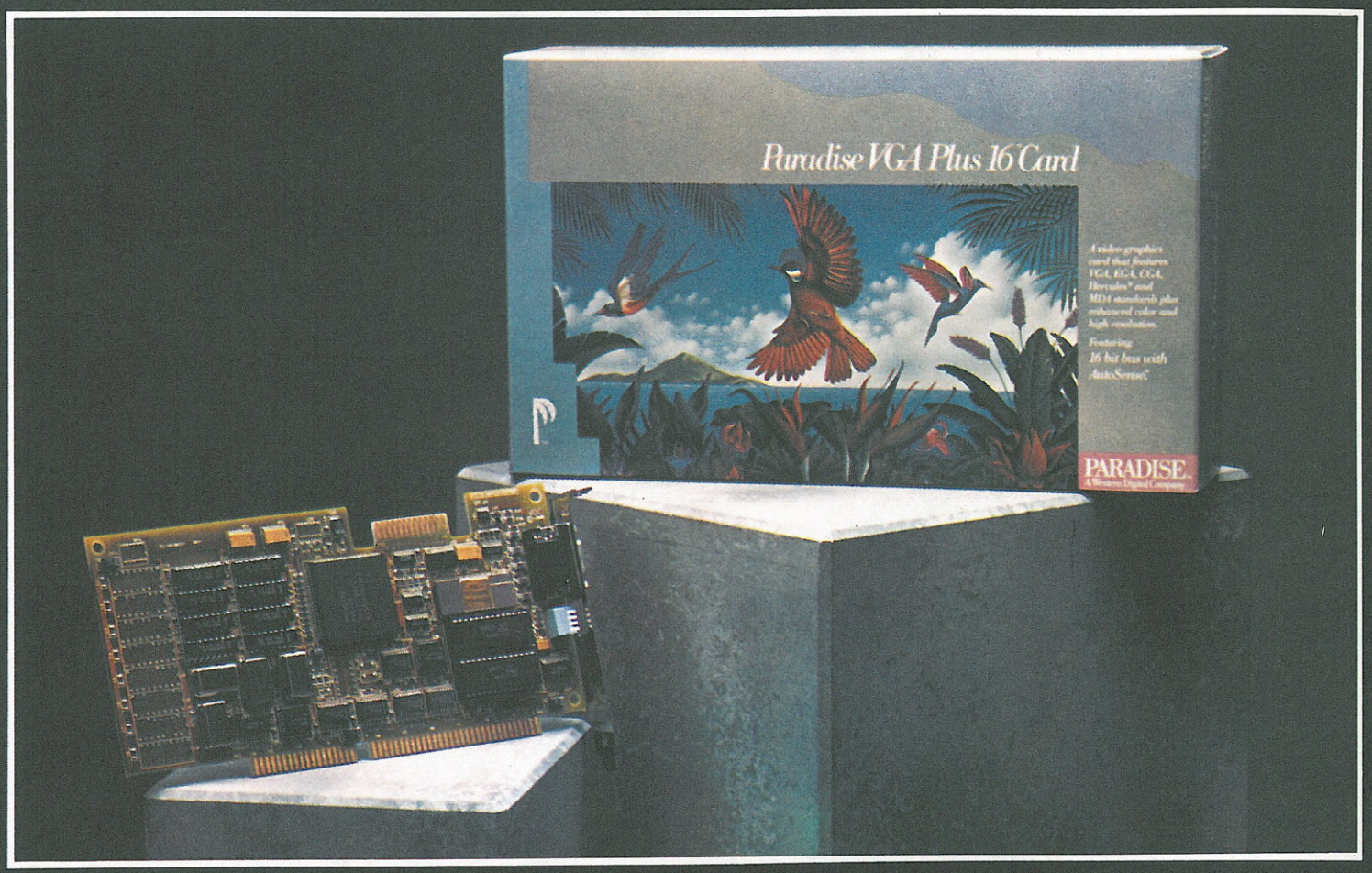
μ mikro számítógép magazin

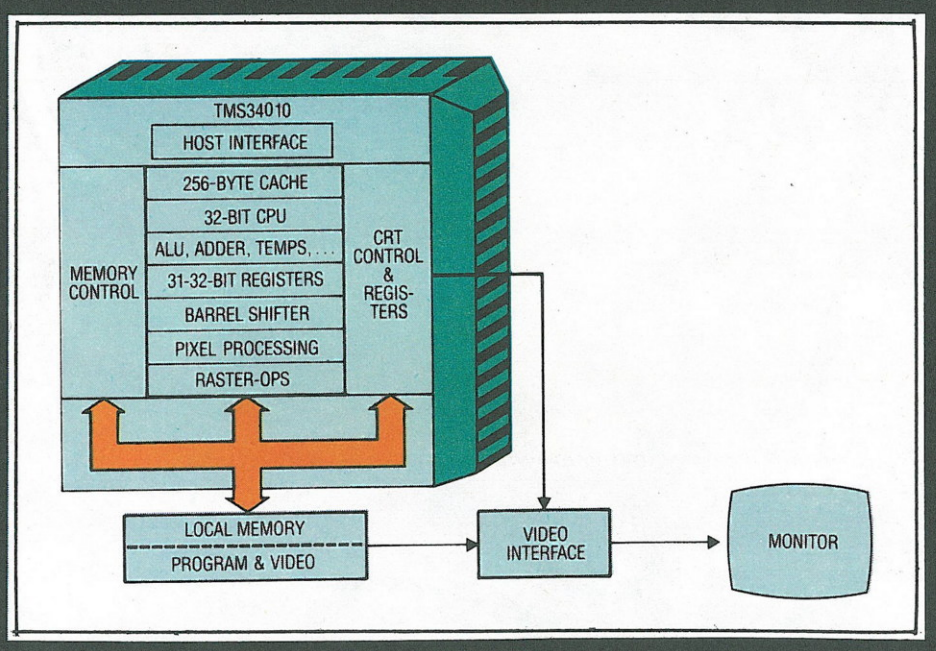
Ára: 30 Ft

TARTSUK A TEMPÓT!

1989/1



VGA Plus 16 kártya



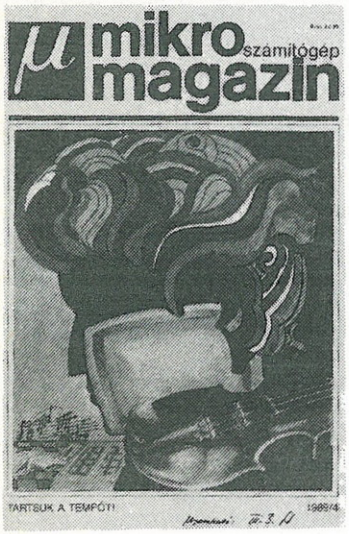
TMS34010
processzoros rendszer tömbvázlata

μ mikro számítógép magazin

7. ÉVFOLYAM
1989/4. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

Címképünk:
Velekey József Lajos munkája



Felelős szerkesztő:
Könyves Tóth Pál

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

Levélcím:
1371 Budapest
Pf. 433

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

Levélcím:
1368 Budapest 5. Pf. 240

Telefon: 329-349

Felelős kiadó:
Tóth Istvánné ügyvezető főtitkárhelyettes

Terjeszti a Magyar Posta Előfizethető a hírlapkézbesítő hivataloknál és a Posta Hírlap-előfizetési és Lapellátási Irodáján (1900 Budapest XIII., Lehel u. 10/A) vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta.
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra, 1389 Budapest, Pf. 149 és a Magyar Média 1932 Budapest, Pf. 279 88—1552

Szikra Lapnyomda
Budapest (89—0316)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

TARTALOM

„Minden ember életében van olyan időszak, amikor a balszerencséből valami jó fakad. Amikor a dolgok olyan baljós fordulatot vesznek, hogy az ember nem tehet mást: üstökön ragadja a sorsot és jól megrázza."
(Lee Jacocca: Egy menedzser élete)

Levél a kórházból II.

Úgy látszik, hogy a kórházi levél műfaj most már évente előjön az életemben, pedig nagyon nem szeretem a publicisztikának ezt a formáját. Különösen akkor nem, ha mindez a saját ostobaságom következménye: egy rizikómentes előzés — gondoltam én —, frontális találkozás egy teherautóval — határozott a sors. Szerencsére a mellettem ülő fiatal kollégámmal együtt megúsztuk. Hál' istennek ő kisebb sérüléssel, én egy kicsit bonyolultabb módon, de azért remélem, hogy elfogadható idő alatt ismét mozgékony leszek.

Eddig persze mindez magánügy, de közben a Mikroszámítógép Magazin várja a szerkesztőségi cikket, és most — sajnos — van elég időm töprengeni a számítástechnika ügyes-bajos dolgain, a kényszerű pihenőm alatt.

A gondolkodásomat segítette, de a hangulatomat nem nagyon emelte, hogy néhány napig mozdulatlanul kellett feküdnöm, mert a lábamra gipszkolonc került. A második napon pedig már úgy éreztem, hogy a hátam kilyukad a sok fekvéstől, és akkor még a rendszeresen visszatérő szurkálások örömeit nem is említettem.

Az sem tesz túlságosan vidámmá, hogy éppen ma kezdődik az NJSZT Országos Elnökség ülése, amelyet másfélszáz kilométerről próbálok követni, ami persze lehetetlen, de legalább gondolatban ott vagyok, és Balatonföldváron kínomban egy nyílt levelet írtam az elnökség tagjainak.

Azt hiszem ugyanis, hogy nagyon fontos kérdésekről van szó. Különösen ma, amikor egyre inkább anyagiasodó, profitcentrikus világban élünk, és végül is azt kell az NJSZT vezetőségének elhatároznia, hogy a társaságot egy nyereségorientált vállalkozássá alakítja át, ahol a prioritásokat a jövedelmezőség határozza meg, vagy pedig megpróbálja a társadalmi célokat követni, és támogatások útján a megfelelő anyagiakat megszerezni.

Én ahhoz a kisebbséghez tartozom, amely az utóbbi elképzelésekkel ért egyet, annak ellenére, hogy tudom, az állam a támogatásokat az élet szinte valamennyi területén mérsékli vagy teljes egészében megszünteti.

Voltaképpen arról is szó lesz, hogy ma hogyan tudják a tudományos egyesületek, például a Neumann János Számítógéptudományi Társaság is, a működésükhöz szükséges anyagiakat előteremteni akkor, amikor a bevételi forrásaik, azaz az egyéni tagdíjak és a rendezvények alaposan megadóztatott bevételei egyre szűkülnek. Hiszen a pénzügyi kormányzat a tudományos egyesületeket sem kezeli másként, mint a termelővállalatokat.

Nagyon nem szeretném, ha ez az írás nyafogósra sikerülne, de a tudományos egyesületek jövőjét, anyagi lehetőségeit egyre nehezebbnek látom. Sőt, azt is megkockáztatom, hogy a szakvállalatok anyagi és erkölcsi segítsége nélkül bizonyos feladatokat egyszerűen lehetetlen lesz teljesíteni.

A közelmúltban két kísérletem is volt — az egyik az írás pillanatában még tart —, hogy „magasztos" egyesületi céljaink teljesítéséhez társadalmi támogatást szerezzek. Az egyik éppen a Mikroszámítógép Magazinnak a jelenlegi formában való továbbéltetése, a másik pedig a nyíregyházi állandó számítástechnikatörténeti kiállítás létrehozása volt.

Azt minden olvasónk tudja, hogy a Magazin folyamatosan anyagi nehézségekkel küzd; tudniillik a jelenlegi 30 forintos példányonkénti ár a nyomda-, papír- és szerkesztőségi költségeket nem fedezi. Valószínűleg nem halogathatjuk tovább a lap árának a mintegy 50 százalékos emelését, mert az egyre fokozódó mértékű inflációt sajnos követnünk kell, de még így sem tudjuk a lapeladásból a költségeinket teljes mértékben fedezni. A Mikroszámítógép Magazin az amatőrök, az ifjúság lapja, még akkor is, ha az előfizetőink egy igen kis hányada az érdeklődő felnőttek közül kerül ki. Végeredményben tehát — nekem ez a véleményem — a lap árát nem az előállítási költséghez, hanem az olvasók jövedelmi viszonyaihoz kell igazítani, és a veszteségeket más forrásból kell fedeznünk. A lap pénzügyi egyensúlyát könnyen helyreállíthatnánk például, ha a Neumann társaság jogi tagvállalatai vállalnák, hogy évente egyszer-kétszer hirdetnek nálunk. Ez az árbevétel a veszteségünket teljes mértékében fedezné.

Ugyanígy — a társaság költségvetésében — egyelőre nincs fedezet a nyíregyházi hazai számítástechnika-történeti kiállítás létrehozására sem. Ha valaki nem tudná: 1959. január 29-én jelent meg az újságokban, hogy elkészült az első magyar elektronikus számítógép, vagyis 1989. január 28-án volt 30 éves a magyar számítástechnika. Ráadásul Tóth Endre, az Országos Műszaki Múzeum főmuzeológusa a korabeli dokumentumokból kiderítette, hogy 1964. január 27-én alakult meg a MTESZ keretében az IKOSZ, az NJSZT elődje, az Információfeldolgozási, Kibernetikai és Operációkutatási Szakosztály, tehát ez év januárjában lett 25 esztendős a Neumann János Számítógéptudományi Társaság is.

Ezért javasoltam — ha már az évfordulók ilyen jól összejöttek, és ráadásul sikerült megőriznünk egy csomó öreg számítógépet is —, tiszteljük meg a szakmát egy állandó kiállítás létrehozásával. A Szabolcs-Szatmár megyei vezetők ajánlkoztak társul a vállalkozáshoz, és vállalták egy ilyen kiállítás költségeinek legnagyobb részét is. De azért legyünk őszinték, ez az ügy nem csak Szabolcs—Szatmár megye ügye. Amikor a körlevelet a megyei szervezeteknek elküldtem, a megyéknek egy kis része sajnálkozott, hogy nem tudja a feltételeket biztosítani, a nagyobbik része pedig még csak nem is válaszolt. E tények tudatában a szabolcsiak felajánlását különösen meg kell becsülnünk.

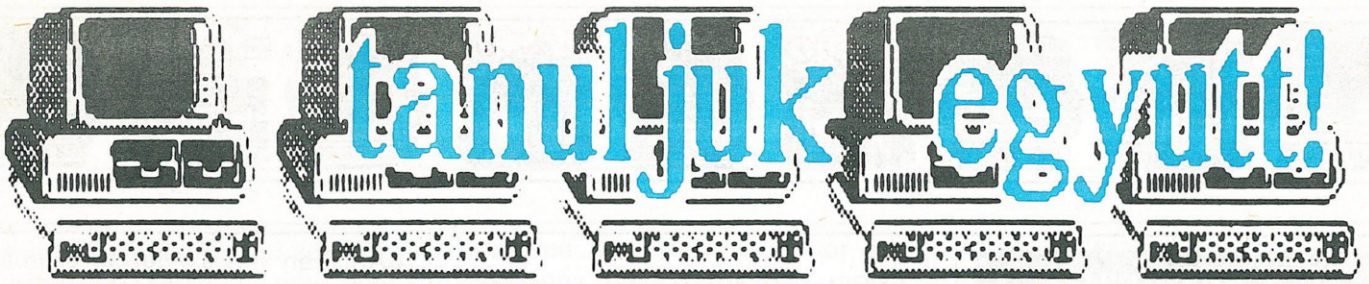
A kiállítás költségeit nyilván nem lehet csak a megyére hárítani. Amit még össze kellene szednünk, az hozzávetőlegesen 3 millió forint, hogy a gépeket a helyszínre szállíthassuk, a tablókat elkészítsük, egyszóval a kiállítást méltóvá tegyük az évfordulóhoz. Ezt a 3 millió forintot ismét csak az NJSZT kétszáz feletti jogi tagközösségétől kértem, örök optimistaként azt gondolva, hogy egy ilyen akció nem csak a szíveket, de a bukszákat is megnyitja. A körlevélre összesen öt cég válaszolt. Az MTA és a Comporgan jelentős összeget, a másik három néhány ezer forintot ajánlott, illetve bíztató szavakat küldött támogatásként.

Én persze érthetetlennek tartom a többi kétszáz-egynéhány cég magatartását. Az ugyanis elképzelhetetlen számomra, hogy néhány tízezer forint annyira a tönk szélére taszítana egy-két országos nagyvállalatot, hogy még csak nem is válaszolnak. Vagy — ez a másik lehetőség — annyira érdektelen ma a vállalati szféra az ilyen társadalmi ügyek iránt, hogy nem is szabad támogatásért hozzájuk fordulni.

Szóval fekszem az ágyon itt Dunaújvárosban, és élvezem a korábbi kormányzat döntéseinek eredményét a hazai telefonhelyzettel kapcsolatban. A kórházból ugyanis alig negyven perc alatt tudom például a Vasművet megkapni, Budapestet viszont kora reggel vagy nagyon késő este, akkor is csak igen nehezen. Tehát úgy el vagyok vágva a világtól, mintha egy korallzátonyon élnék, közben az idő megy, és gyakorlatilag nem történik semmi.

Most azon gondolkozom, hogy nem lettem-e pesszimista egy autóbaleset következtében. Vagy ma ez a realitás?

Kovács Győző



A PASCAL REJTELMEI

6. Negyedik programunk

Talán nem merészség azt feltételezni, hogy sorozatunk előző részeinek elsajátításával olvasóink már kellő nyelvismerettel vértezhették fel magukat. Éppen ezért a továbbiakban komolyabb feladatokra vállalkozhatunk, ugyanakkor mélyebbre hatolhatunk a Pascal titkosabb régióiba is.

Ezúttal tíz integer típusú adat (szám) növekvő sorrendbe rendezése és a képernyőre kivitele lesz a feladatunk. A rendezendő adatokat egy tömbváltozóban helyezzük el. Egy tömbváltozó deklarációja a következő formájú:

var tomb:array [min..max] of tipus

Ha a tömb elemei — típusai — standard típusok, külön típusdeklarációra nincs szükség. Az **array** kulcsszót követő szögletes zárójelben a tömb első és utolsó elemének indexét kell megadni. Az index értéke csak **integer** szám lehet.

Részletesen foglalkozunk a globális és lokális változókkal és a változók érvényességi (hatás-) körével. Először fogjuk alkalmazni a feltételes vezérlésátadás legegyszerűbb eszközét, az **if..then** utasítást.

6.1 Globális és lokális változók

Eddigi programjainkban a felhasznált változókat a program elején, a programfejet közvetlenül követő deklarációs részben definiáltuk. E változók a program bármely részében — az eljárásokban is — alkalmazhatók voltak, értékükhöz hozzáférhettünk. Az ilyen változók érvényességi köre tehát az egész program, elnevezésük globális változó.

A Pascal lehetőséget ad arra is, hogy változókat deklaráljunk a programhierarchiában tetszőleges helyen lévő eljárásokban is. E változók hatásköre csak az adott vagy az abból hívott — a hierarchiában alacsonyabb szinten lévő — eljárás. Ezek a változók a program magasabb szintjein vagy a programfa más ágain lévő helyekről, például eljárásokból nem érhetők el; ha rájuk hivatkozunk, akkor **unknown identifier** (ismeretlen azonosító) hibajelzést kapunk. Az ilyen módon deklarált változók neve lokális változó.

A globális és lokális változó meghatározást természetesen rugalmasan és értelemszerűen kell alkalmazni: például egy eljárásban deklarált változó az adott eljárásból hívott minden eljárás számára globálisnak tekintendő.

Gyakran előfordul, hogy egy változó érvényességi körén belül, történetesen egy hívott eljárásban, a változót más tartalommal ruházzuk fel, egy új deklarációval újradefiniáljuk. Ilyenkor az új deklaráció hatáskörén belül a változó eredeti értékét elveszti, majd abból kilépve, azt visszanyeri.

Az elmondottakat illusztrálja a *15. ábrán* látható program, a futás eredményét pedig a *16. ábra.*

Az **i** az **ujradeklaral** program globális változója. Az **ujra** eljárásban mint lokális változót újradefiniáltuk és **i**-nek 1 értéket adtunk.

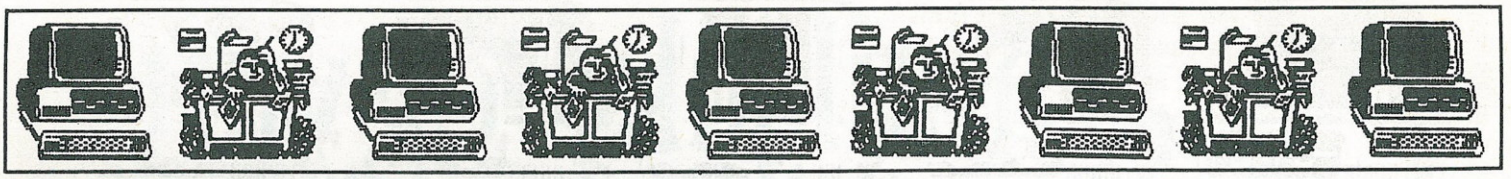
A főprogram kezdetén az **i** értéke 2 lesz. Ezt kivisszük a képernyőre, majd az **ujra** eljárást hívjuk. A kivitelkor a képernyőn most 1 jelenik meg. A program következő **writeln**-je már az újradeklarálás hatáskörén kívül van, tehát az **i** visszakapja eredeti értékét (2), és kiírásra most ez kerül.

A figyelmes olvasó ezek után joggal vetheti fel: a gyakorlati problémák megoldása során csak ritkán alkalmazunk „elszigetelt" lokális változókkal dolgozó eljárásokat, majd minden esetben szükség van a hívó és a hívott programrészek közötti adatcserére is! A Pascal erre is igen egyszerű lehetőséget nyújt; ezzel a következő részben, Az eljárások paraméterei (7.1) cím alatt foglalkozunk.

6.2 Az if..then utasítás

Az **if..then** a Pascal legegyszerűbb feltételes utasítása. Az **if** után általában tetszőleges kifejezést (vagy **boolean** típusú változót) írhatunk. A megkötés mindössze annyi, hogy a kifejezés kiértékelése után az eredmény **boolean** típusú legyen. A **then** után csak egyetlen utasítás állhat, ez azonban a **begin** és az **end** kulcsszavakkal képzett összetett utasítás is lehet. A megkötés végső soron tehát nem korlátoz, a **then** után tetszőlegesen hosszú programrész írható.

Az **if..then** utasítások gyakorlatilag tetszőleges mélységben egymásba skatulyázhatók, azaz a **then** után újabb **if** állhat stb.



6.3 Program tíz integer típusú adat sorba rendezésére

Végezetül a *17. ábrán* — minden különösebb magyarázat nélkül — bemutatjuk a feladatot megoldó program listáját. A tíz adatot egy tízelemű tömb elemeiként definiáltuk és kezeltük. (A program a közismert buborék algoritmus alapján végzi el a rendezést.) A futtatáskor keletkező screen a *18. ábrán* látható.

7. Teljes erőbedobással az eljárásokért

Ahogyan azt a sorozat előző részében már említettük, egyre mélyebbre hatolunk a Pascal rejtelmeibe. Állításunk igazságát mi sem bizonyítja jobban mint az, amin az olvasónak most „át kell rágnia magát". Egyebek között ugyanis a Pascal egyik fő erősségével, a programkörnyezettől független eljárásokkal, azok készítésével és alkalmazásával fogunk foglalkozni. Megjegyzendő, hogy a BASIC-en nevelkedettek számára az ismeretanyag valószínűleg nehezebbnek bizonyul majd, mint a tiszta lappal indulóknak. A téma jelentősége miatt a szokásosnál több magyarázatot és egészen egyszerű példákat adunk.

7.1 Az eljárások paraméterei

Kiindulásként tételezzük fel, hogy rendelkezünk egy, már megírt eljárással, amely két — az eljárást hívó programból származó — számot összead, majd az összeget a hívó programnak visszaadja. Ennek az eljárásnak a listáját láthatjuk a *19. ábrán.*

A dologban semmi furcsaság nem lenne, ha a változódeklarációkat nem az eljárásnév utáni zárójelben, hanem a már megszokott helyen találnánk. Az ilyen módon deklarált változók a hívó program

```
program szamsor;
{tömbök használata, globális és lokális változók, 
if then}
var i:integer;
    tomb:array [1..10] of integer;
{globális változók: i,tomb}
procedure bevitel;
begin
   clrscr;writeln('Negyedik gyakorló programunk');
   writeln('Kérek 10 egész számot');
   for i:=1 to 10 do
      readln(tomb[i])
end;
procedure rendez;
var j,temp:integer;
{lokális változók: j,temp}
begin
   for j:=1 to 9 do
      for i:=1 to 9 do
         if tomb[i]>tomb[i+1] then
            begin
               temp:=tomb[i];
               tomb[i]:=tomb[i+1];
               tomb[i+1]:=temp
            end
end;
procedure kivitel;
begin
   gotoxy(30,2);writeln('Az adatok rendezve');
   for i:=1 to 10 do
      begin
      gotoxy(30,2+i);
      writeln(tomb[i])
   end
end;
begin {főprogram kezdet}
   bevitel;
   rendez;
   kivitel
end.
```

17. ábra

```
program ujradeklaral;
var i:integer;  {i első deklarációja}
procedure ujra;
var i:integer;  {i ujradeklarálása}
begin
   i:=i;
   writeln('i ujradefiniált=',i)
end;
begin  {főprogram}
   clrscr;
   i:=2;
   writeln('i eredeti=',i);
   ujra;  {i új értéket kap}
   writeln('i eredeti=',i)  {i értéke ismét a régi}
end.
```

15. ábra

16. ábra

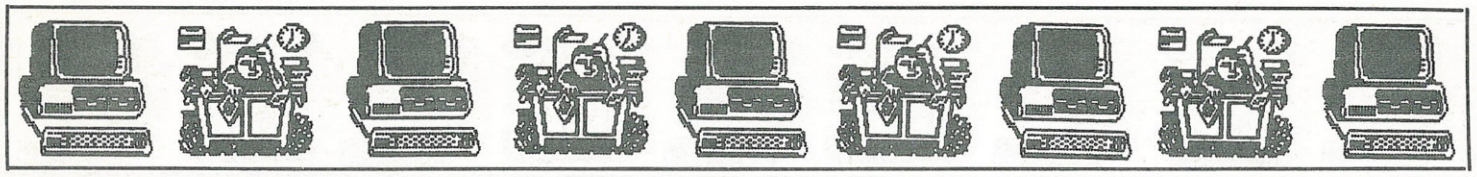
```
i eredeti=2
i ujradefiniált=1
i eredeti=2

>
```

18. ábra

```
Negyedik gyakorló programunk
Kérek 10 egész számot          Az adatok rendezve
87                             -12
11                             1
-12                            4
45                             6
22                             11
31                             20
20                             22
4                              31
6                              45
1                              87

>
```



és a hívott eljárás közötti adatcseré szolgálják. Feladatuk más, mint egy program vagy eljárás már ismert módon, az ún. deklarációs részben definiált változóinak. E különbség miatt a Pascal más elnevezést használt megjelölésükre: formális paramétereknek nevezi őket.

Közülük azok, amelyek a kezdő zárójeltől a **var** alapjelig vannak felsorolva, a bemeneti paraméterek, azaz értéküket a hívó programból kapják (példánkban az **a** és a **b**). A **var** alapjel után lévők az eljárás kimeneti paraméterei, ezek az eljárás végrehajtása után a hívó programnak adnak át értéket (példánkban ilyen az **osszeg**).

Már most — az eljárás hívásának bemutatás előtt — megemlítjük, hogy a formális paraméterként definiált változókra ugyanazok a szabályok érvényesek, mint a lokális változókra; mivel hatáskörük csak az adott és az abból hívott eljárásokra terjed ki, a változónevek kiválasztására speciális korlátozás nincs. (Közömbös, hogy az eljárás paramétereit a program más részében változóként vagy paraméterként alkalmaztuk-e vagy sem.) Egyébként éppen ez az a tény, ami az eljárásnak a programkörnyezettől való teljes függetlenségét biztosítja!

A kérdés most már csak az, hogyan kapnak a formális paraméterek értéket? Ennek megoldása sem bonyolult: az eljárás hívásakor az eljárásnév után zárójelben, egymástól vesszővel elválasztva fel kell sorolni az eljárásnak átadni, illetve onnan átvenni kívánt adatokat, azaz a szám- vagy szövegkonstansokat, változókat, kifejezéseket. A felsorolásban szereplő elemek elnevezése: aktuális paraméterek.

A 19. ábrán bemutatott eljárást használó program listája látható a *20. ábrán.*

A program a billentyűzetről beolvas két számot (**op1** és **op2**). Ezután hívja az **osszead** eljárást. A paraméterátadás a következő:

— **op1** értéke átadódik **a**-nak,

— **op2** értéke átadódik **b**-nek,

— az eljárás elvégzi az összeadást, az eredmény az **osszeg** változóba kerül,

— az **osszeg** értéke átadódik **szumma**-nak.

A bemutatott példából két nyilvánvaló szabályra derül fény:

— a formális és aktuális paraméterek száma azonos legyen,

— a formális és aktuális paraméterek típusa azonos vagy legalábbis kompatibilis legyen.

A szabályok magyarázata igen egyszerű. A program lefordításakor a fordító helyet foglal el a tárban a formális paraméterek listája alapján. Ha ezekre a helyekre nem illeszkednek a program futása közben az aktuális paraméterek, ez hibához vezet. A paraméterek és a tárhelyek illeszkedésének feltétele: azonos számú és azonos helyfoglalású (kompatibilis) paraméter.

A típuskompatibilitást a különböző Pascal reprezentációk különféleképpen kezelik. Általános útmutatásként az fogadható el, hogy a különböző típusok akkor kompatibilisek, ha azonos alaptípusra vezethetők vissza. Például várhatólag kompatibilis egy húszkarakteres sztring és egy húszelemű karaktertömb. Mindezért legcélszerűbb, ha a birtokunkban levő fordítót kipróbáljuk a kompatibilisnek vélelmezhető típusokkal. Nekünk egy bombabiztos javaslatunk van: egyelőre ne kompatibilisnek vélt, hanem azonos típusú paraméterekkel dolgozzunk!

Az érdekesség és a tanulság kedvéért **szumma** nevű programunk egy másik változatát is bemutatjuk. Ez a program ugyanazokat a neveket használja a programban és az eljárásban, sőt formális és aktuális paraméterként is. A program listája a *21. ábrán* látható.

A program a sok adat azonos neve mellett is működik. Ennek magyarázatát, indoklását az olvasóra bízzuk! Ha valakinek nehézségei támadnának, javasoljuk, hogy olvassa végig még egyszer a globális és lokális változókról leírtakat (6.1 rész).

N. I.

19. ábra

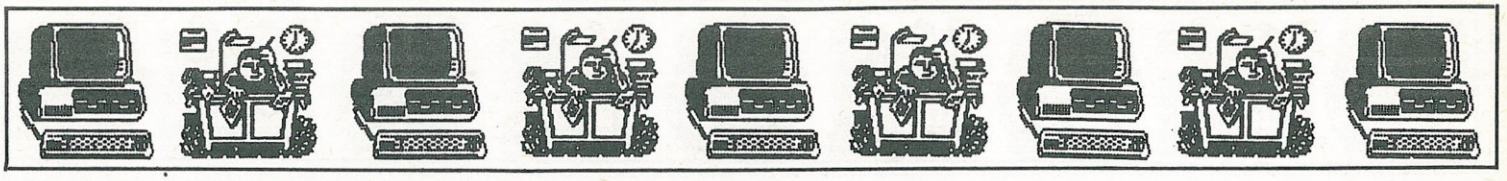
```
procedure osszaead(op1,op2:real;var szumma:real);
begin
   szumma:=op1+op2
end;
```

20. ábra

```
program szumma;
var op1,op2,szumma:real;
procedure osszead(a,b:real;var osszeg:real);
begin
   osszeg:=a+b
end;
begin
   write('1. összeadandó:');readln(op1);
   write('2. összeadandó:');readln(op2);
   osszead(op1,op2,szumma);
   writeln('Az összeg:   ',szumma)
end.
```

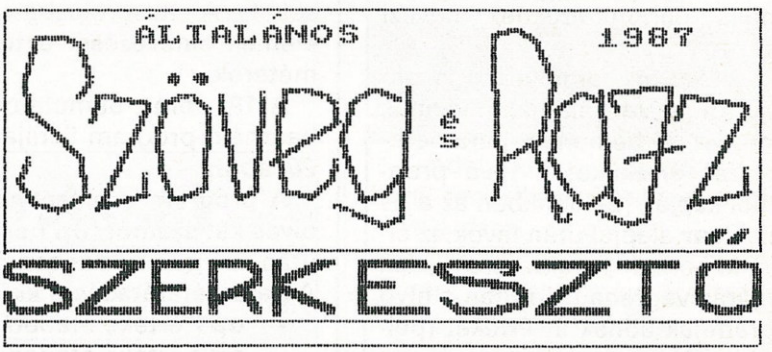
21. ábra

```
program szumma;
var op1,op2,szumma:real;
procedure osszead(op1,op2:real;var szumma:real);
begin
   szumma:=op1+op2
end;
begin
   clrscr;
   write('1. összeadandó:');readln(op1);
   write('2. összeadandó:');readln(op2);
   osszead(op1,op2,szumma);
   writeln('Az összeg:   ',szumma)
end.
```



Commodore Plus/4

Program-ajánlat



Tettre kész mindenes

A professzionális kivitelű program segítséget nyújt levelek, iskolai segédletek (feladatlapok, tesztlapok stb.) és tetszőleges írásos dokumentumok szerkesztésére. Az elkészítendő nyomtatványok sokszínűségét elősegíti az, hogy a program képes valódi grafikus ábrák szerkesztésére és a szövegbe illesztésére, valamint lehetőséget ad magyar ékezetes, cirill, görög és „indexbe írt" karakterkészletek használatára.

Az irat elkészítéséhez egy százsoros, soronként 50 karakteres szerkesztőterület áll rendelkezésre. Ebből a képernyőn huszonegy sor és soronként 35 karakter látható. A képernyő többi részét a fent és baloldalt állandóan jelen lévő ikonmenü tölti ki. A felső menü általános célú funkciók kiválasztására szolgál, az oldalsó a grafika szerkesztését segíti. A kurzort vezérlő billentyűkkel természetesen a szöveg bármelyik része hozzáférhető. A szerkesztőterület akármelyik részén tetszés szerint helyezhetünk el ábrákat, szöveget.

A számítógép képernyőszerkesztőjéhez hasonlóan használhatók az INST/DEL és a RETURN billentyűk. A HOME a képernyő, a CLEAR az irat kezdőpozíciójára állítja a kurzort. A billentyűk önmagukban kis, SHIFT-tel együtt nagy, a C=-ral együtt pedig ékezetes betűket jelenítenek meg. További szerkesztési funkciókat az ikonmenük segítségével végezhetünk el.

A megfelelő ikont — piktogramot — a kurzorral választhatjuk ki, amely a menüben mint nyíl jelenik meg. A menüelem aktivizálása a SHIFT billentyű lenyomásával érhető el.

A szöveg írása közben bármikor válthatunk karakterkészletet. Az F1, F2, F3 és a HELP billentyűk rendre a magyar, a cirill, a görög és az index betűkészlet váltóiként alkalmazhatók.

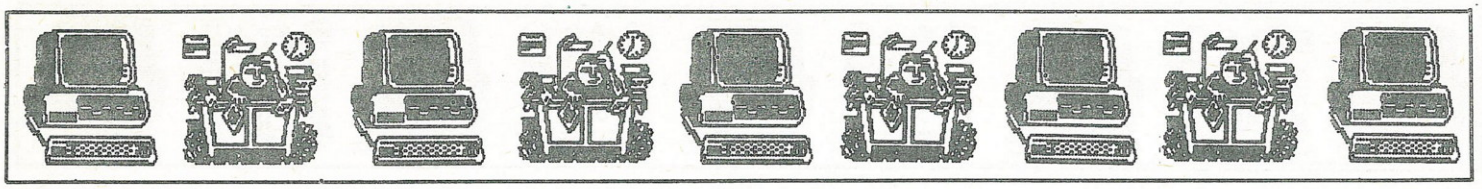
Gyakran lehet szükség arra, hogy nyomtatáskor egy szövegrészt a papír közepére vagy jobb szélére helyezzünk. Ezt valósíthatjuk meg a CONTROL billentyű használatával. Egyszeri lenyomás hatására a képernyő egy sora a papíron középre, ismételt lenyomáskor a papír jobb oldalára kerül.

Billentyűzet megfeleltetési táblázat a cirill betűk használatához:

а-a	з-z	п-p	ч-@
б-b	и-i	р-r	ш-s
в-v	й-j	с-q	щ-w
г-g	к-k	т-t	ы-ü
д-d	л-l	у-u	ь-y
е-e	м-m	ф-f	э-d
ё-ó	н-n	х-h	ю-ú
ж-x	о-o	ц-c	я-á

Billentyűzet megfeleltetési táblázat a görög betűk használatához:

α-a	ι-i	ρ-r
β-b	κ-k	σ-s
γ-c	λ-l	τ-t
δ-d	μ-m	υ-y
ε-e	ν-n	φ-j
ζ-z	ξ-f	χ-x
η-h	ο-o	ψ-w
ϑ-g	π-p	ω-u



Ha a kurzor egy grafikus kép területéhez ér, akkor a képben az aktuális ceruzaheggyel ellátott nyíl jelenik meg. Ez mindaddig „él", amíg a kép területét el nem hagyjuk. A grafikus menübe a grafikus területből annak bal oldali választófalán át juthatunk be. A funkciót az általános menüvel kapcsolatban már leírtakhoz hasonlóan választhatjuk ki.

A programot írták: Petrov Ferenc és Petrov Péter

Forgalmazza a Bács-Kiskun Megyei Pedagógiai Intézet

6001 Kecskemét, Katona József tér 8. Pf. 180.

Egy kép a program képernyőjéről:

Tempótartás kopogással

A Magazin tavalyi 6., 7. és 10. számaiban már jelentek meg zenei képességeket vizsgáló rövid programok C64-re. Ennek a gépnek a hanggenerátora és belső órája lehetővé teszi, hogy olyan zenei képességekről szerezzünk pontos adatokat, amiket korábban a legjobb fülű szakemberek is csak szubjektíven értékelhettek. Természetesen nem kizárt, hogy a zenét nem tanulók is alávessék magukat a megmérettetésnek. Az ismertetendő téma az életben sok helyütt felbukkan. Hiánya vagy megléte befolyásolhatja hétköznapjainkat. A téma nem más, mint a tempótartás.

```
5 DIM X(16),Y(16):READ H:FOR I=1 TO H
10 READ Y(I):O=O+Y(I):NEXT
15 O=O-Y(I-1)
20 DATA 7,6,2,4,4,4,4,8
25 S=54272:FOR I=0 TO 24:POKE S+I,0:NEXT
30 POKE 53281,1:POKE S+24,15:POKE S+5,24
35 M=INT(RND(1)*7)+8
40 PRINT CHR$(147):FOR I=1 TO H:TI$="000000"
45 POKE S+4,129:POKE S+1,50
50 IF TI<Y(I)*M THEN 50
55 POKE S+4,128:NEXT
60 PRINT "MI LEGYEN A FOLYTATAS?":PRINT
65 PRINT "1. UJBOL MEGHALLGATOD A RITMUST"
70 PRINT "2. VISSZAKOPOGOD":POKE 198,0
75 GET R$:IF R$="" THEN 75
80 IF R$="1" THEN 40
85 PRINT CHR$(147)"ELKEZDHETED"
90 POKE 198,0:FOR I=1 TO H:TI$="000000"
95 GET R$:IF R$="" THEN 95
100 IF I=1 THEN PRINT CHR$(147):NEXT I
105 X(I-1)=TI:NEXT
110 FOR I=1 TO H:SU=SU+X(I):NEXT:K=SU/(M*O)
115 IF K<=1 THEN SZ$="GYORSABB":GOTO 125
120 SZ$="LASSABB"
125 PRINT " A TE KOPOGASOD";
130 PRINT ABS(INT(10000*(1-K)+.5)/100);
135 PRINT "%-KAL VOLT "
140 PRINT SZ$" AZ EREDETINEL":PRINT
145 PRINT " AZ EGYES UTESEK ELTERESEI"
150 PRINT "(A TE TEMPOD SZERINT) %-ban:"
155 FOR I=1 TO H-1:A=100*X(I)/(Y(I)*M*K)
160 B=INT(10*A+.5)/10-100
165 C=INT(100*B+.5)/100:PRINTTAB(13)C
170 NEXT:END
```

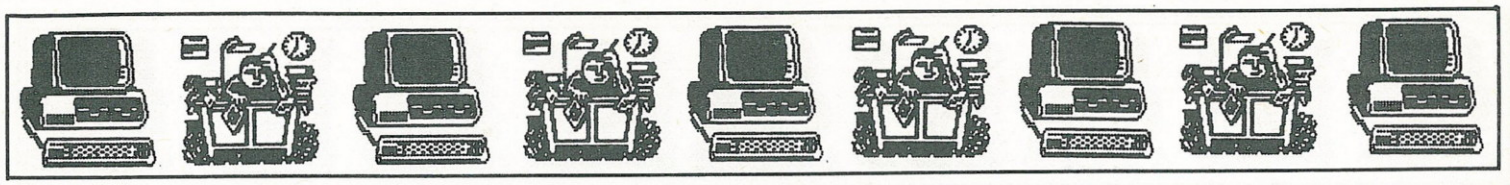
A program egy közismert ritmust, a magyar Himnusz kezdősorát szólaltatja meg (40—55-ös sorok). A megszólalás tempóját a program határozza meg (35-ös sor). Ezt kell visszakopogni valamelyik billentyűn (90—110-es sorok). Ezután a program úgy méri a tempó tartását, hogy összehasonlítja a hangoztatás és a kopogás idejét (125—140-es sorok). Ez azonban nem egészen korrekt, mivel az egyes hibák szerencsés esetben kompenzálhatják is egymást. Ezért az egyes ütések hosszát is összehasonlítja az ideálissal (145—170-es sorok), úgy, mintha az etalon is a visszakopogás átlagos tempójában szólalt volna meg (155-ös sor). Zenét tanulóknak hasznos lenne az is, ha az egyes ritmusképletek pontosságát önmagukban is vizsgálnánk. Ez azonban lényegesen bonyolítaná a programot. E két vizsgálat akkor nyújt igazán reális képet, ha a programot többször futtatjuk egymás után, s így többféle tempóban halljuk a ritmust.

A kiértékeléshez az alábbi megközelítő adatokat lehet figyelembe venni. Ha a kopogás és hangoztatás időtartama közötti eltérés 0-2 százalék között van, az kiváló eredmény; 2-5 százalék közötti érték jó, az 5-10 százalék már rossz eredménynek számít. 10 százalék fölött bizonyára más tempót akart a felhasználó megvalósítani, ezért ezt nem lehet még rosszabbnak minősíteni.

Ha az egyes ütközések eltérése 0-6 százalék között van, az kiváló; 6-15 százalék között jó, 15-30 százalék között rossz eredménynek minősíthető. 30 százalék fölött valószínűleg téves volt a ritmus értelmezése.

A program lehetőséget nyújt más ritmusok bevitelére is. Ilyenkor a 20-as sorban elsőként azt kell közölni, hogy hány hangból áll a ritmus; nálunk hétből. Utána a ritmust relatív értékekkel kell megadni. Célszerű a negyed értékű hanghoz a 4-es számot rendelni. Ekkor a nyolcad a 2-nek, a tizenhatod az 1-nek stb. felel meg. Ha a negyed érték a 6-os számot kapja, akkor triola és nyolcad egymás mellett is megszólaltatható.

Kalmár Gyula



„KEYBOARD" Hangos billentyűzet

Commodore Plus/4-hez

```
Ø 5 F 5   L D A  ↓ $ Ø 2        Uj interrupt vektor
Ø 5 F 7   S T A    $ Ø 3 1 2
Ø 5 F A   L D A  ↓ $ Ø 6
Ø 5 F C   S T A    $ Ø 3 1 3
Ø 5 F F   J M P    $ 8 7 Ø 3    Visszaugrás BASIC- be
Ø 6 Ø 2   J S R    $ C F B F    Interrupt folytatási imitáció
Ø 6 Ø 5   J S R    $ C E C D
Ø 6 Ø 8   L D A    $ F B
Ø 6 Ø A   P H A
Ø 6 Ø B   L D A  ↓ $ Ø Ø
Ø 6 Ø D   S T A    $ F B
Ø 6 Ø F   P H P
Ø 6 1 Ø   S T A    $ Ø 5 4 3    Billentyüzet olvasás
Ø 6 1 3   L D Y  ↓ $ 4 Ø
Ø 6 1 5   S T Y    $ C 6
Ø 6 1 7   J S R    $ D B 7 Ø
Ø 6 1 A   T A X
Ø 6 1 B   C P X  ↓ $ F F        Van megnyomott billentyü ?
Ø 6 1 D   B E Q    $ Ø 6 3 4    Nincs
Ø 6 1 F   L D A  ↓ $ 1 1        Hang beállítás, bekapcsolás
Ø 6 2 1   L D X  ↓ $ Ø 3
Ø 6 2 3   L D Y  ↓ $ 2 8
Ø 6 2 5   S T A    $ F F Ø F
Ø 6 2 8   S T X    $ F F 1 Ø
Ø 6 2 B   S T Y    $ F F 1 1
Ø 6 2 E   J S R    $ D B 2 5    Billentyüzet olvasás befejezése
Ø 6 3 1   J M P    $ C E 5 4    Interrupt befejezés
Ø 6 3 4   L D A  ↓ $ Ø Ø        Hang kikapcsolás
Ø 6 3 6   S T A    $ F F 1 1
Ø 6 3 9   J S R    $ D C Ø 1    Kiugrás billentyüzet olvasásból
Ø 6 3 C   J M P    $ C E 5 4    Interrupt befejezés
```

Az alábbi program MONITOR üzemmódban történő begépelés vagy betöltés után G05F5 paranccsal indítható. A rendszerváltozók által nem használt területen helyezkedik el, így az egyéb programokat nem zavarja. Futtatása esetén minden egyes billentyű lenyomásakor hangjelzést kapunk mindaddig, míg a gépet RESET-tel újra nem indítjuk vagy ki nem kapcsoljuk. A hang frekvenciáját a 061F sor értékének átírásával tudjuk megváltoztatni.

