paramzőbe írunk. Megőrizzük a vermen, mert jól jön majd, mikor helyet foglalunk a rekord adatainak. A LETESZ az adatokat a szótár „tetejére" írja a, és C, szavakkal.

```
: LETESZ (mezőhossz_msz . . . mezőhossz_1 msz — rekordhossz)
    DUP C,          (letesszük a mezők számát)
    0               (ez lesz mindig az éppen kiszámított relatív cím)
    SWAP 0 DO       (annyiszor ismételünk, ahány mező van)
                    (a vermen: a még el nem használt mezőhosszok és az)
                    (adott mező relatív címe)
    +               (új relatív cím: a régi + a következő mezőhossz)
    DUP,            (az egyik példányt a szótárba írjuk, a másik)
                    (kell a ciklus folytatásához)
    LOOP
;
```

Hogyan találjuk meg a relatív címet, ha a paramző címét tudjuk? Ha x a paramző címe, akkor az

x címen találjuk a mezők számát,
x + 1 címen a második mező relatív címét (ha van második mező),
x + 3 címen a harmadikét (ha van harmadik mező),
.
.
.
x + 2* (k − 1) − 1 címen a k-adikét, ha 2 <= k <= mezők száma.

Az első mező relatív címe egyszerűen 0.

Megírjuk azt a programot, amelyik a paramző kezdőcíméből és a mezősorszámból a fentiek alapján előállítja a mező relatív címét, értelmezhetetlen mezősorszám esetén pedig „kiszáll".

```
: RELCI (pmcím msorsz — pmcím relcím)
        (először ellenőrizzük a mezősorszámot)
    DUP 1< >R (msorsz < 1 ? A flaget „félretesszük")
    OVER C @  (elővesszük a paramzőből a mezők számát)
    OVER<     (mezők száma < msorsz?)
    R> OR     (a két flagből egy hibaflag)
```

```
    IF. "hibás mezősorszám" QUIT ENDIF
      (ide csak akkor jutunk el, ha a mezősorszám jó.)
      (A vermen: pmcím msorsz)
          1– DUP IF (a relatív címet csak akkor kell a
           memóriában keresnünk, ha msorsz≠1)
            2* 1– OVER+ (a relatív cím helye)
            @
    ENDIF
          (az ELSE-ág üres: ha a msorsz 1 volt, akkor a ver-
          men)
          (@relatív cím helyén éppen 0 van)
;
```

A nehezén túlvagyunk, nekifuthatunk a REK-nek:

```
: REK
      <BUILDS (hossz_msz . . .hossz_1 msz —)
          (itt jön, ami a „rekord" létrehozásánál történni fog)
           LETESZ (beírtuk a paramzőbe a mezők számát)
                  (és a relatív címeket. A vermen a rekord-
                  hossz.)
           ALLOT  (lefoglaljuk a rekord adatainak helyét)
    DOES> (így működnek majd a „rekordok" avagy „mező-
          címszavak")
          (a vermen: msorsz pmcím)
          SWAP RELCI (a vermen: pmcím relcím)
          (a mező relatív címét tudjuk, most kiszámítjuk a)
          (rekord kezdőcímét)
          SWAP DUP C @ (a verem tetején a mezők szá-
                       ma)
          2*           (ennyi bájtot foglalnak a rela-
                       tív címek)
          1+           (és még egyet a mezők száma)
          +            (hozzáadjuk a paramző kezdő-
                       címéhez)
                       (ezzel megvan a rekord kezdő-
                       címe)
          +            (hozzáadjuk a relatív kezdőcí-
                       met)
;                      (megvan a mezőcím).
```

Ezzel megoldottuk, hogy rekordunkat mezőnként elérjük. Hogyan hivatkozhatnák az egész rekordra?

Írunk egy szót, amely a veremre teszi a rekord kezdőcímét és hosszát. A szó neve TELJES. A rekordot a TELJES szó után, a nevével adjuk meg, például így:

```
TELJES KOLCS
```

Ha mondjuk egy rekord a virtuális memória 10. blokkjának elején kezdődik, akkor így csalhatjuk be a KOLCS adatterületére (hogy ott az egyes mezőkkel dolgozhassunk)

```
10 BLOCK TELJES KOLCS CMOVE
```

Lássuk a TELJES forrásszövegét:

```
: TELJES (— kezdőcím hossz)
    —FIND (elolvassuk a TELJES utáni szót és megpróbáljuk)
          (megkeresni a szótárban)
  IF      (megtaláltuk)
      DROP (eldobjuk a hosszúságbájtot)
      CFA  (a paramzőcímből előállítjuk a kódmezőcímet)
      1 SWAP EXECUTE
           (a kódmezőcím segítségével „lefuttatjuk")
           (a szót, mégpedig úgy, hogy 1-et adunk neki a
           vermen)
           (Az első mező címét, azaz a rekord kezdőcímét
           kapjuk.)
           (Azt is tudjuk, hogy a rekord előtt közvetlenül,
           a)
           (rekord kezdőcíménél 2-vel kisebb címen van a
           rekordhossz)
           DUP 2– @
      ELDE .„nincs ilyen rekord QUIT
      ENDIF
;
```

Elképzelhető, hogy valaki nem akarja a rekordot a mezőcímszó paramzőjébe ki-be mozgatni, ott szeretné elérni az egyes mezőket, ahol valamiért úgyis vannak, például a virtuális memóriában. Írjunk ezeknek a felhasználóknak egy REK2 definiáló szót, amely egy bármely memóriaterületet tagolni tudó sablon! A REK2-vel létrehozott rekordok paramzője nem tartalmazza a rekordot, csak egy rá mutató pointert. Definiáláskor a mezőhosszok és a mezőszám tetejébe ezt a pointert is meg kell adnunk a vermen.

A REK2-vel definiált mezőcímszavak szintén a mező címét adják a sorszám alapján.

A REK2-vel definiált szavak paramzője:
2 bájton a rekord kezdőcíme,
1 bájton a mezők száma,
mezőnként 2 bájton a következő mező relatív címe a rekord elejéhez képest.

Ha például a könyvbarát a szótárban egy külön erre való területen akarja a rekordjait tartani, akkor ír egy területdefiniáló szót, amely definiáláskor megadott számú bájtot foglal a szótárban, a definiált szavak pedig a veremre teszik a lefoglalt terület címét:

```
: AREA  <BUILDS (hosszúság —) ALLOT
        DOES> (— területcím) ;
```

létrehozza magának a területet: 146 AREA KOLCS2—ADAT majd a rekordot, amely ezt a területet mezőnként elérhetővé teszi:

```
6 80 60 (mezőhosszak) 3 (mezők száma) KOLCS2—ADAT
```

REK2 KOLCS2. A REK2 nem sokban tér el a REK-től:

```
: REK2
  <BUILDS (hossz_msz . . .hossz_1 msz rekordcím —)
  (beírjuk a paramzőbe a rekordcímet)
  (innen nagyrészt ugyanazt kell tennünk, mint az előbb)
       LETESZ
       DROP (csak nem kell lefoglalni az adatok helyét)
 DOES> (mezősorsz — mezőcím)
       (a vermen: mezősorsz pmcím)
        2+ (ha átlépjük a paramző két első bájtját,)
       SWAP RELCI (akkor ugyanúgy vannak elrendezve
                  az adatok)
                  (mint az előbb, tehát ugyanúgy kapjuk
                  a mező)
                  (relatív címét)
       SWAP 2– @  (a rekord kezdőcíme viszont a paramé-
                  ter-)
                  (mező elején van)
          +       (kezdőcím+relatív cím)
;
```

Hogyan tehetjük át a REK2-vel készített sablont egy másik memóriaterületre? Írunk rá egy szót, a neve: RECIM. Az új kezdőcímet a vermen, a rekordot pedig a RECIM után, a nevével adjuk meg.

Ha például az imént KOLCS2-vel a virtuális memória 10. blokkjának elején elhelyezett rekordot akarjuk mezőnként látni, ezt kell tennünk:

```
10 BLOCK RECIM KOLCS2
: RECIM (új kezdőcím —)
     —FIND (megkeressük a szótárban a szót)
     IF    (megtaláltuk)
        DROP (el a hosszúságbájttal)
        2+   (itt kezdődnek az adataink; a paramző első
             szava)
             (a FORTH adminisztrációs céljait szolgálja)
        !    (beírjuk a paramző elejére a megadott címet)
     ELSE. „nincs ilyen rekord QUIT
     ENDIF
```

Folytatás a következő számban.

ADORJÁN NOÉMI

Z80 programozási gyakorlatok 4.

Az előző részekben 16 bites aritmetikáról volt szó. Most nézzünk meg néhány, karakterek kezelésével kapcsolatos problémát! Ilyen feladatok nagyon gyakran előfordulnak a különböző fordítók és assemblerek írásakor, de hasonló problémával szembetalálkozhat bárki már egy BASIC programátsorszámozó írásakor is. Legyen ez az első feladat!

4.1. Írjunk olyan szubrutint, amely a memóriában levő BASIC program sorszámait egyesével átsorszámozza. A GOTO, GOSUB és egyéb, a sorszámokra hivatkozó utasításokat hagyjuk figyelmen kívül! Egy sor szerkezete a Microsoft-BASIC változatoknál:

2 bájt: sorszám MSB, LSB sorrendben
(MSB: Most Significant Byte = nagyobb helyiértékű bájt;
LSB: Least Significant Byte = kisebb helyiértékű bájt angol nyelvű rövidítése)
2 bájt: a programsor hossza n
a programsor n bájt.

A program kezdetére mutasson egy PROG nevű rendszerváltozó, és tartson addig a program, amíg a sorszám kisebb mint 9999.

A futás során a BC regiszterpár tartalmazza a sorszámok új értékeit, HL mutat az éppen vizsgált sor elejére. Először mindig a régi sorszámot töltjük DE-be, hogy megvizsgáljuk, nem nagyobb-e, mint 9999. Ha nem nagyobb, akkor betesszük a helyére a BC értékét, majd DE-be a sor hosszát töltjük. Ekkor HL a sor első bájtjára mutat, a sorszám és a hosszmutató utánra. Ehhez hozzáadva a sor hosszát, HL a következő sor sorszámának első bájtjára mutat. *(4.1. program)*

A következő példa része lehet egy assembler editorának. Az assembly sorok beírásakor nem követelheti meg az assembler a felhasználótól, hogy a címke és a mnemonik, illetve a mnemonik és az operandus közé pontosan egy szóközt tegyen. Legalább egyet viszont megkövetelhet, hiszen szükség van valamilyen elválasztó karakterre. A felhasználó beírhat két szó közé akárhány szóközt, a programsorban ezeket tárolni azonban helypazarlás. Az assembly programok listázása amúgy is kötött formátumú, azaz a mnemonik, az operandus az adott oszlopba kerül a ténylegesen beírt szóközök számától függetlenül, ezért értelme sem lenne, ha a sorban az editor bent hagyná a szóközöket.

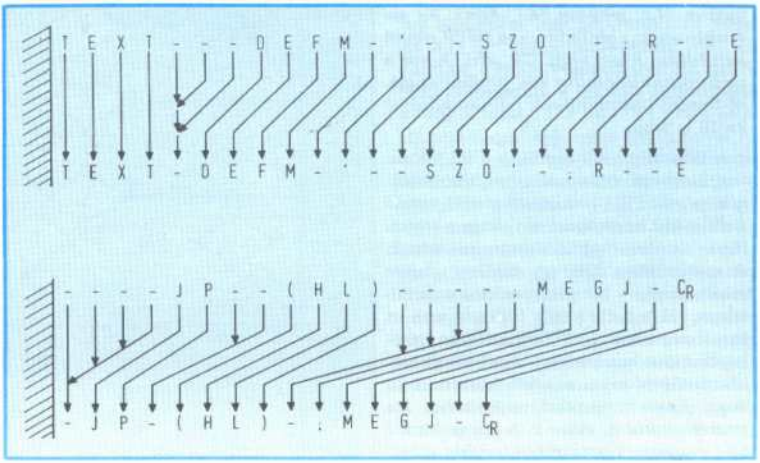
4.2. Írjunk olyan szubrutint, amely egy assembly sorból kiszedi a felesleges szóközöket, azaz bármely két szó között egy szóközt hagy! A sorban aposztrófok vagy idézőjelek között szereplő szövegkonstansokból nem szed ki egyetlen szóközt sem, hiszen itt ezeknek szerepük van, és nem szedi ki a szóközöket a sor végén álló, pontosvesszővel elválasztott megjegyzésből sem. A sor kezdődjön az INPB (input buffer) címen, és tartson a lezáró kocsi vissza (CR = 13) karakterig. *(4.2. program)*

A feladat megoldására elsőnek adódó algoritmus az, hogy elkezdünk lépkedni a programsoron, és ahol egynél több szóközt találunk, ott a sor végét egy LDIR utasítással lejjebb hozzuk. Járható út, de az a hátránya, hogy valahányszor lejjebb hozzuk a sor végét, meg kell keresnünk a sort lezáró CR karaktert, vagy folyamatosan regisztrálnunk kell, hogy meddig tart a sor. Ha ez mind megvan, akkor még mindig elgondolkozhatunk azon, hogy például a sor végén álló CR karaktert hányszor raktuk arrébb.

Próbáljuk meg inkább úgy megírni a programot, hogy az a futása során átmásolja a sort egy másik pufferbe, de ahol több szóközt talál, azok közül mindig csak egyet másol át. Pontosabban: a futás során a forrásszöveg aktuális karakterére mutat egy mutató, amit minden egyes karakter átmásolása után növelünk eggyel, és a fogadó puffer első szabad karakterére is mutat egy mutató, amelyet szintén minden karakter átmásolása után növelünk eggyel, kivéve, ha az átmásolt karakter és az előzőleg átmásolt karakter is szóköz volt. Így ha valahol egymás után két szóköz van, akkor az első átmásolódik, majd a mutató rááll a következő karakter helyére. A második szóköz is átkerül a fogadó pufferbe, de a mutató ekkor nem nő, és így a következő karakter felülírja a második szóközt. Innen már csak egy lépés a végső megoldás.

Azt kell még végiggondolnunk, hogy a fogadó és a forráspuffer lehet ugyanaz. Amíg nem találunk felesleges szóközt, addig a karaktereket az eredeti helyükre teszszük vissza, és mihelyt találunk felesleges szóközt, a célmutató lemarad a forrásmutató mögött, így nem fordulhat elő, hogy még át nem másolt karakterre töltünk rá. Ez a módszer általában csak akkor használható, ha a végső formátum rövidebb, mint az eredeti.

A programban a forrás- és célmutató a HL és a DE regiszterpár. A ciklusban annak jelzésére, hogy az előző karakter szóköz volt-e, a zéró jelzőbitet használjuk, ezért a ciklusba való belépés előtt ezt törölni kell. Ha idézőjelet vagy aposztrófot találunk, akkor a következő idézőjelig, illetve aposztrófig előrefut a program. Amikor ezt megtalálta, akkor visszatér a fő ciklusba, de ezelőtt a zéró jelzőbitet törölni kell, különben az esetleg ezután jövő szóközöket teljes



4.1. ábra

egészében kitörölné a program. A zéró jelzőbit törlésére azért alkalmas az OR H utasítás, mert HL a bemeneti pufferbe mutat, és így H nem nulla, hacsak a puffer nem a nulladik lapon van. A Z80-as rendszerekben a memória kis címein általában ROM van, ezért nem valószínű, hogy H nulla legyen. (A CP/M operációs rendszerben is különleges jelentősége van a nulladik lapnak, és bár ott is RAM van, program csak 100h felett szokott futni.) Amennyiben mégis a nulladik lapon van a puffer, úgy valamilyen más utasítást kell a zéró jelzőbit törlésére használni.

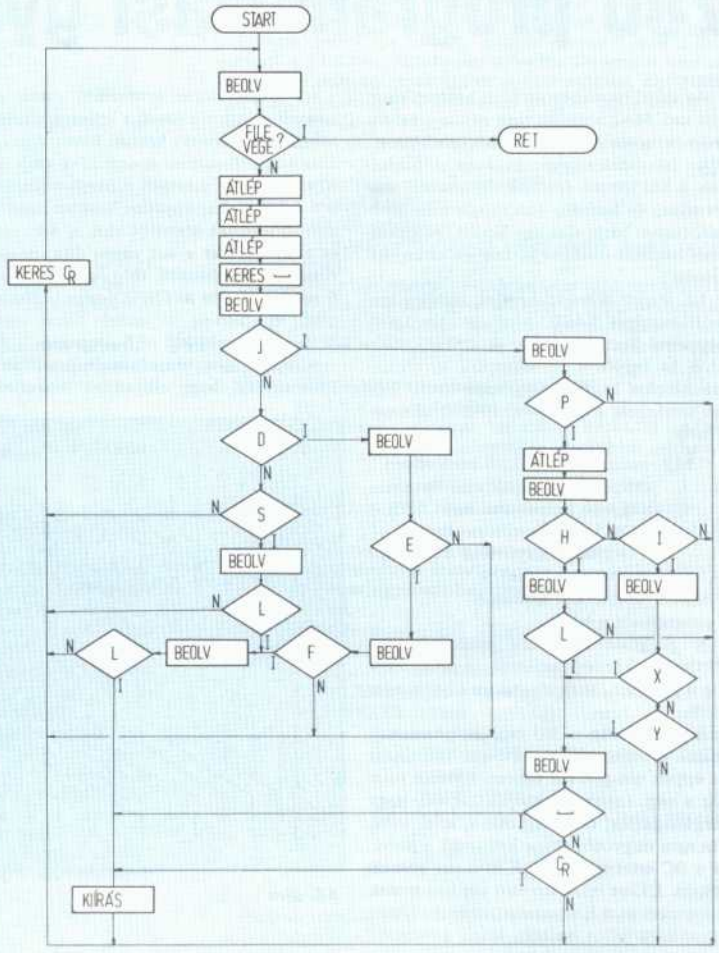
A harmadik feladat tulajdonképpen a második megfordítása. Amikor az editor listázza a sorokat, akkor a szóközöknek megfelelően tabulálja a sort, például úgy, hogy a címkék kerülnek az első hat oszlopba, a mnemonikok a 8—11. oszlopokba, az operandusok pedig a 13. oszloptól kezdődnek.

4.3. Írjunk szubrutint, amely egy assembly programsort kiír a képernyőre, tabulálva *(4.3. program, 4.1. ábra)*. Az assembly sor egy pufferben van INPB címtől kezdődően, a sor végét CR jelzi. A sor a memóriában tömörítve van, például az előző feladat szubrutinjának felhasználásával került a tárba.

A kiíratáshoz felhasználjuk a PRNA rutint, amely az akkumulátorban levő értéknek, mint ASCII kódnak megfelelő karaktert írja ki a képernyőre úgy, hogy a regiszterek értékei változatlanok maradnak. A szubrutinban DE egy táblázat elejére mutat, amely a tabulátorpozíciókat tartalmazza, HL mindig a még éppen ki nem írt karakterre mutat, és a C regiszterben számláljuk a kiírt karaktereket. Amikor egy szóközt találunk, akkor annyi szóközt írunk ki, hogy eljussunk a tabulátorpozícióig. Ha ezen túljutottunk, akkor csak egy szóközt ír ki a program. Tekintve az assembly szövegek szintaktikáját, nem okoz bajt, hogy az idézőjeleket és aposztrofokat nem vesszük figyelembe, hiszen ezek csak az utasítás után, az operandusban fordulhatnak elő, azaz az utolsó tabulátorpozíció után, és így pontosan egy szóközt ír ki a program, valahányszor egy szóközt talál.

A táblázat végét nulla jelzi, és amikor erre mutat a DE, akkor a fő ciklusba való visszatérés előtt DE-t csökkentjük eggyel, így a táblázat utolsó értékén nem lép túl. Próbáljuk úgy megírni a programot, hogy a megjegyzéseket, ha nem a sor elején kezdődnek, egységesen például a 20. pozícióra helyezze.

4.4. A memóriában a PROG címtől kezdve van egy assembly program a következő formában: minden programsor első két bájtja a programsor sorszáma BCD formátumban. A címkét, a mnemonikot és az operandust egy-egy szóköz előzi meg. Minden sor végét kocsi vissza (CR = 13) jelzi, a fájl végén pedig EOF = 255 található.



4.2. ábra

Írassuk ki azokat a sorokat, amelyekben DEFL direktíva, SLL mnemonik, vagy zárójel nélküli JP HL, JP IX, JP IY szerepel! *(4.4. program)*

Bizonyára sokan rájöttek, hogy ez a formátum a Spectrum gép EDITAS vagy más néven MONEDITAS nevű assembler formátuma. Igaz, a cikksorozat elején gépfüggetlenséget ígértem, de azt hiszem, senki nem kerül így hátrányosabb helyzetbe, mint ha egy valóban gépfüggetlen, soha nem is volt assembler formátumára oldanánk meg ugyanezt a feladatot. Meg kell még jegyezni, hogy ez az assembler a fájl elejére egy CR karaktert tesz.

Miért éppen ezeket a sorokat írassuk ki?

```
SORSZ   LD    BC,1        ;az első sorszám
        LD    HL,(PROG)   ;a program eleje
SZAMOZ  LD    D,(HL)      ;régi
        INC   HL          ;    sorszám
        LD    E,(HL)      ;      DE-be
        PUSH  HL          ;HL elmentése
        EX    DE,HL       ;sorszám HL-be
        LD    DE,9999+1   ;sorszám határ
        OR    A           ;carry=0
        SBC   HL,DE       ;carry beállítása
        POP   HL          ;HL vissza
        RET   NC          ;nagy a sorszám
        LD    (HL),C      ;új
        DEC   HL          ;    sorszám
        LD    (HL),B      ;      elrakása
        INC   HL          ;HL a hossz
        INC   HL          ;  bájtra mutat
        LD    E,(HL)      ;a sor
        INC   HL          ;  hossza
        LD    D,(HL)      ;    DE-be
        INC   HL          ;HL a sor elejére
        ADD   HL,DE       ;HL az új sor elejére
        INC   BC          ;sorszám növelése
        JR    SZAMOZ      ;következő sor
```

4.1. program. BASIC-átsorszámozás (BC a sorszám, HL a címmutató)

```
NOSP    LD    HL,INPB ;HL-be, DE-be
        LD    D,H     ;az inputbuffer
        LD    E,L     ;kezdőcíme
        OR    H       ;zero törlése
CIKL    LD    A,(HL)  ;egy karakter
        LD    (DE),A  ;átmásolása
        JR    NZ,SNV  ;ha az előző nem
        CP    ' '     ;szóköz
        JR    Z,SKIP  ;mindkettő szóköz
SNV     INC   DE      ;cél mutató
SKIP    INC   HL      ;forrás mutató
        CP    '"'     ;idézőjel
        LD    B,A     ;B-be az a jel
        JR    Z,FUT   ;  kerül amire
        CP    '"'     ;    vissza kell
        JR    Z,FUT   ;      törni a fő
        CP    ':'     ;        ciklusra
        LD    B,CR    ;megjegyzés
        JR    Z,FUT   ;  a sor végéig
        CP    B       ;sorvégjel ?
        RET   Z
        CP    ' '     ;zero
        JR    CIKL    ;    beállítása
FUT     LD    A,(HL)  ;egy karakter
        LD    (DE),A  ;      átmásolása
        INC   HL      ;mindkettőt
        INC   DE      ;    növeljük
        CP    CR      ;a sor végénél
        RET   Z       ;  visszatérés
        CP    B       ;
        JR    NZ,FUT  ;
        OR    H       ;zero törlése
        JR    CIKL    ;fő ciklusra
;
CR      EQU   13      ;sorvégjel
```

4.2. program. Sor tömörítése, felesleges szóközök kihagyása

```
PRTLIN  LD    DE,TBTBL  ;tabulátor tábla
        LD    HL,INPB   ;buffer eleje
        LD    C,0       ;a kiírt karakterek száma
PRINT   LD    A,(HL)    ;egy karakter
        INC   HL        ;mutató léptetése
        CP    ' '       ;ha szóköz
        JR    Z,TAB     ;  akkor tabulálás
        CALL  PRNA      ;karakter kiírása
        INC   C         ;számlás
        CP    CR        ;sorvége
        JR    NZ,PRINT
        RET
TAB     CALL  PRNA      ;egy szóköz mindenképpen
        LD    A,(DE)    ;tabulátor pozíció
        INC   C         ;számlás
        CP    C
        LD    A,' '
        JR    NC,TAB    ;tabulátor előtt
        INC   DE        ;következő tabulátor
        LD    A,(DE)    ;ha
        OR    A         ;  van
        JR    NZ,PRINT  ;  még
        DEC   DE        ;vissza az utolsóra
        JR    PRINT
TBTBL   DEFB  8,13      ;tabulátor pozíciók-1
        DEFB  0         ;tábla vége jelző
;
CR      EQU   13        ;sorvégjel
```

4.3. program. Sorkiíratás tabulálva

```
        ORG   40000     ;fordítási cím
        LOAD  40000     ;töltési  cím
PROG    EQU   50000     ;itt kezdődik az
                        ;EDITAS szöveg
CR      EQU   13        ;sorvégjel
EOF     EQU   0FFH      ;fájl végjel
;makro definíciók
BEOLV   MACR            ;makro def.
        INC   HL
        LD    A,(HL)
        ENDM
ATLEP   MACR            ;makro def.
        INC   HL
        ENDM
KERES   MACR  &0        ;makro def.
        LD    BC,0      ;operandussal!
        LD    A,&0      ;hivathozás az arg.-re
        CPIR
        DEC   HL
        ENDM            ;makródefiníciók vége
;*****************************************
INIT    LD    HL,PROG   ;HL mutató
START   BEOLV           ;egy karakter beolv.
        LD    D,H       ;HL elmentése
        LD    E,L       ;a kiíratás számára
        CP    EOF       ;fájl vége
        RET   Z
        ATLEP           ;sorszám
        ATLEP           ;  átlépése
        ATLEP           ;szóköz átlépése
        KERES ' '       ;második szóköz
        BEOLV           ; utáni karakter
        CP    'J'
        JR    Z,JPHL
        CP    'D'
        JR    Z,DEF
        CP    'S'
        JR    Z,SLL
UJSOR   KERES CR        ;a sor végének keresése
        JR    START
;
SLL     BEOLV
        CP    'L'
        JR    NZ,UJSOR  ;nem SLL
UTLSL   BEOLV
        CP    'L'
        JR    NZ,UJSOR  ;nem SLL
        JR    KIIRAS    ;a mnemonik SLL
;
DEF     BEOLV
        CP    'E'

        JR    NZ,UJSOR  ;nem DEFL
        BEOLV
        CP    'F'
        JR    NZ,UJSOR  ;nem DEFL
        JR    UTLSL     ;utolsó L
JPHL    BEOLV
        CP    'P'
        JR    NZ,UJSOR  ;nem JP
        ATLEP           ;mnemonik utáni szóköz
        BEOLV
        CP    'H'
        JR    NZ,IXORIY ;nem HL , IX vagy IY
        BEOLV
        CP    'L'
        JR    NZ,UJSOR  ;nem JP HL
LEZAR   BEOLV
        CP    ' '
        JR    Z,KIIRAS
        CP    CR
        JR    Z,KIIRAS
        JR    UJSOR     ;folytatódik az op.
IXORIY  CP    'I'
        JR    NZ,UJSOR  ;nem is IX vagy IY
        BEOLV
        CP    'X'
        JR    Z,LEZAR   ;operandus
        CP    'Y'
        JR    Z,LEZAR   ;  lezárása
        JR    UJSOR     ;nem X vagy Y az I után
KIIRAS  PUSH  HL        ;A sor kiírása
        EX    DE,HL     ;HL-be a sor kezdocíme
        LD    A,(HL)    ;első két számjegy
        INC   HL
        CALL  DIGITS    ;számkiírás
        LD    A,(HL)    ;második két számjegy
        INC   HL
        CALL  DIGITS    ;számkiírás
KIR     LD    A,(HL)    ;karakter
        INC   HL        ;lépés
        CALL  PRNA      ;kiírás
        CP    CR        ;sorvége
        JR    NZ,KIR
        POP   HL
        JP    START
PRNA    PUSH  AF        ;kiíratő rutin
        PUSH  BC        ;minden regisztert
        PUSH  DE        ;megőriz
        PUSH  HL
        RST   10H       ;SPECTRUM
        POP   HL
        POP   DE
        POP   BC
        POP   AF
        RET
DIGITS  PUSH  AF        ;két számjegy kiírása
        SRL   A         ;a felső
        SRL   A         ;  félbájt
        SRL   A         ;    alsóra
        SRL   A         ;      hozd
        OR    30H       ;konverzió ASCII-ba
        CALL  PRNA      ;kiírás
        POP   AF        ;vissza az eredeti
        AND   0FH       ;felső félbájt törlése
        OR    30H       ;konverzió ASCII-ba
        JR    PRNA
        END
```

4.4. program. EDITAS nem szabvány sorainak listázása

A megoldás folyamatát a *4.2. ábra* mutatja. Minden sorban először el kell jutni a mnemonikig, azaz a második szóköz utánra. Ha a sor első karaktere nem J, S vagy D, akkor a sort nem kell kiírnunk. Ekkor ellépkedünk a sor végéig, és vesszük a következő sort. Ha a karakter S vagy D, akkor meg kell vizsgálni, hogy LL vagy EFL karakterek következnek-e! Ha a mnemonik első karaktere J, akkor azt kell ellenőriznünk, hogy a második P, és az egy szóköz után következő argumentum HL, IX vagy IY! Ez a lépés két részből áll. Ellenőrizni kell, hogy az operandus így kezdődik, és azt is, hogy utána szóköz áll-e. Elképzelhető ugyanis olyan címke, amely a HL, IX vagy IY karakterpár valamelyikével kezdődik, például HLESS = HeaderLESS.

