Ára: 30 Ft

μ mikro számítógép magazin

NEM AUTÓPÁLYA, SZÁMÍTÓGÉP!

1989/1

ENTEC 386

A Superpower AT Compatible Personal Computer

Az Intel 80386 alapú, 32-bites hazai típusok 1987 tavaszi megjelenése után, másfél évvel később, az elmúlt ősszel mutatta be a következő szocialista ország az ilyen alapú gépeket: Bulgáriában egyszerre több típus is megjelent. Míg a Pravec 386 leváért is kapható, a Sport 386 és az Entec 386 viszont csak olyan bolgár intézmény által vásárolható, amely keményvalutával tud fizetni.

μ mikro számítógép magazin

7. ÉVFOLYAM
1989/1. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

Címképünk:
Kiss Ilona 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:
Havass Miklós főtitkár

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 (88—1801)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

TARTALOM

„Napjainkban az átlagemberek állandóan találkoznak a számítógéppel. Így van ez a munkabérrel, a jövedelemadóval, a bolti hitelszámlákkal, a bankszámlával, a hitelkártyákkal és számos más mindennapos tevékenységgel kapcsolatban. Sőt azt mondhatjuk, hogy ezek az alkalmazások nem kényelmi célokat szolgálnak, hanem kimondottan szükségszerűek. (. . .)

Mindennek folytán az emberiség világképe visszafordíthatatlanul megváltozott; az emberek életmódja átalakult, és ez az átalakulás tovább fog folytatódni azon nehézségek és problémák hatására, amelyeket a számítógép a társadalomban kivált."

(H. H. Goldstine: A számítógép Pascaltól Neumannig. 1972.)

30 éve készült el az M–3, az első magyar elektronikus számítógép

1959. január 21-e óta őrzöm az „Esti Hírlap" egykori számát és az első oldalról a fényképes tudósítást, „Elkészült az első magyarországi számítógép."

Harminc év, fél emberöltő, az ember el sem hiszi, hogy ennyi idő eltelt azóta, hogy az első programok lefutottak az M—3-on, az első magyar elektronikus számítógépen, az akkor hihetetlennek tetsző 50 művelet/mp sebességgel. Az újságba százezer művelet/óra adat került, (ami kevesebb, mint az általunk mért teljesítmény) valószínűleg azért, mert a százezer tekintélyesebb számnak tűnt, mint az ötven.

A szakemberek még ma is vitáznak arról, hogy az M—3 mennyiben tekinthető az első *magyar* számítógépnek.

Ha az Olvasók megengedik, akkor most ismét elmondom a magam, lehet, hogy túlságosan is egyéni és nem is biztos, hogy teljesen elfogulatlan véleményét. Az én kronológiám szerint az első hazai digitális, automata számítógépet 1955-ben a sajnos csak nagyon szűk körben ismert MESZ—1-et Kozma László, a Budapesti Műszaki Egyetem professzora építette. Ez a gép a szó igazi értelmében nem volt számítógép, hiszen a programot egy külső tárolóból (speciális lyukkártya, 45 darab 12 bites utasítás és 9 konstans fért rá) olvasta le, és csak az adatokat tárolta a belső tárolóban, így — miután nem volt tárolt programú gép — az ENIAC-nak volt elvileg rokona. A MESZ—1 architektúrájában, de bizonyos konstrukciós megoldásokban sem követte közeli rokonait, például a MARK-sorozat első példányait, hiszen Kozma professzor bizonyos célokra, történetesen tíz ívpontos, a telefonközpontokban használt léptetőgépeket alkalmazott. Azért sem ismerhette H. H. Aiken, a Mark gépek konstruktőrének munkáját, mert mint a Standard per egyik fővádlottja éppen börtönben volt, és így nem juthatott hozzá a megfelelő szakirodalomhoz.

Az M—3 elektroncsöves áramkörökből épült, mágnesdob memóriájában együtt tárolta az adatokat és az utasításokat, — ez a gép tekinthető az első magyar tárolt programú elektronikus számítógépnek.

Nemrégiben, amikor néhány barátom és ismerősöm társaságában elmondtam, hogy szeretném, ha megemlékeznénk az M—3 elkészültének 30 éves évfordulójáról, a társaság egyik tagja nem kevés éllel jegyezte meg, hogy „teljesen felesleges, hiszen az M—3 nem magyar, hanem szovjet gép volt, és így nyugodtan megemlékezhetnénk az Ural 1—2-ről, vagy a Razdanról is. Barátom véleményével nyilván nem értettem egyet, és hogy mennyire nem volt igaza, azt szeretném a továbbiakban bizonyítani.

Azt senki sem vitatja, hogy az M—3 szovjet tervek alapján készült, ezek a tervek — ha jól emlékszem — valamikor 1957-ben érkeztek Magyarországra.

A Kibernetikai Kutató Csoport — a Műszeripari Kutató Intézet egyik osztályaként — azzal a céllal, már előbb, 1956-ban alakult, hogy *megtervezzék* és *megépítsék* az első hazai elektronikus számítógépet.

A csoport létrejötte Tarján Rezsőnek volt köszönhető, akit nem sokkal előbb rehabilitáltak, és aki — Kozmához hasonlóan — a börtönben kezdett el egy elektronikus számítógép építésének a gondolatával foglalkozni. Tarján B—1-nek (Budapest 1-nek) nevezte az elektroncsöves, illetve nikkel-késleltető művonalas áramkörökből tervezett gépet. Tarján Rezső úgy mondta nekem, hogy az ENIAC-ot tekintette modellnek, amelyről egy elég jó leírás eljutott hozzá.

1957-ben a Műszeripari Kutató Intézetnek a Tarján vezette osztálya a Magyar Tudományos Akadémia önálló csoportjává vált (MTA Kibernetikai Kutató Csoport), igazgatójává Varga Sándort, tudományos igazgatóhelyettessé pedig Tarján Rezsőt nevezték ki.

A kinevezéseket inkább politikai, mint szakmai meggondolás motiválta, ti. Varga Sándor addig a Minisztertanács titkárságát vezette, és ebből a pozíciójából került az igazgatói székbe, a hírek szerint azért, mert állást kerestek számára. Varga a szovjet emigráns csoporthoz tartozott. Varga Jenő, közgazdász unokaöccse lévén, igen jó szovjet tudományos kapcsolatokkal rendelkezett. Azt is beszélték róla, hogy nem volt rossz automatikai szakember, hiszen a háború alatt ő tervezte a szovjet T—34-es páncélosok lövegét vezérlő rendszert. (Egyszer megkérdeztem tőle, hogy igaz-e ez az információ, a kérdést elhárította, se nem cáfolta, se nem igazolta.)

Varga és Tarján talán soha nem értettek igazán egyet a KKCS feladatait illetően, miután mint láttuk, Tarján egy saját tervezésű számítógépet akart építeni, Varga pedig az akkori tudományos szokásoknak megfelelően a számítógépépítést szovjet kooperációban képzelte el.

Nem volt kétséges, hogy az utóbbi elképzelés fog győzni, és persze győzött is, hiszen hamarosan létrejött a szerződés a szovjet és a magyar akadémia között az M—3 terveinek az átvételére. Az M—3 egy közepes teljesítményű gép volt, a szovjet választékban nem a legnagyobb — hiszen a BESZM már akkor működött — de azt hiszem, akkor ez a gép látszott a legkorszerűbbnek. A választást még ma is szerencsésnek tartom, mert akkor ezzel a feladattal a nagyon sok kezdő szakemberből álló kutatócsoport még éppen meg tudott birkózni. Tarján — véleményem szerint — nem elvtelenül és nem megalkuvásból, de elfogadta a döntést, és teljes szívvel igyekezett volna az M—3 építését segíteni. A feltételes mód itt azért fontos, mert Varga Tarjánt igazán sohasem engedte beleszólni az építési munkákba, annak vezetését, a döntéseket mindig fenntartotta magának.

A munka valamikor 1957-ben indult, a munkatársak egy része akkor már hónapok óta a kutatócsoportban dolgozott (Szanyi László, dr. Edelényi László, Sándor Ferenc, Bóka András, aki azonban nem sokkal az indulás után kivált az M—3-as fejlesztői csoportból), néhányunkat akkor vettek fel közvetlenül az egyetemről (Dömölki Bálint, Szelezsán János, Hajnalné Márkus Emília, Veidinger László, Molnár Imre, Podhradszky Sándor, Kovács Győző). A csoportot igen kiváló technikusok és szakmunkások segítették, mint például, Kardos Kálmán, Ercsei István, Pólya Endre, Ficze Sándor, Horváth Mária, Dani János, Jámbor Antal, Suhajda János, Piller Ignác, akik szinte az első pillanattól kezdve részt vettek a munkában.

A fejlesztők száma, amikor a feladat nagysága már jól látszott, tovább növekedett, csatlakozott hozzánk Szentiványi Tibor, Németh Pál, Drasny József, Ábrahám István, akiknek jelentős szerepük volt a gép elkészültében.

Az M—3-at a KKCS vezetése már az első pillanattól kezdve különféle gyakorlati feladatok megoldására szánta. A KKCS két erőssége volt a nagyon dinamikus matematikai, illetve az alkalmazásokat előkészítő közgazdasági részleg. A már felsorolt szakembereken kívül itt dolgoztak dr. Aczél István, majd kicsit később Lőcs Gyula , Révész György, Gergely József és Frey Tamás, illetve Krekó Béla, Kornai János, Ganczer Sándor és még sokan mások, kül- és beltagok egyaránt.

A KKCS nagyon sokféle kutatási feladatot vállalt, főleg olyanokat, amelyeket kizárólag számítógéppel lehetett megoldani, így például támogatta a nyelvészeti kutatásokat is, Kiefer Ferenc és Vargha Dénes munkáját.

A kutatócsoportban hardverfejlesztés is folyt, például Hatvany József NC-vezérlést, Münnich Antal digitális áramköröket, Bóka András, Ladányi József és Czili Gyuláné pedig mágnesmagos logikai áramköröket fejlesztett.

A gép üzemeltetése voltaképpen a hivatalos átadás után kezdődött. Varga Sándor — kitűnő érzékkel — G.P. Lopato szovjet mérnököt hívta el a gép működőképességének a bizonyítására. Lopato elvtárs egyike volt az M—3 konstruktőreinek és egyben vezetője a szovjet M—3 építésének és üzembe állításának.

Itt kell talán megjegyezni, hogy a szovjet és a magyar példánnyal egyidőben, talán valamivel később egy kínai M—3-as építése is megkezdődött. 1985-ben véletlenül összefutottam Sun Quiangan professzorral, akiről igen gyorsan kiderült, hogy majdnem ugyanaz volt a feladata a kínai M—3-as fejlesztésében, mint nekem a mi M—3-asunknál, mondanom sem kell, hogy nehezen tudtuk abbahagyni a váratlan fordulatot hozó beszélgetést.

Nem tudom, hogy a hivatalos, Lopato-féle átvételre pontosan mikor került sor, van aki 1959 januárjára, mások decemberre esküsznek. Az átvételt követően Varga Sándor a KKCS-t átszervezte, megszüntetett az üzemeltetési osztály, és megindult a gép szabályszerű üzemeltetése. Új, addig ismeretlen munkaköröket kellett létrehoznunk, az addigi fejlesztőmérnököket és technikusokat karbantartó szakemberekké kellett átképezni, néhány matematikus már programozónak nevezte magát, és mint addig ismeretlen státus, elkezdték munkájukat az operátorok, akik közül az első Várkonyi Zsolt volt, majd Götzy Ilona, Kovács Győzőné és Varga Gabriella.

Az akkori szokásoknak megfelelően a gépet folyamatosan továbbfejlesztettük. Az áramkörök működésének biztonságossá tétele állandó feladatunk volt, ennek érdekében az egyik legnagyobb munkát a normál rádiócsöveknek a hosszú élettartamú csövekre való átcserélése jelentette. Hamarosan még egy mágnesdobot kapcsoltunk a géphez, és így megháromszoroztuk a memória kapacitását 1 kszóról 3,2 kszóra.

A legnagyobb hardverfejlesztést azonban egy 1 kszó kapacitású gyors ferritmemória vásárlása és a géphez kapcsolása jelentette, aminek következtében a gép teljesítménye (a műveletek végrehajtási sebessége) mintegy három nagyságrenddel növekedett. Hasonlóan nagy lépést jelentett, amikor a matematikusok fejlesztési munkájának eredményeként a gépi kódú programozás helyett már mnemonikus kódban, majd pedig M—3 autokódban lehetett programokat írni.

Ami a programozást illeti, a KKCS matematikusai a különféle táblázatok kiszámításától hamar eljutottak a nagy népgazdasági modellek, szállítási problémák megoldásáig, ezeket a feladatokat a csoport közgazdászai szervezték és fogalmazták meg.

Bonyolult és egészen más típusú feladatot jelentett például Tóth Árpád és Ady Endre műveinek nyelvstatisztikai vizsgálata és nem kevésbé az új Erzsébet híd terhelési eredményeinek feldolgozása a mai számítógépekhez mérten igen kis teljesítményű számítógépen.

Az M—3 1965-ig üzemelt Budapesten, akkor az Akadémia egy új Ural 2 gépet vásárolt, ami már „gyártott" és nagyobb teljesítményű gép volt, így azután az M—3-ra már nem volt tovább szükség. Ekkor kezdődött az M—3 életének második szakasza Szegeden, a JATE Kibernetikai Laboratóriumban. Kalmár László akadémikus a gépet főleg oktatási célokra szánta, de a laboratórium munkatársai a gépen nagyon sok alkalmazói feladatot is megoldottak.

Muszka Dánielnek és munkatársainak sikerült a gép üzemi megbízhatóságát növelni és azt hosszú ideig megfelelő szinten is tudták tartani. Ami nem sikerült: az M—3-at 1968-ban leszerelték, és miután a múzeum nem fogadta be, szétszedték és „kimérték" a JATE intézetei között, ma már csak néhány alegység és néhány mágnesdob maradt meg emlékeztetőül.

Nemrégiben még megvolt a temesvári múzeumban egy másik M—3-as mágnesdob is, ami az első, a temesvári egyetemen épült román gép memóriájaként szolgált. William Löwenfeld és Joseph Kaufmann építettek ott egy gépet, amelynek nem volt memóriája, ezért csak kézi adagolással lehetett vele műveleteket végrehajtani. Dr. Aczél István segítségével készíttettünk egy dobmemóriát a temesváriaknak, majd még kettőt Bukarestbe, egy akkor meg nem nevezett felhasználónak.

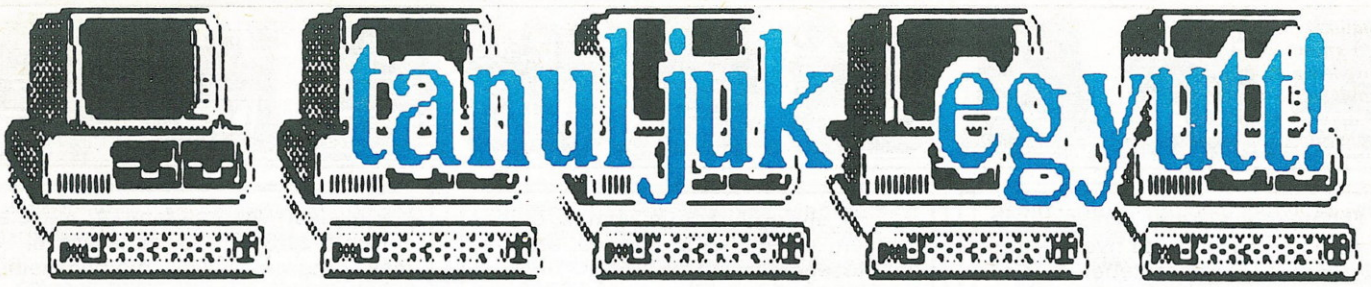
A temesvári gép sokáig és jól működött, majd a kétségkívül attraktív memória a múzeumba került, amit onnan tudtam meg, hogy valaki, aki ismerte itthoni munkáinkat, küldött egy képet emlékül.

Visszatérve a barátaimmal folytatott beszélgetésre, talán elhiszi nekem a kedves Olvasó, hogy az M—3 annak ellenére, hogy az alaptervek a Szovjetuniónában készültek, mégiscsak az első magyar elektronikus számítógép volt, ami elindította a számítástechnika fejlődését Magyarországon.

Azt sem lehet figyelmen kívül hagyni, hogy Neumann János és H.H. Goldstine az első modern (Neumann elvű) számítógépet, az IAS gépet 1952 novemberében, azaz csak *hét* évvel az M—3 átadása előtt mutatták be, de a többi „világhírű" gép sem volt készen sokkal korábban, hiszen Eckert és Mauchly is csak 1951 márciusában fejezték be a UNIVAC fejlesztését, ami pedig még nem volt igazi számítógép. Az első — sikeres — IBM számítógépcsaládból a 701-es modellnek a feladatására is csak 1953. április 7-én, míg az első, köznapi használatra készült IBM-szériagépnek, az IBM 702-esnek a bemutatására 1955. február 1-jén, azaz hat illetve négy évvel az M—3 átadása előtt került sor.

Ezt az írást emlékeztetőnek szántam, így a tanulságok elemzését meghagyom majd egy következő jubileumra.

Kovács Győző



Nyílt levél az Olvasóhoz

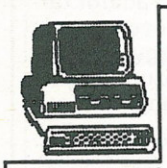
A Mikroszámítógép Magazin szerkesztősége elhatározta az eddig Iskola — számítógép *nevet viselő rovat megújítását. E rovat tartalmának és az olvasók körének meghatározása miatt kérte az Országos Pedagógiai Intézet számítástechnikai szaktanácsadó csoportjának véleményét is. Mi, a csoport tagjai — a számítástechnikát már hosszú évek óta tanítók, azt művelők és szolgálók — a szerkesztőséggel egyetértésben alakítottuk ki álláspontunkat. Elsősorban a tanárokhoz, a középiskolás diákokhoz és a számítástechnikát nem jövendő hivatásuknak választó, de azzal foglalkozó egyetemistákhoz, főiskolásokhoz kívánunk szólni: mindazokhoz, akik a számítástechnikát és annak alkalmazási lehetőségeit alkotóan művelni szeretnék és közkinccsé kívánják tenni.*

A rovatban megpróbálunk olyan cikkeket, írásokat, ötleteket, leveleket és riportokat közölni, amelyek az ismertetett célokat szolgálják, és segítséget nyújtanak a sokszor zsákutcába vezető munka és önképzés elkerüléséhez. Tervezett témáink között szerepelnek alkalmazói programok ismertetései, még kevéssé alkalmazott programnyelvek megismerését és elsajátítását célzó cikkek vagy cikksorozatok, hardver- és szoftverbővítések, oktatási alkalmazások, újdonságok és mindaz, amit önök, az olvasók kérnek és javasolnak. Kérjük ehhez segítségüket, várjuk bíráló — esetleg egyetértésüket kifejező — leveleiket, és nem utolsósorban a talán legfontosabbat: aktív közreműködésüket, közlésre szánt írásaikat.

Nagy Imre
rovatvezető

A PASCAL REJTELMEI

Alapozás „meztelen" elemekkel



A nyolcbites házi számítógépek (ezután HC-k) megjelenésével és tömeges elterjedésével megnyílt a kapuja a programozás világának mindazok előtt, akiket ez érdekelt. Azóta néhány év eltelt, és egyértelműen kiderült: sokaknak ez a világ csak részben tárult fel, vagy a kapu volt inkább csak afféle kémlelőnyílás. A HC-k adottságaiból következett, hogy a gép segítségével megoldandó problémákat csak BASIC — egy szűk rétegnek assembly — nyelven lehetett megfogalmazni. Hiába választhattak a számtalan alkalmazást segítő programból, ezek többnyire csak célfeladatok megoldásában nyújtottak segítséget. A felhasználó egyéni és/vagy egyedi problémáinak feldolgozása sokszor igen körülményesnek bizonyult. Végül is tehát a HC-k birodalmában a programozóknak nincs jól kezelhető, univerzális nyelvi eszközük.

A professzionális gépek (ezután PC-k) jobb hardverjük — 16 bites processzoruk, 8—16 bites belső adatsíneik, nagy központi és háttértár-kapacitásuk — révén BASIC-kel is hatékonyabban használhatók ugyan, de elsősorban a nagy tárkapacitásuk azt is lehetővé tette, hogy több és minőségileg is jobb nyelvi támogatás kerüljön a felhasználók kezébe. Ezek közül kiforrottnak, könnyen elsajátíthatónak és jól alkalmazhatónak tűnik a Pascal. Az sem elhanyagolható tény, hogy a HC-k egy részére is léteznek különféle Pascal-fordítók, így sorozatunkból nemcsak azok tanulhatnak, akiknek PC-jük van, esetleg ilyen gép várományosai, hanem a Spectrum-, az Atari- és a C64-tulajdonosok is.

1. Röviden a Pascal nyelvről

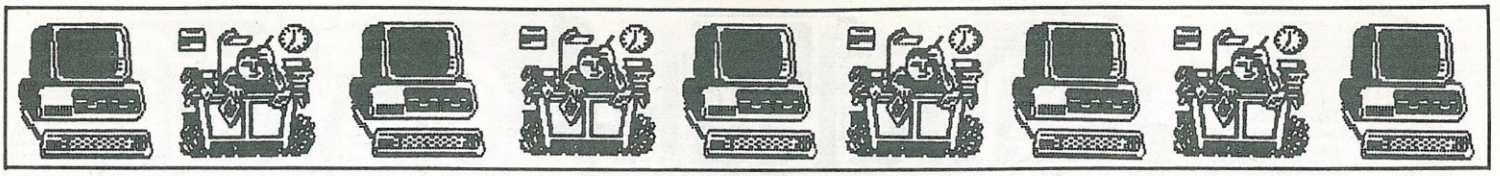
Mint a mai magas szintű programnyelvek többségének, a Pascalnak is vannak előzményei. Sokak véleménye szerint őse az ALGOL. Ezt bizonyítja a nyelvi elemek — operátorok, változótípusok, kulcsszavak stb. — hasonlósága és a nyelv adta strukturálási lehetőség is.

A Pascal eredeti definícióját a Zürichi Műszaki Egyetem professzora, Niklaus Wirth alkotta meg. A különböző gépi reprezentációk a szabványos Pascaltól általában többletszolgáltatásaikban térnek el; az eredeti nyelvi alapokat többé-kevésbé tartalmazzák, kivétel talán a C64-re készült G-Pascal, melynek az eredeti nyelvhez nem sok köze van.

Sorozatunkban a Borland International cég CP/M és MS/DOS operációs rendszerek alatt futó Turbo Pascal nyelvének 3.XX (3.0, 3.01, 3.1 stb.) verzióját választottuk alapul. Programjaink e változat nyelvi készletének, szintaxisának és egyéb lehetőségeinek — korántsem teljes — felhasználásával fognak készülni.

Mivel a Magyarországon hozzáférhető PC-k általában IBM kompatibilisek, így az MS/DOS 3.1 alatt futtatható változattal foglalkozunk. A 3.XX verzióknak sem minden lehetőségét használjuk fel, igyekszünk csak azokra a nyelvi elemekre támaszkodni, amelyek azonosak a szabványos Pascal elemeivel, illetve azoktól lényegesen nem térnek el. Néhány Turbo Pascal specialitást csak azért mutatunk be és alkalmazunk, mert segítségükkel programozási munkánkat lényegesen egyszerűsíthetjük.

A sorozat a számítógép-programozás alapvető kérdéseivel nem foglalko-



zik. Feltételezzük, hogy olvasóink ebben és egy magas szintű nyelv — például a BASIC — ismeretében már némi jártasságra tettek szert. A teljes sorozat jobb áttekinthetősége céljából az egyes részeket folyamatos „tizedes" sorszámozással látjuk el; az ábrák számozása hasonlóan folyamatos lesz, így megkönnyítjük a hivatkozásokat is.

Célunk nem Pascal-tankönyv vagy kézikönyv írása és részletekben való közlése. Ezért a nyelv megismertetését egy-egy komolyabb programfeladat köré csoportosítjuk, az egyes programok mindig az előző ismeretekre épülnek, és mindig csak a szükséges „újdonságokkal" foglalkozunk részletesen.

1.1 A Pascal elemei

A KARAKTERKÉSZLET

Betűk: az angol ábécé kis- és nagybetűi és az aláhúzás jel.

Számok: 0—9

A szóköz (space) karakter

Speciális karakterek: + − * / = ^ < > ( ) [ ] { } . , : ; ' # $

LEFOGLALT SZAVAK (KULCSSZAVAK)

A lefoglalt szavak (reserved words) a Pascal „beépített" elemei, azonosítóként nem alkalmazhatók és nem definiálhatók újra, azaz más jelentéssel nem ruházhatók fel. Tekintsük át e szavakat:

and	**array**	**begin**
case	**const**	**div**
do	**downto**	**else**
end	**file**	**for**
function	**goto**	**if**
in	**label**	**mod**
nil	**not**	**of**
or	**packed**	**procedure**
program	**record**	**repeat**
set	**then**	**to**
type	**until**	**var**
while	**with**	

A standard Pascalnak nem elemei, de a Turbo Pascalban lefoglaltak a következő szavak is:

absolute, external, inline, shl, shr, string, xor.

A Turbo Pascalban nagyszámú, egyéb előre definiált azonosító is van (konstansok, adattípusok jelölései, eljárás- és függvénynevek). Részletes felsorolásuktól — elsősorban nagy számuk miatt — most eltekintünk. Ilyenek például:

clrscr (képernyőtörlés eljárás)

cos (koszinusz függvény)

char (a „karakter" adattípus jelölése)

Ezek elvileg újradefiniálhatók, de ha ez nem okvetlenül szükséges, ezzel a lehetőséggel ne éljünk!

ADATTÍPUSOK

A Pascal négyféle standard — külön meghatározást, definiálást nem igénylő — adattípust ismer, melyek a következők:

- **integer** (kétbájtos egész: −32768-tól 32767-ig)
- **real** (valós: lebegőpontos szám 11 decimális jegy pontossággal, két decimális jegyből álló kitevővel: a kitevő értéke −38-tól 38-ig)
- **boolean** (logikai: értéke false: hamis, vagy true: igaz; az értékek relációja: false < true)
- **char** (karakter: egy bájtban tárolt ASCII kód)

A Turbo Pascal ismeri a **byte** adattípust is (egybájtos egész). Mivel ennek jelentősége csekély, alkalmazni nem fogjuk, az egybájtos egészek tárolására az integer típust használjuk. Turbo Pascal jellegzetesség a **string** (karakterlánc) típus is. Ez a hagyományos Pascal rendszerekben nem létezik, helyette egydimenziós karakter **(char)**-tömböt alkalmaznak.

OPERÁTOROK

Az operátorokat — műveleti jeleket — precedenciájuk, azaz a műveleti sorrend szempontjából öt csoportra osztja a Pascal. Ezek a műveleti sorrend szerint az alábbiak:

1. − (mínusz előjel)
2. **not** (logikai tagadás)
3. *** / div mod and** (szorzás, osztás, az osztás eredményének egész része, az osztás maradéka, logikai és)
4. **+ − or xor** (összeadás, kivonás, logikai vagy, logikai kizáró vagy)
5. **= <> < > <= >=** (egyenlő, nem egyenlő, kisebb, nagyobb, kisebb vagy egyenlő, nagyobb vagy egyenlő)

A BASIC-ben gyakorlottaknak külön kihangsúlyozzuk, hogy a relációs operátorok alkalmazásánál különös figyelemmel járjanak el, ugyanis a relációs jelek a BASIC-ben nem operátorok. Például az

A < 2 or A > 10

kifejezés BASIC-ben

(A < 2) or (A > 10)

formában, Pascalban pedig

A < (2 or A) > 10

formában értékelődik ki. Ezért hasonló esetekben a szükséges műveleti sorrendet zárójelekkel kell jelölni. Egyenrangú műveleteknél balról jobbra kell a kiértékelést végrehajtani. A műveleti sorrend zárójelek alkalmazásával — más programnyelvek megoldásaihoz hasonlóan — tetszés szerint befolyásolható.

KIFEJEZÉSEK

A kifejezések operátorokkal összekapcsolt adatok (konstansok, változók). A következőkben azt vizsgáljuk, hogy a különböző operátorok milyen adattípusokon végezhetnek műveleteket. Hogy a szabályokat egyszerűen fogalmazhassuk meg, tegyünk egy megszorítást: a kettőnél több operandusú műveletekkel külön foglalkozunk.

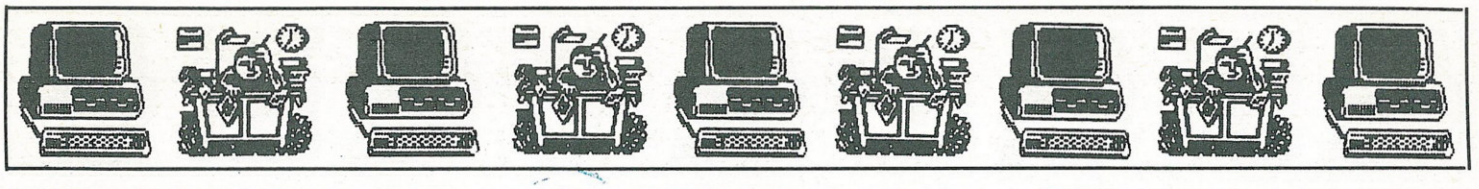
Ezzel a megkötéssel a szabályok igen egyszerűek. Egy aritmetikai kifejezés értékének típusa mindig azonos a benne szereplő adatok típusával. Kivétel ez alól a **real** és az **integer** típusok keverhetősége: ilyenkor az eredmény mindig **real. Integer** adatokon végzett műveleteknél az osztás eredménye mindig **real,** a többi műveletnél **integer. Boolean** típusokkal értelemszerűen csak logikai műveletek végezhetők. Ilyenkor az eredmény is **boolean.** Minden relációs operátorral végzett művelet **boolean** típusú eredményt szolgáltat.

Ha az operandusok száma kettőnél több, az iméntiekben megfogalmazott szabályok általános érvényéről nem kell lemondanunk. Egy több adatot tartalmazó kifejezés kiértékelése ugyanis mindig elvégezhető úgy, hogy egyszerre csak két adattal foglalkozunk, azaz a bonyolult kifejezést kétoperandusos műveletek sorozatára bontogatjuk le.

UTASÍTÁSOK

A Pascal standard utasításkészlete kicsi. Ennek magyarázata elsősorban az ún. eljárások használatának lehetősége. Az eljárások — egyszerűen és nem teljesen szakszerűen fogalmazva — a felhasználók által definiált utasítások. Ezeknek a száma viszont a programozó igényétől, képzettségétől, gyakorlatától függően szinte tetszőleges lehet. A szabványos Pascal is, de főként az egyes Turbo Pascal változatok a programozási munka megkönnyítésére számos „beépített" eljárást tartalmaznak.

Szerepüket és felhasználásukat tekintve az eljárások leginkább más programnyelvek szubrutinjaihoz hasonlóak, de a programkörnyezettel való kommunikáció szempontjából azoktól lényegesen különböznek; az eljárásoknak egyszerű eszközökkel lehet adatokat átadni, illetve azokból a környezetbe kivinni. Az eljárások ezenkívül gyakorlatilag függetlenek a programkörnyezettől is, például ugyanazon



változónevek az eljárásban, illetve azon kívül más-más jelentéssel bírhatnak.

Nézzük ezek után az egyes utasításokat, most csak csoportosítva és felsorolva, a funkciók részletes magyarázata nélkül:

1. := (értékadás)
2. **GOTO** utasítás (feltétel nélküli vezérlésátadás)
3. Feltételes utasítások (**boolean** változók vagy kifejezések értékétől függően végeznek programelágazást):
   — **if..then** (végrehajtás, ha a vizsgálat eredménye **true**)
   — **if..then..else** (ha a vizsgálat eredménye **true,** a **then**-ágon, ha **false,** az **else**-ágon fut le a program)
   — **case..of** (többirányú elágazás)
4. Ciklusutasítások
   — **for..do** (egyszerű ciklus)
   — **while..do** (addig ismétel, ameddig az utasításban megadott feltétel értéke **true**)
   — **repeat..until** (ha a ciklusmag lefutása után a megadott feltétel **true,** a ciklusnak vége)

Bár funkcióját tekintve nem utasítástípus, de a Pascal egyik jellegzetessége az ún. összetett utasítás. Kezdetét a **begin,** végét az **end** kulcsszó jelzi. Például:

```
begin
  utasítás_1
  utasítás_2
  ...
  ...
  ...
  utasítás_n
end
```

Az összetett utasítások alkalmazása akkor célszerű például, ha feltételes utasítással nem egyetlen utasításra, hanem egy sorozatra akarjuk átadni a vezérlést. A figyelmes olvasónak feltűnhetett, hogy az utasítások között nincsenek I/O műveletekre vonatkozók, ernyőképet vagy nyomtatási képet vezérlő utasítások stb. Ezeket a Pascalban eljárások valósítják meg (például **read**-bevitel, **write**-kivitel stb.).

Mind a standardban, mind a Turbo Pascalban van még néhány utasítás, amelyekről nem szóltunk. Elképzeléseink szerint a sorozat későbbi — „haladó szintű" — részeiben ezek ismertetésére is sort kerítünk.

1.2 A Pascal program szerkezete, az alapvető szintaxisszabályok

A Pascal program elkészítése gondos tervezést kíván. Az előírások szigorúak, amelyeket feltétlenül be kell tartani. A követelmények szerint készített program viszont jól áttekinthető, az egyes részek egymástól függetlenül megírhatók — például több programozó párhuzamos munkájával —, és a program egyszerűen átszerkeszthető, módosítható.

A Pascal program három fő részből áll:

— fejből,
— deklarációkból,
— programtörzsből (a program végrehajtható részéből).

A fej általában csak a program azonosítását szolgáló név, de megadhatók olyan paraméterek is — adatok, változók nevei —, amelyeknek segítségével a program a környezetével kommunikál. E lehetőséget mi nem fogjuk kihasználni, csak olyan programokat készítünk, amelyek önállóak, más programokkal nem tartanak kapcsolatot.

Ebben az esetben a programfej igen egyszerű, a **program** kulcsszóval kezdődik, amit szóközzel elválasztva követ a program neve. Például:

```
program Adatfeldolgozas;
program rendezes;
```

A példákból látható a Pascal szintaxisának egyik alapvető szabálya is. Minden deklaráció, utasítás stb. után a ; jelet kell alkalmazni, tehát nem elegendő az ENTER (RETURN) billentyű lenyomása. A ;-vel lezárt utasításokból viszont többet is írhatunk egy fizikai sorba. A fizikai sor mérete maximálisan 126 karakter, de az áttekinthetőség miatt nem célszerű az adott monitorra jellemző sorméretet — ami az IBM monitoroknál 80 karakter/sor — túllépni.

A deklarációk öt csoportra bonthatók. Ha valamelyikre nincsen szükség, természetesen elhagyható. A különféle deklarációknak megszabott sorrendje van:

— címke **(label)** deklarációk,
— konstans **(const)** deklarációk,
— típus **(type)** deklarációk,
— változó **(var)** deklarációk,
— eljárás **(procedure)** és függvény **(function)** deklarációk.

Az egyes deklarációkat a zárójelben írt lefoglalt szavakkal végezhetjük el. Nézzük ezeket most sorban!

CÍMKEDEKLARÁCIÓ

Mivel a Pascalban az utasítássoroknak nincs sorszámuk — mint ahogy a BASIC-ben van —, a **GOTO** utasítás számára az ugrási célt meg kell jelölni. Erre szolgál a címke. Egy címkével ellátott sor formája:

```
ide: write(a);
```

A címke azonosítója (példánkban: ide) után a : jel szükséges, azt követi az utasítás. (Azzal, hogy ez most mit jelent, nem foglalkozunk.) A címke deklarációja:

```
label ide;
```

Egy deklarációs sorban több címke is megadható, egymástól a , jellel elválasztva. Például:

```
label ide, hiba, vege;
```

A Pascal mindaddig címkeként értelmezi a szavakat, amíg egy újabb lefoglalt szót nem talál; ezért az egyes címkék külön sorba is írhatók, ahogyan ezt a következő példa szemlélteti:

```
label ide;
      hiba;
      vege;
```

A tabulált írásmód sem szükséges, ez csak az áttekinthetőség javítását szolgálja. Az ilyen, jól áttekinthető megoldást majd a későbbiekben, a programok írásakor is alkalmazzuk, a programok logikailag összetartozó részeinek egyes sorait azonos vízszintes tabulációval fogjuk kezdeni.

KONSTANSDEKLARÁCIÓ

A konstansdeklaráció azonosítókhoz (nevekhez) szám- vagy szövegkonstansokat rendel hozzá. A forma a következő:

```
const adat=325;
      gyorsulas=9.81;
      cím='TURBO PASCAL';
```

A Turbo Pascalnak van néhány előre definiált konstansa: közülük komolyabb jelentősége a **pi**-nek van, melynek értéke: 3,1415926536E+0001.

TÍPUSDEKLARÁCIÓ

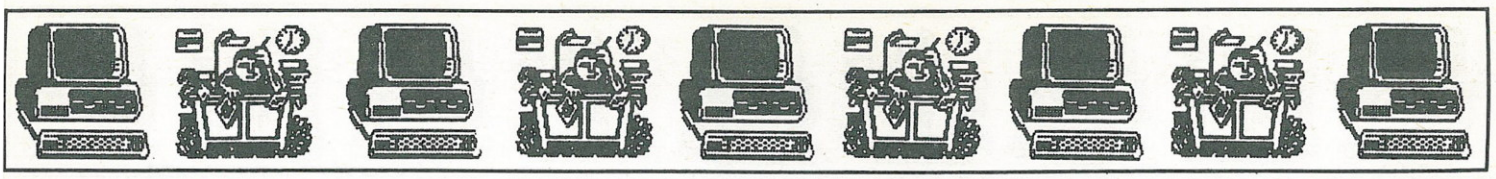
A standard adattípusokból — **integer, real, boolean, char,** a Turbo Pascalban a **string** is — a felhasználó a típusdeklaráció segítségével újabb adattípusokat alkothat. A következőkben néhány jellegzetes típusdeklarációt mutatunk be:

```
type ketbajt=integer;
     nap=(H, K, Sze, Cs, P, Szo, V);
     nev=array [1..30] of char;
     nevsor=array [1..10] of nev;
```

Az első deklaráció a ketbajt elnevezésű típust integerként definiálja.

A második deklaráció az ún. felsorolási, enumeratív típusmegadás: a nap típus lehetséges értékeit a zárójelben soroljuk fel, egymástól a , karakterrel elválasztva.

A harmadik példa egy nev típusú, 30 karakterből álló tömböt deklarál. A tömböt az **array** kulcsszó jelzi, ezt követi szögletes zárójelek között a tömbindex minimális és maximális értékének megadása. A deklaráció a tömbelemek típusának leírásával fejeződik be: **of char** (típusmegjelölés).



A teljesség kedvéért jegyezzük meg, hogy a bemutatott deklarációval a Turbo Pascalban egyenértékű a **string [30]** definíció.

A negyedik deklaráció felhasználja a már létező nev típust: egy nevsor típust definiál, amely tulajdonképpen egy maximum 10, egyenként maximum 30 karakterből álló tömb.

VÁLTOZÓDEKLARÁCIÓ

A változódeklarációban a változók típusát kell megadni. A Pascal egyik fontos előírása, hogy a programban használt összes változót deklarálni kell. Ha a fordító nem deklarált változót észlel fordítás közben, hibajelzés kíséretében **(unknown identifier)** megáll. A deklarációt a **var** kulcsszó vezeti be:

```
var adat:integer;
    betu, jel:char;
    igazsag:boolean;
    dolgozo_neve:nev;
    lista:nevsor;
```

A változóneveket a : karakter követi, ezután kell megadni a változó típusát, amely a standard vagy a **type** deklarációban definiált típusok bármelyike lehet.

ELJÁRÁS- ÉS FÜGGVÉNYDEKLARÁCIÓ

Az eljárások és függvények a más programnyelvekből már ismert szubrutinokhoz hasonló szerkezetek. Mivel a strukturált programozás talán legfontosabb eszközei, jelentőségük igen nagy. A szerkezet szempontjából mind az eljárások, mind a függvények önálló programnak tekinthetők, így felépítésük azokéhoz hasonló:

— fej,
— deklarációk,
— eljárás- vagy függvénytörzs.

A különbség mindössze annyi, hogy míg a programfej a **program** kulcsszóval kezdődik, az eljárásoknál és a függvényeknél erre a célra a **procedure,** illetve a **function** kulcsszavak szolgálnak.

Az eljárások és a függvények a környezettel való kommunikációhoz paraméterek — többnyire változók értékeinek — átvételét és átadását igénylik. Ezeket a név után zárójelek között kell felsorolni. Mivel az eljárásokkal és a függvényekkel később részletesen foglalkozunk, így most a paraméterátvételt és -átadást bővebben nem tárgyaljuk meg.

A programok — és most már tudjuk, hogy ez az eljárásokra és a függvényekre is vonatkozik — végrehajtható utasításokból álló része a törzs. Ennek a résznek a behatárolására vannak a **begin** és az **end** kulcsszavak. Ezek nem utasítások, csak a fordítóprogramnak jelzik a program (eljárás, függvény) kezdetét és végét; hasonlóan, mint ahogyan azt az összetett utasításokkal kapcsolatban már említettük.

Alapvető szintaxisszabály, hogy egy programban — annak minden ágán — azonos legyen a **begin** és az **end** szavak száma. A program végét jelző **end** után a . karakter álljon, minden más esetben a ; jelet kell alkalmazni. Mivel a begin nem utasítás, utána a ; nem szükséges. Ugyanezért nem kell alkalmazni ;-t az **end** előtt lévő utasítás végén sem.

A szintaxissal a sorozat egyes részeiben — ha arra szükség van — később is foglalkozunk. Az induláshoz, úgy véljük, ennyi éppen elegendő.

N. I.

Új utasítások C Plus/4-re

Több grafikus képernyő használata

Aki gyakran készít grafikát, valószínűleg hiányolta azt a lehetőséget, hogy több grafikus képernyőt használhasson egyszerre, és az ezeken elkészített rajzokat gyorsan váltogathassa a monitoron. Ezt a hiányosságot igyekszik pótolni a most ismertetendő gépi kódú program.

Alkalmazásával lehetősége nyílik a felhasználónak arra, hogy egyszerre több grafikus képernyőt definiáljon, ezekre rajzokat készítsen és azokat cserélgethesse. A program BASIC-utasításként hívható. Szintaxisa:

PICTURE P

ahol P a képernyő száma. Értéke alaphelyzetben 0 és 3 között lehet (a programba való „belepiszkálással" a 4 is használható paraméterként, azaz öt képernyő valósítható meg; erről egyébként a későbbiekben szó lesz).

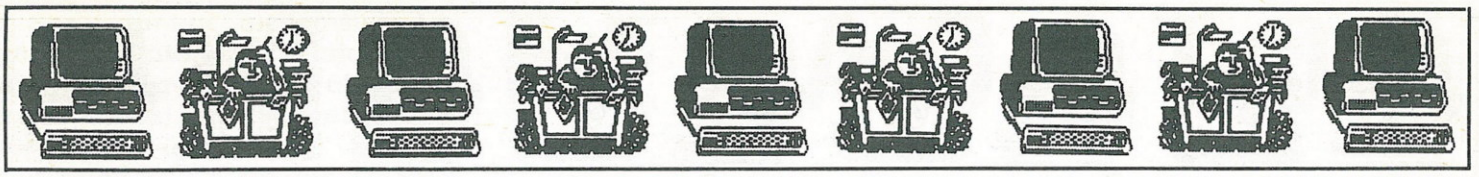
A programot „hexdump" listán mellékeljük. Közlése így a legegyszerűbb, és betöltése is így jár a legkevesebb munkával. Azok számára, akik a programot módosítani kívánják, akár gépi kódban, akár disassemblálás útján, részletesebben is szólunk a program működéséről.

