ENTERPRESS

Magazin az ENTERPRISE felhasználóknak

2020/1-2. január - április

SymbiFace 3

Teljes értékű hardver Enterprise-ra

Egy nehéz időszak közepén

Írta: Matusa István (Tutus)

A 2020-as év sajnos nem úgy kezdődött, ahogy elterveztük. Március közepén **kitört egy világjárvány,** mely a mai napig is tart. Hiába enyhült júniusra, sajnos itt van velünk és bármikor várható, hogy újra robban...

Pont akkor, mikor az **Enterprise 128-as számítógép 35. születésnapját ünnepeltük volna** (2020. május 16-án). Javában benne voltunk a szervezésben, kiküldtük a VIP meghívókat, megvolt a megfelelő helyszín stb. Külföldről többen már lefoglalták a repülőjegyeket és szállást intéztek erre az időre. Sajnos két klubtalálkozót és a jubileumi találkozónk is elmaradt a járvány miatt. A jubileumi találkozóra befizetett adományokat természetesen felhasználjuk a tervezett őszi időpontra, reméljük, hogy ekkor tudjuk pótolni ezt a nagy napot.

Az **ENTERPRESS Magazin** első, dupla száma is csak most tud megjelenni. Természetesen pótoljuk az elmaradt klubtalálkozókat is, valamint az ENTERPRESS Magazinok is megjelennek ebben az évben!

Nyilván ez a nehéz időszak a **folyamatban lévő fejlesztéseket is befolyásolta,** hiszen sokunknak kellett otthonról dolgozni úgy, hogy közben a család is otthon volt. Érthető, hogy örült az ember, ha a saját home office munkáját el tudta látni ilyen helyzetben.

Persze azért vannak jó híreink is! A **SymbiFace 3 csodakártya és hozzá tartozó csatoló kártya** (amit *Gflorez* fejlesztett) is megjelent, rendelhető. Ahogy a cikkben is olvasható, még pár drivert kell írni a kártyához, de ezek folyamatban vannak.

Közben többen teszteltük **Bruce EPNET kártyáját,** és így kiderült több hiba is. Ez is érthető, hiszen Bruce pár éve hirtelen abba kellett, hogy hagyja a fejlesztést egy rajta kívül álló dolog miatt. Dolgozunk azon, hogy az EPNET kártyához legyen SymbOS driver *(amint sikerül elkészítenem LGB-nek egy EPNET kártyát – a szerk.)*

Elindult június elején egy másik nagyon érdekes projekt is: az **Enterprise 128 megvalósítása FPGA-n keresztül.** Ennek előzménye, hogy próbáltuk felvenni a kapcsolatot **Nick Toop**-al, aki a Nick video chipet készítette Enterprise-ra. Válaszolt is, de sajnos nem tud már ezzel foglalkozni. Nagyon elkeserítette az az időszak, mikor a Nick chip terveit meg kellett semmisítenie... De ettől függetlenül jött a jó hír **ron**-tól, hogy próbálják megoldani az FPGA-s rendszert. Erről is olvashattok egy rövid cikket.

Talán észrevettétek már az előző magazinjainkban is, hogy **sok régebbi program leírást is közöltünk.** Igaz, hogy ezek nem jelentek még meg az ENTERPRESS-ben, de mi szeretnénk inkább az új EP szoftverekről és hardverekről beszámolni. Így arra kérünk mindenkit (továbbra is!), ha valami újdonsága van Enterprise-os témában, küldje el nekünk! 2015 óta egész jól haladtunk, szerencsére elég színvonalas tartalommal (szerintem).

Igaz, úgy tűnik, továbbra sem maradunk **új játékok nélkül,** újabb CPC, TVC, Spectrum játékkonverziókkal gazdagodtunk az elmúlt időszakban *Geco* és *Povi* jóvoltából.

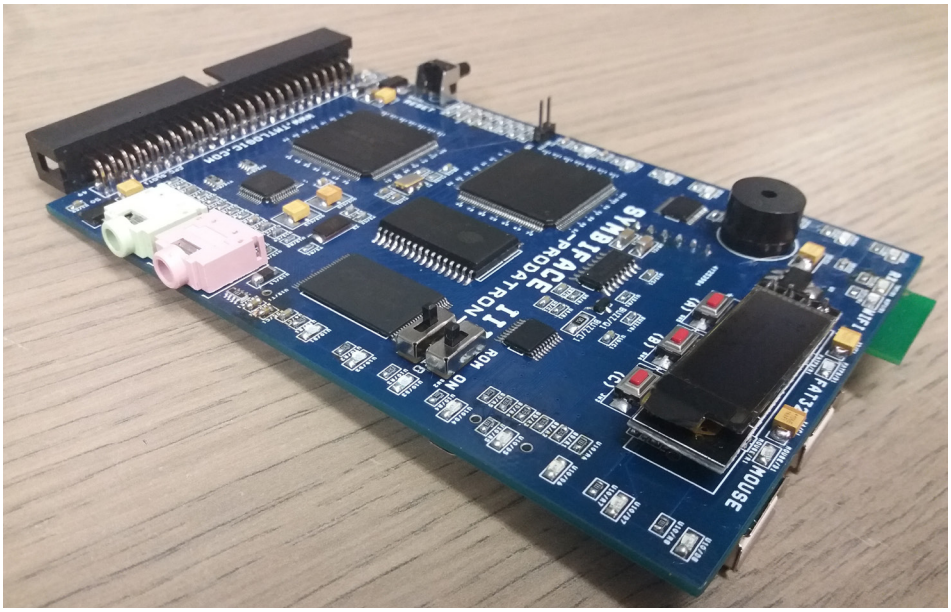
Folyamatban van az **Enterprise Klub mobil applikációjának a továbbfejlesztése** is, valamint tervezzük, hogy ne csak Android telefonokon, hanem iPhone készülékeken is használható legyen ez az alkalmazás. Vannak még **más egyéb terveink is,** melyekkel a járvány miatt ismét csúszásba kerültünk, de nem adjuk fel és nem mondunk le ezekről sem!

ENTERPRISE FOREVER!

SymbiFace 3

Teljes értékű hardver Enterprise-ra

Mi a SymbiFace3?



Az SF3 egy többfunkciós bővítőkártya, amelyet arra terveztek, hogy több számítógépre is adaptálható legyen. A SymbOS operációs rendszert megosztó négy számítógépen való használatra szánták (CPC, MSX, Enterprise és PCW). Alkotója, **Prodatron,** aki **Hans**-ot (TMTLogic) insipirálta arra, hogy a kártyát olyan közös hardverként építse fel, amely lehetővé teszi a jobb kompatibilitást a mindegyik gépre írt szoftverrel. A nevet egy klasszikus CPC IDE adapterről (SymbiFace2) örökölte, ám új felépítése és funkcióinak száma teljesen különbözik. A kifejezést a „Symbiosis" (ez Prodatron munkacsoportja) és az „Interfész" szavak alkotják.

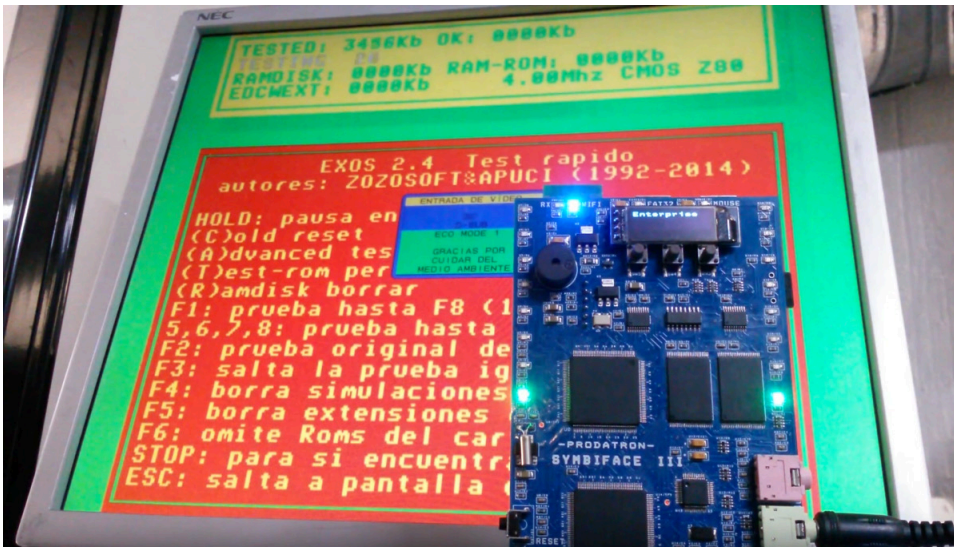
A SymbiFace3-at azonban először a CPC sorozatú számítógépekbe ültették be, tehát rendelkezik a CPC-csatlakozóval. Ezután a másik három számítógépnek valamilyen adapterre van szüksége az illesztéshez.

A probléma itt az, hogy nem mind a 4 számítógép ugyanazt a Z80 portot használja, és abszolút mindegyik saját módján használja ezeket a memória eléréséhez. Ezért további munkára van még szükség, hogy ez mind a négy gépen jól jűködjön. Ezen a versenyen a CPC teljesen adaptált és integrált állapotban van, az Enterprise és az MSX adaptált, de nem teljesen integrált, és a PCW-re csak most kezdődött meg ez a folyamat.

Milyen funkciókkal rendelkezik?

Mindenekelőtt: memória, a RAM / ROM összes kombinációja, egészen az ígéretes 4 MB-os címtartomány eléréséig. Ezt egy kötegelt fájl segítségével lehet konfigurálni, amelyet bekapcsolt állapotban nagyon gyorsan végrehajt a rendszer. Ezután az EP reset után nem veszíti el a betöltött konfigurációt.

Másodszor: zene, akár stream lejátszásával, mint például a Web rádio vagy a MIDI, vagy többféle zenefájl lejátszásával. Ez utóbbi kihasználja a klasszikus MSX MP3 cartridge-el (Sunrise) való kompatibilitását, hogy képes legyen MP3 fájlok lejátszására a SymAmpon a SymbOS-on belül. Az SF3 összes zenei része egyszerű AT parancsokkal vagy a Basic IN / OUT parancsaival vezérelhető.



Harmadszor: hatalmas tároló kapacitás, SD-kártya olvasóval és USB-csatlakozóval, minden Enterprise képes dolgozni a modern tárolórendszerekkel, miközben kompatibilis a klasszikus fejlesztésekkel, mint például az SD kártya vagy az EXDOS felület. Még meg kell írni az illesztőprogramot a rendszer

integrálásához (EXOS és SymbOS), de az összes fájlfunkció már elérhető az OUT / IN parancsok segítségével.

Negyedszer: Wifi. Csakúgy, mint az előzőnél itt is illesztőprogramra lesz szükség ahhoz, hogy integrálja azt a rendszerbe (EXOS és SymbOS), de működése már IN / OUT parancsokkal is kezelhető.

További HID USB porttal rendelkezik a különféle vezérlők csatlakoztatásához. Az USB egeret felismeri és már be van építve az Enterprise egér illesztőprogramjának utolsó verziójába. Itt még szükség van egy illesztőprogramra a SymbOS számára, de mivel a CPC felhasználói már használják a SymbOS portján, úgy gondolom, hogy Prodatron hamarosan beilleszti az EP verziójára.
Vannak más hardverek is, például egy **valós idejű óra,** egy **OLED képernyő 3 gombbal, sztereó mikrofon ki és bemenet 3,5-ös jack csatlakozóval, két oldalsó LED sor a kártya mindkét oldalán,** amelyek VU méterként működnek, és egy **FTDI port,** amely felhasználható a MIDI IN / OUT csatlakozás esetén is.
De a legfontosabb része a nagyteljesítményű **ARM processzor,** amely az összes modult kezeli, és a **CPLD,** amely az SF3 csatolására szolgál minden más számítógép számára. Ennek a rugalmas rendszernek köszönhetően folyamatosan érkeznek az újabb ötletek.
Néhány kis demo programot írtak annak bemutatására, hogy miként lehet kihasználni az SF3 előnyeit, például a SEPlayer, a Basic, TNC-EP.BAS zenefájlok lejátszásához, amelyet más SF3 felhasználókkal való csevegéshez használnak, még a többi számítógépen is, vagy a Quigs környezettel írt WebRadio SymbOS segédprogram, amely mind a négy számítógép számára lehetővé teszi az internetes rádióállomások hallgatását. De a lista hosszabb.

Mikor jelenik meg?

A kártyából van raktárkészleten, tehát rendelhető. Az egyetlen probléma az volt, hogy hogyan lehet azt az Enterprise-hez csatlakoztatni, de most sok önálló adapter van. Csatlakoztathatók közvetlenül egy Enterprise-ra vagy a buszkiterjesztő hátsó csatlakozójára.

Ha egy igazi slot adaptert szeretnénk, a Bus Expander-nél, akkor barátom, Wilco már megtervezte, és néhány napon belül néhány darab PCB-t fogok megrendelni Kínából. A vírusválság késleltette a folyamatot. Az árát körülbelül 10 euróban, nem szerelt (csak panel) verzióban 7 euróban becsültem. Érdemesebb csoportosan rendelni, hogy a szállítási költségeket csökkentsük.

Mennyibe kerül?

Az SF3 ára 110 Euro, amelyet Prodatron a TMTLogic-en keresztül értékesít (tmtlogic@gmail.com). A szállítási költség Németországból Magyarországra körülbelül 15 Euro. Ismét jobb, ha csoportosan rendeltek, a szállítási költségek csökkentése érdekében.
Igen, igen, nem olcsó, de a hosszú ideje ígért funkciókból sokat, ha nem is mindegyikét lehet használni, így a használat során szerintem megéri a költségeket.

