Ára: 49,- Ft

ENTERPRESS

KÉTHAVILAP AZ ENTERPRISE SZÁMÍTÓGÉPEK FELHASZNÁLÓINAK

II. évfolyam
1. szám

1991. január — február

MÁTRIX Kft.

Olcsó programok

Mindig vegyes érzelmekkel olvastam azokat a hirdetéseket, amelyekben a hirdető több száz programot kínál olcsón eladásra. Teljesen nyilvánvaló, hogy ezek a programok nem a hirdető saját munkái, hiszen az ENTERPRISE magyarországi megjelenése óta eltelt három és fél évben ennyit el sem lehet készíteni. Ha pedig mégis saját munkák, akkor a tudásuk, színvonaluk erősen megkérdőjelezhető. Előfordulhat ugyan az is, hogy a hirdető ismerősei, barátai programjait árusítja, de ennek minimális a valószínűsége.

Sokat siránkoztunk már azon, hogy kevés a gépre készült komoly szoftver, a létezők pedig túlságosan drágák. Ezért aztán könnyen kísértésbe esünk, amikor egy neppertől ugyanezeket 10-15 forintért beszerezhetjük. A két dolog között azonban szoros, egymás természetének ellentmondó kapcsolat van: ne várjunk addig semmilyen professzionális szoftvert, amíg a piacon a nepperek is hirdethetik magukat! Milyen érzés az, amikor a program írója saját munkáját látja egy nepper listáján, bagóért kimérve? Nyilván ezek után semmilyen nagyobb munkába nem fog bele, pedig milyen nagy szükségünk lenne például egy igazi szövegszerkesztőre, adatbáziskezelőre, táblázatkezelőre, C fordítóra, különféle grafikus programokra és sok minden más alkalmazásra.

Egy profi szoftver kifejlesztésére az ENTERPRISE esetében több hónapra van szükség. Az országban nem sok olyan ember van, aki ezt főállásban csinálja, egyszerűen azért, mert nem lehet megélni belőle. Egy ilyen munka elvégzése után a programozó kap —jó esetben— néhányszor tízezer forintot, és kész. Ha ugyanezt az energiát arra fordítja, hogy mondjuk PC-re ír egy könyvelő programot, akkor az előző összeg többszöröse üti a markát. Ugyanakkor bármilyen jó gép is az ENTERPRISE, a világ már igencsak messze-messze előtte jár: a 16 bites processzorok, a VGA kártyák, a jobbnál jobb rendszerfejlesztési eszközök, a soktaszkos-sokfelhasználós operációs rendszerek, az MS Windows 3.0-ák stb. korában (a nagyobb kategóriákról nem is beszélve) egy profi programozó számára manapság már nem karrier egy Z80 alapú házi számítógépre alkalmazásokat írni.

Azok az "amatőrök" viszont, akik iskola, vagy munka után leülnek gépeikhez, és megszállottként elkezdik verni a billentyűzetet, néha egészen jó dolgokat csinálnak. Ezek a munkák sok kívánni valót hagynak maguk után, de nagyjából megfelelnek annak a célnak, amire eredetileg szánták őket. Eleve irreális lenne elvárni az ilyen művektől azt a szintet, amit mondjuk egy korrektül megtervezett, csoportosan több évig fejlesztett szoftver képvisel. Ezek a művek tehát korántsem az igaziak, de vajon azok a programok, amelyeket az áruházakban vásárolhattunk meg több száz forintért, nem ilyenek voltak..?

A felhasználónak reális áron szoftvert kell adni: programot és leírást. Nálunk eddig (?) az volt a gyakorlat, hogy a program mellé odaadtak valami fecnit, felhasználói leírás gyanánt. A programok mindig nagyon drágák voltak, így maga a forgalmazó késztette másolásra a felhasználókat. Ördögi kör alakult ki: a drága anyagokat csak kis példányszámban tudta a forgalmazó eladni, a drágaság miatt a felhasználók nem vették, hanem másolták a programokat. A forgalmazó továbbra is tartotta (esetleg emelte) árait, a géptulajdonosok pedig egyre jobb másoló programokat használtak, és felvirágzott a nepperek kora. Itt tartunk ma, a legális és korrekt ENTERPRISE ellátás pillanatnyilag a nullával egyenlő.

A dolgokat tehát valahol a helyükre kell tenni: ki kell szorítani a neppereket a piacról, meg kell fizetni a fejlesztőt, olcsón kell adni, és meg kell venni a szoftvert! Az ENTERPRESS a cél elérése érdekében soha nem fog olyan hirdetéseket közölni, amelyben a hirdető hitelessége akár csak egy kicsit is kérdéses. Szeretnénk egy programküldő szolgálatot elindítani, amelynek lényege a következő lenne: azok a géptulajdonosok, akik összetetteb programokat írnak, a lapban néhány mondatos ismertetőt tennének közzé róluk, a lap olvasóinak megvételre kínálhatnák azokat. A legjobb programok sem kerülnének 300 forintnál többe, az átlagár valahol 150 és 200 forint között mozogna, a beérkező megrendeléseket pedig az ENTERPRESS teljesítené, a befolyt összeg nagyobb részét átutalná a szerzőnek. Az ilyen árak nem megfizethetetlenek, a géptulajdonosok, kisebb csoportjaik néha megengedhetnék maguknak, hogy megrendeljenek egy legális példányt az őket érdeklők közül. Nem garantálhatjuk, hogy egy-egy szoftverből 20-30 darabnál többet el tudunk adni, de szerencsére ezek a programíró emberek nem az ENTERPRISE programok készítésére alapozzák egzisztenciájukat, így bizonyára pár ezer forintnál nagyobb fizetségre eleve nem is számítanak.

A programküldő szolgálatot még csak szervezzük, ezért vannak a mondatok feltételes módban. Az ENTERPRISE rossz ellátását nem tudjuk, és nem is akarjuk megoldani, ez egyébként sem a mi feladatunk. Csupán a helyzetet szeretnénk a tőlünk telhető módon egy kicsit normalizálni.

Hajnal Csaba

TARTALOM

ENTERPRESS

Kéthavilap
az ENTERPRISE számítógépek
felhasználóinak

II. évfolyam 1. szám
1991. január-február

Kiadja a
MÁTRIX Számítástechnikai, Kereskedelmi és Szolgáltató Kft.
Székesfehérvár
Felelős kiadó:
Annus István
ügyvezető
Felelős szerkesztő:
Ujlaki László (UL)
A lap szerkesztői:
Hajnal Csaba (HCs)
DevilSoft (Devil)
Bozai Gábor (BG)
Címlap:
Németh Ferenc
Technikai szerkesztő:
Bartha István
A szerkesztőség és a kiadó címe:
8000 Székesfehérvár
Dózsa György tér 10.
Telefon: (22) 16-520/141
Telefax: (22) 11-585

Levélcím:
ENTERPRESS
1399 Budapest
Pf. 701/334

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

Nyomja a
VIDEOTON Nyomda
Székesfehérvár
Felelős vezető:
Gombaszögi József

ISSN 0866-1820
Terjeszti a
Magyar Posta
Előfizethető a HELIR Bp. 1900 címen
Előfizetési díj
egy évre 294 Ft, fél évre 147 Ft
Ára: 49,- Ft

Következő számunk március 20-án jelenik meg

Assembly

3. rész

Az előző részben, gondolom, már sokak számára világossá vált az egyes utasításcsoportokon belüli utasításformák működésének logikája, hiszen ezek egymással teljesen összefüggőek. Éppen ezért ebben a részben — a gyorsabb haladás érdekében — illusztrációként a továbbiakban csak egy-két rövid példát hozok fel. Kicsit előreszaladva: a sorozat ötödik részében már az operációs rendszerrel szeretnék foglalkozni, ezért arra kérem Olvasóimat, hogy írják meg azokat a témákat, amelyekkel kapcsolatban EXOS példaprogramokat szeretnének látni.

Aritmetika (folytatás)

Összeadás és kivonás 16 biten

Ezekről az utasításokról fontos tudnunk, hogy a C (valamint a számunkra érdektelen N és H) biten kívül más jelzőbitet nem állítanak, így az eredmények feldolgozásához a későbbiekben különféle trükköket kell majd alkalmaznunk. Itt a HL regiszterpárnak van kitüntetett szerepe, demonstrálják ezt az

```
ADD HL,BC
ADD HL,DE
ADD HL,HL
ADD HL,SP
```

utasítások, melyek a HL regiszterhez adják a feltüntetett regiszter értékét. Túlcsordulásnál a 15. bitről van átvitel, így ha HL-ben például 57269, BC-ben pedig 8267 van, akkor az

```
ADD HL,BC
```

végrehajtása után HL-ben 0 lesz, és a C bit 1-be billen. Az indexregiszterekhez adhatjuk más regiszterek értékét az

```
ADD IX,BC
ADD IX,DE
ADD IX,IX
ADD IX,SP
ADD IY,BC
ADD IY,DE
ADD IY,IY
ADD IY,SP
```

utasításokkal. Gondolom a működésüket nem kell külön példával illusztrálni, helyette inkább meg kell említeni, hogy a Z80-nak nincsenek olyan utasításai, amelyekkel regiszter tartalmat vonhatnánk ki az indexregiszterek valamelyikéből. Az indexregiszterekhez nem adható hozzá a C bit értéke sem, hiszen ezek a regiszterek (mint a nevük is mutatja) valamilyen adat indexelt elérésére szolgálnak, nem pedig aritmetikai műveletek operandusaiként.

A regisztertartalmakon kívül a C bit értékét is hozzáadhatjuk HL-hez az

```
ADC HL,BC
ADC HL,DE
ADC HL,HL
ADC HL,SP
```

utasításokkal. Ha előző példánál maradunk azzal a különbséggel, hogy kiindulásnál a C bit értéke 1, akkor az

```
ADC HL,BC
```

végrehajtását követően HL-ben 1 lesz, és C bit továbbra is túlcsordulást jelez az 1-be billenésével.

Nincsen 16 bites SUB utasítás, csak elsődlegesen a HL regiszterből kivonó

```
SBC HL,BC
SBC HL,DE
SBC HL,HL
SBC HL,SP
```

utasítások léteznek. Legyen HL-ben 57269, BC-ben 57268, és C bit 1. Az

```
SBC HL,BC
```

után HL-ben 0 lesz, hiszen HL-ből BC-t és C bitet (azaz egyet) is kivont az utasítás. Ha induláskor BC-ben is 57269 lett volna, akkor a műveletet követően HL-ben 65535 és C bit értéke ismét 1 (az alulcsordulást jelezve) lenne.

Logikai műveletek

A Z80 négyféle logikai műveletet tud elvégezni: AND (ÉS), OR (VAGY), XOR (KIZÁRÓ VAGY), NOT (NEM). Először nézzük meg a műveletek jelentését az ún. igazságtáblázatokon!

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

NOT	0	1
	1	0

A processzor ezeket a műveleteket bitről bitre elvégzi a teljes regiszteren, közben a C bit mindig 0 lesz, Z pedig a művelet eredményről ad felvilágosítást. Mint ahogy azt már megszoktuk, a művelet befejezésekor az eredmény az akkumulátorban lesz.

Az AND műveletek

Használhatjuk az

```
AND n
AND A
AND B
AND C
AND D
AND E
AND H
AND L
AND (HL)
AND (IX+d)
AND (IY+d)
```

utasításokat. Tegyük fel, hogy A-ban 147 van, E-ben pedig 117. Az

```
AND E
```

végrehajtása után A = 17 lesz, mert

```
A 10010011
E 01110101
----------
A 00010001
```

(a legalsó sor a művelet utáni akkumulátortartalmat jelképezi). Az AND művelettel a gyakorlatban ún. maszkolásokat végezhetünk: egyes biteket, bitcsoportokat egyszerűen nullába állíthatunk.

Az OR műveletek

A lehetséges utasítások szervezése az AND-hez idomul.

```
OR n
OR A
OR B
OR C
OR D
OR E
OR H
OR L
OR (HL)
OR (IX+d)
OR (IY+d)
```

Az A és az E regiszterek tartalma legyen ugyanaz, mint az előző példában, így az

```
OR E
```

eredményeként A-ba 247 került, mert

```
A  10010011
E  01110101
-----------
A  11110111
```

Az OR művelettel biteket, bitcsoportokat határozottan 1-be állíthatunk.

A XOR műveletek

Már sejthetjük az utasításformákat:

```
XOR n
XOR A
XOR B
XOR C
XOR D
XOR E
XOR H
XOR L
XOR (HL)
XOR (IX+d)
XOR (IY+d)
```

Maradjunk továbbra is az előző példa kiinduló állapotánál, így a

```
XOR E
```

művelet után az akkumulátor 230-al lesz egyenlő. A magyarázat:

```
A   10010011
E   01110101
---------------
A   11100110
```

Ha ismét végrehajtjuk az utasítást, akkor az A regiszter tartalma az eredeti 147 lesz, mert

```
A   11100110
E   01110101
---------------
A   10010011
```

A XOR művelet ismételt végrehajtásával ezek szerint biteket, bitcsoportokat tudunk billegtetni. A műveletnek ezt a kellemes tulajdonságát szokás többnyire a sprite-ok kezelésénél is kihasználni, mert egyszerű a sprite kirakása, visszavétele, az elrontott háttér pedig automatikusan helyreáll.

A NOT művelet

A műveletet a CPL (ComPLement, komplementál, kiegészít) utasítás hajtja végre. Legyen az A regiszter tartalma 147, így a

```
CPL
```

végrehajtását követően az akkumulátorban 108 lesz, hiszen az összes bit az ellenkezőjére változik:

```
A10010011
---------
A01101100
```

Összehasonlító utasítások

Mint az utasításcsoport elnevezése mutatja, ezekkel a műveletekkel két adatot hasonlíthatunk össze. A "központi illetékes" szerepét itt is az akkumulátor tölti be, mindig ez szolgáltatja az egyik operandust. A végrehajtást követően a jelzőbitek az eredménynek megfelelően beállnak, az összehasonlított regiszterek tartalma változatlan marad. A CP (ComPare, összehasonlítás) műveletek lehetséges formái:

```
CP n
CP A
CP B
CP C
CP D
CP E
CP H
CP L
CP (HL)
CP (IX+d)
CP (IY+d)
```

Nézzük meg a C és a Z bitek viselkedését három alapvető esetben! Legyen az akkumulátor tartalma végig 147, a másik operandust E tartalmazza, így a

```
CP E
```

műveletet hajtjuk majd végre az összes példában.

