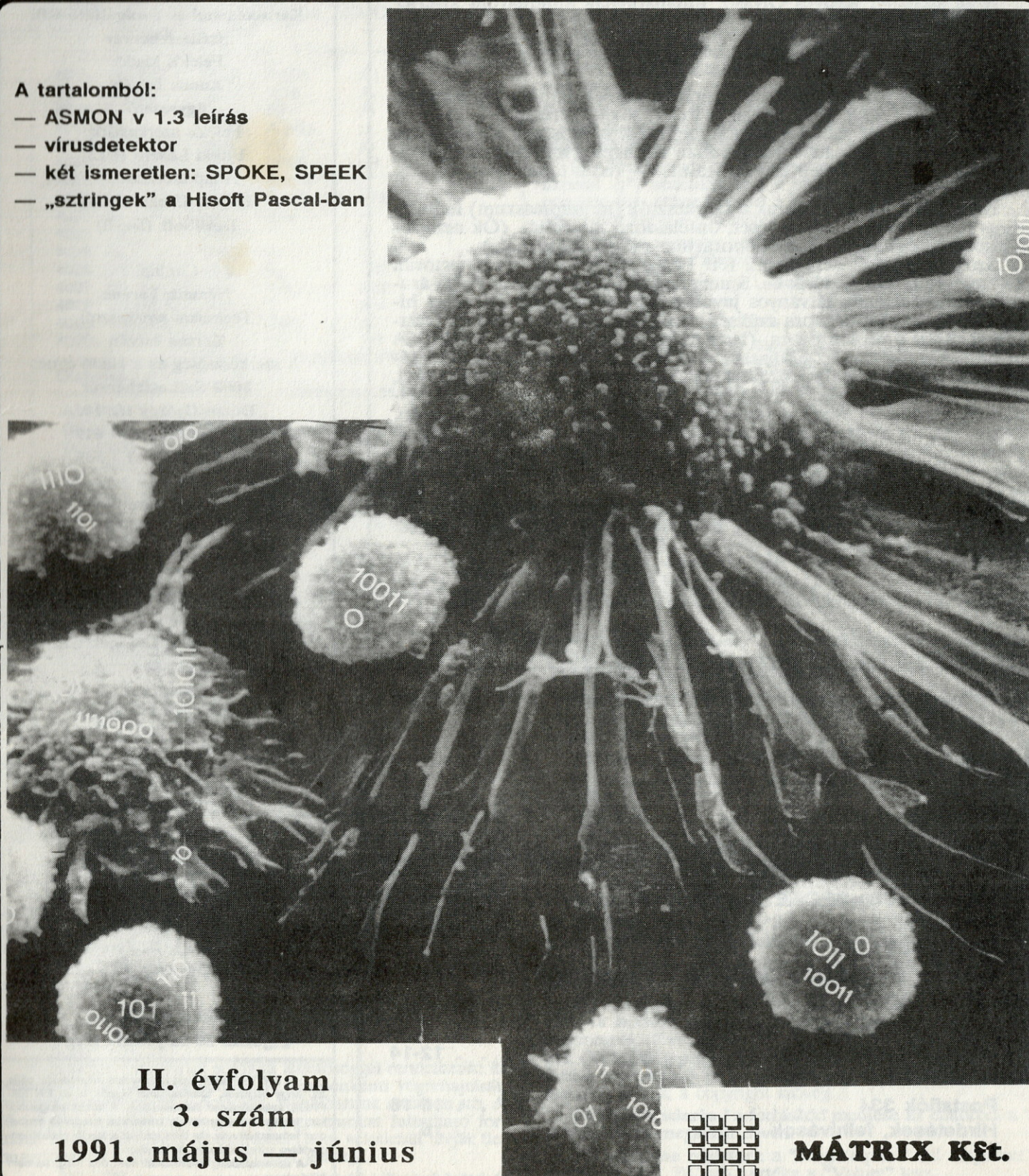


ENTERPRESS

KÉTHAVILAP AZ ENTERPRISE SZÁMÍTÓGÉPEK FELHASZNÁLÓINAK

A tartalomból:

- ASMON v 1.3 leírás
- vírusdetektor
- két ismeretlen: SPOKE, SPEEK
- „sztringek” a Hisoft Pascal-ban



II. évfolyam
3. szám
1991. május — június



MÁTRIX Kft.

Kedves Olvasó!

Megértésére, türelmére apellálok, amikor e néhány sor elolvasására kérem. Eddig Ön ezen a helyen a lap szakmai arculatát biztosító szerkesztők néhány gondolatát, üzenetét, tervét olvashatta. Most azonban szót kér a kiadó, ugyanis: jubileumi számhoz érkezünk! A lap fennállása óta ez az ötödik „kísérleti” szám:

— ötödik kísérlet arra, hogy a kis hazánkban ENTERPRISE-t használók közel húszeszeres táborából megtaláljunk néhányat,

— ötödik kísérlet arra, hogy a kiváló terjesztési tapasztalatokkal rendelkező Postának a ma már erősödő konkurenciával szemben bizonyítási lehetőséget adjunk,

— ötödik kísérlet arra, hogy társaságunknak egyéb tevékenységei mellett gazdaságos lapkiadási üzletágat biztosítsunk.

Nem áltatom, Kedves Olvasó: pillanatnyilag mindhárom kísérletünk vesztesre áll.

Mielőtt lapunk ellenzéke — eljutva eddig az olvasásban — fékeveszett ünneplésbe kezdene, őket is türelemre intjük: mi még nem adjuk fel!

Nem adjuk fel, mert a hozzánk érkező levelek alapján úgy érezzük, hogy az ENTERPRESS-t sokan szeretik, sokan keresik.

Nem adjuk fel, mert most már bennünk is erősödik a bizonyítás vágya: napjaink lapkiadási hullámában a rengeteg hirdetés, szex, pletyka mellett a földi élet egyéb hívságainak — tudás tapasztalat, korszerűség — is helve van.

Nem adjuk fel, mert látjuk a szerkesztők (ld. impresszum) fantasztikus (és fanatikus) lelkesedését, önfeláldozó munkáját. (Ők nem főállásban és egyelőre bizony honorárium nélkül dolgoznak.)

Szóval egyelőre nem adjuk fel! Persze nem azért, mert tartósan veszteségre rendezkedünk be. S nem azért, mert hiszünk a Posta árusítási terjesztésének látványos javulásában. Nem adjuk fel, mert hiszünk abban, hogy a lapra szükség van. Ezért kérjük a Kedves Olvasót: segítsen a terjesztésben. (Nem utcai rikkancsokat keresünk!!) A legnagyobb segítség az előfizetés. Ezt lehet a Posta közreműködésével (reméljük a jelenlegi 425 előfizető megkapja ezt a szolgáltatást), s lehet a kiadón keresztül is intézni. Társaságunk vállalja egyedi példányok elküldését, vagy a rendszeres postázását, előfizetés alapján. (Az előfizetés megrendelhető levélben (MÁTRIX Kft., 8000. Székesfehérvár, Dózsa Gy. tér 10.) A budapestieknek némi előnyt jelenthet, hogy a lap bármelyik példánya kapható jelenleg is a Műszaki Könyv-áruházban (VI. ker. Liszt F. tér 9.).

Remélve, hogy az ötödik — jubileumi — szám jó értelemben vett fordulópont lesz a lap életében, további hasznos, kellemes időtöltést kívánok a lap olvasóinak.

A kiadó nevében:

Annus István

TARTALOM**KURZUS**

Assemly 5.	3-5
A Pascal 5.	6-7
Lehetőségek Páratlan Tárháza (LPT) 4.	8

PROGRAMOZÁSTECHNIKA

Az ismeretlen SPOKE és SPEEK	9
Virus ?	10

TIPPEK-TRÜKKÖK

Köztük: színkeverés, sztringek a Pascal-ban	11
---	----

KÖNYVED MŰFAJ

Green Hill, Super Robin Hood, Joeblade, Nether Earth	12-14
---	-------

MINDENFÉLE

Postafiók 334	15-16
Hirdetések, felhívások	16

ENTERPRESS

Kéthavilap
az ENTERPRISE számítógépek
felhasználóinak

II. évfolyam 3. szám
1991. május — június

Kiadja a

**MÁTRIX Számítástechnikai,
Kereskedelmi és Szolgáltató Kft.**

Székesfehérvár

Felelős kiadó:

Annus István

ügyvezető

Felelős szerkesztő:

Ujlaki László (UL)

A lap szerkesztői:

Hajnal Csaba (HCs)

DevilSoft (Devil)

Címlap:

Németh Ferenc

Technikai szerkesztő:

Bartha István

A szerkesztőség és a kiadó címe:

8000 Székesfehérvár

Dózsa György tér 10.

Telefon: (22) 12 - 619

Telefax: (22) 11 - 585

Levélcím:

ENTERPRESS

1399 Budapest

PF. 701/334

Lapunk az ENTERPRISE Computers
GmbH kelet-európai képviselőjének, a
VTGe Electronics Ltd.-nek szakmai
támogatását élvezi.

Nyomja a

VIDEOTON Nyomda

Székesfehérvár

Felelős vezető:

Gombaszögi József

ISSN 0866-1820

Terjeszti a

Magyar Posta

Előfizethető a HELIR Bp. 1900 címen

Előfizetési díj

egy évre 294 Ft, fél évre 147 Ft

Ára: 49,- Ft

Következő számunk

július 20-án jelenik meg

Az ENTERPRESS-ben közreadott információk célja az, hogy segítsék, tudnivalókkal lássák el az ENTERPRISE számítógépek felhasználóit. A közölt programokat, kapcsolási rajzokat és leírásokat mindenki szabadon felhasználhatja, de tilos azokat a kiadó írásbeli engedélye nélkül másolni, terjeszteni.

A szerkesztőség kéziratokat nem őriz meg, és nem küld vissza.

Assembly

5. rész

Legutóbb nagy általánosságban megismerkedtünk az AS-MON-nal, mint komplett assembler/editor/debugger programmal. Ebben a fejezetben egy ténylegesen használható leírást kapunk a rendszerről.

Az ASMON kétféle változatban terjedt el: az egyikben autostartos alkalmazói programként, a másikban rendszerbővítőként inicializálódik. Jelen leírás az előbbi 1.3-as változatára vonatkozik, igazán lényeges eltérés nincs a különböző verziók között.

A betöltést követően megjelenik az ASMON képernyője, amely három főrésze tagolódik:

4015	C6	FC	ADD	A,FC	;F
4017	C6	0E	ADD	A,0E	;F.
4019	C7		RST	00	;G
401A	8E		ADC	A,(HL)	;
401B	C7		RST	00	;G
401C	18	C8	JR	3FE6	;H
401E	5E		LD	E,(HL)	;^
401F	CD	3A C9	CALL	C93A	;M:l
4022	5F		LD	E,A	;
4023	C9		RET		;l
4024	2C		INC	L	;
4025	CB	3F	SRL	A	;K?
4027	CB	40	BIT	0,B	;K@
4029	CB	3D	SRL	L	;K=
402B	CB	3E	SRL	(HL)	;K
402D	CB	D3	SET	2,E	;KS
PRINTER:OFF-					
Command >					
INPUT:ASCII-					
AF=00FF	MZ-H-ENC	..			
BC=0000	FF FF FF FF	00 00 00 00	00 00		
DE=0000	FF FF FF FF	00 00 00 00	00 00		
HL=0000	FF FF FF FF	00 00 00 00	00 00		
IX=0434	00 00 00 00	70 02 11 0F	6F 00		
IY=0000	FF FF FF FF	00 00 00 00	00 00		
SP=0200	04 00 BF 05	C3 00 C0 C3	55 01		
PC=1000	00	NOP			

A felső, legnagyobb ablakban 16 sorban jeleníthető meg a kód listája, a memóriatartalom (vagy népszerűbb nevén a memóriadump), itt végezhetjük el a memóriatartalom módosítását, ide fródnak a rendszer üzenetei stb.

A két csík által bezárt középső sort parancssornak nevezhetjük. A "Command >" promptnál nyomjuk le az egyes parancsokhoz tartozó billentyűket, és a szükséges további paramétereiket is ebben a sorban adjuk meg. A sor vége feletti "PRINTER:" felirat tájékoztat a nyomtatási státuszról, a sor alatti "INPUT:" pedig azt árulja el, hogy hexadecimális, decimális vagy éppen karakteres formában kell megadnunk a paramétereiket az illető parancsnak.

A legelső ablak bal szélén a regiszterek tartalmát látjuk. A legelső sorban az F regiszter bitjeinek állását vizsgálhatjuk. Ha valamelyik bit értéke 1, akkor jelenik meg a hozzátartozó betű.

A BC, DE, HL, IX, IY, SP regisztertartalmak mellett tíz értékből álló dumpot látunk. Ez a dump úgy képződik, hogy a rendszer az adott regiszter által címzett memóriarekesszel szomszédos négy alsóbb és öt felsőbb című rekesz tartalmát is kijelzi. Ez a lehetőség főleg az indexregiszterekkel, veremkezeléssel kapcsolatos vizsgálatoknál nagyon hasznos.

A legelső sorban a PC értéke mellett az éppen megcímezett utasítást láthatjuk.

A debugger

A debugger viszonylag sokfajta lehetőséggel rendelkezik: itt végezhetjük a program fordítását, lépésenkénti végrehajtását, futtatását; kezelhetjük a portokat, vájkálhatunk a tárban stb. A fő feladata tehát, hogy segítsen programunkat futtatható formába önteni, a hibákat kibogarázni (nem véletlenül hívják debugger-nek, a "bug" ugyanis bogarat jelent).