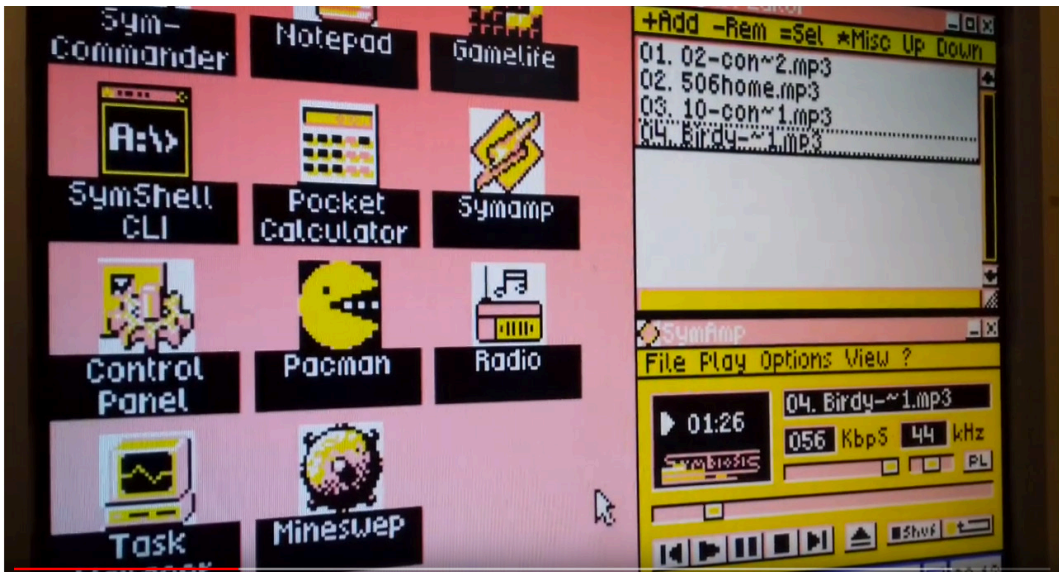
Mit kell még tennünk, hogy használhassuk?

Az SF3 csak a mellékelt DFU kábelt tartalmazza (a firmware frissítéséhez szükséges egy USB-USB-kábel), tehát ha már rendelkezik az említett Enterprise adapterrel, akkor csak a szokásos USB PEN drive-ra és egy micro SD-kártyára van szükséged. Amíg várunk az EXDOS 3.0-ra, hogy jobb formátumú legyen mind a FAT16, 1 partícióra (a FAT 16 nagy korlátozással rendelkezik 4GB-os partíciónként, így jobb a többi helyet kihasználatlanul hagyni)
Ahhoz, hogy hangot kapj az SF3-ból, szükséged lesz egy rövid Jack-Jack (3,5 mm-es sztereo) audio kábelre, amelyet az SF3 kimenetről az adapter Enterprise bemeneti hangjára csatlakoztathatsz. Varázslatosan az Enterprise közvetlenül egyesíti az SF3 hangot a belső hanggal.

Miután elkészült, csak néhány konfigurációs fájlt kell átvinned, hogy a SymbiFace3 szolgáltatásait élvezhesd.

Letöltheted az Enterprise SF3 kézikönyvet, konfigurációs és példafájlokat a következő helyről:

http://tmtlogic.com/



Hangkeltés a régi számítógépek basicjében

Írta: Bodnár Tamás (Szipucsu)

Régi számítógépeken zenét programozni legegyszerűbben a beépített basic programozási nyelven lehet. A hangchip négyszögjelét a legtöbb számítógépen a SOUND utasítással szólaltathatjuk meg, egyes gépeken a SOUND mellett vagy helyett más utasítás is rendelkezésre áll.

A SOUND utasítás
A SOUND utasításnak megadhatunk paramétereket. Így állíthatjuk be elsősorban a hang magasságát, hosszát, egyes számítógépeken a hang más tulajdonságait is, pl. hangerejét. Néhány gépen az egyes paramétereknek külön nevük is van, ilyenkor a paraméterek sorrendje tetszőleges. Más gépeken nincs külön nevük a paramétereknek, ilyenkor a paraméterek sorrendje kötött, az adott sorszámú paraméterrel lehet az adott hangtulajdonságot állítani.

Enterprise-on már a korábbi Enterpress számokból ismerhetjük a SOUND utasítás rejtelmeit. Az alap, hogy a PITCH paraméter után a hangmagasságot, a DURATION után a hosszt adhatjuk meg, de ezeken kívül is még sok paraméter megadható (LEFT, RIGHT, STYLE, SYNC, ENVELOPE, INTERRUPT):

SOUND PITCH [hangmagasság],DURATION [hossz], ...

[hangmagasság] - 0 és 127 közötti érték

Hangmagasságból az 1 jelenti a legmélyebb C hangot, és félhangonként a magasabb hangok felé haladva eggyel növelni kell a szám értékét. Így a 2 Cisz, a 3 D, a 4 Disz, az 5 E, a 6 F hangot jelöl, és így tovább. A normál C hangot a 37 jelöli. Mindig 12-t kell hozzáadni az értékhez, ha egy oktávval magasabb hangot akarunk megszólaltatni. (Így a C hangot az egyes oktávokban a következő számok jelölik: 1, 13, 25, 37, 49, 61, 73, 85.) A legmagasabb hangot 127-tel érhetjük el elméletileg, gyakorlatilag azonban nem használunk ilyen magas hangokat a zenében.
A SOUND paramétereinek sorrendje Enterprise-on tetszőleges, hiszen a paraméter neve egyértelműen utal arra.
Burkológörbét is megadhatunk az ENVELOPE NUMBER utasítással. Ennek használatát biztos mindenki ismeri, a gépkönyvben részletesen le van írva.
Egyedülálló Enterprise-on a PING parancs, mely nem paraméterezhető, és egy magas, lecsengő hangot ad ki. Hasonló, mint amikor írógéppel a sor végére értünk.

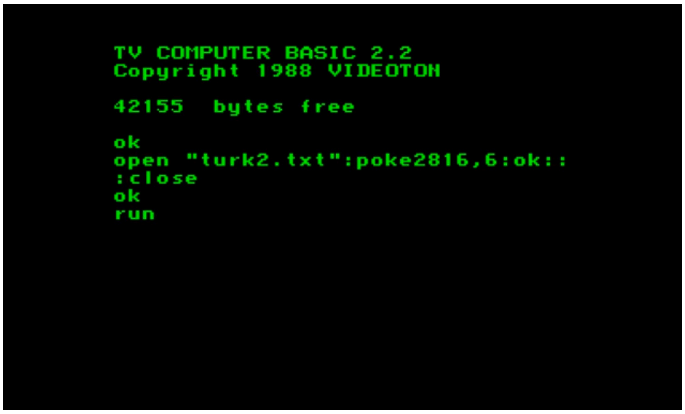
Videoton TVC-n egyetlen hangcsatornát használhatunk. Ez azt jelenti, hogy lényegében csak egyetlen egy szólamban szólaltathatunk meg zenét, egyidejűleg több hang nem szólhat egyszerre. (Azonban megoldható, hogy két hang egymást nagyon gyorsan váltva szóljon párhuzamosan, így hasonló érzetet kelt, mintha egyszerre szólnának.)
TVC-n egy sound utasítás a következőképpen néz ki:

SOUND; PITCH [hangmagasság],DURATION [hossz],-VOLUME [hangerő]

; - nincs interrupt
[hangmagasság] - 0 és 4095 közötti érték
[hangerő] - 0 és 15 közötti érték

A DURATION után 1/50 másodpercben adhatjuk meg a hang hosszát, akárcsak Enterprise-on, tehát az 50-es érték 1 másodperc hosszú hangot eredményez. (Azonban TVC-n a DURATION 1 rövidebb hangot eredményez, mint EP-n, így TVC-n ezt kihasználva alaphelyzetben változatosabb hangzások állíthatók elő egyetlen hangcsatornán.)
A PITCH paramétere az azonos szavak ellenére teljesen más, mint Enterprise-on. A legmélyebb elérhető zenei hangot a 110 adja, ez egy G hang. Megadható ennél kisebb szám is, de még a 0 sem mélyebb annyival, hogy fél

hanggal nagyobb legyen az eltérés a G-hez képest. A legmagasabb elérhető zenei hang a dokumentáció szerint 4047, ez egy H hang. (A dokumentáció pdf formátumban letölthető, annak az 57. oldalán található meg, milyen zenei hanghoz milyen PITCH érték tartozik.) Megadható ennél nagyobb szám is, azonban ilyen magas hangok esetében kevesebb frekvenciát tud a TVC hangchipje megszólaltatni, mint az alacsonyabb frekvenciáknál, és nem jelöl egyik egész érték sem olyan frekvenciát, amely adott zenei hangnak megfelelne, törteket pedig nem támogat a TVC PITCH paramétere (az Enterprise-é igen). Mély hangokból többet tud a gép megszólaltatni, mint magasból, egy mély G hangnak 110 felel meg, míg az emelletti Gisz-nek 334, tehát 224-et kell a 110-hez hozzáadni, hogy annál egy félhanggal magasabb hangot kapjunk, a többi szám köztes hangmagasságot jelöl. A 4047-tel jelölt H hangnál egy félhanggal mélyebb hangot a 4044-gyel adhatjuk meg, tehát itt már csak 3 a különbség, ami lefedi a köztes hangokat. A két félhang közötti PITCH-beli különbség fokozatosan csökken, ahogy a mély hangok felől a magas hangok felé haladunk. Bonyolult képlettel számítható ki, hogy egyes zenei hangokhoz milyen PITCH érték tartozik, így a leginkább az használható, hogy az előre elkészített táblázatból (lásd a fent linkelt dokumentációt) mindig megnézzük, milyen PITCH érték tartozik a kívánt zenei hanghoz.
TVC-n a SOUND utasítás alapból mindig megszakítja azt a hangot, amely korábban megszólalt, de még szól. (Olyan, mintha EP-n kiadnának a sound mellé az interrupt paramétert is.) Így alapból eléggé gyorsan „ledarálna" egy zenét a gép, ha sosem várná meg az előző hang végét. Hogy kikapcsoljuk ezt, meg kell adni egy pontosvesszőt a SOUND után, így az adott hang mindenképpen addig fog szólni, amekkora hosszúság meg van adva.



TVC-n a hangerőt a SOUND utasítás VOLUME paraméterével adhatjuk meg. A SOUND VOLUME 0 nem ad ki hangot, a VOLUME 1 a leghalkabb, a VOLUME 15 a leghangosabb. A SOUND VOLUME 16 nem ad ki hangot (tehát olyan, mintha 0-át adtunk volna meg), a SOUND 17 olyan, mint az 1, és ez így megy egészen 255-ig. A SOUND VOLUME n+(m*16) ugyanazt a hangerőt eredményezi, mint ha csak n lenne megadva, de csak amíg 256 alatt van az eredmény, mert a 256 már hibát okoz. (Enterprise-on a hangerőt a LEFT és a RIGHT paraméterekkel állíthatjuk, külön a bal és a jobb csatornán 0 és 255 számot megadva, 0 értékre nincs hang, 255 a leghangosabb. Enterprise-on sem adhatunk meg 255-nél nagyobb számot, és megadhatunk negatív számokat -256-ig: a -1 a leghangosabb, a -256-ra pedig nincs hang.)

A Plus 4 két hangcsatornát kezel, ezek közül az egyiken csak tiszta négyszögjel, a másikon vagy zaj, vagy tiszta négyszögjel szólaltatható meg, zenéhez inkább a tiszta négyszögjel praktikus, bár dobszerű effektekhez a zaj is használható lehet. Plus 4-en nem kell a SOUND utasítás mellett PITCH és DURATION szavakat megadni, hanem a hangmagasságot és hanghosszt jelölő számot egymástól vesszővel elválasztva adhatjuk meg, és a csatorna számát is mindig meg kell adni. Egyik paraméter sem hagyható el, sorrendjük kötött, mivel csak a sorrendjükből derül ki, mit jelölnek, hiszen PITCH és egyéb szavak nem utalnak a paraméterre:

SOUND [csatorna száma],[hangmagasság],[hossz]

[hangmagasság] - 0 és 1022 közötti érték

A csatorna száma 1, 2 vagy 3 lehet, attól függően, hogy melyik csatornát akarjuk megszólaltatni, ill. tiszta négyszögjelet vagy zajt akarunk-e hallani. Hangmagasságnak 0 és 1023 közötti szám adható meg, 0 a legmélyebb, 1022 a legmagasabb hang, 1023 pedig szintén a legmélyebb hang. A legmélyebb hang egy 2. oktávbeli A hang, ezt PAL rendszerben a 16-os szám jelöli. (Plus 4-en más hangmagasság-értékek tartoznak PAL és NTSC rendszerhez. Az egyszerűség kedvéért mi csak a PAL-lal foglalkozunk.) A 16-os A-nál egy félhanggal magasabb hangra, az Asz-ra a 73-as számmal hivatkozhatunk. A dokumentált elérhető legmagasabb zenei hang a 6. oktávbeli A hang, erre 961-gyel hivatkozhatunk. Az ennél egy félhanggal mélyebb hang száma 957. (A plus4world oldalán megtekinthető, mely zenei hangokhoz milyen értéket kell megadnunk.)
A hang hosszát az Enterprise-hoz és TVC-hez hasonlóan 1/50 másodpercekben adhatjuk meg, így az 50-es szám egy másodperc hosszúságú hangot eredményez. (NTSC rendszer esetén 50 helyett 60 jelöli az 1 másodperc hosszúságú hangot.)
A hangerőt nem a SOUND paramétereként adhatjuk meg, hanem külön utasítással állíthatjuk a globális hangerőt. A VOL után 0 és 8 közötti szám adható meg, 0 értékre csend van, 1 a leghalkabb, 8 pedig a leghangosabb hangerő. A VOL mindkét csatornára kifejti hatását, tehát nem állítható a két csatorna hangereje külön-külön.
Plus4-en tehát feltétlenül mind a 3 paramétert meg kell adni, és szigorúan a fenti sorrendben, hiszen a PITCH, DURATION, stb. szavak nem utalnak a paraméterre. Ennél több paramétere nincs ezen a gépen a SOUND utasításnak. Ehhez képest Enterprise-on és TVC-n tetszőleges sorrendben adhatjuk meg a hanghosszt, hangerőt, hangmagasságot, vagy akár el is hagyhatjuk egyik-másik paramtétert, vagy akár az összeset, mert minden paraméternek van alapértelmezése, így önmagában egy SOUND utasítás is eredményez hangot. Plus4-en ez nem lehetséges.
Videoton és Plus/4 gépen bonyolult képlettel számítható ki, hogy egy adott zenei hanghoz milyen hangmagasság-értéket kell megadnunk a SOUND utasításban.