A program Commodore Plus/4-es és C16-os gépeken futtatható.

Lovas Sándor

„SEARCH. S" Hangos keresőprogram

C Plus/4 magnóhoz

Ez a program az adatmagnóról beérkező jeleket teszi hallhatóvá a monitor vagy a tv-készülék hangszóróján keresztül. Indítása MONITOR üzemmódban, G05F5 paranccsal történhet. A PLAY billentyű lenyomásával a kazettán tárolt jelek hallhatóvá lesznek, miáltal a programok keresése, ellenőrzése egyszerűbbé válik.

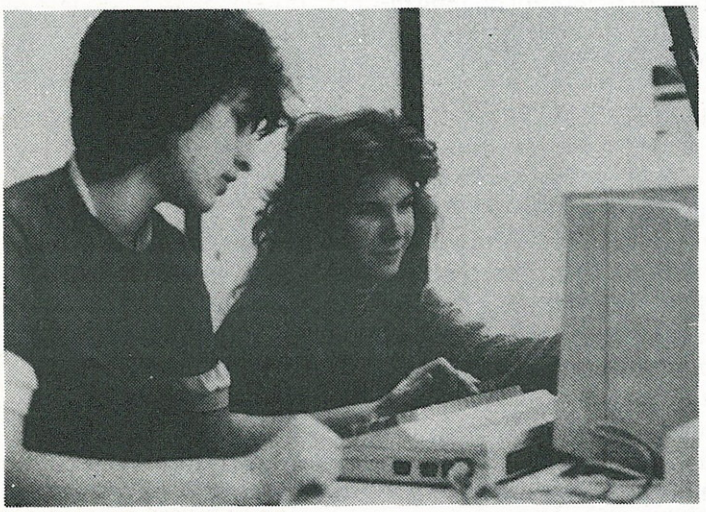
Ha BASIC-ben Sound utasítást akarunk használni, RESET-tel állítsuk le a gépi kódú programjainkat!!!

L. S.

```
Ø 5 F 5   L D A  ↓ $ Ø 2        Uj interrupt vektor beállitása
Ø 5 F 7   S T A    $ Ø 3 1 2
Ø 5 F A   L D A  ↓ $ Ø 6
Ø 5 F C   S T A    $ Ø 3 1 3
Ø 5 F F   J M P    $ 8 7 Ø 3    Vissza BASIC- be
Ø 6 Ø 2   L D A    $ F D 1 Ø    PLAY gomb le van- e nyomva ?
Ø 6 Ø 5   R O R
Ø 6 Ø 6   R O R
Ø 6 Ø 7   R O R
Ø 6 Ø 8   B C S    $ Ø 6 2 4    Nincs
Ø 6 Ø A   L D A    $ Ø 1        Beérkező bit, Ø ?
Ø 6 Ø C   R O L
Ø 6 Ø D   R O L
Ø 6 Ø E   R O L
Ø 6 Ø F   R O L
Ø 6 1 Ø   B C C    $ Ø 6 2 4    Igen
Ø 6 1 2   L D A  ↓ $ 7 7        Nem, 1 , hang beállitás, eng.
Ø 6 1 4   L D X  ↓ $ Ø 3
Ø 6 1 6   L D y  ↓ $ 2 8
Ø 6 1 8   S T A    $ F F Ø F
Ø 6 1 B   S T A    $ F F 1 Ø
Ø 6 1 E   S T A    $ F F 1 1
Ø 6 2 1   J M P    $ C E 4 2    Vissza az eredeti Interrupthoz
Ø 6 2 4   L D A  ↓ $ Ø Ø        Hang kikapcsolás
Ø 6 2 6   S T A    $ F F 1 1
Ø 6 2 9   J M P    $ C E 4 2
Ø 6 2 C   n o p
```

Az írógéptől a számítógépig

Akik fellapozták az 1987/88-as tanévre szóló Fővárosi Pályaválasztási Tanácsadó c. kiadványt, még hiába keresték benne az Árendás közi — Újpalota — Közlekedési és Közgazdasági Szakközépiskola címét. Pedig ebben az iskolában már tavaly szeptember óta folyik a leendő közlekedés-gépészeti technikusok és közgazdasági szakemberek képzése.



Eddig ez nem jelent semmi szenzációt, hiszen bizonyára nemcsak ebben az intézményben kezdik meg első ízben a szakemberképzést. Érdeklődésünk inkább arra irányult, hogy hogyan tudja egy újonnan induló iskola a napjainkban oly fontos, a jövőben pedig egyenesen nélkülözhetetlen számítástechnika oktatásának feltételeit megteremteni. Erről Donázy István igazgatótól kaptunk tájékoztatást.

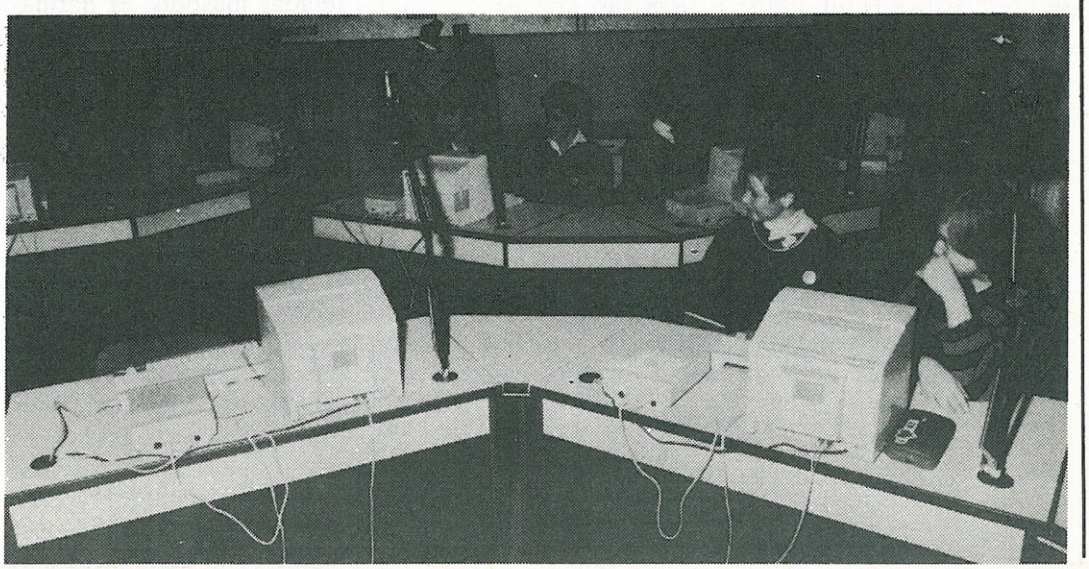
Az épületet eredetileg általános iskolának szánták, de a demográfiai hullám ismert alakulása miatt szükségessé vált középfokú intézménnyé való átalakítása. Ez rendkívül sok pénzt emésztett fel, ezért nem is gondolhattak arra, hogy önerőből komolyabb számítástechnika-oktatási hátteret teremtsenek. A gépbeszerzésekhez a pénzt a Budapest Főváros Tanácsa V. B. művelődésügyi főosztálya, valamint a Közlekedési Minisztérium közreműködésével a közlekedési vállalatok — pl. BKV, Hungarocamion, Volán Tefu stb. — biztosították. Persze ez utóbbiak saját jól felfogott érdekükből tették ezt. Ezeknél a vállalatoknál a régi könyvelőgépeket már lecserélték IBM PC XT vagy AT kategóriájú gépekre. A jelenlegi oktatási rendszerben az iskolák többnyire csak követni próbálják a számítástechnikai igényeket. Az iskolavezetőség itt előrelátó volt, és a régi latin közmondás szellemében csakugyan az életnek tanítanak. Hogyan? A számítástechnika oktatásához két terem áll rendelkezésre.

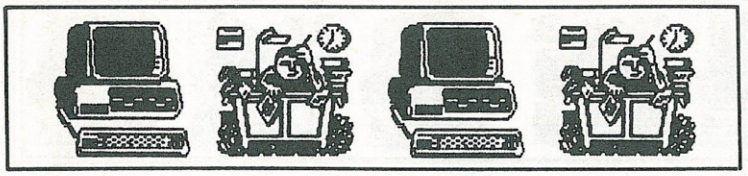
Sikerült beszerezniük tíz C64-et, a TII kedvezményes áron biztosított tíz IBM PC XT és egy AT típusú gépet, melyekhez winchesterek, színes monitorok és nyomtatók is rendelkezésre állnak. A C64-es gépeket arra használják, hogy azok is megismerkedhessenek a számítógéppel, akiknek otthon nincs gépük, tehát ezek amolyan BASIC-kóstolgatásra szolgálnak.

Amikor Kun Emese tanárnő vezetésével elsajátították az alapokat, akkor jön az igazi munka a PC-ken. Az említett közlekedési vállalatok a tervek szerint biztosítják az iskola részére a saját, élesben futó rendszereiket, amelyeket a hallgatók tananyagként sajátítanak el. Azok a hallgatók, akik tanulmányi szerződést kötnek valamelyik vállalattal, így „kész" alkalmazási szakemberként képesek már az első munkanaptól dolgozni, hiszen megtanulták a leckét.

Az átlagosnál érdeklődőbb hallgatók számára a tanterven felül a számítástechnikai szakkörök is rendelkezésre állnak. Az iskola várja az olyan pedagógusok jelentkezését, akik kedvet éreznek a számítástechnika oktatásához.

Sz. S.





Írni nehéz...

Kettőt egy csapásra

A mellékelt listán lévő programot kettős céllal bocsátjuk közre. Részben segíteni kívánunk azoknak (lásd a későbbieket), akik a programot a tanulásban vagy a tanításban alkalmazni kívánják, másrészt pedig gondolkodásra szeretnénk késztetni azokat, akik kedvet éreznek egy program logikájának kibogozásához. A program C Plus/4-es gépre készült; természetesen a C16-tulajdonosok minden változtatás nélkül használhatják.

A program a nem összetett szavakat szótagolja. Sajnos az ékezetes betűket nem ismeri. Ezek kezelése is megoldható lett volna ugyan, de ebben az esetben — célunkkal ellentétben — éppen az egyszerűséget vesztettük volna el. A magyar nyelv elválasztási szabályait az ékezetes vagy ékezet nélküli magánhangzók használata egyébként sem befolyásolja.

Mint ahogyan ezt az általános iskolák tanárai és a korosztályba tartozó gyerekek szülei egyaránt tudják, a kisebbek egy részének nagy gondot okoz a szótagolás, vagy ami ezzel egyenértékű, a szavak elválasztása. Éppen ezért a program kiválóan alkalmas a kicsiknek gyakorlásra, illetve a felnőtteknek a gyakoroltatásra.

Hogy másoknak, a BASIC-ben már többé-kevésbé járatosaknak is hasznosat nyújtsunk, azt javasoljuk, próbálkozzanak meg a program működésének megfejtésével. Ez a feladat a program látszólagos egyszerűsége ellenére sok fejtörést fog okozni. Azok azonban, akik a feladattal eredményesen megbirkóznak, sikerélményre számíthatnak. Az ő dolgukon könnyítendő hívjuk fel a figyelmet a 150-es sorra: ehhez úgy férhetnek hozzá, ha a sorban levő REM utasítást a programból kiirtják.

Horváthné Tőkei Zsuzsanna

```
10 DIMB$(21):DATAA,E,I,O,U,CS,DZ,GY,LY,NY,SZ,TY,ZS
20 DATACCS,DDZ,GGY,LLY,NNY,SSZ,TTY,ZZS
30 FORI=1TO21:READB$(I):NEXTI
40 C$="":SCNCLR:PRINT:INPUT" A TAGOLANDO SZO";A$
50 FORI=1TOLEN(A$)
60 FORJ=1TO5:IFB$(J)=MID$(A$,I,1)THENC$=C$+"0":GOTO140
70 NEXTJ:C$=C$+"1"
80 FORJ=6TO21
90 IFB$(J)=MID$(A$,I,2)THENC$=C$+"2":I=I+1:GOTO140
100 IFB$(J)=MID$(A$,I,3)THENC$=C$+"22":I=I+2:GOTO140
110 IFMID$(A$,I,3)="DZS"THENC$=C$+"13":I=I+2:GOTO140
120 IFMID$(A$,I,4)="DDZS"THENC$=C$+"133":I=I+3:GOTO140
130 NEXTJ
140 NEXTI:C$=C$+"#"
150 PRINT:REM PRINTC$
160 P1=1:I=1
170 IFMID$(C$,I,1)="0"THEN190
180 I=I+1:GOTO170
190 IFMID$(C$,I,1)="#"THEN280
200 I=I+1:IFMID$(C$,I,1)="0"THEN220
210 GOTO190
220 P=VAL(MID$(C$,I-1,1)):P2=I-P
230 PRINTMID$(A$,P1,P2-P1);
240 IFMID$(C$,P2,2)="22"THENPRINTMID$(A$,P2+1,1);:GOTO260
250 IFMID$(C$,P2+1,2)="33"THENPRINTMID$(A$,P2+1,2);
260 PRINT"-";
270 P1=P2:GOTO190
280 PRINTMID$(A$,P2,255)
290 CHAR,10,23,"NYOMJ LE EGY BILLENTYUT!"
300 GETKEYX$:RESTORE:GOTO30
```

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

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

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

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

10. feladat: Labirintusrajzolás

Írjon programot, amely véletlen labirintust rajzol! A labirintusnak a következő tulajdonságai legyenek:

1. Egy be- és egy kijárata legyen.
2. A bejárattól a kijáratig egy és csak egy út vezessen.
3. A bejárattól a labirintus bármely pontjára el lehessen jutni.

Megoldás

Véletlen labirintus rajzolására számtalan módszer képzelhető el. A legegyszerűbb az lenne, ha a képernyőre véletlenszerűen vonalakat húzgálnánk. Ennek az ötletnek a továbbfejlesztése az, hogy a vonalakat nem teljes összevisszaságban húzzuk, hanem a kezdő- és a végpont koordinátáira valamilyen megkötést teszünk. Ha a vonalakat csak páros koordinátájú pontok között húzzuk, akkor egész szép labirintusokat kaphatunk.

Természetesen az ilyen labirintusok nem teljesítik a feladat második és harmadik feltételét. Szinte biztosan tartalmaznak zárt, minden oldalról fallal körülhatárolt részeket, a bejárattól a kijáratig vezető út létezése pedig igencsak kétséges. Ha ezeket a hibákat valahogyan megszüntetnénk, máris megkapnánk a megoldást.

Egy másik lehetséges megoldáshoz juthatunk a következő gondolatmenettel.

Induljunk ki a még felosztatlan táblából! Véletlenszerűen választott helyen osszuk ketté a téglalapot egy olyan fallal, amelyen pontosan egy átjáró van (lásd az *ábrákat*).

Ismételjük meg ezt az eljárást a jobb és a bal oldali téglalapra!

Most végezzük el ugyanezt a keletkezett négy téglalapra! Folytassuk az eljárást mindaddig, míg a téglalapok tovább már értelmesen nem oszthatók!

Az így kapott labirintus a kívánalmaknak minden szempontból megfelel, bármely pontból bármely más pontba pontosan egy út vezet.

Ezenkívül okoskodhatunk a következőképpen is. Ha megvizsgálunk egy, a feladat feltételeinek megfelelő labirintust, láthatjuk, hogy az két összefüggő falrendszerből (fából) áll. A két falrendszer között halad a labirintuson átvezető út. Ha tudunk olyan programot írni, amely ezt a két véletlenszerű fát megrajzolja, akkor lényegében már meg is oldottuk a feladatot. Ilyen fát a következőképpen rajzolhatunk.

Jegyezzük fel egy naplóba a fa meglévő ágainak azon pontjait, ahonnan további ágakat lehet növeszteni. Minden lépésben válasszunk ki véletlenszerűen egy-egy pontot. Vizsgáljuk meg, hogy a négy lehetséges irány közül valamelyikben továbbléphetünk-e. Ha nem, töröljük a pontot a naplóból — egyébként pedig az egyik lehetséges irányba növesszük tovább az ágat, és a kapott új pontot jegyezzük be a naplóba. Amikor elfogynak a folytatható ágak, a labirintus rajzolása befejeződött.

A felsorolt módszerek mindegyike egyaránt alkal-

FELADATOK — MEGOLDÁSOK

mas a feladat megoldására, de talán a másodikat egy kicsivel könnyebb beprogramozni.

A közölt program is ezt használja. A Turbo Pascal 4.0 nyelven megírt program IBM PC-n tetszőleges grafikus adapterrel működtethető.

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

Az eljárás paraméterként megkapja a kettéosztandó téglalap adatait: a bal felső sarok koordinátáit, a téglalap szélességét és hosszúságát. Amennyiben a téglalap szélesebb vagy magasabb mint az SZ konstansban tárolt folyosószélesség, az osztófal (oszt) és a falon átvezető átjáró (lyuk) helyének kiszámítása következik. A következő két sor a fal megrajzolását látja el.

A fal két oldalán keletkezett téglalapok további felosztását Xfelez vagy Yfelez végzi KOMPL konstans értékétől függően eltérő valószínűséggel. Ezzel a labirintus bonyolultságát lehet szabályozni.

A főprogram feladata már csak a grafikus mód beállítása, a külső keret, a be- és kijárat megrajzolása. A labirintus rajzolását a főprogramban meghívott Xfelez végzi el. Végül még szerepel egy ellenőrzés: a FloodFill utasítás kiszínezi a labirintust. Ha az egész képernyőt betölti a szín, az jelzi, hogy a labirintus bármely pontja elérhető a bejárattól indulva.

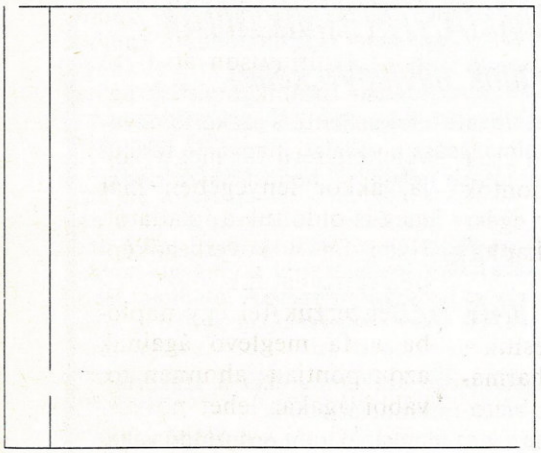
A következő feladat témája bizonyára sokak előtt ismeretes.

11. feladat:
Egér a labirintusban 1.

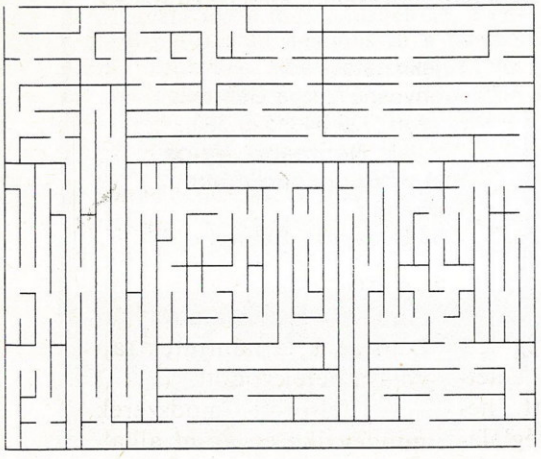
Tegyük az egeret a labirintus bejáratához, a sajtot pedig a kijáratához. Megtalálja-e az egér a sajtot? Ön, mint az egér jó barátja, elhatározta, hogy segít neki. Hogyan? Írjon erre programot! — Tegyük fel, hogy a labirintus térképe nem ismert, az egér csak az őt közvetlenül körülzáró falakat látja.

Pintér Gábor

A tábla az első kettéosztás után



A kész labirintus



```
{--------------------------------------}
{                                      }
{     Labirintus rajzolo program       }
{                                      }
{              Pinter  Gabor           }
{                                      }
{                1988.10.06.           }
{                                      }
{--------------------------------------}

program labirintus;


uses
  Graph;

const
  SZ=20; { Folyoso szelesseg }
  KOMPL=8; { Labirintus bonyolultsag }

var
  MaxX, Maxy : integer;
  Ch         : char;
  Gd, Gm     : integer;

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


{ Fuggolegesen ketteosztja a tablat }
procedure Xfelez(
  x,y,  { Bal felso sarok koordinatai }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztofal helyzete: }
  oszt : integer;
  { Az osztofalon a lyuk helye: }
  lyuk : integer;
begin
{ Kell e tovabb osztani? }
if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
  { Fal rajzolasa: }
  oszt:=x+SZ*Random(dx div SZ -1 )+SZ;
  lyuk:=y+SZ*Random(dy div SZ);
  Line(oszt,y,oszt,lyuk);
  Line(oszt,lyuk+SZ,oszt,y+dy);
  { A fal ket oldalanak levo
    tablak tovabbosztasa: }
  if Random(10)>KOMPL
    then
    Xfelez(x,y,oszt-x,dy)
    else
    Yfelez(x,y,oszt-x,dy);
  if Random(10)>KOMPL
    then
    Xfelez(oszt,y,x+dx-oszt,dy)
    else

    Yfelez(oszt,y,x+dx-oszt,dy)
  end {if}
end {proc};
```

```
{ Vizszintesen ketteosztja a tablat }
procedure Yfelez(
  x,y,  { Bal felso sarok koordinatai }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztofal helyzete: }
  oszt : integer;
  { Az osztofalon a lyuk helye: }
  lyuk : integer;

begin
{ Kell e tovabb osztani? }
if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
  { Fal rajzolasa: }
  oszt:=y+SZ*Random(dy div SZ - 1)+SZ;
  lyuk:=x+SZ*Random(dx div SZ);
  Line(x,oszt,lyuk,oszt);
  Line(lyuk+SZ,oszt,x+dx,oszt);
  { A fal ket oldalanak levo
    tablak tovabbosztasa: }
  if Random(10)<=KOMPL
    then
    Xfelez(x,y,dx,oszt-y)
    else
    Yfelez(x,y,dx,oszt-y);
  if Random(10)<=KOMPL
    then
    Xfelez(x,oszt,dx,y+dy-oszt)
    else
    Yfelez(x,oszt,dx,y+dy-oszt)
  end {if}
end {proc};




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

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

  { Labirintus rajzolas }
  Xfelez(0,0,MaxX,MaxY);

  { Vege }
  Readln(ch);

  CloseGraph;
until ch=#13
end.
```

Pályázati felhívás

Idén nyáron ismét megrendezik Novoszibirszkben a Fiatal Programozók Nyári Táborát, amelyen a Neumann János Számítógéptudományi Társaság Magyarországról összesen négy fő részvételét biztosítja. A tábor munkájában az a négy fő vehet részt — az NJSZT költségén —, aki az alábbi feladatra a legjobb megoldást küldi be.

FELADAT: A felhasználó által adott n számú, mindkét oldalán fehér kártyalapot kiterítünk az asztalra. A kártyákat felfelé néző oldalukon 1-től n-ig sorszámozzuk, majd megfordítjuk és összekeverjük. Az újabb terítést követően a kártyák fehéren maradt oldalát is sorszámmal látjuk el. Ha kész, ismét megkeverjük a lapokat, de közben meg is fordítunk néhányat. Végül az asztalra helyezett kártyák adatait — külön a felfelé, külön a lefelé néző oldalukon szereplő számokat — balról jobbra sorrendben feljegyezzük.

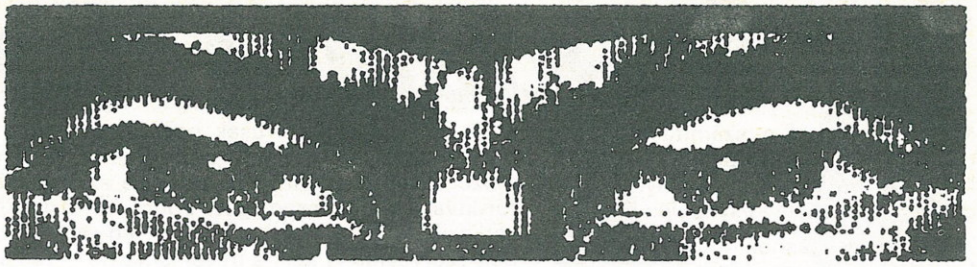
BEKÜLDENDŐ: az a BASIC nyelven írt, printerrel készített programlista és az az algoritmus (blokkdiagram), amely a fenti adatok alapján meghatározza, mely kártyákat kell megfordítanunk ahhoz, hogy a kártyák felfelé néző oldalán az 1-től n-ig terjedő számok mindegyike pontosan egyszer szerepeljen.

Fontos szempont az algoritmus, illetve program elbírálásánál, hogy gyors legyen, és minél kevesebb lépésből álljon.

Beküldési határidő: 1989. április 15.
Cím: Mikroszámítógép Magazin Szerkesztősége
1371 Budapest, Pf. 433.

A nagy sikerű, The last ninja című játék folytatásáról van szó. Ha lehet, még egy fokkal jobb animációja és zenéje, valamint szebb kivitelű grafikája van, mint elődjének.

Történetünk New Yorkban játszódik, amint az a grafikából is kiderül. S még valami, amiben bizonyosak lehetünk: elbizakodottságra semmi okunk a leírás biztonságában sem, mert az ördöngösségekkel újfent meg kell küzdenünk, ahányszor csak birokra kelünk.



The Central Park

Induláskor a Central Park egyik térzenei pódiumán állunk. Menjünk be a függöny mögötti terembe és verjük meg a rejtett levezető lépcső kapcsolóját őrző illetőt. Nyomjuk meg a szemközti falon található kapcsolót — ez beléptünkkor villogással hívja fel magára a figyelmet, mint minden fontos tárgy, amit össze kell szednünk vagy használnunk kell a játék során —, és térjünk vissza a nagydob mellé, ahol közben megnyílt a levezető lépcső ajtaja.

Essünk le — emberünk ruganyosan földet ér. Szedjük fel a bal alsó sarokban lévő kis kulcsot és távozzunk a nyitott ajtón. (A leesés négyszögébe való visszalépéskor visszakerülünk a pódiumra.)

Kilépéskor a parkban találjuk magunkat, és a szemben látható kapu kulcsa van a zsebünkben. De még ne menjünk tovább!

Utasítsuk rendre a bal alsó sarokban mászkáló rendőrt — ezt ő is megteszi velünk, ha túl közel kerülünk hozzá —, és távozzunk. Ezen a pályán az út mellett található díszkockából kivehetjük kedvenc dobócsillagunkat. Tovább haladva lefelé, egy rosszindulatú zsonglőr próbál kilőni minket. Ugráljuk át a lövedékeket. A következő terembe lépve egy újság hívja fel magára a figyelmet. Vegyük fel, és másszunk fel a vasrácson. Balra haladva újabb fegyverünket, egy botot látunk. Fáradjunk el érte és térjünk vissza a kertkapuhoz. Takarékoskodjunk a dobócsillaggal, mert nagy szükségünk lesz rá máshol is!A kapunál elmehetünk felfelé egy falás hamburgerért és onnan jobbra egy WC-hez, amit egy zord vécés néni őriz.

Nyissuk ki a kaput és menjünk be. Ugorjunk be a leszáguldó csónakba, majd onnan a másik partra. A méhek elől szaladjunk a bal felső sarokba, és ugorjunk egy nagyot, ugyanis a látszat ellenére ott egy sziget található. A szigeten lökjük el a csónakot a bottal, majd térjünk vissza szúrós barátainkhoz. Most balra távozzunk, és ismételjük meg az előző pályán tanult csónakos mutatványt. A túlparton induljunk lefelé.

The street

Az utcára lépve induljunk felfelé, a következő pálya után menjünk át a lámpás zebrán, lehetőleg zöldnél, különben elüt egy jóképű motoros. A lámpa után menjünk balra. Az itt található lelkes amatőrt verjük meg és rúgjuk be a szemközti ajtót. Vegyük le a falon lévő kardot. Visszafelé menet ne a lámpás, hanem a sima zebrán keljünk át, és óvakodva a virágcserepektől, menjünk tovább. Vegyük fel a hamburgert, majd a következő pályán a csavargó előtti üveget és menjünk át a zebrán. Tovább haladva intézzük el a két botos urat és kutassuk át a második bódéját. Egy fura szerszámot találunk, ami nagyon fontos. Térjünk vissza a rendőrig (a hamburgeres kocsi előtti pálya), ahol egy zebra vezet lefelé. Menjünk át rajta. Innen már csak három pályányi a kiindulópontunk. Menjünk újra felfelé, de most a rendőr mellett forduljunk jobbra. Itt találunk egy csatornafedőt, amit a szerszámmal felnyithatunk. Ereszkedjünk alá.

The sewers

Induljunk jobbra, majd onnan a második pályán vegyük fel a földön lévő szerszámot. Most folytassuk utunkat felfelé az előző pályán, a nyílást átugorva. Ugorjunk át ismét és várjuk meg, amíg a gusztusos pók elgaloppozik, és siessünk utána. Innen már csak háromteremnyire van a kiindulópont. Ott emeljük fel a vasrácsot a szerszámmal, és szálljunk alá. A leérkezést követő pályán menjünk be a középső ajtón. Ezután botos barátunk mellett a bal szélső ajtón menjünk be. Várjuk meg a patkányinváziót, majd a következő patkánycsapatot átugorva lejtsünk tovább egyenesen. Innentől a második pályán szintén a bal szélső ajtó a jó. Továbbhaladáskor egy éhes krokodil társaságát élvezhetjük. Térjünk vissza az előző terembe, és gyújtsuk meg a fáklyánál a csavargó elől felvett üveget. Ezzel kedveskedjünk a krokodilnak egy jól célzott dobással. A fényes jelenség után látogassuk meg a néhai krokodil lakását.

The basement

Induljunk a ládák között balra (eleinte ez nem lesz könnyű). A verekedés után másszunk fel a létrán és menjünk vissza az előző terembe a mágneskártyáért. Fussunk tovább a galérián, egész annak végéig. Ott lépjünk be a szűk ajtón és szedjük fel a csirkecombot. Menjünk vissza a létrához és másszunk le.

A csillét átugorva térjünk vissza a galéria végéhez, és menjünk ki lent. A „rázós" síneket átugorva vegyük fel a hamburgert. Folytassuk utunkat a ládatornyokig. Ugorjunk először a túlpart felé esőre, onnan a legalsóra, majd tovább a kicsire, és végül a másik oldalra. Máris mehetünk tovább! A laboratóriumon és a vegyszerraktáron túli teremben mártsuk a csirkecombot a villogó méregbe. Térjünk vissza a laborba, de most az ajtótól jobbra távozzunk. Nyújtsuk át a mérgezett húst az itt szendergő párducnak. A következő teremben helyezzük a mágneskártyát a leolvasóba, amire kinyílik az ajtó.

The office

Robogjunk végig a folyosón egészen az első jobbra nyíló ajtóig. Menjünk be és olvassuk le számítógépről a kódot. Ezt jegyezzük fel, mert szükségünk lesz rá. Menjünk tovább két szobával, a két irodaajtóhoz. Térjünk be a szembeni irodába. Itt az íróasztalon található egy titkos kapcsoló, ami a mellette lévő ajtót nyitja. Aktivizáljuk a kapcsolót, lépjünk be, másszunk fel a létrán. A fegyverműhely után evickéljünk el a ventilátor előtt. Kapirgáljunk a mellette lévő vasrács alján, mire az felnyílik. Az épület párkányán jobbra indulva egyensúlyozzunk egészen a létráig. Másszunk fel és fussunk tovább, majd kapaszkodjunk fel a helikopterre.

The mansion

Ugorjunk le a gépről a leghátsó bástyafokra, különben ledobnak. Ugorjunk át az előtte levőre, és onnan a különállóra. Erről hulljunk alá a tetőn lévő vízszintes sávra, és fussunk tovább. A következő pályán essünk be a tetőablakon. Száguldjunk a lépcsőlejáróig, és térjünk be az ott jobbra található szobába. Szedjük fel a kötelet és menjünk le a lépcsőn. A lépcső és a növény között van egy kis nyílás, csússzunk be rajta. Motozzunk a lenti kapcsolótáblán, ami az ajtót nyitja. A mini labirintuson át a kazánházba jutunk. Állítsuk el a gőzcsapot a jobb oldali gombnál, különben megpörkölődünk. Induljunk tovább.

The final battle

Verekedjük át magunkat a következő terembe. Rántsuk fel az írásjelekkel ellátott függönyt. Nyomjuk meg a páncélszekrény elejét, majd állítsuk be az Office-ban leolvasott kódot. A szekrény kinyílik, mi pedig vegyük ki belőle a tárgyat. A megjelenő fehér harcost öljük meg a csillag közepén, majd sietve gyújtsuk meg a csillag csúcsain a gyertyákat. A tárgyat visszatehetjük a szekrénybe.

Ezek után a gép kifejezi elismerését a gonosz shogunon aratott győzelmünkért.

Tass Csaba

The last ninja II.

BELEÉLÉSTŐL A TÁLALÁSIG

PROGRAMOZÁSI STÍLUSOK

Mindenkivel, aki a számítástechnikát műveli, netán programoz, előfordul, hogy kap egy feladatot, amelyre működő programot kell készítenie.

A programozásnak mindig vannak jó és rossz velejárói, lényege azonban, hogy változatos. Minden feladat más és más, végül is ez adja a munka szépségét. Jó programot csak úgy lehet írni, ha valaki beleéli magát az adott feladatba, látja ez értelmét, és gondol arra, hogy a programját, „gyerekét", „szellemi szüleményét" használni fogják. E felfogás nélkül nem lehet, s nincs is értelme hozzálátni a munkához.

De lássunk neki, kezdjünk el gondolkodni! Sohase feledjük figyelembe venni a megvalósítás gyakorlati korlátait. Ezen értem a hardverbázist, az anyagi lehetőségeket, az emberi pontatlanságot, a felhasználó kívánságait stb.

Ilyenkor felmerül a kérdés, hogy mi most programozunk vagy épp egy folyamatot szervezünk meg? Erre a kérdésre azt mondhatom, hogy mindkettőt tesszük. Mert a programozás és a szervezés ilyen szinten nem különíthető el egymástól. Ámbár ha belegondolunk, ez az állításom csak egyoldalú, hiszen a szervező a munkáját a programozó nélkül is el tudja végezni, de ez fordítva nem egészen igaz. Ebből következik, hogy a programozónak egyben szerveznie is kell, s mivel ismeri a feladatot, átlátja, átláthatja a legcélszerűbb gyakorlati megvalósítást. Viszont, ha az adott feladat oly nagy, hogy több programozónak kell dolgozni rajta, akkor mindenképpen ajánlatos egy rendszerszervező, aki a feladat legmagasabb szintjén összehangolja az egyes programozók munkáját. De az adott programozóra eső feladat megvalósításába nem szól bele, neki konkrét *bemenő* adatok után konkrét *kimenő* adatok kellenek. Ha egy ilyen nagy munkához túl sok szervezőt használunk, akkor a jó rendszer helyett egy túlbonyolított, lassú, nem hatékony valamit kapunk, ez pedig senkinek nem érdeke.

A legfontosabb alappillér a felhasználótól kapott bizonylatok és kimeneti listák terve, erre kell minden be/kiviteli műveletet a későbbiekben felépíteni. Ha a feladat nagysága, milyensége, bonyolultsága megkívánja, akkor mindenképpen folyamatábrát vagy blokksémát, illetve számunkra jól értelmezhető feladattervet kell készíteni. Ha ezzel megvagyunk, akkor már gondolkodhatunk a gyakorlati megvalósításon. Fontos a be/kiviteli rekordtervek elkészítése; ezt célszerű a felhasználóval közösen megcsinálni a későbbi bonyodalmak elkerülése céljából.

Már csak egy igazán nélkülözhetetlen lépésünk van hátra: a rendszer interaktivitásának megtervezése. Ezen a képernyőterveket és a menüstruktúrákat értem. Lényeges, hogy ne csak jól, megbízhatóan működjön a rendszerünk, hanem szép is legyen, ami amolyan lelki tükör a felhasználó számára, aki mit sem sejt a program belsejéről. Ebben a fázisban össze kell hangolnunk az esztétikát a praktikussággal, mert egyiknek sem szabad elnyomnia a másikat. Ezt a szakács tudásával és ízlésével lehet a legjobban bebizonyítani: ugyebár az ételnek finomnak kell lennie, hogy a kuncsaftok elfogyasszák, de ez nem elég, az étel külleme, esztétikája, tálalása is fontos. Egy jó program ezt mindenképpen figyelembe veszi!

A gyakorlati megvalósítás folyamatában gondolnunk kell a felhasználóra is, aki kezeli a rendszert, tehát gondoskodnunk kell a betanításról és a jó programleírásról is.

Ahhoz, hogy az itt leírtakról egy kis vizuális fogalmunk legyen, képzeljük el a következő gyakorlati feladatot.

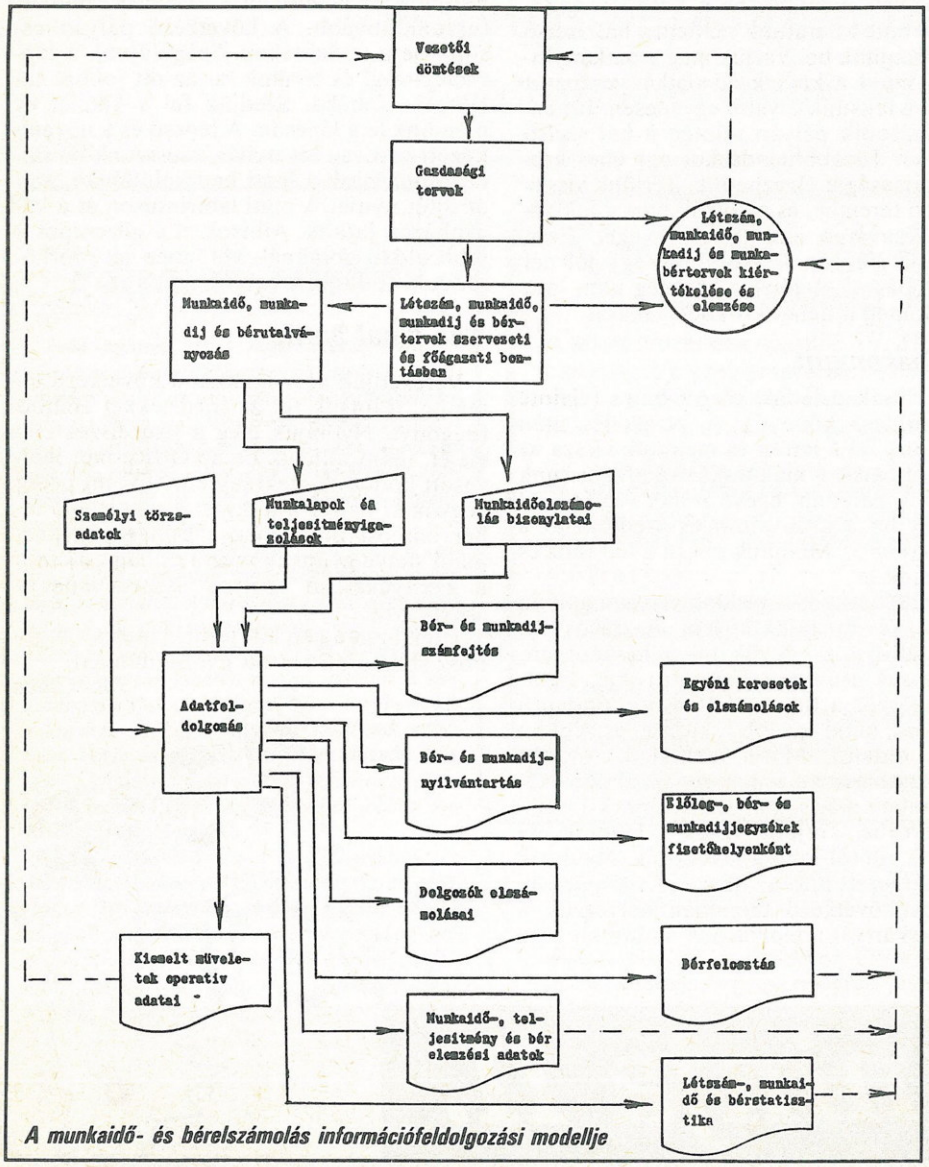
Munkaidő- és bérelszámolás, nyilvántartás

A munkaidő-, munkadíj- és bérelszámolás munkaigényessége közismert. Az e területen eddig végzett szervezési munkák tapasztalatai azt igazolják, hogy helyes szervezési megoldásokkal ez a munkaigényesség jelentősen csökkenthető.

A munkaigényesség abból ered, hogy a munkaidő-, munkadíj- és bérelszámolás igen sokrétű információfeldolgozást tartalmaz, ezek manuális megoldása többnyire csak külön-külön munkafolyamatokkal biztosítható.

A gépi feldolgozás eszközei már lehetőséget teremtenek arra, hogy az alapbizonylatok adatait egyidejűleg több szempont szerint feldolgozzák, tehát az e területen igényelt információk kevés vagy együtemű feldolgozásából származnak. Az adatfeldolgozás rendszerének kidolgozásánál alapvető követelmény, hogy a rendszer az alapbizonylatok együtemű feldolgozásával lehetővé tegye a bérszámfejtést, a bérjegyzék elkészítését, a munkaidő- és bérnyilvántartás vezetését, a létszám-, a munkaidő- és a teljesítményadatok megfelelő részletezésű gyűjtését, a bérfelosztást, a létszám-, a munkaidő- és a bérstatisztika elkészítését és a munkavállalók különféle tartozásainak nyilvántartását.

Bartos Gyula



A munkaidő- és bérelszámolás információfeldolgozási modellje

ÖRÖKÉLET

Kerecsényi István megörvendeztetett bennünket azzal, hogy C64-re készült játékok örökélet-kódjait elküldte hozzánk. Saját bevallása szerint kedvenc szórakozása, hogy különböző játékprogramokba ilyen POKE-okat beépít. Úgy gondolta, megosztaná ezeket olvasóinkkal is. Íme hát:

SUICIDE EXPRESS (csak lemezre): POKE 12231,234: POKE 12232,234: POKE 12233,234

STARWARRIOR: POKE 40788,255

GUNFIGHTER (RUN/STOP RESTORE UTÁN):
POKE 49963,255:POKE 49965,0:SYS 50887

CATCH AND KILL!: POKE 6397,PEEK(23):SYS 9152
Hatása, hogy a játék elviselhetetlenül lelassul

ANIHILATOR: 6295,11

MRS PAC MAN: 8090,173

COSTA CAPERS: POKE 23843, életek száma

BOMBO: POKE 8887, keret színe (a játék megoldásában ugyan nem könnyít, de érdekes hatásokat érhetünk el vele)

ELITE II: POKE 45229,69:POKE 51211,96 (kalóz űrhajók nem támadnak)

GORF: POKE 7240,234:POKE 7241,234

SMURF: POKE 3221,96 (nem harapós a farkas)

THE GIANT EPIRE: POKE 36211,0:36912,0:36913,0

NEPTUNES DAUGHTER: 7870,80

PLANET DESTROYER: 9926,234 (a keretben nincsenek lövőállások)

PRESENTS III: Írjuk be a listára, ha rekordot döntöttünk: CHEAT 64. Ezek után nem halálos egyetlen ellenség sem

FROGGER: POKE 22341,173

WARDERBORDER: Mikor a második lemezoldalról elkezd tölteni, RESET gombbal állítsuk le, majd töltsünk be egy C000-nál kezdődő monitorprogramot, írjuk be az M utasítással a 8400-tól kezdődően a következő bájtsorozatot:
M 8400 A6 00 EA EA EA 8D 00 40
M 8408 9D 15 D0 EA EA 60 00 00
A programot a G 8800 utasítással indíthatjuk monitorból, illetve X utasítás után SYS 34816-tal. Ennek az utasítássorozatnak a hatása: a bolygó közelében a gravitáció elhanyagolhatóvá válik, megkönnyítve az egyébként igen nehézkes manőverezést.