Az ASMON összes parancsa a megfelelő billentyű leütésé-

vel aktivizálható. A parancsok listáját a [H] (Help, segítség) billentyű lenyomásával kapjuk meg. A parancsok paraméterként számokat és szövegeket is várhatnak. A számokat hexadecimális vagy decimális formában adhatjuk meg, a számrendszerek között a [CTRL]+[F8] billentyűkkel válthatunk. Ez a billentyű-kombináció egyébként a szükséges helyeken mindig működik, ugyanúgy, mint az [ESC], amellyel bármilyen helyzetből, bármelyik helyről visszaléphetünk a parancssorba.

Sok opció alapértelmezéssel rendelkezik, így ha az ott szereplő érték számunkra megfelelő, akkor nincs más dolgunk, mint hogy az [ENTER]-t leüssük.

A következőkben az angol gépeknél megjelenő help szerinti sorrendben kerülnek ismertetésre a funkciók. A német gépeken csak annyi az eltérés, hogy néhány parancs más billentyűre indul el. Ezen különbségekről a [H] lenyomásával szerezhetünk tudomást a német gépek tulajdonosai. (Sajnos, szerkesztőségünknek egyetlen német gépe sincsen. A szerk.)

[A] Assemble : a fordítási opcióknak megfelelően elindul az editorban lévő forráslista fordítása.

[B] Breakpoints : két töréspont ("Breakpoint 1:" és "Breakpoint 2:") beállítása. Ha a program futás közben a törésponthoz jut, akkor megáll a végrehajtása, így ismét az AS-MON-hoz kerül a főszerep.

[C] Copy memory : memóriatartalom másolása a "Start:"-tól az "End:"-ig a "Destination:" cílcímre.

[D] Dump memory : memóriatartalom megjelenítése a "Start:" címtől.

[E] Edit source : belépés az ASMON editorba.

[F] Fill memory : a memória feltöltése a "Start:"-tól az "End:"-ig a "Value:" értékkel.

[G] Go : a lefordított program indítása a "Start:" címtől.

[H] Help : a helpüzenetek megjelenítése.

[I] Set function key : a "Key number:" sorszámú funkció-billentyűhöz a "Key string:" idézőjelek között megadott sztringet rendelhetjük. Az ASMON az editorba lépve elfelejti a megadott sztringeket, így ez a lehetőség használhatatlan.

[J] Swap register : a regiszterkészletek cseréje, tartalmuk kijelzése. Az alsó ablak jobb oldalán az "ALT-REGS" felirat jelenik meg.

[K] Read source : a "File name:"-nél megadott nevű forráslista beolvasása az editorba.

[L] Disassemble : a lefordított kód listázása a "Start:" címtől.

[M] Modify memory : a memóriatartalom módosítása a "Start:" címtől. A számmező és az ASCII mező között az [ALT]+[F8] billentyűkkel ugrálhatunk.

[N] Number cruncher : kiszámolja a "Value:"-nál megadott szám 16-os, 10-es, 2-es számrendszerbeli és ASCII megfelelőjét.

[O] Output port : a "Port:" portra a "Value:" értéket írja.

[P] Printer ON/OFF : bekapcsolt (ON) állapotban a megjelenítést a nyomtatóra irányítja.

[Q] Query port : megjeleníti a "Port:" port tartalmát.

[R] Read BIN file : a "Start:"-tól az "End:"-ig a "File name:" nevű tárgykód betöltése.

[S] Save BIN file : a memóriában a "Start:"-tól az "End:"-ig címig elhelyezkedő tárgykód elmentése "File name:" néven.

[T] Trace memory : a "Start:" címtől "Steps:" számú utasítás végrehajtásának lépésenkénti végrehajtása, nyomkövetése.

[U] Untrace : ez a furcsára sikeredett funkció elvileg ugyanaz, mint az előző, csak itt nincs kijelzés.

[V] View status : megjelenik az EXOS verziószáma; a CPU-ra lapozott szegmensek száma (Page0...Page3); a szabad szegmensek (Free segment(s)), az ASMON munkaszegmenseinek (Working segment(s)), az üzemképtelen szegmensek (Defective segment(s)) és az összes szegmens (Total RAM segment(s)) száma; a copyright szöveg.

[W] Write source : a forráskód mentése az editorból a "File name:"-nél megadott néven.

[X] eXamine register : a "Registerpair:"-nél megadott regiszterpár (AF, BC stb.) értéke a "Value:" lesz.

[Y] **Options disassembler** : a listázáskor megjelenő memóriacímekhez hozzáadhatjuk a "Memory offset"-nél megadott értéket, a címek kiírását az "Addresses" opcióval tilthatjuk/engedélyezhetjük (a "NO/YES" válaszokat bármelyik billentyű leütésével változathatjuk).

[Z] **Options assemble** : "Assembly listing" - fordítás közben listázza-e forráslistát, vagy sem; "List conditions" - fordítás közben listázza-e a feltételeket, vagy sem (a feltételekről később); "Force Pass 2" - az első menet esetleges hibáitól függetlenül végrehajtódjék-e a második fordítási menet, vagy ne; "Memory assembly" - memóriába fordítson, vagy ne (ha a "Memory assembly"-re "YES"-t válaszolunk, akkor a "Memory offset"-tel a forráslistában megadott fordítási címtől eltérhetünk); "Object file name:" - a tárgykód neve. Ha nem adjuk meg a tárgykód nevét, akkor nem tesz fel több kérdést a rendszer, ellenkező esetben még a közvetvezőkre kell válaszolnunk: "EXOS module header" - készítsen-e az ASMÓN a tárgykód elé EXOS fejléctet, vagy sem; ha "YES"-t válaszolunk, akkor meg kell adnunk a típusát az "EXOS module type:"-nál.

[I] **Options printer** : "Output to" - a listázást az editorba (EDITOR), vagy a nyomtatóra (PRINTER:) irányítja (ha PRINTER:-t választunk, akkor a "Use:"-zal akár fájlba is listázhatunk); "Return sends" - a listázott sorok végére CR/LF (Carriage Return, kocsivissza; Line Feed, soremelés) karakterek, vagy csak CR kerüljön; "Lines:" - egy oldalra kerülő sorok száma; "Chars.:" - karakterek száma egy sorban; "Left margin" - a bal margó mérete karakterben; "Bottom margin" - az alsó margó mérete karakterben; "Header:" - a lap tetejére kerülő fejléc szövege.

[N] **Options editor** : az editorban tárolt forráslista az "Editor buffer start:"-tól az "Editor buffer end:"-ig tart.

[L] **Load module** : a "File name:"-nél megadott nevű EXOS rendszerbővítő betöltése.

[@] **Symbol table** : a szimbólumtábla tartalmának megjelenítése.

[=] **Find string** : a "Start:" címtől a "Search:"-nál idézőjelek között megadott sztring keresése. Ha idézőjelek nélkül számokat sorolunk fel, akkor a számsorozatot keresi.

[:] **EXOS string** : parancs küldése az EXOS-nak.

Az editor

Az editorba az [E] leütésével léphetünk. Az editor gyakorlatilag minden olyan dolgot tud, ami fontos. Az editorba lépve az alsó két sorban a funkcióbillentyűkhöz rendelt lehetőségeket láthatjuk.

[F1] **Find** : Keresés

Find string : sztring keresés. Az "F" után beírt szöveget keresi a kurzor aktuális helyétől a szöveg vége felé haladva.

Find & repl : keresés és helyettesítés. Az "R" után először a keresendő-helyettesítendő, a ">" után a helyettesítő sztringet kell megadni.

Cont search : a keresés folytatása.

Repl string : a megtalált sztringet kicseréli a megadottal.

Repl & cont : a helyettesítés után folytatja a keresést.

[F2] **Tab** : Tabulátorok

Set tabmark : új tabulátorpozíció lesz a kurzor aktuális helyén.

Res tabmark : törli azt a tabulátort, amelyen a kurzor éppen áll.

Clr tabmark : törli az összes tabulátort.

Stand mark : beállítja az alapértelmezésű tabulátorokat.

[F3] **Copy** : Másolás

Mark start : a blokk eleje a kurzor helyén lesz.

Mark end : a blokk vége a kurzor helyén lesz.

Copy block : az előzőleg kijelölt blokk másolása a kurzor helyére.

Del block : a kijelölt blokk törlése.

Print block : a kijelölt blokk printelése.

Clr marks : a blokkjelzők törlése.

[F4] **File** : Fájlműveletek

Load file : a "File name:"-nél megadott című forrásfájl betöltése az editorba.

Save file : a "File name:"-nél megadott néven elmenti az

editorban lévő forrásfájl.

Save block : a "File name:"-nél megadott néven elmenti a kijelölt blokkot. E funkció csak akkor működik jól, ha az editor kurzora a blokk előtt van.

Kill text : törli az editor tartalmát, ha a "Delete text Y/N" kérdésre [Y]-t válaszolunk.

[F6] **Free** : Szabad helyek

A "Text" után a forráslista hosszát, a "Free" a még az editorba írható karakterek számát jelenti bájttban.

[F7] **Help** : help az editor lehetőségeiről

A szerkesztő billentyűk a szokásos funkciókat valósítják meg, ezért ezek külön nem érdemes felsorolni. A help nem tartalmazza az [ALT]+[JOYUP] és az [ALT]+[JOYDOWN] lehetőségeket, amelyekkel egyenesen a forráslista elejére/végére ugorhatunk. Az editornak egyetlen (óriási!) hiányossága, hogy nem tud folyamatosan beszűrűsös (INSERT) üzemmódban dolgozni.

[F8] **Exit** : Kilépés az editorból

A billentyű leütése után visszajutunk a debugger parancsorába.

Az assembler

Az assembler feladata, hogy az editorban megírt forráslistát lefordítsa, futtatható kódot generáljon. A minél kényelmesebb programozói munka érdekében a fordító sokfajta lehetőséget kínál: négyféle számrendszert használhatunk, a szimbólumokkal a fordítási időben műveleteket végezhetünk, támogatja a makrók használatát stb. Vegyük sorba ezeket az adottságokat!

Számrendszerek

Az assembler ugyanazokat a jelöléseket használja, mint amelyekkel a sorozat előző részeiben mi is alkalmaztunk: a számok végén egy-egy betűvel jelöljük a számrendszert. Ha nem adunk meg a szám után betűt, akkor az assembler az alapértelmezésnek megfelelő számrendszerben (i. .RADIX) értelmezi.

Decimális : a végén "d" betű, pl. 67d.

Hexadecimális : a végén "h" betű, pl. 0ff27h. Ha a szám alfanumerikus karakterrel (azaz betűvel) kezdődik, akkor a szám elé nullát kitenni, mert a fordító címkeként értékeli ki, és hibát fog jelezni.

Bináris : a végén "b" betű, pl. 11001001b.

Oktális : a végén "o" betű, pl. 73o.

Értékkadás másképpen is történhet:

Ha idézőjelek közé tesszük egy karaktert, akkor annak ASCII megfelelőjét fogjuk megkapni, pl. LD B,"A", amely az LD B,041h-val megegyező.

Szimbólikus jelölés is használható. A szimbólum egy, a fordítás idején létező értéket jelképez a fordítónak. Legyen például a "CHAR" szimbólum értéke 041h. Ekkor az LD B,CHAR hatására a fordító az LD B,041h kódot fogja előállítani.

Az ún. **cím számláló értékét** is megkaphatjuk, ha a "\$" jelölést használjuk. A fordítás indításakor meg kell adnunk a fordítás kezdőcímét, erre fog kerülni az első utasítás. A cím számláló értéke a fordítás során az utasítások hosszának megfelelően állandóan növekszik, mindig annak a tárrekesznek a címét tartalmazza, amelyre az illető utasítás kerülni fog. Ez a lehetőség az EXOS-nál különösen hasznos, mert pl. az ún. escape szekvenciák elküldésekor tudnunk kell azok hosszát.

A szimbólumokkal végezhető műveletek

Legyen a két szimbólum A és B, ekkor a következő műveletek végezhetőek el:

A+B összeadás

A-B kivonás

A*B szorzás

A/B osztás

A MOD B maradék

A AND B ÉS művelet

A OR B VAGY művelet

A XOR B KIZÁRÓ VAGY művelet

A SHL B A bitjei B helyel balra mozognak

A SHR B A bitjei B helytel jobbra mozognak
NOT A a biteket negálja (egyes komplementum)
-A kettes komplementum képzése
LOW A az alacsony bájt képzése
HIGH A a magas bájt képzése

Használatok még az

A < B, A > B, A < = B, A = > B, A = B, A < > B relációk.

Ezek után lássuk a fordítást vezérlő utasításokat, az ún. *assembler direktívákat*. A direktívákat a sor elejénél legalább egy szóköznyi (célszerűen egy tabulátorny) helytel beljebb kell írni; a szimbólum- és a címkeveket viszont mindig a sor elejére kell tennünk, máskülönben a fordító utasításként próbálja értelmezni.

ORG n A fordítás az "n" címtől fog kezdődni, minden programot ezzel kell kezdeni, pl. **ORG 0C00Ah**.

.PHASE n Ez a direktíva tulajdonképpen egy második címszámlálót indít el az "n" címtől, a fordítás természetesen erre az új címre történik. Párhuzamosan az elsődleges címszámláló értéke is megfelelően növekszik.

.DEPHASE Ismét az elsődleges címszámlálót lesz a főszerep.

DEFB n Az "n" bájtot a címszámláló által meghatározott tárrekeszbe teszi (DEFine Byte, bájt definiálás).

DB n L. DEFB n.

DEFW n Az "n" szót a címszámláló által megcímzett helyre teszi (DEFine Word, szó definiálás).

DW n L. DEFW n.

DEFS n A kódban "n" bájtnyi helytel üresen hagy (DEFine Storage, "tárolóhely" definiálás)

DS n L. DEFS n.

DEFM "text" Az idézőjelek között álló szöveget letárolja (DEFine Message, "üzenet" definiálás).

sybm EQU value Egy adott szimbólumot egyenlővé tehetünk egy értékkel, pl. az **esc EQU 01Bh** után az "esc" szimbólum 27-el lesz egyenértékű (EQUAl, egyenlő).