Bár az Enterprise-on csak 0 és 127 között vannak a hangmagasság értékei, ez a gép tudja a három közül a legtöbb hangmagasságot megszólaltatni, részben azért, mert törteket is megadhatunk PITCH értéknek, részben pedig azért, mert az egész PITCH értékeket használva is a hangterjedelem 8 oktáv. Az Enterprise a PITCH értékeket felhasználóbarát módon kezeli, az egész értékek mindig konkrét zenei hangokat jelentenek. Azonban magas hang kevesebb van Enterprise-on is, mint mély, bár a PITCH értékek eloszlása látszólag egyenletes. Ennek ellenére kb. 100 fölött több egymás melletti PITCH érték is ugyanazt a hangot jelöli, és a törtek is. Így ha lefuttatunk egy FOR ciklust, mely a gép egész hangterjedelmét megszólaltatja, Enterprise-on a magas hangok tartományában hosszabb ideig fogjuk hallani ugyanazt a hangot többször is, míg TVC-n és Plus/4-en ilyenkor gyorsabban követik egymást a hangok, mivel utóbbi gépeknél a hangmagasság-értékek mind konkrét, különböző hangokat jelölnek.

Egy normál C hangot a következő utasítással tudunk megszólaltatni a három gépen egy másodpercig:

Enterprise:
SOUND PITCH 37,DURATION 50

Videoton TVC:
SOUND PITCH 3349,DURATION 50

Commodore Plus 4:
SOUND 1,600,50

A következő basic programok a Boci-boci tarka című világhírű dalt játsszák le a három gépen:

Enterprise:

```
        IS-BASIC        program    0
ok
LIST
   90 PROGRAM "boci.bas"
  100 RESTORE
  110 FOR A=1 TO 31
  120   READ G
  130   SOUND PITCH G,DURATION 10
  140 NEXT
  150 DATA 37,41,37,41,44,127,44,127,
      37,41
  160 DATA 37,41,44,127,44,127,49,48,
      46,44
  170 DATA 42,127,46,127,44,42,41,39,
      37,127
  180 DATA 37
ok
```

Videoton TVC:

```
ok
list
  100 RESTORE
  110 FOR A=1 TO 31
  120 READ G
  130 SOUND; PITCH G,DURATION 10
  140 NEXT
  150 DATA 3349,3503,3349,3503,35
98,4095,3598,4095,3349,3503
  160 DATA 3349,3503,3598,4095,35
98,4095,3723,3701,3652,3598
  170 DATA 3537,4095,3652,4095,35
98,3537,3503,3431,3349,4095
  180 DATA 3349
ok
```

Commodore Plus 4:

```
READY.
LIST

90 VOL 7
100 RESTORE
110 FOR A=1 TO 31
120 READ G
130 SOUND 1,G,10
140 NEXT
150 DATA 600,688,600,688,741,1022,741,10
22,600,688
160 DATA 600,688,741,1022,741,1022,812,8
00,772,741
170 DATA 707,1022,772,1022,741,707,688,6
46,600,1022
180 DATA 600

READY.
```

A 264-es családba tartozik a Plus/4-gyel együtt a Commodore 16 és Commodore 116 is többek között, ezek egymással kompatibilisek (megfelelő memóriabővítés esetén), a basic-jük is teljesen ugyanaz, a SOUND utasítás mindegyikben ugyanazon az elven működik.
A régi számítógépek basic-je nagyon hasonló módon kezeli a hangkeltést. Mindegyiknek az a lényege, hogy a hangot kiadó utasítás (a legtöbb esetben SOUND) után számokkal kell megadni a hangmagasságot és a hosszt, csak az egyes hangmagasságokat gépenként eltérő számok jelölik. Az eddig tárgyalt gépeknél a hossz hasonló elven adható meg, 1/50-ed másodpercben.
Az Amstrad CPC sound utasításának felépítése nagyon hasonló a Plus/4-eséhez, egymástól vesszővel elválasztva kell megadni első helyen a csatorna számát, utána a hangmagasságot, végül a hanghosszt. További paraméterek is megadhatóak itt, többek között a hangerő, de ezek a paraméterek elhagyhatók, akárcsak a hanghossz, csak az első két paraméter kötelező. Így egy minimalista sound utasítás CPC-n minimum két paraméterrel rendelkezik, pl.

SOUND 1,239

ahol csak a csatorna számát és a hangmagasságot adtuk meg, a többi paraméter alapértelmezés szerinti.
CPC-n három csatorna létezik, ezeket A, B és C betűjellel jelölik. Az A csatorna balról, a B csatorna jobbról, a C középről szól. Az A csatornához a SOUND utasításban az 1, a B csatornához a 2, a C-hez a 4-es számot kell megadni. Ha egy hangot egyszerre több csatornán is meg akarunk szólaltatni, akkor a megfelelő csatornák számjegyét (1, 2 ill. 4) össze kell adni. A 3-as szám tehát nem a C csatornát jelöli, hanem azt, hogy az 1-es és a 2-es csatorna is aktív. (Enterprise-on hasonló elven működik a joy függvény, ahol a megfelelő irányok értékeit kell összeadnunk azok kombinációjának eléréséhez.) A csatorna számához különböző értékek hozzáadásával az egyes csatornákat összehangolhatjuk, késleltethetjük, vagy az Enterprise-on ismert CLEAR SOUND utasításához hasonló hatást élhetünk el. A késleltetett hangokat a RELEASE paranccsal szólaltathatjuk meg.
Hangmagasságnak 0 és 4095 közötti szám adható meg a TVC-hez hasonlóan, azonban minél kisebb ez a szám, annál magasabb hangot jelöl, a legmélyebb hang pedig a 4095. (Ezzel szemben Enterprise-on, TVC-n és Plus/4-en is minél kisebb a szám, annál mélyebb a hang.) A hangterjedelem itt 8 oktáv. Hanghossznak 100-at megadva

kapunk 1 másodperces hangot.
Egy egy másodperces, normál C hang:

SOUND 1,239,100

A cpcpwiki oldalon egy PDF fájlban megtalálható a gép dokumentációja, melyben a 7. fejezet 24. oldalán megtalálható a táblázat, hogy melyik zenei hangnak milyen szám felel meg. Tehát, a TVC-hez és a Plus4-hez hasonlóan bonyolult képlettel számítható ki CPC-n is, hogy mely zenei hangnak milyen érték felel meg. A hanghosszt jelölő szám pont fele olyan hosszú hangot jelöl, mint Enterprise-on, TVC-n vagy Plus/4-en, ott az 50 jelöli az 1 másodperc hosszúságú hangot, míg CPC-n a 100. Ha nem adjuk meg a hanghosszt, az alapértelmezés 20 lesz (0.2 másodperc).
A SOUND első két paraméterét (csatorna és hangmagasság) CPC-n kötelező megadni. A harmadik paramétert, a hosszt nem kötelező megadni, és a további paraméterek sem kötelezőek. Összesen 7 paraméter adható meg:

SOUND [csatorna száma],[hangmagasság],[hossz],[hangerő],[volume envelope],[tone envelope],[zaj típusa]

A Plus/4-gyel párhuzamba állítható a csatornaszám, hangmagasság, hanghossz megadása. Utána sorrendben következik a hangerő, utána a hangerő burkolójának (envelope) azonosítója, a hangmagasság burkolójának azonosítója, végül a zaj típusa.
A hangerő a TVC-hez hasonlóan 0 és 15 közötti szám lehet.
Míg Enterprise-on az ENVELOPE NUMBER paranccsal definiálhatunk burkológörbét, melyben a hangmagasság és a hanghossz változásait is megadhatjuk, CPC-n külön kell definiálnunk burkológörbét a hangerőnek és a hangmagasságnak. Az előbbit az ENV paranccsal (V mint Volume), az utóbbit az ENT paranccsal (T mint Tone) hozhatjuk létre. Például az ENV 1,50,20,20 parancsban az 1. szám jelzi a burkológörbe azonosítóját (hasonlóan az ENVELOPE NUMBER esetéhez), ezt az azonosítót kell megadni a SOUND utasítás 5. paramétereként. Az ENT 2,10,-4,20 olyan burkológörbét definiál, melynél a hang lejátszása során a hangmagasság fog változni. Ennek a burkolónak a bekapcsolásához itt a SOUND utasítás 6. paraméterében 2-t kell megadni.
Az utolsó paraméterrel megadhatjuk, hogy akarunk-e zajt megszólaltatni az adott csatornán, és ha igen, akkor mi legyen a frekvenciája. A zaj paramétere 0 és 31 közötti szám lehet, 0-ra nincsen zaj, az 1 a legmélyebb, a 31 a legmagasabb hangú zajt eredményezi. A zaj hasonló, mint Enterprise-on a magas torzításnál megadott PITCH értékek, csak jóval kisebb a zaj frekvenciatartománya. CPC-n tehát egyetlen csatornán egyszerre szólhat zaj és tiszta hang is, a SOUND második paramétereként a tiszta hang frekvenciáját, a 7. paramétereként a zaj frekvenciáját adhatjuk meg, a két frekvencia tehát egymástól független. Ha az adott csatornán a tiszta hangot vagy a zajt ki akarjuk kapcsolni, akkor az adott paraméterbe nullát kell megadni, illetve a zaj paraméterét a legvégéről el is hagyhatjuk.
CPC-n a paraméterek sorrendje a SOUND parancsban kötött. Ha nem akarjuk megadni valamelyik paramétert, de a következő paramétert igen, akkor egyszerűen ki kell hagynunk, ilyenkor vesszők kerülnek egymás mellé. Ha például nem adunk meg sem a hangerőhöz, sem a hangmagassághoz burkológörbét, de zajt meg akarunk adni, akkor így nézhet ki egy SOUND utasítás:

SOUND 1,300,100,7,,,10

A CPC dokumentációjában (linket lásd feljebb) bővebben olvashatunk a SOUND utasításról és a burkológörbék működéséről az 1. fejezet 65. oldalától kezdődően, valamint a 8. fejezet 35. oldalától kezdődően.

A BEEP utasítás és egyebek
ZX Spectrumon sem túlságosan eltérő a helyzet. Ott azonban a BEEP utasítást használhatjuk hangkeltésre. A paraméterek sorrendje kötött, de eltér a Plus/4-től vagy a CPC-től, először kell megadni a hanghosszt másodpercekben, utána a hangmagasságot:

BEEP [hanghossz másodpercekben],[hangmagasság]

A 0 jelöli a normál C hangot, és félhangonként felfele haladva mindig egyet kell ehhez hozzáadni, hasonlóan az Enterprise-hoz. (Így pl. Cisz 1, D 2, Disz 3, E 4.) Azonban az Enterprise-nál az 1 a legmélyebb C hangot jelöli, és nem a normál C hangot, mint Spectrumon. Ha a normál C hangnál mélyebb hangokat akarunk, akkor negatív számokat kell megadni (pl. -1 lesz a H). Továbbá, Enterprise-on az 1-es szám jelöl C hangot (a 0 egy H hangot jelöl), Spectrumon pedig a 0 jelöl C hangot. Így, ha a BEEP-ben megadott hangmagasságot számoljuk át SOUND PITCH értékre, 1-et hozzá kell adnunk (illetve ezen kívül még 3*12-t is hozzá kell adnunk, hogy ugyanabban az oktávban legyen az adott hang).
Spectrumon egy másodperces normál C hang így fest:

BEEP 1,0

Tehát az 1 utal az egy másodpercre, a 0 pedig a C hangra. A BEEP 1,12 egy oktávval magasabb C hangot eredményez. A Worldofspectrum oldalán bővebben olvashatunk a BEEP utasítás használatáról, illetve a Sam Coupé felhasználói kézikönyve (lásd lejjebb) is elég bőven kifejti, hogy mely zenei hangok milyen BEPP értékekkel érhetők el.

Spectrum 128k-n a BEEP mellett a PLAY utasítás áll még rendelkezésre. Ezzel az utasítással sokkal könnyebben írhatunk zenét, mint a többi gépen, ahol az egyes zenei hangokhoz tartozó számokban nem mindig van sok logika. Három szöveges változót adhatunk meg a PLAY utasítás után, melyekben a három csatorna hangjait adhatjuk meg:

PLAY a$,b$,c$

A hangokra a zenében is használt nevükkel hivatkozhatunk. A hangok nevei mellett természetesen egyéb karaktereket is elhelyezhetünk, melyekkel a hang hosszát, az oktávot, a tempót és egyéb paramétereket állíthatjuk. Egy

C hang így szólaltatható meg:

PLAY „c”

A PLAY utasításról bővebben a World of Spectrum oldalán lehet olvasni, illetve a Spectrum Világ 5. számában.

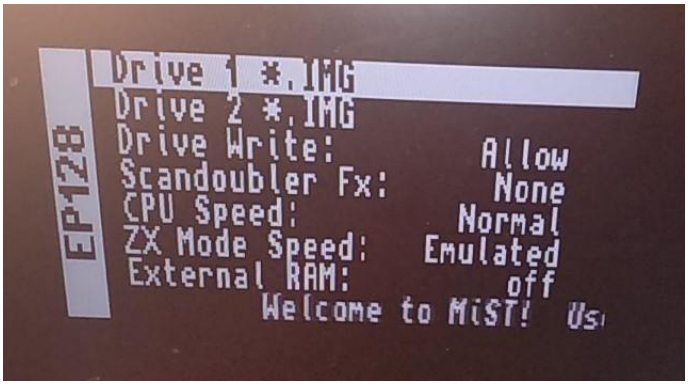
Sam Coupén is használható a BEEP, ezen kívül létezik SOUND utasítás is. Utóbbi használata kissé bonyolult, mivel közvetlenül a hangchip regisztereit piszkálhatjuk vele. Egyedülálló ezen a géptípuson a ZAP, POW, ZOOM, BOOM parancs, melyek négy különböző hangeffektet szólaltatnak meg. Ez a megoldás kissé az Enterprise PING utasítására emlékeztet. A speccy.cz oldalán megtalálható a Sam Coupé felhasználói kézikönyve, ebben a 93. oldalon kezdődik a hangkeltéssel foglalkozó fejezet.

Commodore 64-en a hangkeltéshez használható basic utasítás a poke, ez egy külön világ.