1., Legyen E=147, azaz A=E. Eredményként Z=1, C=0 lesz.

2., Legyen E=146, azaz A>E. Eredményként Z=0, C=0 lesz

3., Legyen E=148, azaz A<E. Eredményként Z=0, C=1 lesz.

A megértésben sokat segíthet, ha tudjuk, hogy a CP utasítások tulajdonképpen kivonást végeznek az eredmény visszaírása nélkül.

Ugró utasítások

Az ugró utasításokkal tulajdonképpen közvetett módon a PC regiszterbe töltünk új adatot, így a processzor a tár más helyéről olvassa be az utasításokat. PC-relatív és abszolút ugrásokat hajthatunk végre, akár feltételesen is, a feltételbitek megváltozása nélkül. A JP (JumP, ugrás) utasítások feltétel nélküli, abszolút formái a következők:

```
JP addr
JP (HL)
JP (IX)
JP (IY)
```

Tegyük fel, hogy addr egyenlő 57269-el, a

```
JP 57269
```

utasítás pedig a 34567-os címen van. A végrehajtás pillanatában PC már a JP utasítás után elhelyezkedő következő utasításra mutat, azaz PC=34570 (a JP addr utasítás 3 bájt hosszú). A processzor ebben a pillanatban azonban felismeri az ugró utasítást, és betölti a PC-be az 57269-es értéket, így a következő utasítást már innen olvassa be, nem pedig a 34570-es címről. Az utasítások feltétel nélküliek, hiszen a végrehajtásuk mindig bekövetkezik. Abszolútak, mert egy konkrét címre hivatkoznak. Feltételes abszolút ugrásokat hajthatunk végre a

```
JP Z,addr
JP NZ,addr
JP C,addr
JP NC,addr
```

és a számunkra egyelőre lényegtelen

```
JP P,addr
JP PE,addr
JP PO,addr
JP M,addr
```

utasításokkal. Szeretném hangsúlyozni, hogy az utasításokban szereplő Z, NZ, C, NC feltételek nem az adott feltételbit értékére, hanem az előzőleg elvégzett művelet eredményére vonatkoznak! A

```
JP Z,addr
```

utasítás például akkor hajtódik végre, ha mondjuk az előző kivonás eredménye nulla volt (ekkor Z bit 1), nem pedig olyankor, amikor a Z bit értéke 0.

Feltétel nélküli relatív ugrást (Jump Relative, relatív ugrás) hajthatunk végre a

```
JR d
```

utasítással. A "d" egy kettes komplemensben ábrázolt számot jelent, aminek báziscímét a JR utasítás utáni cím szolgáltatja. Ez annyit jelent, hogy magától a 2 bájt hosszú JR utasítástól számított -126 ... +129 tartományon belül tudunk ugrani. Az ugrást tehát a JR utasításhoz viszonyítva hajthatjuk végre. Feltételekhez köthetjük a vezérlésátadást a

```
JR Z,d
JR NZ,d
JR C,d
JR NC,d
```

utasításokkal. A "d" értékek kiszámítása természetesen nem a mi feladatunk, ezt majd az assembler megteszi.

Az Z80-nak van egy különleges ciklusszervező utasítása is. A

```
DJNZ d
```

utasítás (Decrement Jump if Non Zero, csökkentés és ugrás, ha nem nulla) a B regiszter tartalmát csökkenti, és amíg az a nullát el nem éri, addig a vezérlés a "d" által meghatározott relatív címen folytatódik. Ha a B regiszter értéke nulla lesz, a DJNZ utasítás után következő utasítást hajtja végre a processzor.

Szubrutinok

A programokban több helyen gyakran ugyanazt a feladatot kell elvégeztetnünk a processzorral (pl. képernyőtörlés, billentyűleütésre várakozás stb.). Ilyen esetekben nem fordítjuk be újra és újra a programrészt a főprogramba, hanem csak egyszer írjuk meg, és a szükséges helyekről ezt az egy rutint hívjuk meg, a rutin lefutása után ott folytatódik a végrehajtás, ahol az abbamaradt. Ezeket a "magányos", de sok helyről hívott programrutinokat nevezzük szubrutinoknak. A sorozat második részében már leírtam az SP regiszter szerepét a szubrutinhívásnál. Ezért itt most csak annyit említenék meg, hogy a regiszterek pillanatnyi állapotának megőrzése a mi feladatunk. Az előző mondat pontos jelentését, és a probléma megoldását később részletezem. A szubrutinhívó utasítások a feltételbiteket nem állítják.

Az egyetlen feltétel nélküli szubrutinhívó CALL (CALL, hívás) utasítás a

```
CALL addr
```

Feltételhez köthetjük a szubrutin meghívását a

```
CALL Z,addr
CALL NZ,addr
CALL C,addr
CALL NC,addr
```

és a

```
CALL P,addr
CALL PE,addr
CALL PO,addr
CALL M,addr
```

utasításokkal.

A szubrutin lefutása után a végrehajtásnak valahogy vissza kell kerülnie a hívó programhoz. Erre szolgálnak a RET (RETurn, vissza) utasítások. Feltétel nélküli formája a

```
RET
```

míg a feltételhez kötöttek a

```
RET Z
RET NZ
RET C
RET NC
RET P
RET PE
RET PO
RET M
```

utasítások, melyek az JP, JR utasításokhoz hasonlóan nem módosítják az F regiszter bitjeit. Nézzünk egy példát szubrutinhívásra konkrét számokkal! Kiindulási állapotban SP=1000, a CALL 3000 utasítás pedig a 2000-es címen legyen (tehát a szubrutin a 3000-es címen kezdődik). A CALL parancs hatására a processzor az SP által címzett verem tetejére teszi a CALL utasítás után következő utasítás címét, a

veremre 2003 kerül, hiszen a CALL utasítás 3 bájt hosszú. Ezután SP értéke kettővel csökken (998), így az már készen áll az esetleges további hívások címeinek elrakásához. A PC-be betöltődik a 3000-es érték, elkezdődik a szubrutin végrehajtása. A rutin végén álló RET hatására az SP értéke kettővel növekszik (ismét 1000), és a verem tetején lévő érték betöltődik a PC-be, így a program végrehajtása 2003-as címen folytatódik.

Az egybájtos RST (ReSTart, újrakezd) utasítások is szubrutinhívást valósítanak meg, de az

```
RST 0h
RST 8h
RST 10h
RST 18h
RST 20h
RST 28h
RST 30h
RST 38h
```

utasítások sokkal rugalmasabbak a CALL-oknál. Az utasítások végén álló számok memóriahelyekre hivatkoznak. Például az

```
RST 0h
```

hatására a vezérlés a 0h címen kezdődő rutinra adódik át, ahol egy újabb szubrutinhívás, ugróutasítás stb. állhat. A dolog rugalmassága abban nyilvánul meg, hogy a főprogramnak mindig csak RST utasítást kell generálnia, de az RST utasítással meghívott rutinban az újabb hívások a főprogram tudta nélkül módosíthatók. Ha például a 0-ás címre egy CALL 57269 utasítást teszünk, majd később ezt mondjuk kicseréljük egy CALL 43923-al (mert mondjuk elkészült a meghívandó rutin újabb verziója), akkor a főprogramból továbbra is az RST 0h utasítást használhatjuk anélkül, hogy tudnánk a végrehajtandó rutin kezdőcímének megváltozásáról. Az RST utasítással hívott rutinból is RET-tel lehet visszatérni, az utasítás nem állítja a jelzőbiteket. Az ENTERPRISE-ban az RST utasításoknak rendkívül fontos szerepük van, többek között az EXOS-t is ezek segítségével hívjuk, lévén az EXOS maga is egy nagy szubrutingyűjtemény.

Veremkezelő utasítások

Nincs sok dolgunk, hiszen már eddig is jónéhány veremkezelő utasítást ismerünk. Az összes olyan ide sorolható, amelyben szerepel az SP regiszter. A veremmel kapcsolatos, a feltételbiteket nem állító utasítások két csoportját kell még megismernünk. Az egyik a veremre teszi az illető regiszter tartalmát, ezek a

```
PUSH AF
PUSH BC
PUSH DE
PUSH HL
PUSH IX
PUSH IY
```

utasítások. A másik csoport az adott regiszterbe tölti a verem tetején lévő szót:

```
POP AF
POP BC
POP DE
POP HL
POP IX
POP IY
```

A veremre tehát ezekkel az utasításokkal tehetünk, és vehetünk le ideiglenesen ott tárolt adatot. Ezt a módszert használjuk akkor, amikor például a szubrutinok elején elmentjük az összes, a rutinban használt regiszter tartalmát azért, hogy a főprogram a szubrutin hívás előtti helyes adatokkal tudjon majd tovább futni a szubrutinból való visszatérés után (erre utaltam a "regiszterek pillanatnyi állapotának megőrzése"-ként). Ha a szubrutinnak a vermen keresztül adunk át paramétereket, akkor ne feledkezzünk meg arról, hogy a verem tetején PC értéke van, a paraméterek ez alatt vannak! Látható, hogy csak olyan veremkezelő utasítások vannak, amelyekben regiszterpár szerepel.

Bitműveletek

A bitműveletekkel regiszterek, tárrekeszek bitjeit tudjuk 1-be vagy 0-ba állítani, illetve meg tudjuk vizsgálni a bitek állását. Ezekből az utasításokból olyan sok van (összesen 240), hogy értelmetlen felsorolni az összeset, helyette inkább egy kissé általánosított formába öntve nézzük meg őket. Vezessük be a "b" jelölést, ahol "b" a bájton belüli bitsorszámot jelöli (b7...b0), így 7>=b>=0.

Bitállítást végeznek a SET (SET, beállítás) utasítások:

```
SET b,A
SET b,B
SET b,C
SET b,D
SET b,E
SET b,H
SET b,L
SET b,(HL)
SET b,(IX+d)
SET b,(IY+d)
```

Példaképpen emeljük ki a

```
SET 3,E
```

utasítást, melynek végrehajtása után az E regiszter 3. bitje 1-be billen. A biteket törölhetjük a RES (RESet, törlés) csoport utasításaival:

```
RES b,A
RES b,B
RES b,C
RES b,D
RES b,E
RES b,H
RES b,L
RES b,(HL)
RES b,(IX+d)
RES b,(IY+d)
```

Ha az előző példán felbuzdulva megvizsgáljuk a

```
RES 3,E
```

utasítást, akkor a végrehajtása után az E regiszter 3. bitje törlődik. A BIT és a RES utasítások a feltételbiteket nem állítják, nem így a bitvizsgáló BIT (BIt Test, bitvizsgálat) utasítások:

```
BIT b,A
BIT b,B
BIT b,C
BIT b,D
BIT b,E
BIT b,H
BIT b,L
BIT b,(HL)
BIT b,(IX+d)
BIT b,(IY+d)
```

Ha a vizsgált bit értéke 1, akkor Z bit 0, egyébként Z bit 1 lesz; C bit egyik esetben sem változik. Legyen az E regiszterben 8, ekkor a

```
BIT 3,E
```

utasítás végrehajtását követően a Z bit 0 lesz, jelezve a 3. bit 1-es értékét.

Portműveletek

A Z80 portjain keresztül kommunikál más áramkörökkel, perifériákkal (a port kaput jelent), nincs ez másképpen az ENTERPRISE-ban sem: portműveletekkel lapozhatjuk a memóriát; az EXOS rutinjai is ilyen műveletekkel végzik a kooprocesszorok, a nyomtató, a magnó stb. vezérlését. A portok is saját címmel azonosíthatók, ezeket portcímeknek hívjuk. A Z80-nak 256 portcíme van, 1 bájtos adatokat olvashatunk be róluk, illetve írhatunk ki rájuk.

A beolvasást az IN (INput, bemenő) utasításokkal végezhetjük. Az akkumulátorba az

```
IN A,(p)
```

utasítással tudunk közvetlenül adatot beolvasni a "p" portcímről (0<=p<=255). Ez a közvetlen címzésű forma nem állítja a jelzőbiteket. Legyen a portcím B3h, ekkor az

```
IN A,(B3h)
```

után az akkumulátorban lesz a B3h port tartalma. Ha a port címét indirekt módon akarjuk megadni, és más regiszterben szeretnénk megkapni a port értékét, akkor az

```
IN A,(C)
IN B,(C)
IN C,(C)
IN D,(C)
IN E,(C)
IN H,(C)
IN L,(C)
```

utasítások küzül választhatunk, ahol C a C regisztert jelenti, amely a port címét tartalmazza. A beolvasott értéknek megfelelően megváltozik a Z bit, a C bit változatlan marad. Portra írni az OUT (OUTput, kimenő) utasításokkal tudunk, közvetlen címmegadással az akkumulátorból az

```
OUT (p),A
```

utasítással küldhetünk adatot valamelyik eszköznek. Indirekt módúak az

```
OUT (C),A
OUT (C),B
OUT (C),C
OUT (C),D
OUT (C),E
OUT (C)H
OUT (C),L
```

utasítások. Az OUT utasítások nem módosítják a jelzőbiteket.

(folytatjuk)

-HCs-

A PASCAL

Második rész

A hosszúra nyúlt bevezetés után lássuk a Pascal jellemzőit.

A Pascalt ismertető könyvek általában a nyelv legalacsonyabb szintű építőelemeivel kezdik a tárgyalást, ezekből építik fel az egyre bonyolultabb szerkezeteket. Ebben a sorozatban ellenkező úton járunk: a legmagasabb szintről kezdve haladunk a legalacsonyabbak felé. Ez egyébként megfelel a korszerű programtervezési módszereknél alkalmazott top-down, azaz felülről lefelé haladó tervezési eljárásnak. Ugyanakkor azt is lehetővé teszi, hogy *in medias res* - azonnal a dolgok közepébe vágva - érdekesebbé és olvashatóbbá tegyük a leírást.