A folyamatábra jelölései:

ÁTLÉP	egy karakter átlépése
BEOLV	a következő karakter beolvasása
KERES char	keresés a megadott karakterig.

A folyamatábra alapján az assembly szöveg elkészítése már automatikusan megy. Az ÁTLÉP helyére mindenhova INC HL utasítást, a BEOLV helyett INC HL LD A,(HL) utasításpárt, a KERES helyett pedig az LD A, char ; LD BC,0 ; CPIR ; DEC HL utasítássort kell beírni a programba. Az assembly nyelvű program gépi kódra való fordításában az assemblerek vannak a segítségünkre. Valamivel nagyobb segítséget nyújtanak az ún. makróassemblerek. Ha ilyen assemblerrel dolgozunk, akkor lehetőségünk van arra, hogy valamely utasítássorozatnak nevet adjunk (makródefiníció) például az INC HL ; LD A, (HL) utasításpárt elnevezhetjük BEOLV-nak, és a programunkban az utasítássorozat helyett csupán a nevet kell leírni (makróhivatkozás). A definícióban paramétereket is megadhatunk, amelyek helyére a hivatkozáskor a megadott, konkrét argumentum kerül. A makrók segítségével sokkal olvashatóbb programokat készíthetünk.

Az egyes assemblerek makródefiníciójának szintakszisa különböző lehet, ezért meg kell nézni az assembler leírásában, hogy tudja-e kezelni a makrókat és hogy hogyan. A 4. program is egy makroassembler segítségével készült.

VERHÁS PÉTER

A DEFL direktívát nem minden assembler ismeri. Az SLL nem standard Zilog utasítás. Ez azt jelenti, hogy a processzort gyártó cég nem garantálja, hogy bármely sorozatban készített gyártmánya azonos módon reagál egy ilyen utasításra. Az ilyen utasítások használata ellenjavallt, és ezért az assemblerek többsége nem is ismeri fel ezek mnemonikjait. A JP HL utasítás a JP (HL) alakban szabványos, és így terjedt el. Az EDITAS azért engedi meg a zárójel nélküli alakot, mert készítői úgy gondolták, ez az alak jobban kifejezi az utasítás hatását. Felfogás kérdése, mert egyrészt a HL, és nem a HL által mutatott címen elhelyezkedő két bájt értéke kerül a PC-be, másrészt mint minden ugróutasításnál, címre hivatkozunk és nem a HL-re kerül a vezérlés. A zárójeles alak terjedt el!

Figyelem!

A TURBO interfész ZX-Spectrumhoz c. cikk (1986. szeptemberi szám) 2. ábrájában az alsó IC beültetése **fordított!** Az IC-t pozicionáló félkör **a jobb oldalon van.** A hibáért olvasóink szíves elnézését kérjük.

A MINŐSÉGÜGY KÖZÜGY

Van-e kiút a zsákutcából?

„Jó pap holtig tanul" tartja a közmondás. *Tanul,* lehet, hogy azért mert jó, lehet, hogy azért mert pap, azért azonban bizonyosan mert *van annyi tanulnivaló, hogy holta előtt nem jut a végére.* Jaj tehát annak aki *befejezettnek tekinti* saját vagy vállalata fejlődését. Jaj annak aki azt hiszi (akár spontán akár mert mások elhitették vele), hogy *már eleget tanult, eleget fejlődött.*

Az önelégültség a jó minőség legbiztosabb ellenszere. Más általános szabály a minőséggel kapcsolatos (minőséghez kapcsolódó) feladatok megoldására nincs. Nem is lehet, hiszen az ilyen feladatok megfogalmazására sincs semmilyen általános eljárás, csak az, hogy *fejlődni kell, mégpedig minden szempontból.* A fejlődés, ami gyakran, de nem minden vonatkozásban egybeesik a megújulással, újítással, természettörvény. Az emberre és munkájára is vonatkozik. Ennek a fejlődésnek persze lehetnek és vannak is kisebb-nagyobb zsákutcái. Ezek némelyike elkerülhető, mások elkerülhetetlenek. Egy — talán nem mindenki által ismert — példa. Gondos felmérések alapján padlókefélő gépek gyártására gyárat létesítenek. Alighogy ez elkezd termelni, megjelennek a piacon a parkettlakkok, és egyik napról a másikra kiszorítják a parkett viasszal bekenés után kefével való fényesítését. Egy másik zsákutca, amelyet viszont el lehetett volna kerülni, *a számítástechnika mai oktatásának egyoldalúsága.* Nyilvánvaló tény, hogy a számítástechnika létének, fejlődésének meghatározója a gép. Amely *feladatmegoldó gép.* A feladatok lehetnek például információtárolási, információkeresési, sokféle kérdésre válaszadási, technológiai rendszerek, automaták, robotok, szerszámgépek stb. irányítási feladatai és számolási, számítási feladatok is. Mégis a név által (számoló-számító) szuggeráltan ma az oktatás számítástechnika címen lényegében gépi nyelveket (leginkább csak egyet) oktat. Pedig e „nyelv-tan" csak nagyon kicsi és sokadlagos fontosságú része a számítástechnikának. A számítástechnika pedig nagyon kicsi (de fontos!) része a gépalkalmazás tudományának. *Egy fa odújába (BASIC odú) bebújva, még a fát sem látják, nemhogy az erdőt . . .*

Olyanok akiknek csak felületes, másodkézből szerzett ismereteik vannak, akik nem ismerik a gép teljes alkalmazási körét, és így áttekintésük sincs róla, vitatkozgatnak azon, hogy számítógép, vagy logikai gép vagy matematikai gép a „computer". (Mások ez idő alatt keményen dolgoznak vele.) Van aki a gépet szövegszerkesztésre használja, legyen tehát *szerkesztő gép?* Van ahol a legnagyobb hasznot annak révén hajtja a „computer" hogy hosszú-hosszú ideig képes kis helyen megbízhatóan információt tárolni. Legyen tehát *tárológép?* Másutt főleg rajzi munka készítését végeztetik vele, lenne emiatt *rajzológép?* Vagy „szakértői rendszereknél" *szakértő vagy tanácsadó gép?* E kérdéseket érdemi munka helyett a meddő vitaexhibicionizmusnak átengedve, a minőséget mindenkor fenyegető egyoldalúság és szűklátókörűség veszélyére hívjuk fel a figyelmet.

Különösen káros az egyoldalúság és a szűklátókörűség az ismeretnyújtás, az oktatás, a képzés minőségére. E minőség pedig sorsdöntő a jövő minőségére vonatkozóan. Ezért hangsúlyozzuk, hogy káros és nagyon veszedelmes a mai egyoldalú minőségfogalom, amely termékek és szolgáltatások minőségére korlátozódik.

Nem termékek és szolgáltatások, hanem egész rendszerek, egész organizmusok jó minőségű működésével kell foglalkozni. Ebből következik majd, hogy jó (megfelelő) lesz a termékek és szolgáltatások minősége is. Ennek fordítottja azonban nem igaz.

Mint már mondottuk, a tévedések, félrecsúszások, hibák néha elkerülhetetlenek. Nehéz (ha egyáltalán megoldható) az a feladat, hogy pontosan megállapítsuk *„a helyes irányt".* Így *„a helyes iránytól való eltérés"* megállapítására sem vághatjuk rá általában, hogy mindig megoldott probléma.

Mivel minden (valamilyen kívánalomrendszerhez viszonyítva) többé kevésbé tökéletlen, hibás, selejtes, rendkívül nagy szerepe van a hibának, a hibával kapcsolatos tudományos tevékenységnek. Hibátanunk, hibaelméletünk azonban mind a mai napig nincs, nemcsak a számítástechnikában hanem az élet egyetlen területén sem. Ezzel kapcsolatos, hogy ha hiba van, akkor illik javítani. Javítás-, fejlesztéstudományunk sincsen. Csoda tehát, hogy a hibák kijavítása nemritkán újabb nagyobb hibákkal történik? A ló egyik oldaláról a ló másik oldalán a földre kerülve?

Van értékelemzés (nagyon hasznos gyakorlati „tudomány"), de nincs hibaelemzés. És nincs javítás- és fejlesztéselemzés, javítás és fejlesztéstudomány sem. Ez utóbbiak folyamatokkal foglalkoznak, jó minőségű elvégzésükhöz szükség volna *folyamattanra, folyamatelméletre, folyamatok tudományára.* Ez sincs.

Tennivaló van tehát bőven. A „nincsek" felsorolását programadásnak szánva ugyan, nem ringatjuk magunkat olyan ábrándokban, hogy az összes „nincs"-re a közeli holnap jó minőségű „van"-okkal felel. Különben is „hamar munka ritkán jó" *(minőségű).* Sikernek könyvelhetnénk el, ha nem túl hosszú idő alatt összegyűlne és általános használatba kerülne a legfontosabb minőségmutatók, minőségjellemzők és minősítési eljárások gyakorlatban kielégítően jól használható készlete.

Ezzel szoros kapcsolatban van a minősítés, rangsorolás témaköre. Már tárgyaltuk, hogy csak konkrét igénnyel szembeállítva van értelme az igény kielégítésének mértékéről beszélni. Azok a széltében hosszában propagált módszerek, amelyek nyilatkoznak pl. egy cipő jóságáról anélkül, hogy a lábról mértéket vettek volna, sőt bármit is tudnának róla, általudományos konjunktúratermékek csupán. Fontos azonban megjegyezni, hogy lehet egy cipő átlagos megfelelőségéről beszélni, ha egy sokaság minden egyes elemére vonatkozó megfelelőségét átlagoljuk. Így összehasonlíthatunk pl. termékeket, szolgáltatásokat egy populációra vonatkozó átlagos megfelelőségük alapján. A kérdés csupán az, hogy ez az átlagolási művelet mikor vezet valósághű eredményre, mikor engedhető meg egyáltalán, és milyen valóban hasznosítható minősítési, rangsorolási ítéletek képezhetők, amelyekben ilyen átlagokat szerepeltetünk.

Az új, még megalkotandó *minőségtudománynak* foglalkoznia kell a minőség — valamire vonatkozó megfelelőség — helytől, időtől és környezettől való függésével is. Ennek fontosságát szemléltetheti a számítástechnika-piac egy rövidesen bekövetkező válságjelensége is. A rendkívül gyors hardverfejlődés miatt rövidesen hatalmas tömegű lesz az a szoftver, amely elavuló hardverek miatt használhatatlanná válik. Ennek a pazarlásfolyamatnak csak lényegtelen lassítója lehet a nyelvi azonosság miatti portabilitás azaz átvihetőség (nem pedig hordozhatóság, ahogyan sokan hibásan fordítják!). Várhatóan szükség lesz széles körű *értékkimentő tevékenységre.* Ennek eszközei a cross assemblerekre és más fordítóprogramokra, átültetőprogramokra fognak épülni.

Egy program minősége tehát lehet, hogy csökken egy darabig. Aztán amikor a gépet, amelyen futott kidobják, értéke, minősége újra megváltozhat. Ha lesz kezünk ügyében egy megfelelő átültetőprogram, cross assembler vagy más fordítóprogram, lehet, hogy értéke nagyot ugrik felfelé, ha azonban nem lesz, ez az ugrás lefelé történik és nem feltételesen, hanem biztosra vehetően.

Reméljük, hogy példáink meggyőzően bizonyították, hogy a minőségzsákutcákból az egyik kiút biztosan az *előrelátás* lesz, ez pedig nehezen képzelhető el a *körültekintés* nélkül.

Barátunk, a számítógép

Testközelben

(Mottó: ... Azzal fecséreled el idődet, hogy a mások kezében lévő szerszámokat kívánod. És nem veszed észre, hogy a te kezedben is vannak szerszámok. Mások ugyan, de éppoly használhatók ... — M. Quist)

Tegyük fel, hogy ötször öt az huszonöt! Mosolyogsz barátom? Ez a mosoly kínos vigyorrá torzult szegény kollégánk arcán, amikor a vásár egyik számítógépes standja előtt huncut szemű bácsika állt meg. — No, fiam, meg tudja-e mondani az a fene masinád, hogy mennyi ötször öt? — kérdezte. Jeles szakemberünk erre irtózatos kapkodásba fogott. 24, 18, hibaüzenetek. Két perc múlva azonban produkálta az eredményt. — Szép! — mondta az öreg, talán tolnai székely, és pökött. Talán nem ártana, ha ilyen helyekre is küldenénk valakit, aki ért az annyira lenézett pedagógus szakmához. Amit egyszer, érdeme szerint, tiszteltek. Mert akkor a jeles szakember nem pirult volna, hanem így szólt volna: — Bácsika, ez nem erre való! De hogy termett idén a híres szekszárdi vörös? És azzal lett volna egy örök és hűséges barátja — a számítógépnek.

Bocsássanak meg nekem, ha most kicsit személyes leszek. Tudom, tízéves fiataloknak talán nem mondok újat. Tegnap voltam negyven, és 16 éve vagyok a szakmában. És pár hete kezelek először számítógépet. Szabad-e személyes élményeimről beszámolnom? Ha nem érdekel, akkor is várlak a következő cikkrészletben!

Olvastam arról, hogy idős emberek, akik magukra maradnak, hosszabb ideig élnek, ha van egy becézhető, testközeli kis állatka a közelükben. Ne nézzetek bolondnak! Én szentül hiszek abban, hogy ennyire fontos a testközeliség a mi szakmánkban is. És háborút átélt öregjeinknek miért ne válhatna hobbijukká barátunk? Mindig csak a fiatalokra gondolunk? De ez más kérdés. Ha érdekel testközeli barátságom a számítógéppel, akkor figyelj!

Egy furcsa írógép

Tizenhat éve írógéppel keresem a kenyerem azok szerencséjére, akiknek nem kell kéziratomat — reménytelenül — olvasni. Ez az írógép előttem picit más. Mennyiben?

A dinnyék felcserélték a betűket. És nincsenek ékezetes betűk. Speciel itt átprogramozták a szögletes zárójelet „é" betűvé. De ezt nem használom, mert másutt hátha nem így tettek? Apropó! Dörmögnek nyelvészeink, hogy a számítástechnikában miért használunk ékezet nélküli, a magyar nyelvet rontó betűket. És tettek-e ők, valakik, valamit azért, hogy ebben a bizonyos „átprogramozásban" legyen valamiféle szabvány? Mint az írógépen? Tartom magam annyira jó magyarnak, mint ők. De piacról és forgalmazott szoftverből él az ember.

Jobbra-balra speciális billentyűk. Lenyomtam az egyiket, mire minden eltűnt, amit addig nehezen megírtam. Mögöttem jóízű röhögés: a balfácán. Jobb ezekhez nem nyúlni! Vagy mégis? Pár nap múlva varázslatokat műveltem segítségükkel, pedig már nem vagyok tanulékony. Az is igaz, hogy nélkülük is lehet boldogulni. Meg az is, hogy egyes rendszerkészítők éppen e bűvöletben élnek. De róluk majd később szólok.

Legfurcsább az, hogy minden zajtalan — és nincs papír. Állítom, hogy Murphy törvénye részben igaz. Az öreg Erikán az esetek tíz százalékában igenis a helyes ütődik le az összekavarodott betűkarok közül. Esélyem a villanysebességű írógépnél összezsugorodott. Minden leütődik.

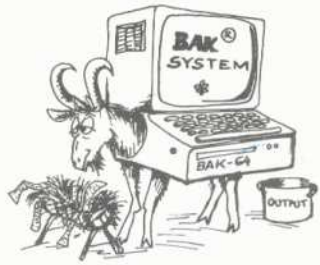
A lényeg: ez az írógép az úgynevezett szabványos bemeneti egység, a konzol. Mondanivalómat itt közöltem kis barátommal, a számítógéppel. Igen elnagyoltan fogalmazva közlendőm háromféle lehet: adhatok adatot, adhatok információfeldolgozási tevékenységre történő felhívást és mindenféle általános vezérlést. Néha ezek keverednek. Csak ne felejtsek el erre még visszatérni!

Naphosszat tévézek

Előttem, egy lapos doboz tetején, picinyke tévékészülék. Ha valamit gépelek, az ott jelenik meg. Szürke alapon zöld betűk. Majd sárga alapon feketék. Nincs papírom, tehát szükségem van erre a ketyerére, de egy ideig nem szerettem. Később igen.

Nem szerettem, mert mindig úgy kell fordítanom, hogy ne essen rá fény. Akkor ugyanis becsillog. Láttam én már színesebb eszközöket is, de nem voltam elbűvölve. Ha a focivébén barátom, akinél együtt szoktuk nézni a meccseket, így állítja be a színes tévéjét, akkor közös a felhördülés: Te, ez túl éles! Valami pasztell szint kérünk! Szó, mi szó, öreg Erikámon egy napig vagyok képes gépelni baj nélkül, itt pedig két-három óra után holtfáradt vagyok. Barátom, Te is vigyázz a szemedre!

A pici tévé a szabványos kijelzőegység. Megjeleníti azt is, amit én az írógépen írok, meg azt is, amit nem akarok. Előkelő üzenet: „system failed" (azaz minden bedőlött). Meg aztán csak 24 sort tudok írni, miközben egy normális gépelt oldal ugyebár 30 sor. No, majd kibékülünk. Hiszen nem kellene ezt a cikket sem újragépelni, ha be akarok toldani egy szakaszt, vagy valamit javítani kívánok. Mindezt láthatom a tévén. Kis tolás jobbra, egy sor beillesztése — gyerekjáték.



Csak ne lenne ilyen hülye színű!

A másik írógép

Krrrr, ... nyekk. Krrrrr, ... nyekk. Nem nádi rigó. Hanem a „szabványos kimeneti egység" — helyett bekapcsolható írógép. Valódi papírral, bár nem írókarokkal. Kell apró bűvölet, ameddig ő fog megindulni. De már a latinok is megmondták: a képernyő elszáll, az írás megmarad. Így örülök, amikor mellettem duruzsol, papírt kapok mindenről, amiről akarom. Kicsit szokatlan, hogy barátom mindent kétszer mond: képernyőn is, nyomtatón is. De nem mindig. Van, amikor választhatok.

Központi egység

Súlyos szavak. Központi is és egység is. Pedig valójában lapos doboz. Kollégáim nem engedték, hogy csavarhúzó eszközletével belülről is megnézzem. Pedig akkor jobban érteném. Szerintük viszont egy darabig utána ők sem értenék. Furcsa emberek.

Annyi tény, hogy én valamit az első írógépen beviszek, mire vagy a tévén vagy a másik írógépen megjelenik az, amit beviszek, vagy akár valamiféle eredmény is, amit nem vittem be. Vagyis ez az eszköz dolgozik. Olyasféle, mint a kérődzők gyomra. Angolmániás körökben beszélnek input-feldolgozás-output láncról, holott bemenet-feldolgozás-kimenet láncról illenék szólniuk. Mivel a feldolgozás kétségkívül adatfeldolgozás, én a lényeget így látom:

Nem tagadhatom, hogy már tudom: a központi egység nem giliszta. Nem egyszerű közvetítő a két vég között. De erről majd később! Hiszen van itt még más fontos részlet is!

Lejárt lemezek helyett

Ha fáradt vagyok, ma is szívesen olvasom „Zsül", vagy ahogyan én ejtettem: Jólesz Verne híres regényét, a Kétévi

vakációt. No, abban még a „vincseszter" a vadkacsák távoli ártalmára szolgáló lőfegyver volt. Most meg, lám, rögzített — csakis csavarhúzóval kiszedhető — lemez (nem engedték). A miénk kicsi. 20 Mbájt információ tárolására alkalmas. Kicsi? Te jóságos ég! Ifjúi zavaromban írtam egy-két könyvet. Ha egy oldalt írtam, akkor az ugyebár 60 betűször harminc sor. Egy könyv tehát 555 oldallal számolva is csak 1 millió karakter. A gyorstüzelő vincseszterre tehát 20 könyv fér el . . . Kinek kell ennyi információ?

Azután a lapos doboz két részébe bedugdoshatok még két, franciás becézésű diszkettet, azaz lemezecskét is. Ez remek találmány! Nagygépes kollégáim kínlódnak: ki védi meg az adatot a jogtalan hozzáférés ellen? Ki, hát én! Kiveszem a „floppit" és hazaviszem. És szeretném látni azt a számítógépes gengsztert, aki el tudja lopni az én adatomat, az én programomat! Ha fiatal lennék, ha új és ifjú szerelmem lenne, ha mindkettőnknek lenne számítógépe, akkor lágy floppin küldenék neki még lágyabb szerelmes leveleket. No, ezt utánozzátok nagygépen!

Hogy ezek mire valók? Egyszerű. Valahányszor bekapcsolom a gépet, mindig beüthetném a háztartási elszámolásomra szolgáló műveletek sorát. Nem teszem. Egyszer bevittem, most nyugodjék békében a vincseszteren vagy a floppyn. (Utóbbin, hogy nejem ne lássa.) Így vagyok az adatokkal is. Tény, hogy egyszer annyit költöttem, hogy . . . Nem fogom ezt minden alkalommal közölni még barátommal, a számítógéppel sem. Jegyezze meg, és kész! Vagyis a lemezek az ismétlődően felhasználandó programjaim és adataim tárolására szolgálnak. Az én kényelmemért. Ezért a bemenet-adatfeldolgozás-kimenet képlet nem is olyan egyszerű.

Mi tagadás, elsőre így láttam barátunkat. Nem beszélek most, az első nász hevületében a gondokról. Fogok, amint tapasztalok. Ismétlem: nem az értőkhöz szóltam. Picit talán hozzájuk is. Hogy felkészüljenek az első szerelmet követő, elkerülhetetlen csalódásokra. De: az első szerelem mindig szép és igaz. Még testközelben is.

DR. HALASSY BÉLA

A korlátozások:

— A lefoglalható terület hosszát a 6510-es mikroprocesszor indexelési rendszere határozza meg. Az egybájtos indexregiszterek miatt a tárban 2 BASIC-sor nem kerülhet távolabb egymástól, mint 256 bájt.

— A lefoglalt területen nem lehet Ø bájt, mivel azt a rendszer a BASIC-utasítássorok elválasztására használja.

PROGRAMTITKOSÍTÁS

A gépi kódú program elhelyezésének egy ritkán használt módját ismertetem, amikor a BASIC-utasítások között foglalunk helyet a gépi kódú program részére. Az így lefoglalt terület felhasználására szolgáló példaprogramban egy olyan gépi kódú rutint helyezünk el a BASIC programon belül, amely titkosítja azt.

Területfoglalás a BASIC programon belül

A BASIC-utasításokat a gép a következőképpen tárolja. A program első sorát a SOB (43,44) által meghatározott címtől kezdi elhelyezni. Az első két bájtra egy mutató kerül, amely a következő BASIC-sor elhelyezési címét adja meg. A mutató utáni két bájt a BASIC-utasítás sorszámát tartalmazza. Ez a kétbájtos elhelyezés határozza meg a legnagyobb használható sorszámot (1. ábra).

1. ábra

Minden BASIC-sor a fenti elrendezésben van tárolva. Az utolsó sor után következő mutató tartalma ØØ. Ez jelzi, hogy nincs több utasítássor.

— A lefoglalt területre nem kerülhet BASIC-vezérlés, mert az ott lévő bájtokat nem lehet BASIC-utasításoknak értelmezni.

— A program listázásakor a LIST rutin sem tudja a lefoglalt területet értelmezni, ezért a listázást a lefoglalt területet követő első BASIC-sortól kezdve érdemes végezni.

Milyen előnyökkel jár mégis a tárterület lefoglalásának ez a módja?

— A SAVE és a LOAD parancsok a programmal együtt a lefoglalt területet is kimentik, illetve betöltik.

— Nem kell a BASIC-mutatókat megváltoztatni.

— Ha a program első utasításai közül valamelyikben lefoglaltunk egy területet, akkor az ezt követő utasításokat szabadon módosíthatjuk (átírhatjuk, kiegészíthetjük és törölhetjük) anélkül, hogy ez a lefoglalt területre bármilyen hatással lenne.