A program működése

A PICTURE utasítás a grafikus képernyők átváltását blokkmozgatással végzi. Erre azért van szükség, mert — bár a TED lehetőséget ad mind a grafikus képernyő, mind a videomátrix áthelyezésére — a BASIC a $2000-től kezdődő képernyőt használja. A program mindig a behívott képernyőt cseréli fel az aktuális képernyővel. Az ehhez szükséges információt a $1003—$100A táblázatból veszi.

Mivel a négy képernyő használatával a BASIC-terület 17 661 bájtra csökken, ez határt szab a BASIC program terjedelmének is. Ha nincs szükség négy képernyőre, akkor a fennmaradó területet fel lehet szabadítani a BASIC számára. Ehhez át kell írni a BASIC-terület kezdőcímét ($002C—$002D) az utolsó grafikus képernyő utáni címre (ezek kezdőcímeinek felső [értékesebb] bájtjai a $1007—$100A rekeszekben vannak tárolva). Ezután a $10AA címen át kell írni az LDY # $03-at a kívánt képernyőszámnál eggyel kisebbre, és a $10A6 címen a CPX # $04 utasítást a képernyők számának megfelelően.

Ha öt képernyőt kívánunk használni, akkor a $1007—$100A címeken lévő táblázatot át kell helyezni a $1008-tól kezdődő területre.



$1004-re #04-et, $100C-re #B8-at kell írni. A $10AA és a $10A6-os címen lévő utasítások operandusait eggyel meg kell növelni, és a BASIC kezdőcímét $E001-re kell beállítani. Ha ezután a BASIC-be visszatérünk és kiadunk egy NEW parancsot, öt képernyőt használhatunk, de a BASIC-terület 7407 bájtra csökken.

Karakterek írása tetszőleges grafikus koordinátákra

A BASIC CHAR utasításának hátránya, hogy a grafikus képernyő használatakor is csak a karakteres képernyő pozícióira lehet szöveget kiíratni. Bizonyos esetekben (zsúfolt rajzok) jól alkalmazható egy olyan program, amellyel BASIC-utasítás formájában a grafikus képernyő tetszőleges pozícióira is írhatunk. Ha a szöveg nem túl hosszú, elfogadható minőségű vízszintes finomscroll is megvalósítható.

E program is BASIC-utasításból érhető el. Formája:

WRITE X,Y,A,B,A$,C

ahol a WRITE után írt paraméterek jelentése a következő:

X és Y a szöveg kezdőpozíciójának bal felső koordinátái,

A és B az X, illetve Y irányú nagyítás (értékeik a program alaphelyzetében 1, 2, 3 és 4),

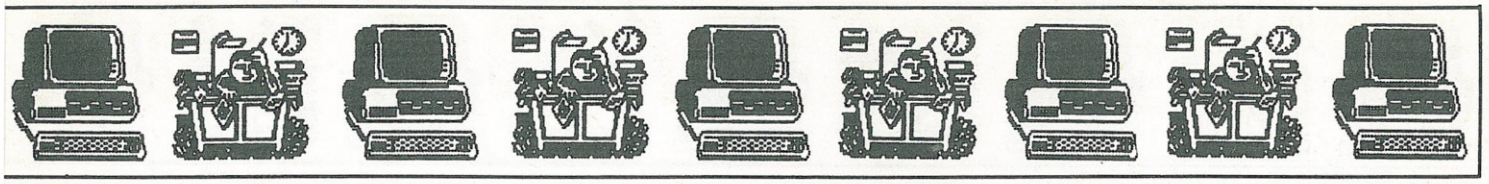
A$ a kiíratni kívánt szöveg,

C az inverzjelző (ha C=0, a kiírás normál, C=1 esetén a kiírás inverz formájú, C elhagyható, ekkor a program 0 értéket feltételez).

Az eddig elmondottakból az is kiderül, hogy a program lehetőséget ad a karakterek nagyítására is. A=1 és B=1 értékek mellett a karakterek mérete az alapméret. Ha 4-nél nagyobb nagyítást akarunk, akkor a $11FD-n lévő CPX#$05 utasítást kell megváltoztatni úgy, hogy az operandus a kívánt maximális nagyításnál eggyel nagyobb szám legyen.

A karaktereket a program a $D000-on kezdődő karaktergenerátorból veszi. Szükség esetén ezt is meg lehet változtatni vagy a $D400 címre, vagy a saját — a RAM-ban elhelyezett — karaktergenerátor kezdőcímére. Ehhez a $1230 címen az ADC#$D0 utasításban át kell írni az operandust, úgy, hogy az a karaktergenerátor kezdőcímének felső (értékesebb) bájtját tartalmazza. Ha a saját karaktergenerátorunk a $8000 cím felett van, a RAM átváltásáról is gondoskodnunk kell.

```
>1000 00 00 00 00 01 02 03 18
>1008 40 68 90 00 00 00 00 00
>1010 78 8D 3F FF B9 F8 FC 8D
>1018 3E FF 58 60 A2 10 F9 10
>1020 B4 11 DB 13 4F 13 50 49
>1028 43 54 55 52 C5 49 4E 56
>1030 45 52 D3 57 52 49 54 C5
>1038 43 4C 4F 43 4B 4F 46 C6
>1040 43 4C 4F 43 CB 00 00 A9
>1048 00 85 8C 85 8D A5 87 60
>1050 A9 10 A0 26 20 07 8A 90
>1058 04 C8 18 90 03 20 79 04
>1060 4C 6A 89 AA A0 10 84 23
>1068 A0 26 84 22 4C 9E 8B E9
>1070 80 0A A8 B9 1D 10 48 B9
>1078 1C 10 48 4C 73 04 50 10
>1080 63 10 6F 10 A0 05 B9 7E
>1088 10 99 0C 03 88 10 F7 A9
>1090 B8 85 2C 84 75 C8 84 2B
>1098 98 91 2B E6 2B 20 7E 8A
>10A0 4C 03 87 20 84 9D E0 04
>10A8 B0 42 A0 03 8A D9 03 10
>10B0 F0 03 88 10 F8 AD 03 10
>10B8 8E 03 10 99 03 10 B9 07
>10C0 10 85 15 A0 00 84 14 84
>10C8 22 A9 18 85 23 A2 28 78
>10D0 8D 3F FF B1 14 48 B1 22
>10D8 91 14 68 91 22 C8 D0 F3
>10E0 E6 15 E6 23 CA D0 EC 8D
>10E8 3E FF 58 60 4C 1C 99 C9
>10F0 01 90 06 D0 F7 C0 40 B0
>10F8 F3 60 20 E1 9D 20 EF 10
>1100 84 F2 85 F3 20 D8 9D E0
>1108 C8 B0 E1 86 87 20 DE 9D
>1110 20 EF 10 20 D8 9D E0 C8
>1118 B0 D2 86 96 20 BF C7 A5
>1120 15 C5 F3 90 C7 D0 06 A5
>1128 14 C5 F2 90 BF A5 96 C5
>1130 87 90 B9 A5 F3 85 8D A5
>1138 F2 85 8C A0 00 20 66 11
>1140 51 57 91 57 E6 8C D0 02
>1148 E6 8D A5 8D C5 15 D0 ED
>1150 A5 14 C5 8C B0 E7 E6 87
>1158 A5 96 C5 87 B0 D5 60 06
>1160 57 2A CA D0 FA 60 A5 87
>1168 29 F8 85 57 A9 00 85 58
>1170 A2 03 20 5F 11 85 5A A6
>1178 57 86 59 A2 02 20 5F 11
>1180 85 58 A5 57 65 59 85 57
>1188 A5 58 65 5A 69 20 85 58
>1190 A5 8C 29 F8 65 57 85 57
>1198 A5 8D 65 58 85 58 A5 87
>11A0 29 07 65 57 85 57 90 02
>11A8 E6 58 A5 8C 29 07 AA BD
>11B0 89 C2 60 EA EA 20 E1 9D
>11B8 C9 01 90 06 D0 46 C0 40
>11C0 B0 42 20 D8 9D E0 C8 B0
>11C8 3B 86 F2 20 F6 11 86 F3
>11D0 20 F6 11 86 F4 20 91 94
>11D8 20 48 9C F0 D5 85 96 8E
>11E0 18 12 8C 19 12 20 A5 C3
>11E8 E0 02 B0 18 8A 6A 66 B1
>11F0 20 BF C7 D0 12 EA 20 D8
>11F8 9D E0 00 F0 05 E0 05 B0
>1200 01 60 68 68 4C 1C 99 A0
>1208 00 84 C0 A5 F4 85 5B A2
>1210 00 20 A9 12 20 B6 12 BD
>1218 F5 FC 8D 3E FF 58 C9 40
>1220 90 02 E9 40 85 FC A9 00
>1228 A2 03 06 FC 2A CA D0 FA
>1230 69 D0 85 FD A4 C0 B1 FC
>1238 85 5C A0 08 84 F6 A5 F3
>1240 85 F1 20 66 11 85 5D A4
>1248 F6 A9 00 38 2A 88 D0 FC
>1250 F0 6B EA 28 F0 04 11 57
>1258 D0 04 49 FF 31 57 91 57
>1260 E6 8C D0 02 E6 8D A5 8D
>1268 F0 16 A9 3F C5 8C B0 10
>1270 A4 F4 20 47 10 69 08 88
>1278 D0 FB C9 C8 B0 11 85 87
>1280 C6 F1 D0 BE C6 F6 D0 B6
>1288 A6 C1 E8 E4 96 D0 85 E6
>1290 F2 A5 F2 C9 C8 B0 11 C6
>1298 5B F0 03 4C 0F 12 A4 C0
>12A0 C8 C0 08 F0 03 4C 09 12
>12A8 60 A5 14 85 8C A5 15 85
>12B0 8D A5 F2 85 87 60 86 C1
>12B8 78 8D 3F FF 60 25 5C 08
>12C0 24 B1 10 08 28 F0 03 98
>12C8 F0 01 8A 08 A5 5D D0 83
>12D0 CE FD 07 10 08 A9 2B 8D
>12D8 FD 07 20 25 13 20 25 13
>12E0 2C FF 07 10 0D A2 02 A0
>12E8 07 B5 A3 20 12 13 88 CA
>12F0 10 F7 20 11 CF AD 10 FD
>12F8 29 04 D0 0D 2C FC 07 30
>1300 0E A5 01 29 F7 85 01 D0
>1308 06 8D FC 07 20 B0 E3 4C
>1310 45 CE 48 20 1B 13 68 4A
>1318 4A 4A 4A 29 0F 18 69 30
>1320 99 00 0C 88 60 EE FE 07
>1328 A9 33 20 F5 13 D0 1F A2
>1330 02 B5 A3 F8 69 00 C9 60
>1338 D0 11 A9 00 95 A3 CA D0
>1340 F0 A5 A3 69 00 C9 24 D0
>1348 02 A9 00 95 A3 D8 60 EA
>1350 F0 4D 20 84 9D E0 28 B0
>1358 6E 86 14 20 D8 9D E0 19
>1360 B0 65 A5 14 7D 02 D8 8D
>1368 21 13 BD 1B D8 69 00 8D
>1370 22 13 20 79 04 F0 28 20
>1378 91 94 20 48 9C 8E 15 10
>1380 8C 16 10 C9 06 D0 40 A8
>1388 4A AA 20 CD 13 85 15 20
>1390 CD 13 0A 0A 0A 0A 05 15
>1398 95 A2 CA D0 ED EA EA A5
>13A0 A3 C9 24 B0 0A A9 5A C5
>13A8 A4 90 04 C5 A5 B0 08 A9
>13B0 00 85 A3 85 A4 85 A5 A9
>13B8 FF 8D FF 07 A9 D0 8D 12
>13C0 03 A9 12 8D 13 03 60 4C
>13C8 1C 99 4C 24 93 88 20 10
>13D0 10 C9 3A B0 F5 C9 30 90
>13D8 F1 29 0F 60 20 84 9D E0
>13E0 02 B0 E4 E0 00 F0 0A A9
>13E8 F2 8D 12 03 A9 12 8D 13
>13F0 03 8E FF 07 60 CD FE 07
>13F8 D0 05 A9 00 8D FE 07 60
```



Grafikus alakzatok invertálása

Elsősorban grafikus programok készítésénél jól jöhet az a lehetőség, hogy egy meghatározott alakzatot (téglalap befoglaló formában lévő ábra- vagy szövegrészletet) invertálni, esetleg villogtatni tudunk. Ezt a lehetőséget a BASIC nem nyújtja. Erre a célra készítettük azt a gépi kódú alprogramot, amelyet BASIC-ből az

INVERZ X1, Y1, X2, Y2

utasítással érhetünk el. X1 és Y1 a téglalap bal felső, X2 és Y2 a téglalap jobb alsó koordinátáit jelöli. Ha az alakzat nem túl nagy, az egymást követő invertálások még elfogadható minőségű villogást eredményeznek.

Rezidens szoftveróra

Egyes feladatok (például ernyőképek tervezése) megoldása során szükség lehet egy olyan utasításra, amellyel a futó programtól függetlenül megjeleníthetjük az időt a képernyő tetszőleges pozícióján. Erre a célra készült a BASIC utasításként hívható gépi kódú alprogram.

Az utasítás szintaxisa:

CLOCK X, Y, A$

illetve

CLOCKOFF P

A CLOCK utasítás a végrehajtás során átírja a raszter megszakítási vektort, ezzel kikapcsolja az eredeti rendszerórát. Ennek a helyén fog működni az új óra. Ha az utasítást nem követik paraméterek, akkor az óra bekapcsolása után minden rendszermegszakításnál az óra kiíródik az utoljára meghatározott helyre. Ha az utasítás után X, illetve Y paraméter áll, akkor az óra az X (vízszintes) és az Y (függőleges) karakteresképernyő-pozícióktól kezdődően kerül a képernyőre. További paraméter az A$, amellyel be lehet állítani az órát. Ennek hatkarakteres sztringnek kell lennie: két-két számjegy tartalmazza az órát, a percet és a másodpercet. Érvénytelen adatok esetén (például 25 óra, 71 perc stb.) az óra 00 00 00 állásból indul. A CLOCKOFF utasítással az óra kikapcsolható. Ha az utasításban paraméterként 0-t adunk meg, az óra tovább „jár", csak kijelzésre nem kerül. Az óra leállítása a CLOCKOFF 1 utasítással végezhető el; ilyenkor természetesen a képernyőre kiírás is megszűnik. Az óra egy később kiadott CLOCK utasítással újraindítható.

A program működése

A feladat megoldásához egy „saját" órát működtetek a gépben. Erre egyrészt azért volt szükség, mert a rendszerórát csak bizonyos átváltások után lehetne kiíratni, ami lassítaná a gép működését (az átváltások csak az interrupt rutin alatt végezhetők el), másrészt a Plus/4 órája meglehetősen pontatlan (1 óra alatt 10—40 másodperc hibát is tapasztaltam különböző gépeken).

A program annak megfelelően működik, hogy az óra engedélyezve van-e. Ha igen, akkor minden megszakítás után „növeli" az órát, tartalmát kiírja a képernyőre és ezután elvégzi a magnókezelést. Ha az óra működése tiltva van, csak a magnókezelést végzi el, majd visszatér a megszakítás rutinba.

Maga az óra a $ØØA3—$ØØA5 címeken található. Az egy-egy bájtban lévő két számjegy BCD kódú, így a kiírás gyorsan és egyszerűen megoldható. Az óra pontos beállítása — ha ez szükséges — a $12Ø6 és a $1329 címeken lévő időzítők átírásával végezhető el.

A programok használata

A négy új utasítás alprogramjai egyetlen gépi kódú programként készültek el.

A programot MONITOR üzemmódban kell beírni a mellékelt hexdump lista alapján. Ezt a munkát nagy figyelemmel végezzük el, és alaposan ellenőrizzük. Rövidke BASIC programokkal próbáljuk ki a lehetőségeket, csak ezután lássunk neki a komolyabb feladatoknak. A program bevitelekor végzett gondos munka sok kellemetlenségtől kímélhet meg.

A bevitt programot (szintén MONITOR-ból) az

S "programnév",1,1ØØ1,14ØØ

paranccsal menthetjük ki kazettára. Lemezre íráskor a perifériaszámot értelemszerűen módosítanunk kell.

Az adathordozón lévő program MONITOR-ból és BASIC-ből egyaránt betölthető. Indítása MONITOR üzemmódban a G1Ø84 paranccsal lehetséges. Az elindított program először átírja a felhasználói BASIC-utasításra vonatkozó részeket, lefoglalja a négy grafikus képernyő helyét, a BASIC-terület kezdetét $B8Ø1-re helyezi át, majd visszatér az interpreterhez. Ezután az új utasítások az ismertetett szintaxisszabályok betartásával a standard utasításokhoz hasonlóan használhatók.

Nagy Ferenc

FELADATOK — MEGOLDÁSOK

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 majd, 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!

7. feladat: Tizenötös játék 1.

Írjon programot, amely kirajzolja a tizenötös játékot, lehetővé teszi a számok megfelelő mozgatását, és össze tudja keverni a játékot! Ügyeljen arra, hogy az összekevert állás kirakható legyen!

MEGOLDÁS

A szokásostól eltérően vizsgáljuk meg rögtön a programot! Ezt azért tehetjük meg, mert a ZX—Spectrumra Beta Basic 3.1-ben írt program olyannyira strukturált felépítésű, hogy még a nyelvet nem ismerők is követhetik.

Ahhoz, hogy a táblát kényelmesen kezelhessük, annak valamilyen belső ábrázolást kell választani, célszerűen egy 4×4-es tömböt. A programunkban ez t$(4,4). A tömb elemei a számok egytől tizenötig, a táblán levő számoknak megfelelően és az üres mezőt jelképező nulla.

Mivel az üres mező helye mind a mozgatáskor, mind a kirajzoláskor különösen fontos, a tx és ty ennek koordinátáit tartalmazza.

Most vizsgáljuk meg a képet! Jól láthatóan két részre tagolódik: egy álló — a játék összekeverése során sem változó részre, amely a keretből és a tájékoztató részből áll — és a számokból álló részre, amely mozgatható.

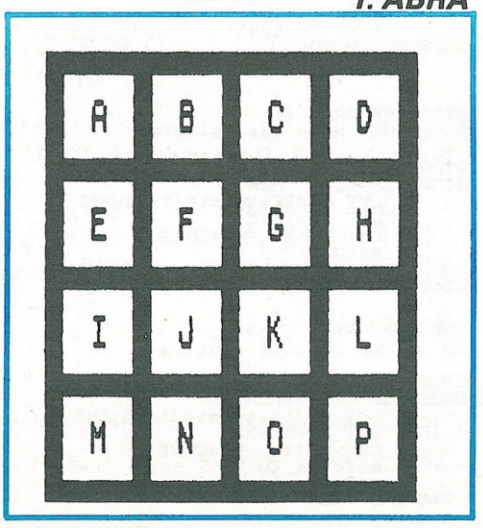
A kiindulási helyzet beállítása ily módon három fő részre tagolódik: a kép nem változó részének kirajzolására, t$ feltöltésére a kiindulási helyzetnek megfelelően, és a t$-ben tárolt állás megjelenítésére a képernyőn. E három feladatot látja el az alapszubrutin.

A kép nem változó részének kirajzolásához a Kép nevű rutint hívja meg. Ez jóformán csak PRINT utasításokból áll. A kiindulási helyzet beállítását az 147Ø—15ØØ-ig látható kettős ciklus végzi. Ezzel egy időben megjelennek a számok a képernyőn, a mező szubrutin segítségével. A szubrutin két paramétere a kirajzolandó mező két koordinátája. Az 125Ø-es sorban látható számítás egyedüli célja, hogy a megfelelő nagyságú szám a számára előírt helyzetbe, pontosan a mező közepére kerüljön. Ugyanitt töröljük az üres mezőt is.

A négyirányú mozgatást a le, fel, jobbra és balra eljárások végzik. Felépítésük annyira hasonló, hogy elegendő egyet megvizsgálni. Nézzük tehát a le nevű rutint!

Lefelé mozdítani csak akkor lehet, ha az üres mező nem a legfelső sorban áll, azaz ha ty>1. Ekkor a mozdítás hatására az üres mező (tx, ty) és a fölötte álló mező (tx,ty—1) helyet cserél. Semelyik másik nem változik, így elég e kettőnek az újrarajzolása.

1. ÁBRA



Végül ty-t az üres mező új helyének megfelelően kell változtatni.

Ezek után a főprogram megértése nem jelenthet különösebb nehézséget. A kiindulási helyzet beállítása után egy végtelen ciklus belsejében történik a billentyűzet vizsgálata és a lenyomott billentyűnek megfelelő szubrutin meghívása. A feltételek kicsit összetettebbek, hogy lehetővé tegyék a játék irányítását Kempston botkormánnyal is, a képernyőn feltüntetett billentyűkön kívül.

A feladat legnehezebb része, a tábla összekeverése azonban még hátra van. Nem elegendő ugyanis a számokat kiborítani a dobozból és valamely tetszőleges sorrendben visszatenni, mivel az összes lehetséges elrendezésnek (ezek száma 16!= 20 922 789 888 000) csak a fele rakható ki.

A következő algoritmus alkalmas annak eldöntésére, hogy egy adott állás kirakható avagy nem.

1. Legyen n az elrendezésben az A pozícióban levő szám (lásd az *1. ábrát*). Számolja meg, hogy az A-t követő betűpozícióba hány n-nél kisebb szám van! (Az üres hely „értéke" legyen most 16.)

2. Végezze el ugyanezt a műveletet mind a 16 pozícióra, A-tól P-ig, és adja össze a kapott eredményeket!

3. Ha az üres mező a B, D, E, G, J, L, M, O mező valamelyike, adjon hozzá az összeghez egyet.

4. Az adott elrendezés kirakható, ha az összeg páros.

Tehát az összekeverést megvalósíthatjuk úgy, hogy a számokat tetszőlegesen összekeverjük, majd az előbbi algoritmussal a kapott állást ellenőrizzük. De mit tegyen az, aki nem ismeri ezt az algoritmust? Megoldható a feladat enélkül is?

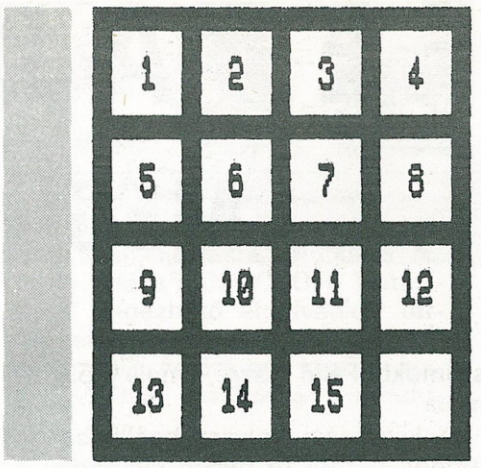
Lehetséges olyan megoldás, hogy a számokat nem vesszük ki a dobozból, hanem csak egymás mellett elcsúsztatjuk. Így biztosan kirakható álláshoz jutunk.

Ezen az elven működik a Kever nevű szubrutin. A h$-be a 3Ø5Ø-es sorba 1234 kerül. Mindegyik szám egyiránynak felel meg úgy, hogy az ellen-

tétes irányok kódjának összege 5, azaz balra (1)+jobbra (4)=5=le (2)+fel (3). A négy irányból ezután töröljük azt, amit nem lehet meglépni a tábla széle miatt (3070—3100) és az előző iránnyal (r) ellentéteset. Ez utóbbi az oda-vissza mozgatás elkerülésére szolgál.

A 3110—3130-as sorokban 1$-be a még nem törölt elemeket, a lehetséges mozgatási irányokat gyűjtjük össze, majd a 3140—3150-ben ezek valamelyike véletlenszerűen kiválasztott irányba mozgatható. Ez ismétlődik mindaddig, míg a játékos valamely billentyű lenyomásával meg nem állítja.

A most következő feladat bizonyára alaposan próbára teszi minden feladatmegoldónk tudását.



2. ÁBRA

8. feladat: Tizenötös játék 2.

Írjon programot (algoritmust), amely ki tudja rakni a tizenötös játékot *(2. ábra)*! A megoldásban hivatkozhat a most közölt program bármely szubrutinjára.

Pintér Gábor

```
 10 REM  Tizenotos jatek
          1988.06.29.
          Pinter Gabor

 20 RUN 6000
 30 CLEAR
    DEF KEY ERASE
    LET rt=PEEK 23730+256*PEEK 2373
    1
    SAVE "15/2 V1.0" LINE 40
    POKE PEEK 23631+256*PEEK 23632+
    2,181
    PAUSE 30
    SAVE "cBeta 3.1"CODE rt+1,65536
    -rt
    STOP
 40 BORDER 7
    PAPER 7
    INK 0
    CLEAR rt
    LOAD "cBeta 3.1"CODE
    CLS
    RANDOMIZE USR 58419
 50 LIST FORMAT 1
    CSIZE 7,8
 60

1000 DEF PROC kep
1010  REM A kep nem valtozo
       reszet rajzolja ki.

1020  LOCAL i
1030  CLS
1040  PRINT CSIZE 8,11;AT 1,22;"TIZE
      NOTOS"'TAB 24;"JATEK"''TAB 23
      ;"5-BALRA"'TAB 23;"6-FEL"'TAB
      23;"7-LE"'TAB 23;"8-JOBBRA"''T
      AB 23;"0-KEVERD"'TAB 25;"OSSZE
      "''TAB 23;"SPACE-"'TAB 25;"RAK
      D KI"
1050  CSIZE 8
      INK 5
      PRINT AT 0,0
1060  FOR i=1 TO 21
       PRINT "                    "
      NEXT i
1070  INK 0
1080 END PROC


1200 DEF PROC mezo x,y
1210  REM Az (x,y) mezot
       rajzolja meg.

1220  LOCAL s$
1230  LET s$=STR$ CODE t$(x,y)
1240  CSIZE 32
1250  IF CODE t$(x,y) THEN
       PLOT PAPER 6;40*x-32,199-40*y
       "  "
       CSIZE 8,16
       PLOT PAPER 6;40*x-16-4*LEN s$
       ,191-40*y,s$
      ELSE
       PLOT PAPER 5;40*x-32,199-40*y
       "  "
1260  CSIZE 8
      PAPER 7
1270 END PROC
```

```
1400 DEF PROC alap
1410  REM A kezdoallapot
       beallitasa.

1420  LOCAL i
1430  DIM t$(4,4)
      REM A tabla.
1440  LET tx=4
      LET ty=4
      REM Az ures mezo helye.
1450  RANDOMIZE
1460  kep
1470  FOR i=1 TO 4
       FOR j=1 TO 4
1480    IF i=4 AND j=4 THEN
         LET t$(j,i)=CHR$ 0
        ELSE
         LET t$(j,i)=CHR$ (4*i+j-4)
1490    mezo j,i
1500   NEXT j
      NEXT i
1510 END PROC


2000 DEF PROC le
2010  REM Mozditas lefele.

2020  BEEP .02,30
2030  IF ty>1 THEN
       LET t$(tx,ty)=t$(tx,ty-1)
       LET ty=ty-1
       LET t$(tx,ty)=CHR$ 0
       mezo tx,ty
       mezo tx,ty+1
2040 END PROC


2100 DEF PROC fel
2110  REM Mozditas felfele.

2120  BEEP .02,27
2130  IF ty<4 THEN
       LET t$(tx,ty)=t$(tx,ty+1)
       LET ty=ty+1
       LET t$(tx,ty)=CHR$ 0
       mezo tx,ty
       mezo tx,ty-1
2140 END PROC

2200 DEF PROC jobbra
2210  REM Mozditas jobbra.

2220  BEEP .02,17
2230  IF tx>1 THEN
       LET t$(tx,ty)=t$(tx-1,ty)
       LET tx=tx-1
       LET t$(tx,ty)=CHR$ 0
       mezo tx,ty
       mezo tx+1,ty
2240 END PROC
```

```
2300 DEF PROC balra
2310  REM Mozditas balra.

2320  BEEP .02,23
2330  IF tx<4 THEN
       LET t$(tx,ty)=t$(tx+1,ty)
       LET tx=tx+1
       LET t$(tx,ty)=CHR$ 0
       mezo tx,ty
       mezo tx-1,ty
2340 END PROC


3000 DEF PROC kever
3010  REM A jatek osszekeverese.

3020  LOCAL i,r,h$,l$
3030  LET r=1
3040  DO
3050   LET h$="1234"
3060   LET h$(5-r)=" "
3070   IF tx=4 THEN
        LET h$(1)=" "
3080   IF tx=1 THEN
        LET h$(4)=" "
3090   IF ty=4 THEN
        LET h$(3)=" "
3100   IF ty=1 THEN
        LET h$(2)=" "
3110   LET l$=""
3120   FOR i=1 TO 4
        IF h$(i)<>" " THEN
         LET l$=l$+h$(i)
3130   NEXT i
3140   LET r=VAL l$(INT (LEN l$*RND)
       +1)
3150   ON r
        balra
        le
        fel
        jobbra
3160  LOOP WHILE INKEY$="" AND (IN 3
      1>=32 OR IN 31=0)
3170  DO
      LOOP UNTIL INKEY$="" AND (IN 3
      1>=32 OR IN 31=0)
3180 END PROC


6000 REM A foprogram
6010 alap
     LET tn=0
6020 DO
6030  LET k$=INKEY$
      LET k=IN 31
6040  IF (k=0 OR k>=32) AND k$="" TH
      EN
       GO TO 6030
6050  IF k<4 AND k>=2 OR k$="5" OR k
      $=CHR$ 8 THEN
       balra
6060  IF k<2 AND k>=1 OR k$="8" OR k
      $=CHR$ 9 THEN
       jobbra
6070  IF k<8 AND k>=4 OR k$="6" OR k
      $=CHR$ 10 THEN
       le
6080  IF k<16 AND k>=8 OR k$="7" OR
      k$=CHR$ 11 THEN
       fel
6090  IF k<32 AND k>=16 OR k$="0" TH
      EN
       kever
6130 LOOP
6140
```

Takarékos megszakítás ZX-Spectrumra

Az alábbi gépi kódú rutinnal az IM 2-es megszakítás használatakor ZX-Spectrum memóriájában, helyet lehet megtakarítani.

A megszakítás rutinjának címét a géphez kapcsolt perifériáról érkező bájt (alapgépnél ennek értéke 255) és az I regiszter határozza meg, I 256+bájt módon. Így kapunk egy címet, ami az általunk elkészített vektortáblázat egyik elemére mutat, ahol megtalálható az interrupt rutin címe.

A táblázat 256 bájtot foglal el, ha különböző perifériák alkalmazásakor, különböző rutinokat akarunk végrehajtatni. Ha azonban mindig egy helyen szeretnénk folytatni a programot, akkor ehhez 128-szor kell ugyanazt a címet tárolni.

A ROM segít takarékoskodni, mert kihasználatlan területei 255-tel (FFh) feltöltöttek, mint például 16128—16383-ig (3F00h—3FFh) terjedő része. Ez lesz a vektortáblázat. Megszakításkor a 65535-ös(FFFFh) címre kerül a vezérlés. Ezután az első értelmezési hely a 0. (0000h) lesz, amelyen egy DI utasítás található. Ez a ROM első bájtja. A DI kódja 243 (F3h). Ha egy JR utasítást teszünk elé, akkor ez 13 lépést jelent vissza, vagyis a program a 65524 (FFF4h) címen folytatódik.

Ahogy a példa is mutatja, mindössze 12 bájtot foglalunk le, illetve még ennyit sem, hiszen a megszakítás bekapcsolása elfér a két ugrás között. Ellenőrzésnek egy vonalka jelenik meg a képernyő bal felső sarkában, s ez egészen a kikapcsolásig marad is.

Bekapcsolás: 65527 (FFF7h)
Kikapcsolás: 65516 (FFECh)

Lugossy Norbert

```
                ORG #FFE2
FFE2 INTR PUSH AF           ;INTERRUPT
FFE3      LD   A,#FF        ;RUTIN
FFE5      LD   (#4000),A
FFE8      POP  AF
FFE9      RST #38
FFEA      RETI
FFEC      LD   A,#3F        ;KIKAPCSOLAS
FFEE      LD   I,A
FFF0      IM   1
FFF2      EI
FFF3      RET
FFF4      JP   INTR         ;UGRAS AZ
                            INTERRUPTRA
FFF7      LD   A,#3A        ;BEKAPCSOLAS
FFF9      LD   I,A
FFFB      IM   2
FFFD      EI
FFFE      RET
FFFF      DEFB #18          ;A #0000-VAL
                            ;JR #F8 LESZ
          (C) 1987 BY L'SSY SOFT
```


```
 1          1 VOL 8
10 GRAPHIC3,1
20 CIRCLE1,18,38,18:CIRC-
LE1,140,38,18
30 CIRCLE1,18,160,18:CIRC-
LE1,140,160,18
40 FORI=1TO16
50 FORO=1TO7
55 FORX=1TO11:NEXTX
60 COLOR3,I,O:PAINT3,18,38,1:PA-
INT3,140,38,1:PAINT3,18,160,1:PA-
INT3,140,160,1
61 COLOR4,I,O
70 NEXTO
80 NEXTI
90 GOTO 40
```

ÖTLETEK C16-HOZ ÉS C PLUS/4-HEZ

NTSC üzemmód. — Könnyen zavarba ejthetjük számítógépes barátunkat, ha begépeljük az alábbi parancsot, amikor nem néz a gépre, illetve a tévére. A parancs begépelése után zavarossá, csíkossá válik tévékészülékünk képernyője. Persze ez az apró csalafintaság is felhasználható, esetleg RESET gomb megnyomásakor a program zavarossá varázsolja a képernyőt, zavarba ejtve ezzel a felhasználót. A hatás lényegében annak köszönhető, hogy a gépet az utasításunkkal NTSC üzemmódba kapcsoljuk PAL helyett. NTSC üzemmódra váltó parancs: POKE65287,PEEK(6587)OR 64 Visszakapcsolás PAL üzemmódra: POKE65287,PEEK(65287)AND 191

Érdekes fényhatást mutat be az alábbi program a sokszínű grafikus képernyőn. Érdemes ellesni a program trükkjét, mert a saját programunkban is könnyen felhasználhatjuk!!!

Érdemes a programot úgy továbbfejleszteni, hogy ne látsszon, ahogy a körök kialakulnak és ahogy befestjük őket feketére. Én úgy gondolom, a kialakulás idejére érdemes a képernyőt feketére váltani, majd ha a kialakulás megtörtént, akkor újra fehérre váltani. A programot még érdekesebbé tehetjük, ha hanggal kombináljuk az alábbi módon:

61 COLOR4,I,O:SOUND1,ix63x61,1

így a feljebb leírt módon kijavított program már meg is szólal. A programon még sok mindent tovább lehet fejleszteni, más ötlettel kombinálni, de ezt már az olvasókra bízom.

Hangtorzítás. — Az alábbi módon könnyen változtathatjuk a zenei programunk által kiadott hang torzságát.

SOUND1,O,U:SOUND2,P,G

A legkisebb torzítást érhetjük el, ha az O és a P változó értékei egyenlőek. Legnagyobb ajánlott torzítást úgy érhetjük el, ha az O és P változó különbsége 10. Ennél nagyobb torzítás nem ajánlott. Akkor is érdekes hanghatást érhetünk el, ha az U és a G változó értéke különböző, mert így a 2. oszcillátoron hosszabb ideig szól a hang mint az elsőn, ennek következtében különböző hatású dallamokat játszathatunk le gépünkkel.

Különleges hangzás! — Az előbb leírt torzításnál is izgalmasabb hangzással kecsegtet a következő módszer, amit ugyancsak könnyen beépíthetünk saját programunkba.

10 fori=1TO10:reada,b:sound1,a,b:sound2,a1,b:sound1,1020,4

20 data"

itt vannak elhelyezve az A és a B változó értékei, azaz a dallam adatai. Egyszerűen összehasonlíthatjuk a „SIMA" sound utasítással keltett hangot a fentebb leírt módszer által elhangzott hanggal. A különbséget még egy „botfülű" is megérzi.

Bácsi Péter

Mindenki, aki foglalkozott már számítástechnikával, sőt még programokat is ír, más és más stílusban oldja meg a feladatokat. A módszerek között vannak jók és kevésbé jók, vagy inkább célszerűek és célszerűtlenek.

Programozói szemmel nézve nagyon sok dolgot figyelembe kell venni egy-egy program, programrendszer megírásánál. Ezek a problémák kis programocskáknál nem is olyan feltűnőek, de egy terjedelmesebb munkánál már katasztrofális zűrzavarhoz vezethetnek. Mindig az a legfontosabb, hogy figyeljünk a program tagoltságára, átláthatóságára, mert ha netán később javítani vagy bővíteni kell az adott szoftvert, akkor így sokkal gyorsabban tudjuk ezt megtenni. Természetesen ha nagyon nagy vagy munkaigényes a feladatunk, minden esetben blokksémát és feladattervet kell készíteni, amit manapság nagyon sokan elhanyagolnak. Ez az oka annak, hogy olyan kevés hazánkban az igazán színvonalas szoftver.

Cikkemben — egy kicsit elszakadva a házi számítógépektől — inkább az IBM PC felé tekintek, s a példáimat is ebből a világból merítem. Az érthetőség kedvéért az *1. listán* bemutatom a rossz, a *2. listán* pedig a jó programot egy adott feladatra (IBM PC XT/AT, PL/I fordító).

Az első program nagyon átláthatatlan, amiért már az ilyen kis feladatnál is gondban lehetünk a megfejtéssel. A második program ugyanazt a feladatot oldja meg: szalagról a fájl végéig olvassa a 80 bájtos rekordokat, s ha a 9-10. bájt az „EZ" szöveget tartalmazza, akkor az egész kártya tartalmát kinyomtatja. Ugyanúgy persze, mint az előző kis program, de érezhető a különbség a kivitelezésben.

Természetesen ez más nyelveknél is fontos követelmény, mivel nagyon kellemetlen, ha később a javításnál vagy bővítésnél az áttekinthetetlenség miatt szinte új programot kell írnunk.

Bartos Gyula

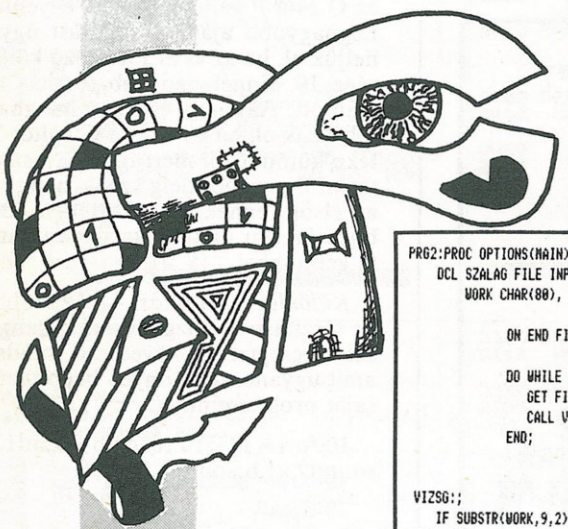
Programozási stílusok

Csiszolatlan rövidség

```
PRG1:PROC OPTIONS(MAIN);

DCL SZALAG FILE INPUT ENV(MEDIUM(SYS001,2400)F(80)), WORK CHAR(80),
    ON END FILE(SZALAG) GOTO VEG;
    DO WHILE('1'B);
    GET FILE(SZALAG) EDIT(WORK)(A(80));
    IF SUBSTR(WORK,9,2)='EZ' THEN DO PUT SKIP EDIT(WORK)(A(80)); END;
    END;
 VEG:END PRG;
```

1. lista



2. lista

```
PRG2:PROC OPTIONS(MAIN);
    DCL SZALAG FILE INPUT ENV(MEDIUM(SYS001,2400)F(80),
        WORK CHAR(80),

            ON END FILE(SZALAG) GOTO VEG;

            DO WHILE ('1'B);
               GET FILE(SZALAG) EDIT(WORK)(A(80));
               CALL VIZSG;
            END;


VIZSG:;
   IF SUBSTR(WORK,9,2)='EZ' THEN CALL PRINT;
END VIZSG;


PRINT:;
   PUT SKIP EDIT(WORK)(A(80));
END PRINT;


VEG:;
   END PRG;
```

A TÖMÖRÍTÉS TERMÉSZETRAJZA

BÁJTPRÉS

Egyre több olyan program készül, amelynek feladata, hogy programokat és adatállományokat tömörítsen, azaz rövidebbé tegyen. Könnyítendő a dolgon, ilyen programok írásához kívánunk segítséget nyújtani.

Tulajdonképpen minden tömörítő azonos vázra épül:

a) A rövidítendő fájl beolvasása. Ehhez általában saját rutint kell írni, ami a $D000—$DFFF közötti RAM-ba — a CIA, VIC, SID chipek alá — is tud tölteni. Ezután a betöltött programot legtöbbször rámásolják a memória végére.

b) Tömörítés valamilyen algoritmus szerint. Ez az a pont, ahol az egyes tömörítők különböznek egymástól.

c) A szétpakoló hozzáfűzése a tömörített állomány elejéhez. Erre azért van szükség, mert a tömörített program önmagában működésképtelen, elindítás előtt vissza kell állítani az eredeti állapotát („szét kell pakolni"). Ez természetesen azt jelenti, hogy csak akkor érdemes tömöríteni, ha több bájttal rövidül a program, mint ahány bájt hosszú a szétpakoló.

d) Kimentés. Ezután illik kipróbálni a tömörített programot.

A továbbiakban megkíséreljük felvázolni néhány tömörítőalgoritmus alapjait.

A bájtos tömörítés

Ezzel a módszerrel lehet leghatékonyabban tömöríteni az egymás után következő azonos bájtokat. Az eljárás lényege: végigolvassuk a rövidítendő fájlt, és ha közben egymás utáni azonos bájtokat találunk, akkor azokat a következő három bájttal helyettesítjük:

1 — jelzőbájt,
2 — az ismétlődő bájt,
3 — hányszor ismétlődik.

A jelzőbájt jelzi a szétpakolónak, hogy a következő bájtot ismételnie kell. Elvileg bármelyik 0 és 255 közötti számot kinevezhetjük jelzőbitnek (például a $00-t, mert azt könnyű letesztelni). Ajánlatosabb azonban azt az értéket választani, ami az eredeti állományban a legkevesebbszer fordul elő. Ez azért jó, mert a jelzésre használt bájtot akkor is tömöríteni kell, ha csak egyszer szerepel. Ellenkező esetben — ha például a $A2 bájtot használjuk jelzőbájtnak — a szétpakoló egy $A2 olvasása után nem tudná eldönteni, hogy az a tömörítést jelzi-e, vagy csak egy egyszerű $A2 bájt volt

az eredeti fájlban. Tehát az adott esetben az eredeti programban lévő minden egyes $A2 helyére a következő sorozat kerül: $A2 $A2 $Ø1. Mivel ez két bájt veszteség, ezért a jelzőbájtot úgy érdemes megválasztani, hogy ilyen eset minél kevesebbszer forduljon elő.