Ahhoz, hogy a menet közben példákat is megadhassunk, előfordulhat, hogy időnként még nem definiált dolgokra is hivatkoznunk kell; ezeknek az értelmezését vagy az olvasó józan belátására bízzuk, vagy ideiglenesen megmagyarázzuk.

A Pascal program

Az első, amivel meg kell ismerkednünk, a program felépítése. Míg a Basic-ben nincs különös megkötés a program fogalmára - programnak számít minden, amiben Basic utasítások vannak -, addig a Pascal szigorúbb szerkezete meghatározza a program fogalmát. A Pascal program egy programfejből és egy programtörzsből (a terminológia szerint *blokkból*) áll. Ezeket a megfelelő elválaszójellel kell egymástól (és a külvilágtól) elválasztani. **(1. ábra)**

A programfej jelzi a program kezdetét és azonosítja a programot. A pontosvessző a - kötelező - elválasztójel.

A blokk vagy programtörzs végez minden hasznos tevékenységet. A blokk után álló pont a program végét jelzi. Ne hagyjuk el!

Maga a programfej a **PROGRAM** kulcsszóból és a program nevéből áll. A legtöbb fordítónak mindegy, hogy nagy- vagy kisbetűvel írjuk-e. A program neve egy tetszőleges azonosító, de célszerű a program tényleges nevét megadni. **(2. ábra)**

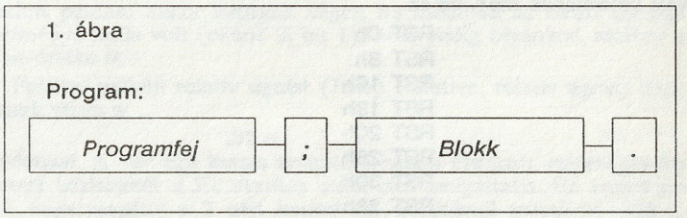
A blokk két fő részből áll: a deklarációs részből és az utasításrészből. **(3. ábra)**

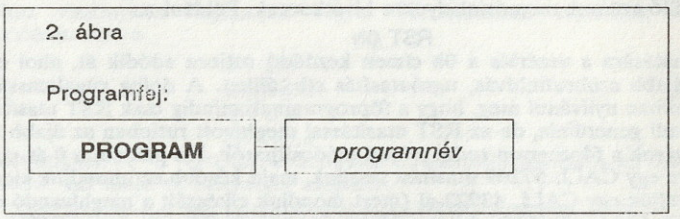
A deklarációs rész egyrészt meghatározza az adatszerkezeteket, amelyekkel a program dolgozni fog, másrészt - az adatszerkezetekhez hasonlóan - meghatározza a program által használni kívánt absztrakt eljárásokat. Az utasításrész - más programnyelvek terminológiája szerint a "főprogram" - végzi a tulajdonképpeni műveleteket. Mind a deklarációs rész, mind az utasításrész lehet üres is.

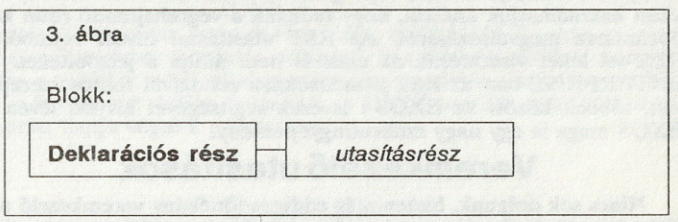
A deklarációs rész a konstansdefiníciókat, típusdefiníciókat, változódefiníciókat, valamint az eljárás- és függvénydefiníciókat tartalmazza. Ezekre hamarosan kitérünk. Az utasításrészt a **BEGIN** és az **END** kulcsszó határolja. Ezeket úgy kell felfogni, mint egyfajta zárójeleket; a nevük is utasítászárójel. Minden, ami az utasítászárójel-páron belül van, együtt kezelendő, egyetlen utasításnak számít (lásd az utasításssorozat fogalmát a bevezető részben). Az eljárási rész után egy pont áll, ami az egész programot lezárja. A legegyszerűbb Pascal program tehát így néz ki:

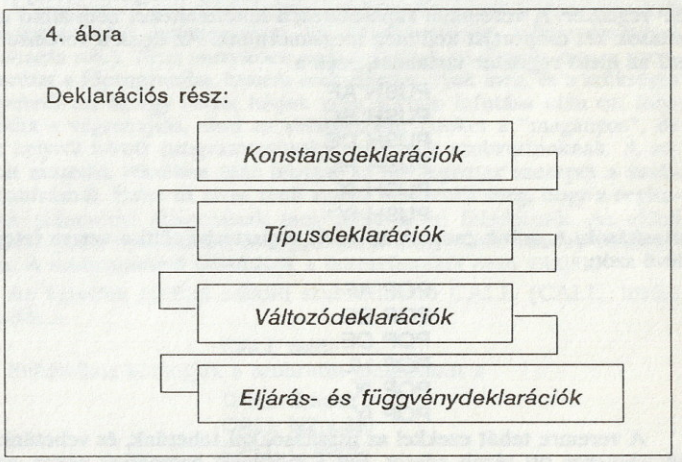
```
PROGRAM programnév;
BEGIN
END.
```

Ez a program az égvilágon semmit nem csinál, viszont kipróbálható vele a fordítóprogram működése. A programnév helyére egy tetszőleges azonosítót - célszerűen a program nevét - kell beírni. Az azonosító - mint a legtöbb programnyelvben - betűvel kezdődő és betűvel vagy számmal folytatódó karaktersorozat. A hosszát egyes fordítóprogramok 8 vagy esetleg 16 karakterben korlátozhatják, vagy csak az első *n* karakterét tekintik érvényesnek; mások tetszőleges hosszúságig elfogadják. Akármit is enged meg a fordítónk, ne vigyük túlzásba! Egyikünk sem képes első látásra megkülönböztetni olyan azonosítókat, mint XXXXXXXXXXABBABABABBA vagy XXXXXXXXXABBABBABABA.









Azoknak, akik elsőre ennél többet akarnak, a program eljárási részét kibővítjük egy utasítással, amely a később tárgyalandó WRITELN eljárást használja; mivel ez már nem minta, hanem konkrét, működő program, a programnév helyére egy konkrét nevet írtunk be.

```
PROGRAM elso_programom;
  BEGIN
    WRITELN( 'Tessék, itt vagyok!' );
  END.
```

A program, elvárásunknak megfelelően, az aposztrofok közötti szöveget írja majd ki a képernyőre.

A deklarációs rész

A Pascal program deklarációs része határozza meg, mint már említettük, azokat az adat- és eljárási szerkezeteket, amelyeket a program később használni fog. A deklarációs rész elemei (ebben sorrendben): a konstansdeklarációk, a típusdeklarációk, a változódeklarációk, és - egymással tetszőleges sorrendben keverve - az eljárás- és függvénydeklarációk. Az összes felsorolt elem el is hagyható. Megjegyezzük még, hogy egyes Pascal-implementációk, így a Turbo-Pascal is, nem ragaszkodnak a deklarációk itt meghatározott kötött sorrendjéhez, az egyes deklarációk keverve is előfordulhatnak.

A deklarációs rész tulajdonképpen a címkedeklarációkkal kezdődne. A címke szerepe, hogy **GOTO** utasítással ráugorhassunk; mivel azonban nincs olyan program, amelyik ne lenne ugrás nélkül is megírható, a sorozatban mind a címkéket, mind a **GOTO** utasítást elfelejtjük. Aki ragaszkodik az ugráláshoz, a számtalan szakkönyv egyikében utánanézhet, hogyan teheti programját áttekinthetetlenné és hibákra érzékennyé. **(4. ábra)**

A Basic nyelvben a konstans tulajdonképpen egy érték, amelyre felírásával hivatkozunk; ez az úgynevezett alaki konstans. Számos esetben azonban a konstansok értéke a program fejlesztése során változhat. Például menet közben rájövünk, hogy más méretű képernyőt vagy ablakot kell használnunk, mint amilyent eredetileg akartunk. Ilyenkor, ha a képernyő vagy ablak vízszintes és függőleges méretét, illetve az ettől függő egyéb állandókat számértékként (alaki konstansként) adtuk meg, a program módosítása emberfeletti nehézségekbe ütközhet: meg kell keresni az összes konstans összes előfordulási helyét és átírni az új értékre. Más esetben igen kellemetlen lehet a konstans értékét állandóan kikeresni és a kellő pontossággal beírni a kifejezésekbe. Az előrelátó programozó ilyen esetekben a konstans helyett változót használ, amit a program elején feltölt a megfelelő értékkel. Egy megváltozott értéket ilyenkor csak egyetlen egy helyen kell módosítani. A hibalehetőség itt az, hogy elfelejtjük a kezdeti értékadást; illetve valahol a programban tévedésből megváltoztatjuk a "konstans" értékét. A Pascal bevezeti a név szerinti konstans fogalmát. A konstans ekkor egy ugyanolyan azonosító, mint a változók azonosítója, de értéke állandó. Ugyanúgy szerepelhet kifejezésekben, mint a változók, de új értéket nem kaphat. A konstansdeklarációk rész felsorolja az ilyen név szerinti konstansokat, meghatározva értéküket. A deklarációkat a CONST kulcsszó vezeti be, ezt követik az egyes konkrét konstansdefiníciók. A példában - a Pascal szabályainak megfelelően - kapcsos zárójelben álló megjegyzés magyarázza az egyes konstansok szerepét.

```
CONST
  X_MAX = 1329;          { a max. vízszintes felbontás }
  Y_MAX = 759;           { a max. függőleges felbontás }
  UZENET = '*** Hibás érték! ***';  { hibaüzenet }
  SZOKOZ = ' ';          { a szóköz }
  URES_HELY = SZOKOZ;  { egy konstans egy másikra hivatkozik }
  PI = 3.14159265358;
```

Látható, hogy az összes ismert típussal megadható konstans (a fordítóprogram a konstans alakjából következtet az alkalmazandó típusra). Az X_MAX és az Y_MAX konstans így egész típusú lesz; az UZENET sztring típusú, míg a SZOKOZ (és a rajta keresztül definiált URES_HELY) karakter típusú. A PI konstans természetesen valós típusú lesz. (A Pascal implementációk - így a Turbo-Pascal is - a Pi értékét egy beépített argumentum nélküli függvénnyel teszik hozzáférhetővé; ilyen esetben nem szükséges a PI konstanst deklarálnunk.) A Pascal egyes továbbfejlesztései további lehetőségeket engednek meg, úgymint (fordítási időben kiszámítható eredményű) kifejezések alkalmazása a konstansok definíciójában, illetve a tipizált konstans; ezeket majd később tárgyaljuk.

A típusdeklarációk

A Pascal nyelv igazi gazdagsága a felhasználó által definiálható típusoknál mutatkozik meg igazán. A típusdeklarációk segítségével a mindenkori feladatnak legjobban megfelelő adattípusok hozhatók létre. Ez a programozási feladat bonyolultságát több nagyságrenddel is csökkentheti. A saját típusok létrehozásának két módja van: új típus vagy egy meglévő (szabványos vagy korábban a felhasználó által definiált) típus szűkítésével hozható létre, vagy pedig ugyanilyen meglévő típusok kombinálásával. Először gyorsan tekintsük át a nyelvben már meglévő, szabványos típusokat, röviden kitérve gépi reprezentációjukra is.

Az egész típus

Az egész (INTEGER) típus az egész számok egy részhalmazát veheti fel értékként. Mivel a gépi megvalósításánál (a gyorsaság miatt) rendszerint két bájton ábrázolják, a szokásos értéktartománya a kettes komplemensű aritmetika szabályainak megfelelően -32768-tól +32767-ig terjed. Ez elég kevésnek tűnik, de a gyakorlat azt mutatja, hogy az egész típust rendszerint csak ciklusszervezésre, számlálásra, és még néhány egyszerűbb feladatra használják, erre pedig általában ez az értéktartomány elegendő. Ha nagyobb értékekkel kell dolgoznunk, használjuk a valós típust. Az adott Pascal implemetációban a használható legnagyobb egész értéket a MAXINT (argumentum nélküli) függvény adja meg.

Egyes Pascal implementációk ismerik még az egy bájt hosszúságú egész típust is, ennek neve BYTE, értéktartománya 0-tól 255-ig terjed. A bájt típus mind az egész, mind pedig a (később ismertetendő) karakter típussal kompatibilis, azaz értéket adhat át azoknak vagy vehet át ezektől típuskonverzió nélkül. Egyes implementációk ugyancsak ismerik a szó (WORD) típust; ezt szintén két bájton ábrázolják, de előjel nélküli számként értelmezve. Az értéktartomány így 0-tól 65535-ig terjed. Segítségével nagyobb ciklusok szervezhetők, mint az egész típussal, és a 16 bites címeknél a teljes címtartomány átfogható.

A valós típus

A valós (REAL) típus a valós számok egy részhalmazát veheti fel értékként. Az egyes gépi megvalósításokban csak az a közös, hogy minden esetben lebegőpontos számként ábrázolják; egyébként jelentős eltérések lehetnek mind a maximális értékben, mind az elérhető (relatív) pontosságban. A Turbo-Pascal például a valós típust 6 bájton ábrázolja: egy bájton az exponenst kettes komplemens alakban, öt bájton pedig a mantisszát, ugyancsak kettes komplemens alakban. Az értéktartomány így a -9.9999999999E+37 értéktől a +9.9999999999E+37 értékig terjed, míg az exponens értéktartománya -38-tól +37-ig érvényes. Az értékes tizedesjegyek száma 11. Más rendszerekben a megoldás más lehet.

(folytatjuk)

—UL—

Lehetőségek Páratlan Tárháza (LPT) 2.

Az első részben megismerkedtünk a *pixel* és *lpixel* grafikus üzemmódokkal, és ahogy azt jeleztem, most megnézzük a többi képmódot is.

A grafikus videomód harmadik tagja az attribútum mód. Ezzel a képtípussal találkozhatunk legtöbbször a játékprogramokban, mivel a ZX Spectrum csak ezt ismeri, és a legtöbb játékot Spectrumról konvertálták át. Ezt a videomódot az ENTERPRISE programozók nem szívesen alkalmazzák, pedig hátrányai mellet sok előnyös tulajdonsága is van. De vajon hogyan épül fel egy attribútum kép?

