II. évfolyam 6. szám * 1991. november — december * Ára: 49,- Ft

ENTERPRESS

KÉTHAVILAP AZ ENTERPRISE SZÁMÍTÓGÉPEK FELHASZNÁLÓINAK

Tudósítás

Az egyik kórház belgyógyászati osztályára elgyengült, ismeretlen egyedet szállítottak be a mentők. Az orvosok szerint az eset nem mindennapi, az egyed ugyanis nem hasonlít az emberre. Lapos, négyzetes formájú, külseje fehér, testfelületének nagyrészén fekete foltok találhatók. Laboratóriumi vizsgálatok kimutatták, hogy az egyed összesen nyolc rétegből áll. Rendkívüli érdekessége, hogy a rétegek bármelyikébe egy, a könyvlapozáshoz hasonló mozdulattal be lehet hatolni, és így szélessége duplájára növekszik. Az ápolók megfigyelései, mérései szerint az egyed papírkönnyű, 297 mm magas, jóképű, könnyed gyűrődik.

Az egyed egy eddig ismeretlen kór miatt gyengült le. Diétás nővérek próbálták étellel, itallal kínálni, de a beteg semmit sem fogyasztott el. Ennek okára az ultrahangvizsgálatok és a röntgenfelvételek adtak később magyarázatot: kiderült, hogy az egyednek nincsenek csontjai, belső szervei, így emésztése sincs. Szervezetének 99,999%-a papírból, a maradék 0,001% nyomdafestékből áll. Az orvosok megállapították, hogy ilyen gyenge állapotban februárig élhet a beteg. Az életkörülmények felderítéséhez a rendőrség segítségét kérték a kórház dolgozói.

Az egyed származási helyének felderítésével egyszerre több nyomozócsoport is foglalkozott. A szálak néhány fiatal emberhez vezettek, akik bevallották: az egyed nem más, mint egy, általuk csak "Enterpress"-nek nevezett újság. Elmondásuk szerint az újság többször elpanaszolta nekik létbizonytalanságát, gyengeségét; a lap valószínűleg ezért vitette magát a kórházba. A fiatal emberek egyébként tudták a bajok okát: az újságnak csak olvasói bizalmára van szüksége, így hosszú életű, erős szaklap lehet. A gyógyulás annál gyorsabb lesz, minél többen akarják a lap megerősödését - állítják a pszichológusok.

Sajtótechnikai szakértők szerint a bizalom hosszútávú kinyilvánításának formája az ún. előfizetés. Ez egy illetve fél évre szólhat. Az elsőhöz 294 Ft-ot, a másodikhoz 147 Ft-ot kell küldeni a következő címre: Mátrix KFT., 8000 Székesfehérvár, Honvéd utca 8.

Bosszúságaink

Tudjuk, hogy a szöveges képernyőn a háttér és a karakterek színe megváltoztatható. 40 karakteres üzemmódban kétféle papír-tinta színpárt használhatunk. Ha azonban letöröljük a képernyőt, és átállítjuk a tintaszínt, a gép erre nem reagál, a tintaszín nem állítódik át!

Egyik programomban az alábbi sorok szerepeltek:

```
SET #102:PALETTE GREEN,RED,GREEN,BLACK
```

(azaz zöld háttér előtt piros és fekete betűk), majd később

```
CLEAR SCREEN
SET #102:INK 3
```

Erre a gép zavartalanul piros betűket kezdett el írni. A megoldás: legalább egyszer írni kell a lapra mielőtt átállítjuk a színeket. Legegyszerűbb egy üres PRINT utasítás:

```
CLEAR SCREEN
PRINT
SET #102:INK 3
```

Ekkor már a helyes színt kapjuk.

Szalontai Andrea

Újabb furcsaság

Sok Basic interpreter nélkülözi az olyan elemi szolgáltatásokat, mint például az IS-Basic rendszer RENUMBER parancsa, amely újraszámozza a programsorokat. Bátran kijelenthetjük, hogy egy ilyen rendszer gyakorlatilag játszogatáson kívül bármi másra használhatatlan.

Létezik olyan újraszámozási megoldás, amelyik csak a sorok elején álló sorszámokat számozza át, a rájuk való hivatkozásokat már nem. Például, ha van egy GOSUB 1210 utasítás a programban, az újraszámozás után ez az utasítás máshova fog belépni, mint előtte. Hogy egy ilyen megoldás mit ér, nem kell taglalni.

Az IS-Basic rendszer RENUMBER parancsa intelligens, legalábbis első látásra: a GOTO és a GOSUB utasítások hivatkozásait is átszámozza (más kérdés, hogy az IS-Basic-ben ez az a két utasítás, amelyiket szerencsére soha sem kell használni). Nagy meglepetésünkre azonban az EXLINE függvénnyel együtt pl. egy IF utasításban szereplő hivatkozás az átszámozásnál nem változik, így az eredetileg egy adott utasításban előforduló hiba vizsgálata elromlik.

Bizonyos szempontból talán érthető, miért maradt benn a rendszerben ez a hiba (ha egyáltalán észrevették és foglalkoztak vele): szinte lehetetlen ugyanis kideríteni, milyen módon kellene "átszámozni" a sorszámot az alábbi típusú utasításban:

```
IF EXLINE+20=A*B-320 THEN ...
```

A kiút: kerüljük az olyan programmegoldásokat, ahol a sorszámnak bármiféle jelentősége van (azaz ne használjuk, mivel egyébként sem tanácsos, a GOTO és a GOSUB utasítást sem). Ha a hibakezelésnél nem megoldható, hogy minden egyes kritikus programszakaszt önálló WHEN ... END WHEN blokkba tegyünk, vezessünk be egy változót (a főprogram elején deklaráljuk, hogy feltétlenül globális legyen), és ennek minden kritikus művelet előtt adjunk más és más értéket. A HANDLER-ben így garantáltan azonosíthatjuk a hibát.

Billentyűzet érdekesség

Az EDITOR:-nak sajátos dolgai vannak. Egyes billentyű kombinációkkal utánozni lehet "normális" billentyűket. Kétnyelvű gépemen a Shift+Alt és a táblázat első oszlopában látható billentyűket egyszerre lenyomva a következőket tapasztaltam:

Előállítás	Hatása	ASCII kódja
Y	Enter	13
X	Shift+Bal	185
C	Shift+Fel	177
V	Hold	-
B	Shift+Jobb	189
N	Stop	-

Bor Gergely

TARTALOM 91/6

Assembly

8. rész

A sorozat előző részében tisztáztuk a perifériakezelő és a csatorna fogalmát, megismerkedtünk hét EXOS funkcióhívással. Ebben a fejezetben példaprogramon keresztül egy újabb fontos hívást fogok bemutatni.

A blokkírás és a szekvenciák

A múltkori példában felhasználtuk az EXOS 8 funkciót, de nem esett szó ennek teljesebb használatáról. Ismétlésképpen annyit kell felidézni a funkcióról, hogy segítségével egy csatornára (A) adott számú bájtot (BC) tudunk a tár adott helyéről (DE) küldeni. Ez egy meglehetősen általános ismertetése az EXOS 8-nak, nézzünk három példát, mire használható még:

- A memóriában levő adatokat tudjuk háttértárra (magnó, lemezegység) menteni. Értelemszerűen A-ba a háttértárat azonosító csatorna számát, BC-be a kimentendő bájtok számát, DE-be az adatblokk memóriabeli kezdőcímét töltjük.
- Ugyanígy küldhetünk adatblokkot a hálózatnak, a soros vonalnak, a nyomtatónak, az editornak, a hangkezelőnek.
- A képernyőkre szövegeket írhatunk, és *escape-szekvenciákat* küldhetünk. A szövegmegjelenítés világos, de mit jelent az escape-szekvencia?

Nos, az escape-szekvencia egyfajta különleges vezérlőkódok sorozatát jelenti. Nem csak a VIDEO: eszköznél értelmezett, de jelentőségük talán itt a legnagyobb. Az ASCII 27 (ez az escape kódja) karakterrel kezdődő szekvenciák különféle parancsokat jelentenek az eszköznek. Szekvenciákkal tudjuk például a kurzort ki- és bekapcsolni, a képernyő scrollozását tilthatjuk, engedélyezhetjük; olvasni tudjuk a kurzor pozíciót, a színeket tudjuk beállítani stb. (A képkezeléssel kapcsolatos szekvenciákat a könyv a III. 4.2.-nál sorolja fel.)

A példaprogram

A blokkírás testvére az EXOS 7 - karakter írása funkció. Ennél is A-ba annak a csatornának a számát kell töltenünk, amelyikbe "potytyantani" akarunk, a B-be pedig az elküldendő karakter ASCII kódját tesszük. A képernyőkezeléssel kapcsolatos első példa (*1.lista*) ezt és a szekvenciák használatát is bemutatja. (A könnyebb magyarázhatóság kedvéért a LinePrint segítségével felsorszámoztam a sorokat, természetesen a sorszámokat nem kell bepötyögni.)

A program a következőt teszi: megnyit egy 40*27-es hardver-szöveges képernyőt, majd ennek egészén ciklikusan megjeleníti az ASCII 33-127 karaktertartományt. Az ilyenfajta program a kezdők első munkái közé tartozik.

Milyen feladatokat kell megoldanunk a rutinban?

1., Meg kell nyitnunk a csatornát.

2., Ki kell alakítanunk a karaktereket megjelenítő rutint.

3., Le kell zárnunk a csatornát és jöhet a visszatérés.

Az 1. és 3. pont sima ügy. A megjelenítő rutint sokféleképpen ki lehet alakítani, én a következőket teszem:

- Az akkumulátort duplán kihasználom: ebben tárolom a megjelenítndő karakter kódját, és a kódot ciklusszámlálóként is felhasználom.
- Felhasználok még két regisztert (D és E) ciklusokat alakítva ki velük, amelyeket csupán arra használok, hogy pontosan 40*27 darab, azaz 1080 karaktert jelenítsek meg.
- Nem feledkezem el a karakter megjelenítésről, és a szabatos veremkezelésről.

Ez az első olyan program, amelyben makródefiníciót (3-8) használunk. Ügyeljünk arra, hogy a *@val* paraméter előtt ne álljon szóköz, mert így az ASMON nem hajlandó átvenni!

A 10-11 sorokban a képernyőméreteket adjuk meg, majd a 14-22-ben beállítjuk a szükséges változókat, és megnyitjuk 10-es számon a videocsatornát. A 25-28-as sorokban a 76-78 részen deklarált 3 escape-szekvenciát küldjük el a csatornába: 76-kikapcsolja a kurzort, 77-kikapcsolja a scrollozást (set scroll off), végül 78-megadja a képernyő színeit (set palette c0,c1,c2,c3,c4,c5,c6,c7). A 31-36 részben a már ismerős "DISP"-speciális funkciót hívjuk.

És most lássuk a lényeget! 39-66 végzi el a ciklikusan a karaktermegjelenítést. 39-nél az akkumulátorba töltjük a legelsőnek kiírandó karakter kódját (ASCII 33, ami maga a felkiáltójel!). A karakterkódot egészen 128-ig növeljük majd. Ha még nem érte ezt el, akkor mindig visszaugrunk cycl_0-ra (40), ellenkező esetben pedig lezárjuk a csatornát, és jöhet a RET.

40-nél a veremre tesszük a kódot azért, mert A-t az EXOS hívásoknál kell majd használnunk. A 41-43-ban kiírjuk a csatornába az ASCII 30-at; ennek hatására a képernyő letörlése nélkül a bal felső sarokba (HOME) ugrik a kurzor.

A 44-es sorban a D regiszterbe töltjük a képernyő függőleges méretét (az oszlopok magassága), így D a külső ciklus (cycl_1) számlálója lesz.

```
1         org 01000h
2  ;EXOS valtozo allitasa makro
3  evar    macro @exvar,@val
4          ld b,1
5          ld c,@exvar
6          ld d,@val
7          exos 16
8          endm
9  ;kepernyomeretek
10 x_size  equ 40  ; video x
11 y_size  equ 27  ; video_y
12
13 ;a kepernyo definialasa
14         evar 22,0          ;video mode
15         evar 23,0          ;video color
16         evar 24,x_size     ;video X
17         evar 25,y_size     ;video Y
18
19 ;a kepernyo csatorna megnyitasa
20         ld a,10            ;a csat. szama
21         ld de,vid          ;mutato a nevre
22         exos 1             ;csat. nyitas
23
24 ;a szekvenciak elkuldese
25         ld a,10            ;ide kuldunk
26         ld bc,seqlen       ;ennyi bajtnyi
27         ld de,seq          ;itt kezdodo
28         exos 8             ;szekvenciat
29
30 ;megjelenites (DISP)
31         ld a,10
32         ld b,1
33         ld e,1
34         ld c,1
35         ld d,y_size
36         exos 11
37
38 ;a karakterek megjelenitese
39         ld a,33 ;Ascii 33=!
40 cycl_0  push af ;karakterkod a verembe
41         ld a,10 ;csatornaszam
42         ld b,30 ;Ascii 30=kurzor HOME
43         exos 7  ;karakter ki
44         ld d,y_size        ;sorok szama
45                            ;D a kulso ciklus
46                            ;azaz cycl_1
47                            ;ciklusszamlaloja
48 cycl_1  ld e,x_size        ;oszlopok szama
49                            ;E a belso ciklus
50                            ;azaz cycl_2
51                            ;ciklusszamlaloja
52 cycl_2  pop bc  ;karakterkod a verembol
53         push bc ;vissza a verembe
54         push de ;ciklusvaltozok mentese
55         ld a,10 ;ide irunk
56         exos 7  ;karakter ki
57         pop de  ;ciklusvaltozok vissza
58         dec e   ;belso szaml. csokkentese
59         jr nz,cycl_2 ;nem nulla -> cycl_2
60         dec d   ;a belso ciklus veget ert
61                 ;kulso szaml. csokkentese
62         jr nz,cycl_1 ;nem nulla -> cycl_1
63         pop af  ;karakterkod A-ba
64         inc a   ;novelese
65         cp 128  ;(Ascii) 128?
66         jr nz,cycl_0 ;kisebb, -> cycl_0
67
68 ; a videocsat. lezarasa
69         ld a,10
70         exos 3
71         ret
72
73 vid      db 6
74         defm "VIDEO:"
75 ; a szekvenciablokk
76 seq      db 27,"o"          ;kurzor kikapcs.
77         db 27,"s"          ;scroll kikapcs.
78         db 27,"C",255,32,56,32,0,0,0,0
79                            ;palettaszinek
80 seqlen   equ $-seq          ;szekv. hossz
81         end
```

1. lista

Ezután (48) E-be a vízszintes méretet (a sorok hossza) töltjük, E a belső ciklus lefutásainak számát (cycl_2) fogja vezérelni. *A karakterek kiírását úgy végezzük, hogy E darabot írunk ki D darab sorban.*

A cycl_2 (52) kezdeténél a verem tetején van a megjelenítendő karakter kódja. Ebben a megoldásban kihasználjuk, hogy A az AF-ben és B a BC-ben a magasabb (high) bájtok. Az 52 után tehát B-ben lesz a karakterkód. Ezután gyorsan újra elmentjük a karakterkódot (53), B tartalma természetesen nem romlik el a PUSH után! Mivel EXOS hívás következik, így (54) elmentjük DE tartalmát. 55-56-ban megjelenik a karakter. (Elnézést, amiért megszakítom a magyarázatot! Mióta a sorozatot írom, nem érkezik hozzám semmilyen olvasói kérdés, hozzászólás. Kezdem úgy érezni, hogy senki sem olvassa az Assembly sorozatot. Kérek minden olyan olvasót, aki ezt a szándékosan a szövegbe rejtett felhívást olvassa: kérdezzen, szóljon hozzá! Ha nincs semmi kérdése, akkor is dobjon fel egy levelezőlapot, és legalább ennyit írjon: "Olvasom a sorozatot." Hálásan köszönöm!) Levesszük a veremről DE-t (57), majd a belső ciklus számlálóját csökkentjük (58), és ha az nem érte el még a nullát (azaz még nem futott le annyiszor, ahány karakter hosszú a sor), akkor visszaugrik (59) cycl_2-re. Ezután lezajlik ugyanaz, amiről ebben a bekezdésben írtam.

Ha az E tartalma nullára csökkent, az azt jelenti, hogy egy sort végigírt a rutin. Egy (újabb) sor készen van, tehát csökkenteni kell a külső ciklus (sorok) számlálóját eggyel (60). Ha D értéke nem nulla (62), akkor még nincs teleírva a képernyő, folytatni kell cycl_1-et (l. a megelőző két bekezdést). Tehát ez a ciklus gondoskodik a szükséges számú (y_size) sor megjelenítéséről.