Összegzésként, a hangmagasság paraméterezésében megkülönböztethetünk felhasználóbarát és nem felhasználóbarát megoldásokat. Felhasználóbarát az Enterprise SOUND, és a ZX Spectrum ill. számos más gép BEEP utasítása. Ezeken a gépeken aránylag könnyen kiszámolható, hogy adott zenei hanghoz milyen értéket kell megadnunk, míg pl. TVC-n, Plus/4-en és CPC-n egy átlagos, zenéhez értő felhasználó számára nem mondható egyszerűnek a hangmagasság értékeinek a kiszámolása. A BEEP utasítás és az Enterprise SOUND utasítása annyiból is előnyös, hogy adott számú félhangnak megfelelő számjegy hozzáadásával egy dallam valamennyi hangmagasság-értékéhez könnyen transzponálhatjuk a dallamot. Például, 12-t hozzáadva a hangmagasságot jelölő változóhoz egy oktávval magasabb hangon játszhatjuk le a dallamot, bármilyen hangmagasságról legyen is szó. Más gépeken ez nem lehetséges.
A legemberbarátibb a Spectrum 128 PLAY utasítása, mivel maguknak a zenei hangoknak a nevét adhatjuk meg.

Az Enterprise 128 megvalósítása FPGA-n keresztül

Nemrég alakítottunk ki egy kis számú retro számítógépes csoportot, akik képesek voltak több dolgot megvalósítani, beleértve az Oric magját Microdisc-rel. Mindenesetre a csoportban vannak olyan felhasználók, akik nagy ismeretekkel rendelkeznek a Z80-ról és az Enterprise hardverről. Jelenleg meglehetősen szilárd, kitartó csoportunk van, és mindenki számára nyitva állunk, aki hozzájárulhat a dokumentációhoz és az ismeretekhez.
A fő cél az Enterprise 128 megvalósítás elérése, amely olyan FPGA táblákon működik, mint a MiST, MiSTica és SiDi, és a későbbi az FPGA-k, mint például a MiSTer.



A két legnagyobb probléma a Nick és Dave, mivel nincs pontos dokumentáció, és csak emulátorok és kisebb dokumentumok állnak rendelkezésünkre, de ezt mindannyian tudjuk már. Gflorez is a csapatban van, mivel ő az egyik legjobb Enterprise-os nagykövetünk. Gflorez lesz kapcsolatunk az Enteprise Forever fórum és a RetroWiki, valamint a Telegram csoportok között.
A projektet nem szeretnénk elkapkodni, nyilván sok időt vesz majd igénybe. Ez mindenféle feltétel és haszon nélkül történik!

Erőfeszítéseinket arra kell összpontosítanunk, hogy Nick és Dave a lehető legjobb megvalósítást kapja. Ez kulcsfontosságú, mivel a mag többi része többé-kevésbé már meg van határozva.
Az első dolog, amire szükségünk van, hogy rendelkezzünk egy Nick's Video bázissal, hogy elkezdhessük a képernyőn a megjelenítést, és fokozatosan hozzáadjuk az összes többi fejlesztést. A Dave része könnyebbnek tűnik. Reméljük, hogy érdekesnek találjátok ezt a projektet, és meghívunk téged a részvételre. Létrehozunk egy topicot spanyolul a RetroWiki-n, hogy tudjuk cserélni az ismereteket és a benyomásokat.
Ez egyelőre minden, azonnal értesítünk téged, és reméljük, hogy tetszik kezdeményezésünk!

ron, (Enterprise Forever Fórum)

És a jó hír! Az első kép mely megjelent az EP FPGA-ról

Egérkezelés a Lemmings-ben II.

Írta: Povázsay Zoltán (Povi)

A varF57B változónk egy számláló, ami nullázódik, ha épp lenyomunk egy bilentyűt, a billentyű nyomva tartott állapotában pedig minden megszakításkor eggyel növekszik az értéke, egészen 4-ig. Ha a számláló értéke kisebb, mint 4, akkor a lassan mozog a kurzor, ha a változó egyelnő 4-gyel, és nem vagyunk a játéktér szélén, akkor „begyorsul" a kurzor.

A fentiekből következik, hogy a kurzor függőleges koordinátáit az 0F56BH címen lévő változó tartalmazza. Hogy gyanúnkat beigazoljuk, lássuk a C nyelvre átírt lefelé mozgató rutint, ahol sejtésünk szerint a varF56B-nek növekednie kéne:

```
if (varF57A & 2 == 0)
{
    varF57A = 2;
    varF57B = 0;
}
else if (varF57B < 4)
{
    varF57B += 1;
}
if (varF56B < 120)
{
    varF56B += 1;
    varF56C = 1;
    if (varF57B == 4 && varF56B < 113)
    {
        varF56B += 8;
    }
}
```

A fentiek alapján gyanúnk beigazolódott, és most már azt is tudjuk, hogy a kurzor a 0 és a 120 közötti értéket veheti fel. Mivel az egérkezelő rutintól egy relatív elmozdulást kapunk, a kurzor mozgatása gyerekjátéknak tűnik (most csak a függőleges irányt nézzük):

```
CURSOR_Y += Y_REL;
if (CURSOR_Y > 120) CURSOR_Y = 120;
else if (CURSOR_Y < 0) CURSOR_Y = 0;
```

Ezek után már gyerekjáték lesz kibővíteni a HandleMouse rutinunkat! A MouseRoutine függvénytől az L regiszterben kapjuk meg a függőleges relatív elmozdulás értékét, azonban a felfele irányt itt a pozitív, a lefele irányt a negatív előjelű szám jelenti, tehát előbb negálni kell az értékét. Lássuk, hogyan folytatódik tehát a HandleMouse rutinunk a noBtn címkétől:

```
noBtn:

; check vertical movement
        ld   a, l        ; Y_REL
        neg
        jr   z, noVer    ; if (Y_REL == 0)
                         ; no vertical
                         ; movement
        ld   l, a        ; L = -Y_REL
        ld   a, (CURSOR_Y)
        jp   m, fel

lefele: add  a, l
        cp   121         ; if (CURSOR_Y >
                         ; 120) CURSOR_Y =
                         ; 120;
        jr   c, l4
        ld   a, 120
        jr   l4

fel:    add  a, l
        jp   p, l4       ; if (CURSOR_Y < 0)
        xor  a           ; CURSOR_Y = 0;
```


```
l4:     ld   (CURSOR_Y), a
noVer:
; TODO: horizontal movement
        ret
```

Az L regisztert áttöltjük az A regiszterbe, majd értékét negáljuk. Ha az eredmény nulla, akkor nem volt függőleges elmozdulás, és a program a noVer címkétől fut tovább. Ha volt elmozdulás, akkor az L regiszterbe betöltjük az immár előjelhelyes (negatív = fel, pozitív = le) függőleges relatív elmozdulás értékét, az A regiszterbe az egérkurzor Y koordinátájának értékét, majd összeadjuk őket (eredmény az A regiszterben). Ha az eredmény illegális pozicióra mutatna (felfelé, vagy lefelé kiszaladna a kurzor a képernyőről), akkor az eredményt át kell írni a szélső értékékre (0-ra, vagy 120-ra), majd végül elmentjük az A regisztert (most már az egérkurzor új értéke van benne) a CURSOR_Y változó memóriaterületére.
A vízszintes mozgás nagyon hasonló a fentiekhez, lássuk a jobbra irány esetében a C-sített kódott:

```
if (varF57C & 1 == 0)
{
    varF57C = 1;
    varF57D = 0;
}
else if (varF57D < 4)
{
    varF57D += 1;
}
if (varF56B == 248) goto 0C960H;
varF56B += 1;
varF56C = 1;
if (varF57D == 4 && varF56B < 241)
    {
        varF56B += 8;
    }
}
```

Ismerős a szerkezet: a varF57C változónk egy hasonló bitfield, mint a varF57A: itt a 0. bit jelenti a jobbra irányt, és az 1. bit a balra irányt. A varF57D egy ugyanolyan számláló, mint a függőleges irány esetében a varF57B. A kurzor vízszintes koordinátáját pedig a 0F56B címen lévő változó tárolja, két szélső értéke a 0 és a 248 (ezekben az esetekben a program futása a 0xc952, vagy a 0xc960 címeken folytatódik, ezekre később visszatérünk). Ezek alapján az már csak az alábbi kódot kell megírnunk assembly-ben:

```
CURSOR_X += X_REL;
if (CURSOR_X > 248) CURSOR_X = 248;
else if (CURSOR_X < 0) CURSOR_X = 0;
```

Ami így néz ki (a H regiszterben a relatív vízszintes elmozdulás van, negatív előjel a jobbra, pozitív előjel a balra irány):

```
noVer:

CURSOR_X:    EQU   0xf56b

; check horizontal movement
        ld   a, h        ; X_REL
        or   a
        jr   z, noHor
        jp   p, balra

jobbra: neg
        ld   c, a
        ld   a, (CURSOR_X)
        add  a, c
        jr   c, .l11          ; if (A > 255)
                              ; A = 248
        cp   249
        jr   c, l2            ; if (A > 248)
                              ; A = 248
.l11:   ld   a, 248
        jr   l2

balra:  ld   a, (CURSOR_X)
        sub  h                ; A = A - H
        jr   nc, l2           ; if (a < 0)
                              ; a = 0
        xor  a
l2:     ld   (CURSOR_X), a
noHor:
; TODO: auto scroll at horizontal edge
        ret
```

A fenti kódrészlet TODO sorából látható, hogy kimaradt még egy igen fontos feature: a pálya automata vízszintes scrollozása, akkor, amikor a kurzor a képernyő szélén van. Most térjünk vissza a fentebb már említett esetre, amikor a kód a 0xc952, vagy 0xc960 címektől fut tovább: mivel ezekre akkor ugrunk rá, amikor a kurzor a képernyő bal, vagy jobb szélén található, könnyen rájöhetünk, hogy ez valójában nem csinál mást, mint pont azt at automata scrollozást, amire nekünk most szükségünk van! Egészítsük ki tehát a HandleMouse rutinunkat a noHor címkétől:

```
noHor:

; check horizontal edge for auto scroll
        ld   a, (CURSOR_X)
        or   a
        jp   z, 0xc952
        cp   248
        jp   z, 0xc960
        ret
```

A fenti néhány soros kód beillesztésével a játékok már egérrel is irányítható. A megoldás ugyan nem tökéletes, mivel a kurzor „mozgástere" megmaradt az eredeti játékban is tapasztalttal: azaz a játéktér alá, a lemmingek cselekvését kijelölő sávba nem tudjuk vinni a kurzort, azokat funkciókat továbbra is a számbillentyűkkel (0..9) kell kiválasztani.

Játékújdonságok

Írta: Kiss László (Lacika)

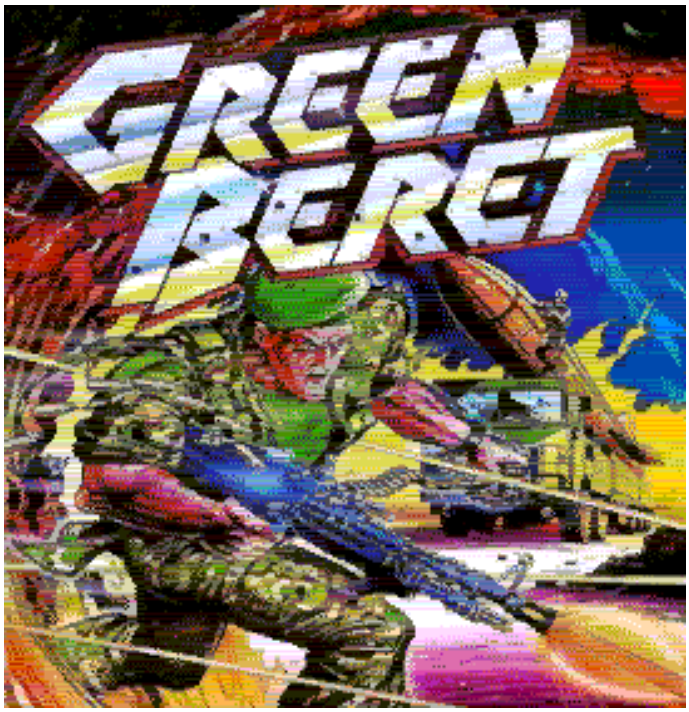
Úgy tűnik továbbra sem maradunk új játékok nélkül, újabb CPC, TVC, Spectrum játékkonverziókkal gazdagodtunk az elmúlt időszakban Geco és Povi jóvoltából.

Inside Outing

A játékban egy profi tolvajt alakítunk, akit azonban most - rendhagyó módon - eljövendő „hőstetteink" helyszínének tulajdonosa bérelt fel: a nemrég elhunyt Crutcher úr özvegye bízott meg, hogy megtalálja a megboldogult hagyatékát. Crutcher úr ugyanis nem bízott meg a bankokban, vagyonát drágakőbe fektette, amiket ósdi házukban rejtett el. Sajnos ezek rejtekhelyét feleségével sem osztotta meg... Feladatunk 12 drágakő felkutatása a ház 28 szobájában, amiket egyenként Lady Crutcher szobájába kell vinnünk és - szó szerint - a hölgy orra elé leraknunk. Ez irányú tényked éseinket a leromlott állapotú, ódon házban randalírozó elhízott patkányok és megvadult háziállatok nehezítik.

Az eseményeket izometrikus nézetében látjuk. A berendezési tárgyakkal való interakció azonban szabadabb, mint az Ultimate játékokban: A berendezési tárgyakat tolhatjuk, húzhatjuk, kinyithatjuk. Ez lehetőséget adott a készítőknek, hogy egyik-másik drágakövet egészen elmeroggyant helyekre rejtsenek.

A játék Spectrum verziója monokróm, Geco ezért inkább a CPC verziót konvertálta. Ez nem csak színes, de nagyobb képernyőméretet is használ. A végeredmény igencsak látványos!

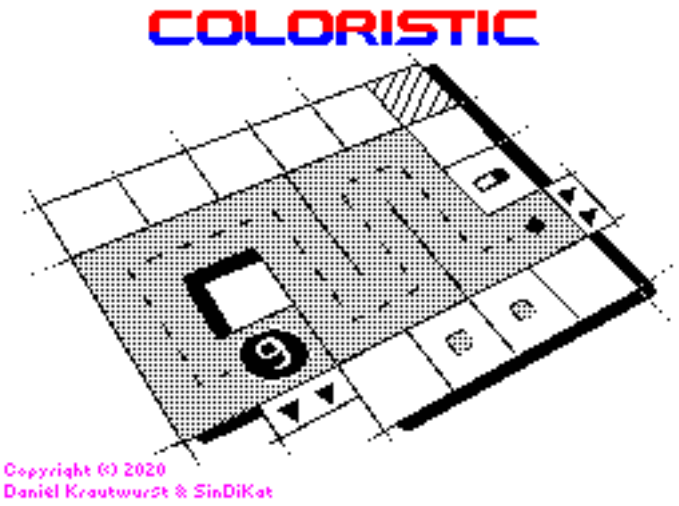


Green Beret

A Green Beret arcade játéknak a Spectrum verzió mellett MSX, C64, C16, CPC változata is borzolta a korabeli 8-bites home computer tulajdonosok lelki világát. Ezek közül talán a legértékelhetőbb látványvilág a CPC verzióban tárul a szemünk elé, így Geco ezt is konvertálta azok kedvéért, azok nem elégednek meg a régi Spectrum konverzióval.