A leglényegesebb, hogy különválik a képpont- és a színterület. Az LPB-ben az elsődleges video adatcím (LD1) adja meg a képterület memóriacímét, a másodlagos adatcím (LD2) pedig a színterület memóriacímét. A képpontok területe hasonló a kétszínű üzemmód memóriaterületéhez, vagyis minden képpontnak egy bit felel meg. Ha a bit beállított állapotú, akkor a bitnek megfelelő pont tintaszínű lesz, ellenkező esetben papírszínű. Azt, hogy a papír és tintaszínek valójában mit jelentenek, a színterületen adhatjuk meg. A képpontmemória minden bájtjához tartozik ugyanis egy attribútumbájt, amelynek felépítése a következő:

b0..b3 Tintaszín palettaszám. (0..15)

b4..b7 Papírszín palettaszám. (0..15)

Mint az a fentiekből is látható, az attribútum módban 16 szín használatára van lehetőségünk, olyan felbontás mellett, mint amilyen a 4 színű pixel módé. Örömünk azonban nem lehet teljes, mert egy képbájthoz csak egy attribútumbájt tartozhat. Ebből következően az egy bájtban lévő nyolc ponthoz csak két színből választhatjuk ki az aktuálisat, hasonlóan a C64-es képkezeléséhez. A legnagyobb hátrány azonban az attribútum mód használatakor az, hogy nincs olyan rajzolóprogram, amely megfelelően kezelné azt.

Még egy fontos dologra fel kell hívnom a figyelmet: az attribútum videomódot kétszínű színmóddal kell használni, az LPB második bájtjának 14h-nak kell lennie.

Ezzel a grafikus videomódok ismertetésének végére értünk, most lássuk a karakteres képmódokat!

A karakteres üzemmódokból három féle áll a rendelkezésünkre, amelyek rendre a következők: 256 karakteres (CH 256), 128 karakteres (CH 128) és 64 karakteres (CH 64) mód. A három üzemmód csak a használható karakterek számában különbözik egymástól. Mindhárom módban az LD1 tartalmazza a karakterkódok memóriacímét, ahol a kijelzendő karakterek ASCII kódjait tároljuk. Az LD2 a karakterkészlet kezdőcímét tartalmazza, elosztva az üzemmódban használható karakterek számával, tehát CH 256 módban 256-tal, CH 128 módban 128-cal és CH 64 módban 64-gyel kell osztani a karakterkészlet Nick-címét. A karakterkészlet felépítése a következő: először a karakterek első bájtját tároljuk az ASCII kód szerinti sorrendben, majd a másodikat és így tovább. Azt, hogy hány bájt (illetve pont) magas legyen egy karakter, az LPB első bájtjával tudjuk beállítani, így 1 és 256 között bármilyen magasságú karaktereket használhatunk.

A karakterkészletben lévő bájtok a képernyőn a színmódtól függően kerülnek megjelenítésre, hasonlóan a pixel grafikus módnál megismertekhez. A különböző színű karakterek megjelenítéséhez az LPB-ben két vezérlőbitet -nevezetesen az LS BALT-tot és az MS BALT-tot- használhatunk. A vezérlőbitek használatáról részletesen a következő részben szólok majd.

Ezzel az LPB ismertetésének a végére értünk. Ha valami nem volt teljesen érthető, vagy nem volt kellően részletes, kérem írják meg, és akkor visszatérek rá.

Most pedig jöhetnek az ígért rutinok, amelyekkel különböző effekteket állíthatunk elő. Először azonban szükség van egy LPT-re, amelyet az alábbi rutinnal építhetünk fel. A program ASMON-nal fordítható, de egyszerűen átvihető más assemblerre is. A rutin az elején található változók beállítása után bármilyen pixel grafikus képernyő előállítására rávehető. Természetesen egy kis átalakítással karakteres és attribútum képernyők is kreálhatók vele (ekkor az LD2 pozíciót is ki kell tölteni). A rutin automatikusan kiszámolja a keret méreteit, és a szinkronizálást is elvégzi. Meghívása a CALL MAKELPT utasítással lehetséges. Visszatérés előtt a rutin bekapcsolja az előállított LPT-t. Fontos, hogy hívás előtt belapozzuk azt a szegmenst, amelyikre az LPT kerül.

(folytatjuk)

-DEVIL-

```
        .RADIX  10H     ;alapert. szamrendszer
LNUM    EQU 100         ;LPB-k szama
LPT     EQU 4000        ;LPT Z80 kezdocime
LPT2    EQU 0           ;LPT Nick-kezdocime
VIDEO   EQU 4000        ;videomemoria Nick-kezdocime
MODS    EQU 1           ;egy MODSOR pixelsorainak szama
VIDM    EQU 12          ;szin/videomod (LPB 2.byte)
LM      EQU 0BH         ;bal margo
RM      EQU 33          ;jobb margo
C0      EQU 0           ;palettaszinek COL0-COL7
C1      EQU 1
C2      EQU 2
C3      EQU 3
C4      EQU 4
C5      EQU 5
C6      EQU 6
C7      EQU 7
LENGHT  EQU ((RM MOD 3FH)-(LM MOD 3FH))*2*MODS
BORDER  EQU (134-LNUM*MODS)/2
MAKELPT LD A,LNUM MOD 100       ;ciklusvaltozo
        LD DE,LPT       ;LPT Z80-as cime
        EXX
        LD HL,LPT+4     ;aktualis LPB LD1 pozicioja
        LD DE,VIDEO     ;videomemoria Nick-cime
        EXX
NEXTLPB LD HL,LPB
        LD BC,10
        LDIR            ;egy LPB letarolasa
        EXX
        LD (HL),E       ;LD1 pozicio feltoltese
        INC HL
        LD (HL),D
        LD BC,0FH
        ADD HL,BC       ;kovetkezo LPB LD1 pozicioja
        EX DE,HL
        LD BC,LENGHT
        ADD HL,BC       ;kovetkezo LPB videomemoria cime
        EX DE,HL
        EXX
        DEC A           ;ciklusvaltozo csokkentese
        JR NZ,NEXTLPB
        LD HL,SYNC
        LD BC,LENSYNC
        LDIR            ;szinkron LPB-k letarolasa
        LD HL,LPT2
        SRA H           ;LPT Nick-cimenek osztasa 16-tal
        RR L
        SRA H
        RR L
        SRA H
        RR L
        SRA H
        RR L
        LD A,L
        OUT (82),A      ;az LPT bekapcsolasa
        LD A,H
        OUT (83),A
        OR 40
        OUT (83),A
        OR 80
        OUT (83),A
        RET
LPB     DEFB 100-MODS,VIDM,LM,RM,0,0,0,0
        DEFB C0,C1,C2,C3,C4,C5,C6,C7
SYNC    DEFB 100-BORDER+0CH,12,3FH,0,0,0
        DEFB 0,0,0,0,0,0,0,0,0,0
        DEFB 0FCH,10,0,3FH,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0
        DEFB 0FFH,10,3FH,20,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0
        DEFB 100-BORDER-0CH,93,3FH,0,0,0
        DEFB 0,0,0,0,0,0,0,0,0,0
LENSYNC EQU $-SYNC
```

Betöltési problémák

Valószínűleg az összes EXDOS felhasználónak az volt az első dolga, amikor üzembehelyezte lemezegységes rendszerét, hogy a kazettán lévő programjait átmásolta lemezre. A Spectrumról átírt játékprogramok másolásánál valószínűleg nem is volt sok gondja, csak a magnó lassúsága. Az eredeti ENTERPRISE játék- és felhasználói programok nagy része azonban már több gondot okozott. Most az ilyen programok lemezessé konvertálásához szeretnék néhány hasznos tanácsot adni.

A gép megjelenésekor még nem lehetett lemezmeghajtót illeszteni az ENTERPRISE-hoz, ezért az akkor készült programok eleve magnós konfigurációhoz készültek. Átírás szempontjából ezek a programok is több csoportba sorolhatók.

Általában minden programnak van egy betöltője, ami lehet egy rövid Basic program, vagy egy gépi kódú, 5-ös fejléccel rendelkező, ún. új alkalmazói program (NAP=New Application Program). Ez a betöltő elindulása után beolvassa a teljes programot, majd átadja annak a vezérlést.

Ahhoz, hogy be tudjunk tölteni valamit, először meg kell nyitni egy csatornát egy eszközre. Basic-ben ez egy OPEN #csatorna:"eszköz:fájlnév" utasítással lehetséges, ahol a *csatorna* a csatorna száma, az *eszköz* lehet TAPE: a magnó esetében, a lemezegységnél pedig a meghajtó azonosítója. Ha nem írunk eszköznevet, akkor a csatorna az alapértelmezésű eszközre fog vonatkozni, ami alapkonfigurációnál a magnó.

Ha a géphez lemezegységet csatlakoztatunk, akkor az lesz az alapértelmezésben használt eszköz. Lehetőség van arra is, hogy fájlnevet sem adunk meg, ekkor a magnós konfigurációnál a szalagon elsőnek megtalált fájlt, lemeznél pedig a START nevű fájlt nyitja meg az EXOS.

A csatornanyitás gépi kódban sem sokkal bonyolultabb. Az A regiszterbe a csatornaszámot, a DE regiszterpárba pedig a fájlnév memóriabeli kezdőcímét kell betölteni, és végül egy EXOS 1 hívást kiadni. A fájlnév megadása hasonló, mint a Basicben, azzal a különbséggel, hogy az első bájt a karakterfüzér hosszát adja meg bájtokban.

A két lehetőség keveredésével is találkozhatunk: a betöltő Basic nyelvű, de gépi kódban történik a beolvasás. Ilyenkor a Basic program az ALLOCATE és CODE utasításokkal előállít egy gépi kódú rutint, majd CALL USR utasítással meghívja azt.

Az eddigi információk alapján már valószínűleg érthetővé válik az a jelenség, amelynél egy lemezre másolt programot próbálunk betölteni, és az első blokk után leáll a gép, amely aztán kazettáról akar olvasni, vagy egyszerűen lefagy. Ha az első eset fordul elő -amikor kazettáról akar a gép olvasni-, akkor a csatornanyitásban a TAPE: eszköznév szerepel. Ezen lehet a legkönnyebben segíteni, egyszerűen át kell írni a TAPE: szöveget a program második fájljának nevére.

Ha a betöltő Basic és találunk benne egy OPEN #csatorna:"TAPE:..." utasítást, akkor azt javítsuk ki úgy, hogy az idézőjelben csak a fájl neve szerepeljen. Ha a betöltő gépi kódú, akkor töltsük azt be egy assemblerbe (célszerű az ASMON-t használni, mert az megadja a töltés végcímét), és keressük meg a TAPE: sztringet. Ha megtaláltuk, javítsuk ki, de vigyázzunk rá, hogy a hosszbájt is jó legyen. Basicből hívott gépi kód esetén a HEX$ függvény argumentumában kell megkeresnünk az "54,41,50,45,3A" számsort, ami a TAPE: sztring ASCII megfelelője hexadecimális alakban. Ezt már könnyedén ki tudjuk javítani (itt is figyeljünk a hosszbájtra).

A másik esetben nem is próbál a gép továbbtölteni, hanem csak lefagy. Ez akkor fordul elő, ha a megnyitáskor sem eszköznév, sem pedig fájlnév nem volt megadva. Magnónál ez nem gond, mert a soronkövetkező fájl betöltődik. Az EXDOS azonban megnézi, hogy van-e START nevű fájl a lemezen, és ha nincs, akkor egy hibakóddal visszatér. Ez az eset Basic betöltőnél nem fordul elő. Gépi kódnál (Basic-ből hívottnál is) meg kell keresni, hogy hol nyitja meg a csatornát a program. Ez gépi kódban egy EXOS 1 hívást, hexadumpnál egy "F7 01" számsort jelent. A hívás előtt mindig van egy "LD DE,cím" utasítás, ahol a "cím" a fájl nevének memóriabeli kezdőcímével egyenlő. A hexadumpban ez a következőképpen néz ki: "11 x1 x2", ahol az x1 az alsó, x2 a cím felső bájtja tehát a cím=x1+256*x2. Ha cím által meghatározott memóriarész tartalmát megnézzük, akkor látjuk, hogy ott egy 00 bájt szerepel. Általában itt nincs is akkora hely, amelyen elférne a betöltendő fájl neve, így kell keresnünk valahol annyi üres helyet a programban, ahol a név elfér (ha máshol nem, akkor a program végén). Ide beírjuk a fájl nevét hosszbájttal az elején, majd azt a címet ahová írtuk, beírjuk az "LD DE,cím" utasításba operanduskénti (x1, x2 helyére). A gépi kódú programot az assemblerbe az 10F0h címre érdemes tölteni, így a programban szereplő címek 1000h értékkel lesznek eltolva.

Ezek voltak az egyszerűbb esetek (már amenyire ez egyszerű lehet). Ezeknél valamivel bonyolultabb a helyzet, ha nem egy fájlt tölt be a betöltő, hanem többet és mindegyikhez ugyanaz a fájlnév van rendelve (ami lehet 00 is). Ebben az esetben mindegyik csatornanyitásnál át kell írni a majdani fájlnevek kezdőcímeit (ezek kerülnek DE-be) különböző értékekre (ott, ahol erre van hely a programban), és az egyes címekre be kell írni a fájlneveket. Ilyenkor általában már csak a program után van annyi helyünk, ahol ez elfér, tehát hosszabb lesz a betöltő. Ha az 10F0h címre töltöttük a programot, akkor az 10F2h címen található a hossza, amit át kell írni az új hosszértékre.

Ezeknél bonyolutabb esetek megoldására már csak a gépi kódú programozásban jártasak vállalkozhatnak eredményesen.