Mivel az ismétlések számát is egy bájton tároljuk, ezért maximum 255 egyforma bájtot lehet három bájtra tömöríteni. Ha például $3D a jelzőbájt, akkor 400 darab $FA bájt így tömöríthető: $3D $FA $FF $3D $FA $91.

Egy konkrét példa. Tételezzük fel, hogy a $BB fordult elő a legkevesebbszer az eredeti programban.

Eredeti:

$72 $AØ $1D $1D $1D $1D $1D $1D $1D $1D $1D $1D $1D $1D $9B $BB $41

Tömörített:

$72 $AØ $BB $1D $ØB $9B $BB $BB $Ø1 $41

Ez a legegyszerűbb bájtos tömörítés. A hatékonyság növelése érdekében ezt lehet még egy kicsit bonyolítani.

Általában 10 kbájtos méretig még van remény arra, hogy létezik olyan bájt, amelyik egyszer sem fordul elő az eredeti állományban. Ebben az esetben a fent említett probléma nem fordulhat elő. Az ennél nagyobb fájlokat logikai blokkokra bonthatjuk úgy, hogy minden blokkban csak 255-féle bájt legyen. Minden blokk elejére fel kell írni a következő blokk kezdőcímét (2 bájt) és azt a bájtot, amelyik nem szerepel benne (ezt használjuk majd jelzőbájtnak az adott blokkban). Ezek után már nem fordulhat elő, hogy 1, 2 vagy 3 egyforma bájtot kell tömörítenünk. Tehát a tömörítés harmadik bájtja (darabszám) jelentheti azt, hogy az azonos bájtok száma mennyivel több négynél.

Megtehetjük azt is, hogy nem 255, hanem 254 különböző bájt megtalálásáig bővítjük az egyes blokkokat. Így két jelzőbájtunk lesz, a másikkal mindig a leggyakoribb bájt tömörítését jelezzük. Történetesen ha a két jelzőbájt $F5 és $47, a leggyakrabban tömörített bájt a $FF, akkor 100 darab $ØØ tömörítése: $F5, $ØØ $6Ø ($6Ø=96, négynél ennyivel több bájtot tömörítettünk), 32 darab $FF tömörítése: $47 $1C. Ilyenkor persze a másik jelzőbájtot is fel kell írni a blokk elejére, valamint azt a bájtot is, amelyiknek a tömörítését jelzi.

Természetesen ezek a finomítások bonyolultabb — azaz hosszabb és lassúbb — szétpakolót igényelnek, így nem biztos, hogy mindig érdemes alkalmazni őket. Egy intelligens tömörítő többféle algoritmussal próbálkozik, kiválasztja a leghatékonyabbat, és azzal tömörít.

(Folytatjuk)

Deierl András—Molnár Imre

Kedves Olvasónk!

Ha netán a figyelmed elkerülte, vagy most veszed először a Magazinunkat kezedbe, megismételjük a tavaly októberben megjelent felhívásunkat. Mert bizonyára neked is van otthon számítógéped, és biztos neked is okozott problémát programkészleted hiányossága, vagy egyszerűen jó lenne valakivel — valakikkel — a gépen keresztül ismeretséget kötni. Ebben próbálunk segíteni annyiban, hogy elkezdjük a hozzád hasonló érdeklődésű, s hozzánk levelet írók címeinek összegyűjtését.

Írj te is! Írd meg, hogy kikkel, milyen típusú gépről, milyen szinten kívánsz levelezni. Küldj levelet, benne egy saját címeddel ellátott, felbélyegzett borítékot, és munkánkért cserébe egy számítógépekkel kapcsolatos írást, rövidebb programot, trükkök ismertetését, véleményt vagy valamit, ami szerinted érdekes lehet egy újság hasábján. (Tegyük hozzá, ez utóbbi kérésünknek, noha sokan üdvözölték kezdeményezésünket, csak néhányan tettek eleget. Reméljük a jövőben nemcsak címkívánsággal árasztotok el bennünket. — sic!)

A borítékokat márciusban fogjuk visszaküldeni az addig összegyűlt címek listájával. Ha az akció sikerül, természetesen folytatjuk, s újabb leveled esetén újabb címeket küldünk.

Csipegető, azaz a régi-új
Diákrovat szerkesztősége

Pályázati felhívás

Az 1989 márciusában megrendezésre kerülő IV. Nemzetközi Mikroszámítógépes Találkozó keretében megrendezzük az

OKTATÓPROGRAMOK

egyéni és csapat országos versenyét, 1989. március 18-án és 19-én.

A kiírt kategóriák: **Biológia—kémia**
Fizika
Humán (két különböző témájú)
Matematika
Nyelv
Szakirányú (két különböző témájú)
Egyéb (két különböző témájú)

Az egyéni versenyben bárki (kivéve kisvállalkozások és vállalatok) részt vehet, és az adott kategória díjáért versenyezhet! Egy kategóriában minden pályázó csak egy programmal szerepelhet.
A csapatversenyben iskolák indulhatnak (általános iskola, szakmunkásképző, szakközépiskola, gimnázium és felsőoktatás csoportban). Minden csapatnak legalább 5 különböző kategóriában kell előre beneveznie. Egy iskola több csapattal is benevezhet. A csapat tanárokból és diákokból állhat (de legalább egy tanulónak önálló programmal kell rendelkeznie).
Jelentkezési határidő: 1989. február 12.
Cím: NJSZT „Oktatóprogramok", 1368 Bp. V. ker. Pf. 240
Jelentkezési lapjukon a program nevét, rövid leírását és azt is tüntessék fel, milyen gépre, milyen nyelven készült.

Versenybizottság

Pályázati felhívás

Az utóbbi időben egyre több család rendelkezik otthonában is számítógéppel. A munkahelyek többségében is terjed a professzionális gépeket felhasználó (IBM PC/XT, IBM/AT) számítástechnika. Fontos lenne — különösen a részmunkaidőben dolgozóknak —, hogy munkahelyi feladatukat otthon is tudják végezni! A probléma nem megoldott hazánkban, és a távközlési infrastruktúra fejletlensége még sokáig fogja akadályozni az adatátvitel hálózaton történő megoldását.

Ezért hirdetjük meg a

MUNKAHELY-OTTHON

című pályázatot.

Feltételek: Adott egy tetszőleges otthoni gép (PC) és egy tetszőleges típusú munkahelyi gép. Köztük az adathordozó valamilyen mágneses eszköz (floppy vagy kazetta).
Elkészítendő és bemutatandó: mindkét gépet működtető szoftver, amely bizonyítja működését!
Jelentkezési határidő: 1989. február 19.
Cím: NJSZT, „Munkahely-Otthon" pályázat, Bp. V. ker., Báthori u. 16. 1054
A verseny időpontja: 1989. március 19., a IV. Nemzetközi Mikroszámítógépes Találkozó keretében. A jelentkezési lapjukon tüntessék fel a program nevét és a két géptípust.

Versenybizottság

Mi villog itt?

A képernyőterületek vagy feliratok kiemelésére igen jó a terület villogtatása. Az eredeti interpreter erre nem képes. Megfelelő attributum a C64-ben nincs, így a színinformációt használtam fel erre a célra.

Az általam készített kiegészítés lehetővé teszi, hogy egy előre meghatározott színű felirat adott frekvenciával villogjon direkt módban, vagy programból be, illetve ki lehessen kapcsolni. Csak a meghatározott színű felirat vagy képernyőterület villog, azt más színnel felülírva a villogás megszűnik.

A működés elve az, hogy a színtárban megkeressük a kijelölt színnek megfelelő bájtokat, és a képtárban az ugyanannak a címnek megfelelő karaktereket inverzükre változtatjuk.

A program a megszakításrutint használja — az IRQ vektor átírása folytán —, de működése a BASIC program futását nem lassítja le.

Az assembler programot az *1. lista* tartalmazza (kazettapuffer területére). A program a memória szabad helyén helyezhető el, be-, illetve kikapcsolása

'SYS CÍM,SZÍN,FREKVENCIA

beírásával lehetséges.

A SZÍN értéke 0-tól 15-ig, a FREKVENCIA értéke 1-től 255-ig terjedhet — kis érték gyors, nagy érték lassú villogást eredményez. A normál villogásnak megfelelő érték 20—25.

A BASIC program a *2. listán* látható.

Kocsis Gábor

2. lista

```
1 REM 2.SZ. LISTA  *** VILLOG BASIC ***
2 REM (C) KOCSIS GABOR
3 REM START/STOP= SYS CIM,SZIN,SEBESSEG
5 :
10 S=0:I=0:INPUT"CIM: ";C
11 READA:IFA=-1THEN13
12 S=S+A:POKEC+I,A:I=I+1:GOTO11
13 IFS=15891THENPRINT"RENDBEN":END
14 PRINT"ADATHIBA !!!"
15 DATA 32,253,174,32,158,183,134,247
16 DATA 32,155,183,134,249,173,20,3
17 DATA 201,49,240,11,169,49,162,234
18 DATA 141,20,3,142,21,3,96,120
19 DATA 169,104,162,3,141,20,3,142
20 DATA 21,3,88,96,165,248,240,4
21 DATA 198,248,16,44,165,249,133,248
22 DATA 169,0,160,216,162,4,133,251
23 DATA 133,253,134,252,132,254,160,0
24 DATA 177,253,41,15,197,247,208,6
25 DATA 177,251,73,128,145,251,200,208
26 DATA 239,230,252,230,254,202,208,230
27 DATA 76,49,234,230,252,230,254,202
28 DATA 208,229,76,49,234,-1
```

```
  1.SZ.LISTA  (C)KOCSIS GABOR
100 SYS9*4096        *** VILLOGAS ***
110 .OPT OO,P        *** A C-64-EL ***
120 *=828
130 IRQ =$314        AZ IRQ VEKTOR CíME
140 SZíN =$D800      A SZíNTÁR KEZDETE
150 KEP =$400        A KÉPERNYŐ KEZDETE
154 SZ =247          A 'SZíN' TÁROLÓHELYE
155 COUNT =248       A 'SZÁMLÁLÓ' HELYE
156 SZAM =249        A 'SEBESSÉG'
160 JSR $AEFD        ',' VIZSGÁLATA
170 JSR $B79E        SZíN ÉRTÉK BEVÉTELE,
180 STX SZ           TÁROLÁSA.
190 JSR $B79B        A SEBESSÉG BEVÉTELE,
200 STX SZAM         TÁROLÁSA.
210 LDA IRQ          AZ IRQ AZ EREDETI ?
220 CMP #$31         HA IGEN, AKKOR
230 BEQ IND          LECSERÉLNI A SAJÁT
240 LDA #$31         RUTIN CíMÉVEL !
250 LDX #$EA         HA A SAJÁT RUTIN,
260 STA IRQ          VISSZAÁLLíTANI AZ
270 STX IRQ+1        AZ EREDETI $EA31-RE.
280 RTS              VISSZA BASIC-BE !
290 IND SEI            IRQ CíM CSERE
300 LDA #<KEZD       EZUTÁN A MEGSZAKíTÁS
310 LDX #>KEZD+1     A 'SAJÁT RUTIN
320 STA IRQ          CíMÉVEL KEZDŐDIK.
330 STX IRQ+1
340 CLI
350 RTS          A SAJÁT RUTIN KEZDETE
360 KEZD LDA COUNT   SZÁMLÁLÓ = 0 ?
370 BEQ FOLY         HA IGEN,AKKOR TOVÁBB
380 DEC COUNT        HA NEM, SZÁMLÁLÓ
390 BPL KESZ         CSÖKKENTÉS ÉS VISSZA
400 FOLY LDA SZAM    A SZÁMLÁLÓ ÚJRATÖL-
410 STA COUNT        TÉSE.
420 LDA #0
430 LDY #$D8         A KÉPERNYŐ ÉS A
440 LDX #4           SZíNTÁR KEZDŐCíMEK
450 STA $FB          BETÖLTÉSE $FB-$FC,
460 STA $FD          ILLETVE   $FD-$FE
470 STX $FC          CíMEKRE AZ INDIREKT
480 STY $FE          INDEXELT CíMZÉSHEZ.
490 LO1 LDY #0         A CIKLUS
500 LA LDA ($FD),Y     A  SZíN-INFORMÁCIÓ
510 AND #$0F             KIKAPUZÁSA
520 CMP SZ             ES HASONLíTÁSA
530 BNE T2           'SZíN'-HEZ. HA MEG-
540 LDA ($FB),Y      EGYEZIK, A HOZZÁTAR-
550 EOR #$80         TOZÓ KARAKTERT
560 STA ($FB),Y      INVERZRE VÁLTOZTATNI
570 T2  INY          ES VISSZAíRNI !
580 BNE LA           EZT 1024-SZER KELL
590 INC $FC          ELVÉGEZNI "
600 INC $FE          A CíMEK NÖVELÉSE
610 DEX
620 BNE LO1          HA MEGVAN, VISSZA AZ
630 KESZ JMP $EA31   IRQ RUTINRA !
```

1. lista

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

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

TOP - lista

BBS

	Felhasználói progamok	IBM	AMIGA	C-128	C-64	C+4 (16)	SPECTR.	ENTERP.	TVC
1	Geos 1.6 d.				•				
2	Windows	•			•		•	•	
3	Giga Cad				•				
4	Printfox	•			•				
5	Rockmon v5.0				•				
6	News room				•				
7	Printmaster	•			•				
8	Music master				•				
9	Letter design				•				
10	Disk manager				•				

	Játék programok	IBM	AMIGA	C-128	C-64	C+4 (16)	SPECTR.	ENTERP.	TVC
1	Skyfox II.				•				
2	Alt. w. g.				•				
3	Sim flight				•				
4	Pentis	•							
5	Elite						•		
6	The guild of t.				•				
7	Nodes of y.				•				
8	Defender etc.		•		•				
9	Renegade						•		
10	Impossible II.				•		•		

Programozási fogások és melléfogások

Ebben a most induló sorozatban elterjedt programozási hibákat mutatok be, ismertetve kiküszöbölésük módját is. A nagyító alá kerülő programok, illetve programrészletek magyar nyelven nyomtatásban megjelent írásokból származnak. A hibák különböző súlyosságúak. Van amelyik teljesen rossz eredményt ad, van amelyik csak a szükségesnél jóval hosszabb futási idejével kelti fel érdeklődésünket, és olyan is akad, amely csupán terjengőssége folytán okoz felesleges gépelési munkát kipróbálójának. A példaprogramok közlésével nem azok szerzőinek „cikizése" a célom, hanem a rossz programozási stílus kiküszöbölése. Ezt azért is fontosnak tartom, mert a hibás programok nagy része éppen oktatási céllal jelent meg.

A most bemutatásra kerülő programokat C64-en futtattam, de más gépeken is használhatók, legfeljebb az eredmény módosul csekély mértékben.

Az Élet és Tudomány 1984/30. számában, az azóta megszűnt *ZSEBISKOLA* 7. fordulójában jelent meg a következő feladat:

„2. feladat (Bukits Róbert — szombathelyi olvasónk javaslata): Írjunk BASIC nyelvű programot, mely előállítja az összes olyan 100-nál kisebb számpárt, melyeket kivonva és összeadva négyzetszámot kapunk."

A feladat megtalálható Donald D. Spencer *Játékok BASIC nyelven* című könyvében is, mely magyarul 1983-ban jelent meg a SZÁMALK kiadásában. A 97. oldalról gépeltem be az *1. listán* látható programot, mely Commodore gépen — a szerző állításával ellentétben — nem állítja elő az összes kívánt számpárt, hanem a 31-ből csupán a *2.* (futási) *listán* látható kilencet.

A hiba a 150-es és 160-as sorban van. A szerző figyelmen kívül hagyta azt a fontos programozási szabályt, hogy *valós számok összehasonlításánál nem szabad egyenlőségre vizsgálni.* A *3. listán* láthatjuk az összehasonlítás helyes módját: két valós számot akkor tekintünk egyenlőnek, ha különbségük abszolút értéke kisebb egy előre meghatározott kis pozitív számnál. Esetünkben viszonylag nagy számot választottunk, de a feladat jellegéből adódóan ennek a százszorosa is megfelelő lenne.

Az SQR függvény mindig pozitív értéket ad, így könnyen beláthatjuk, hogy a vizsgált sorokban az ABS függvény argumentuma mindig pozitív lesz. Ebben az esetben a *4. listán* látható módon az ABS függvény is elhagyható. Valóban, az eredeti program két sorát a 4. lista szerint módosítva, a program mind a 31 számpárt előállítja. Csak az a baj, hogy *nagyon lassan.* A fejléc kiírása után kezdve a mérést 40671 TI-egységet kaptam futási időként, ami több mint 11 perc. (Egy TI-egység 1/60 másodpercnek felel meg.) Az 1E—5 konstans változóval való helyettesítésével és a szóközök elhagyásával sikerült a futási időből valamennyit lefaragni, de az így is 10 perc fölött maradt, és ezen már a program tömörítésével (a HELP ⌘C parancsával) sem sikerült segíteni.

1. lista

```
100 rem negyzetszamok
110 print " n          p          n+p          p-n"
120 print
130 for n=1 to 100
140 for p=n+1 to 100
150 if sqr(n+p) <> int(sqr(n+p)) then 180
160 if sqr(p-n) <> int(sqr(p-n)) then 180
170 print n,p,n+p,p-n
180 next p
190 next n
200 end
```

2. lista

n	p	n+p	p-n
6	10	16	4
16	65	81	49
24	25	49	1
24	40	64	16
30	34	64	4
36	85	121	49
40	41	81	1
60	61	121	1
96	100	196	4

3. lista

```
    ...
150 if abs(sqr(n+p)-int(sqr(n+p)))>1e-5 then 180
160 if abs(sqr(p-n)-int(sqr(p-n)))>1e-5 then 180
    ...
```

4. lista

```
    ...
150 if sqr(n+p)-int(sqr(n+p))>1e-5 then 180
160 if sqr(p-n)-int(sqr(p-n))>1e-5 then 180
    ...
```

5. lista

```
100 for j=1 to 12
110 j2=j*j
120 for i=j+2 to 14 step 2
130 i2=i*i
140 a=(i2+j2)/2
150 if a>100 then 190
160 b=(i2-j2)/2
170 print b,a,i2,j2
180 next i
190 next j
```

6. lista

```
10 print " a     b     a+b    a-b"
20 for a=1 to 100
30 for b=a+1 to 100
40 if sqr(a+b)<int(sqr(a+b)) then 70
50 if sqr(a-b)>int(sqr(a-b)) then 70
60 print a,b,a+b,a-b
70 next b
80 next a
90 end
```

A könyvben látottnál jobb algoritmust kerestem. Megfordítottam a problémát: négyzetszámpárokhoz kerestem megfelelő számpárokat. Ehhez felhasználtam a következő egyszerű algebrai egyenlőségeket:

((A+B)+(A−B))/2=A
((A+B)−(A−B))/2=B

A zárójelekben lévő A+B, illetve A−B kifejezéseket négyzetszámokkal helyettesítve jutunk el a megoldáshoz, ahol A és B a keresett számok. Belátható, hogy a négyzetszámoknak azonos párosságúaknak kell lenniük, különben A és B nem lennének egészek. Az eredeti programhoz hasonlóan kizárhatjuk azt az esetet is, amikor a négyzetszám önmagának és nullának az összegeként és különbségeként jön létre.

Az új program az *5. listán* található. Eredménye — a listázott számnégyesek sorrendjétől eltekintve — megegyezik a könyvből begépeltével, de annál 248-szor (!) gyorsabban fut. A mért futási idő 164 TI-egység, azaz 3 másodpercnél is kevesebb. Még feltűnőbb a gyorsulás, hogy ennek az időnek több mint felét a PRINT utasítás végrehajtása veszi igénybe.

Még egy részlet magyarázatra szorul. A négyzetszámok előállítása nem hatványozással, hanem az alapszámok önmagával való szorzásával történik. Ez gyakori programgyorsítási fogás, de itt nem ez az elsődleges cél, hanem a hatványozás pontatlanságából eredő hibák elkerülése. Ez különösen Commodore gépeken kellemetlen. Aki még nem tapasztalta, próbálja meg kiszámítani, menynyi hét a négyzeten.

Végül a *6. listán* bemutatom a ZSEBISKOLA 9. részében (az Élet és Tudomány 1984/33. számában) közölt — dr. Appel György rovatvezető szerinti helyes — megoldást. Pontatlanul idézem, ugyanis az eredeti programlista írógéppel készült, és a 1Ø-es sorban a PRINT utasítás után *alsó idézőjel* áll, és azt számítógépes programlistán nehezen tudnám megvalósítani.

Mivel B értéke mindig nagyobb A-énál, ezért az 5Ø-es sorban az SQR függvény argumentuma negatív lesz. Ezt — tudomásom szerint — egyetlen BASIC-változat sem engedi meg. Magyarázatot, helyreigazítást azóta sem találtam. Lehet, hogy nem az Élet és Tudományban, hanem valamely más kiadványban jelent meg, ezért kerülte el a figyelmemet. Ha valaki tud ilyesmiről, kérem, hogy tájékoztasson!

Barna László

A PROGRAMOZ

ALAPKÉRDÉSEI V.

Egy program

Ötrészes sorozatunkban mindarról szót ejtünk, amit olvasóinknak érdemes átgondolniuk, ha programozásra adják a fejüket, vagy ha valamely programról véleményt kívánnak alkotni. Ezek az írások a szerzőnek lapunkban közölt korábbi fejtegetéseivel együtt (utalunk „Az adattípus fejlődése" és a „Függvények és utasítások" című sorozatokra, lásd 1987/8. és 9., valamint 1987/12., 1988/1. és 2. számainkat), teljes körűen és közérthetően elemzik e sorozat címében jelzett (problematikát) témát.

Amióta a szoftver árucikké vált, jó néhány tanulmány jelent meg a programok életciklusáról (life cycle-jéről). A teória szerint a program élete a következő szakaszokra osztható: a feladat meghatározása (specifikáció), a rendszerterv elkészítése, a program megírása és ellenőrzése (implementáció), a program munkába állítása, folyamatos karbantartása. Majd ezután néhányszor szerét ejthetik a program kisebb átalakításának és továbbfejlesztésének, ami után ismét a munka és karbantartás következik, végül a programot kiselejtezik.

Ez a modell nagyjából fedi a piacra készülő programok tényleges életét. Ami nem véletlen, hiszen egy árunál nagyon fontos, hogy mennyi idő alatt állítják elő, milyen a szerviz, és mennyi ideig lehet majd használni. A programok jelentős részének azonban nem ilyen az élete: általában nem így születnek és élnek a kutatási feladatokat megoldó programok, s megint más a hobbiprogramoknak és különösen azoknak a segédprogramoknak a sorsa, amelyeket saját munkájuk megkönnyítésére írnak a programozók.

Míg az első csoportba tartozó programok egész életét tervszerűség hatja át, addig a másik csoportot inkább valami spontán evolúció, ösztönös fejlődés jellemzi. — Egyszer valakinek valamire volt egy jó ötlete, írt belőle egy kis programocskát, amit ügyesen tudott használni. Valaki más meglátta, hozzátett néhány ötletet, vagyis ő létrehozta a továbbfejlesztett változatot stb., mindaddig, míg végül a kezdeti ötletből kikerekedett egy komoly program.

AZ „ONTOGENEZIS" KÖVETKEZMÉNYEI

Mindkét programfajtának vannak előnyös és hátrányos tulajdonságai. A „megtervezett programok általában könnyen leírhatóan viselkednek, viselkedésük többé-kevésbé dokumentálva is van, és jól védekeznek a felhasználók hibái ellen is. Ugyanakkor gyakran túl nagyok, és esetenként meglehetősen lassúak is, mert olyan részeket is tartalmaznak, amelyekről eredetileg úgy gondolták, hogy szükség lesz rájuk, de a gyakorlatban feleslegesnek bizonyultak. Az „evolúcióval létrejött" programok rendszerint ügyesek és hatékonyak, de gyakran számos különböző változatban élnek, ami alaposan megzavarhatja a felhasználókat. Emiatt persze a felhasználó gyakran befejezetlennek és néha következetlennek találja őket.

A tiszta típusok között természetesen sok átmenet létezik. A BASIC nyelv eredetileg egy megtervezett, de végtele-

ÁS

élete

nül egyszerű nyelv volt, amelyet azután — mivel volt benne egy-két jó ötlet — elkezdtek tupírozni, míg végül a Dartmouth BASIC egy rendkívül nagy és kissé illogikus programmá fejlődött. Ez a BASIC még nagyszámítógépen futott. A személyi számítógépek megjelenésével az eredeti nyelv ötletei ismét aktuálissá váltak, de minden gyártó saját ízlésének megfelelően alakította át és fejlesztette tovább a nyelvet. A káosz után ma ismét lassan rendeződik a helyzet, úgy tűnik, hogy a Microsoft BASIC lassan hallgatólagos szabványnyá válik. Az utóbbi időben fontos minősítője egy BASIC programnak, hogy Microsoft kompatibilis-e.

A C nyelv evolúcióval fejlődött ki a B nyelvből, de elkészülte után szerzői (B. W. Kernighan és D. M. Richie) egy olyan pontos és érthető leírását adták, ami azóta szabványnak számít, és senki sem kívánt eltérni tőle. Annál is inkább, mert a Unix operációs rendszer kapcsán több nagyon ügyes C nyelvű program készült, melyek bármelyikét minimális erőfeszítéssel lehet átvinni bármelyik másik gépre, ha ott is van C.

A Pascal nyelv kibontakozása és térhódítása éppen ellentétes az iméntivel. Ezt szerzője (N. Wirth) először papíron, filozófiailag tervezte meg. A filozófia nagyon sikeres volt, a Pascal nyelv minden utána született nyelv egyik kiindulópontjává vált. Nem egészen ez a helyzet magával a nyelvvel, amely a gyakorlatban gyöngének bizonyult, és ezért továbbfejlesztésekre ingerelte alkalmazóit. Mivel nem volt, aki a fejlesztéseket koordinálja — Wirth akkor már egy másik nyelvvel, a Modulával foglalkozott —, a különböző Pascalok különböző, egymással inkompatibilis változatokban alakultak tovább.

A fentiekből is látható, hogy egy program minőségét és népszerűségét nem az határozza meg, hogy előre kidolgozott tervek vagy önfejlődés útján keletkezett — hacsak olyan értelemben nem, hogy egy igazán sikeres programnak mindkét fázison keresztül kell mennie.

A „MEGTERVEZETT" PROGRAMOK

Az irodalomban sokszor vitatott kérdés: lehet-e egyből jó programot írni. Vannak, akik nemcsak hisznek ebben, hanem megpróbáltak receptet is kidolgozni rá. Ezek a receptek filozofikus állásfoglalások, gyakorlati tapasztalatok, matematikai tételek és kemény munkaszervezési előírások keverékéből állnak. A legmélyén minden ilyen módszertan két általános érvényű bölcsességre épül:

— Előbb gondold végig, mit akarsz csinálni, és csak azután fogj hozzá.

— Több szem többet lát.

Az én személyes véleményem: igenis lehet egy csapásra hibátlan programot írni, feltéve, hogy rutinfeladatról van szó, továbbá lehetőségünk van a korábban kidolgozott és bevált programokból részek átvételére, és ugyanakkor betartjuk az előbb említett két elvet.

A feladat meghatározása és a környezeti feltételek megismerése az egész munka szempontjából alapvető fontosságú. Amikor egy új feladatot kap az ember, gyakran tapasztalja, hogy valójában a megrendelő sem tudja értelmesen elmondani, mit is szeretne. A rutinos programozó is csak a harmadik, negyedik hasonló feladat után tudja már, hogy mit kell megkérdeznie. Éppen ezért nagyon fontos, hogy a megbeszélések végén szűkszavúan és nagyon földhözragadt módon, de írásban összeállítsák azt, hogy mit is kell csinálni. Nem helyes, ha ennek a fázisnak az eredményét kilókban akarjuk leméri. — Sok beszédnek sok az alja, továbbá: fától nem látni az erdőt — újabb két pótolhatatlan bölcsesség, utolérhetetlen tömörségű „filozófia".

A program elkészítésének második fázisa a rendszerterv megírása. Rendszerterven mindenki mást ért. Az egyik rendszerterv szabad elmélkedés arról, hogy miért úgy csináljuk a dolgokat, ahogy csináljuk. A másik egy durva blokkvázlat, hogy milyen nagyobb lépésekből fog állni a program. A harmadik a program működésének részletes leírása pszeudokódban. Ez így is van rendjén. Különböző feladatok és különböző munkacsoportok más és más rendszertervet igényelnek. Összeszokott csapat, ha rutinmunkája van, megelégedhet egy egészen nagyvonalú vázlattal. Viszont, ha ugyanezt a munkát újoncokra bízzuk, a tervet nemcsak az apró részletekig kell lebontani, hanem többször közösen át kell rágnunk rajta magunkat, miközben az a kolléga premizálandó legjobban, aki a tervben a legtöbb hibát találja meg.

Egy dolog van, ami minden rendszertervnek elkerülhetetlen tartozéka: tisztázni kell, hogy a program milyen részekből fog állni, ezeknek mi lesz a feladatuk, és hogyan kapcsolódnak egymáshoz. A kapcsolódási felületet (az interfészt) a lehető legnagyobb pontossággal előre ki kell kötni, mert csak így képzelhető el, hogy a munkát hatékonyan fel tudjuk osztani, és hogy az elkészülő program áttekinthető és javítható részekből áll majd.

Ezután következik a program kódolása. Ha valaki tudja, hogy milyen feladatot, milyen úton kell megoldania, és a kódolás egy-két évi programozói munkaviszony után még gondot okoz neki, az jobb, ha azonnal új munkakör után néz. Ez természetesen nem jelenti azt, hogy egy program kódolása egyértelmű, könnyű és gyors munkával jár. Bizony több lehetséges megoldást kell végiggondolni, és néha ki is kell próbálni, amíg a legjobbat megtaláljuk, de hát végül is ez a szakmánk.

A kódolás után a program tesztelése következik. A rátermett programozó a tesztelésre nem akkor gondol először, amikor a program elkészült, hanem a munka legelejétől kezdve állandóan. Jobb, ha már a specifikáció során tisztázzuk: tud-e a megrendelő olyan adatokat adni, még a program elkészülte előtt, amelyekkel a programot ki lehet próbálni. Ha ez nem lehetséges — ami sajnos gyakran előfordul —, legjobb, ha a munkacsoport egyik emberét csak a tesztadatok előkészítésének szenteljük. Tapasztalni fogjuk, hogy ez nem kisebb munka, mint a programírás. Természetesen a kívülálló tesztje nem elegendő a program ellenőrzésére, csak a programíró ismeri annyira saját programját, hogy annak minden ágát ellenőrizni tudja.

Összegezve azt mondhatjuk, hogy amíg a program a kész állapotba eljut, át kell esnie a specifikációkészítés, a rendszertervezés, a kódolás és a tesztelés fázisain. A tapasztalat azt mutatja, hogy a jó programozók az idő felét töltik el a tervezéssel és az idő negyedét a teszteléssel, javítgatással, majd csak a maradék egynegyedét a programírással. Ez a főnökökre súlyos felelősséget hárít, mert tulajdonképpen a programírás az egyetlen olyan tevékenység, ami egyszerűen — a megírt sorok számával — mérhető, a többi feladatrész számonkérése komoly szellemi munkát igényel.

AZ „ÖNFEJLŐDŐ" PROGRAMOK

Az önfejlődő programoknak természetesen nincs specifikációjuk és rendszertervük, csak nőnek, mint a bolondgombák. Ezek között is vannak azonban tisztességesen megírt programok, mint ahogy akadnak összebuheráltak a másik kategóriában is.

Amíg a program egyszerű, jól strukurált és áttekinthető, sokan hajlanak rá, hogy fejlesztés ürügyén kicsit beleszabdaljanak, beletoldozgassanak. Ezek során azonban a program lassan áttekinthetetlenné válik, és eljön a pillanat, amikor a következő ötlet „belevarázsolása" nehezebb munka lesz, mint újraírni a programot.

Ha a program eredetileg is tisztességesen készült és minden egyes továbbfejlesztésnél vigyáztunk arra, hogy olyan is maradjon — ami némi többletmunkát kíván a toldozgatáshoz képest —, akkor a programot nagyon hosszú ideig lehet folyamatosan fejlesztgetni

anélkül, hogy a teljes újraírásra szükség lenne.

Egy egyszerű, de gondosan kódolt program eleve úgy néz ki, hogy egy keretből és egy magból áll. A keret foglalkozik a szükséges segédadatok előkészítésével, a bemeneti-kimeneti fájlok megnyitásával, majd a munka végén a fájlok bezárásával stb. és meghívja a magot (gyakran ciklusban), ami az érdemi munkát végzi.

A mag a legegyszerűbb esetben egy függvény, ahogy azonban a feladat növekszik, bonyolódik, idővel több függvényre bomlik szét. Amikor a program elérte azt a tömeget, hogy egyben már nem lehet vagy nem érdemes lefordítani, a függvényeket funkcióik szerint külön fordítható modulokba osztjuk be. A modulok létrehozásának egyik fontos vezérlőelve, hogy ez a függvény- és adatgyűjtemény lehetőleg más programokban is felhasználható legyen. Idővel bizonyos modulok is elérik a kritikus tömeget, és több modulra hasadnak szét.

VAN-E ÉLET A HALÁL UTÁN?

Bármilyen ötletes, hatékony és gondosan kivitelezett a programunk, az idő nem múlik el felette nyomtalanul. Ami fontos volt, lényegtelenné válik, ami jelentéktelennek tűnt, egyszerre nagy hangsúlyt kap. Mint minden műszaki cikknél, a szoftvernél is van erkölcsi elavulás. Az esetek nagyobbik részében egy hatékonyabb, kényelmesebb program jelenik meg a színen, és lassan elcsábítja a régitől a felhasználókat. Van olyan is, hogy egy új programozó csillag tündöklik föl, akinek nem tetszik a mi kőbaltás módszerünk; máskor viszont új gépet dobnak piacra, ami használhatatlanná teszi a régi szerszámokat.

Az előbbivel nincs gond: amikor a túlnyomó többség már az új eszközt használja önként vagy parancsszóra, mi is kénytelenek leszünk cserbenhagyni a régi programot, és a program meghal.

Nem ilyen egyértelmű a helyzet, amikor a program alkalmatlanná válik az új helyzetben — például nem tud bizonyos új eszközöket, gépet, perifériát, fájlrendszert, adatbázist stb. kezelni —, de nincs olyan eszköz, amelyik ellátná a régi és az új feladatot együtt, vagy nagyon rossz hatásfokkal teszi. Ilyenkor nincs más hátra, mint a program újraírása, ami megtörténhet nagyon jól strukturált és tudatosan tervezett, bevált programokkal is.

A CIKLUS — ÉS E SOROZAT — VÉGE

A program újraírását a program céljainak újragondolásával kell kezdeni. Egy-egy ilyen komoly átgondolás néha azt eredményezi, hogy az utódprogram lényegesen eltér a régitől. Felmerül a kérdés, hogy ez a program még „az" a program-e, vagy már valami egészen más. Ilyenkor új nevet vagy legalábbis friss verziószámot adunk a programnak. Ha az új program mindazt tudja, amit a régi, akkor általában csak újabb verzióról beszélünk. Ha viszont bizonyos dolgokat másként old meg, célszerű másik nevet adni neki. Ezután a programot elölről kell tervezni és programozni. Így a program „megtervezett" programmá válik, és ha jól sikerült, akkor később egy újabb evolúciós folyamat kiindulópontja lehet.

Farkas Ernő

Az operációs

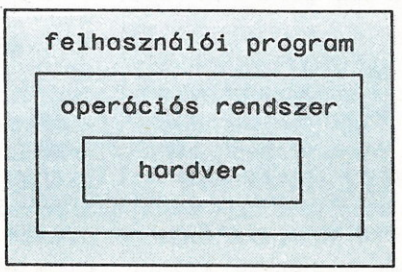
Ezt a cikket az 1989 februárjában induló operációs rendszerekkel foglalkozó sorozatunk előfutárjaként, figyelemfelkeltőnek szánjuk.

Ha bekapcsoljuk mondjuk a Commodore 64-es számítógépünket, egy BASIC-ül „értő" környezettel találjuk szemben magunkat. Ez természetesen nem hardversajátosság, hanem annak következménye, hogy a gépen BASIC interpreter — egy tulajdonképpeni operációs rendszer — fut.

Már az első számítógépek megjelenésekor világossá vált, hogy bizonyos alaptevékenységek — például a programbetöltés, a bemenet-kimenet kezelése stb. — minden programban ismétlődnek. Célszerű volt ezeket előre megírni és programkönyvtárakként a programozók rendelkezésére bocsátani. Így születtek meg az első operációs rendszerek és így vált ketté a programozói munka rendszerprogramozói és felhasználói programozói részre. Később a többfeladatos időosztásos (ún. multiprogramozott) környezetekben szükségessé vált a számítógép erőforrásainak — a nyomtatónak, mágnesszalagegységnek, a mágneslemeznek, a központi memóriának — az egymással párhuzamosan futó programok közötti hatékony és „ütközésmentes" elosztása. Ezt a feladatot is az operációs rendszer látja el. A párbeszédes üzemmódú rendszerek megjelenésekor a felhasználóval való kapcsolattartás, a parancsok értelmezése és végrehajtása is az operációs rendszer feladata lett. Ismeretes, hogy az első időkben a programokat, adatokat általában lyukkártyára lyukasztották és egymás után — kötegelve — dolgoztatták fel a géppel. A képernyős-billentyűzetes perifériák csak később jelentek meg.

AZ OPERÁCIÓS RENDSZER FELADATAI

Az operációs rendszer tehát valahol a hardver és a felhasználói program között áll *(1. ábra)*. A felhasználói programok a számítógép erőforrásaihoz csak az operációs rendszeren keresztül férhetnek hozzá. Az operációs rendszer „elrejtheti" a számítógép-



1. ábra

rendszerek adatkezelése

hardverek különbségeit a felhasználók és a felhasználói programok elől. Ezáltal lehetővé válik, hogy ugyanaz a program különböző típusú gépeken azonos módon fusson, ha a gépek ugyanazt az operációs rendszert alkalmazzák. Az operációs rendszer a gép „gyengeségeit" is eltakarhatja; megnövelheti például a központi memória látszólagos méretét — természetesen a működési sebesség rovására — a mágneslemez-területből, vagy a sebességet azáltal, hogy a mágneslemez-terület egy részét állandóan a központi memóriában tárolja. Ezen alapul az a mondás, hogy „az elefánt nem más, mint operációs rendszerrel ellátott egér".

Az operációs rendszer rendszerint néhány hasznos segédprogramot is tartalmaz: szövegszerkesztőket, fordító-szerkesztőprogramokat, rendező-válogató-összehasonlító programot, esetleg adatbázis-kezelőt; ezenkívül általában bő választékát kínálja olyan jellegű szubrutinoknak, mint többek között a dátum és a pontos idő stb., melyekre a programozónak szüksége van.

BEMENETI-KIMENETI MŰVELETEK VEZÉRLÉSE

A továbbiakban az operációs rendszerek bemenet-kimenet kezelő feladatát tekintjük át. Hogy ez történetesen egy mágneslemeznél milyen sokrétű feladat, azt az alábbiakban láthatjuk.

Maga az operációs rendszer is egy „többrétegű" program. Ez azt jelenti, hogy feladatát hierarchikusan szervezett részfeladatokra bontva látja el. Minden programszint a saját részfeladatáért felelős, és az alatta, illetve felette lévő szintekkel tart kapcsolatot; utasításokat és adatokat küld, illetve fogad. Minden operációs rendszernek más-más hierarchikus felépítése van. A *2. ábra* egy képzeletbeli „minta operációs rendszer" felépítését szemlélteti. A legalsó szint közvetlenül a hardvert kezeli, a legfelső a felhasználói programmal — vagy közvetlenül a felhasználóval — tart kapcsolatot.

A mágneslemez-meghajtó berendezés a következő parancsokat fogadhatja *(3. ábra)*:

— az író/olvasófej pozicionálása egy adott sáv adott szektorára,
— az adott szektor írása,
— az adott szektor olvasása.

Az operációs rendszer legalsó szintje ezeket a parancsokat továbbítja a hardvernek.

A felhasználó általában nem szektoronként, hanem logikailag összefüggő adatállományonként „látja" a lemezt. Ilyenkor egy előre kijelölt lemezterületen — a katalógusterületen — nyilván kell tartani, hogy melyik adatállomány a lemez mely szektorait foglalja el. A rendszer egy másik szintje a katalógust kezeli, és az adatállomány-hivatkozásokat „lefordítja" az alsó szint számára fizikai címhivatkozásokká. Az adatállományok törlésekor a felszabaduló területek nyilvántartása, újrakiosztása, a „feldarabolás" megakadályozása is ennek a rétegnek a feladata. A felső szinteknek tehát már nem kell az adatállományok fizikai helyével törődniük.

Ez a programszint az alábbi utasításokat fogadja: egy adatállomány létrehozása (ha még nem létezik), megnyitása (ha már létezik), lezárása, írása, olvasása, törlése.

Az írás/olvasás módja is többféle lehet. A legegyszerűbb esetben az adatállomány adatait sorban, egymás után akarjuk elérni. Bonyolultabb alkalmazások, adatbáziskezelő programok „rafináltabb" elérési módokat is igényelnek. Ilyenek az ún. indexszekvenciális, direkt stb. elérési módok. Ezeknek a kezelése az operációs rendszer megfelelő rétegének a feladata lehet.

Egy további szint foglalkozhat a többszörös hozzáférés vezérlésével, ha a gépet egyszerre több felhasználó használja — ahogyan a mikrogép kategória „fölötti" gépeknél általában ez a helyzet —, ugyanarra az adatállományra egyszerre több felhasználó is hivatkozhat. Ilyenkor az illetéktelen hozzáférés elleni védelem, illetve az adatütközés veszélyének — például ugyanazt az adatállományt egyszerre többen akarják írni — elkerülése érdekében a katagólusba azt is be kell jegyezni, hogy mely adatállományt mely felhasználók milyen módban (olvasás, írás, mindkettő vagy egyik sem) érhetik el. Azt is nyilván kell tartani, hogy ki az adatállomány tulajdonosa, kinek engedélyezett a törlés, ki változtathatja meg a többiek elérési módját stb. Általában a többszörös hozzáférés olyan ütemezési problémákat vethet fel, amelyeket az operációs rendszer tervezésekor külön figyelembe kell venni.