A játék nagyvonalakban ugyanaz, azaz a cél, hogy négy szinten keresztülrohanva kiszabadítsuk a túszokat. Apróbb eltérések azért akadnak: A program eredetileg a japán piacra készült, így egyáltalán nem mondható könnyűnek, de a Spectrum verzióhoz képest még így is sokkal nehezebb! A pályák hosszúak, az ellenfelek a legrosszabbkor érkeznek. A felugorva támadó katonák nem szökellve

közelednek, így nem tudunk előre felkészülni, hogyan fognak támadni. Ez utóbbi lehet hogy bug, mert így a játék sérthetetlenség nélkül gyakorlatilag játszhatatlan. Az irányításban fölöttébb bosszantó eltérés, hogy a tűz-gombot nem lehet nyomva tartani! Egyetlen vigaszunk, hogy a játékot 5 élettel kezdjük.



Coloristic

Vadonatúj (2020-as) Spectrumra készült területkitöltős logikai játék amit Povinak köszönhetően immár Ep-n is futtathatunk. Célunk a játéktér bejárása úgy, hogy minden mezőre csak egyszer léphetünk rá. Az a mező, amire már ráléptünk, elszíneződik. A furfang az, hogy mindezt 1-4 különböző színnel kell megtennünk. Minden szín start mezőjén, egy korongon látható szám jelzi, hogy az adott színnel még hány lépést tehetünk meg. Nehezebb szinteken néhány trükkös mező tovább bonyolítja a helyzetet:
A nyilakkal jelzett mezőkre csak a megadott irányból lehet rálépni.

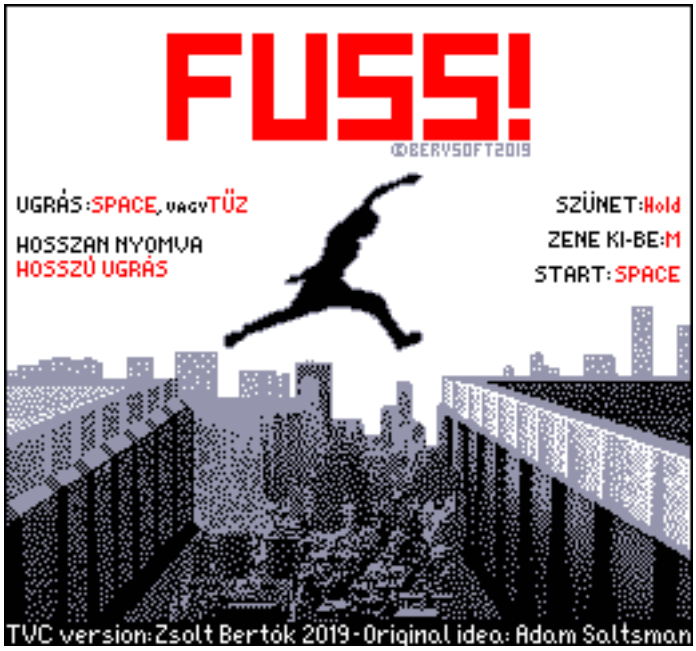
A kis színes kockákkal jelzett mezőre csak a megadott színnel lehet rálépni.

A kétállású kapcsoló a hozzá tartozó befalazott mezőt teszi szabaddá.

A +1, +2, +3 jelzésű mezőkre rálépve a még hátralévő lépésekhez hozzáadódik a mezőn jelzett lépésszám.
A fő nehézséget természetesen a bejárandó pályák szabálytalan alakja jelenti. Annyi kavarás ebben is van, hogy a pálya szélén látható szürke hátterű nyilak átléptetnek a pálya túloldalára. A játékban összesen 80 nehézségi szint (feladvány) van. Ezt szerencsére nem kell egyhuzamban lenyomnunk, minden pálya teljesítése után megkapjuk a következő szint kódját. A kódokat megadva a kívánt szinttől folytathatjuk a játékot.

Fuss!

Bertók Zsolt (Bery) TVC játéka egy régi, de mind a mai napig népszerű játékstílust képvisel: egyetlen (ugrás) billentyűvel irányítható runner játék. Konkrétan a 2009-es Canabalt c. játék TVC adaptációja, amiben talán a



legszebb TVC-s oldalirányú scroll-t láthatjuk. Ráadásul a grafika több mint funkcionális, kimondottan szép, az eredeti játékra emlékeztet, és a TVC (és Enterprise) korabeli elemek is bele lettek csempészve. A történet egy ilyen játéknál kb. mindegy, de azt is kapunk: egy járvány miatt az emberek pánikba esnek és menekülnek a városokból. Az utcák tele vannak fertőzöttekkel, az utak pedig bedugultak az autóáradattól. Egy módon juthatsz ki a városból: menj fel a tetőre és fuss! Még szerencse, hogy Superman-t, pláne Mike Powell-t, Javier Sotomayor-t megszégyenítően nagyokat tudsz ugrani...

Ugrani a SPACE vagy a botkormány tűz gombjával lehet. Ha nyomva tartjuk, nagyobbat ugrunk. Egyetlen életünk van, ha leesünk valahonnan (értendő ez alatt az az eset is, ha a hídszerkezet nem vízszintes részére sétálunk), vége a játéknak és a program kiírja a megtett távolságot méterben. A városhatár 1888 méterre van a lakóhelyünktől...
A játékot Geco konvertálta. A program EXOS kompatibilis, fut EP64-en is megfelelő sebességgel (egy picit gyorsítani kellett a háttér kirajzoló rutinon), némi időzítési problémával.

Donkey Kong Junior

Donkey Kong először 1981-ben az azonos nevű játékban tűnt fel. Ott Mario (akkori nevén még Jumpman) a főszereplő, aki egy bajba jutott hölgyet (később a Pauline nevet kapta) szabadított ki. A majom a következő évben újra megjelent a folytatásban, a Donkey Kong Juniorban, ahol Donkey Kong-ot ejti foglyul és zárja egy ketrecben az immár Mario nevű szereplő. Donkey Kong Junior vállalkozik a feladatra, és az ő bőrébe bújhat a játékos, hogy megmentse Donkey Kongot. A játékból a számtalan port között kvarcjáték is készült, aminek ugyan vajmi kevés köze volt az eredeti arcade játékhoz, de a szereplők tökéletesen felismerhetők benne. Kiss Károly készített Videoton TV Computerre egy nagyon szép kvarcjáték „remake-et”, amit Geco konvertált át pár extrával.

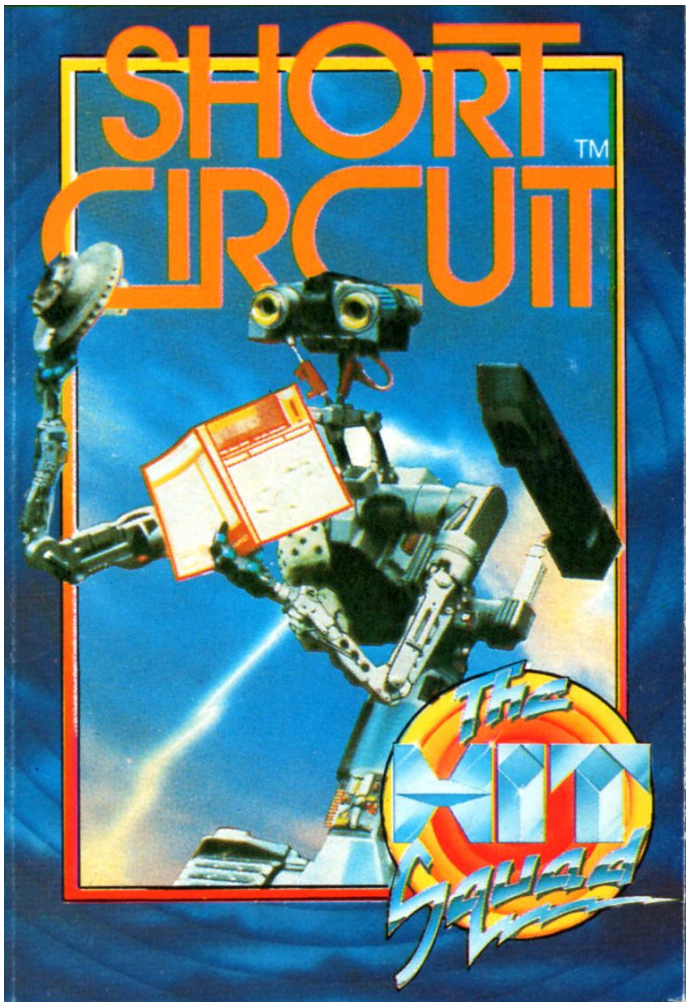

```
MARIO FOGSÁGBA EJTETTE
DONKEY KONGOT!

Az a feladatod, hogy kicselezd
a csapdákat és juss fel a ket-
recekhez a madarakat kikerülve
utána négyszer kapd el a lengő
kulcsot, ettől Donkey Kong ki-
szabadul Mario fogságából.
Irányítás:          Joystick
Ugrás:              Space/Tűz
Hang ki/be:         S
Nappal/Éjszaka:     F5/F6
Kulcs elkapása:     Balra+Ugrás
```

A játék természetesen egyetlen képernyőn zajlik. Főhősünkkel el kell jutnunk, Kong ketrecéig, miközben kicselezzük az alsó emeleten repkedő madarakat, és Mario által eleresztett csapdákat (amik inkább harapós fogprotézisnek néznek ki). A tűz megnyomásával felugorhatunk az indákra, így elkerülve a száguldozó csapdákat, és a felső platformra is a jobb oldali indára felkapaszkodva mászhatunk fel. A ketrec előtt, egy jól időzített ugrással (tűz + balra) elkaphatjuk a lengő kulcsot, amikor az ketrechez legközelebb van. Ha nem jó időben ugrunk a kulcsért, akkor lezuhanunk. Ezt a produkciót négyszer kell előadnunk, hogy Kong kiszabaduljon, ekkor a jutalompontok felmarkolása után kezdődik az egész elölről. A fán lógó kókuszdiót leszakítva lezuhan, ha eltalál egy protézist és / vagy madarat, jutalom pontot kapunk.

CrossFire

Egy újabb keletű TVS-s lövölde. A játékmenet némi hasonlóságot mutat a Galaxian játékokkal, PacMan elemekkel fűszerezve...(?!?) Bizonyára mindenki megelégedésére a rácsos felosztású játéktéren az ellenfelek nem csak felülről támadnak, hanem a képernyő mind a négy széléről. Ráadásul, ha a közelünkbe érnek, célirányosan üldözőbe vesznek! Miközben a túlélésért küzdünk, a játéktéren megjelenő extrák felvételével is gyarapíthatjuk pontszámunkat.



Short Circuit

Az Ocean játéka az 1986-os azonos című film alapján készült. A történet főhőse egy lézerfegyverrel felszerelt katonai robot, ami villámcsapás következtében meghibásodik. A hibajelenség több mint furcsa: a gép „életre kel" és megszökik a Nova Robotics cégtől. Az alaptörténet ellenére nem egy Terminator jellegű filmet kell elképzelni, hanem egy vígjátékot.
A játék két részre oszlik. Az első részben el kell szökni a Nova komplexumból. Mielőtt azonban távozhatnánk a kijáraton, meg kell szereznie három kiegészítő modult. Ez egy „tárgyhasználós, berendezés átkutatós, számítógép terminál hackelős", kalandjáték elemekkel tarkított mutatvány lesz. Szerencsére a komplexum teljesen néptelen, a rejtvény jelenti a bonyodalmakat.
A második rész egyfajta ügyességi „jutalomjáték" (bár annak azért elég nehéz...). Folyamatosan jobbra haladva kell elmenekülni immár a szabadban. A minket üldözőket lézersugarakkal megbéníthatjuk, vagy esetleg átugorhatjuk és kerülgetjük az akadályokat.

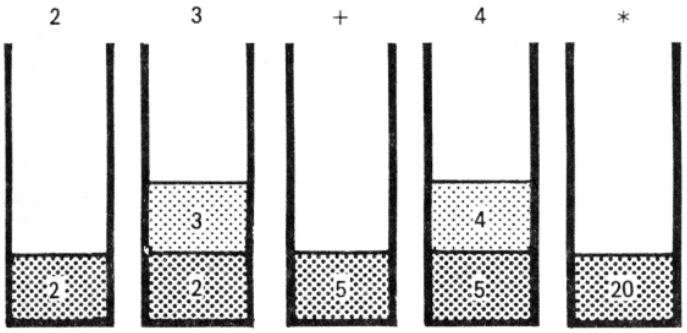
ENTERPRISE pólók!

IS-FORTH - 2. rész

Intelligent Software - 1985
rendszerbővítő, FORTH programozási nyelv

1.3. Számolni tanulunk - kicsit szokatlan lesz...

Miből áll a FORTH aritmetika? Természetesen FORTH szavakból. Ezek nevei olyan rövidek, hogy a naivabbak műveleti jelnek is vélhetik. A négy alapművelet: + , - , * , / . Mindegyik a vermen várja a két számot, amelyeken a műveletet végzi (azaz a művelet két operandusát); ezeket le is emeli a veremről és helyükbe a művelet eredménye kerül. Erre a következő ábrán láthatunk példát.