Ilyen eset az, amikor a program konkrét memóriacímeket használ a rendszerszegmensben, vagy olyan memóriaterületet igénylő videocsatornát akar megnyitni, amelyhez nincs elég hely, ugyanis az EXDOS lefoglal kb. 4 kilobájtot. Ekkor a fájlok betöltése után elfelejtetjük a géppel a lemezegység meglétét, amit a következő módon lehet elérni: a rendszerszegmensben az EXOS ROM-ok táblázatát át kell írni úgy, hogy csak az 1-es és a Basic szegmens maradjon meg. Az ABBDh címtől 0,0,0,0,1,0,0,0,x,0,0,0,0 számsort kell beírni ahol az "x" a Basic szegmens száma (angol gépnél 4, németnél 5), majd a BF95h és a BF97h címre (a rendszerszegmensre) ABBDh-t kell írni. Végezetül egy EXOS 0 hívást kell végrehajtatni a C regiszter 0 értéke mellett.

Az eddig felsorolt esetek a régebben (1985) készült programoknál fordultak elő. Az újabb készítésű programokat — gondolok itt az "a" Studio és a Novotrade programjaira — már szándékosan készítették úgy, hogy azok csak magnóról induljanak el, ez ugyanis egyfajta programvédelemként szolgál.

Az egyik megoldás az, amikor a programot saját betöltővel látják el, így ezeket a programokat az EXOS nem képes beolvasni. Ilyenek például az "EGGS_OF_DEATH", a "MIRROR_WORLD" című játékprogramok. A megoldás az, hogy a fájlokat a saját loaderükkel betöltjük, majd EXOS formátumban kimentjük. A betöltőt pedig átírjuk olyanra, hogy az az EXOS-t használja beolvasásra a speciális loader helyett.

A másik programvédelmi lehetőséget a "FINE_PEN" spritetervező programnál figyelhetjük meg. Az elv a következő: a betöltő egy 6-os fejlécű abszolút rendszerbővítő, amely a betöltődés után elindul. Korrekt működés helyett azonban végrehajt egy EXOS 0 hívást, melynek hatására újrainicializálódnak a beépített és bővítő perifériák, valamint rendszerbővítők. Az alapperifériák után a rendszerbővítő ismét megkapja a vezérlést, hogy inicializálódjon. Ehelyett ismét mást csinál: átveszi a vezérlést, és elkezdi betölteni a főprogramot. Mivel csak az alapperifériák inicializálása történt meg, ezért az EXOS nem tud róla, hogy lemezegység is van a konfigurációban. Ennek megoldására az Örökéletkódoknál mutatok egy példát.

-DEVIL-

Az EnterSPRITE animátor

Az első ENTERPRISE programok egyike volt az EnterSPRITE, amely 8 sprite kezelését támogatja Basic-ben 11 utasítással és 2 függvénnyel. Az $$REL fejlécű, felhasználói áthelyezhető modulként inicializálódó programot eredetileg kazettán árusították, de lemezről is elindul.

A Basic bővítő parancsai, függvényei a következők:

**SLOAD <sorszám>,<$név>** : sprite betöltése. A programban a sprite-okhoz egy-egy sorszámot kell rendelni, ez alapján hivatkozhatunk rájuk, a <sorszám> 1 és 8 között lehet. A <$név> azt a nevet jelenti, amellyel a sprite-ot a tervezés után rögzítettük. Természetesen itt megadható az eszköznév, a meghajtó, és az elérési útvonal is.
Példa: SLOAD 4,"B:\SPRED\UMBRELLA.SPR"

ANIMATE <sorszám>,[fázis_1],[fázis_2],...[fázis_15] : egy sprite-on belül az animáció során megjelenítendő fázisok sorrendjének megadása. A fázisok tetszőleges sorrendben követhetik egymást, megadásuk nem kötelező. A sprite tehát 8 fázist tartalmaz, ezzel a paranccsal a fázisok megjelenési sorrendjét állíthatjuk be. Ha az animátor az utolsóhoz ért, akkor előről kezdi a fázisok kirakását.
Példa: ANIMATE 4,5,4,8,6,4,3,1,6

SPEEDAN <sorszám>,<sebesség> : az animáció sebességének megadása egy sprite-ra vonatkozóan. A <sebesség> értékét egy 0-tól 255-ig terjedő számmal adhatjuk meg. A leggyorsabb mozgást az 1-es érték adja, a 0 gyakorlatilag leállítja az animációt. Figyelembe kell venni, hogy az animáció sebessége erősen függ a betöltött sprite-ok számától.
Példa: SPEEDAN 4,10

ANIMON <sorszám> : elindítja az adott sprite animációját.
Példa: ANIMON 4

ANIMOFF <sorszám > : leállítja az adott sprite animációját.
Példa: ANIMOFF 4

**DIRECTION <sorszám>,<$irány>** : az adott sprite mozgásirányának megadása. A mozgás 8 irányba történhet, az <$irány> az égtájak angol neveinek kezdőbetűjét jelenti:

- "n" (north, észak) fel
- "s" (south, dél) le
- "e" (east, kelet) jobbra
- "w" (west, nyugat) balra
- "ne" jobbra fel
- "nw" balra fel
- "se" jobbra le
- "sw" balra le
- "" nincs mozgás

Példa: DIRECTION 4,"NE"

SPEEDSPR <sorszám>,<sebesség> : egy sprite sebességének megadása a mozgáshoz. A <sebesség> értéke itt is egy 0 és 255 közötti szám, az 1-es eredményezi a leglassabb mozgást, 0-ra megáll a sprite.
Példa: SPEEDSPR 4,5

POSITION <sorszám>,<xpos>,<ypos> : adott sprite adott pozícióra helyezése. A képernyő bal felső sarka az origó. A paraméterek lehetséges értékei: 0<=xpos<=71, 0<=ypos<=147.
Példa: POSITION 4,71,147

SPRON <sorszám> : adott sprite láthatóvá tétele. A parancs kiadása után a beállított jellemzőknek megfelelően megjelenik a sprite.
Példa: SPRON 4

SPROFF <sorszám> : adott sprite eltüntetése.
Példa: SPROFF 4

INIT : inicializálja a kezelőt (kiiktatja a megszakítási rutinját).
Példa: INIT

XPOS(<sorszám>) : visszaadja az adott sprite X irányú pozícióját. A visszaadott érték illeszkedik a POSITION parancsnál közölt tartományhoz.
Példa: X4=XPOS(4)

YPOS(<sorszám>) : visszaadja az adott sprite Y irányú pozícióját. A visszaadott érték illeszkedik a POSITION parancsnál közölt tartományhoz.
Példa: Y4=YPOS(4)

Az EnterSPRITE inicializációja után elsőként az előzőleg már megtervezett sprite-okat kell betölteni (SLOAD). A sprite paraméterek (ANIMATE, SPEEDAN, ANIMON, DIRECTION, SPEEDSPR, POSITION) beállítását követően a GRAPHICS HIRES 16 parancs kiadása, majd a sprite-ok bekapcsolása (SPRON) következik. Ha a kezelőt ki akarjuk kapcsolni, akkor adjuk ki az INIT parancsot, és minden más "sprite-független" műveletet csak ezután végezzünk el.

Szerencsés lett volna egy sprite definiáló utasítást is beépíteni a parancsok közé. Valószínűleg a fejlesztők azért nem tették meg ezt, mert a színek kezelése a felhasználó szempontjából nehézkes lett volna (a sprite-ot alkotó összes pontról meg kellene mondani, hogy milyen színű legyen). Ilyen formán viszont az sprite animátor a FINE_PEN nélkül nem használható. (Ügyes árukapcsolás!)

Az EnterSPRITE tehát csak a GRAPHICS HIRES 16 parancs hatására keletkező képernyőn működik helyesen. Sajnos hiába definiáljuk át a 101-es csatornát más méretűre, a program ezt nem képes lekezelni, marad hát a 40*20-as méret. A két-, négy-, kétszázötvenhatszínű képernyőkön pedig szemetet jelenít meg. Érdekes látvány az is, amikor bekapcsolt kezelő mellett 80 karakteres szöveges képernyőre váltunk át, ahol a szövegeken időnként "szöszmöszök" úsznak át.

A parancsoknál is tapasztalható némi inkorrektség. Az INIT például nem állítja alaphelyzetbe a sprite változókat, csupán a megszakítási rutint iktatja ki. Az INIT-nek ez a hiányossága előnynek is felfogható, hiszen ha egyszer beállítottuk a sprite-okat, majd valamilyen okból kikapcsoljuk a kezelőt, akkor a fáradságos munkával beállított paraméterek megőrződnek. Nagyobb baj a felhasználói megszakítási rutinnal való "garázdálkodása": ha valaki a Basic programja mellett még saját IT rutint is szeretne használni, akkor először le kell kezelnie az EnterSPRITE rutinját. Nem kellően átgondolt a sebességértékek jelentése sem: a SPEEDAN-nél az 1-es érték a leggyorsabb, ugyanez a SPEEDSPR-nél a leglassabb mozgást eredményezi. A POSITION az x és y pozíciónak a valóságban 16 bites értékeket vesz át, csak az MSB-t nem veszi figyelembe. Így aztán origónak a 256 többszörösei is megfelelnek. A POSITION <sorszám>,65351,65427 hatására például a jobb alsó sarokba áll be a sprite.

A FINE_PEN ismertetésénél már említettem, hogy az EnterSPRITE működése közben megáll az óra. Hiába adjuk ki később az INIT parancsot, az óra ekkor sem indul el, a TIME$ konstans értéket szolgáltat! Mivel az EnterSPRITE-ot leginkább játékprogramok "alá" készítették, az ilyen programokban pedig gyakran van szükség az idő mérésére, az óra kikapcsolása súlyos hibája az EnterSPRITE-nak.

Az animátor nem kellően intelligens, csak meglehetősen egyedi környezetben működik. Célszerűbb lett volna rendszerbővítőként, vagy perifériakezelőként megírni, így sokkal univerzálisabban lehetne használni. Jó lenne, ha a kezelő a főprogramtól teljesen függetlenül figyelne bizonyos előre beállítható eseményeket (pl. sprite ütközés), és azt az esemény bekövetkezésekor szoftver interrupton keresztül értesítené. Hiányzik a sprite prioritás beállíthatósága is.

Az EnterSPRITE végülis kezdetnek nem rossz, de még sokat lehet csiszolni rajta. De vajon van-e valaki, aki hajlandó erre?

-HCs-

Dőlt betűk egyszerűen

Előfordulhat, hogy megunjuk a megszokott karakterkészletet, és valami másra vágyunk, mondjuk dőlt betűket akarunk látni a képernyőn (a dőlt betűtípus neve az angol szakirodalomban italic). Semmi akadálya, hogy emberfeletti erőfeszítéssel átdefiniáljuk a karaktergenerátort, de van egy sokkal egyszerűbb módszer is. A mellékelt programmal fél perc alatt elfogadható kinézetű dőlt karaktereket kapunk. A program az összes karakter felső három pontsorát egy-egy ponttal jobbra, az alsó három sort egy-egy ponttal balra tolja; a középső három sor a helyén marad.

```
100 PROGRAM "BETUDONTO.BAS"
110 NUMERIC A,I
120 LET A=46208
130 FOR I=A TO A+3*128-1
140   POKE I,PEEK(I)/2
150 NEXT
160 FOR I=A+6*128 TO A+9*128-1
170   POKE I,MOD(2*PEEK(I),256)
180 NEXT
```

Nincs akadálya annak sem, hogy balra dőlő betűket állítsunk elő. Ehhez mindössze a POKE utasítást tartalmazó két sort kell egymással felcserélni.

Megáll az idő

A rendszeróra nem működik a magnóról olvasás és a magnóra írás közben, ezért az ilyen műveleteknél és utánuk nem érdemes az órára hagyatkozni; a "magnózás" befejeztével érdemes újra beállítani az időt.

A problémát feltehetően az okozza, hogy a magnó hangfrekvenciás jelének előállítása és visszakódolása a processzort teljesen igénybe veszi, és így ilyenkor nem lehet megengedni az órát rendes körülmények között "mozgató" megszakítást. A megoldás -az akkumulátorról állandóan működő CMOS hardverórát most nem számítva- az lett volna, hogy a magnókezelő interfész mellékesen kiszámítja, hogy mennyi ideig foglalkozott a hangelőállítással vagy megfejtéssel, és időnként (pl. blokk-közökben) helyesbítené az időt.

Jellemtelen kurzorkarakter

Az EXOS videokezelője megengedi a kurzorkarakter beállítását, így ha például Basic-ben kiadjuk a

SET CURSOR CHARACTER 95

parancsot, akkor a villogó négyzet (ASCII 142) helyett villogó aláhúzáskarakter jelzi a kurzor helyét. Ez szép és jó egészen addig, amíg nem nyomjuk le kétszer az [ALT]+[INS] kombinációt (insert üzemmód be/ki), ekkor ugyanis ismét a villogó négyzet lesz a kurzor. A hibát valószínűleg az EDITOR: okozza, amely az insert kurzor bekapcsolása előtt nem menti el a kurzorkarakter eredeti kódját.

Ismeretlen hibatípusok

A számítógép használata során az egyik legkellemetlenebb élmény, amikor váratlanul hibajelzést kapunk. Különösen bosszantó lehet azonban egy hibaüzenet, ha nem ismerjük a megfejtését.