Fele(más)nyomtatás

SIC utasításokból álló egyszerűbb programok vagy csak adatok is átvihetők a C16-ról a HT-ra. *Gyurkovits Gergely*

Magyarországon nagyobb számban először a HT—1080Z típusú iskolaszámítógép terjedt el. A hosszú idő alatt, amíg uralta ezt a címet, ha tehette, sok iskola vásárolt hozzá nyomtatót. Manapság azonban az oktatásban népszerűbbek lettek a Commodore gépek. Ha a C16-hoz vagy C Plus/4-hez nincs nyomtató, viszont van egy HT és egy hozzá kapcsolt nyomtató, akkor a következő programok segítségével és a két gép összekapcsolásával a C16-ról nyomtathatunk.

Ez természetesen csak példa a felhasználásra, tehát az eljárás alkalmazható — a programok módosításával — adatok továbbítására a C16-ból a HT-ba. Az adatátvitel sorosan működik: a C16 soros buszát kell a HT digitális be- kimenetével összekapcsolni, az *ábra* szerint.

A HT programját célszerű SYSTEM paranccsal betölthető formában kazettára venni.

A gépeket lehetőleg kikapcsolt állapotban kapcsoljuk össze. A nyomtatás kezdetekor először a programokat be kell tölteni a gépekbe, majd a HT programját monitorból G7000 paranccsal (vagy SYSTEM/28672) indítsuk el. Miután betöltöttük a C16-ba a nyomtatandó programot, és ha jó a két gép közötti csatlakozás, akkor a SYS 1536 parancs után minden üzenet, amit a C16 kiír, a HT nyomtatóján is megjelenik. Tehát ha például a C16-ba betöltött BASIC programot akarjuk kinyomtatni, akkor a LIST paranccsal kilistázzuk, így a lista megjelenik a nyomtatón is. Az adatok továbbítása a HT-ba két vezetéken, „handshake" elven lehetséges, ezért a C16 listázásának sebessége a nyomtatóhoz fog igazodni.

A gépeket összekapcsoló háromszálas vezetéken levő soros busz, illetve a HT digitális port csatlakozójának beszerzése sajnos többé-kevésbé gondot jelent. Ennek ellenére Commodore nyomtató hiányában olcsó és egyszerű megoldás a HT-val való összekapcsolás. A programok fejlesztésével akár mindkét gépen létező BA-

A HT programja

```
 1                          DERKO
 2
 3                                  ORG  7000H
 4                                  LOAD 7000H
 5 7000 3E07                        LD   A,7
 6 7002 D31F                        OUT  (31),A
 7 7004 3E80                        LD   A,128
 8 7006 D31E                        OUT  (30),A
 9 7008 3E0F                        LD   A,15
10 700A D31F                        OUT  (31),A
11 700C D31E                        OUT  (30),A
12 700E 0608                        LD   B,8
13 7010 3E0E               HT:      LD   A,14
14 7012 D31F                        OUT  (31),A
15 7014 DB1F                        IN   A,(31)
16 7016 1F                          RRA
17 7017 38F5                        JR   C,HT-2
18 7019 3E0F                        LD   A,15
19 701B D31F                        OUT  (31),A
20 701D AF                          XOR  A
21 701E D31E                        OUT  (30),A
22 7020 3E0D                        LD   A,13
23 7022 3D                          DEC  A
24 7023 20FD                        JR   NZ,$-1
25 7025 3E0E                        LD   A,14
26 7027 D31F                        OUT  (31),A
27 7029 DB1F                        IN   A,(31)
28 702B 1F                          RRA
29 702C CB11                        RL   C
30 702E 3E0F                        LD   A,15
31 7030 D31F                        OUT  (31),A
32 7032 D31E                        OUT  (30),A
33 7034 3E0D                        LD   A,13
34 7036 3D                          DEC  A
35 7037 20FD                        JR   NZ,$-1
36 7039 10D5                        DJNZ HT
37 703B 79                          LD   A,C
38 703C CD3B00                      CALL 003BH
39 703F CD3300                      CALL 0033H
40 7042 C30070                      JP   7000H
41                                  END
```

A C16 programja

```
. 0600  A9 C8     LDA #$C8
. 0602  85 01     STA $01
. 0604  A9 0F     LDA #$0F
. 0606  A2 06     LDX #$06
. 0608  8D 24 03  STA $0324
. 060B  8E 25 03  STX $0325
. 060E  00        BRK
. 060F  78        SEI
. 0610  86 D0     STX $D0
. 0612  A2 08     LDX #$08
. 0614  48        PHA
. 0615  48        PHA
. 0616  A5 01     LDA $01
. 0618  09 01     ORA #$01
. 061A  85 01     STA $01
. 061C  A5 01     LDA $01
. 061E  29 40     AND #$40
. 0620  D0 FA     BNE $061C
. 0622  68        PLA
. 0623  0A        ASL
. 0624  48        PHA
. 0625  08        PHP
. 0626  A5 01     LDA $01
. 0628  4A        LSR
. 0629  28        PLP
. 062A  2A        ROL
. 062B  49 01     EOR #$01
. 062D  85 01     STA $01
. 062F  A5 01     LDA $01
. 0631  29 40     AND #$40
. 0633  F0 FA     BEQ $062F
. 0635  A5 01     LDA $01
. 0637  29 FE     AND #$FE
. 0639  85 01     STA $01
. 063B  CA        DEX
. 063C  D0 D8     BNE $0616
. 063E  68        PLA
. 063F  68        PLA
. 0640  A6 D0     LDX $D0
. 0642  58        CLI
. 0643  4C 4B EC  JMP $EC4B
```

```
10 PRINTCHR$(14),CHR$(8)
20 DATA162,128,142,15,48,142,18,48,160,0
25 DATA141,62,255,185,0,128,153,0,128
27 DATA200,208
30 DATA244,232,224,253,208,231,162,73
35 DATA189,0,255,157,0,255,232,208,247
37 DATA162,0,189
40 DATA58,48,157,247
50 DATA3,232,224,38,208,245,32,247,3,141
55 DATA63,255,96,162,0,189,10,4,141,2,4
57 DATA169,63
60 DATA141,115,4,232,224,8,208,240,96
65 DATA130,158,172,183,194,205,216,227
70 DATA173,18,255,41,251,141,18,255,76
75 DATA14,206
80 FORI=0TO95:READA:POKE12288+I,A:NEXTI
90 POKE13000,120:POKE13001,96:SYS13000
100 SYS12288:POKE789,4:POKE788,18
110 POKE13000,234:POKE13001,96:SYS13000
120 POKE65532,18:POKE65533,128
125 POKE37041,156
130 POKE34391,70:POKE34395,244
140 POKE46906,2:POKE52592,70:POKE2042,0
150 POKE33811,186:POKE33812,127
160 REM****  KARAKTEREK  ***
170 DATAD780,2,4,10,3C,66,7E,66,0:REMAA
180 DATAD788,4,8,7C,60,78,60,7C,0:REMEE
190 DATAD710,72,24,20,20,20,20,70,0:REMI
200 DATAD7C8,4,8,3C,66,66,66,3C,0:REMOO
210 DATAD730,14,0,3C,66,66,66,3C,0:REMO
220 DATAD6D8,A,14,3C,66,66,66,3C,0:REMO
230 DATAD7C0,8,10,66,66,66,66,3C,0:REMUU
240 DATAD6E0,14,0,66,66,66,66,3C,0:REMU
250 DATAD6E8,14,28,66,66,66,66,3C,0:REMU
260 DATAD6F8,4,8,3C,6,3E,46,3E,0:REMA
270 DATAD600,4,8,3C,66,7E,60,3E,0:REME
280 DATAD400,2,4,30,30,30,30,30,0:REMI
290 DATAD4D8,4,8,3C,66,66,66,3C,0:REMOO
300 DATAD4E8,8,10,66,66,66,66,3C,0:REMUU
310 FORI=0TO13:READA$:FORY=0TO7
320 READB$:POKEDEC(A$)+Y,DEC(B$):NEXTY,I
330 REM **** UTASITASOK ***
340 B$(1)="VEG SZAMOL KOVETKEZ ADAT BE "
345 B$(1)=B$(1)+"B TOM OLVAS@ MEN FUS@ "
347 B$(1)=B$(1)+"H@UJR@SZUBRUTIN VISSZ@"
350 B$(1)=B$(1)+"JEGYE@ALL MUTAT VAR TOL"
355 B$(1)=B$(1)+"T@MENT@EL DE TA IRJ IR"
357 B$(1)=B$(1)+" FOLYTLIST@CL CELGEP "
360 B$(1)=B$(1)+"NYIS@ZAR FOG U TAB TOL"
365 B$(1)=B$(1)+"F/SPC AKKO NE LEPE@ "
367 B$(1)=B$(1)+"_"+CHR$(222)+"E@VAG "
370 B$(1)=B$(1)+" SG/IN AB@US FR 0"
380 B$(2)="PO@GYO/VELLO EX DO@SI/TA/AT/"
385 B$(2)=B$(2)+"LE@TA STR ER KO KAR BA"
387 B$(2)=B$(2)+"L JOBB KOZEP INDUL RG_"
388 B$(2)=B$(2)+"RCL_"
390 B$(2)=B$(2)+"RLU JOYSTIC/PIXEL DE HE"
395 B$(2)=B$(2)+"X ERR HKA VMIN HIB@TRA"
397 B$(2)=B$(2)+"TRO/TROF ZEN ERTAU "
398 B$(2)=B$(2)+"ATI_"
400 B$(2)=B$(2)+"GRAFIK@FES IR KOCK@KO_"
405 B$(2)=B$(2)+"BTE KTE VONAL PONI SZI/T"
407 B$(2)=B$(2)+"OROL KOCS SO@A@C IKLKI "
410 B$(2)=B$(2)+"TARTALO@"
420 B$(3)="KIMEN BETOL INICIALVOL SZABA"
425 B$(3)=B$(3)+"MASOLUJNEXALL M@BASIC"
427 B$(3)=B$(3)+"ATSZA GOM COMPUTE FORM"
430 B$(3)=B$(3)+"@IDAI BA_@"
440 B$(4)="0"
450 A=1:B=1:C=0
460 D=ASC(MID$(B$(A),B,1))
470 IFD=48THENB=1:A=A+1:GOTO460
480 IFA=4THENGOTO520
490 POKE33166+C,D
500 C=C+1:B=B+1
510 GOTO460
520 REM***HIBAJELZESEK***
530 A$="TJL SOK FIL FILE NYITV@NEM NYIT"
535 A$=A$+"OTT FIL NEM L"+CHR$(96)+"TEZ"
537 A$=A$+CHR$(126)+" FIL "
540 A$=A$+"NEM L"+CHR$(96)+"TEZ"
545 A$=A$+CHR$(126)+" EGYS"+CHR$(127)
547 A$=A$+"!"
550 A$=A$+"NEM BE FIL NEM KI FIL HI"
555 A$=A$+CHR$(96)+"NYZI FILE N"
557 A$=A$+CHR$(127)+"X"+CHR$(127)
560 A$=A$+"RV"+CHR$(127)+"NYTELEN C@\"
570 A$=A$+"'KOVETKEZO','SZAMOLJ'N"
575 A$=A$+CHR$(127)+"LK"+CHR$(124)
577 A$=A$+"LNYELVTANI HIB@"
580 B$="'VISSZA','SZUBRUTIN'N"+CHR$(127)
585 B$=B$+"LK"+CHR$(124)+"L"
590 B$=B$+"KEV"+CHR$(96)+"S ADAI "
595 B$=B$+CHR$(96)+"RV"+CHR$(96)
597 B$=B$+"NYTELEN MENNYIS"+CHR$(96)
600 B$=B$+"!TJLCSORDUL"+CHR$(127)
605 B$=B$+"@KICSI A T"+CHR$(127)+"_"
610 B$=B$+"NEML"+CHR$(96)+"TEZ"
615 B$=B$+CHR$(126)+" UTAS@T"+CHR$(127)
620 B$=B$+"@HELYTELEN TOMBINDE@"
630 B$=B$+"T"+CHR$(126)+"MB IJRA DIMENZ"
635 B$=B$+"ION"+CHR$(127)+"L"+CHR$(127)
640 B$=B$+"@OSZT"+CHR$(127)+"S NULL"
645 B$=B$+CHR$(127)+"VAL"+CHR$(96)+"RV"
650 B$=B$+CHR$(96)+"NYTELEN PARANC@NEM "
655 B$=B$+"EGYEZ"+CHR$(123)+" T@PU@TJL "
657 B$=B$+"HOSSZ KARAKTERL"+CHR$(127)
660 B$=B$+"N HIB"+CHR$(127)+"S ADAI TJL "
665 B$=B$+"BONYOLULT K"+CHR$(96)+"PLEI "
670 C$="NEM FOLYTATHATO\NEM DEFINI"
675 C$=C$+CHR$(127)+"LT F"+CHR$(124)
677 C$=C$+"GGV"+CHR$(96)+"N!"
680 C$=C$+"ELT"+CHR$(96)+"R"+CHR$(96)
685 C$=C$+"@BET"+CHR$(126)+"LT"+CHR$(96)
690 C$=C$+"SI HIB@MEGSZAK@T"+CHR$(127)
695 C$=C$+"@'HIBA','TRAP'N"+CHR$(96)
697 C$=C$+"LK"+CHR$(124)+"LAZ','CIKL'N"
700 C$=C$+CHR$(127)+"LK"+CHR$(124)+"L'C"
705 C$=C$+"IKL','AZ'N"+CHR$(96)+"LK"
710 C$=C$+CHR$(124)+"LCSAK PARANCSMIDBA"
715 C$=C$+"/NEM VOLT 'GRAFIKA'UTAS@T"
717 C$=C$+CHR$(127)+"@ROSSZ LEME@"
720 FORI=1TOLEN(A$)
725 POKE62533+I,ASC(MID$(A$,I,1)):NEXTI
730 L=62533:FORI=1TOLEN(B$)
735 POKEL+I+LEN(A$),ASC(MID$(B$,I,1))
737 NEXTI
740 K=LEN(A$)+LEN(B$):FORI=1TOLEN(C$)
745 POKE62533+I+K,ASC(MID$(C$,I,1))
747 NEXTI
750 FORI=0TO5:POKE34548+I,32:NEXTI
760 POKE34549,144
770 REM***MAGNO***
780 A$="NYOMJ":B$="GOMBOT ":C$="EZ  ?"
785 D$="K SZ !"
790 FORI=1TOLEN(A$)
800 POKE58151+I,ASC(MID$(A$,I,1)):NEXTI
810 FORI=1TOLEN(B$)
820 POKE58182+I,ASC(MID$(B$,I,1)):NEXTI
830 FORI=1TOLEN(C$)
840 POKE59883+I,ASC(MID$(C$,I,1)):NEXTI
850 FORI=1TOLEN(D$)
860 POKE34419+I,ASC(MID$(D$,I,1)):NEXTI
870 REM **  UJ SZAVAK  **
880 DATA169,96,141,82,139,162,128,142
885 DATA186,127,138,32,147,139,32,79,255
887 DATA13,29,29
890 DATA145,29,29,29,29,29,29,29,29,29
895 DATA29,29,0,141,62,255,174,186,127
900 DATA138,160,255,132,73,32,147,139
905 DATA141,63,255,169,13,32,210,255,174
910 DATA186,127,232,224,255,208,200,169
915 DATA201,141,82,139,96:REM **COMPUTER
920 RESTORE880
930 FORI=0TO68:READA:POKE32699+I,A:NEXTI
940 KEY 1,"GRAFIKA"
945 KEY 2,"LOAD"+CHR$(13)
947 KEY 3,"COMPUTER"+CHR$(13)
950 KEY 6,"FUSS"+CHR$(13)
955 KEY 7,"LISTA"+CHR$(13)
960 KEY 4,"":KEY 5,"":KEY 8,""
970 PRINT"    /|@_| |@@ - X3.5 ";
975 PRINT"28416 | || @ _ ":NEW
```

„ÁTÜLTETÉS" GYEREKEKNEK IS

Bizonyára sokan gondoltak már arra, hogy jó lenne magyar nyelven programozni a számítógépet. A következő programmal, amely C Plus/4-es számítógépre készült, ennek már semmi akadálya. Kisebb gyerekeknek is segíthet a programozás elsajátításában.

A PROGRAM ÁLTAL NYÚJTOTT LEHETŐSÉGEK:

- **magyar nyelvű utasítások, parancsok**
- **ékezetes betűkészlet**
- **magyar nyelvű hibajelzések**
- **magyar nyelvű magnókezelési felhívások**
- **COMPUTER utasítás, amelynek segítségével megtudhatjuk az új utasításokat**
- **RESET letiltás**

A PROGRAM RÉSZEI:

1Ø—11Ø A gépi kódú rész betöltése és futtatása
16Ø—32Ø Karakterek átírása
33Ø—51Ø Új utasítások
52Ø—76Ø Hibajelzések
77Ø—86Ø Magnókezelési jelzések
87Ø—93Ø COMPUTER utasítás
94Ø—96Ø Funkcióbillentyűk beprogramozása
97Ø „HUNGARY BASIC V 3.5 28416 BYTES FREE" kiírása

A gépi kód részei:

3ØØØH ROM—RAM másolás
Ø3F7H BASIC RAM-ra kapcsolás
Ø412H Karaktertár RAM-ra kapcsolása
Kilépés a programból: POKE 1Ø24,62:SYS 1Ø15Belépés a programba: POKE 1Ø24,63:SYS 1Ø15Végleges kilépés: SYS 65526

ÉRDEKESSÉGEK:

- **Z/Y betűk cseréje: POKE 57394,89:POKE 574Ø7,9Ø**
- **Karaktertár DØØØH-tól szabadon felülírható**
- **A betöltött BASIC programot „magyarosítja"**

Az előbbiekben leírt POKE és SYS utasítások a program elindítása után már nem élnek. A magyar megfelelőjüket a COMPUTER utasítással találhatjuk meg. Ajánlatos a programot beírás után kimenteni, mert így sok bosszúságtól kíméljük meg magunkat.

Horváth Gábor

Listánkat felhasználói, illetve játékprogramokból állítjuk összse. A legjobbakat, legérdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprise-ra, TVC-re, Atarira és IBM-re készült programrangsorokat várunk havonta.

Címünk
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
Diákszerkesztőség

	Felhasználói programok	IBM PC	AMIGA	C-128	C-64	C+4,16	SPECT.	ENTER.	TVC
1	Giga Paint				*				
2	Giga Cad +				*				
3	Printfox	*			*				
4	News room	*			*				
5	Geos 1.6 d.				*				
6	Windows	*			*		*	*	
7	Disk m. 3.2				*				
8	WordStar5	*							
9	Cheese				*				
10	Geos 1.32				*				

	Játék programok	IBM PC	AMIGA	C-128	C-64	C+4,16	SPECT.	ENTER.	TVC
	Zak McKraken				*				
	Test Drive II.	*			*				
	Last Ninja II.				*				
	Skyfox II.				*				
	Renegade						*		
	Pentis	*							
	Slap fight				*				
	Microp.soccer				*				
	Impossible II.				*		*		
	Pacmania +				*				

bicskiller

A számítógépek motorja III.

Erőforrás-kezelés

Alapelvek

Amikor még a számítógépes rendszerekben a hardver ára dominált, erőforráson a rendszer fontosabb hardverkomponenseit értették: a központi egységet, a memóriát, a B/K csatornákat és a különböző perifériákat. Az idők folyamán ez a szemlélet megváltozott, az erőforrások köre a szoftver erőforrásokkal, azaz programokkal és adatokkal bővült. A szoftver erőforrások — mivel ezek élőmunka-tartalma magas és egyre növekvő — értéke emelkedik, a hardver erőforrásoké pedig élőmunka-tartalmukkal együtt csökken. Gondoljunk például egy kiterjedt bankhálózatban tárolt adatok értékére; elvesztésük jóformán helyrehozhatatlan kárt okozna.

Általában elmondható, hogy az operációs rendszerek egyik fontos funkciója a teljes rendszer erőforrásainak kezelése. Ez azt jelenti, hogy az operációs rendszer gondoskodik az erőforrások elosztásáról, védett használatáról és az elosztás során kialakuló versenyhelyzetek kezeléséről.

Az erőforrás kezelésekor az operációs rendszer maga is létrehoz később elosztandó erőforrásokat. Ilyenek például a virtuális tárkezelésnél a lapok, a B/K során használt pufferek, vagy akár az adatállományok.

Az erőforrás-kezelés két alapvető célja a számítógéprendszer működésének valamiképpen gazdaságossá tétele és a holtpontthelyzetek kivédése. A két cél annyiban összefügg, hogy ha lemondanánk a gazdaságos üzemről, történetesen az erőforrások osztott felhasználásáról, a holtpont kérdése fel sem merülne. A fejezetben az erőforrással való gazdálkodásnak csupán magasabb szintű stratégiáival foglalkozunk, a fontosabb konkrét erőforrások — processzor, memória, állományok, programok — ütemezésének külön fejezeteket szentelünk.

Gazdaságosság

A számítógép használati értékét elsősorban az a haszon határozza meg, amit a gép használata közvetlenül vagy közvetve hajt. Ilyen lehet a termelési költség csökkentése, az adatok tárolása, a felhasználók termelékenységének növekedése és az információk minőségileg értékesebb és jobban ütemezett szolgáltatása. A bevételt természetesen terhelik a hardverköltségek, az üzemeltetés és az alkalmazási rendszerek fejlesztési vagy beszerzési költségei.

A gazdaságos üzem valamilyen költségfüggvény értékének minimalizálását jelenti. A probléma általában ott van, hogy a költségfüggvény nem univerzális, hanem függ a számítógépes rendszer felhasználásának körülményeitől. Az operációs rendszerek általában két — sajnos egymásnak ellentmondó — költségfüggvénnyel szoktak dolgozni. Az egyik az erőforrások minél teljesebb kihasználtságára törekszik, a másik a rendszer válaszkészségét igyekszik optimalizálni. Az általános stratégián kívül megfogalmazhatók — nem ellentmondásmentes — konkrétabb célok is:

— az átlagos átfutási idő minimalizálása,

— a lassú válaszok számának minimalizálása,

— a hardver kihasználtságának maximalizálása,— maximális kihasználtság adott válaszidőkorlát esetén.

Elvileg az a jó — és vannak is ilyen operációs rendszerek —, ha egy nagy rendszernél az üzemeltető befolyásolhatja az erőforrás-kezelési stratégiát.

Az elmúlt húsz év alatt a hardver árával együtt a gazdaságosság megítélése is változott. A tendencia az, hogy a maximális kihasználtság háttérbe szorul a válaszkészséggel szemben. A valósidejű rendszereknél ez mindig is így volt, de újabban általánossá vált az időosztásos rendszereken való fejlesztési tevékenység is, ami szintén a válaszkészségre helyezi a hangsúlyt. A személyi számítógépek elterjedése ugyancsak ebbe az irányba hatott.

A lefoglalás munkafázisa

A klasszikus nagygépes feldolgozásban egy ütemező az indítható feladatok közül olyat próbált választani, amelyhez az erőforrások — memória, adatállományok és perifériák — adottak voltak. Ilyenkor a két esemény, az ütemezés és az erőforrás lekötése összekeveredett: a rendszer az F1 feladathoz például lefoglalta a memóriát (mert ezt meg tudta tenni), majd üzenetet küldött az operátornak, hogy tegye fel azt a mágnesszalagot, amelyen az A1 állomány van. Ezután az ütemező esetleg az F2 feladathoz is foglalt memóriát és egy lemezfájlt, majd üzent egy másik mágnesszalagért. E tevékenységek eredményeként még egyik feladat sem indult el, de már le van kötve a memória és két szalagos egység. A feladat csak akkor indul, ha az ütemező minden erőforrásigényét „összegyűjtögette". Ezt a módszert nevezhetnénk statikus lefoglalásnak is, hiszen az erőforrásokat a feladat indítása előtt lekötik. Ennek az a nagy hátránya, hogy az erőforrások akkor is függő — más feladat számára nem használható — állapotban lehetnek, amikor tulajdonképpen még senki nem használja őket. A statikus rendszerben egy feladat meghatározhatatlan ideig várakozhat az indítható munkák egymásutánjában, ameddig sorra kerül. Ugyanakkor, ha egy feladat végrehajtása elindult, erőforrás hiánya miatt nem szakad meg.

A statikus lefoglalást természetes módon szorította ki a dinamikus módszer, amelyben az indítandó feladatot minden előzetes erőforrásigény kielégítése nélkül választjuk ki. A szükséges erőforrásokat a feladat maga kérheti futása alatt az operációs rendszertől. Az ütemező csak azt teszi lehetővé, hogy a feladat részt vehet az erőforrásokért folyó versenyben. A felhasználónak persze előny a gyors indítás, ami természetesen még nem garantálja a gyors befejezést. A rendszer számára az éppen nem használt erőforrások szabaddá tétele és újra hasznosíthatósága a fontos, ami azonban függ a programozótól is. Ilyen stratégiával nagymértékben növekedhet a rendszer átbocsátóképessége, ha minden folyamat csak addig tart, ameddig arra okvetlenül szüksége van (például a programozó nem feledkezik meg a már feldolgozott állományok azonnali lezárásáról).

Dinamikus stratégiát követnek a modernebb kötegelt, valamint az időosztásos rendszerek, ahol a statikus módszer szóba sem jöhetne.

Holtpont

A holtpont tulajdonképpen az erőforrásokkal gazdálkodó, azok megosztott használatát lehetővé tevő algoritmusok mellékhatása, melynek kezelése egy fejlett operációs rendszerből sem hiányozhat. A folyamatok egy halmaza akkor kerül holtponthelyzetbe, ha mindegyikük egy olyan esemény bekövetkezésére vár, amit csak a halmaz valamelyik folyamata tud kiváltani. Az erőforrásokkal kapcsolatban az esemény értelemszerűen azok lekötése és elengedése lehet. A dinamikus lefoglalási stratégiának megfelelően egy folyamat a következő sorrendben kerül egy erőforrással kapcsolatba:

— Igénylés. Ha az igény nem teljesíthető azonnal — például az erőforrást egy másik folyamat használja —, az igénylő várakozásra kényszerül.

— Használat.

— Felszabadítás.

Az igénylés és a felszabadítás mindenképpen az operációs rendszer privilegizált rutinjainak hívását jelenti, a használat elvileg a felhasználó hatáskörében is megoldható, de általában az alsó szinten itt is a rendszerrutinok működnek.

A holtpontnak a kezelése érdekében kidolgoztak egy — a holtpont bekövetkezéséhez szükséges — feltételrendszert.

E szerint egy rendszer csak akkor kerülhet holtponthelyzetbe, ha a következő négy feltétel egyidejűleg teljesül:

— Kölcsönös kizárás. Legalább egy erőforrás nem osztható módon foglalt, azaz egyszerre csak egy folyamat használhatja. Ha egy másik folyamat is igényli, várnia kell az erőforrás felszabadulásáig.

— Várás közben leköt. Van olyan folyamat, amely leköt legalább egy erőforrást, miközben várakozik más folyamatok által használt erőforrás(ok)ra.

— Rablás nincs. Az erőforrások egy folyamattól erőszakkal nem vehetők el. A felszabadítást a folyamat mindig önkéntesen végzi.

— Ciklikus várakozás. Létezik a folyamatok egy olyan { p0, p1, . . . pn } halmaza, hogy p0 egy p1 által lefoglalt, p1 egy p2 által lefoglalt, . . . pn egy p0 által lefoglalt erőforrásra várakozik.

A feltételek nem teljesen függetlenek, a „ciklikus várakozás" például tartalmazza a „várás közben leköt" feltételt.

A holtpont alapvetően kétféleképpen kezelhető: bizonyos szabályok betartásával megelőzzük, vagy bekövetkezését felismerjük, majd a rendszert felélesztjük. Mint várható, a megelőzés az „olcsóbb", mi csak ezzel foglalkozunk.

A holtpont megelőzése

Arról kell gondoskodni, hogy a négy szükséges feltétel közül legalább egy ne teljesüljön. Ennek érdekében mindegyiket közelebbről is meg kell vizsgálni.

A kölcsönös kizárás feltételével nem sokra lehet jutni. Ez olyan eszközökre vonatkozik, amelyeket nem használhat két folyamat párhuzamosan (pl. egy sornyomtató). A párhuzamosan használható erőforrásokat viszont nem célszerű kölcsönösen kizáró módon kezelni, ezért ezek nem játszanak szerepet a holtpont kialakulásában. Ha például egy csak olvasható állományt tekintünk, azt akárhány folyamat használhatja egyidejűleg, tehát arra senkinek nem kell várnia.

Mivel azonban egyes erőforrások természetüknél fogva nem oszthatók, a kölcsönös kizárás nem küszöbölhető ki és így a holtpont nem előzhető meg.

Ha azt akarjuk, hogy a „várás közben leköt" feltétel soha ne teljesüljön, lehetővé kell tenni, hogy valahányszor egy folyamat erőforrást igényel, ne tartson másik erőforrást lekötve. Ez legegyszerűbben a statikus lefoglalási stratégiával biztosítható, azaz a folyamat csak minden erőforrás előzetes megszerzése után indulhat.

A másik megoldás, hogy minden folyamat csak akkor igényelhet, ha nincs lekötött erőforrása. Példaként tekintsünk egy folyamatot, amely cserélhető lemezről átmásol egy állományt a fix lemezre, ott rendezi azt, majd nyomtatón kiírja az eredményt. Az első megoldás szerint a folyamat indulás előtt igényli és lefoglalja a cserélhető lemezt, a fix lemezt és a nyomtatót. A nyomtatót végig leköti, pedig csak az utolsó fázisban van rá szüksége. A másik megoldás először csak a kétféle lemezt igényli és elvégzi a másolást, valamint a rendezést, majd felszabadítja a lemezeket. Ezután újra igényli a fix lemezt és a nyomtatót. Ha megkapja, kinyomtatja a rendezett állományt, minden erőforrását felszabadítja és a folyamat befejeződik.

Mindkét megoldásnak az a súlyos hátránya, hogy az erőforrások kihasználtsága igen rossz lehet, a lekötött erőforrások hosszú ideig kihasználatlanul állhatnak. A másik probléma, hogy nem korlátos annak a folyamatnak a várakozási ideje, amelyik több, gyakran használt erőforrást igényel, hiszen az összes igen ritkán válik egyszerre szabaddá. Ezt a jelenséget szokás „éheztetésnek"-nek is nevezni.

A harmadik feltétel az volt, hogy egy folyamattól nem lehet elrabolni a már megszerzett erőforrásait. Ha azt akarjuk, hogy ez ne teljesüljön, a következőképpen járhatunk el. Ha egy folyamat, amelyik már lekötött valamilyen erőforrást, újabb olyan erőforrást igényel, amit nem lehet azonnal hozzárendelni — azaz a folyamat várakozni kényszerül —, az összes eddig lekötött erőforrását el kell rabolni, erőszakkal fel kell szabadítani. Az elvettek felkerülnek azon erőforrások listájára, melyekre a folyamat várakozik. A folyamat csak akkor folytatódik, ha egyaránt visszaszerzi a régi és az új erőforrásait.

Ugyanezt a célt úgy is elérhetjük, hogy ha egy folyamat erőforrást igényel, először megnézzük, elérhető-e. Ha igen, lekötjük, ha nem, ellenőrizzük, hogy olyan folyamathoz kötött-e, amelyik további erőforrásra vár. Ha igen, elraboljuk az igényelt erőforrást a várakozó folyamattól és odaadjuk az igénylőnek. Ha az erőforrás nem hozzáférhető, vagy más okból várakozó folyamathoz tartozik, az igénylőnek várnia kell. Miközben várakozik, egyes erőforrásait elrabolhatják, de csak akkor, ha egy másik folyamat igényli azokat. A folyamat akkor indulhat újra, ha megkapja az újonnan igényelt és visszaszerzi a várakozás közben esetleg elvesztett erőforrásait.

Ez a módszer jól alkalmazható olyan erőforrásokra, melyek állapota könnyen menthető és visszaállítható (például a központi egység regiszterei vagy a memória). Általában nem használható sornyomtatóknál és mágnesszalagoknál. Ennek ellenére a THE rendszerben a sornyomtató elrabolható, az esetleg összekeveredett listákat az operátor szétválogatja.

Azért, hogy a ciklikus várakozás kiküszöbölhető legyen, az összes erőforrásra teljes rendezettséget kell létrehozni. Minden erőforrástípushoz egy egyedi egész számot rendelünk, melynek segítségével bármely két erőforrásról eldönthető, hogy a mi rendezési sorrendünk szerint melyik előzi meg a másikat.

Legyen Ei az i. erőforrás és a hozzá tartozó egész számot jelöljük F(Ei)-vel. A következő szabályok betartásával a holtpont megelőzhető.

A folyamatok csak növekvő sorrendben igényelhetnek erőforrásokat. Egy folyamat először akárhány példányt igényelhet mondjuk az Ei erőforrásból. Ezután a folyamat csak akkor jogosult egy Ej erőforrást igényelni, ha $F(Ej) > F(Ei)$. Ha azonos típusú erőforrásból több példányra van szükség, ezt egyetlen igényléskénti kell kezelni. Más szavakkal ez azt jelenti, hogy az Ej típusú erőforrást igénylő folyamattól egyszerűen megköveteljük minden olyan Ei erőforrásának felszabadítását, melyre $F(Ei) > F(Ej)$.

Könnyű belátni, hogy a fenti módszer nem engedi meg a ciklikus várakozást, mert a szigorú rendezettség miatt a várakozási kör nem tud bezárulni.

Megjegyezzük, hogy az F függvény értékeit az erőforrások természetes felhasználási sorrendjének megfelelően célszerű definiálni. A lemezek például általában előbb használatosak, mint a nyomtatók.

A fenti algoritmusok közös tulajdonsága, hogy az igénylések megszorításával célt érünk, vagyis valamelyik szükséges feltétel nem teljesül, de eközben esetleg lecsökken az eszközök kihasználtsága és a rendszer áteresztőképessége.

A holtpont úgy is elkerülhető, ha a folyamatok igényeiről további információt kérünk. E modellek közül az a legegyszerűbb és leghasznosabb, amelyik minden folyamattól megköveteli, hogy az egyes erőforrástípusokból legfeljebb hányat igényel. Ennek az információnak a birtokában olyan algoritmus konstruálható, amelylyel a holtpont biztosan elkerülhető.

A holtpontot elkerülő algoritmus dinamikusan vizsgálja az erőforrás kiosztásának az állapotváltozását és kiküszöböli a ciklikus várakozást. Az erőforrás-kiosztási állapot a még szabad és a már lekötött erőforrások számával, valamint a folyamatok maximális igényeivel jellemezhető. Egy állapot biztonságos, ha minden folyamat igényét valamilyen sorrendben ki lehet elégíteni a ciklikus várakozás bekövetkezésének veszélye nélkül. A folyamatok egy F1, F2 . . Fn sorozata akkor biztonságos, ha minden Fi-re a még igényelhető erőforrások vagy azonnal megkaphatók, vagy olyan Fj folyamatok birtokában vannak, melyekre $j < i$. Ebben az esetben ugyanis, ha Fi igényei nem elégíthetők ki azonnal, Fi várhat, ameddig az összes Fj befejeződik. Akkor lekötheti az igényeit, és erőforrásait szabaddá téve befejeződhet. Fi befejezése után $F(i+1)$ igényei teljesíthetők stb. Ha ilyen sorozat nem létezik, a rendszer állapota bizonytalan. Természetesen nem minden bizonytalan állapot holtpontállapot, csak annyi igaz, hogy bizonytalan állapotban a rendszer nem tudja megakadályozni, hogy a folyamatok olyan igényekkel léphessenek fel, ami holtponthoz vezet.

Bakos Tamás

Logika kontra algoritmus

Tisztelt Bakos Tamás!

A Szovjetunióban tanuló magyar diák vagyok. Jelenleg IBM PC-n dolgozom, de ismerek néhány nyelvmegvalósítást Spectrumon, Enterprise-on és Commodore 64-en is.

Jelenleg a számítógépes nyelvek fejlődését próbálom rövid elemzésnek alávetni, így az Ön cikke, ami a Mikroszámítógép Magazin 1988/11. számában jelent meg, egybeesik az érdeklődési területemmel.

Meglepett, hogy Ön a cikkében a Magyarországon még mindig nagyon erős irányvonalat támogatja, mely nem ismeri el, hogy a logikai nyelvek, például a PROLOG, illetve még a LISP is közelebb állnak az emberi gondolkodáshoz, mint az algoritmikus nyelvek. A cikkében megjelent táblázathoz szeretnék néhány megjegyzést fűzni:

— Azt hiszem, ha a PC-n elterjedt programozási nyelveket hasonlítjuk össze, nem lehet figyelmen kívül hagyni az „object-oriented" nyelveket, melyek a programozás új dimenzióját nyitják meg. Ilyenek például a Smalltalk, az Objective—C, a Mesa, melyek tudomásom szerint PC variánsban is léteznek.

A megvalósítás

— A PROLOG éppen az IBM PC-n jött ki egy olyan megoldásban (Turbo—PROLOG), amely az értelmező és a fordító tulajdonságait egyesíti magában.

Az olvashatóság

— A COBOL olvashatósága szerintem rossz, felesleges az a részekre osztás, melyet alkalmaz (lehet, hogy a PC variáns jobb, még nem találkoztam vele).

— A LISP olvashatósága megegyezik a FORTH-éval, mivel mindkettő ugyanolyan mélyről próbálja felépíteni a feladatait. Ami a LISP-ben a kezdőket zavarhatja — főleg más nyelvek után — az a sok zárójel (melyekben elkövetett hibát az értelmező azonnal kijelzi), illetve a logikai feltételek nagyon szigorúan strukturált felépítése. Ezenkívül még a listák kezelése lehet szokatlan más nyelvek után. Ezek viszont éppen azok a tulajdonságok, melyek a többi nyelvvel szembeni előnyeit alkotják.

A bővíthetőség

— Nem egészen értem, mit értünk pontosan ezen, hiszen itt nem a programozó képességei a döntőek, hanem az, hogy milyen alaprendszerből kiindulva próbálja megoldani a feladatot. (A grafika problémája teljesen másként néz ki egy rendszerben, ahol a programozó a gépi rutinokat kénytelen ráírni, mint ahol ezek a rutinok már be vannak építve.)

Strukturáltság

(Nincs megjegyzésem.)

A fejlesztői környezet

— Ugyanúgy, mint a bővíthetőség, nagyon függ az éppen felhasznált rendszertől.

— Az ún. „magas szintű" nyelvek már nagyon hasonlítanak egymáshoz, általában grafikai lehetőségeket nyújtanak, segítenek a program jobb strukturálásában, már „environment" alakban vannak, így a fordító, illetve az értelmező közötti különbség nem lényeges.

Az elsajátíthatóság

*— Helyette inkább az **emberközeliség** fogalmát vezetném be.*

— A magas szintű nyelvek elsajátítása között nem lehet különösebb szintet megjelölni, viszont közülük a COBOL (rossz strukturáltságával), illetve a PL/I (túlságosan bonyolult) nem emberközeliek.

— A LISP szerintem legalább a FORTH szintjére teendő. Sajnos emberközeliségét rontják az olvashatóság azon problémái, melyeket már kifejtettem.

— A PROLOG emberközelibb, mint a magas szintű nyelvek, ezért elsajátíthatósága is jobb. Megismerve a standard eljárásokat, szinte szabályok ismerete nélkül — természetesen az alapelv ismeretével — nagy bonyolultságú programokat lehet írni.

Lipták Gábor

Programozási fogások és melléfogások

A sorozat eddigi három részében a gazdaságtalan algoritmusok kiválasztására mutattam példákat. Legutóbb **A BASIC nyelvű programozás ábécéje** című könyvből idéztem a szerzők által javasolt algoritmust és programot a legnagyobb közös osztó kiszámítására, azután egy sokkal gyorsabb eljárást mutattam be ugyanerre. Akkor elmulasztottam megírni azt, hogy az említett könyv tankönyv, mégpedig **Dusza Árpád—Varga Antal: A BASIC nyelv c. tankönyv változatlan kiadása.** Erről a tankönyvről olvastam a „régi" Számítástechnika 1986. júliusi számának 15. oldalán egy könyvismertetést, melynek idézem néhány mondatát:

„Dusza Árpád és Varga Antal könyve ízig-vérig hagyományos tankönyv, mégpedig kiemelkedően a legjobb eddig a magyar piacon. (. . .) Ha ezt a könyvet kapta volna minden középiskola a HT géphez, akkor a magyar iskolaszámítógép-program nem félkarú óriás lenne a közepesen fejlett világ törpéi között."

Ezek után képzeljük el, hogy milyen programok jelennek meg a **nem kiemelkedően legjobb** tankönyvekben.

Az itt következő példákat olyan könyvekből választottam ki, melyek nem tankönyvek ugyan, de nem titkolt céljuk a programozási fogások oktatási célú bemutatása. A bemutatásra kerülő hibák nem súlyosak, veszélyük gyakoriságukban van, s főleg abban, hogy a programok szerzői műveiket követendő példaként állítják a tapasztalatlan olvasók elé, így terjesztve a rossz programozási stílust. Nem utolsósorban terjengősségükkel felesleges munkával terhelik azokat, akik kipróbálás céljából begépelik programjaikat.

Téglás Erzsébet Példák gazdasági jellegű feladatok megoldására BASIC nyelven című könyvét 1987-ben (!) adta ki az OMIKK. Íme az első feladat a könyv 7. oldaláról: „Írjunk programot, amely egy hétjegyű telefonszámról megállapítja, hogy budapesti-e!" A megoldás rövid ismertetéséből kiderül, hogy a budapesti telefonszámot arról ismerhetjük fel, hogy 1-es számjeggyel kezdődik. A 8. oldalon találhatjuk meg az *1/a listán* látható ötsoros programot. Most tekinsünk el attól az apróságtól, hogy a ciklusból csak a STOP és a RESTORE billentyűk egyidejű lenyomásával lehet kilépni, ezt a könyv szerzője is csak átmeneti megoldásnak szánta. A második sorban a THEN és a GOTO együttes használata felesleges, az 50-es sor END utasítása szintén. Ami nagyon csúnya, az a második sorban lévő GOTO utasítás, amely egy másik GOTO-ra ugrat. Érthetetlen, miért nem lehet rögtön a 10-es sorra ugrani? Próbáljuk meg!

További egyszerűsítésre ad módot, ha az IF utasítás feltételét megfordítjuk. A módosított program az *1/b listán* látható.

A következő három példa **Könözsi László—Gál István—Vasvári József Csupa játék C16-ra** című könyvéből van, mely a Műszaki Könyvkiadónál jelent meg

```
10 INPUT"TELEFONSZAM";SZ$
20 IF LEFT$(SZ$,1)<>"1" THEN GOTO 40
30 PRINT"BUDAPESTI"
40 GOTO 10
50 END
```

1/a lista

```
10 INPUT "TELEFONSZAM"; SZ$
20 IF LEFT$(SZ$,1)="1" THEN PRINT "BUDAP
ESTI"
40 GOTO 10
```

1/b lista

```
        ...
280 IFA$="I"THEN300
290 GOTO310
300 GOTO320
310 X9=0:Y9=0:GOTO340
320 ...
```

2/a lista

```
        ...
280 IF A$="I" THEN 320
310 X9=0 : Y9=0 : GOTO340
320 ...
```

2/b lista

```
    ...
280 IF A$<>"I" THEN X9=0 : Y9=0 : GOTO 3
40
320 ...
```

2/c lista

```
    ...
590 IFX1<1THENGOTO640
600 IFX1>25THENGOTO640
610 IFY1<1THENGOTO640
620 IFY1<40THENGOTO640
630 GOTO340
640 ...
```

3/a lista

1987-ben. A programrészleteket eredeti környezetükből kiemeltem, de lényegükön nem változtattam. Ezek is jól illusztrálják a GOTO meggondolatlan használatát.

A könyv 55. oldalán egy torpedójátékot mutatnak be. Idézek a bevezető szövegből: „A játékot dr. Kocsis András (. . .) könyve alapján egy szakkörön elemeztük, és úgy találtuk, hogy kezdő programozók számára több tanulságos szerkezeti elemet tartalmaz, tehát C16-os gépen is ajánlatos a programot bemutatni." Egy tanulságos szerkezeti elemet látunk a *2/a listán.* Javítását a *2/b listán* mutatom be, további egyszerűsítése — melyben az eredeti négy programsorból egy lesz — a *2/c listán* látható.

A *3/a listán* ugyanennek a programnak egy másik részlete látható. Ebből azt tanulhatjuk meg, hogy hogyan lehet egy (legfeljebb két) programsort ötre növelni a hatékonyság növelése nélkül. Nem tartom valószínűnek, hogy a szerzők a hosszú programsortól ijedtek meg, hiszen a könyv más helyén 88 karakternél hosszabb — több utasítást tartalmazó — programsort is láttam, amely utasítás rövidítés nélkül be sem írható. A helyes megoldás a *3/b listán* van. A *3/c listán* látható változat még jobb, de ismerni kell hozzá a logikai műveletek átalakításának szabályait.

