ENTERPRESS

Magazin az ENTERPRISE felhasználóknak

2020/3-6. május - december

File Transfer Enterprise-ra

Játék-fejlesztésem története I.

2020 - az elveszett esztendő...

Írta: Matusa István (Tutus)

Valljuk be férfiasan, senki sem gondolt arra, hogy a mi életünkben ilyen helyzettel kell szembenéznünk! Korlátozások, maszkviselés, káosz, mindenki mást mond, egy nap sem telik el, megcáfolják azt a kijelentést melyet pár órával azelőtt mondtak...

Pont akkor, mikor végre szépen beindult az Enterprise Klubunk. 2020-ban, mikor már nem lehetett összejövetelt szervezni, volt egy próbálkozásunk. Egy online klubot próbáltunk szervezni az interneten, de érdeklődés hiányában erre sem került sor. Őszintén megvallva ez érthető, hiszen nekünk fontos volt az, hogy fizikailag találkozzunk egymással, ott legyenek Enterprise gépeink stb. Nem mindig a gépről és az ezzel kapcsolatos témákról beszélgettünk és ezért is nagyon jó volt az Enterprise Klub!

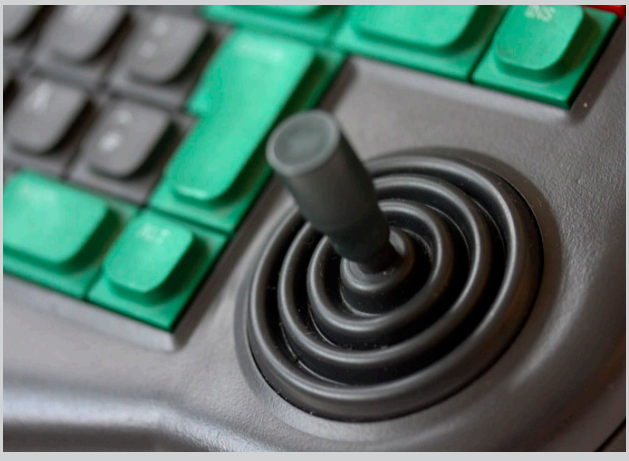
2021 első féléve sem lesz sajnos jobb, úgy tűnik. A 2020-ban befizetett tagdíjak egy részét átcsoportosítjuk 2021-re, valamint az elmaradt jubileumi találkozóra tartalékoljuk és a be nem kalkulált postai kézbesítési díjakra. Természetesen a 2020-as Enterpress Magazinokat minden előfizetőnk megkapja nyomtatott formátumban is postai úton.

Ami a fejlesztéseket érinti, a 2020-as év nem csak a vírus miatt volt „katasztrófa". Sajnos több fejlesztőnknek voltak más jellegű problémái is, így ezek a fejlesztések tovább tartanak majd.

Geco írt egy fantasztikus másoló programot Enterprise-ra, melyet végre felhasználóbarát formában lehet kezelni (lásd. magazinunk 3. oldalán), valamint több játékprogramot is átírt 2020-ban. Köszönjük!

Hardver szempontból nyilvánvaló, hogy az SF3 kártya lett a 2020-as év szenzációja. Hans már a következő verzión dolgozik, melynek neve RSF3 lesz és jóval több lehetőséget kínál majd a CPC valamint Enterprise felhasználók részére.

Az Enterprise Klub elindított egy új weboldalt:
https://enterprise-world.net
Itt megtalálhatóak a legfontosabb programok, kiegészítők, driverek, 3D nyomtató fájlok, melyeket minden Enterprise felhasználó használ. Az oldalt folyamatosan fejlesztjük!



File Transfer Enterprise-ra

- Geco, 2020.

Írta: Matusa István (Tutus)

Évek óta szerettem volna végre egy igazi, jól működő másoló programot Enterprise-ra!

Nézzük át, hogy eddig milyen másoló programok voltak kedvenc gépünkre. A legjobbakból válogattunk.