Mit tegyünk akkor, ha a számmal jelzett hibakód nem található meg sem a Basic leírásban, sem az EXOS, az EXDOS, sem az IS-DOS kézikönyvben? Ebben a rovatban megpróbálunk megfejtést adni az ismeretlen hibajelzésekre. Kérjük mindazokat, akik ilyen "titkosított" hibajelzéssel találkoznak, mind pedig azokat, akik tudják vagy sejtik ezek megfejtését, hogy osszák meg velünk tapasztalataikat. Kezdetnek két ilyen kóbor hiba:

*** Error 185 A fájl katalógus szerinti és tényleges mérete nem egyezik meg egymással.

*** Error 187 Érvénytelen .COM fájl.

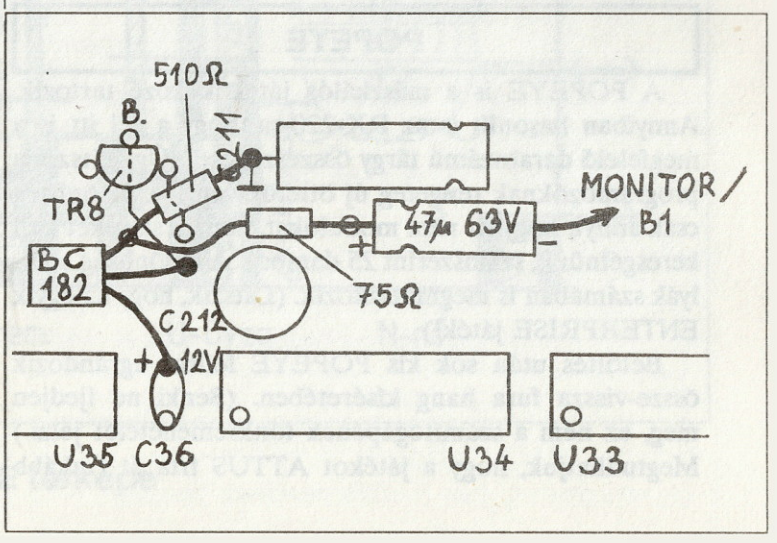
Színes videokimenet 2.

Most lapunk első számában ismertetett illesztőfokozat beépítését ismertetjük. A munkát csak az elektronikai szerelésben jártasaknak ajánljuk! Sok "hibaforrást" építhetünk be a gép szét- és összeszerelése során, vagy ha nincs megfelelő szerszámunk például a precíz forrasztáshoz.

Először minden csatlakozót eltávolítunk, kivesszük a Basic kártyát a gép bal oldalából. Kicsavarjuk az alul lévő 14 darab kereszthornyú csavart. A billentyűzetet óvatosan felbillentve távolíthatjuk el. Ekkor kivehető a funkcióbillentyűk címkéinek átlátszó tartója. A felső burkolatot fokozatosan és több helyen felfelé csúsztatjuk. Óvatosan emeljük le, mert a fóliavezeték elszakadhat! Kihúzzuk az alappanelon lévő két csatlakozóból a fóliát. A fémdobozzal árnyékolt RF modulátorhoz közel találjuk meg a *mellékelt ábrán* és a panelon a pozíciószámmal jelölt alkatrészeket. A BC182 tranzisztor kollektorát a C36-os (100nF) kondenzátor "felső" lábához (+12V), a bázist (középső láb) a TR8 bázisára menő fóliaponthoz forrasztjuk. Az emitterre csatlakozik az 510 ohmos és a 75 ohmos ellenállás. Mindegyik ellenállás kisméretű legyen, a kivezetésekre szigetelést húzva akadályozzuk meg a zárlatot! Az 510 ohmos ellenállást a C211-es elektrolit kondenzátor negatív lábához (földpont) visszük. A 75 ohmot sorbakötjük a 47 mikrofarados elektrolit kondenzátorral, annak negatív pólusát vezetjük vékony vezetékkel a monitorcsatlakozó B1-es pontjára (ez a magnócsatlakozók felőli felső aranyozott csík). A földpont a monitorcsatlakozón az A2 és a B2 érintkező, azaz alul és felül a második pont. Az összeszerelést a fóliacsatlakozók visszadugásával kezdjük. Két kézzel nagyon óvatosan, a megtörést elkerülve billegtetjük helyre a vékony fóliasávokat. Kockázatos művelet! Igazítsuk helyére a bekapcsolásjelző LED fényvezető búráját is! Finoman visszatesszük a felső burkolatot, majd a címkék műanyag tartóját. A billentyűzet következik, ez rögzíti az előbbi címketartót, majd jöhet a 14 csavar és a Basic ROM.

Ha a monitorcsatlakozót is bekötöttük, bármilyen színes TV vagy monitor videobemenetére csatlakozva megvizsgálhatjuk a képminőséget. Akár videomagnóra is rögzíthetjük számítógépünk színes ábráit.

-BG-



RX-220

Az RX-220 - ha skatulyázni szeretnénk eme játékot - a mászkálós játékok közé tartozik. Grafikusként természetesen mindig a grafikát helyezem előtérbe, így itt is meg kell hogy említsek néhány kifogást. A sprite-ok közül például a játékos által mozgatandó robot nem nyerte meg a tetszésemet. Viszont van a játékban néhány nagyon profi módon megrajzolt részlet is. Ezek közé tartozik például a téglafal.

Ennyi bevezető után térjünk rá magára a játékra! A betöltés után egy dallamos muzsika csendül fel, s közben olvashatunk a játék céljáról egy scroll segítségével: "RX 220-nak is szétszórtan elhelyezett energiamodult kell összeszednie ellenséges környezetben." (Vajon kinek kell még? Talán a szomszéd öregasszonynak? Vagy talán a csernobili reaktor egyik munkatársának?) Hááát, a program fogalmazásmódjáról ne beszéljünk! Biztosan a NOVOSOFT új találmánya szerint lehet egy darab energiamodult szétszórtan elhelyezni. Ha végigolvastuk ezt a hasznos információkat nyújtó mondatot, valamint megtudtuk azt is, hogy a játék 1989 novemberében készült, akkor már indulhatunk is. A tűzgomb illetve a [SPACE] megnyomásával indíthatjuk a játékot. Az irányítás külső vagy belső botkormánnyal, valamint az [A], [O], [P], [Q], [SPACE] billentyűkkel történik.

A játék célját már tudjuk, és a térkép segítségével már könnyen végig is játszhatjuk. De néhány jótanács mindig jól jön! Néhány modul felvételekor a térképen jelölt falak illetve a talaj megnyílhat, valamint a falak néhány helyen kapcsolók segítségével omlanak le. Természetesen ez is jelölve van a térképen.

Ha összeszedtük az összes modult, gépünk gratulál nekünk, hogy nagyon ügyesek vagyunk, mivel sikerült összeszednünk az összes modult, és visszajutottunk a földre. (Miért, eljöttünk onnan egyáltalán?) Majd ismét gratulálnak nekünk, (csak el ne teljünk magunkkal), hogy bejutottunk a legjobbak közé. Ha itt a nevünk beírása után nem nyomjuk meg az [ENTER]-t, akkor még meg is tapsol minket a számítógép. (Hát igen, aki tud, az tud!)

És végül egy kis könnyítés: aki örökélet nélkül nem tudja végigjátszani a játékot, az olvassa el két oldallal hátrébb az örökélet bevitelének módját!

POPEYE

A POPEYE is a mászkálós játékok közé tartozik. Annyiban hasonlít is az RX-220-ra, hogy a cél itt is a megfelelő darabszámú tárgy összeszedése. (Úgy látszik a programozóknak rengeteg új ötletük van.) A különbség csak annyi, hogy itt nem modulokat, hanem szíveket kell keresgélnünk, számszerint 25 darabot. A különbség a pályák számában is megmutatkozik. (Látszik, hogy az egyik ENTERPRISE játék!)

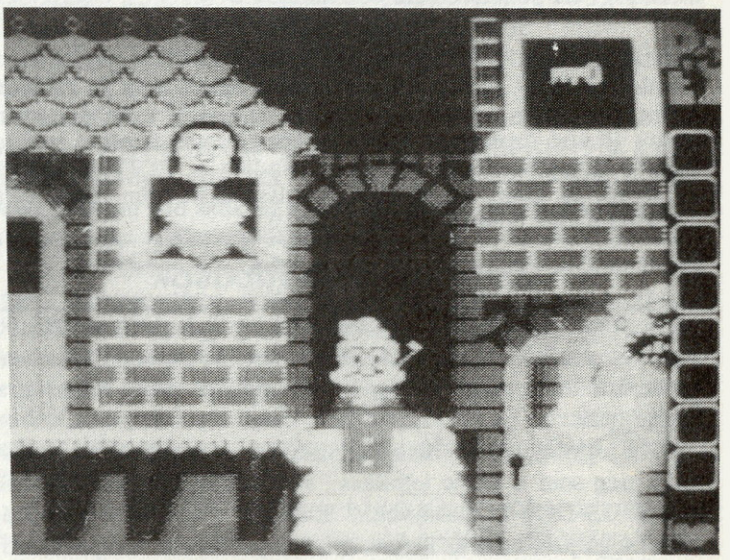
Betöltés után sok kis POPEYE felirat ugrándozik össze-vissza fura hang kíséretében. (Senki ne ijedjen meg, ez nem a számítógépének tönkremenetelét jelzi.) Megtudhatjuk, hogy a játékot ATTUS írta át (inkább nem szólok egy rossz szót sem!), valamint az irányítás módjáról is tájékoztatást kapunk: az [O], [P], [Q], [A] billentyűket használhatjuk. Aki nem akarja tönkretenni a billentyűzetét, az se ijedjen meg, hiszen POPEYE bármelyik botkormány ráncigálására reagál. A további billentyűk a következők: [H] - Hold, [SPACE] - Stop. (Aki már megszokta a lövöldözős játékokat, és nincs külső joystick-ja, az kösse hátra az egyik kezét, nehogy véletlenül megnyomja a [SPACE]-t!) A szívgyűjtési akció (kisdobos koromban csináltam ilyet, csak akkor papírral) a [SPACE] illetve a tűzgomb megnyomására indul.

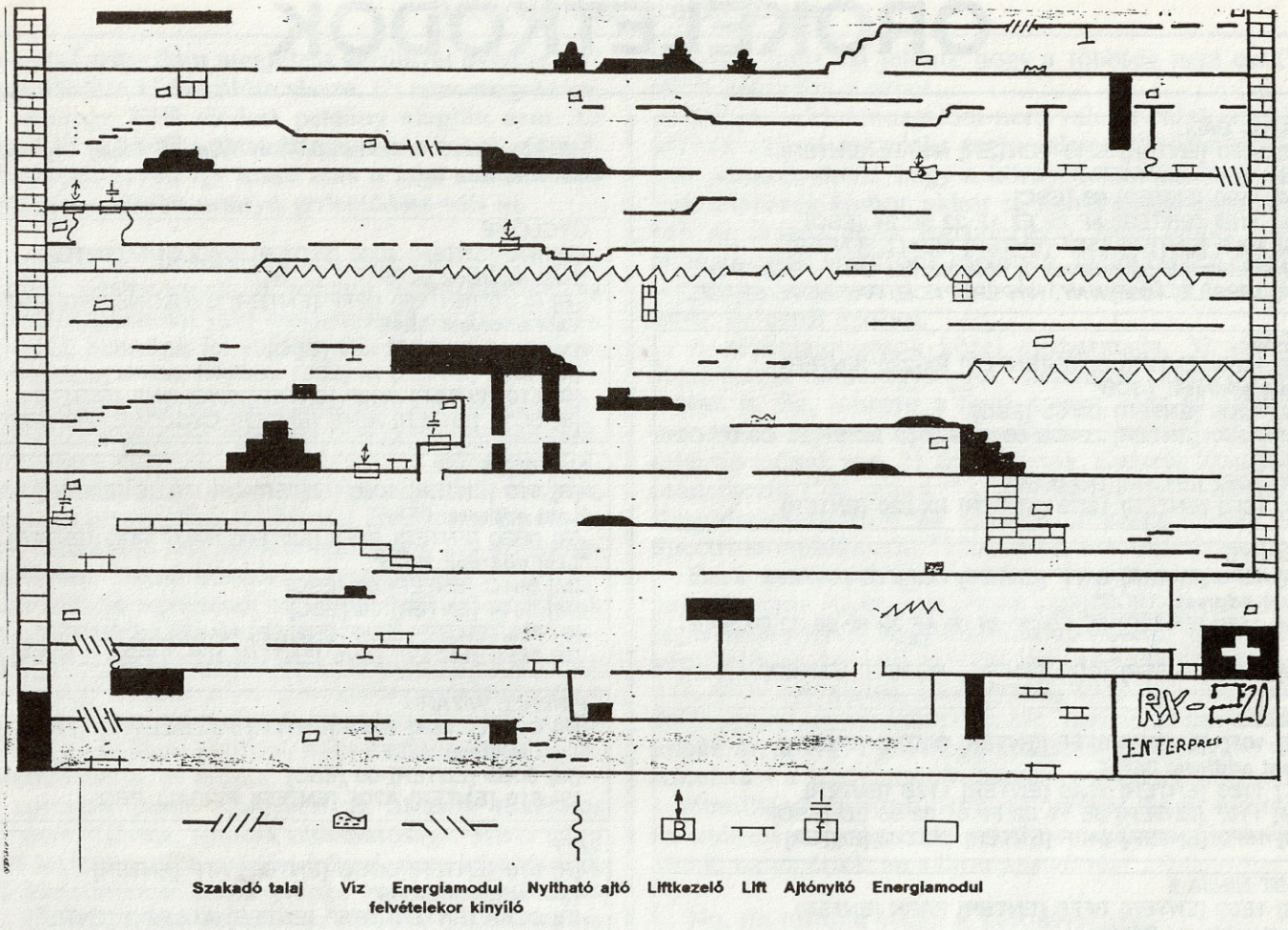
Ha megtettük a fent leírtakat, akkor máris megpillanthatjuk POPEYE-t egy lépcső tetején. Még ne csináljunk semmit! Nézzünk a képernyő jobb oldalára. (Kétbalkezesek hátrányban!) Itt láthatjuk az eltelt időt, a még összegyűjtésre váró szívek számát, és a nálunk lévő tárgyakat. Ezek a következők lehetnek: Szív (S) - a céljáról majd később, Spenót (P) - az életeink számát növeli, Kulcs (K) - az ajtók nyitásához szükségeltetik (ki gondolta volna), Érem (C) - a félkarú rabló kezeléséhez kell, Üveg (Ü) - a sárkány ellen véd (úgy tanultam, hogy sárkány ellen sárkányfű), Nő (N) - hozzá kell a szíveket vinni, és odaadni neki, melyekért óriási puszikat kapunk, ez egyébként a játék célja. Ha mind a 25 szívet odaadtuk neki, és ezzel bebizonyítottuk igaz szerelmünket, akkor beenged a házába, és furcsa mozgolódásra lehetünk figyelmesek. (A kis perverz...!)