INCLUDE filename Az "filename" nevű forráslistát lefordítja, a kódhoz illeszti. Az "inkludolást" nagyobb programok készítésekor szokás használni. Ilyenkor a már jól működő részeket külön fájlba mentjük, és a szükséges helyeken betöltjük őket. Így az editorban mindig csak az aktuális, fejlesztés alatt álló rutin van. Ez a direktíva akkor is nagyon hasznos, ha a forráslistánk teljes mérete meghaladja az editorba tölthető 46KB-ot. Nagy jelentősége van a makrókönyvtárak aktivizálásánál is, l. alább.

END A forráslista és ezzel együtt a fordítás végét jelzi.

mname MACRO parameter(s) A makródefiniáció kezdetét jelöli. A makró egy olyan paraméterezhető, csak a fordítás idején létező utasításegyüttes, amely csak akkor kerül be a kódba, ha a nevével hivatkozunk rá. A makró definiálása tehát önmagában még nem eredményez semmiféle kódot. A makró és a szubrutin között az a különbség, hogy a szubrutin a kódban csak egyszer fordul elő, és ezt az egy rutint hívjuk meg távolról; a makró viszont minden esetben lefordításra kerül, ha hivatkozunk rá, így a makróban szereplő utasítások több helyen is szerepelnek majd a kódban. Egy példa a makró definiálásra és használatra:

; először a definíció

wrchr

MACRO @char

ld a,070h

ld b,@char

rst 030h

defb 2
endm

; majd a használat

org 01000h
wrchr "A"
end

A példában egy olyan makrót készítettünk, amely bemeneti paraméterként a kiírandó karakter ASCII kódját várja, majd ezt a megfelelő EXOS hívással kiteszi az ASMON debugger képernyőre. A tulajdonképpeni programban a makró nevét ("wrchr") a paraméter, azaz egy "A" betű követi, ez fog megjelenni. Ha megvizsgálunk a generált kódot, akkor azt látnánk, hogy az ugyanaz, mint a makró (mi más is lehetne..?), csupán az "ld b,@char" helyére "ld b,041h" került.

Az ASMON egyik súlyos hiányossága, hogy nem támogatja a feltételes paraméterátadást.

ENDM A makró végét jelzi, l. az előző példát.

EXOS n Egy előredefiniált, az operációs rendszer hívását támogató makró. "n" a hívni kívánt funkció számát jelenti. Az előbbi példában az "rst 030h" és a "defb 7" sorok helyett "exos 7"-t is írhattunk volna.

EXITM A direktívát az alább ismertetendő feltételes fordítást vezérlő utasításokkal együtt arra használhatjuk, hogy fordítás alatt kiugorjunk a makró belsejéből (**EXIT** Macro, kilépés a makróból).

IF kifejezés

utasítások1

ELSE

utasítások2

ENDIF Az IF..ELSE..ENDIF szerkezettel feltételes fordítást végezhetünk. Ha a "kifejezés" igaz, azaz értéke egy, akkor az "utasítások1", egyébként az "utasítások2" lesz lefordítva. Az ELSE ág elhagyható.

IFT Ua., mint az IF (IF True, ha igaz).

IFF Az IF-től annyiban különbözik, hogy az "utasítások1" a feltétel hamissága esetén kerül lefordításra (IF False, ha hamis).

IF1 A fordítás első menetében igaz.

IF2 A fordítás második menetében igaz.

.SFCOND A fordítási feltételek listázását kapcsolja be.

.LFCOND A fordítási feltételek listázását kapcsolja ki.

.TFCOND A fordítási feltételek listázását kapcsolja be/ki. E három direktíva a debugger "Options Assemble/List conditions" részét utánozza.

.LIST E direktíva már a listázás vezérlésére vonatkozik. A direktíva hatására fordításkor bekapcsolódik a listázás.

.XLIST Kikapcsolja a listázást.

.PRINTX text A "text" szöveget beleírja a fordítási listába. Ha a szöveg első karaktere a "%" jel, akkor a jelet követő kifejezés értékét jeleníti meg.

TITLE str Nyomtatásnál a lap fejlécébe kerül az "str" sztring.

.RADIX n Az alapértelmezésű számrendszer "n" lesz.

Az ASMON saját kódjában szerepelnek a relokálható állományok fordításához szükséges **INI0FS**, **SETRTP**, **RESRTP**, **ASEG**, **CSEG** stb. direktívák is. Mivel azonban a rendszer képtelen korrek relokálható fájlokat készíteni, ezek a direktívák használhatatlanok.

(folytatjuk)

-HCs-

Fizessen elő a

Hobby Elektronika és a Rádiótechnika

folyóiratokra! Így biztosan mindig hozzájut!

A cím: 1387 Budapest, Pf. 603. Tel.: 117 - 0262

A szerkesztőségben regisztrált HE előfizetőknek ingyenes nyák-film melléklet.

A PASCAL

5. rész

A halmaz (set) típus

A matematikában a halmaz valamilyen szempontból azonos jellemzőjű objektumok összessége. A Pascal nyelvben is bevezették a halmaz típust; ez egy adott - megszámlálható - típushoz tartozó értékek együttesét jelenti.

Miért van erre szükség? Mindenki, aki készített már komolyabb programot, találkozott azzal a problémával, hogy meg kell állapítani egy érték hovatartozását. Numerikus értékeknél legtöbbször nincs is gond, egyszerűen megvizsgáljuk, hogy az érték belesik-e az előírt tartományba:

```
IF ( ERTEK >= ALSO_HATAR ) AND ( ERTEK <= FELSO_HATAR ) THEN
```

Egy fordítóprogram készítésekor azonban olyan feltételeket kell vizsgálni, hogy az adott karakter pl. betű-e vagy szám, esetleg frásjel. Szerencsére az ASCII kódok esetében a betűk egy (a kis- és nagybetűket külön kezelve két) folyamatos kódtartományba tartoznak, így a vizsgálat megoldható a fenti módszer kiterjesztésével. Eláruljuk, hogy ez nem minden kódrendszerben van így - az IBM nagygépeken használt EBCDIC (ejtsd "ebszidik") kódkészlet sem ilyen -; az frásjelek viszont valóban össze-vissza helyezkednek el a kódtartományban.

A Pascal az ún. halmazgenerátor segítségével generálni tud az alaptípusának - egy megszámlálható típusnak - az elemeiből egy halmazt (a halmazgenerátor programbeli alakja egy szögleteszárjel-pár, amelyben felsoroljuk vagy folytonos tartományként megadjuk az alaptípusnak azokat az elemeket, amelyeket az adott halmazhoz akarunk rendelni). Lássunk erre először pár példát. Itt most halmazokat generálunk egy fordítóprogram számára:

```
CONST
  TAB = CHR( 9 ); { HiSoft Pascal; TURBO-
                  Pascalnál TAB = #9 };
  PONT = '.';
  VESSZO = ',';
  PONTOSVESSZO = ':';

VAR
  KISBETU, { ezeket }
  NAGYBETU, { a változókat }
  BETU, { karakter- }
  SZAMJEGY, { halmaznak }
  HEXA_SZAMJEGY, { deklaráltuk }
  ELVALASZTOJEL : SET OF CHAR;

...

BEGIN
  KISBETU := [ 'a' .. 'z' ]; { a kisbetűk halmaza
                             a-tól z-ig }
  NAGYBETU := [ 'A' .. 'Z' ]; { a nagybetűk
                               halmaza A-tól Z-ig }
  BETU := KISBETU + NAGYBETU; { a két
                               előző
                               halmaz
                               uniója }

  SZAMJEGY := [ '0' .. '9' ];
  HEXA_SZAMJEGY := SZAMJEGY + [ 'A' .. 'F' ];
  ELVALASZTOJEL := [ PONT, VESSZO, PONTOSVESSZO,
  TAB ];
END;
```

Ezután gyerekjáték megállapítani, hogy egy karakter beletartozik-e

valamelyik halmazba:

```
IF C IN BETU THEN ...
```

```
IF C IN ( BETU + SZAMJEGY ) THEN ...
```

Az IN halmazművelet, értelemszerűen, igaz eredményt ad, ha egy érték beletartozik az adott halmazba. A második programsorban a betűk és a számjegyek közös halmazába tartozást vizsgáltuk.

Természetesen konstansként is definiálhattuk volna ugyanezeket a halmazokat:

```
CONST
```

```
KISBETU = [ 'a' .. 'z' ];
```

```
stb.
```

A halmazok között megengedett az értékadás; ha A és B halmaz típusú, akkor ez

```
A := B
```

alakban írható. A természetesen változó, B tetszőleges - halmaz típusú - kifejezés lehet.

```
Az
```

```
A + B
```

kifejezés, mint már láttuk, a két halmaz unióját - közös halmazát - jelenti (vagyis azokat az elemeket, amelyek akár az A, akár a B halmazba tartoznak); az

```
A * B
```

a két halmaz közös részét - intersekciónak - (vagyis azokat az elemeket, amelyek egyidejűleg mind a két halmazhoz tartoznak); az

```
A - B
```

a két halmaz különbségét (A-nak azt a részhalmazát, amelyik nem tartozik ugyanakkor B-hez is).

Vizsgálhatjuk két halmaz egyenlőségét, azaz, hogy ugyanazokat és csak ugyanazokat az elemeket tartalmazzák-e:

```
A = B
```

két halmaz egyenlőtlenségét, azaz, hogy nem tartalmazznak azonos elemet:

```
A <> B
```

és azt, hogy az egyik halmaz részhalmaza-e a másiknak:

```
A <= B
```

vagy megfordítva, a másik az egyiknek

```
A >= B
```

Írjuk még fel az üres halmazt:

```
[ ]
```

Ez nyilván egy olyan halmaz, amelynek nincs egyetlen eleme sem. Szóljunk pár szót a halmaz típus gépi ábrázolásáról. Egy adott alaptípusból képzett összes lehetséges halmazban az összes elemnek csak egy tulajdonságát használjuk ki, mégpedig, hogy beletartozik-e az adott halmazba vagy sem; így az elemek ábrázolásához elegendő egy-egy bit. Például, a karakter alaptípusból képzett összes halmaz egy 256 bites (32 bájtos) mezővel ábrázolható, ahol az egyes bitpozíciókon 1 jelenti, hogy az adott karakter eleme a halmaznak, és 0, ha nem eleme. Így az üres karakterhalmaz gépi ábrázolása

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
... (még 24 x 8 nulla)
```

ahol a bal szélső nulla a 00 hexa kódú karaktert (pontosabban annak hiányát) jelenti, az utolsó - a példában már nem ábrázolt - nulla pedig az FF hexa kódú karakterhez tartozik.

A csak a számjegyeket tartalmazó halmaz gépi ábrázolása pedig

```
00000000 00000000 00000000 00000000 00000000 00000000 11111111 11000000 ...
(itt az első egyes a 0, az utolsó a 9 számjegyre tartozik). Az elválasztójeleket tartalmazó halmaz ábrázolása pedig (lásd a fenti példát)
```

```
00000000 01000000 00000000 00000000 00000000 00001010 00000000 00010000 ...
```

az összes karaktert tartalmazó halmazé pedig