A *P-3 jelű program* maximum 240 bájtot tud lefoglalni a BASIC program első utasításában. A program a következő módon használható. Bevisszük a P-3 jelű programot és RUN paranccsal elindítjuk. A program megkérdezi, hogy hány szabad bájtra van szükségünk. A válasz megadása után megkapjuk az első és az utolsó szabad bájt címét. A program ezután betölt egy gépi kódú programot, amely törli a P-3 jelű programot, annak első három sora kivételével.

2. ábra

Ha valamelyik BASIC-sor kezdetét jelző mutatót átírjuk, akkor a rendszer a következő utasítássort az átírt címen fogja keresni, és így két tárolt BASIC-sor között egy területet lefoglalhatunk (2. ábra).

Az így lefoglalt terület azonban nem lehet tetszés szerinti hosszúságú, és nem is használható minden korlátozás nélkül.

P-3

```
1  REM
2  REM
100  PRINT "[CLR][CD][CD]       [RVS]  **
      MEMORY ALLOCATION **  "
105  PRINT "          **  ( TARFOGLALAS )
      **  "
110  INPUT "[CD] FELSZABADITANDO BYTOK S
     ZAMA              (MAX240)";N
115  IF  N < 1  OR  N > 240  THEN  110
120  E =  PEEK(43) + 256 *  PEEK(44) + 1
     2
130  PRINT "[CD]   ELSO SZABAD BYTE:";E
140  VE = E + N + 1:U = VE - 2: PRINT "
       UTSO SZABAD BYTE:";U
150  GOSUB  300: POKE  831,L: POKE  832,
     H
170  VE = VE + 29: GOSUB  300: POKE  833
     ,L: POKE  834,H
200  PRINT "[CD]          [RVS] VARJ ":
      FOR  I = 1 TO  114: READ  D: POKE
      834 + I,D:S = S + D: NEXT
210  IF  S <  > 12792 THEN  PRINT "HIBAS
      DATA": END
220  PRINT "[CU] ELSO SZABAD BASIC SORSZ
     AM= 3"
240  SYS  864: LIST : END
300  H =  INT(VE / 256):L = VE - 256 * H
     : RETURN
500  DATA 2,0,143,42,65,76,76,79,67,39
510  DATA 68,46,77,69,77,46,66,89,32,66
520  DATA 46,90,83,79,77,42,0,0,0,162
530  DATA 4,189,62,3,149,250,202,208,248
     ,160
540  DATA 7,165,251,145,43,200,165,252,1
     45,43
550  DATA 160,31,185,65,3,145,251,136,20
     8,248
560  DATA 160,0,165,253,145,251,200,165,
     254,145
570  DATA 251,24,165,253,105,2,133,45,13
     3,47
580  DATA 133,49,165,254,105,0,133,46,13
     3,48
590  DATA 133,50,56,173,63,3,229,43,168,
     136
600  DATA 169,0,145,43,169,42,136,145,43
     ,192
610  DATA 12,208,249,96
```

A P-3 program

P-4

```
100  PRINT "[CD][CD][CR][CR]VARJ"
110  FOR  I = 2061  TO  2264: READ  A:
      POKE I,A:S = S + A: NEXT
120  IF  S <  > 21861  THEN  PRINT "DAT
     A ERROR": END
130  FOR I = 2294  TO  2296: POKE  I,0:
      NEXT : POKE 45,249: CLR : END
1000  DATA  24 , 165 , 61 , 105 , 15
1002  DATA  133 , 34 , 165 , 62 , 133
1004  DATA  35 , 144 , 2 , 230 , 35
1006  DATA  32 , 211 , 8 , 162 , 1
1008  DATA  189 , 63 , 8 , 73 , 1
1010  DATA  32 , 210 , 255 , 232 , 224
1012  DATA  25 , 208 , 243 , 32 , 211
1014  DATA  8 , 162 , 1 , 189 , 88
1016  DATA  8 , 32 , 210 , 255 , 232
1018  DATA  224 , 12 , 208 , 245 , 240
1020  DATA  48 , 33 , 43 , 66 , 78
1022  DATA  69 , 68 , 69 , 33 , 76
1024  DATA  68 , 76 , 47 , 33 , 90
1026  DATA  66 , 92 , 33 , 67 , 47
1028  DATA  91 , 82 , 78 , 76 , 43
1030  DATA  33 , 32 , 80 , 65 , 83
1032  DATA  83 , 87 , 79 , 82 , 68
1034  DATA  58 , 63 , 32 , 32 , 87
1036  DATA  65 , 73 , 84 , 33 , 32
1038  DATA  255 , 255 , 255 , 255 , 169
1040  DATA  3 , 133 , 36 , 32 , 228
1042  DATA  255 , 240 , 251 , 166 , 36
1044  DATA  157 , 108 , 8 , 202 , 134
1046  DATA  36 , 16 , 241 , 32 , 211
1048  DATA  8 , 162 , 1 , 189 , 100
1050  DATA  8 , 32 , 210 , 255 , 232
1052  DATA  224 , 7 , 208 , 245 , 14
1054  DATA  111 , 8 , 46 , 110 , 8
1056  DATA  46 , 109 , 8 , 46 , 108
1058  DATA  8 , 144 , 3 , 238 , 111
1060  DATA  8 , 234 , 160 , 1 , 177
1062  DATA  34 , 240 , 18 , 205 , 108
1064  DATA  8 , 240 , 5 , 77 , 108
1066  DATA  8 , 145 , 34 , 230 , 34
1068  DATA  208 , 218 , 230 , 35 , 208
1070  DATA  214 , 200 , 177 , 34 , 200
1072  DATA  17 , 34 , 240 , 13 , 24
1074  DATA  169 , 5 , 101 , 34 , 133
1076  DATA  34 , 144 , 197 , 230 , 35
1078  DATA  176 , 193 , 96 , 169 , 13
1080  DATA  32 , 210 , 255 , 96
```

A P-4 program

A 0-ás sorszámú sorra azért van szükség, hogy átugorjuk az 1-es sorszámú sort. Az 1-es sor REM-je után annyi * karaktert helyez el a program, ahány szabad bájtot kértünk. A 2-es sort ugyanennyi bájttal feljebb tolja a tárban, majd a sor kezdetét jelző mutatót az 1-es sorban átírja az áthelyezett 2-es sornak megfelelően.

A program lefutása után egy három sorból álló programot kapunk. A * karakterekkel feltöltött terület helyére betölthetünk például egy gépi kódú rutint, amit a programunk használhat. Ekkor a *-ok helyén más karakterek fognak megjelenni, de ez a program többi részére nincs hatással.

Programtitkosítás

A következő részben a P-3 programmal lefoglalunk egy területet, és ott elhelyezünk egy gépi kódú programot, amely a BASIC programot titkosítja.

A titkosítás a KIZÁRÓ-VAGY logikai műveletet használja. Ennek lényege, hogy az eredmény akkor 1, ha vagy csak az egyik vagy csak a másik bemeneten van 1. Ha

A P-4/próba program

P-4/PROBA

```
10  PRINT "[CLR][CD][CD] TITKOSITO PR
    "
20  SYS 2061
30  PRINT "[CD][CD]  [RVS]EZ MAR ATI
     RESZ[CD][CD]"
40  FOR  I = 1 TO 10
50  PRINT I,I ↑ 2
60  NEXT
70  END
```

mindkét bemeneten Ø vagy mindkét bemeneten 1 van, akkor az eredmény Ø. A KIZÁRÓ-VAGY igazságtáblázata:

A bemenet	B bemenet	C eredmény
Ø	Ø	Ø
Ø	1	1
1	Ø	1
1	1	Ø

Vegyünk tetszés szerinti számokat az A és a B helyére. Legyen például A=92, B=58. Írjuk fel ezeket bináris alakban, és végezzük el a KIZÁRÓ-VAGY műveletet.

	128	64	32	16	8	4	2	1
A= 92=	Ø	1	Ø	1	1	1	Ø	Ø
B= 58=	Ø	Ø	1	1	1	Ø	1	Ø
C=1Ø2=	Ø	1	1	Ø	Ø	1	1	Ø
B= 58=	Ø	Ø	1	1	1	Ø	1	Ø
A= 92=	Ø	1	Ø	1	1	1	Ø	Ø

A táblázat segítségével könnyen megérthetjük a program által használt titkosítási módszert. A titkosítandó program valamelyik bájtját az A képviseli. A titkosítást a B kóddal végezzük. A két bájt között elvégzett KIZÁRÓ-VAGY művelet eredménye, C lesz a program titkosított változata. Ha ezt tároljuk a lemezen, akkor azt hiába hívja be valaki, sem futtatni, sem listázás útján megnézni nem lehet, és a lemez-monitorprogramok sem adnak a programról érthető információt. A program eredeti formára csak a titkosításkor használt kóddal kódolható vissza. Ezt látjuk a táblázat utolsó két sorában. Ha a titkosított változat C és a titkosító kód B között ismét elvégezzük a KIZÁRÓ-VAGY műveletet, akkor ismét visszakapjuk az eredeti A változatot.

A *P-4 jelű program* a BASIC program kívánt részét titkosítja a fenti módszerrel, mégpedig úgy, hogy a program minden bájtját más-más kóddal kódolja át, megnehezítve ezzel a megfejtést. A titkosítás kódjának kezdő értékét a program használója 4 karakterben adhatja meg. Ez a 4 karakter lesz a jelszó (PASSWORD), amivel a program elindítható. A jelszót a program nem tárolja, így azt onnan nem lehet visszakeresni. A felhasználó tetszés szerinti 4 karaktert választhat jelszónak, és a jelszót bármikor meg is változtathatja. A program mindig azzal a jelszóval kódolható vissza, amivel az átkódolás történt.

A titkosított program elkészítésének menete a következő. Vigyük be a P-3 jelű programot, majd RUN-nal történő elindítása után a gép kérdésére adjuk meg a következő adatot:

FELSZABADÍTANDÓ BÁJTOK SZÁMA: 2Ø4

Ekkor a P-3 program az ott leírtak szerint készít egy 3 utasítássorból álló programot. Az 1-es sorszámú utasításban helyet foglal a titkosító program számára. Ezt a 3 utasítássort közvetlenül a billentyűzetről nem tudjuk beírni, mert az 1-es utasítássor önmagában is 6 képernyősort foglal el, a billentyűzetről viszont csak 2 képernyősornyi utasításokat készíthetünk. NEW parancs kiadása nélkül, folyamatosan írjuk be a P-4 jelű programot. Ekkor a P-3 által készített 3 utasítássoros program és a P-4 jelű program a tárban egy programként fog szerepelni. Ha van valamilyen kiegészített BASIC programunk, akkor a két program összefűzését az APPEND utasítással is elvégezhetjük, de ekkor is a már lefuttatott P-3 programhoz kell hozzáfűzni a P-4-et.

Ezután RUN paranccsal indítsuk el az összefűzött programot. Ekkor a 11Ø-es sorban lévő rész feltölti az 1-es sorban lefoglalt területet a titkosító programmal. A 12Ø-as sor ellenőrzi, hogy helyes adatokat tartalmaznak-e a DATA utasítások. A 13Ø-as sor törli a P-4-es programot, és ismét 3 utasítássorból álló programot kapunk, ami csak abban különbözik a P-3 által készített programtól, hogy az 1-es utasításba a *-ok helyére a gépi kódú titkosító program került.

Az így elkészített titkosító programot lemezen tároljuk, majd tetszés szerinti programot hozzáfűzve, a hozzáfűzött programot titkosíthatjuk.

Próbaként írjuk hozzá a kapott titkosító programhoz a *P-4/próba jelű programot.*

A 1Ø-es sor jelképez egy tetszés szerinti hosszúságú programrészt, ami a program indításakor megjelenő képernyőképet készíti el. Ez a programrész még nem titkosított.

A 2Ø-as sor hívja a gépi kódú rutint. A hívást 2-esnél nagyobb sorszámú utasításban a programban bárhol elhelyezhetjük. A hívás után a program megkérdezi a PASSWORD-t. Addig nem történik semmi, amíg válaszként 4 karaktert nem adunk meg. A PASSWORD megadása után a hívás utáni programrész átkódolódik. Mint már láttuk, a kódolás oda-vissza egyformán történik, így a kódolatlan program titkosításra, a titkosított pedig visszakódolásra kerül. Az első futásnál tehát egy kódolatlan program titkosítása történik a megadott PASSWORD-del. A program SYNTAX ERROR jelzéssel leáll, mivel a titkosított részt a gép nem érti. Egy most kiadott SAVE parancs a titkosított programot a gépi kódú titkosító programmal együtt fogja tárolni. Ezután ha a titkosított programunkat lemezről betöltjük és elindítjuk, akkor először megjelenik a képernyőn a nem titkosított rész által készített kép, majd a program ismét a PASSWORD-t fogja kérni. Az eredeti program csak akkor áll helyre, ha jó PASSWORD-t adunk meg. Ekkor végrehajtódik a program többi része is, a P-4/próba 3Ø-as sorától.

A titkosító program után a P-4/próba helyett bármilyen más programot elhelyezhetünk. A titkosítást is és a visszakódolást is SYS 2061-gyel hívhatjuk, és mindig csak a hívás utáni rész lesz átkódolva.

ZSOM BÉLA

ZX—SPECTRUM

Sprite-rajzoló

A rutin a BETA-BASIC PLOT X, Y, $-jához hasonlóan működik. Öt bemenő adatra van szükség. AD1-en az X koordináta, AD2-n az Y koordináta van, az origó pedig a bal felső sarokban. A rutin az alsó két sorba is tud rajzolni. AD3-on a sprite szélessége karakterben, AD4-en magassága pixelben, AD5-ön és AD5+1-en a báziscím van (AD5 az alacsonyabb helyiérték). A sprite-ot nem karakterenként, hanem soronként kell eltárolni.

```
1Ø REM E=AZ AD5 ÁLTAL MEGHATÁROZOTT CÍM
2Ø FOR F=Ø TO 7
3Ø POKE E,PEEK(USR"A"+F):LET E=E+1
4Ø POKE E,PEEK(USR"B"+F):LET E=E+1
5Ø NEXT F
6Ø FOR F=Ø TO 7
7Ø POKE E,PEEK(USR"C"+F):LET E=E+1
8Ø POKE E,PEEK(USR"D"+F):LET E=E+1
9Ø NEXT F
```

Ez a program egy $\begin{matrix}AB\\CD\end{matrix}$ négyzetet tárol el.

A rutin használatánál még a következőket kell figyelembe venni:

— a rutin csak relatív ugrásokat tartalmaz,

— a printer-puffer első néhány bájtját munkaterületnek használja,

— a „rendszerváltozók" értékeit megváltoztatja,

— a kiíratás XOR-ral (OVER 1) történik,

— az ATTR bájtokat nem állítja át.

BORBÁS ZOLTÁN

```
1Ø REM C 1986. BORBAS ZOLTAN
2Ø INPUT "KEZDŐCÍM=", Q
3Ø DIM A(5,2)
4Ø FOR F=1 TO 5
5Ø LET A$="AD"+STR$ F+"="
6Ø INPUT (A$), A(F,1)
7Ø LET A(F,2)=INT (A(F,1)/256): LET A(F,1)=A(F,1)-256*A(F,2)
8Ø NEXT F
9Ø LET S=Ø
1ØØ FOR F=Q TO Q+143
11Ø READ A: POKE F,A: LET S=S+A
12Ø NEXT F
13Ø IF $=1Ø496+2*(A(1,1)+A(1,2))+3*(A(2,1)+A(",2))+4*(A(3,1)+A(3,2))+2*(A(4,1)+A(4,2))+2*(A(5,2)) THEN PRINT "ELGÉPELÉS!": STOP
```

```
140 DATA 217,58,A(1,1),A(1,2),230,7
150 DATA 71, 58, A(1,1), A(1,2), 203, 63
160 DATA 203, 63, 203, 63, 79, 217
170 DATA 42, A(5,1), A(5,2), 17, 0, 91
180 DATA 58, A(+,1), A(3,2), 79, 6, 0
190 DATA 203, 176, 34, A(5,1), A(5,2), 235
200 DATA 54, 0, 58, A(3,1), A(3,2), 60
210 DATA 79, 217, 120, 217, 246, 0
220 DATA 40, 12, 65, 33, 0, 91
230 DATA 203, 30, 35, 16, 251, 61
240 DATA 32, 244, 58, A(2,1), A(2,2), 71
250 DATA 203, 7, 203, 7, 230, 224
260 DATA 111, 62, 7, 160, 79, 203
270 DATA 8, 203, 8, 203, 8, 62
280 DATA 24, 160, 177, 103, 1,0
290 DATA 64, 9, 217, 121, 217, 79
300 DATA 6, 0, 9, 58, A(2,1), A(2,2)
310 DATA 60, 50, A(2,1), A(2,2), 17, 0
320 DATA 91, 58, A(3,1), A(3,2), 60, 71
330 DATA 26, 174, 119, 35, 19, 16
340 DATA 249, 58, A(4,1), A(4,2), 61, 50
350 DATA A(4,1), A(4,2), 32, 144, 58,
    A(3,1)
360 DATA A(3,2), 60, 71, 33, 0, 91
370 DATA 54, 0, 35, 16, 251, 201
```

```
    EXX
    LD      A, (AD1)
    AND     7
    LD      B,A
    LD      A, (AD1)
    SRL     A
    SRL     A
    SRL     A
    LD      C,A
    EXX
G5  LD      HL, (AD5)
    LD      DE, 23296
    LD      A, (AD3)
    LD      C,A
    LD      B,0
    LDIR
    LD      (AD5), HL
    EX      DE, HL
    LD      (HL), 0
    LD      A, (AD3)
    INC     A
    LD      C,A
    EXX
    LD      A, B
    EXX
    OR      0
    JR      Z,G1
G3  LD      B,C
    LD      HL, 23296
G2  RR      (HL)
    INC     HL
    DJNZ    G2
    DEC     A
    JR      NZ, G3
G1  LD      A, (AD2)
    LD      B,A
    RLC     A
    RLC     A
    AND     224
    LD      L,A
    LD      A, 7
    AND     B
    LD      C,A
    RRC     B
    RRC     B
    RRC     B
    LD      A, 24
    AND     B
    OR      C
    LD      H,A
    LD      BC, 16384
    ADD     HL, BC
    EXX
    LD      A,C
    EXX
    LD      C, A
    LD      B, 0
    ADD     HL, BC
    LD      A, (AD2)
    IVC     A
    LD      (AD2),A
    LD      DE, 23296
    LD      A, (AD3)
    INC     A
    LD      B, A
G4  LD      A, (DE)
    XOR     (HL)
    LD      (HL),A
    INC     HL
    INC     DE
    DJNZ    G4
    LD      A, (AD4)
    DEC     A
    LD      (AD4),A
    JR      NZ,G5
    LD      A, (AD3)
    INC     A
    LD      B,A
    LD      HL, 23296
G6  LD      (HL), 0
    INC     HL
    DJNZ    G6
    RET
```

Programlista-kódok

A programlisták jobb olvashatósága érdekében az eredeti listázó program által használt grafikus jeleket a továbbiakban minden programlistán az alábbi, szögletes zárójelek közé tett emlékeztető kódok helyettesítik. Ezek a kódok általában megfelelnek a billentyűn is használt kódoknak. A program beírásakor az ilyen zárójelben lévő kifejezések helyett a következő billentyűket kell használni. (A szakirodalom a szögletes zárójelek helyett kisebb-nagyobb jeleket használ.)

ZSOM BÉLA

Kód	Beírandó	Jelentés	
[BLK]	CTRL+1	Black	fekete
[BLU]	CTRL+7	Blue	Kék
[BROWN]	C= +2	Brown	Barna
[CD]	CRSR DOWN		Cursor le
[CL]	CRSR LEFT		Cursor balra
[CLR]	CLR	Clear	Képernyő törlés
[CR]	CRSR RIGHT		Cursor jobbra
[CU]	CRSR UP		Cursor fel
[CYN]	CTRL+4	CYAN	Cián kék
[DEL]	DEL	Delete	Törlés
[F1]-[F8]	F1-F8		F1-F8
[GRN]	CTRL+6	Green	Zöld
[GREY 1]	C= +4	Grey 1	Szürke 1
[GREY 2]	C= +5	Grey 2	Szürke 2
[GREY 3]	C= +8	Grey 3	Szürke 3
[HOME]	HOME	CRSR Home	Cursor kezdő pozíció
[INST]	INST	Insert	Beszúrás
[LT.BLU]	C= +6	Light blue	világoskék
[LT.GRN]	C= +7	L.Green	Világoszöld
[LT.RED]	C= +3	Light red	Világos Piros
[ORANGE]	C= +1	Orange	Narancs
[PUR]	CTRL+5	Purple	Lila
[RED]	CTRL+3	Red	Piros
[RVO]	CTRL+0	RVS OFF	Invers ki
[RVS]	CTRL+9	RVS ON	Invers be
[WHT]	CTRL+2	White	Fehér
[YEL]	CTRL+8	Yellow	Sárga

... egy könyvet

(Webster: Introduction to PASCAL. Heyden and Son Ltd., 1976.),

mely mindössze 129 oldalon jó bevezetést ad a PASCAL nyelvbe. A könyv végén függelékként megtalálható a PASCAL nyelv metanyelvű leírása. Erről jutott eszünkbe: *mi is a metanyelv?* Azoknak, akiknek érdeklődését ez a kis írás felkelti, ismét csak Wirth számítástechnikai alapműnek számító könyvét ajánljuk (Algoritmusok + Adatstruktúrák = Programok. Bp. Műszaki Könyvkiadó, 1982.).

Idegen nyelveket általában egy másik nyelven szoktak leírni, szótárak, nyelvtankönyvek, olvasókönyvek, feladatgyűjtemények formájában. Így egy-egy idegen nyelvet leíró könyvek gyakran akár egész könyvtárat tehetnek ki, és mégsem írják le pontosan, minden részletében az idegen nyelvet.

A számítógépek programozására kitalált mesterséges nyelvek egyrészt jóval egyszerűbbek, másrészt lényegesen szigorúbbak mint az élő vagy hajdani természetes nyelvek. Mivel a számítógépek nem elnézőek a nyelvtani (szintaktikai) és jelentéstani (szemantikai) hibákkal szemben, ezért a programozási nyelveket lényegesen pontosabban kell definiálni, leírni, mint a természetes nyelveket. Sőt, a „lényegesen pontosabb" nem is helyes kifejezés: a számítógépek programnyelvét *egészen, minden részletében pontosan* kell leírni.

Ha a programozási nyelvek leírásához természetes nyelveket használunk, akkor a leírás igen terjedelmes és helyenként felettehetően még mindig pontatlan lesz. Ezért a programozási nyelvek definiálására, leírására ún. metanyelveket fejlesztettek ki, melyek formalizmusa hasonló a matematikai összefüggések leírására használt formanyelvhez.

A metanyelvekkel szemben támasztott alapvető követelmény, hogy egyszerűbbek és tömörebbek legyenek, mint azok a nyelvek, amelyeket segítségükkel le akarunk írni — hiszen nem mennénk sokra azzal, ha egy bonyolult programozási nyelv leírásához egy még bonyolultabb metanyelvet használnánk.

Ezeket az egyszerűbb metanyelveket is le kell azonban írni, meg kell értetni valahogy a használójával. A metanyelvek leírására is lehetne elvben egy „meta-metanyelvet" stb. alkotni — vannak is ilyenek —, de a programozási nyelvek leírására használt metanyelvek már annyira egyszerűek, hogy általában néhány egyezményes jelölés és a hozzá fűzött magyar, angol stb. nyelvű magyarázat elegendő hozzájuk.

Metanyelvet természetesen nemcsak a számítástechnika, hanem a hagyományos nyelvészet is használhat. Az idegen szavak szótára a metanyelvet a következőképpen definiálja: „metanyelv a vizsgálat tárgyát képező nyelv valamennyi elemét magasabb logikai szinten *egyértelműen* meghatározó fogalomrendszer". A „meta" görög eredetű előtag itt azt jelenti, hogy valamilyen „nyelven túli" nyelvről van szó. Ismeretes például a „metakritika" kifejezés, melyen a kritika kritikáját értik — tehát a kritikáról írt kritikát.

El kell ismerni, hogy ez eddig elég bonyolultan hangzott. A számítástechnikával foglalkozók a metanyelvek használatát azonban nehezen tudják kikerülni: előbbutóbb beleütköznek abba, hogy a programozási nyelveket ezek segítségével kell megérteniök. A kezdők persze ma általában először a BASIC programozási nyelvvel ismerkednek meg, mégpedig úgy, hogy elolvassák a BASIC-et valamilyen számukra érthető, természetes nyelven leíró könyvet. Az ilyen könyvek rendszerint nagyon vastagok, mégis pontatlanok, ismétlésekkel vannak teli és sokszor csak nehezen áttekinthetők.

A programozási nyelvek leírására ma az ún. BNF: *BACKUS—NAUR—FORM* a legelterjedtebb metanyelv, melyet két kifejlesztőjéről neveztek el. Lássunk most néhány példát az egész technika megvilágítására: némi bevezető után „alkossunk" és írjunk le mesterséges nyelveket. Nevezzük el ezeket — mivel a gyakorlatban semmire sem lesznek jók — mondjuk „KUKA" nyelveknek.

Minden nyelv alapja a SZÓTÁR, melynek elemeit általában SZAVAK-nak nevezzük. Ami a szó a beszélt nyelvekben, az a SZIMBÓLUM a programozási nyelvekben. A nyelvek tulajdonsága, hogy a SZAVAK (SZIMBÓLUMOK) bizonyos sorát, egymásutánját helyesnek, más sorát pedig helytelennek, hibásnak lehet tekinteni.

— „A kutya alszik." Magyarul szintaktikailag és szemantikailag egyaránt helyes mondat.

— „A kutya kukorékol." Szintaktikailag nem kifogásolható, szemantikailag viszont hibás.

— „Látok a kutya." Szintaktikailag hibás mondat.

A nyelvtan az, ami eldönthetővé teszi, hogy a nyelv szavainak valamilyen egymásutánja (a mondat) helyes-e.

A nyelvtan, a szintaxis nem más, mint szabálygyűjtemény. Próbáljunk BNF-metanyelven definiálni egy ilyen egyszerű, valójában csak a leírási mód érzékeltetésére való nyelvet az alábbiak szerint:

(1) <mondat> ::= <alany> <állítmány>.

(2) <alany> ::= a kutya | a macska

(3) <állítmány> ::= alszik | eszik

Az (1) összefüggést úgy kell olvasni, hogy kitalált nyelvünkben a mondat mindig alannyal kezdődik, melyet kötelezően állítmány követ, és a mondat mindig ponttal („.") zárul. Az összefüggés azt mondja, hogy e nyelvben az alany kétfajta lehet: vagy a „kutya", vagy a „macska".

Az (1), (2) és (3) összefüggéssel definiált nyelv, amit nevezzünk el KUKA nyelvnek, nagyon primitív. Mindössze négy darab szintaktikailag helyes mondat állítható összse benne:

a) A kutya alszik.
b) A kutya eszik.
c) A macska alszik.
d) A macska eszik.