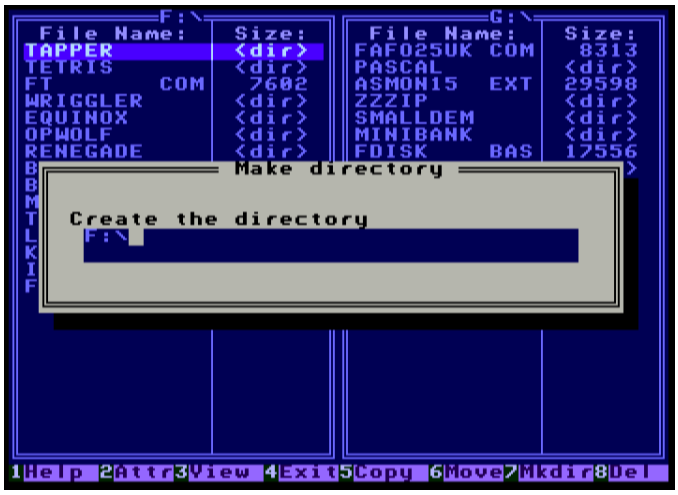
Elsőként az **EPDOS** volt a legjobb, de sajnos itt sem volt két részre osztott képrernyő két meghajtónak. Egy képernyőn jelent meg a file lista, ki kellett jelölni a másolandó fájlokat, COPY és kért a program egy útvonalat, ahova másolni szerettük volna a fájljainkat.

PGCOPY. Miután behívtuk a programot lefoglal egy hatalmas RAMDISK-et (a megkérdezésünk nélkül), amit a másoláshoz fogunk felhasználni. Itt is egy képernyőn jelenik meg a fájl lista.

Kcopy 2.0. A Basic másoló program lemezről lemezre, lemezről magnóra, vagy magnóról lemezre tud másolni (magnóról magnóra tehát nem tudunk másolni). További funkcióként lehetőségünk nyílik file-ok törlésére a kiválasztott lemezen. Extra funkcióként a normál és lassú magnósebesség mellett választhatunk „félturbó" sebességet is. Másolásnál megadhatjuk, hogy a file-ok dátumát a másoláskori dátumra változtassa. A másoláskor megjelenő listából a file-okat az ENTER megnyomásával választhatjuk ki, vagy a TAB megnyomásával kiválasztva megadhatjuk, milyen néven mentsük el a cél lemezen a file-t. a másolást a lista tetején látható SAVE parancsot kiválasztva (ENTER) indíthatjuk. A másoláshoz RAMDISK-et használ.

SymCommander. A SymbOS grafikus operációs rendszer alatt használható másoló program. Itt már két részre osztott képernyő van a meghajtóknak. Sajnos így sem használható teljesen, mivel csak az SD kártyán lévő első két partíciót tudja használni, a kiterjesztett partíciókat nem... Évek óta „könyörgünk" emiatt Prodatronnak, de semmi sem történt. Megjegyzem, és lehet, hogy ezért nem leszek népszerű: a SymbOS tulajdonképpen egy „látvány program", melyet igazán nem tudunk teljes mértékben használni. Az új SF3 kártyához nincsenek disk és net driverek, így sok program és lehetőség nem működik SymbOS alatt... Igaz, hogy ilyenkor megy a visszamutogatás, hogy az EXDOS 3.0 FAT16-os rendszere sem készült még el...

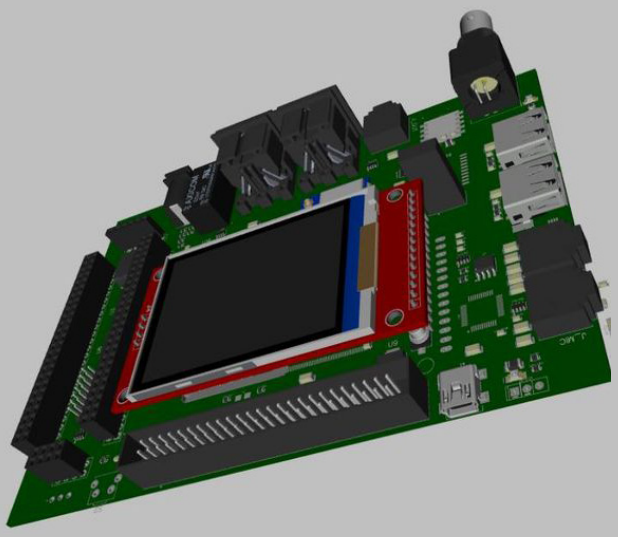
De nincs gond, itt van Geco programja, az FT - File Transfer! Végre tehát megvalósult az álmom, bár fura az, hogy csak én szerettem volna ilyen programot?
Itt megosztott képernyő van a két meghajtónak. A színek tökéletesek, minden jól átlátható és nem kellett a TEXT 80-as képernyőt használni, szépen elfért minden. Annyi, hogy, a joystick-et balra és jobbra mozgatva változik: fájl méret, dátum, idő, file attribútum.
Az INS billentyűvel jelölhetjük ki a másolandó, áthelyezendő, törlendő fájlokat.