```
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111 ...
```

Mindezzel persze nem kell törődnünk, a fordítóprogram megcsinálja helyettünk a leképezést.

A fájl (magyarul állomány) típus

A fájl típus ugyanúgy azonos típusú elemekből áll, mint a tömb

típus, azonban azzal ellentétben az elemeinek száma nem meghatározott. Az elemek sorban egymás után következnek, és egyszerre csak egy elemhez férünk hozzá. Ezután már nem meglepő, hogy a fájl típus megvalósítása a köznapi fájl fogalommal azonos módon történik, azaz a fájl típus valamilyen háttértárolón (mágnesszalag, mágneslemez) vagy periférián (terminal, modem) megjelenő adatok kezelésére szolgál. A file típus elemei tetszőleges típusúak lehetnek, azaz pl. karakter, egész vagy valós, rekord vagy tömb. A karakter-fájl (FILE OF CHAR) mellett a Pascal kényelmi okokból megkülönböztet ún. szövegfájl (TEXT), amely ugyan karakterekből áll, de benne a karakterek szövegsorokat alkotnak.

Ha deklarálunk egy fájl típusú változót:

VAR

F : FILE OF INTEGER;

akkor automatikusan létrejön egy F^{\wedge} változó is, az adott fájl pufferváltozója. Ez, mint egy ablak, mindig megmutatja a fájl valamelyik elemét. Az "ablak" végig tud lépkedni a fájl elemein, így tudunk ezekhez hozzáférni. Ha a fájl utolsó elemén is túmentünk, a pufferváltozó értéke meghatározatlan.

Minden fájlhoz tartozik még egy logikai típusú függvény is, az

EOF (F)

függvény (end of file), amely igaz értéket ad, ha a pufferváltozó a fájl végén túla került.

A fájl típus kezeléséhez négy alapvető eljárást használunk.

RESET (F) Egy létező fájl elejére állítja a pufferváltozót, elkezdhetjük (újra)olvasni a fájl tartalmát;

REWRITE (F) Egy nemlétező fájl létrehoz, vagy egy létező fájl töröl, és az elejére állítja a pufferváltozót, elkezdhetjük a fájl írását;

GET (F) Egy elemmel továbblépteti a pufferváltozót a fájlban, a következő elemet tudjuk olvasni;

PUT (F) Egy elemmel továbblépteti a pufferváltozót a fájlban, a következő elemet tudjuk írni.

Maga az írás és az olvasás a pufferváltozó segítségével végezhető, azaz pl.

$F^{\wedge} := 34$; PUT (F);

írás esetén, és

V := F^{\wedge} ; GET (F);

olvasás esetén (itt a V változót korábban ugyanolyan típusúnak kellett deklarálni, mint az F file elemeinek típusa).

Bár ezekkel az eljárásokkal mindenfajta fájlművelet elvégezhető, a Pascal - megint csak kényelmi okokból - újabb két eljárást vezet be:

READ (F, V)

amely a V változóba teszi a pufferváltozó értékét, majd továbblép a fájlban:

WRITE (F, E)

amely az E kifejezés értékét a fájl végére írja, majd ugyancsak továbblép a fájlban.

További kényelmeit nyújt az, hogy a READ és a WRITE eljárásnak több paramétere is lehet, így egyetlen eljáráshívással több értéket is olvashatunk vagy írhatunk.

READ (F, V1, V2, ... Vn)

és

WRITE (F, K1, K2, ... Kn)

A szövegfájlok

Bár a szöveget, azaz karakterek sorozatát tartalmazó fájl definíciójában

FILE OF CHAR

alakban, a Pascal mégis bevezeti a speciális szövegfájl (TEXT) fogalmát. Ez abban különbözik a FILE OF CHAR típustól, hogy feltételezi azt, hogy a szöveg sorokra tagolódik. Ezt a sorokra tagolódást a Pascal helvettünk kezeli, így sokkal egyszerűbb a szövegsorok feldolgozása.

Ennek megvalósításához a Pascal bevezet két új eljárást és egy függvényt:

READLN (F) A beolvasás alatt lévő fájl következő sorára pozícionálja a pufferváltozót;

WRITELN (F) Lezárja az eddig írt szövegsort. Az ezután kiírt karakterek új sorba kerülnek majd.

Kényelmi okokból a READLN és a WRITELN eljárásnak paramétere is lehetnek, ezekre ugyanaz érvényes, mint a READ és WRITE eljárásparamétereire; így a beolvasás (kiírás) és az új sorra állás egyetlen utasítással elvégezhető. A sorváltás mindig az összes paraméter beolvasása (kiírása) után történik.

Az

EOLN (F)

(end of line) függvény TRUE értéket ad, ha a pufferváltozó a sor végére mutat a beolvasott fájlban.

A mutató (pointer) típus

Az összes eddig tárgyalt adatszerkezet közös vonása, hogy statikusak: valahol, egy blokkban deklaráljuk őket, ezzel "létrejönnek", és mindaddig "élnek", amíg a programvégrehajtás ki nem lép az adott blokkból. Szükség lehet azonban olyan adatokra, amelyeknek a mennyiségét, struktúráját nem ismerjük a program írása idején; ezeket menet közben, a program végrehajtása során kell létrehozni és esetleg megszüntetni. Ezek a dinamikus változók.

A dinamikus változókra nem lehet név szerint hivatkozni, hiszen nem szerepelnek semmilyen deklarációban. Kezelésükhöz a Pascal a mutató (angolul pointer) típust használja. A mutató típusú változó, értelemszerűen, "rámutat" a dinamikus változóra; maga a dinamikus változó tetszőleges típusú lehet.

A

TYPE P = \wedge T

deklaráció a P típust olyan mutató típusnak deklarálja, amelyik valamilyen T típusú (dinamikus) változóra mutat.

Deklaráljunk most egy ilyen P típusú változót, tehát egy mutatót:

VAR MUTATO : P

Ha MUTATO értéket kapott, akkor valamelyik T típusú elemre mutat. Lehetséges még egy olyan eset, hogy a mutató éppen nem mutat semmire: értéke ekkor a NIL (magyarul semmi) nevű speciális konstans (ez gyakorlatilag ugyanaz, mint az egész vagy valós változó 0 értéke).

Hogyan tudunk most a mutatott, a dinamikus változóra hivatkozni? A mutatón keresztül:

MUTATO \wedge

jelentése az a (T típusú) változó, amelyre MUTATO éppen most mutat.

A dinamikus változók létrehozására a NEW eljárás szolgál; a

NEW (MUTATO)

eljáráshívás létrehozza a T típusú dinamikus változót (helyet foglal számára a memóriában), és a MUTATO változónak az új dinamikus változóra mutató értéket ad. A

DISPOSE (MUTATO)

eljárás megszünteti azt a dinamikus változót, amelyikre a MUTATO az adott pillanatban éppen mutat.

Ha az A és B változó - a dinamikus változó típusával egyezően - T típusú, akkor az

A := MUTATO \wedge

értékdadás A-ba tölti a dinamikus, mutatott, változó pillanatnyi értékét, a

MUTATO \wedge := B

értékdadás pedig a dinamikus változóba tölti B értékét.

A dinamikus változókka számtalan érdekes dolgot lehet csinálni. Az egyik ilyen az adatrekordok "felfűzése". Mivel a dinamikus változó tetszőleges típusú lehet, senki sem akadályozhat meg bennünket abban, hogy a mutató egy olyan rekordra mutasson, amelyik önmaga is egy (vagy több) ugyanilyen mutatót tartalmaz. Ha mindegyik rekordban a mutató valamelyik másik rekordra mutat, a mutatókon "lépkelve" akkor is végigmenhetünk a rekordok láncolatán, ha azok fizikailag nem sorrendben, hanem össze-vissza helyezkednek el:

P0 → P1 → Pn → NIL

Az utolsó rekord mutatója nem mutat sehová (értéke NIL), hiszen nincs több elem a láncban.

Ez azt jelenti, hogy ha egy új rekordot hozunk létre egy rendezett láncban, akkor azt úgy tudjuk a "helyére" tenni, hogy nem kell az összes rekordot átrendezni, az új elemet egyszerűen "beáncsoljuk" a megfelelő helyre: az előző elem mutatóját ráállítjuk a beszűrt elemre, a beszűrt elem mutatóját pedig a következő elemre.

A másik érdekes lehetőség, ha a rekord több mutatót is tartalmaz. Ekkor ugyanazokon a rekordokon többféle láncot követhetünk végig: pl. egy könyvtári adatbázisban az egyik lánc mentén a szerző neve szerint, a másik lánc mentén a cím szerint, egy harmadikon a kiadás éve szerint lehet "rendezve" ugyanaz az adattömeg.

Ilyen pointeres, mutatós adatszerkezettel lehet ábrázolni a számítástechnikában, de más területeken is közkedvelt fa-struktúrákat, vagy a matematikusok által szerett grafokat. Valós tolvamatok modellezésénél szinte elkerülhetetlen a dinamikus adatszerkezetek alkalmazása

Lehetőségek Páratlan Tárháza (LPT) 4

A legutóbbi részben már elkezdtem a beígért látványos rutinok bemutatását egy scroll-rutinnal. Most ezt folytatva két, demókból jól ismert látványelem megvalósítása kerül sorra. Mielőtt azonban a rutinok ismertetésébe kezdenék, szeretnék néhány szót szólni a színekről, merthogy erről még nem volt szó.

Valószínűleg mindenki előtt ismert tény, hogy a TV-technikában a színek három összetevőből - a vörös, a zöld és a kék színárnyalatok keverékéből - állnak. Ugyanezt a módszert használják a számítógépek is. Az ENTERPRISE 8 árnyalatot tud megkülönböztetni a vörös és a zöld színből és négyet a kékből. Ez alapján könnyen kiszámolható, hogy összesen $8 \cdot 8 \cdot 4 = 256$ színárnyalatot képes megjeleníteni. Ahhoz, hogy egy színek megállapíthatassak a kódját ismernünk kell a színkódok felépítését. Ez a következő:

b7 b6 b5 b4 b3 b2 b1 b0

G0 R0 B0 G1 R1 B1 G2 R2

Ez egy bináris szám, ahol R a vörös, G a zöld, B a kék színt jelenti, a betűk melletti szám pedig a színerősség bináris értékének helyiértéke. Például, ha 5-ös erősségű vöröset akarunk előállítani, akkor R2 R1 R0=101b vagyis a színkód 01000001b=81h. Ezek alapján már bármilyen szín kódját kiszámolhatjuk.

Ezután a kis kitérő után lássuk a két rutint! Az első egy hullámozgató rutin, amely a megadott LPT által definiált képet egy táblázatnak megfelelően hullámozgatja vízszintes irányban. Működése röviden a következő: a táblázat elemeinek megfelelő értékkel megváltoztatja az egymás után következő LPB-k videomemória-címét, az alapértelmezésbelihez képest. Ha a táblázat végére ért, akkor az első elemnél folytatja. A rutinban szereplő táblázat egy 32 elemű szinusz-tábla, ami könnyen megváltoztatható, így akármilyen alakzatnak megfelelő hullámozgást elérhetünk. Ha meg akarjuk változtatni a táblázat méretét, akkor a rutinban lévő mindkét "AND 1FH" utasítást operandusát át kell írni. Így csak kettő hatványainak megfelelő elemszámokat tudunk megvalósítani.

A második rutinnal háttérszínezést tudunk előállítani. Ezt általában raszternek hívják. Ha megnézzük a programcska működését láthatjuk, hogy először visszaállítja az összes LPB COL0 mezőjét a megadott háttérszínre, majd kirakja a megadott számú rasztert. A raszterek kirakásánál elsőként ellenőrzi, hogy nem érte-e el valamelyik megadott végállást. Ha igen akkor irányt vált, majd kiszámolja az új pozíciót, és sorban beállítja az LPB-k COL0 mezőjét a táblázatban megadott színekre. A rutinnal tetszőleges számú rasztert mozgathatunk, amelyekre külön-külön megadhatjuk az alsó és felső határpozícióját, sebességét, méretét és színeit.

Mindkét rutinnál alapvető követelmény, hogy a megadott LPT-ben minden LPB csak egy pixelsort definiáljon, mint ahogyan azt az előző részben szereplő scroll-rutinnal bemutattam. Ezenkívül a nemkívánatos villogások elkerülése érdekében ajánlatos videomegszakításból hívni őket.

-DEVIL-

	.RADIX 10H	
LPTADR	EQU 8000	;LPT Z80-AS CIME
LPTLEN	EQU 80	;LPB-K SZAMA
VIDADDR	EQU 4000	;VIDEOMEMORIA NICK-CIME
LINELEN	EQU 50	;EGY PIXELSOR HOSSZA BAJTOKBAN
WAWES	LD A,0	;BELEPESI PONT
	INC A	;HULLAMSZAMLALO
	AND 1FH	;NOVELESE ES
	LD (WAWES+1),A	;TAROLASA
	LD C,A	
	LD B,LPTLEN	;CIKLUSSZAMLALO
	LD DE,VIDADDR	
	LD HL,LPTADR+4	
NEXTLIN	PUSH BC	;CIKLUS KEZDETE
	PUSH DE	;REGISZTEREK
	PUSH HL	;MENTESE
	LD HL,WAWETAB	;TABLAZATBOL
	LD A,C	;C-EDIK ELEM
	ADD A,L	;KIOLVASASA
	LD L,A	
	ADC A,H	
	SUB L	
	LD H,A	
	LD B,0	
	LD C,(HL)	
	BIT 7,C	;HA NEGATIV
	JR Z,NONEGAT	;AKKOR KIVONAS
	LD B,OFFH	
NONEGAT	EX DE,HL	;VIDEOCIM
	ADD HL,BC	;ALLITASA A
	EX DE,HL	;TABLAZATNAK
	POP HL	;MEGFELELOEN
	LD (HL),E	;BEIRASA AZ
	INC HL	;AKTUALIS
	LD (HL),D	;LPB-BE
	LD BC,OFH	;KOVETKEZO LPB
	ADD HL,BC	;CIMMEZOJENEK

	POP DE	;SZAMOLASA
	EX DE,HL	;KOVETKEZO PIXELSOR
	LD BC,LINELEN	;ALAP VIDEOCIMENEK
	ADD HL,BC	;KISZAMITASA
	EX DE,HL	
	POP BC	
	LD A,C	;TABLAZATMUTATO
	INC A	;NOVELESE
	AND 1FH	;MODULO 32
	LD C,A	
	DJNZ NEXTLIN	;CIKLUS
	RET	
WAWETAB	DEFB 0,2,3,4,6,7,7,8	;32 ELEMU
	DEFB 8,8,7,7,6,4,3,2	;TABLAZAT
	DEFB 0,-2,-3,-4,-6	;AMELY ALAPJAN
	DEFB -7,-7,-8,-8,-8	;MEGY A HULLAM
	DEFB -7,-7,-6,-4,-3,-2	
LPTADDR	EQU 8000	;LPT Z80-AS CIME
RASTNUM	EQU 3	;RASZTEREK SZAMA
LPBNUM	EQU 0	;LPB-K SZAMA
BGRCOL	EQU 0	;HATTERSZIN
RASTER	LD B,LPBNUM	;BELEPESI PONT
	LD A,BGRCOL	;HATTERSZIN A-BA
	LD HL,LPTADDR+8	;ELSO LPB SZINCIME
	LD DE,10	;EGY LPB HOSSZA
BACKGR	LD (HL),A	;HATTERSZIN
	ADD HL,DE	;VISSZAALLITASA
	DJNZ BACKGR	;ALAPERTELMEZESRE
	LD B,RASTNUM	;CIKLUSVALTOZO BEALL
	LD HL,RDATA	;ADATTABLAZAT CIME
NRAST	PUSH BC	;CIKLUS KEZDETE
	LD D,H	;TABLAZATCIM
	LD E,L	;ATTOLTESE DE-BE
	LD A,(HL)	;AKTUALIS POZICIO
	INC HL	;KIOLVASASA
	LD B,A	
	CP (HL)	;FELSO VEGALLAS?
	INC HL	
	JR C,BACK	;UGRAS HA IGEN
	CP (HL)	;ALSO VEGALLAS?
	JR NC,BACK	;UGRAS HA IGEN
	INC HL	
	JR NOBACK	
BACK	INC HL	;IRANYVALTAS
	LD A,(HL)	
	NEG	
	LD (HL),A	
NOBACK	LD A,B	;AKTUALIS POZICIO
	ADD A,(HL)	;NOVELESE A
	INC HL	;SEBESSEGEL
	LD (DE),A	;UJ POZICIO
	EX DE,HL	;ELMENTESE
	LD L,A	;RASZTER ELSO
	LD H,0	;PIXELSORHAHOZ
	ADD HL,HL	;TARTOZO LPB
	ADD HL,HL	;CIMENEK
	ADD HL,HL	;KISZAMOLASA
	ADD HL,HL	
	LD BC,LPTADDR+8	
	ADD HL,BC	
	EX DE,HL	
	LD A,(HL)	;RASZTER MERETE
	INC HL	
NLINE	LDI	;RASZTER KIRAKASA
	EX DE,HL	
	LD BC,OFH	;KOVETKEZO LPB
	ADD HL,BC	;CIMENEK SZAMOLASA
	EX DE,HL	
	DEC A	
	JR NZ,NLINE	
	POP BC	;KOVETKEZO
	DJNZ NRAST	;RASZTER
	RET	;VISSZATERES
RDATA	DEFB 2,2,0EFH,1,0DH	;RASZTERADATOK
	DEFB 40,8,48,1,41	;SORRENDEN:
	DEFB 9,49,9,41,1	;AKTUALIS POZICIO
	DEFB 48,8,40	;FELSO VEGALLAS
	DEFB 70,10,0C0,2,15	;ALSO VEGALLAS
	DEFB 80,10,90,2	;SEBESSEG
	DEFB 2,82,82,12	;MERET
	DEFB 12,92,92,92	;A MERETNEK MEGFELEO
	DEFB 12,12,82,82	
	DEFB 2,2,90,10,80	;SZAMU SZINKOD
	DEFB 30,20,0E0,3,9	
	DEFB 20,4,4,24,24	
	DEFB 24,4,4,20	

A TANDEM Kereskedelmi és Szolgáltató Kft-nél előnyös áron vásárolható IBM kompatibilis számítógép-részegységeket, háttértárolókat, lemezeket és egyéb kiegészítőket. Cím: 1132 BUDAPEST, Visegrádi u. 6. Telefon: 112-8604

Az ismeretlen SPOKE és SPEEK

Olvasóink közül jónéhányan kifogásolták, hogy a közölt programok egy részében magyarázat nélkül használjuk a SPOKE és a SPEEK utasításokat. Kicsit értetlenül fogadtuk ezeket a kifogásokat, hisz nincs ezekben az utasításokban semmi különleges. Szeretnénk azonban elejét venni a további zsörtölődésnek, így most egy külön cikket szentelünk e témának. Csak remélni merjük, hogy ezek után már bátran használhatjuk e két igen ügyes utasítást.

Gépünk 128KB-nyi RAM-mal rendelkezik, ezt a memóriát a gép tervezői 16KB-os részekre, ún. *szegmensekre* osztották. A teljes RAM az alapkiéptésű gépben 8 darab ilyen szegmensre tagolódik. A prospektusokból megtudhatjuk, hogy a gép maximálisan 4MB-os memóriát tud kezelni, ebbe a ROM és a RAM szegmensek egyaránt beleértendők. A Z80-as CPU 64KB memóriát képes egyszerre kezelni, így a CPU-n egy időben négy szegmens lóghat. A CPU a *lapjain* látja a szegmenseket. A lapok mérete igazodik a szegmensek méretéhez, egy lap egy 16KB-os tartományt fed le. A laptartományok:

0.lap	00000-16383
1.lap	16384-32767
2.lap	32768-49151
3.lap	49152-65536

A szegmensek cserélgetését memórialapozásnak, röviden csak *lapozásnak* nevezzük. Fizikailag a lapozás egy meglehetősen komplex feladat, nem véletlenül szükséges hozzá egy külön, memórialapozó funkciót (is) ellátó kooprocesszor, a Dave. Basic-ből is van lehetőség a direkt memórialapozásra: a 176-179 portok tartalma dönti el, hogy a CPU adott lapján melyik szegmens van. A 176-os port a 0.laphoz, a 179-es port a 3.laphoz tartozik; az IN és az OUT utasításokkal e portok tartalmát szabadon olvashatjuk, írhatjuk. (A 0. és a 3.lapokat lehetőség szerint ne zaklassuk!)

Minden szegmensnek saját azonosítószáma van, ezt szokás *szegmensszámnak* (vagy nagyolós körökben "szegmenscímnak") nevezni. Például az EXDOS vezérlőprogramja a 32-es és a 33-as ROM szegmenseken helyezkedik el. Ha igazán jó programokat akarunk készíteni, akkor a szegmensek számozását és szerepüket pontosan kell ismernünk. Lehetetlenség lenne e cikk, de akár a teljes lap terjedelmében a gépben lévő szegmenseket, feladatukat ismertetni, így ettől eltekintünk. Itt most elegendő annyi ismeret, hogy az EXOS a RAM szegmenseket a 248-255 tartományba sorolja, és a 255-ös szegmens az ún. *rendszersegmens*. Az alapvető lehetőségeknél (pl. gépi kód, POKE, PEEK) tudnunk kell, hogy melyik szegmens és a szegmens belüli melyik címen lévő adattal akarunk műveleteket végezni, és hogy mindezt a CPU melyik lapján kívánjuk tenni.

A két utasítás (SPOKE, SPEEK) elsősorban a közvetlen memórialapozástól mentesít bennünket. Basicben tehát elegendő a szegmensszámot, és a szegmens belüli címet az ún. *ofszetcímet* tudnunk, a tár állapotának könyvelésével (melyik szegmens melyik lapon van stb.) nem kell foglalkoznunk.

A SPOKE (Segment POKE) paranccsal bájtot tudunk leírni egy adott szegmens ofszetcíme, az utasítás helyes formája: *SPOKE(szegmensszám,ofszet cím,érték)*

A szegmensszám 0 és 255, az ofszet cím 0 és 16383, az érték 0 és 255 közötti, illetve azzal egyenlő is lehet. Az ofszet cím helyére a megadottnál nagyobb értéket is felvehetünk, az utasítás csak a cím alsó 14 bitjét veszi figyelembe. Így ha például a SPOKE(200,0,0) helyett SPOKE(200,65536,0) frunk, ugyanazt az eredményt kapjuk.

Konkrét példaként színezzük át a képernyő első sorát a ha-

gyományos POKE, majd pedig a SPOKE utasítással!

Basic alaphelyzetben a sorparamétertábla (l. LPT sorozat) első COL0 bájtja a 47368-as címen van. Ha ekkor kiadjuk a

POKE 47368,255

parancsot, akkor a képernyő tetején egy fehér csík lesz látható. A sorparamétertábla mindig a 255-ös sorszámú szegmensben van, ez lesz tehát a szegmensszám. Az ofszetcímet megkaphatjuk, ha vesszük a cím 16384-es modulusát (osztás utáni maradékát), a MOD(47368,16384)-et. Ennek eredménye 14600, jöhet a

SPOKE 255,14600,255

parancs, az eredmény ugyanaz, mint fent. Van azonban egy igen lényeges különbség: ha valamilyen okból eltűnik a 2-es lapról a 255-ös szegmens, és mi kiadjuk a POKE 47368,255 parancsot, akkor egészen biztosak lehetünk abban, hogy rossz helyre ment a 255-ös érték. A POKE paranccsal egyébként is csfnján kell bánni, mert igencsak katasztrófális helyzeteket idézhetünk elő a rendszerben...

Ugyanez nem fordulhat elő a SPOKE-al! A ROM Basic ugyanis saját, belső memórialapozásával gondoskodik arról, hogy az érték garantáltan jó helyre kerüljön, megmentve a felhasználót ettől a feladattól.

A SPEEK (Segment PEEK) függvényel bájtot tudunk beolvasni egy szegmens adott címéről, a forma:

érték = SPEEK(szegmensszám,ofszet cím)

A szegmensszámra, az ofszetcíme, az értékre ugyanaz vonatkozik, mint ahogy azt fentebb megtudtuk.

Az előző példa folytatásaként olvassuk be a szín kódját, de a "látvány" kedvéért lapozzuk le a 255-ös szegmenst a 2-es lapról, adjuk ki:

OUT 178,0

Most teljesen közömbös, hogy miért épen a nullás szegmenst rakjuk a helyébe. A lényeg az, hogy ha a

PRINT PEEK(47368)

parancsot végrehajtja a Basic, akkor 252-öt fog kiírni, ami biztosan rossz, hiszen az imént 255-öt küldtünk oda (csak épen másik szegmensre). De a

PRINT SPEEK(255,14200)

után a helyes 255 jelenik meg a képernyőn.

A példa egy kicsit erőltetett, hiszen mi magunk tüntettük el a szegmenst, de ugyanezt az EXOS is teljes joggal megethette volna.

Ha valaki elsősorban Basic-ben írja programjait, akkor előbb-utóbb gyorsítani szeretné őket, ekkor veti be a Zzip Basic fordítót. Köztudott, hogy a Zzip ún. integer fordító, azaz a programban szereplő számok csak a -32768...+32767 tartományba eshetnek. Ha egy ilyen programba a minimál Basic POKE, PEEK parancsait használjuk, akkor nagy valószínűséggel az őket követő címértékek láttán kiakad a fordító. Ezt a problémát a SPOKE, SPEEK kiküszöböli.

A cikk közepe táján azzal büszkélkedtünk, hogy a két utasítást használva nincs szükségünk memórialapozásra. Ez így is van, de ne feledkezzünk meg a 16KB-nál nagyobb memóriatartományok kezeléséről sem! Tegyük fel, hogy megnyitottunk egy grafikus lapot, amelynek mérete 20KB, azaz két videosegmensben helyezkedik el. Ha ezt a 20KB-os tartományt kezelni akarjuk, akkor a műveletek előtt mindig el kell tudnunk dönteni, hogy az ofszet cím éppen melyik szegmensre vonatkozik.

Ha a közvetlen memóriakezelés vagy a SPOKE/SPEEK nyújtotta lehetőségek között kell döntenünk, akkor ugyanúgy járhatunk csak el, mint általában: a feladat pontos ismeretében, alapos mérlegelés után kell a megfelelő megoldást kiválasztanunk.

Színuszgörbe

A grafikával ismerkedők első dolga szokott lenni, hogy kirajzolják a színuszgömbét. Ezt teszi a listán látható program is, melyet érdemes kipróbálni a paraméterek változtatásával is.