Utolsó példánk a könyv 46. oldalán kezdődő lóversenyprogramból való. A *4/a listán* bemutatott szerkezet a programban összesen háromszor fordul elő, különböző adatokkal. Az idézett részben egy programhiba is van, melyet csak a feladat ismeretében fedezhetünk fel: a 370-es sorban 1001 helyett 1000 lenne helyes. Az END utasítás itt feleslegesebb, mint az első példában, mert ott legalább programzáró funkciót lehetne belemagyarázni, de itt még azt sem. Az egyszerűsítés módja a *4/b listán* látható.

Barna László

```
590 IF X1<1 OR X1>25 OR Y1<1 OR Y1>40 TH
EN 640
630 GOTO 340
640 ...
```

3/b lista

```
590 IF X1>0 AND X1<26 AND Y1>0 AND Y1<41
 THEN 340
640 ...
```

3/c lista

```
      ...
370 IF PENZ>1001 THEN 400
380 GOTO 420
390 END
400 ...
```

4/a lista

```
      ...
370 IF PENZ<1001 THEN 420
400 ...
```

4/b lista

Legyen-e sorellenőrző kód?

A Magazin ez évi 1. számának 39. oldalán S. E. részletesen kifejti, hogy a lapban közölt programlisták miért nem tartalmaznak sorellenőrző számokat. Bár abban egyetértek vele, hogy ezek közlése nehézségekbe ütközik, úgy gondolom, hogy a cikkben leírt érveinek nagy része nem helytálló.

Szükségtelen, hogy a kódellenőrző program a lapnak minden számában megjelenjen, és az sem szükségszerű, hogy minden géptípushoz legyen ilyen program. A hazai példaként felhozott Mikrovilág programozási rovatában 1988-ban öt különböző géptípusra közöltek programokat, de kódellenőrző programot csak C64-hez és C Plus/4-hez adtak. (A C16-ost itt nem tekintem külön géptípusnak, mivel programozás szempontjából nem tér el a C Plus/4-től.) Az említett Commodore gépek mindegyikére az év folyamán több mint húsz számban jelent meg egy vagy több program, de a kódellenőrző programokat összesen négy-négy alkalommal jelentették meg.

Külföldi példára is hivatkozhatok. Az NSZK-ban megjelenő „kevésgépes" 64'-er magazin rendszeresen közöl programokat C Plus/4-re is, de ellenőrző kód nélkül. A C64-esre készült Checksummer 64 sem jelenik meg minden számban, bár a lap terjedelme ezt megengedné. A V3 jelű változatot az 1985/8. számban közölték, az utána következő legalább öt számban egyszer sem, legfeljebb annyit, hogy megcímzett és a postai szállításhoz elegendő bélyeggel ellátott válaszboríték ellenében bárkinek megküldik a listáját a gépi kódú programok begépelését segítő MSE program listájával együtt.

S. E. állításával ellentétben a BASIC programok listáiban az ellenőrző kód kevés helyet foglal el, nem növeli számottevően a terjedelmét, és ez is könnyen kompenzálható.

A példaként felhozott 1988/10. számban közölt BASIC programokhoz — a Tandy CoCora írt egyetlen egysoros (sőt „egysorosok") program kivételével — jócskán elkelne az ellenőrző kód, ha másért nem, legalább azért, hogy az olvasó biztos legyen abban, nem ő gépelte be rosszul a szöveget, hanem valóban a közölt program működik hibásan. Arra a C Plus/4-re készült programra gondolok itt, melyet fáradságos munkával beírtam a gépbe, aztán ugyanannyi — ha nem több — munkát fordítottam az esetleges gépelési hibák keresésére. Végül kiderült, hogy a program nemcsak feleslegesen hosszú, de hibásan is működik.

Valós gond, hogy szükség lenne kódellenőrző programokra, ezeket el kellene készíteni. Valószínűnek tartom, hogy valamely külföldi lapból való átvétel nem úszható meg anyagi vagy jogi következmények nélkül, bár tudok néhány ilyen irányú kísérletről. Zsom Béla a Formázott listázás című, kétrészes írásában (Mikroszámítógép Magazin, 1987/9—10. szám) javasolja a Compute!'s Gazette PROOFREADER nevű kódellenőrző programjának átvételét, közölte a magyar szövegű programlistát is. Tudomásom szerint azóta semmi sem történt e témával kapcsolatban.

S. E. cikkében olvastam Gregorits József javaslatát, hogy vegyük át a Mikrovilág kódellenőrző programját. Ez még kevésbé járható út, mint az előbbi. Egyrészt azért, mert a Mikrovilág ragaszkodik a megjelent írásokkal kapcsolatos jogaihoz, másrészt azért, mert az említett program — tudomásom szerint — idegen szellemi termék jogtalan kisajátítása, olyasmi, amit szépirodalmi berkekben plágiumnak neveznek. Akkor inkább maradjunk Zsom Béla javaslatánál, mert a PROOFREADER négy különböző Commodore géptípus kódellenőrzését oldja meg egymaga.

Másik probléma — amit S. E. is említ — a listázás. A kódellenőrző számot alkalmazó lapoknak a programokat általában mágneses adathordozón — lemezen vagy magnókazettán — kell beküldeni, és azokat központilag egységesen listázzák. Ez nálunk sajnos megoldhatatlan, de arra kellene törekedni, hogy a szerkesztőség a külső munkatársaktól is csak jól használható programlistákat fogadjon el, s ha az ellenőrző kódokat bevezetnénk, lehetőleg a kód feltüntetésével együtt. A Commodore gépekhez Zsom Béla említett cikkében erre is láthatunk megoldást.

Nagyon örülnénk, ha a Mikroszámítógép Magazin is csatlakozna az ellenőrző kódot alkalmazó lapok táborához, annak ellenére, hogy sokgépes lap. Úgy gondolom, sok boszszúságtól és felesleges fáradságtól mentesülnének az olvasók.

B. L.

Rekurzió átalakítása iterációvá

A programozáselmélet irodalma elég sokat foglalkozik rekurzív algoritmusok nem rekurzívvá való átírásával. Legtöbbször azonban megállnak annál, hogy a jobb, illetve bal rekurziót ciklussá írják át. Jobb rekurzióról beszélünk, ha a rekurzív hívás az eljárás végén található, és bal rekurzióról, ha a hívás az eljárásba belépés után következik. Átírás szempontjából hasonló a helyzet, ha a rekurzív hívás az eljárás közepén található. A jobb rekurzió átírása különösen egyszerű; sima ciklussá szervezhető. A bal rekurzió már egy fokkal bonyolultabb: egy verem szervezése és kezelő utasításainak megírása után két különálló ciklussá esik szét a feladat. Az első ciklusban sorra letesszük (push) a megőrizni kívánt elemeket, majd a másik ciklusban szabadítjuk fel (pop) az elemeket. A szerzők általában bemutatják, hogy milyen feladatoknak van olyan tulajdonságuk, hogy ily módon átírhatók. Az utóbbi időben is találkozhatott a magyar olvasó ilyen tárgyú cikkel, illetve könnyvvel.

Ebben a cikkben azonban a fentieknél általánosabb rekurzív algoritmusok ciklussá átírását mutatjuk be. Célunk, hogy a gyakorlatban előforduló, de nem ilyen „szabályosan" viselkedő algoritmusokkal foglalkozzunk. Nehéz megfogalmazni az ilyen algoritmusok általános szerkezetét. Röviden: olyan algoritmusokra gondolunk, amelyekben a rekurzív hívások száma nem egy, hanem többszörös. A jobb megértés kedvéért egy konkrét algoritmusból kiindulva szeretnénk az általános megfogalmazási és átírási szabályig eljutni.

Nézzük példának a bináris fa preorder bejárásának rekurzív algoritmusát!

A bináris fa egy-egy szögpontjáról a következő adatokat tároljuk:

```
 ---------------------------------------------------
| BAL ( CIM ) | szögponti elem | JOBB (CIM )|  , vagyis
 ---------------------------------------------------
baloldali részfa                    jobboldali részfa
    cime                                 cime

 A terminálisok BAL(CIM)-e, illetve JOBB(CIM)-e 0.  /


Az egyszerü rekurziv algoritmus a következö:

PREORD eljárás (CIM)
   MÜVELET (CIM)
   Ha BAL (CIM) > 0 akkor
                    PREORD (BAL(CIM))
                    PREORD (JOBB(CIM))
   Elágazás vége
PREORD vége.
```

Egy szögpontból 0 vagy 2 élt tételezünk fel.

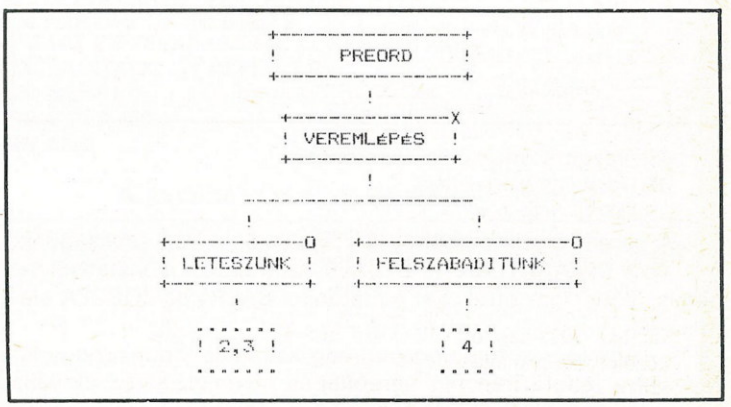
Természetesen most is adott az a gondolat, hogy rekurziónk ciklussá szervezését egy veremszerkezettel oldjuk meg. A preorder bejárás lényegéből kitűnik, hogy itt nincs meg az a szabályosság a veremlépésekben, mint az említett cikkek feladataiban. Ez a bejárás a „felülről lefelé", „balról jobbra" jelszavakkal jellemezhető. Vagyis először egy szögpontból kiindulva, annak legmélyebb szintű bal oldali terminálisáig haladunk (PREORD)-szinten: első hívások egymásutánja, VEREM-szinten push műveletek egymásutánja), majd az ezen élre kapcsolódó részfák bejárása következik, sorrendben alulról felfelé. (PREORD-szinten a két rekurzív eljárás hívása, illetve az azokból való visszatérések váltakozása; VEREM-szinten szintén a push és pop műveletek előre láthatatlan sorozata.)

A továbbiakban egységes magyar nyelvű elnevezéseket használunk: a push eljárást LETESZ eljárásnak, a pop eljárást FELSZABADÍT eljárásnak nevezzük.

Algoritmusunk váza a következő:

```
PREORD eljárás (CIM):
                .
                .
                .
        VEREMLÉPÉS ciklus
              Elágazás
                    LETESZÜNK esetén
                                LETESZ(CIM)
                                MÜVELET(CIM)
                    FELSZABADITUNK esetén
                                FELSZABADIT
              Elágazás vége
        VEREMLÉPÉS ciklus vége
PREORD eljárás vége.
```

Látjuk, hogy egy teljesen általános veremkezelésen alapuló eljárás szerkezetét adtuk meg. Ábrázoljuk szemléletesen Jackson-féle diagrammal:



A tevékenységek:
2 LETESZ (CIM)
3 MÜVELET (CIM)
4 FELSZABADÍT

A doboz jobb sarkában található X jel az iterációt, a 0 jel a szelekciót jelenti.

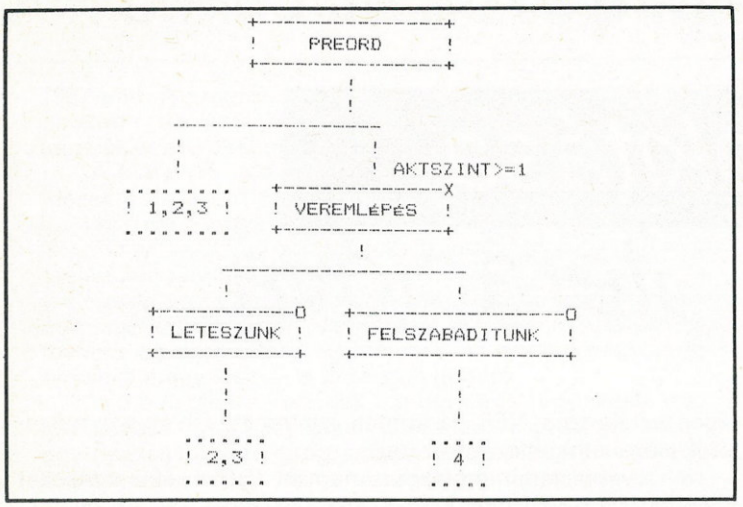
A következő lépésben meg kellene határozni a konstrukciós műveletek feltételeit. Első kérdés: meddig hajtsuk végre a veremlépés ciklust? Hogy determinisztikus legyen a szerkezet, alkalmazzuk az előreolvasás analógiájaként az első veremlépést, azaz a verem inicializálása (fájlnyitás) után egy előrehozott

LETESZ (CIM)
MÜVELET (CIM)

utasításlépést — egyszeres előreolvasást — alkalmazunk.

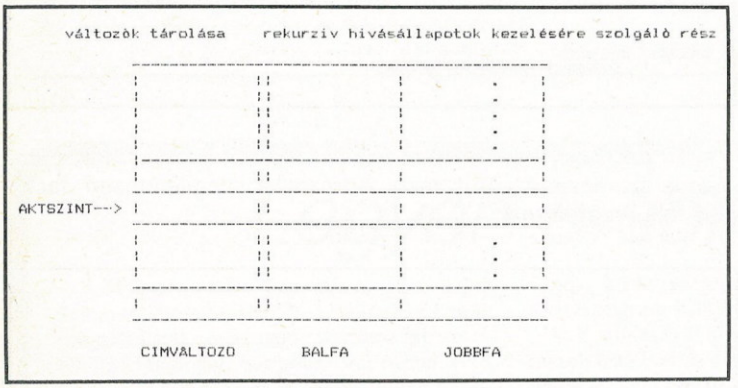
A verem aktuális szintjét mutatja egy AKTSZINT változó (az egyszerűség kedvéért globális változónak tekintsük): alkalmas a ciklus terminálására (van_még_rekord, illetve nem fájl_vég analógiájára).

Ekkor a pontosított programszerkezeti diagram az alábbi:



A tevékenységek:
1 INICIALIZÁL
2 LETESZ (CIM)
3 MŰVELET (CIM)
4 FELSZABADÍT

A következő lépés a szelekció kiértékelése. A verem egy-egy szintjén nem elég a változó(ka)t tárolni — esetünkben a szögpont címét —, hanem információra van szükségünk arról, hogy az azon a szinten lehetséges rekurzív hívásoknak megfelelő veremműveletek közül melyiket kell aktualizálnunk. A verem a következőképpen szervezhető jól:



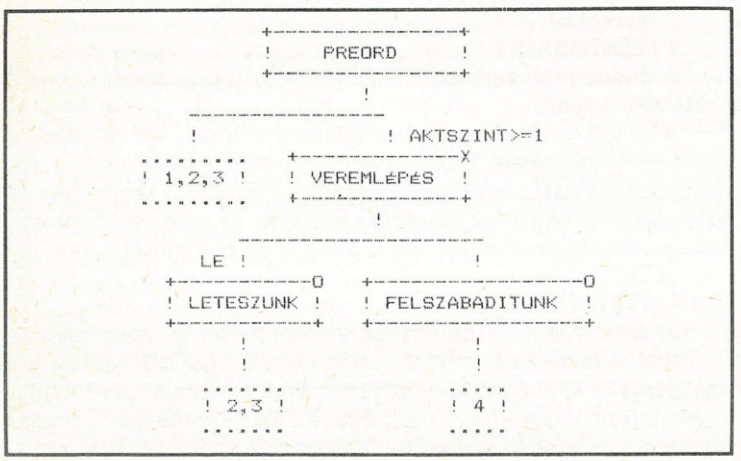
A javasolt szimbolikus konstansok:
BEJÁRATLAN = HAMIS
BEJÁRT = IGAZ

A verem értelmezéséhez a következőkre van szükségünk:

— a BEJÁRATLAN és BEJÁRT szimbolikus konstansokkal hamis, illetve igaz értékeket adhatunk a BALFA és JOBBFA elemeinek,

— egy logikai függvénnyel meg kell állapítanunk, hogy lefelé vagy felfelé irányuló veremlépés következik; az aktuális szinten bejártunk már minden részfát. Az eljárás neve legyen LE!

Ekkor a diagram az alábbi lesz:



Nézzük meg, hogyan valósulnak meg a fentiek pszeudokódban:

```
PREORD eljárás(CIM):
        INICIALIZÁL eljárás
        LETESZ eljárás(CIM)
        MŰVELET eljárás(CIM)
        VEREMLÉPÉS ciklus amig AKTSZINT >= 1
                ha LE(CIM) akkor
                        LETESZ eljárás(CIM)
                        MŰVELET eljárás(CIM)
                különben
                        FELSZABADIT eljárás
        VEREMLÉPÉS ciklus vége
PREORD eljárás vége.


INICIALIZÁL eljárás:
        SZINT:=1
        ciklus amig SZINT <= VEREMMÉRET
                VEREM(SZINT,BALFA):=VEREM(SZINT,JOBBFA):=BEJÁRATLAN
                SZINT:=SZINT + 1
        ciklus vége
        AKTSZINT:= 0
INICIALIZÁL eljárás vége.


LETESZ eljárás (CIM):
        VEREM(AKTSZINT,CIMVÁLTOZÓ):=CIM
        AKTSZINT:=AKTSZINT + 1
LETESZ eljárás vége.


FELSZABADIT eljárás
        VEREM(AKTSZINT,BALFA):=VEREM(AKTSZINT,JOBBFA):=BEJÁRATLAN
        AKTSZINT:=AKTSZINT - 1
FELSZABADIT vége.


LE logikai eljárás(CIM):
   Ha BAL(CIM)=0 akkor
                LE:=HAMIS
      különben
      ha VEREM(AKTSZINT,BALFA)=BEJÁRATLAN akkor
                                        LE:=IGAZ
                                        VEREM(AKTSZINT,BALFA):=BEJÁRT
                                        CIM:=BAL(CIM)
         különben
         ha VEREM(AKTSZINT,JOBBFA)=BEJÁRATLAN akkor
                                                LE:=IGAZ
                                                VEREM(AKTSZINT,JOBBFA):=BEJÁRT
                                                CIM:=JOBB(CIM)
            különben
            LE:=HAMIS
         Elágazás vége
      Elágazás vége
   Elágazás vége
LE eljárás vége.
```

Általánosan elmondhatjuk, hogy azok a rekurzív algoritmusok, amelyekben valamilyen predikátum teljesülésének függvényében többszörös meghívásra kerül sor, a fent említett módon átírhatók nem rekurzívvá:

— a fenti vezérlő modul egy az egyben, a veremkezelő eljárások szinte egy az egyben alkalmazhatók, csak

— a verem „állapotjelző" részét kell a feladathoz igazítottan definiálni, valamint

— feladatonként a LE logikai eljárást meg kell írni.

Látható, hogy a javasolt veremszerkezet lényegében kétdimenziós. Az első dimenziója a szokásos felhasználást jelenti a változók tárolására, a második dimenzió pedig az egy szinten való rekurzív hívások állapotait fejezi ki.

Belátható, bár itt nincs alkalmunk bizonyítani, hogy e kétdimenziós veremszerkezettel és a definiált algoritmusokkal olyan rekurzív sémák is átírhatók ciklussá, amelyek a tetszőleges számú programváltozót megengedő folyamatábra sémájába nem transzformálhatók.

Írásunk egyébként nem csupán elméleti eszmefuttatás. Valós feladatok megoldásaként alakult ki. Nevezetesen egy bonyolult rendszerprogramozási feladatot kellett megoldani olyan nyelvi környezetben (FORTRAN), amelyben nem volt rekurzió. A Jackson-féle programtervezési módszert alkalmazva kényszerült rá a fejlesztő a szerkezetek ilyen tiszta és általános kialakítására és a szerkezet által sugallt veremszerkezet megfogalmazására.

Bánné dr. Varga Gabriella

Rekurzió átalakítása iterációvá

A BASIC programok tárolása a memóriában

Az Enterprise is, mint más számítógépek, a BASIC programokat az értelmezés meggyorsítása érdekében átalakítva, kódolva tárolja a memóriájában.

Az Enterprise BASIC memóriája a nullás lap szegmensen a 4827 (=12DBh) címen kezdődik. Mivel ez a szegmens (248=OF8h) normális esetben mindig a nullás lapon van, ezért a BASIC programok ezen a címen mindig elérhetők. Az ennél nagyobb címek ezen a szegmensen csak a BASIC editornak vannak fenntartva, de természetesen kellő óvatossággal itt is elhelyezhetünk gépi kódú programokat. A programok a memóriában tárolva is megtartják soros felépítésüket, és a sorok sorszám szerinti sorrendben állnak. Egy memóriában tárolt BASIC-sor felépítése a következő:

1. bájt: a sor memóriabeli hossza bájtokban. E bájt címéhez hozzáadva ezt az értéket, megkapjuk a következő sor első bájtjának a címét.

2—3. bájt: a sor sorszáma bináris alakban — tehát elöl az alcsonyabb helyiértékű (LSB), utána a magasabb helyiértékű bájt (MSB) — van.

4. bájt: blokkmélység. Ennek a bájtnak az értéke mondja meg a listázó rutinnak, hogy az áttekinthető, tördelt listázási kép érdekében ezt a sort milyen mélyen kell megjeleníteni. Alapértéke 0, ekkor a sor közvetlenül a sorszám mögé íródik ki. Ha 1 az értéke, akkor két plusz szóköz szúródik a sorszám mögé, ha 2, akkor 4 szóköz stb.

5-n. bájt: a BASIC utasítások kódolva.

n+1. bájt: 00, azaz a sorvégjel (ERW).

Ezután közvetlenül a következő sor adatai jönnek, ugyanilyen struktúrában. Hogyha a soron következő BASIC-sor első bájtjának, azaz a hosszúságbájtnak az értéke nulla, ez a program végét jelzi. Egyébként a NEW utasítás sem csinál mást, mint az első programsor hosszúságbájtját törli. Az BASIC-értelmező tehát azt hiszi, hogy nincs a tárban érvényes program. Ennek ismeretében a NEW utasítással meggondolatlanul kitörölt programunkat „restaurálni" tudjuk.

Ezek után talán érthető a kézikönyvnek az a kitétele, hogy egy BASIC programsor maximálisan 250 karakterből állhat: ugyanis a sor térbeli képéhez hozzájön még plusz öt bájt. Így kaphatjuk meg az egy bájton maximálisan ábrázolható 255-ös értéket. A kézikönyv szintén írja, hogy a programsorszámok 1 és 9999 között lehetnek. Ez így is van, ha a BASIC-szerkesztővel dolgozunk. Ám utólag például POKE utasítással felülírhatjuk a sorszámokat 0 és 9999-nél nagyobb értékre. Ekkor is értelmes, futtatható programot kapunk, csak a sorszámok helyett értelmetlen karakterek íródnak ki, és a programsort nem lehet editálni, módosítani. Ezzel elérhetjük azt, hogy avatatlan kezek ne tudjanak belekontárkodni programjainkba, vagy mi ne tudjuk véletlenül kitörölni a sort.

A sor érdemi része nem ASCII kódok formájában tárolódik, hanem ún. BASIC kódokkal. Ezek kódolhatnak egyrészt karaktereket — például kettőspontot —, utalhatnak arra, hogy utánuk milyen típusú adat következik — történetesen sztring —, illetve lehetnek tokenek. A tokenek BASIC kulcsszavak helyett álló kódokat jelentenek. A mellékelt táblázatban megtalálhatók a BASIC kódok és a tokenek.

A különleges kódok rövidítéseinek magyarázata:

ERW (End of RoW): a sorvégjel. Ez jelenti a BASIC-értelmező számára azt, hogy az adott sor befejeződött.

FP (Floating Point): e kód után álló 6 bájtot az értelmező lebegőpontos számként próbálja értelmezni. A 6 bájt felépítése a következő: BY4 BY3 BY2 BY1 BY0 EXP, ahol a BY szimbólumú bájtok adják a mantisszát, az EXP pedig az exponenst. Az expontens 7. bitje az előjelbit: ha az 1 értékű, a szám negatív lesz. A maradék bitek értékéből 64-et (40h) levonva megkapjuk az exponens tényleges értékét. A mantisszát BCD számok alkotják. Itt négy biten ábrázolódik egy tizes számrendszerbeli számjegy. Az öt bájtból a legelső (BY4) adja a legalacsonyabbb helyiértékű számjegyeket, a legutolsó (BY0) pedig a legmagasabbakat. Az egy bájtban ábrázolt két számjegy közül az elsőnek van nagyobb helyi értéke. A tizedespont a legnagyobb értékű bájt (BY0) két digitje közé értendő. Eszerint a 35428 a következőképpen ábrázolódik:

$35428 = 3{,}5428 \times 10^4$ dec.: 0, 0, 8, 66, 53, 68

hexa: 00 00 08 42 35 44

INT (INTeger): e kód után 2 bájton ábrázolt egész szám következik. Az Enterprise csak a −9999...9999 intervallumba eső számokat kódolja így, de megérti az ilyen alakban megadott nagyobb számokat is. Emiatt ezt az alakot kis egész alaknak is hívják. A szám maga rendesen gépi ábrázolású, tehát a bináris értelmű bájtok LSB MSB sorrendben követik egymást.

RWN (RoW Number): e kód teljesen az INT mintájára felépülő egész számra utal, amit az értelmező BASIC sorszámnak fog fel. Tehát ilyen adat áll a GOTO, GOSUB stb. utasítások után.

STR (STRing): ezt a kódot egy karakterfüzér, sztring követi. Az ábrázolás ASCII kódokkal, a határoló idézőjelek nélkül lehetséges. A STR karaktert egy hosszbájt, majd a sztring karakterei követik.

NUV (NUmeric Variable): az ezt követő karaktersorozat egy BASIC változó azonosítója, illetve egy olyan tokenné át nem íródó BASIC kulcsszó neve, amely nem „$" karakterre végződik. Bővebbet a következő kódnál.

STV (STring Variable): e kódra következő karaktersorozat egy sztringváltozó azonosítója, illetve egy olyan tokenné át nem íródó BASIC kulcsszó neve, amely „$" karakterre végződik. Mindkét kódból 32 darab van. Ennek az az oka, hogy ebben a kódban vonták össze az azonosítók hosszát is. A kódok 0—4. bitjei adják meg a név hosszát. Emiatt lehet egy változó azonosítója maximum 31 karakteres.

TKN (ToKeN): a következő bájt egy BASIC utasítás tokenje. Több szóból álló utasításoknál csak az első szó tokenizálódik, a többi változóazonosító formájában kódolódik. Hasonlóképpen nem íródnak át tokenné a függvények, opciók nevei sem.

A programlistában lévő szóközök nem tárolódnak, azokat az értelmező listázóprogramja szúrja be értelem szerint. Egyébként a fent említett változásokat leszámítva a BASIC program térbeli szintaxisa megegyezik az editorba való begépeléskor előírtakkal.

Az elmondottak megvilágosításához álljon itt egy mintaprogram és annak memóriabeli képe decimális, valamint hexadecimális számokkal.

```
100 OPTION ANGLE DEGREES
110 LET ALPHA=23: LET MAX=35428
120 LET NAME$="ENTERPRISE"
130 FOR I=0 TO 100 STEP 2
140 LET P=PEEK(MAX+I): LET P$=STR$(P)
150 PRINTé102, AT ALPHA,3:P$:" ":
160 NEXT I
170 GOTO 120
180 END
```

A decimális memórialista soronként:

```
21,100,0,0,96,50,37,65,78,71,76,69,39,68,69,71,82,69,83,0
32,110,0,0,96,40,37,65,76,80,72,65,19,194,23,0,16,96,40,35,77,
        65,88,19,198,0,0,128,66,53,68,0
26,120,0,0,96,40,69,78,65,77,69,36,19,128,10,69,78,84,69,82,
        80,82,73,83,69,0
27,130,0,0,96,31,33,73,19,194,0,0,34,84,79,194,100,0,36,83,84,
        69,80,194,2,0,0
40,140,0,1,96,40,33,80,19,36,80,69,69,75,8,35,77,65,88,11,33,
        73,9,16,96,40,66,80,36,19,68,83,84,82,36,8,32,80,9,
        0
34,150,0,1,96,56,3,194,102,0,12,34,65,84,37,65,76,80,72,65,12,
        194,3,0,16,66,80,36,17,128,1,32,17,0
9,160,0,0,96,47,33,73,0
10,170,0,0,96,33,152,120,0,0
7,180,0,0,96,23,0
0,0,0
```

A hexadecimális lista:

```
15 64 00 00 60 32 25 41 4E 47 4C 45 27 44 45 47 52 45 45 53 00
20 6E 00 00 60 28 25 41 40 50 48 41 13 C2 17 00 10 60 28 23 4D
        41 58 13 C4 00 00 80 42 35 44 00
1A 78 00 00 60 28 45 4E 41 4D 45 24 13 80 0A 45 4E 54 45 52 50
        52 49 53 45 00
1B 82 00 00 60 1F 21 49 13 C2 00 00 22 54 4F C2 64 00 23 53 54
        45 50 C2 02 00 00
29 8C 00 01 60 28 21 50 13 24 50 45 45 4B 08 23 4D 41 58 0B 21
        49 09 10 60 28 42 50 24 13 44 53 54 52 24 08 21 50
        09 00
22 96 00 01 60 38 03 C2 66 00 0C 22 41 54 25 41 4C 50 48 41 0C
        C2 03 00 10 42 50 24 11 80 01 20 11 00
09 A0 00 00 60 2F 21 49 00
0A AA 00 00 60 21 A2 78 00 00
07 B4 00 00 60 17 00
00 00 00
```

A BASIC program után közvetlenül a változók területe kezdődik, vagyis ha a szerkesztőt megkerülve bővítjük programunkat, megtörténhet, hogy az értelmező nem ismervén a BASIC programterületnek új végcímét, a változókkal felülírja azt. Ezért ilyenkor célszerű néhány felesleges REM sorral helyet biztosítani az illegális programbővítésnek.

A BASIC KÓDOK ÉS TOKENEK TÁBLÁZATA

dec	hex	kód	token	dec	hex	kód	token
Ø	ØØ	<ERW>	ALLOCATE	53	35	<NUV>	PLOT
1	Ø1		ASK	54	36	<NUV>	POKE
2	Ø2	<ERW>	AUTO	55	37	<NUV>	SPOKE
3	Ø3	#	CALL	56	38	<NUV>	PRINT
4	Ø4	$	CAPTURE	57	39	<NUV>	PROGRAM
5	Ø5	%	CASE	58	3A	<NUV>	RANDOMIZE
6	Ø6	&	CAUSE	59	3B	<NUV>	READ
7	Ø7	'	CLEAR	6Ø	3C	<NUV>	REDIRECT
8	Ø8	(	CLOSE	61	3D	<NUV>	REM
9	Ø9	)	CODE	62	3E	<NUV>	RENUMBER
1Ø	ØA	*	CONTINUE	63	3F	<NUV>	RESTORE
11	ØB	+	COPY	64	4Ø	<STV>	RETRY
12	ØC	,	DATA	65	41	<STV>	RETURN
13	ØD	–	DEF*	66	42	<STV>	RUN
14	ØE	.	DEF*	67	43	<STV>	SAVE
15	ØF	/	DELETE	68	44	<STV>	SELECT
16	1Ø	:	DIM	69	45	<STV>	SET
17	11	;	DISPLAY	7Ø	46	<STV>	SOUND
18	12	<	DO	71	47	<STV>	START
19	13	=	CHAIN	72	48	<STV>	STOP
2Ø	14	>	EDIT	73	49	<STV>	INFO
21	15	<>	ELSE	74	4A	<STV>	STRING
22	16	<=	ELSE IF	75	4B	<STV>	TEXT
23	17	>=	END	76	4C	<STV>	TOGGLE
24	18	Ä	END DEF	77	4D	<STV>	TRACE
25	19	Ö	END HANDLER	78	4E	<STV>	TYPE
26	1A	Ü	END IF	79	4F	<STV>	VERIFY
27	1B		END SELECT	8Ø	5Ø	<STV>	WHEN
28	1C		END WHEN	81	51	<STV>	!
29	1D		ENVELOPE	82	52	<STV>	LLIST
3Ø	1E		EXIT	83	53	<STV>	LPRINT
31	1F		FOR	84	54	<STV>	EXT
32	2Ø	<NUV>	GOSUB	85	55	<STV>	GET
33	21	<NUV>	GOTO	86	56	<STV>	FLUSH
34	22	<NUV>	GRAPHICS	87	57	<STV>	LOOK
35	23	<NUV>	HANDLER	88	58	<STV>	PING
36	24	<NUV>	IMAGE	89	59	<STV>	DATE
37	25	<NUV>	IF*	9Ø	5A	<STV>	TIME
38	26	<NUV>	IF**	91	5B	<STV>	WAIT
39	27	<NUV>	INPUT	92	5C	<STV>	ON
4Ø	28	<NUV>	LET	93	5D	<STV>	
41	29	<NUV>	LINE	94	5E	<STV>	
42	2A	<NUV>	LIST	95	5F	<STV>	
43	2B	<NUV>	LOAD	96	6Ø	<TKN>	
44	2C	<NUV>	LOOP	:			
45	2D	<NUV>	MERGE	128	8Ø	<STR>	
46	2E	<NUV>	NEW	:			
47	2F	<NUV>	NEXT	162	A2	<RWN>	
48	3Ø	<NUV>	NUMERIC	:			
49	31	<NUV>	OPEN	194	C2	<INT>	
5Ø	32	<NUV>	OPTION	:			
51	33	<NUV>	OK	198	C6	<FP>	
52	34	<NUV>	OUT				

Megjegyzés a táblázathoz:

Mint látható, a táblázatban két DEF, illetve IF token szerepel. A -tal jelölt tokenek egyetlen BASIC sorban elhelyezkedő utasításra, míg a -tal jelöltek utasításblokkot kezdő parancsra vonatkoznak.

Az ON szimbólum az ON .. GOTO utasítástokenje, sohasem az opciókapcsoló ON szócskáé.

A LINE INPUT utasításban csak a LINE tokenizálódik, az INPUT nem; ha azonban az INPUT önmagában alkot utasítást, akkor átíródik tokenné.

A Mikroszámítógép Magazin 1988/8. számában Dósa Istvánnak a 26—27. oldalon megjelent cikkét (Memória, grafika, billentyűzet) szeretném kiegészíteni. A billentyűzetcsoportokat leíró 2. táblázatnak része még két billentyű és ezeknek a kódjai. Az x=9-es csoporthoz tartozó I, O, P billentyűk után tehát még két sort kell illeszteni:

x=9	
I	254
O	251
P	239
U	247
+	223

Ezenkívül a táblázat szemléletesebbé is tehető: kérdéses ugyanis, hogy milyen értéket kapunk akkor, ha az I és a P billentyűket egyszerre tartjuk lenyomva. Ez a probléma akkor fordul elő a leggyakrabban, ha a botkormány állását kívánjuk így megállapítani, és szükségünk van a ferde állás leolvasására is. A billentyűk ábrázolásának szokásosabb módja a billentyűmátrix megadása. Ilyenkor a 181-es portra kiírandó értékek a táblázat sorait, míg a beolvasáskor kapott bájt bitjei a mátrix oszlopait adják. A billentyűmátrix német tasztatúrájú gépeknél a következő:

a PORT—181-re írandó értékek	a portról kiolvasott bájt bitjei:							
	Ø.	1.	2.	3.	4.	5.	6.	7.
Ø:	N	>	B	C	V	X	Y	<LTSH>
1:	H	<LOCK>	G	D	F	S	A	<CTRL>
2:	U	Q	Z	R	T	E	W	<TAB>
3:	7	1	6	4	5	3	2	<ESC>
4:	<F4>	<F8>	<F3>	<F6>	<F5>	<F7>	<F2>	<F1>
5:	8		9	ß	Ø	'	<ERA>	
6:	J		K	Ö	L	Ä	#	
7:	<STOP>	<DWN>	<RGT>	<UP>	<HOLD>	<LFT>	<ENT>	<ALT>
8:	M	<DEL>	,	-	.	<RTSH>	<SP>	<INS>
9:	I		O	Ü	P	+		

Ha a billentyű éppen le van nyomva, az őt reprezentáló bit Ø értékű, ugyanakkor ha nincs lenyomva, a bit értéke 1. Eképpen könnyen kiszámítható, hogy ha a botkormány jobbra—fel állású és előzőleg a portra 7-et írtunk, a leolvasott bájt értéke 243. Ugyanígy I és P együttes nyomásakor, ha 9 lett a portra kiírva, a kapott érték 238 lesz.

A jelölésekről: RTSH jelöli a jobb oldali, LTSH a bal oldali SHIFT-billentyűt. A botkormány állásai: DWN — le, RGT — jobbra, UP — fel, LFT — balra. Az SP jelöli a betűköz-billentyűt, az ENT az ENTERT, az ERA az ERASE-t, a HOLD felel meg a német gépeken a PAUSE billentyűnek.

Vigyázni kell arra — különösen letiltott megszakításkor —, hogy a 181-es port felső 3 bitje más értelmű. A 6. és a 7. bitek állítják be a távvezérlő kimeneteket, az 5. bit pedig a magnetofon ki-bekapcsolását vezérli.

Racskó Tamás

Fedezzük fel együtt!

A beépített függvények számát tekintve nem lehet okunk panaszra. Az ábécé szerinti felsorolásuk a Felhasználói kézikönyv 153. oldalán kezdődik. Mi most más rendezőelvet követünk. A függvényekből csoportokat alakítunk ki az egyszerűbb eligazodás kedvéért. A csoportosításunkon kívül természetesen más szempontok alapján is lehet osztályozni a függvényeket: ismerem — nem ismerem, már alkalmaztam — még nem alkalmaztam stb. Ezek a megkülönböztetések is segítségünkre lehetnek a tanulásban.

Az Enterprise belső függvényeit a következőképpen csoportosíthatjuk:

A színfüggvények: BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE és az RGB.

Miután ezeket már ismerjük, a soknak tűnő függvények száma máris csökkent. Az RGB függvényről még lesz szó a 27. programban.

A matematikai függvények. Ezekből van a legtöbb, ezért további csoportokat hozunk létre. Egyparaméteresek: nevüket a 28. program DATA listájában találjuk meg. Kétparaméteresek: ezeket a 31. és 32. listán mutatjuk be. Átalakítást végzők: DEG, RAD és BIN. Tömbfüggvények: LBOUND, SIZE, UBOUND. Alkalmazásukra a 28. programban, a 475—478-as sorokban találunk példát. Beolvasó és kiíratási függvények (beolvasó: IN, JOY, kiíratási: TAB). A számítógép állapotát jelző függvények:

Függvényeink függvényében

INF, PEEK, SPEKK, FREE, EXTYPE, EXLINE, FBS. A gépi szubrutin hívása: USR.

A színekről már volt szó, ennélfogva nem ismeretlenek előttünk a színek megadására szolgáló függvények. Arról, hogy ezek a kulcsszavak tényleg függvények, meggyőződhetünk, ha kiadjuk például a PRINT BLUE parancsot. A függvény értéke a két szín kódja lesz.

A *27. listán* levő programmal azt a 256 színt mutatjuk be, amivel a számítógépen dolgozhatunk. Az RGB függvény paramétereivel az alapszínek — piros, zöld, kék — arányait adhatjuk meg. A megjelenítés helyének kiszámításához a MOD és az INT függvényeket alkalmazzuk.

```
  100 REM     --- 27. program ---
  110 LET L=0
  120 GRAPHICS HIRES 256
  130 FOR R=.125 TO 1 STEP .125
  140    FOR G=.125 TO 1 STEP .125
  150       FOR B=.25 TO 1 STEP .25
  160          PRINT "szinkod:";RGB(R,G,B)
  170          PRINT " R:";R,"   G:";G,"B:"
;B
  180          SET INK RGB(R,G,B)
  190          LET J=32*MOD(L,32):LET I=64*
INT(L/32)
  200          PLOT 120+J,600-I;120+J,600-I
-32
  210          LET L=L+1
  220          GET Q$
  230          IF Q$="" THEN 220
  240       NEXT
  250    NEXT
  260 NEXT
```

27. lista

```
100 REM    --- 28. program ---
110 GRAPHICS HIRES 4
120 PLOT 640,719;640,0
130 PLOT 0,360;1279,360
140 LET E=100
150 FOR I=0 TO 1279 STEP 4
160    LET X=(I-640)/E
170    LET Y=SIN(X)
180    LET J=360+E*Y
190    IF J<0 OR J>719 THEN 210
200    PLOT I,J
210 NEXT
```

28. lista

29. lista

```
100 REM    --- 29. program ---
110 GRAPHICS HIRES 4
120 PLOT 640,719;640,0
130 PLOT 0,360;1279,360
140 LET E=100:LET V=0
150 SET INK 3
160 WHEN EXCEPTION USE HIBA
170 · FOR I=0 TO 1279 STEP 4
180       LET X=(I-640)/E
190       LET Y=COT(X)
200       LET J=360+E*Y
210       PLOT I,J
220    NEXT
230 END WHEN
240 END
250 REM    --- hibakezelo ---
260 HANDLER HIBA
270    PRINT EXTYPE
280    CONTINUE
290 END HANDLER
```

30. lista

```
   100 REM      --- 30.program ---
   105 REM -- fuggvenyek abrazolasa --
   110 TEXT
   120 DIM Q$(30)
   130 NUMERIC K,U,V
   140 GOSUB 430
   150 LET E=100
   160 LET A=1:LET B=1279
   170 IF V=2 OR V=3 THEN LET A=640-E:LET
  B=640+E
   180 IF V=14 OR V=15 OR V=16 THEN LET A
 =641
   190 IF V=20 THEN LET A=640
   200 IF V=17 THEN LET A=0
   210 GRAPHICS HIRES 4
   220 GOSUB 610
   230 GOSUB 270
   240 END
   250 !
   260 REM    --- abrazolas ---
   270 FOR I=A TO B STEP 4
   280    LET X=(I-640)/E
   290    GOSUB 730
   300    LET J=360+E*Y
   310    LET T=3
   320    IF J<0 THEN LET T=0:LET J=0
   330    IF J>719 THEN LET T=0:LET J=719
   340    SET INK T
   350    IF (V>10 AND V<14) OR V=17 OR V=
5 THEN
   360       PLOT I,J
   370    ELSE
   380       PLOT I,J;
   390    END IF
   400 NEXT
   410 RETURN
   420 !
   430 REM    --- kivalasztas ---
   440 LET N=1
   450 DO
   460    READ Q$(N)
   470    LET N=N+1
   475    IF UBOUND(Q$)+1<N THEN
   476       PRINT "A DIM-ben a felso index
et novelni kell!"
   477       END
   478    END IF
   480 LOOP UNTIL Q$(N-1)="*"
   490 LET N=N-2:LET V=1
   500 PRINT AT 20,4:"kereses-joy  /  kiv
alasztas-space"
   510 PRINT AT 12,15:" ";V;CHR$(161):PRI
NT AT 12,20:":";Q$(V);"(x) "
   520 LET W=JOY(0)
   530 IF W=16 THEN RETURN
   540 IF W<>8 AND W<>4 THEN 520
   550 IF W=8 THEN LET V=V-1
   560 IF W=4 THEN LET V=V+1
   570 IF V=0 THEN LET V=N
   580 IF V=N+1 THEN LET V=1
   590 GOTO 510
   600 !
   610 REM  --- koordinata-rendszer ---
   620 PLOT 628,710,:PRINT #101:"^ Y"
   630 PLOT 1250,370,:PRINT #101:">"
   640 PLOT 1228,324,:PRINT #101:"X"
   650 PLOT 640,710;640,0
   660 PLOT 0,360;1279,360
   670 PLOT 640+E,368;640+E,352
   680 PLOT 632+E,320,:PRINT #101:"1"
   690 SET INK 2
   700 PLOT 30,700,:PRINT #101:Q$(V);"(X)
"
   710 RETURN
   720 !
   730 REM    --- fuggvenyek ---
   740 ON V GOSUB 760,770,780,790,800,810
,820,830,840,850,860,870,880,890,900,910
,920,930,940,950,960,970
   750 RETURN
   760 LET Y=ABS(X):RETURN
   770 LET Y=ACOS(X):RETURN
   780 LET Y=ASIN(X):RETURN
   790 LET Y=ATN(X):RETURN
   800 LET Y=CEIL(X):RETURN
   810 LET Y=COS(X):RETURN
   820 LET Y=COSH(X):RETURN
   830 LET Y=COT(X):RETURN
   840 LET Y=CSC(X):RETURN
   850 LET Y=EXP(X):RETURN
   860 LET Y=FP(X):RETURN
   870 LET Y=INT(X):RETURN
   880 LET Y=IP(X):RETURN
   890 LET Y=LOG(X):RETURN
   900 LET Y=LOG10(X):RETURN
   910 LET Y=LOG2(X):RETURN
   920 LET Y=SGN(X):RETURN
   930 LET Y=SIN(X):RETURN
   940 LET Y=SINH(X):RETURN
   950 LET Y=SQR(X):RETURN
   960 LET Y=TAN(X):RETURN
   970 LET Y=TANH(X):RETURN
   980 DATA abs,acos,asin,atn,ceil,cos,co
sh,cot,csc,exp,fp,int,ip,log,log10,log2,
sgn,sin,sinh,sqr,tan,tanh,*
```

31. lista

```
  100 REM     --- 31. program ---
  110 TEXT
  120 INPUT AT 3,2,PROMPT "a,b (b>0):":A
,B
  130 IF B=0 THEN
  140    PRINT "0-val nem lehet osztani!"
  150    END
  160 END IF
  170 PRINT AT 5,3:"MOD(A,B)=" MOD(A,B);
  180 IF B<0 THEN 200
  190 PRINT TAB(20) "A-B*INT(A/B)=" A-B*
INT(A/B)
  200 PRINT AT 7,3:"REM(A,B)=" REM(A,B);
  210 PRINT TAB(20) "A-B*IP(A/B)=" A-B*I
P(A/B)
  220 PRINT AT 9,3:"CEIL(A)=" CEIL(A);
  230 PRINT TAB(20) "-INT(-A)="-INT(-A)
  240 PRINT AT 11,3:"A/B=" A/B
  250 PRINT AT 13,3:"FP(A/B)=" FP(A/B);
  260 PRINT AT 15,3:"A/B-IP(A/B)" A/B-IP
(A/B)
  270 RANDOMIZE :LET N=RND(6)-3
  280 REM  let n=int(6*rnd)-3
  290 PRINT AT 17,3:"ROUND(A/B," N ")="
ROUND(A/B,N)
  300 PRINT AT 19,3:"INT(a/b*10^" N "+0.
5)/10^" N " =" INT(A/B*10^N+.5)/10^N
```

32. lista

```
  100 REM     --- 32. program ---
  110 TEXT
  120 INPUT AT 3,2,PROMPT "a,b:":A,B
  130 LET K=MIN(A,B):LET V=MAX(A,B)
  140 PRINT AT 5,4:"°a,b§ egesz szamai:"
CHR$(13)
  150 FOR I=CEIL(K) TO INT(V)
  160    PRINT I;
  170 NEXT
  180 PRINT
```

Az egyváltozós matematikai függvények vizsgálatához, megismeréséhez három programot készítettünk. A *28. listán* látható program elég rövid. Tekinthető „egy-két sorosnak" is. Sajnos, az Enterprise-tulajdonosok az „egy-két soros programok" írásában nem rúghatnak labdába, mert a gép csak áttekinthető programok írását segíti elő és teszi lehetővé. Lehet, hogy az utasítások száma jobban jellemzi a program készítőjének ötletességét?