Nem helyes mondatok ebben a KUKA nyelvben például:

— A kutya a macska.
— Alszik eszik.
— A macska.
— Eszik a macska.
— Alszik a kutya.

A BNF-ben megadott (1) összefüggés szerint a KUKA nyelvben a mondat alannyal kezdődik, utána kötelezően állítmány következik, és a mondat végén kötelező kitenni a pontot.

Definiáljunk most BNF segítségével egy módosított KUKA nyelvet. Nevezzük ezt KUKA—2 nyelvnek. A továbbiakban az (1) összefüggés helyett legyen érvényes az alábbi:

(4) <mondat> ::= { <alany> <állítmány> | <állítmány> <alany> }

A KUKA—2 mesterséges nyelv is igen primitív. Szintaxisát tehát a (4), (2) és (3) összefüggés írja le. A kapcsos zárójel — mint erre még visszatérünk — a BNF-ben azt jelenti, hogy az egymás alatt álló szerkezetek közül egyet választhatok.

A (4) összefüggést úgy kell olvasni, hogy ebben a nyelvben kétféle felépítésű mondat lehetséges: olyan, amely alannyal kezdődik, amit állítmány követ, vagy olyan, amely állítmánnyal kezdődik, amit alany követ. Mindkét mondat kötelezően ponttal zárul.

A három összefüggés (4), (2), (3) szerint a KUKA—2 nyelven a kiinduló KUKA nyelvhez képest további négy helyes mondatot írhatunk le:

e) Alszik a kutya.
f) Eszik a kutya.
g) Alszik a macska.
h) Eszik a macska.

A KUKA—2 általunk kitalált nyelven tehát összesen nyolc helyes mondat írható le.

Ha most a (4) és az (1) helyett az alábbi (5) összefüggést léptetem életbe:

(5) <mondat> ::=[<alany>] <állítmány>.

akkor ezen a — mondjuk KUKA—3 — nyelven az első négy helyes mondat, az a), b), c) és d) köre az alábbi két mondattal bővül:

i) Eszik.
j) Alszik.

A KUKA—3 nyelven tehát összesen hat szintaktikailag helyes mondat írható le.

A szögletes zárójel a BNF-ben azt jelenti ugyanis, hogy a benne foglalt szintaktikai elemet ha akarom odaírom, ha nem akarom, akkor nem. Az (5) összefüggés tehát így olvasandó: a KUKA—3 nyelven a mondat mindenképpen állítmányt tartalmaz és ponttal zárul. Az állítmány előtt áll*hat* alany.

Gyakorlatképpen definiáljunk még egy újabb, mondjuk KUKA—4 nyelvet így:

(6) <mondat> ::=

::= { <helyhatározó> | <kérdőszó> } van {.|?}

(7) <helyhatározó> ::= itt | ott

(8) <kérdőszó> ::= hol | merre

A (6), (7) és (8) összefüggésekkel definiált, KUKA—4-nek nevezett nyelven az alábbi, szintaktikailag helyes mondatok írhatók le:

k) Itt van.
l) Ott van.
m) Hol van?
n) Merre van?
o) Itt van?
p) Ott van?
q) Hol van.
r) Merre van.

Az utolsó két mondat KUKA—4 nyelven szintaktikailag helyes, mert megfelel a (6), (7), (8) előírásnak, annak ellenére, hogy a mondat végére kérdőjel kívánkozna.

A fenti példákban néhány nagyon egyszerű „nyelvet" definiáltunk BNF-ben: a KUKA, a KUKA—2, a KUKA—3 és a KUKA—4 nevű mesterséges, semmi gyakorlati célra nem jó nyelveket.

Most elnézést kérve azoktól, akik úgy érezték, hogy jobb lett volna az előre közölt, a fantázia megmozgatását célzó példák helyett előbb a BNF formalizmusát, jelölésrendszerét röviden bemutatni, pótoljuk ezt a hiányt. A BNF korrekt leírása mintegy két nyomtatott oldalon elfér. Ennyi helyünk itt nincs, ezért csak a legfontosabb, a fenti példák megértéséhez szükséges jelölésekre térünk vissza.

— A két kettőspontot követő egyenlőségjel ("::=") a BNF definíciós jele. A jel bal oldalán lévő fogalmat meghatározom mindazzal, ami a jobb oldalon van.

— A hegyes zárójelek között ("< >") a később definiálandó fogalmak neve áll.

— Függőleges vonallal ("|") választjuk el a nyelv mindazon elemeit, melyek bármelyike megfelel a definíciós összefüggés bal oldalán található fogalomnak.

— A kapcsos zárójel ("{ }") azt jelenti, hogy a benne egymás alá írt szerkezetek közül egyet választhatunk.

— A szögletes zárójel („[ ]") azt jelenti, hogy az abban foglalt szerkezeti elem választásunk szerint vagy van, vagy nincs.

Nézzünk most egy számítástechnikai példát! Minden programozási nyelv egyik legfontosabb része a nyelv alapvető, tovább nem bontható elemeinek, a karakterkészletnek a definiálása, például így:

(1) számjegy::=Ø | 1 | 2 | 3 | 4
| 5 | 6 | 7 | 8 | 9 |

(2) betű::= A | B | C | D | E | F | G | H |
| I | J | K | L | M | N | O | P |
| R | S | T | U | V | Z | X
| Y |
| W |

(3) különleges karakter::=$ | % | @ | ↑ | < |
= | . | , | : | * | : |
| + | − |

(4) karakter::=

::= { <számjegy> | <betű> | <különleges karakter> }

(5) változó név::=

::=<betű>[{ <betű> | <számjegy> }] [{$ | %}]

(6) pozitív egész szám::=

::=<számjegy>[. . .]

Ez eddig persze csak egy töredéke valamely nyelv leírásának. Nézzük meg, hogyan kell az egyes sorokat olvasni:

ad 1. Az (1) nem mást mond, mint azt, hogy az adott programozási nyelvben számjegyként a Ø-t vagy az 1-et vagy a 2-t stb. értelmezzük — egészen 9-ig. Emlékeztetünk rá: a „::=" jel a BNF definíciós jel. Az „egyenletet" úgy kell olvasni, hogy ami a bal oldalon van, mindaz lehet, ami a jobb oldalon fel van sorolva. A felsorolás elemeit függőleges vonalak választják el egymástól.

Tehát *számjegy* lehet a Ø, az 1, a 2 stb.

De: számjegy pl. *nem lehet* „Ø2" vagy „%" stb.

A függőleges vonal értelme: „vagy". „Vagy ez lehet, vagy az", de például kettő vagy három stb. egyszerre nem lehet.

ad 2. A (2) összefüggés ugyanezt mondja a betűkre. Itt meghatároztuk, hogy mi lehet betű. Például a „H" betű, de „h" az itt leírt nyelvben *nem* betű. A példában leírt nyelv tehát például *nem* a BASIC, mert a BASIC megengedi a kisbetűk használatát is.

ad 3. Mint (1) és (2). De a példából szándékosan elhagytuk a törtvonalat (a „/"-t). Az itt leírt nyelvben például az osztást nem lehet majd törtvonallal jelölni, mást kell kitalálni helyette.

ad 4. Ez valami új. A kapcsos zárójel a BNF-ben azt mondja, hogy a karakter definíciószerűen vagy számjegy, vagy betű, vagy különleges karakter. Itt tehát ezeket együttesen nevezem összefoglaló néven karakternek. Eszerint karakter lehet például „C", „%", „8", de nem lehet karakter az „É" vagy a szögletes zárójel, a „[".

ad 5. Ez a definíció Commodore 64 BASIC-szerű. Azt mondja (úgy kell olvasni), hogy a változónév mindig betűvel kezdődik, melyet vagy követ, vagy nem egy betű, vagy egy számjegy és ezt vagy követi, vagy nem egy „$"-jel vagy egy „%"-jel.

Mindaz, ami szögletes zárójelben áll, ha akarom van, ha akarom nincs. A szögletes zárójelben lévő kapcsos zárójel ugyanazt jelenti, mint előbb; az egymás alatt álló dolgokból vagy az egyik van, vagy a másik — a kettő együtt nem állhat. Például (5) szerint változónév lehet D5$, F%, W9, AA stb. De nem lehet például 5, 5A, %C, $7 stb.

ad 6. Itt megint valami új BNF-fogás (szabály) van. A szögletes zárójelbe tett három pont azt jelenti, hogy az az előtt álló valamit tetszés szerint megismételhetem — beleértve a Ø-szori megismétlést is. Tehát pozitív egész szám ebben a példanyelvben lehet Ø, 11, 12Ø77, ØØØØ, Ø2 stb., de nem lehet például +1, 2*3, −1356 stb.

Biztosan lesznek olyanok, akiknek az olvasás közben már eszébe jutott, hogy a metanyelveket fel lehetne használni fordítóprogramok (compilerek, assemblerek) generálására. Egy ilyen generálóprogram bemenete lehetne a nyelv leírása mondjuk BNF-ben és kimenete az a fordítóprogram, amellyel a BNF-ben definiált nyelven megírt programot fordíthatom például gépi kódra. Ennek a kérdésnek a további boncolgatása azonban már túl messze vezetne.

— KE —

A Commodore Plus/4

Mint ismeretes, a Tudományszervezési és Informatikai Intézet iskolaszámítógépek szállítására kiírt pályázatának általános iskolai kategóriájában megosztott első helyen végzett a Commodore 16 számítógép. Mivel azonban ezt a gépet már csak elvétve, kisebb tételekben lehet beszerezni, a korábbi bírálóbizottság a C16-tal program szempontjából teljesen kompatibilis, de a 60 kbájt szabad memóriával rendelkező Commodore Plus/4-et elfogadta helyettesítő gépnek. A szállítók a C16 korábban megállapodott áráért vagy annál még olcsóbban fogják a gépet a Tudományszervezési és Informatikai Intézeten keresztül az iskolák rendelkezésére bocsátani.

PLUS/4

A gyártó Commodore cég a gépet a C64 utódjának szánta. Az utódlás megkönnyítésére igyekezett a C64 perifériaválasztékát átmenteni, ezért a leglényegesebbnek tartott eszközöket, a lemezegységet és a nyomtatót változatlan formában használhatóvá tette. A többi perifériára ez nem vonatkozik, és nem vonatkozik a programok legnagyobb részére sem. Valószínűleg ez utóbbi okozta azt, hogy a cég elgondolása nem vált be.

A régebbi Commodore gépek külső kialakításával szakítva, jóval korszerűbb és célszerűbb formát kapott (lásd az *1. képet*). A *2.* és *3. képen* oldalnézetben látható, hogy perifériaválasztéka azonos az eddig használt Commodore gépeknél megszokottakkal (a VC20 és C64 analóg bemenetei hiányoznak).

A BILLENTYŰZET

A billentyűk formája „ujjhoz simuló", ezért használatuk is könnyebb. Előnyös megoldás, hogy a gyakran használt billentyűket az írógépeknél megszokottól eltérő módon helyezték el, és a *, =, + jelek stb. betűváltó nélkül használhatók. Célszerű megoldás volt az is, hogy a kurzormozgató billentyűket egy csoportba helyezték, olyan formát kialakítva, hogy a billentyűkön elhelyezett nagyméretű nyilak közvetlenül mutatják a billentyű lenyomásának hatására bekövetkező kurzormozgatási irányt.

1. kép

A képernyőt szerkesztő billentyűk (INST/DEL, CLR/HOME) és a módosító (SHIFT, CONTROL, C=) billentyűk elősegítik a képernyőn a javítások megvalósítását. Ez utóbbiak segítségével ún. grafikus karaktereket lehet előállítani, változtatni lehet a színeket, villogtatni a karaktereket (FLASH), lehet listázni és nyomtatást leállítani, szövegüzemmódot beállítani. Az ESC billentyű és 18 betű kombinációja többek között lehetővé teszi, hogy a képernyőn kijelölt részterülettel, az úgynevezett ablakkal tudunk dolgozni: kijelölni, törölni, azon belül képernyőszerkesztő műveleteket végrehajtani, mozgatni. Igen célszerűen használhatók az előre programozott függvénybillentyűk: ezekkel lehet a beépített programokat bekapcsolni, lemezparancsokat kiadni, képernyőt törölni, RUN és LIST indítást megvalósítani, hibahelyet jeleztetni.

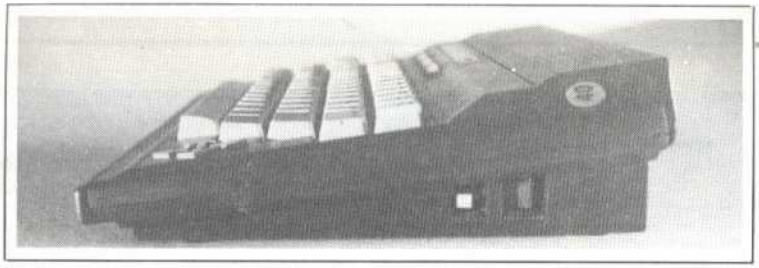
A betűkészlet tartalmazza a teljes magyar ábécét, de ahogy már korábbi cikkeimben is ismertettem, a 8×8-as betűmátrix miatt ez bizonyos problémákat okoz. Ezek közül a legjelentősebb az, hogy az összenyomott betűk használatának ellenére a hosszú ékezetes betűk, valamint az f, l, t összeérnek a felettük levő g, j, y betűkkel, és olvashatatlanná folynak össze. Ha pedig p alá Ő, Ű kerül, eltűnik az első ékezet, és itt például az Ő helyett Ó lesz.

A CSATLAKOZÓK

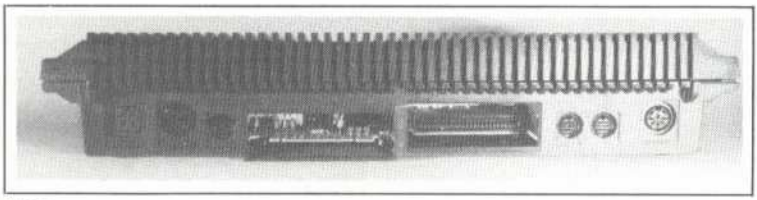
Már említettem, hogy a csatlakozók megfelelnek a korábbi Commodore gépeknél megszokottaknak. Kialakításuk olyan, hogy nem cserélhetők össze, és így nem okozhatják a gép meghibásodását. Egy részük (a magnetofon, a botkormány, a tápegység kivételével a többi) megegyezik a C64-ével. Sajnos, a magnetofon és a botkormány mini DIN típusú, más cégek nem gyártják, így ára lehetetlenül magas. Például csak e három csatlakozó ára mintegy 30%-a (!!) a C16 gép árának.

A gép sokkal masszívabb, professzionálisabb mechanikai felépítésű, mint akár a C16, akár a C64.

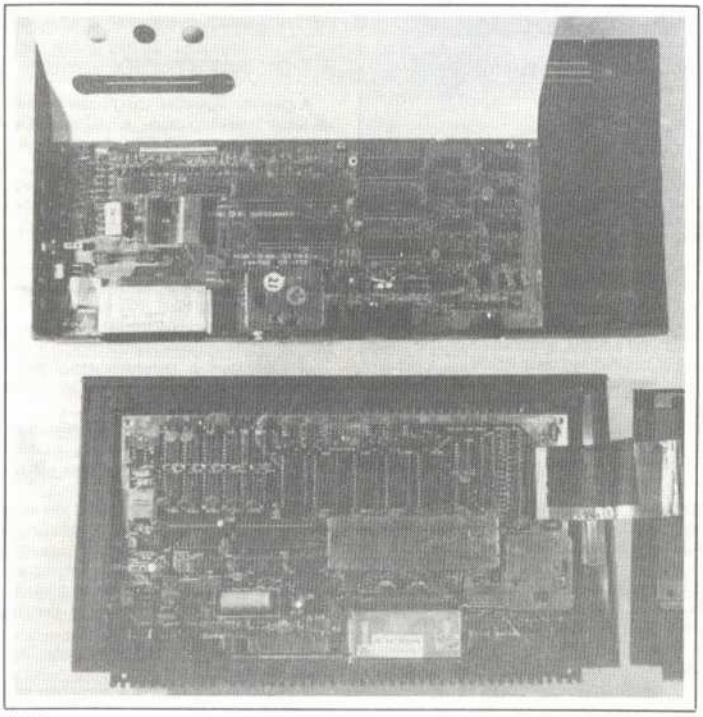
Oldalán található a csatlakozókon és a ki/be kapcsolókon kívül a RESET gomb; segít-



2. kép



3. kép



4. kép

ségével történhet a BASIC „hideg" és „meleg" indítása is. (A kétféle indítás a „törlés" milyenségében különbözik.)

A Plus/4 gépek mind az amerikai, mind a nyugat-európai tévék használatára alkalmasak: két változatban készültek. Itthon csak a PAL rendszerű, illetve a kétnormás tévék adnak színes képet és hangot. A legolcsóbb és igen elterjedt szovjet televíziók nem alkalmasak a gép színes képének előállítására, mivel csak SECAM rendszerűek.

Egy Elektronika C—401 típusú gépen sikerült elérnem, hogy a keret barnásvörös, a háttér zöld, a betűk sötétkékek voltak, illetve bizonyos idő múlva automatikusan átváltódtak szürkére, sárgára, feketére. Ezt minden alkalommal reprodukálni tudtam, de a színes kép stabilitása meglehetősen gyenge volt, elég könnyen átugrott a második színkombinációra, és onnan már csak újrahangolással lehetett az első állapotba vinni.

Az 1. képen is látható a gép felső oldalán hátul elhelyezett nagyméretű hűtőbordázat, melynek köszönhetően a gép 120 órás folyamatos üzemelés alatt is alig langyosodott.

A géphez viszonylag nagy teljesítményű tápegység tartozik, melynek felvett teljesítménye 46 VA (a C16 tápegységénél ez az érték 14 VA). A tápegység a 120 órás vizsgálat során sem melegedett fel annyira, hogy forrónak lett volna érezhető. Itt jegyzem meg, hogy valamennyi tapasztalatomat egyetlen gép vizsgálatával szereztem.

A BELSŐ FELÉPÍTÉS

A *4. képen* látható a C16 és a Plus/4 belseje. A vizsgált gép lényegileg egykártyás — a második kártya a billentyűzetkezelő.

A már korábbi cikkeimben is elemzett betűmátrix-problémákra nem kívánok újra visszatérni, mert véleményem szerint azokat már kellő részletességgel tárgyaltam.

A gyártó két, speciálisan erre a két géptípusra tervezett alkatrészt használ. Az egyik egy 6502 típusú mikroprocesszorral kompatíbilis, és különböző kiegészítő egységeket tartalmazó mikroprocesszor (típusszáma 8501), a másik a perifériák kezelésére szolgáló, ún. TED (típusszáma 8360). E két alkatrészt más cég nem gyártja, kiskereskedelmi forgalomban — ez nemcsak Magyarországra vonatkozik — nem szerezhető be. A forgalmazó — ahogy ezt az iskolaszámítógép-pályázat eredményéről szóló cikkünkben is leírtuk — vállalta, hogy gondoskodik a tartalék alkatrészekről. E speciális alkatrészeken kívül további, csak ennél a két gépnél használt alkatrészek is vannak, például a párhuzamos csatornákat kezelő 6529-es típusú és a BASIC—ROM (típusszáma 318006). A másik ROM IC a két géptípusnál különböző. Ennek a gépnek a specialitása a 8551 típusszámú perifériakezelő és a beépített programokat tartalmazó két ROM (ez utóbbiak típusszáma 317053, 317054). A programozható memória számára szolgáló dinamikus RAM IC a géptípus kifejlesztésének idején, két évvel ezelőtt, Nyugaton általánosan használt 4164 típus.

A további alkatrészek a karaktergenerátor ROM és a 9 egyéb integrált áramkör, valamint a dekóder rész. Összehasonlítva a TV-Computer ismertetőjével (1986. 5. szám), látható, hogy az alkatrészválaszték különbözősége miatt ez a gép sokkal kevesebb alkatrészt tartalmaz.

A kazettaegység a Commodore gépek egyik általános gyenge pontja. Speciális magnetofont igényel, és átviteli sebessége nagyon alacsony. Ezen segíthet például a SUPERTAPE-program és a hardvermódosítás, amellyel mintegy 12—24-szeres sebesség érhető el, és lehetségessé válik közönséges magnetofon használata is.

```
10 FOR I=1000001 TO 1000003 STEP 2
20 FOR D=3 TO SQR(I) STEP 2
30 IF I/D=INT(I/D) THEN 60
40 NEXT D
50 PRINT I
60 NEXT I
```

1. lista

```
10 PMODE 4,1: PCLS: SCREEN 1,1: REM EL
OEKEESZITEES
20 DIM A(10): DRAW"BM 120,96;U10L10D10
10": GET(105,81)-(125,101),A,G: REM RA
ZOLAAS, ELTAAROLAAS
30 FOR I=1 TO 33: FOR J=1 TO 33: PUT(I
,J)-(I+20,J+20),A,PSET: NEXT J: NEXT I
```

3. lista

Előnye a gépnek, hogy jól használható képernyőszerkesztője van. Ha azonban a sorban ott marad valami nem kívánt rész, az is belekerülhet a programba, aminek elkerülésére szolgál az ESC Q; hatására a kurzortól jobbra levő rész törlődik.

A GÉPPEL EGYÜTT SZÁLLÍTOTT PROGRAMOK

Eltérően a belföldi forgalomban levő gépektől, ROM-ba rögzítetten szövegszerkesztő, adatbázis-kezelő, táblázatkezelő és grafikus programcsomag is a gép beépített tartozéka. Ezek használatával azonban e cikk keretében nem tudunk foglalkozni, de külön cikkben visszatérünk rá.

A BASIC

A gép BASIC-változatát a Microsoft cég készítette. Sokkal kedvezőbb tulajdonságú, mint az ugyancsak e cég által készített C64 változat. Felsorolom a szerintem előnyös tulajdonságokat: az automatikus sorszámozás lehetősége (AUTO parancs); a nem kívánt sorok/sortartományok törlésének lehetősége (DELETE parancs); a függvénygombok jelentésének programozhatósága maximum 128 karakteres sorozatig adható értelmezéssel (KEY utasítás); az újraszámozás lehetősége (RENUMBER parancs); a különböző lemezegység-vezérlő parancsok/utasítások, melyekkel nem foglalkozom részletesen, mert az iskolákba magnetofonos készülékek kerülnek; tetszőleges helyzetű négyszöget rajzoló utasítás (BOX); a rajzszöveg keverhetősége a képernyőn (CHAR utasítás); a kör/ellipszis/sokszög, illetve ezek részleteit rajzoló és forgató utasítás megléte (CIRCLE); a vonal/vonalsorozat rajzolásnál a relatív derékszögű és polár koordináták használatának lehetősége (DRAW és : illetve előjel); a formált kiíratás lehetősége (PRINT USING utasítás) és az így kiírt jelek újraprogramozhatósága (PUDEF utasítás); a RESTORE sorszám utasítás létezése; a hiba esetén folytatás lehetősége (RESUME utasítás) a sor kihagyásával, illetve adott helyen történő folytatással; a hibakezelés lehetősége (TRAP utasítás); a futás követésének lehetősége (TRON utasítás), a számkonvertálás lehetősége (DEC, HEX$), a részkarakter-sorozat kereső utasítás megléte (INSTR); az egésztípusú változók használata; hogy a Plus/4 angol könyv néhány nem szokványos (például Sinclair típusú) BASIC-átalakítás módját is tárgyalja, valamint hogy a MID$ utasítás az összehasonlítás bal oldalán is használható.

```
1 SI=7000
2 DIM FL(7001)
3 PRINT"ONLY 1 ITERATION"
5 CO=0
6 FOR I=0 TO SI
7 FL(I)=1
8 NEXT I
9 FOR I=0 TO SI
10 IF FL(I)=0 THEN 18
11 PR=I+I+3
12 K=I+PR
13 IF K>SI THEN 17
14 FL(K)=0
15 K=K+PR
16 GOTO 13
17 CO=CO+1
18 NEXT I
19 PRINT CO,"PRIMES"
```

2. lista

Részben a gép, részben a BASIC hátránya, hogy a hanggenerátor meglehetősen pontatlan, és hogy hangkeltésnél nem a zenei skála hangjait vagy a frekvenciát kell megadni, hanem egy táblázatból kikeresett értéket (SOUND), ami a C16 magyar kézikönyvének 144. oldalán levő képlet szerint csak nagyon pontatlanul határozza meg a hangok magasságát, még a hallható tartományban is (például kb. 10-11 kHz esetén 1013 ez az érték), a zenei skála előállítása bonyolult. Például egy oktáv előállítása:

10 VOL 8: FOR I=0 TO 7: READ X: SOUND 1,X,10: NEXT X

20 DATA 812, 836, 856, 865, 882, 898, 912, 918

ugyanez a Dragonnál:

10 PLAY "L2T30V3102CDEFGAB03C"

ahol L a hanghossz, T a tempó, V a hangerő, 0 melyik oktáv.

Külön kiemelendő előnynek tartom a strukturált nyelvi szerkezeteket: a DO UNTIL (feltétel) vagy WHILE (feltétel) . . . EXIT LOOP UNTIL (feltétel) vagy WHILE (feltétel) utasításokat. Ez egyébként azt jelenti, hogy ha az UNTIL áll az adott helyen, addig ismétli, amíg nem igaz, illetve ha WHILE áll az adott helyen, addig ismétli, amíg igaz.

Gép és BASIC	Idő (s)
C64, beépített BASIC	255
Apple II+, Applesoft	230
IBM PC, BASICA	190
IBM PC, ZBASIC	1
Plus/4	325
Dragon, beépített BASIC	280
Dragon gyors, beépített BASIC	135

A prímszámteszt-eredmények táblázata

A DOKUMENTÁCIÓ

A géphez egyelőre még csak angol nyelvű kézikönyvet kaptam. A tájékoztatás szerint a magyar nyelvű kézikönyv ennek pontos fordítása lesz. A kézikönyv első fejezetében a csatlakozók, az üzembe helyezés, a hibák ismertetése található jól érthetően, kimondottan kezdők számára. A kezdők számára előnyös a sok magyarázó ábra, példa, ami a könyv további fejezeteiben is hasonló módon található. Az ún. BASIC Enciklopédia szintén érthetően, világosan ismerteti a parancsokat, utasításokat, függvényeket, változókat és műveleteket.

A könyvben a keresést megkönnyíti a kulcsszavak és egyéb tárgyszavak ábécérendbe szedett mutatója.

A könyv tárgyalja a gépbe beépített gépi kódú monitort is. Remélhetőleg, a C16 kézikönyvétől eltérően, leírása a magyar nyelvű kézikönyvben is benne lesz!

A rendelkezésemre álló rövid idő ellenére néhány tévedést is találtam a kézikönyvben. Így például a GSHAPE/SSHAPE utasításpár ismertetésénél szereplő példa mind a C16 (angol és magyar), mind a Plus/4 kézikönyvében hibás. A hiba az, hogy a karakteres változóként megadott

"ADAT$" helyett vagy A$ vagy "ADAT" állhat csak.