A TAB billenytű megnyomásával át tudunk ugrani a másik meghajóhoz.
F1-re HELP-et kapunk a program használatáról. F2-vel tudjuk módosítani a fájlok attribútumát. F3 megnyomására a fájl tartalmát tudjuk megtekinteni karakteresen valamint hexadecimálisan. F4-gyel tudunk kilépni a programból. F5 a COPY parancs. F6-os billentyű: fálok mozgatása vagy átnevezése. F7: könyvtár létrehozása.

Végül az F8-as funkcióbillentyűvel tudunk fájlokat törölni (előtte az INS-el kell kiválasztanunk ha több fájlt szeretnénk törölni). Geco megoldotta már az SF3 kártya fájlkezelését is ebben a programban, valamint külső fájlkezelő parancsokkal is elérhető az SF3 (az SF3-hoz csatlakoztatott Pendrive-on lévő tartalom).

Következő számunkban erről írunk majd.



Elkészültek az RSF3 kártya tervei

Szinte még be sem fejeződött az SF3 kártya fejlesztése, Hans már elkészítette egy új kártya terveit, mely RSF3 névre hallgat.

Több újdonság is kerül majd erre a kártyára, többek között:

- dualcore ARM 480 mhz-es processzor
- TFT érintőképernyős kijelző
- új gyorsabb Wi-Fi modul
- MIDI csatlakozók stb.

Játékfejlesztésem története I.

Az Emberkétől a Hungry Creature-ig

Írta: Bodnár Tamás (Szipucsu)

A 80-as évek vége felé írtam első „játékprogramomat", amely az Emberke volt. A program négyszög alakú karaktereket véletlenszerűen szétszórt a képernyőn, az emberke alakú karaktert pedig a belső botkormánnyal lehetett mozgatni. Egy percünk volt, hogy minél több négyszöget összeszedjünk. A program még a négyszöggel való ütközést sem érzékelte, és ha kimentünk a játéktérből, hibaüzenettel leállt. De azért így is jól el lehetett szórakozni vele. Egy picit mindig szépítettem rajta, így idővel lett Emberke 10 is. Később nagyon megörültem, hogy a Hetedhéten túl című Enterprise-os könyvben a Hamikában volt ütközésérzékelés, így onnan el tudtam lopni a GET #102:A$ elvét, ami lekérdezi a kurzor pozíciójában lévő karaktert. Így már számolta az összegyűjtött négyszögeket, és a játéktér körül keretkarakterek (ez a szó visszafele olvasva is ugyanúgy van!) is voltak, a játéktéren belül pedig falak.

Hála a Zzzip-nek, rájöttem, hogy ebből még többet is ki lehet hozni, így megjelentek a pályán a szörnyek is. Innen egy kicsit komolyabbra fordult a programozás. A játék pályáit DATA sorokban tároltam. Hogy gyorsabb legyen, azt találtam ki, hogy a pálya kirajzolása után a program a képernyő tartalmát, az összes karaktert a képernyőről végignézi, és beolvassa egy tömbbe (a KEP$(X,Y) tömbbe), így a pálya falai abban a tömbben voltak benne. A szörnyek így nem a képernyőről olvasták ki, hogy hol van fal és merre lehet menni, hanem a tömbből, ez gyorsabb volt. Így csak azt kellett a képernyőről kiolvasni, hogy az emberkénk hova lép, szóközre, bogyóra, netalán a szörny karakterére.