```
100 PROGRAM "SINUS.BAS"
110 GRAPHICS HIRES 2
111 SET PALETTE 0,255
120 FOR X=0 TO 1279 STEP 8
130 PLOT X,350*SIN(X/50)+360;
140 NEXT
```

Színkeverés

Kedvenc masinánk 256 szín használatát teszi lehetővé. Ha egy-egy szöveges képernyőt ízlésszerű szeretnénk tenni, akkor a megfelelő színek kiválasztása alapvetően fontos. Ez a gépben lévő színekavalkádból meg lehetőséges körülményes, ezért célszerű a listán látható programot segítségül hívni. Nincs más dolgunk, minthogy "billegtessük" a belső botkormányt, figyeljük a képernyő színeit, és lejegyezzük a megfelelő színkombináció kódját.

```
100 PROGRAM "COLORKEY.BAS"
110 LET B=0:LET F=255
120 DO
130 SELECT CASE JOY(0)
140 CASE 1
150 LET B=B+1 BAND 255
160 CASE 2
170 LET B=B-1 BAND 255
180 CASE 4
190 LET F=F-1 BAND 255
200 CASE 8
210 LET F=F+1 BAND 255
220 CASE ELSE
230 END SELECT
240 SET #102:PALETTE B,F,B,F
250 SET 27,B
260 PRINT #102,AT 1,1:B,F
270 LOOP
```

Directory-fájlba

Az idő előrehaladtával szükségünk lehet arra, hogy felidézhesük lemezeink tartalmát. Ha emlékezetünk erre túl rövidnek bizonyulna, úgy célszerű ezt a feladatot is a gépre bízni: a directory listázását írnyítsuk át egy fájlba, és később csak ezt "tájpoltjuk ki". Basic-ben a megoldás:

```
OPEN #1:"A:DIRINFO" ACCESS OUTPUT
SET 4,1
:DIR
CLOSE #1
SET 4,0
```

Ezek után a directory aktuális tartalma a DIRINFO fájlba kerül. Csak megjegyezzük, hogy így akár (az egyenlőre) vírusmentes programjaink eredeti hosszát is feljegyezhetjük a későbbi összehasonlíthatóság érdekében...

Sztringek a Pascal-ban

A Hisoft Pascal az eredeti Pascal-t olyannyira hűen valósítja meg az Exos-on, hogy a rendszer nem ismeri a STRING típust. Megoldást jelent a karaktertömbök használata, amelyekből annyiféle kell tipizálnunk, ahány különböző hosszúságút akarunk használni. Ha esetleg escape-szekvenciát (azaz nem ábrázolható karaktereket) akarunk elküldeni az Exos-nak, akkor ne is próbálkozzunk a CONCAT()-al, hiszen ez már alapértelmezésben feltételezi a STRING típus meglétét. Csak úgy érhetünk célt, ha a tömb minden elemét külön-külön megadjuk... Ez van!

Megint egy START program!

Egyik előző számunkban már írtunk arról, hogy mennyire megkönnyítheti életünket egy-két ügyesen megírt START program. Most is közreadunk egyet ezekből, ezúttal nem csak a programfejlesztőknek, hanem azoknak is, akik játszani is szeretnek.

Egy lemezen gyakran igen sok játékprogram fér el; a normál kétoldalon is átlagosan nyolc, de például a 3,5 hüvelykes (vagy 5,25 hüvelykes nem igazán szabványos) 720 kilobájtoson 16, az 1,44 megabájtoson pedig kb. 32. Egy-egy játék rendszerint 2-3, de néha 4-5 vagy még több fájlból áll. Hogy el ne vesszünk a félszáz különböző fájlnev között, érdemes a játékok 2., 3. és további fájljait HIDDEN (rejtett) attribútummal ellátni, csak az elsőt hagyni meg normálnak. Ekkor csak egy-egy fájl látható a katalógusban játékként, ennek neve pedig rendszerint megegyezik a játék nevével. Így már könnyebb a választás (az EXDOS pedig gond nélkül megtalálja a rejtett fájlokat).

Hogy a dolog még egyszerűbb legyen, írhatunk egy rövid segédprogramot, amelyet azután START néven minden lemezünkre rámásolunk. Az F1 billentyű megnyomására a program elindul, és kiírja a katalógust. A kiírásán kívül a margótól beljebb villogó, megváltozott színű kurzor figyelmeztet arra, hogy most nem az IS-BASIC parancsértelmezőjével vagyunk kommunikációs kapcsolatban. A kurzorral a megfelelő programnevet tartalmazó sorra állva rögtön kiválaszthatjuk a futtatni kívánt programot, amit a programunk egy egyszerű trükkel elindít.

A program az illendőség kedvéért beállítja a dátumot és az időt, ha az nem lenne beállítva; ez persze játékprogram betöltésekor nem igazán fontos, ilyenkor a 120 ... 160-as sorszámú programrész elhagyható.

Csak megjegyezzük, hogy az itt közölt program tudomásunk szerint a lehető legegyszerűbb módszer programok menüből való indítására (jelenleg - az olcsó beszédfelismerő alrendszerek piacradobásáig és az ENTERPRISE-hoz illesztésük megvalósításáig - ez a legkorszerűbbnek mondható eljárás az ember-gép kapcsolat területén). Emiatt hivalkodik kis programunk az előkelő MENÜ kiírással.

Mint minden egyszerű megoldásnak, ennek is megvannak a gyenge pontjai: senki sem akadályoz meg bennünket abban, hogy a képernyő tetszőleges pontjára vigyük a kurzort, és ott nyomjunk ENTER-t; ekkor ne csodálkozzunk, ha esetleg kellemetlen hibaüzeneteket kapunk. Nincs persze akadálya annak, hogy egy minden igényt kielégítő, álmobeli menüprogramot csináljunk, de ez a legegyszerűbb esetben is nagyságrendekkel több erőfeszítést igényel. Talán majd valamelyik következő számban...

Csak a rend kedvéért jegyezzük meg, hogy vannak, akik az egyes játékokhoz tartozó fájlokat játékként külön alkönyvtárban tárolják; emellett szól az, hogy a különböző játékok esetleg azonos nevű fájlokat használnak, ezek ekkor nem zavarhatják egymást. Ebben az esetben a program jelenlegi egyszerű egyszerűségében nem alkalmas a programok indítására. Nincs azonban akadálya annak, hogy képessé tegyük alkönyvtárak végigbogarászására is.