Amikor a képernyő megtelik az adott kódú karakterrel, akkor levesszük a veremről a kódot (63), majd a következő kitételhez növeljük értékét (64). Megvizsgáljuk, hogy elérte-e már a 128-at (65). Ha nem, akkor a cycl_0-ra ugrunk vissza (66), és megjelenítjük az újabb karaktereket.

A 127-es kódú karakter megjelenítése után A tartalma 128 lesz, véget érnek a ciklusok. Nincs más hátra, mint lezárni a videocsatornát(69-70), majd visszatérni.

Fordítsuk le és indítsuk el a programot, de az indítás után ne ijedjen meg senki! Nem a gép órajele esett vissza 5 Hz-re, a program az EXOS hívások miatt ilyen lassú. Íme, egy ékes bizonyíték arra, hogy gyors képernyőkezelést csak a közvetlen videomemória-írással érhetünk el. A második program (*2.lista*) ilyen elven működik, értelmezése a következő részben kerül sorra.

-HCs-

```
1          org 01000h
2
3  ;EXOS valtozo allitasa
4  evar    macro @exvar,@val
5          ld bc,256+@exvar
6          ld d,@val
7          exos 16
8          endm
9
10 ;kepernyomeretek
11 x_size  equ 40
12 y_size  equ 27
13
14 ;a csatorna szama
15 vchan   equ 10
16
17 ;a valtozok beallitasa
18         evar 22,0
19         evar 23,0
20         evar 24,x_size
21         evar 25,y_size
22
23 ;a csatorna megnyitasa
24         ld a,vchan
25         ld de,vid
26         exos 1
27
28 ;a szekvenciak elkuldese
29         ld a,vchan
30         ld bc,seqlen
31         ld de,seq
32         exos 8
33
34 ;megjelenites (DISP)
35         ld a,vchan
36         ld bc,256+1
37         ld de,y_size*256+1
38         exos 11
39
40 ;a kepernyo kezdocimenek lekerdezese
41         ld a,vchan      ;a csat. szama
42         ld b,3          ;ADDR-lekerdezes
43         exos 11
44
45         push bc         ;BC-ben kapjuk meg
46                         ;a Nick-fele kezdocimet
47         pop hl          ;bekerul HL-be
48
49 ;a szegmensszam kepzese
50         ld a,b          ;a kezdocim felso resze
51                         ;(High) A-ba
52         and 11000000b   ;csak a felso 2 bit kell
53         rlca            ;beleptetes jobbrol
54         rlca            ;beleptetes jobbrol
55         or 11111100b    ;a felso 6 bit bevitele
56                         ;A-ban a szegmensszam!!!
57         ld b,a          ;B-ben a szegmensszam
58
59 ;a szegm.cim eltolasa a 16K-s (Page 1) tartomanyl
60         cp 255          ;szegm=255?
61         jr nz,seg254    ;nem
62         ld de,32768     ;igen,32K eltolas
63 seg254  cp 254          ;szegm=254?
64         jr nz,seg253    ;nem
65         ld de,49152     ;igen,48K eltolas
66 seg253  cp 253          ;szegm=253?
67         jr nz,seg252    ;nem
68         ld de,0         ;igen,nincs eltolas
69 seg252  cp 252          ;szegm=252?
70         jr nz,segpage   ;nem
71         ld de,16384     ;igen,16K eltolas
72
73 ;a Z80 cim kepzese a Nick cimmel es az eltolassal
74 segpage add hl,de        ;HL-ben Page 1 cim
75
76 ;P1 es P2 lapok mentese (2 szegmens esetere)
77         in a,(0B2h)     ;P2
78         push af         ;veremre
79         in a,(0B1h)     ;P1
80         push af         ;veremre
81         ld a,b          ;B-ben a szegm. volt
82         out (0B1h),a    ;a videoszegmensek
83         inc a           ;belapozasa a
84         out (0B2h),a    ;P1, P2-es lapokra
85
86 ;az osszes karakter megjelenitese
87         xor a           ;Ascii 0
88 cycl_0  push hl         ;kezdocim mentese
89         ld d,y_size     ;sorok szama
90 cycl_1  ld e,x_size     ;oszlopok szama
91 cycl_2  ld (hl),a       ;iras a memoriaba
92         inc hl          ;a kovetkezo hely
93         dec e           ;belso szaml. csokk.
94         jr nz,cycl_2    ;nem nulla -> cycl_2
95         dec d           ;kulso szaml. csokk.
96         jr nz,cycl_1    ;nem nulla -> cycl_1
97         pop hl          ;kezdocim visszaallitasa
98         inc a           ;kod es szamlalo novelese
99         jr nz,cycl_0    ;nem null -> cycl_0
100
101 ;a lapok visszaallitasa
102        pop af          ;veremrol
103        out (0B1h),a    ;P1
104        pop af          ;veremrol
105        out (0B2h),a    ;P2
106
107 ;a videocsat lezarasa
108        ld a,vchan
109        exos 3
110        ret
111
112 vid    db 6
113        defm "VIDEO:"
114 ;a szekvenciablokk
115 seq    db 27,"o"       ;kurzor kikapcs.
116        db 27,"C",255,32,32,255,0,0,0,0
117                        ;palettaszinek
118 seqlen equ $-seq       ;szekv. hossz
119        end
```

2. lista

A Pascal

8. rész

A függvények

Az előző folytatásban az eljárásokról beszéltünk. Most megismerjük az eljárások egy különleges fajtáját, a *függvényeket.*

A függvény egy olyan eljárás, amelyiknek saját értéke van. Ezt az értéket a *nevén* - azonosítóján - keresztül hordozza. Ez azt jelenti, hogy a függvényazonosító bármilyen kifejezésbe beírható (természetesen a típusösszeférhetőség figyelembe vételével), és ott úgy viselkedik, mint egy változó. Az eltérés az, hogy a változó mindaddig megtartja a korábban felvett értékét, amíg újat nem adunk neki, a függvény értéke viszont minden egyes hivatkozásnál újra kiszámítódik. A másik eltérés, hogy a függvénynek *paraméterei* (argumentumai) lehetnek, ugyanúgy, mint a normál (PROCEDURE) eljárásnak.

A Pascal nyelv számos beépített függvénnyel rendelkezik, ezeket majd később részletesen megismerjük. Most csak néhány példát. A valós típusú

SIN(X)

függvény a valós argumentum szinuszát számítja ki. Az

ORD(X)

függvény bármilyen megszámlálható típusú argumentum típuson belüli sorszámát adja meg. Így ha X karakter, ORD(X) a karakterkódot adja vissza. Az argumentum nélküli MAXINT függvény az ábrázolható legnagyobb egész számot adja eredményül.

Saját függvények

Természetesen saját függvényeink is lehetnek, ezeket a program (vagy más blokk) deklarációs részében kell meghatározni; a függvénydeklarációk az eljárásdeklarációkkal vegyesen szerepelhetnek. (Mint tudjuk, a Turbo-Pascalban a deklarációk egészen szabadon keveredhetnek egymással.)

A függvény deklarációját a FUNCTION kulcsszó vezeti be. A továbbiakban a függvény deklarációjára ugyanaz vonatkozik, mint az eljáráséra, tehát a formális paraméterek behelyettesítése az aktuális paraméterekkel ugyanúgy megtörténik majd híváskor, mint az eljárások esetében. Mivel a függvénynek értéke is van, a típusát is deklarálni kell, ezt a deklaráció végén kettőspont után megadott típusazonosítóval érhetjük el.

Egyetlen fontos szabály van még: a függvény törzsében a függvénynek értéket kell adni, azaz a függvény nevét szerepeltetni kell egy (esetleg több) értékadás bal oldalán. Ha ezt elmulasztjuk, a függvény a hívásakor értelmetlen értéket ad majd át.

Lássunk egy példát függvénydeklarációra:

```
FUNCTION SZINUSZNEGYZET( X : REAL ) : REAL;
BEGIN
  SZINUSZNEGYZET := SIN( X ) * SIN( X );
END { SZINUSZNEGYZET };
```

és a hívásra:

```
S := SZINUSZNEGYZET( PI / 2 )
+ KOSZINUSZNEGYZET( PI / 2 );
```

(feltételezve, hogy KOSZINUSZNEGYZET megírása SZINUSZNEGYZET ismeretében nem okozhat gondot).

Itt sommásan látjuk az elmondottakat: a függvényt, amelynek argumentuma és saját értéke is valós, az értékadást a függvényazonosítóra és a hívást egy konkrét esetben.

Egy másik:

```
FUNCTION TURESEN_BELUL( X, Y, EPS : REAL ) :
  BOOLEAN;
BEGIN
  IF ABS( X - Y ) <= EPS THEN
    TURESEN_BELUL := TRUE
  ELSE
    TURESEN_BELUL := FALSE
  { END IF };
END { TURESEN_BELUL };
```

Ez a függvény igaz (TRUE) értéket ad vissza, ha az X és az Y valós változó különbsége kisebb, mint az EPS változóval megadott tűrés, és hamis (FALSE) értéket ellenkező esetben. (A példabeli vizsgálat numerikus integrálás és egyéb iteratív számítások esetén fordul elő.)

(A Pascal persze lehetőséget ad ennek jóval egyszerűbb kiszámítására: a BEGIN ... END utasítászárójel-pár közé elég annyit beírni, hogy

```
TURESEN_BELUL := ( ABS( X - Y ) <= EPS );
```

hiszen a zárójelben álló kifejezés típusa logikai (BOOLEAN), és ez a szintén logikai típusú TURESEN_BELUL függvényazonosítónak értékül adható.)

Egyvalamivel óvatosnak kell lennünk: ha a függvény azonosítója a függvény törzsében értékadás jobb oldalán szerepel, ez rekurzív hívást eredményez, hiszen a függvényérték kiszámításához a függvényt ismételten meg kell hívni. Ez nem feltétlenül hiba, csak akkor, ha nem megfelelően alkalmazzuk.

Próbáljuk meg egy korábban deklarált tömb elemeit összeadni az OSSZEG függvénnyel!

```
TYPE
  ARRAYTYPE = ARRAY[ 1 .. 100 ] OF INTEGER;
...
FUNCTION OSSZEG( X : ARRAYTYPE; N : INTEGER ) :
  INTEGER;
VAR
  I : INTEGER; { CIKLUSSZÁMLÁLÓ }
BEGIN
  OSSZEG := 0;
  FOR I := 1 TO N DO
    OSSZEG := OSSZEG + X[ I ]
  { END FOR I };
END { OSSZEG };
```

A fordító szerencsére "kiadkad" azon, hogy az értékadás jobb oldalán az OSSZEG azonosítónál hiányoznak a paraméterek. A fordító ugyanis "tudja", hogy OSSZEG nem változó, hanem függvényazonosító. Ellenkező esetben a függvény újra és újra meghívná önmagát, amíg a verem meg nem telik, és a program vagy elszáll, vagy valamilyen hibaüzenetet kapunk. A feladat megoldása helyesen:

```
FUNCTION OSSZEG( X : ARRAYTYPE; N : INTEGER ) :
  INTEGER;
VAR
  I,          { CIKLUSSZÁMLÁLÓ }
  SZUMMA      { IDEIGLENES VÁLTOZÓ }
        : INTEGER;
BEGIN
  FOR I := 1 TO N DO
    SZUMMA := SZUMMA + X[ I ]
  { END FOR I };
  OSSZEG := SZUMMA;
END { OSSZEG };
```

Amint látjuk, be kellett vezetni egy ideiglenes változót (SZUMMA), amelyben az összegzést elvégezzük, és csak a számítás legvégén adjuk ezt az összeget értékül a függvényazonosítónak. OSSZEG így csak a bal oldalon szerepel értékadó utasításban, így nem lesz rekurzív hívás.

Ugyancsak rekurzív hívást eredményez, ha egy függvény ugyan nem önmagát hívja, hanem egy másikat, az viszont újra meghívja az elsőt. Tegyük fel, hogy egy grafikai programhoz gyors egész típusú szögfüggvényekre van szükségünk (a Pascal beépített sin és cos függvénye - logikusan - valós típusú). Matekóráinkról emlékszünk, hogy

$$\sin^2(alfa) + \cos^2(alfa) = 1$$

azaz

$$\sin(alfa) = \sqrt{1 - \cos^2(alfa)}$$

és

$$\cos(alfa) = \sqrt{1 - \sin^2(alfa)}$$

Gyorsan megírjuk a két függvényt (feltételezve, hogy az egész típusú INT_SQRT függvény már létezik):

```
FUNCTION INT_SIN( X : INTEGER ) : INTEGER;
BEGIN
  INT_SIN := INT_SQRT( 1 - INT_COS( X ) *
    INT_COS( X ) );
END { INT_SIN };
FUNCTION INT_COS( X : INTEGER ) : INTEGER;
BEGIN
  INT_COS := INT_SQRT( 1 - INT_SIN( X ) *
    INT_SIN( X ) );
END { INT_COS };
```

A program (a mindjárt ismertetendő kiegészítéssel) lefordul, de futtatáskor elszáll, hiszen a két függvény vég nélkül hívogatja egymást. Az ENTERPRISE-on is futó Turbo-Pascal 3.01 esetén például majdnem 3000 oda-vissza hívás történik, mielőtt a program belehalna. (Csak megjegyezzük, hogy ez a példa nem csak ebből az egy sebből vérzik, matematikailag is hibás.)

Persze, az ilyen típusú rekurzió is megengedett, ha megfelelően alkalmazzuk (gondoskodunk arról, hogy a hívások lánca valamikor megszakadjon).

Ügyeljünk arra, hogy bár a Pascal nyelv leírása tetszőleges típusazonosítót megenged a függvény típusának deklarációjában, a fordítók általában korlátozzák a megengedett típusokat. A Turbo-Pascal például csak skalár jellegű típust enged meg, így itt sem rekord, sem tömb nem használható. (A STRING skalárnak számít, az ARRAY OF CHAR viszont már tömb!)

Mit tegyünk, ha mindenképpen tömböt vagy rekordot kell visszaadnunk eredményül? Nagyon egyszerű: használjunk függvény helyett eljárást, az eredményt pedig változóparaméterként kaphatjuk meg. Ugyanezt tehetjük, ha a függvénynek egynél több paramétert kell visszaadnia. A fájlokkal végzett műveletekhez a FILE (és az ezzel rokon TEXT) változót név szerint kell paraméterként átadni, hiszen a művelet során változik az értéke:

```
TYPE
  NAME_TYPE = STRING;
  ACCESS_TYPE = ( FOR_READ, FOR_WRITE );
FUNCTION OPENFILE( VAR F      : TEXT;
  FILE_NAME : NAME_TYPE;
  ACCESS    : ACCESS_TYPE ) : BOOLEAN;
BEGIN
...
END { OPENFILE };
```

Itt az OPENFILE függvény (nem részletezett módon) megpróbálja megnyitni a fájlt, és ha ez sikerül, a függvényérték TRUE lesz, ellenkező esetben FALSE. A függvény hívása (pl. egy listázó programrészben) ilyen lehet:

```
IF OPENFILE( DATAFILE, 'DATA.DAT', FOR_READ )
THEN BEGIN
  WHILE NOT EOF( DATAFILE ) DO BEGIN
    READLN( DATAFILE, SZOVEGSOR );
    WRITELN( SZOVEGSOR );
  END { WHILE };
  CLOSE( DATAFILE );
END ELSE BEGIN
  WRITELN;
  WRITE( '*** Nem tudom megnyitni az adatfájlt! ' );
  READLN;
END { IF };
```

A példa egyúttal ízelítőt ad a helyes programozási stílusból (megjegyezve, hogy nem feltétlenül ez az egyedül üdvözítő megoldás); SZOVEGSOR valamilyen stringként van deklarálva, lehetőleg a példát tartalmazó eljárásban lokális változóként.

Az előzetes deklaráció

A két egymást hívogató függvény néhány példával korábbról úgy, ahogy van, nem fordítható le. A Pascal ugyanis minden esetben csak már deklarált objektumokra való hivatkozást fogad el. Ez érthető is, hiszen a fordító nem tud mit kezdeni egy azonosítóval, ha azt sem tudja, hogy az konstans, típus, változó vagy függvény, sem pedig, hogy milyen típusú. Így az INT_SIN függvény fordításakor az INT_COS azonosítót még nem ismeri, így hibát jelez. Mivel azonban ilyen jellegű rekurzióra szükség lehet, a probléma megkerülhető az ú.n. előzetes (FORWARD) deklaráció segétségével. Ennek a módja az, hogy a később definiálandó függvény vagy eljárás fejét felírjuk és pontosvesszővel elválasztva utánaírjuk a FORWARD kulcsszót. Ekkor már hivatkozhatunk a nevére, magát a függvényt vagy eljárást ráérünk később definiálni. A példa ennek fényében helyesen így néz ki:

```
FUNCTION INT_COS( X : INTEGER ) :
  INTEGER; FORWARD;
FUNCTION INT_SIN( X : INTEGER ) :
  INTEGER;
BEGIN
  INT_SIN := INT_SQRT( 1 - INT_COS( X ) *
    INT_COS( X ) );
END { INT_SIN };
FUNCTION INT_COS;
BEGIN
  INT_COS := INT_SQRT( 1 - INT_SIN( X ) *
    INT_SIN( X ) );
END { INT_COS };
```

Figyeljük meg, hogy a függvény tényleges definíciójában már nem adtunk meg sem típust, sem pedig a formális paraméterek listáját, hiszen mindkettő már ismert a fordító számára az előzetes deklarációból.