(Az adott lépés után a veremben levő adatokat rajzoltuk meg.)
A

2 3 + 4 *

sorozat ugyanazt a számítást végzi, mint a (mondjuk) BASIC nyelven írt (2+3)*4.
Az utóbbi, megszokottabb jelölésmódot infixnek nevezzük, szemben a FORTH (többeket visszariasztó) postfix jelölésével. Az elnevezések azt tükrözik, hogy a műveleti jel az infix írásmódban a két operandus között (in) van, a pontfixben pedig az operandusok után (post). A postfix megszokásához mankóul szolgálhat a következő:

Az operandusok sorrendje a postfix írásmódban ugyanaz, mint az infixben, csak a műveleti jel helye változik.

infix postfix
1 + 1 1 1 +
2 - 4 2 4 -
6 / 3 6 3 /
(10+2)*2 10 2 + 2 *
3+2*4 2 4 * 3 +

Ez azt jelenti, hogy pl. kivonásnál a szó a kivonandót várja a verem tetején, alatta pedig a kisebbítendőt. A művelet elvégzése után a veremben a különbség lesz. A tevékenységet a következő ábra szemlélteti:

Ezt a FORTH programoknál így szokás dokumentálni:

(kisebbítendő kivonandó - - - - különbség)

Zárójelet azért szoktunk írni, mert így az egyes szavak hatása a veremre a FORTH forrásszövegben is feltüntethető.
A (ugyanis FORTH alapszó. Működése: a záró zárójelig „takarja" a szöveget, amelyet az interpreternek adunk; így a nyitó és záró zárójel közötti részt az interpreter el sem olvassa, nemhogy végrehajtaná. Tessék kipróbálni! A nyitó és záró zárójelnek egy sorban kell lennie. Így lehet FORTH-ban dokumentálni.
A veremhatás jelölésének sémája:

(előtte - - - - utána)

Ha az elemek sorrendjét nézzük, egyszerűen úgy kell képzelni, mintha a vermet jobbra döntenénk.
A négy alapművelet veremhatása:

+ (összeadandó1 összeadandó2 - - - - összeg)
- (kisebbítendő kivonandó - - - - különbség)
* (szorzandó1 szorzandó2 - - - - szorzat)
/ (osztandó osztó - - - - hányados)

A számolást segíti a következő néhány, az eddigiek alapján könnyen megérthető szó:
MIN (n1 n2 - - - - min) A két szám közül a kisebbet adja.
MAX (n1 n2 - - - - max) A két szám közül a nagyobbat adja.
MOD (n1 n2 - - - - mar) Az n1/n2 osztás maradékát adja.
/MOD (n1 n2 - - - - mar hany) Az n1/n2 osztás maradékát és hányadosát is megkapjuk.
ABS (n - - - - n1) Az n abszolút értékét adja.
NEGATE (n - - - - n1) A kapott szám -1-szeresét adja.

A Forth-nak egész számokkal dolgozó aritmetikája van. Ezért az osztást különleges figyelemmel kell kezelni. A példa kedvéért, ha a hetet elosztjuk hárommal, az eredmény kettő lesz. A pontos válasz persze az lenne, hogy a 7-ben a 3 kétszer van meg , a maradék pedig egy. A /MOD operátor megoldja az említett problémát. Eredményként itt megkapjuk a hányadost és a maradékot is a veremben.
Példa:

```
7 3 /MOD . .
```

A veremből kapott két szám a 2 és 1 lesz.
Ezzel szemben a MOD operátor csak a maradék osztását teszi a verembe.

A következő példaprogram hőmérsékletet számol át Fahrenheit-ből Clesius fokba:

```
: DEGCON CR 32 - 5 * 9 / . ;
```

Tegyük fel, hogy 144 Fahrenheit fokot akarunk átszámolni, Ehhez a következőt kell beírnunk: 144 DEGCON

Véletlenszám-generátor
Gyakran használatos játékoknál, demonstrációknál. Igen könnyű a használata. Véletlenszám generálható 0 és 65536 között. Ha például 0 és 100 között akarunk véletlenszámot generálni és azt akarjuk, hogy ez a verembe kerüljön, a következő utasítást kell adni:

```
101 RND
```

látható, hogy a verembe előzőleg egy számot kell tölteni. Ez a szám eggyel nagyobb kell, hogy legyen a kívánt felső határnál.

Példa: Dobókocka szimuláció
Írjunk egy olyan programot, ami egy dobókockát szimulál, Ennek kapcsán néhány új szóra is szükségünk lesz. A KEY megállítja a program futását, és csak akkor folytatja, ha a felhasználó leüt egy billentyűt. A BEGIN...REPEAT végtelen ciklus (később részletesen beszélünk róla), addig fut amíg le nem állítjuk a STOP billentyűvel.

```
: DICE BEGIN 6 RND 1+ . CR KEY REPEAT ;
```

Amikor a DICE végrehajtásra kerül, egy 0 és 5 közötti véletlenszám generálódik. Miután nekünk 1 és 6 közti számokra van szükségünk, hozzáadunk egyet a generált számhoz. Ezután kiíratjuk ezt, majd a CR hatására a kurzor egy sort lejjebb megy. Ekkor a program mindaddig fel van függesztve, amíg egy billentyűt le nem ütünk. A program mindaddig újraindul, amíg a STOP-ot le nem ütjük.

Számrendszerek
A FORTH alapértelmezésben 10-es számrendszerben dolgozik, de bármikor megváltoztathatjuk a használt számrendszert:

```
BINARY - Kettes (Bináris) számrendszer.
OCTAL - Nyolcas (oktális) számrendszer.
DECIMAL - Tízes (decimális) számrendszer
HEX - 16-os (hexadecimális) számrendszer
```

A fenti szavak a veremre semmilyen hatást nem gyakorolnak, a BASE rendszerváltozóba töltik a megfelelő értéket (az OCTAL szó pl. 8-at). (Lásd 3.4. fejezet) Ha átállítjuk a használt számrendszert, a veremben tárolt értékek azonnal az új számrendszerben használhatóak a továbbiakban.
A FORTH az összes aritmetikai műveletet bináris formában végzi. A bemeneti számokat binárissá alakítja, így számolja ki a művelet eredményét, majd a kiíratás előtt a számokat visszaalakítja a megfelelő számrendszerbe.
Példa: hogyan konvertáljuk az 1234 decimális számot hexadecimális számmá?

```
DECIMAL 1234 HEX . DECIMAL
```

A válasz 4D2. Megjegyezzük, a DECIMAL a sor elején azt jelenti, hogy biztosak legyünk abban, hogy tízes számrendszerben adjuk be az 1234-et. A végén kiadott DECIMAL pedig biztosítja, hogy a parancs végrehajtása után továbbra is tízes számrendszerben dolgozhassunk
Hogyan alakítunk bináris számod decimálissá?

```
BINARY 010001001010 DECIMAL .
```

A válasz 1098.
A BASE rendszerváltozó direkt állításával tetszőleges egyéb számrendszert is használhatunk:

```
DECIMAL 7 BASE !
```

Ez a gépet átállítja hetes számrendszerbe. Látszólag valószínűtlennek tűnik a 32-es számrendszer használata. A hasznossága ennek a számrendszernek abban áll, hogy az alfanumerikus karakterek számként tárolhatók, így szavakat tudunk számok formájában letárolni, összehasonlítani és egyéb műveleteket végezni velük.

1.4. Számábrázolás

Előjeles egyszeres pontosságú számok: A Forth-ban a számok általában 16 bites bináris számként vannak kezelve (egyszeres pontosság). Ebből az következik, hogy a számok -32768 és +32767 közöttiek lehetnek. Az eddigi példáinkban csak ilyen számok szerepeltek A „ . „ operátor a számokat ebben a formábaban értelmezi és olyan számrendszerré alakítva írja ki, amiben éppen dolgozunk.
Előjel nélküli egyszeres pontosságú számok: Ha csak pozitív számokat használunk, akkor lehetőség var a számok 0 és 65535 közti ábrázolására. Az ilyen számok kiíratására egy másik előjel nélküli kiíró utasítást kell használni, ami a következő:

```
U. (n - - - - )
```

Dupla pontosságú számok: Ha nagyobb számokra van szükségünk, dupla pontossági számokat kell használnunk. Ezek 32 bites mennyiségként tárolódnak és az előjeles típus -2147483648 és +2147483647 közt lehet, az előjel nélküli pedig 0 és 4294967295 közt. A legtöbb dupla pontosságú művelet „D” előjellel van ellátva Az alábbiakban megadjuk néhány gyakran használt funkció dupla pontossági változatát:

egyszeres pontosság:	dupla pontosság:
.	D.
+	D+
-	D-
*	D*
U.	DU.

Ahhoz, hogy egy dupla pontosságú számot tegyünk a verembe, csak annyit kell tenni, hogy egy tizedespontot teszünk a szám végére. Ha Forth-ban bármikor tizedespont szerepel a számban, ez azt jelenti, hogy dupla pontosságúnak kell azt értelmezni.
Példaképp nézzük a következőt:

```
1234. 2000. D+ D.
```

Az eredmény: 3234
Helytelen eredményt ad viszont a

```
66000. .
```

mert a „.” szó csak a szám felső két byte-ját veszik ki a veremből, a másik kettőt „othagyja”.
A dupla pontosságért sebességbeli hátránnyal kell megfizetni.
Az előjeltelen dupla pontosságú számokat a következő utasítással tudjuk kiíratni:

```
DU. ( nn - - - - )
```

Hogyan konvertálunk 16 bites számokat 32 bitessé?
Egyszeres pontosságú számokat dupla pontosságúvá a következő jel segítségével alakíthatjuk át:

```
S->D (n - - - - nn)
```

Ha együtt használunk dupla és egyszeres pontosságú számokat, akkor ez gyakran használatos. Például:

```
1234 S->D D.
```

1.5. Összehasonlító és logikai műveletek

Hogyan hasonlítunk össze a FORTH-ban két számot? Természetesen az az első, hogy a veremre tesszük őket (így, mivel az operandusokat adjuk meg először, az összehasonlító műveletek írásmódja is postfix). Utána behívjuk valamelyik összehasonlító műveletet. Ezek a következők: < , > , =. Használatukhoz nem árt észben tartani, hogy:
az operandusok sorrendje a postfix írásmódban ugyanaz, mint az infixben, csak a műveleti jel helye változik.

A

```
2 3 <
```

művelet eredménye például az lesz, hogy a < egy igaz értékű jelzőt tesz a veremre.

A jelző

A jelző, angolul flag (ejtsd: fleg) arra való, hogy valaminek az igaz vagy hamis voltát jelezze. Ennek a két lehetőségnek az ábrázolására általában - így a FORTH-ban is - számokat használunk. FORTH-ban:

megállapodás szerint a jelző hamis, ha értéke 0, és igaz, ha bármi más.

Az összehasonlító műveletek „jól nevelt” jelzőket szolgáltatnak, amelyek értéke 0 vagy -1.

A TRUE szó hatására -1 kerül a verem tetejére.
A FALSE szó hatására 0 kerül a verem tetejére.

Írjunk egy szót amely arról tudósít, hogy mit gondol a klaviatúránál ülő felhasználó! A tudósítás a veremre tett jelzővel történik. A felhasználó lelkivilágában pedig a következő kérdéssel mélyedünk el: IGEN VAGY NEM?
Most várunk, amíg megnyomja valamelyik billentyűt. A vermen akkor adunk igaz értéket, ha a nagy I betűt nyomta le.

Ehhez meg kell tanulnunk azt a szót, amelyik kivárja, hogy valamelyik billentyűt megnyomják a billentyűzeten, s a billentyűnek megfelelő kódot a veremre teszi. Ez a szó a KEY (- - - - kód) (A KEY (ejtsd: kí) angol szó több dolgot is jelent, itt valószínűleg a „billentyű” fordítás a legtalálóbb.) A

```
KEY
```

után minden megáll, amíg meg nem nyomunk egy billentyűt. A képernyőn nem látjuk, mit nyomtunk meg (nem írja vissza, mint máskor), csak, hogy az interpreter OK-t küld. A karakterkód a veremben van - EMIT-tel kiírathatjuk a karaktert vagy .-tal (pont) a kódját..
Kicsit kényelmesebb, ha az ember látja is, hogy mit ír. Íme egy program, amely a KEY-hez hasonlóan bevárja egy billentyű lenyomását és a veremre teszi a megfelelő kódot, sőt még ki is írja a karaktert a képernyőre:

```
: ECHO KEY DUP EMIT ;
```

Ezek után az igen-nem program (figyelembe véve, hogy az I betű kódja 73) a következő:

```
: IVN CR ." IGEN,vagy NEM?" ECHO 73 = ;
```

Az eddig látott adattípusok
Két, már ismert szó:
. (szám - - - -) kiírja a vermen talált számot a képernyőre;
EMIT (kód - - - -) kiírja a vermen megadott karakterkódnak megfelelő karaktert a képernyőre.

Mindkettő egy elemet használ a veremről. A verem egy eleme egy 16 bites gépi szó. (Gépi szó: 16 jegyű, 2-es számrendszerbeli - azaz bináris - szám, másképpen: egy 16 elemű, 0 és 1 értékeket tartalmazó sorozat.) A . ezen egy 16 bites, előjeles számot feltételez (látni fogjuk, hogyan lehet ennél hosszabb számokkal dolgozni), az EMIT pedig egy karakterkódot, amely egyébként elférne 1 byte-on (8 biten). Az EMIT a 2 byte-ból álló gépi szónak az egyik byte-ját egyszerűen figyelmen kívül hagyja!

Adott esetben például a vermen 42 hever (binárisan; ugyanis a veremben csak gépi ábrázolású számok vannak). Honnan lehet tudni, hogy ez „melyik” 42: előjeles szám, a karakter kódja, vagy - már tudjuk, ez is lehetséges - egy „igaz” jelző?

A FORTH-ban az adatok típusa csak attól függ, hogy milyen műveletet végzünk rajtuk.

A 42 tehát karakterkód, ha az EMIT használja, és előjeles szám, ha a . (pont)

```
42 EMIT * OK
42 . 42 OK
```

A + például előjeles számnak tekinti a verem felső két elemét. Ha valaki mégis a KEY-vel kapott karakterkódot felejti ott, az vessen magára.
A szó veremhatásának leírásakor az elemeket jelölő betűk az elemek típusát is közlik. Az eddig látott típusok:

c	karakterkód (character),
n	16 bites, előjeles szám (number),
f	jelző (flag).