Rövid programunk azonban jó néhány dologra hibát jelez. Most egy nagyon durva megoldást adunk, ami arra lesz jó, hogy megismerkedjünk a hibák kezelésével.

A *29. listáról* beírható programba beépítettünk egy hibafigyelést és az azt lekezelő HANDLER-blokkot. A WHEN-blokkba azt a programrészletet tettük, ahol várjuk a hiba megjelenését. A hiba bekövetkezésekor a végrehajtás a HANDLER-blokkban folytatódik. Itt most csak a hiba kódját íratjuk ki az EXTYPE függvénnyel. A CONTINUE hatására a hibát jelző sort követő soron folytatódik a program. Egy másik, a RETRY utasítással arra a sorra lehet visszatérni, amelyikben a hiba bekövetkezett.

A különböző függvények ábrázolásakor — ha a hibakódokat figyeljük — az is kiderül, hogy mi az, amit vizsgálnunk kell. Mire figyeljünk?

— Ismernünk kell a függvények értelmezési tartományát.

— Azokat a pontokat, amelyek nem esnek a képernyőre, ne próbáljuk meg kirajzolni!

Nézzük ezek után a matematikai függvények legismertebb és legnagyobb csoportjának bemutatását a *30. program* segítségével, amelyik több szubrutinból áll. Vizsgáljuk meg mindegyiket a végrehajtásuk sorrendjében!

Kiválasztás szubrutin. Egyszerű „menü", ahol először feltöltjük a Q$ vektort a függvények nevével. Itt látunk példát az egyik tömbfüggvény alkalmazására.

A JOY függvénnyel a botkormány helyzetét kérdezzük le, aminek a mozgatásával kikereshetjük az ábrázolandó függvényt. A választás a SPACE billentyű lenyomásával lehetséges. Ez lesz a beépített, azaz a 0-ás botkormány „tűz" billentyűje.

A DATA-ba újabb függvényeket is írhatunk a * karakter elé. A DIM utasítás harminc névnek ad helyet (a 0. indexet nem használjuk). Ha új függvényeket írunk be, meg kell határoznunk még az értelmezési tartományt és az ábrázolás módját is. Ez utóbbi azt jelenti, hogy pontokkal vagy vonalakkal rajzoljunk-e.

Koordináta-rendszer szubrutin. Bemenő paraméterként az E-t, azaz az egységnek választott képernyőpontok számát és a függvény sorszámát, a V-t kell megadni.

Ábrázolás szubrutin. Mielőtt belépnénk a szubrutinba, határozzuk meg az értelmezési tartományt, azaz A és B értékét.

A szakadásos függvények közül néhánynak az ábrázolását úgy oldottuk meg, hogy egy kis ügyeskedéssel kikerültük azokban a pontokban a függvény értékének kiszámítását, ahol nincs értelmezve. Ezért lesz A=1. Az SGN függvényben az SGN(0) értékének ábrázolása miatt A=0. X>0 esetén A=641, X>=0-nál A=640. A 11, 12, 13, 17-es és az 5-ös sorszámú függvények ábráját pontokból alakítottuk ki, a többinél összekötöttük a pontokat (350—380-as sorok).

Függvények szubrutin. Ezzel az ON GOSUB utasítást is bemutatjuk. Újabb függvény beírásakor az utasítássor sorszámát ide is be kell írni. A függvények közül az INT, IP, CEIL és az FP függvényekre szeretném felhívni a figyelmet. Alkalmazásukra bemutatunk néhány példát a *31. lista* programjával. Van ugyan néhány kétparaméteres függvény, amelyekkel ezek az átalakítások, számítások egyszerűbben megoldhatók, de úgy gondolom, érdemes összehasonlítani a megoldásokat.

Eddig még az RND függvénnyel nem foglalkoztunk. Nem is soroltuk be egyik csoportba sem. A két alak miatt nem döntöttem: RND(6) a hatnál kisebb nem negatív egész számok közül választ véletlenszerűen (270-es sor). Ugyanez történik a 280-as sorban is — a REM-ben mutatom be — az RND-vel. Az RND 0 és 1 közé eső véletlenszámot állít elő. Hármat azért vonunk ki, hogy negatív számokat is kapjunk.

Gyakorlásul próbáljuk megoldani a következő feladatot: írassuk ki az [A;B] egész számait! Másképpen fogalmazva: írassuk ki az A-nál nem kisebb és B-nél nem nagyobb egész számokat, ha A<B teljesedik! A megoldást hasonlítsuk össze a *32. listán* látható programmal!

Dusza Árpád

Mi a manó?

Magyarország=Budapest. Sok panaszos levelet kapunk vidéki Enterprise-tulajdonosoktól, hogy nem jutnak hozzá helybéli áruházakban programokhoz, szakirodalomhoz és kiegészítőkhöz. Sajnos saját országjáró körutainkon ugyanezt tapasztaltuk. Nincs jó hírünk sem, mivel részben a vidéki áruházak felkészületlensége (a szakértelem hiánya), részben az elmúlt évben kialakult Centrum—Novotrade viszály miatt sem, illetve csak nagyon mérsékelten rendelnek számítástechnikai eszközöket. A kialakult helyzettel ugyan egyáltalán nem értünk egyet, azonban csak azt tanácsolhatjuk az Enterprise-osoknak, hogyha biztosra akarnak menni, a kijelölt márkaáruházakat keressék fel. Ezek Budapesten vannak! Emiatt a vidékieknek sajnos utazniuk kell. A szakosodott üzletek a következők: az Úttörő, az Otthon, a Kispesti és a Novotrade 2C Áruház. Nagyon nem örülünk annak, hogy e remek és olcsó gép körül újabb vihar dúl.

*

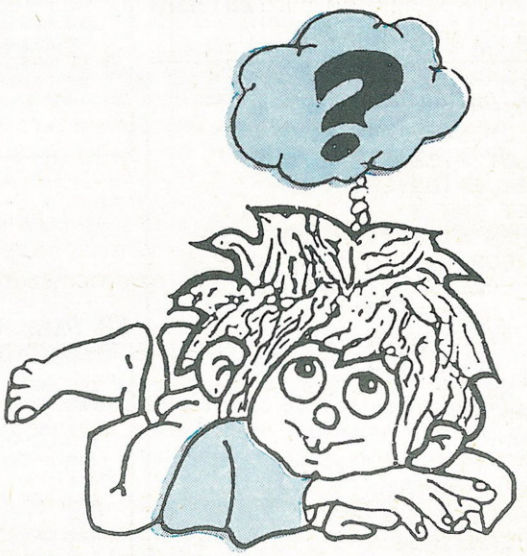
LORIGRAPH-kiterjesztés. Minden bizonnyal sok Enterprise-tulajdonosnak megvan a LORIGRAPH rajzolóprogram. Ennek egyik jó tulajdonsága, hogy egyszerűen átírható vele a gépünk karakterkészlete, és az szalagra is menthető. A következő rövid program a szalagra mentett karakterkészlet más, saját fejlesztésű programjainkban való felhasználását teszi lehetővé.

```
10 ALLOCATE 50
20 CODE KAR=HEX$ ("18,06")
30 CODE TAP=HEX$ ("05,54,41,50,45,3A")
40 CODE     =HEX$ ("3E,6A,11")
50 CODE     =WORD$ (TAP)
60 CODE      =HEX$  ("F7,01,3E,6A,01,80,B4,F7,
   06,3E,6A,3E,6A,F7,03,C9")
500 CALL USR(KAR,0)
```

A programot Fendrik László küldte be.

●

Monogram. Írjunk az ST FLAG rendszerváltozóba (1Ah=26) 42-t. Érdekes dolog történik. Az állapotsorban a ROM programok íróinak monogramjai jelennek meg.



Gaetsch Günterné rajza

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozónak

Hardver

rendelkeznie kell alapfokú áramköri hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretekre.

MIKROPROCESSZOROS RENDSZEREK TERVEZÉSE ÉS FEJLESZTÉSE

A cikksorozat eddigi részeiben összefoglaltuk azokat az alapvető hardverismereteket, amelyekre alapozva hozzáfoghatunk kisebb mikroprocesszoros rendszerek fejlesztéséhez.

A fejlesztést lépésről lépésre haladva mutatjuk be, többek között azért, hogy eloszlassuk a mikroprocesszoros rendszerek fejlesztésével kapcsolatos misztikus légkört, és egyszerű, jól alkalmazható gyakorlati tanácsokat adjunk a rendszereket készítőknek. Az *ábrán* összefoglaltuk a rendszerfejlesztés lépéseit.

A mikroprocesszoros rendszerek fejlesztésével kapcsolatos alapvető nehézség abban van, hogy egy ilyen feladat megoldása kettős tevékenységet igényel:

— meg kell tervezni és valósítani a működő áramkört, a hardvert,

— meg kell írni a működtető és az adott feladatot a kialakított hardverrel megvalósító programot, a szoftvert.

Mivel ez a két tevékenység szorosan kapcsolódik egymáshoz, ezért nélkülözhetetlen, hogy a fejlesztő bővében legyen mindkét témakör alapvető ismereteinek. Az ilyen rendszerek tervezői és fejlesztői már nem „szoftveres" vagy „hardveres" szakemberek, hanem mindkettőt — bár az egyiket általában jobban — fejlesztők.

A cikksorozat eddigi részeiben sem törekedtünk minden kérdés pontos, részletes tárgyalására, és a továbbiakban is azt a gyakorlati utat kívánjuk bemutatni, hogyan lesz egy feladat meghatározásából az azt megvalósító mikroprocesszoros rendszer.

Teljes általánosságban minden ilyen rendszer úgy tekinthető, hogy a rendszer bemenetére jutó jelekből valamilyen algoritmus — módszer — alapján előállítja a kimenő jeleket.

A továbbiakban először összefoglaljuk a rendszerfejlesztéssel kapcsolatos elméleti alapismereteket, majd egy konkrét, jól használható mikroprocesszoros rendszert is bemutatunk.

A mikroprocesszort tartalmazó rendszerek létrehozásakor a következő lépéseket kell elvégeznünk: a feladat megfogalmazását, az áramköri tervezést, a működtető program megtervezését, a kódolást, a hibakeresést, a programvizsgálatot (tesztelést) és a dokumentálást.

Egy működő berendezés megvalósításánál mindegyiknek nagy jelentősége van. A kódolás — ami a program megírását jelenti — csupán egy a többi között, és nem is mindig a leglényegesebb. Foglaljuk össze az egyes lépésekkel kapcsolatos legfontosabb ismereteket!

A FELADAT MEGFOGALMAZÁSA

A mikroprocesszorral megoldandó feladatok szükségessé teszik számos tény és körülmény pontos megszabását. Feltétlenül tisztázandó az alapvető kérdés, hogy valóban szükséges-e mikroprocesszoros rendszer alkalmazása.

A bemenetek definiálása

Először a rendszer bemeneteit kell definiálni. Fel kell tehát sorolni, hogy a feladat megoldása milyen bemeneteket igényel, és jellemezni is kell azokat.

— Milyen a bemenő jelek formája, időbeli viselkedése?

— Mi jelzi a bemenő jel aktív állapotát?

— Milyen hosszú ideig aktív a bemenő jel?

— Milyen gyakran — gyorsan — változik a bemenő jel, képes-e a program a bemenő jel változását követni?

— A bemenő jel kapcsolatban van-e más bemenő vagy kimenő jelekkel?

A kimenetek definiálása

Ehhez hasonló kérdéseket kell megválaszolni a rendszer kimeneteinek definiálásakor is.

A feldolgozó rész definiálása

Ez az a programrész, ami a bemenő jelekből és adatokból előállítja a kimenő jeleket és adatokat:

— Mi az alapvető algoritmus, ami a bemenetek alapján megadja a kimeneti jeleket?

— Milyen időkorlátok vannak? Milyen gyorsan kell a feldolgozást végrehajtani?

— Milyen memóriaterület-korlátozások vannak? Van-e valami korlát a programot tartalmazó memóriarészre vagy az adatokat tartalmazó memóriarészre vonatkozólag?

— Milyen szabványos — esetleg már meglévő — programrészleteket, szubrutinokat használhatunk fel?

— Milyen speciális esetek vannak, a program hogyan kezeli ezeket?

— Az eredményeknek milyen pontosaknak kell lenniük?

— Hogyan tudja a feldolgozás során fellépő hibákat a program kezelni?

Hibakezelés

Nagyon fontos a program működése közben fellépő hibák felderítése és

megfelelő módon való kezelése. A hibakezeléssel kapcsolatban a következő szempontokat kell figyelembe venni:

— Milyen hibák lehetségesek?

— Mely hibák a legvalószínűbbek? (Ha emberi beavatkozás is van, az emberi hibák a leggyakoribbak.)

— Milyen hibák maradhatnak rejtve egy ideig a rendszerben?

— A rendszer hogyan tudja a hibát a legkevesebb következménnyel elhárítani?

Az emberi tényező

Sok mikroprocesszor alapú rendszer emberi közreműködéssel dolgozik. Az ilyen rendszerek tervezésekor ezt a tényt fokozottan figyelembe kell venni, mivel alapvető minőségi jellemzőjük az ember-gép kapcsolat, az ún. „humán interfész" megfelelő kialakítása. A legfontosabb szempontok:

— Milyen beviteli módszer, beavatkozás a legtermészetesebb a kezelőnek?

— Milyen megjelenítési, visszajelzési módszert a legcélszerűbb alkalmazni (vizuális, akusztikus, mechanikus jelzések)?

— A kezelő hogyan kap jelzést a rendszer hibás működéséről?

— Melyek a kezelő által elkövetett legvalószínűbb hibák?

— A kezelő beavatkozására adott rendszerválaszok egyértelműek?

— A kijelzési képet a kezelő könnyen olvashatja és érthető?

Ezek tisztázása után létrejön a tervezés alapjául szolgáló rendszerterv, amiből kiindulva elkezdhető a rendszer hardverjének a megtervezése.

AZ ÁRAMKÖRI TERVEZÉS

A megválaszolandó kérdések:

— Mennyi RAM/EPROM memória szükséges?

— Hány be/kimeneti áramkört kell használni?

— Milyen egyéb kiegészítő áramkörök szükségesek az adott feladat megoldásához?

A hardver elkészítése meglehetősen eszköz- és munkaigényes.

Lépései:

— a hardver megtervezése, a kapcsolási rajz elkészítése,

— a rajz alapján a nyomtatott áramköri lemez elkészítése,

— az alkatrészek beültetése a lemezre,

— a kész hardver bemérése, az egyes funkciók helyességének ellenőrzése.

A fejlesztési munkának ettől a részétől megkímél az, hogy sok olyan komplett, mikroprocesszoros rendszert tartalmazó bemért kártya kapható, ami önálló mikroszámítógépként használható. Ezek RAM és EPROM memóriát, valamint jó néhány be- és kimenetet tartalmaznak, ami szerint eligazodhatunk a megfelelő kiválasztásában. Valószínűleg a feladatunkhoz pontosan illő kártyát nem fogunk találni. A kiválasztáskor ezért kompromisszumokat kell kötni, és előfordulhat, hogy néhány áramköri részt esetleg külön kell megépítenünk. Egy ilyen — talán legegyszerűbb — kártya leírása a magazin 1987/8. számában található: 8 be- és 8 kimenete van, maximum 64 kbájt EPROM használható, és RAM-ot nem tartalmaz. Egy másik, univerzálisan használható kártya tervét a sorozat egy későbbi részében közöljük.

A PROGRAM TERVEZÉSE

Ebben a lépésben a részletesen megfogalmazott feladatot programként realizáljuk. Bár a cikksorozat címe a hardverrel, az áramkörökkel kapcsolatos ismeretek bemutatására utal, most a teljesség kedvéért a programtervezésről, mint a rendszertervezés fontos lépéséről kissé részletesebben is szólnunk kell. A programtervezésben a következő módszereket használják: a folyamatábra-készítést, a moduláris programozást, a strukturált programozást és a felülről lefelé tervezést.

Az első kettő az assembler programozásnál használható jól, az utóbbiakat pedig magas szintű programozási nyelveken (Pascal, C) alkalmazhatják.

Mielőtt ezeket a módszereket összefoglalnák, soroljuk fel a programtervezésnél használt legfontosabb alapelveket:

— Célszerű kis lépésekben haladni és fejleszteni.

— A nagyobb részfeladatokat kis, egymástól logikailag elkülönülő modulokból kell felépíteni, mivel ezek önállóan tesztelhetők, és esetleges megváltoztatásuk nem igényli a teljes rendszer újratervezését.

— A feladatnak megfelelő programvezérlés lehetőleg egymás utáni, egyenként végrehajtható részekből álljon, és ne ide-oda ugrásokból, áttekinthetetlen programhurkokból, ciklusokból. Ez a hibakeresést is megkönnyíti.

— Célszerű minél több grafikus, vizuális leírást alkalmazni (a folyamatábra módszernek ez az előnye).

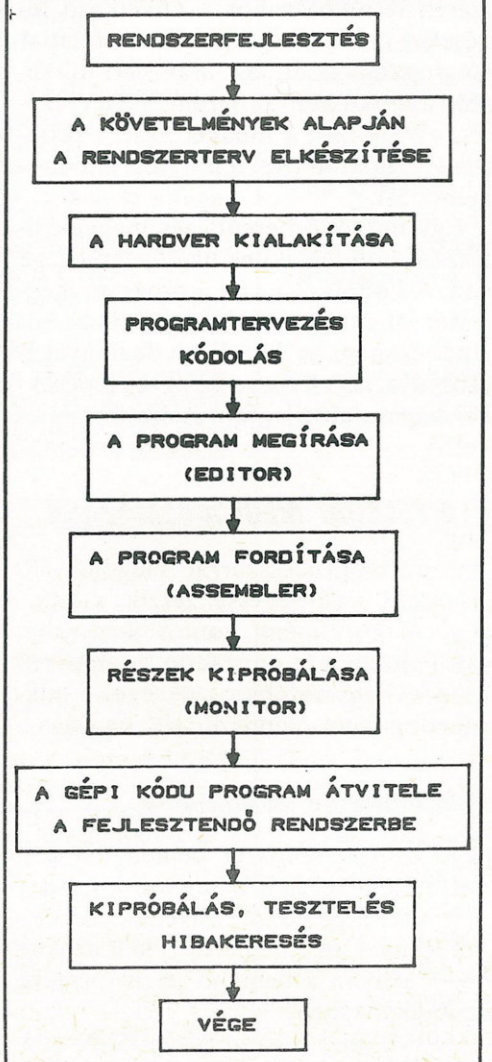
— A megfogalmazások, fogalmak egyszerűek és világosak legyenek.

— Olyan algoritmusokat kell felhasználni, amelyek ismertek és többszörösen kipróbáltak.

— A programtervezéskor figyelembe kell venni azt, hogy melyek azok a tényezők, paraméterek, amelyek megváltozhatnak.

— A kódolást csak a programtervezés teljes befejezése után szabad elkezdeni.

A RENDSZERFEJLESZTÉS BLOKKVÁZLATA



A folyamatábra módszer

Nagyon jól alkalmazható módszer. Legkiválóbb tulajdonsága az, hogy a programot grafikusan lehet megtervezni, és ezáltal jól láthatók az összefüggések és a programrészek közötti kapcsolatok.

Előnyei ezenkívül még, hogy

— szabványos szimbólumokkal lehet dolgozni, így széles körben használható,

— a folyamatábrák megérthetők programozási ismeretek nélkül is,

— a folyamatábrával mind a teljes program, mind az egyes részei leírhatók.

Hátrányai:

— A folyamatábrát bonyolult meg-

tervezni, megrajzolni, változtatni — újra kell rajzolni —, kivéve a legegyszerűbb eseteket.

— Nincs egyszerű módszer a folyamatábra tesztelésére.

— Annak eldöntése, hogy részletes legyen-e a folyamatábra, nem egyszerű. Ha túl tömör, nem adja vissza a program minden fontos részletét, ha pedig túl részletes, akkor nagyon nehéz a program áttekintése.

— A folyamatábra csak a program szervezését mutatja, nem mutatja az adatszervezést és a be/kimeneti modulok felépítését.

— A folyamatábrában könnyű a nyilakat ide-oda húzni, aminek az a következménye, hogy a kódoláskor sok hurkot, ide-oda ugrásokat kell megvalósítani.

Moduláris programozás

Azt a módszert, amikor a teljes programot részekre — alprogramokra vagy más néven modulokra — osztjuk, moduláris programozásnak nevezzük. Ennek a módszernek az alapvető problémája az, hogy a tervező hogyan ossza fel a programot modulokra és miként egyesítse a modulokat komplett, működő programmá. E módszernek az előnyei:

— Az egyes modulokat könnyű megírni, tesztelni, a bennük lévő hibákat megkeresni.

— Egy modul valószínűleg sok helyen is felhasználható — más programokban is —, mert többnyire elég általános. Célszerű egy modulkönyvtárat létrehozni.

— Több programozó is könnyen együttműködhet a különböző modulok megírásánál.

— Egy-egy hiba általában egy modulhoz kapcsolódik csak, így könnyebb kijavítani.

Hátrányai:

— A modulok egymáshoz illesztése gondokkal járhat, különösen ha különböző programozók írták a modulokat.

— A modulok nagyon pontos és részletes dokumentációt igényelnek, mert a program más részeire is hatással vannak.

— A modulok tesztelése és a hibakeresés elég nehézkes lehet, mivel az egyes modulok olyan bemenő adatokat igényelnek, amelyeket más modulok állítanak elő. Ezért az ilyen modulok kipróbálásához külön olyan programrészeket kell írni, melyek az adott modulnak mintaadatokat generálnak. Ez járulékos — és valójában nem szükséges — programozást igényel.

— A modulok gyakran több futásidőt és memóriahelyet igényelnek, mivel hogy a modulokon belül gyakran ugyanazok a részfeladatok megismétlődnek.

Strukturált programozás

Ennél a módszernél — mint ahogy a neve is utal rá — a programokat a programozó alapstruktúrákból építi fel. Nevezzünk egy utasítást vagy utasítássorozatot S-nek. A jelölés felhasználásával a következő alapstruktúrák adódnak:

1. Szekvenciális struktúra: S1, S2, S3 . . . struktúrák egy programban sorban egymás után hajtódnak végre.

2. Feltételes struktúra: ha C igaz, akkor S1, különben S2. Itt C egy feltételt jelent.

3. Hurokstruktúra: amíg C igaz, csináld S-t. Itt C egy feltételt jelent.

4. Indexstruktúra: I esetén SØ, S1, . . ., Sn, ahol I az index és 0,1, . . ., n értékű lehet. Ha I = 0, akkor SØ, ha I = 1, akkor S1, . . . ha I = n, akkor Sn hajtódik végre.

Matematikailag bebizonyítható, hogy e négy struktúra felhasználásával tetszőleges program felépíthető. Ennek a módszernek az előnyei, hogy

— a műveletek sorrendjét könnyű követni, ezért könnyű a hibát megkeresni és tesztelni,

— az így felépített program már önmagát dokumentálja, tehát könnyen olvasható,

— ezt a módszert használva megnövekszik a programozói teljesítmény.

Hátrányai:

— Az ilyen programírást csak néhány magas szintű programozói nyelv (például PL/M, PL/Z, Pascal) támogatja utasításaival. Az assemblerben programozónak külön át kell alakítania egy ilyen módon megírt programot, de előnyösen használható dokumentációs célokra.

— A strukturált program általában hosszabb futásidejű és több memóriát foglal el, mint a nem strukturált változat.

— A korlátozott számú struktúrával alkalmanként nagyon nehéz és bonyolult a feladatot megvalósítani. Az, hogy bármely program ezekkel a struktúrákkal megírható, igaz, de nem biztos, hogy a megírt program hatékony, kis tárigényű és gyors lesz!

— Az egymásba ágyazott „ha . . . akkor . . . különben" struktúrák esetleg nehezen olvashatók.

Végül is a strukturált programozást akkor ésszerű használni, ha programunk elég nagy — több mint 1000 utasítás —, a futásidő és a memóriafelhasználás nem kritikus követelmény.

A memóriák olcsóbbá válása, valamint a nagyobb sebességű mikroprocesszorok megjelenése és elterjedése mind a strukturált programozás szélesebb körű felhasználásának irányába hat, mert jól karbantartható, technologizálható programozást tesz lehetővé.

Felülről lefelé programozás

Amikor a struktúrákból vagy modulokból a programot össze akarjuk állítani, felmerül az a kérdés, hogyan teszteljük az egyes részeket és miképpen, mennyire hatnak egymásra. A szokásos eljárás az, hogy az egyes részeket külön-külön, sokszor fáradságos munkával teszteljük, majd a részek egymáshoz illesztése (az integrálás) a végére marad. Az a módszer, ami a tényleges programkörnyezetben lehetővé teszi a tesztelést és hibakeresést, a felülről lefelé programozás.

A módszer lényege, hogy először a teljes, átfogó programot írjuk meg, a benne szereplő egyes feladatokat, eljárásokat csak kijelöljük — egy nevet adunk nekik —, és csupán definiáljuk a név mögött álló feladatot. A következő lépésben ezeket a feladatokat — ha szükséges — ismét alapfeladatokra bontjuk, ismét definiáljuk azokat és így tovább. Mikor a lebontásban eljutottunk az elemi szinthez, onnan kezdjük a program megírását. Innen kezdődik a felfelé való programírás és a minden szinten elvégezhető tesztelés.

A módszer alkalmazásával olyan programunk lehet, melynek hatékonysága bizonytalan. Másrészről hátránya az is, hogy a meglévő programokat, rutinokat nem biztos, hogy könnyen felhasználhatjuk, ugyanakkor a módszer nem eredményez általánosan felhasználható modulokat sem.

A fentiekben röviden összefoglaltuk az assembler programok fejlesztésénél felhasználható programtervezési módszereket. Nem adtunk egy kizárólagosan jó, „csak ezt használd" módszert, mert ha ilyen lenne, nem kellett volna a kérdéssel ennyit foglalkozni. Az egyes módszereket azért mutattuk be, hogy a feladat megoldójának ötleteket, módszereket adjunk, melyekből a feladat megoldása közben meríthet. Az igazi programtervezésnek világosnak, (mások által is!) érthetőnek, áttekinthetőnek kell lennie, amit csak sok és pontos munkával, gyakorlással érhetünk el.

Dr. Kónya László

BÖRZE

TUTTI

ELECTROCOOP KISSZÖVETKEZET

— IBM PC kompatibilis gépek
— HARDVERTELEPÍTÉS SZERVIZ ÉS GARANCIA
— SZOFTVERES TÁMOGATÁS
— RÖVID HATÁRIDŐ

Cím: Bp., Üllői út 81. 1091
Tel.: 334-354

ENET

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

COBRA

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1097 Budapest, IX. Illatos út 7. Telefon: 476-160/388

SZÁMLAKÉSZÍTÉSTŐL A KÖNYVELÉSIG

— COBRA—CONTO ügyviteli programrendszer moduljai
— Számlakészítő, Számlanyilvántartó, Bér- és jövedelemszámfejtő, Főkönyvi könyvelő, Anyagnyilvántartó, Általános könyvelési

PROGRAMOK!
Árak: 19 000—49 000 Ft-ig
KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

ADATÁTVITEL!

MIKROPO BUDAPEST

Komplex számítástechnikai szolgáltatás!

Iroda:
VI., Nagymező u. 51.
Tel.: 325-768

ASY

Telefon: 415-166

Kereskedelmi és Szoftveriroda
1061 Bp., Liszt F. tér 10.
Telex: 22-4378

— ASY—16 SZUPERMIKRO számítógép (12 terminál, VME busz, UNIX)
— ASY—16/XT IBM kompatíbilis mikroszámítógép
— ASY—16/AT IBM kompatíbilis mikroszámítógép
— különleges igényeket kielégítő billentyűzetek
— szoftverfejlesztések

PERIFÉRIA

Elektronikai Fejlesztő és Szolgáltató Kisszövetkezet
Bp. VII., Peterdy u. 30.
Telefon: 213-588

AJÁNLATA:

— P—XT: 140 E Ft-tól + áfa
— P—AT: 200 E Ft-tól + áfa
igény szerinti konfigurációk
— FX—1000 Printer: 75 E Ft + áfa
— 40 MB-os WINCHESTER:
86 E Ft + áfa

procontrol

pc®

Kisszövetkezet megnyitotta

COMPUTER SZAKÜZLETÉT

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

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

BROTHER AX 15 típusú,
margarétafejes,
elektronikus írógép

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

MIGERT

ECONORG

SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

1146 Bp., AJTÓSI DÜRER SOR 10.
Levélcím:
1393 Pf.: 319.
Telefon: 421-974
Telex: 22-6544

SZAKÜZLETEK:

1. Bp. VI., Szinyei Merse u. 1.
   Tel.: 127-628. Tx.: 22-6684
2. 9022 Győr, Lukács S. u. 18.
   Tel.: 96-14808. Tx.: 02-4679
3. 3100 Salgótarján, Ady Endre u. 1.
   Tel.: 32-10971. Tx.: 22-9380

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

Alkatrészek, kártyák

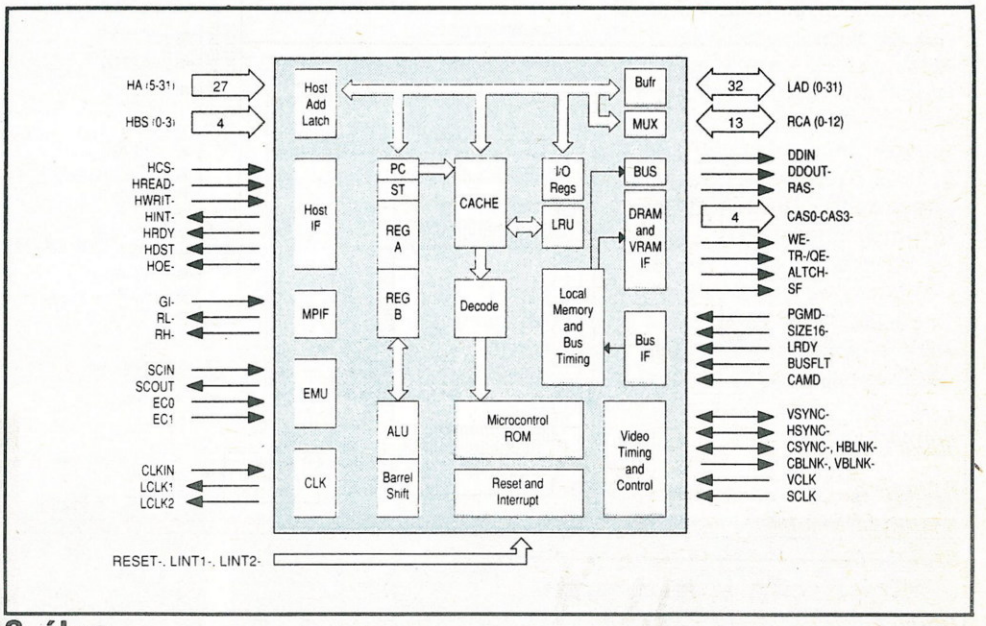
Texas IC-k

A Texas cég az egyik legnagyobb — ha nem a legnagyobb — név a félvezetőgyártásban. Fejlesztései a tranzisztorgyártás megkezdésétől, az első IC-ken keresztül a mai grafikus szuperalkatrészekig, mindig a technológiai élvonalat képviselték. Ezenfelül a japánok előretöréséig a világ legnagyobb félvezetőgyártója volt, jelenleg is az első három között van.

A borító belső oldalán látható annak a TI 34010 grafikus processzornak a tömbvázlata, amellyel nagy felbontású, nagy sebességű, sok színt előállító kártyákat terveztek, amelyek ma már széles körben elterjedtek. Miből áll és milyen is ez a kártya?

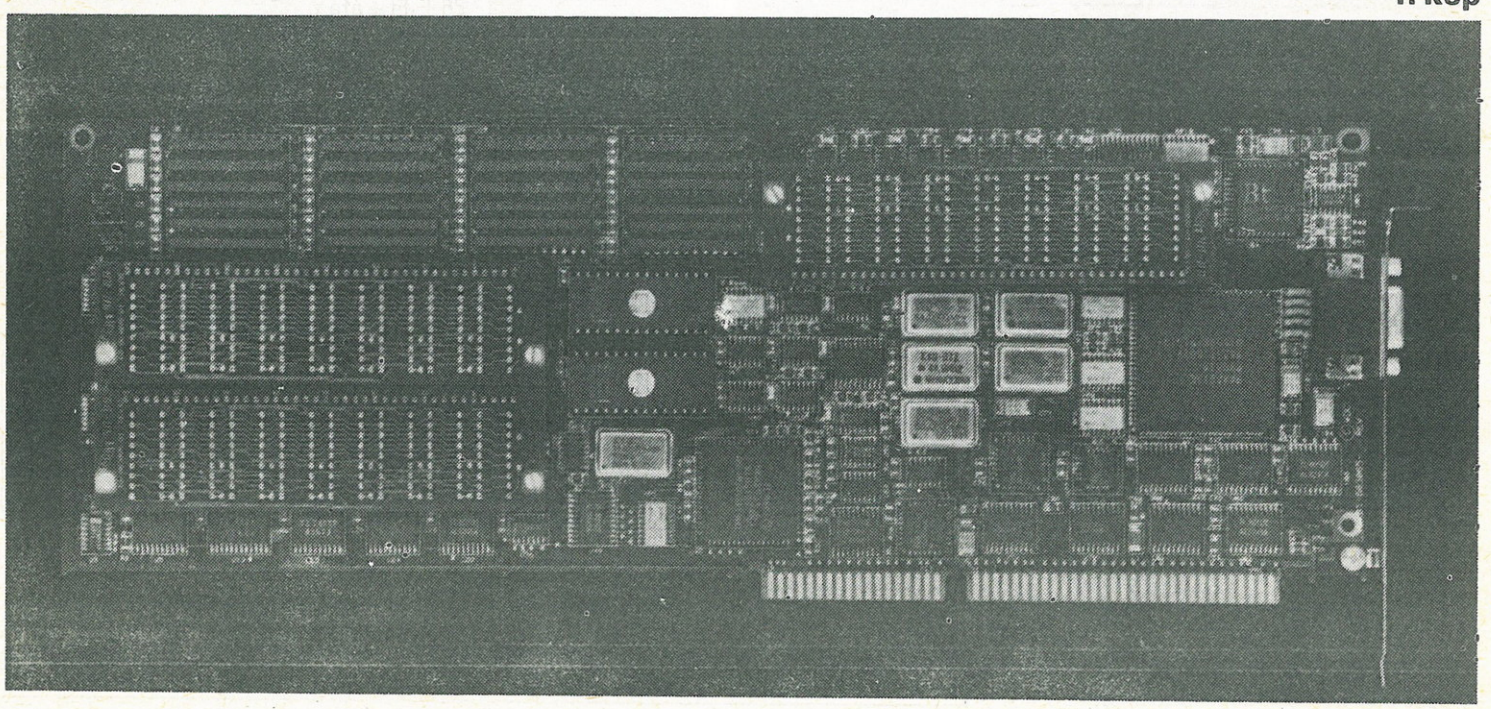
Egy 32 bites CMOS mikroprocesszor van benne, grafikus rendszerekhez optimalizálva. Különlegessége — az alkalmazási célnak megfelelően tervezve — a bitenként címezhető, 1 Gbit címterület, a képernyővezérlő. A cég most továbblépett. Kiegészítette a TMS340 családot. Ellátta kiegészítőkkel, mint például az *1.* és *2. ábrán (hátsó borító)* látható TMS34061 típusú videovezérlővel és a TMS34070 típusú színes palettával. Az előbbi bármilyen általános célú mikroprocesszorral együttműködhet, és kiküszöböli azt, hogy a szöveget és grafikát csak külön lehessen kezelni. Irányítani képes bármilyen monitort és 64 k/256 k video-, valamint ugyanekkora dinamikus tárat. A színes paletta 4096 szín közül választhatja ki a képernyőre vitt 16-ot, és 1024 × 768 pont megcímzését teszi lehetővé.

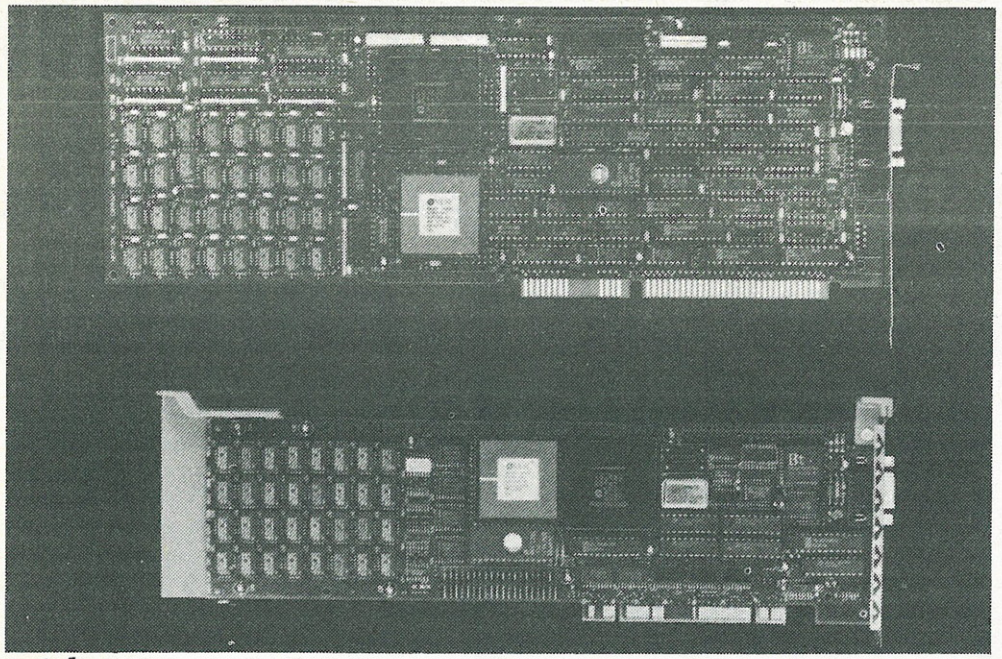
Ezenkívül tökéletesítették a grafikus processzorokat, aminek eredményeképpen egy új processzort hoztak létre, a TMS34020-as típust *(3. ábra)*. Ez mintegy ötvenszer gyorsabb, mint az elődje, azzal felülről kompatibilis, multiprocesszoros üzemmódban dolgozhat más hasonló vagy egyéb processzorokkal. A TMS34020-ast kiegészítették a TMS34082 típusú lebegőpontos processzorral *(4. ábra)*, amivel 40 Mf lop érhető el (ez tízszerese a PC-koprocesszorokénak!). A TMS44C251 CMOS videotár *(5. ábra)* 1 Mbit kapacitású, 4 bites szervezésű. A három új alkatrésszel és néhány kiegészítővel készült 1024 × 1024, egyenként 8 bites színes képpontú rendszer blokkvázlatát, mint az egyik legfontosabb alkalmazási példáét mutatja a *6. ábra*.



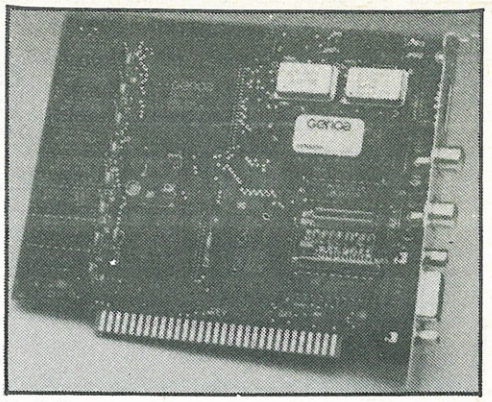
3. ábra

1. kép





2. kép

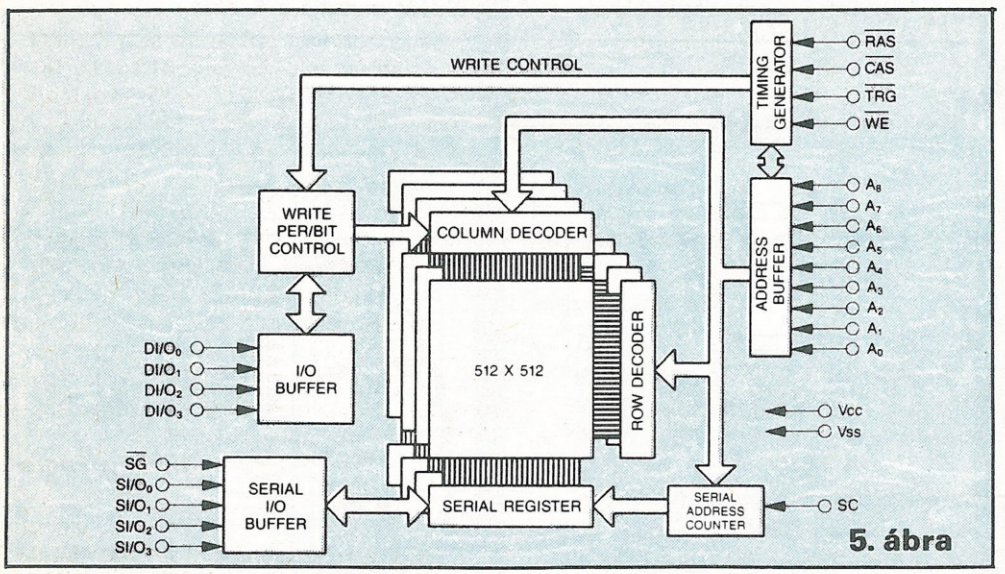


3. kép

dise — a Western Digital leányvállalata — PVGA1A IC-re egy Verticom MX elnevezésű sorozatot dolgozott ki *(1. kép)*. Az AT kompatibilis kártyák 1024 × 768 darab 16 vagy 256 színű pontot kezelhetnek. A kártyán AutoCAD meghajtó is található a CAD-alkalmazások megkönnyítésére. A kártyákkal elsősorban a jelenleg mintegy százezer darabos CAD-piacot szeretnék meghódítani, de remélik, hogy a jövőre várható mintegy három és félszeres eladások is főleg az övék lesznek. (A piaci adatok a Dataquest nevű piackutató cégtől származnak.)

A még nagyobb igényű — és pénzű — vásárlóknak készült a cég Verticom HX sorozatának két új tagja *(2. kép)*. A kártyák AT, illetve MCA kompatibilisek, maximálisan 2 Mbájt RAM tárat tartalmaznak, és hogy mit képesek nyújtani, az a hátsó borítón látható. A TwinFocus nevű eszközzel az AutoCAD-használók, ahogy ezt a kép is mutatja, egyidejűleg két interaktív, nagyítható ablakkal dolgozhatnak.