Ügyeljünk arra, hogy az egyes megvalósításokban a FORWARD deklaráció szintaksziss eltérő lehet!

Végül pedig lássunk egy példát a "hasznos" rekurzióra! Minden tankönyv a faktoriális számítását hozza fel példának a rekurzióra, hozzátéve, hogy egyáltalán nem ez a számítás egyetlen lehetséges (és legegyszerűbb!) módja; mi sem tesszük másként.

Mint tudjuk, *n!* az *n* szám faktoriálisát, azaz az összes természetes szám szorzatát jelenti 1-től *n*-ig. Így

```
1! = 1
2! = 1 x 2 = 2
3! = 1 x 2 x 3 = 6
4! = 1 x 2 x 3 x 4 = 24
```

stb., hozzátéve, hogy megállapodás szerint

```
0! = 1
```

Bár ennek alapján a faktoriális értéke egy 1-től *n*-ig menő ciklussal igen egyszerűen kiszámítható, a rekurzív függvényhívás menetének megértéséhez mégis az alábbi függvényt írjuk meg:

```
FUNCTION FACT( N : INTEGER ): INTEGER;
BEGIN
  IF N  0 THEN BEGIN
    FACT := N * FACT( N - 1 );
  END ELSE BEGIN
    FACT := 1;
  END { IF };
END { FACT };
```

A módszer azon alapszik, hogy a definíció alapján bármely n értéknél

```
n! = n * ( n - 1 )!
```

azaz bármelyik szám faktoriálisa kiszámítható a nála 1-gyel kisebb szám faktoriálisának és önmagának a szorzataként.

Hívjuk meg a függvényt a

```
WRITELN( FACT( 3 ) );
```

utasítással! A híváskor N felveszi a 3 értéket, így a függvény törzsében a feltételvizsgálat eredménye IGAZ; a függvény ki akarja számolni a 3 * 2! szorzatot, de ehhez szüksége van 2! értékére. Meghívja tehát újra a FACT függvényt, de most már N = 2 argumentummal.

Az új híváskor a függvény adatterülete újra generálódik, így N előző híváskori értéke nem vész el.

A függvény most újra végrehajtja a feltételvizsgálatot, és mivel N értéke (2) nagyobb nullánál, megint a szorzást hajtaná végre. Ehhez szüksége van 1! értékére, ehhez meghívja a FACT függvényt N = 1 argumentummal.

Az újabb hívásnál az előzőek szerint játszódik le minden, a függvény harmadszor is meghívja önmagát, most N = 0 argumentummal. Végre történik valami: a feltétel nem teljesül, az ELSE ág hajtódik végre, a függvény a FACT = 1 értéket kapja, és a vezérlés visszaadódik a hívás helyének.

A hívó azonban ugyanez a FACT függvény volt, ahol is N értéke 1. Végrehajtódik a FACT := 1 * 1 értékadás, a vezérlés - és a függvényérték - visszaadódik a hivónak. A hívó a FACT függvény, de most N értéke 2. Végrehajtódik a FACT := 2 * 1 értékadás, a hívó megint megkapja a függvényértéket. Itt N értéke 3, a szorzás eredménye átadódik a legelső hívónak, ez a főprogrambeli WRITELN utasítás, amely ki is írja a kapott 6 értéket. Ennyi az egész.

Aki még sohasem írt használt rekurziót, rajzolja fel magának egymás alá az egyes hívási szinteket, feltüntetve N és FACT értékét a hívások és visszatérések láncán haladva, mondjuk, FACT(4) esetén! Valahogy így:

Szint	Hívás vagy visszatérés	N értéke a függvény törzsében	FACT értéke visszatéréskor
0 -> 1	FACT(N)	4	-
1 -> 2	FACT(N-1)	3	-
2 -> 3	FACT(N-1)	2	-
3 -> 4	FACT(N-1)	1	-
4	nincs hívás	0	1
4 -> 3	visszatérés	1	1
3 -> 2	visszatérés	2	2
2 -> 1	visszatérés	3	6
1 -> 0	visszatérés	4	24

A rekurzív technika alkalmazásánál azért ügyeljünk arra, hogy bár elvileg a rekurzió tetszőleges mélységű lehet, gyakorlatilag mindig van egy határ, ez pedig a verem mélysége. Ne felejtsük el, hogy minden újabb rekurzív hívásnál a verembe kerül a visszatérési cím és a paraméterek, és ugyanide kerülnek az eljárás vagy függvény lokális változói. Amint láttuk, a Turbo-Pascal az ENTERPRISE-on néhány ezres szintet enged meg. Ha túl sok paramétert vagy változót használunk, a verem jóval hamarabb megtelik. Ha semmiképp nem elegendő a memória, fordítsuk le lemezre a programot, azaz készítsünk .COM fájlt és azt futtassuk (HiSoft Pascal esetén fordítsuk magnóra, azaz készítsünk betölthető programot), ekkor jóval több memóriát kapunk.

- UL -

IS-Forth

2.rész

A Forth Assembler

Az IS-Forth olyan szótárat is tartalmaz, amely a Z80 assembler programozást segíti. Olyan esetben, ha a kész programot (alkalmazást) fel kell gyorsítani, ezt úgy érik el, hogy a sebesség-kritikus részeket újraírják ("kihegyezik") Forth-assemblerben, majd az egész programot újrafordítják.

Az assembler, Forth, és a fordított lengyel jelölés

Mint a Forth része, az assembler is a fordított lengyel jelölés elvén dolgozik, amely szerint az operandusok (paraméterek) megelőzik a operációt (műveletet).

A Z80 regiszterek konstansként vannak definiálva, a többi utasítás pedig olyan szóként működik, amely a paramétereit a veremről szerzi meg, a helyes műveleti kód előállításához, majd pedig ezt a kódot fordítja be a következő szabad szótárbeli memóriahelyre. A regiszterek, és feltételkódok 16-bites kostansként vannak definiálva, a felső (magasabb helyiértékű) bájt tartalmazza regiszter típusát (8-bites, 16-bites), az alsó bájt pedig azt, hogy pontosan melyik regisztert kell a műveletnél (az előbb leírt formában) felhasználni. A típus bájt értékei a hexadecimális C000 felett vannak.

A regiszter konstansok és értékeik (hexadecimálisan):

B	F000	(BC)	F800
C	F001	(DE)	F810
D	F002	I	F807
E	F003	R	F80F
H	F004		
L	F005	BC	E000
(HL)	F006	DE	E001
A	F007	HL	E002
(IX+)	F606	AF	E004
(IX-	F706	IX	EE02
(IY+)	F406	IY	EC02
(IY-)	F506	SP	E003

A feltételkódok (a feltételes vezérlésátadó utsításokhoz) szintén konstansok. Az értékek a következőképpen alakulnak:

NC	F000
C	F001
NZ	F002
Z	F003
PO	F004
PE	F005
P	F006
M	F007

Látható, hogy az értékek megegyeznek a regiszteréknél definiáltakkal.

A IX és IY regiszterek a következőképpen használhatók: A Z80 (IY+d) ill. (IX-d) hivatkozások megfelelnek a (IY+) és (IX-) kódoknak olyan formában, hogy a eltolás (displacement) megelőzze a kódot. Például az

```
LD A,(IY+10)
```

utasítás a Forth-assemblerben a

```
10 (IY+) A LD,
```

szekvenciának felel meg.

Az utasításkészlet

Az utasítások a Zilog Z80 által használt mnemonikokat követik, persze a fordított lengyel jelölésnek megfelelően.

Itt következnek az utasítások (Z80 formában, és a megfelelő Forth frámódban):

Z80	Forth	Z80	Forth
ADC A,r	r A ADC,	IND	IND,
ADC A,n	n A ADC,	INDR	INDR,
ADC HL,rr	rr HL ADC,	INI	INI,
ADD A,r	r A ADD,	INIR	INIR,
ADD A,n	n A ADD,	JP (HL)	HL JP(),
ADD HL,rr	rr HL ADD,	JP (IX)	IX JP(),
ADD IX,rr	rr IX ADD,	JP (IY)	IY JP(),
ADD IY,rr	rr IY ADD,	JP nn	nn JP(),
AND r	r AND,	JP cc,nn	nn cc ?JP,
AND n	n AND,	JR d	addr JR,
BIT b,r	r b BIT	JR cc,d	addr cc ?JR,
CALL nn	nn CALL	LD (BC),A	A (BC) LD,
CALL cc,nn	nn cc ?CALL,	LD (DE),A	A (DE) LD,
CCF	CCF,	LD R,A	R A LD,
CP r	r CP,	LD I,A	A I LD,
CP n	n CP,	LD A,(DE)	(BC) A LD,
CPD	CPD,	LD A,(DE)	(DE) A LD,
CPDR	CPDR,	LD A,R	A R LD,
CPI	CPI,	LD A,I	I A LD,
CPIR	CPIR,	LD r,r'	r' r LD,
CPL	CPL,	LD r,n	n r LD,
DAA	DAA,	LD (nn),A	A nn () LD,
DEC r	r DEC,	LD (nn),HL	HL nn () LD,
DEC rr	rr DEC,	LD (nn),DE	DE nn () LD,
DI	DI,	LD (nn),SP	SP nn () LD,
DJNZ d	addr DJNZ,	LD (nn),BC	BC nn () LD,
EI	EI,	LD rr,nn	nn rr LD,
EX (SP),HL	HL SP () EX,	LD (rr),nn	nn rr () LD,
EX (SP),IX	IX SP () EX,	LD SP,HL	HL SP LD,
EX (SP),IY	IY SP () EX,	LD SP,IX	IX SP LD,
EX AF,AF'	AF AF EX,	LD SP,IY	IY SP LD,
EX DE,HL	HL DE EX,	LDD	LDD,
EXX	EXX,	LDDR	LDDR,
HALT	HALT,	LDI	LDI,
IM n	n IM,	LDIR	LDIR,
IN A,(C)	C A IN(),	NEG	NEG,
IN A,(n)	n A IN(),	NOP	NOP,
INC r	r INC,	OR r	r OR,
INC rr	rr INC,	OR n	n OR,
OTDR	OTDR,	RR r	r RR,
OTIR	OTIR,	RRC r	r RRC,
OUTD	OUTD,	RRD	RRD,
OUTI	OUTI,	RST n	n RST,
POP rr	rr POP,	SBC A,r	r A SBC,
PUSH rr	rr PUSH,	SBC A,n	n A SBC,
RES b,r	r b RES,	SBC HL,rr	rr HL SBC,
RET	RET,	SCF	SCF,
RET cc	cc ?RET,	SET b,r	r b SET,
RETI	RETI,	SLA r	r SLA,
RETN	RETN,	SRA r	r SRA,
RL r	r RL,	SRL r	r SRL,
RLA	RLA,	SUB r	r SUB,
RLC r	r RLC,	SUB n	n SUB,
RLCA	RLCA,	XOR r	r XOR,
RLD	RLD,	XOR n	n XOR,

A jelölések magyarázata:

r = 8-bites regiszter pl. A,
rr = 16-bites regiszter pl. DE,
n = 8 bites adat pl. 240 ha a számrendszer decimális,
nn = 16-bites adat
cc = feltételkód pl. PO,
b = bitszám pl. 4 - adott regiszter 4.bitjének befolyásolásához,
d = kettes komplemensű 8 - bites eltolás - relatív ugrásokhoz.

Amint az látható, a Forth-assemblerben a szavak végét vessző (,) jelzi.

Mindez a következőkre vezethető vissza:

- Forth konvenció, hogy azok a szavak amelyek a szótárba fordítanak, vesszővel végződjenek (pl. C,),
- a vessző jól elkülönîti az assembler-szavakat a Forth-szavaktól, s így nem lehet őket összekeverni (pl. XOR és XOR,),
- az általános Forth-assembler kód tartalmazhat több utasítást is egy adott soron belül (szemben a konvecionális assemblerek egy utasítás - egy sor szemléletével). A vessző a szavak végén az egyes utasítások szétválasztását (elkülöníthetőségét) segíti.

Forth-assembler szavak definiálása:

CODE név ... END-CODE

vagy

:név ... ;CODE ... END-CODE

ahol a *név* természetesen a szó nevét jelenti a ... pedig assembler-, illetve (a második forma első részében) Forth-szavakat jelent.

Az IS-Forth a következő regisztereket használja:

SP - paraméter verem (a legfelső két elem külön regiszterben !),
IY - visszatérési verem mutató,
IX - a belső interpreter (NEXT) címe,
HL' - interpreter pointer (a végrehajtásra következő szó címe),
DE' - a végrehajtandó szó CFA címe,
HL, DE - a paraméter verem legfelső két eleme.

vége

Légrádi Gábor

Az Életjáték (2)

Az előző számban megismerkedtünk az Életjáték szabályaival, és átgondoltuk a megprogramozásával kapcsolatos problémák egy részét. Most lássunk hozzá a programtervezésnek! Mielőtt bárki nekiesne, hogy

```
10 LET I = 1 TO ...
```

vagy a mindenre, csak korszerű programozásra nem alkalmas folyamatábrát kezdenénk rajzolni, próbáljunk nagyobb léptékben gondolkodni. A program két nagyobb egységre bontható. Az egyikben mindenféle beállítást végzünk: kezdőértéket adunk a használt változóknak, letöröljük a képernyőt, beállítjuk a kiinduló konfigurációt stb. Nevezzük ezt a részt ELŐKÉSZÍTÉS-nek. A másik programrész a tulajdonképpeni játék, amelyben az újabb és újabb állapotokat generáljuk. Legyen ennek a neve JÁTÉK. Programunk tehát most így néz ki az ú.n. struktogramm-technikával:

ÉLETJÁTÉK:

ELŐKÉSZÍTÉS
JÁTÉK

és így az ennek megfelelő pszeudonyelven:

```
ÉLETJÁTÉK:
  ELŐKÉSZÍTÉS;
  JÁTÉK;
END (ÉLETJÁTÉK)
```

Mielőtt bármilyen elhamarkodott következtetést tennénk e két nagyobb egység felépítésére vonatkozóan (tehát mielőtt

```
100 FOR I = 1 TO 24
110     FOR J = 1 TO 40
```

jellegű dolgokat kezdenénk írkálni), gondoljuk át, milyen módon lehet az egymást követő generációkat a gépen ábrázolni! Ehhez nyilván valamiféle táblázatra lesz szükségünk. A feladat megfogalmazásából azonban az is következik, hogy egyetlen táblázat nem elegendő, hiszen az adott elemkonfiguráció csak akkor változhat meg, ha már a teljes jelen konfigurációt kiértékeltük, ellenkező esetben a megoldás módjától függő, hamis eredményt kapnánk.

Kézenfekvő volna egy olyan táblázatot használni, amelynek minden eleme két-két állapotot tárolna: a jelenlegit és a kiértékelés alatti következőt. Sajnos, a BASIC nyelvek többsége, így az IS-BASIC sem engedi meg összetett típusok használatát. Maradna egy háromdimenziós tömb lehetősége, amelyben az első két dimenzió értelemszerűen a sor- és az oszlopkoordináta volna, a harmadik pedig mindössze két értékkel bírna, a jelenlegi és a következő állapot tárolására. Sajnos, az IS-BASIC csak kétdimenziós tömböt enged meg. Summa-summárom, bármennyire ellenkezik a józan programozói gondolkodással, marad a két különböző táblázat használata. Ennél most ne is menjünk tovább.

A program pszeudokódja most így néz ki:

```
ÉLETJÁTÉK:
VAR
  TÁBLA-1
  TÁBLA-2
BEGIN
  ELŐKÉSZÍTÉS;
  JÁTÉK;
END
```

A program JÁTÉK részéről tudjuk, hogy ismétlődő folyamat:

```
JÁTÉK:
  REPEAT
    LÉPÉS
  END-REPEAT;
END (JÁTÉK)
```

Ez grafikusan így néz ki:

JÁTÉK:

REPEAT
LÉPÉS

Az ELŐKÉSZÍTÉS részt pedig így bontjuk ki:

```
ELŐKÉSZÍTÉS:
  ALAPHELYZET (TÁBLA-1, TÁBLA-2);
  KEZDŐKONFIGURÁCIÓ (TÁBLA-1);
  EGYÉB;
END
```

Önkényesen TÁBLA-1-et választottuk a kezdő konfiguráció számára. A másik alternatíva ugyanilyen jó lett volna; mivel a döntésnek nincs igazán jelentősége, nem töprengünk rajta sokáig (a tett halála az okoskodás).