A PRINT # utasításnál a leírásban megtévesztő a második # (a C16 magyar kézikönyvének 110. oldalán ugyanez található).

Abban is bízom, hogy átkerül a magyar fordításba is az egyes részeknél a részletes memóriatérkép és a soros csatorna ismertetése is.

A VIZSGÁLÓPROGRAMOK

A TV-Computer termékismertetőjénél még csak egyetlen vizsgálóprogram-csoporttal történt a vizsgálat. Ugyanezekkel a

```
10 DIM A$(40): GRAPHIC 1,1: REM ELOEKEESZITEES
15 BOX 1,20,20,120,120: REM RAJZOLAAS
20 FOR I=0 TO 4: FOR J=0 TO 7: SSHAPE A$(K),I*40,J*40,(I+1)*40,(J+1)*40
30 GSHAPE A$(K),I*40,J*40: K=K+1: NEXT J: NEXT I
```

4. lista

```
10 PCLEAR 8: PMODE 4,5: PCLS: PMODE 4,1
:PCLS: SCREEN 1,1: REM ELOEKEESZITEES
15 DRAW"BM 120,120;U100L100
D100R100": REM RAJZOLAAS
19 POKE 65497,0
20 FOR I=1 TO 1E3: PMODE 4,5: SCREEN 1
,1: PMODE 4,1: SCREEN 1,1: NEXT I
21 POKE 65496,0: POKE 65494,0
```

5. lista

```
10 DIM A$(40): GRAPHIC 1,1: REM ELOEKEESZITEES
15 BOX 1,105,81,125,101: SSHAPE A$,105,81,125,101
20 FOR I=1 TO 33: FOR J=1 TO 33: GSHAPE A$,I,J,0: NEXT J: NEXT I
30 GSHAPE A$(K),I*40,J*40: K=K+1: NEXT J: NEXT I
```

6. lista

programokkal a gép gyorsaságának — pontosabban a gép és a BASIC együttes gyorsaságának — ellenőrzését elvégeztem. Az eredmények gyakorlatilag azonosak voltak az ÖTLET-ben a C16 típusú gépre közölt értékekkel. Ezt a vizsgálatot kiterjesztettem egyrészt az irodalomban használt egyéb, másrészt néhány általam készített programra.

A HCC NEWSLETTER-teszt *(1. lista)* nem annyira a gépek és a BASIC gyorsaságát ellenőrizni, mint inkább a pontosságot. A gépnél először a belső ciklusból a 30. sorban a D=101 értéknél ugrik ki, ami helyes is, másodszor azonban D=999 értéknél, ami hibás. A helyes érték a kiugrás nélküli végigfutás lenne. Az ok, hogy a helyes

I/D=1001.004

és

INT(I/D)=1001

helyett egyenlőséget kapunk.

Így bár a futási idő azonos a helyes értékkel, a végrehajtás azonban nem. A kapott érték 8,1 másodperc, míg a Dragonnál 8,0, illetve a gyors üzemben 4,1 másodperc volt.

A *2. listában* látható ún. prímszámteszt a BYTE című szaklapból származik. Az eredmények a táblázatban találhatók. A táblázatból jól látható, hogy nemcsak a géptől függ a futás gyorsasága, hiszen az IBM PC kétféle BASIC változata között óriási eltérés adódik.

Bár a vizsgált gépkategóriába eső gépek fontos jellemzője a grafikai lehetőség, mégsem terjedtek el összehasonlítások a grafika gyorsaságának és teljesítőképességének vizsgálatára. Ennek egyik fő oka, hogy a gépek grafikai lehetőségei korábban nagyon eltérőek voltak. Megkísérlem ezt a hiányosságot részben pótolni azzal, hogy két vizsgálatot ajánlok. Az egyik a teljes képmezők eltételét, illetve elővételét vizsgálja, a másik pedig egy blokk mozgatását a képernyőn. Mind a kettő a gépek grafikájának egy-egy jellemző és igen gyakran használt mozzanatát vizsgálja. Az elsőnél a teljes képmezőt eltéve a memóriába, egy másik képmezőt vetetek elő, és ezt ezerszer megismétlem. A grafikus vizsgálatokat egyelőre még csak a Plus/4 és a Dragon géptípusokkal végeztem.

Amennyiben az olvasók ezt a módszert más gépeknél is használják, szívesen közöljük a vizsgálatokkal kapott eredményeket, ha a vizsgálatoknál használt programot is közlik.

A *3.* és *4. lista* tartalmazza a Dragon és a Plus/4 gépekre írt programokat.

A 4. listából látható, hogy itt nincs ezerszeres ismétlés, mert a képváltás sebessége ezt szükségtelenné tette. A Dragonnál a képváltásra fordított idő 16 ms (nem elírás, valóban ennyi!), gyors üzem esetén 9, míg a Plus/4 gépnél 37 s (ez sem elírás). Az utóbbi gépnél PEEK és POKE utasításokkal is meg lehet valósítani a képcserét. A módszer azonban kedvezőtlenebb eredményt adott (180 s). Más gépeknél, amelyeknél nincs hasonló grafikus utasítás, ez az egyetlen járható út. Az itt használt program a következő:

```
5 PRINT"BEADANDÓ, A KEERT KEEPTE-
  RULET KEZDOECIME EES A KEERT
  PONTOK SZAAMA"
10 INPUT A, B, C
20 FOR I=0 TO C:X=PEEK
  (A+I):Y=PEEK(B+I): POKE B+I,X:
  POKE A+I, Y: NEXT I
```

A blokkmozgatásnál a Plus/4 által elrakható legnagyobb blokkméretet megközelítő értékkel dolgoztam. Ez egy 40 egység élhosszúságú négyzet volt. A blokkot 1000 különböző helyre helyezve, a kapott átlagos időket használtam fel. A Dragon esetében *(5. lista)* ez 66 ms, míg a Plus/4 esetében *(6. lista)* 320 ms.

ÖSSZEFOGLALÁS

Egyetlen gép rövid idejű vizsgálata alapján megállapíthattam, hogy egy, az árához képest igen teljesítőképes gépet vizsgáltam. A gép jó használhatóságának azonban két olyan feltétele van, melyet tudomásom szerint még nem biztosítottak az iskolákban: a lemezegységgel és a színes monitorral való ellátás. — *Az Iskola-számítógép rovat vezetőjének, Varga Andrásnak a megjegyzése: Erről tárgyalások folynak.*

DR. SIMONYI ENDRE

Műszaki adatok:

Tápfeszültség: 9,5 V (csak saját tápegységével üzemeltethető)
Teljesítményfelvétel: max. 46 VA
Tár: 64 kbájt ROM (operációs rendszer, BASIC 3.5 interpreter, beépített programok), 64 kbájt RAM
Processzor: 8501 mikroprocesszor 1,6 MHz órajellel
Grafika: 40 oszlop × 25 sor szöveg
15 szín 8 fényerősséggel + fekete
Nagy- és kisbetűk, grafikus jelek
Finomgrafika (320 × 200 pixel) teljes BASIC támogatással
Osztott képernyő (szöveg és grafika keverve)
Hang: 2 hanggenerátor, egyik zajgenerátorként is használható, 8 hangerősségű szint
Billentyűzet: 66 billentyű, 4 programozható funkcióbillentyű, 4 kurzorbillentyű
Be/kimenet: soros interfész (lemez, nyomtató csatlakoztatásához)
Kazettaegység-csatlakozó
Modulcsatlakozó
2 botkormány-csatlakozó
Videokimenet (PAL kompozit jel)
Hang be- és kimenet (a videocsatlakozón)
Tévékimenet (kb. 36. csatorna, PAL rendszerű)
Méret: 40 × 8 × 20,5 cm
Tömeg: 1750 g

Fiatalok! Figyelem!

Folytatjuk az 1986/8., szeptemberi számunkban meghirdetett rejtvénypályázatunkat.

Az e havi feladat:

3. Két HT gépet az USER PORT-okon összekapcsolunk. A cél az egyik gép (adó) billentyűzetéről egybájtos értékek átvitele a másik gép (vevő) képernyőjére. Az átvitel párhuzamos (egyszerre egy bájt és szinkronbit). A billentyűzetről történő bevitelre az egyszerűség kedvéért használd az INPUT utasítást! Készítsd el az adó és a vevő programját! Megengedhető az a feltételezés, hogy mindig az adó programja indul először.

A beküldési határidő:
1986. november 20.
A megoldásokat írásban kérjük a következő címre:

Mikroszámítógép Magazin Szerkesztősége, Bp., Fő u. 68. IV. em. 452. 1027
A borítékra írjátok rá: DIGITÁL rejtvénypályázat.
Sok sikert kíván a szerkesztőség!

PTA–4000
ROM-leírás II.

Az előző rész a VAL utasítás szubrutincímével fejeződött be. Itt folytatjuk.

LEN
D9DD—D9F2.
RIGHT$ — LEFT$ — MID$
D9F3—DA5C.
MEM
DA5D—DA7Ø

Egyéb szubrutinok

16 bites szorzás
DA71—DAA7.
Az Ureg tartalmának kettes komplemensét előállító szubrutin
DAA8—DB86.
Az AR—X-ben levő szám formátumának vizsgálata
DB87—DBBØ.
Változó formátum vizsgálata
DBB1—DBBB.
Az Ureg felöltése paraméterrel
DBBC—DC15.
Az AR—X tartalmát áttölti a BASIC verembe
DC16—DCA5.
Egy közelebbről meg nem határozott segédrutin
DCA6—DCB5.
Szintaktikai vizsgálat a következő jel vagy kulcsszókód betöltésével együtt
DCB6*.
Ugyanez töltés nélkül
DCB7—DCC4.
Annak a vizsgálata, hogy vége van-e a sornak vagy az utasításnak
DCC5.
Mint az előző, de nem tölti be a következő karaktert
DCC6—DCD3.
A következő karakter vagy kulcsszókód szintaktikai vizsgálata
DCD4.
Az Ureg-ben levő karakter vagy kulcsszókód szintaktikai vizsgálata
DCD5—DCE8.
Visszaugrás
DCE9—DCEC.
Mint az előző
DCED—DCF8.
Mint az előző
DCF9—DCFC.
Mint az előző
DCFD—DDØ7.
A következő karakter vagy kulcsszókód betöltése az Uregbe
DDØ8—DD12.
Az Ureg tartalmától függően csökkenti az Yreg tartalmát
DD13—DD19.
Az AR—X vizsgálata
DD1A—DD2C.
Egy aritmetikai rutin
DD2D—DDB4.
Egy másik szubrutin folytatása
DDB5—DDC7.
Az Xreg feltöltése paraméterrel
DDC8—DDD8.
Az AR—X eltárolása
DDD9—DE5D.
Egy közelebbről meg nem határozott szubrutin
DE5E—DE81.
TIME
DE82—DE96.
Karaktersorozat-hossz meghatározás
DE97—DEAE.
Az AR—X feltöltése karaktersorozatok adataival
DEAF—DEBB.
Ugyanez az Xreg-gel
DEBC—DEDØ

A programparaméterek aktualizálása
DED1—DEE2.
Mint az előző
DEE3—DFØE.
Szintaktikai vizsgálat
DFØF—DF22.
Mint az előző
DF23—DF3A.
Üzemmódvizsgálat
DF3B—DF62.
Szintaktikai vizsgálat
DF63—DF71.
A következő sorkezdet meghatározása
DF72—DF7F.
A programparaméterek aktualizálása
DF8Ø—DF92.
Az Xreg feltöltése a program kezdőcímével
DF93—DFF4.
Egy közelebbről meg nem határozott szubrutin
DFF5—DFF9.
Az Ureg feltöltése a programhosszal
DFFA—DFFF.
RESET rutin
EØØØ—E167.
PC—PORT paraméterek
E168—E17Ø.
NMI rutin
E171—E22A.
Mint az előző
E22B.
TIMER-megszakító
E22C—E233.
PV-lapozás
E234—E242.
Karakterbeadás
E243—E33E.
Automatikus kikapcsolás
E33F—E42B.
Egy karakter beadása
E42C—E45Ø.
BREAK-kérdés
E451—E456.
OPN
E457—E49F.

Karakterkiegyenlítés
E4AØ—E4A7.
A következő kulcsszótáblázat megkeresése
E4A8—E4B6.
A kulcsszótáblázat kiterjesztése
E4B7—E4EA.
PRINT
E4EB—E572.
TIMER-mód változtatása
E573—E5CØ.
BEEP
E5C1—E654.
BEEP be/ki
E655—E6A4.
PAUSE
E6A5—E7AB.
GPRINT
E7AC—E83D.

GCURSOR
E83E—E847.
CURSOR
E848—E864.
CLS
E865—E869.
WAIT
E86A—E88B.
Időkésleltetés
E88C—EA77.
USING
EA78—EB3F.
Egy közelebbről meg nem határozott szubrutin
EB40—EC5B.
Karakter átvitele a kimeneti pufferbe
EC5C—EC73.
Egy közelebbről meg nem határozott szubrutin
EC74—ECEA.
Mint az előző
ECEB—ECFF.
LCD-re adatkiadás
ED00—ED5A.
Karakterek kiadása
ED5B—EDAA.
Mátrixpointer vizsgálata
EDAB—EDB0.
Mátrixpointer növelése
EDB1—EDF5.
Egy bitminta kiadása az LCD-re
EDF6—EE1D.
Mátrixcím-meghatározás
EE1F—EE70.
Az LCD törlése
EE71—EECA.
POINT
EECB—EEFF.
A fénypont villogtatásának leállítása
EF00—EF80.
A kimeneti puffer törlése
EF81—EFB9.

Aritmetikai szubrutinok

Összeadás
EFBA—F01A.
Szorzás
F01A—F07F.
Az AR—X érték
F080—F083.
Osztás
F084—F0E8.

Újabb utasítások

SQR
F0E9—F160.
LN
F161—F164.
LOG
F165—F1CA.
EXP
F1CB—F390.
COS
F391—F39D.
TAN
F39E—F3A1.
SIN
F3A2—F491.
ACS
F492—F495.
ATN
F496—F499.
ASN
F49A—F530.
DEG
F531—F563.
DMS
F564—F596.
ABS
F597—F59C.
SGN
F59D—F5BD.
INT
F5BE—F5DC.
RND
F5DD—F640.
RANDOM
F641—F660.

Egyéb szubrutinok

Az AR—X átalakítása BCD formára
F661—F662.
Mint az előző, csak az előjel az akkumulátorban
F663—F6B3.
Az AR—X-ben levő BCD érték átalakítása ASCII értékké
F6B4—F6E5.
Előjelvizsgálat
F6E6—F6FA.
Abszolútérték-képzés az AR—X-ben
F6FB—F706.
Az AR—X áttöltése az AR—S-be
F707—F70C.
Az AR—X áttöltése az AR—Y-ba
F70D—F714.
Az AR—S áttöltése az AR—Y-ba
F715—F728.
Az előjel és a mantissza áttöltése az AR—X-ből az AR—Y-ba
F729—F72E.
Mint az előző, csak az AR—Z-be
F72F—F73C.
Az AR—Y áttöltése az AR—X-be
F73D—F746.
Az előjel és mantissza kitörlése az AR—Y-ból
F747—F756.
Az AR—X törlése
F757—F75E.
Az előjel és mantissza törlése az AR—X-ből
F75F—F762.
Az UL bájt kezdőcímének törlése az Xreg-ből
F763—F774.
Az előjel és mantissza 4 bittel jobbra léptetése
F775—F79B.
Mint ez előző, csak balra
F79C—F7A6.
Egy véletlenszám betöltése az AR—X-be
F7A7—F7AF.
A felsőbájt pointer átvitele az X-, Y-, Ureg-ből az aritmetikai regiszterbe
F7B0—F7B4.
Az AR—X és AR—S cseréje
F7B5—F7B8.
Az AR—X és AR—Y cseréje
F7B9—F7CB.
Az AR—X U, AR—Y mantisszáinak összeadása és az eredmény betöltése az AR—X-be
F7CC—F7CD.
Decimális összeadás
F7CE—F88A.
Az AR—Y feltöltése
F88B—F88E.
Mint az előző
F88F—FA73.
A kulcsszótábla kiterjesztése
FA74—FA88.
Kulcsszófeldolgozás
FA89—FB29.
PV-lapozás
FB2A—FB9C.
Egy közelebbről meg nem határozott szubrutin
FB9D—FBDF.
Matematikai állandók
FBE0—FC9F.
Karaktermaradvány-tábla
FCA0—FE7F.
Billentyűzet alapfunkciók
FE80—FEBF.
Mint az előző, csak SHIFT-tel
FEC0—FEFF.
CALL vektor tábla
FF00—FFFF.

DR. SIMONYI ENDRE

ADOK – VESZEK – CSERÉLEK

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

■ Spectrum Plus gépemet adatrögzítővel, programozható interfésszel, bő szakirodalommal és programokkal elcserélném Commodore 64-re vagy eladnám készpénzért. Telefon: 343-100/295 Szabó József.

■ Plus/4 és C16 programokat cserélek. Horváth Károly, Szeged, Gera Sándor u. 55. 6729.

VZ—200 Color számítógépemet eladnám vagy elcserélném (ráfizetéssel) 16 k-s Spectrumra. Telefon: 76/27-192.

■ Commodore VIC—20 eladó datasettel, programokkal. Frikker Gábor, Hercegkút, Petőfi Sándor út 74. 3958.

■ Keresem a „Your Computer" magazin 1984. decemberi, 1985. májusi, októberi, novemberi, decemberi számait megvételre. Telefon: 156-411 du. 4—9-ig.

PTA—4000 Gépi kódú miniprogramok

INVERZ DISPLAY hívása: CALL 17408

17408_{dec}

```
68 70 BE 44 07 68 71 6A 4D B5 FF
2D 2E 88 06 9A
```

SCROLL BALRA CALL 17424

17424_{dec}

```
58 71 48 72 BE 44 31 58 72 48 73
BE 44 31 F4 72 FE F6 72 4C F4 71
FE 24 F1 2A A4 F1 28 F6 D7 4C 9A
5A FE 4A 00 6A 4D F5 88 03 9A
```

HANGSKÁLA CALL 17467

17467_{dec}

```
48 01 4A 02 6A FF BE E6 03 88 05
BE E6 D3 B5 FF 64 26 99 09 9A
```

ORGONA CALL 17488

Billentyűk: A—Z; kilépés: BREAK

17488_{dec}

```
BE E4 2C DF 81 0E B3 2C 58 44 1A
15 2A 48 01 4A 0A BE E6 6F FD A5
F2 0B B9 02 9B 1C 9A F5 95 B9 C3
D0 B4 A0 8F 7A 7D 76 5F 77 84 6A
55 FF BB DC AA 84 A8 E8 CD 98 E5
```

SCROLL JOBBRA CALL 17543

17543_{dec}

```
F4 D6 4C F6 D6 FE F4 D7 4C 24 F1
2A A4 F1 28 F6 D5 FE 48 D6 58 D6
BE 44 A4 48 D7 58 D7 4A 4B 5A 4D
6A 4D 47 53 88 04 9A
```

Betöltés: POKE 17408,&68,&70,&BE . . . stb.

MÉSZÁROS CSABA

Ki ad magyarázatot?

A DRAW utasítás magyarázata

A DRAW X, Y, φ alakú Spectrum-utasításban X és Y, X,Y számú lépés megtételét jelenti vízszintesen, illetve függőlegesen, φ pedig az (X_0, Y_0), (X_0+X, Y_0+Y) pontok közé húzható húrhoz tartozó középponti szöget jelenti (lásd az *ábrát*).

A 3 paraméteres DRAW tehát a 2π-nél rövidebb ívek rajzolására való. Az ívrajzoló ciklus a körívet egyenesekkel közelíti. Azért, hogy a körtől való eltéréseket korlátozzák, az elemi egyenessel közelített ívek számát (N) a „φ" függvényében határozzák meg.

$N = 4 * INT(ABS(\varphi) * SQR((ABS(X) + ABS(Y))/(ABS(SIN(\varphi/2)))/8) + 4$

Ennek a képletnek az értéke

N = 4, 8, 12, . . ., 252 lehet. Azért van 252-re korlátozva, hogy a kapott szám 1 bájton ábrázolható legyen.

Nagy középponti szög esetén a jól közelített körhöz 252-nél több ívdarabra lenne szükség, de a gép csak 252-t húz. Minél nagyobb ez a különbség, annál inkább láthatóvá válnak a húrok.

Abban az esetben, ha d $\varphi = \frac{\varphi}{N}$ értéke valamilyen nevezetes szög, például 72°, 120° stb., az egymás után húzott húrok ötszöget, háromszöget adnak.

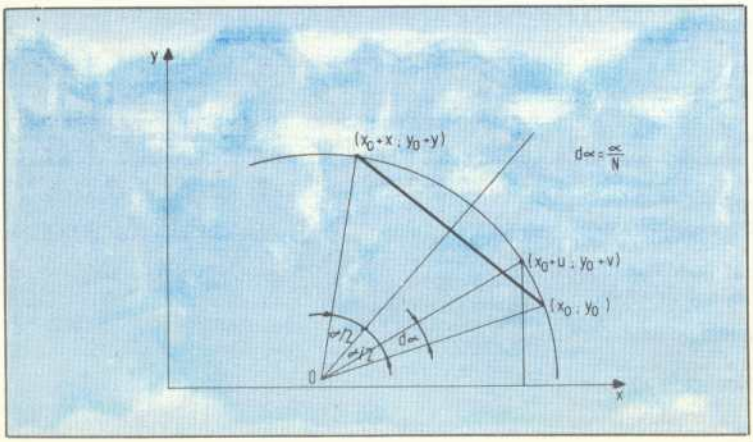
Mivel az ívszám (252) sokkal nagyobb, mint az oldalak száma, ezért többször végigmegy ugyanazon a vonalon.

Abban az esetben, ha a húrok végpontja nem ér vissza a kezdőpontra, akkor az ábrát forgatja, és a ciklus lefutása után a legtöbb esetben körgyűrűt kapunk.

Fontos tudni, hogy ha $\sin \frac{\varphi}{2} = 0$, akkor a DRAW csak az (X_0, Y_0) és $(X+X_0, Y_0+Y)$ pontok közötti egyenest húzza meg.

Lehet próbálkozni a következő képletekkel:

1. N = 252
2. $\varphi = \frac{2\pi}{n} N$ n = az oldalak száma
3. $\varphi \neq \pm 2k\pi$



A hibajelzésekről

Leggyakoribb a B hiba (Integer out of range)

1. Vonalak jelennek meg.

A hibajelzés a DRAW-LINE rutinból jön. Az utolsó pont címe a képernyő RAM-on kívülre kerülne.

2. A kezdőpont megjelenik, de más nem a hibajelzésen kívül.

DRAW-LINE rutin ún. kétbájtos egész alakban fogadja az $X_i Y_i$ értéket, de X_i, Y_i nem fér el 16 biten.

HIVESSY BÉLA

Lothar Englisch:
Gépi kódú programozás a Commodore 64-esen
(Budapest, 1985. DATA BECKER—NOVOTRADE Rt., 125 oldal. Ára: 241,— Ft.)

A könyv a gépi kódú programozás tankönyvének igényével lép fel, de alapos tanulmányozása mellett rengeteg gyakorlásra is szükség van a gépi kódú programozás elsajátításához, hatékony alkalmazásához.

Az első rész a 6510 processzor regisztereit, a címzési módokat és funkciójuk szerint csoportosítva a gépi utasításokat ismerteti. Ezt követi a gépi kódú programok készítésének technikájáról szóló rész, mely az assembler fogalmát és használatát is tárgyalja. A fejezet tartalmaz egy teljes BASIC nyelven megírt „tanulói" assembler programot és a 6510-es processzor egy egylépéses szimulátorát a gépi kódú programok tesztelésére. A következő rész néhány rövid példát mutat be gyakorlásként, majd megmutatja a BASIC interpreter és a BASIC bővítések ejárásainak felhasználási lehetőségeit. Végül a disassembler feladatával és egy BASIC nyelven megírt „tanulói" disassembler listájával, működésével ismerkedhet meg az olvasó.

Hosszabb gépi kódú programok készítéséhez a PROFI-ASS 64 2.0-t ajánlja a szerző, ismertetve használatának szintaxisát. A könyvet átszámítási és utasítástáblázatok zárják.

Angerhausen—Brückmann—Englisch—Gerits:
A Commodore 64-es belső felépítése
(Budapest, 1985. DATA BECKER—NOVOTRADE Rt., 316 oldal. Ára: 355,— Ft.)

Az első fejezet a hardverfelépítés elvi vázlatával indul, majd a 6510-es processzor lábkiosztásának ismertetése után a lehetséges tárfelosztásokról ad összefoglalót. A második fejezet a hangszintetizátorral foglalkozik. Közli a 6581-es SID lábkiosztását, regisztereinek leírását, a hang létrehozásának elvi vázlatát. Kitér a beépített A/D átalakítók szerepére, majd a SID programozását mutatja be néhány példával illusztrálva.

A harmadik fejezet témája a 6569-es VIC processzor. Lábkiosztása és regisztereinek leírása után a VIC ábrázolási módjait tárgyalja, teljes terjedelmében közli a GRAFIK AID programot, és használatát is bemutatja, majd leírja a szellemgrafika lehetőségeit, műveleteit. A negyedik fejezet a 6526-os CIA processzorok ismertetését adja. A lábkiosztás és a regiszterek után az I/0 pontokat, majd a valós idejű órát, végül a soros buszt mutatja be. Az ötödik fejezet 6 gépi kódú rutint tartalmaz.

A hatodik fejezet a gépi kódú és az assembly szintű programozás kérdéseivel foglalkozik, feltételezve, hogy az olvasó ismeri már a téma alapjait. Közli a C64 operációs rendszerét alkotó legfontosabb rutinok kezdőcímeit tartalmazó táblázatot, majd a gépi kódú programok adatforgalmát tárgyalja, példákkal illusztrálva. Bemutatja a gép és a perifériák közötti adatmozgatási lehetőségeket, hangsúlyozva az RS 232 nyújtotta megoldásokat.

A hetedik fejezet a haladó programozók számára hasznos módon, táblázatos formában nyújtja a C64 és a VIC 20 nullás lapjának és fontosabb tárcímeinek ismertetését, valamint a BASIC interpreter rutinjainak kezdőcímeit, és tartalmazza a C64 és a VIC 20, illetve a C64 és a CBM 8000 gépek legfontosabb eljárásai kezdőcímeinek összehasonlító táblázatát.

A nyolcadik fejezet elején a ROM rutinok megnevezését és kezdőcímeit tartalmazó táblázatot találjuk. Ezt követi a könyv legfontosabb része, a C64 teljes, magyarázatokkal kiegészített ROM listája, melyet a kapcsolási rajzhoz magyarázatként készített hardverleírás követ.