A cég Paradise leányvállalata a kispénzű felhasználók számára adott új megoldást a borító belső oldalán látható VGA Plus 16 kártyával. Ennek felbontása megegyezik az IBM VGA előírással (erről sorozatunk egy korábbi részében már volt szó). Különleges tulajdonsága, hogy megvizsgálja a számítógéprendszer egyéb részeinek konfigurációját, és amit talál, ahhoz alkalmazkodik.

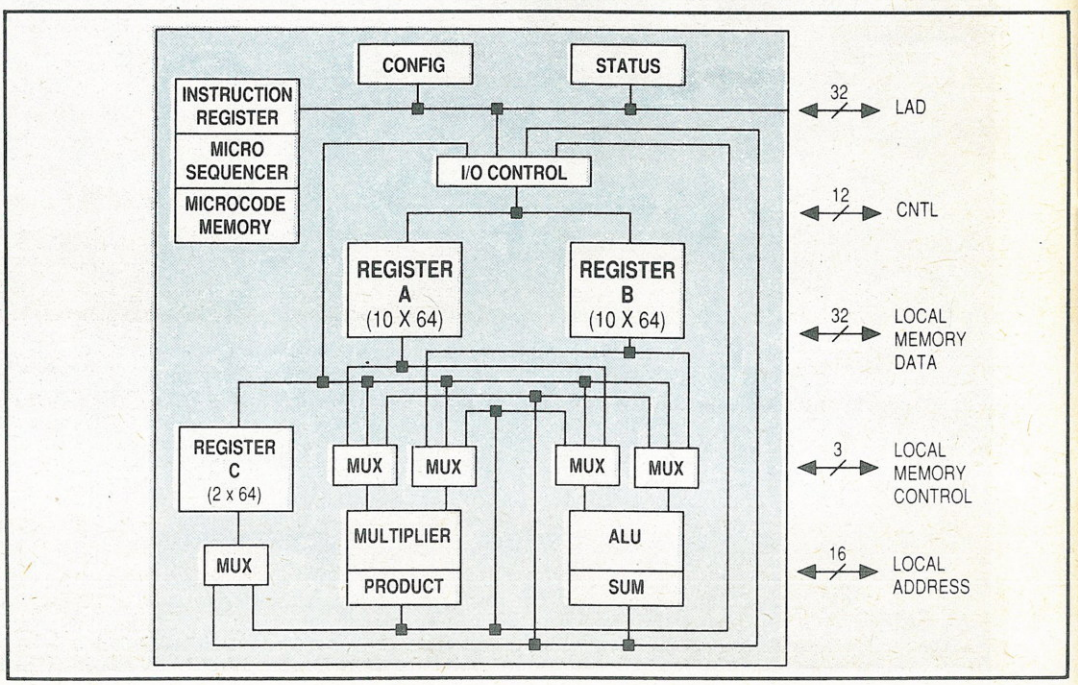


5. ábra

Motorola nyomtató-vezérlők

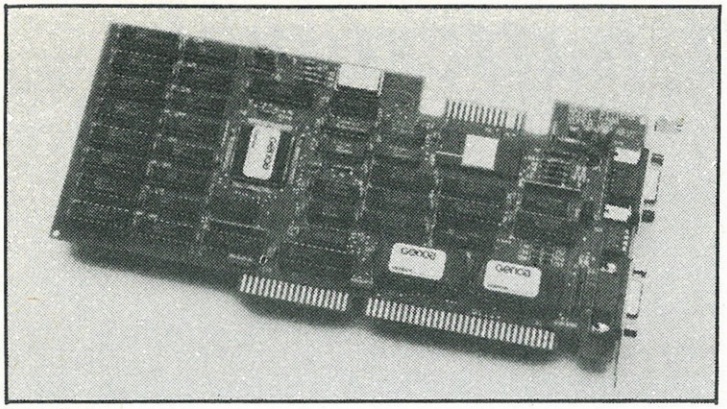
A cég kétmikronos, kétrétegű fém- és egymikronos háromrétegű CMOS technológiájának alkalmazásával két lézernyomtató-vezérlő IC-t hozott létre. Az LPC1 mikroprocesszor-illesztőt, dinamikus RAM-kontrollert és DMA videoillesztőt, az ALPC1 ezenfelül a rasztermű-veletekhez kiegészítőt is tartalmaz *(7. ábra)*. A mikroprocesszor a 680 × 0 sorozat lehet, a tár 8 Mbájtig 256 kbites vagy 1 Mbites tokokból állhat. A cég együttműködve a Personal Computer Products Inc. nevű vállalattal, komplett lézervezérlő kártyákat *(8. ábra)* is gyárt. Ezek képesek emulálni a HP LaserJet II-t, az Epson FX—80-at, a HPGL 7475A-t (plotter) és kompatibilisek a PostScript lapleíró nyelvvel.

4. ábra

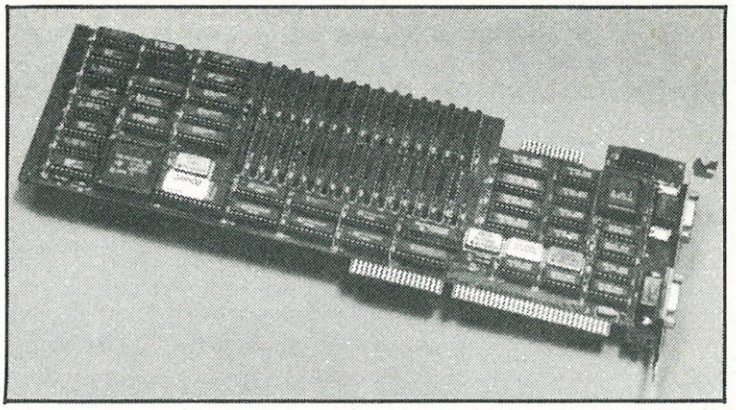


Western Digital kártyák

A cég a már említett TI34010-re és a sorozat egy korábbi cikkében tárgyalt Para-

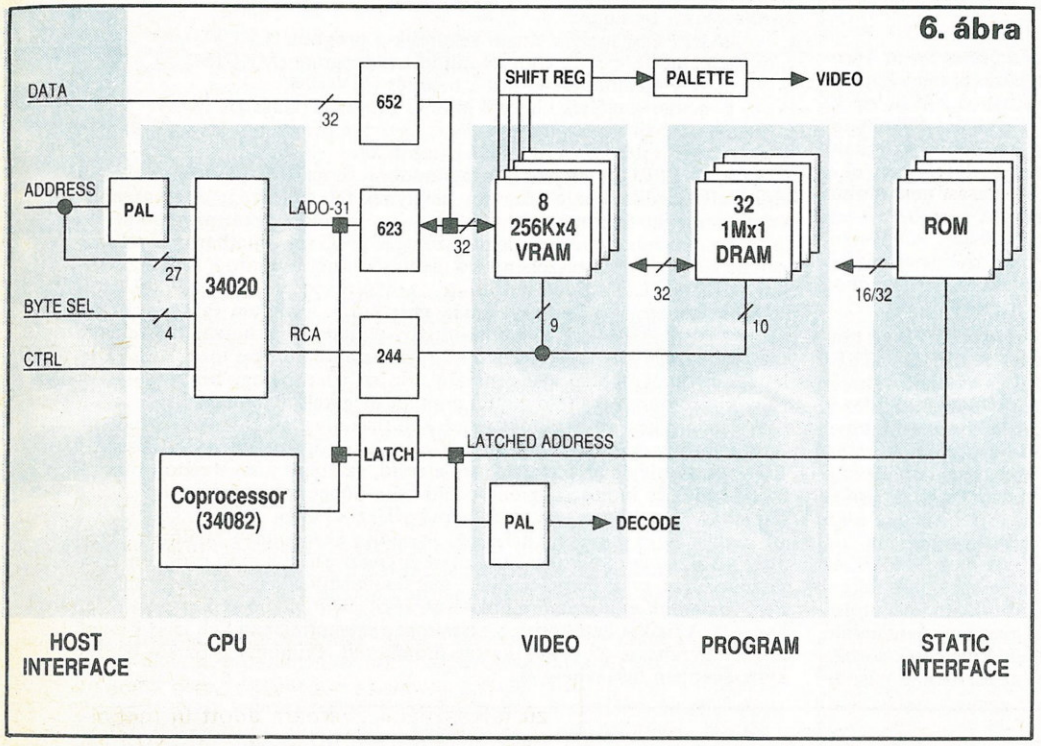


4. kép



5. kép

6. ábra

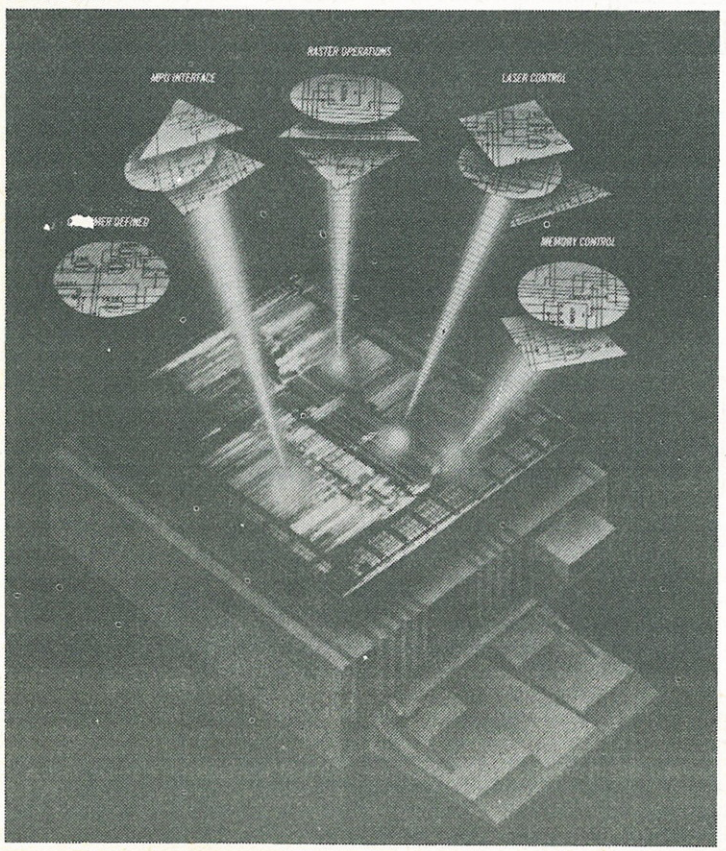


GENOA kártyák

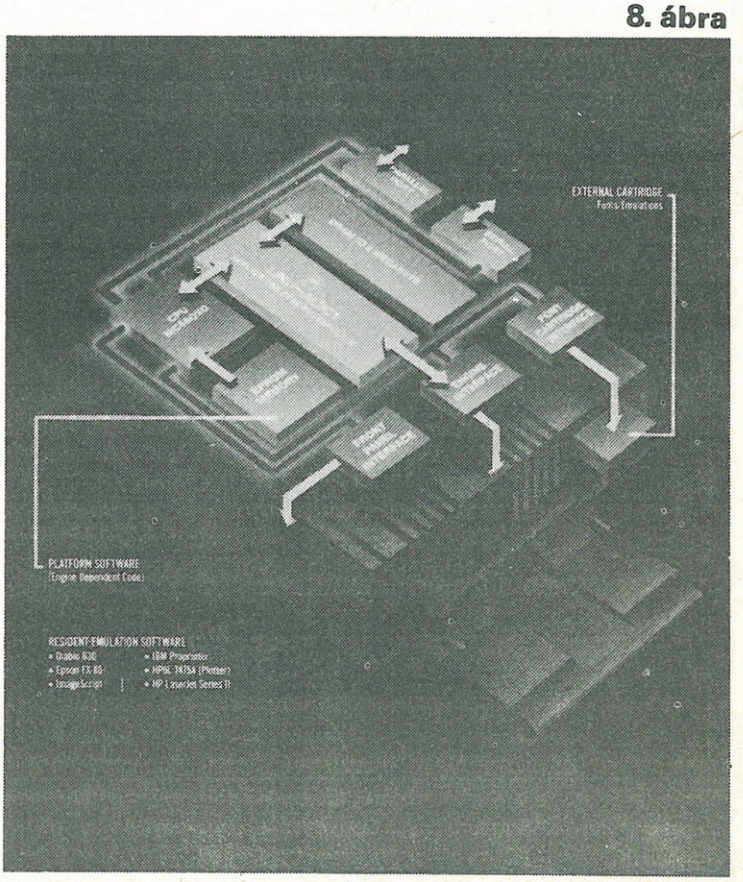
Az ötéves cég a PC-piacon az aránylag jó minőségű és olcsó grafikus kártyáiról ismert. Ő kezdte el például a 800 × 600 pont felbontású EGA kártyák gyártását. Ezt, az azóta is egyre szélesebb körben használt ASIC (Application-Specific Integrated Circuits = alkalmazásspecifikus IC) technológiának köszönhették. Új termékük a SuperEGA HiRes+ *(3. kép)* lehetővé teszi, hogy monokrom monitorral EGA és EGA monitorral Hercules kompatibilis szoftver dolgozhasson együtt. A SUPERVGA 16—Bit Adapter *(4. kép)* 16 bites busz és saját RAMBIOS használatával megkétszerezi a VGA-termékek működési sebességét. Végül a SuperHGA *(5. kép)* kártyán levő RISC processzorral (erről a típusról korábban már írtam) mintegy százszor gyorsabbá tehetők a VGA funkciók, és egyúttal sokkal egyszerűbbé teszik a működtető szoftvert.

Simonyi Endre

7. ábra



8. ábra



A felvirágoztatott háttértároló

A leggyakrabban használt, például szövegszerkesztő, assembler, monitor programoknál előnyös, ha ezeket gyorsan és egyszerűen tudjuk a számítógép memóriájába tölteni és futtatni. Azoknál a számítógépeknél, melyeknek lemezmeghajtójuk van, nem gond, de nagyon sok személyi számítógép tulajdonosának még nincs ilyen meghajtója, hanem háttértárolónak kazettás magnetofont használ. Ez az eszköz azonban viszonylag lassú, és sokszor jogos türelmetlenséggel várjuk programunk betöltésének befejezését. Hibás betöltéskor pedig kezdhetjük az egészet elölről. Jogos tehát az igény egy olyan eszközre, ami a következő feltételeknek tesz eleget:

— tárolja a programokat,
— egyszerű vele a betöltés,
— minél kevesebb vezetékkel kapcsolódik a számítógéphez,
— a legtöbb személyi számítógéphez kapcsolható, azaz típusfüggetlen,
— az ára nem magas.

Az ezeket a feltételeket kielégítő eszközt Rögzített Tartalmú Soros Tárolónak, a továbbiakban RTST-nek nevezzük. Tárolóeszközként nagy kapacitású EPROM memóriát alkalmazunk — például 16 kbájtos 27 128-as, 32 kbájtos 27 256-os vagy 64 kbájtos 27 512-es típust —, ezek ára folyamatosan csökken. Az egyes személyi számítógépekhez kapható bedugható („plug-in") modulok is ilyen memóriát tartalmaznak. Az RTST-ben új a tartalom kiolvasásának a módja: a memóriából való kiolvasást nem párhuzamosan, hanem sorosan, bitenként valósíthatjuk meg. A sorosan érkező bitekből a bájtokat a számítógépben lévő program rakja össze. A következőkben leírt megoldás már nem általános, hanem a Primo számítógépre készült, de a leírtak alapján más számítógépekre is adaptálható. A Primóhoz készült RTST kapcsolási rajza (0,5 Mbájt) az *ábrán* követhető végig.

A kapcsolási rajzon szereplő ötpólusú csatlakozó bekötése a Primo botkormány (B) csatlakozójának bekötésével egyezik meg. A törlő (RSTBE) bemenetre adott jellel alaphelyzetbe hozzuk a V10, V11 és V14 jelű számlálókat. Ezután kiolvassuk a (DKI) kimeneten keresztül a V15-ös multiplexer Y adatkimenetét, ami az EPROM nulladik címén lévő bájt nulladik bitje. A léptető bemenetre (CLBE) adott jellel léptetjük a V14-es felső 8-as számlálóját, és ismét kiolvassuk a multiplexer kimenetét, ami a megcímzett bájt első bitje. A nyolcadik léptetéskor a multiplexert címző számláló ismét alaphelyzetbe kerül, azaz ismét a megcímzett bájt nulladik bitje lesz a kimeneten. Azonban az átvitelen megjelenő jel a V11, V10-es címszámláló láncot is lépteti, és a megcímzett 1. címen lévő bájt jelenik meg az EPROM adatkimenetein. A kiolvasást ekként ismételve, sorban egymás után, bitenként kiolvasódik az EPROM tartalma. Az első EPROM-tokban lévő utolsó bájt kiolvasása után a V14-es IC alsó számlálója is egyet lép. A számláló kimenete a V13-as dekódolóra kapcsolódik, melynek kimenetei sorban egymás után kiválasztják az EPROM-tokokat, ebben az esetben a másodikat. A kiolvasás tehát a következő tokból folytatódik. A V13-as 12. lábára adott H szintű jellel az egész kártya működését tilthatjuk.

Az RTST a számítógéptől két kimeneti és egy bemeneti bitet igényel. A kapcsolás maximum nyolc darab 64 kbájtos tokkal használható, azaz a tárolási kapacitás maximum 512 kbájt lehet! Természetesen kisebb kapacitású EPROM-ok is használhatók, a JI jelű átkötésmező megfelelő összekötéseivel.

A következő lépés az RTST-t működtető szoftver megtervezése és megvalósítása. A szoftver legfontosabb eleme egy olyan betöltőprogram, amely képes a fenti hardver működtetésére, azaz az EPROM tartalmát a memória egy adott kezdőcímétől képes betölteni, majd a betöltött programot elindítani. Az EPROM tartalmát célszerű két részre bontani: az első rész tartalmazza az EPROM második részében található programokra vonatkozó információkat, azt hogy:

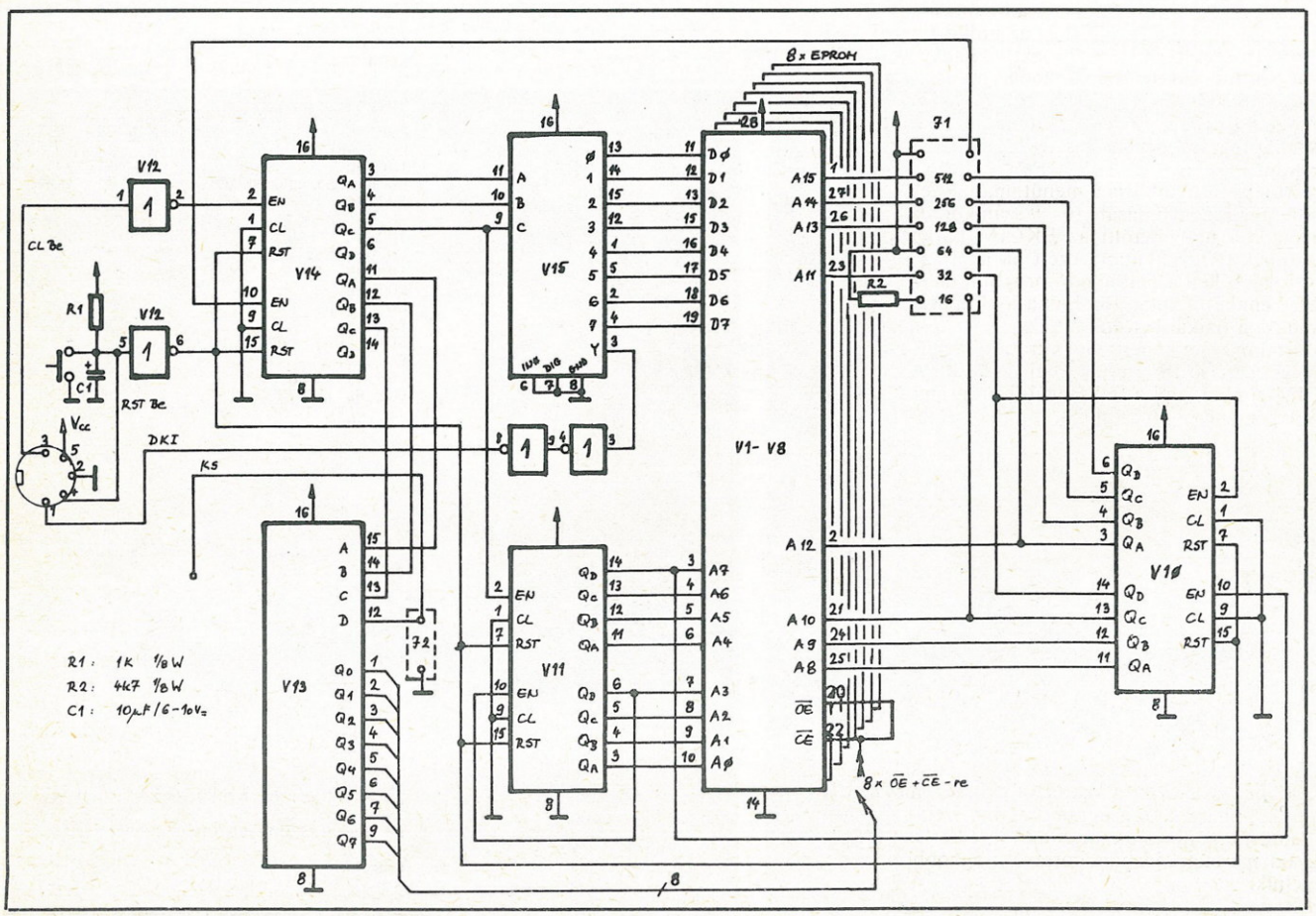
— az EPROM melyik címén kezdődik a program (EKCIM),
— a memória melyik címétől töltődik a program (MKCIM),
— melyik memóriacímig tart a betöltés (MVCIM),
— a memória melyik címéről indul a program futása (ICIM).

Ez a 4×2 bájt egyértelműen meghatározza a programot, és ezek alapján a program betölthető és futtatása elindítható.

Az első EPROM elején, a programokra vonatkozó paramétertáblázat előtt helyezkedik el az a program, amelynek feladata, hogy a felhasználónak egy menüt felkínálva, kiválaszthassa a futtatni kívánt programot. Az így felépített rendszerrel elvileg tetszőleges program futtatható, mivel csupán az EPROM első részében lévő menü szövegét és a program paramétereit kell megfelelő módon beírni.

A következő — a fentiek alapján működő — program Primo számítógépre készült. Mivel a Primo botkormány-csatlakozóját használtuk fel, ami csak egy kimeneti vonalat tartalmaz, ezért az áramköröket törlő RESET jelet egy nyomógombbal kell generálni. Ez azt jelenti, hogy bekapcsoláskor vagy új program betöltése előtt a gombot meg kell nyomni.

A Primóra készült betöltőprogram két, fizikailag is különálló részből áll. Az első program a tulajdonképpeni fizikai betöltő, amely az RTST első EPROM-jának elején lévő menüt megjelenítő, valamint a kiválasztott programot betöltő és indító programot tölti be a képernyő-memória utolsó 256 bájtjába. Ez a tartomány 64-es gépeknél az FF00H, a 48-as gépeknél a BF00H, a 32-es gépeknél a 7F00H memória kezdőcímen található. Azért esett a választás erre a területre, mert egyrészt a futtatandó program kiválasztása után ez a program már „megsemmisülhet", másrészt ily módon nem foglalunk el a memóriából olyan részt, amit valamelyik program felhasznál. A fizikai betöltőnek a számítógép memóriájában kell lennie azért, hogy működtesse az RTST kártya áramköreit. Primónál a fizikai betöltő kétféleképpen helyezhető el:



RTSTOL betöltőprogram

```
                ***Z80 PROGRAM***

Programnév: RTSTOL                  Programhossz:  38 bájt

A program feladata: RTST programok betöltése

Felhasznált regiszterek: A, BC, DE, HL, IY


 1  ;ROMBAN LEVÖ BETOLTÖ
 2  067B                        ORG 67BH
 3  067B   21 00 FF    RTSTOL:  LD HL,0FF00H ;MKCIM
 4  067E   11 FF FF             LD DE,0FFFFH ;MVCIM
 5  0681   FD 21 00 FF          LD IY,0FF00H ;ICIM
 6  0685   06 08        TOLT:   LD B,8        ;EGY BAJT BEOLV.
 7  0687   DB 40        IDE:    IN A,(40H)    ;OLVASAS
 8  0689   1F                   RRA
 9  068A   CB 19                RR C          ;TOLAS C-BE
10  068C   3E 9D                LD A,9DH      ;SZAMLALÓ LÉPTETÉS
11  068E   D3 00                OUT (0H),A    ; 0 SZINT KI
12  0690   3E ED                LD A,0EDH
13  0692   D3 00                OUT (0H),A    ; 1 SZINT KI
14  0694   10 F1                DJNZ IDE      ; HA VAN MÉG A BAJTBÓL
15  0696   71                   LD (HL),C     ; BAJT TÁROLÁSA
16  0697   23                   INC HL        ; KÖVETKEZÖ BAJT
17  0698   E5                   PUSH HL
18  0699   AF                   XOR A
19  069A   ED 52                SBC HL,DE     ;UTOLSÓ BAJT VOLT?
20  069C   E1                   POP HL
21  069D   38 E6                JR C,TOLT     ;HA NEM, VISSZA
22  069F   FD E9                JP (IY)       ;UGRÁS ICIM-RE
23  06A1                        END
```

1. Fizikai betöltő a BASIC ROM-ban

A Primo BASIC-et tartalmazó EPROM-tokokban van olyan „üres" hely, amit a BASIC nem használ, ide a fizikai betöltőprogram beírható. Ez azonban azzal a kényelmetlenséggel jár, hogy a Primóba beforrasztott EPROM-tokok közül az üres helyet tartalmazó tokot ki kell emelni, majd ismét vissza kell helyezni (legjobb a helyére foglalatot forrasztani). Egészen pontosan: az alsó 4 k memóriatartományt tartalmazó tok — a panel szélén lévő — 67BH-6A0H címére kell beírni az RTSTOL nevű betöltőprogramot.

2. Fizikai betöltő szalagról

Semmiféle fizikai beavatkozást nem igényel, ha a fizikai betöltőprogramot szalagról betölthető öninduló assembler programmá alakítjuk. Ekkor a program a 4127H-414FH RAM-területre töltődik. A fizikai betöltő indítási címe 4134H. Ha ez a betöltött program nem sérül meg, újra elindítható a ?call(16679) utasítással.

A fizikai betöltőprogramnak olyan a kialakítása, hogy mind a menükiíró és -kiválasztó programot, mind a segítségével kiválasztott felhasználói programot képes betölteni és elindítani. Ezt úgy értük el, hogy a fizikai betöltőt két részre tagoltuk: az első rész állítja be a betöltési és indítási paramétereket, a második rész pedig a betöltést és az indítást végzi. Először a fizikai betöltőt az RTSTOL címkén keresztül aktivizáljuk, amivel a menüprogramot töltjük be és indítjuk el. A menüből kiválasztott felhasználói program betöltési paramétereit a regiszterekbe töltve, a vezérlés az STOLT címen keresztül ismét a fizikai betöltőnek adható át, ami betölti és elindítja a programot. A menüprogram három részre osztható. Az első rész az RTSTPR nevű program.

A program egyrészt a képernyőre írja a menüt, majd a felhasználói programot kiválasztó billentyű megnyomására (0—9 számkódok) várakozik. Értelmezhető kód megadása után betölti az EKCIM paramétert. Mivel a számlálók Ø-ról indulnak (RESET után), ezért ha nem az első programot választottuk, akkor tovább kell a betöltendő program elejére lépni az EPROM-okban (CSLEP címke). Ezután történik a többi paraméter betöltése, majd a vezérlés átadása a fizikai betöltőre.

A második rész tartalmazza a képernyőre kiírandó menüszöveget (TEXT címke). A szöveg végét az első FFH értékű bájt jelöli ki. A harmadik rész egy táblázat (TKCIM címke), ami a felhasználói programok paramétereit tartalmazza. A példaként négy programot tartalmazó táblázat az alábbi.

Programnév	EKCIM	MKCIM	MVCIM	ICIM
EDI	100	42E8	5548	5538
PBUG	1360	43C0	53BF	43C0
EPR.PRG	2400	E400	E75F	E400
SZMAS	2800	5700	5870	5700

(EDI = Editor-assembler, PBUG = Monitor-disassembler, EPR.PRG = 2716/2732 EPROM programozó, SZMAS = szalagmásoló program)

A szöveg és a táblázat még nem használt bájtjainak értéke FFH. Ezért újabb programokat úgy illeszthetünk be a rendszerbe, hogy azokat egyszerűen hozzáírjuk a menüszöveghez, és a program paramétereit beírjuk a táblázatba. A meglévő EPROM-tartalmakat nem kell módosítani. Nagyobb kapacitású RTST alkalmazásakor kicsinek bizonyulhat a menüszöveg mezője, és a maximum tíz felhasználói program paramétereit befogadó táblázat is. Aggodalomra azonban nincs ok, mert mindkét korlát bővíthető úgy, hogy a menüprogram méretét 256 bájt hosszról nagyobbra — 512 vagy 1024 bájtra — növeljük.

K. L.

RTSTPR program

```
                ***Z80 PROGRAM***

Programnév: RTSTPR                  Programhossz: MAX.256 bájt

A program feladata: Az RTST directory betöltése és a
                    kiválasztott program indítása

Felhasznált regiszterek: A, BC, DE, HL, IY


 1          ;RTST PROGRAMOK BETOLTÖJE
 2  0000                        ORG 0H
 3          ;
 4          FF60      MENU:     EQU 0FF60H ;SZOVEG KEZDETE
 5          FFB0      TABL:     EQU 0FFB0H ;PARAM.TABLAZAT KEZDETE
 6          0015      DSPL:     EQU 15H ;PRIMO KEPERNYOKEZELO
 7          0025      KLAV:     EQU 25H ;PRIMO BILLENTYUKEZELO
 8          0685      STOLT:    EQU 685H ;FIZIKAI BETOLTO A ROMBAN
 9          ;
10  0000   21 60 FF  KIIR:     LD HL,MENU
11  0003   7E        ISMET:    LD A,(HL)
12  0004   FE FF               CP 0FFH ;SZOVEG VEGE?
13  0006   28 06               JR Z,UJBI ;IGEN
14  0008   CD 15 00            CALL DSPL ;KIIRAS A KEPERNYORE
15  000B   23                  INC HL ;UJ KARAKTER
16  000C   18 F5               JR ISMET
17  000E   CD 25 00     UJBI:  CALL KLAV ;VARAS BILLENTYURE
18  0011   28 FB               JR Z,UJBI ; HA NEM NYOMTAK
19  0013   D6 30               SUB 30H
20  0015   FE 00               CP 0H      ; SZAMBILLENTYU ?
21  0017   38 F5               JR C,UJBI ; <0 ?
22  0019   FE 0A               CP 0AH
23  001B   30 F1               JR NC,UJBI ; >9 ?
24  001D   07           TOLT:  RLCA ; * 2
25  001E   07                   RLCA ; * 4
26  001F   07                   RLCA ; * 8
27  0020   4F                   LD C,A
28  0021   06 00               LD B,0 ;BC=RELATIV TABLACIM
29  0023   21 B0 FF            LD HL,TABL ;HL=TABLAZAT KEZDOCIME
30  0026   09                   ADD HL,BC ;TENYLEGES CIM
31  0027   4E                   LD C,(HL)
32  0028   23                   INC HL
33  0029   46                   LD B,(HL) ;BC=EKCIM
34  002A   23                   INC HL
35  002B   78                   LD A,B
36  002C   FE FF               CP 0FFH
37  002E   28 D0               JR Z,KIIR ;EHHEZ NINCS PROGRAM
38  0030   E5                   PUSH HL
39  0031   05                   DEC B ; MERT 256-OT MAR LEPTUNK
40  0032   C5           CSLEP:  PUSH BC ;FELLEPTETETES AZ EPROMBAN
41  0033   06 08               LD B,8 ; EGY BAJT
42  0035   3E 9D        CSLEP1: LD A,9DH ;SZAMLALÓ LÉPTETÉS
43  0037   D3 00               OUT (0),A ; 0 KIVITELE
44  0039   3E ED               LD A,0EDH
45  003B   D3 00               OUT (0),A ; 1 KIVITELE
46  003D   10 F6               DJNZ CSLEP1
47  003F   C1                   POP BC ;BC=EKCIM
48  0040   78                   LD A,B
49  0041   B1                   OR C    ;BC=0 ?
50  0042   0B                   DEC BC
51  0043   20 ED               JR NZ,CSLEP ;HA BC NEM 0
52  0045   E1                   POP HL
53  0046   4E                   LD C,(HL)
54  0047   23                   INC HL
55  0048   46                   LD B,(HL) ;BC=MKCIM
56  0049   23                   INC HL
57  004A   5E                   LD E,(HL)
58  004B   23                   INC HL
59  004C   56                   LD D,(HL) ;DE=MVCIM
60  004D   23                   INC HL
61  004E   C5                   PUSH BC
62  004F   4E                   LD C,(HL)
63  0050   23                   INC HL
64  0051   46                   LD B,(HL) ; BC=ICIM
65  0052   C5                   PUSH BC
66  0053   FD E1               POP IY ;IY=ICIM
67  0055   E1                   POP HL ;HL=MKCIM
68  0056   C3 85 06            JP SSDTOLT ;FIZIKAI BETOLTO ROMBAN
69          ;
70  0060                        ORG 60H ; MENU SZOVEGE
71  0060   02 52 54 53 54       DB 02H,"RTST PROGRAMOK:",12H
    0065   20 50 52 4F 47
    006A   52 41 4D 4F 4B
    006F   3A 12
72  0071   0D 0D 30 3D 3E       DB 0DH,0DH,"0=>EDI"
    0076   45 44 49
73  0079   0D 31 3D 3E 50       DB 0DH,"1=>P-BUG"
    007E   2D 42 55 47
74  0082   0D 32 3D 3E 45       DB 0DH,"2=>EP.PRG"
    0087   50 2E 50 52 47
75  008C   0D 33 3D 3E 53       DB 0DH,"3=>SZ.MAS"
    0091   5A 2E 4D 41 53
76  0096   0D                   DB 0DH ; KIIRAS VEGE
77
78  00B0                        ORG 0B0H;TABL: (PARAMETERTABLAZAT)
79  00B0   00 01E8 4348 55      DW 0100H,43E8H,5548H,5538H ;EDI
    00B6   3855
80  00B8   60 13C0 43BF 53      DW 1360H,43C0H,53BFH,43C0H ;P-BUG
    00BE   C043
81  00C0   00 2400 E45F E7      DW 2400H,E400H,E75FH,E400H ;EP.PRG
    00C6   00E4
82  00C8   00 2800 5770 58      DW 2800H,5700H,5870H,5700H ;SZ.MAS
    00CE   0057
83  00D0                        END
```

Konverziós program IBM PC-re

Az IBM PC gépeken máig sincs egységes, teljes magyar ékezetes karakterkészlet. Az eredeti IBM karakterkészlet tartalmaz ékezetes betűket, de néhány — például a nagy hosszú Ú stb. — hiányzik.

Ezen csak a karaktergenerátor EPROM cseréjével lehet segíteni. A Videoton VT110-esnél ezt meg is tették, amitől a gép elvesztette az eredetiekkel való teljes kompatibilitását. Az is elképzelhető, hogy a hiányzó karaktereket más — megegyezés szerinti — karakterekkel helyettesítsük, például hosszú ő=Θ.

Sajnos a VT110 módosított karaktergenerátorában a hiányzó karaktereket olyan kódokra helyezték, amelyekből néhány még véletlenül sem hasonlít az eredeti IBM megfelelőjére, például a hosszú I=Ç [128]. Ezért ha VT110-en készült szövegfájlt akarunk IBM gépen kinyomtatni, a fájlt konvertálni kell. Bemutatok a VT110 → IBM konverzióra egy megoldást.

A program a DOS (V2.0 vagy későbbi) Standard perifériakezelési lehetőségét használja ki. A DOS öt Standard egységet ismer:

Név	Rövidítve	Fájlszám	Alapértelmezés
Standard Input	STDIN	0	CON
Standard Output	STDOUT	1	CON
Standard Error	STDERR	2	CON
Standard Auxiliary	STDAUX	3	AUX
Standard List	STDLST	4	PRN

A DOS az STDIN-t, az STDOUT-ot és az STDERR-t használja a felhasználóval való kommunikációra. Mindegyik a konzolra vonatkozik. A CON bemenetként a billentyűzetet, kimenetként a képernyőt jelenti. A másik kettő a printer (PRN) és az első soros vonal (AUX) kezelésére szolgál.

Ezek az eszközök mindig megtalálhatók a megadott fájlszámokon (0—4). Ugyanúgy kezelhetők, mintha fájlok lennének, írni-olvasni lehet őket.

Ezenkívül a felhasználónak lehetősége van arra is, hogy az STDIN és STDOUT más egységre vagy fájlra irányításával elérhesse például azt, hogy egy program kimenetét ne a képernyő-

VT110 — IBM karakterkonvertáló program

```
;vt.asm
;
;   VT110 - IBM Karakter konvertáló program
;
;
;   (c) 1988 Szinyei Gerzson  KKVMF-EABI
;

DOS             equ 21h     ; DOS Funkciók Interruptja
STDIN           equ 0       ; Standard Input fileszáma
STDOUT          equ 1       ; Standard Output fileszáma
READ            equ 3fh     ; Fn 3f-olvasás file-ból vagy egységről
WRITE           equ 40h     ; Fn 40-irás file-ba vagy egységre
PRINT_STRING    equ 09h     ; Fn 09-karakterlác kiiratása
NR              equ 1       ; Irandó-olvasandó karakterek száma
CR              equ 13      ; Kocsi vissza
LF              equ 10      ; Soremelés
EOT             equ '$'     ; Szöveg vége a DOS PRINT_STRING-nek

VTsegm          segment 'code'
                assume CS:VTsegm,DS:VTsegm
                org 80h
parnum:         db ?                        ;Paraméter sor hossza (PSP:80h)
                org 100h                    ; .COM program miatt
VT              proc    far
                cmp byte ptr parnum,0   ;Van paraméter?

                jne loop                    ;Igen->program futás

                mov dx,offset text      ;Nem->bevezetö szöveg
                mov ah,PRINT_STRING     ;kiirása
                int DOS
                jmp end_process         ;Vissza a DOS-hoz

text:       db 'VT110 »»» IBM  Karakter konvertáló program',CR,LF
            db CR,LF
            db 'Hivása:  VT <bemeneti fájlnév >kimeneti fájlnév',CR,LF
            db EOT

loop:           mov ah,READ             ;Egyetlen karakter
                mov bx,STDIN            ;olvasása a Standard
                mov dx,offset char      ;Input-ról
                mov cx,NR               ;
                int DOS                 ;

                jc error                ;OK?

                cmp ax,1                ;EOF?
                jne end_process         ;

                mov al,byte ptr char    ;A beolvasott karak-
                mov bx,offset Newtable  ;ter konverziója a
                xlat                    ;'Newtable' alapján
                mov byte ptr char,al    ;

                mov ah,WRITE            ;A konvertált karakter
                mov bx,STDOUT           ;kiirása a Standard
                mov dx,offset char      ;Output-ra
                mov cx,NR               ;
                int DOS                 ;

                jc error                ;OK?

                cmp ax,1                ;LEMEZ TELE?
                jne error

                jmp short loop

end_process:    mov ax,4C00h            ;Visszatérés a DOS-hoz 0-ás
                int DOS                 ;ERRORLEVEL-lel (ok)

error:          mov ax,4C01h            ;Visszatérés a DOS-hoz 1-es
                int DOS                 ;ERRORLEVEL-lel (hiba)

char:           db 0                    ;Karakter tároló

                org 200h                ;A Debug-gal, vagy
                                        ;PCTools-szal való könnyebb
                                        ;szerkesztés miatt
Newtable:                               ;A módositásokat tartalmazó
                                        ;tábla

        db  0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
        db  16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
        db  ' !"#$%&',39,'()*+,-./'
        db  '0123456789:;<=>?'
        db  '@ABCDEFGHIJKLMNO'
        db  'PQRSTUVWXYZ[\]^_'
        db  '`abcdefghijklmno'
        db  'pqrstuvwxyz{|}~'

        db  8ch,'üéâäàåçëë',0e9h,93h,95h,98h,'ÄÅ'   ;ezekben a sorokban
        db  'æÆöòûù',0a2h,'öÜ¢£¥₧ƒ'                ;van változás az e-
        db  'áíóúñÑªº¿⌐¬½¼¡«»'                     ;redetihez képest

        db  '░▒▓│┤╡╢╖╕╣║╗╝╜╛┐'
        db  '└┴┬├─┼╞╟╚╔╩╦╠═╬╧'
        db  '╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀'
        db  'αßΓπΣσµτΦΘΩδ∞Ø∈∩'
        db  '≡±≥≤⌠⌡÷≈°∙·√ⁿ²■',255


VT              endp
VTsegm          ends
                end VT

; Az eredeti karaktertábla

        db  0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
        db  16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
        db  ' !"#$%&',39,'()*+,-./'
        db  '0123456789:;<=>?'
        db  '@ABCDEFGHIJKLMNO'
        db  'PQRSTUVWXYZ[\]^_'
        db  '`abcdefghijklmno'
        db  'pqrstuvwxyz{|}~'
        db  'ÇüéâäàåçëëèïîìÄÅ'
        db  'æÆöòûùÿÖÜ¢£¥₧ƒ'
        db  'áíóúñÑªº¿⌐¬½¼¡«»'
        db  '░▒▓│┤╡╢╖╕╣║╗╝╜╛┐'
        db  '└┴┬├─┼╞╟╚╔╩╦╠═╬╧'
        db  '╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀'
        db  'αßΓπΣσµτΦΘΩδ∞Ø∈∩'
        db  '≡±≥≤⌠⌡÷≈°∙·√ⁿ²■',255
```

re, hanem egy fájlban küldje és később szerkessze vagy nyomtassa.

Természetesen ez csak akkor működik, ha az adott program az STDOUT-ot használja kiíratásra, nem pedig ezt megkerülve, mondjuk közvetlenül ír a videomemóriába.

A Standard IO a DOS kijelentkező promptjánál irányítható át, a végrehajtandó program neve után beírt speciális karakterekkel. Ezek a karakterek: <, >, »,¦.

Jelentésük:

<fájl vagy egységnév. Az STDIN a billentyűzet helyett a megadott fájlból vagy egységből olvas.

>fájl vagy egységnév. Az STDOUT a képernyő helyett a megadott fájlba vagy egységre ír. A fájl régi tartalma elvész.

»fájlnév. Ugyanaz, mint >fájlnév, de a fájlba hozzáfűzéssel (APPEND) ír.

prog1¦ prog2. Az első program STDOUT-ját egy ideiglenes fájlba írja az aktuális alkönyvtárban, azután innen veszi a második program STDIN-jét. A második program vége után letörli az ideiglenes fájlt. Ezekről az átirányításokról a parancsfeldolgozó (COMMAND.COM) gondoskodik, a hívott program számára láthatatlanul. Például:

```
DIR »CATALOG        a directoryt hozzáfűzi a
                    CATALOG nevű fájlhoz.
```

A VT.COM készítése: az assembly listát írjuk be egy .ASM kiterjesztésű fájlba (VT.ASM). Ezután fordítsuk le az alábbi módon:

```
MASM VT;
LINK VT;
EXE2BIN VT.EXE VT.COM
```

Ezután már megvan a VT.COM fájl, amit az alábbi módokon használhatunk:

```
TYPE UZENET ¦VT >PRN
```

Az UZENET fájl tartalmát konvertálva kinyomtatja.

```
VT <forrás fájl neve >cél fájl neve
```

Ekkor a VT a forrásfájlt konvertálja és tárolja a célfájlban.

Ha a programot paraméterek nélkül futtatjuk, kiírja a helyes szintaxisát. Természetesen fájlnevek helyett adhatunk meg egységneveket is, a kimeneti fájlnév elhagyása esetén pedig automatikusan a képernyőre ír.

A bejelentkező szöveget mindenki tetszés szerint módosíthatja a forráslistában vagy a kész programban, ekkor azonban a szöveg hossza nem változhat.

Eredeti (VT11Ø)		Konvertált (IBM)	
Kód	Karakter	Kód	Karakter
AØH	á	AØH	á
82H	é	82H	é
A1H	í	A1H	í
98H	ó	A2H	ó «
94H	ö	94H	ö
8BH	ő	93H	ô «
A3H	ú	A3H	ú
81H	ü	81H	ü
96H	ű	96H	û
8FH	Á	8FH	Á
9ØH	É	9ØH	É
8ØH	Í	8CH	î «
8CH	Ó	95H	ò «
99H	Ö	99H	Ö
8AH	Ő	E9H	Θ «
97H	Ú	97H	ù
9AH	Ü	9AH	Ü
8DH	Ű	98H	ÿ «

Az ékezetes karakterek konverziós táblája

A VT program természetesen megfelelően módosítva más feladatok ellátására is képes (például kisbetű-nagybetű konverzióra). Ehhez a konverziós táblát a Debug rendszerprogrammal vagy a népszerű PCTools programmal módosíthatjuk.