Ennek grafikus ábrázolását az olvasóra bízzuk. Ha valaki a teljes programot fel akarja rajzolni, elegendő a magasabb szintű ábrába a megfelelő téglalapok helyére berajzolni azok részletesebb rajzát. Mi azonban nem grandiózusságra törekszünk, hanem egyidőben csak egy-egy jól definiálható részfeladatra szeretnénk koncentrálni.

Most értünk el oda, hogy a táblázatok pontosabb definiálását már nem lehet tovább halasztani. Kézenfekvő, hogy valami ilyesfajta deklarációjuk legyen:

```
VAR
  TÁBLA-1,
  TÁBLA-2 : ARRAY [SOROK, OSZLOPOK] OF
            ADATELEM;
```

Egyáltalán nem sietünk viszont meghatározni, mit értünk sorokon és oszlopokon, sem pedig adatelemen.

Ha viszont ismerjük a táblázatok szerkezetét, pontosíthatjuk a JÁTÉK programrész LÉPÉS elemét:

```
LÉPÉS:
  SOROKON-VÉGIG DO
    OSZLOPOKON-VÉGIG DO
      TEDD-AMIT-KELL;
    END-DO;
  END-DO;
END (LÉPÉS)
```

Grafikusan:

SOROKON VÉGIG
OSZLOPOKON VÉGIG
TEDD AMIT KELL

Tovább nem tudunk menni, mert nem tudjuk, MIT-KELL-TENNI (sic). Előbb valamiféle döntést kell hoznunk az adatok ábrázolásáról. Egy pillanatra megfeledkezve rossz BASIC-es beidegződéseinkről és az azzal járó 0 és 1 vagy hasonló választékról, meghatározzuk a táblázatelemek típusát:

```
TYPE
  ADATELEM : (ÉL, NEM-ÉL);
```

Ennek alapján most már elég részletes megoldást tudunk adni az ALAPHELYZET programrészre:

```
ALAPHELYZET:
  SOROKON-VÉGIG DO
    OSZLOPOKON-VÉGIG DO
      TÁBLA-1 [SOR, OSZLOP] := NEM-ÉL;
      TÁBLA-2 [SOR, OSZLOP] := NEM-ÉL;
    END DO
  END-DO
END (ALAPHELYZET)
```

A két táblázatot alaphelyzetbe állíthattuk volna két külön ciklusban is, de intuitív módon gyorsabbnak találtuk ezt a megoldást. A kezdő konfiguráció beállításával most nem törődünk; a program kipróbálásához majd megadunk valami egyszerű elrendezést.

Itt az ideje, hogy komolyan belenézzünk a megoldás részleteibe. Jól tudjuk, hogy egy adott cellában az ott lévő sejt megmaradása vagy elpusztulása, illetve egy üres cellában egy sejt születése vagy nem születése valamilyen módon a cella szomszédaitól, egészen pontosan a szomszédai számától függ. Ez pontosan elegendő információ a folytatáshoz. Definiálunk egy FÜGGVÉNY nevű függvényt, amelynek bemenő paraméterei a szomszédok száma és a cella saját állapota (ÉL vagy NEM-ÉL), a függvényérték pedig ugyanilyen típusú, vagyis azt mondja meg, hogy a következő pillanatban ugyanitt lesz-e sejt vagy nem.

```
TEDD-AMIT-KELL:
  TÁBLAEELEM (MÁSIK-TÁBLA) :=
    FÜGGVÉNY (SZOMSZÉDOK,
    TÁBLAELEM (EGYIK-TÁBLA));
END (TEDD-AMIT-KELL)
```

Most segítene, ha a táblázatokat értelmesen meg tudtuk volna határozni (sebaj, majd egy értelmesebb programnyelven!), így kénytelenek vagyunk egy ostobább megoldást alkalmazni (a "...maga mindent kétszer mond, kétszer mond..." fajtából): *(folytatás a 15. oldalon)*

LEMEZMÁSOLÓ PROGRAM

Ha csak egy meghajtóval rendelkezünk, nem célszerű a szektorról szektorra másoló DISKCOPY használata. A 16 kilobájtos puffert használó program csak két meghajtó esetén másol gyorsan. A DCOPY csak két azonosra formázott lemezt képes kezelni, azonban kihasználja a teljes szabad memóriát. Alapkiépítésű gép esetében 5-ször kevesebb csere szükséges. A szegmensek maximális száma 32, ennyit kezel az IS-DOS. A

DCOPY a: b:

parancs kiadása után bejelentkezik, majd várakozik az ENTER lenyomásáig, ezzel lehetőséget adva a lemez kicserélésére.

```
100 PROGRAM "dcopy.bet"
110 OPEN #1:"dcopy.com" ACCESS OUTPUT
120 DO
130   READ IF MISSING EXIT DO:A$
140   DO UNTIL A$=""
150     PRINT #1:HEX$(A$(:2));
160     LET A$=LTRIM$(A$(3:))
170   LOOP
180 LOOP
190 CLOSE #1
200 DATA C33A 010D 0A44 434F 5059 2056 312E 3020
210 DATA 2048 736F 6674 2031 3939 312E 0D0A 5072
220 DATA 6573 7320 454E 5445 5220 746F 2063 6F6E
230 DATA 7469 6E75 652E 2E2E 0D0A 3EFF 1103 0101
240 DATA 1B00 F708 C227 0221 B502 E53A 0700 E6C0
250 DATA FEC0 3EF7 C227 02DB B123 77DB B223 77E5
260 DATA 0E82 CD05 00E1 2371 28F5 D1B7 ED52 2DEB
270 DATA 7321 8100 CD17 02D6 4132 AF02 5F3E 3ACD
280 DATA 1E02 3E20 CD1E 02CD 1702 D641 32B0 02BB
290 DATA 3EAF CA27 023E 3ACD 1E02 CD17 02B7 C221
300 DATA 023E FF11 1E01 011C 00F7 08C2 2702 3EFE
310 DATA F705 1100 3C0E 1ACD 0500 1100 002A AF02
320 DATA 2601 0E2F CD05 0020 5E2A 133C 2B22 B102
330 DATA 1100 400E 1ACD 0500 1101 00CD 6202 1100
340 DATA 3E0E 1ACD 0500 1100 002A B002 2601 0E2F
350 DATA CD05 0020 323A 153C 2A15 3EBD 2027 1100
360 DATA 400E 1ACD 0500 1101 00CD 2F02 CD62 022A
370 DATA B302 7CB5 20F3 C77E 23FE 2028 FAC9 BE23
380 DATA C83E AE18 023E 9647 B70E 80CD 0500 C721
390 DATA B502 237E D3B1 D92A B302 0120 00AF ED42
400 DATA 3004 094D 2E00 22B3 02C5 D9C1 0C0D C8E5
410 DATA 2AB0 0261 C50E 30CD 0500 20CB E119 EBE1
420 DATA 18D0 2100 0022 B302 21B5 0246 D5C5 237E
430 DATA D3B1 E52A B102 0120 00B7 ED42 3004 094D
440 DATA 2E00 22B1 020C 0D28 1E2A B302 0922 B302
450 DATA 2AAF 0261 C50E 2FCD 0500 FEA4 2889 FEE5
460 DATA 2885 E119 EBAF 3CE1 C128 0210 C0D1 C901
470 DATA 0200 0000 00
```

MOVE RENDSZERBŐVÍTŐ

A géptulajdonosok nagy része televízióval használja az ENTERPRISE-t. Legtöbb esetben a kép nincs a képernyő közepén. Ezen kíván segíteni a MOVE bővítő. A parancskészletét lekérhetjük a :HELP MOVE parancs segítségével (LEFT, RIGHT, UP, DOWN). A parancs több rendszerből is kiadható, pl. BASIC-ból :UP, WP-ből F8 UP. A videocsatorna ismételt megnyitása esetén újra be kell állítani a vízszintes pozíciót.

```
100 PROGRAM "move.bet"
110 OPEN #1:"move" ACCESS OUTPUT
120 DO
130   READ IF MISSING EXIT DO:A$
140   DO UNTIL A$=""
150     PRINT #1:HEX$(A$(:2));
160     LET A$=LTRIM$(A$(3:))
170   LOOP
180 LOOP
190 CLOSE #1
200 DATA 0006 B101 0000 0000 0000 0000 0000 0000
210 DATA 79FE 0228 28FE 03C0 78B7 200D D911 3FC0
220 DATA 0118 003E FFF7 08D9 C921 C8C0 CD86 C0D0
230 DATA 113F C001 4700 3EFF F708 0E00 C921 D0C0
240 DATA CD86 C0D0 E94D 4F56 4520 2064 6973 706C
250 DATA 6179 2070 6F73 6974 696F 6E0D 0A28 6329
260 DATA 2048 534F 4654 3A20 3139 3931 2E20 436F
270 DATA 6D6D 616E 6473 3A20 4C45 4654 2C52 4947
280 DATA 4854 2C55 502C 444F 574E 0D0A C5D5 D9D1
290 DATA C1D9 1348 0600 CD99 C03E 004F D8D9 C97E
300 DATA 23B7 C8B9 2809 C602 856F 8C95 6718 F0C5
310 DATA D51A 13ED A120 10EA ABC0 7E23 666F D9D1
320 DATA C11A 90D9 4737 C909 2323 D1C1 18D1 044D
330 DATA 4F56 4500 0000 044C 4546 54EC C005 5249
340 DATA 4748 54F0 C002 5550 A1C1 0444 4F57 4EAE
350 DATA C100 0EFF 1802 0E01 DD21 1089 1110 0006
360 DATA 18DD 7E02 E63F 67DD 7E03 E63F BC38 0ACD
370 DATA 91C1 DD23 CD91 C1DD 2BDD 1910 E421 C0BF
380 DATA 2B7E 2BB7 2004 0E00 AFC9 562B 5ED3 B121
390 DATA 0700 19D5 118B C106 061A 13BE 2328 03E1
400 DATA 18DE 10F5 D1DB B147 D921 BDBF 2B7E 2BB7
410 DATA 2004 0E00 AFC9 562B 5ED3 B1D5 DDE1 D90D
420 DATA 7EF2 BB20 28DD 7EF3 BA20 22DD 7EF4 BB20
430 DATA 1CF3 DBB0 F5DB B13D D3B0 D511 DDFF DD19
440 DATA D1CD 91C1 DD23 CD91 C1F1 D3B0 FBD9 EB18
450 DATA BB05 5649 4445 4FDD 7E02 67E6 C06F 7C81
460 DATA E63F B5DD 7702 C921 108B 3420 0235 C921
470 DATA C0BA 35C9 21C0 BA34 2002 35C9 2110 BB35
480 DATA C9
```

FCODE BASIC-BŐVÍTŐ

Az IS-BASIC támogatja a gépi kódban írt rutinok használatát. Sok időt takaríthatunk meg, ha ezt assemblerrel készítjük. A CODE segítségével történő munkaigényes adatbevitelt, jelentősen egyszerűsíti az FCODE bővítés használata. Az utasítás ellenőrzi a fejléc alapján, hogy a kód elfér-e az ALLOCATE parancs által lefoglalt területen, valamint hogy 2-es vagy 5-ös típusról van-e szó. Hiba esetén a futás megszakad.

A kódfájl létrehozása:

Ha a kód abszolút címzést nem tartalmaz, elkészíthető 5-ös fejléccel, pl. ASMON vagy GEN használatával.

Abszolút címzés használata esetén áthelyezhető formában kell a kódot lefordítani, azaz 2-es fejléccel, pl. GEN "A,,R 2" fordítással.

Abszolút címzés mellett 5-ös fejlécet csak akkor használhatunk, ha a pontos betöltési címet ismerjük, valamint ez a cím nem fog esetleg más alkalommal módosulni. Változást okoz pl. BASIC-bővítő, vagy több ALLOCATE-kód használat. A pontos címet lekérdezhetjük a PRINT PEEK(540)+PEEK(541)*256 utasítással. Ezt a címet adjuk meg ORG kezdőcím-direktívaként kódunk elején.

Beolvasás:

Nyitott csatornáról:

```
OPEN #1:"ESZKÖZ:FÁJL"
FCODE változó=#1 vagy FCODE =#1
CLOSE #1
```

Eszköz : fájlról (a 106-os csatorna lesz megnyitva, majd lezárva).

```
FCODE változó="ESZKÖZ:FÁJL"
```

vagy

```
FCODE ="ESZKÖZ:FÁJL"
```

(Hsoft)

```
100 PROGRAM "fcode.bet"
110 OPEN #1:"fcode" ACCESS OUTPUT
120 DO
130   READ IF MISSING EXIT DO:A$
140   DO UNTIL A$=""
150     PRINT #1:HEX$(A$(:2));
160     LET A$=LTRIM$(A$(3:))
170   LOOP
180 LOOP
190 CLOSE #1
200 DATA 0002 D600 0000 0000 0000 0000 0000 0000
210 DATA 7696 C640 2759 7846 563D 2CC4 0F80 0C06
220 DATA 0038 0ACE 2C90 0000 0300 C000 103C 0204
230 DATA 8014 80A4 6219 3C88 456B 8F40 0C91 D008
240 DATA 04FE 1008 030D 7460 01DA 5B0E 0080 27F0
250 DATA 01CC 4672 119C 0467 0119 C046 716B 8800
260 DATA 03E0 985C 4400 7F00 C501 768B 9000 3E35
270 DATA 0CA3 F003 B4B6 2801 04DE E016 FB5C 2C00
280 DATA 1F3F C300 768B F000 328D 200A F194 6600
290 DATA 1D46 2000 0840 0011 2480 DEE0 6101 4842
300 DATA 1712 0449 2030 D2DC 4044 0986 9FC0 27A8
310 DATA A008 FE02 8807 2ED2 592C 06ED 2D87 0042
320 DATA A100 096E ED29 3B48 4210 0050 7024 1507
330 DATA 0040 9788 EA13 000A 008F 7030 6004 F70F
340 DATA 0802 C220 E009 EA3A 82C0 0B71 4018 7503
350 DATA C01E E036 FBC5 92CD 9E1F EE76 6CEC FF6F
360 DATA 8000 1800 0A00 0000 0000 0000 0000 0000
370 DATA 0000 00
```

Elkészült az EPDOS version 1.2 lemezkezelő program. Mintegy 50 funkciós menü, 6-féle diszkeditor, univerzális FORMAT, COPY, DCOPY, DEL, UNDEL, CHKDSK, gyors CD, nyomtatás stb. A program másik része bármely rendszerből ható parancsbővítéseket tartalmaz, pl.: EXOS, VIDEO, MONITOR és DISK direkt utasításokat. A program ára 32 K-s epromban, gyorsteszttel: 900 Ft.

Buszkiterjesztő egység több bővítőkártya illesztésére

Mint ismeretes, az ENTERPRISE lehetőségeit kibővítő különböző kártyák (pl. a lemezvezérlő) a gép jobboldalán található "SYTEM BUS" (rendszerbusz) csatlakozósávra dugaszolhatók. Ezen a csatlakozón keresztül hozzáférhetünk a gép hardveréhez, rendelkezésre áll az összes cím-, és adatvonal, a mikroprocesszor vezérlővonalai, külső sprite-bemenetek, és még egy pár "hasznos" vezeték, így szinte mindent megvalósíthatunk, amire csak szükségünk lehet, sőt, néhányat a szükségtelen dolgok közül is.

A gond akkor jelentkezik, ha több kártyát (pl. lemezvezérlő + óra/naptár) óhajtunk egyszerre a géphez csatlakoztatni. Ehhez egyrészt nincs elég hely, másrészt a gép vonalai sem terhelhetők tetszőlegesen. Három, vagy még több kártya csatlakoztatásához már feltétlenül kell néhány meghajtó áramkör. Mindezek miatt terveztem az öt+egy kártya csatlakoztatását lehetővé tévő buszkiterjesztő egységet, amelyet a következőkben mutatok be.

Tekintsük a kapcsolási rajzot (*1. ábra*)! Mint látható, a vonalaknak mintegy a felét erősítjük. U3, U2 és U4 illeszti a címvezetékeket (A0-A21), U1 pedig a kimenő vezérlőjeleket. C3-C6 hidegíti az integrált áramköröket, és U5 állítja elő számukra a +5V tápfeszültséget. C1 az ENTERPRISE-ból jövő +9V-ot szűri, C2 pedig a stabilizátort hidegíti. Mind a +9V, mind a +5V megjelenik a kártyafoglalatokon, így a kisebb kártyák használhatják a buszkiterjesztő stabilizátorát, a nagyobbak pedig saját maguk stabilizálhatják tápfeszültségüket (ami a zavarérzékenységet is csökkenti) a +9V-ból.