```
100 PROGRAM "START.BAS"
110 STRING X$
120 EXT "var 79 185"
130 IF DATE$="19800000" THEN
140 EXT "date"
150 EXT "time"
160 END IF
170 SET #102:PALETTE BLACK, GREEN, BLACK, CYAN
180 CLEAR SCREEN
190 PRINT :PRINT
200 PRINT " PROGRAMBETOLTO MENU"
210 PRINT
220 EXT "dir"
230 PRINT
240 PRINT " Alljon a kurzorral az indítani"
250 PRINT " kívánt program sorába és nyomjon"
260 PRINT " ENTER-t!"
270 LINE INPUT PROMPT " ":X$
280 IF LTRIM$(X$)<>" " THEN
290 LET X$=RTRIM$(X$(1:8))&"."&LTRIM$(X$(10:12))
300 RUN X$
310 END IF
320 SET #102:PALETTE BLACK, GREEN, BLACK, RED
```

GRANGE HILL

Ahányszor a számítógép mellé ülök, hogy az ENTERPRISE beépített szövegszerkesztőjével elkezdjek egy játék-leírást, mindig az jut az eszembe, hogy vajon miért nem írnak az olvasók?! Miért nem írják meg, mit szeretnének olvasni az ENTERPRESS hasábjain? Mindig tanácsstalan vagyok, hogy vajon a POPEYE, vagy a mostani GRANGE HILL érdekel-e egyáltalán valakit? Ezért arra szeretnék kérni minden olvasót, hogy írja meg, melyik játékkal nem tud mit kezdeni, melyik az a játék, amelyiket végig szeretne játszani, de nem tudja mit kell csinálni, legyen az kaland-, akció- vagy logikai játék. A GET DEXTER befejezését még senki nem írta meg! Pedig azt hiszem, minden olvasó izgatottan várja, mit is kezdünk, hogyan meneküljünk meg. Talán senkit sem érdekelnek ezek a játékok? Vagy ezt a rovatot senki sem olvassa? Én biztos vagyok benne, hogy igenis, kellenek a játékleírások! Várom a leveleket!

Most pedig térjünk rá a cikk igazi lényegére, kezdjük a GRANGE HILL-el. A játék egy BBC tévésorozat alapján készült Matthew Rhodes, Nick Vincent és John Pickford jóvoltából. A grafikát Jeremy Nelson, a zenét pedig Dave Whittaker - aki a legközismertebb zeneszerző a Spectrumosok körében - készítette 1987-ben.

Piacra az ARGUS PRESS SOFTRWARE LTD dobta ki. Ha mindezt elolvastuk, és végignéztük a digitalizált képeket, akkor nyomjuk meg a tűzgombot! A játékot mindhárom joystickról irányíthatjuk.

Gonch, akit a játék elején már megismerhettünk, egyik délután barátjával, Holloval sétál hazafelé, de a házuk ajtaja előtt jut eszébe, hogy nemcsak későn ér haza (hiszen az iskola mellett lévő olasz fagyizót vétek lenne kihagyni), de még a vadonatúj walkman-jét (vagy magyarul sétáló magnóját) is az iskolában hagyta. Az anyukája pedig azt mondta, amikor Bécsben, a Maria Hilfer strassén megvették a Keleti pályaudvarnál 7 forintért váltott maradék schilling-ből, hogy úgy vigyázzon rá, mint a szeme fényére, mert ha elveszti, megöli. Talán csak viccelt? Nem! Ő nem szokott viccelni. Mikor mindezt elmesélte barátjának, Hollo megkérdezte: "Most mihez kezdessz?". Gonch válasza frappáns, egyedülálló és egyben ijesztő, horrorfilmből illő volt: "Nem tudom!" Aztán még egyszer végiggondolta a történeteket, és hozzátette most már szelíd, szerény hangon: "Visszamegyek a suliba". "De az már zárva van!" - mondta Hollo, mire Gonch rávágta: "Nem baj! Majd betörök!". Hollo azonmód kapott az alkalmon, és kijelentette: "Veled megyek!"

Itt lépünk be a játékba. Merre is induljunk el? Próbáljuk meg balra! Oooh! Ez egy rossz ötlet volt! Gonch anyja nem viccelt, megölt minket. Talán kezdjük újra a játékot, és most másszunk fel a kezdő képernyőn lévő telefonfülke tetejére. Ugorjunk át a fal tetejére, menjünk végig rajta és a végén essünk le. Jobbra egy kutya zárja el az utunkat, így menjünk balra. Itt vegyük fel a horgászbotot (Fishing Rod), és másszunk vissza a falon. Menjünk vissza egészen a startpályáig. Útközben vegyük fel a zseblámpát is (Torch), amely elem hiányában nem működik. A kezdő pályától menjünk jobbra ameddig csak tudunk. Vegyük fel a teleszkópot (Telescope), majd menjünk vissza egy pályát, és essünk le a lyuknál. Menjünk jobbra a csatornáig, ahol használjuk a horgászbotot (USE/FISHING ROD/"Fish Bone"). Majd menjünk visszafelé, útközben vegyük fel a törött széklábat (Broken Chair Leg). Lesz egy létra felfelé, itt másszunk fel. Vegyük fel a történelemlétkönyvet (History Book), majd menjünk lent tovább balra. Itt van egy papírrepülő, próbáljuk meg felvenni. Nem sikerült, hiszen túl magas van. Gondolkozzunk! Talán ha ráállunk valamire? De mire? A zseblámpa még szét-törne alattunk. A teleszkóp sem éppen erre való. Talán a történelemlétkönyv? Ha már eddigi életünk során nem használtuk semmire, most próbáljuk meg! (USE/HISTORY BOOK/"Stand on book". Most már sikerült felvenni a repcsit. Menjünk vissza a kutyaéhoz, és adjuk oda neki a csontot (GIVE/ROLF/BONE). A kutya a csontot rágva eláll az utunk-

ból. Menjünk tovább, és az utunkba kerülő tárgyakat vegyük fel. (A gyufát a következőképpen vehetjük fel: USE/PAPER PLANE/"Throw fly at matches). Majd ahol elakadunk, ott másszunk fel a vasrácsra, és menjünk el balra. Mint látjuk, itt Imelda állja utunkat. Amikor hozzászólunk, a válasza csak ennyi: "Tűnj a francba! Ha közelebb jössz ...". Hát nem valami kedves hozzánk. Próbáljunk meg jobbra menni! Egy metrő-alagútba jutottunk, ahol egy elgázolt macska fekszik (Dead Cat). Vegyük fel. Mint tudjuk, a lányok könnyűsáka, hát próbáljuk meg odaadni Imeldának, hátha elhúzza a csikot. Igazunk volt. Ha továbbmegyünk, és az akadályokat (Bollards) átugorjuk, akkor a csatorna túoldalára jutunk, ahol felvehetjük a gyertyát (Candle). Majd menjünk jobbra, túl a metrő-alagúton, és az utunkba kerülő oszlopra másszunk fel. Az első fal tetejére ugorjunk át, és vegyük fel a filctollat (Felt Pen). Tovább haladva egy lakattal lezárt ajtó zárja el az utunkat, amit a törött széklábbal leverhetünk (USE/CHAIR LEG/"Hit lock"). Kérjük el Hollotól a kulcsot (TALK/HOLLO/"Give me key"). Menjünk be az iskolába, és a labirintusban keressük meg azt a létrát, amely a STAFFROOM-ba vezet. Innen a kedves olvasóra bízom a befejezést, nehogy ellustuljon.

SUPER ROBIN HOOD

1247 december 24-e van, amikor is Huntington lordja, Robert Earl lányát ismeretlen tettesek elrabolták. Így hát elindult, hogy a veszéllyel dacolva megmentse egyetlen, s gyönyörű szép leányát, aki hetedhét határon híres volt szépségéről.

Ennyi a története a CODEMASTERS programjának. Betöltés után nyomjuk meg az [O]-t. Sorban beállíthatjuk, hogy legyenek-e hangok, zene, illetve joystick-kal akarunk-e játszani. Ha igen, akkor kiválaszthatjuk a megfelelőt. Nyomjunk meg egy gombot (bármit az [O]-n kívüli!), és máris kezdődhet a játék. Feladatunk hihetetlenül egyszerű! Nem más, minthogy a POPEYE-hez hasonlóan az összes szívet összegyűjtsük. Ehhez viszont fel kell vennünk minden kulcsot, mert a lifteket, illetve a mozgó talajokat csak így hozhatjuk működésbe, csak így juthatunk át rajtuk. A játékban vannak még érmék is, de ha örökenergiával játszunk, akkor ezekre semmi szükségünk sincsen, hiszen ezek az egészségünket növelik. (Ezt a jobb alsó sarokban láthatjuk.) Az akadályok közül csak a katonákat érdemes megemlíteni, akik fáradságot nem ismervé lövöldöznek a vakvilágba. De örök egészségnél ezek sem jelentenek akadályt. (Ezt úgy érhetjük el, ha betöltés közben lenyomva tartjuk a [D] billentyűt.)

Egy jótanács: ha megvan az örök "health", akkor a fel-le mozgó szörnyek tetejére ráugorva felugorhatunk olyan helyekre is, ahova egyébként nehezen jutnánk fel.

JOEBLADE

Halló! Itt Sas 203534! Súlyom jelentkezz! - szólalt meg a CB egyik szombat éjjel. Az órára pillantottál, s láttad, hogy már éjfél is elmúlt. Vajon mi történhetett? Mi az a sürgős ügy, ami miatt kiverték a szemedből az álmot. "Itt Súlyom 425871-es! Vétel!" - mondtad, miközben már a kommandósrühád darabjait kapkodtad magadra. "Súlyom! Azonnal gyere a New York-i CIA központba! Sürgős! Vétel!" - recsegte a rádió, majd mély csend borította el a szobát. Gyorsan befejezted az öltözködést, s tíz perc múlva már a repülőtéren voltál. A repülés nagyon kimerített, de nem volt időd ezzel törődni. A CIA és az FBI főnökei fogadtak. Végre elmondták, hogy miért van ez a felhajtás. Nem sokat értettél belőle a melletted rohagáló zöldsapkások miatt. Kezedbe nyomtak egy telexet, melyen a következő állt: "TOP SECRET! Crax Bloodfinger terroristái elrabolták a hat világhatalom államfőit az ENSZ-ben tartott megbeszélés közben. Ezek a következők: G. Bush amerikai, M. Gorbacsov szov-

jet, T. King angol, F. Mitterand francia, Kohl német és Kaifu japán államfők. A túszoikat a New York-i állami börtönbe hurcolták, melyet előzőleg már elfoglaltak. Az önök feladata a következő: a legjobb emberüket juttassák be a börtönbe, a börtön védelmi bombáit aktivizálják, és szabadítsák ki a hat elnököt. A bent lévő emberüknek 20 perc van a menekülésre az első bomba aktivizálásától számítva. Utána a bomba robban. Kérjük a történet titokban tartását!!!" Ekkor már tudtuk, hogy rád esett a választás. Másodpercek alatt felgyereztünk, majd a börtönhöz vittek. Az utolsó emberi hang amit hallottál a kiképződ torkán szakadt fel: "Sok sikert, Joe!"

Ez az előzménye Col Swimbourné játékának, melyet az [S] billentyű lenyomásával indíthatunk. Az irányítás belső illetve külső joystickkal történhet. A képernyő alsó részén láthatjuk a pontszámunkat (SCORE), energiánkat, a még rendelkezésünkre álló időt (CLOCK), a már megmentett túszoikat (HOSTAGES), cellakulcsaink számát (KEYS), valamint a már aktivizált bombák számát (BOMBS). (A bombákat úgy aktivizálhatjuk, hogy az ABCDE betűket ABC sorrendbe állítjuk. Ezt szintén a joystick mozgásával tehetjük meg.) Találhatunk még itt az életbenmaradáshoz szükséges ételt (SOME FOOD), ellenséges ruhát (ENEMY UNIFORM), valamint lőszert (AMMUNITION). Tehát, mint a telexből megtudtuk, a cél (nem a ...) a bombák (6 db) aktivizálása, a kormányfők (6 db) kiszabadítása, és ezek után a megmenekülés. A játék izgalmas, igazi profi módon megvalósított akciójáték, melyet a 128 KB-os jó zene, illetve a zenei és más egyéb effektek csak feldobnak. Azt hiszem, jó szórakozást nyújt mindenkinek. S ha még bele is tudjuk élni magunkat...!

Aki ezek után sem tudja teljesíteni a küldetést, az kövesse az FBI megfigyelői által összeállított tervet: balra 1 képernyőt/kulcsot felvenni/ke/balra 1 képet/túszt megmenteni/le/kulcsot magunkhoz venni/balra 1 szobát/fel/balra 3 képernyőt/le/balra 3 képet/bombát aktivizálni/jobbra 3 szobát/fel/jobbra egy képet/le/le/ismét jobbra egy képet/bomba időszerkezetét beállítani/fel/fel/jobbra 1 képet/le/le/ismételten jobbra 1 képet/kulcsot felvenni/jobbra két szobát/le/balra egy szobát/le/kulcsot magunkkal vinni/fel/bombát aktivizálni/jobbra kétszer/túszt kiszabadítani/jobbra 1 szobát/le/le/le/balra 3 képernyőt/fel/jobbra egy szobát/fel/jobbra még egy szobát/kulcsot felvenni/le/túszt kikötözni/fel/balra egy szobát/le/balra ismét egy szobát/le/kétszer balra/fel/fel/jobbra egyszer/kulcsot magunkhoz venni/le/bombát beállítani/fel/balra/le/le/le/jobbra 5 képet/le/kulcsot felvenni/jobbra háromszor/le/balra/túszt kiszabadítani/le/balra 3 képet/bomba időszerkezetét beállítani/fel/balra háromszor/túszt kikötözni/le/jobbra/le/balra 2 szobát/le/jobbra/kulcsot felvenni/jobbra/le/balra/le/kulcsot magunkkal vinni/balra/fel/túszt megszabadítani átkozott köteleitől/le/jobbra/fel/jobbra/fel/balra kétszer/fel/balra/le/le/balra/le/balra/bombát aktivizálni/jobbra/fel/jobbra/fel/fel/négyszer balra/fel/fel/jobbra/fel/fel/balra ismét négyszer/fel/jobbra 5 képernyőt/fel/fel/fel/balra 2 szobát/fel/balra 3 képet/fel/jobbra/fel/jobbra/fel/jobbra - és a gép máris gratulál az eredményes akcióhoz, mellyel a harmadik világháború kitorésének veszélye tünt el. Még annyit, hogy a térképjelölések a betöltés utáni legelső játékot mutatják. Ha újrakezdjük, akkor a különböző tárgyak helyei felcserélődhetnek.

NETHER EARTH

A Joeblade után a Nether Earth-ben már igen előreugrottunk az időben. 2176-ban a civilizáció már olyan fokra hágott, hogy az emberiség képes volt önállóan gondolkodni tudó, és így harcra képes robotembereket készíteni. Ezek segítségével került hatalomra a sötétség birodalmának fejedelme, a pusztító vezére, Dark Evil, akinek a legújabb terve is megvolt már: elpusztítani a Földet az emberiséggel együtt, és az őt szolgáló robotemberekkel egy új civilizációt kialakítani. A földi halandóknak egyetlen egy reményük maradt csak: szembeszállni a gonosszal. Így jött létre Duke Mytalker

vezetésével egy Galaxisközi Lázadó Csoport (Galács). A csoport tagjainak száma napról napra nőtt, és el is jött az a nap, mikor már biztos szembeszállhatnak Dark Evil sötét robotseregével. Sikerült elfoglalniuk az ellenség egy robotgyártó központját, melynek segítségével már ők is tudtak harci robotokat gyártani. 2376-ban a sötétség 100 év diktatúrája elbukni látszott. A Nether bolygón megkezdődött a véres, szörnyű, pusztító háború, mely lehet, hogy az emberiség utolsó háborúja lesz. A tét óriási: Élet vagy Halál.

A játékot betöltés után a [6]-os billentyűvel indíthatjuk. Az irányítás bármelyik joystickkal illetve a [Q] - fel, [A] - le, [O] - balra, [P] - jobbra, [SPACE] - tűz gombokkal történhet, az [5] - save, és az [I] - stop billentyűk használhatók még. A harcban egy rádióirányítású radarral felszerelt minikonstrukcióval veszünk részt, és ennek a segítségével gyárthatunk illetve irányíthatunk robotokat. (Nem is sokat tudna tenni egy ember ebben a nukleáris háborúban.) A radar segítségével a képernyő alján láthatjuk a bolygó térképét, természetesen csak a radar hatósugarában. A jobb oldalon fent láthatjuk az eltelt időt napokban illetve órákban. Alatta pedig a menükezelésnek szorítottak helyet a programozók. Ha éppen nem menüpontok vannak itt - mint a játék kezdetekor, - akkor a sötétség erőinek illetve a lázadó emberek eszközeinek számát láthatjuk. (Warbases - központok, melyek ha a mieink, akkor a tetejükön egy H betű van; Electrics - elektromomodulok; Nuclear - Nukleáris bomba; Phasers - lézergyű; Missiles - rakéták; Cannon - ágyúk; Chassis - mozgató berendezések; Robots - robotok.) Ez alatt pedig a már elfoglalt gyárak által gyártott eszközök számát, illetve a General felirat mellett a nyersanyagot láthatjuk.

Robotokat a következőképp szerelhetünk össze: szálljunk rá egy központra, természetesen egy olyanra, ami a már a sajátunk (amelyiken van H betű). Ekkor egy menü jelenik meg. Ha csak véletlenül jöttünk ide, akkor menjünk az EXIT MENU felírra, és nyomjuk meg a tűzgombot. Ha nem, akkor először a bal oldalon lévő alkatrészekből válasszuk ki a nekünk megfelelőket a tűzgomb megnyomásával. (Ha mégsem kell, nyomjuk meg ismét a tűzgombot.)

A következő alkatrészek állnak a rendelkezésünkre: ELECTRONICS - Elektronikus vezérlő. Ha van a robotban ilyen, akkor tud védekezni, tehát ha ráló valaki, akkor visszalő. 3 credits értékű. NUCLEAR - Nukleáris bomba. Ez a legdrágább (20 credits). Nem sok értelme van, csak akkor érdemes használni, ha a sok hullá miatt már nem tudunk valahol átmenni, vagy egy szűk folyosón kéne átmenünk, és a folyosó másik végén túlságosan nyűzsögnek az ellenség robotjai. Használatakor a környezetünkben lévő épületek romba dőlnek. PHASERS - Lézergyű. Ára 4 credits, a leghatékonyabb fegyver. MISSILES - Rakéta. Ugyan annyiba kerül mint a lézer, de nem olyan hatékony. CANNON - Ágyú. A legolcsóbb fegyver (2 credits), a hatékonyságán meg is látszik. ANTI GRAV - Anti gravitációs talp. Ezzel mindenen átjuthatunk, de az ára is borsos (10 credits). TRACKS - Lánctalp. 5 credits-be kerül, de nem tudunk vele mindenhol átmenni. RIPOD - Lépegető láb, 3 credits-ért. Ha felépítettük az izlésünknek megfelelő robotot, akkor menjünk a START ROBOT felírra, és nyomjuk meg a tűzgombot. Ekkor a robotunk életre kel. Természetesen be is kell őket programozni. Ezt úgy tehetjük meg, hogy rászállunk valamelyik robotra. Ekkor megjelenik egy újabb menü. A DIRECT CONTROLL-lal saját magunk vezérelhetjük a robotot. A GIVE ORDERS kiválasztásakor egy újabb menüt kapunk. Itt a STOP AND DEFEND-el kapcsolhatjuk ki a robotot. Az ADVANCE ?? MILES illetve a RETREAT ?? MILES menüpontokkal előre illetve hátraküldhetjük a robotokat, de a maximális határ 50 mérföld. A SEARCH & DESTROY-ba lépve ellenséges robotokat, gyárakat illetve robotgyártó központokat tehetünk a földdel egyenlővé. A SEARCH & CAPTURE-val pedig semleges illetve ellenséges gyárakat és központokat foglaltathatunk el robotjainkkal. A COMBAT MODE menüpontjaival harcolni tudunk közvetlen irányítással. Innen a STOP COMBAT-tal lehet kilépni. Végül a LEAVE ROBOT segítségével szakadhatunk el a kiválasztott robottól.