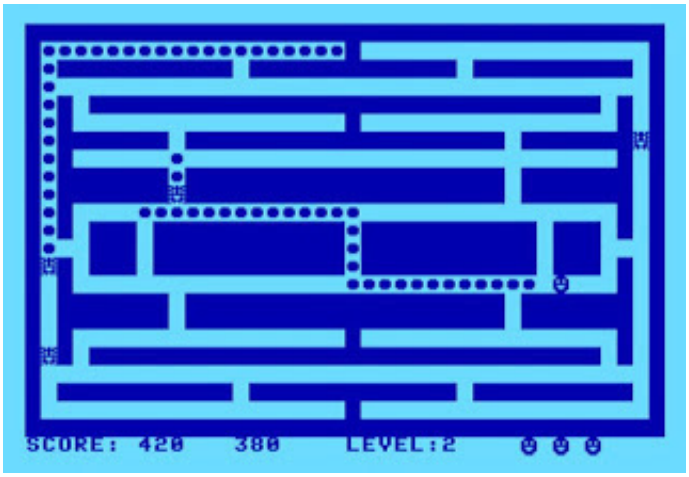
Addig még csak-csak elvoltam a PRINT #102,AT utasítással, a GET #102:A$-gel, a JOY és az RND függvénnyel, viszont a szörnyek mozgása kicsit más volt. A pacman játékokban megfigyelhetjük, hogy a szörnyek általában egy adott irányba mozognak, amíg kereszteződéshez nem érnek, és ott újra felvesznek valamilyen irányt. Ezt próbáltam én is megvalósítani valahogy. Négy szörny volt a pályán, mindegyiknél figyelni kellett, hogy kereszteződéshez vagy zsákutcához érnek-e, és akkor új irányt kellett nekik meghatározni. Ezt először, kisebb küzdelmek árán így sikerült elérnem valamikor 1995 őszén, ha jól emlékszem:

```
3750 LET UT,FEL,LE,JO,BA=0
3760 IF KEP$(SZO1(A)-1,SZO2(A))=" „ THEN LET
     UT=UT+1:LET FEL=1
3770 IF KEP$(SZO1(A)+1,SZO2(A))=" „ THEN LET
     UT=UT+4:LET LE=1
3780 IF KEP$(SZO1(A),SZO2(A)-1)=" „ THEN LET
     UT=UT+8:LET BA=1
3790 IF KEP$(SZO1(A),SZO2(A)+1)=" „ THEN LET
     UT=UT+2:LET JO=1
3800 IF UT=5 OR UT=10 THEN GOTO 3970
3810 IF FEL+LE+JO+BA=0 THEN LET
     I1(A),I2(A)=0:GOTO 3970
3820 IF FEL+LE+JO+BA=1 THEN LET
     I1(A)=I1(A)*(-1):LET I2(A)=I2(A)*(-1):GOTO 3970
3830 SELECT CASE RND(4)
3840 CASE 0
3850  IF FEL=1 THEN LET I1(A)=-1:LET I2(A)=0:GOTO 3960
3860   LET I1(A)=1:LET I2(A)=0
3870 CASE 1
3880  IF LE=1 THEN LET I1(A)=1:LET I2(A)=0:GOTO 3960
3890   LET I1(A)=-1:LET I2(A)=0
3900 CASE 2
3910  IF JO=1 THEN LET I1(A)=0:LET I2(A)=1:GOTO 3960
3920   LET I1(A)=0:LET I2(A)=-1
3930 CASE 3
3940  IF BA=1 THEN LET I1(A)=0:LET I2(A)=-1:GOTO 3960
3950   LET I1(A)=0:LET I2(A)=1
3960 END SELECT
3970 RETURN
```

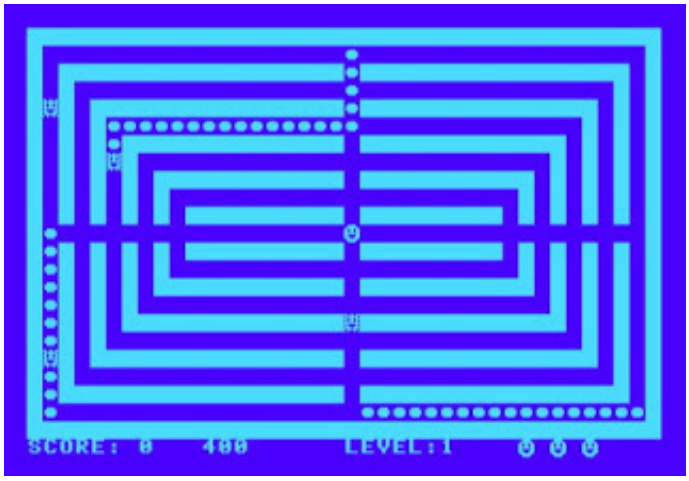
Addig módosítgattam ezt az eljárást, amíg úgy nem működött, ahogy elképzeltem, utána meg már nem foglalkoztam vele, csak örültem, hogy működik. Pedig ez eléggé szedett-vedett, és lehetne rajta szépíteni. A 3760-3790-es sorokban megnézi a program, hogy az adott szörny alatt, fölött, tőle jobbra és balra