LOGIKAI PERIFÉRIÁK

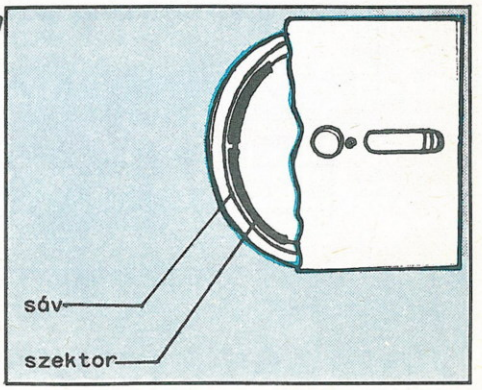
Léteznek olyan programok, amelyeknek logikailag lényegtelen, hogy bemenetük és kimenetük mögött valójában milyen fizikai eszköz van. Egy rendezőprogram például ugyanúgy működik, ha bemenetét mágnesszalagról kapja és kimenetét a nyomtatóra kell küldenie; olyan, mintha bemenete egy mágneslemez-adatállo-

felhasználói programok
logikai perifériák
adatállomány-elérési módok (pl indexszekvenciális)
mágneslemez tartalomjegyzék kezelése
erőforrások kiosztása (többszörös hozzáférés)
fizikai eszközkezelés
H A R D V E R

operációs rendszer

2. ábra

3. ábra



mány, kimenete egy másik mágneslemez-adatállomány lenne. A rendezőprogram írója ekkor ún. „logikai perifériákra" hivatkozik. Programjának környezetét a program futtatásakor jelölik ki, esetleg futtatásonként másmás környezetet. A fejlettebb operációs rendszerek lehetőséget nyújtanak erre, és a futtatáskor kijelölt környezet szerinti perifériavezérlő programokat aktivizálják. Mindezt a *4. ábra* szemlélteti.

A „kötegelt feldolgozást" végző OS operációs rendszerben egy RENDEZ nevű rendezőprogram számára a környezet kijelölése például az alábbi, ún. „Job control" utasításokkal lehetséges:

```
.
.
//   EXEC PGM=RENDEZ
//BE DD UNIT=TAPE,VOL=SER=
       SZALAG100,DISP=(OLD, KEEP),
//   DCB=(LRECL=80,RECFM=
       FB),LABEL=(3,NL)
.
.
```

A RENDEZ program írója csak annyit tud, hogy a „BE" nevű logikai perifériáról kapja a bemenetet (amit rendezni kell) és a „KI" logikai perifériára kell az eredményt kiírnia. A fenti Job control utasításokkal azt definiáltuk, hogy a „BE" a „SZALAG100" nevű mágnesszalag, a „KI" pedig a sornyomtató legyen. Futás közben a rendezőprogramnak az OS rendszer adja a bemeneti adatokat, a kimenetet szintén ő továbbítja a sornyomtatóhoz. Megjegyzendő, hogy az OS operációs rendszerhez tartozó SORT rendezőprogram környezetkijelölése ennél bonyolultabb. A logikai perifériák kezelését, a konkrét fizikai perifériák hozzárendelését az operációs rendszer egy magasabb rétege végzi.

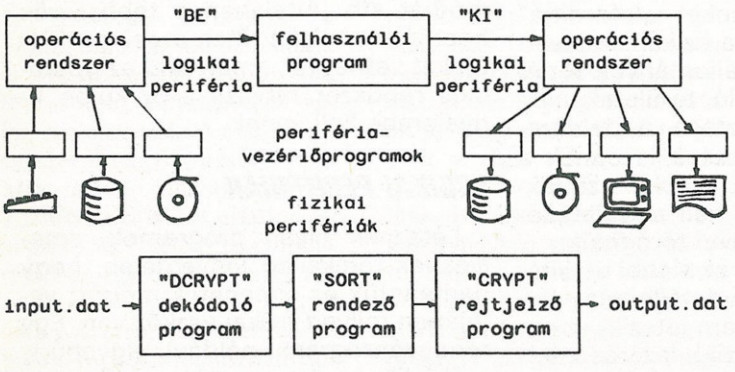
Az MS—DOS és a Unix operációs rendszerben a programok számára a „standard bemenet és kimenet" érhető el. Ezek alapértelmezésben a billentyűzetet, illetve a képernyőt jelentik. A standard bemenet és kimenet átdefiniálható, például a következőképpen: a

sort < input.dat > PRT

parancs az „input.dat" nevű adatállományt rendezi, és az eredményt a nyomtatóra küldi ki. Sőt, az egyes programok „sorba kapcsolhatók" úgy, hogy az egyik kimenete a másik bemenete. Tekintsük az alábbi MS—DOS parancsot:

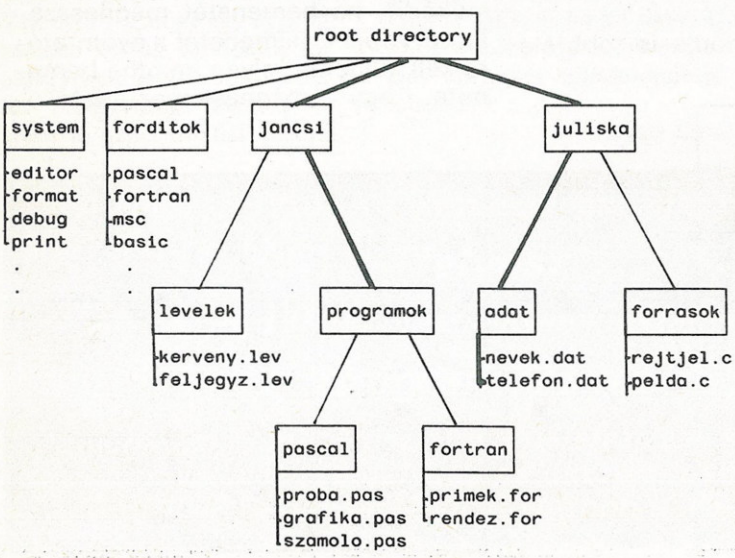
dcrypt < input .dat sort crypt > output.dat

Ez a rejtjelezett „input.dat" adatállományt dekódolja, rendezi, és az eredményt — rejtjelezés után — az „output. dat" adatállományba írja ki *(5. ábra)*. Eközben mindhárom program (a kódoló, a rendező és a dekódoló) írója programjában csupán a standard be- és kimenetre hivatkozott, mit sem tudva a környezetről, amelyben programja futni fog!



4. ábra

5. ábra



6. ábra

CÍMJEGYZÉKEK

Egy mágneslemez általában nagyon sok adatállományt tartalmaz. Ahhoz, hogy valamiképpen rendet tarthassunk, saját adatállományainkat másokétól megkülönböztethessük, szükség van valamiféle csoportosítási lehetőségre. Az MS—DOS és a Unix operációs rendszerek lehetővé teszik, hogy adatállományainkat címjegyzékekbe (directoryba) csoportosítsuk.

A címjegyzék egy fastruktúrához hasonlít, ahol a fa gyökere az ún. „root directory", levelei pedig az alcímjegyzékek. Egy címjegyzék tartalmazhat adatállományokat és/vagy al-al-al címjegyzékeket. Így ha egy gépet — pontosabban egy mágneslemezt — többen használnak, saját adatállományait ki-ki különböző címjegyzékekben tarthatja, esetleg téma szerint további alcímjegyzékekbe csoportosítva.

Egy tipikus címjegyzékstruktúrát mutat be a *6. ábra.* Itt a „SYSTEM" címjegyzék a mindenki által használt segédprogramokat, a „FORDÍTOK" pedig néhány nyelv fordítóprogramját tartalmazza. A gépet ketten használják: „JANCSI" leveleket és Pascal, illetve FORTRAN nyelvű programokat, „JULISKA" pedig adatbázisokat és C forrásprogramokat tárol, téma szerint külön-külön alcímjegyzékben.

Az operációs rendszer számára aktuális címjegyzéket jelölhetünk ki. Ekkor csak azok az adatállományok „látszanak" közvetlenül, amelyek az aktuális címjegyzékben vannak. Más címjegyzékbeli állomány eléréséhez meg kell adnunk az elérési utat is. Például, ha „JULISKA" adatbázisaiból a „TELEFON.DBF" állományra akarunk hivatkozni, azt az alábbi módon tehetjük meg:

\JULISKA\ADAT\TELEFON.DBF

Itt a „\" karakter az elválasztójel szerepét tölti be. A kezdeti „\" azt mutatja, hogy az út a fa gyökeréből indul. Ha ezt elhagynánk, az aktuális címjegyzékből indulna a keresési út. Az út elemei között szereplő „.\" a közvetlenül „fölöttünk" lévő, ún. szülő címjegyzéket jelenti. A fenti adatállomány a \JANCSI\PROGRAMOK címjegyzékből így is elérhető:

..\..\JULISKA\ADAT\TELEFON.DBF

Ezt az utat a 6. ábrán vastag vonallal jelöltük. Az első megadási forma általánosabb, hiszen független attól, hogy éppen mi is az aktuális címjegyzék.

Szentjóbi Ottó

Az NJSZT a Tavaszi Fesztivál ideje alatt, az Utazás '89 kiállítással egy időben, az IPV-vel együttműködve megrendezi a

μ' 89

IV. ORSZÁGOS MIKROSZÁMÍTÓGÉPES TALÁLKOZÓT 1989. március 17—22. között.

A találkozót — immár hagyományként —

„A SZÁMÍTÁSTECHNIKA MINDENKIÉ, A SZÁMÍTÁSTECHNIKA MINDENKIÉRT" informatikai kiállítással egybekötve, a BNV területén fogjuk megrendezni.

A kiállítás először kap tematikus vezérfonalat: szeretnénk az idén az idegenforgalmi alkalmazásokat hangsúlyozni. A látogatóknak ismét változatos programokat ajánlunk, pályázatot írunk ki (ld. 13. o.) oktatóprogramok, házi építésű számítástechnikai eszközök, munkahely-otthon, számítógépes zenei verseny témában. Hobby Computer Club-bemutatók, ember—gép sakkviadalok, bridzs, szakkönyvvásár színesítik a programot.

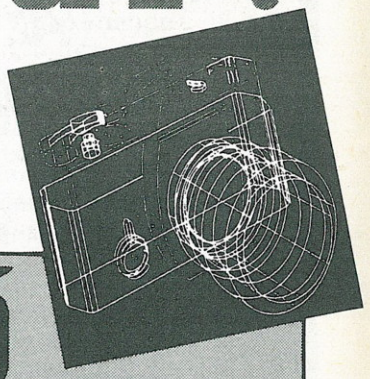
Kérjük olvasóink javaslatait és közreműködését, hogy a μ' 89 találkozó ismét sikeres és eredményes rendezvény legyen a számítástechnika népszerűsítése és a társadalom informatizálása program keretén belül.

Levelezési cím: 1054 Budapest, Báthori u. 16.

Javaslataikat előre is köszöni:

a szervezőbizottság

Szoftveripar: JELEN ÉS JÖVŐ



Éppen egy éve, az Economist 1988. januári számában jelent meg egy tanulmány „A számítástechnika jövője" címmel. Ez a cikk sok olyan kérdést érint, amely a világ vezető országaiban akkortájt részben napi aktualitású, részben perspektivikus jellegű volt.

Úgy érezzük, nálunk, Magyarországon ennyi idő múltán sem vesztettek jelentőségükből az akkor megfogalmazottak; „követő stratégiánkból" fakadóan néhány dolog pedig még mindig csak a távoli jövőben válhat a hazai számítástechnika igazi piaci tényezőjévé.

Ezért is érezzük szükségesnek, hogy alapvetően ennek az Economistban olvasott cikknek a tényeire támaszkodva indítsunk egy olyan sorozatot, amely a szoftveripar főbb vonásait, trendjeit, megújulásának eredményeit összegzi — a hazai piac sajátosságainak megfelelően elsősorban a PC-kategóriában.

1987-ben a világ hardveriparának összes eladási értéke mintegy három és félszer akkora volt, mint a szoftveriparé. De míg a hardverek értékesítésének csaknem a fele egy cég — az IBM — nevéhez fűződik, a világ legnagyobb szoftvercégének, a Computer Associatesnek összes forgalma is mindössze két százalékban részesedik a világ szoftverpiacából. S bár a szoftveripar évi növekedési üteme csaknem húszszázalékos, az IBM-éhez hasonló domináns szerepre vélhetően még sokáig nem törhet senki.

SZABVÁNYOK ÉS OSZTÁLYOK

Három „szabványos" program uralkodik ma a mikrogépek világában: az IBM és vele kompatibilis PC-k operációs rendszere — az MS—DOS —, a Lotus 1—2—3 nevű táblázatszámító-program, valamint a dBASE adatbázis-kezelő. Ide sorolható még az információnak az adatbázisból való helyreállítására szolgáló SQL nyelv és néhány különböző típusú „szabványhálózat" is, bár ezek jelentősége nem mérhető a „három nagyhoz".

Mindennek a „koronája", a szabványok szabványa azonban még hiányzik, főként mivel a szoftveripar „szabványainak" mindegyike csak saját szűkebb környezetében domináns, nem illeszkedik a többihez. A hasonló szigeteket egymáshoz kapcsolni általános, közös szabványokkal lehetne, ennek a folyamatnak azonban gátat szabnak az eltérő piaci érdekek. Abban egyetértenek a szoftveróriások, hogy a munkafolyamatok automatizálásához elengedhetetlen az integrálás, csakhogy az integráció középpontjában mindegyik a saját rendszerét szeretné látni.

Elméletileg az Ashton-Tate-nek kellene a legjobb helyzetben lennie. Mivel dBASE programjuk őrzi a más programokat kezelő információkat, ebből automatikusan következnének, hogy a dBASE mondja meg a többi programnak, miként tudják ezeket az információkat használni. Az új verzió, a tavaly napvilágot látott dBASE IV. két, kulcsfontosságú újítást tartalmaz. Az első, hogy egy dBASE képes több különböző számítógép és program kiszolgálására a hálózatba kapcsolódva. A második, hogy el van látva egy olyan programnyelvvel, a dBASE SQL-lel, amellyel a többi program kommunikálni tud, és képesek lesznek erre egymással is.

Ha a piac is úgy gondolja, az Ashton-Tate ellátja mind az integrált programok által használt ismertetőt, mind pedig a programnyelvet használati utasítással. Ily módon a felhasználók is jól járnak, mert a hálózatban dolgozó dBASE elvégezheti ugyanazt a munkát, amit egy kis- vagy nagygépes adatbázis — csak sokkal olcsóbban és rugalmasabban.

Az Ashton-Tate számára csak az a bökkenő, hogy sokan mások is felismerték ezt a kínálkozó alkalmat.

A szoftver világa nagyjából két részre osztható. A bázist a rendszerszoftverek szolgáltatják. Ilyenek a programnyelvek, az operációs rendszerek, melyek a számítógépi hardver összességét támogatják a többi program érdekében, valamint a „más programok futását segítő programok", beleértve a hálózati-kommunikációs szoftvertermékeket és így tovább.

A rendszerszoftverek haszonélvezőit nevezik felhasználói szoftvereknek. Ezek azok, amelyekkel a számítógépet használó ember találkozik: szövegfeldolgozók, táblázatszámító-programok, adatbáziskezelők és így tovább. Operációs rendszerek nélkül dolgozni, jó felhasználói szoftvereket előállítani sokkal, de sokkal nehezebb lenne. Minden programnak ezeregy dolgot kellene tudnia.

Sürgetve a szoftver integrálását, a szoftvercégek megpróbálnak ehhez a hierarchiához egy harmadik szintet hozzáadni. Eszerint olyan szoftvert kell készíteni, amely a felhasználói programokra épülne. A kulcsprobléma: kidolgozni, hogy mely programokra lehet építeni másikat.

„NAGYBANI SZOFTVER"

Természetesen a kommunikáció iránti igény nemcsak horizontális, hanem vertikális irányban is megfogalmazódott. Így a mikro-, mini- és nagygépes szoftvereknek is „szót kellene érteniük" egymással. Ma a világon körülbelül negyvenezer IBM nagyszámítógép található, az IBM és IBM-kompatibilis PC-k száma ugyanakkor jócskán meghaladta már a 10 milliót. Érthető hát, hogy a nagygépes cégek csak nehezen állják a versenyt: az agresszív mikroszámítógépes piacnak köszönhetően ezek a hatalmas arányeltolódások csak fokozódnak.

A nagygépes cégeknek az az egyetlen esélyük, ha kihasználják az elektronikus formában jelentkező, robbanásszerű információsigény-növekedést. A nagygépes eladók óriási erőfeszítéseket tesznek annak érdekében, hogy a képességek erőösszevonásával mint központi adatkönyvtárak működhessenek, és ezáltal hatásukat kiterjeszthessék a számítógépekre, amelyek így kénytelenek velük együttműködve dolgozni.

Ekképpen az adatbázis-szoftver szinte észrevétlenül az összes nagygépes szoftveres cég parancsnoki hajójává válik. Az ADR a Datacom alatt evez, a Software AG az Adabas által nőtt hatalmasra, a Cullinet ajánlata az IDMS. Az IBM két adatbázist ajánl: az IMS-t és a DB2-t.

A többi cég különböző módon növekszik. A Computer Associatesnek egy Unicenter névre hallgató programgyűjtemé-

nye van, amely a nagyszámítógépek futását hivatott segíteni. Az MSA szoftvere olyan menedzseri információs funkciókat ötvöz, mint a készletezési könyvvitel és a számlázás.

És amit minden nagygépes ajánl, az az út, amely növekvő súllyal veszi figyelembe a felhasználói igényeket a megrendelések teljesítésekor. Eszközéül az úgynevezett negyedik generációs nyelvek szolgálnak. Ezek óriási mértékben leegyszerűsítik a programozás műveletét, mivel bizonyos részfeladatokat elhagynak, a programozónak csak a számítógépnek adandó utasításokkal kell törődnie. A legtöbb esetben a program megírása szinte olyan egyszerű, mint egy formanyomtatvány kitöltése, s ez a termelékenység segíthet a vevőnek abban, hogy felülről lefelé érje el az integráltságot. A nagygépek programozásakor felügyelőként dolgozva, a nagygépes segítheti más programok és a számítógépek együttes munkáját.

Ami talán a leghathatósabb érv ebben a megközelítésben, az igazgatási vonatkozású. Mivel az összes információ a gépben van, bárki meggyőződhet róla, hogy ezek valódiak és naprakészek.

Sokan a Digital Equipment miniszámítógépeiben találják meg a kompromisszumot. A cég VAX gépeinek választéka az íróasztali, közel PC-s nagyságrendű gépektől egészen a majdnem nagygépekig terjed. És mindegyik ugyanazzal a szoftverrel képes futni, és nagyrészt maguk a nagyszámítógépes eladók menedzselték ezt a növekedési folyamatot a piacon.

Manapság az IBM megpróbálja a nagy- és kisgépek közötti választást még könnyebbé tenni. Úgy tervezik, hogy ezt általános szabványosítással érik el, amely képes lenne ugyanazon programokat futtatni nagygépeiken és PC-iken egyaránt. Ha ez sikerülne, akkor ledőlnének azok a korlátok, amelyek jelenleg elválasztják egymástól a nagygépes és a mikrogépes szoftverpiacot. És ekkor az IBM alakíthatná az integrálási folyamatot néhány gyártói szoftver segítségével. Úgy tűnik, ennek a nagygépes szoftvereladók cseppet sem örülnének . . .

SAA

A System Application Architecture (SAA) már hivatalos ígéret a szabványosításra. Ez a rendszer széleskörűen — a legtöbb IBM gépen — képes lesz könnyen kezelhető programok futtatására, ami lehetőséget nyújt majd a programok együttdolgozására és az adatmegszakításra.

A SAA ígérete szerint az IBM 9370-es nagygépén ugyanazon programok futtathatók lesznek, amelyek az új PS/2 PC-ken futnak. Ezáltal leegyszerűsödik az a munka, amely arra irányul, hogy az IBM számítógépek kommunikálhassanak egymással és más típusú gépekkel is.

Az SAA a szabványok három csoportjából áll: általános programozási interfészből, általános kommunikációs támogatásból és általános felhasználói hozzáférhetőségből.

Mindenki, aki a szoftveriparban tevékenykedik, valamilyen formában érdekelve van a Microsoft új OS/2 operációs rendszerének sikerében vagy bukásában. Az IBM-nek, amely segédkezett az OS/2 kifejlesztésében, ez valósítja majd meg az SAA szabványok mikroszámítógépes változatait, és egyben az első igazi megmérettetés lesz az SAA számára a piacon.

OS/2

Mi az OS/2-nek az a különleges tulajdonsága, amely alkalmassá teszi a PC-t több program egyidejű futtatására (a zsargonban ezt „multi-tasking"-nak nevezik)? A többfeladatos üzemmód nélkül bizonyíthatóan sokkal idegesítőbb procedúra rábírni a programokat az együttfutásra. A DOS például egyszerre csak egy dolgot képes elvégezni. Emiatt a felhasználó a billentyűzet mellett várakozhat akár jó néhány percet is, míg programja lefut, mondjuk azért, hogy hozzáférjen az adatbázis adataihoz. A többfeladatos üzemmel mindez a háttérben megy végbe, miközben a számítógép felhasználója másik munkát végezhet.

Van azonban egy korlát, ami miatt nem minden program képes a többfeladatos üzemre. Azok közül a programok közül, amelyek a DOS-ra íródtak — pedig a ma szoftvereinek túlnyomó többsége ilyen —, még az OS/2-vel is egy időben csak egy képes futni. De csak 1990-ig, mikorra is a Microsoft az OS/2 új változatát ígéri, amely már felhasználná az Intel 80386-os mikroprocesszorának előnyeit is, és így a felhasználók képesek lesznek több DOS-program egyidejű futtatására.

A többfeladatos üzemmód trükkje abban áll, hogy minden számítógépnek csak egy mikroprocesszora van, amely „szétosztható" több különféle program között.

A multi-tasking operációs rendszer például általában nagy sebességgel kapcsol át programról programra, mindegyik részére néhány ezredmásodpercig biztosítva a mikroprocesszorral való kapcsolatot, mielőtt a következő programra váltana. Hogy kézben tartsa ezt a „trükköt", az operációs rendszernek teljes mértékben irányítása alatt kell tartania a számítógépet és mindent, ami hozzá kapcsolódik. Ily módon például lehetőség nyílik arra, hogy egy program használni kezdje a nyomtatót, mielőtt egy másik befejezte volna vele a munkáját.

Ugyanakkor az OS/2-t fejlesztő Microsoft teljességgel tisztában van vele, hogy úgysem lehet ellátni az OS/2-t mindazzal, amit a felhasználók még hozzátennének. Ezért előkészületük során több mindent tettek a bővítés érdekében. A legfontosabb ezek közül — nevezzük dinamikus csatolásnak — ténylegesen képessé teszi a programozót arra, hogy saját kódját beolvassza az operációs rendszerbe, többékevésbé azonos szinten azzal, amellyel a Microsoft eleve ellátja azt.

A két első, aki haszonélvezője az OS/2-t bővítő dinamikus csatolás nyújtotta előnyöknek, ténylegesen a két kifejlesztő, a Microsoft és az IBM.

Tavaly jelent meg a „Presentation Manager" az OS/2-höz, amely finomítja és egyszerűbbé teszi azt a módot, ahogy a felhasználó utasítást ad a programnak. Egy másik bővítés, amely a LAN Manager névre hallgat, ki fogja bővíteni az operációs rendszert úgy, hogy lehetségessé válik a kapcsolattartás más számítógépekkel helyi hálózatokon (LAN) keresztül.

Egy harmadik, egyedül az IBM által meghirdetett bővítés képessé teszi majd az operációs rendszert arra, hogy információt kapjon az alatta futó programok számára az adatbázisokból.

A másik ügy, amelyben megbotolva hibázott az OS/2-vel a Microsoft, a szürke ló, amellyel a programozók kegyeiért versengett: a Unix.

Az 1970-es években az AT&T Bell Laboratóriumban dolgozó Ken Thompson és Dennis Ritchie által készített Unix talán minden idők legnagyobb szoftverújítása. Sajnos azonban, egyben talán minden idők legrosszabbul menedzselt programterméke is.

UNIX

A Unixot nagyrészt Thompson egymaga hozta létre, mivel a már meglevő operációs rendszerekről „szörnyű" volt a véleménye. És ez sok évig megoldott olyan dolgokat, amelyek az OS/2 és más termékek részéről csak ígéretek voltak: egy gyors, rugalmas környezetet, amely megkönnyíti több program együttműködését és adatcseréjét. A Unisys összes számítógépére ráépítette a Unixot — a kicsikre és nagyokra egyaránt —, így érve el a hordozhatóságot, amelyért az IBM még csak küzdött az SAA-val. Két kaliforniai cég, a Locus és a Microport Unix-ajánlásai képesek úgy megsokszorozni a DOS-programokat, hogy azok egyszerre futhassanak, a kiváló OS/2 pedig még két évig nem lesz képes erre.

A DOS és az OS/2 tervezésének nagy része, ez tény, Unix-inspirációjú volt. De a technikailag csodálatos Unixot kereskedelmileg rontották el. Amikor megírták, még az AT&T volt monopolhelyzetben, és tiltott volt kutatásainak felhasználása. Nyilvános környezetbe helyezve, a Unix nagymértékben megerősödött az egyetemeken és a mérnöki intézetekben. Amíg az ilyen, technikailag hozzáférhető felhasználók növelték a Unix képességeit, addig egyre komplexebb lett, és egyre érthetetlenebb a hozzá nem értők számára.

Sajnos az operációs rendszernek készült néhány különböző, nem kompatibilis változata is — így az egyikben megírt program nem biztos, hogy működött a másikban.

Az AT&T véghezvitte, hogy egy szabványos Unix képes lehet kisebb erőfeszítéssel a lehetséges felhasználói programokat íróknak nagyobb piacot biztosítani, és ezért kiterjeszti a hozzáférhető Unix-szoftverek mennyiségét. Az AT&T ekkor megmagyarázhatatlan módon megbízott két társaságot, hogy hozzon létre két szabványt. Egyikük a mérnöki munkahelyeket készítő, gyorsan növekvő Sun Microsystems. Még érthetetlenebb, hogy a másik a Microsoft. A Microsoftnak valóban olyan példányszámú Unix-eladásai voltak, mint senki másnak (250 000). De az OS/2 jövetelével többé nem volt érdekelve a Unix folyamatos sikerének erősítésében.

Ezzel együtt vagy a Unix, vagy még valószínűbb, hogy az OS/2 létrehozhatja az integráláshoz szükséges bővítéseket, hogy a felhasználói programok futtathatók legyenek alattuk.

Bíró András—Horváth Péter

LÜKTETHET A SZÍV

ANIMÁCIÓS BOSZORKÁNYKODÁS

A személyi számítógépek egyik legvonzóbb „képessége" a nagyfelbontású grafikának az alkalmazási lehetősége. Egy szemléltető program, egy jelenség szimulációja a legegyszerűbb grafikával is többet mond, mint egy táblázat, számsor vagy szöveges információ. A játékprogramok többsége pedig el sem képzelhető grafika nélkül. A legnagyobb hatást természetesen a rajzok, ábrarészletek animációjával, rajzfilmszerű mozgatásával érhetjük el.

Ha a megjelenítendő ábra egy periodikusan változó objektum (például egy lüktető szív a vérkörökkel vagy egy forgó kommutátoros motor), a képernyőn megelevenedő látvány magyarázó ereje igen nagy. Így nem csupán a felépítés, hanem a működés is szemléltethető. Ily módon a személyi számítógépek például az oktatástechnikában elérhetik az oktatófilmek, modellek hatékonyságát, sőt az egyéni hozzáférés és — interaktív kivitelnél — a beavatkozás (lassítás, leállítás, megfordítás, szöveges információ kérése az egyes fázisokban stb.) lehetősége révén minőségileg új szintet is jelenthetnek. Az elérhető látványosság révén természetesen bármilyen más programtípusnál is „megtérül" a befektetett többletenergia, ami géptípusonként igencsak eltérő lehet.

Az Enterprise 128 a hazánkban elterjedt személyi számítógépek között egyedülálló mértékben segíti az animációt. Sőt a hardver által lehetővé tett szolgáltatások nagy része BASIC-ből is elérhető. Ezen a géptípuson tisztán BASIC nyelven, viszonylag kis munkával írható olyan (vagy jobb) program, mint amilyen más típuson csak speciális alkalmazói szoftver (például C64, Giga Movie) vagy gépi kódú rutinok felhasználásával készíthető.

A folyamatos, vibrálásmentes mozgásérzet elérésének kulcsa:

— A látható képernyőn ne legyen nagyobb változás. Az egyes fázisok közötti átalakulás a háttérben, „láthatatlanul" menjen végbe!

— Az egyes fázisrajzok kellő gyakorisággal váltsák egymást! Elvileg másodpercenként legalább 16 képváltás szükséges a folytonosságérzethez.

Az Enterprise IS—BASIC-ben mindkét feltétel teljesíthető. A munka során igen sok grafikus csatorna (egy-egy rajzlap) nyitható. Ezek bármelyikére rajzolhatunk akár láthatóan, akár „vakon" a háttérben, és bármikor megjeleníthetők a képernyőn. Ezzel teljesül az első feltétel: egy-egy elkészített fázisrajz készen áll a memóriában a megjelenítésre várva.

A második feltétel is a hallatlanul sokoldalú videoegység (NICK-chip) és a kiszolgáló szoftver révén teljesíthető. A két fázisrajz közötti átkapcsolás gyakorlatilag pillanatszerű: az egyik térképet még az első rajz alapján generálja a chip, az 1/25-öd másodperc múlva következő képen már a másik rajzot látjuk.

Természetesen nem kis probléma a fázisrajzok időre való elkészítése. Az elvileg ideális megoldás — amelyben az új rajzot a megjelenések közötti időben a program készíti el — a BASIC sebességével nemigen érhető el. Csak egészen egyszerű változások esetén kapunk viszonylag folytonos mozgásképet. Ebben a technikában lényegében két lapon dolgozunk. A mozgássor felépítésének algoritmusa a következő lehet:

— az n. fázisrajz elkészítése az 1. lapon
— az 1. lap megjelenítése
— a 2. lap előkészítése [az (n−1). fázisrajz törlése]
— az (n+1). rajz elkészítése a 2. lapon
— a 2. lap megjelenítése
— az 1. lap előkészítése (az n. rajz törlése)
— az (n+2). rajz elkészítése az 1. lapon: vissza a ciklus elejére.

Lényegesen egyszerűbb a helyzet, ha egy ciklikusan ismétlődő rajzsorozatot szeretnénk megjeleníteni, vagy ha tetszőleges sorrendben, de adott rajzkészletből dolgozunk. Ekkor ugyanis a fázisrajzok (bemutatás előtt) tetszőleges idő alatt készíthetők el az egyes videolapokon. A megjelenítéskor ismét élvezhetjük az Enterprise előnyeit. A többi géptípuson általában egy — vagy igen kevés — memóriaterület jeleníthető meg. A kijelzésre szánt képet ideiglenes helyéről egy blokkmozgató rutinnal kell erre a területre vinni, aminek végrehajtási ideje általában nem elhanyagolható. Az Enterprise-on a felső négy szegmensben bármilyen memóriaterület megjeleníthető. Ezért ezen a gépen nem kell adatokat mozgatni. Elég a NICK-chippel „közölni", hogy honnan vegye a képinformációt, ami — még a BASIC interpreter közvetítésével is — elég gyors az animációhoz.

Az egyetlen szűk keresztmetszetet a lapok száma jelenti, ami persze függ a lapok választott méretétől is. Például a mintaprogramban választott 10×10-es lapoknál az operációs rendszer 29 lapnak foglal helyet a memóriában. Hatékonyabb memóriagazdálkodást az operációs rendszer ismeretében, gépi kódú programmal érhetünk el.

A *listán* látható mintaprogram — az animációs lehetőségek egyszerű illusztrációjaként — két egymást metsző színes kört mozgat egy 10×10 karakternyi méretű grafikus ablakban.

130—160 beállítja a fázisrajzok közös videofelvételeit
— nagyfelbontású grafikus üzem
— négy szín
— 10×10-es méret
200—230 megnyit 20 videolapot, és ezeken elkészíti a fázisrajzokat
250—260 billentyűlenyomásra vár
300—390 megjeleníti a kész fázisrajzokat
300—330 „előre" sorrendben
350—380 „hátra" sorrendben
390 végtelen ciklusban ismétli a megjelenítést
430—450 késleltetés az optimális mozgásérzet eléréséhez
500 megjeleníti a megnyitott lapot (ez a sor futtatáskor elhagyható)
510 színválaszték
520 megrajzolja a lap keretét
530—570 megrajzolja és sárgára festi az első kört
580—620 megrajzolja és pirosra festi a második kört
630—650 zöldre festi az első kör másodikkal nem közös részét

```
  100 !
  110 !ELOKESZITES
  120 !
  130 SET VIDEO MODE 1
  140 SET VIDEO COLOR 1
  150 SET VIDEO X 10
  160 SET VIDEO Y 10
  170 !
  180 !LAPOK NYITASA
  190 !
  200 FOR N=1 TO 20
  210   OPEN #N:"VIDEO:"
  220   CALL RAJZOK(N)
  230 NEXT
  240 !
  250 LET A$=INKEY$
  260 IF A$="" THEN GOTO 250
  270 !
  280 ! MEGJELENITES
  290 !
  300 FOR N=1 TO 20
  310   DISPLAY #N:AT 1 FROM 1
TO 10
  320   GOSUB 430
  330 NEXT N
  340 !
  350 FOR N=20 TO 1 STEP-1
  360   DISPLAY #N:AT 1 FROM 1
TO 10
  370   GOSUB 430
  380 NEXT
  390 GOTO 300
  400 !
  410 !KESLELTETES
  420 !
  430 FOR T=1 TO 10
  440 NEXT
  450 RETURN
  460 !
  470 !FAZISRAJZOK
  480 !
  490 DEF RAJZOK(N)
  500   DISPLAY #N:AT 1 FROM 1
TO 10
  510   SET #N:PALETTE BLACK,G
REEN,YELLOW,RED
  520   PLOT #N:0,0;319,0;319,
359;0,359;0,0,
  530   SET #N:INK 1
  540   PLOT #N:-10+16*N,160,
  550   PLOT #N:ELLIPSE 70,70,
  560   SET #N:INK 2
  570   PLOT #N:PAINT
  580   SET #N:INK 3
  590   PLOT #N:328-16*N,200,
  600   PLOT #N:ELLIPSE 70,70,
  610   PLOT #N:328-16*N,245,
  620   PLOT #N:PAINT
  630   SET #N:INK 1
  640   PLOT #N:-10+16*N,130,
  650   PLOT #N:PAINT
  660 END DEF
```

Más rajzsorozatnál a 170—310-es sorok helyett építhető be a saját rutin vagy az azt hívó utasítás. Természetesen az előre-hátra pörgetés is fölösleges lehet. A DISPLAY utasítások AT utáni paraméterével adhatjuk meg, hogy a 10 sornyi grafikus ablak a képernyő hányadik sorától jelenjen meg. Ha az animált ablakot egy előre megrajzolt grafikus képernyő közepére „vetítjük", ügyes rajztechnikával elérhető, hogy a mozgó képrészletet alul-felül egy olyan álló háttér egészíti ki, mint amilyen például egy motor forgó- és állórésze.

Felföld József

MEGKÉRDEZTÜK AZ ENTERPRISE -RÓL

Néhány napos magyarországi látogatásakor, megragadva az alkalmat, H. M. Windisch úrral, az NSZK-beli ENTERPRISE Computers GmbH igazgatójával sikerült interjút készítenünk, aki szívesen válaszolt az ENTERPRISE cég eddigi tevékenységére és terveire vonatkozó kérdéseinkre.

M. M. Az Enterprise általános megítélés szerint teljesen újszerű és remek konstrukció. Összefoglalná-e a gép fejlesztésének „filozófiáját"?

H. M. W. Az Enterprise számítógépeket nagyon kedvezőtlen körülmények között bocsátottuk útjukra. Bár az előzetes elképzelések közül nagyon sok helyénvalónak bizonyult, mégis súlyos hibákat követtünk el, amelyeket nem lehetett korrigálni. A műszaki koncepciónk kiindulópontja és feltétele az volt, hogy egy átlagos teljesítményű, piacképes, a termék és annak tudása szerint úgymond „ármegfelelő" gépet állítsunk elő. Az Enterprise ismerőinek nem újdonság, hogy az operációs rendszer és ennek bővítései, az IS—DOS, az EXDOS és az EXOS ma még egyetlen más hasonló gépnél sem érik el ezt a szintet a teljesítmény szempontjából.

M. M. Az Enterprise a rossz marketingtevékenység miatt bukott meg a nyugat-európai országokban. Miben nyilvánult ez meg?

H. M. W. A piaci bevezetés első szakaszában többször módosítottuk a termék nevét. A bejelentett szállítási időpontokat nem tartottuk be a két speciális chippel, a Dave-vel és a Nick-kel kapcsolatos műszaki nehézségek miatt. Ezenkívül téves marketing döntések születtek, érthetetlenül rossz árpolitikát folytattunk. Mindezek a tervezett termelési költségek tekintélyes túllépéséhez és a várt, illetve kalkulált eladások kieséséhez vezettek. Mivel az Enterprise-t a Mahtani család finanszírozta, ez azt eredményezte, hogy az 1985/86. években az NSZK-ba irányuló és a döntő piaci fázisban meghatározott eladási célokhoz nem állt rendelkezésre elegendő pénz. A fejlesztésre húszmillió USA-dollárt költöttek, és így az értékesítésre nem maradt semmi. Ehhez járult még, hogy a müncheni ENTERPRISE üzletvezetése olyan üzleti stratégiát követett, amely egyszerűen nem vált be. Így bár 1985-ben Münchenben még további ötmillió dollárt költöttek a fejlesztésre, a célokat mégsem érték el. 1986 elején az ENTERPRISE úgy döntött, hogy konszolidálja tevékenységét. Ezzel oda jutottak, hogy 1986-ban a müncheni cég kivételével az összes külföldi vállalatukat bezárták, a londoni központ pedig önálló elszámolású lett.

M. M. Milyen tevékenységet folytat a müncheni cég?

H. M. W. Az ENTERPRISE Computers GmbH München a hardver és a szoftver területén minden jogot elnyert, és a fejlesztésekben is központként illetékes. Az utóbbi két évben az ENTERPRISE folyamatosan javított a működési módszerén és számos szoftvert fejlesztett. A hardvert aktualizálták, teljesítményét javították és továbbfejlesztették. Ma az eredeti Enterprise mellett olyan, már ismert perifériák vannak — EXDOS controller, nyomtatók, monitorok, tárbővítők egészen 576 kbájtig, valamint egy sor komplett lemezmeghajtó —, amelyek kapacitása 180-tól 720 kbájtig terjed. Sőt lehetőségünk van már arra, hogy 60 Mbájtos összkapacitású, két merevlemezegységet az Enterprise-hoz csatlakoztassunk. A fenti fejlesztéseken kívül éppen most dolgoznak két univerzális hardverbővítő rendszeren, melyeknek segítségével az Enterprise a tetszőleges szintre bővíthető. Így többek között mérési adatok regisztrálására (áram, feszültség, ellenállás, hőmérséklet stb.). Többféle kártya is az előkészítésnél tart. Az Enterprise mérőhely-számlálóként való alkalmazásának semmi sem áll útjában.

M. M. Elterjedt a hír, hogy új modelleken is dolgoznak. Mondana erről néhány szót?

H. M. W. A már ismert 64 és 128 k-hoz tartozó kiegészítőkön kívül természetesen egy következő modellen is dolgozunk, amely a professzionális igényeket is kielégíti majd. A magyarországi Enterprise-sikerek arra késztették a céget, hogy vizsgáljuk meg, vajon az alapgép és tartozékai gyártására Magyarországon adottak-e a megbízható műszaki feltételek, és versenyképes áron tudnák-e gyártani. Már feladtunk rendelést hat mintapanel elkészítésére. A későbbiekben az ENTERPRISE érdekelt lenne egy, a magyar vállalatokkal folytatandó kooperációban, amely az Enterprise gépet használók részére a kellő mennyiséget és természetesen a sokoldalú hardver- és szoftverkínálatot elérhetővé tenné. A gyártás Magyarországon kívüli igényeket is kielégíthetne. Amit még hiányolunk Magyarországon, az egy Enterprise újság. Olyan, mint például Münchenben az úgynevezett „Enterprise-t használók csoportja" nevű, az „ENTERPRISE USER GROUPS", amelyet klubújságként adunk ki. Ez egy hotline (forródrót), amelyhez általában minden Enterprise-rajongó, a programok előállításához indíttatást érző szabadon hozzájuthat. Az olvasók többek között pályázatokat és versenyeket nyerhetnek. A müncheni ENTERPRISE szívesen támogatna egy hasonló lapot Magyarországon is.

Hajnal Csaba — Pinke György

Fedezzük fel együtt!

Gyönyörű színkavalkád

Az előző részben az ismerkedés módszerével és a különböző geometriai alakzatok mozgatásával foglalkoztunk. Most próbáljuk ki a gép színezési lehetőségeit.

A *6. listán* lévő programmal az Enterprise 256 színében gyönyörködhetünk, és megtudhatjuk a színek kódszámait. A kódszámok kiírásához a karakterláncokkal végezhető műveleteket és függvényeket kell felismernünk (F/51).

A program sorainak magyarázata:

160 A K+I numerikus kifejezés értékét karakterlánccá alakítja az STR$ függvény.

170 A LEN függvény a K$ karakterlánc-változó értékének hosszát, a karakterek számát határozza meg.

190 A TAB függvény hatására az L. karakterpozícióba írja ki K$ értékének N-2. karakterét.

220—240 5 sorral feljebb visszük a kurzort a 176-os kódszámú karakter ismételt kiírásával.

260—270 A GET utasítással a Q$ változóba azt a karaktert olvassuk be, amelyiknek a billentyűjét lenyomjuk. Ha nem nyomunk le egy billentyűt sem, akkor Q$ értéke az „üres karakter". Ezt vizsgáljuk a 270-es sorban.

Egyszerre 18 vonalat tudunk kirajzolni a programmal, ezért a K értékét kell beállítani a 120-as sorban a program futtatása előtt, ha a többi színt is látni akarjuk.

Ismerős kép jelenik meg a képernyőn, ha a *7. lista* programját begépeljük és elindítjuk.

140—180 Különböző színekkel egymás mellé kiíratjuk az X$ változó értékének karaktereit.

200—230 Véletlenszerűen választott színekkel kiírjuk a véletlenszerűen kiválasztott karaktereket. Az ENTERPRISE szót a grafikus képernyőre, a másik szöveget a szöveges képernyőre írjuk ki.

A 2, 4 és a 16 színű grafikus képernyő színezésének vizsgálatát segíti a *8. lista* beírása. A SET PALETTE utasításban nyolc szín kódszámát kell megadnunk. A programban az első nyolc színkódot soroltuk fel. A SET BIAS a következő nyolc színt jelöli ki. A SET INK utasítással — a 2, 4 és 16 színű képernyőnél — a PALETTE és a BIAS utasításokkal kijelölt színek sorszámait kell megadnunk. Ezt változtatjuk a FOR ciklussal.

A program futtatása után adjuk ki a SET BIAS 0 parancsot! Azt látjuk, hogy a második nyolc szín ugyanaz lesz, mint az első nyolc. Ez a nyolc szín csak akkor változik, ha a SET BIAS parancsban hétnél nagyobb kódszámot adunk meg.

A kétszínű és négyszínű képernyőkön csak az első kettő, illetve az első négy színt használhatjuk: a 0, 1, illetve a 0, 1, 2, 3 sorszámúakat. Nagyobb sorszám megadásakor a számítógép néggyel osztja a sorszámot, és a maradék jelöli ki a színt. Adjuk ki a SET PALETTE BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE parancsot! A színeket megadhatjuk az RGB függvénnyel is (F/85). A grafikus képernyőre írunk a 170-es és a 200-as sorokban. A számítógép tárolja a grafikus képernyő tartalmát. Ezt a lapot kell kijelölnünk, ha írni akarunk a grafikus képernyőre.