Így dokumentáljuk az összehasonlító műveleteket:

<	(n1 n2 - - - - f)	igaz a jelző, ha n1<n2;
>	(n1 n2 - - - - f)	igaz a jelző, ha n1>n2;
=	(n1 n2 - - - - f)	igaz a jelző, ha n1=n2;
U<	(n1 n2 - - - - f)	igaz a jelző, ha n1<n2; az előjeleket figyelmen kívül hagyja
D<	(nn1 nn2 - - - - f)	igaz a jelző, ha nn1<nn2; nn1 és nn2 dupla pontosságú érték
D=	(nn1 nn2 - - - - f)	igaz a jelző, ha nn1=nn2; nn1 és nn2 dupla pontosságú érték
DU<	(nn1 nn2 - - - - f)	igaz a jelző, ha nn1<nn2; nn1 és nn2 dupla pontosságú érték; az előjeleket figyelmen kívül hagyja

Miért kell egy jelzőnek „jól neveltnek" lennie?

Írunk egy szót, amely egy jelzővel közli, hogy a veremben talált szám 0 és 9 közé esik-e. A szó neve legyen 1JEGY, veremhatása pedig: (n - - - - f). Meg tudjuk már vizsgálni, hogy egy szám kisebb-e 10-nél (egész számokról lévén szó, ez ugyanaz, mintha a „nem nagyobb-e 9-nél" kérdésre válaszolnánk), és azt is, hogy nagyobb-e -1-nél. A két jelzőből egy ún. logikai ÉS művelettel kapjuk meg, hogy a két válasz egyszerre igaz-e.

A logikai ÉS két logikai értékből állít elő egy harmadikat: ha a két érték igaz volt, akkor a művelet eredménye igaz, egyébként hamis.A jelzők közötti ÉS műveletet az

```
AND
```

FORTH alapszóval lehet megvalósítani. (AND magyarul: ÉS.)
Vigyázat: az AND a logikai „és" műveletet a két operandus bináris alakjának minden egyes bitjével elvégzi! Ha p1. a veremben 2 és 1 volt, azaz binárisan

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

és

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

akkor a logikai ÉS eredménye:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

vagyis 0 lesz, mivel az egymásnak megfelelő bitek közül az egyik mindig 0. Holott mi a 2-t is, az 1-et is igaz értéknek tekintjük, így az AND-nek a mi logikánk szerint igaz értéket kellett volna adnia.Erről a kényelmetlenségről (amely más alkalommal kényelem) tudnunk kell, pillanatnyilag azonban fölösleges miatta aggódnunk; az összehasonlító műveletek „jól nevelt", 0 vagy 1 értékű jelzővel kedveskednek, amelyekkel nem állhat elő a fenti félrekapcsolás.

```
: 1JEGY DUP -1 > SWAP 10 < AND ;
```

A másik fontos művelet a logikai VAGY, amely szintén két logikai értékből ad egy harmadikat. Az eredmény igaz lesz, ha a két logikai érték közül legalább az egyik igaz. Tehát akkor és csak akkor kapunk hamis-at, ha mindkét operaadus hamis volt. Láthatóan ez a VAGY nem felel meg a magyar nyelv VAGY szavának. Magyarul ilyeneket mondunk:

„Vagy láng csap az ódon, vad vármegyeházra vagy itt ül a lelkünk tovább, leigázva",

és ezt úgy értjük, hogy a két lehetőség kizárja egymást. Az előbbi VAGY-ot megengedő VAGY-nak hívjuk, hogy megkülönböztessük a magyar VAGY-ra jobban hasonlító kizáró VAGY-tól. A kizáró VAGY akkor ad igaz eredményt, ha a kapott logikai értékek közül az egyik igaz, a másik nem. VAGY-nak általában a megengedő VAGY-ot hívjuk, ha a kizáró VAGY-ra gondolunk, végigmondjuk a nevét. Ennek megfelelően a két FORTH szó:

```
OR (VAGY)
```

és XOR (eXclusive OR, Kizáró VAGY).

Ezek is bitenként működnek, mint az AND, de az összehasonlító műveletektől kapott „jól nevelt" jelzőknél ez nem jelent különbséget.

Nézzük az 1JEGY-gyel ellentétesen működő TOBBJEGY (n - - - - f) szót, amely akkor ad igaz jelzőt, ha a kapott szám nem esik 0 és 9 közé (azaz kisebb 0-nál vagy nagyobb 9-nél):

```
: TOBBJEGY DUP 0 < SWAP 9 > OR ;
```

Az 1JEGY-gyel ellentétesen működő TOBBJEGY-et persze könnyebb úgy megírni, hogy felhasználjuk az 1JEGY-et. Egy olyan művelet kell hozzá (negálás, komplementálás), amely megváltoztatja a vermen levő jelző jelentését: az igaz jelzőből 0-t, a hamis, azaz 0 értékű jelzőből 1 -et csinál. Ez a szó nem szerepel a szabvány FIG-FORTH 1.1. alapszavak között, de az IS-FORTH-ban megvan, és megírni sem nehéz:

```
: NOT 0 = ;
```

Így a

```
: TOBBJEGY 1JEGY NOT ;
```

működése ugyanaz lesz, mint az előbb definiált másik TOBBJEGY-é.

Folytatjuk!

dBase II. 2.43 (IS-DOS) – V. rész

8. Adatbázis készítése meglévő állományok felhasználásával

Alaptevékenységként az elsők között említettük, hogyan kell egy adatbázist létrehozni, megteremteni. Most azzal foglalkozzunk egy kicsit, hogy létező adatbázisaink felhasználásával hogyan tudunk új adatbázis-állományokat készíteni.

8.1. Adatbázis szerkezetének módosítása

Először egy olyan paranccsal ismerkedjünk meg, amelynek elsődleges célja nem az, hogy új állományt hozzon létre. Az adatbázisokkal végzett munka során bármikor szükség lehet az aktív adatbázis szerkezetének módosítására, például azért, mert egy meződefiníciót rosszul rögzítettünk, és javítani szeretnénk. Erre szolgál a

MODIFY STRUCTURE

parancs. (Csak az aktív adatbázis szerkezetét lehet így módosítani.) A parancs megjelenési formája hasonló, mint a CREATE parancsé, de a jelenleg létező mezők tulajdonságaikkal együtt ki vannak bírva. Nekem leginkább úgy sikerült módosítani egy mező tulajdonságait, ha törlöm a sort (CTRL+G), majd újra beírjuk az egész, hasonlóan a CREATE parancsnál megismertekhez: vesszővel elválasztva a paramétereket. A funkció értelmét megkérdőjelezi, hogy bármit is módosítunk, mindenképpen elvész az adatbázis egész tartalma!

8.2. Az adatbázis részeinek másolása

A következő parancsok már vadonatúj lemezes adatbázis-állományokat hoznak létre, felhasználva a létező adatbázisokat (az új állományok típusjele alapértelmezés szerint „.DBF”). Az aktív adatbázisról vagy annak egy jól meghatározott részéről a

COPY TO <új állománynév> [<érvényességi kör>] [WHILE <feltétel>] [FOR <feltétel>] [FIELD <mezőlista>]

paranccsal készíthetünk másolatot. Az elhagyható paraméterek megfelelő definiálásával befolyásolhatjuk, hogy az adatbázisnak mely rekordjai és mezői kerüljenek át az új állományba. Paraméterek nélkül a teljes adatbázisról készít másolatot, de nem szabad elfelejteni, hogy az esetleges szűrőfeltételt és specifikált mezőlistát, valamint a főindex sorrendjét (ha van) figyelembe veszi ez a parancs is. Az átkerülő mezők tartalmukkal együtt másolódnak az új állományba. A létrejött állomány egyelőre a lemezen található, használat előtt a USE paranccsal meg kell nyitni. A COPY parancs végrehajtása után üzenetet kapunk, amelyből megtudjuk a másolt rekordok számát.

Másolhatjuk úgy is az aktív adatbázist, hogy a tartalmát nem mentjük át a másolatba. Ilyenkor az állománynak csupán a szerkezetét struktúráját - kopírozzuk le. A parancs:

COPY STRUCTURE TO <új állománynév> [FIELDS <mezőlista>]

A szerkezetről készített másolatban csak a mezők közti válogatásra van lehetőségünk. A rekordok nem kerülnek át az új állományba, amelynek így nem lesz egyetlen rekordja sem. Ezt is külön meg kell nyitni, ha használni szeretnénk.

Az üres szerkezetet megnyitása után valamely más állomány segítségével feltölthetjük az

APPEND FROM <állománynév> [FOR <feltétel>]

paranccsal. A forrásállománynak, amelyből a rekordokat át akarjuk emelni, zártnak kell lennie. A „FOR” paraméter feltételében csak olyan mezőnevek szerepelhetnek, amelyek mindkét állományban megtalálhatók. A parancs hatására az azonos nevű és típusú mezők feltöltődnek.

Az aktív adatbázis szerkezetét egy speciális állományba is bemásolhatjuk, amely maga is egy adatbázis, de rekordjaiban a másolt állomány meződefinícióit tartalmazza. Ezt a speciális adatbázist (melynek szerkezetét ugyanúgy a rendszer hozza létre, mint a katalógusállományokét) „szerkezeti állománynak” nevezzük; lehetőséget Nyújt arra, hogy rekordtartalmát módosítva - akár programból is - új adatbázist készítsünk. A szerkezeti állományt létrehozó parancs:

COPY TO <állománynév> STRUCTURE EXTENDED

Az újonnan keletkező állomány, ha másként nem intézkedünk, „.DBF” típusjelet kap, és pontosan annyi rekordja lesz, ahány mezője van az aktív adatbázisnak, amelynek a szerkezetét tartalmazza. (A szerkezeti állományt is külön meg kell nyitni.) A szerkezeti állomány felépítése kötött (ha ki akarjuk használni adottságait, ne változtassunk a meződefinícióin). Négy mezője van, a következő tulajdonságokkal:

FIELD:NAME (karakteres típusú, 10 byte hosszú) - egy korábbi, illetve elkészítendő mező nevét tartalmazhatja, azaz betűket és számjegyeket

FIELD:TYPE (karakteres típusú, 1 byte hosszú) - az iménti mező típusát tartalmazhatja, azaz a C, N, L betűk valamelyikét

FIELD:LEN (numerikus, 3 byte hosszú) - egy lehetséges mezőhosszt tartalmaz, azaz 1 és 254 közötti egész számot

FIELD:DEC (numerikus, 3 byte hosszú) - a mezőhosszhoz tartozó tizedesjegyek számát tárolja

Amikor ezzel a szerkezeti állománnyal dolgozunk (mint adatbázissal, azaz a tartalmát módosítjuk), a dBASE nem ellenőrzi, hogy „értelmes" dolgok kerülnek-e a rekordjaiba vagy sem. Az esetleges hibák csak a szerkezeti állomány felhasználásakor derülnek ki, amikor új adatbázist szeretnénk segítségével létrehozni.

Vadonatúj adatbázis készíthető egy szerkezeti állomány felhasználásával, az alábbi paranccsal:

CREATE <állománynév> FROM <szerkezeti állomány neve> EXTENDED

(A parancs kiadása előtt a szerkezeti állományt le kell zárni.)
E parancs hatására létrejön egy új adatbázis-állomány olyan struktúrával, amilyent a szerkezeti állomány rekordjaiban elhelyeztünk. Az új állomány rekordjait nekünk kell feltölteni.
E két parancs (a COPY STRUCTURE EXTENDED és a CREATE FROM) felhasználásával olyan programokat írhatunk, amelyek futásuk közben építik fel a később használt adatbázisokat. (Kicsit csal az előbbi kijelentésünk, mert pusztán ezzel a két paranccsal nem lehet megoldani a paraméterezett programok írását. Ehhez szükségünk van még néhány speciális függvényre, amelyek ismertetésére a programozásról szóló fejezetben térünk ki.)

8.3. Az összegzetállomány

Egy aktív, numerikus mező(ke)t tartalmazó adatbázisról készíthetünk ún. összeg- vagy összegzetállományt, amelyben az eredeti állomány azonos kulcsú rekordjaiból egyetlen rekord jön létre; ez a numerikus mezőkben az eredeti adatbázis megfelelő rekordjainak összegét tartalmazza.

Az összegzetállományt elkészítő parancs végrehajtáskor feltételezi hogy az aktív adatbázis az összegzési kulcs alapján rendezett. A logikai rendet veszi figyelembe, tehát aktív indexállomány esetén a főindex által meghatározott sorrendet; ha nincs főindex, akkor a fizikai sorrend szerint megy végig a rekordokon. Ez azt jelenti, hogy mindenféleképpen végrehajtódik a parancs, csak rendezetlen állomány esetén nem összegez minden egyforma kulcsú rekordot, csupán azokat, melyek egymást követően helyezkednek el a logikai rendben. Az összegzetállományt a következő paranccsal állíthatjuk elő:

TOTAL TO <állománynév> ON <kulcsmező> [FIELDS <mezőlista>] [FOR <kifejezés>] [WHILE <kifejezés>]

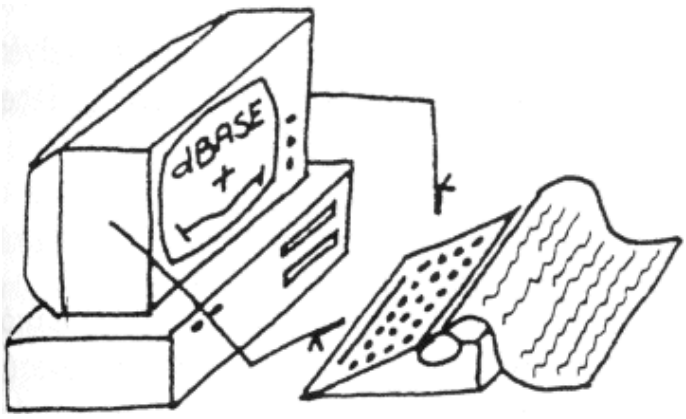
A parancs „FIELDS" paramétere nem az elkészülő adatbázis mezőit határozza meg, hanem azokat a numerikus mezőket, amelyekben végre kell hajtani az összegzést. Alapértelmezés szerint minden numerikus mezőt összegez. (A parancs többi paraméterét már ismerjük.)