Englisch—Szczepanowski:
A VC—1541-es lemezegység programozása
(Budapest, 1985. DATA BECKER—NOVOTRADE Rt., 280 oldal. Ára: 355,— Ft.)

Az első fejezet bevezetés a VC—1541 programozásába. Bemutatja a programok lemezen való tárolásával kapcsolatos parancsokat, majd a lemezegység rendszerutasításait és a soros adattárolás teendőit. A relatív adattárolás tárgyalása során ismerteti a REL típusú fájlokkal végezhető műveleteket. Példát mutat egy rekord kulcs alapján történő keresésére, módosítására. A második fejezet alkalmazási példákkal illusztrálva mutatja be a lemez blokkjainak közvetlen elérésére szolgáló BLOCK utasításokat, majd a DOS közvetlen elérését lehetővé tevő MEMORY utasításokat.

A harmadik fejezet ismerteti a lemezegység tárfelépítését, megadja a rendszerváltozók címeit. Foglalkozik a BAM, a tartalomjegyzék, valamint a relatív fájlok oldalszektorainak felépítésével, szerepével. Ez a fejezet tartalmazza a könyv legfontosabb részét, a magyarázatokkal kiegészített, teljes DOS listát.

A negyedik fejezet teljes programokat közöl a fájlok védelmére, paramétereik kirására. Ezt követi a TEST/DEMO lemez segédprogramjainak rövid ismertetése, majd három gépi kódú rutin a fájlok kezelésének megkönnyítésére. A fejezet egy teljes Disk-Monitor program ismertetésével zárul. Az utolsó fejezet összehasonlítja a különböző CBM lemezegységeket.

Fekete István:
Matematika és számítástechnika 1. Melléklet: Programok a matematika és számítástechnika 1. című könyvhöz
(Budapest, 1986. Műszaki Könyvkiadó, 205 oldal. Ára: 66,— Ft.)

A könyv, mely eredetileg főiskolai jegyzetnek készült, végigvezeti az olvasót a műszaki gyakorlatból adódó matematikai problémák számítógépes megoldásának útján. A szerző ismerteti a gyakorlati problémákból való modellalkotás módját, az algoritmus, vagyis a képletes-szöveges megoldási utasításrendszer ismeretében pedig bevezet a számítógépes programírásba.

A kötet főbb fejezetei: Emberi és gépi gondolkozás és számolás; Szám, vektor, mátrix; Számsorozatok, sorok, határérték; Egyváltozós függvények és differenciálásuk. A könyvet számítógépes programok gyűjteménye egészíti ki.

DaCosta, F.:
A kalandprogram írásának rejtelmei
Hogyan írjunk BASIC nyelven az iskolaszámítógépre kalandprogramot
(Budapest, 1986. Műszaki Könyvkiadó, 232 oldal. Ára: 63,— Ft.)

A könyv a 16 kbájtos TRS-80 személyi számítógép programozásával foglalkozik. A BASIC nyelvű játékprogramokon keresztül — amelyeknek teljes programlistáját, működési leírását, tervezését is ismerteti — bevezeti az olvasót a korszerű strukturált programozási módszerekbe, s ugyanakkor számos olyan, kis számítógépeknél hasznos fogást közöl, amelyek segítségével csökkenthető a programok tárigénye és futási ideje. A könyvbeli programok Microsoft BASIC-ben készültek, amely a HT-1080Z iskolaszámítógép BASIC-változata is, így valamennyi program e géptípuson — de más személyi számítógépeken is — változtatás nélkül lefuttatható.

Ismerd meg a BASIC nyelvjárásait!
COMMODORE 64, COMMODORE VIC 20, SHARP PC—1500
Szerk.: Kőhegyi János
(Budapest, 1986. Műszaki Könyvkiadó, 186 oldal. Ára: 65,— Ft.)

A könyv oldalszámokkal hivatkozva Alcock: Ismerd meg a BASIC nyelvet! c. munkájára, az alcímben felsorolt három (ill. a SHARP-1500 magyar megfelelőjével, a PTA-4000-rel együtt négy) személyiszámítógép-típus BASIC programnyelvének egyedi jellemzőit mutatja be. Nem ismétli az Alcock-műben leírtakat, de azzal azonos, kézírásos formában, közvetlen stílusban szól a hazánkban is slágernek számító gépekről, tanácsokat, ötleteket, tájékoztatást ad a gépek felhasználóinak.

ÉNEKES FERENC

A teljesítményviszonyok

A maximális átviteli teljesítményre törekvő tervezésnél figyelembe kell venni a hálózati architektúrát, annak megvalósítását (kábel), az elérési módot és annak késleltetés-átbocsátóképesség karakterisztikáját, a hálózattól várt szolgáltatásokat stb., mivel a hálózat teljesítményét ezek együttesen határozzák meg. Az erre vonatkozó vizsgálatokat az IBM zürchi kutatóközpontjában végezték el.

Egy helyi hálózat teljesítményét tehát a választott hálózatelérési eljárás késleltetés-átbocsátóképesség karakterisztikája, a hálózat átviteli sebessége, az átviteli közeg hossza és a hálózatba kapcsolt munkaállomások száma szempontjából kell vizsgálni, de jelentős szerepük van azoknak a tényezőknek is, melyeken keresztül a felhasználó „látja" a hálózatot (a fájlkezelő rendszer, az alkalmazott protokoll és annak szintje, a felhasználói interfész, a puffer mérete stb.).

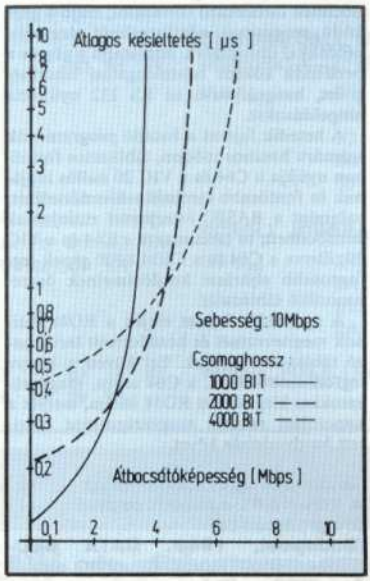
CSMA/CD

A vizsgálatok azt mutatták, hogy a CSMA/CD protokollal működő rendszerben a forgalomtorlódás miatt előálló többszörös csomagütközés csak igen erős csatornatúlterhelés esetén fordul elő. (Ez okozza a teljesítmény csökkenését.)

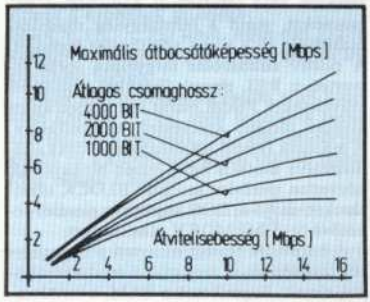
A hálózatelérési protokoll

A helyi hálózatokra vonatkozó nemzetközi előírásokat az IEEE 802 szabvány tartalmazza.

A hálózatelérési protokoll olyan szabályt jelent, amelynek segítségével egy állomás forgalmazási jogot szerez a hálózaton olyan céllal, hogy adatokat vigyen át egy másik állomásra, illetve adatokat hívjon le onnan. A CSMA/CD protokollal működő rendszer a csatornafoglaltsághoz az adásismétlések közötti kivárásokkal (résidő) igazodik. Ennek érdekében választják ezt az időt a két forgalmazott csomag közötti adásszüneti idő többszörösére. Ez az idő mindig nagyobb, mint a kábelen mérhető jelterjedés ideje: 10 Mbps-os sebességű rendszer kivárási ideje 51 µs. Ami a csomaghosszakat illeti, célszerű a CSMA/CD rendszer üzenetcsomagjainak hosszát minimalizálni, például 10 Mbps esetén 512 bitre választani. Ezáltal az ütközések miatti ismétlések alkalmával kisebb csomagokat kell újra elküldeni, ami ugyancsak a túlterhelt rendszer teljesítménycsökkentését igyekszik ellensúlyozni.



1. ábra. CSMA/CD rendszer késleltetés-átbocsátóképesség karakterisztikája a csomaghosszúság függvényében

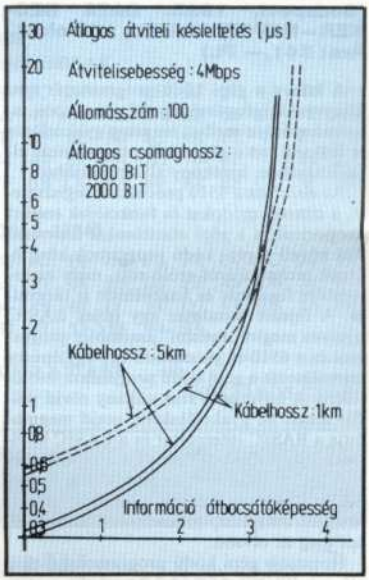


2. ábra. A hálózat átbocsátóképességének függése az átviteli sebességtől és az átlagos csomaghosszúságtól

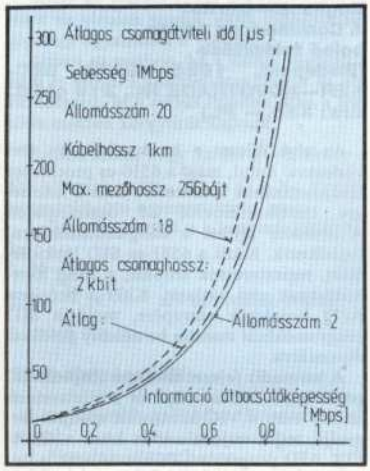
A CSMA/CD rendszer vétel oldali szinkronizmusát a csomagok előtt küldött szinkronizáló impulzusok biztosítják. Ezek segítségével szinkronizálódik fel a vételi oldal, mielőtt elvégzi a címdekódolást (neki szól-e az üzenet).

Teljesítménykarakterisztikák

Az egyes állomások forgalmazási jogát vezérelt jelzéses (token passing) rendszerben egymás után sorolják, és minden állomásnak egy adott hosszúságú „időszelet"

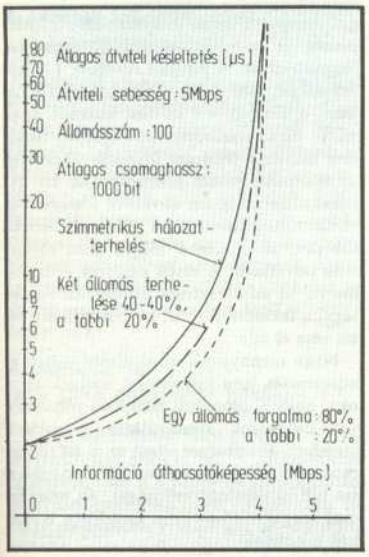


3. ábra. Vezérelt jelzéses gyűrűs hálózat átlagos csomagátviteli késleltetés-információátbocsátó képesség karakterisztikája 4 Mbps átviteli sebesség, 100 munkaállomás ás 1-5 km kábelhossz esetén

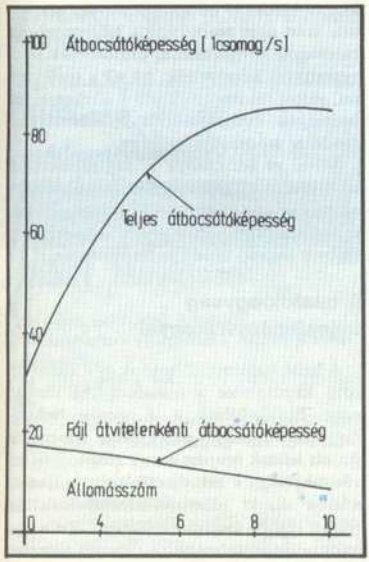


4. ábra. A vezérelt jelzéses gyűrűs hálózat késleltetés-átbocsátóképesség karakterisztikája hosszú adási idő birtoklása esetén

áll rendelkezésére üzenetcsomagja továbbítására. A forgalmazás után legalább annyi időt kell kivárnia, mint amennyi a hálózat

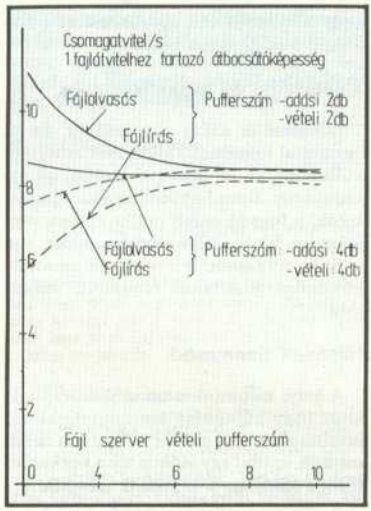


5. ábra. Vezérelt jelzéses buszhálózat átlagos késleltetés-átbocsátóképesség karakterisztikája

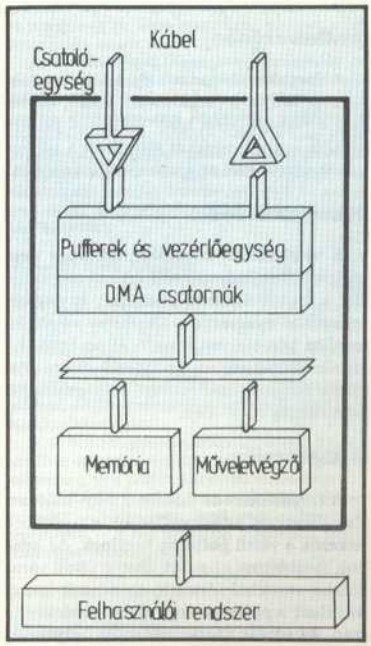


6. ábra. A teljes csomagátbocsátó-képesség és a fájl átvitelenkénti átbocsátóképesség-változása az állomásszám függvényében

jelterjedési ideje; ezalatt érzékeli az előző adás végét és kezdi meg saját adását.



7. ábra. Az átvitel közbeni csomagvesztés és a fogadó oldali pufferméret közötti kapcsolat



8. ábra. A csatolóegység blokkvázlata

Az *1. ábra* egy 10 Mbps-os CSMA/CD rendszer késleltetés-átbocsátóképesség karakterisztikáját szemlélteti a csomaghosszúsággal paraméterezve. A feltüntetett átlagos átviteli késleltetésen a csomagnak az adás helyéről a vétel helyére való továbbításához szükséges idő értendő, az átbocsátóképesség az egy időegység alatt sikeresen továbbított bitek számát jelenti.

A hálózat átbocsátóképessége függ a forgalmazott csomagok hosszától; rövidebb csomaghossz esetén kisebb az átlagos késleltetés, azaz nagyobb az átbocsátóképesség. A teljesítménycsökkenés a csomagok egyre gyakoribbá váló ütközése miatt áll elő, és kb. 2,5 Mbps-nál kezdi éreztetni hatását.

A hálózat átbocsátóképességének az átviteli sebességtől és az átlagos csomaghosszúságtól való függése a *2. ábrán* látható.

A fentiekből is érzékelhető, hogy a CSMA/CD protokoll hatékonysága főleg nagyobb sebességek és kisebb csomaghosszúságok esetén csökken erőteljesen.

A vezérelt jelzéses gyűrűs hálózat

A vezérelt jelzéses gyűrűs (token ring) hálózat széles körben elterjedt kiváló tulajdonságai miatt: egyszerű elrendezés, hatékony átviteli eljárás, nagy teljesítmény, garantált válaszidők, mérsékelt beruházásigény stb. Hátránya viszont a kisebb megbízhatóság, ami megfelelő hozzáférési eljárás és hálózati elrendezés (topológia) segítségével ellensúlyozható.

Teljesítménykarakterisztikák

Az állomások közötti átviteli késleltetés a kábelkésleltetésből (kb. 5 µs/km), a csatlakoztatások okozta késleltetésekből és a nyugtázás beállításából tevődik össze. A helyi hálózati protokollok különböző mértékben érzékenyek a rendszer paramétereinek változásaira. A *3. ábra* egy vezérelt jelzéses gyűrűs hálózat átlagos átviteli késleltetés-információátbocsátó képesség karakterisztikáját szemlélteti 4 Mbps átviteli sebesség, 100 munkaállomás, valamint 1 km és 5 km kábelhosszak esetén. A karakterisztikából látható, hogy a kábelhossz változása alig befolyásolja a rendszer teljesítményviszonyait, az átviteli sebesség 4 Mbps-ról 16 Mbps-ra való növelése a karakterisztikát önmagával párhuzamosan (jobbra) tolja el. Az ábrából az is látható, hogy a karakterisztika alig érzékeny a csomaghossz változására.

Egy rendszer teljesítményét elsősorban az alkalmazott protokoll hatékonysága és a felhasználók számára nyújtott szolgáltatások minősége jellemzi, ez pedig elsősorban

a hozzáférési időrés alatt továbbítható csomagok számával (1-256 lehet) mérhető.

Ami a terhelés megoszlását illeti, a vizsgált 20 állomásos rendszerben két állomás adta a teljes forgalom 40%-át és 18 állomás a forgalom 1,1%-át. A vizsgálat tapasztalata az volt, hogy rövid adási idők esetén a kevéssé kihasznált állomások átviteli késleltetése kicsi, a terhelt állomásoké nagy, hosszú átfutási idők esetén pedig a helyzet éppen fordított. Ennek az az oka, hogy a hosszú adási időt az erősen terhelt állomások hosszabb üzeneteikkel jobban ki tudták használni, mint a kevésbé terhelt állomások. Hosszú adási idő esetén a rendszer késleltetés-átbocsátóképesség karakterisztikáját a *4. ábra* mutatja.

A vezérelt jelzéses buszhálózat (token busz)

Ez a rendszer egyesíti a buszarchitektúra és a vezérelt jelzéses hozzáférés előnyeit, azaz jó hatásfoka nagy terhelés esetén is megmarad, hozzáférési rendszere jól definiált, viszonylag érzéketlen az átviteli sebesség változására és a busz (kábel) fizikai hosszára.

Hozzáférési rendszer

A buszhoz való hozzáférést a token buszhálózat munkaállomásai maguk vezérlik: adás végén az újabb adás jogát automatikusan továbbadják a következő címre. A buszhálózat nélkülözi a gyűrűs hálózatnál tapasztalható karbantartási kényelmetlenségeket (kezdeti beállítások, új állomások installálása stb.), a tokenidőket az egyes állomások maguk időzítik. Az adási jog továbbadása előtt az állomások ellenőrzik a következő cím aktív állapotát, hiba esetén automatikusan korrigálják a forgalom sorrendjét.

Teljesítménykarakterisztikák

Az *5. ábra* egy 5 Mbps-os 100 állomásos, 1000 bites csomaghosszakkal működő vezérelt jelzéses buszhálózat késleltetés-átbocsátóképesség karakterisztikáját szemlélteti azokban az esetekben, amikor 1. minden állomás egyformán forgalmaz (szimmetrikus terhelés); 2. két állomás adja a forgalom 40-40%-át, a többi pedig a maradék 20%-ot; 3. egy állomás adja a forgalom 80%-át, az összes többi a maradék 20%-ot (aszimmetrikus terhelés). A rendszer átlagos átviteli késleltetése meglehetősen nagy, 2 ms körüli, ami a protokoll vezérlésével együttjáró tetemes rendszeradminisztrációval függ össze. A vizsgálatok azt mutatták, hogy az aszimmetria növekedésével a késleltetés kismértékben csökkent.

Adatátvitel és hibaellenőrzés

Ellenőrzésre akkor van szükség, ha az egymással forgalmazó állomások sebessége különböző, és az átmeneti tárolók mérete korlátozott. Ilyen esetben főleg az állomáshibák, a fogadó oldali puffer hiánya vagy időzítési hibák kiküszöbölése miatt van szükség az ellenőrzésre, amely a protokoll különböző adatátviteli rétegeiben valósul meg.

Hálózati üzemmód

A helyi hálózatok a munkaállomásokon kívül nagy kapacitású lemezegység-kezelő számítógépet (fájl szerver állomás) is tartalmaznak, ezáltal egy időben több konkurens fájlhoz fordulást is ki tudnak szolgálni.

A szerver állomással való kapcsolat felépítését, bontását, az átvitel vezérlését és a hibaellenőrzést a működtető protokoll vezérli.

Átvitelvezérlés

A forgalmazó „adás" típusú csomagokban helyezi el a forgalmazni kívánt adatokat, a fogadó „vétel kész" típusú — pozitív nyugta — csomagokkal nyugtázza a sikeres vételt, illetve kéri az újabb adatok küldését.

Hibaellenőrzés

A vétel helyén hibásnak ítélt vagy nem megfelelő sorszámozással érkező csomagokat a fogadó állomás „eldobja", és negatív nyugtával nyugtázza a sikertelen vételt, ismétlést kérve a forgalmazó állomástól. Az egyes állomások azt is figyelik, hogy történt-e forgalmazási kísérlet a forgalmazási jogosultság ideje alatt.

A fájlkezelés

A forgalmazandó adatok a helyi hálózati csatolóegység adási pufferébe, a gyűrűből érkezők a vételi pufferbe kerülnek. Az adatok továbbítása az adási, illetve vételi várakozási sorokból történik, az adatok innen kerülnek a pufferbe vagy olvasódnak ki onnan. Az adatok várakozási sorba helyezését és onnan való kiolvasását a fájlkezelő vezérli (lemezre írás és kivitel a lemezről).

A munkaállomások egy időben vagy forgalmaznak vagy fogadnak adatokat.

A fájlátvitel lehet adási vagy vétel irányú. A *6. ábra* a csomagátbocsátó-képességet mutatja a teljes rendszer-átbocsátóképesség és a munkaállomások számának függvényében. A vizsgált rendszer 1 Mbps sebességű, 500 bites csomagokkal forgalmaz, a csatoló 4-4 puffert használ mind adás-, mind vételirányban. Alacsony forgalom esetén a rendszer átbocsátóképessége az állomásszámmal arányosan nő kb. 5-7 munkaállomásig, ezt követően a szerver állomás túlterhelődik, és a rendszer átbocsátóképessége telítésbe megy, azaz tovább nem növelhető. A vételi pufferek csökkentésével az adatvesztések száma nő. Veszteségnek tekinthető az is, ha pufferhiány miatt vész el adat.

Nagy mennyiségű adat vétele esetén az adatvesztés igen kismértékű, aminek az az oka, hogy a protokoll előnyben részesíti a vett csomagok kiszolgálását az adással szemben. Veszteséget jelent az is, ha túl sok csomagot kell fogadni, de ebből csak keveset tud az állomás nyugtázni. Az adatcsomag forgalmazását ilyen esetekben is meg kell ismételni.

Az átvitel közbeni csomagvesztés és a fogadó oldali pufferméretek közötti összefüggés a *7. ábrán* látható. A karakterisztikából kitűnik, hogy ha a pufferméret kicsi, csökken az átvitel és nő a csomagvesztések száma, mert nem marad idő a hibátlanul vett csomagok nyugtázására, ezáltal ezek is elveszettként kezelendők; ha nő a pufferméret, akkor jut elegendő idő a sikeresen vett csomagok nyugtázásra is, és ezzel nő a rendszer átbocsátóképessége.

Mivel az adásirányú átvitel csökkenését a sikeres nyugtázások számának növekedése részben ellensúlyozza, összességében a teljes átbocsátóképesség érzéketlen a fájl szerver fogadó oldali pufferméretére.

A csatolóegység teljesítményviszonyai

A helyi hálózati állomások és a kábel közötti kapcsolatot a csatolóegység teremti meg. Blokkvázlata a *8. ábrán* látható. Funkcióit cikkünk előző része ismertette. Az ott leírtak csupán annyi kiegészítést kívánnak, hogy a vett és előtárolt (pufferelt) adatok direkt memória-hozzáférésen keresztül (DMA csatorna) töltődnek be a csatolóegység-memóriába. Az összes ismertetett műveletet a csatolóegység belső műveletvégző egysége látja el. Ha vételnél nem áll rendelkezésre elegendő üres fogadópuffer vagy szabad DMA csatorna, a vett csomag elvész. Ennek elkerülésére olyan protokollt dolgoztak ki, amely a csomagokat

akkor is fogadja, ha a két csomag érkezése közötti idő minimális.

Hasonló módon történik az adás is, csak fordított irányban. Létezik olyan megoldás is, melynél a csomag adatrésze a csatoló memóriája helyett közvetlenül a fogadóállomás belső memóriájába íródik be.

A csatoló teljesítményét befolyásoló tényezők

Az átvitel közbeni csomagvesztések számát vagy megfelelő méretű puffertár alkalmazásával, vagy a vett csomagok minél gyorsabb lekezelésével lehet csökkenteni. Az utóbbi célt szolgálják a kizárólag vételi feladatokat ellátó, gyors működésű DMA csatornák, melyek üres puffer és szabad DMA csatorna esetén átveszik a csomagot; ha ezek nem állnak rendelkezésre, akkor egy csomagfogadó pufferbe töltik be az adatot arra az időre, amíg a fenti feltételek teljesülnek.

A pufferek és a DMA csatornák hatása

Vonalpufferelléssel csökkenthető a csomagvesztések száma, így a teljesítménycsökkenés csak a 8 Mbps-os sebességtartományban észlelhető ismét. A sebesség további növelésével (16 Mbps) egyre inkább a DMA csatorna és a csatolóegység műveletvégző teljesítménye válik „szűk keresztmetszetté". Mindebből érzékelhető, hogy egy rendszer átviteli teljesítménye elsősorban a DMA csatornák számának és csak másodsorban a vonali pufferek számának növelésével javítható.

CSEH KÁLMÁN

Az olvasó írja

Mindig nagyon várjuk olvasóink leveleit, most különösen, hiszen meg szeretnénk ismerni a véleményüket, mielőtt megkezdjük a lap 5. évfolyamának szerkesztését. Szeretnénk, ha elmondanák, melyik rovat tetszik és melyik nem, miről szeretnének még olvasni, mi hiányzik és mi a felesleges. Várjuk leveleiket.

Heidrich Attila, Lenинváros,

Bartók Béla u. 4. V. e. 3580

A júniusi számban megjelent egy rövid megjegyzés az árral kapcsolatban. Nos, ami engem illet — és nem hiszem, hogy egyedül lennék — inkább elnéznék néhány „coca-cola"-t a lapban, minthogy ennél is drágább legyen. Jó nagyon, hogy havonta jelenik meg a magazin, de éppen ezért havi 30 Ft-nál több már sok lenne.

Mi is azt hisszük.

ifj. Malek Miklós, Budapest,

Bajcsy-Zs. u. 45/B 1158

V. osztályos tanuló vagyok. Van egy Commodore 64-es gépem. Sajnos kevés lapjukban a C 64-es játékprogram. Tudom, hogy ebből a gépből a legkínosabb valami grafikát, ill. játékprogramot kihozni, ezért fordulok Önökhöz. Egy kis programot szeretnék, mellyel valamit, pl. egy karaktert lehet mozgatni fel, le, jobbra, balra. A programot ha lehet joystick-re, ha nem, a billentyűzetre tessék írni.