Említést érdemel még a kártya géppel ellentétes oldalán található panelélcsatlakozó, amely két célra alkalmazható. Egyrészt úgy, ahogy van, kompatíbilis a lemezvezérlő kártya tartozékaként forgalmazott ún. "Bus Bridge"-dzsel, így egy kártyacsatlakozó fölforrasztása után közvetlenül ide dugható a gyári lemezvezérlő kártya, vagy a Spectrum-emulátor. Másrészt némi átalakítás (egy fűrészelés, egy vágás, egy forrasztás) után ugyanolyan panelélcsatlakozót kaphatunk belőle, mint a gépé, így ide az eredetileg a gépbe dugható kártyákat (gigantomániások figyelem: akár még egy buszkiterjesztő egységet is, összesen tíz+egy kártya számára!) tehetjük.

Fölmerülhet a kérdés: miért csak kimeneti vonalakat erősítünk, és miért pont azokat? A válasz három részből áll.

Először: a kimaradt kimenetek vagy speciálisak (pl. HALT, -RFSH), amelyeket valószínűleg csak egy kártya használ, vagy amúgy is LS TTL áramkörökről jönnek (ilyen a 8 MHz, ezért van szüksége valakinek az ENTERPRISE kapcsolási rajzára!), ezért nem igényelnek külön meghajtót.

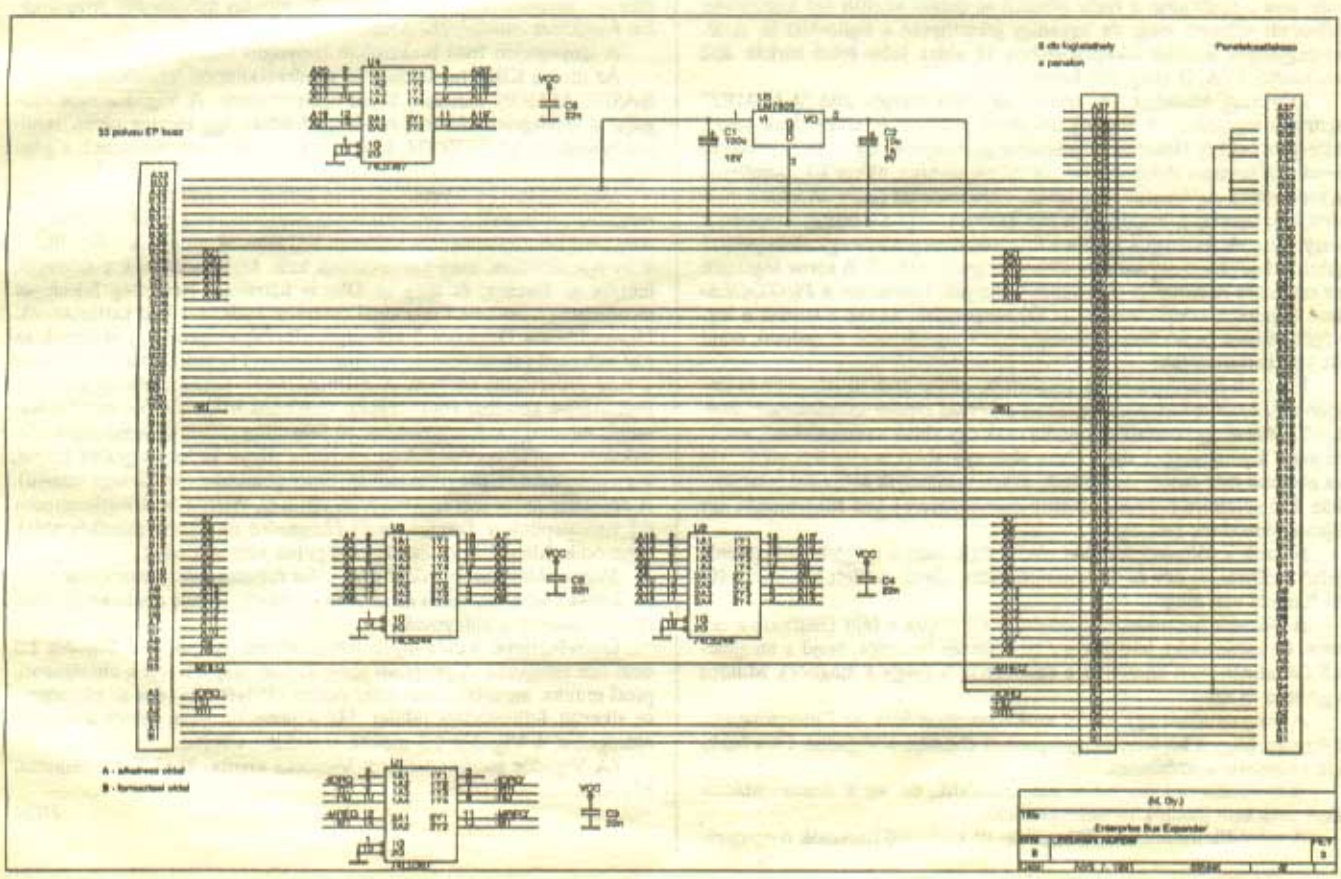
Másodszor: a számítógép digitális bemenetei legfeljebb egységnyi terhelést jelentenek a külvilág számára, amelyekkel jól megbirkóznak a külső kártyák rendszerint LS TTL kimenetei.

Harmadszor: a kétirányú adatbusz egy központi kártyán, kis késleltetéssel történő erősítése komplikált. Ehhez minden bővítőkártyán ki kellene alakítani egy olyan jelet, ami a megfelelő címtartományban az adatmeghajtót vezérli, és ezen jeleket kell a buszkiterjesztőn logikai VAGY-kapcsolatba hozni. Ez még nem olyan nagy feladat, hiszen az említett jelekre az egyes kártyákon általában szükség van, ha nem a központi, akkor a kártyán lévő adatmeghajtó vezérlésére. A probléma abban áll, hogy annak a bizonyos VAGY-kapcsolatnak a létrehozása számottevő késleltetést okoz, így a központi kártyán történő adatbuszerősítés esetén célszerű lenne elhagyni a kártyákon lévő adatmeghajtókat. A kérdést a lemezvezérlő kártya döntötte el. Mivel azon rajta van az adatmeghajtó áramkör, én is mindegyik kártyára ráterveztem azt, ezáltal a központi meghajtó elhagyhatóvá vált, elkerülve ezzel az adatvonalakon fellépő további jelkésleltetést.

A kártya fizikai megvalósításánál a fő szempont az volt, hogy a meghajtóáramkörök minél közelebb kerüljenek a számítógéphez, mivel a gép kíméletlenül megtorolja az akár csak tíz cm-es toldóvezetéket a rendszerbuszán: bizonytalanná válik a működése, állandóan lefagy, stb., stb. A meghajtóáramkörök után már viszonylag messze is elvezethetők a jelek, a RAM-bővítő kártyát leszámítva, az ugyanis a géphez legközelebbi foglalatban érzi legjobban magát.

Ha valamely Nyájas Olvasót érdekli a buszkiterjesztő egység, illetve az ahhoz csatlakoztatható többi bővítőkártya, a következő címen érdeklődhet:

M. GY. Hard-Szoft
Mészáros Gyula, 1029 Bp., Zsíroshegyi út 110.



Vigados 1.3: *Veled vagy nélküled?*

A Mikrovilág olvasói előtt nem újdonság a Vigados 1.3 lemezkezelő program, amelyről ott rövidebb leírást közöltek. Ezért fogadtuk nagy örömmel a KEYSOFT hozzánk elküldött csomagját, mert így alkalmunk nyílt a gyakorlatban is megismernünk.

Tesztkonfigurációnk egy alapgépből, egy színes monitorból, egy 3.5"-os és egy 5.25"-os lemezegységből állt. A készítő ajánlása ellenére nem használtunk memóriabővítést, sem egeret.

A program elindítása után megjelenik a főmenü és a copyright szöveg. Amennyiben a program megsérült, vagy illegális verzió, úgy a bejelentkező képernyő tájékoztat erről. *A program bizonytalanul indul*: a rendszert a 3.5"-os "gyári" (értsd Enterprise) lemezegységről próbáltuk betölteni. A harmadik rendszerindítás után a Vigados valahogy összeszokott a meghajtóval, így végre kegyeskedett betöltődni. Ez különösebben nem volt szimpatikus...

Ha sikerült a startolás, akkor a főmenü és öt ablak jelenik meg, mindegyik ablak egy-egy lemezegységhez kapcsolódik (A: - E:). Szokatlan, hogy a *rendszer alapértelmezésben 4 fizikai lemezegységet feltételez* a konfigurációban, holott a gazdagabb felhasználóknak sincs kettőnél több.

A Vigados attribútum típusú grafikus felületen kommunikál a felhasználóval, magyar nyelven. Messze nincsenek kihasználva a képernyőtípus lehetőségei. A főmenü és az ablakok színének megválasztása nem esztétikus, kissé izlésrontó. A főmenü *Infó* pontja például világos mustársárga alapon, világoszöld betűkkel van kiírva. (Egyébként csak a dokumentációból tudtuk megállapítani, hogy "Infó" van odaírva.)

A Vigados legördülős menürendszerének egy részénél szövegekre, más részénél ikonokra kell rámutatnunk (egérrel, külső illetve belső botkormánnyal), de a vezérléssel van egy kis baj! *A menüvezérlésnek ugyanis az a lényege, hogy a felhasználónak kényelmes használatot biztosítson. Nos, a Vigados ezt nem teljesíti*, de erre alább még visszatérünk.

A programot először a *szép kivitelű, jónak minősíthető, 8 oldalas dokumentáció* nélkül próbáltuk használni. Ezzel a program tanulhatóságát, egyértelműségét akartuk vizsgálni, és a funkciók felének használatára sikerült is rájönnünk. Ezután a dokumentációhoz fordultunk, de annak elolvasása után sem sikerült mindent (*Mozgatás*) kielégítően tisztáznunk.

A főmenü *6 főfunkciót* tartalmaz, sok mást (tartalomjegyzék megjelenítés, fájl-, könyvtárkijelölés, könyvtárváltás, mozgás a könyvtárban stb.) közvetlenül a meghajtókhoz tartozó ablakokban kell elvégeznünk. Az egyes "ablakfunkciókat" az ablakok tetején álló ikonokkal aktivizálhatjuk. Véleményünk szerint *a nehezen felismerhető ikonok egy része nem utal a funkcióra*: a radír például az összes kijelölt fájl kijelöltségi állapotát szünteti meg, de ugyanígy jelenthetné a fájltörlést is. A lemezegységek közötti átkapcsoláshoz az ablak jobb felső sarkán álló azonosítóra (A, B stb.) kell kattintanunk.

Directory kéréshez (*A: egység*) az ablak tetején álló "A-LEMEZ" felíratra mutatunk. A katalóguslistában a fájlok és könyvtárak nevét, kiterjesztésüket láthatjuk *alfabetikus sorrendben. Ez a sorbarendezés rendkívül hasznos dolog* (ha valakit mégis zavarna, akkor kikapcsolhatja a rendezést). A Vigados megjelenti a jellemzőket (csak olvasható, hidden, rendszerfájl), ezenkívül a fájl méretét vagy a készítés időpontját vagy dátumát írathatjuk ki. Ha a listában szereplő név egy alkönyvtárat jelöl, akkor előtte egy sarkára állított négyzet látható. A sorok legelején az esetleges kijelöltségi sorszámot láthatjuk, hasonlóan a *PiciTOOLS*-hoz. Sajátos a könyvtárváltás (CD) megoldása: annak a sornak a legvégére állunk (a semmire), amelyben a könyvtár neve olvasható, majd itt kell kattintanunk.

Ha több lemezegységgel dolgozunk, akkor *csak az aktuálishoz tartozó ablak látható tökéletesen. Ezzel a készítő eredeti szándékának ellenkezőjét érte el*: gyakorlatilag mindig csak egy ablak hasznosítható, a többi nem, a programhoz ilyen elven akár egy ablak is elég lett volna. Ha az ablakok már összekuszálódtak, akkor valamelyik ablak bal felső sarkán álló valamire (egyesek szerint hamburgerre) kell rákattintani, így újrarendeződnek (arrange).

Nézzük a főfunkciókat! Az első a **Fájl**, ahol a megjelenítés (view), jelző beállítás, új név és új dátummegadás, törlés mellett három további funkció van még:

A *másolás* használata egyértelmű: kijelöljük a fájlt (ráállunk a nevére, és kattintunk), lehívjuk a *Fájl-Másolás* funkciót, majd a megjelenő *Cél* almenüben kijelöljük a célmeghajtót (vagy a magnót). Mindez egyszerű és szép.

A *programtöltés* egy eddig szokatlan megoldás az Enterprise-nál. Lényege, hogy a kiválasztott programot elindítja a Vigados. Ez a funkció *megnyerte tetszésünket*.

A *mozgatás* valahogy biztosan működik, de mi a dokumentációt átolvasva sem jöttünk rá használatára...

A második főfunkció a **Könyvtár**. Itt található funkciók megegyeznek a *Fájl*-éval, de értelemszerűen könyvtárakra vonatkoznak. A mozgatást itt sem tudtuk kipróbálni, ugyanakkor nagyon jól használható a *Szüntet* funkció. Ezzel egy teljes könyvtárat törölhetünk úgy, hogy vannak benne fájlok. *Az EXDOS nem tud ilyet!*

A következő főfunkció a **Speciál** névre hallgat. Itt főleg az EXOS rendszerváltozókat állíthatjuk. *A funkciónak szépre sikerült, egyértelmű jelentéssel bíró ikonokat tartalmazó almenüje van, ilyenre kellett volna készíteni az egész Vigadost.* Műveleteket végezhetünk

- a *lemezekkel*: diskcopy, lemeznév, formattálás. (*Ez utóbbi rendkívül jól* 850, 840, 800, 720, 440, 430, 400 és 360 KB-ra formattálhatunk. Ha a 720 KB-os lemezt 850 KB-ra formázzuk, akkor azt az EXDOS a Vigados nélkül is tudja használni. Erre csak annyit mondhattunk: *Nem semmi!*)
- a *lemezegységekkel*: hozzárendelés (A: meghajtóhoz rendelhetjük a B:-t stb.), ellenőrzés, léptetési sebesség.
- a *memóriával*: ramdisk-et nyithatunk, statisztikát kérhetünk a memória állapotáról.
- az *egérrel*: kurzor mozgási sebesség.
- a *billentyűzettel*: az ismétlés és a várakozás ideje, click hang.
- a *hangszóróval*: ki illetve bekapcsolhatjuk.
- a *magnóval*: REM1, REM2, hang ki/be, felvételi szint, felvételi sebesség.
- a *nyomtatóval*: kinyomtatja az aktuális ablak tartalmát.
- a *képernyővel*: a lemezegységekhez tartozó ablakok nyitása, zárása, mozgatása lehetséges itt.
- az *órával*: olvasni, beállítani lehet. Ha a felhasználó birtokában van a KEYSOFT Speed-Test EPROM-jának, akkor az órakijelzés folyamatos.

Az **Opciók** főfunkciónál az ablakokban megjelenítendő fájljellemzőket tudjuk megadni: kívánságra láthatjuk a rejtett és a rendszerfájlokat; eldönthetjük, hogy a fájl neve után méretét, keletkezési dátumát vagy idejét akarjuk-e látni. Itt kapcsolható a ABC szerinti rendezés, és a biztosítás: ha valamilyen veszélyes műveletet akarunk végezni, akkor a biztosítás bekapcsolása esetén a Vigados minden esetben megerősítést vár. Enterprise-os körökben (még) szokatlan a Vigados remek *Maszk* funkciója: meghatározhatunk egy névmaszkot, és csak az ennek megfelelő fájlok, könyvtárak jelennek meg. Például ha a kiterjesztés részbe PAS-t írunk, akkor csak azok a fájlok jelennek meg, amelyeknek .PAS a kiterjesztésük. Ugyancsak újszerűnek számít a *Beállítás mentés* funkció: ha a Vigados-t egyszer beállítjuk olyanra, ami nekünk megfelel, akkor a beállítást elmenthetjük, és legközelebb már így jelentkezik be (Save setup). *Ez minden felhasználói programokat készítőnek mintaértékű lehet!*

A lényegtelen **Infó** funkcióban copyright szöveget olvashatunk.

Az utolsó **Kilép** funkciónál 15 rendszerkilépési lehetőség (pl. WP, BASIC, ASMON indítása) közül választhatunk. A Vigados *nem vizsgálja a ténylegesen elérhető rendszerbővítőket*, így esetleg olyan rendszerbővítőket (pl. FORTH, LISP) is kínál, amelyek nincsenek a gépben.