Futtassuk le a 7. programot, és nyomjuk le a DISPLAY TEXT funkcióbillentyűt (SHIFT FUNCTION 5), majd a DISPLAY GRAPHICS funkcióbillentyűt (SHIFT FUNCTION 6). Így hívhatjuk elő a grafikus, illetve a szöveges képernyőt.

Nyomjuk le a DISPLAY TEXT funkcióbillentyűt, és adjuk ki a

DISPLAY £101:AT 6 FORM 1 TO 5

parancsot! A grafikus képernyő első öt sorát előhívtuk a 101-es lapról, és megjelenítettük a 6. sortól kezdődően.

Nyomjuk le a TEXT billentyűt és adjuk ki a

DISPLAY £102:AT 12 FROM 1 TO 4

parancsot! A 12. sortól kezdődően ugyanaz, mint az első négy sorban. A 101-es lap tehát a grafikus, a 102-es a szöveges képernyőt tárolja. A többi 100 lapról később lesz szó.

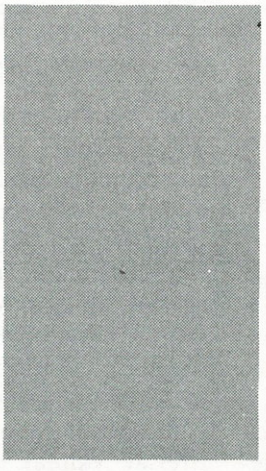
A grafikus képernyőkön a PLOT utasítással jelölhetjük ki a kiírás helyét. A GRAPHICS ATTRIBUTES utasítással olyan grafikus képernyőn dolgozhatunk, amelyen a kiírás helyét a sor és soron belül a karakterpozícióval adhatjuk meg (*9. lista*). A SET ATTRIBUTES utasítás a karakterek és a rajz színét befolyásolja (F/149).

A *10. listán* látható program talán ötletet ad egy játékprogram megírásához.

Felfedező utunk során arra mindig kell időt szakítanunk, hogy ismereteinket, tapasztalatainkat rendszerezzük és kiegészítsük újabb, talán kevésbé fontos, nélkülözhető, ugyanakkor munkánkat megkönnyítő vagy újabb lehetőségeket nyújtó ismeretekkel.

Nézzük a programok begépelését, javítását, módosítását. A betűk, számok, írás- és műveleti jelek begépelésére szolgáló billentyűkön kívül az ENTER, SHIFT, valamint az ERASE, DEL és az INS billentyűket használtuk eddig. Gyorsabban mozgat-

6. lista



```
100 REM ---6.program---
110 GRAPHICS LORES 256
120 LET K=220
130 FOR I=0 TO 18
140   SET INK K+I
150   PLOT 64*I,0;64*I,719
160   LET K$=STR$(K+I)
170   LET N=LEN(K$)
180   LET L=2*I+1
190   PRINT TAB(L) K$(N-2)
200   PRINT TAB(L) K$(N-1)
210   PRINT TAB(L) K$(N)
220   FOR J=1 TO 5
230     PRINT CHR$(176);
240   NEXT
250 NEXT
260 GET Q$
270 IF Q$="" THEN 260
```

7. lista

```
  100 REM  --- 7. program ---
  110 REM   --- enterprise ---
  120 GRAPHICS HIRES 256
  130 RANDOMIZE
  140 LET X$="ENTERPRISE"
  150 FOR I=1 TO 10
  160   SET INK RND(256)
  170   PLOT I*128-128,360,
  180   PRINT #101:X$(I)
  190 NEXT
  200 PRINT TAB(5) "C 1985 Intelligent Sof
tware Ltd"
  210 SET INK RND(256)
  220 LET I=RND(10)
  230 PLOT I*128,360,
  240 PRINT #101:X$(I+1)
  250 GET Q$
  260 IF Q$="" THEN 210
```

hatjuk a kurzort és gyorsabban végezhetjük a javítást, ha tudjuk, mi történik akkor, amikor a botkormányt mozgatjuk, vagy az ERASE, az INS, a DEL billentyűk valamelyikét lenyomjuk, miközben lenyomva tartjuk a SHIFT vagy a CTRL billentyűt (lásd a Felhasználói kézikönyv 39—40. oldalán). Irassuk ki többször valamelyik programot a képernyőre, és próbáljuk ki, mi történik a billentyűk használatakor.

A programkezelő parancsok közül a legfontosabb a NEW, a RUN és a LIST parancs. Az AUTO parancs megkímél a sorszámok begépelésétől, a DELETE paranccsal utasítássorozatokat törölhetünk, a RENUMBER átsorszámozza a programot. A parancsok közül néhányat a funkcióbillentyűkkel is megadhatunk (Felhasználói kézikönyv, 46. oldal).

Tegyünk különbséget a HOLD és a STOP billentyűk hatása között. Az első csak leállítja, a második megszakítja a program futását. Megszakításkor a program futása a STOP billentyű lenyomása után, a CONTINUE parancs hatására folytatódik. A programok mentése a SAVE, ellenőrzése a VERIFY, beolvasása a LOAD paranccsal lehetséges. A magnetofon távvezérlésének ki- és bekapcsolására a REM1, illetve a REM2 parancsok szolgálnak. Ez attól függ, hogy a távvezérlőt melyik bemenettel kötöttük össze.

A parancsok részletes leírását a Felhasználói kézikönyvben a 119. oldaltól kezdődően kell keresnünk.

A következő részekben az Enterprise grafikus képességeivel foglalkozunk.

Dusza Árpád

8. lista

```
  100 REM    --- 8. program ---
  110 REM --- szinek 2-4-16 ---
  120 GRAPHICS LORES 16
  130 SET PALETTE 0,1,2,3,4,5,6,7
  140 SET BIAS 8
  150 FOR I=0 TO 15
  160    SET INK I
  170    PLOT I*64,0;I*64,600
  180    PLOT 0,719,:PRINT #101:I
  190    GET Q$
  200    IF Q$="" THEN 190
  210    PLOT 0,719,:PRINT #101:CHR$(159) C
HR$(159) CHR$(159)
  220 NEXT
```

9. lista

```
  100 REM    --- 9. program ---
  110 REM    --- vegyes kepernyo ---
  120 GRAPHICS ATTRIBUTE
  130 SET BORDER 42
  140 SET PALETTE 0,25,125,12,2,13,38,255
  150 SET BIAS 60
  160 LET X$="MISKOLC":LET J=15
  170 FOR K=0 TO 7
  180    SET ATTRIBUTES 2^K
  190    LET L=LEN(X$)*32+32:LET I=2*K+2
  200    LET X=J*32-48:LET Y=720-I*36-18
  210    SET INK 8+K
  220    PRINT #101,AT I,J:X$ "    " 2^K
  230    SET INK K
  240    PLOT X,Y;X+L,Y;X+L,Y+72;X,Y+72;X,Y
  250 NEXT
```

10. lista

```
  100 REM  --- 10. program ---
  110 GRAPHICS ATTRIBUTE
  120 LET J=0:LET X$=" "&CHR$(154)
  130 PRINT #101,AT 10,J:X$
  140 LET J=J+1
  150 IF J<40 THEN 130
```

Mi a manó?

Német nyelvű gép átépítése. Az ENTERPRISE Computers GmbH München cég közölte szerkesztőségünkkel azt, hogyan lehet a német nyelvű gépet angol nyelvűvé átépíteni. Ehhez egy darab UM RIN típusú miniatűr kapcsoló kell, aminek rendelési száma MS 24 L 244. Az átalakítást a következőképpen csináljuk:

— A gép tetejét rögzítő csavarokat vegyük ki, így a tető levehető. Húzzuk ki az érinkezőket.
— A 08—59-es EPROM-ot az U1 csatlakozó aljzatba helyezzük.
— A 08—46—A számú ROM-ot az UÁ-es csatlakozóba dugjuk.
— Távolítsuk el az LK1 összekötést, azaz vágjuk el a huzalt.
— Ki kell iktatni az *1. ábra* szerinti kapcsolót.
— Fúrjunk a ROM felső részébe egy öt mm átmérőjű menetet a *2. ábra* alapján.
— Helyezzük be az érintkezőket a dobozház alsó részébe.
— Tegyük vissza a gép tetejét, rögzítsük a csavarokat.
— A kapcsolót a hozzá mellékelt csavarokkal erősítsük fel.

Ezzel átkapcsolhatóvá tettük a gépet. Ha a kapcsoló zárt állásban van, akkor a gép német nyelven üzemel. Nyitott állapotban angol nyelvű BASIC-kel dolgozik. A gép üzem közben is átkapcsolható, de ez hideg indítást eredményez.

A következő számunkban részletes leírást közlünk az átépítésről és megkérdezzük a Professional szerviz vezetőjének véleményét is.

A német nyelvű gépen lehetőség van a grafika tárolására a :VSAVE és a :VLOAD utasítás illetve parancs segítségével. A :VSAVE paranccsal kazettára vagy lemezre kimenthetjük az általunk megnyitott grafikus csatorna tartalmát. Parancsmódban ez a következőképpen történik:

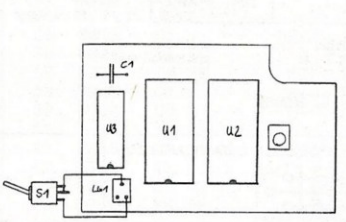
:VSAVE#[csat. sz.]: "[eszköz]:[adatnév]"

A csatornaszámhoz az általunk megnyitott videocsatorna számát kell beírni. Az eszköz helyére az adatrögzítő típusát, amelyen tárolni akarunk (TAPE vagy A). Az adatnév megengedett hosszúságú karakterlánc lehet.

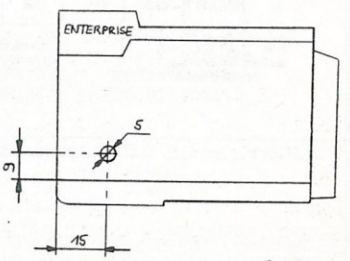
Ha BASIC programból utasításként adjuk ki, akkor a szintaktikája a következőképpen alakul:

EXT "VSAVE" #[csat. sz.]: ""[eszköz]:[adatnév]""""

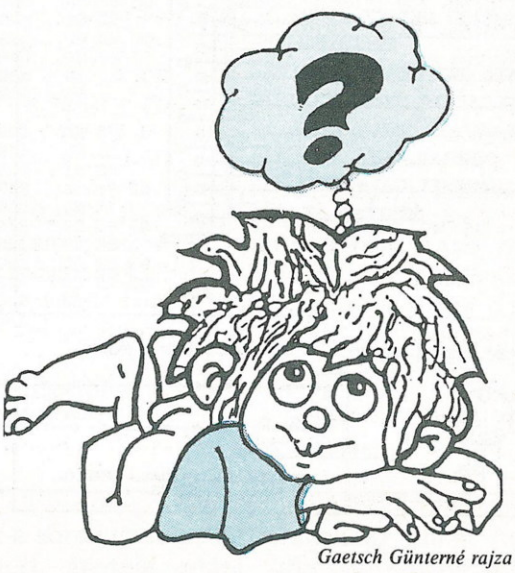
Figyelembe kell venni, hogy a VSAVE paranccsal csak a grafikát tárolhatjuk, a kiválasztott színeket nem. A tárolt grafikát a VLOAD paranccsal tölthetjük be a tetszőlegesen megnyitott videolapra. Természetesen csak a kép tartalmát kapjuk vissza, a színeket újra be kell állítani a BASIC progamból. A szintaktikája megegyezik a VSAVE paranccsal illetve utasítással.



1. ábra



2. ábra



Gaetsch Günterné rajza

Hardver

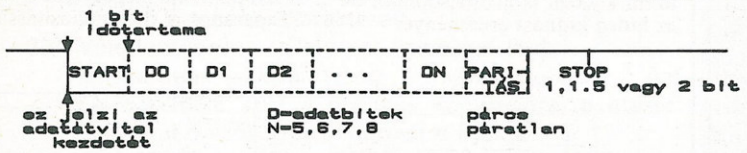
A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan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

SOROS ADATÁTVITEL

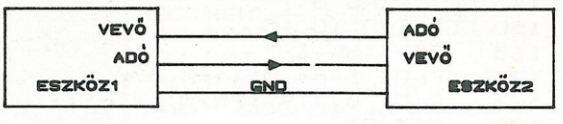
A személyi számítógépeknél az egyik legjobban elterjedt adatátviteli forma a soros átvitel. Ez azt jelenti, hogy a bitcsoportba kódolt információt egy vonalon, bitenként, sorban egymás után visszük át. Természetesen ez lassítja az információátviteli sebességet, de olcsó, mert az információ áramlásához csupán néhány vezeték szükséges. Soros adatátvitelnél definiálni kell az adatátvitel mikéntjét (protokollját), és az átvitel fizikai megvalósítását, vagyis az áramkört, ami az átvitelt megvalósítja. A továbbiakban csupán a legelterjedtebb szabványos RS232C soros adatátvitelt mutatjuk be, anélkül, hogy teljességre törekednénk. A gyakorlati szempontokat szem előtt tartva, kitérünk mind az átviteli protokoll, mind a fizikai kialakítás kérdéseire.

Ahhoz, hogy az ADÓ-ról érkező biteket a VEVŐ egyértelműen azonosíthassa, szükséges, hogy szinkronban legyenek, azaz például egy bájt ötödik bitjét kiküldve, azt a VEVŐ is annak tekintse. Ennek egyik megoldási módja, hogy a bitek küldésével párhuzamosan egy órajelet is küldünk, és az órajel valamelyik fizikai jellemzőjéhez — éléhez, szintjéhez — rendeljünk az átvitt bit érvényességét. Ez az órajel szinkront teremt az ADÓ és VEVŐ között, amit szinkron soros adatátvitelnek nevezünk. Ehhez hasonló a C64 soros vonalán is az információáramlás.

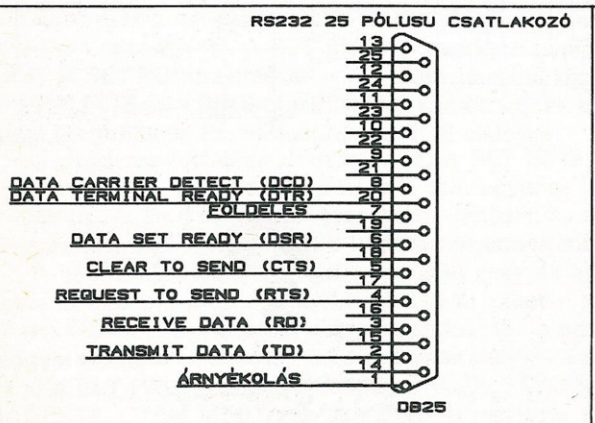
Az aszinkron soros átvitelnél nincs külön órajelet hordozó vonal, hanem a bitcsoportos átviteli mód hozza létre az ADÓ és a VEVŐ szinkronját. Természetesen ehhez az együtt-



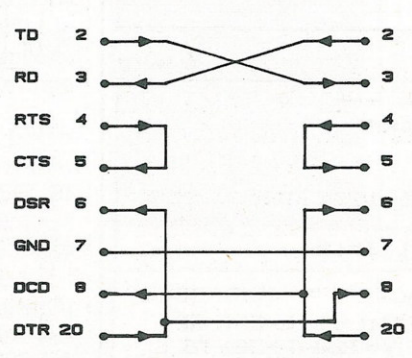
1. ábra



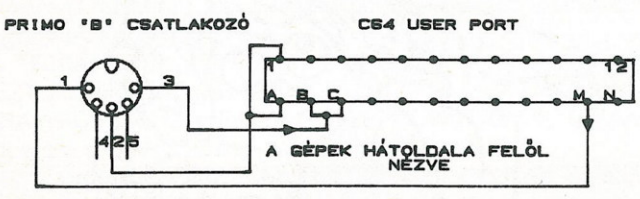
2. ábra



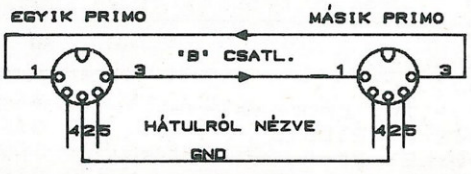
3. ábra



4. ábra



5. ábra



6. ábra

futást megvalósító járulékos információhoz járulékos biteket is fel kell használni. Ezek a START és a STOP bitek, amelyeket keretező (framing, ejtsd: fréming) biteknek is szoktak nevezni, mivel a tényleges információt „keretbe foglalják".

A START bit jelzi, hogy utána következnek a tényleges információt hordozó adatbitek, a STOP bit(ek) pedig ezek végét jelzi(k). Az aszinkron soros protokoll szerint, ha a soros vonalon nincs információátvitel, a vonal állapota logikai 1 szinten van. Az adatátvitel kezdetekor az ADÓ a vonalat egy bit átvitelének idejéig logikai 0 szintre húzza le (START bit), majd utána következik az adatbitek átvitele. Az átvitt adatbitekből álló bitcsoport végére az ADÓ STOP bit(ek)ből álló logikai 1 szintű információt helyez el. A VEVŐ az adás tényéről a vonal 1-0 állapotváltozásából szerez tudomást. Ezután egy bit átvitelének idejéig várakozva, az adatbiteket veszi, a STOP bitek érkezése alatt pedig már figyelheti a vonalon ismét megjelenő 1-0 állapotváltozást, ami a következő bitcsoport adásának kezdetét jelöli. A leírtakat az *1. ábra* illusztrálja.

Fontos megemlítenünk a vonalon időegység alatt átvitt információ sebességét, amit bitütemnek vagy közismert angol kifejezéssel baud-rate (bódrét)-nek nevezünk, és bit/s-ban mérünk. Tipikus, szabványosan használt értékei az alábbiak:

Bit/s	Egy bit átvitelének ideje (ms)
19 200	0,0521
9 600	0,1041
4 800	0,2082
1 200	0,8333
600	1,6666
300	3,3333
150	6,6666

Az adatátvitelkor az esetleges átviteli hibák felderítését megkísérelhetjük úgy, hogy az átviendő adatbitcsoportot egy ún. paritásbittel egészítjük ki úgy, hogy az így kiegészített adatcsoportban lévő 1 értékű bitek száma páros (páros paritás) vagy páratlan (páratlan paritás) legyen. Ekképpen elérhetjük azt, hogy az ADÓ oldalán az 1-es értékű bitek száma mindig páros/páratlan legyen, így a VEVŐ oldalon az egy bitváltozás miatti hiba felderíthető.

Legyen például az adatunk 01101001 alakú. Páratlan paritás választásakor a paritásbit értéke 1, vagyis az 100101101 sorrendben (a legkisebb helyiértékű bit megy ki először!) küldjük ki az adatokat. Tegyük fel, hogy ezt a VEVŐ hibásan veszi, történetesen a 0. adatbitet, ami eredetileg 1 értékű volt, 0-nak értékeli: 000101101. A VEVŐ oldalán a páratlan paritás ellenőrzésekor ez a hiba kiderül, mivel az adatcsoportban lévő 1 értékű bitek száma páros lesz. Ezt a hibát paritáshibának nevezzük.

Természetesen, ha egyszerre két vagy több bit értéke változik meg, a paritásellenőrzés a hibát nem mindig jelzi. Statisztikailag igazolható, hogy az ilyen „hibacsomósodások" valószínűsége a gyakorlatban csekély, és a legtöbb esetben a paritásellenőrzés az adatátvitelt elég megbízhatóvá teszi. A fentiek alapján a soros adatátviteli protokoll konkrét kialakításánál a következőket kell meghatározni:

— az adatbitek számát: ez gyakorlatilag 5,6,7 vagy 8;

— a paritásbitet: használunk-e paritásbitet vagy nem, ha igen, páros vagy páratlan paritást;

— a STOP bitek számát: ez a soros vonalnak a bitcsoport átvitele utáni garantált logikai 1 állapotának az idejét határozza meg, az egy bit átviteléhez szükséges idővel kifejezve.

```
                    ***Z80 PROGRAM***

Programnév: SOROS                       Programhossz: 105 bájt

A program feladata: RS 232 szabványnak megfelelô soros adat-
                    átviteli formátum generálása : 1200 baud
                    1 START 8 ADAT 2 STOP bit

Bemenô adatok: A: beolvasott bájt

Kimenô adatok: A: kiküldött bájt

Felhasznált regiszterek: A

Megjegyzés: Rutinok hívása: CALL RSBE, CALL RSKI

                         **********

            LISTA:

  1          ;RS 232 HANDLER (KEZELÖ) PRIMORA
  2          ;1200 BAUD 1 START 8 ADAT 2 STOP BIT
  3  8000                    ORG 8000H
  4          ;
  5          0040     PORTBE: EQU 40H
  6          0000     PORTKI: EQU 00H
  7          403B      TUKOR: EQU 403BH :Ø.PORT MASOLATA
  8          ;
  9  8000    C5         RSBE: PUSH BC ; ADATOT FOGADÓ RUTIN
 10  8001    DB 40       BE1: IN A,(PORTBE)
 11  8003    CB 47            BIT Ø,A ; A Ø. BIT AZ AKTÍV
 12  8005    20 FA            JR NZ,BE1 ; LEFUTÓ ÉL FIGYELÉSE
 13  8007    CD 2E 80         CALL DEL2 ; FÉL BITIDÖ KÉSL.
 14  800A    DB 40            IN A,(PORTBE)
 15  800C    CB 47            BIT Ø,A
 16  800E    20 F1            JR NZ,BE1 ; CSAK ZAVAR VOLT
 17  8010    06 09            LD B,9 ; START+8 ADATBIT
 18  8012    DB 40     HUROK: IN A,(PORTBE)
 19  8014    CB 47            BIT Ø,A ; AZ ELSÖ A START BIT
 20  8016    37               SCF ; CY=1
 21  8017    20 01            JR NZ,BE2 ; HA A BIT = 1
 22  8019    3F               CCF ; CY=Ø
 23  801A    CB 19       BE2: RR C ; C-BE CY
 24  801C    CD 2E 80         CALL DEL2
 25  801F    CD 2E 80         CALL DEL2 ; EGY BITIDÖ KÉSL.
 26  8022    10 EE            DJNZ HUROK ; A TÖBBI BIT
 27  8024    04               INC B ; B=1
 28  8025    DB 40            IN A,(PORTBE)
 29  8027    CB 47            BIT Ø,A
 30  8029    20 E7            JR NZ,HUROK ; NEM STOP BIT JÖTT
 31  802B    79               LD A,C ; BAJT A-BAN
 32  802C    C1               POP BC
 33  802D    C9               RET
 34          ;-------------------------------
 35  802E    C5         DEL2: PUSH BC  ; KÉSLELTETÖ RUTIN
 36  802F    06 4C            LD B,4CH ; EZ A PARAMÉTER!
 37  8031    10 FE       KES: DJNZ KES ; IDÖZÍTÖ HUROK
 38  8033    C1               POP BC   ; FÉL BITIDÖ
 39  8034    C9               RET
 40          ;-------------------------------
 41  8035    C5         RSKI: PUSH BC ; ADATOT KIVIVÖ RUTIN
 42  8036    4F               LD C,A ; C=ADATBAJT
 43  8037    CD 52 80         CALL SENDØ ; START BIT = Ø
 44  803A    06 08            LD B,8
 45  803C    CB 39       KI1: SRL C
 46  803E    38 05            JR C,KI2 ; HA A CY-BE 1-ES VAN
 47  8040    CD 52 80         CALL SENDØ ; KÜLÖNBEN Ø KI
 48  8043    18 03            JR KI3
 49  8045    CD 59 80    KI2: CALL SEND1 ; EGY BITIDÖIG 1 KI
 50  8048    10 F2       KI3: DJNZ KI1
 51  804A    CD 59 80         CALL SEND1
 52  804D    CD 59 80         CALL SEND1 ; EZ A KÉT STOP BIT
 53  8050    C1               POP BC
 54  8051    C9               RET
 55          ;-------------------------------
 56  8052    3A 3B 40  SENDØ: LD A,(TUKOR) ;Ø KIKÜLDÉSE
 57  8055    CB B7            RES 6,A
 58  8057    18 05            JR KI4
 59  8059    3A 3B 40  SEND1: LD A,(TUKOR) ;1 KIKÜLDÉSE
 60  805C    CB F7            SET 6,A
 61  805E    32 3B 40    KI4: LD (TUKOR),A ; TAROLÁS
 62  8061    D3 00            OUT (PORTKI),A
 63  8063    CD 2E 80         CALL DEL2
 64  8066    CD 2E 80         CALL DEL2
 65  8069    C9               RET
 66  806A                     END


Lines Assembled :  66
Assembly Errors :  Ø

                         **********
```

Hossza 1, 1,5 vagy 2 bit lehet. A legrövidebb az egy bit, és ez teszi lehetővé, hogy a VEVŐ a következő bitcsoport vételéhez szükséges szinkronizáló START bit 1-0 lefutó élének érzékelésére felkészüljön. Két STOP bit használata akkor előnyös, ha valamilyen okból azonnal szükséges a vett adatbitek feldolgozása;

— az adatátviteli sebességet, ami igen fontos adat, mert ez határozza meg alapvetően az ADÓ és a VEVŐ szinkronizmusát.

Mivel a soros adatátvitelt széles körben használják, ezért céláramköröket fejlesztettek ki a megvalósítására. Ezeknél az ADÓ oldalán csupán az adatbitcsoportot kell párhuzamosan a bemenetekre adni, az áramkör elvégzi a sorosítást, a paritás-, a START, STOP bitekkel való kiegészítést és az átvitelt. A vevő oldalon a vett soros adatokból vevőáramkör képzi a bitcsoportot. Ezek az áramkörök programozhatók, azaz vezérlőkódokkal megadható az átvitel jellege (szinkron vagy aszinkron) és a soros adatátviteli protokoll. Ezeket az általában mikroprocesszorokhoz kapcsolódó periféria-áramköröket betűszavakkal szokták jelölni, ahol az egyes betűk jelentése:

U — univerzális
S — szinkron
A — aszinkron
R — vevő
T — adó

Az egyik leggyakrabban használt és talán legismertebb típus az Intel 8251 USART (Univerzális Szinkron-Aszinkron Vevő-Adó) áramköre. Az átvitel kisebb (1-2 m) távolságokra TTL szintű jelekkel is lehetséges. Ha az információt mindkét irányba át akarjuk vinni, akkor három vezetéket használunk (*2. ábra*). A nyilak az információáramlás irányát jelölik. A föld (GND) vezeték a készülékek jelszintjének közös rögzítését szolgálja. Látható, hogy két személyi számítógép közötti adatcsere ilyen vonalhármason megoldható, ahol az ADÓ jelű kivezetés egy kimeneti kapu egy bitje, a VEVŐ kivezetés pedig egy bemeneti kapu bitje lehet.

AZ RS232C ÁTVITELI SZABVÁNYRÓL

Nagyon sok személyi számítógépnek van RS232C szabványú soros kimenete. Ennek sikeres felhasználása megkívánja, hogy röviden megismerkedjünk a szabvány lényegével. Az előzőekben leírtakban a soros adatátvitel logikai kialakításával foglalkoztunk, de nem írtunk arról, hogy az átvitelhez ténylegesen milyen fizikai összeköttetés szükséges, milyen jel hordozza az információt stb. Bár történt arra utalás, hogy kisebb távolságokra az átvitel közvetlenül, TTL szintek segítségével megoldható, de nagyobb távolságokra ez a megoldás — a zavarérzékenysége miatt — nem alkalmazható. A fejlődés során kialakított rendszerek nagy részénél szabványossá vált megoldást valósítottak meg.

Mivel a legtöbb soros adatátviteli alkalmazásnál a tényleges átvitel a telefonvonalakra kapcsolódó, a bináris 0,1 állapotot két különböző frekvenciájú jellé alakító, ún. modemeken keresztül valósult meg, ezért az ezeket a modemeket vezérlő, illetve állapotukat jelző jelek is a szabvány részeivé váltak.

A szabvány olyan soros, bináris adatátvitelt támogat, ahol az adatátviteli sebesség 0 és 20 000 bit/s között változik. Az átviteli távolság maximális értéke mintegy 25 méter. A szabvány jeleket, szabványos jelszinteket és szabványos csatlakozót(!) is meghatároz.

A szabványos csatlakozó DB—25 típusú, 25 pólusú. Kivezetési pontjainak elrendezése és jelölése a *3. ábrán* látható. A legegyszerűbb összeköttetés kialakításához mindössze három pont, a 2. láb (TD), a 3. láb (RD) és a 7. láb (GND) három vezetékkel való összekötése szükséges. A jelszintek: a szabvány a logikai 0 állapothoz a vezeték +3...+15 V-os feszültségét, a logikai 1 állapothoz a −3...−15 V-os feszültségét rendeli hozzá. Ezért a TTL jelszinteket át kell alakítani ilyen jelszintekké. Más kialakítások egyéb jelek alkalmazását is igénylik (vételkészség, adáskészség, ezek ellenőrzése), ezért ilyenkor a *4. ábrán* látható összeköttetéseket kell megvalósítani.

Konkrét megoldásként nézzünk egy programmal megvalósított soros TTL szintű adatátvitelt. A Primo személyi számítógépnek nincs soros adatátvitelt támogató szoftvere, ezért a *listán* közölt, Z80 assembler programmal egy bájt kivételét és behozatalát valósíthatjuk meg. Az adatátviteli sebesség 1200 baud, formája: 1 START, 8 ADAT, 2 STOP bit. Az adatátvitel a botkormány (B) csatlakozóján valósítható meg, ahol a következő pontokat használjuk:

Tuchel csatlakozópont	Funkció	Megjegyzés
2	GND	—
3	Adatkimenet	00H című kimeneti port 6. bitje
1	Adatbemenet	40H című bemeneti port 0. bitje

Az RSBE rutin egy bájtot olvas az A regiszterbe, az RSKI rutin egy bájtot küld ki sorosan. Mindkét rutin programhúrokkal időzít, a DEL2 rutin fél bitidőnyi késlekedést valósít meg.

Utasítás	T ciklusok száma
CALL	17
PUSH	11
DJNZ	988=76×13 (B<>0) (4CH=76)
DJNZ	8 (B=0)
POP	10
RET	10
Összesen:	1044

Mivel 2,5 MHz-es órajelnél egy ciklus ideje 400 ns, ezért ez 1044×0,4=417,6 μs. Mivel egy bit átvitelének ideje 1/1200=833 μs, valóban jó az időzítés beállítása. A szubrutin csupán egy bájt ki/bevitelét valósítja meg. Több bájtból álló blokkok mozgatásához már összetettebb átviteli programra van szükség. A legegyszerűbb esetben meg kell adni a forrásblokk memóriabeli kezdőcímét, az átviendő bájtok számát és a célhely memória kezdőcímét. Ez utóbbi két adatot is célszerű az ADÓ oldaláról átvinni.

Befejezésül az *5.* és *6. ábrán* a kétirányú, soros, TTL szintű összeköttetés megvalósítását mutatjuk be C64-Primo, illetve Primo-Primo számítógépek között. A C64 soros vonala BASIC-ből programozható.

Kónya László

BÖRZE

TUTTI

ELECTROCOOP KISSZÖVETKEZET

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

ECOSOFT
Számítástechnikai Szolgáltató Kisszövetkezet

IBM PC/AT
kompatibilis számítógép
ár: 229 000 Ft-tól
TURBO/32
32 bit, 24 MHz
ár: 479 000 Ft-tól.
Az általunk forgalmazott eszközök lízingelhetők is.
Tel.: 863-677

COBRA

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

KISSZÖVETKEZETEK!

Egyedülálló kínálatunk:

Számlakészítő program	19 900,— Ft
Számlanyilvántartó program	24 900,— Ft
Bér- és jövedelem-számfejtő program	24 900,— Ft
Főkönyvi könyvelőprogram	44 900,— Ft
	114 600,— Ft
COBRA—CONTO programcsomag	99 000,— Ft
IBM—XT kompatibilis számítógép	169 000,— Ft
STAR LC—10 nyomtató	49 000,— Ft
	317 000,— Ft

helyett mindezt már
299 000,— Ft-ért is megvásárolhatja!

MIKROPO BUDAPEST
IRODA:
VI., Nagymező u. 51.
TEL.: 325-768

VÁLASZTÉKUNKBÓL

— adatátviteli rendszerek
MIKROMOD, MODEM-CSALÁD
— CAD-rendszerek
nagy teljesítményű perifériák
magyar VERSACAD
— szuper mini számítógépek

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
— **CRT TERMINÁLOK**
VT—52, QUT—102, Siemens 8160
— **BILLENTYŰZETEK**
— **MONITOROK**
— **INTEGRÁLT VÁLLALATI INFORMÁCIÓS RENDSZER**

UNIX környezetben üzemeltethető.

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 90 E Ft + áfa

pc®

ez a védjegy a megbízható

procontrol

termékeket jelöli

— Profi XT, AT, 386 gépek
— PLATON folyamatirányítók
— PORTAPRINT blokkolóórák
— T80 biztonsági rendszerek
— BCR vonalkód eszközök

PROCONTROL KISSZÖVETKEZET
6725 SZEGED,
VERESÁCS U. 28/B
TEL.: 62/21-165, 28-985,
TELEX: 82-726

ERIKA

a legújabb hordozható irodai elektronikus írógép.
CSÖNDES, PONTOS, MEGBÍZHATÓ,
ára: 25 200 Ft,
áfa-val együtt.
VÁSÁROLJA MEG nálunk:
Bp. VI., Népköztársaság útja 2.

ECONORG SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

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

Kínálatunkból:

a 80 386-os mikroproceszorra alapozott, multiuser üzemmódú

ACER SYS 32

rendszer, amely széleskörűen, változtatható kiépítésben, alap és alkalmazói szoftverekkel együtt kapható.

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

IBM PS/2 utánzatok

Nemsokára már két éve lesz, hogy az IBM elkezdte gyártani a PS/2 sorozatot, s bár az XT-k, AT-k nem szorultak még ki az amerikai piacról sem — sőt, az utánzatokból még mindig sokkal többet adnak el, mint az IBM új sorozatából —, lassan a piac kezd átalakulni. A korszerűbb technológiával gyártott, teljesítőképesebb gépek, ha a kellően naggyá növelt gyártókapacitás és az árpolitika ezt lehetővé teszi, akár még olcsóbbak is lehetnek, mint az eddigi utánzatok.

Sorozatunk egy régebbi részében, az 1988/6. számban ismertettük az IBM új sorozatának a korábbitól eltérő sajátságait. Ugyanitt írtunk az utánozhatóság egyik problémájáról, amely az új, nagyobb felbontású grafika használatával kapcsolatos. Beszámoltunk az első, az IBM-től független megoldásról is. Ezt a Western Digital cég leányvállalata, a Paradise kezdte gyártani. A cég PVGA1 nevű IC-je nemcsak kompatibilis az eredetivel, de még nagyobb felbontást is lehetővé tesz. Ezen az alkatrészen alapuló kártyákat is gyártott ugyanez a cég.

Az anyavállalat továbbment, kidolgozta a PS/2 sorozat Model 30-cal kompatibilis FE2011 típusú IC-t, amely billentyűzet, egér és tárkezelő; a WD 57C65 lemezmeghajtót, amely a Model 50 és 60 kiszolgálására is alkalmas, és létrehozta a WD 16C552-t, amely aszinkron kommunikációs IC mindegyikhez. A Model 50/60-hoz az előbb említetteken kívül az FE5000 perifériakezelő, az FE5010 és 5030 tárkezelő, valamint az FE5020 adat/címpuffer készült el. Végül akkor beszámoltam a WD 1006V—MC1 és WD 1007V—MC1 típusú winchester-hajtó kártyákról, amelyeket a Model 50/60/80-hoz csináltak. Összegezve: akkor egyes részmegoldások már készen voltak, de teljes gép nem, sőt még az alapkártya sem.

Időközben három új, nagy jelentőségű termékcsoport, egy teljesebb IC-készlet, teljes kártyák, valamint teljes gép jelent meg.

A Chips and Technologies Inc. IC-sorozata

A cég néhány évvel ezelőtt először az XT, majd az AT típusokkal kompatibilis IC-k gyártásával indult. Ezek a kártyák legtöbb IC-jét helyettesítő, tömör, CHIPS feliratú IC-k ma már a legtöbb kártyán megtalálhatók.

Az alkatrészek kompatibilitását az eredeti BIOS, az önellenőrző rutin, az IBM PS/2 Tutorial Program és az OS/2 futtatásával ellenőrizték az új, Model 80/50/30 kompatibilis alkatrészeknél.

A Model 80 kompatibilis CHIPS/280 hét alkatrészből áll *(1. kép)*: a 82C325 rendszerlogikából, a 82C607 multifunkciós kontroller analóg adatszeparátorból, a 82C451 VGA IC-ből, a 82C322 memóriakontrollerből, a 82C321 mikroprocesszor(busz)periféria kontrollerből, a 82C223 DMA kontrollerből és a 82C226 rendszerperiféria kontrollerből. Az IC-készletet használva 59 alkatrészből állítható elő egy teljes gép.

A CHIPS/250 a Model 50/60-hoz készített készlet *(2. kép)*, hasonló feladatokat lát el.

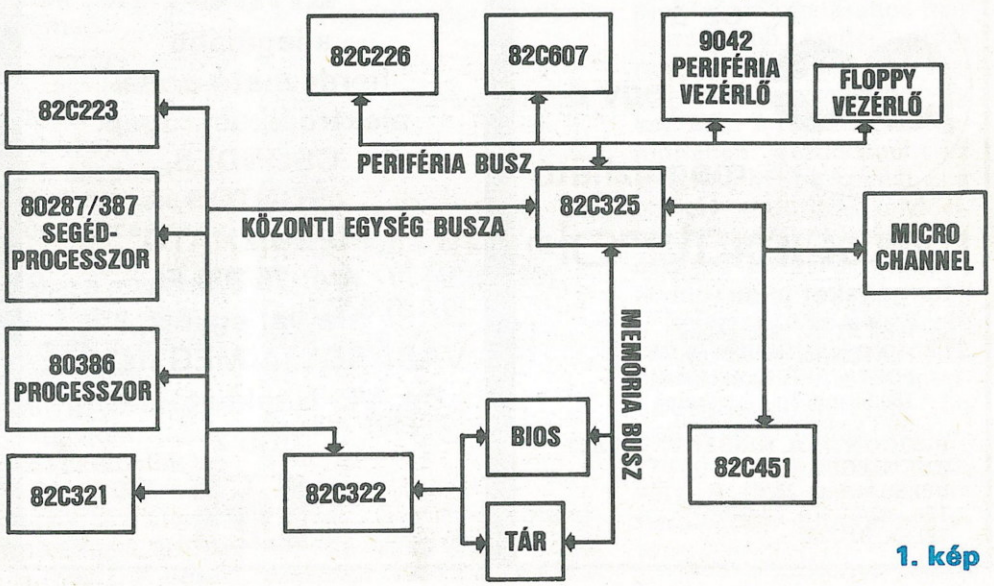
Végül a Model 25/30-hoz mikroprocesszort is készítettek 82C100 típusszámmal. Ez használható 8, illetve 16 bites processzorként egyaránt. A 82C101 típusú 8 bites mikroprocesszort használva Super XT készíthető.

Az első kettőhöz készült a CHIPS/450 VGA IC-készlet *(3. kép)*, amely a 82C451 VGA kontrollerből és a 82C452 szuper VGA kontrollerből áll. Ez utóbbi a normál VGA üzemmódokon felül a 640×480 pontos 256 színű, a 900×700 pontos 16 színű és az 1280×960 pontos monokrom üzemmódot is vezérli, grafikus kurzorral és a grafikus szöveg gyors beírásmódjával kiegészítve.

A készletekkel megvalósított alapkártya *(4. kép)* a Model 80-nál 113 alkatrésszel kevesebbet tartalmaz, mint az eredeti IBM-kártya. Ugyanez az adat a Model 50-nél 53. A nagy alkatrészszám-csökkenés alapján Gordon A. Campbell, a cég elnöke azt várja, hogy a 32 bites gépek számára új piac nyílik, az ún. desk-top kategóriában. Ugyancsak jelentős előnye a készleteknek, hogy a sebességük jóval nagyobb az eredeti konstrukcióénál. A tárelérés ideje mintegy egyharmaddal, a perifériaműveletek mintegy felével csökkentek.

Western Digital kártya

Az IBM régi sorozata hagyományos alkatrészválasztékra épült, amiért is utánzása nem jelentett különösebben nehéz problémát. Mindazok a cégek, amelyek megvásárolták a ROM BIOS szoftverének forgalmazási jogát, gyárthatták az egész gépet. Az új sorozatnál a helyzet alapvetően más. A gépek jó néhány speciális, erre a célra tervezett, nagy integráltságú alkatrészt alkalmaznak, és felületszerelt technológiával készülnek. Ezen okok miatt az utánzás csak magas szintű és nagy mennyiségű mérnöki munkával érhető el, és természetesen igen fejlett gyártástechnológiával. Ez nem a csekély beruházással



1. kép

megvalósítható utánzó cégek paradicsoma. A kompatibilitás biztosítása ezért eddig csak keveseknek és kivált azoknak sikerült, akiknek nagyszerű fejlesztőgárdájuk van.

Az első teljes alapkártyát — ahogy az első IC-ket is — a Western Digital cég gyártja. A kártya teljesen kompatibilis *(5. kép)*. A WD30-WDM nevű kártyán 80C86 mikroprocesszor található. A kártya lelke a már említett FE2011 IC. Ez kompatibilis a 8237A DMA kontrollerrel, a megszakításvezérlővel (beleértve a Model 30 minden kiterjesztését), a 8253 időzítővel és a 8255 perifériakezelővel. Ezenkívül tartalmaz buszvezérlőt, DRAM-kezelőt, cím/adat puffert is. Ráadásul billentyűzet- és egérvezérlést, szoftverrel megadható órafrekvenciát, perifériadekódolást valósít meg. Megfelel a LIM (Lotus—IBM-Microsoft) Extended Memory V. 4.0 előírásainak. Így 640 k tárat használhat, 64, 256, 1024 kbites DRAM IC-ket fogadva.

A kártyán a felületszerelt technológiával készült WD 37C65 típusszámmal olyan IC található, amelyik egyaránt kezeli a 3,5 és az 5,25" méretű, kis és nagy sűrűségű hajlékonylemezeket.

A WDXT—GEN merevlemezegység-vezérlő mind az MFM (hagyományos), mind a megnövelt kapacitású RLL ST506 műveletekre alkalmas. A kártyát IBM 44 és WD 40 pólusú csatlakozóval is ellátták. A videofunkciókat a már ugyancsak tárgyalt PVGA1 IC valósítja meg.

Az egyetlen vásárolt alkatrész — pontosabban annak tártartalma — a korábbi IBM-utánzatokhoz a BIOS szoftvert készítő Phoenix Technologies terméke. Ez a cég készítette el ehhez is a BIOS szoftvert, és annak használati jogát eladta a Western Digitalnak. A kártya egyik érdekessége, hogy a Tandon cég által gyártott intelligens merevlemezegységgel külön vezérlő IC használata nélkül képes dolgozni.

A Western Digital cég olyan nagy mennyiségű kártya gyártására számít, hogy megvásárolta az egyik legnagyobb alapkártyagyártót, a Faraday céget.