Az utolsó hat szívet úgy szerezhetjük meg, ha a félkarú rablóval kiíratjuk a POPEYE nevet. A gép kezelése igen egyszerű: a kart ugrással ránthatjuk meg, az egyes ablakokat pedig a lépcsőn felmászva állíthatjuk meg, ha jobbra húzzuk a joystick-ot. Ha már kiraktuk a megfelelő szót, akkor a betűk szívekké alakulnak át, és ezeket felvehetjük.

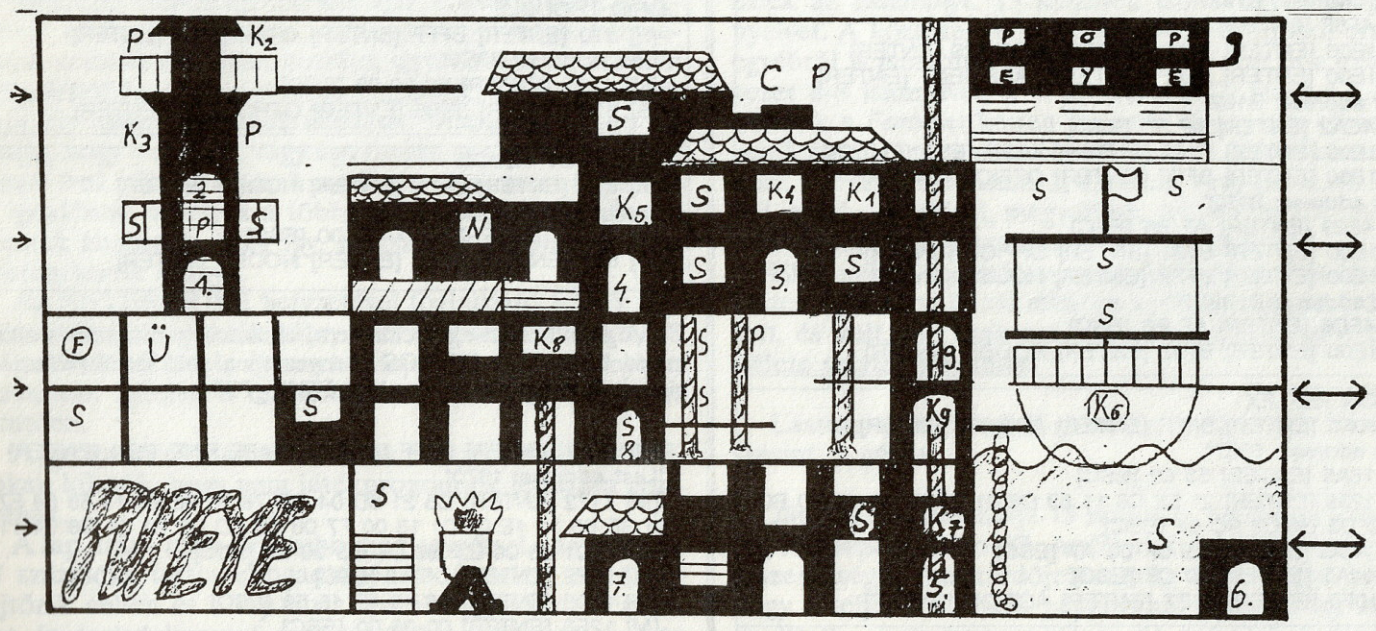
És még valami: a térképen nincsen jelölve a legfelső szinten mozgó HÉRV (Helyközi Érdekeltségű Repülő Vasút.) Ezzel közlekedhetünk ott, ahol nincs talaj a lábunk alatt.

Felhívom az olvasók figyelmét arra, hogy a következő számban egy vadonatúj ENTERPRISE játék, a MAHJONGG leírására kerül sor.





Az RX 220 térképe



A POPEYE térképe

ÖRÖKÉLETKÓDOK

MAGIC BALL
-[R] 10F0 [ENTER] BFFF [ENTER] MBALL [ENTER]
"Last address: 156F"
-[M] 1190 [ENTER] 66 [ESC]
-[M] 11E7 [ENTER] AF 32 F7 1F 32 3E 24 [ESC]
-[S] 10F0 [ENTER] 156F [ENTER] MBALL [ENTER]
A használható pályanevek a következők: LAND, SEA, ROAD, AIR, MISSILE, HIGHWAY, HYPERSPACE, RAINBOW, SPACE.

RX-220
-[R] 10F0 [ENTER] BFFF [ENTER] RX-220 [ENTER]
"Last address: 13CB"
-[M] 12D6 [ENTER] D0 03 [ESC]
-[M] 13D0 [ENTER] 22 CB 56 22 CD 56 22 CF 56 C3 06 C0 [ESC]
-[M] 10F2 [ENTER] DC [ESC]
-[S] 10F0 [ENTER] 13DB [ENTER] RX-220 [ENTER]

POPEYE
-[R] 10F0 [ENTER] BFFF [ENTER] POPEYE [ENTER]
"Last address: 12DC"
-[M] 11F9 [ENTER] 3E C9 32 38 00 AF 32 82 66 32 D7 6A 21 71 75 34 [ESC]
-[S] 10F0 [ENTER] 12DC [ENTER] POPEYE [ENTER]

DIZZY II
-[R] 10F0 [ENTER] BFFF [ENTER] DIZZY2 [ENTER]
"Last address: 34FF"
-[C] 1183 [ENTER] 11A6 [ENTER] 117B [ENTER]
-[M] 119F [ENTER] 3E 18 32 F9 61 32 66 ED [ESC]
-[S] 10F0 [ENTER] 34FF [ENTER] DIZZY2 [ENTER]

LAST NINJA II
-[R] 1E00 [ENTER] BFFF [ENTER] PARK [ENTER]
"Last address: B192"
-[M] 4EE2 [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] PARK [ENTER]
-[R] 1E00 [ENTER] BFFF [ENTER] STREET [ENTER]
"Last address: B192"
-[M] 4C99 [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] STREET [ENTER]
-[R] 1E00 [ENTER] BFFF [ENTER] SEWERS [ENTER]
"Last address: B192"
-[M] 4F8F [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] SEWERS [ENTER]
-[R] 1E00 [ENTER] BFFF [ENTER] BASEMENT [ENTER]
"Last address: B192"
-[M] 4EA2 [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] BASEMENT [ENTER]
-[R] 1E00 [ENTER] BFFF [ENTER] OFFICE [ENTER]
"Last address: B192"
-[M] 4E29 [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] OFFICE [ENTER]
-[R] 1E00 [ENTER] BFFF [ENTER] HOUSE [ENTER]
"Last address: B192"
-[M] 4FD6 [ENTER] AF B6 [ESC]
-[S] 1E00 [ENTER] B192 [ENTER] HOUSE [ENTER]

ACTION REFLEX
-[R] 10F0 [ENTER] BFFF [ENTER] ACTION [ENTER]
"Last address: 16E7"
-[M] 16A5 [ENTER] 53 02 [ESC]
-[M] 1258 [ENTER] 21 52 C6 11 53 C6 01 09 00 36 00 ED B0 3E 30 D3 B4 C3 9C 04 [ESC]
-[M] 149C [ENTER] 3E 0F D3 A7 [ESC]
-[M] 14A5 [ENTER] 50 C3 [ESC]
-[S] 10F0 [ENTER] 16E7 [ENTER] ACTION [ENTER]

DEATHWISH 3
-[R] 10F0 [ENTER] BFFF [ENTER] DEATHWIS [ENTER]
"Last address: 43C1"
-[M] 11B8 [ENTER] C8 33 [ESC]
-[M] 43C8 [ENTER] 21 DB 33 11 F3 80 01 05 00 ED B0 3E 99 32 59 81 C3 40 6C 3E 18 32 D4 9B [ESC]
-[M] 10F2 [ENTER] E0 [ESC]
-[S] 10F0 [ENTER] 43DF [ENTER] DEATHWIS [ENTER]

A következő örökéletkódokat *Németh József* budapesti olvasónk küldte be:

CYCLONE
-[R] 810 [ENTER] 1000 [ENTER] CYCLONE [ENTER]
"Last address: 0C0F"
-[R] 0C10 [ENTER] BFFF [ENTER] CYCLONE.PRG [ENTER]
"Last address: AE4E"
-[M] 441C [ENTER] xx-1 [ESC]
-[M] 43F8 [ENTER] xx [ESC] xx:életek száma
-[S] 810 [ENTER] 0C0F [ENTER] CYCLONE [ENTER]
-[S] 0C10 [ENTER] AE4E [ENTER] CYCLONE.PRG [ENTER]

NAUTILUS
-[R] 810 [ENTER] 1000 [ENTER] NAUTILUS [ENTER]
"Last address: 0FAF"
-[R] 0FB0 [ENTER] BFFF [ENTER] NAUT_PRG [ENTER]
"Last address: B075"
-[M] 581C [ENTER] 00 [ESC]
-[M] 5844 [ENTER] 00 [ESC]
-[S] 810 [ENTER] 0FAF [ENTER] NAUTILUS [ENTER]
-[S] 0FB0 [ENTER] B075 [ENTER] NAUT_PRG [ENTER]

PINBALL WIZARD
-[R] 810 [ENTER] BFFF [ENTER] PINBALL.PRG [ENTER]
"Last address: A70F"
-[M] 30B5 [ENTER] 00 [ESC]
-[S] 810 [ENTER] A70F [ENTER] PINBALL.PRG

ATF
-[R] 810 [ENTER] 0CCC [ENTER] ATF [ENTER]
"Last address: 0CA9"
-[R] 0CAA [ENTER] BFFF [ENTER] ATF.PRG [ENTER]
"Last address: B1A9"
-[M] 2F92 [ENTER] 01 [ESC]
-[M] 31D9 [ENTER] FF [ESC]
-[M] 35D4 [ENTER] 3E 03 00 [ESC]
-[M] 4579 [ENTER] 3E 03 00 [ESC]
-[S] 810 [ENTER] 0CA9 [ENTER] ATF [ENTER]
-[S] 0CAA [ENTER] B1A9 [ENTER] ATF.PRG [ENTER]

GRID TROUBLE
-[R] 810 [ENTER] BFFF [ENTER] GTROUBLE [ENTER]
"Last address: 33B6"
-[M] 1325 [ENTER] 00 00 00 [ESC]
-[S] 810 [ENTER] 33B6 [ENTER] GTROUBLE [ENTER]

NODES OF YESOD
-[R] 810 [ENTER] BFFF [ENTER] NODES [ENTER]
"Last address: AFE0"
-[M] 0A78 [ENTER] 00 00 00 00 [ESC]
-[S] 810 [ENTER] AFE0 [ENTER] NODES [ENTER]

Végül álljon itt egy példa arra, amikor egy programot alkalmassá teszünk az EXDOS környezetben való betöltődésre. A kiszemelt program az elterjedt FINE_PEN.

-[R] 10F0 [ENTER] BFFF [ENTER] TAPE:FINE_PEN [ENTER]
"Last address: 1370"
-[M] 1172 [ENTER] C5 21 6D 04 CD 78 04 3E 66 11 66 04 F7 01 3E 66 11 4E 04 01 18 00 F7 08 3E FD D3 B1 21 B8 04 11 FC 5D 01 06 00 ED B0 C1 C9 00 00 [ESC]
-[M] 11F2 [ENTER] DF 19 [ESC]
-[M] 1208 [ENTER] 46 49 4E 45 50 [ESC]
-[M] 12F3 [ENTER] 00 00 00 [ESC]
-[S] 10F0 [ENTER] 1370 [ENTER] FINE_PEN [ENTER]

A program második részét "FINEP" néven kell a lemezre másolni.

-DEVIL-

A kérdőívek eredménye

A lap első számában megjelent kérdőívet összesen 87 olvasónk töltötte ki és küldte vissza. Ez nem nagy szám, még a mintegy 2000 eladott példány alapján sem. Az olvasók 4,35 százaléka tisztelt meg bennünket véleményének kifejezésével; így kissé más a kép, sok hasonló felmérés jóval kisebb arányú érdeklődést vált ki.

Az olvasók döntő többségének megfelel a lap címe, 95 százalék találta jónak. A néhány "ellenszavazat" az Interpress Magazinnal való hasonlatosságot kifogásolja, talán joggal; bennünk fel sem merült a gyanú az esetleges tévesztésre.

A lap tartalmával kapcsolatban már inkább megoszlanak a vélemények: 82 százalék találja elfogadhatónak. Az "ellentábor" viszont igen megosztott, mindegyik olvasó mást hiányol és mást tart feleslegesnek. Sajnos, van, aki csak kategorikus NEM-mel válaszolt a kérdésre, így nem tudhatjuk, hogyan javíthatnánk a lap színvonalán a kedvére. Sokan keveslik a terjedelmet.

A lap külseje ugyancsak az olvasók 82 százalékának felel meg. A legtöbben a színes borítót hiányolják; sajnos, ez jelenleg elfogadhatatlanul megdrágítaná a lapot. Érdekes, hogy mások pont ellenkezőleg, feleslegesnek tartják a címlapot, szerintük a cím alatt azonnal "hasznos" anyaggal kellene kezdeni, mint a napilapoknál, jobban kihasználva a 16 oldalt.

Az egyetlen pont, amelyben az olvasók tökéletes egyetértésre jutottak, listáink olvashatósága. Mivel a lap a lehető legkorszerűbb módon készül, nem küszködünk rosszul kinyomtatott listák gyenge fénymásolataival, és nem is készítünk rejtvényeket az olvasók bosszantására a listák újragépelésével.

Nem mindenki szeret véget nem érő listákat begépelni: csak az olvasók 71 százaléka látna szívesen hosszabb programokat a lapban. Van, aki csak a Basic programok megjelentetése ellen berzenkedik; de hát egy valamirevaló assembly program oldalak sokaságát töltené meg...

Meglepő módon igen sokan ajánlkoztak cikket, programot küldeni az újságba. Ezt a kérdést mégis kénytelenek voltunk óvatosan kezelni, ugyanis ez sok esetben csak felelőtlen ígérgetésnek bizonyult: az önkéntesek nem kis része egy másik kérdésre válaszolva utóbb elárulja, hogy nemigen, vagy egyáltalán nem szokott programot írni... Ezért itt csak azokat a válaszokat fogadtuk el igenlőnek, amelyek a többi válasz alapján megalapozottnak látszanak. Még így, megszűrve is 55 százalékos a jelentkezők aránya.