Postafiók. 334.

A FORTH programozási nyelv érdeklő *Kövi Miklós* szabad-szállási olvasónkat. *"A veremkezelést megkönnyítő utasításoknak egy részét találtam csak meg az IS-FORTH szótárában, az SP@ és az RP@ nem szerepel, pedig a FORTH tankönyvek hivatkoznak ezekre. Az az ötletem támadt, hogy létrehozom magam a hiányzó SP@ nevű szót (ez a veremmutató címét adja meg)."* Olvasónk a továbbiakban leírja a verem megtalálására tett kalandos próbálkozását. A siker nem teljes: *"Az általam írt SP@ szó 4-gyel magasabb memóriacímre adott vissza, mint ahol a veremben írt számok valószínűleg elhelyezkednek. Ráadásul a verem tetején lévő számot nem találtam."*

Kedves Kövi Miklós! Az a kivételes szerencse jutott az Ön osztályrészeül, hogy szerkesztő kollégánk "előző életében" elég sokat foglalkozott a FORTH-szal. Egyebek között maga is megpróbálkozott egy FORTH rendszer megírásával, majd a szabványnak számító F.I.G. FORTH rendszert adaptálta egy 68- és hőkori 8 bites mikroszámítógépre. Nos, a választát továbbítjuk most Önnek (és mindenkinek, aki érdeklődik e nem mindennapi programozási eszköz iránt).

A FORTH rendszer egyik jellemzője a rendkívüli gyorsaság. Annak ellenére, hogy a FORTH "program" futása mindig interpretálva valósul meg, a rendszer majdnem egy gépi kódú program sebességével dolgozik. Ehhez a rendkívüli gyors működéshez a FORTH úgynevezett belső interpreterét, amelyik az utasítások láncolatának végrehajtását szervezi, a lehető legjobban "ki kellett hegyezni". Ezt úgy érik el, hogy a rendszer legfontosabb eszközeit nem a memóriában valósítják meg, hanem a mikroprocesszor regisztereit használják fel. Az IS-FORTH belső szerkezetét részleteiben nem ismerem, de a F.I.G. FORTH például - pont ellentétben a laikus első elgondolásával - a mikroprocesszor veremmutatóját használja FORTH adatverem-mutatónak (SP), és egy másik regiszterpárt a vizuális verem mutatójának (RP). Az olvasónk által észlelt 4 bajtnyi eltérést ez megmagyaráz kettőt: a Z80 mikroprocesszor SP regisztere a verem legfelső eleme fölé mutat, azaz 2-vel kisebb memóriacímre tartalmaz, mint ahol a felső elem van (a veremmutató neve másképpen "szabadhely-mutató").

A fennmaradó két bajtnyi eltérést, illetve a legfelső elem hiányát pedig egyszerűen az okozza, hogy a legfelső elem soha nincs fizikailag is a veremben (illetve csak szükség esetén, ideiglenesen kerül oda). Mivel minden művelet eredménye a verembe kerül, a következő utasítás pedig úgyis ezzel az eredménnyel fog majd foglalkozni, felesleges az értéket minden utasítás után a memóriába írni, majd a következő pillanatban újra visszaolvasni: a verem teteje a mikroprocesszor valamelyik regiszterpárja (pl. a HL). Nyilván ugyanezt vagy hasonló megoldást alkalmazták az IS-FORTH-nál is.

Pulai Sándor zalaegerszegi olvasónknak a beépített szövegszerkesztővel gyúlt meg a baja:

"Definiálom az ékezetes karaktereket, majd beállítom a szövegszerkesztő jobb margóját, és elkezdem a szöveg beírását. Amikor eljutok a jobb margóhoz úgy, hogy a szó teljes hosszában nem fér el a margóig, akkor a gép az ékezetes betűnél "eltöri" a szót, ahelyett, hogy azt teljesen átvinné a következő sorba. Kérném a segítségüket, hogyan tudnám "megmondani" a gépnek, hogy melyik karakterből lett ékezetes betű?"

Kedves Pulai Sándor, a legkönnyebb volna egy cinikus választ adni: használjon szövegszerkesztőt! Az ironia természetesen nem Önön csattan, hanem a nagyképű WP-nek nevezett gyenge kis "szövegszerkesztőcskén" (a WP, azaz Word Processzor rövidítés a korszerű számítógépeken alkalmazott sokoldalú, intelligens szövegszerkesztőket jelenti).

Sajnos, a WP-nek nem ez az egyetlen betegsége. Sok-sok jellemzőjét kellene megváltoztatni ahhoz, hogy szövegszerkesztőnek lehessen nevezni. Csak párat említünk:

Egy szövegszerkesztőnek meg kell tudnia keresnie egy megadott szövegrész előfordulásait a szövegben, illetve ezeket ki kell tudnia cserélni egy másik megadott szövegrészre. Lehetővé kell tenni egy nagyobb szövegrész kijelölését, hogy ezen valamilyen műveletet lehessen végezni, pl. le lehessen törölni. Meg kell engedni a szöveg egy részének kimentését külön fájlba, illetve egy meglévő szöveg bemásolását a szerkesztett szövegbe (a WP ezt a bemásolást csak ASCII fájlként - a PRINT utasítással kiírt - szöveggel engedi meg, és ezt is igen bután). A PRINT utasítás is "buta", felesleges, hogy a szöveget a képernyőn is mozgatva, sokszorosára lassítsa a kiírást.

Elképesztően ostoba dolog, hogy ha betelik a szöveg szer-

kesztéséhez használható mindössze 16 kilobájtos puffer, a szöveg eleje minden figyelmeztetés nélkül elvész (pontosabban, a tapasztaltabbakat figyelmezteti a státusz sor jobb szélén megjelenő szám).

Értekes anyagaink sorozatosan mehetnek tönkre azáltal, hogy a WP nem kezeli tisztességesen a fájlkat. Egy szövegszerkesztőnek automatikusan el kell mentenie a fájl régi tartalmát, amikor valami változtatást végzünk az anyagban, hogy bármikor visszaállítható legyen az eredeti tartalom. A WP-vel, ha pont a fájl mentésekor történik valami hiba, mondjuk feszültségkimaradás, elvész az anyagunk. Azt sem akadályozza meg a program, hogy egy létező fájl tartalmát tévedésből felülírjuk.

A megoldás: egy tisztességes szövegszerkesztő használata. Sajnos, nem tudunk ilyennek a tömeges árusításáról a Centrum áruházak egész országra kiterjedő ENTERPRISE-os hálózatában...

Még egy mondatot azoknak, akik az ENTERPRESS-ben szándékoznak megjelentetni frászműveket: ne törődjenek az "eltört" szavakkal, az általunk kidolgozott továbbfeldolgozási eljárás során azok automatikusan "összeragadnak" majd...

Szabó Gábor győri olvasónk több furcsaságot talált gépében (csak az angol nyelvű gépen próbálkozott):

*"Ha egy funkcióbillentyűt üres stringre programozunk, azaz pl. beírjuk, hogy SET FKEY 1", az adott billentyű lenyomásával mindig megállíthatjuk a program listázását vagy futását. Így a program megállítható még SÉT INTERRUPT STOP OFF esetén is. A listázás azonnal megáll, ha megnyomjuk a billentyűt. Ha programfutás közben nyomjuk meg, akkor a következő adatbeviteli utasításnál áll meg. Ha például van egy INPUT utasítás a programban, beírjuk az adatot, majd még az ENTER megnyomása előtt megnyomjuk a funkcióbillentyűt; a program a következő utasításnál *** STOP key pressed. üzenettel megáll. Hasonló a helyzet a GET utasításnál is."*

Ha automata sorszámozással írjuk programjainkat, és valamelyik sorban az INPUT utasítás IF MISSING feltételes alakját használjuk, a sorszámozás 610 körül folytatódik. Legtöbbször 611-es és 612-es sort ír ki a gép.

Végezetül: mindenhol kerestem, hogy hol lehet gépi kóddal olyan programot betölteni, amelynek nem ismerjük a hosszát, de sehol sem találtam rá megoldást. Az EXOS 6 kéri a program hosszát. Talán EXOS 5-tel kell? Kérem, válaszoljanak!

Az üres stringre programozott funkcióbillentyű valóban megállítja a program futását vagy a listázást, de ez nem meglepő: a leírás említi is, hogy az üres string megszakítást okoz.

Az automata sorszámozás és az IF MISSING "összeakadása" egy igazi "bug", azaz programhiba. Különösen, ha kipróbáljuk, hogy bizony a 610-es sor utánról is "visszamegy" ebbe a tartományba, felülírva az esetleg létező 611-es sort. Egyébként kipróbáltuk, hogy az IF MISSING-ben tevékenységként sorszámot megadva a 611-es, EXIT DO, EXIT FOR vagy EXIT DEF-et megadva a 616-os sorra ment a sorszámozás; LINE INPUT esetén ez a két érték 617 és 622 volt. Akárhogy is, várjuk a forgalmazó választ!

Programbetöltési problémájára szerencsére van gyógyír. Mivel egy file beolvasásakor egyáltalán nem biztos, hogy ismert a hosszúság, a betöltő funkciót úgy kell megcsinálni, hogy képes legyen az ismeretlen hosszúságot kezelni. Nos, az EXOS 6-os funkció éppen ilyen. A híváskor a BC regiszterpárba nem a blokk hosszát kell tölteni, hanem azt a maximális hosszúságot, amit még elfogadunk (amennyinek helyet csináltunk a memóriában). Ha a program ennél rövidebb, a betöltés után a BC regiszterpár mutatja, hogy a beolvasott blokk mennyivel volt rövidebb, mint amit vártunk; a DE regiszterpár pedig a pufferekben a beolvasott blokk utáni első szabad bájtra mutat. Az akkumulátor ekkor a .EOF (0E4H) kóddal jelzi, hogy elértük a fájl végét.

Ha ellenkezőleg, a BC regiszterpárban 0 van, akkor sikerült a puffert teleolvasni a blokkal; azonban lehet, hogy a fájl hosszabb, még tovább kell (vagy lehet) olvasnunk.

Az EXOS 5 funkciókóddal is olvashatunk, de ezt csak akkor érdemes használni, ha a fájl valóban bájtonként akarjuk kezelni, pl. listázunk a képernyőre.

Fóti Marcell péceli olvasónk súlyos kijelentéseket tesz a gép hangzásbeli képességeivel kapcsolatban:

"A lap előző számában közreadtam néhány igen egyszerű dal-lamocskát azért, hogy bemutassam a gép hangkezelőjének használatát. Az egyszerűbbtől a bonyolultabb felé haladva (több szólam használata, más hangszínek kikísérletezése) azonban hama-

rosan megmutatkoznak a ROM0 SOUND: kezelőjének hiányosságai, a haraver huanvosságokról nem is beszélve. Megpróbáltam a gép szupernek titulált hangelállítást szintetizátorprogram írására kényszeríteni, mivel ilyet meg nem láttam erre a gépre. (Most nem a beépített D/A átalakítót használó programokról beszélek, hanem övönről, amú modulál.)

Gonaotiam, a negycsatornás sztereó hang chippel a világ összes hangját elő lehet állítani, de valahogy nem sikerült. Kuarcom okai: először is a beépített hangkezelőről el kell mondani, hogy bonvolultsága ellenére (több szólam kezelése) "normális" hangzást nem tudok vele készíteni. ("Normális" hangzás alatt hangszerek hangjának utánzását értem.) Ugyanis egyrészt a burkológörbekezelő (továbbiakban EP-GEN) túl nagy lépésközzel doigozik (1/150 s), amú az egy fázison belüli gyors frekvenciaváltásoknál (lásd elektronikus dobok) és amplitúdóváltozásoknál nálinaó is (lepcsős): másrészt több szólamú zenénél maximum torzításokat használhatunk, hogy a hangzások még elviselhetők, és a szólamok megkülönböztethetők legyenek. Ha gyakran használjuk a SYNC szinkronizációt, egyszerűen lebetegszik a gép, és csak egy finom kis RESET térít magához. (Talán a SOUND inkorrekt módon bánik a csatornákkal?) Van még egy oka annak, hogy végül is feladtam a beépített hangkezelő bővítését, megpeáig az, hogy az EP-GEN nem teszi lehetővé, hogy egy hang megszólalása alatt a jelalakot változtassam. Ez pedig fontos volna sok hangzásnál, pl. a gítárhanggal. Mi a megoldás? Egy új hangkezelő írása! Az EP-GEN rövidebb fázisokkal doigozzon, és változtassa a jelalakot a hangban, ha kell.

Gépünknek csak négyszöggenerátorai vannak. Sebaj! Különböző szűrőkkel manipulálva a többi hullámformát is előállíthatjuk. Most jön az újabb csalódás: csak felüáteresztő szűrőink vannak, amelyek a felharmonikusokat engedik át, de ennél sokkal fontosabb lenne egy aluláteresztő szűrő, amely lenyesegetné a felharmonikusokat, és így közelítene a jel a szinusz formájához. További hátrány, hogy a szűrőket nem lehet függetlenül vezérelni, hanem csak egy másik hangcsatornával, amú ettől "foglalt" lesz.

Mind Ezek miatt állítom, hogy az ENTERPRISE-zal nem lehet jó hangzást előállítani. Mégis, milyen sok játéknak van jó zenéje (pl. MAGICBALL)! Ez igaz, a zene jó, de a hangzás az "lavór"!

A RING modulátorról mostanáig bölcsen hallgattam. Ennek az az oka, hogy bár tisztában vagyok fizikai (hullámtan) működésével, de fogalmam sincs arról, hogy mire jó. Ugyanez áll a különböző polinomszámlálókra is.

Levelem témáját vitaindítónak szántam. Aki nálam jártasabb a témában, kérem írja meg, hogy miért nincs igazam (mert remélem nem ilyen sötét a kép, mint ahogy lefestettem), és/vagy magyarázza el a RING modulátort, a polinomszámlálókat stb.

A levélíróval teljesen egyetértünk. Az ENTERPRISE hangjával kapcsolatosan egyébként is valamiféle úr van: nincsenek zeneprogramok, a játékok alatt is többnyire csak igen gyengécske (legtöbbször Spectrum) dallamok szólnak, szerkesztőségünkhöz - ezt nem számítva - még egyetlen zenei témájú cikk sem érkezett. Lehet, hogy az ENTERPRISE e zenei némaságban is hasonlít egy kicsit a PC-kre, és örökké csak "pittyegő" szoftverek lesznek hozzá? Ki tudja, mindenesetre érdeklődve várjuk a válaszleveleket.

Végezetül minden Kedves Olvasónknak! Kérjük Önöket, hogy ha megtisztelnék bennünket levelükkel, programot vagy egyéb ötletet küldenek, a küldemény lapjait vagy tűzzék össze, vagy minden egyes lapra írják rá nevüket és címüket! A leg gondosabb kezelés mellett sem garantálható, ugyanis, hogy egy-egy levél, vagy egy levél egy-két lapja el ne keveredjen a többivel vagy a borítéktól. Így érhetjük csak el, hogy minden írás valóban a szerzője neve alatt jelenjen meg (és - indokolt esetben - a honoráriumot is el tudjuk küldeni).

Segítségüket eddig is és ezután is köszöni:

A felelős szerkesztő

mikrovilág

Az ENTERPRESS előző számai korlátozott példányszámban még megrendelhetők a kiadó címén (MÁTRIX Kft. 8000 Székesfehérvár, Dózsa Gy. tér 10.), vagy megvásárolhatók a Műszaki Könyvtárházban (Bp. VI. ker. Liszt F. tér 9.) és a FOKUSZ Könyvtárházban (Bp. VII. ker. Rákóczi út 14).

Tisztelt Olvasóink!

Arra kérjük Önöket, hogy utórendeléseiket és megrendeléseiket ne a szerkesztőség, hanem a kiadó (Mátrix Kft.) címére küldjék, mert így sokkal gyorsabban juthatnak hozzá kedvenc lapjukhoz.

Köszönjük!

Apróhirdetések

ENTERPRISE gépemhez színes, euro-scart csatlakozós, vízszintes sorozással rendelkező Philips monitort keresek (használt is lehet, 80 karakteres képernyő is olvasható legyen).
Laczhégyi László (22)15-113, (22)12-312/222 mellék

ENTERPRISE lemez meghajtót vennék.
Kulcsár Zoltán, 2600 Vác, Csaba u. 11.

ENTERPRISE GÉPEMET LEMEZEGYSÉGGEL BŐVÍTENÉM. ÁRAJÁNLATOT KÉREK! SZARKA ENDRE, 8500 PÁPA, MIKES K. u. 11.

ENTERPRISE-hoz EPROM égető megrendelhető. Égethető típusok: 2716 — 27256.
Barna Miklós 6000 Kecskemét, Vitorla u. 36.

LEVELEZÉS

A géppel kapcsolatos témákban leveleznék:
Bognár Balázs, 9443 Petőháza, Bartók Béla u. 11.
Budai Attila, 2119 Pécel, Várhegy út 12.

ENTERPRISE klubok,
ENTERPRISE köré szerveződött baráti társaságok,
ENTERPRISE témákban levelezni kívánók címeit várjuk!

A Mikrovilág minden számában két oldalnyi terjedelemben foglalkozik ENTERPRISE-os témákkal.

hirdetések, felhívások

HIRDETÉSFELVÉTEL

Az apróhirdetések ára: 1 Ft karakterenként. A szöveget és a befizetést igazoló nyugtát (rözsaszín postautalványon) az alábbi címre kérjük elküldeni:

MÁTRIX Kft.

ENTERPRESS

8000 Székesfehérvár

Dózsa György tér 10.

Bankszámlaszám: OTP 679-022096-9

Közületi hirdetőknak kívánságra hirdetési árajánlatot küldünk.

Megjegyzés: a nem saját fejlesztésű szoftverek másolásával foglalkozó üzletelők hirdetéseit nem áll módunkban elfogadni.

Tisztelt leendő Szerzőtársak!

Kérjük Önöket, hogy anyagaik elkészítése, beküldése előtt feltétlenül kérjenek ingyenes tájékoztatót a feltételekről, a szerkesztőség által felállított tartalmi és formai elvárásokról.

Az érdeklődők leveleit levélcímünkre várjuk:
ENTERPRESS

1399 Budapest, Pf. 701/334