A könnyebb módosítás kedvéért került a karaktertábla kerek (2ØØH) címre a fájlban.

A Debuggal készült hexa dumpon láthatjuk a jelenlegi konverziós táblát. Az átírandó karakter helyére (2ØØH + a karakter kódja) beírjuk a helyette látni kívánt karakter kódját.

Szinnyei Gerzson

Hexa dump a karaktertábláról

```
A>debug VT.COM
-d 200 2ff
ØFEØ:Ø2ØØ  ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7-Ø8 Ø9 ØA ØB ØC ØD ØE ØF  ................
ØFEØ:Ø21Ø  1Ø 11 12 13 14 15 16 17-18 19 1A 1B 1C 1D 1E 1F  ................
ØFEØ:Ø22Ø  2Ø 21 22 23 24 25 26 27-28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
ØFEØ:Ø23Ø  3Ø 31 32 33 34 35 36 37-38 39 3A 3B 3C 3D 3E 3F  Ø123456789:;<=>?
ØFEØ:Ø24Ø  4Ø 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
ØFEØ:Ø25Ø  5Ø 51 52 53 54 55 56 57-58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
ØFEØ:Ø26Ø  6Ø 61 62 63 64 65 66 67-68 69 6A 6B 6C 6D 6E 6F  `abcdefghijklmno
ØFEØ:Ø27Ø  7Ø 71 72 73 74 75 76 77-78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}~.
ØFEØ:Ø28Ø  8C 81 82 83 84 85 86 87-88 89 E9 93 95 98 8E 8F  ................
ØFEØ:Ø29Ø  9Ø 91 92 93 94 95 96 97-A2 99 9A 98 9C 9D 9E 9F  ................
ØFEØ:Ø2AØ  AØ A1 A2 A3 A4 A5 A6 A7-A8 A9 AA AB AC AD AE AF  ................
ØFEØ:Ø2BØ  BØ B1 B2 B3 B4 B5 B6 B7-B8 B9 BA BB BC BD BE BF  ................
ØFEØ:Ø2CØ  CØ C1 C2 C3 C4 C5 C6 C7-C8 C9 CA CB CC CD CE CF  ................
ØFEØ:Ø2DØ  DØ D1 D2 D3 D4 D5 D6 D7-D8 D9 DA DB DC DD DE DF  ................
ØFEØ:Ø2EØ  EØ E1 E2 E3 E4 E5 E6 E7-E8 E9 EA EB EC ED EE EF  ................
ØFEØ:Ø2FØ  FØ F1 F2 F3 F4 F5 F6 F7-F8 F9 FA FB FC FD FE FF  ................
```

A The Newsroom magyar változata II. rész

A Newsroom tulajdonságainak összefoglalása és a billentyűzet lehetőségeinek korlátai után most tüzetesebben foglalkozunk a billentyűzet felhasználásával.

Amint a billentyűzet elrendezési táblázatából látható *(3. ábra)*, a betű sorszámát kiszámíthatjuk úgy, hogy az ASCII kódjához hozzáadunk 29-et.

Ha végignézzük a programlemezt, négy karaktertáblázatot találunk rajta:

sfnts — kisméretű betűk,
lfnt0 — Serif betűtípus,
lfnt1 — Sans Serif betűtípus,
lfnt2 — English betűtípus.

(A rövidítések a small fonts, illetve a large font szavakból származnak.)

A karakterkészlet számlistájának elején a betöltési cím áll, majd a betűk említett sorrendjében a betűk szélessége következik, ami tehát kilencvennyolc szélességadat. Ha a szélességet eggyel kisebbre vesszük, a betűk egymáshoz simulnak; így valósíthatjuk meg a kötött írást.

A kisméretű betűk egymás után, egy állományban helyezkednek el. Tehát a 98. után a 99. helyen a Sans Serif számára tervezett szóköz áll (ennek egyetlen adata van: a szélesség), utána a 100. helyen ismét felkiáltójel áll. Vagyis ha tudjuk egy betű helyét az első táblázatban, adjunk hozzá 96-ot, ott található ugyanaz a betű a másik betűtípussal. A szélességtáblázat után találjuk a betűk alakját.

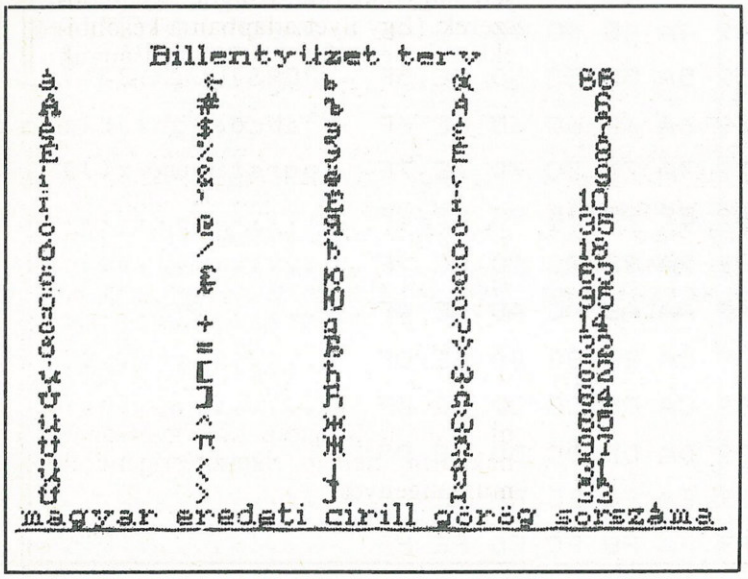
A kisméretű betűk 8 × 8-as pontmátrixból épülnek fel úgy, hogy az első adat a betű tetejét írja le, a következők pedig lefelé haladva írják le a betűt. A 203. bájton tehát 128 áll, mert a felkiáltójel bal felső pontja a betűtörzs tetejéig ér. Az idézőjel tárolását a *4. ábra* szemlélteti.

A nagyméretű betűk a lemezen három állományban találhatók, a fent említett neveken. Méretük 16-szor 16 pixel. Ekkora felbontóképességgel már igen bonyolult betűformák hozhatók létre. Ezt példázza e cikk főcíme, amely az English betűtípussal készült.

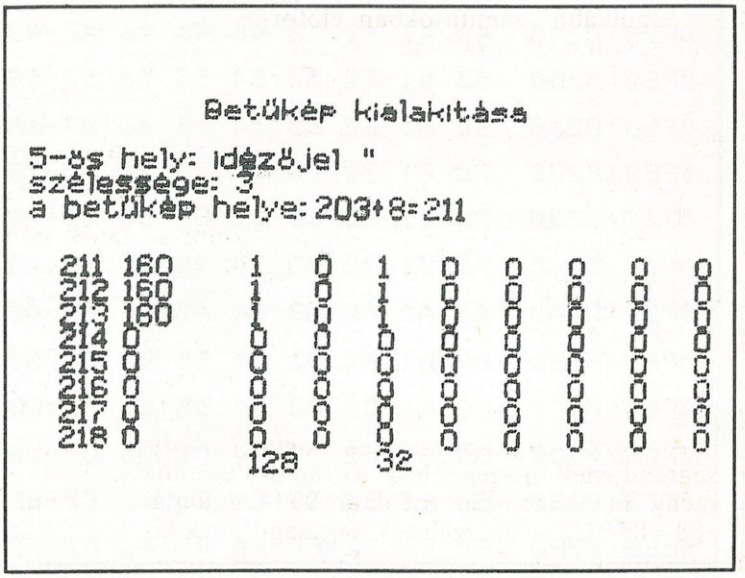
A betűformák tárolása tulajdonképpen redundáns, mert például a felkiáltójel *(5. ábra)* három-, illetve ötpixelnyi méretű, s ezeket tizenhat lehetséges helyre írjuk be. Ha végignézzük a betűformák kódtáblázatát, ezeken a helyeken nullákkal váltakozó számokat találunk; a nullák a betűkép jobb oldali, használatlan elemeit írják le: 32 0 112 0 248 0 112 0 112 0 stb. Az 5. ábra e számok elhelyezkedését magyarázza.

A billentyűzetkiosztási terv, mint említettem, két elemében az Easy Scriptéhez igazodik. Ennek következtében az ö és ü betűk a fontjelen, illetve a hatványozás jelen vannak, ezek a Newsroomban is shiftelhetők. Minden más betűt igyekeztem úgy elhelyezni, hogy lehetőleg a megfelelő ékezet nélküli alapbetű közelében legyen. A gépen meglévő páros billentyűket — szögletes zárójelek és relációjelek — egymáshoz tartozó betűk leírására veszem

3. ábra



4. ábra



igénybe. Ezért itt találhatók az u, ú, illetve ü, ű betűk. A többi a „maradék" billentyűkön foglal helyet. Az elhelyezés természetesen minden hasonló programnál végiggondolandó. Az olvasó a betűket máshogyan is elrendezheti.

A cirill betűkkel megint csak gondban voltam. Ezeknek a száma kevesebb, mint a magyar betűké. Ezért megpróbáltam elhelyezni a szerbhorvát betűket is. Ez azonban csak részben sikerült. A „gy" és a jésített „cs" igénybe vett négy billentyűt. El kellett hagynom a jésített „l", „n" és „dzs" betűket. Néhány szöveg áttanulmányozása után úgy találtam, hogy ezeket le lehet írni összetett betűkkel, és ez nem számít súlyos helyesírási hibának. Ezzel szemben elengedhetetlenül szükséges a latin „j" betű, ami nem pótolható a cirill megfelelőjével; elhagyása súlyos helyesírási hiba. Minthogy azonban orosz nyelvű szövegeket sűrűbben írunk, a „j" billentyűre a cirill „j" betű került, a latin a relációjelekre.

Töprengésre késztetett a görög betűk billentyűzeten való elhelyezése is. A betűket általában hangalakjuknak megfelelően rendeztem el. A hasonló hangértékű betűknél az alábbiak szerint jártam el.

Az omikron az „o" betűre került, ezért az omegát az „u" betűre tettem. Az „e" betűre az epszilont kellett tennem, ezért az éta a „j" betűre került. A „c" billentyűre helyeztem az étát, éles ékezettel. A tau helye a „t" billentyű, ezért a thétát a „v" billentyűre tettem. A „q" betű gazdátlan, ezt arra használtam fel, hogy a szóvégi szigma önálló formáját is leírhassam. A pszi betű a gazdátlan „w" billentyűn helyezkedik el. Azokat a billentyűket, amelyeken magyar ékezetes betűk voltak, ékezetes görög betűk céljára tartottam fenn.

```
Nagyméretű betűkép tárolása

4-es hely: felkiáltójel , ! (English)
szélessége: 5
a betűkép helye: 131-től 162-ig.
131 :  32  132 :  0    0 0 1 0 0 0 0 0 0 0 .
133 :  112 134 :  0    0 1 1 1 0 0 0 0 0 0 .
135 :  248 136 :  0    1 1 1 1 1 0 0 0 0 0 .
137 :  112 138 :  0    0 1 1 1 0 0 0 0 0 0 .
139 :  112 140 :  0    0 1 1 1 0 0 0 0 0 0 .
141 :  112 142 :  0    0 1 1 1 0 0 0 0 0 0 .
143 :  32  144 :  0    0 0 1 0 0 0 0 0 0 0 .
145 :  32  146 :  0    0 0 1 0 0 0 0 0 0 0 .
147 :  32  148 :  0    0 0 1 0 0 0 0 0 0 0 .
149 :  32  160 :  0    0 0 1 0 0 0 0 0 0 0 .
151 :  0   152 :  0    0 0 0 0 0 0 0 0 0 0 .
153 :  0   154 :  0    0 0 0 0 0 0 0 0 0 0 .
155 :  32  156 :  0    0 0 1 0 0 0 0 0 0 0 .
157 :  112 158 :  0    0 1 1 1 0 0 0 0 0 0 .
 .     .     .  .     . s t b . . . . . . .
```

5. ábra

A Newsroom betűválasztékát a jövőben gyarapítani szeretném. Legközelebbi terveim között a szláv huszita írás betűinek elkészítése szerepel, hogy cseh, szlovák, lengyel, szerbhorvát szövegek írását is lehetővé tegyem.

Dr. Zana János

Adom a magyarázatot!

Az egyenletrendszert megoldó program hibáit hosszasan elemeztem. A legutóbbi számban megemlítettem: ennek az az oka, hogy a probléma igen általános. Bár az egyenletrendszerek megoldása is igen gyakran nélkülözhetetlen, ezúttal még általánosabb témákkal foglalkozom. Ez idáig a gépi számítás korlátozott számtartomány-használatából adódó gondokról írtam. A számtartomány túllépését leginkább számításokban előforduló szorzások/osztások okozzák. A továbbiakban a számokat pozitívnak veszem. A negatív előjel az összeadásból kivonást csinál és viszont. A szorzásnál és az osztásnál nincs jelentősége.

Két szám összeadásánál az eredmény nagyságrendje legfeljebb egygyel lehet nagyobb, mint a nagyobbik összeadandó. Például $999+88=1087$. Az összeadandók nagyságrendje 3 és 2, az eredményé 4. A kivonás csak csökkentheti a nagyságrendet.

A szorzásnál a szorzandók nagyságrendjének összege lesz az eredmény nagyságrendje (például $999 \cdot 88 = 87\,912$). A szorzandók nagyságrendje itt is 3 és 2, az eredményé 5. Itt tehát a növekedés sokkal nagyobb lehet, mint az összeadásnál.

Az osztásnál a nagyságrend kivonódik. Az eredmény nagyságrendje tehát kisebb lesz, mint az osztandóé, ha az osztó egynél nagyobb szám. Amennyiben az osztó egynél kisebb, akkor a helyzet azonos a szorzáséval, hiszen egész számot (vagy törtet) törtel úgy osztunk, hogy az osztó reciprok értékével szorzunk.

Eddig nem foglalkoztam a számítások pontatlansága miatt előforduló és itt is meglévő hibaforrással. Legyen történetesen a gépi számítás pontossága 12 jegy, legyen az elvégzendő művelet kivonás, folytatódjon a program az egyik irányban, ha az eredmény nem zérus, és egy másik irányban, ha zérus. Legyen az egyik szám $1+10^{-12}$, a másik 1. A különbség ebből adódóan helyesen 10^{-12}, de a gép szerint 0. Így tehát a program rossz irányban folytatódik.

Milyen hibák származnak ebből esetünkben? A törteknek mind a számlálója, mind a nevezője tartalmaz kivonást. A számlálóknál ez csak pontatlan eredményt idéz elő, a nevezőnél azonban akkor is „nincs megoldás" — vagyis a nevező zérus — választ kaphatunk, ha van. Ez a helyzet, ha számaink mondjuk azonosak az iménti példában szereplőkkel.

Van-e megoldás erre a problémára? Nincs. Korlátlan pontossággal nem lehet számolni, de nagyobbal igen. Léteznek a szokásosnál lényegesen nagyobb pontosságot adó módszerek. (Egy ilyet a lapban a későbbiekben ismertetünk.) Közös hátrányuk azonban, hogy időigényesek, ezért lényegesen növelik a gépi időt.

Összegezve az eddigieket: egy, a hagyományos számítási módszereinkkel megoldható, tiszta, egyszerű feladat gépre vitele bonyolult és ugyanakkor alapos megfontolást kíván. Nem lehet egyszerűen, gépiesen átvenni a klasszikus matematikai módszereket. Az ilyen feladatokat megoldó programokat gazdaságosabb készen megvenni, vagy ha lehet, átvenni, mint a program kidolgozásának nekilátni, mert az elkészítés igencsak munkaigényes.

Simonyi

OLVASTUNK...

Elektronok — elektronika

Az elektronika szóban egy elemi részecske neve húzódik meg: az *elektroné*. Miért játszik pont ez az elemi részecske olyan meghatározó szerepet mai számítógépeinkben, rádiókészülékeinkben, kvarcóráinkban, telefonközpontjainkban — felsorolni is lehetetlen, még hol mindenhol? Például miért nem *„protonikus"* vagy *„neutronikus"* szerkezetek terjedtek el világunkban? A válasz egyszerű. Ennek az elemi részecskének számunkra kitüntetetten *jó tulajdonságai* vannak:

— *tömege* igen kicsi ($\sim 9{,}1 \times 10^{-31}$kg). Ez jó, mert felgyorsításához — mozgatásához — nagyon kevés energia szükséges;

— kis tömege ellenére igen „erős" — elektromos-mágneses — *kölcsönhatásban* van környezetével (töltése $\sim -1{,}6 \times 10^{-19}$C). Emiatt jól érzékelhető, jól „terelhető", energiatárolásra alkalmas;

— sok olyan, a nekünk fontos hőmérséklet-tartományban, ahol gépeink működnek, szilárd halmazállapotú anyag van — ezeket nevezzük elektromos vezető anyagoknak —, melyben az elektronok nagy tömegben, „szabadon" (például a kristályszerkezethez nem kötötten) vannak jelen. És sok olyan anyag is — ezek szigetelők —, melyben ilyen „szabad" elektronok gyakorlatilag nincsenek. Emiatt az elektronok útvonala könnyen kijelölhető, olcsón, történetesen rézhuzalból, megépíthető;

— *stabil*, nem nyelődik el, mint akár a *foton*.

Amikor gépesíteni kívánunk valamit, nagyon fontos, hogy olyan munkaeszközünk legyen, amivel számtalan műveletet — például rengeteg gondolat feljegyzését — végezhetünk el igen gyorsan, igen kis energiával és teljesítménnyel. A legegyszerűbb, naponta gyakorolt formája a feljegyzésnek, hogy ceruzával strigulákat húzunk egy papírlapra. Az elektronikában ennek az felelhet meg, hogy egy mikroszkópikus méretű elektrontartályt (kondenzátort) feltöltünk, ma legalább mintegy ezer elektronnal. Ha a tartály megtelik: ott a strigula, ha a kondenzátort kisütjük: kiradíroztuk a strigulát. Elektronokkal dolgozva mindezt megtehetjük — ha úri kedvünk úgy kívánja — akár százmilliószor is másodpercenként.

Egy bitnyi információ (hogy ez mit jelent, arról a Magazin 1987/10. számában már írtunk) feljegyzése elektronokkal ma mintegy 10^{-16} joule energiát igényel, és még nem tartunk a fejlődési folyamat végén; az elméleti határ mai ismereteink szerint 10^{-21} joule nagyságrendnél van. Hogy ez technikailag mennyire közelíthető meg, az a jövő kérdése. Mindenesetre úgy látszik, van még jócskán lehetőség a fejlődésre.

Elektronika — erősáramú technika

Nem minden szerkezetet nevezünk *elektronikusnak*, ami elektronok tulajdonságain alapuló hatások kihasználásával működik. A határ az elektronikus és az ún. erősáramú szerkezetek között elmosódó, inkább gyakorlati, mint elméleti. Egy nagy teljesítményű villamos motort vagy áramfejlesztő generátort senkinek sem jutna eszébe *elektronikusnak* nevezni, holott működésükben kétségtelenül a mozgó elektronok játsszák a kulcsszerepet. Hasonlóképpen nem szokták elektronikusnak nevezni azokat a szerkezeteket, melyek *nagyobb teljesítmények* átalakítását szolgálják vagy melyekben *mozgó mechanikus* alkatrészek jelentős szerepet kapnak. Nem szokták például elektronikus szerkezeteknek tekinteni a mozgó alkatrészeket nélkülöző félvezető alapú átalakítókat sem, melyeket többek között számítóközpontok szünetmentes áramellátására alkalmaznak. (Ezek egy akkumulátortelep egyenfeszültségét alakítják át 220 V-os hálózati feszültséggé nagy teljesítményű félvezetők segítségével.) De — hogy egy ellenpéldát is idézzünk — már a nagy teljesítményű számítógépekbe beépített, ugyancsak nagy teljesítményű, ún. *tápegységeket*, melyek a hálózati feszültséget alakítják át nagy teljesítményű félvezető áramkörök segítségével a gépbe beépített integrált áramkörök tömegének energiaellátásához szükséges (ma tipikusan 5 V körüli) egyenfeszültséggé, azokat már elektronikus eszközöknek szoktuk tekinteni.

Szerepe van az elektronika és az erősáramú technika elhatárolásában az egyetemi, a főiskolai stb. oktatásnak és az ipar világszerte nagyjából egyöntetű, de sokszor esetleges szakosodásának is: külön képeznek például szakembereket „erősáramú" és külön „gyengeáramú" technikára; általában más profilú vállalatok gyártanak áramátalakítókat és mások számítógépekbe beépítésre szánt tápegységeket stb.

A furcsa határesetek számbavétele helyett ma elfogadható egyszerűsítéssel azt mondhatjuk: *elektronikus szerkezeteknek ma és még jó ideig azokat nevezhetjük, melyek működésük lényegét tekintve tipikusan félvezető alapú, integrált áramkörökből épülnek fel.* — A továbbiakban eszmefuttatásunkat a *digitális* elektronikára korlátozzuk. Hogy ez mit jelent, arról lapunk 1987/9. számában írtunk.

Integrált áramkörök — elektronikus szerkezetek

A *digitális*, félvezető alapú integrált áramkörök nagyon sok, mikroszkópikus méretű „elektrontartályt" — kondenzátort — és az elektronok mozgatását szabályozó, igen gyorsan vezérelhető (nyitható és csukható) kapcsolót, tranzisztort tartalmaznak. Ma néhány négyzetmilliméternyi területen, a chipen vagy morzsán néhány százezer vagy akár néhány millió tartály, kapcsoló helyezhető el úgy, hogy a tartályok és a kapcsolók akár a másodperc százmilliomod része alatt nyithatók-csukhatók, feltölthetők-kisüthetők. Egy-egy ilyen integrált áramkör üzemben tartása ma, a „szivárgások", „súrlódások", a csatlakoztatások stb. miatt, tizedwatt nagyságrendű teljesítményt igényel. (Ha egymillió mikroszkópikus méretű kondenzátort töltögetnénk és sütögetnénk ki másodpercenként egymilliószor, és kisüléskor hagynánk az energiát elveszni, akkor ehhez csupán 10^{-4} watt teljesítményre lenne szükség!)

A digitális elektronikus eszközök tervezése — nem túl nagy egyszerűsítéssel — nem más, mint sok millió tartály és vezérelhető kapcsoló célszerű összeköttetésének meghatározása. A mai átlagos feladataink megoldásához szükséges darabszámok azonban akkorák, hogy nagyon szakszerű és gondos *strukturálás* és az ennek megfelelő munkaszervezés-munkamegosztás bevezetése nélkül az összeköttetések tervezése megoldhatatlan lenne.

Az elektronikán belül szervezetileg is (külön vállalatok, külön szakképzés formájában stb.) elkülönülnek azok, akik a félvezető lapkán belüli összeköttetésekkel — mikroelektronikával — foglalkoznak, azoktól, akik *kész*, integrált áramkörökből építkeznek. Ennélfogva elkülönülnek tehát a szerelőipartól, az eszköztervezéstől, a berendezésgyártástól és általában a számítástechnikától stb.

A félvezető alapú integrált áramkörökből felépülő elektronikus szerkezetek (hardverek) készülhetnek valamilyen meghatározott célra és lehetnek „univerzálisak". A potenciálisan „univerzális" számítógépek egy-egy konkrét feladat elvégzésére programozhatók (szoftver). A programok maguk is gyakran olyan sok lépésből — utasításból, parancsból — tevődnek össze (megjegyzem, egy átlagos könyv hozzávetőlegesen 20 ezer sorból áll; vannak 10^5, 10^6 utasításból álló, sőt még ennél nagyobb programok is!), hogy komoly elméleti alapokon nyugvó strukturálás (mint a nyelvi, fordítói, operációs, „run-time", keret- stb. rendszerek) nélkül áttekinthetetlenek lennének. A mindezekkel való foglalkozásnak az a jelentősége, hogy az emberi gondolkodás, a megismerés, az absztrakció, a tanulás, az intelligencia, a nyelv új dimenzióit feszegeti, tárja fel. Mindez az elektronikus elven működő szerkezetekhez tapad, mindezt az elektronizáció részeként tartjuk számon. Valószínű, hogy épp ez a „tapadék" az, ami az elektronizációval kapcsolatban hatásaiban és feltételeiben leginkább tanulmányozásra érdemes.

Az elektronizáltság, mint a fejlettség fokmérője?

Napjainkban az országok egymás közötti nem hivatalos versenyében a sorrendet az elektronizáltság fokának megfelelően szokták meghatározni. Úgy tűnik, az elektronizáció nem pusztán technikai-technológiai, hanem fontos társadalmi és politikai kérdéssé is vált. Emiatt gyakran merül fel a kérdés, mérhető-e egy-egy országban az *elektronizáltság* foka.

Ha elfogadjuk a korábbi gondolatmenetet, akkor — mint előbb már megfogalmaztuk — célszerű kis egyszerűsítéssel azt mondhatjuk, hogy minden *elektronizáció*, ami alapvetően, lényegét tekintve félvezető alapú in-

és most először az elektronika területére invitáljuk önöket, majd hírt adunk egy amerikai felsőoktatással kapcsolatos könyvről. Mindenki tudja, hogy mai számítógépeink elektronikusak, de vajon hányan tudják azt is, mit jelent ez?

tegrált áramkörök beépítéséből áll, továbbá olyan eszközök felhasználásából, amikben ilyenek vannak.

Úgy tűnhet, ezek után a „verseny" kiértékeléséhez nem kellene mást csinálni, mint azt összeszámlálni — valamilyen ésszerű súlyozással —, hogy a vizsgált ország, népesség egy-egy tagjára hány millió elektron-„tartály+kapcsoló" jut. Minél több, annál elektronizáltabb a vizsgált ország, a vizsgált népesség. A súlyozás alapja lehetne például a maximális működési sebesség, a fajlagos energiaszükséglet, a megbízhatóság, a környezetállóság stb. Nagyobb súlyt kapnának esetleg a gyorsabb, magasabb hőmérsékleten működésre képes, megbízhatóbb integrált áramkörök. Feltételezhetjük, hogy az integrált áramkörök száma átlagban meghatározza a felhasználásukhoz szükséges *szoftver* mennyiségét is.

Mindez elméletileg szép, csakhogy amíg a hazánkban gyártott integrált áramköröket meg az importáltakat számításba lehet venni, ki fogja összeszámlálni a legkülönfélébb importált eszközökbe, szerelvényekbe — a hazai gyártók által megvásárolt ún. „produktív importba" — beépítetteket is? A mi külgazdasági nyitottságunknál ez nem elhanyagolható mennyiség. Továbbá: igaz-e, ha például egy gépkocsiba beépített mechanikus szerkezetű órát felvált egy elektronikus szerkezetű óra, akkor ez feltétlenül a fejlettség, előrehaladottság jelzője?!

Mindazon „gyanús" dolog közül, ami a világ arculatát valószínűleg (talán?) meg fogja változtatni — ha egyáltalán ki lehet ragadni egyet vagy néhányat a sok közül —, az nem *általában* az elektronizálás lesz, hanem annak egy része, következménye, *a szellemi munka át- avagy felértékelődése:*

— a szellemi munka gépesíthető (rutin-) részének gépesítése és ezáltal a munka humanizálása (?);

— a gondolkodás új területeinek meghódítása (?);

— az intelligencia, a tanulás, a nyelvek stb. mélyebb megértése.

Valószínű, hogy egy-egy ország elektronizáltsága szoros összefüggésben van az adott országban működő számítógépek számával, illetőleg e számítógépek *összteljesítményével.* Ha ez így lenne, akkor egy-egy ország, népesség elektronizáltsági fokának jellemzésére használhatnánk az egy főre jutó számítástechnikai teljesítmény mérőszámát. A különböző számítógépek teljesítményének közös nevezőre hozása — egy vagy néhány mérőszámmal való jellemzése — sem egyszerű feladat. Ebből a szempontból érdekes például a különféle, például a COCOM tilalmi listáknál alkalmazott számítás.

Most pedig következzék a híradás Allan Bloom „The Closing of the American Mind" (Simon and Schuster, 1987. 392 oldal) könyvéről.

Az amerikai gondolkodás beszűküléséről

szól Bloom professzor könyve, amihez a neves amerikai író, Saul Bellow írt előszót. Arról, hogy szerinte az amerikai felsőoktatás hogyan értette félre a demokráciát és miként szegényítette el a mai diákok lelkivilágát. A szerző, aki maga is tanított patinás, ún. ivy league egyetemeken, Saul Bellow szerint kiváló könyvet írt már Shakespeare politikájáról, lefordította Platón „Köztársaság"-át, Rousseau „Emile"-jét. Ilyen ajánlólevéllel nehéz lesz egy kézlegyintéssel elintézni súlyos, kritikai mondanivalóját. Olyan tekintélyek szólnak mellette, mint Szókratész, Platón, Arisztotelész, Machiavelli, Rousseau és Kant.

A könyv három részre tagolódik: *a diákokról,* a *nihilizmusról és az amerikai életstílusról,* végül az *egyetemről* szóló ra.

Rovatunk gyakorlatához híven, általános áttekintés helyett vegyünk inkább nagyító alá egy kisebb, jellemző részletet. A diákokról szóló első rész négy fejezetre, a negyedik pedig több alfejezetre oszlik: Tiszta lappal; Könyvek; Zene; Kapcsolatok: Egocentrikusság, Egyenlőség, Faji kérdések, Nemek közötti kapcsolat, Állandó kötődések hiánya, Válás, Szerelem, Erosz. Nézzük meg közelebbről a nemek közötti kapcsolatról szóló alfejezetet.

Itt a szerző arról ír, hogy a közhiedelemmel ellentétben Amerika nem valamiféle műveletlen, antiintellektuális ország, ahol az elméletek legfeljebb csak árra kellenek, hogy igazolják a célokhoz szükséges cselekedeteket. Éppen ellenkezőleg: Amerika olyan színpad, ahol a tragédiák és komédiák főszerepeit mindig is a *nagy elméletek* játszották. Amerika a szerző szerint filozófusok és tanítványaik országa. Minden cselekedetet a *szabadság* és *egyenlőség* elve vezérel.

Ez a két vezérelv most fellép a legmagánabb magánélet színpadára, hogy eljátssza utolsó főszerepei egyikét. A nemek kapcsolata — és minden, ami még hozzá tartozik: a szerelem, a házasság, a család — most ebben a végjátékban egy nagy nemzeti elgondolás, terv tárgya lett. De a természet itt úgy látszik, ellenáll az emberi törekvésnek, ellenáll annak, hogy az ember ide is kiterjessze a mindenható eszmék hatalmát.

A nemek közötti kapcsolatokat két hullámban érte kihívás jellegű változás. Az elsőt *szexuális forradalomnak* nevezhetjük, a második a *feminizmus.* Az elsőnek a *szabadság,* a másodiknak az *egyenlőség* volt a zászlajára írva. A kettő egy darabig jól megvolt egymással, de már Tocquville is megmondta: a szabadság és egyenlőség nehezen egyeztethető össze. A feministák harcot kezdtek a pornográfia ellen. A pornográfia védelmezői a szólásszabadság páncélzatát öltötték fel. A „forradalmi" feministák így furcsa mód hirtelen egy táborba kerültek a közösségi morált, a hagyományos nemi szerepeket védő *konzervatívokkal,* akik tabunak tekintettek volna minden kapcsolatot aközött, amit az emberek olvasnak vagy képeken látnak és aközött, amit egymással csinálnak. És e küzdelem hátterében ott álldogáltak, kezüket aggodalmasan tördelve a *liberálisok,* akik mindkét félnek szerettek volna igazat adni, de nem tudtak.

A szexuális kapcsolatok felszabadításáért indított mozgalom az ösztönöket fel akarta szabadítani a puritán bibliai örökség, az eredendő bűn eszméjének terhe alól. A hatvanas évek fokozatosan kitapogatták volna a szókimondás határait, de azok végül is egyszerűen szétolvadtak anélkül, hogy nagy feltűnés támadt volna ebből. A szülők és tanárok azon is elég hamar túlették magukat, hogy a diákok kezdtek együtt élni és hálni. Az erkölcsi gátlások, a betegségektől való félelem, a terhesség kockázata, megfelelő hely hiánya — szóval mindaz, ami korábban ennek útjában állt — egyszerre csak eltűnt. A diákok — különösen a lányok — egyszerre csak elkezdték nem szégyellni azt, hogy kimutassák szexuális érdeklődésüket, sikereiket. Az együttlakás, mely a húszas években még veszélyesnek, a harmincasnegyvenes években még jó esetben is legfeljebb bohémkodásnak minősült, egyszerre olyan természetessé kezdett válni, mint valami lánycserkész-csapatbeli tagság. Azért kell azt mondani, hogy „különösen a lányoknál", mert a tradicionális gondolkodás (hit?) szerint a fiúk mindig készen lettek volna habozás nélkül erre, és e szereposztás a fiatal nőktől várta volna el a szemérmes tartózkodást. Az első hullámban tehát ki kellett kapcsolni ezt a tradicionális női szemérmet. Mivel ezt a szemérmet pusztán valamiféle konvenciónak, szokásnak fogták fel, nem jelentett problémát a leküzdése. Ez az első hullám, a szabadság csak hangsúlyosabbá tette a nemek közötti különbséget: mind a férfiak, mind a nők hangsúlyosabban férfiak és nők szerettek volna lenni.

Ez a nagy felszabadulás azonnali boldogságot ígért, felszabadítást érzékeink ezeréves sötét elnyomása alól, feloldódást valamiféle nagy, folytonos bacchanáliában. De a ketrecbe zárt fenevadról hamarosan kiderült, hogy nem oroszlán, hanem nagyon is háziasított macska csupán. Ami a nagy felszabadítás során kiderült, az az, hogy az egész szenvedély nem is olyan veszedelmes, jobb zöld utat engedni neki, mint a lázadást elnyomni. Bloom professzor egyszer megkérdezte diákjait: hogy lehet az, hogy nemrég még a szülők rászóltak a kimaradós lányukra: „Nehogy nekem még egyszer leoltsd a bejárati lámpát!", most meg akkor sem szólnak, ha a fiuk barátjával alszik. Erre azt a feleletet kapta egy csinos, nagyon normális, fiatal aszszonykától, hogy: „Nagy dolog!" És ez az, ez a szenvedélynélküliség a legjellemzőbb eredménye az első „forradalmi hullámnak".

A második hullám, a feminizmus, a felszabadítás köntösében érkezett. Sokkal inkább a természeti kötöttségektől, mint a megszokástól való felszabadításaiban. Ezért komorabb, erotikamentes, valamiféle elvontabb társadalmi reformtervezet volt. Kevésbé állt szemben a fennálló törvényekkel. De, hogy mi a végső cél — mint Freud is utalt rá —, nem volt olyan világos.

A szóhasználat eltolódott az *„éljünk természetesen"*-től (amellyel legtöbbször nagyon is meghatározott testi funkciókra utaltak) olyasmire, mint „legyél saját magad", „teljesítsd ki magad", „magad határozd meg prioritásaidat", „alakítsd ki magad saját életstílusodat" stb. A nőmozgalom Bloom professzor könyve szerint nem az emberi természetre alapozott. Az alfejezet a továbbiakban elsősorban a feminista eszmék hatását boncolgatja igen tanulságos módon.

E rövid ízelítővel reméljük, sikerült olvasóink érdeklődését felkelteni a könyv iránt.

— KE —

PROGRAMTERMÉK

Zeneközelben – programmal!

Volt egy korszak, amikor a számítógépes zeneszerzést gúnyiratainkban emlegették, mint a rothadó kapitalizmus újabb, velejéig romlott ötletét. Azóta – a személyi számítógép eljövetelével – óriási néptömegek kaptak lehetőséget az önálló számítógépes zenélésre – még a keleti blokkban is! —, és személyesen győződhetnek meg a dolog értelméről vagy értelmetlenségéről. Azoknak a szerencséseknek, akik tehetik, nagy kedvük telik az akármilyen bárgyú zenei lehetőségekben is.

A Novotrade-től kapott egyik legújabb programcsomagban két zenepedagógus próbálkozik a zenetanítás számítógépes támogatásával. A gépi lehetőségek előnyeit és korlátait figyelembe véve, nem határozták el az énektanárok – és önmaguk? — lecserélését, sőt kompromisszumot kellett kötniük a géppel támogatható és az iskolai tantervben előírt ismeretek körére vonatkozóan is. Az elkészült oktatóprogram-csomag így „három és feledik" generációsnak tekinthető, amennyiben bizonyos mértékig támaszkodik az előírt tematikára, de ahol csak lehet, elrugaszkodik attól, és a számítógépes lehetőségeknek megfelelően átstrukturálja az átadandó ismereteket.

A program jellemzőit a szokásos, összefoglaló adatok táblázata mutatja.

A programcsomag kezelése nagyon egyszerű. A kazettákról a betöltés elég hosszú időt vesz igénybe a turbóbetöltő ellenére is, különösen, ha a szalag végén lévő programot kell betölteni. A problémán gyermekem segített egy jó trükkel. Elmondom, hátha más is tudja hasznosítani. Amikor a turbóbetöltő már elindult, akkor leállítjuk a kazettaolvasót. Kivesszük belőle a kazettát, és egy normál magnón gyorsan előrecsévéljük a szalagot a betölteni kívánt program előtti hangtalan szalagszakaszra. Ha szerencsénk van, a kazettát visszatehetjük a magnóba, és szinte azonnal töltődni kezd a keresett program (ha füllel sikerült eltalálni az üres szakaszt!). A programcsomagot hosszabb kazettákra írták fel, hiszen azokon három-öt program is található. Az igazság az, hogy ezzel a programcsomaggal tantermi órákon csak mágneslemezzel lehetne jól érvényesülni. A lemezegység azonban a C Plus/4-ek környékén fehér holló.

A program dokumentációja elég lakonikus, de sajnos ebben nemzetközi normákhoz igazodik. A programkazettákhoz általában csak annyi leírást mellékelnek, ami a csomagolóanyagon elfér. Ugyanakkor elmondható, hogy a programok öndokumentált jellegűek. Döntő többségében tudni lehet, hogy egy adott képen éppen mit lehet csinálni és mit nem. A borítón a szöveget sajnos írógéppel írták, ezért a hosszú ú és í betű helyett azok rövid párja szerepel.

A Ritmus I. kazetta a demón kívül az általános iskolai I—IV., a Ritmus II. kazetta az V—VII. osztályos ritmusanyagot tartalmazza. A Dallam I—II. nem kötődik közvetlenül évfolyamtantervekhez, hanem olyan zenei alapelvekhez, amelyek keresztül-kasul átszövik az általános iskolai tananyagot.

A szerzők ügyesen és játékosan oldották meg a ritmusok gyakoroltatását. A szóközbillentyű a C Plus/4-en talán elég stabil erre a célra. A ritmusokat meghallgatás után utánozni kell, méghozzá igen kényes pontossággal. Ez a pontossági igény már kezdő nehézségi foknál is igen erős. A Novotrade-nek emiatt sokan panaszkodtak, hogy nem jó a program. Ez nem így van. Követelményeket támaszt. A nehézségi fok egyébként beállítható, tehát a gyengébbek ritmusérzéke javítható, a tehetségesebbeké pedig tovább fokozható. Ezt nagyon helyes alapelvnek tekintem.

A programcsomagban sikerült elég szép képi megjelenítést alkalmazni. A kottarajzok igazán szépek, világosak és érthetők. A dallamgyakorlatokban egy helyen azonban nehezen jöttem rá, hogy az egynegyedes hangnak két kottafej hely kell a kottában. Mindenesetre ügyesen szervezték a kották bekérését, a „zenegrafikus inputot". A menüvezérlés egyszerű, de a célnak megfelelő. Nem kell, hogy zenepedagógusok ilyen témákban túl nagy erőfeszítést igénylő megoldást hozzanak.

Jó poén, hogy a mollhangsor-gyakorlatok közül a 9. éppen Beethoven IX. szimfóniájából az örömódát idézi. Emellett úgy érzem, hogy a szerzők a Bartók—Kodály hagyományok megőrzésére törekedtek, amennyiben a dallamanyag főként magyar népdalokra épül.

A gyakorlatok némelyike egy becsapós kérdés/válasz üzenettel lepett meg: folytathatom (i/n), de a nem választ ignorálta. Zárt hurokba kényszerít, azaz nem engedi, hogy kiszálljak a feladat megoldása nélkül, ami alapjában véve nem rossz elv. Ilyenkor én csak "i" választ írnék a programomban. A programcsomag drill módszerei egyébként jó eredményeket ígérnek, ha a tanuló elég szorgalmas. El tudom hinni, hogy a programmal olyan tanuló — vagy akár felnőtt! — ritmusérzéke és zenei készsége is hatékonyan javítható, aki tipikus zenei analfabétának ismeri magát. Már csak az a lehetőség is indukáló, hogy a tanórai stressz nélkül, nyugodt környezetben is lehet tanulni a programcsomaggal, és a haladási sebességet is kiki magához igazíthatja. Minimális szorgalom persze nélkülözhetetlen a sikerhez. Korszerű elemnek tartom azt a megoldást, hogy a program lehetőleg segíteni akar. Például bizonyos szituációban a tanuló kézbizonytalansága ellenére is addig tartja ki a hangot, amíg kellene.

A tartalom elemzésére nem vállalkozhatok, jóllehet énekkarokban énekeltem, de mikor volt az már. A rendelkezésre álló rövid idő alatt elég kevés modulba sikerült beletekinteni. Maguk a szerzők is kapacitálnak arra, hogy a számítógépes oktatás sohasem helyettesítheti a tanárt, hiszen a gép az éneklést már nem tudja visszaellenőrizni, mert nem hall (egyelőre!). A négykazettányi méret mégis azt sugallja, hogy a szakmabeli szerzők az elérhető maximumra törekedtek.

Én sem dicsekszem azzal, hogy már a programcsomag legelső moduljai is olyan nehéznek bizonyultak számomra, miszerint gyermekeim ügyességét kellett igénybe vennem, ha túl akartam jutni legalább egy-egy típusfeladaton. Mégis megerősítem, hogy ez csak azt mutatja, a programcsomag — nagyon helyesen — erőfeszítésre serkentő követelményeket állít a tanuló elé.

A program az általános iskolai munkát és bármely korosztály önképzését támogathatja, ami önmagában is igen jelentős teljesítmény.

Zsadányi Pál

ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Novotrade Rt.
Terméknév:	Ének-zene Ritmus I-II. Dallam I-II.
Szerzők:	Brusznyai Margit Polgár László
Géptípus:	C Plus/4
Hordozó:	négy kazetta
Dokumentáció:	két-két oldal a kazettaborítókon
Ár:	Ritmus I. 798 Ft Ritmus II. 798 Ft Dallam I. 610 Ft Dallam II. 498 Ft

MINŐSÍTŐ ADATOK

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

A figuratípusok jutalompontjai

Az értékelőfüggvény főbb összetevőit az eddigiekben már részletesen tárgyaltuk, sőt Dap Hartman ANALYSE programjának segítségével kielemzett nyolcszázharminckét mesterjátszma elemzésének eredményét is bemutattuk. A sakkprogramok fejlődése azonban nem áll meg, a sakkprogramozók mindig újabb szempontok figyelembevételével próbálnak javítani programjukon.

A híres Chess 4.5 sakkprogram értékelőfüggvénye az általunk is már jól ismert komponensek pontjain kívül minden figurának külön bónuszt adott, amelynek nagyságát számos tényező határozta meg. Ezek a bónuszok nagymértékben növelték a program játékerejét. Ezt az bizonyítja, hogy 1970-ben, amikor a Northwestern Egyetem diákjai — David Slate, Larry Atkin és Keit Gorlan — elkészültek sakkprogramjukkal, megnyerték vele Észak-Amerika első bajnokságát. Az ezt követő tíz évben is csak kétszer adták át első helyüket, először 1974-ben a kanadai Ribbit-programnak, majd 1978-ban a Belle-nek, amely később ugyanígy zsinórban nyerte meg Észak-Amerika bajnokságait.

Dap Hartman, akinek tanulmányából már többször idéztünk, feldolgozta a Chess 4.5 programban használt bónuszokat is. Ellenőrizte, hogy a figurák elhelyezkedésére adott jutalompontok milyen mértékben reálisak. Ehhez 832 mesterjátszmát használt fel, amelyekben minden lépés megtétele után kiszámította a jutalompontokat. Ezután kiszámította az összes játszmára vonatkoztatva az azonos lépésszámhoz tartozó bónuszok átlagát mindkét fél számára, amit a féllépésszám függvényében ábrázolt külön a nyertes és a vesztes játékosnál (a vezér bónuszának függvényét az *1. ábrán* láthatjuk).

Nézzük, hogyan alakulnak a bónuszok az egyes figuratípusokra! A vezérre nagyon egyszerű a jutalompont kiszámítása:

$Fq = 0{,}8 \cdot A - 8 \cdot D$, ahol

A: a vezér által megtámadott olyan mezőknek a száma, amelyeket nem támad ellenséges gyalog, futó vagy huszár.

D: a vezér és az ellenséges király sorkülönbségének és oszlopkülönbségének a minimuma.