szóköz van-e, azaz nincs-e fal, és ha szabad a pálya, akkor az UT változó értékét módosítja, és a túlbiztosítás érdekében a FEL, LE, JO, BA változók értékét is értelemszerűen 1-re módosítja 0-ról. Ezzel eltárolja a gép, hogy merre van lehetséges útirány a szörny számára. A 3800-as sorban megvizsgáljuk, hogy esetleg pont csak függőlegesen mindkét irányba, vagy csak vízszintesen mindkét irányba tudunk-e mozogni, és ha igen, akkor nem történik semmi, hanem tartjuk az irányt, tehát egyirányú folyosón tovább megyünk arra, amerre addig is tartottunk. A 3810-es sorban azt biztosítjuk, hogy ha a szörny mind a négy oldalról körül van véve fallal, akkor semerre ne mozduljon el. A 3820-as sorban azt ellenőrizzük, hogy zsákutcában vagyunk-e, vagyis csak egyetlen irányba mozoghatunk-e. Ilyenkor feltételezzük, hogy az egyetlen lehetséges irányból jöttünk, így az irányt megváltoztatjuk. Ez az irányváltás, és az előbb említett folyosón haladás is csak akkor lehetséges, ha már adott volt egy irány, tehát rögtön a játék indulásakor, amikor az irányt meghatározzuk, ez még nem működik. A pálya indulásakor mindig meg kellett határozni az irányt egy külön eljárással. A 3830-as sortól pedig akkor adunk egy véletlenszerű irányt a lehetséges irányok közül a szörnynek, ha se nem folyosón haladunk, se nem zsákutcába nem értünk, se nem volt a szörny körülvéve minden oldalról fallal.

Egy probléma volt ezzel a pacman típusú játékkal: a képernyőn lévő bogyókat a szörnyek felülírták a mozgásukkal. Bonyolultnak láttam megcsinálni, hogy a szörnyek megjegyezzék, hogy bogyóra vagy szóközre léptek-e rá, és azt rakják ki maguk után. Emlékeztem, hogy általános iskolában talán számítógépes szakkör keretében, vagy ki tudja, mikor, előkerült a Plus/4-es Squirm című játék, ahol a pacmanhez hasonlóan bogyókat kell gyűjteni, a pályán kígyók mozognak, melyekkel nem szabad összeütközni. Ott úgy működött a játék, hogy az egyik kígyó mindig bogyókat hagyott maga után, a többi kígyó pedig szóközt, és 200 bogyót kellett összegyűjteni a pálya teljesítéséhez.



Innen jött az ötletem, hogy a Hungry Creature játékban két szörny mindig bogyókat, a másik kettő pedig mindig szóközöket hagyjon maga után, és bizonyos menynyiségű bogyó összeszedése kellett a szint teljesítéséhez. Ezt sokkal könnyebbnek láttam megvalósítani, mint hogy a szörnyek megjegyezzék, hogy bogyóra vagy szóközre léptek-e.



Nagy örömömre összejött ebből a Hungry Creature című játék, ami Zzzippel lefordítva pont játszható sebességgel működik. Igaz, az egész képernyő egyszínű, mert karakteres képernyőn fut a program, így csak egy tintaszín és egy háttérszín van, de legalább pályánként változik a paletta.

Szerettem zenéket komponálni DATA sorokba írt számok segítségével, így volt néhány szerzeményem. Ezek közül az egyiket betettem a menübe háttérzenének.
Gondoltam, toplista is kellene, hogy beírhassuk játék után a nevünket, a program pedig beillessze pontszámunk alapján a nevünket a megfelelő helyre a rangsorban. Ennek megvalósításáról nem sok fogalmam volt, viszont eszembe jutott, hogy a Felhasználói kézikönyvben volt egy példaprogram, amely 10 számot rendez nagyság szerint sorba:

```
140 NUMERIC TOMB(1 TO 10)
150 NUMERIC VALT,SZAM,MAX
…
250 LET VEG=10
260 FOR X=1 TO 10
270   LET MAX=0
280   FOR Y=1 TO VEG
290     IF TOMB(Y)>MAX THEN LET MAX=TOMB(Y)
300     IF TOMB(Y)=MAX THEN LET SZAM=Y
310   NEXT Y
320   LET VALT=TOMB(VEG)
330   LET TOMB(VEG)=MAX
340   LET TOMB(SZAM)=VALT
350   LET VEG=VEG-1
360 NEXT X
370 FOR X=1 TO 10
380   PRINT TOMB(X)
390 NEXT X
```