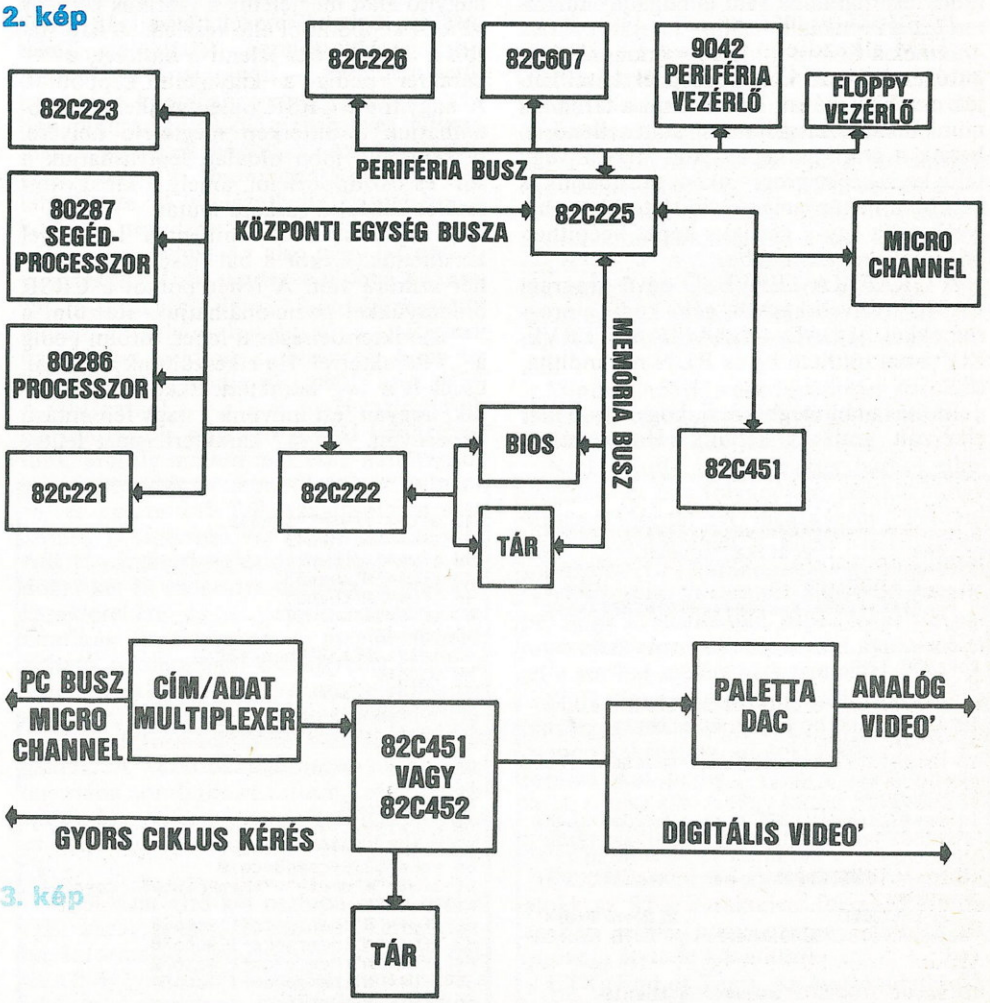
A Western Digital tájékoztatója szerint árpolitikájukat úgy alakították ki, hogy áraik — bár a kártya 25 százalékkal gyorsabb, mint az IBM-é — alacsonyabbak legyenek az IBM árainál. Remélik, hogy hamarosan sikerül egy olyan hajlékonylemezes, monokrom terméket előállítaniuk, melynek ára a hazai számítógépekével megegyezik.

A Compatible Computer Co. számítógépe

A vállalat az említett kártyákra épülő számítógép gyártását kezdte meg. A torony formájú gép intelligens billentyűzetet és beépített monitort tartalmaz. A gépről jelenleg azon kívül, hogy működésében megfelel az IBM Model 30-nak, más információt nem közöltek.

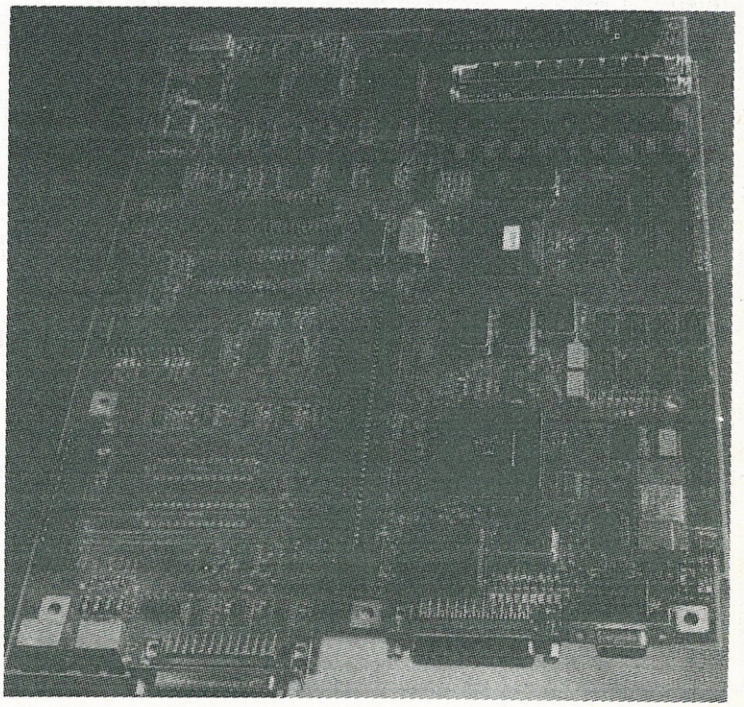
Simonyi Endre

2. kép

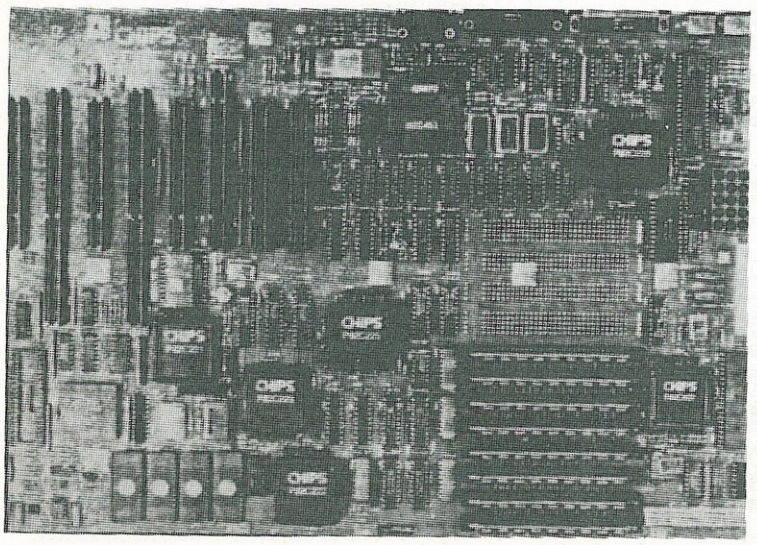


3. kép

5. kép



4. kép



COMMODORE 64

BITTÉRKÉP JÁTSZI KÖNNYEDSÉGGEL

Bármilyen nagy felbontású grafikus bittérkép elkészítése igen fáradságos munka, s a szemet sem kíméli. Ráadásul a grafikus rendszerprogramok sem támogatják túlzottan ezt a munkát.

Ennek a C64-re íródott programnak a segítségével játszi könnyedséggel készíthetjük el a bittérképet, közvetlenül a tárban, s munkánkat szempillantás alatt ellenőrizhetjük a grafikus képernyőn. A kész vagy félig kész képet programként tárolhatjuk, s később a nekünk megfelelő tárterületre helyezhetjük. Így a grafikus képet beépíthetjük saját programunkba.

A GRAFIKA TERVEZŐ nevű program BASIC nyelven készült, gépi kódú alprogramokkal. LOAD "GRAFIKA TERVEZŐ",8-cal tölthető be és RUN-nal indítható.

Indítás után megkérdezi, hogy van-e már elkészült grafikus képünk. Ha a válasz "N", akkor előkészíti a memóriaterületet, törli a grafikus képernyőt, és kívánságunk szerint beállítja a színeket. Ezután mintegy nagyító alatt megjelenik a grafikus kép egy 32 × 24 képpontból álló részlete. A nagyításon a "." karakter jelenti a hátteret, a "*" karakter pedig a kigyújtott képpontot. A nagyítót a CRSR billentyűkkel pozicionálhatjuk a bittérkép megfelelő helyére. A képernyő jobb oldalán leolvashatjuk a sor- és oszloppozícióit, amely a kinagyított terület bal felső sarkára mutat.

A rajzolást a "*" billentyű leütésével kezdhetjük. Ekkor a bal felső képpont fehér színűre vált. A fehér pontot a CRSR billentyűkkel pozicionálhatjuk. Rajzolni a "*" karakter beírásával lehet, törölni pedig a "." karakterrel. Ha elkészültünk a rajzzal, üssük le a "←" karaktert. Ekkor megláthatjuk, hogyan fest művünk a nagy felbontású képernyőn. A "≠" karaktert ismét leütve visszatérhetünk a rajzoláshoz. Ha lemezre akarjuk menteni rajzunkat, akkor a grafikus kép alatt a "@" karaktert üssük le, természetesen előzőleg készítsünk egy formázott lemezt a meghajtóba.

Ha az így kimentett képen tovább akarunk dolgozni, akkor a program első kérdésére "I"-vel válaszoljunk. Ekkor a program bekéri a kimentett képfájl nevét, ami nálunk "+kep+" lesz. Ezt a nevet a program automatikusan adja az első kimentéskor. Ha valamilyen okból megállítottuk a program futását vagy nem sikerült a képet kimenteni, például elfelejtettünk lemezt tenni a meghajtóba, a STOP/RESTORE leütése után indítsuk el újra a programot RUN-nal, de amikor a képfájl nevét kéri, csak a RETURN-t üssük le. Ekkor tovább dolgozhatunk a tárban lévő képen.

A program egyébként kazettáról is működik, de ahhoz írjuk be a következő programsort:

```
18 poke 49500,1:poke 49505,5:
   poke 49532,1:poke 49507,117
```

A grafikus kép kazettára mentésénél a "@" karakter leütése után indítsuk a magnót felvételre, anélkül, hogy a PRESS RECORD AND PLAY felirat látszódna.

Radnóti Tibor

```
1 rem************************************
2 rem*          Grafika tervezo         *
3 rem*                                  *
4 rem* Radnoti Tibor              1988 *
5 rem************************************
6 :
7 Poke53272,23:Poke53280,.:Poke53281,12:
Poke650,128
8 goto16
9 Poke211,co:Poke214,cs:sys58640:return
10 inPut"■Ird be a keP-file nevet";q$
11 if q$=""then28
12 Poke49539,len(q$):forx=1tolen(q$):Pok
ex+49556,asc(mid$(q$,x,1)):next:sys49531
13 goto28
14 n=a*320+b*8+16384:hi=int(n/256):lo=(n
-hi*256):Poke251,lo:Poke252,hi:sys49152
15 goto39
16 Print"■■■■■■■■Van mar elkeszult graf
ikus keP ?"
17 Print"■■■                    ■I■gen/■N■em"
:gosub82:ifs<>52481thenPrint"DATA hiba":
end
18 :
19 getw$:ifw$<>"i"andw$<>"n"then19
20 ifw$="i"then10
21 Print"■Allitsd be a GRAFIKUS kePernyo
szineit !"
22 inPut"■■       PaPir (hatter)(0-15):"
;P$:P=val(P$):ifP>15thenP=0
23 Poke49475,P
24 inPut"■       Keret (0-15):          ";
P$:ke=val(P$):ifke>15thenke=0
25 inPut"■       Tinta (rajz)(0-15):  ";
P$:r=val(P$):ifr=Pthenr=P+1
26 ifr>15thenr=1
27 Poke49475,Peek(49475)orr*16:sys49451
28 Print"■"spc(32)"■■0-------";
29 PrintspC(32)"1-OszloP";:PrintspC(32)"
2-■-----■-"spc(32)"3-■-----■-";
30 PrintspC(32)"4-■-----■-"spc(32)"5----
---"spc(32)"6--Sor--";
31 PrintspC(32)"7-■-----■-"spc(32)"■0■-■
------■-"spc(32)"■1■-■-----■-";
32 PrintspC(32)"■2■-------"spc(32)"■3■--
------"spc(32)"■4■-Rajz-■*■";
33 PrintspC(32)"■5■-------"spc(32)"■6■-K
esz-■+■"spc(32)"■7■-------";
34 PrintspC(32)"■■0-------";
35 PrintspC(32)"1-Mentes";:PrintspC(32)"
2-------"spc(32)"3---■■■---";
36 PrintspC(32)"4-------"spC(32)"5 Mozga
-"spC(32)"6--tas:-";
37 PrintspC(32)"7-------0123456■0123456
7■01234567■01234567■---■■CRSR■";
38 Poke2023,160:Poke56295,.:goto14
39 cs=3:co=35:gosub9
40 PrintspC(3-len(str$(b)))b:cs=8:gosub9
:PrintspC(3-len(str$(a)))a
41 Poke198,.
```

```
42 getw$:ifw$=""then42
43 ifw$="■"then49
44 ifw$="■"then51
45 ifw$="■"then53
46 ifw$="■"then55
47 ifw$="*"then57
48 goto42
49 a=a+1:ifa>22thena=22
50 goto14
51 a=a-1:ifa<0thena=.
52 goto14
53 b=b+1:ifb>36thenb=36
54 goto14
55 b=b-1:ifb<0thenb=.
56 goto14
57 co=0:cs=0
58 Poke198,.:gosub9:Poke55296+cs*40+co,1
59 getw$:ifw$=""then59
60 Poke55296+cs*40+co,0
61 ifw$="*"orw$="."thenPrintw$;:co=co+1:
P=1:goto72
62 ifw$="■"thencs=cs+1:goto68
63 ifw$="■"thencs=cs-1:goto70
64 ifw$="■"thenco=co+1:goto72
65 ifw$="■"thenco=co-1:goto74
66 ifw$="←"then76
67 goto58
68 ifcs>23thencs=23
69 goto58
70 ifcs<0thencs=0
71 goto58
72 ifco>31thenco=0:cs=cs+1:goto68
73 goto58
74 ifco<0thenco=31:cs=cs-1:goto70
75 goto58
76 sys49326:Poke53280,ke
77 Poke56576,Peek(56576)and254:Poke53272
,0+8*16:Poke53265,59
78 getw$:ifw$=""then78
79 ifw$="@"thensys49491
80 Poke56576,151:Poke53272,23:Poke
53265,27:Poke53280,.
81 goto14
82 fori=49152to49561
83 readx:Pokei,x:s=s+x:next:return
84 data32,126,192,72,72,160,0,169,128,
 162,0,72,49,251,208,5,169,46
85 data76,23,192,169,42,157,0,4,104,74,
 232,224,8,208,234,173,24,192
86 data24,105,40,144,3,238,25,192,141,
 24,192,200,192,8,208,211,206,25
87 data192,173,24,192,56,233,56,176,3,
 206,25,192,141,24,192,32,137,192
88 data104,24,105,1,201,4,208,180,32,
 149,192,238,25,192,173,24,192,24
89 data105,32,144,3,238,25,192,141,24,
 192,104,24,105,1,201,3,240,6
90 data72,169,0,76,4,192,32,163,192,141,
 25,192,169,0,141,24,192,96
91 data165,251,133,253,165,252,133,254,
 169,0,96,165,251,24,105,8,144,2
```

```
92 data230,252,133,251,96,230,252,165,
 251,24,105,32,144,2,230,252,133,251
93 data96,165,253,133,251,165,254,133,
 252,169,4,96,32,126,192,72,72,160
94 data0,162,0,177,251,10,145,251,189,
 0,4,201,42,208,5,177,251,9
95 data1,145,251,232,224,8,208,233,173,
 189,192,24,105,40,144,3,238,190
96 data192,141,189,192,200,192,8,208,
 212,206,190,192,173,189,192,56,233,56
97 data176,3,206,190,192,141,189,192,
 32,137,192,104,24,105,1,201,4,208
98 data181,32,149,192,238,190,192,173,
 189,192,24,105,32,144,3,238,190,192
99 data141,189,192,104,24,105,1,201,3,
 240,6,72,169,0,76,178,192,32
100 data163,192,141,190,192,169,0,141,
 189,192,96,169,0,160,0,162,0,157
101 data0,64,232,224,0,208,248,238,51,
 193,200,56,192,32,144,239,9,112
102 data192,36,208,233,169,64,141,51,
 193,169,0,141,50,193,96,169,0,133
103 data251,169,64,133,252,162,8,32,
 186,255,169,7,162,115,160,193,32,189
104 data255,169,251,162,233,160,99,32,
 216,255,96,64,58,43,75,69,80,43
105 data0,162,8,160,0,32,186,255,169,5,
 162,149,160,193,32,189,255,169
106 data0,162,0,160,64,32,213,255,96,
 43,75,69,80,43
```

A Programban szereplo specialis karakterek:

"■" = SHIFT/CLR
"■" = CTRL/9
"■" = CTRL/0
"■" = CTRL/7
"■" = CTRL/1
"■" = CRSR←
"■" = SHIFT/CRSR←
"■" = CRSR↑
"■" = SHIFT/CRSR↑

Figyelem!
A 28-37 sorokkal rajzoljuk meg a kepernyot. A szokoz karaktereket a konnyebb olvashatosag kedveert a '-' karakter helyettesiti.

A KIVESÉZETT KÓDKIRÁLY

Karakterek ábrázolása

A számítógép és a külvilág vagy egy másik számítógép közötti kapcsolatteremtéskor az információkat kódolva visszük át. A bitcsoportoknak — amelyek elvileg tetszőleges számú bitből állhatnak — jelentést tulajdonítunk (kódolást végzünk). Az információ átvitelekor ezeket a bitcsoportokat továbbítjuk, és a vevő oldalon a jelentésüknek megfelelően értelmezzük, azaz dekódoljuk azokat. Természetesen a bitcsoportokban lévő bitek száma és a bitcsoportokhoz rendelt jelentés sokféle lehet. A karakterek ábrázolásánál azonban — a számítástechnika és informatika fejlődése során — csaknem kizárólag az ASCII kódrendszer vált egyeduralkodóvá, és gyakorlatilag a személyi számítógépekben szinte kizárólagosan ezt használják.

Az ASCII kódrendszer

Az ASCII rövidítés, az American Standard Code for Information Interchange — amerikai szabványos kód az információ kölcsönös cseréjére — kifejezés rövidítése. Az ilyen módon kódolt bitcsoportokat ASCII karaktereknek nevezzük. Az ASCII karakterkészlet 128 hétbites, különböző kódot tartalmaz, amelyek mindegyike egy egyedi karaktert reprezentál. Természetesen felmerülhet a kérdés, hogy miért hét-, és nem nyolcbites kódot választottak, hiszen ekkor 256 különféle kód volna lehetséges, ami a bájtos tárolási módhoz is illeszkedne. Az ASCII kód ANSI X3.4—1977-es szabványának függelékében szerepel az a megállapítás, hogy minimum hét bit a legtöbb felhasználásban elegendő. Ez érthető is, mert ha az angol ábécét tekintjük, annak 26 kis-, 26 nagybetűje, az írásjelek (vessző, kérdőjel stb.), valamint a 10 szám együttesen már 64 különféle karaktert jelent, aminek kódolásához már hat bit szükséges. Ezek alapján belátható, hogy a hétbites kód a legtöbb gyakori írásszimbólumot képes reprezentálni, és bővíthető, mert olyankor, amikor a hét bit és az ezzel biztosított 128-fajta kód nem elegendő, a nyolcadik bit felhasználásával a kódolható szimbólumok száma 256-ra nőhet.

Szövegeket tartalmazó szimbólumtáblákban a táblabeli szövegek végét olyképpen is jelölhetik, hogy a befejező karakter nyolcadik bitjét 1-be állítják. Ezt a bővíthetőséget használják ki például a magyar nyelvben lévő ékezetes kis- és nagybetűk kódolására. A nyolcadik biten egyet tartalmazó kódokat használva és az ASCII kód alapértelmezését változatlanul hagyva hozták létre a magyar ábécének megfelelő, bővített szabványos ASCII kódrendszert. Nyolc bitre kiegészített karakterkészletet használnak az IBM PC személyi számítógépben (World Trade Character Set), amelynek alsó 128 eleme az ASCII kódrendszerrel egyezik meg, a felső 128 elemhez rendelt karakterek felölelik a spanyol, német, svéd, görög stb. ábécé speciális karaktereit. Ezenkívül speciális grafikus karaktereket is tartalmaz, amelyekkel könnyű keretezéseket, táblázatokat kialakítani. Természetesen olyan nyelvnél — kínai, japán —, ahol nem betűírás van, a kódolás igen nagy gond, és csak bonyolult megoldások segítségével oldható meg a szövegek kódolása. Az ASCII karaktereket az *1. ábrán* látható táblázat foglalja össze. A táblázatbeli felépítéshez a kód alsó négy bitjét és a felső három bitjét különválasztottuk. Az alsó négy bit növekvő értékek szerint rendezve alkotja a táblázat 16 sorát, és a felső három bit hasonló módon a nyolc oszlopát alkotja a táblázatnak. Tehát például kiolvasható, hogy az A betű ASCII kódja 01000001, azaz 41H. A táblázatból a 0., 1., 6. és 7. oszlopot elhagyva, az ún. szűkített ASCII kódhoz jutunk, ami ily módon már csak hatbites, de a legfontosabb 64 szimbólumot tartalmazza, és egyszerűbb felhasználásokban sokszor ez is elegendő. Az ANSI szabvány az ASCII karakterkészlet definiálásakor a kódokat két fő csoportra osztotta: a grafikus karakterekére és a vezérlőkarakterekére. Grafikus karaktereken a megjeleníthető, látható, nyomtatható karaktereket értjük, a vezérlőkarakterek pedig a megjelenítés vezérlésére, formájának kialakítására, valamint az információcsere vezérlésére szolgálnak. A vezérlőkaraktereket három kategóriába soroljuk: az információcserét vezérlőkébe, a formátumot befolyásolókéba és abba, amelyek az információt elkülönítik.

A táblázat első két oszlopa és az utolsó DEL karakter tartozik ezekbe a kategóriákba. Információcsere-vezérlő karakterre példa a ØAH kódú EOT karakter, amit annak a jelzésére használnak, hogy a karakterek átvitele befejeződött, és ez a kód jelöli, hogy nincs több átviendő karakter. Formátumbefolyásoló karakterekkel lehet a karaktersorozat megjelenési formáját befolyásolni. Például az LF (ØAH) Line Feed (soremelés) karakter hatására a karakterek megjelenítése az adott pozícióban, de új sorban folytatódik. Például az A,B,C,D,LF,E,F karaktersorozat az

```
ABCD
    EF
```

formában jelenik meg.

Az információt elkülönítő karakterek arra alkalmasak, hogy az információkat logikailag elkülönítsék. Ekképpen különböző hosszúságú karaktersorozatok — rekordok — vihetők át. Ha például három különböző hosszúságú rekordot akarunk átvinni, akkor a rekordokat a Record Separator (RS) (1EH) karakterrel lehet egymástól elválasztani. A vezérlőkarakterek némelyike a fentiek egyikébe sem sorolható be, ezeket általános vezérlőkaraktereknek nevezzük. A *2. ábra* foglalja össze az eddig elmondottakat.

A vezérlőkarakterek jelentése

Az ANSI szabvány minden ASCII karaktert részletesen meghatároz. A vezérlőkarakterek értelmezése és jelentése nem közismert, ezért a következőkben ezeket mutatjuk be, hogy jobban megértsük egy adott készüléknél, berendezésnél, mi a felhasználásuk célja és értelme.

Ahogy ezt már az elején is említettük, az ASCII kódokat információátvitelnél használják. Ezért a vezérlőkarakterek nagy része az átvitel miként jének, protokolljának kialakítására szolgál. Mivel ennek részletezése meghaladja e cikk kereteit, ezért a vezérlőkarakterek funkcióinak ismertetésekor csak nagyon röviden utalunk a velük kapcsolatos átviteli definíciókra. Néhány adatátvitellel kapcsolatos bevezető ismeretet azonban röviden összefoglalunk.

Az adatátvitelkor az ADÓ valamilyen összeköttetésben van a VEVŐ-vel, és a kódolt információt átadja. Ez az információátvitel általában ASCII kódú karakterek segítségével történik, és a vezérlőkarakterek szolgálnak az adatátvitel tényleges végrehajtásának megvalósítására. Az adatátvitelnek a szabályait hívjuk átviteli protokollnak.

Minden ASCII vezérlőkarakter speciális vezérlési feladatot lát el. A következőkben röviden ismertetjük jelentésüket.

NUL (Null). Ez a karakter bárhol elhelyezhető az adatfolyamban, anélkül, hogy információtartalmát megzavarná. Például lassú nyomtatóknál a kocsi vissza (CR), soremelés (LF) karaktereket egy vagy több NUL karakter követhet, aminek az a szerepe, hogy a mechanika képes legyen a parancsokat végrehajtani, azaz a nyomtatófejet a sor bal szélére visszavinni.

SOH (Start of Heading). Adatátvitelnél a tényleges adatokat blokkonként, karaktercsoportonként, az adatokra vonatkozó információkat pedig — hány adat, milyen fajta, típusa stb. — egy külön blokkban viszik át. Ennek a speciális blokknak a kezdetét jelöli a SOH karakter.

STX (Start of Text). Az előbbi speciális blokk az STX karakterrel fejeződik be, és egyben jelöli, hogy ezután adatblokkok (szöveg) átvitele következik.

ETX (End of Text). Az utolsó adatblokk befejezését jelöli (szöveg vége).

EOT (End of Transmission). Ezt a karaktert szokás használni a teljes átvitel befejezésére. Maga az átvitel több speciális blokkból és az azokat követő adatblokkokból állhat.

Hogy jobban megértsük ezeknek a vezérlőkaraktereknek a jelentését, tegyük fel, hogy egy terminálra — aminek címe mondjuk legyen 16 — ki akarjuk vinni a "STOP" üzenetet. Az üzenet négy karakter hosszúságú. A szabványos átvitel szerinti karaktersorozat:

SOH,1,6,STX,S,T,O,P,ETX,4,EOT

Természetesen a konkrét megvalósításokban még más specifikus részek is lehetnek az üzenetben, de ezek sokrétűsége miatt azok külön vizsgálatát javasoljuk.

ETB (End of Transmission Block). Ez a karakter használható egy-egy adatblokk átvitelekor a végső lezáró karakterként.

ENQ (Enquiry). Az adatátviteli rendszerekben, ha választ várunk egy távolabbi állomástól, ezt a karaktert küldjük ki (ki vagy?), hogy az beküldje az azonosítóját és az állapotára (státusára) vonatkozó információt.

A KIVESÉZETT KÓDKIRÁLY

ACK (Acknowledge). Ezt a jelet a vevő küldi ki, hogy választ kapjon a küldőtől.

NAK (Negative Acknowledge). A vevő küldi ki az adónak, ha valamilyen okból nem képes az adóval együttműködni, mert foglalt.

Az ENQ, ACK és NAK általában a tényleges átviteli protokoll kialakítására szolgál.

BEL (Bell). Vezérlőkarakter a figyelem felhívására; ha a vevő ezt veszi, általában hallható — csengő — hangjelzést ad.

BS (Backspace). A formátumot befolyásolja. Kiküldésével a vevőben ugyanabban a sorban egy pozícióval való visszalépés és esetleg az utolsó karakter törlése hajtódik végre.

HT (Horizontal Tabulation). Szintén formátumvezérlő. A vevő a jel hatására az aktuális karakterpozícióból a következő, előre meghatározott tabulátorpozícióba lép. Általában sorok vízszintes osztására használható. Például ha egy nyomtatón a sor elejétől számítva minden nyolcadik hely a tabulátorpozíció (ez a szokásos), akkor a HT, A, HT, B karaktersorozat hatására a nyomtatási kép az alábbi lesz:

```
A       B
12345678 ← pozíciók
HT nélkül:  AB
```

LF (Line Feed). Soremelés az aktuális pozícióban, de a következő sorban folytatódik a nyomtatás. Sokszor e karakter vétele nem a fenti hatást, hanem újsor-parancsot jelent, azaz ekkor a nyomtatás az új sor első pozíciójában folytatódik, tehát hatása a CR,LF vagy LF,CR vételekor bekövetkező hatással egyezik meg. Természetesen ilyenkor ezt az adó és a vevő protokolljában figyelembe kell venni.

VT (Vertical Tabulation). Formátumvezérlő, ugyanabban a pozícióban, de előre meghatározott sor átlépése után folytatódik a nyomtatás.

FF (Form Feed). Formátumvezérlő, ugyanabban a pozícióban, de a szövegformátum — például lap — következő oldalának előre meghatározott sorában folytatódik a nyomtatás. Megállapodás kérdése: itt is lehetséges az első karakterpozícióra lépés (a gyakorlatban egyszerűen a következő lap elejére lép).

CR (Carriage Return). Kocsi vissza, ugyanannak a sornak az első pozíciójába lép.

SO (Shift Out). Az SI karakterrel együtt a grafikus karakterkészlet kiterjesztésére szolgál. Ekkor az SI karakter vételéig az ASCII grafikus karaktereknek más — például grafikai szimbólumok — jelentése van.

SI (Shift In). Vétele után visszaáll az eredeti állapot- és karakterértelmezés.

DLE (Data Link Escape). Átvitelvezérlő karakter, ami az ezt követő korlátozott számú karakter jelentését megváltoztatja. Kizárólag további adatátviteli vezérlő funkciók biztosítására való.

DC1, DC2, DC3, DC4 (Device Controls). Vezérlőkarakterek. A DC1 és DC3 karakterek szokásos használata a különböző átviteli sebességű adók és vevők közötti adatátvitel vezérlése az ún. XON/XOFF protokoll szerint. Ezt részletesebben később, a soros adatátvitelről szóló részben tárgyaljuk.

SYN (Synchronous Idle). Az átvitelvezérlő karaktert a soros szinkron adatátviteli rendszerekben használják.

CAN (Cancel). Vétele azt jelzi a vevőnek, hogy a küldött adatban hiba van vagy az adatot törölni kell. Pontos jelentését adott esetben külön kell definiálni.

EM (End of Medium). Vezérlőkarakter, ami az adatokat tartalmazó adathordozó fizikai végét, befejeződését jelzi.

SUB (Substitute). Vezérlőkarakter, amit hibás vagy érvénytelen karakter helyettesítésére használnak.

ESC (Escape). Vezérlőkarakter, a kódrendszer kiterjesztésére. A karakter egy jelölőkarakter, ami az utána következő véges számú bitalakzat speciális értelmezését jelzi. Szokásos megoldás nyomtatóknál a nyomtató paramétereinek beállításához, felhasználó által definiált karakternek a nyomtató elektronikába való betöltéséhez, az ún. „escape szekvencia" használata (például TMS 120 nyomtató).

FS, GS, RS, US (File-, Group-, Record-, Unit-Separator). Elválasztók, amelyek fájl-, csoport-, rekordstruktúrájú adatok elválasztására használhatók. FS a legmagasabb „rendű" elválasztó — azaz a struktúra legmagasabb szintjén álló egységek szétválasztására szolgál —, az US pedig a legalacsonyabb.

DEL (Delete). Karakter, ami az utolsónak bevitt karaktert helyettesíti, felülírja, gyakorlatilag törli. Mivel nem nyomtatható és egyéb jelentése nincs, ezért adatátvitelkor helyet és időt kitöltő karakternek használható.

Ahogy a felsorolásból is látható, az ASCII vezérlőkarakterek alapvetően a karakterorientált átviteli protokoll kialakításának támogatását végzik. Ilyen szempontból az ASCII karakterkódkészletet adatátviteli, kommunikációs kódnak is nevezhetjük. Még három, a gyakorlat miatt fontos megjegyzést kell tenni az ASCII kódkészlettel kapcsolatban.

A billentyűzet kialakítása

Mivel a billentyűzetről jövő információk — billentyűlenyomások — általában ASCII karakterként értelmeződnek, de ugyanakkor a billentyűzet általában kevesebb gombot tartalmaz, mint a minden billentyűnyomás külön-külön, egyértelmű azonosításához szükséges 128 gomb, ezért a billentyűzeteken általában két speciális gomb: a SHIFT (sift) és a CTRL (kontroll) nevű billentyű található. Az ASCII kódtáblázat ötödik és hetedik, illetve hatodik és nyolcadik oszlopának betűkarakterei csak egy bitben, az ötödikben különböznek. Ezért a billentyűzeten nincs külön kis- és nagybetű, hanem alaphelyzetben a billentyűhöz rendelt kód jelenik meg a kimeneten, ha pedig közben a SHIFT billentyűt is lenyomjuk, akkor annak a „siftelt" változata. A billentyűkkel való takarékosság és a hagyományos írógép-billentyűzethez való hasonlóság miatt a negyedik oszlop „siftelt" verziója a harmadik oszlop. A CTRL billentyű segítségével a nem látható vezérlőkarakterek billentyűzetről való előállítására van lehetőség.

Az ötödik és a hatodik oszlophoz rendelt billentyűk valamelyikének és a CTRL billentyűinek együttes megnyomásakor az első vagy második oszlop ugyanazon sorához rendelt vezérlőkarakterek generálódnak. Például a CTRL és a D együttes megnyomásakor a billentyűzet kimenetén az ETX (Ø4H) ASCII kód jelenik meg. További kódkiterjesztésre szolgál az ALT jelű billentyű. A fentiekben leírt, billentyűzetre vonatkozó konvenciók általában igazak, de adott esetben mindig meg kell győződni a billentyűkhöz rendelt kódokat tartalmazó táblázat alapján, hogy ezeket a konvenciókat a tervező hogyan valósította meg; például milyen kód generálódik a CTRL, a SHIFT és valamilyen betűbillentyű együttes megnyomásakor. Hasonló módon, a kiküldött karaktereket megjelenítő kijelzők vagy a nyomtatók helyes használatához is ismerni kell a vezérlőkarakterek szerepét és értelmezését, amelyek leírását a készülékek gépkönyvei tartalmazzák.

A karakterek sorrendje

Számítástechnikai alkalmazásokban nagy jelentősége van szövegek sorba rendezésénél, kezelésénél a karakterek sorrendjé-

Az ASCII kódkészlet

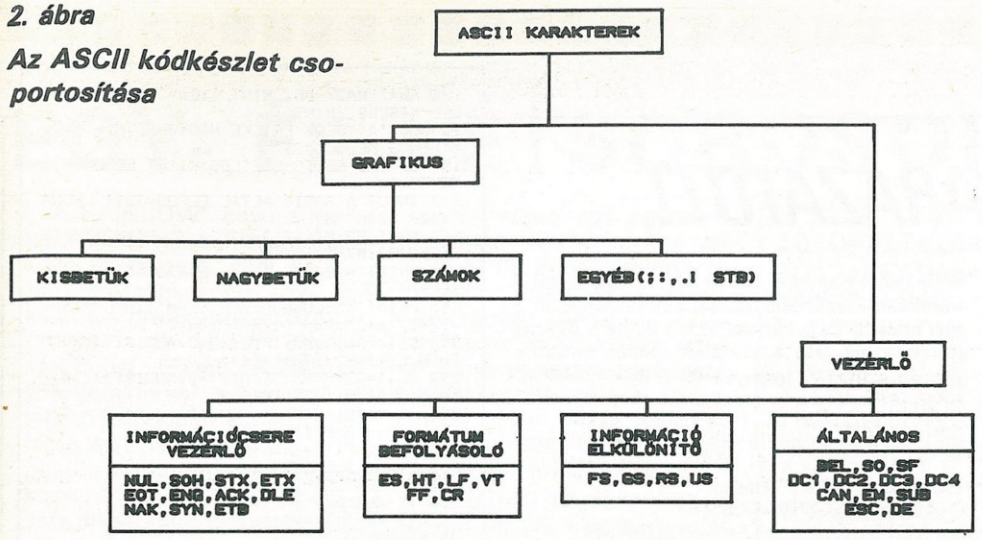
1. ábra

TABLE 4-1 ASCII character matrix

Least significant bits (3, 2, 1, 0)	Most significant bits (6, 5, 4) 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P		p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N		n	
1111	SI	US	/	?	O	—	o	DEL

2. ábra

Az ASCII kódkészlet csoportosítása



nek, aminek helyes megoldása alapvető feltétel volt a kódkészlet kialakításánál. A sorrendiséget természetesen a karakterekhez tartozó bináris szám nagysága határozza meg, mivel a használt algoritmusok is alapvetően numerikus műveleteket — összehasonlítás, összeadás, kivonás stb. — használnak. Így a sorban az első a NUL (ØØH), az utolsó a DEL (7HF) karakter. A betűkódok értéke az alfabetikus sorrend szerint változik, de a betűközt (SP) kell az első betűkarakternek tekinteni, hogy helyes sorrend alakuljon ki olyan vezetéknevek rendezésekor is, mint például JUHOS és JUHOSI. Természetesen a nemzeti ábécé betűivel kiterjesztett karakterkészlet használatakor a sorrendiség szempontjára is figyelemmel kell lenni.

Kritikai észrevétel

Az ASCII kódrendszert még abban az időszakban alakították ki, amikor az adatmegjelenítő perifériák szinte kizárólag mechanikus működésűek voltak (teletype, telex, géptávíró). Ezért a kódrendszerből hiányoznak azok a vezérlőkódok, amelyek a már csaknem kizárólagosan használt képernyő-orientált rendszerekben használatosak: a kurzormozgató, a képernyőtörlő stb. funkciókhoz rendelt kódok. Mivel ezekről a szabvány nem rendelkezik, ezért bizonyos inkompatibilitás van az egyes megjelenítők között, aszerint, hogy melyik vezérlőkaraktereket rendelték hozzá az adott funkcióhoz.

A szabványos, magyar betűket tartalmazó kódrendszer

A számítástechnika egyik legfontosabb gyakorlati alkalmazása a szövegszerkesztés és szövegfeldolgozás. Mivel a számítástechnikai eszközök a szabványos hétbites ASCII kódrendszert használják — ami az angol ábécé 26 kis- és nagybetűjét tartalmazza —, gondot okoz a magyar nyelv speciális ékezetes betűinek hiánya. Az ilyen ún. speciális vagy nemzeti karakterek szükségessége miatt a berendezések tervezői lehetőséget teremtettek a kódrendszer kiterjesztésére, azaz speciális karakterek bevitelére és megjelenítésére.

Ezt általában úgy tették, hogy az ASCII kódrendszert nyolcbites kóddá terjesztették ki. Ilyenkor a 8ØH-FFH tartomány használható a speciális jelek kódolására. Magyarországon az MSZ 7795-ös szabvány határozza meg a magyar ábécé ékezetes betűihez rendelt kódokat, de sajnos, látva és ismerve a számítógépekhez elterjedt különféle ékezetes betűs kódhozzárendeléseket, valószínűnek látszik, hogy ezeket „hasra-csapásos" alapon csinálták. Ez a sokféleség természetesen azzal a hátránnyal jár, hogy például az otthon, személyi számítógépen megírt szöveget egy másik rendszerbe csak kódkonverzióval — ami legtöbbször a két rendszerben eltérő módon kódolt ékezetes magyar betűk konverzióját jelenti — lehet átvinni. Az egységes kódolás elterjesztése érdekében összefoglaljuk a fenti szabvány magyar nyelvű kiegészítő karakterkészletére vonatkozó előírásait.

A szabvány meghatározza a magyar ékezetes betűkhöz és néhány további, a szövegekben felhasznált karakterhez rendelt kódokat. A következőkben ezeket nem táblázatos formában, hanem szabványos megnevezésükkel, a kódok növekvő sorrendjében ismertetjük.

NBSP: Rögzített szóköz (No-Break SPace). Kód: AØH.

Képi megjelenítésében megegyezik a szokványos szóközével (üres karakterpozíció), de a szövegben azt a helyet jelzi, ahol nem lehet sorvégi elválasztást alkalmazni a szöveg kialakításakor. Ha például nem akarjuk az SN 74LSØØ kifejezés esetleges sorvégi kettéválasztását, akkor a kifejezést a következő módon kódoljuk: SN<NBSP>74SLØØ.

§ Paragrafusjel Kód: A7H

SHY Elhagyható kötőjel (Soft-HYphen) Kód: ADH

Képi megjelenésében hasonló a szokásos kötőjelhez, de csak sorvégi elválasztáskor jelenik meg. Történetesen hosszabb szavaknál alkalmazva, ezzel a kóddal adjuk meg azokat a helyeket, ahol a szó a sor végén esetleg elválasztható.

Megjegyzések:

— Bár a felsorolásból nem nyilvánvaló, de megjegyezzük, hogy a kis- és nagybetűk — hasonlóan az ASCII angol betűkhöz — a kódtáblázat azonos sorában helyezkednek el, mégpedig úgy, hogy csupán egy bitben különböznek.

— Az eredetileg német nemzeti karaktereknek, az „umlaut"-nak a karakterek közötti szerepeltetését az indokolja, hogy egyrészt a magyar írógépeken szerepel, másrészt a német nyelvű idegen kifejezések leírását megkönnyíti.

K. L.

°	Fokjel — felül kör	Kód: BØH
Á	Nagy A betű, hosszú ékezettel	Kód: C1H
Ä	Nagy A betű, kettős rövid ékezettel	Kód: C4H
É	Nagy E betű, hosszú ékezettel	Kód: C9H
Í	Nagy I betű, hosszú ékezettel	Kód: CDH
Ó	Nagy O betű, hosszú ékezettel	Kód: D3H
Ő	Nagy O betű, kettős hosszú ékezettel	Kód: D5H
Ö	Nagy O betű, kettős rövid ékezettel	Kód: D6H
×	Szorzókereszt	Kód: D7H
Ú	Nagy U betű, hosszú ékezettel	Kód: DAH
Ű	Nagy U betű, kettős hosszú ékezettel	Kód: DBH
Ü	Nagy U betű, kettős rövid ékezettel	Kód: DCH
á	Kis A betű, hosszú ékezettel	Kód: E1H
ä	Kis A betű, kettős rövid ékezettel	Kód: E4H
é	Kis E betű, hosszú ékezettel	Kód: E9H
í	Kis I betű, hosszú ékezettel	Kód: EDH
ó	Kis O betű, hosszú ékezettel	Kód: F3H
ő	Kis O betű, kettős hosszú ékezettel	Kód: F5H
ö	Kis O betű, kettős rövid ékezettel	Kód: F6H
:	Osztásjel	Kód: D7H
ú	Kis U betű, hosszú ékezettel	Kód: FAH
ű	Kis U betű, kettős hosszú ékezettel	Kód: FBH
ü	Kis U betű, kettős rövid ékezettel	Kód: FCH

ADOM A MAGYARÁZATOT!

Előző számunkban egy egyenletrendszert megoldó programmal kapcsolatos problémával foglalkoztam. Ígértem, hogy az ezt megvalósító programot ismertetem most.

Foglaljuk össze a programkészítéshez a problémát!

1. A program túlcsordulással leáll, ha
— valamelyik (DX, DY, illetve DX, DY, DZ) számláló,
— a nevező (D), vagy
— valamelyik (X, Y, illetve X, Y, Z) tört értéke meghaladja M2-t, a BASIC-fordító számára létező legnagyobb számot. Ezek közül legutóbb csak az első csoporttal foglalkoztam, mert a többi hatása azonos.