Az a szándékunk, hogy játékprogramokat ezután is közlünk, de nem sokat. Olvasóink leveleiből azt látjuk, hogy sokkal szívesebben látják a lapban a programozási ötleteket, a programozás technológiájáról írt cikkeket, mint a kész játékprogramokat, általában azoknak a programoknak a listáit, amelyeket valamilyen hasznos célra lehet felhasználni (oktatás, adattárolás stb.). Ez volt az oka annak, hogy a játékprogram rovat terjedelmét csökkentettük.

Kemecsei Zsolt, Komárom,

Arany J. u. 11/A 2900

Júliusi számuk DIÁKROVAT-ában örömmel fedeztem fel Káli Csaba írását. Sok hardverkiegészítést sikerült már elkészíteni SPECTRUM-omhoz, de fényceruzarajzot még nem találtam a szakirodalomban.

A szerző felajánlotta, hogy segítséget nyújt a szoftver elkészítésében is, válaszboríték és bélyeg ellenében, de a pontos címe nem jelent meg.

Ezért kérem Önöket, hogy írják meg nekem — ha lehet — Káli Csaba címét, vagy — ha sok az érdeklődő — a lap hasábjain jelentessék meg a szükséges programokat.

Zóka Gábor, Nagyatád,

Rozsnyói u. 5/A 7500

Rendszeres olvasója vagyok lapjuknak, és az e havi számukban (július) találtam egy érdekes cikket. A *Fényceruza ZX-Spectrumhoz* című cikkről van szó. További tanácsokért fordulnék az íróhoz, Káli Csabához, ezért kérem, küldjék meg nekem a címét, ahová írhatok.

Káli Csaba címe: Piarista Gimnázium, Bp., Mikszáth Kálmán tér 1. 1088. Sikeres fényceruzaépítést kívánok!

Képíró Róbert, Budapest

Imre u. 5. II. 14. 1093

Relatíve elég régen vagyok lapjuknak — eleinte — szorgalmas olvasója, majd előfizetője is. Sajnos most több Spectrum-os sorstársammal arra a megállapításra jutottunk, hogy lemondunk az előfizetésről, mert nem állunk olyan jól anyagilag, hogy „Commodore-centrikus" folyóiratokat is vásároljunk, illetve azokra előfizessünk, akkor, mikor a Sinclair témák mind csökkenőbb arányokban fordulnak elő, sőt lassan mind gyakrabban teljesen hiányoznak is.

Gondolom, nem vigasz az sem, hogy nemcsak az Önök lapjával szemben jutottunk erre az elhatározásra, hanem még 2 másik lappal is, mert teljesen felesleges olyasmikre pénzt kiadni, amit ha egyikünk egy példányt megvesz belőle, akkor tapasztalhatjuk, hogy „Sinclair szempont"-ból nem érdemes megvenni és raktározni. Az eddigi tapasztalatok szerint így igen kevés példányt fogunk megvásárolni feleslegesen. Természetesen nem revolverezni akarok ezzel, csupán a saját anyagi helyzetünkön kívánunk segíteni, hiszen Önöknek egypár spectrumos elvesztése nem jelenthet problémát. Csak azért talán érdemes lenne foglalkozni a gondolattal, hogy tekintettel az országban folyó „Commodore-kampányra" és a „Commodore Újság"-ra, na meg arra is, hogy nálunk reklámozni „kell" a sok megvásárolt és raktáron volt gépet — nem indokolt a saját sajtóval rendelkező tábort a többiek rovására támogatni.

Abban a titkolt reményben is ragadtam tollat, hogy talán majd egyszer újra érdemes lesz megvásárolni minden példányt, és akkor újra előfizethetünk is rá (rájuk).

Reagálásukat szeretnénk oly módon tapasztalni, hogy újra a régi, nem részrehajló lapot vehetnénk kezünkbe, addig is a magam és több spectrumos kollégám nevében búcsúzom lapjuktól.

Sajnáljuk, a búcsú sohasem öröm, kiváltképp nem az, ha hamis indokkal szakítják meg velünk a kapcsolatot. Nem vagyunk Commodore lap, és remélem nem is leszünk egyetlen géptípus, pl. a Sinclair elkötelezettje sem. Az a célunk, hogy az informatika társadalmi méretű elterjesztését segítsük, olyan cikkeket, programleírásokat, hardverötleteket, híreket, tanácsokat hozzunk, amelyek az olvasóink informatikai ismereteit szélesítik. Gépspecifikus programokat, példaprogramokat igyekszünk elsősorban a hazai gyártású gépekre közölni (Primo, TV computer, HT), illetve a legelterjedtebb külföldi gépekre, amelyek közül a Commodore-ra, illetve a Sinclairre hoztuk eddig a legtöbb programot. Hogy melyik gépre hányat, azt az dönti el, hogy milyen választékból válogathatunk. Azokat a programokat részesítjük előnyben, amelyek a legjobbak, a program jól strukturált, szellemes programozási megoldásokat tartalmaz. Nem tudna néhány ilyen Sinclair programot küldeni?

Zsurka Gábor, Szeged-Tarján,

Bite Pál u. 2/A III. 10. 6723

A szegedi Radnóti Miklós Gimnázium harmadikos tanulója vagyok, speciális biológiai osz-

tályban tanulok. Körülbelül egy éve vásároltunk egy A—48-as Primót. Néhány hónapig csak BASIC-ben írtam programjaimat, később azonban érdeklődni kezdtem a gépi szintű programozás iránt. Komolyabb gépi kódú programok elkészítéséhez szükségem volt néhány ROM-ban található rutinra, de ezek helyét nem ismertem, mivel ilyen leírás nem kapható a Primóhoz. Ezért készítettem magamnak egy BASIC nyelvű disassembler programot, s ezzel vizsgálom a memóriát. A vizsgálódás során már elég sok rutin helyét megállapítottam. Ha Önök is így gondolják, ezeket a kezdőcímeket néhány hozzáfűzéssel együtt megírom, biztosan vannak, akiket érdekelne. Bár a gyártó feladata lenne, hogy tájékoztassa a Primo-tulajdonosokat ilyen és ehhez hasonló kérdésekről, például egy könyv formájában.

A Primóval kapcsolatos vitához csak annyit: teljesen egyetértek Mohila Károllyal. Valószínűleg sokkal könnyebben írhatnék programokat, ha a kezemben volna egy részletes leírás a Primóról, azonban így, hogy gyakorlatilag mindent magamnak kell megkeresnem a ROM-ban, sok és sokféle problémával kerülök szembe, olyanokkal, amelyekkel egy könyvet olvasva nem találkoznék, s melyeket megoldva programozástechnikai fogásokat ismerek meg, s ezeket saját programjaimban is használni tudom. Mindezek ellenére jól jönne egy használható leírás.

Magával a géppel nagyon meg vagyok elégedve. Egyszer kellett biztosítékot cserélni, de ettől eltekintve tökéletesen működik, pedig nyáron néha egész nap be van kapcsolva. Lehet, hogy mi „kifogtunk" egy átlagosnál jobb minőségű gépet, de azok a problémák, amiket eddig a Primóval kapcsolatban hallottam, olvastam, még sosem jelentkeztek. A billentyűzet pont megfelelő érzékenységű — bár a gyakrabban használt billentyűknél kopik —, a tévé képe teljesen jó, a magnóra olvasás biztonsága — egy Sanyo M2502U típusú magnóval — gyakorlatilag 100%-os. Nagyon megszerettem ezt a kis gépet, és szerintem kár lenne a tájékoztatás hiánya miatt azok kedvét elvenni tőle, akiknek nincs türelmük vagy idejük végigbogarászni a memóriát. (Kíváncsian várom a bővített változat, a Pro-Primo ismertetőjét.)

S ha már így belejöttem a dicsérésbe, akkor az Önök lapja sem maradhat ki. Nagyon jól elkészített, kitűnő lapnak tartom. Azt hiszem, legértékesebb tulajdonsága a sokszínűsége; a számítástechnika rengeteg oldalának bemutatása lendületessé, ezért élvezetessé teszi a lap szerkezetét. Az arányok is nagyon jól kialakultak. Szerintem a kész programok mellett egy árnyalattal több memóriatérképre, táblázatra lenne szükség, vagyis olyan konkrét adatokra, „finomságokra", amelyek ötletek adásával elősegítik az otthoni programkészítést. Hasonló dolgokra gondolok, mint a tavalyi Z80 utasításstáblázat, amit remek ötletnek tartok.

Sokszor van úgy, hogy az egyik olvasói levél felel a másikra. Ez a levél is némileg felelet Képíró Róbert kritikájára. A Primóval nekünk sincsenek rossz tapasztalataink, a részletesebb leírásokat mások is hiányolják.

Smid Iván, Borsodnádasd,

Kossuth út 20. 3671

Én egy éve foglalkozom számítástechnikával C—64-esen. December óta rendszeres olvasója vagyok a Magazinnak, sőt most sikerült beszereznem néhány régebbi példányt. Többnyire elégedett vagyok a lappal, de pl. szívesen látnám, ha az „Adok — veszek — cserélek" rovat nagyobb terjedelmű volna. Tulajdonképpen nem is ezért írtam, inkább azért, mert van néhány problémám, ami azt hiszem közérdekű. Először az örökzöld probléma: a szoftverdokumentáció. Magyarországon a programok nagy része csere útján terjedt el. Sajnos ez az elterjedés a dokumentációkra már csak nagyon kismértékben érvényes. Nekem is rengeteg olyan programom van, amit sajnos kezelési utasítás híján csak korlátozott mértékben tudok használni. Itt elsősorban a játékprogramokat értem, de idetartoznak még a legkülönfélébb egyéb programok is. Ami most konkrétan érdekelne, az a Flight Simulation 2. Ugyanis sikerült kb. egy fél éve átvennem, s már azóta bajlódom vele. Egy pár dologra már rájöttem, de sajnos ez nagyon kevés. Nemrégen kezembe akadt egy könyv, az 1001 Játék és a Graphic Basic címmel. Ebben találtam néhány utasítást a szimulátor kezeléséhez, s itt azt írták róla, hogy az angol nyelvű 220 oldalas dokumentációját már sokan kritizálták a szűkszavúsága miatt. Abban kérem az Önök segítségét, hogy amennyiben valami módon tudnának segíteni, annak nagyon örülnék. S van még egy másik program is, a Collossus Chess 2.0 nevű, amivel ugyan sakkozni már tudok, de még sok mindent nem ismerek rajta. Kérem, ha tudnak, segítsenek!

S végül az utolsó problémám. Az Ötlet egyik számában megjelent a Commodore Újság. Abban olvastam, hogy be lehet lépni a Commodore Szekcióba különböző ún. páholybérlettel. A diákpáholyban pl. 60 Ft a belépti díj. Nos én rögtön befizettem ezt az összeget (ennek több mint egy hónapja), és azóta semmi. Lehetséges, hogy ez nem Önökre tartozik, de azért leírtam, hátha más okul belőle.

Levelét azért közlöm, hátha az olvasók közül valaki segíteni tud, mi a kereskedelmi forgalomban kapható programokkal nem foglalkozunk.

Mészáros László, Leninváros,

Malinovszkij u. 16. F/4. 3580

A BME Villamosmérnöki karának hallgatója vagyok, s rendszeresen olvasom (az általunk „igazi Impulzusnak" nevezett) kari lapot. Ebben mindig szerepel egy „Szakmai Oldal" rovat, amely az elektronika legújabb (legfeljebb néhány hónapos) csodáiról, érdekességeiről számol be röviden, viszonylag közérhető formában. Úgy gondolom, a Mikromagazin oldalait is színesítené egy-egy ilyen vidám hangulatú, de komoly témájú cikk. Mivel az Impulzus kéthetenként jelenik meg, s ára számonként csupán 1 forint, különösebb anyagi megterhelés nélkül még választhatnak is havi két ismertető közül. Ízelítőül mellékelten elküldöm a legújabban (pontosabban az Impi utolsó tavaszi számában) megjelent Szakmai Oldalt, amely a Fairchild cég egyik legfrissebb termékét, egy 32-bites szuperprocesszort ismerteti.

Remélem, mind a cikk tartalma, mind — kissé könnyed — fogalmazása megnyeri tetszésüket.

Nem ismertem az „igazi Impulzus"-t, nem rossz lap, a címlap fényképét itt közlöm.

A cikk tartalma tetszett, a stílusa más, mint a µM-ban megszokott cikkeké. Általában nem közlünk máshol megjelent cikkeket, de szívesen vennénk, ha a szerkesztőség tagjai eredeti írásokat küldenének — a µM stílusában.

Sok sikert kívánok.

Fülöp Miklós, Budapest,

XIV. utca 20. 1224

Szelet vetett és vihart aratott dr. Simonyi Endre pályázata, amit a Számítástechnika 86. áprilisi számában, fizetett hirdetés formájában tett közzé. A vihar egyik részese — a Rádió 168 óra c. műsora mellett — én voltam. Levelem olvasható a lap 86. júniusi számában. A pályázatot kiíró ugyanott válaszol is rá, de magyarázat olvasható az Impulzus legutóbbi számában is.

Nos, én nem vonom kétségbe dr. Simonyi szándékának nemességét, azt, hogy keserű tapasztalatain okulva másokat megmentsen ezen keserűségektől. Jobb lett volna azonban ezt az akciót nem „pályázat"-nak nevezni, nem pályázat formájában lebonyolítani. Jobb lett volna erre PJT, GMK vagy akármi. Például az egyesület. Ma már tíz személy egyetértésével alakítható egyesület, ahol tagdíjat lehet szedni, ötleteket lehet nyilvántartani — díjazás fejében — szakértőket lehet felkérni stb. Végül, ha befut a százezer forintot meghaladó alkotás, akkor a többlet feléből támogatni lehet a siker kapujában álló, de meg nem értett „tagok" nehéz ötleteinek megvalósítását.

A 8. számban beszámoltunk a pályázat eredményéről.

Fekete Sándor, Oradea 3700 Breiner Béla u. 59/A Bihor Romania

Nagyon szépen köszönöm a levelét, amelyben közölte a digitális audio szakembernek a címét. De nem lehetne egy ilyen felhívást három sorban közölni a µM-ban az én nevemben a levelezési rovatban? Amely valahogy úgy szóljon, hogy mindazok, akik digitális audio problémával foglalkoznak, próbáljanak közös kapcsolatot felvenni és egy kis klubot létesíteni. Az egyik alcsoport a digitális audióval, a másik a digitális videóval foglalkozna, mert már erre is kacsingatni kellene és a gyakorlat felé venni az irányt.

Ime a felhívás, akik digitális audio-technika iránt érdeklődnek, felvehetik a kapcsolatot Fekete Sándorral.

Ezzel zárom is a szerkesztő postáját, megismételve kérésemet, hogy a következő évi számok szerkesztéséhez várjuk olvasóink ötleteit és javaslatait.

Kovács Győző

BITEK ÉS FIGURÁK

Állásértékelés VI.

Királytámadás és védelem II.

A királytámadás és -védelem első részének ismertetése kapcsán a sakktábla különféle súlyozásait tárgyaltuk. Most más szempontokat veszünk figyelembe.

Nagyon fontos, hogy a király előtti gyalogok a megfelelő helyen legyenek. Feleslegesen ne lépjünk velük, mert a királyállás kritikusan meggyengülhet. Rövid sánc esetén fontos az f3/f6 és h3/h6 mezők védelme, mert ha itt a középjátékban ellenséges figurák jelennek meg, akkor általában el is dőlt a játszma sorsa. Az ellentétes oldalú sáncolás esetében a támadás legjellegzetesebb formája a kölcsönös gyalog-előnyomulás, a vonalnyitás az ellenfél királyállásával szemben, valamint az átütő erejű tisztrohamozás. Itt az nyer, aki előbb jut el az ellenfél királyához.

Az alábbi, Polgár I.—Flesch J. játszma, amelyet 1965-ben a XX. magyar bajnokságon játszottak, jó példa az említett elvek jelentőségének bemutatásához.

1. d4, Hf6 2. c4, e6 3. Hf3, d5 4. Hc3, c6 5. e3, Hbd7 6. Fd3, Fb4 7. Vc2, dc: A centrum ilyen korai feladása nem szükségszerű. Sötét megvárhatta volna, míg világos előbb-utóbb a3-at játszik.

8. Fc4:, Fd6 9. Fd2! Sötét elárulta, hogy e6-e5-re és világos rövid sánc esetén király elleni tiszttámadásra törekszik. Ezért világos fordít egyet a kormányrúdon, és előkészíti a hosszú oldalra történő sáncolást.

9. —, 0—0 10. 0—0—0, Ve7 11. Bhe1, b5 12. Fd3, Fb7 13. e4!, e5 14. de:, Fe5: 15. He5:, He5: 16. Ff1, Hfd7 17. Kb1, Hb6 Sötét módszeresen elfoglalja a c4 pontot, de mire jó ez? A világos gyalogoknak a királyszárny megtisztulása által létrejött mozgékonysága veszélyesebb.

18. f4, Hec4 19. Fc1, a5 20. e5, Fa6 21. He4, a4 22. Bd3, Fc8 A fenyegető *Bh3 és Hf6+* valamint a *Vh7:+* ellen.

23. g4!, Fg4: Nyilván nem mohóság eredménye, hanem az előbbi fenyegetés felújítása ellen veszi fel a küzdelmet. Ez azonban megnyitja a sötét király előtti g vonalat, ami újabb fenyegetéseket idéz elő.

24. Fh3!? Sajátos megoldás, erős volt a kézenfekvő 24. Bg3 is. **24. —, Fe6 25. Bg1, Kh8 26. Bg7:!!, Bg8** A bástya ütésére 27. Bg3+, Kh8 28. Hf6! nyer.

27. Bh7:+!, Kh7: 28. Hf6+, Kh8 29. Bg3!, Bg6! 30. Fe6: (Itt egyszerűbben nyert 30. *Bg6:, fg: 31. Vg6:, Fg8 32. Ff5!)*

30. —, Ve6: 31. f5! *(31. Bg6:?, fg: 32. Vg6:-ra a Vh3 véd!)*

31. —, Ve5: 32. Bh3+, Kg7 33. Hh5+, Kf8 34. fg:, fg: 35. Vg6: Anyagilag az állás a bonyodalmak után kiegyenlítődött, de a sötét király teljesen védtelenné vált.

35. —, Ke7 36. Hf4!, Bh8 37. Hd3, Vd4 38. Bf3, Hd6 39. Fg5+, Kd7 40. Ff6, Vd5 (40. *—, Ve4-re 41. Ve4:, He4: 42. He5+!* megvédi az f3 bástyát *Hd2+* esetére, és csak azután üt *h8*-ra.)

41. Fh8:, Hbc4 42. Vg7+, Kd8 43. Bf8+, He8 44. Vd4, Ke7 45. Vd5: Sötét feladta.

Ebben a játszmában konkrétan láttuk egy gyalogroham technikáját, a királyállás előtti nyílt vonalak szerepét, az f6 és a h7 pont gyengeségén alapuló kombinációk jelentőségét.

A következő játszma mindkét félnek azonosan rövid oldara történő sáncolása mellett példázza a király elleni támadást. Ebben az esetben a támadó félnek a változatokat mélyebben és pontosabban kell előre számítania, mert a támadás során a saját királyállását is meggyengíti. Azt kell pontosan értékelnie, hogy királyállásának gyengülése vagy az ellenséges királyállás elleni támadás nyom-e többet a latban.

A Bilek I.—F. Gheorghiu játszmában (Bukarest, 1968.) világos kinyitja a f vonalat, így királyállásának f2 pontja meggyengül, de cserébe sötét g7 gyalogját eltávolítja a g vonalról, így a sötét király előtti gyalog eltűnik, a vonal félig nyitottá válik. A g vonal megnyitása miatt sötét nem tudja védeni a mattfenyegetéseket.

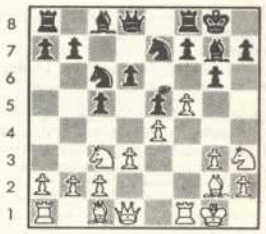
1. e4, c5 2. Hc3, Hc6 3. g3, g6 4. Fg2, Fg7 5. d3, d6 A zárt szicíliai védelem klasszikus változata.

6. f4, e5 Az f gyalog korai lépésére, amelyet különösen Spasszkij alkalmazott sikerrel Geller elleni páros mérkőzésén, a legjobb valószínűleg az e6, Hge7-es felállás.

7. Hh3!? (Itt általában a 7. Hf3, Hge7 8. 0—0, 0—0 9. Fe3, ef: 10. gf:, f5 változatot választják, ami egyenlő játékra vezet. Világos újításának az az előnye, hogy ha sötét gyanútlanul a szokásos fejlődési módot választja, mint ebben a játszmában is, akkor meglepetésszerű, valószínűleg döntő támadásnak teszi ki magát.)

7. —, Hge7 8. 0—0, 0—0? Nem gondolnánk, hogy ez a természetes lépés máris döntő hiba. Hd4! tartaná az állás egyensúlyát.

9. f5!! Gyalog- és minőségáldozat bevezetése. Sötét számára a legjobb volt most az ajándék elhárítása *9. —, f6*-tal, amire világos kissé jobban áll.



9. —, gf:? 10. ef: Ff5: 11. Bf5:!!, Hf5: 12. Fe4! Hfd4 (12. — Fe7-re 13. Fh7:+! Kh7: 14. Vh5+, Kg8 15. Hg5, Be8 16. Vf7:+, Kh8 19. He6+ vezérnyerés szép befejezés lett volna.)

13. Vh5!, Be8 (nem segít *13. —, f5* sem, *14. Fd5+, Kh8 15. Hg5, h6 16. Vg6* miatt.)

14. Vh7:+, Kf8 15. Fg5!, Vd7 16. Hd5!, Be6 Fenyegetett 17. *Fh6 és Hf6,* mindkettő azonnali nyeréssel.

17. Bf1, Hc2: 18. Fg6!, H2d4 19. Fh6! sötét feladta.

Ezekben a játszmákban szép példákat láttunk az ellenfél királyának megtámadására. A nagymesterek az ilyen támadásokat sokszor nem számítják végig, intuíciójukat használják fel, és nagy tapasztalatukra támaszkodva indítják el a támadást. Ilyen képessége a számítógépnek természetesen nincs. Mielőtt azonban pontosabban megvizsgálnánk, hogy milyen szempontokat kell a számítógépnek figyelembe vennie, hogy támadása sikeres legyen, érdemes még egy játszmát tüzetesebben megvizsgálni, amelyben az egyik fél az ellenfele meggyengült királyállására építi a támadását. Erre jó példa a Ballá Z.—R. Spielmann játszma, melyet következő számunkban ismertetünk.

KOVÁCS P. ATTILA

A Mephisto-sztori

Korábbi írásainkban rámutattunk azokra a tényezőkre, amelyeket célszerű figyelembe venni, amikor valaki arról dönt, hogy milyen sakkszámítógépet vásároljon. A következőkben az egyes gyárak termékeinek tulajdonságait, képességeit, hozzávetőleges árát ismertetjük.

Verseny a világpiacon

Az elmúlt esztendők tapasztalatai azt mutatják, hogy a sakkszámítógépek iránt a világban megmutatkozó érdeklődést nem több, mint öt-hat gyár ki tudja elégíteni. Ezek között igen nagy a verseny, amelynek tényezői: a játéktudás, a kivitel, a számítógép nyújtotta egyéb szolgáltatások, információk stb. az eddig közölthekből ismertek olvasóink előtt. S ezekhez járul a vásárlási döntést befolyásoló egyik legfontosabb komponens: az ár. A versenyben nem egy cég elhullott már, itt-ott újak tűnnek fel; a piac állandó mozgásban van. Arra törekszünk, hogy azokat a cégeket, s ezeknek olyan termékeit ismertessük, amelyek ma a piacon vannak; a régebbiekre csak itt-ott, a nyújtott kép teljesebbé tétele érdekében utalunk. Elsőként az élenjáró müncheni Hegener + Glaser cég méltán népszerű Mephistóival foglakozunk.

A Mephisto I és II

Több USA-beli és hongkongi cég már készülékeinek harmadik generációját hozta forgalomba, amikor 1980-ban — körülbelül az angol Intelligent Chess-szel egy időben — megjelent az első európai gyártású és összeszerelésű sakkszámítógép, az NSZK-beli Mephisto.

Mind külsejében, mind kezelésmódjában lényegesen eltért a többitől; egyszerű, könnyen kezelhető, üzembiztos kicsi fekete doboz. Térfogata 17 × 11 × 4,5 cm, súlya ½ kg. 17 nyomógombja van, amelyek egyedi, kettős vagy hármas funkciót töltenek be. Kijelzője négy betű, illetve szám megjelenítésére alkalmas. Elemmel és hálózatról is működtethető.

A lépéseket az indulási és érkezési mezőnek megfelelő gombok megnyomásával lehet betáplálni (például E2E4), ami megjelenik a kijelzőn, s ugyanígy jelenik meg a válaszlépés. A játszmát természetesen sakktáblán követni kell. A többi sakkszámítógéphez hasonlóan beállítható többféle fokozatra, s ez a rendkívül egyszerű hardver az elhasznált gondolkodási idő, az éppen elemzett lépés, az ellenfélnek ajánlott lépés (ha az van soron) jelzésére is képes; továbbá hadállás, feladvány táplálható be, amelyeket a maga módján megold.

Programját Thomas Nitsche és Elmar Henne készítette, akik folyamatosan dolgoztak tovább annak fejlesztésén. Ez annál sürgősebb volt, mert a Hegener + Glaser cég a Mephistóval első ízben valósította meg a moduláris rendszert, a program cserélhetőségét. Így nem kellett új készüléket vásárolnia annak, aki alig több mint egy esztendővel később, 1981 őszén meg akarta szerezni a lényegesen továbbfejlesztett Mephisto II programot. Hozzátesszük: a Mephisto ára 600 nyugatnémet márka volt, a Mephisto II-é 700 márka, de a programé mindössze 200, és amikor az új program megjelent, a régivel ellátott készülékeket már leszállított áron, kb. 500 márkáért árusították.

A Mephisto I-gyel és II-vel kapcsolatos adatokat azért ismertetjük, mert ezek a számítógépek ma is forgalomban vannak, s a második program napjainkban is jó középkategóriát képvisel. Ma már 500 márka körül kapható sok helyen a Mephisto II is, közben ugyanis elkészült ugyanerre a hardverre a harmadik program. Moduláris rendszerükkel és viszonylag olcsó áraikkal a készülékek gyorsan népszerűvé váltak, annak ellenére, hogy a Mephisto a „nagyok" versenyébe balszerencsés körülmények miatt csak 1982-ben tudott bekapcsolódni. 1981 szeptemberében Travemündében rendezték a mikroszámítógépek 2. világbajnokságát, s Mephistót (saját országában!) kizárták a nevezők közül, mert a verseny indulásakor még nem volt az üzletekben kapható, ami a kereskedelmi forgalomban lévő készülékek kategóriájában a részvétel feltétele volt. A kísérleti programok csoportjában nem lett volna esélye, ezek a már gyártás alatt lévőkkel szemben általában pár hónapnyi előnnyel rendelkeznek.