Még néhány észrevétel: *Nagyon kényelmetlennek tartjuk a szövegmegadás (pl. maszk, kötetnév) módját.* Megjelenik az ablak, de a szöveg még nem írható be, hanem külön rá kell állnunk mondjuk a *Név* illetve a *Jel* stb. felíratra, ahol kattintanunk kell. Miután beírtuk a szöveget, leütjük az Enter-t, és még az Oké is hátravan. Rengeteg felesleges mozdulat! (A *rename* funkciónál összesen nyolcszor kell kattintanunk. Ugyanehhez a DOS-ból 3 billentyűt (REN) elegendő...) Hiányzik az *Esc* billentyű (abortálás) és az *Enter* (bevitel) figyelése, bár helyenként a *Stop* (abortálás) hatásos. A menürendszer kezdetleges, az almenük lezárásának kezelése rossz. Példa: Belépünk a *Speciál*-ba, majd a *Memóriá*-ba, majd a *Statisztiká*-ba. A *Statisztika ablak* lezárásához a *Memóriá*-ba kell visszalépnünk (a *Statisztika* aljára kellene egy *Oké* felirat, vagy figyelhetné a program az *Esc*, *Enter* gombokat, és ezekkel lezárni). A *Memória ablak* szerencsére nem tűnik el, viszont ha következetesen (!) visszalépünk a *Speciál* menü *Hangszóró*-ra, akkor mindkét ablak bezáródik. Miért? Ilyen és hasonló példa még sok akad.

Nagyon biztosan működik a Vigados hibakezelése: nincs olyan helyzet, kritikus hiba, amelyben ne találná fel magát. Sajnos a hibaablakoknál is jelentkeznek a hiányosságok.

Összefoglalva: Kevés olyan lemezművelet van, amit a Vigados 1:3 nem tud elvégezni. A program lemezkezelő magja nagyon kidolgozott, profi munka, azonban a rendszer összes előnyét eltakarja az elégségesre sikerült felhasználói felület. Ha a program ezen részét a készítő átdolgozza, a Vigados 1.3 méltán számíthat sikerre.

(A Vigados megrendelhető: Vicsotka Gyula, 2143 Kerepestarcsa, Pf.: 41)

-HCs-

LED demó

Az alábbi rövid rutin a lemezegységek sajátságos felhasználására mutat példát. Hogy mit csinál? Nem árulom el! (A programot 5-ös fejléccel kell fordítani.)

K.System'S

```
        org 0100h
port    equ 24
        di
c1      ld b,3
c2      ld c,b
        ld a,1
        out (port),a
        ld a,(szam)
        ld b,a
c4      djnz c4
        ld a,2
        out (port),a
        ld a,(szam)
        xor 255
        ld b,a
c5      djnz c5
        ld b,c
        djnz c2
        ld hl,szam
        inc (hl)
        ld a,(hl)
        cp 255
        jr nz,c1
c6      ld b,3
c7      ld c,b
        ld a,1
        out (port),a
        ld a,(szam)
        ld b,a
c8      djnz c8
        ld a,2
        out (port),a
        ld a,(szam)
        xor 255
        ld b,a
c9      djnz c9
        ld b,c
        djnz c7
        ld hl,szam
        dec (hl)
        ld a,(hl)
        cp 0
        jr nz,c6
        jr c1
szam    defs 1
        end
```

Szevasztok, ENTERPRISE-osok!

Szeretnék bemutatkozni. EPY-nek hívnak, 1991. szeptember 1-jén születtem (nem a bolondok napján!). Apukám (JOVI) és anyukám - az ENTERPRESS olvasóinak egyik tagja, akinek hosszú szőke haja, zöld szeme, csodás alakja van - gyártott, no nem nagyiparban, és nem az ágyban. Beszélni már születésem pillanata óta tudok, a Ti sajnálatotokra! Feladatom az, hogy az ENTERPRESS "Könnyed műfajához" küldött levelekre válaszoljak. Már itt szeretnék elnézést kérni mindenkitől, aki levelet ír, hogy nem méltatom hosszú válaszra, de ez a játékleírások elől rabolná el a helyet. Várom a további leveleket! Címem: ENTERPRESS-EPY, 1399 Budapest, Pf. 701/334.

Gulyás Árpád Véméndről azt írja, hogy a magnója és a gépe is rossz. Örökéletkódokat kaptam tőle. Köszönöm!

Feczkó Kálmán Veszprémből levelében három kérdésre vár választ, mindhárom a *KING OF THE CASTLE*-lel kapcsolatos.

1., *"Hogyan lehet a kereskedőnél eladni és venni tárgyakat?"* Ha a kereskedőnél megnyomjuk a **[T]** billentyűt, megjelenik a nála lévő összes tárgy listája, sorszámokkal ellátva. Ha meg szeretnénk venni valamit, nyomjuk meg a megfelelő szám gombját. A vétel úgy lehetséges, hogy a *CASH* felirat alatt lévő összegből levonodik a tárgy ára. Ha nincs elég pénzünk a gép *"You can't afford to buy that"* felirattal tudatja ezt velünk. (Nálad szerintem az volt a baj, hogy még nem adtál el semmit, így a CASH felirat alatt egy nagy nulla állt. Az *OBJECTS* alatt csak a nálunk lévő tárgyak értéke áll, de az nem készpénz.) Eladni pedig úgy lehet, hogy a [T] után nem választunk ki semmit, hanem a **[SPACE]**-t nyomjuk meg, így a vételhez hasonlóan egy szám leütésével adjuk el a nálunk lévő tárgyat. Ha pl. megnyomtuk az [1]-est, de a lista még mindig látszik, akkor a [SPACE]-szel lépjünk tovább, mert egy újabb szám hatására csak leteszi a tárgyat a jó öreg Bűvös Lovag.

2., *"Hogyan lehet a* Slimey Lower Maze *labirintusból a* Gort the trader's room *nevű szobába jutni?"* A térkép alapján - mondhatnám. A labirintus bal alsó sarkában lévő kijáraton át.

3., *"Hogyan kell az örökéletkódokat bevinni?"* Betöltjük az ASMON (SIMON) nevű felhasználói programocskát. Ezután már csak annyi a dolgunk, hogy a szögletes zárójelben lévő betűket leütjük, és az utánuk álló számokat illetve szavakat beírjuk. A KING-nél a második fájlt kell betölteni! Egyébként a zárójeles betűk a következőket jelentik ASMON nyelven: **[R]**-betöltés, **[M]**-módosítás, **[S]**-kimentés. (Ne aggódj, én sem értek ebből egy kukkot sem! Don't worry, be EPY!) (Kiejtve: dont vöri, bí ipáj. Lásd még lent a szerző saját kiejtési verzióját is! *A szerk.*)

Akik leírásokat kérnek, azoknak csak annyit tudok üzenni, hogy kéréseiteket megpróbálom teljesíteni, és minden eszközzel megpróbálom kiharcolni a terjeszkedést a levelező rovat számára, de erre csak akkor lesz lehetőség, ha a "Könnyed műfaj" rovat is tud majd terjeszkedni vala(ha-ha-ha).

Kreiner Attila Budapestről egy *Sorcery* leírást küldött. Nagyon-nagyon-nagyon szépen köszönöm neki, és elnézés szeretnék kérni, hogy semmi visszajelzést nem kapott, de az átfutási idő miatt csak most tudok válaszolni. A most küldött kazettát pedig sajnos felhasználatlanul kell visszaküldenem, mert azon nem programhangok, hanem csak nyávogás volt. (Lehet, hogy a macskád átkapcsolta a magnót mikrofonra, és énekelt nekem egy dalt. Neki üzenem, hogy sajnos nem beszélek több idegen nyelvet, csak az angolt!) Várom a jó szövegfájlt!

Most pedig búcsúzom! Ne feledjétek a jelmondatot: HAPPY, HAPPY EPY!!! (Gyengébbek, illetve angolul nem tudók kedvéért a kiejtés: Hepi, hepi, epi.)

-EPY-

DIZZY II - THE TREASURE ISLAND

Ahogy az előző számban megígértem, íme, itt a DIZZY sorozat második tagjának, a *Kincses Szigetnek* a leírása. Az 1987-ben megjelent DIZZY I sikerén felbuzdulva a **CODE MASTERS** programozói ismét életre keltették a kis záptojást, aki újabb kalandok részesévé válik egy szigeten, ahonnan haza szeretne jutni. De ehhez szüksége van egy hajóra és egy kis készpénzre is, amivel a sziget határőrét le tudja fizetni. A szigeten minden megtalálható, a detonátortól kezdve a búvárszemüvegen át a bibliáig. (Kivéve a legfontosabbat, egy újságosstandot, ahol DIZZY megvásárolhatná a legújabb ENTERPRESS-t. Talán ezért szeretne visszatérni Tojásland-be! *-EPY*) [*A szerk. megj.*: Ott is hiába keresné, a Kincsesszigeti Posta inkább raktározza a lapot, nem terjeszti...]

Az irányítás történhet *External-*, *Internal Joy*-jal, illetve billentyűzettel. Ha az External-lal akarunk játszani, akkor nyomjuk meg a botkormányon a tűzgombot. Ha az Internal esik jobban kézre, akkor húzzuk azt lefelé, és máris indul a játék. Aki pedig nem sajnálja a billentyűzetét, az az [ENTER]-t nyomja le. Ha elkezdtük a játékot, akkor ismerkedjünk egy picit az irányítással. Ugrani a tűzgomb lenyomásával (billentyűzeten a SPACE), jobbra/balra mászkálni pedig a joy megfelelő irányba való húzásával (billentyűzeten X/Z, illetve német gépeken X/Y billentyűvel) tudunk. Tárgyat felvenni a joy lefelé húzásával (az ENTER billnetyűvel) tudunk .

Erről egy kicsit bővebben: Dizzykénél egyszerre összesen három tárgy lehet. Ha nincs nála semmi, akkor mindhárom helyen *NOTHING* áll. Ezt a képernyő tetején láthatjuk. Ha meg szeretnénk cserélni a tárgyak sorrendjét, akkor le kell őket tenni, majd a megfelelő sorrendben felvenni. Ha felveszünk egy tárgyat, akkor a legfelsőt Dizzy lerakja. (Ezért kell a búvármaszkot majd mindig az utolsó helyen tartani, ha a vízbe megyünk.) Ennyit erről, most pedig kezdődjék a nagy kaland a kincses szigeten!

A játék legelején egy tó mellett állunk. Mivel tudjuk, hogy a határőrséget le kell fizetnünk, így a jobb oldalon látható pénzérmét próbáljuk meg felvenni. Sajnos, DIZZY megfullad. A következtetés: ne menjünk a vízbe, ha nem tudunk teniszezni. (Ha-ha-ha! Ez most hülyeség volt!) Marad az egyetlen lehetséges út, balra. De először menjünk a bal oldali bokorhoz, és próbáljuk meg felvenni. Jé, a kezünkben maradt! Ez egy védett növény (*PROTECTED SPECIES*). De ez bennünket ne érdekeljen (Érdekelje a zöldeket! *-EPY*), hanem a bokor mögött rejtőző pénzdarabot (*01*) vegyük fel. A zöldséget tegyük is le. Menjünk tovább balra, ahol egy sziklafal az utunkat állja. A két pálmafa között viszont egy öreg ládát találunk (*OLD SOLID CHEST*), amit ha felveszünk és a szikla mellé tesszük, akkor tovább tudunk jutni. Ugorjunk hát tovább, s a következő pályán vegyük fel a második coint is (*02*). Menjünk tovább, és a híd előtt lévő növényt is vegyük fel. Itt is van egy pénz (*03*). A zöldségtől szabaduljunk meg ismét, és a hídon lévő coint is vegyük fel (*04*).

Továbbhaladva egy sírkőbe botlunk, és egy mellette fekvő papírtekercsbe. Olvassuk el! (Álljunk elé, és húzzuk lefelé a joy-t.) *"Egy legenda szól Hookjaw-ról, a kalózról, aki itt fekszik, s őrködik a kincse felett."* Találhatunk még itt egy fogkrémes tubust (*TOOTHPASTE*) is, de erre nincs szükségünk. (Hacsak nem lenne fogkrém, mert akkor Dizzy-t bekenve nyugodtan mászkálhatnánk a vízben, nem vesztené el kalciumtartalmát! *-EPY*) Menjünk hát tovább, s egy újabb tekercsre bukkanunk, melyen ez áll: *"Snoggles faház-komplexum - elhagyatott, amióta a turisták fenyegető hada feldúlta nyugalmát."* Továbbhaladva egy rakás gomba (*CLUMP OF MUSHROOMS*) mögött találunk egy érmét (*05*). Majd a következő pályán egy újabbat (*06*). De elfogyott az út!

Forduljunk hát vissza, és a *Snoggles-komplexum*nál ugorjunk fel a házhoz. A jobb oldali ablakot leszedve (*MISTY GLASS OF WINDOW*) még egy coint találunk (*07*). Felvétele, és természetesen az ablak letétele után haladjunk tovább balra. Ahol az imént nem tudtunk továbbmenni, ott most egy ketrec vár ránk. Próbáljunk meg átmenni alatta - majd kezdjük újra a játékot, hiszen csak egy életünk van. Átmenni nem tudunk alatta. Kúszni itt nem tudunk (Azt csak RICK DANGEROUS tud! *-EPY*), hát próbáljunk meg ugrani. Ha az utolsó lépcsőfokról ugrunk, akkor nem halunk meg! Éppen a szemben lévő platóra érkeztünk, ahol egy újabb tárgy vár ránk. (Egy Sinclair Elhasználói Magazin (*SINCLAIR ABUSER MAG*), amely a Spectrum tönkretételétől a joystick széttöréséig minden jó dolgot tartalmaz! *-EPY*) Erre nincs szükségünk, csak megtévesztésből van itt. Menjünk hát tovább!

Mikor egy felfelé vezető lépcsőhöz érkeztünk, álljunk az első lépcsőfokra, és húzzuk le a joy-t. Nahát, felvettük a fa kérgét! Tegyük is le, de közben vegyük fel az így előkerült pénzdarabot (*08*). Továbbhaladva jobbra fent egy érmét láthatunk. Ugorjunk fel érte. (Ha rosszul ugrottunk, akkor újrakezdés...! *-EPY*) De vigyázzunk, mert a plató felett egy újabb ketrec áll. (Vajon ezeket ki tette ide, hisz még egyetlen embert sem láttunk! *-EPY*) A ketrec kikerülése érdekében még az előző pálya legszéléről

kell ugranunk. Majd ugorjunk ismét jobbra, és fel is vettük a következő coint (*09*). Ugorjunk vissza a platóra jobb oldalára, onnan ugorjunk jobbra ismét. Itt menjünk el jobbra, ahol egy kavicsot láthatunk. A kavics nem más, mint egy kapcsoló, amelyhez ha hozzáérünk, az átgurul a túloldalra, és az alattunk lévő platón egy újabb lépcsőfok jelenik meg. Ezután menjünk le oda, ahol az előző coint felvettük, és essünk le az alattunk lévő platóra. Innen menjünk jobbra, és a következő pálya bal szélén ugorjunk felfelé, majd jobbra haladva meglátjuk azt a platót, amelyet a kapcsolóval helyeztünk oda. Ugorjunk erre fel, majd a másik pályán a második fánál ismét fel kell ugrani. Innen a balra lévő platóra, majd balra menve a házhoz ugorjunk át. A ház túlsó végén vegyük fel az infravörös detonátort (*INFRARED DETONATOR*). Majd ismét tegyük meg az előző utat visszafelé egészen a kavicsig.

A ház alatti platónál, a képernyő jobb szélén egy coin van elrejtve a farács (*WOODEN SAFETY RAIL*) mögött (*10*). Innen a balra lévő platóra ugorjunk, ahonnan feljuthatunk egy házig. (Ez már nem komplexum, a turistáknak nem volt erejük idáig felszenvedni magukat. *-EPY*) Ugorjunk tovább jobbra-fel, s a következő pályán ismét egy házat találunk. (Tiszta lakótelep az őserdő közepén! *-EPY*) A ház túlsó végén egy búvárszemüveget találunk (*RUBBER SNORKEL*). Vegyük fel, menjünk a plató bal szélére, és ugorjunk egyet a semmibe. Ha jól csináltuk, két kőomlás közé érkezünk, ahol egy zöldség mögött újabb érmét találunk (*11*). Jöjjünk el jobbra, essünk le, majd menjünk balra. Egy kőomlás állja az utunkat. De van itt egy papírtekercs, amelyet elolvasva megtudhatjuk, hogy robbantási területen járunk. Meneküljünk jobbra, de először tegyük itt le a detonátort. Ugorjunk fel a platóra, menjünk jobbra, és ugorjunk fel a jobbra-fent lévő platóra. Igyekezzünk a plató másik végéig, és ott a balról a harmadik fa mellé. Próbáljunk meg felvenni egy tárgyat. Ismét a farács mögött találhattunk egy pénzdarabot (*12*). A farácsot természetesen tegyük vissza. Majd a plató széléről ugorjunk fel jobbra, majd erről a platónak a közepéről felfelé. Innen ugorjunk át balra, majd vegyük fel a jobbra lévő (13), illetve a balra lévő érmét is (*14*). Menjünk vissza oda, ahol a farács mögött volt a pénz, és menjünk a bal oldali pályára. De ott ne menjünk tovább, hanem a képernyő jobb szélén ugorjunk felfelé. Egy újabb platóra érkezünk.