Az ENTERPRISE helyzetével foglalkozó írások természetesen az olvasók túlnyomó többségét érdekli, 88 százalékuk válaszolt igennel. Majdnem ilyen arányban szavaznak igennel a teszteredményeket közlő írásokat érintően.

A programleírások az olvasók 95 százalékát érdekli. Sokan kikötik, hogy nem játékprogramok leírására gondolnak...

A lap árát - több-kevesebb megjegyzéssel - az olvasók 77 százaléka tartja elfogadhatónak. Érdekes kettősség rejtőzik ebben és a lap tartalmáról faggatódzó kérdésben (a terjedelemmel együtt). Sokan válaszoltak úgy, hogy az első kérdésnél a tartalommal elégedettek, de hozzátették, hogy az ár kicsit magas. Ennél a kérdésnél ugyanazok a nyilatkozók az árat elfogadják, de a lap terjedelmét keveslik.

Olvasóink több, mint fele, 55 százalék nyilatkozott úgy, hogy szokott hosszabb programokat írni. Ez igen jó arány, mert azt jelenti, hogy a többség nem csak játékra használja a gépet.

Sajnos, a klubmozgalom nem valami élénk, az olvasóknak alig 11 százaléka tagja valamelyik klubnak. Sokan panaszkodnak, hogy a környéken nincs klub, vagy nem ismernek klubot, akkor is, ha történetesen helyben van az ország egyik legismertebb, legnagyobb klubja, mint az egyik levélírónk esetében. Igény esetén elvállaljuk, hogy összehozzuk a klubokat a tagokkal és megfordítva; de erről máshol.

A géptulajdonosok közel egyharmada, 32 százalék rendelkezik lemezegységgel, néhányan kettővel vagy többel is. Ez, ismerve a honi árakat, elég jó aránynak mondható. Nyomtatója, ugyanilyen okokból, még kevesebb olvasónak van, 21 százaléknak. Egérrel 10 százalék rendelkezik ("de minek?" - kérdezi az egyik válaszoló). Beszédszintetizátora az olvasók 5 százalékának van. Spectrum-emulátorral 13 százalék rendelkezik.

Ezek az adatok igen érdekes következtetésekre adnak alkalmat. Az eladott gépek számát 20 ezerre becsülve, és feltételezve, hogy a kérdőívre küldött válaszok jól képviselik az átlagot, ez 6400 lemezegységes konfigurációt, 4400 nyomtatót, 2000 egeret, 1000 szintetizátort és 2600 Spectrum-emulátort jelent. Ennek az eszközkészletnek az értéke - jelenlegi, általában alacsonyabb áron számítva - a legóvatosabb becslés szerint is meghaladja a félmilliárd forintot! Rajtunk múlik, hogy miután a százmilliós nagyságrendű profitot zsebrevágók továbbálltak, ez az érték ne váljon szeméttéé.

No, de térjünk vissza a kérdőívekhez.

A felhasználók mindössze 5 százaléka vallja úgy, hogy semmilyen programnyelvet nem ismer. A *Basic*-et majdnem mindenki ismeri, 94 százalék nyilatkozott eképpen. Örvendetes az egyéb programnyelveket ismerők viszonylag magas aránya: 37 százalék ismeri a *Pascal*t, 51 százalék az *assembly*t. 13 százalék ismer egyéb programnyelvet. A közlékenyebb nyilatkozók jóvoltából erről az egyébről is ismerünk egyet s mást: a *Forth* és a *dBase II* vezet 8-8 ismerővel, a *C* következik 7, majd a *LISP* 4 ponttal; a *Fortran*t ketten ismerik, és egy-egy olvasó örvendeztetett meg bennünket olyan egzotikumokkal, mint a *Prolog*, a *Modula-2*, a *Fokál*, vagy az *AML/SCARA*, amely utóbbiról, megvalljuk, mi sem hallottunk (ez, mint kiderült, egy szerelőrobot-vezérlő nyelv).

Elég nehéz ezekből az adatokból bármit is leszűrni, hiszen nyilván volt, aki minden nyelvet beírt, amiről hallott, és volt, aki csak azt, amit használ és saját értékítélete szerint jól ismer.

Lássuk végül, hogy ki mire használja, saját bevallása szerint, a gépét.

Játékra (is) használja 75 százalék, programozásra pedig meglepő módon 87. Anélkül, hogy a válaszolók jóhiszeműségét megkérdőjeleznénk, feltételezni merjük, hogy a teljes húszezres gépparknál ez az arány inkább fordított... Feladatmegoldásra 64 százalék használja a gépet, és szövegszerkesztésre is 41 százalék. (Ez utóbbiak számát gyarapítjuk mi, a szerkesztőség tagjai is, hiszen az újság első három száma majdnem teljesen ENTERPRISE-on készült.) 4 százalék volt az "egyéb" kategória, amelyről - mivel a nyilatkozó nem árulta el - nem tudjuk, hogy mi is lehet.

Postafiók 334

Patek Alajos, Budapest. Kedves Patek úr! Engedje meg, hogy így, az ENTERPRISE tulajdonosok, lapunk olvasói előtt szóljunk Önhöz, hiszen Ön is mindent megtett azért, hogy az ENTERPRÉSS híre minél több géptulajdonoshoz eljusson. Bámulatos és tiszteletreméltó az az elszántság, ahogy Ön korát meghazudtolva, egyes-egyedül kiállt az ENTERPRISE-os tábor érdekében. Hihetetlen, hogy a mai világban valaki önzetlenül, nyugdíjából finanszírozva, díjtalan szolgáltatást nyújtson csupán egy ügy mellett érzett elkötelezettségétől vezérelve. Utólagos jóváhagyásával idézzük a szerényen levélnek nevezett, olvasóinak postán elküldött újságja, az *ENTERPRISE-SZOLGÁLAT* vezérelveit: *...lehetőséget nyújt érdekeink kinyilvánítására, lehetőséget nyújt egymás problémáinak megoldása terén segítség nyújtására, értelmes és hasznos programok igen mérsékelt díjakért, vagy díjtalanul történő terjesztésére. ...Nincs kötelező tagdíjféleség, nincs előfizetési díj. Aki mégis kész megosztani a felmerülő költségeket, az hozzájárulhat ehhez 100 Ft megküldésével. Akinek ez gondot okoz, vagy körülményei nem engedik meg, költséghozzájárulás nélkül is minden hónapban meg fogja kapni levelemet, annak híreit, közlöm hirdetéseit, kéréseit, ötleteit.*

Az ENTERPRISE-SZOLGÁLAT címe: Patek Alajos, 1131 Budapest, Faludi u. 28.

Szalai Ödön budapesti olvasónk már a negyedik levelét írja: *Ami tetszik: csökkent a betűméret, és így több az információ. A szerkesztőségi cikk minden megállapításával egyetértek. A sorozatokkal kezdek megbarátkozni, csak vajon hány évet kell várni, amíg egy téma végetér? Pláne, ha csak egy fél év múlva jön a folytatás... Legalább rövid gyakorló programokat közöljenek! Hol a hardver-rovat? De jó lenne tudni az EXPANSION PORT csatlakozókiosztását, látni egy buszcsatlakozó, egy RAM+EPROM bővítő, egy floppykontroller kapcsolási rajzát! Kinek van a gépről kapcsolási rajza? Pláne nyákrajz!...Végezetül közöljék a hirdetőkkel, hogy a hirdetés ár nélkül nem ér semmit!*

Megfigyelése helytálló, igyekszünk minél több hasznos információt sűríteni a rendelkezésünkre álló 16 oldalra, ami egyébként szerintünk is kevés, de rajtunk kívülálló okok miatt jelenleg nincs módunk rajta változtatni. A sorozatok hátulütője tényleg az, hogy nehéz kivárni a következő folytatást. Ugyanakkor vannak témák, amelyekről nem lehet egy-egy rövid cikkben szólni, igénylik a nagyobb terjedelmet. Reméljük azért, hogy nem kell évekig várni a sorozat befejezésére (vagy pont ellenkezőleg, olyan siker lesz, hogy az olvasók ezrei nem engedik, hogy befejezzük, újabb és újabb folytatás kiagyalására kényszerítve a szerzőket?). A folytatásra szintén nem kell fél évet várni, láthatja, hogy kéthavonta megbízhatóan megjelenünk. Megpróbáljuk a sorozatainkat élénkké, színessé tenni, hogy egy-egy rész elolvasása is önmagában hasznos legyen. Közlünk majd rövid gyakorló programokat is, bár ezzel mások nemtetszését vívjuk majd ki... Teljesen egyetértünk a kapcsolási rajzokkal kapcsolatos kívánságával. Kezünkben van a rajzok egy része, de ez még nem ok az örömre: egyrészt nem hivatalos forrásból származnak, másrészt a közlésnek jogi vonatkozásai is vannak (meg kell szereznünk a szerzői jog tulajdonosának hozzájárulását). Ezek megoldása nem könnyű, de várhatóan lesz eredmény. A hirdetőknek szóló tanácsa helytálló: ha létrejön az üzlet, előbb-utóbb úgyis "színt kell vallani", egyszerűsíti az eljárást, ha a hirdetésben az ár is szerepel. Végezetül elnézését kérjük a lassúságért, első levelére küldött válaszunk egy postai osztályozógép hibájából darabokra szaggatva visszaérkezett címünkre. Köszönjük a "ránk zúdított" sok-sok értékes észrevételt, tanácsot; amit lehet megfogadunk belőlük.

Bejó Gábor pilismaróti olvasónk által küldött örökéletkódokat, valamint ajánlatát a további ilyen kódok közlésére továbbadtuk az ezzel foglalkozó "szaklektornak".

Ifj. Szabados Sándor sződligeti olvasónk írja: *15 éves ENTERPRISE-os felhasználó vagyok. Már több, mint két éve megvan a gépem, de nem nagyon van rá egy értelmes program. Már azon voltam, hogy eladom ezt a gépet, és spórolok egy Amigára, de ebben az évben egyre több játékprogram lett "PRISE"-ra is. ... Elhatároztam, hogy írok nektek, mert az újságot nem találtam égen-földön. Ha létezik, légyszíves* (Sic!), *küldjetek egy tiszteletpéldányt. Ha már átírtátok a* Platoon-*t, akkor küldjétek el azt is utánvétellel...*

Az Amiga nem rossz gép, de örülünk, hogy az ENTERPRISE-zal is elégedett vagy végre. Sajnos, az újság első száma csak 2000 példányban jelenhetett meg, ez egyértelműen kevésnek bizonyult. Ugyanakkor az egyenetlen terítés miatt mégis visszajött egy kevés példány. Amíg futja a készletből, utánvétellel elküldjük az első és a második számot az érdeklődőknek. Tiszteletpéldány a szerzőknek jár, esetleg még azoknak, akik sokat tettek a lapért. Ki kell azonban ábrándítanunk: a más gépre készült programokat nem mi írjuk át, így nem is küldhetünk ezekből. Ugyanakkor vállaljuk majd a későbbiekben színvonalas felhasználói programok, eredeti, nem átírt játékprogramok terjesztését, amely egyértelműen a szerző szellemi terméke.

Fejes Ferenc kiskunfélegyházai olvasónk a hozzánk eljuttatott leírásból ítélve komoly, sokoldalú EPROM-égetőt készített és forgalmaz elfogadhatónak tűnő áron. Ez sok olvasónkat érdekelné, ezért javasoljuk, hogy ajánlatát a kellő tömörségűre átfogalmazva hirdetésként jelenttesse meg.

Több kedves olvasónk ajánlkozott, hogy segíteni szeretne a lap terjesztésében, mégpedig önzetlenül. Segíteni akarásukat köszönjük, a kapcsolatot levélben felvesszük velük.

A szerkesztőség a címére érkezett küldeményekből a közérdeklődésre számot tartónak ítélt leveleket, teljes egészükben vagy részleteikben, a válasszal egyetemben közli. Ehhez a levélírók hozzájárulását külön megkérni nincs módjában. A szerkesztőség fenntartja a jogot, hogy a közölt részleteken, azok eredeti jelentését meg nem változtatva, terjedelmi vagy stilisztikai változtatásokat eszközöljön. Megértésüket köszönjük.

Az ENTERPRESS előző két száma korlátozott számban még megvásárolható a kiadónál (MÁTRIX Kft. 8000 Székesfehérvár, Dózsa Gy. tér 10.), és a Műszaki Könyváruházban (Bp. VI. ker. Liszt F. tér 9.).

APRÓHIRDETÉSEK

ELADÓ: ENTERPRISE számítógép+512 KB-os bővítő+floppy vezérlő+5.25"/720 KB-os lemezegység+trafó+Spectrum emulátor+magnó+tartozékok külön-külön is! **Molnár Sándor**, 1182 Budapest, Zalatnai u. 1. Tel.: VSZM 1810-950/135 (Molnárné)

ELADÓ memóriabővítős EXDOS lemezillesztő 8500-; 3.5"-es 720 KB-os új lemezegység 7500 Ft-ért. **Bozai Gábor**, 8000 Székesfehérvár, Budai út 92. fsz. 5.

LEVELEZÉS

A géppel kapcsolatos témákban leveleznének:

Bognár Balázs, 9443 Petőháza, Bartók Béla u. 11.

Budai Attila, 2119 Pécel, Várhegy út 12.

ENTERPRISE klubok, ENTERPRISE köré szerveződött baráti társaságok, ENTERPRISE témákban levelezni kívánók címeit várjuk! A címeket lapunkban rendszeresen közöljük.

hirdetések, felhívások

HIRDETÉSFELVÉTEL

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

MÁTRIX Kft.
ENTERPRESS
8000 Székesfehérvár
Dózsa György tér 10.

Bankszámlaszám: OTP 679-022096-9

Közületi hirdetőknek kívánságra hirdetési árajánlatot küldünk.

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

TISZTELT LEENDŐ SZERZŐTÁRSAK!

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

Az érdeklődők leveleit levélcímünkre várjuk:

ENTERPRESS
1399 Budapest
Pf. 701/334