A Hegener + Glaser cég a második programmal jelentős fejlesztéseket hajtott végre. Az első az volt, hogy 1982 őszétől több mint 50 százalékkal gyorsabb — 6,1 MHz sebességű — processzorral hozták forgalomba, változatlan áron. Ezt házilag nem lehetett cserélni, de a gyárban bárkinek beépítették az új processzort. Ezzel a készülék játékereje megnövekedett, hiszen ugyanannyi idő alatt mélyebben tudott számítani, és ebben az évben már jó eredményeket ért el a nemzetközi versenyeken.

Ugyanakkor forgalomba hoztak a 64 mező alatt mágneses szenzorokkal ellátott, kétféle sakktáblát: egyet műanyagból, amelyet *ESB 3000*, s egy másik, luxuskivitelűt fából, melyet *ESB 6000* jellel láttak el. Ennél szebb kivitelű sakkszámítógép azóta sem jelent meg a világon! Méretük 30 × 30 × 2,5, illetve 50 × 50 × 8 cm. Ezek fiókjába kell a fekete dobozt behelyezni, és a fiókban a táblához lapos kábellel csatlakoztatni, aminek eredményeképpen a lépéseket egyszerűen a talpukban mágnest tartalmazó figuráknak az egyik mezőről a másikra történő áthelyezése útján lehet a táblán megtenni. Minden mező sarkában kis dióda van. Két dióda kigyulladása jelzi az indulási és érkezési mezőt, és ugyanígy jelennek meg a számítógép ellenlépései. Ha a fiókot becsukjuk, olyan, mintha a puszta sakk-készlet ellen játszanánk! Ha viszont kinyitjuk, a számítógép kijelzőjén változatlanul megkaphatjuk mindazokat az információkat, amelyekről fentebb szóltunk. Az ESB 3000-es 400 márkáért, az ESB 6000 pedig 900 márkáért kapható. A valóban gyönyörű tábla és bábkészlet jóval többe kerül a számítógépnél! (A legújabb típusú Mephistók megjelenése óta, amiről legközelebb szólunk, ezeket is kínálják alkalmi vételre.)

Ezzel a müncheni cég nemcsak a piac legszebb sakkszámítógépét hozta forgalomba, de egyben elsők között alkalmazta az azóta szinte egyeduralkodóvá vált szenzoros technikát. Lefelé is szélesítette választékát, kialakítva a *Mephisto Juniort*, amely zsebben hordható, és játéka olcsó árához képest kielégítő. 300 márkáért árulják az üzletekben.

A Mephisto III

Nitsche és Henne, a cég programozói még 1982-ben közzétették tanulmányukat „Das Mephisto-Konzept" (Mephisto koncepció) címen, amely forradalmi változásokat helyezett kilátásba a mikrogépek programozása terén. Lényege, hogy az elemzés mélységének fokozása érdekében a lehetséges lépések között alapos előválogatást hajtanak végre; de a tanulmány emellett feltárt egy sor olyan stratégiai és taktikai alapelemet is, amellyel a gépi játék lényeges finomítására törekedtek. Amikor a III-as program 1983-ban a piacra került, kiderült, hogy a vártnál sokkal erőteljesebben tér el minden addig ismert sakkprogramtól.

Itt csak utalni kívánunk rá, hogy a program másodpercenként mintegy 5-6 lépést vizsgál — illetve értékeli az utánuk bekövetkező hadállásokat —, szemben a korábbi Mephistók — és más programok — másodpercenkénti 600-1000-es vizsgálati sebességével. Ennek fejében az értékelésre kiválasztott variációkat nagy mélységig, a szerzők szerint versenyfokozaton 19 féllépésig (ami túlzásnak tűnik) elemzi. Ezzel mindenesetre óriási léptek előre a programozási technikában, azzal együtt, hogy a gyakorlat bebizonyította: a program valóban képes több lépésre történő számítást igénylő kombinációkra, de ugyanakkor nemegyszer eléggé elemi hibákat vét: a várható ellenlépések közül esetleg erőseket is kihagy számításainak köréből. Máig sem dőlt el, hogy valójában a Mephisto III lényegesen jobban sakkozik-e a Mephisto II-nél, azzal együtt, hogy a III-as program hatékonyságát hamarosan igen jelentős hardverfejlesztéssel is fokozták.

Mephisto I és II

A Mephisto III ugyanis még sokkal több „mellékszolgáltatásra" képes elődjénél. A cég fejlesztőrészlege mintaszerűen használta ki a kisméretű számítógép viszonylag szűk és sokban kötött adottságait. Aki az új modult megvásárolta, öt megváltozott feliratú nyomógombot kapott hozzá, amelyek ugyanúgy mint a program, házilag kicserélhetők a régiekkel. A nyomógombok funkciója részben megváltozott, lényegesen megnőtt; ugyanakkor a kezelési mód egyszerű elemei — a lépések megtétele, viszavétele, az éppen elemzett lépés lekérése stb. — ugyanazok maradtak. Az új program tárolója 32 kbájtra bővült, 9 fokozatú, de beállítható ezektől eltérő időbeosztás is. A kijelzőre váltakozva le lehet hívni a gondolkodási időt, az állás értékelését hexadecimális számrendszerrel, az elemzés mélységét, vagyis hogy a program számításában hányadik féllépés hányadik változatánál tart, a soron lévő lépés számsorrendjét, hogy hányadik lépésnél tart a játszma, öt féllépésig az éppen elemzett változatot. Bővült a betáplált megnyitástár és pótolták a korábbi Mephisto-programok egy hiányosságát: a nyolcadik, illetve első sort elérő gyalog tetszés szerinti tisztté alakulhat át (a korábbi programok csak vezérátváltozást ismertek). Az új programnak egyben „oktató" képessége is van, megfelelő gombnyomásra megmutatja egy elkövetett hiba eredetét.

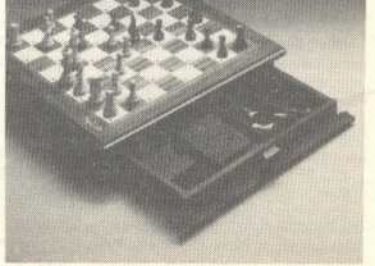
Kijelenthetjük: a mai legkorszerűbb sakkszámítógépek sem képesek a váltott sakkjátszmákkal kapcsolatban több információ nyújtására, mint a Mephisto III program, amely természetesen szintén működtethető az ESB sakktáblákhoz csatlakoztatva.

Ára 300 márka, a II-es program cseréjéért 200 márkát kérnek. A Mephisto III készülék hardverrel együtt azonban nem kerül többe a Mephisto II-nél, 700 márka az ára; a komplett Mephisto ESB III ugyancsak változatlanul 1600 márkába kerül.

Legközelebb a Mephistók legújabb, 1983 óta megjelent típusait ismertetjük.

LINDNER LÁSZLÓ

Mephisto II és III ESB



HOMEHOKI

A HOMEHOKI hasonlóan működik, mint a hazánkban is kapható tv-játék hokija. Két játékos játszhatja. Mindkettőnek két-két ütője van, amiket gombnyomással irányíthat. Az egyik játékos az A és Z billentyűket, a másik az L és > billentyűket használhatja. A játék megkezdése előtt a nehézségi fokot 1-től 9-ig be lehet állítani, ahol 1 a legerősebb fokozat. A játék befejezésekor az N billentyűvel lehet új játszmát kezdeményezni.

A teljes program gépi kódban íródott. Betöltési tárcíme: 5ØØØH. Innen indítható monitorból vagy CALL 2Ø48Ø BASIC paranccsal. Mivel a rutin végtelenített, csak RESET-tel állítható le.

A rutin 5ØØØ-től 5Ø16-ig terjedő része a fejléc kiíratását és a nehézségi fok bevételét végzi. 5Ø17—5Ø93 programrész a pályát rajzolja ki és a változók kezdőértékét állítja be. A játékprogram vezérlését az 5Ø94 és 5ØE7 közötti rutin végzi. A program futását szinkronizálja a képszerkesztéshez úgy, hogy a két szomszédos képszerkesztés között a labda és az ütők mozgatását irányítja. A nehézségi fok ebben az esetben azt jelenti, hogy egy-egy labdamozgatáshoz hány ütőmozgatás tartozzon. Így az ütők sebessége állandó, a labdáé pedig a nehézségi foktól függ.

A labda irányváltoztatásakor — attól függően, mi változtatta meg irányát — hangjelzés keletkezik. Ha a labda kapuba jut, vagyis * karakterbe ütközik, a hangjelzés változása mellett a gól „elkönyvelése" is megtörténik. Amennyiben valamelyik játékos elérte a 15 pontot, a rutin „befagy". Erre azért van szükség, mert a játék hevében az utolsó gólnál lenyomott gomb a játék továbbfutását eredményezné, megszüntetve az eredmény kiiratását! Ebből az állapotból az N gomb megnyomásával lehet kilépni.

Az ütők két jól elkülöníthető részből állnak. A négy karakter közül a felső kettő n, az alsó kettő pedig u betű. Ha a labda az n betűnek ütközik, eredeti irányától függetlenül felfelé, u betűről pedig lefelé pattan. Ezzel a megoldással a labdát meg lehet csavarni, megetetve vele az ellenfelet! Az ütők hátulról sem átlátszóak, így tapasztalatunk szerint meglehetősen gyakori jelenség az öngól.

Azoknak a felhasználóknak, akik szeretnének belepiszkálni a programba, közöljük a felhasznált rutinok kezdőcímeit és értelmüket.

5ØE9 Eredménykiiratő rutin.

5114 Szövegkiiratő rutin, a DE regiszter mutat a kiírandó szöveg első karakterére, míg a szöveg utolsó 6ØH karaktere zárja azt.

511C Vonalhúzó rutin, HL-ben a kezdőcím, DE-ben a lépésköz, és az A-ban a kirakandó karakter ASCII kódja. B regiszterbe kell írni a kirakandó karakterek számát. Ezt a rutint a pályarajzoló program hívja.

5121 Ütő 1. gombvizsgáló rutin.

5136 Ütő 2. gombvizsgáló rutin.

```
HOMEHOKI

5000   11 8B 52 21 01 C0 CD 14
5008   51 DF FE 3A 30 FB FE 31
5010   38 F7 E6 0F 32 20 53 3E
5018   0C EF 21 7D C0 06 21 36
5020   A0 23 10 FB 21 9D C3 06
5028   21 36 A0 23 10 FB 21 A5
5030   C0 11 28 00 06 04 3E A0
5038   CD 1C 51 21 C5 C0 06 04
5040   CD 1C 51 21 FD C2 06 04
5048   CD 1C 51 21 1D C3 06 04
5050   CD 1C 51 3E 2A 21 1B C1
5058   06 0D CD 1C 51 21 3F C1
5060   06 0D CD 1C 51 21 1C C1
5068   11 E0 01 06 02 CD 1C 51
5070   21 3E C1 06 02 CD 1C 51
5078   21 28 01 22 18 53 21 46
5080   C1 22 1A 53 21 16 C2 22
5088   1C 53 21 FF 00 22 1E 53
5090   AF 32 21 53 3E 01 F5 ED
5098   4B 1E 53 CD E9 50 78 FE
50A0   0F 28 05 79 FE 0F 20 09
50A8   F1 DF FE 4E 20 FB C3 00
50B0   50 F1 3D 28 08 21 13 C1
50B8   22 16 53 18 03 21 F7 C0
50C0   22 16 53 3A 21 53 A7 C4
50C8   77 52 AF 32 21 53 3A 20
53D0   53 47 78 C5 CD 21 51 CD
50D8   36 51 C1 10 F5 CD A4 51
50E0   30 E1 2A 16 53 36 60 18
50E8   AD F5 C5 21 30 C0 11 09
50F0   53 CD 14 51 C1 F1 3D 20
50F8   03 0C 18 01 04 ED 43 1E
5100   53 AF 80 27 CD A5 01 2A
5108   14 40 CD 14 51 AF 81 27
5110   CD A5 01 C9 22 14 40 E7
5118   C8 EF 18 FB 77 19 10 FC
5120   C9 3A 6F 3A 2F A7 C8 FE
5128   06 C8 FE 04 2A 1A 53 CD
5130   4B 51 22 1A 53 C9 3A D7
5138   3A 2F A7 C8 FE 50 C8 FE
5140   40 2A 1C 53 CD 4B 51 22
5148   1C 53 C9 E5 16 00 30 24
5150   1E 28 A7 ED 52 7E 1E 14
5158   19 B6 E1 FE 60 20 0F 1E
5160   78 19 36 60 1E 14 19 36
5168   60 1E B4 A7 ED 52 E5 CD
5170   8B 51 E1 C9 1E A0 19 7E
5178   1E 14 19 B6 E1 FE 60 20
5180   ED 36 60 1E 14 19 36 60
5188   19 18 E3 E5 CD 93 51 E1
5190   1E 14 19 1E 28 3E ED 3C
5198   05 02 77 19 10 FC C8 AF
51A0   3E F5 18 F4 DD 2A 16 53
51A8   3A 18 53 FE 28 20 16 DD
51B0   7E 27 DD B6 29 E6 02 28
51B8   20 3E 0F 32 21 53 3E A8
51C0   32 18 53 18 14 DD 7E D9
51C8   DD B6 D7 E6 10 28 0A 3E
51D0   0F 32 21 53 3E 28 32 18
51D8   53 3A 18 53 FE 28 20 13
51E0   DD 7E 28 FE 60 28 1D 3E
51E8   0F 32 21 53 3E A8 32 18
51F0   53 18 11 DD 7E D8 FE 60
51F8   28 0A 3E 0F 32 21 53 3E
5200   28 32 18 53 3A 19 53 FE
5208   01 20 20 DD 7E 01 FE 2A
5210   20 09 3E 1F 32 21 53 3E
5218   02 37 C9 FE 60 28 2A 3E
5220   0F 32 21 53 3E 91 32 19
5228   53 18 1E DD 7E FF FE 2A
5230   20 09 3E 1F 32 21 53 3E
5238   01 37 C9 FE 60 28 0A 3E
5240   0F 32 21 53 3E 01 32 19
5248   53 3A 19 53 DD E5 E1 36
5250   60 16 00 FE 01 20 04 5F
5258   19 18 05 1E 01 A7 ED 52
5260   3A 18 53 FE 28 20 04 5F
5268   19 18 05 1E 28 A7 ED 52
5270   36 91 22 16 53 A7 C9 6F
5278   17 E6 3F 4F 26 3C 45 CB
5280   1E 10 FC 7C EE 3C 67 0D
5288   20 F4 C9 0C 08 08 08 08
5290   08 08 0E 20 20 20 20 20
5298   20 A0 A0 A0 A0 A0 A0 A0
52A0   A0 A0 A0 A0 0D 0E 20 20
52A8   20 20 20 20 A0 0E 0E A0
52B0   0D 0E 20 20 20 20 20 20
52B8   A0 20 46 20 4F 20 43 20
52C0   49 20 A0 0D 0E 20 20 20
52C8   20 20 20 A0 0E 0E A0 0D
52D0   0E 20 20 20 20 20 20 A0
52D8   A0 A0 A0 A0 A0 A0 A0 A0
52E0   A0 A0 0D 0D 0D 20 20 20
52E8   20 20 49 72 64 20 62 65
52F0   20 61 20 6E 65 66 7C 7A
52F8   73 7C 67 69 20 66 6F 6B
5300   6F 74 20 28 31 2D 39 29
5308   60 5B 6C 6C 7B 73 0E 60
5310   20 20 3A 20 20 60 00 00
```

HOMELAB

514B Ütőmozgató rutin. A HL regiszterbe kell beírni egy-egy ütőpár bal oldali tagja legfelső karakterének kezdőcímét, valamint a CY-ba a mozgatás irányát: CY = 1 felfelé, CY = Ø lefelé. Ezeket az értékeket a gombvizsgáló rutinok állítják be. Ha az ütők valamelyike falba vagy labdába ütközik, a léptetés nem történik meg, tehát a labdát nem lehet a falhoz szorítva „kipukkasztani"! Visszatéréskor a HL regiszter tartalmazza az említett ütő új pozícióját. Ez a programrész mind a két ütőt egyszerre elmozdítja!

51A4 Labda vagy korong mozgatását végző rutin. A rendszerváltozók közül automatikusan elővesszi a labda pozícióját, majd a labda környezetét vizsgálva elvégzi a mozgatást. Ha a labda ütközik, akkor beállítja a labda új irányát és egy változón keresztül jelzi, hogy milyen hangjelzés szükséges ekkor. Ha a labda útját úgy sikerül elállni, hogy az semerre sem tud pattanni, akkor mindig az ütő „húzza a rövidebbet".

5277 Hanggeneráló rutin. A hangjelzés magasságát az L regiszterbe kell beírni, amit a program számlálónak használ fel. Ezt a rutint a labdamozgató rutin által szabályozva, a vezérlő program feltételesen hívja. Eredetileg a számláló úgy van beállítva, hogy a hangjelzés ideje beférjen két képszerkesztés közé. Ha ez nem így lenne, akkor irányváltozásnál a labda mozgása lelassulna.

KANICS MIKLÓS

ZX—SPECTRUM

Kiegészítés a „Tréningmódok" c. cikkhez

A februári számban olvastam Pintér Tibor cikkét a különböző játékok örökéletűvé tételéről. A Manic Minerben szerinte maximum 32 élet lehetséges, én viszont „örök életet" ajánlok: a RAND USR elé POKE 35136,Ø. Ez a POKE használható a játék második változatához, a Manic Miner 2-höz is.

Sabre Wulf: a BASIC loaderben PRINT USR elé POKE 44786,Ø-t beírva érdekes hatást érhetünk el: ellenség nélkül, veszett sebességgel szaladgálhatunk a dzsungelben. Ha mégis kedve támadna valakinek harcolni, az nyomja meg a „FIRE" gombot! Ez a módszer különösen térképkészítésnél használható.

CZIRÁKI GÁBOR
LGL Software
Szekszárd

PÁLYÁZATI FELHÍVÁS

Az NSZJT, a szekszárdi Garay János Gimnázium és a Mikroszámítógép Magazin szerkesztősége ismét pályázatot hirdet az alábbi kategóriákban:

1. Új, önálló játékprogram készítése HT—1080Z, Sinclair Spectrum, Commodore 16 és 64 gépre.

2. Valamely tantárgyhoz kapcsolódó, a tanítási órát segítő, illetve a tanulók önálló tanulását támogató oktatóprogram készítése a fent említett gépekhez.

A pályázaton részt vehet minden általános és középiskolás tanuló, valamint első éves egyetemista.

A pályázat beadási határideje: 1987. január 31.

A programot mágnesszalag-kazettán (a kazettára többször felvéve), vagy mágneslemezen (floppy) rövid leírás kíséretében (mit tud, hogyan működik a játék, hanyadik osztály mely tantárgyának melyik anyagrészéhez kapcsolódik stb.), jeligével ellátva (külön zárt borítékban a név, lakcím vagy iskola) kérjük beküldeni a Mikroszámítógép Magazin szerkesztősége (Budapest, Fő u. 68. 1027) címére.

A szerkesztőségnek joga van a pályázaton részt vett programok közlésére, amiért a szokásos honoráriumot fizeti. A döntő 10-10 résztvevőjét az NJSZT tagjaiból, a Garay Gimnázium tanáraiból és a szerkesztőség munkatársaiból álló előzsűri választja ki.

A döntő — melyen az előzsűri által kiválasztott 10-10 program versenyez — Szekszárdon a Garay János Gimnáziumban rendezendő Garay-napok alkalmából, 1987 márciusában lesz.

A döntőbe jutott tanulókat a Garay János Gimnázium vendégül látja.

Mindkét kategóriában az első három helyezett programot díjazzuk, és mindkét kategóriában kiadjuk a közönség díját.

NJSZT
Dömölki Bálint
elnök

Garay János. Ált. Gim.
Zentai András
igazgató

Mikroszámítógép Magazin szerkesztőség
Kovács Győző
a szerkesztőbizottság vezetője

Építész tervező rendszer

Míg az általános célú tervezőprogramok rajzi elemekkel dolgoznak, egy új elvű hazai szoftver, az ArchiCAD először a háromdimenziós geometriai modellt építi fel, s ezt azután igazi modellezőprogramként kezeli. Ennek eredményeképpen

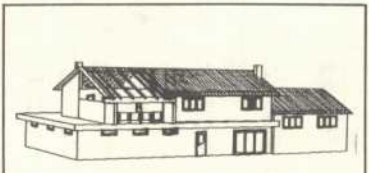
— megjeleníthetjük a homlokzati nézeteket, különféle axonometrikus (izometria, dimetria, monometria stb.) vagy valódi perspektív képet szerkeszthetünk kívánság szerinti nézőpontból:
— tetszőleges forgatásokat, tükrözéseket, torzításokat végezhetünk a modellel,
— a program önállóan tünteti el a takart vonalakat,
— fényiránytól függő árnyékolást alkalmazhatunk a plasztikus hatás kiemelésére,
— teljesen általános térbeli metszeteket készíthetünk.

Az ArchiCAD szoftver, mint a neve is jelzi, elsősorban építészeti tervezéshez készült. Ezért a felsorolt szokatlan lehetőségeken kívül a már klasszikusnak számító szolgáltatásokat is nyújtja: az ajtó, ablak és egyéb elemek kiválasztása a felhasználó által összeállított könyvtárból, ezek szabad paraméterezése, majd elhelyezése a falnyomvonalakba való illesztéssel; automatikus külső és belső méretezés: helyiségtérfogatok és -terület számítása stb.

A felhasználónak lehetősége van megnevezés- és egységár-katalógus kialakítására. Ennek alapján a program képes konszignációs listák és előzetes árkalkulációk készítésére. Mindez dBASE III formátumú állományok segítségével történik, melyekkel rendkívül széles lehetőség nyílik a katalógusok felépítésére, a listázások szempontjainak bővítésére, beleértve a más kapcsolódó programokhoz való illesztést is.

A Softinvest által forgalmazott ArchiCAD programrendszert a hazai Graphisoft Kisszövetkezet dolgozta ki IBM PC XT és AT, valamint az azzal kompatibilis gépek részére. A hazai alkalmazásba vétel mellett már Nyugat-Európában is szép piaci sikereket könyvelhet el: az NSZK-ban tucatnyi helyen megvásárolták, de referenciarendszer működik már Franciaországban is.

Az ArchiCAD által készített axonometrikus kép



Tizenöt éves

Születésnapját ünnepli a mikroprocesszor: az első, az Intel 4004 1971 novemberében került piacra. Ez volt az első olyan termék, amely egyetlen morzsában integrálta a számítógépek központi feldolgozó egységének valamennyi funkcióját. Az áramkör több mint 4000 tranzisztort tartalmazott, de a nevében szereplő 4-es számjegyek nem erre utalnak, hanem arra, hogy az adat- és műveleti forgalom 4 bites egységekben történt.

Az évforduló tiszteletére érdemes felidézni 1985/3. számunk alapján az Intel 4004 mikroprocesszor létrejöttének történetét. Egy japán zsebszámológépgyár, a Bizcomp Corp. azzal kereste meg 1969-ben az Intelt, hogy az készítsen számára néhány nagy integráltságú céláramkört a berendezéseihez. A tervezés már akkor sem került kevésbe, így az első mérlegelésnél az Intel el akarta vetni az üzleti ajánlatot. A későbbi felülvizsgálat során azonban az Intel egyik fiatal mérnökének, Marcian E. Hoffnak mentőötlete támadt. Azt javasolta, hogy készítsenek egy általános célú processzort, majd azt programozzák be úgy, hogy képes legyen támogatni az igényelt számológép-funkciókat. Így született meg az Intel 4004 típusú mikroprocesszor.

Számítógéppel a paradicsomháborúban

Tavasszal hatalmas vitákat váltott ki az olcsó, 99 forintos, importból származó primőr paradicsom megjelenése a 400 forintos hazai terméssű mellett. Sokan állították, hogy a 160—200 forintos önköltség miatt ez az akció elveszi a hazai termelői kedvet. A Sárszentmihályi Állami Gazdaság inotai üvegházában másképpen álltak hozzá a kérdéshez: a piacképesség érdekében radikálisan mérsékelni akarták az önköltséget, és ehhez a számítógépet hívták segítségül. A gép feladata egy költségtakarékos, de egyszersmind rendkívül nagy technológiai pontosságot követelő új eljárás vezénylése.

A primőr paradicsom termesztésénél kőzetgyapotos, vízkultúrás módszert alkalmaznak, amellyel várhatóan 30 százalékkal csökkenthetik a hőengergia-felhasználást és 40 százalékkal növelhetik a termés mennyiségét. Az eljárásnál a termőföldet szabvány méretű edényekbe adagolt kőzetgyapot, tőzeg és perlit helyettesíti, a növények táplálékát pedig a számítógép által adagolt tápoldat biztosítja. A módszernek több előnye van: a megfelelő időben adagolt tápoldat a szokásosnál intenzívebb, gyorsabb növekedésre serkenti a növényeket, s mivel a magasabb hőmérsékletet csak a gyökérzetnél kell biztosítani, ez energiatakarékosságot is lehetővé tesz.

A tavasszal végzett főpróba során a módszer beváltotta a hozzá fűzött reményeket: a magasba nyúló indákon 7-8 „emeleten" is érlelődtek a bogyók, s nem ritkák a 11-12 paradicsomból álló „fürtök" sem.

ÖN TUDJA,

HOGY AZ **Szki** STABIL PARTNER!

Szki
Sci-L
Scitel

SZÁMÍTÁSTECHNIKAI KUTATÓ INTÉZET ÉS INNOVÁCIÓS KÖZPONT
SZÁMÍTÁSTECHNIKAI INFORMATIKAI FEJLESZTŐ LEÁNYVÁLLALAT
SZÁMÍTÁSTECHNIKAI FEJLESZTŐ LEASING LEÁNYVÁLLALAT

1251 Budapest, Pf. 19. Telefon: 360-180

MI TUDJUK,

HOGY A JÖVŐBEN IS SZÁMÍT RÁNK!

A SZÁMÍTÁSTECHNIKA HATÉKONY ALKALMAZÁSÁHOZ
A FEJLESZTŐ ÉS A FELHASZNÁLÓ SZOROS,
FOLYAMATOS KAPCSOLATA NÉLKÜLÖZHETETLEN,
EZÉRT

MŰKÖDJÜNK EGYÜTT

A JÖVŐ ÉRDEKÉBEN!

NAGY TELJESÍTMÉNYŰ IBM PC/XT kompatíbilis KONFIGURÁCIÓK és rendszerbővítő elemek

PROPER PROFESSZIONÁLIS SZEMÉLYI SZÁMÍTÓGÉPCSALÁD

További felvilágosítás: RENDSZERÉRTÉKESÍTŐ IRODA Tel.: 153-204

NAGY TELJESÍTMÉNY