Menjünk balra, vegyük fel az érmét (*15*), majd vissza, és a legvékonyabb fatörzs mögül is vegyük fel a coint (*16*). (A fatörzset a szokásos módon tegyük vissza.) Most ugorjunk fel arra a platóra (jobbra), amelynek a legalsó lépcsőfokát a kallantyú elmozdításával jelenttettünk meg. Menjünk jobbra, és a következő helyszínen balról a második fánál ugorjunk egyet felfelé. Itt menjünk teljesen jobbra (közben megfigyelhetjük, hogy ide is állított valaki egy csapdát), és a plató végén vegyük fel a ház melletti éles üvegkardot (*SHARP GLASS SWORD*). Alattunk egy másik tárgy is van, ezért menjünk le érte. (A csapda kikerülése érdekében az utolsó előtti fatörzsnél ugorjunk!) Egy kis fényképezőgépet találtunk (*SMALL VIDEO CAMERA*). Zsebeink megteltek, menjünk hát a tóparti fövenyhez (startpálya).

Tegyük le a parton a fényképezőgépet és az üvegkardot. Helyezzük el a búvármaszkot úgy, hogy a harmadik helyen legyen (nehogy letegyük, ha a vízben felveszünk valamit, különben megzápulunk). Menjünk be a tengerbe, és vegyük fel azt az pénzdarabot, amelyik a vesztünket okozta a játék legelején (*17*). Menjünk tovább! Itt egy halat és egy polipot látunk. Nehogy hozzáérjünk valamelyikhez, mert a hal tojással táplálkozik, a polip pedig híres arról, hogy szereti a búvármaszkokat ellopkodni a szerencsétlen tojásokról. (Egyébként mindegyik tengeri élőlény DIZZY halálát okozza!) Menjünk tovább! A *Titanic* roncsát láthatjuk, amit elleptek a rákok és a polipok. A roncson lévő coint vegyük fel (*18*), majd jobbra a rákot átugorva a sósvízi ásót (*SALT WATER SPADE*). Majd visszamászva menjünk tovább jobbra. Itt egy koponyát találunk. Vegyük fel, és a mögötte levő érmét se hagyjuk a vízben rozsdásodni (*19*). Menjünk vissza a partra! A koponyát hagyjuk itt, és a tárgyak sorrendjét cseréljük meg. Legfelül az ásó legyen, utána az üvegkard, legalul a búvármaszk.

Ezután menjünk vissza oda, ahol a koponya volt. Itt egy követ láthatunk a jobb oldalon, amelyet valami furcsán mozgat. Mintha valami ki szeretne bújni a föld alól. (Biztosan egy viziszörny! Hú, de horrorisztikus egy játék! *-EPY*) Menjünk oda, és tegyük le az ásót. Egy gázkitörésre leltünk, s ahogy kiástuk, a gázbuborékok formájában szabadul ki ezeréves fogságából. Álljunk rá az egyik buborékra, amely egy medúza felé visz bennünket. Ugorjunk hát le róla, mielőtt a medúza karjai közé kerülnénk. (Vigyázzunk az itt úszó kishalak csoportjára is!) Majd jobbra menve, és útközben a coint felvéve (*20*) menjünk ki a vízből.

A szárazföldön lépdeljünk tovább. Útközben a következő pénzdarabot felvéve (*21*) menjünk el az első kunyhóig. Itt egy totemoszlop mellett ismét egy gombarakást találhatunk. (Hohó! Fogadjunk, hogy egy coin lesz mögötte?! *-EPY*) Vegyük fel. Ha nem is sikerült, de a mellette lévő szikla (*BIG RED HEAVY ROCK*) mögött egy pénzdarabot lelünk (*22*). A "kavicsot" el is dobhatjuk, majd menjünk tovább. Egy újabb kunyhót látunk, ahol egy bajszos hapsi áll. (Az első értelmesnek tűnő élőlény! *-EPY*) Ez az úriember új vállakozást indított, műszaki cikkek adás-vételével foglalkozik.

Tegyünk le még a ház mellett mindent, menjünk oda hozzá, és húzzuk le a joy-t! Az eladó bácsi a következőkkel kezd el untatni bennünket: *"Gondolom, el szeretnéd hagyni a szigetet. Adok neked egy csónakot némi értékes holmiért cserébe."* A készletünk: Egy búvármaszk - erre még szükség lehet. Tényleg, az üvegkard. Á, ez sem jó! Nem egy harcos lélek ez. Megvan! A fényképezőgép! Vegyük fel, majd tegyük le elé. Az úriember megköszöni, majd ad nekünk egy szárított (porított?) csónakot (*DEHYDRATED BOAT*), mellé pedig egy használati utasítást: El kell vinni a stégre. Vegyük fel az itt lévő coint (*23*), majd menjünk tovább jobbra a stégig. (Útközben két ház között a zöldség mögött is van egy coin (*24*).) Majd álljunk a stég szélére, és tegyük le a csónakot. Itt van egy aranykulcsocska (*LARGE GOLDEN KEY*) is, ezt vegyük fel.

Menjünk vissza az áruházba, és kérdezzük meg ismét a palit. Most azt mondja, hogy valószínűleg szükségünk van egy motorra is. Van neki valahol, de az kerülni fog nekünk valamibe. Vegyük vissza a búvármaszkot, valamint az üvegkardot, majd menjünk tovább balra. Egy idő után rátalálunk egy sírkőre, s alatta egy nagy lyukat látunk. Tegyük le a sírkőre az üvegkardot. A föld megnyílik a lábunk alatt, de ne essünk még le! Tegyük a búvármaszkot a tárgyaink között az utolsó helyre. (Gyengébbek kedvéért a tárgyak helyén egymás alatt a következő kell hogy álljon: *NOTHING/NOTHING/A RUBBER SNORKEL*. *-EPY*) Most már bátran leeshetünk.

Mindkét oldalon egy egy tekercs áll. A bal oldalit olvassuk el. Ez áll rajta: *"Vigyázat! Mágikus vízfal!"* Ez, illetve a pályán lévő kis tócsa arra utal, hogy balra víz van. Nem baj, nálunk van a búvármaszk, menjünk nyugodtan balra. Itt néhány dinamitrudat találunk (*STICKS OF DYNAMITE*). Emlékezzünk csak vissza! Még a másik szigeten a játék elején be próbáltunk menni egy barlangba, de az be volt omolva. Ott az állt: *"Robbantási terület!"*, a dinamitot ott kell majd felhasználni, ezért vegyük magunkhoz. Majd menjünk vissza, és olvassuk el a másik írást is. Ez azt mondja, hogy a sírkőre tett üvegkarddal nyitható ajtó *Rogeré*, a híres kalózé! Menjünk jobbra. Ho-hó! Itt egy koponya! (Na, most már tényleg fogadjunk, hogy egy érme lesz mögötte! *-EPY*) Vegyük fel, majd a mögötte levő érmét is (*25*). (Nyertem! Nyertem! *-EPY*) Ismét jobbra egy újabb pénzdarabot találunk (*26*), majd essünk vissza a két hordó közé. Itt a földbe süllyesztve is van egy hordó. Ez biztos egy rejtekajtó. De vajon mivel nyitható? Tegyük le a stégen felvett aranykulcsot, s a hordó leereszkedik. Itt is van egy coin (*27*), valamint egy papirusz is. Elolvasva megtudhatjuk, hogy megtaláltuk a kalózok titkos konyháját, amely alkalmas hely az éjféli poharazgatásra. A túloldalon egy mikrohullámú sütőt (*MICROWAVE OVEN*) találunk, ezt vegyük fel. (Ezek valami XX. századi kalózok lehettek! *-EPY*)

Másszunk ki a barlangból, tegyük le a mikrót, majd a sütőt. Illetve a mikrohullámú sütőt. (Na azért! *-EPY*) Most vegyük fel a jobbra lévő pályán található Szent Bibliát (*HOLY BIBLE*), valamint a barlang felett lévő favágó fejszét (*WOODCUTTER'S AXE*). Menjünk át a tavon, és menjünk a hídig! A híd közepén tegyük le a fejszét, amely egy lyukat vág a hídon. Ne essünk le, rendezzük át a tárgyainkat úgy, hogy a búvármaszk az utolsó helyen álljon, a biblia pedig a másodikon. Most essünk le a vízbe. Itt is egy barlang van. A jobbra lévő üregben találunk egy pénzdarabot (*28*), valamint az elátkozott kincset (*CURSED TREASURE*). (Az átoktól véd meg bennünket a biblia.) Ha ezeket felvettük, akkor a bal oldali pályán lévő érmét is vegyük fel (*29*), és a platókon felugrálva kijuthatunk a felszínre. (Hookjaw sírjánál bukkantunk felszínre. Mi van fókalóz, mégsem sikerült a kincset megőrizni? *-EPY*) Most fogjuk a kincset, és vigyük el a bajszos úrgéhez, aki cserébe most egy csónakmotort ad

(*OUTBOARD MOTOR*); igaz, azt mondja, nem tud mit kezdeni a kincsel. (Adja nekem! Majd én felélem! *-EPY*) A bibliát dobjuk el, a motort pedig vigyük a stégre.

Álljunk ismét a szélére (de ne a legszélére!!!), és tegyük le a motort. Ha jól csináltuk, a csónak színe megváltozik. (Ha túlságosan a stég szélén vagyunk, akkor csak simán letesszük, majd pedig felvesszük a motort. Ez igaz lesz a többi alkatrésznél is!) Olvassuk el ezek után az itt látható tekercset is. Az áll rajta, hogy addig nem tudunk elcsónakázni, míg a csónak nem mozog. De mi kell még a csónakhoz? Evezőlapát? De hisz motorunk is van már. Megvan! Benzin! Menjünk hát vissza a tavon keresztül a robbantási területhez, de a barlang felett hagyott dinamitot is vigyük magunkkal. Vegyük fel itt a detonátort, majd menjünk balra, ameddig csak tudunk. Itt tegyük le a dinamitot, és a kicsit hátrébb lévő kő mögé álljunk. Ha itt letesszük a detonátort, akkor bummm!, a dinamit robban, és szabad az út. Menjünk hát balra, és vegyük fel az itt található pénzeszsacskót (*BAG OF GOLD COINS*). Ezt vigyük vissza a túlpartra a bajszos fickóhoz, aki odaadja a benzint (*A GALLON OF PETROL*). Ezt is vigyük a stégre, és tegyük le. A csónak színe ismét változik. Majd menjünk a barlang felett hagyott mikrosütőért, amit szintén adjunk oda neki. Cserébe odaadja az utolsó alkatrészt is, a slusszkulcsot (*IGNITON KEY*), és sajnálkozását fejezi ki, amiért elmegyünk, hiszen mi voltunk a legjobb kuncsaftja - egyébként az egyetlen.

Természetesen a kulcsot is a stégre kell vinnünk. Ha letettük, akkor a csónak elkezd ide-oda úszkálni a vizen. Ugráljunk át rajta Tojásland szigetére. Itt találunk egy tekercset, melyben közlik velünk, hogy sikeresen teljesítettük a feladatot. Továbbmenve a határőrrel találkozunk. Ha nincs nálunk a 30 coin, akkor közli, hogy csak akkor mehetünk át, ha kifizetünk neki 30 aranyat. Ekkor menjünk vissza, és vegyük fel a harmincadik érmét is. Ha viszont mindet felvettük, akkor a következő szózatot intézi hozzánk: "*Mmmm! 30 arany érme! Nem rossz! Azt hiszem, most szeretnéd tovább folytatni a kalandozást. Sajnálom, nem lehet. Most láttad Fantasy World sarkát, már láttál és megcsináltál mindent. De még fogunk találkozni!*" Miután jól lefárasztott a dumájával, nyomjuk meg a hátsó piros gombot, hiszen itt a vége, fuss el véle, átjutottunk Tojáslandbe. (Akkor irány az újságos, meg kell venni az ENTERPRESS-t!!! *-EPY*)

DIZZY II	
Grafika:	6
Zene/FX:	5
Játszhatóság:	7
Összhatás:	6

TEENAGE MUTANT NINJA TURTLES

Dátum: 1990. Helyszín: Észak-Amerikai Egyesült Államok, közismertebb nevén USA. A BAT MAN láz elmúlt, követte az új: a TMNT láz. Az előzmény - csakúgy mint a BAT MAN esetében - egy képregény, majd egy rajzfilm, és végül egy óriási sikerű mozifilm volt. (Remélem, már mindenki úgy veszi a kezébe ezt a lefrást, hogy a filmet betéve tudja. Neeem? No sebaj, a játék úgysem kapcsolódik a filmhez szinte semmiben.)

Egyszer, réges-régen, talán még az elmúlt évszázadban élt Japánban egy teknőc (*TEKI-BREKI*), aki kifigyelte az akváriumból házigazdája, *Ninjutsu* mozdulatait edzés közben. Már egészen jól tudta a mozdulatokat, amikor is egy orvtámadás következtében mestere az életét vesztette, ő a támadót elpusztította. (Kemotoxszal? *-EPY*) (Na, ez már ide is bepofátlankodik? Nem elég neki a levelezői rovat?) Ekkor döbbent csak rá, milyen hatalmas tudás birtokába jutott. Egy vízelvezető csatornába költözött (természetesen a Tehertaxi igénybevételével - *EPY*), ahol négy apró teknőcöt talált. Felnevelte őket, és megtanította nekik a ninják alapfogásait. Mikor látta, hogy mindegyik fiú tehetséges, akkor külön-külön megtanította nekik egy-egy harci eszköz használatát. *Donatellónak* a *Bo* (bot), *Raphaelnek* a *Katana Blade* (kard), *Michelangelónak* a *Sai* (kés) és *Leonardónak* a *Nunchaku* kezelését. (Na, négy karatélyozó teknőc, akik főfoglalkozásban festők? Vagy hogy van ez? *-EPY*) Eddig még rendben is volna, de egyszer csak megismerkedtek egy csodaszép újságfrónővel, *Aprillel* (Nem inkább Decemberrel? *-EPY*), akit egy szép napon elraboltak Shredder emberei. A kis hősök elhatározták, hogy megmentik a barátnőjüket, és szembeszállnak Shredderrel.

A játékot a **MirrorSoft** készítette 1990-ben, a Konami cég licensze alapján. A négy hős New York egyik kikötőnegyedéből indul útnak April megmentésére. A célunk legyőzni Shredder embereit, majd magát a sötét ninját. Ehhez utcákon, csatornanyílásokon és nyitott ajtókon keresztül kell közlekednünk. A csatornák végén Shredder fő embereivel kell megküzdenünk, akik nem is olyan gyenge ellenfelek. Ha az energiánk elfogy, akkor nem halunk meg, hanem csak (?) fogságba kerülünk. Egyszerre csak egy teknőccel kell verekednünk, így ha azt elfogják, még akkor is kiszabadíthatjuk, ha a megfelelő főgengsztert megöljük. Az energiát az utunkon található pizzadarabokkal növelhetjük. (Hmm, az finom! *-EPY*) Az irányítás a joystickkal történik. A tűzgomb megnyomásával pedig a fegyverünket használhatjuk.

Ha a kikötőt sikerült megtisztítani (ellenkező esetben hívj segítségül néhány, az utcákon gyakran látható UV-narancssárga kabátban tevékenykedő teknőcöt! *-EPY*), akkor a víztározóhoz jutunk, amelyet Shredder éppen felrobbantatni készült. A labirintusban úszva kell a bombákat hatástalanítani.