2. A program nem ad megoldást — bár van —, ha a nevezőre igaz

$$0 < ABS(D) < M1 \quad (1)$$

ahol M1 a BASIC-fordító számára létező legkisebb, nullától különböző szám.

3. A program esetleg hibás eredményt ad, ha valamelyik számláló vagy a nevező abszolút értéke kisebb, mint M1. Ezzel a hibatípussal nem foglalkoztam, mert ez csak az eredmények pontosságát befolyásolja.

Ad 1. A vizsgálat nem kezdődhet az ABS (V) > M2 (ahol V a vizsgálandó számláló, illetve nevező vagy tört) feltétel ellenőrzésével, mert a feltétel teljesülése túlcsordulást okoz. Végezzünk el egy olyan előzetes leosztást, ami a túlcsordulást kizárja.

Egyetlen beadott szám sem érheti el M2 értékét, és az egyes számlálók, illetve a nevező kétváltozós esetben két, egyenként kéttényezős szorzat összege. Így értéke legfeljebb $2M2^2$; háromváltozós esetben hat, egyenként háromtényezős szorzat összege, melynek értéke legfeljebb $6M2^3$. Ezekkel végezve a leosztást, a feltétel

$$ABS(V)/(2 \times M2\uparrow 2) > 1/(2 \times M2) \quad (2)$$

illetve

$$ABS(V)/(6 \times M2\uparrow 3) > 1/(6 \times M2\uparrow 2) \quad (3)$$

alakú lesz. A leosztást úgy végezzük, hogy az egyenletrendszer minden együtthatóját M2 értékkel osztjuk, majd az egyes szorzatok előállításánál kettővel — kétváltozós —, illetve hattal — háromváltozós — ismét osztunk. Amennyiben az együtthatók leosztás előtti értéke nem volt nulla, de utána igen, úgy leosztás utáni értéküket M1-re (illetve „– M1"-re) kell módosítani. Ennek oka itt is a téves érték kiküszöbölése. Amennyiben a túlcsordulási tartományba esik az érték, úgy azt fel kell jegyeznünk, mert ha a számláló túlcsordulna és a nevező nem nagyobb leosztás nélkül mint egy, vagyis leosztással

$$ABS(N)/(2 \times M2\uparrow 2) <= 1/(2 \times M2 < 2) \quad (4)$$

illetve

$$ABS(N)/(6 \times M2\uparrow 3) <= 1/(6 \times M2 < 3) \quad (5)$$

akkor a tört értéke nagyobb, mint M2. Ha a nevező ennél nagyobb, akkor a leosztott számlálóból és nevezőből osztással kell kiszámítani a tört értékét. Ez a helyzet akkor is, ha a nevező túlcsordulna. Ha a számláló nem csordulna túl, de a nevező igen, akkor a tört értéke kisebb, mint 1/M2.

A tört értékének vizsgálatára akkor van szükség, ha a számláló nem csordulna túl, mert ha a nevező elég kicsi, akkor a tört értéke túlcsordulhat. Ennek feltétele:

$$ABS(S)/ABS(N) = (ABS(S/2 \times M2\uparrow 2))/(ABS(N/(2 \times M2\uparrow 2))7 > M2$$

ahol S számláló, N nevező, illetve

$$ABS(N/(2 \times M2\uparrow 2)) < ABS(S/(2 \times M2\uparrow 2))/M2 \quad (6)$$

Ilyen túlcsordulás esetén tehát a tört értéke nagyobb, mint M2.

Összefoglalva teendőinket: le kell osztatnunk M2-vel az egyenlet minden együtthatóját, és ha ettől valamelyik nullává válik, M1-re kell módosítani. A leosztott számlálókat, illetve nevezőt úgy kell előállítani, hogy kétváltozós esetben minden tényezőt osztunk kettővel, háromváltozós esetben hattal. Meg kell vizsgálnunk ezekre, hogy (2) és (3) teljesül-e? Ettől függően vagy kiíratjuk: „a tört értéke nagyobb, mint M2", ha a számláló túlcsordulna és (4) vagy (5) teljesül; „a tört értéke kisebb, mint 1/M2", ha csak a nevező csordulna túl; vagy megvizsgáljuk (6) teljesülését, és ha teljesül, kiíratjuk: „a tört értéke nagyobb,mint M2"; végül, ha ez sem teljesül, jön a következő pont.

Ad 2. Az előző pont alapján itt már csak az lehet, hogy a számláló nem csordulna túl. Vizsgálandó

$$M1 < ABS(D/M1) < 1 \quad (7)$$

Az újabb túlcsordulásveszély elkerülésére megvizsgálandó, hogy a számláló osztása nem okoz-e túlcsordulást, azaz fennáll-e ABS(S/M1) > M2

illetve

$$ABS(S/M2) < M1 \quad (8)$$

Ha nem, megvizsgálandó, hogy teljesül-e

$$ABS(S/M1) >= 1 \text{ illetve}$$

$$ABS(S/M2) >= M1/M2 \quad (9)$$

Ha (7) és (8) vagy (7) és (9) igen, akkor ki kell íratni: „a tört értéke nagyobb, mint M2". Ha (7) nem teljesül úgy, hogy a leosztott nevező is zérus, akkor kiírandó: „nincs megoldás, az eddig esetleg kiírt eredmények tévesek", és nem szabad folytatni a számítást. Amennyiben a leosztott tört értéke 1 vagy nagyobb, jön a következő pont.

Ad 3. Itt már csak az okoz problémát, ha (9) nem teljesül, de (7) igen, azaz mind a számláló, mind a nevező kisebb, mint M1. Ilyenkor mind a számlálót, mind a nevezőt szorozni kell 1/M1-gyel, és az eredményeket osztani egymással, és ennek eredményét kell kiíratni.

Végül, ha (7) nem teljesül, de (9) igen, akkor osztható a számláló a nevezővel, és ennek eredményét kell kiíratni. Amint látjuk, mindez igen bonyolult.

```
155 Aø=" NAGYOBB, MINT ":Bø="A TOERT ERT
EKE KISEBB, MINT"
156 Cø="A TOERT ERTEKE NAGYOBB, MINT":Dø
="NINCS MEGOLDAS."
157 Eø="AZ EDDIG ESETLEG KIIRT EREDMENY
TEVES."
162 INPUT"A BASIC ALTAL ELFOGADOTT LEGNA
GYOBB SZAM";M2
163 M3=1/M2:M4=1/(2*M2ü2):M5=M4*M2:M6=1/
(6*M2ü3):M7=1/6
164 PRINT"A BASIC ALTAL ELFOGADOTT LEGKI
SEBB 0-TOL "
165 PRINT"KUELOENBOEZOE SZAM":INPUTM1:M8
=1/M1:Xø="":Yø="":Zø="":Dø=""
252 C(I)=A(I)*M3:IF(C(I)=0)AND(A(I)<>0)
THEN C(I)=M1*SIGN(A(I))
292 D(I)=B(I)*M3:IF(D(I)=0)AND(B(I)<>0)
THEN D(I)=M1*SIGN(B(I))
301 D1=.5*D(1)*C(4)-.5*C(2)*D(2):D2=.5*C
(1)*D(2)-.5*D(1)*C(3)
302 D4=.5*C(1)*C(4)-.5*C(2)*C(3):IF ABS(
D1)>M4*M2 THEN Xø="I"
303 IF ABS(D2)>M4*M2 THEN Yø="I"
304 IF ABS(D4)<=M4 THEN Dø="I"
305 IF(Xø="I")AND(Dø="I") THEN PRINT"X";
A ø;M2:GOTO332
306 IF Dø="I" THEN PRINTBø;M3:GOTO332
307 IF ABS(D4)<ABS(D1)*M3 THEN PRINTCø;M
2:GOTO332
311 IF ABS(D*M8)>=1 THEN 317
312 IF (ABS(D*M8)<M1)AND(D4=0) THEN PRIN
TDø:PRINTEø:GOTO580
313 IF ABS(D1)>M1 THEN PRINTCø;M2:GOTO33
2
314 IF ABS(D1)>=M1*M3 THEN PRINTCø;M2:GO
TO332
315 PRINT"X=";D1/D4:GOTO332
317 IF(ABS(D*M8)<=M1)AND(ABS(D1)>=M1*M3)
 THEN315
331 PRINT"X=";DX/D
332 IF(Yø="I")AND(Dø="I") THEN PRINT"Y";
Aø;M2:GOTO580
333 IF Dø="I" THEN PRINTBø;M3:GOTO580
334 IF ABS(D4)<ABS(D2)*M3 THEN PRINTCø;M
2:GOTO580
341 IF ABS(D*M8)>=1 THEN347
342 IF(ABS(D*M8)<M1)AND(D4=0) THEN PRINT
Dø:PRINTEø:GOTO580
343 IF ABS(D2)>M1 THEN PRINTCø;M2:GOTO058
0
344 IF ABS(D2)>=M1*M3 THEN PRINTCø;M2:GO
TO580
345 PRINT"Y=";D2/D4:GOTO580
347 IF(ABS(D*M8)<=M1)AND(ABS(D2)>=M1*M3)
 THEN345
350 PRINT"Y=";DY/D:GOTO580
```

Nézzük most már a programot! A 155—157-es sorokban bevezetünk néhány segédváltozót. Megjegyzem, hogy a programnak csak a kétváltozós részét ismertetem, mert a másik rész ugyanígy átírható. Az eredeti program néhány — egyébként nélkülözhető — grafikus karaktertől eltekintve a legtöbb BASIC-változatban futtatható, ezért M1 és M2 értékének megadását bízzuk a felhasználóra. Így adjuk be ezeket a program elején (162—165-ös sorok).

A leosztásokat célszerű rögtön az adatbeadásnál végrehajtani (252 és 292-es sor). A számításokat a vizsgálatokkal kell kiegészíteni (301—370-es sorok). A 360-as sort törölni kell.

Vajon most már jó-e a program? Nem! Jobb, mint volt, de még mindig hibás a számítási módszer elve. Ezzel a következő számban foglalkozom.

S. E.

HCC Klub

6 80(×)0 közleményei

Az általunk épített gép (PT 68K) jellemzői

Hardver

Az alapkártya 8 MHz-es 68 000 CPU-t tartalmaz, de 10, 12 és 16 MHz-es is alkalmazható, sőt 68 010-es CPU is használható.

Minimális kiépítettség: alapkártya, 68 000 CPU EPROM, statikus RAM, óra/oszcillátor, 2 db soros port, tápfeszültség-ellátás. Így egysoros terminállal már működőképes.

Az alapkártya teljes kiépítettség esetén a következőket tartalmazza: 68 000 CPU, 1 Mbájt dinamikus RAM, 4 kbájt elemmel ellátott statikus RAM, 32 kbájt EPROM (Basic, monitor, gépi kódú hibakereső, illesztés az SK + DOS-hoz, 4 db soros port, 2 db párhuzamos port, floppy disk vezérlő max 4 db floppyhoz, hanginterfész és hangszóró, óra/naptár IC, csatlakozó memóriabővítéshez max 11 Mbájt-ig, IBM PC billentyűzet illesztő, 6 db IBM XT bővítőhely (max 2 db összesen 128 Mbájt Winchesterhez és minden IBM XT kártyához).

Felhasznált főbb IC: MC68 000 P8CPU, MC68 681 DUART, MC68230P8 periféria illesztő/időzítő, WD1772 floppy disk vezérlő, 1488, 1489 RS232C adó/vevő önálló működéshez, kiegészítések: IBM XT doboz + min 135 W tápegység, IBM kompatibilis billentyűzet, Hercules grafikus kártya, Monitor, floppy meghajtó szoftver.

SK DOS: tartalmaz segédprogramokat (editor, assembler más Basic verzió, IBM lemez írás/ol-

EGY- ÉS KÉTSOROSOK

A RAINBOW[R] nevű amerikai szaklap minden számában közli az olvasók által beküldött legérdekesebb egy- és kétsoros programokat. Ezek a programozók valódi ügyességpróbái, hiszen legfeljebb 255 karakternyi helyet kell a lehető legjobban kihasználniuk (a lap a Tandy cég CoCo típusú számítógépeivel foglalkozik, és az ezekre írt Microsoft BASIC változat ennyi karaktert fogad el egy sorban). Az már szinte ráadás, hogy ezek a programok gyakran még jól használhatók is.

Az *1. listán* látható program átírásához szükséges tudnivalók: a CLS képernyőtörlés; az 1. sorban az „A" utáni jel ennél a nyomtatónál a $ helyett van; a PRINT# –D a nyomtatóutasítás; az INPUT utasítás felesleges, ha folyamatosan akarjuk az egészet kiíratni.

A program az 1989. évi naptárt készíti el az *1. táblázatban* látható formában. Az eredeti 1987-re készült, és természetesen angol szavak rövidítéseivel. Az angol és amerikai naptárban nem a hétfő, hanem a vasárnap az első nap, ezért a táblázatban a vasárnapot az első helyről az utolsóra kellett áthelyezni, míg a többit egy hellyel előbbre kellett tolni.

2. táblázat

```
FHZ ÉT LÉUTPSPTPL

JLD ÉX PÉYXTWTXTP

OQI ÉC UÉDCYBYCYU

YAS ÉM EÉNMILIMIE

DFX ÉR JÉSRNQNRNJ

EGY ÉS KÉTSOROSOK
```

A *2. listával* egy egyszerű titkosírást készítő programot mutatok be. Ennél a magyarra fordításon kívül csak annyit változtattam, hogy a GOTO utasítások beírásával elértem azt, hogy a program várja az újabb titkosítandó szöveg beadását.

A program úgy titkosít, hogy a beadott számmal hátrább helyezi az ábécében a szöveg betűit, illetve, ha így túljutna az utolsó betűn is, akkor elölről kezdi. Csak az angol ábécé betűivel teszi ezt, ezért az ékezetes betűk változatlanok maradnak. A *2. táblázatban* a címet titkosítottam úgy, hogy 1, 5, 10, 20, 25 helyyel toltam el. Végül megadtam az eredeti szöveget is. Az eltolás itt 0 volt, ami persze nem titkosítás.

Simonyi Zsuzsa

```
10 D=2:T=24:CLS:DIMA¤(12),A(12):FOR B=1
TO12:READA¤(B),A(B):NEXTB:FORC=1TO12:PRI
NT#-D,A¤(C);TAB(22)"1989":PRINT#-D:PRINT
#-D," H  K  SZ  CS  P  SZ  V ":FORB=1
 TO A(C):IF T>24 THEN T=0:PRINT#-D
20 PRINT#-D,TAB(T) RIGHT¤(STR¤(B),2);:T=
T+4:NEXT B:PRINT#-D:PRINT#-D:INPUTU:NEXT
 C:DATA JANUAR,31,FEBRUAR,28,MARCIUS,31,
APRILIS,30,MAJUS,31,JUNIUS,30,JULIUS,31,
AUGUSZTUS,31,SZEPTEMBER,30,OKTOBER,31,NO
VEMBER,30,DECEMBER,31
```

1. lista

2. lista

```
10 PRINT#-2:INPUT"SZOEVEG";M¤:INPUT"(1-2
5)";S:PRINT#-2:FORN=1TOLEN(M¤):P¤=MID¤(M
¤,N,1):IF ASC(P¤)<65 OR ASC(P¤)>90 THENP
RINT#-2,P¤;:NEXTN:GOTO10 ELSE K=ASC(P¤)+
S:IF K>90 THEN K=K-26:PRINT#-2,CHR¤(K);:
NEXTN:GOTO10 ELSE PRINT#-2,CHR¤(K);:NEXT
N:GOTO10
```

1. táblázat

JANUAR 1989

H	K	SZ	CS	P	SZ	V
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

APRILIS 1989

H	K	SZ	CS	P	SZ	V
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JULIUS 1989

H	K	SZ	CS	P	SZ	V
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

OKTOBER 1989

H	K	SZ	CS	P	SZ	V
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

FEBRUAR 1989

H	K	SZ	CS	P	SZ	V
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

MAJUS 1989

H	K	SZ	CS	P	SZ	V
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUGUSZTUS 1989

H	K	SZ	CS	P	SZ	V
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

NOVEMBER 1989

H	K	SZ	CS	P	SZ	V
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MARCIUS 1989

H	K	SZ	CS	P	SZ	V
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

JUNIUS 1989

H	K	SZ	CS	P	SZ	V
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

SZEPTEMBER 1989

H	K	SZ	CS	P	SZ	V
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

DECEMBER 1989

H	K	SZ	CS	P	SZ	V
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

vasás, RAM disk megvalósítás, 6809-s emulátor, pluszként eszközmeghajtókat). Minden szabványos 3,5" és 5,25" lemezformátumot ismer. Maximum 10 eszközt képes kezelni, közte 2 db max 128 Mbájt Winchestert.

Egyéb: teljes 'C' compiler, üzleti programok. Részletes leírás a gépről és összeszereléséről a Radio-Electronics 1987/88-as számaiban található.

Mosonyi Balázs

Közöljünk-e sorellenőrző számokat vagy sem?

Rendszeresen kapunk olyan olvasói leveleket, melyekben vagy mint saját ötletüket, vagy mint mástól átveendő megoldást ajánlják, hogy programlistáink minden egyes sora után közöljünk valamilyen ellenőrző számot. Mi pedig ezt eddig nem tettük! Miért?

Előrebocsátom, hogy nem azért (amit jó néhány olvasónk feltételezett), mert nem ismertünk ilyeneket. A nyugati szaksajtó gyakorlatilag azóta ismerteti (és egyes lapokban használja) ezeket, amióta a lapok listákat közölnek. Tájékozottabb olvasóink megfigyelhették, hogy rendszerint az egygépes (ritkábban a néhány gépes) lapokban van csak ilyen. Ezekben a lapokban, minden egyes lapszámban megtalálható az ellenőrző számot előállító program.

Mi nem vagyunk sem egygépes, sem néhány gépes lap. Nézzük meg, hogy például a cikk írásakor megjelent 88/10. számban hányféle gépre írt program listáját közöltük.

Menjünk sorban: HT 1080Z, Commodore Plus/4, ZX-Spectrum, Commodore 64, Enterprise 128, Commodore VIC20, IBM PC, Tandy CoCo. Összesen nyolc. Tételezzük fel, hogy mindegyik program olyan hosszú lenne, mint az, amit a Mikrovilág c. lapból átvételre ajánl GREGOSITS JÓZSEF olvasónk, vagyis 31 sor, amelynek 47 egyhasábos sor felel meg. Ehhez hozzáadódik a gép megnevezése, a használati utasítás, ez összesen kb. 400 sor, azaz kb. 2 lapoldal. Minden számban! Ezzel azonban még nincs vége, hiszen minden programsor után oda kellene írni az ellenőrző számot, ami mintegy újabb 1 oldallal nyújtaná meg a közölt listákat. Összesen tehát 3 oldallal maradna kevesebb akkor, amikor állandóan visszatérő panasza az olvasóknak, hogy (és ezt minden géptípusra követelik) többet közöljünk. (Azt, hogy miből közöljünk többet, röviden úgy foglalhatom össze, hogy az egyik olvasónak ebből, a másiknak másból.)

Ezenfelül ez természetesen nem ad védelmet sem a program hibái ellen, sem a szerkesztőségben történő újrabegépelés hibái ellen. (Amennyiben ugyanis ragaszkodnának ennek a rendszernek az érvényesítéséhez, úgy a legtöbb beküldött programot elvethetnénk vagy begépelhetnénk, hiszen azt is nagyon nehéz sokszor elérnünk, hogy a beküldött listák megfelelően olvashatóak legyenek.)

Összefoglalva: nem közlünk ilyen ellenőrző számokat, mert kevesebb a hasznuk, mint a káruk.

S. E.

UTASÍTÁS VAGY ADAT?

Neumann Jánostól származik a gondolat, az elv, hogy a számítógépek tárában mind az adatokat, mind az azok feldolgozására vonatkozó információt (a programot) egyaránt elhelyezhetjük. E klasszikus elv szerint a tárban az adatok és az utasítások keveredhetnek egymással, és a gép a programot — az utasításokat — ugyanúgy képes módosítani, akár az adatokat. Ha kézbe vennénk egy, ezt az elvet korlátozás nélkül alkalmazó és kihasználó személy által írt, futásra kész programmal és a hozzá tartozó adatokkal feltöltött tárat, és nem ismernénk a program kezdőcímét, akkor általában csak igen fáradságos nyomozással tudnánk kideríteni azt, hogy a különböző helyeken (címeken) található beírások közül melyik **adat** és melyik **utasítás** — vagy annak része. A feldolgozóegység, a CPU a kezdőcímtől indulva ugyanis mindig az éppen végrehajtott utasítás alapján dönti el, honnan (a tár melyik részéből — címéről) vegye a következő utasítást. Bármit, amit ezen a megelőző utasításból következő címen talál, azt megpróbálja utasításként értelmezni: ha nem tudja, akkor a gép megáll. Ha a gép itt nem utasítást — azaz nem olyan bejegyzést, amelyet annak szántunk —, hanem olyan beírást talál, amit lehet *véletlenül* utasításként *is* értelmezni, akkor még rosszabb dolog történik: a gép ezt az *adatot* szándékunk ellenére *„végrehajtja”*, és ki tudja, mi lesz ennek a következménye?

A csupasz gép (a hardver) az utasításokat az adatoktól tehát úgy különíti el, hogy a program a kezdőcímtől kiindulva sorban **rávezeti magát** az utasításokra. Más szavakkal: a tárolt információt aszerint kezeli a gép utasításként vagy adatként, hogy az előzőleg végrehajtott utasítások alapján a feldolgozóegység a tár valamilyen helyén utasítást vagy adatot keres-e. A szétválasztásra vonatkozó információt tehát a *program hordozza,* és a *kezdőcím* a szétválasztásban igen fontos szerepet tölt be.

VÉGREHAJTÁSI SORREND

Az utasítások végrehajtási sorrendjét általában a tárban való elhelyezkedésük *sorrendje* határozza meg. Ezt a „természetes” beírási (leírási) sorrendet a végrehajtáskor lényegében két dolog változtathatja meg: vagy a programba beleírt ún. *vezérlésátadó* utasítás, vagy egy „kívülről” érkező, ún. *megszakításkérés.* A gép a soron következő utasítást az előbbinél nem a tár soron következő rekeszeiben, hanem a programban, a vezérlésátadó utasításban előírt (más) helyen keresi: ezt a címet például akár maga a vezérlésátadó — ugró — utasítás is tartalmazhatja. Az ugrás, a vezérlésátadás lehet feltételekhez kötött is. A feltételek teljesülését a gép az ilyen utasítás végrehajtásának „pillanatában” vizsgálja. Az, hogy ez a feltétel egy adott feladat végrehajtásakor a program futása közben az adott pillanatban beáll-e, már a feladattól, a futás körülményeitől függ.

A megszakítás (angolul: interrupt) mechanizmusa olyan — ezt úgy tekinti a gép —, mintha a programozó minden egyes utasítás után automatikusan beiktatott volna egy feltételes vezérlésátadó utasítást. A „feltétel” ilyenkor az, hogy érkezett-e a géphez megszakításkérés külső jel formájában. Ha a feltétel teljesül, a gép a következő utasítást a tár egy erre a célra, a hardver tervezője által kijelölt rekeszéből veszi. Ennek a rekesznek a címét a számítógép „tudja”. (Ez a „tudás” be van a gépbe huzalozva.) Ettől a címtől kezdve lehet elhelyezni azt a programot — egy-egy ún. interruptkezelő rutint —, amely „megmondja” a gépnek, ilyenkor, azaz megszakításkéréskor, mit kell csinálnia. Ezt a rutint a programozó készítheti el.

A programozó ún. *feltétel nélküli* vezérlésátadó utasításokkal alapvetően a végrehajtás *leírás szerinti* merev sorrendjét változtathatja meg; más szóval a feladat által megadott műveleti struktúrát képezheti le a tár — a leírás, a lista — lineáris struktúrájára. A feltételes vezérlésátadókkal a feladatmegoldáskor az „időben” — sorrend szerint — előre látható választási lehetőségeket programozhatjuk be, a megszakításokkal az „időben” — a végrehajtási sorrend szerint — nem besorolhatókat.

Ha a számítógépnek nem lenne megszakítási mechanizmusa, akkor a program futásához nem szinkronizált eseményeket csak ún. várakozó ciklusokkal, hurkokkal kombinált feltételes utasításokkal tudnánk kezelni, fogadni. A megszakítási mechanizmus tehát az „időben” — sorrendben — előre pontosan nem kalkulálható, nem is mindig bekövetkező események fogadására szolgál. A programban foglalt feltételes vezérlésátadásokkal az adatokon végrehajtott műveletek eredményeként bekövetkező, determinisztikus eseményeket lehet kezelni — a megszakítási mechanizmussal a külső, a program szempontjából nem determinisztikus eseményeket.

A megszakítási mechanizmus lehetővé teszi a következő kis példával jellemezhető feladatok célszerű megoldását. Egyszerre kell mondjuk takarítani és levest főzni. Ha sípoló kuktafazekam van, akkor nyugodtan takaríthatok mindaddig, amíg nem hallom a sípolást (amíg nem érkezik interruptjel). Amikor meghallom, odamegyek a tűzhelyhez stb. Ha nem lenne ilyen fazekam, „megszakítási mechanizmusom”, akkor takarítás közben állandóan szaladgálhatnék a konyhába megnézni, mi van a levessel.

Mint látjuk, gépi kódban — a csupasz géppel, a hardverrel dolgozva — az utasítások *végrehajtási sorrendje* a program szerint lényegében kötött, és ezen a kötöttségen sem a vezérlésátadó utasítások, sem a megszakítások nem változtatnak. Ez a kötöttség nincs összhangban a legtöbb feladat természetével; részfeladataink sorrendje például gyakran felcserélhető lenne, egyes részfeladatokat pedig akár egymással párhuzamosan is végezhetnénk, illetve végeztethetnénk a géppel.

A tár lineáris struktúráján, az utasítások végrehajtásának kötött sorrendjén kívül az ember és a *„csupasz gép”* közötti súlyos *illesztetlenség* további oka az, hogy az elektronika szempontjából célszerű gépi utasítások nagyon távol esnek azoktól a részműveletektől, amelyekre az ember szívesen bontaná megoldandó feladatait. Amikor gépi vagy ahhoz közeli kódban programozunk — manapság ez igen ritka eset —, akkor kénytelenek vagyunk gondolkodásunkba, munkánkba tipikus feladatainktól idegen elemeket bevinni. Ilyenkor mindig nagy a tévesztés, a hiba veszélye.

A csupasz gép, a hardver igen gyors működésű és univerzális — „mindenre” programozható —, de alapfelépítése nem kedvez a jó feladatstrukturálásnak. Kézenfekvő a gondolat: a gép erőforrásainak — sebességének, idejének stb. — egy részét áldozzuk fel annak érdekében, hogy a gép jobban illeszkedjék hozzánk. Valójában tényleg ez történik. A számítástechnika fejlődése szinte nem más, mint ennek a **„sebesség—kényelem”** típusú csereüzletnek a története. A kezelés, a programozás ilyen áron nyert kényelme nem fényűzés, hanem a tervezési, a működési biztonság és megbízhatóság fontos feltétele. A mikroelektronika, a számítógép-fejlesztés elsősorban azért „hajszolja” a sebességet, hogy legyen elegendő *„cserealap”.*

Már régen ott tartunk, hogy száz végrehajtott gépi utasításból általában kevesebb mint húsz szolgálja közvetlenül a feladat megoldását. A többi nyolcvan az egyébként nagyon is szükséges *rezsi,* amely nélkül nem sokra mennénk gépeinkkel: idejük nagyobb részében tétlenül várakoznának arra, hogy csináljunk végre velük valamit. Az „eladott” sebességért — képletesen szólva — a csupasz gépet rétegesen burkokba csomagolhatjuk, és barátságosabbá tehetjük.

VIRTUÁLIS GÉPEK

Kézenfekvő volt a gondolat: programozzuk gépeinket úgy, hogy kedvezőbb „arcot” mutassanak felénk. Miután az új viselkedést megvalósító programot elkészítettük, a gépbe beírtuk és elindítottuk, már akár el is feledkezhetünk az eredeti, „csupasz” gépről. Most már az egészet, a hardvert és a beírt, futó programot együtt tekinthetjük **„a gépnek”.** Nevezhetjük ezt az **új** gépet ún. *virtuális* — látszólagos — gépnek, mert úgy működik, mintha nem is az eredeti, „csupasz” gép lenne, hanem egy másik, amelynek számunkra — ha ügyesek voltunk — sokkal kedvezőbb tulajdonságai vannak, mint az eredetinek voltak.

Akkor feledkezhetnénk el igazán az eredeti, „csupasz” gépről, ha a virtuális tulajdonságokat megvalósító programok eleve fixen be lennének írva a tárba — nem kellene őket külön betölteni —, és ha a gép bekapcsolásakor ezek automatikusan el is indulnának. (A kisebb gépeknél gyakran így is van.) A program ily módon teljesen **„beburkolja”,** elfödi előlünk a számítógépet. Ha most valamilyen okból mondjuk ez az „első szintű” virtuális gép még mindig nem lenne elég kényelmes a számunkra, akkor semmi akadálya nem lenne annak, hogy az előbbi eljárást megismételjük, a virtuális gépet is újból beburkoljuk stb. Az egyes virtuális

és az Updike-sorozat befejezése, valamint a mesterséges intelligencia körbejárása után most egészen másfelé (ismét „földközelben") kalandoztak gondolataink. Olyasmin töprengtünk például, honnan „tudja" a csupasz számítógép (a hardver), hogy egy adott címen adatot vagy utasítást kell-e találnia, és azt honnan, hogy a tár egy adott helyén lévő bejegyzést hogyan kell értelmeznie, mitől függ az utasítások végrehajtási sorrendje, miért hajszolja az elektronika még mindig a sebességet stb.

szinteket megvalósító programok úgy vehetik körbe az eredeti gépet, mint a hagyma belsejét az egymásra boruló héjak. A magasabb szintű virtuális gépek tervezőinek kiinduló „anyaga" az előző szintű virtuális alap.

Elvileg mindenki, aki képes gépi kódban programozni, megvalósíthatja, megalkothatja a maga számára legmegfelelőbb első szintű virtuális gépet. Csakhogy valóban jó, használható gépet — csupaszt vagy virtuálist — tervezni nagy szaktudást, sok tapasztalatot igénylő, fáradságos munka. Ezért ezt rendszerint szakemberek végzik: a „csupasz gépekből" különféle *virtuális* gépeket előállító programokat erre szakosodott vállalkozásokban ugyanúgy tervezik, „gyártják", forgalmazzák, „követik", javítják, mint a hardvert. Az, hogy felhasználóként ki hány és milyen „burkot" — operációs rendszert, interpretert stb. — vásárol meg csupasz gépéhez, más szóval: ki, milyen virtuális gépeket vásárol, az a megoldandó feladatától, pénzétől, idejétől és szakképzettségétől is függ.

Az eladói polcról levehető félkész/kész megoldások választékának meghatározása komoly tervezői-mérnöki megfontolást igényel. A gépek és a tipikus feladatok gondos elemzésével kell azt is meghatározni, hogy a félkész megoldásokon mit, milyen mértékben lehessen változtatni — mennyire legyenek ezek a konkrét helyzetekhez adaptálhatók (rendszergenerálás). Az adaptálható, hajlékonyabb elemekből álló választék kisebb lehet, mint a merev, nem változtatható elemekből álló. A virtuális gépek tipikus igényeit másik oldalról természetesen figyelembe veszik a számítógép tervezői is, akik a mikroelektronika, a félvezető-technika adta lehetőséget igyekeznek legcélszerűbben kihasználni: igyekeznek minél jobb kiinduló „nyersanyagot" szolgáltatni a virtuális gépek tervezőinek. Emellett a „csupasz" gépek tervezői természetesen továbbadják a követelményeket az *elektronikai alkatrészipar* felé. E folyamat eredményeképpen tegnapi virtuális gépeink egyre több tulajdonsága valósul meg a mai fizikai gépekben, a hardverben. Ez jó, mert ami a számítógépben valósul meg, az mindig gyorsabban működik, mint ami a programban. A sebesség növelésére pedig változatlanul nagy szükségünk van.

MAGASABB SZINTŰ GÉPEK— NYELVEK

A legismertebbek, legelterjedtebbek azok a virtuális gépek, amelyek utasításkészletei és ezek felhasználásának szabályai közelebb állnak az emberhez, mint a gépi utasítások, a gépi nyelv. Azt mondhatjuk, hogy az utasítások és a szabályok együttesen nyelvet alkotnak. Ilyen, ún. magas szintű nyelv például a Pascal, a BASIC, az Ada, a COBOL, a Modula—2 stb.

Tudjuk, a számítógép végső soron mindig *csak* gépi utasításokat hajt végre. A tárba írt, gépinél magasabb szintű program ezért a csupasz gép szintjén nem lehet más, mint *feldolgozásra váró adat* (szöveg). A magasabb szintű gép — nyelv — megvalósításának egyik kézenfekvő módja például az alábbi:

— Összeállítunk egy számunkra a gépinél alkalmasabb utasításkészletet, meghatározzuk az utasítások felhasználásának szabályait, megalkotjuk a nyelvet.

— Írunk egy olyan gépi kódú programot, amely képes az előbbi szabályok szerint megírt szövegeket, azaz az ún. magas szintű programokat mint adatokat fogadni és értelmezni, képes megállapítani, hogy az ilyen szövegek milyen magas szintű utasításokat tartalmaznak.

— Minden egyes magas szintű utasításhoz írunk egy-egy gépi kódú megvalósító rutint, amely az utasításban foglaltakat *„kivitelezi"*.

— Gondoskodunk róla, hogy ha az értelmező a szövegben már felismert egy magas szintű utasítást, akkor átadja a vezérlést a megfelelő *megvalósító rutinnak*.

Ebben a megoldásban nyilvánvaló, hogy a gép valójában *csak* akkor dolgozik az eredeti, felhasználói feladat megoldásán, amikor az utasításokat kivitelező rutinok futnak. Az értelmező, a vezérlésátadó működése az eredeti feladat szempontjából: *rezsi*. Így adtuk el a sebességet kényelemért!

SZUBRUTINOK, ELJÁRÁSOK

A magas szintű nyelvek, gépek nagy munkával, széles körű felhasználási igények kielégítésére készülnek. Utasításkészletük bármennyire sokat tudó, bármennyire hajlékony is, minden igényt nem elégíthet ki. Az ebből fakadó problémát a nyelvek tervezői például úgy igyekeznek áthidalni, hogy lehetőséget adnak a felhasználóknak arra, maguk készíthessenek még célszerűbb, még magasabb szintű részfeladataikhoz még inkább illeszkedő építőelemeket — ún. szubrutinokat, eljárásokat, függvényeket, modulokat stb. (Az elnevezések és a felhasználás szabályai az egyes nyelvekben eltérőek lehetnek.) Ezek olyan — általában ismétlődő részfeladatokat megoldó — alprogramok, amelyeket a szükséges helyeken egyetlen utasítással, egyetlen utasításként lehet meghívni.

Az igazán jó nyelv lehetővé teszi hajlékony, az alkalmazás (a hívás) körülményeihez alkalmazhatóan módosítható szubrutinok konstruálását — a szubrutinok, az eljárások ún. *paraméterezését*.

PROGRAMOK MINT ÉPÍTŐELEMEK

Természetesen a szubrutinok, eljárások maguk is épülhetnek az utasítások mellett szubrutinokból, eljárásokból. A bontás, az egymásba skatulyázás mélysége elvileg akármeddig növelhető. A szubrutin hívhat egy másik szubrutint, és ez ismét egy másikat stb. Egyes nyelvek azt is megengedik, hogy a szubrutin saját magát is hívja. Ez rekurzív jellegű feladatmegoldásoknál igen kedvező lehetőség. A **program** az, ami ezeket a gyakran mélyen egymásba ágyazódó építőelemeket egységbe foglalja. A magasabb szintű gépek, nyelvek szintjén a **program** az a *„legmagasabb szintű szubrutin"*, amit a virtuális gép még *szoros egységben* lát, vagyis még szoros egységként kezelni képes.

Amit itt „szoros egységnek" mondunk, az nagyon fontos kapcsolatot jelöl, amely sok mindenben megnyilvánul. Például:

— Az értelmező, a fordítóprogram a nyelv szabályainak betartását csak egy-egy program határain belül képes ellenőrizni. Úgy is mondhatjuk, hogy a program a legnagyobb *nyelvi,* fordítási egység.

— Az egyes részfeladatokat megoldó programrészek (eljárások stb.) közötti ellenőrzött kapcsolat a programokon belül sokkal szorosabb lehet, mint a programok közötti kapcsolat. Például ún. globális változókon keresztül, melyek „láthatóságát" vagy „elrejtettségét" egy programon belül kézben lehet tartani. A programok egymás változóit, eljárásait viszont általában nem ismerik.

Az egymásba ágyazódó, ágyazható eljárások közül tehát a legkülsőt nevezhetjük ki **programnak.** Hogy ezt az egymásra építkezés melyik fázisában tesszük meg, azt mi — a feladat megoldásának tervezői — dönthetjük el. Csábító dolog egy részfeladatot egy erre specializált programmal megoldani. A program befejezett, ellenőrzött, önállóan igen jól tesztelhető egész. A program kapcsolata a „külvilággal" laza és jól kézben tartható; rendszerint ún. adatfájlokon, állományokon keresztül jön létre. Ez egyik esetben előny, de másik esetben, például ha olyan részfeladatokat szeretnénk *együttműködő* programokkal megoldani, amelyek között túl szoros, túl intenzív lenne a kapcsolat, hátrány is lehet.

A programokra bontásnak vannak, lehetnek a feladat strukturálására visszavezethetőkön kívül kényszerűségi okai is. Például:

— A teljes feladatmegoldáshoz szükséges utasításmennyiség nem fér el a tárban, vagy a processzor címzéstartománya kicsinek bizonyul ehhez;

— A program mérete akkora lenne, hogy túl hosszú fordítási időket és nehézkes javításokat eredményezne.

Ha egy-egy feladat megoldása közben cserélni kell a programokat, akkor minőségileg új szintre kerülünk: az ún. **operációs rendszer** szolgáltatásait kell igénybe vennünk; de ez már egy másik cikk témája lehet.

— KE —

MOZGÉKONYSÁG

A mozgékonysággal régebbi számainkban többször is foglalkoztunk, de nagy jelentőségére való tekintettel újra visszatérünk erre a témára.

Dap Hartman holland sakkprogramozó is nagy jelentőséget tulajdonít a mozgékonyságnak, más szóval a mobilitásnak, amivel tanulmányában egy teljes fejezeten keresztül foglalkozik. A következőkben ebből a fejezetből ismertetek egy-két érdekesebb adatot, amelyek 832 nagymesterjászma feldolgozásának eredményei.

A grafikonokról a mozgékonyság három különféle értelmezésének különbsége is leolvasható.

A mozgékonyság három különféle értelmezése a következő:

1. A legális lépések száma.

2. A pszeudolegális lépések száma. Ezek olyan lépések, amelyeket megtehetnénk a táblán, ha nem vennénk figyelembe a sakk lehetőségét, vagyis azt, ha királyunk sakkban marad, miután egy kötésben levő figurával elléptünk a helyéről.

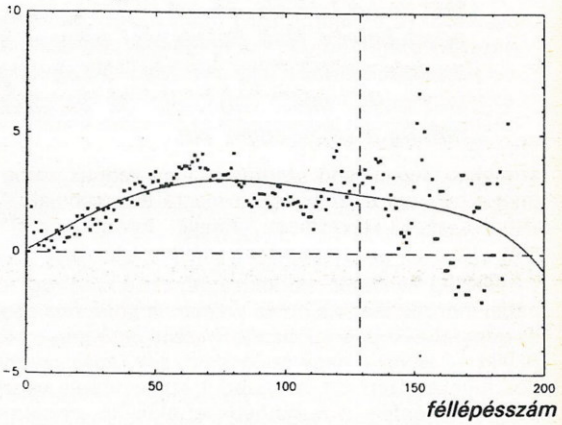
3. De Groot-lépések. Ezt a definíciót De Groot professzor adta, aki a függvényt a következőképpen határozta meg:

Mobilitáson a pszeudolegális lépések számát — ha egyik fél királya sincs sakkban — értjük vagy nem értelmezzük, ha valamelyik fél királya sakkban áll.

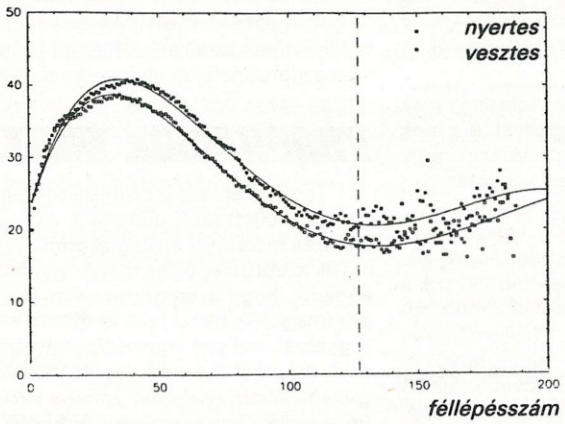
Amelyik sakkprogram az utóbbi definíció szerint számítja a mozgékonysági értéket, az az állásértékelés előtt megvizsgálja, hogy a két király közül nem áll-e sakkban valamelyik, és ha igen, akkor növeli a kutatási mélységet, mélyebben értékel. Ezáltal megnő a játékfa, de az értékelés sokkal megbízhatóbb lesz, megéri a ráfordított gépidőt.