8.4. Adatbázis-állományok összefésülése

Két adatbázis felhasználásával is készíthetünk új állományt, ez a forrásállományok mezőit tartalmazza, megadott feltétel szerint „összefésülve" a rekordokat. A műveletben szereplő mindkét adatbázisnak nyitva kell valamely munkaterületen. A keletkező állomány rekordjainak sorr befolyásolja, hogy a kettő közül melyik az aktív. Az állományok összefésülését a JOIN (= összekapcsol, csatla parancsszóval indíthatjuk:

JOIN TO <állománynév> FOR <kif.> FIELDS <mezőlista>

A FOR paraméter feltétele határozza meg, hogy mely rekordokból készül az új állomány. A végrehajtás során az aktív adatbázison megy végig a rendszer, a logikailag első rekordtól az utolsóig, és egy konkrét rekordon állva, a másik állományból kikeresi az összes olyan rekordot, amely kielégíti az adott feltételt, ezeket a megtalálás sorrendjében beírja adatbázisba. A parancs nem igényli egyik állomány rendezettségét sem, hiszen nem kulcs szerint keres.



9. Nyomtatási lehetőségek

Temérdek begépelt információinkat előbb utóbb szeretnénk papíron is viszontlátni. Az sem mindegy, milyen formában böngészhetjük adatainkat, ezért szükség lehet, hogy (korlátok között) a saját igényeinknek megfelelő formában nyomtathassuk ki adatainkat. Az alaptevékenységek között már megismerkedtünk azzal a lehetőséggel (ld. a SET PRINT ON parancsot), amikor minden nyomtatásra került, ami a képernyőn megjelent, kivéve a teljesképernyő-szerkesztő parancsok eredményét. Ebben a fejezetben olyan a nyomtatási lehetőséggel foglalkozunk, amelyek csak a megjelenítendő információt nyomtatja, az ezt előállító parancsot nem.

9.1. Formázott nyomtatás

Ha az adatbázisokról készítünk listát, azt többnyire valamilyen formátumban szeretnénk látni, ez a forma általában gondos tervezés mellett sem (vagy éppen azért nem) egyezik meg az adatbázis rekordfelépítésével. Például egy listában nem árt, ha két adat között nem egy szóköz van, ahogy az a LIST és DISPLAY parancsok esetében szoká-

sos; jó lenne értelmesebb fejlécet is írni a lista tetejére, a jelenleg LIST és DISPLAY esetén megjelenő mezőnevek, melyek nem biztos, hogy mindenkinek egyértelműen jellemzik a mező tartalmát; problémát okozhat, hogy listáink gyakran hosszabbak egy oldalnál, ilyenkor egy LIST segítségével készült lista teljesen kezelhetetlenné válik. A problémák megoldására létrehozható egy ún. felhasználói listaformátumot tartalmazó állomány. A listaformátum-állomány előállítása és módosítása menüvezérelt paranccsal történik. A listaformátum-állomány az aktív adatbázishoz jön létre. A parancs teljes formája:

REPORT [FORM <formátum-állomány neve>] [<érv. kör.>] [FOR <kif.>] [TO PRINT] [PLAIN]

Első feladatunk létrehozni az új állományt:

REPORT FORM formátum-állomány neve

Ha nem adtunk meg nevet (csak azt írjuk be REPORT), a program először megkérdezi, milyen néven hozza létre a formátum állományt. Az utasítás kiadása után az alábbi kérdéseket teszi fel:
Enter options, m=left margin, l=lines/page, w=page width.
Meg kell adnunk, hol legyen a bal margó, hány sor legyen egy oldalon, és milyen széles legyen egy oldal. Például: m=2, l=15, w=50.

Page heading required? Y/N
Kérünk-e fejlécet az oldalra? Ha igen, a következő kérése, hogy gépeljük be. (Enter page heading.)

Double space report? Y/N
Dupla sortávolsággal írja-e a jelentést?

Are totals required? Y/N
Akarjuk-e összegezni valamelyik mezőt?

Subtotals in report Y/N
Akarunk-e, a mezőkhöz készíteni részösszegzést a lap aljára?

Col. width,
Itt írjuk be, hogy melyik oszlopban mi legyen, és milyen szélességben. (Pl: 15,cikknev) Minden oszlop megadása után rákérdez a föléje írandó fejlécre is (Enter heading), Valamint, ha kívánunk összegezni (Totals Required?), és az adott mező numerikus, rákérdez, hogy akarjuk-e összegezni *(Lásd 1. ábra)*
Ha mindez megvan, a formátumot egy file-ba tárolja, FMT kiterjesztéssel. A kész formátum-állományt ezek után máskor is használhatjuk. Ha minden kész, a jelentés megjelenik a képernyőn. Ez persze csak hevenyészett tájékozódásra jó, szebb jelentéseket programból lehet készíteni.

A kész jelentést bármikor kinyomtathatjuk a

REPORT [FORM <formátum-állomány neve>] [<érv. kör.>][FOR <kif.>] [WHILE <kif.>] TO PRINT

```
                              IS-DOS

ENTER TODAYS DATE  AS MM/DD/YY       OR RETURN FOR NONE  :

***  dBASE II     Ver 2.3B   22 FEB 82
. use karton
. report form jelentes
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH m=2,l=15, w=50
PAGE HEADING? (Y/N) y
ENTER PAGE HEADING: Ez a Karton allomanyhoy keszitett Report fejlece
DOUBLE SPACE REPORT? (Y/N) n
ARE TOTALS REQUIRED? (Y/N) y
SUBTOTALS IN REPORT? (Y/N) n
COL     WIDTH,CONTENTS
001     13,cikknev
ENTER HEADING: Cikknev
002     10,ar
ENTER HEADING: Ar
ARE TOTALS REQUIRED? (Y/N) n
003     5,keszlet
ENTER HEADING: Keszlet
ARE TOTALS REQUIRED? (Y/N) y
004
```

(1. ábra)

Mint láthatjuk, nemcsak a teljes adatbázis nyomtathatjuk, szűrőfeltételekkel meghatározhatjuk a nyomtatásban szereplő rekordokat. Jelentésünkben bármikor gyönyörködhetünk a képernyőn is, ha elhagyjuk a TO PRINT paramétert:

```
                              IS-DOS
004


  PAGE NO. 0001

  Ez a Karton allomanyhoy keszitett Report fejlece

     Cikknev        Ar     Keszl
                             et

  golyostoll        23.50   125
  kek tinta         34.60    50
  cimke              0.90  1340
  piros tinta       43.50    76
  toltotoll        231.00    34
  tuzogep          409.00    12
  tuzokapocs         1.50   500
  miltonkapocs       1.50   500
  ragaszto          12.70    77
  geppapir           0.50  2550
  ** TOTAL **
                           5264

.
```

REPORT [FORM <formátum-állomány neve>] [<érv. kör.>][FOR <kif.>] [WHILE <kif.>]

10. Programozás dBASE II-vel

10.1. Néhány szó a „programozásról”

Egy-egy adatbázis használatakor általában ugyanolyan vagy hasonló tevékenységeket végzünk nap mint nap (például listát készítünk bizonyos rekordokról, egy rekordot kikeresve módosítunk benne valamit stb.). Az eddig megismert párbeszédes használat esetén a szükséges parancsokat minden alkalommal újra és újra be kell gépelni, ami fáradságos és egyáltalán nem illik bele a számítógépek „nagy tudásáról” alkotott képünkbe. Ha ezeket a dBASE parancsokat egy szöveges állományban helyezzük el, olyan utasítássorozatot, programot kapunk, amely egyetlen utasítás begépelésével elindítható és utána már „önállóan” dolgozik ugyanúgy, mint bármely más programozási nyelven PASCAL, C vagy BASIC) megírt program.
A gyakorlattal még nem rendelkező programozóknak is

bátran ajánlhatjuk, hogy próbáljanak meg a dBASE segítségével programot írni (ha már a párbeszédes üzemmódban otthon érzik magukat). A dBase sok-sok hibaüzenete és tesztelési lehetősége előbb-utóbb mindenkit elvezet egy jól működő megoldáshoz. Ha rászántuk magunkat, érdemes a következő munkasorrendet betartani:

a feladat pontos meghatározása, megértése
a kimenő adatok meghatározása és formájuk megtervezése
a szükséges - nyilvántartandó - adatok meghatározása és az állományok megtervezése
azon eljárás - tevékenységsorozat - megtervezése („kitalálása"), amelynek eredménye a szükséges kimenet lesz (ezt az eljárást nevezik algoritmusnak)
kódolás - az eljárás utasításainak lefordítása a dBASE parancsnyelvére
a program kipróbálása és a szükséges javítások elvégzése (ezt valószínűleg senki nem felejti el)

Ne keseredjünk el, ha első programjainkban látszólag több a hiba, mint a helyes utasítás, lehet, hogy egyetlen „fatális" hiba okozza az összes többit. Próbáljuk megtalálni és kijavítani a lehetséges hibákat, de amikor már „működik a remekmű", akkor se üljünk nyugodtan jól megérdemelt babérjainkon! Ugyanis a tapasztalt programozók körében igen elterjedt nézrt, hogy „tökéletes" programot nem lehet írni, ezért feltétlenül őrizzük kritikai érzékünket saját programunkkal szemben is.

10.2. dBASE programok készítése és futtatása

Tekintsük át a programírás manuális részét !
A programállomány elkészítéséhez szükségünk van egy szövegszerkesztő programra. Egyet beépítve tartalmaz a dBASE. Tekintve, hogy programjaink szöveges állományban tárolódnak, olyan szövegszerkesztőt használunk, amilyet akarunk. A szövegszerkesztő aktivizálása következő paranccsal történik:

MODIFY COMMAND <állománynév>

A parancs létrehozza a megadott nevű állományt, abban az esetben, ha még nem létezik, egyébként pedig jelenlegi tartalmát, a szerkesztővel együtt betölti a memóriába. Programíráskor az állománynévből a kiterjesztés elhagyható, ilyenkor automatikusan „.CMD" típusjelet kap az állomány, a megadott név az elkészülő program neve lesz.

A beépített szövegszerkesztő vezérlőbillentyűit a függelékben ismertetjük. (Annyit azért már itt segítek, hogy a szerkesztőből kilépni, és a változtatásokat elmenteni a CTRL+W billentyűkombinációval lehet.)
A programállományokba megkötés nélkül bármilyen dBASE utasítás, tehát a szintaktikai összefoglalóban található összes parancs beírható; ezek között van néhány, amelyet párbeszédes módban nem lehet használni, illetve olyanok, amelyeket programba nem érdemes beleírni. A formai követelmények azonosak a párbeszédes mód szabályaival, egyre mégis felhívjuk a figyelmet, mégpedig arra, hogy minden utasítást külön sorba kell írni, és az ‚ENTER' billentyű megnyomásával kell befejezni. Ezt a szabályt különösen a „páros" parancsok használatánál ne feledjük el. Ha egy parancs fizikailag hosszabb, mint egy sor, akkor vagy folyamatosan írjuk (ilyenkor nincs „sorvégjel", de a következő sorban folytatódik a parancs, mivel beépített szövegszerkesztő automatikus szóátdobást végez), vagy pontosvessző („;") után ‚ENTER'-rel befejezve a sort, a következő sorban folytatható az utasítás.
A programállományban a program elejét nem kell külön utasítás megjelölni. A program végét sem kötelező jelölni, de ezt már nem megtenni, mert a végrehajtást bizonyos helyzetekben megváltoztathatja, Egy program végét jelölő utasítás a következő:

RETURN

A program szövegébe megjegyzéseket („commenteket") is elhelyezhetünk, melyek röviden magyarázzák egy-egy programrész tevékenységét, használatukkal programunk mások és (egy-két hét múlva) a magunk számára is érthetőbb lesz a puszta utasítások olvasásánál. Megjegyzés elhelyezhető külön sorban, a következő két „utasítás" valamelyikével:
NOTE <megjegyzés szövege>
* <megjegyzés szövege>
(Ha többsoros a megjegyzés, minden sor elejére ki kell írni a parancsszót.)
A parancsokat közvetlenül követően, azok sorában is elhelyezhetünk magyarázó szövegeket a „&&" jelek mögött (ez csak a fizikai sor végéig tarthat).

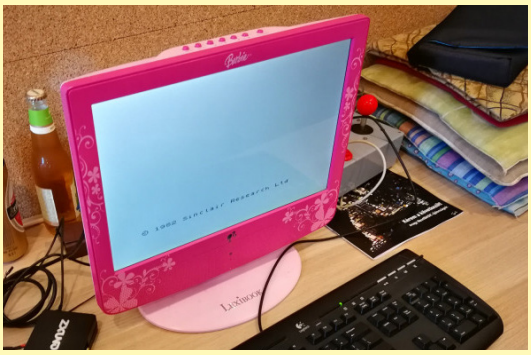
A kész programot a „pont prompt" mellett az alábbi parancs pelésével indíthatjuk:

DO <programállomány-név>

A parancs „.CMD" típusjelet feltételez, de más kiterjesztésű programállomány is elindítható, amennyiben a teljes nevét megadjuk.
Programvégrehajtás alatt (akárcsak párbeszédes módban) a dBASE ún. interpreterként viselkedik - az utasításokat sorról sorra értelmezi és próbálja meg végrehajtani (szinkrontolmácsol a gép és a programunk között). A bekövetkező hibákat azonnal jelzi az interaktív módban megszokott módon. Meg kell azonban jegyeznünk, hogy a hiba nem mindig a kijelzett sorban van.

Folytatjuk!

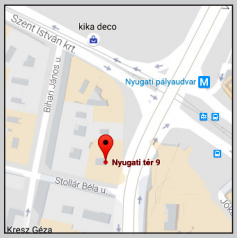
Enterprise Klub 2019. október 12.



ENTERPRISE KLUB

Egy évben 8 alkalommal

Helyszín:
Nyugati Oktatási Központ, Skála terem
Budapest (V. ker.)
Nyugati tér 9.
14 órától 19 óráig



További információ: www.enterpriseklub.hu

Ha te is szeretnél Az ENTERPRESS Magazin szerkesztője lenni, küldj cikket, játékleírást, játékismertetőt, vagy bármit amely az Enterprise számítógéppel kapcsolatos!

A cikkeket erre az e-mail címre küldheted:

info@enterpress.news.hu

ENTERPRISE FOREVER

https://enterpriseforever.com

ENTERPRESS Magazin - 2020/1-2. január - április
Főszerkesztő: Matusa István
Szerkesztőségi főmunkatárs: Németh Zoltán (Zozosoft)
A csapat: geco, Povi, Kiss László, SzörG, szipucsu, lgb, Bakó Róbert, Tamási Istvánné, Kőszegi Ádám, Virág Attila
Design, nyomdai előkészítés: Matusa István
Weboldal: http://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
http://enterpress.news.hu/index.php/magazin