-JOVI-

TEENAGE MUTANT NINJA TURTLES	
Grafika:	3
Zene/FX:	4
Játszhatóság:	5
Összhatás:	4

Örökéletkódok

RICK DANGEROUS
-[R] 10FD [ENTER] BFFF [ENTER] RICK.DTF [ENTER]
Last address: A183
-[M] 117D [ENTER] AF 32 F4 E3 [ESC]
-[S] 10FD [ENTER] A183 [ENTER] RICK.DTF [ENTER]

HAMMERFIST
-[R] 1B00 [ENTER] BFFF [ENTER] HAMEFIST.PRG [ENTER]
Last address: BFFF
A 4. fájlt kell betölteni!
-[M] 7C05 [ENTER] C0 77 00 00 [ESC]
-[S] 1B00 [ENTER] BFFF [ENTER] HAMEFIST.PRG

UNTOUCHABLE
-[R] 4700 [ENTER] BFFF [ENTER] PART-1 [ENTER]
Last address: BDE3
-[M] 6723 [ENTER] 00 [ESC]
-[M] 6788 [ENTER] 00 [ESC]
-[S] 4700 [ENTER] BDE3 [ENTER] PART-1 [ENTER]
-[R] 4700 [ENTER] BFFF [ENTER] PART-2 [ENTER]
Last address: BDE3
-[M] 7033 [ENTER] 00 [ESC]
-[M] 707F [ENTER] 00 [ESC]
-[S] 4700 [ENTER] BDE3 [ENTER] PART-2 [ENTER]
-[R] 4700 [ENTER] BFFF [ENTER] PART-3 [ENTER]
Last address: BDE3
-[M] 7170 [ENTER] 00 [ESC]
-[S] 4700 [ENTER] BDE3 [ENTER] PART-3 [ENTER]
-[R] 4700 [ENTER] BFFF [ENTER] PART-4 [ENTER]
-[M] 72B3 [ENTER] 00 [ESC]
-[M] 7702 [ENTER] 00 [ESC]
-[S] 4700 [ENTER] BDE3 [ENTER] PART-4 [ENTER]

ENTERPRESS TOP TEN

1. Rick Dangeorus I * FIREBIRD
2. Lotus Turbo Esprit Challenge * GREMLINS
3. Myth * SYSTEM 3
4. Untouchables * OCEAN
5. Little Puff In Dragonland * CODE MASTERS
6. Mah Jongg * DEVILSOFT
7. Stuntcar Racer * PETE COOKE
8. Where Time Stood Still * OCEAN
9. Vendetta * SYSTEM 3
10. Swap * CIRCLE

Postafiók 334

Mindig szívdobogtató érzés a szebbik nem képviselőitől levelet kapni. Sajnos, az ENTERPRESS szerkesztőségét nem kényeztetik el a hölgyolvasók; de hát úgy kell nekünk, miért nem divatlapot szerkesztünk?! Így igen nagy örömmel olvastuk **Szalontai Andrea** budapesti kedves olvasónk sok-sok kérdést tartalmazó és egyéb témaköröket érintő levelét.

Olvasónk nagy gondja, hogyan tudna relatív fájl-elérést, indexelt fájlokat létrehozni. Való igaz, hogy szinte minden komolyabb alkalmazásban, ha annak fő feladata az adatmanipulálás, elsőrendű kérdés az adatok elérhetősége. Sajnos, az IS-BASIC csak az minimális soros írást és olvasást teszi elérhetővé. Még az egyébként igen hasznos hozzáírást (APPEND) sem engedi meg. (Ennek az a lényege, hogy egy fájlt megnyitunk olvasásra, elmegyünk a végéig, majd átváltunk írásra, és mintegy folytatva a korábban elkezdett fájlt, hozzáírjuk az újabb adatokat. Ennek hiányában a fájlt be kell olvasni és egy másik fájlba kiírni, hozzáírva az új adatokat. De az új fájlnak más nevet kellett adnunk, és az IS-BASIC az átnevezést sem adja könnyen. És akkor hol vagyunk még a közvetlen eléréstől!)

Ahogy mikroszkóppal nem verünk be szöget - vagy jobb példaként, kalapáccsal nem vizsgálunk zöldmoszatokat -, a számítástechnikában is minden alkalmazásra meg kell találni a megfelelő eszközt. Mivel a gép adva van (itt, e hasábokon nem csak nem akarunk más gépet ajánlgatni, hanem erre nincs is szükség), amin változtatni lehet, az az felhasznált programnyelv vagy egyéb szoftverjellegű segédeszköz. Az ENTERPRISE-on is használható Turbo-Pascal például, egyéb előnyei mellett, a közvetlen fájl-elérést is megvalósítja. Ha programunk csak mellékesen használja a közvetlen elérést, nagyobb része egyéb dolgokat csinál, a Turbo-Pascal lehet az ideális eszköz.

Ha viszont a program lényege az adatmanipuláció, nem szabad hagyományos programnyelvet használni. Csak egy elrettentő példa: a legkisebb utólagos változtatás az adatszerkezetben a teljes program átírását követelheti minden alkalommal! Ilyenkor adatbáziskezelőt kell használni, hiszen ezt arra találták ki, hogy levegye a programozó válláról az adatok kezelésének a gondját, hogy kizárólag az alkalmazásra koncentrálhassunk. Nyilván nem minden olvasónk tudja, mi is az az adatbáziskezelő; de még kevesebben tudják, hogy ez ugyanúgy programozható, mint mondjuk egy BASIC vagy egyéb rendszer, csak más a filozófiája és más jellegű az utasításszerkezete. Az ENTERPRISE-on is fut a dBASE 2.0 valamelyik változata. Adatbázis létrehozásához ezt kell használni.

Andrea másik problémája, hogy hogyan fér el a nem is egészen 64 kilbájtos videomemóriában a gyári programok által használt sok-sok különböző kép. Erről, kedves Andrea, nem is egy, hanem több könyvet lehetne írni. Csak vázolni tudjuk a programok készítői által használt trükköket.

Először is, nem kell az összes képet a videomemóriában tárolni, hiszen azok a hagyományos memóriából is bármikor áttölthetők oda. Igaz, hogy ez csak a program számára szükséges memória rovására mehet. A másik megoldás, és itt van az igazi memóriamegtakarítási lehetőség, az egyszerűen az, hogy nem a teljes képet tárolják. Az egyik variáció, amikor egyszerűbb képelemeket tárolunk, és ezekből építjük fel a képet. Ilyenkor csak a képelemek tárolására kell a hely, a többi memória térképszerűen tartalmazhatja ezeknek az elemeknek a sorrendjét az egyes képeken.

A másik megoldás szerint a képet mindig csak a kellő pillanatban rajzoljuk ki (gépi kódban ez igen gyors lehet), ekkor is nagyságrendekkel csökken a szükséges memória. A két megoldás esetleg kombinálható is.

Ha ez mind kevés, még mindig lehetséges az adattömörítés használata. Ha arra gondolunk, hogy mondjuk egy világos háttéren kell egy sötét ábrát megjelenítenünk, egyáltalán nem szükséges az összes világos és sötét pontot a memóriában (diszken, kazettán) tárolnunk. Elegendő azt megjegyeznünk, hogy hány világos pont után következik majd sötét, és hány sötét után jön újra világos. Ilyen módon megint csak igen gazdaságosan tárolhatók a képek. Természetesen az ilyen ábra kirajzolása időigényes, de gépi kódban ez sem olyan veszélyes. (Csak megjegyezzük, hogy elvileg hasonló módon történik a képek tömörített tárolása a képi adatbázisokban, és hasonló módszerrel rövidítik le a levelek, rajzok, bizonylatok átviteli idejét a telefax készülékek is.)

Ennyit dióhéjban az adatkezelésről és a képtárolásról.

Kedves Andrea! Levelét köszönjük, színkiválasztó programját és játékleírását pedig szívesen megnézzük, küldje el ezeket!

A többi levélírótól elnézést és türelmet kér az elfogult

felelős szerkesztő

Az Életjáték (2)

Folytatás a 7. oldalról

```
TEDD-AMIT-KELL:
  IF MUTATÓ = EGYIK-ÁLLAPOT THEN
    TÁBLA-2 [SOR, OSZLOP] :=
    FÜGGVÉNY(SZOMSZÉDOK,TÁBLA-1[SOR,OSZLOP]);
  ELSE
    TÁBLA-1 [SOR, OSZLOP] :=
    FÜGGVÉNY(SZOMSZÉDOK,TÁBLA-2[SOR,OSZLOP]);
  END-IF;
END (TEDD-AMIT-KELL)
```

Ez feltételezi a globális MUTATÓ változó létezését, amellyel sok bajunk lesz: először is deklarálni kell; másodszor kezdőértéket kell kapnia; harmadszor, minden LÉPÉS után másik (nem más-!) állapotba kell billentenünk. Ezt adminisztráljuk:

```
VAR
  MUTATÓ: (EGYIK-ÁLLAPOT, MÁSIK-ÁLLAPOT);
LÉPÉS:
  SOROKON-VÉGIG DO
    OSZLOPOKON-VÉGIG DO
      TEDD-AMIT-KELL
    END-DO
  END-DO
  INVERTÁL (MUTATÓ);
END (LÉPÉS)
```

és

```
ELŐKÉSZÍTÉS:
  ALAPHELYZET (TÁBLA-1, TÁBLA-2);
  KEZDŐ-KONFIGURÁCIÓ (TÁBLA-1);
  MUTATÓ := EGYIK-ÁLLAPOT;
  EGYÉB;
END (ELŐKÉSZÍTÉS)
```

Most már csak SZOMSZÉDOK és a FÜGGVÉNY függvény szorul némi pontosításra. Az Életjátékban az elem négy él- és négy sarokszomszédja számít. Egy pillanatra elcsábulnánk, hogy írjunk egy kettős ciklust, amelyik az elem előtti sortól és oszloptól az utána következő sorig és oszlopig megy, nem felejtve el kihagyni a számításból magát az elemet; de lerázzuk a sematizmus béklyóját, megérezve, hogy a megoldás lassú lenne. (IS-BASIC-ben a kettős ciklus nagyon lassú. Vegyük észre, hogy a LÉPÉS saját két ciklusával együtt négyszeresen skatulyázott ciklusunk volna!) Inkább leírjuk nyolcszor egymás után (a "kétszer mond" effektust figyelembe véve, 16-szor) az indexkifejezést (egy jó szövegszerkesztővel ez nem probléma):

```
SZOMSZÉDOK (SOR, OSZLOP):
  IF MUTATÓ = EGYIK-ÁLLAPOT THEN
    RETURN WITH    TÁBLA-1 [SOR - 1, OSZLOP - 1]
      + TÁBLA-1 [SOR - 1, OSZLOP]
      + TÁBLA-1 [SOR - 1, OSZLOP + 1]
      + TÁBLA-1 [SOR,     OSZLOP - 1]
        { ITT KIMARAD AZ ELEM MAGA }
      + TÁBLA-1 [SOR,     OSZLOP + 1]
      + TÁBLA-1 [SOR + 1, OSZLOP - 1]
      + TÁBLA-1 [SOR + 1, OSZLOP]
      + TÁBLA-1 [SOR + 1, OSZLOP + 1]
  ELSE
    RETURN WITH {UGYANAZ, MINT AZ ELŐBB,
    CSAK TÁBLA-1 HELYETT TÁBLA-2 MINDENÜTT}
  END-IF
END (SZOMSZÉDOK)

FÜGGVÉNY (SZOMSZ, SAJÁTÉRTÉK):
  IF SAJÁTÉRTÉK = NEM-ÉL THEN
    IF SZOMSZ = 3 THEN
      RETURN WITH ÉL
    ELSE
      RETURN WITH NEM-ÉL
    END-IF
  ELSE { AZAZ HA ÉL A SEJT }
    IF SZOMSZ = 2 OR SZOMSZ = 3 THEN
      RETURN WITH ÉL
    ELSE
      RETURN WITH NEM-ÉL
    END-IF
  END-IF
END (FÜGGVÉNY)
```

(folytatás a következő számban)

PROGRAMOZÁSI VERSENY!!!

Szerkesztőségünk programozási versenyt hirdet az alábbi két kategóriában:

1.,DEMO
Olyan programokat várunk, amelyek
- minél jobban kihasználják a Nick és a Dave lehetőségeit,
- az Enterpress-t propagálják: szerepel az *újság neve*, a *postacímünk* (1399 Budapest, Pf. 701/334), a *kiadó neve* (Mátrix KFT.), *címe* (8000 Székesfehérvár, Honvéd utca 8.) és *telefonszáma* ((06-22) 16-520/280),
- szellemes (magyar illetve angol nyelvű) mondatokat úsztatnak.

Demo programot egyénileg és csoportosan is lehet készíteni. A programnyelvet mindenki maga választhatja meg, és olyan segítséget használ, amilyet csak akar. A lényeg tehát a minél látványosabb program!

2., BASIC JÁTÉKPROGRAM
Olyan Basic nyelvű játékprogramokat várunk, amelyek
- a "PROGRAM 0"-ba befférnek,,
- minél gyorsabbak,
- alapötletük, történetük (lehetőleg) újszerű.

A programot egyének és csoportok is készíthetik. Nem nevezési feltétel, de nem baj, ha a program Zzzip-pel fordítható. A játékprogram fajtája bármilyen lehet: kalandjáték, logikai játék stb. Azok a programok számíthatnak sikerre, amelyek grafikát és zenét is használnak. A program kritikus részein gépi kódú rutinokat (CODE sorok, ROM hívás) is szabad használni, ezt azonban nem szabad túlzásba vinni! A legjobb programot közölni fogjuk.

Beküldési határidő: 1992. március 1.

A Mátrix KFT. szerkesztőségünkkel együtt kategóriánként az alábbi díjakkal jutalmazza az első három helyezettet:

1. díj: 3000 Ft
2. díj: 2000 Ft
3. díj: 1000 Ft

A programokat postacímünkre küldjék:
ENTERPRESS, 1399 Budapest, Pf. 701/334.

mikrovilág

Az ENTERPRESS előző számai korlátozott példányszámban még megrendelhetők a kiadó címén (MÁTRIX Kft. 8000 Székesfehérvár, Honvéd u. 8.), vagy megvásárolhatók a Műszaki Könyváruházban (Bp. VI. ker. Liszt F. tér 9.) és a Fókusz Könyváruházban (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!
A szerkesztőség

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

Apróhirdetések

Eladó egy RPR 210-es KÁRMÁN mátrixnyomtató ENTERPRISE kábellel 6000 Ft-ért.
Bozai Gábor, 8000 Székesfehérvár, József A. u. 70/a. fsz. 1.
Telefon: (22) 10-665

HIRDETÉSFELVÉTEL

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

MÁTRIX Kft.
ENTERPRESS
8000 Székesfehérvár,
Honvéd utca 8.

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

LEVELEZÉS

A géppel kapcsolatos témákban levelezne:
Bognár Balázs, 9443 Petőháza, Bartók Béla u. 11.
Czibere Lajos, 4027 Debrecen, Füredi út 1. III/12.
Feczkó Krisztián, 8200 Veszprém, Ányos u. 1/3. Tel.: (80) 29-493
ifj. Márföldi Béla, 4400 Nyíregyháza, Új u. 28.

KLUB
Budapesti Enterprise klub
VSzM közösségi ház
Budapest, XI. ker. Fehérvári út 120.

Tisztelt leendő Szerzőtársak!
Megszűntek a „szigorú" formai feltételek, írásaikat bármilyen formában (papíron, .WP, .TXT fájlként) el tudjuk fogadni. A programokat továbbra is lemezen/kazettán küldjék be!
Címünk: ENTERPRESS, 1399 Budapest, Pf. 701/334

Szerkesztőségünk
a lap szervezésével kapcsolatos teendők ellátására megbízható, agilis Enterprise rajongót keres.
A jelentkezők leveleit postafiókunkba várjuk.

• • • ENTERPRESS *Kéthavilap az Enterprise számítógépek felhasználóinak* • II. évfolyam 6. szám • 1991. november-december • • • Kiadja a MÁTRIX Kft., Székesfehérvár • Felelős kiadó: Annus István ügyvezető • A kiadó címe: MÁTRIX Kft., 8000 Székesfehérvár, Honvéd utca 8. Telefon:(22) 16-520/280 Telefax:(22) 11-585 • • • Felelős szerkesztő: Ujlaki László • A szerkesztőség tagjai: Hajnal Csaba főszerkesztő, Devilsoft, JOVI, Ari Sándor, Bozai Gábor, Haluska László, Lolasoft, Mészáros Gyula • A szerkesztőség csak levélben érhető el! A cím: ENTERPRESS, 1399 Budapest, Pf. 701/334. • Technikai szerkesztő: Szapper László • • • Nyomja a Duna Print Kft., Dunaújváros • Felelős vezető: Farkas István • • • HU ISSN 0866-1820 • • • Terjeszti a Magyar Posta • Előfizethető a HELIR, 1900 Budapest, vagy a MÁTRIX Kft. címén • Előfizetési díj egy évre 294 Ft, fél évre 147 Ft. • • • Következő számunk januárban jelenik meg. • • • Az ENTERPRESS-ben közreadott információk célja az, hogy segítsék, tudnivalókkal lássák el a gép felhasználóit. A közölt programokat, kapcsolási rajzokat, leírásokat mindenki szabadon felhasználhatja, de tilos azokat a kiadó írásbeli engedélye nélkül másolni, terjeszteni. • A szerkesztőség kéziratokat nem őriz meg, és nem küld vissza • • •