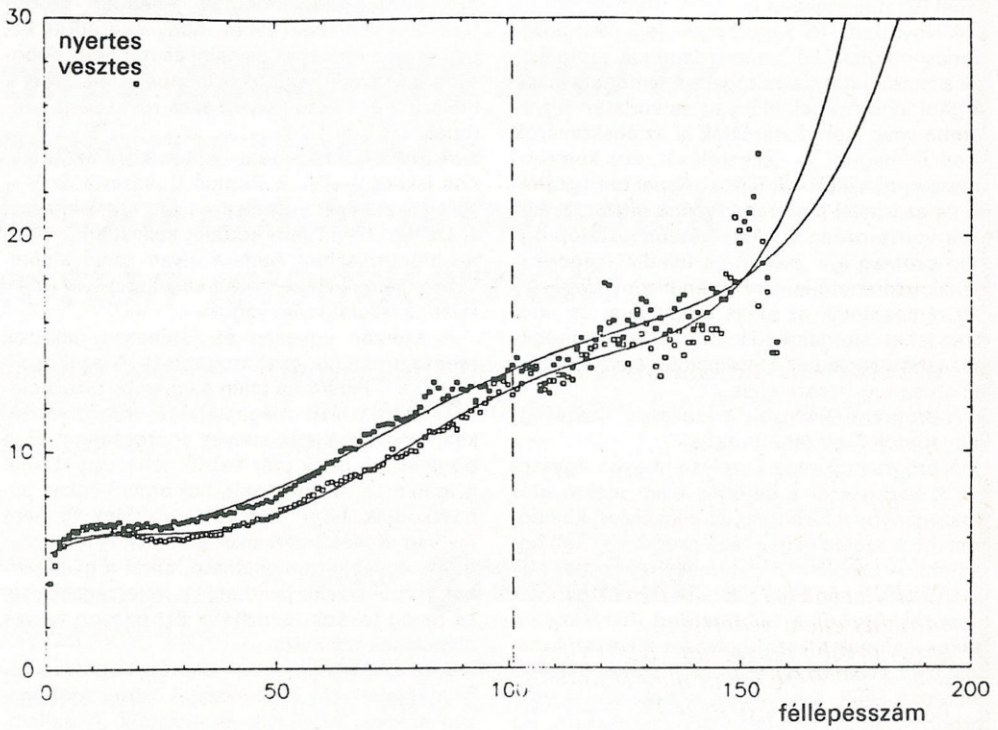
Dap Hartman ANALYSE nevű programja ennél is egyszerűbben számolt a nagymesterjátszmák kiértékelésénél, ugyanis a 0,8-as szorzót elhagyta, ezért a függvény a következőképpen módosult:

$Fq = A - D$, ahol

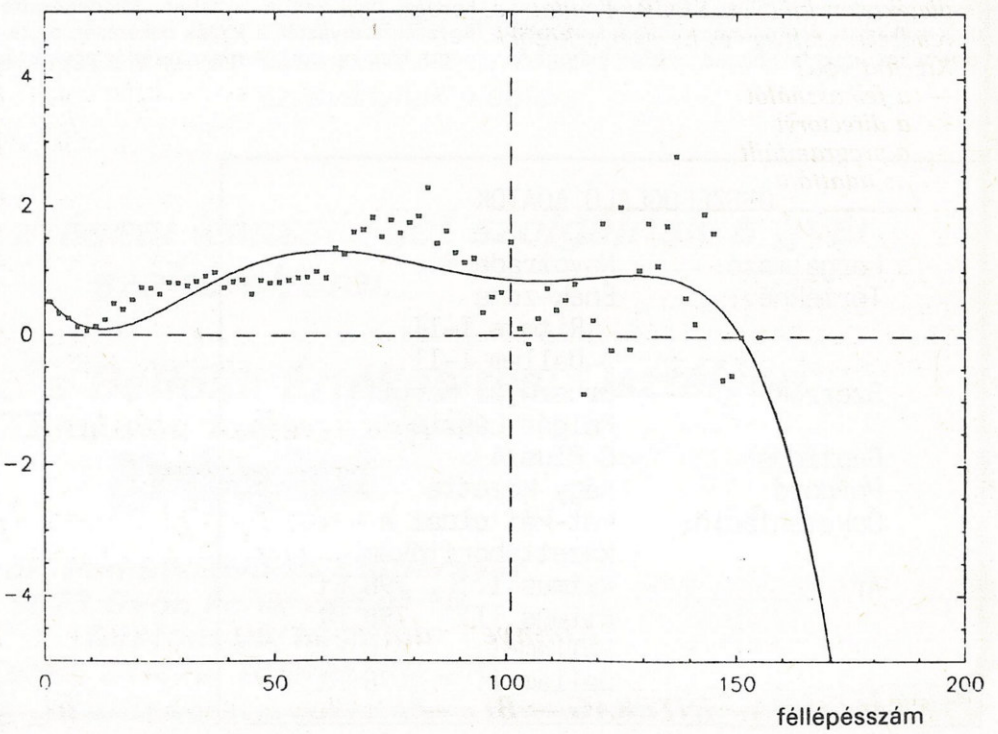
A és D jelentése azonos maradt az eredeti függvényével. Ezzel a módosított függvénnyel számítottuk ki a *2. ábrán* látható hadállás vezéreinek bónuszát, amely világosnál és sötétnél is kilenc. Világos vezér által megtámadott mezők: e2, e1, d1, c1, c2, b2, c3, b4, a5, d3, d4, d5, d6, d7, d8, e3, f4, g5, h6. Ezek közül figyelmen kívül kell hagyni azokat, amelyeket az ellenfél gyalogjával, futójával vagy huszárjával támad. Ezek a következők: b4, a5, d4, d5, d7, d8, h6. Tehát összesen tizenkét olyan mező van, amelyik megfelel az elvárásoknak, ezért erre 12 pontot adunk. Ebből viszont le kell vonni a vezér és az ellenséges király oszlopkülönbségének és sorkülönbségének minimumát. Világos vezér d2-n, sötét király g8-on helyezkedik el, ezért az oszlopkülönbségük a kisebb, mégpedig három. Tehát a végső pontszám a $12 - 3$, azaz 9. Ha a sötét vezért vizsgáljuk hasonló módon, szintén kilencet kapunk.

Kovács P. Attila

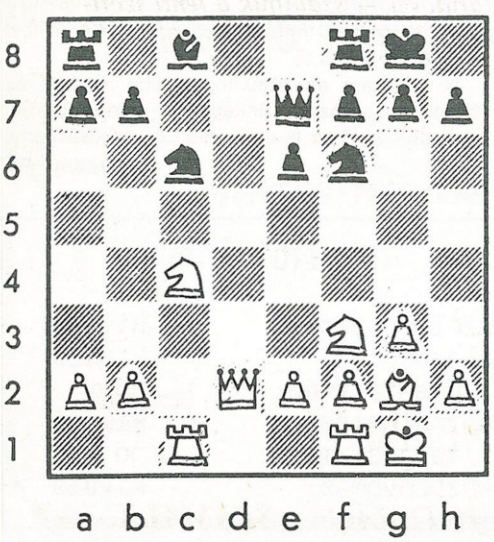
Vezérbónusz **1. ábra**



A nyertes és a vesztes vezérbónuszának különbsége **2. ábra**



Kaszparov—Petroszjan, Bugojno 1982.
A 24. féllépés utáni hadállás



Könnyen választhat a SZÁMALK MENÜ-jéből

A SZÁMALK értesíti az érdeklődőket, hogy a MENEDZSER-üzletága, a SZÁMALK—MENÜ az alábbi termékeket ajánlja.
A MENÜ nagy- és demigrosz-kereskedelmi tevékenységet folytat, főként a professzionális számítástechnika területén (8 bit felett).

Megrendeléseket veszünk fel:
- *a legújabb, komplett külföldi szoftverekre*
- *hardverkiegészítőkre*
- *hardverre*
- *CAD-rendszerekre*
- *tanácsadásra*
- *szervezésre, programozásra*

A SZÁMALK—MENÜ a MICROSOFT magyarországi disztributora: minden MICROSOFT szoftvert és Updateet 3 héten belül szállít írásos megrendelésre. Ajánlatkérésre postafordultával válaszolunk.

Termelőktől és más forgalmazóktól továbbforgalmazásra átveszünk jól dokumentált szoftvereket. Szükség esetén megszervezzük az adaptálást.

MAGYAR TERMÉKEINK

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

SZÜNETMENTES TÁPEGYSÉG *600 VA-es (AKKU-párolgás elleni védettséggel). 3 órás áramkimaradás mellett is tovább dolgozhat IBM PC/XT, AT kompatibilis gépén. A BITBOLT-ban megtekinthető: 1 db AT, 1 db XT és FX 1000. Ára: 120 ezer Ft.*

SECRET
Védi dokumentumait IBM PC/86-286-386 komputerén az illetéktelenektől. A SECRET adat- és programvédelmi rendszer, szoftver- és hardvereszközökkel.
Kit, mit véd?
- *a felhasználót*
- *a directoryt*
- *a programfájlt*
- *az adatfájlt*

Mivel?
- *passworddel*
- *kóddal*
- *hardverkártyával (opció)*
- *EPROM-kulccsal (opció)*

Hogyan?
- *a dekódolhatatlan password-kód kombinációjával*
- *hozzáférés előtt illetékesnek nyit, utána rögtön zár a védelem (a sebességcsökkenés maximum 10%)*
- *hardveres opció esetén a floppy-boot letiltásával*
- *törölhetetlen fájlokkal*
- *átnevezhetetlen fájlokkal*
- *lemásolt védett fájlok illetéktelenek által nem használhatók*

Kitől?
- *Más illetéktelen felhasználótól.*
- *Szükség esetén a systemmenedzsertől is.*

Hol?
- *Egy gépen.*
- *Hálózatba épített több gépen is.*

Meghibásodáskor:
- *A systemmenedzser hardverúton felélesztheti a rendszert. Kérje árajánlatunkat szoftverre, kártyára, kulcsra!*

ELTGUARD szoftver: programvédelmi rendszer szoftverúton. Hierarchikus programvédelem. Korlátozott felhasználói hozzáférés biztosítása programrendszeren belül. Menüvezérlés. Nem csak egy gépen használható. Több mint két éve feltörhetetlen! Ára: 120 ezer Ft. (Adatvédelmi kiegészítő elkészülte után térítésmentesen a korábbi vevőknek.)

MEDANIN
Orvosok, klinikusok és farmakológusok gyakorlati munkáját segíti az adattárolásban és analízisnél. Válaszok biológiai, diagnosztikus és terápiás kérdésekben. MEDikusok ANalízise és INformációs szoftvere a MEDANIN. Készítette prof. dr. Berentei és prof. dr. Kékesi. Ára: 100 ezer Ft.

KIR
Winchesterén és floppyjain tárolt fájlokból directory csoportosításban saját katalógusfájlt készíthet. Megjegyzésekkel elláthatja, csoportosíthatja, dBASE típusú katalógusfájlját tovább feldolgozhatja. Ára: 15 ezer Ft.

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

Kérésre **árkatalógust** *küldünk több mint 1500 legális szoftverre. Szükség esetén tanácsadás.*

A MENÜ iroda rendelési címe:
1123 Budapest, Kapitány u. 6. I/1.
Telefonszám: 565-419
A BITBOLT címe: 1136 Raoul Wallenberg u. 5.
Telefonszám: 110-983

Kérjen levélben ajánlatot, árkatalógust!
Válaszunk után írásban rendeljen!
Rövid szállítási határidő

Tisztelettel a SZÁMALK—MENÜ ÜZLETÁG

SZÁMALK

Könnyen választhat a SZÁMALK MENÜ-jéből!

KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

Marchant, J. J.
Számítógépes műszaki tervezés.
Az előkészítés gyakorlata
(Budapest, 1988.
Műszaki Könyvkiadó, 144 oldal.
Ára: 85,— Ft.)

A digitális számítógépek, valamint grafikus be- és kimeneti eszközeik új korszakot nyitottak a fizikailag nem létező objektumok szemléltetésében. Ilyen objektumok a fejlesztési folyamat kezdeti szakaszában még csak a tervezőmérnök fejében létező műszaki alkotások, amelyeknek megvalósítási módjait papírra írják. A digitális számítógépes rendszerek alkalmazásával a mérnöki műszaki megoldásokat számítógépes belső ábrázolású modellként lehet szemléltetni. A tervező és a gép közötti kommunikáció eszközei: a billentyűzetek, a digitalizáló táblák, a fénytollak, a képernyős megjelenítő eszközök és a számjegyvezérlésű rajzolóberendezések. A digitális számítógépes eszközöket alkalmazó termékfejlesztési folyamat egészét nevezik számítógéppel segített tervezésnek (CAD= Computer Aided Design).

A CAD alkalmazásának előnyei közé tartozik, hogy a tervezői elképzelések szemléltetéséhez nemcsak a klasszikus grafikai alapelemeket — a pontot és a vonalat — lehet felhasználni, hanem felületeket, térfogatokat vagy akár alkatrészeket és szerkezeti egységeket is. A hatékony számítási eljárások — például a végeselemes módszer — lehetővé teszik bonyolult rendszerek viselkedésének számszerű vizsgálatát. Az adatbázisrendszerek megkönnyítik a meglévő megoldások visszakeresését.

A CAD rendszer fejlesztőjének ki kell választania a piacon található nagyszámú rendszerből a számára legmegfelelőbbet, amelynek egyúttal illeszkednie kell a megvalósítandó, számítógéppel integrált gyártórendszeréhez.

Ez a könyv a viszonylag új keletű, CAD-hez kötődő ismeretekben hiánypótló mű, egyúttal tanácsokkal segíti a saját CAD-alkalmazások kialakítását.

Kernighan, B. W.—Pike, R.:
A UNIX operációs rendszer
(Budapest, 1988.
Műszaki Könyvkiadó, 362 oldal.
Ára: 298,— Ft.)

A UNIX a legszűkebb értelemben egy időosztásos operációs rendszer magja, olyan program, amely vezérli a számítógép erőforrásait, és elosztja azokat a felhasználók között. Lehetővé teszi, hogy a felhasználók a programokat futtathassák, vezérli a géphez kapcsolt perifériákat, fájlrendszere van az adatok, a programok és a dokumentumok tárolására.

Tágabb értelemben nemcsak a magot nevezik UNIX-nak, de beleértik az olyan fontos programokat is, mint a fordítókat, szövegszerkesztőket, parancsnyelveket, a fájlokat másoló és kinyomtató programokat. Sikerének titka az, hogy C nyelven íródott, a mikroszámítógépektől kezdve bármely kiépítésű számítógépen futtatható, s másrészt a hozzáférhető és magas szintű nyelven íródott forráskód egyszerűvé teszi a rendszernek még a különleges követelményekhez való illesztését is. Végül a UNIX jó operációs rendszer, különösen programozóknak.

A könyv a UNIX filozófiájával kívánja megismertetni az olvasót. Mivel e filozófia a programok kapcsolatára épül, a különböző fejezetek elsősorban az önálló eszközöket vizsgálják, tárgyalják a programok összekapcsolásának kérdéseit és azt is, hogyan lehet a programokat újabb programok építésére használni.

A könyv nem referenciakönyv, ezért főleg a UNIX alkalmazási stílusára tanít. További információk megszerzéséhez a UNIX Programozói Kézikönyv szükséges.

Honerkamp, M.—Jetter, M.:
Repülés mikroszámítógéppel
(Spectrum, Commodore 64,
Apple II, IBM PC)
(Budapest, 1988.
Műszaki Könyvkiadó, 212 oldal.
Ára: 98,— Ft.)

„Ennél csak a valóságos repülés szebb!" — írhatják a szerzők könyvük előszavában, hiszen szinte minden olyan, repüléssel kapcsolatos tudnivalót sorra vesznek, melyeknek számítógépes szimulálásával a képernyő előtt ülők egy igazi pilótafülkében érezhetik magukat. A kötet a „Flight Simulator II"-vel való repülést ismerteti. A bemutatott repülési manőverek és szabvány repülési eljárások minden más repülésszimulációs programra is alkalmazhatók.

A könyv célja, hogy betekintést nyújtson a repülésbe, és egyúttal élvezetesebbé tegye a repülésszimulátorral való szórakozást. Minden fejezete a szimulációs program, illetve a repülés egy-egy részfejezetét tárgyalja. Ha az olvasó már gyakorlott repülő, néhány fejezet átlapozásával mindjárt a vakrepülésre térhet át. Az élvezetes repüléshez a szükséges repülőtérképeket is megtalálja a kötet végén.

Lajstromolt robotok

A MTESZ-ben működő Magyar Robottechnikai Társaság a KSH segítségével megvizsgálta, hogy hazánkban hány robot dolgozik, s ezek milyen munkát végeznek. A gépiparban 26 vállalat vásárolt robotot. Ezek közül hét különféle gépeket, hat járműveket, öt villamos gépeket, három fémtömegcikkeket, illetve híradástechnikai eszközöket, négy műszereket, egy pedig vaskohászati termékeket gyárt. A robotokat csaknem húszféle munkára alkalmazzák: a legtöbbet ívhegesztésre, különféle szerszámgépek kiszolgálására, présöntésre, továbbá anyagok felületkezelésére. Használnak még robotokat öntvények tisztítására, anyagok lézeres megmunkálására, illetve mechanikai és elektronikai szerelésre.

A Magyar Robottechnikai Társaság az adatok alapján az érdeklődőket azonnal informálni tudja arról, hogy egy-egy speciális robot melyik vállalatnál dolgozik. A tájékoztatást megkapta a társaság 35 tagvállalata s 350 egyéni tagja. A Magyar Robottechnikai Társaság vállalta, hogy ezentúl folyamatosan gyűjt információkat a működő ipari robotokról.

Fakultáció

Lentiben, a 2-es Számú Általános Iskolában 1985-ben számítástechnikai fakultáció indult, ehhez korszerű szaktantermet is kialakítottak. A célnak megfelelő, praktikus pultrendszeren tizenegy Commodore 16 és két Commodore Plus/4 típusú személyi számítógép várja a tanulókat. Az egyik gépet a tanár használja. A központi gép összeköttetésben áll egy „elektronikus táblával" — azaz egy színes televízióval —, így a diákok nyomon követhetik a tanári modellprogramokat és magyarázatokat. A tanulók Junoszty típusú monitoron látják programjukat, s ellenőrzik az esetleges hibákat.

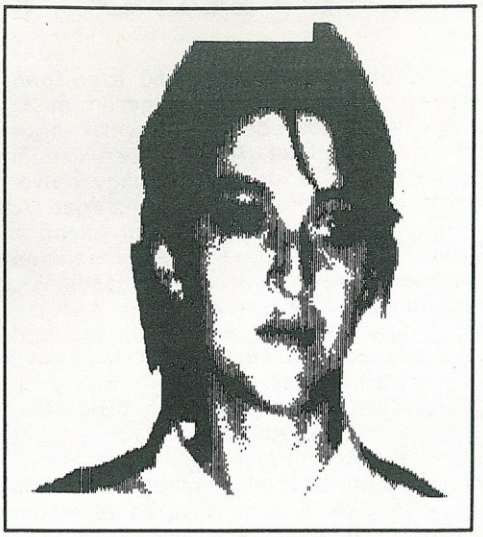
A hetedik és a nyolcadik osztályosok közül 27 tanuló heti két-két órában sajátítja el a számítástechnika alapjait. A számítástechnikai oktatást vezető Fracz József matematika—technika szakos tanár elsősorban a diákok algoritmusos gondolkodását fejleszti. És a foglalkozásokon megismerkedhetnek a gyerekek a számítástechnika alapfogásaival és a személyi számítógép működési elveivel is.

A szaktanterem kialakításában a szülők áldozatvállalásán kívül meghatározó szerepet játszott az ÉPFA, a Fém- és Faipari Szövetkezet, a költségvetési üzem, valamint a megyei és a városi tanács. Együtt több mint félmillió forintot adtak a vállalkozáshoz.

HÍREK — ÉRDEKESSÉGEK

Tárolható a kész kép

A tizenötezer dolgozót foglalkoztató leningrádi Lenelektronmas tudományos-termelő egyesülés a közelmúltban már IBM PC-vel kompatibilis személyi számítógépeket is gyárt. Az Iszkra 1030M típusú gépéhez videoképet feldolgozó programot fejlesztett ki. A videokamerával rögzített képet a számítógép segítségével átalakíthatjuk, retusálhatjuk, módosíthatjuk, sőt feliratozhatjuk is. A képet végső formájában mátrixnyomtatóval jeleníthetjük meg. Ugyanakkor lehetőség van a digitális formájú képnek a mágneses adathordozón való tárolására is.



Az Iszkra 1030M-mel készített, feliratozott portré

ADOK—VESZEK —CSERÉLEK

C64 programokat eladok, lemezen és kazettán. 1988-as programokat cserélek. Micsik Gábor, Szeged, Szent Miklós u. 3. Tel.: (62) 27-771

C128, 1570-es floppy, magnó, 2 db joystick, 20 diszk a legújabb programokkal, sok szakkönyvvel kedvező áron eladó. Antal Attila, Győr, Zöld u. 25. 9028, Tel.: (96) 24-735

Commodore 128D (3 elsődleges üzemmód: C128, C64, CP/M; beépített 1571-es floppy) és 120 db diszk a legújabb programokkal olcsón eladó. Tegyen árajánlatot! Radics Róbert, Kiskunfélegyháza, Szőlő u. 10/4. 6100

Commodore datasett adatmagnó (3000 Ft) és 32 kbájtos cartridge (2200 Ft) kifogástalan állapotban eladók. Muzsik József 587-835 este

Eladó: 31 BIT-LET, 55 Mikrovilág (programlistával), 40 Mikromagazin, 3 könyv + prospektusok. 1000 Ft + utánvétel. Tóth József, Hajdúböszörmény, Tyereskova u. 22. 4220

Enterprise programokat adok és cserélek. Széles programválaszték. Válaszboríték ellenében lista! Cím: Sándor József, Bonyhád, Bezerédi u. 41. 3/5. 7150

Enterprise (128 k) számítógép magnóval, programokkal, kézikönyvekkel 10 000 Ft-ért eladó. Szarvas Imre, Dunakeszi, Jászai M. u. 20. 2120

Fordításokat készítek a **64'er Magazinból**. Több mint 3000 oldalnyi kész anyag! Kedvező árak. Textomat+, GEOS 2.0, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Budapest, Pf.: 400. 1446. Válaszborítékot kérünk!

Hardver, szoftver ötletek, programok cseréjének közvetítése. SYS-TEAM, Kecskemét, Pf.: 57. 6001

L.C.M. - THE NAME OF QUALITY

L.C.M. - a legjobb és leggyorsabb ellátó, programújdonságokkal:

- Commodore 64 (diszk/kazetta): kb. 150 program havonta
- Amiga: több mint 30 diszk a legújabb programokkal havonta
- Atari ST: több mint 20 diszk és több mint 30 program havonta

L.C.M. biztosítja az Ön ellátását a legújabb programokkal a felsorolt géptípusokhoz. Levelezek angolul vagy magyarul: Milosevic Slobodan, 19000 Zajecar, Naselje Avnoj C1/I-39, Jugoszlávia, Tel.: 019-21010

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

PRIMO-sok figyelem! Eladók Primóhoz hardverek és programkazetták! Több mint 1000 program! Kívánságra részletes listát küldök, válaszborítékot kérek! Varsányi Gábor, Nagyatád, Aradi u. 1/IX/C. 7500

Seikosha GP 700/A 8 színű grafikus mátrixprinter olcsón eladó. Érdeklődni lehet nappal a 418-166, este a 312-183-as telefonon Kovács Lászlónál.

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

Spectrum+ (80 k), interfész II., magnó, 15 kazetta, 10 könyv eladó. Új 3,5" lemezek (Real data) eladók. Érdeklődni: Polgár József, Kecskemét, Balaton u. 20. fsz. 30. 6000. Tel.: (76) 47-269

ZX-Spectrumra programok eladók! Mindig a legújabb szoftverek! Ingyenes katalógus! Pilláry Gábor, Pécs, Bajcsy-Zs. u. 4. 3/9. 7622

ZX-Spectrum+ 48 k-s számítógép, Kempston interfésszel, szakkönyvekkel, 300 db felhasználói és 600 db játékprogrammal együtt 15 000 Ft-ért eladó. Kápolnási János, Veszprém, Felszabadulás u. 63/A. 6/38. 8200

Z80 alapú CP/M op.r. mikrogépet dual 5'25" floppyval (800 kbájt/diszk) Philips monitorral (igény esetén eprom programozó adapterrel) és CP/M szoftverekkel olcsón eladó. Az érdeklődéseket nappal a 418-166, este a 312-183-as telefonon Kovács László várja.

Tudja Ön, hogy milyen egyszerű a FASTLOAD CARTRIDGE-el dolgozni? Vállalom bővített karakterkészlet, Reset gomb, IRQ (pause) kapcsoló beépítését, Seikosha SP1200VC - 180 VC nyomtatók magyarosítását, FASTLOAD, SPEEDTAPE programsegítő modulok készítését. Szívesen küldök tájékoztatót! Bártfai Barnabás, Budapest, Móricz Zs. u. 22. 1193. Tel.: 668-411/64 (délelőtt).

VESZEK

Sinclair ZX81 számítógépemhez billentyűzetet keresek. Siófok, Somlói u. 1. 8600. Tel.: (84) 12-431

CSERÉLEK

C16, C Plus/4, C64 programokat cserélek. Listát kérek. Lékó Zoltán, Szolnok, Dr. Csanádi krt. 8. VI/18. 5000

Commodore 16-os, 64-es, 128-as géppel rendelkezőkkel kapcsolatot keresek, hétezer programom van. Amigához 700 lemeznyi programom van. Gyermán Sándor, 23000 Zrenjanin Rade Koncara 23/V. Jugoszlávia

C Plus/4 felhasználói programokat cserélek, kizárólag lemezen. Kálmán Albert, Eger, Rákóczi u. 31. III. 11. 3300

C64-es programokat cserélek kazettán. Budapest, Országbíró u. 4. IX. 38. 1139

C64-es programokat cserélek lemezen. Minőségi választék, leírások is érdekelnek. Válaszokat listával kérném, én is küldök. Péli Gábor, Budapest, Tomori u. 10. 1131. Tel.: 496-032

C64 szuper programokat cserélek kazettán. Listát kérek! Purnhauser Pál, Tárnok, Templom u. 63. 2461

C128 programokért C64 programokat adok. Ugyanitt C64 programok cseréje, lemezen. Kis Bogdán Zsolt, Nagypeterd, Kossuth u. 126. 7912

Enterprise programokat cserélek kazettán. Sántha József, Budapest, Vadaskerti u. 8. fsz. 2. 1021

MSX rendszerű számítógéphez programokat cserélek, néhányat eladok. Jancsurák István, Miskolc, Dráva u. 7. 3528. Tel.: (46) 12-634

Primo programcsere! Több, mint 1000 program! Csak levélben! Listát kérek! Vajai János, Budapest, Nagytétényi u. 234. J. 2/8. 1225

ZX-Spectrum programokat cserélek. Listát kérek! Székely Gyula, Debrecen, Apafa 14. Pf.: 2649. 4002

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

Az Olvasó írja

Úgy látszik, hogy folyamatos invitálásra egyre több olyan olvasói levelet kapunk, ami nemcsak dicsér vagy elmarasztal, de fontos információkat is közöl. Nem titkolt vágyunk, hogy minél több olvasót bevonjunk az újságírásba, hiszen ha sokan írjuk a lapot, akkor valószínűleg gazdagabb lesz, mint ha csak a szerkesztők írásait közöljük. Lássuk a leveleket.

Vass Tamás

Lapjuk rendszeres olvasója vagyok. Főleg az Adok — veszek — cserélek rovatot böngészem. Most mintegy 130 programom van, ebből csaknem 50 programot csere útján szereztem. Akiknek eddig írtam, azok közül ketten válaszoltak tizenötből. Ha a szerkesztőséget érdekli, szívesen rendelkezésükre bocsátok POKE-okat, valamint programokat. Az októberi számban örömmel olvastam azt a Bioritmus nevű programot, ami C Plus/4-re készült. Be is írtam, de a program hibás volt! Ha a születési dátumot 1 nappal megváltoztatjuk és utána megjelenítjük az eredményt, akkor az a csík, amely a mai napot jelzi, 5 nappal ugrik odébb, nem eggyel. Hasonló hiba volt az augusztusi számban közölt Analóg órával. Ezt a programot javítottam, s ha érdekli önöket, elküldöm. A számlapra négyféle variációt készítettem. Kérem, írjon olyan címeket nekem, amelyekre ha írok, cserélhetek C16, illetve C Plus/4-es programokat vagy vehetek.

Sajnos néha előfordul, hogy hibás programot adunk közre, ezért szívesen fogadjuk és közöljük is olvasóink észrevételeit. Ami a kérését illeti, nem gyűjtjük a címeket, de szívesen közöljük a cserére szóló felhívását, ám ehhez ismernünk kellene a postai címét, hogy azt az újságban közreadjuk.

Váraljai Miklós, Budapest

Immár hat éve vagyok olvasója a lapnak, és szinte csak jót tudok mondani róla. Míg az első számok cikkei még inkább csak a figyelemfelkeltésre irányultak, az újság mostani felépítése, hírgazdagsága egészen elsőrendű. Nagyon jónak találom például, hogy a játékprogramok rovat megszűnt (?), ugyanis az ilyen, esetleg féloldalas játékprogramok színvonala csakis sokkal alacsonyabb lehet, mint azoké a programoké, amiket bárki megszerezhet. Az a játékprogram, amit valaki több órai rabszolgamunkával bepötyög (nem beszélve a hibákról) és utána hamar, néhány órán belül megun, nem sokat fejleszti az illető programozási képességeit. Talán ezután majd gépelni tud jobban. Viszont nagyszerűnek tartom, hogy olyan programok használatát ismertetik, amelyek általában elterjedtek, és használatukra igény is lenne. (A kalózmásolás hátránya, hogy nincs leírás.) Baj, hogy csak nagyon nehezen lehet beszerezni az ilyen leírásokat. És ráadásul még olyanok is vannak, akik ezekért horribilis összegeket kérnek, amit néha a felhasználó jobb híján meg is fizet. Az ilyen kalózmásolatokról csak annyit, hogy nem mindig a másolókat kellene szidni. Úgy öt éve az egyik termelőszövetkezet az újonnan vásárolt C64-hez vett egy programot (állami nagyáruházban), mintegy 60 ezer forintért. A programot volt szerencsém átnézni: teljes egészében megegyezik a Simon's BASIC-kel. A különbség csupán annyi, hogy az utasítások magyar nyelvűek és ékezetes a karakterkészlet. Az ilyen programokat miért nem ellenőrizzük? Különben pedig a jelenlegi árak mellett nem csoda, ha inkább másolnak a számítógép-tulajdonosok.

E folyamat ellen szerintem ma már nem sokat tehetnek. Úgy vélem, ma nincs olyan számítógép-tulajdonos, akinek ne lenne ilyen másolt programja. Visszatérve az újságra, nagyon örvendetes, hogy elkezdtek a számítógép egyéb felhasználási lehetőségeivel is alaposabban foglalkozni, gondolok itt a mérésre és vezérlésre vagy az irányításra. Bizonyára ez olyan téma, amelyről nem írhatnak elégszer, és ennél a témakörnél nem szabad sajnálni a magyarázatokat, mert ha valaki valamit a gyakorlatban ront el, annak akár a gép „életébe" kerülő következményei is lehetnek.

Végül lenne egy kérésem. Augusztus óta élvezem a Magyar Néphadsereg vendégszeretetét, és abban az első hónapi nagy kapkodásban lemaradtam a szeptemberi számról. Nagyon szeretném, ha megmondaná, hol lehet beszerezni ezt, mivel eddig az összes lapszám megvan.

A leírtakhoz nincs sok hozzáfűznivalóm, a véleményünk egyezik. Ami a régebbi példányok vásárlását illeti, forduljon bizalommal az NJSZT titkárságához (Bp. V., Báthori u. 16.).

Baráz Tamás, Maglód

Mint tudják, a Sinclair gépgyárnak sok terméke van a ZX81-en és a Spectrumokon kívül, ha azok az enyhén szólva hiányos magyarországi dokumentáció és fölhajtás híján nem is lettek olyan népszerűek, mint az előbbiek (C64, C16 stb.). A Sinclair QL szinte az összes gépet lepipálja magyar—angol karakterkészletével, Super BASIC programozási nyelvével, melyben beépítve megtalálható a föl, le, jobbra, balra scroll rutin (1 parancs!), belső óra, melyet egy paranccsal így ír ki például: 1988 JAN 1. 12:21:30, maradékszámoló függvény, beépített fill (kitöltés), beépített teknősbéka-grafika, számítógép-hálózatra csatlakozás, külön vezérelt ablakok, két beépített lemezmeghajtó, 512 × 256-os felbontás, 256 szín, külön QL monitor, átszínezés, körök, ellipszisek, görbék, vonalak relatív és „sima" koordinátarendszerben, kivillantásos, villámgyors grafika, és hangja is van, de még milyen!

12-féle csatlakozási lehetőség, interfész nélkül! Fájlkezelés, 111-120 különféle utasítás, újrasorszámozás, kilépés gépi kódban, kényszerelv, ROM cartridge csatlakozó (ez az egy nem BASIC). Mindez BASIC!

Ezenkívül a gép kapható 128 k, több Mbájtos kiszereléssel winchesterekkel, floppykkal, ingyen szövegszerkesztő diagram, archívum, költségvetés-táblázatok. Mindezt annyiért, mint a C64. Hát nem csodálatos? És ezt nem ismerik! Erről nem ír senki? Írjanak róla valamit! Vagy ezt a levelet közöljék, ha mód van rá! Sok programot tudok adni, amit közölni lehet, csak előbb szerzek egy nyomtatót.

Igazán élveztem a Sinclair QL-ről írt sorait, de erről nekem az a véleményem, hogy „kinek a pap, kinek a papné". De félretéve a humorizálást, nem tudom, mennyi QL van az országban, mindenesetre hozzánk gyakorlatilag nem érkezik QL program, amit közölhetnénk. Ha tud programot adni, és ezek eredeti, saját maga által készített programok és érdekesek is, akkor nem hiszem, hogy ne közölnénk belőlük néhányat.

Patek Alajos, Budapest

Nemrégen levélben ismertettem önökkel a Novotrade FINE PEN című kazettájával kapcsolatos rossz tapasztalataimat, és valamilyen segítségüket kértem. Aztán telefonon felhívott a Novotrade-től Molnár Attila, és — elismerve, hogy a program kezelési utasítása nem kielégítő — ígéretet tett, hogy egy héten belül megküldik részemre a részletes használati utasítást. Úgy gondolom, kötelességem ezt önökkel is közölni, miután bízom abban, hogy az ügy ezzel lezárul.

Köszönjük az értesítést.

Varga Péter, Budapest

Nem szokásom újságoknak levelet írni, de ha már úgyis küldöm a megrendelőt, akkor fűzök hozzá egy kis dicséretet és miegymást is. Az első szám óta rendszeres olvasója vagyok az újságnak, talán nem is volt olyan lapszám, amit elmulasztottam volna. Majdnem mindegyikben találtam valamit, ami számomra is érdekes volt, bár bevallom, hogy engem egyáltalán nem érdekelnek a játékprogramok és a csak mikrogépeken hasznosítható dolgok, lévén, hogy kezdettől fogva IBM PC gépen volt módom dolgozni. Viszont nagyon tetszik, hogy önök sok olyan cikket közölnek, amelyek nem egy géptípusra való programok, hanem eljárások, algoritmusok, elvek és általánosan hasznosítható ötletek leírásai. Külön örülök annak, hogy az idő haladtával ezeknek a cikkeknek az arányát nőni látom. Szeretném, ha több IBM PC-vel kapcsolatos cikk jelenne meg (ígérem, ha tudok, küldök valamit), és minél több általánosan hasznosítható írás. A könyvismertető cikkek általában nagyon sokára jelennek meg (némelyik könyv már el is fogy addigra), és akkor is elég szűkszavúak. Talán lehetne szólni a könyvek minőségéről is: hasznos-e, érthető-e, érdemes-e megvenni vagy nem ér semmit.

Mindig öröm, ha az olvasók észreveszik és nyugtázzák a szerkesztők törekvéseit. Valóban az a helyzet, hogy erősen mérsékeltük a játékporgramok közlését, mert úgy láttuk, a legtöbb olvasó jobban szeret figyelemfelkeltő cikkeket olvasni, mint szolgaian bepötyögni néhány tíz vagy száz utasítást. Lassan elkezdett a lap a PC kompatibilis gépekkel foglalkozni, és eljön az az idő, amikor egyes korszerűtlen gépekkel már nem is törődünk mi sem.

Valamennyi olvasónknak szívélyes üdvözletemet küldöm:

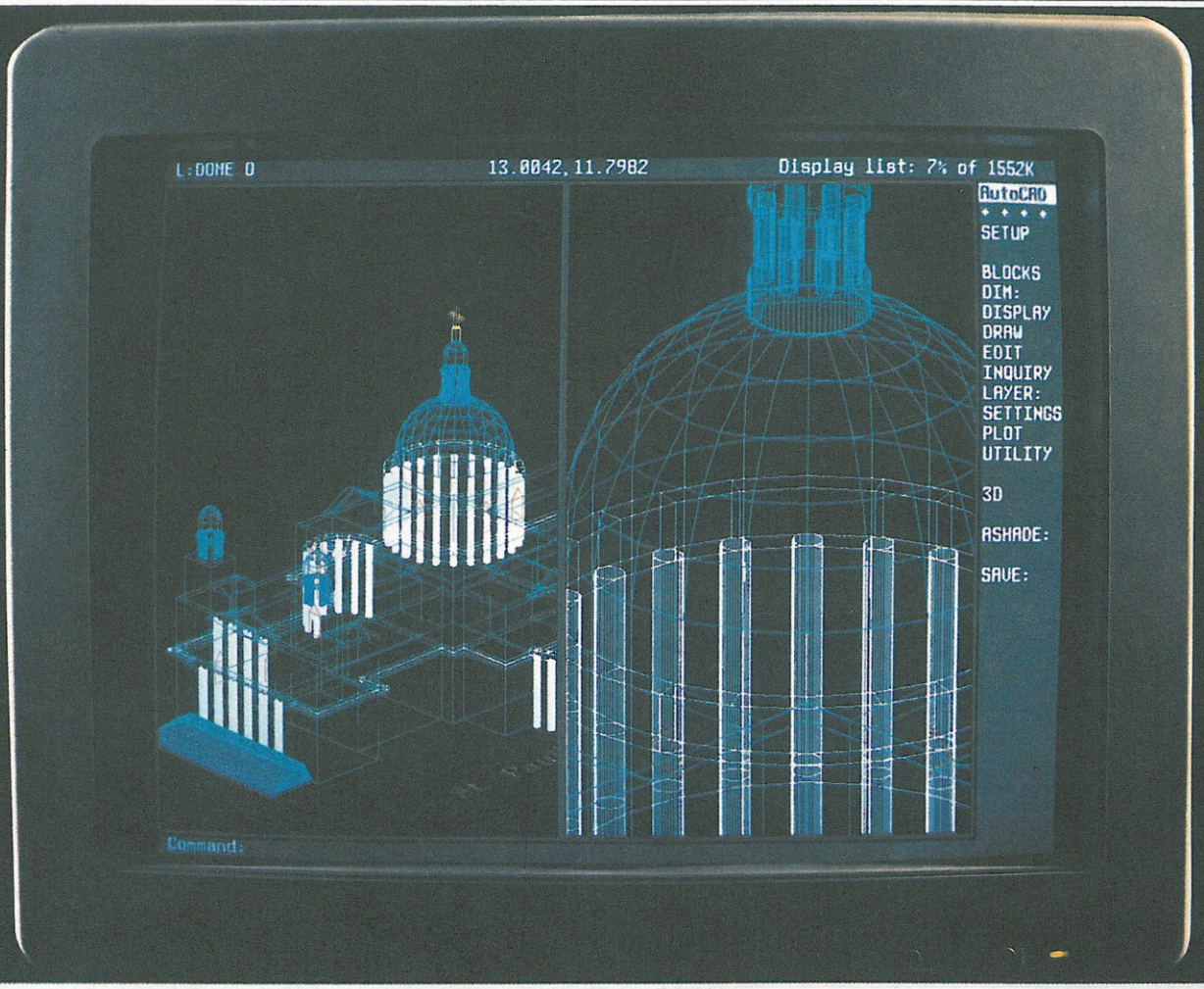
Kovács Győző

A M. A. D. rendszerrel készített polaroid fotó

Digitalizált retus

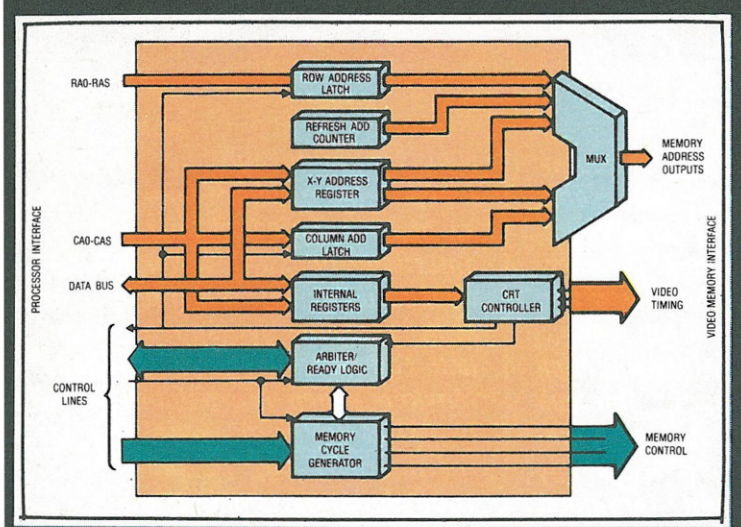
Nemsokára vége a kézi retusálásnak — állítják a fotópiac szakértői —, legalábbis ami a fekete-fehér képeket illeti. Amerikában olyan „számítógépes fotólaborokat" dobtak piacra, amelyek segítségével néhány gombnyomással eltüntethető a fényképről a nagymama fejéből kinövő karácsonyfa vagy a tájképben éktelenkedő villanyoszlop. A dolog nyitja: a képeket digitálisan ábrázolják, így a jeleket sokkal könnyebb megváltoztatni, mint a fotópapíron vagy a filmen, ahol kémiai beavatkozásra van szükség. De hogy kerül a fénykép a képernyőre? Ennek két útja is van. A gyakoribb az, hogy a már kész papírképet bedugják egy 2500 dolláros képolvasóba, amely a fotót úgy pásztázza végig, mint egy fénymásológép. A képernyőre azonban kerülhet olyan kép is, amely soha nem látott fotópapírt; néhány éve ugyanis léteznek olyan fényképezőgépek, amelyek a képet nem filmre, hanem hajlékony mágneslemezre rögzítik. Egy lemezkén ötven kép tárolható. A lényeg tehát az, hogy a fényjeleket digitális jelekké alakítják át. A képernyőn látható képet ezután kedvünkre alakítgathatjuk, s a végleges változatot ismét papírra rögzíthetjük, feltéve, hogy van egy legalább 3 és fél ezer dollárba kerülő, jó minőségű nyomtatónk. A gépi retus egyelőre nem tartozik az olcsó mulatságok közé, de az amerikai fotósok körében már népszerű az új felszerelés. Az árak azonban gyorsan csökkennek. Egy mágneslemezzel működő fényképezőgép ára például jelenleg 2700 dollár, de a gyártók azt ígérik, hogy egy év múlva már 1000 dollárért is lehet kapni.

Érdekesség, hogy Párizsban a M. A. D. cég már szolgáltatásként kínálja az Amiga vagy PC képernyőről — diapozitivra, illetve polaroid papírra — digitális úton készített színes fotót. Az utóbbi egy perc alatt készen van . . .



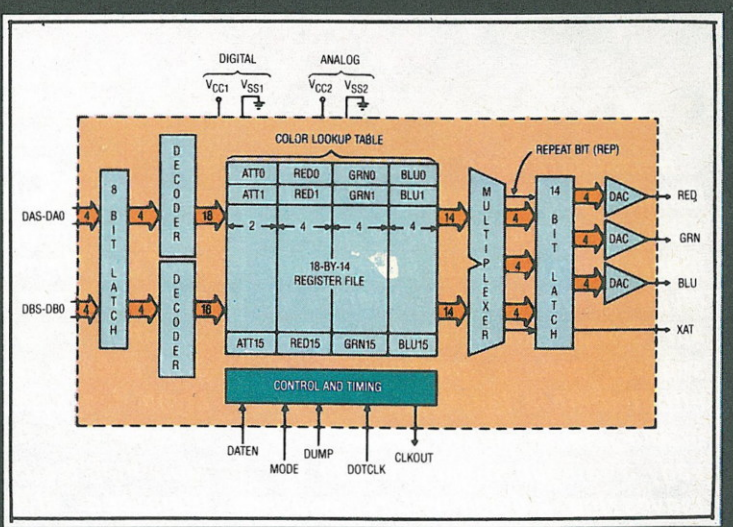
A Verticom HX kártyával és TwinFocus szoftverrel készített kép

1. ábra



A TMS34061
videorendszer-vezérlő folyamatábrája

2. ábra



A TMS34070
színes paletta folyamatábrája