Kovács P. Attila

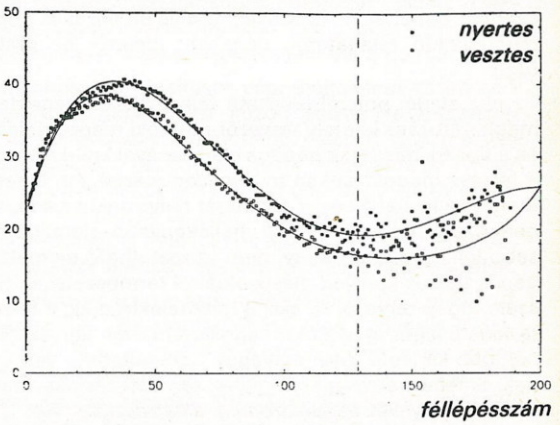
A nyertes és a vesztes legális lépésszámának különbsége



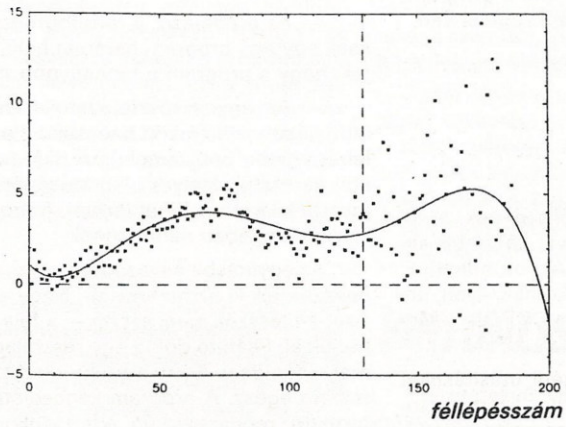
A legális lépések számának átlaga a féllépések függvényében



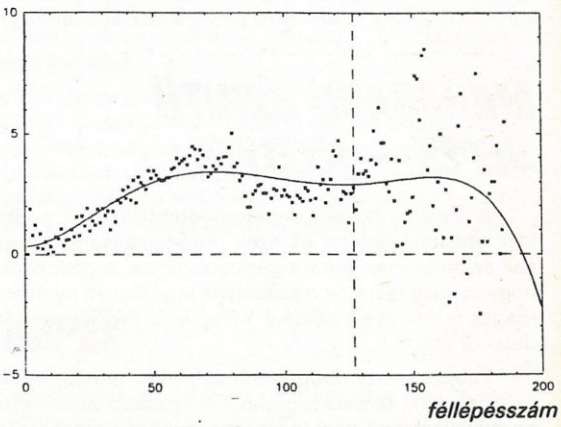
A De Groot-lépések számának átlaga a féllépésszám függvényében



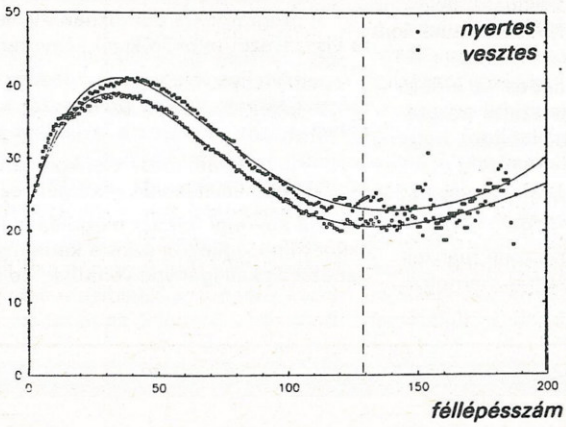
A nyertes és a vesztes pszeudolegális lépésszámának különbsége



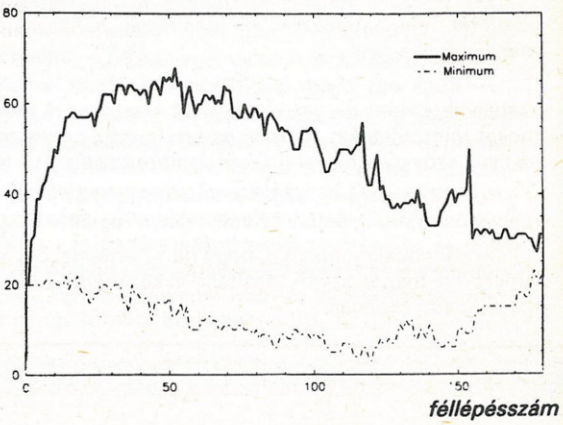
A nyertes és a vesztes De Groot-lépésszámának a különbsége



A pszeudolegális lépések számának átlaga a féllépések függvényében



A pszeudolegális lépések számának maximuma és minimuma a féllépésszám függvényében



Az Olvasó írja

A szerkesztőségben sokszor beszélgettünk arról, hogy a lapnak a tájékoztatáson kívül lehet-e más feladata is. Az elmúlt öt évben sok levelet kaptunk, amelyből kiderült, hogy a Mikroszámítógép Magazin fontos kapcsolatépítő szerepet is betölt a számítástechnikát szerető és a számítógépek használatát jobban vagy kevésbé értő olvasók között. Az első, idézett levél is ezt bizonyítja.

Börzsei Antal, Székesfehérvár

Kedves közreműködésük folytán most kaptam meg az információt a Spectrum és a GLP nyomtató összekapcsolásáról. Dr. Frigyes Ervin úr volt olyan kedves, hogy Afrika távoli részéről juttatta el hozzám értékes tapasztalatait és tanácsait. Ezért a messzemenő gesztusért mind neki, mind a szerkesztőségnek őszinte köszönetemet fejezem ki. A lapnak továbbra is lelkes olvasója maradok.

Örülök, hogy segíthettünk.

Major Tamás, Városföld

Rendszeres olvasója vagyok a lapjuknak, ezért fordulok most a szerkesztőséghez a problémámmal.

Múlt év májusában vásároltam egy Videoton Tv-Computert. Látva a géphez kapható szoftverek szegényes választékát, gondoltam, latba vetem szerény programozói tudásomat, és írok a gépre néhány elfogadható programot. Igen ám, de alig három hónappal a vásárlás után megkaptam a behívót, ezért csak most készült el az első — érzésem és baráti köröm véleménye szerint — terjesztésre méltó játék. Itt jön a bökkenő. Arra gondoltam, hogy a Novotrade-del próbálom kiadatni a programot, de azt sem tudom, hogy miként lehet az ilyen ügyletet lebonyolítani. Ráadásul még az is nehezíti a helyzetet, hogy sorkatonai szolgálatomat sem töltöttem le teljesen.

Kérem, ha tud, adjon tanácsot! Kit kell megkeresni ez ügyben? Ha lehet, írja meg az illetékes címét, esetleg telefonszámát. Esetleg — ha nem esik nehezére — küldje el neki a levélhez csatolt programleírást!

„Pöttöm Peti, a kis erdei manó eltévedt a Sátán birodalmában. Szabadulásának egyetlen módja, hogy elpusztítsa a Démonok Várát, a tornyokon levő zászlók eltalálásával. Útját, mely erdőkön, labirintusokon át vezet, szörnyek, kísértetek és gonosz robotok veszélyeztetik. Kis hősünk bűvös gránátokkal van felszerelve, amelyeket négy irányba és különböző távolságra tud eldobni.

A játék tíz különböző, a játékos mozgásával szinkronban eltolódó helyszínen játszódik. Játszhatja egy, vagy felváltva két személy. A kezdeti életek száma öt. A program pontoz, illetve kijelzi a rekordot.

A program jellemzői:
Név: Pöttöm Peti
Programnyelv: Gépi kód
Irányítható: Billentyűzet, botkormány
Futtatható: 32 és 64 k-s gépen
Adathordozó: Magnókazetta"

A leveléhez mellékelt fenti programleírás — ne haragudjon —, de alig különbözik a többi más kalandjátéktól, így nem hiszem, hogy akár a Novotrade-nél, akár a Videotonnál bárkinek is felcsillanna a szeme, hogy most, íme itt van az a játék, amire a számítógépes világ várt. Ha beküldi, vagy elhozza a kazettát, akkor szívesen elmondjuk a véleményünket a játékról, de azt nem ígérhetjük — látatlanban —, hogy közölni is fogjuk. Hasonlóképpen személyesen kellene eljárnia mind a Novotrade-nél, mind pedig a Videotonnál, hiszen az ötlet, annak ellenére, hogy sokat ér, még azért nem minden. Ezek a professzionális cégek valószínűleg kíváncsiak lesznek a programozás technológiájára is, és valószínűleg elmondják azokat a követelményeket is, amelyeknek a program meg kell, hogy feleljen, ha azt akarja, hogy piacra kerüljön.

Ifj. Fekete László, Budapest

A napokban megvettem az újonnan megjelent GRAFIKA A C64-en című DATA—BECKER NOVOTRADE kiadványt. Örömmel pötyögtem be a grafikai programcsomagot (173. old.), melynek fordításakor az assembler rengeteg hibát jelzett. Ez a program tehát HIBÁSAN van a könyvbe nyomtatva. Hiányoznak a program elejéről a SZIN, a STORE1, a STORE2, az XKH, az XKL és az YK címkék definíciói.

Kérem, küldjék el nekem a program tökéletesen működő változatát! A Mikroszámítógép Magazinban is közzé lehetne tenni!

A Novotrade nevében is köszönjük az észrevételét, amelyet továbbítottunk a cég illetékes szakemberéhez. Leveléhez talán annyi megjegyzést fűznék, hogy jogdíj és más problémák miatt a Mikroszámítógép Magazinban nem közlünk olyan programokat, amelyek valahol, például szakkönyvben megjelentek. Kivétel az olyan eset, ha a szerzői jogokat gyakorló erre fölkér bennünket, akkor szívesen közöljük a helyesbítést. — Végül egy levél, melynek korrektsége önmagáért beszél.

Rátkai István, Esztergom

Ezúton szeretnék elnézést kérni azoktól, akik az Időrégész című programomban elakadtak és levélben fordultak hozzám segítségért — eredménytelenül. Mostanáig katona voltam, se időm, se energiám nem volt, hogy mindenkinek válaszoljak. Remélem, a segítségem nélkül is sikerült végigjátszaniuk a játékot!

* * *

Páris György, a Tudományszervezési és Informatikai Intézet igazgatója az alábbi kiegészítést fűzte a lapunk 1988. novemberi számában a 22—23. oldalon megjelent cikkhez:

A Tudományszervezési és Informatikai Intézet, amely a Művelődési Minisztérium megbízása alapján az iskolaszámítógép-program szervezésében és végrehajtásában 1983 óta részt vesz, a VIDEOTON TVC-k programellátásához is igyekszik hozzájárulni.

Az elmúlt két évben részben más gépeken futó oktatóprogramok átírásával, részben már TVC-re készített programokkal alakította ki jelenlegi programkínálatát, mely 37 különböző programot, valamint programcsomagot (10 programból áll) tartalmaz. Ezek a programok a középiskolai tananyaghoz kapcsolódóan, annak megértését, gyakorlását segítik elő.

A programok többsége a matematika, fizika, földrajz és az idegen nyelvek tárgykörében készült.

Ebben az évben mintegy 600 darab programot, illetv e programcsomagot adott el az intézet az iskolának. A programok választéka és mennyisége folyamatosan bővül részben a tanárok és a diákok körében végzett piackutatási munka révén, részben pályázati úton.

* * *

Leveleiket változatlanul várja:
Kovács Győző

Pályázati felhívás

Az NJSZT, a szekszárdi Garay János Gimnázium és a Mikroszámítógép Magazin szerkesztősége 1989-re is programozási pályázatot hirdet általános iskolás, középiskolás diákok és elsőéves egyetemi hallgatók részére az alábbi kategóriákban:

1. Új, önálló játékprogram készítése HT—1080Z, Sinclair Spectrum, Commodore 16 és 64 és Videoton TV-Computer gépre.
2. Valamely tantárgyhoz kapcsolódó, a tanítási órát segítő, illetve a tanulók önálló tanulását támogató oktatóprogram készítése a fent említett gépekhez.
3. A zsűri az idén ismét külön is elbírálja a pályázatokhoz benyújtott programleírásokat (dokumentáció), a legjobb pályaműveket a verseny szervezői különdíjjal jutalmazzák.

A pályázat benyújtási határideje:
1989. február 15.

A programot mágnesszalag-kazettán (a kazettára többször felvéve), vagy mágneslemezen (floppy) rövid leírás kíséretében (mit tud, hogyan működik a játék, hányadik osztály mely tantárgyához, melyik anyagrészhez kapcsolódik stb.), jeligével ellátva (külön zárt borítékban a név, lakcím vagy iskola) kérjük beküldeni a Mikroszámítógép Magazin szerkesztősége (1371 Budapest, Pf.: 433) címére.

A szerkesztőségnek joga van a pályázaton részt vett programok közlésére, amiért a szokásos honoráriumot fizeti. A döntő 10-10 résztvevőjét az NJTSZ tagjaiból, a Garay Gimnázium tanáraiból és a szerkesztőség munkatársaiból álló előzsűri választja ki. A zsűrinek joga van — ha nem érkezik be elegendő pályázat, vagy a pályaművek színvonala nem éri el a kívánt szintet — a döntő résztvevőinek számát csökkenteni.

A döntő — melyen az előzsűri által kiválasztott programok versenyeznek — Szekszárdon, a Garay János Gimnáziumban rendezendő Garay-napok alkalmából, 1989. március 20-án lesz.

A döntőbe jutott tanulókat a Garay János Gimnázium vendégül látja.

Az 1. és 2. kategóriában az első három helyezett programot díjazzuk, és mindkét kategóriában kiadjuk a közönség díját is.

NJSZT
Dömölki Bálint elnök
Garay János Ált. Gimnázium
Zentai András igazgató
Mikroszámítógép Magazin
Kovács Győző
a szerkesztőbizottság vezetője

Mikroszámítógép Magazin 1989. január

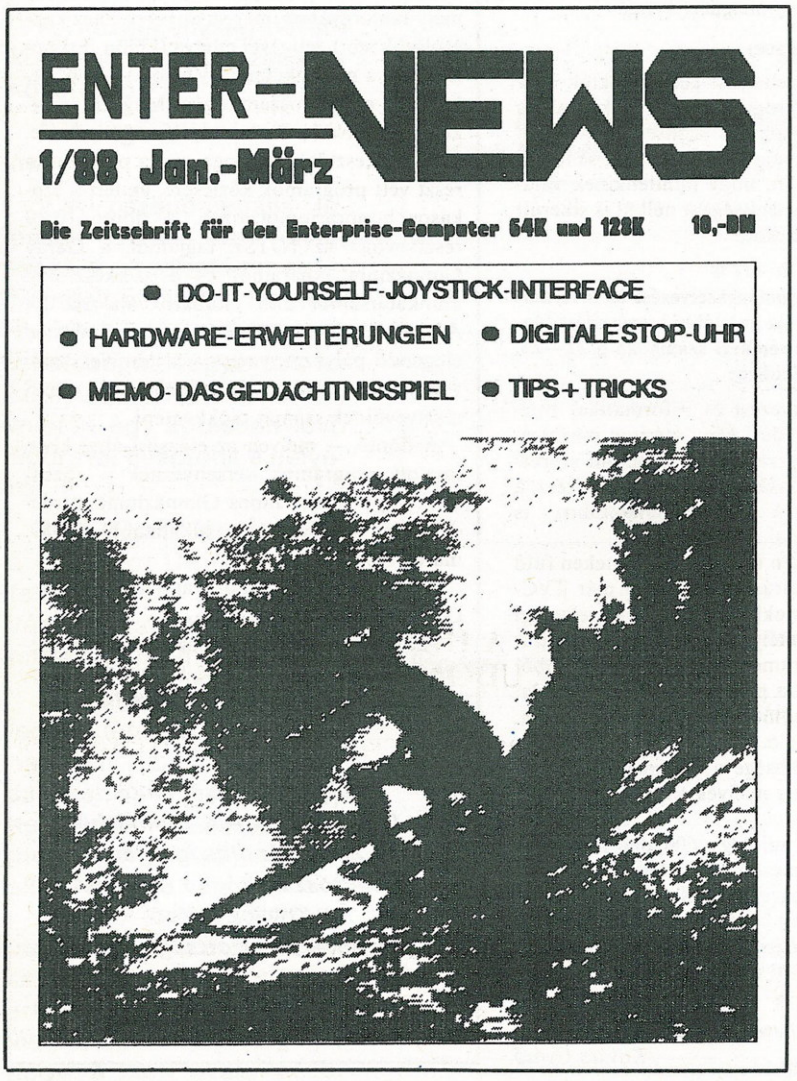
ENTERPRISE® COMPUTERS GMBH

szelvény

Egy sarokkal olcsóbb!!

Az ENTERPRISE COMPUTERS GMBH cég felajánlotta szerkesztőségünknek, hogy részt vesz árengedményes akciónkban. Aki Münchenben jár — ami a világútlevél korában már nem csupán vágyálom — és leadja a magazinból kivágott sarokszelvényt, az az alább felsorolt ENTERPRISE gépek, tartozékok, könyvek árából 15% engedményt kap. Egy sarokszelvény — az eddigieknek megfelelően — csak egy-egy tétel engedményére jogosít. Tehát ha valaki egy csomag kazettát és esetleg egy könyvet is vásárol, annak két szelvényt kell leadnia.

A müncheni cím:
ENTERPRISE COMPUTERS
GMBH
8000 MÜNCHEN 2.
SONNENSTRASSE 3.
NSZK



A CÉG AJÁNLATA:

ENTERPRISE 128K (német BASIC cartridge)	*298 DM*
ENTERPRISE 64K (angol BASIC cartridge)	*198 DM*
EXDOS floppyvezérlő+kézikönyv	*148 DM*
FLOPPY EGYSÉG (5,25 inch, egyoldalas, 40 sávos, 180 Kbájtra formattál)	*198 DM*
FLOPPY EGYSÉG (5,25 inch) + vezérlőegység	*298 DM*
„EP80+" típusú nyomtató (100 kar/s)+illesztőkábel	*348 DM*
SPEAK—EASY beszédszintetizátor	*69 DM*
Mindegyik programkazetta	*15 DM*
Programkazetta-csomag (3 db)	*40 DM*
Programkazetta-csomag (10 db)	*100 DM*
BASIC oktatókönyv	*19 DM*
EXDOS kézikönyv	*19 DM*
IS DOS kézikönyv+floppy	*29 DM*
EXOS (műszaki kézikönyv)	*39 DM*
ENTERPRISE 128K kézikönyv (265 oldalas, német nyelvű)	*10 DM*
Nyomtatókábel (CENTRONICS, 8 bites, párhuzamos)	*24 DM*
Botkormány-kábel	*24 DM*
Monitorkábel (SCART, CYNCH csatlakozóval vagy anélkül)	*24 DM*
RS232C kábel soros nyomtatókhoz	*24 DM*

PROGRAMTERMÉK

A legfrissebb grafikonrajzoló

A laboratóriumi mérések kiértékelése hajdan igen fáradságos munka volt. A számítógépkorszak beköszöntével azután átvették a terhet a különféle görbeillesztő programok. A probléma így meglehetősen egyszerűsödött, ráadásul meghökkentő függvényválasztásokkal kísérletezhetünk perceken belül. A probléma azért mégsem ilyen egyszerű. A munkát csak olyan program támogathatja elég jól, amelyet felkészítettek az ad hoc ötletek fogadására is. Ez programozástechnikailag nem túl könnyű feladat, és erősen rabolja a gépidőt. Az utóbbit az alkalmazó abból veszi észre, hogy lassan kapja meg az eredményeket. Olykor elviselhetetlenül lassan!

Ebben a hónapban a Tudományszervezési és Informatikai Intézet választékából egy ilyen, görbeillesztő témájú oktatóprogramot elemzek. A program teljesen új, igazán még a piacra sem került.

A programleírás, amelyet természetesen még csak kéziratban olvastam, nem túl terjedelmes, de a program működését jól írja le. Fontos körülmény, hogy a program nem valamely konkrét tananyaghoz kapcsolódik, ezért más iskolai tankönyvre nem támaszkodhat. A „felsőbb mennyiségtan" könyvek persze nyújthatnak segítséget a pedagógusnak, de a szakköri diáknak már kevésbé. A dokumentációnak ezért e programnál igen jelentős szerepe van. A végleges változathoz véleményem szerint a kézirat anyaga még nem lesz elég.

A program meglehetősen általános függvényosztályokkal való munkára készült fel, ami dicséretes. Ugyanakkor nem veszi figyelembe a számítógép adta korlátokat. Bizony, nagyon lassan, nyögdécselve bírja csak az algoritmusok nagy terhelését. Bonyolultabb esetekben kifejezetten lassan lehet vele haladni. Az alkalmazók emiatt valószínűleg nem fogják eléggé méltányolni a programmal elérhető igen trükkös lehetőségeket. A program használata tehát kicsit nehézkes.

Ugyanakkor alaposan kihasználja a C Plus/4 grafikai tulajdonságait. Az ábrázolás nagyon szemléletes. Megválaszthatjuk a koordináták skálázását, és lehetőségünk van arra is, hogy az adatokat transzformáljuk a rajzolás előtt. A pontok összekötési módja és a pontok piktogramja választható, valamint három képméret is használható. A kirajzolt grafikonra utólag feliratokat tehetünk, a kész ábrát pedig kinyomtathatjuk mátrixnyomtatón, de tárolhatjuk akár kazettán vagy lemezen is.

Lassúság ide, lassúság oda, ami engem illet, néhány, egyetemen elszenvedett laboratóriumi gyakorlaton szívesen ráhagytam volna az elemzéseket erre a programocskára, ha akkor lett volna ilyen. A program ugyanis lényegében matematikai képleteket tud elfogadni transzformációs formulaként vagy illeszkedő függvényként, ami igen bonyolult összefüggéseket is jelenthet. Persze, illik vigyázni a függvények értelmezési tartományaira. Az X koordináta mellett négy Y koordináta transzformációjáról intézkedhetünk. Az ábrán így vagy négy függvény görbéje vagy két függvény görbéje és hibahatára jeleníthető meg.

Ezzel együtt az adatok bevitelével elég sokat kell nyűglődni, bár a táblázatos beviteli lehetőség mindazon segítséget megadja hozzá, ami egyáltalán lehetséges. Nincs szó tehát arról, hogy a mérőberendezésekről közvetlenül feldolgozhatók lennének az adatok, jóllehet a program transzformációs képességei minden lehetőséget megadnának a műszerről érkező adatok esetleg felmerülő konverziójára. De ne legyünk telhetetlenek. A program már így is túl sokat tud, már ami a processzort illeti, hiszen nyög belé, szegény.

A Grafikonrajzoló, minthogy adatként érkező akármilyen számokkal és beírható formulákkal dolgozik, igen univerzálisan használható. Egyszerűbb dolgoknál a sebességjellemzők sem fognak zavarni. Ha azonban valaki a CRAY—1 pótlására szeretné használni, hát magára vessen.

A programcsomag órai kísérletek eredményeinek elemzésére és szakköri munkákhoz egyaránt javasolható.

Zsadányi Pál

MINŐSÍTŐ ADATOK

Kezelhetőség:	közepes
Teljesség:	kiváló
Dokumentáltság:	jó (jelenleg)
Használhatóság:	jó
Ár/teljesítmény:	kiváló
Összbenyomás:	jó

ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Tudományszervezési és Informatikai Intézet
Terméknév:	Grafikonrajzoló
Szerző:	Nagy Attila
Géptípus:	C Plus/4
Hordozó:	kazetta vagy lemez
Dokumentáció:	14 oldal (még kéziratban láttam)
Ár:	450 + 113 (ÁFA) Ft (kazettán) (ha hajlékonylemezen, akkor még + lemezköltség)

A HÍRADÁSTECHNIKAI TUDOMÁNYOS EGYESÜLET

Mikroszámítógépes programnyelvek és operációs rendszerek szakosztálya előadást tartott 1988. dec. 6-án a MTESZ székházában (Budapest V., Kossuth tér 6—8.) az alábbi címmel:

Gépészeti és elektronikai tervezés az INNOVA CAD gyakorlatában.

Az előadó dr. Karassay Árpád irodavezető volt.

KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

Schultz, Ingo — Pest, Werner:
Nyomtatók
(Budapest, 1988.
Műszaki Könyvkiadó, 169 oldal.
Ára: 167,— Ft)

A nyomtató vásárlásakor több kritériumot is célszerű figyelembe venni, például az írásképminőségét, a nyomtatási sebességet, a megjelenítési lehetőségeket és az illeszthetőséget. Helyesen választani csak az tud, akinek legalább alapismeretei vannak a különböző nyomtatófajtákról és azok teljesítményének jellemzőiről. A könyv ezeket az alapismereteket nyújtja, és így segítséget ad a célnak megfelelő nyomtató kiválasztásához. Azoknak, akiknek már van nyomtatójuk, a kötet információt szolgáltat a saját nyomtatójuk illesztéséhez, ötleteket ad annak optimális használatához. Áttekinti a különböző nyomtatófajtákat — tűmátrix, margarétakeretes, hő-, tintasugár- és lézernyomtatók —, a nyomtatók illesztését a különböző számítógép- és interfésztípusokhoz, a nyomtatók intelligenciáját és megjelenítési lehetőségeit, a nyomtatók tartozékait, valamint a kiválasztás szempontjait. A könyv végén található fontos fogalmak gyűjteménye lehetővé teszi a mű referenciakönyvként való használatát.

Deák László:
Mikroszámítógépes oktatóprogramok készítése és alkalmazása
(Budapest, 1988.
LSI ATSZ, 147 oldal.
Ára: 175,— Ft)

A kiadvány elsősorban azoknak a pedagógusoknak készült, akik olyan iskolában tanítanak, ahol az oktatást iskolai számítógépek — C Plus/4, C16, Primo, a HT sorozat vagy a TV-Computer — segítik.

Benedikti István–Huczka Béla:
Ismerkedés az ENTERPRISE-szal
(Budapest, 1988.
Műszaki Könyvkiadó, 226 oldal.
Ára: 300,— Ft)

A szerzők könyvüket elsősorban azoknak ajánlják, akiknek az Enterprise az első számítógépük. A könyvtárnyi szakirodalomból a gép megismeréséhez nélkülözhetetlen ismeretanyagot választottak ki, amelyet egy-két érdekességgel is kiegészítettek. A cél a számítástechnika és az Enterprise bemutatása volt. Tudományos igényű, mélyebb feltárásra azonban nem vállalkoztak. Arra törekedtek, hogy a leírtak segítségével — ha még nem is a legügyesebben, de — a számítógépet használni lehessen. Az Enterprise házi számítógépként használható. Az eddigi hozzáférhető számítógépek közül kiemelkedik grafikai és akusztikai tulajdonságaival, ára pedig átlagos.

Magyarországi elterjedését elősegíti, hogy mutat némi rokonságot a Videoton TV-Computerével, és az is, hogy sok szakember már régóta ismeri. Ezért várható, hogy építenek majd hozzá különböző hardver- és szoftverillesztőket, és kiegészítik új programokkal is.

Bernáth-Farkas Zoltán:
Commodore—64 file kezelés és Input–Output
(Budapest, 1988.
LSI ATSZ, 141 oldal.
Ára: 119,— Ft)

A könyv a bemenet/kimenet kezelés BASIC és assembly szintjét ismerteti, valamint részletesen foglalkozik a fájlszervezés különféle módozataival is. A programozókon kívül azok is haszonnal forgathatják, akik ebbe a témába még csak most kívánnak belemélyedni.

Papert, Seymour:
Észrengés. A gyermeki gondolkodás titkos útjai
(Budapest, 1988.
SZÁMALK, 165 oldal.
Ára: 160,— Ft)

A könyv szerzője a híres észak-amerikai egyetem, a Massachusetts Institute of Technology mesterséges intelligencia tanszékének professzora, akit a világ a LOGO nyelv megteremtőjeként ismer. A LOGO-t a közvetlen gépi kapcsolat, a modellalkotás, az algoritmikus absztrakció és a teknősgrafika jellemzi. Ez utóbbit azóta már más nyelvekbe is beépítették (Smalltalk, Pascal, ELAN).

A könyv a LOGO nyelv filozófiájáról, céljáról és alkalmazásáról szól pedagógusoknak, szülőknek és azoknak a téma iránt érdeklődőknek, akik a számítógép használatára tanítanak meg másokat vagy tanítják önmagukat.

Pirkó József:
3D. Perspektivikus grafika IBM PC-n Turbo Pascalban
(Budapest, 1988.
LSI ATSZ, 132 oldal.
Ára: 149,— Ft)

A számítástechnika rohamos fejlődésével fejlődésnek indult a számítógépes grafika is. A problémák megoldása során gyakran válik szükségessé a mennyiségek szemléltetésére és bemutatására függvények, grafikonok, diagramok megjelenítése. A számítógépes grafikának sok alkalmazási területe van: a mérnöki tervezésben, a CAD- (számítógéppel segített tervezés) rendszerekben, műszaki rajzok, áramkörök tervezésében. Ezek az alkalmazások igénylik a háromdimenziós megjelenítést, ezen belül is a mozgó, térbeli grafikát. A cél olyan számítógépes algoritmus kidolgozása, amely a lehető legnagyobb mértékben igazodik térlátásunkhoz. A kötet lépésről lépésre követi a számítógépes térlátáshoz vezető utat. Ismerteti a szem anatómiáját és az ezekre épülő matematikai modellt. Mindezt IBM személyi számítógépre készített Turbo Pascal programozási nyelven mutatja be.

JÖVENDÖLÉS

Manapság, amikor a világ sorsát egyre kevésbé a véletlenek határozzák meg, Japánban nemcsak virágzó iparág a jóslás, hanem egyre inkább hódít annak legmodernebb, csúcstechnikát alkalmazó, számítógépesített változata is. Becslések szerint minden negyedik lakos, azaz 30 millió ember évente egyszer jóshoz fordul a távol-keleti szigetországban, hogy kikérje a jövendőmondó „szakemberek" véleményét. Az óriási üzlet csábította egy szoftverház szakértőit, akik az asztrológiától a tarokk-kártya-vetésig, a jövendölés teljes programtárházát kínálva, a lehető legnagyobb matematikai valószínűségre törekedve indítottak hadjáratot annak érdekében, hogy elhódítsák a jövőt kifürkészni akarókat a „kisipari" módszerekkel dolgozó tenyérjósoktól. A Tokióban létrehozott modern jósdában 80 számítógépet alkalmazó hivatásos jós teljesít szolgálatot. Elég, ha az ügyfél a szenzoros tévéképernyő felé fordítja a kezét, a jóslási program azonnal indul. E szolgáltatás egyáltalán nem drága, csupán annyiba kerül, mint egy mozijegy: felnőttek 3000, egyetemisták 2500, középiskolások 2000 jent fizetnek érte.

HUNGAROFLUOR

Az MTA Izotópintézetének munkatársai olyan számítógépes rendszert fejlesztettek ki, amely a gyógyszerek adagolásában segítheti az orvost. A Hungarofluor rendszer egyelőre négy orvosságnak „betegre szabott" beállítását teszi lehetővé. Mind a négy gyógyszerre az a jellemző, hogy szűk a gyógyítási tartománya: ha a beteg kevesebbet kap belőle a kelleténél, nem gyógyul megfelelően, a túladagolás pedig súlyos szervi károsodáshoz vezethet. A mintegy ötévi kutatómunkával kifejlesztett műszer egészen kis mennyiségű vérmintából meghatározza, hogy mennyi gyógyszer jutott a páciens vérébe. A mérés adatait a műszerhez csatlakoztatott Commodore 64-es mikrogép elemzi és értékeli. Ennél fogva egyetlen óra leforgása alatt akár száz vérminta is megvizsgálható különösebb szakértelem nélkül. A műszer kifejlesztői remélik, hogy műszerük mihamarabb sok helyen elterjed. Hiszen ha pontosan adagolják a gyógyszert, gyorsabb a gyógyulás, és az orvos is azzal a megnyugtató tudattal láthatja el munkáját, hogy annyi gyógyszert adott csupán, amennyire feltétlenül szüksége volt a betegnek.

TERET HÓDÍT AZ AUTÓELEKTRONIKA

A személy- és tehergépjárművekbe épített elektronika a következő évtized-

ben minden bizonnyal nagy fejlődésen megy át. Míg az elektronikai berendezésekre a fejlett tőkés országokban ma a gépjárművek előállítási költségéből mintegy 3 százalékot számítanak, a prognózisok szerint ez az arány tíz év alatt eléri a 17 százalékot. Jelenleg az elektronikai rendszereknek mintegy a fele a motort szolgálja, de hamarosan várható más elektronikus kiegészítők — például ajtózárak, fékberendezések — térhódítása is. A gépkocsikhoz előállított elektronikai berendezések száma az 1987. évi 12 millióról 1995-ig várhatóan 100, 2000-ig pedig 200 millióra növekszik.

ŐRSÉGVÁLTÁS NYOMTATÓÉKNÁL?

Az IDC piackutató intézet a következő években gyökeres változásokra számít a nyomtatók NSZK-beli piacán. A mátrixnyomtatóké pedig 19-ről 7 százalékra zsugorodik. Ezzel egyidejűleg a lapnyomtatók, különösen pedig a lézeres nyomtatók nyomtatók ma még 60 százalékos aránya 1992-re 38 százalékra csökken, a sorerőteljes elterjedése várható. Részesedésük az NSZK-ban a számítógép-nyomtatók piacán a jelenlegi 16 százalékkal szemben 1992-ben valószínűleg meghaladja az 50-et.

PI — 200 MILLIÓ SZÁMJEGYBŐL

Kanadában a Hitachi cégnek egy új, közelebbről meg nem jelölt számítógépén előállították a pinek 201 326 000 számjegyét. A feldolgozás hat óráig tartott. Ennek a pontosságnak nincs ugyan gyakorlati jelentősége, de a kanadai kutatók tovább folytatják a munkát, s céljuk, hogy a 400 millió számjegyes pontosságot is elérjék. Mivel a pi végtelen tizedes tört, ezért a számjegyeinek a száma is végtelen, nincs határa.

OLAJFOLTOK NYOMKÖVETÉSE

Skót tengerkutatók új számítógépes programcsomagjával előre jelezhetik a tengerparti fürdőhelyeket szennyezéssel fenyegető olajfoltok mozgását és viselkedését olyan tényezők figyelembevételével, mint a szél, az áramlatok, a párolgás. A PC Oil Slick nevű szimulációs rendszert IBM PC-vel kompatibilis mikrogépeken lehet futtatni. A nagy felbontású színes rendszer lehetővé teszi a rákérdezést is az ominózus területre. Mikor a szimuláció befejeződik, grafikonok sora jelenik meg a térképek mellett, minden részletet megadva a támadó szennyeződésről. A programcsomaggal nemcsak az olajfolt mozgását lehet modellezni, hanem a tervezett tisztítási beavatkozás hatékonyságát is.

OKTATÓRENDSZER CSÚCSTECHNIKÁVAL

A miskolci Zalka Máté gépipari középiskolában csúcstechnika oktatására alkalmas gyártósor szerelése fejeződött be. Hazánkban ilyen magas színvonalú, oktatási célokat szolgáló gyártórendszert még egyetemeken sem állítottak munkába. A miskolci iskola az országos középtávú kutatási-fejlesztési tervpályázat elnyerésével jutott hozzá az oktatórendszerhez. A több millió forint értékű gépsor az osztrák Elco cég gyártmánya, CNC esztergaés marógépekből áll, ezeket egy japán Mitsubishi Move Master 1. típusú robot szolgálja ki. A szerszámgépekhez továbbítja és befogja a munkadarabot, majd egy munkafolyamat befejezése után további megmunkálásra másik szerszámgépre viszi át. A szerszámgépek s a robot munkáját egy IBM PC XT hangolja össze. A rendszer rugalmas: a vezérlő mikrogépbe más programot töltve pillanatok alatt átállítható más-más alkatrészek gyártására.

ADOK—VESZEK —CSERÉLEK

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

ADOK

Apple II gép (64 k), 2 lemezegységgel olcsón eladó. 367-547

Atari program idegen nyelvek tanulásának megkönnyítéséhez. Választék helyenként ingyenes tájékoztató! Dr. Gerő László, Szeged, Budapesti krt. 4/b. VIII/31. 6723

Atari 600XL (16 k) számítógép datasettel, 2 db joystickkal, játékprogramokkal, kézikönyvvel 15 000 Ft-ért eladó. Vértes Balázs, Budapest, Leányka u. 3. 1221, Tel.: 385-867

C64-re írt programmal egyszerűen, gyorsan kiegészíthető BASIC programunk gépi kódú részekkel. Assembler, újszerű hivatkozási lehetőség, új parancsok 2 K-n. Kívánságra tájékoztatót küldök. A program ára használati utasítással (csak kazettán) 599 Ft (utánvéttel). Levélcím: Sződ, Dózsa György u. 261. 2134

C64 + VC 1541 + 1 db joy. + programok lemezen eladók. Árajánlatokat levélben várom. Irimi János, Hajdúnánás, Ady E. krt. 33. I/6. 4080

C64-esemet datasettel, floppyval, joystickkal és rengeteg programmal 35 000 Ft körüli áron eladnám. Válaszokat árajánlattal a következő címre várom: Magyar Attila, Kapuvár, Lenin u. 10. 9330

C64-re játékprogramokat eladok, cserélek (kazettán), listával. Mezei László, Nyírbátor, Kölcsey u. 17. 4300

C64-II, 1541-es floppy, Seikosha SP180-as nyomtató olcsón eladó, külön is. Filep Róbert, Budapest, Szakasits Á. u. 65. IV/33. 1119

Commodore Plus/4 számítógép lemezegységgel, magnetofonnal, 2 db joystickkal készpénzért eladó. Jáki Tamás, Budapest, Juhász Gyula u. 56. III/10. 1039

Commodore 128D + 1581 +1351 +1700 Seikosha 1200 + könyvek, lemezek eladók. Vetor Péter (Jégvirágfagyizó), Budapest, III. Miklós u. 7.

Enterprise programokat adok és cserélek. Választ listával kérek! Kónyár Imréné, Újszász, Hóvirág u. 3. 5052

Sinclair ZX-Spectrum (128 k) + két típusú komplett számítógép eladó. Dr. Hajas, hétköznap este 895-960, napközben 632-070

THOMSON 4121 szines monitor eladó. (RGB, SCART, VIDEO bemenet.) Csatlakoztatható: IBM PC, Amiga, C64-es gépekhez. Filep Róbert, Budapest, Szakasits Á. u. 65. IV/33. 1119

Videoton TV-Computer (32 k), tárbővítővel és programokkal eladó. Szövegszerkesztő, memo, sakk, bridge stb. Ezenkívül egy RPR 210-01 típusú mátrixnyomtató. Telefon: 687-596 napközben

ZX-Spectrum+ botkormánnyal, interfész 2-vel, Computone magnóval, 370 db játékkal és felhasználói programokkal (TAPPER, SKI, STAR, HP4T15M) eladó. Érdeklődni lehet a következő címen: Kovács László, Poroszló, Berzsenyi u. 1/1. 3388

ZX-Spectrum (48 k) + joystick + interfész + játékprogramok eladók. Kovács Zoltán, Fehérgyarmat, Május 14. tér 17. 4900

CSERÉLEK

C16 programokat cserélek kazettán. Cservenák Róbert, Mernye, Jókai u. 14. 7453

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

Enterprise programokat cserélek, elsősorban lemezen (VC 1541). Veresegyház, Pf.: 28. 2112

Enterprise programokat cserélek. Listát kérek! Cím: Három István, Szolnok, Barátság u. 14. 5000

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

TVC-re való (64 k) játék- és demoprogramokat cserélnék. Listát kérek és adok! Szalai László Dániel, Érd, V.ker. Deák F. u. 99. sz. 2030

Pontvadászat

Tavaly indított rejtvénysorozatunk végéhez érkezett. Ebben és a következő számunkban lerójuk még a tartozásunkat, közöljük a feladatok megoldásait. Az 1989/2. számban tesszük közzé a legjobb tíz pontvadász nevét, akiket könyvutalvánnyal jutalmazunk.

Reméljük, feladataink elnyerték az olvasók tetszését.

Dr. Hoffmann Tibor

Az 1988/7. szám 1. feladatára 9 megfejtés, a 2. feladatára 7 megfejtés érkezett. Mindkét feladatra a maximális 3 pontot véve 100 százaléknak, az 1. feladat megoldási százaléka 88,9, a 2. feladaté 52,3 százalék.

AZ 1988/11. SZÁM FELADATAINAK MEGOLDÁSA

1. FELADAT

a) Táblázatban állítsuk össze, hogy az egyes lépésekben a három edény mindegyikében hány liter folyadék van. Az egyes lépések olyan változást hoznak létre, hogy egy eddig folyadékot tartalmazó edény vagy kiürül, vagy megtelik. A táblázatban az edény folyadéktartalmának mértékszáma mellett azt is feltüntetjük egy T betűvel, hogy az edény tele van-e. Az üres edényt nyilván a 0 mértékszám jelzi.

4T	0	2T
4T	2	0
2	2	2T

Ebből is látható, hogy *2* áttöltéssel célhoz érünk. (2 pont)

b) Most jelöljük az egyik folyadék mennyiségét az A szimbólum együtthatójaként és a másikat a B szimbólum együtthatójaként, s az előbbiekhez hasonlóan T-vel jelöljük az edény teljes megtöltöttségét. A táblázat ekkor a következő lehet. (Ld. a táblázatot.)

4AT	0	2BT
A	3AT	2BT
A+2B	3AT	0
$\frac{1}{3}A+\frac{2}{3}B$	3AT	$(\frac{2}{3}A+\frac{4}{3}B)T$
$(\frac{10}{3}A+\frac{2}{3}B)T$	0	$(\frac{2}{3}A+\frac{4}{3}B)T$
$(\frac{10}{3}A+\frac{2}{3}B)T$	$\frac{2}{3}A+\frac{4}{3}B$	0
$\frac{5}{3}A+\frac{1}{3}B$	$\frac{2}{3}A+\frac{4}{3}B$	$(\frac{5}{3}A+\frac{1}{3}B)T$
$\frac{5}{3}A+\frac{1}{3}B$	$(\frac{3}{2}A+\frac{3}{2}B)T$	$\frac{5}{6}A+\frac{1}{6}B$
$(\frac{8}{3}A+\frac{4}{3}B)T$	$\frac{1}{2}A+\frac{1}{2}B$	$\frac{5}{6}A+\frac{1}{6}B$
$(\frac{8}{3}A+\frac{4}{3}B)T$	$\frac{4}{3}A+\frac{2}{3}B$	0
$\frac{4}{3}A+\frac{2}{3}B$	$\frac{4}{3}A+\frac{2}{3}B$	$(\frac{4}{3}A+\frac{2}{3}B)T$

Táblázat

A megoldásunkból látható, hogy *10* áttöltéssel értünk célt. Viszont ebből a vázlatból nem világos, hogy ez-e a legkevesebb szükséges áttöltés. (4 pont)

2. FELADAT

A gépen a következő teendők vannak. Mindkét vektor operatív tárbeli és veremtárbeli kezdőindexét rögzítjük. Ez az f) és g) tevékenységgel *6* μs időt igényel. Ezután két *32* lépéses ciklus következik, melynek a magjában az operatív tár elemeit átvisszük a veremtár elemeibe. Ezek magja az a), b) és c) tevékenységgel $32\times(15+2+1)$ μs = *576* μs, tehát összesen *1152* μs. Az eddigi lépések mind az egyszerű, mind a párhuzamos működésre alkalmassá tett gép esetében szükségesek.

Most az egyszerű gépnél egy olyan ciklusra van szükség, amely a már a veremtárban lévő két tömb elemeit sorban felveszi, összeszorozza és leteszi az egyik tömb elemeinek a helyére. Ehhez először meg kell adnunk, hogy a két tömb első eleme milyen indexszel bír. Ez a g) szerint *2* μs. Ezt egy olyan ciklus követi, amelynek a magjában a d) tevékenység és kétszer a c) tevékenység van. A ciklus *32* lépéses, tehát a szükséges idő $32\times(20+2)$ μs = *704* μs. A párhuzamos működésre alkalmas hardvernél ugyanez csak *20* μs.

A következő lépés, a szorzatok összegzése, megint ugyanaz mind a két géptípusra. Ez egy *32* lépéses ciklus, amelynek a magjában a tömb egyik helyének, például az utolsónak az értékéhez sorban hozzáadjuk a tömb előző elemeit (a részszorzatokat). Ehhez először a tömb utolsó elemének indexét kell megadni (és ez nem változik a ciklus folyamán) és a tömb első indexét, amely már növekszik a ciklus lefutásakor. Ez *2* μs-ot igényel. Végül ennek a *32* lépéses ciklusnak az időigénye így $32\times(30+1)$ μs = 992 μs. Így az összes idő az egyszerű gépnél *2858* μs (6 pont), a *32* párhuzamos processzorral ellátott gépnél pedig *2174* μs (4 pont), ami csak *23,9%*-os időrövidülés. Ennek az időrövidülésnek az ára igen nagy: *31* extra processzor.


PS/2 MODEL 60

COMPFAIR '88


star
LC-10
LC-10 COLOUR
LC-24-10
NX-15
ND-15
NR-15
NB-24-10
NB-24-15

Ilyen volt a jövő

COMPFAIR '88