Ezt a programot addig módosítgattam, amíg el nem értem, hogy a 10 legjobb pontszámot rendezze sorba a pontszámokhoz tartozó névvel együtt. Így névbeírás után a megfelelő helyre kerültünk a toplistában. Itt is számokat kellett sorba rendezni, de ez annyiból nehezebb volt, hogy a számokhoz (pontszámokhoz) nevek is tartoznak, és a pontszámokat a nevekkel együtt kell mozgatni, a nevek pedig már nem számokként, hanem sztringként tárolhatók:

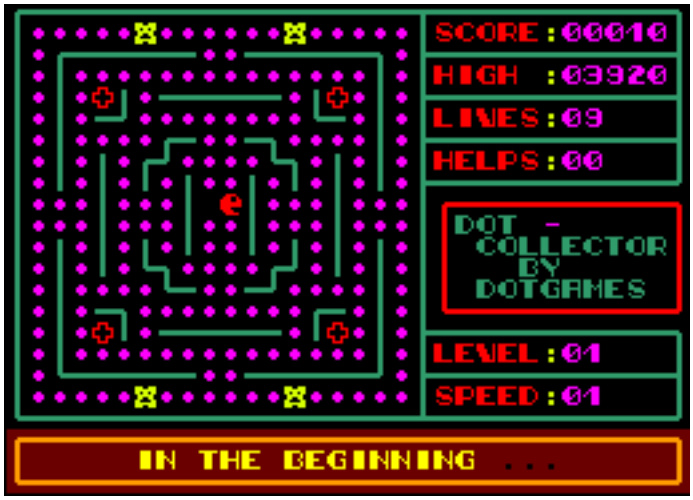
```
140 STRING * 15 N$(1 TO 10),MAX$,VALT$
150 NUMERIC SZAM

6080 LET VEG=10
6090 FOR X=1 TO 10
6100   LET MAX$="99999AAAAAAAAAA"
6110   FOR Y=1 TO VEG
6120     IF N$(Y)<MAX$ THEN LET MAX$=N$(Y)
6130     IF N$(Y)=MAX$ THEN LET SZAM=Y
6140   NEXT Y
6150   LET VALT$=N$(VEG)
6160   LET N$(VEG)=MAX$
6170   LET N$(SZAM)=VALT$
6180   LET VEG=VEG-1
6190 NEXT X
6200 LET MINPONT=VAL(N$(10))
```

Ahogy a MAX$ változóból látható, a pontszámok és a nevek egymáshoz kapcsolását úgy oldottam meg, hogy egyetlen sztringbe írtam bele a pontszámot és a nevet, elöl a pontszámmal.
Sztringeket is nagyság szerint sorba lehet rendezni, mint számokat, de ez kicsit máshogy működik. Sztringeknél elsősorban azt nézi a gép, hogy melyik hosszabb, és az lesz a nagyobb. Ha egyforma hosszúak a sztringek, akkor aszerint dönti el, melyik legyen a nagyobb, hogy az első karaktereknek a kódja melyiknek nagyobb. Hogy egyforma hosszúak legyenek a pontszámot és nevet tároló sztringek, azt találtam ki, hogy az első öt karakter a pontszámot, a következő tíz karakter pedig a nevet tárolja. Ha a pontszám ötnél kevesebb számjegyből áll, az elejére annyi nullát kell lenni, hogy meglegyen az öt számjegy. Ha pedig a név rövidebb tíz karakternél, a végére szóközöket, vagy esztétikai célból inkább pontokat kell tenni. Ezekkel a módosításokkal már a sztringekbe fűzött pontszám+név együttest is a pontszám nagysága szerinti sorrendbe lehet rendezni.
A pontszámtábla képernyőre írásakor pedig először a neveket írjuk ki egymás alá, azaz a sztringet az ötödiktől a tizenötödik karakteréig, utána akár színt váltunk, és az új színnel a nevek után a pontszámokat, amelyek nullákkal az elejükön szebben is mutatnak.
A MINPONT változóra azért volt szükség, mert csak akkor ugrott a program a névbeírás részhez, ha az addigi legkisebb pontszámot meghaladta az új eredmény.
Pályaszerkesztőt is készítettem a játékba, a készített pályákat el lehet menteni, és választhatunk, hogy a beépített vagy a kimentett pályákkal akarunk-e játszani. Ez magnós gépnél nem annyira jó, floppys gépnél elmegy, SD-s gépnél pedig akár egészen előnyös is lehet.

Nagyjából így állt össze a Hungry Creature.

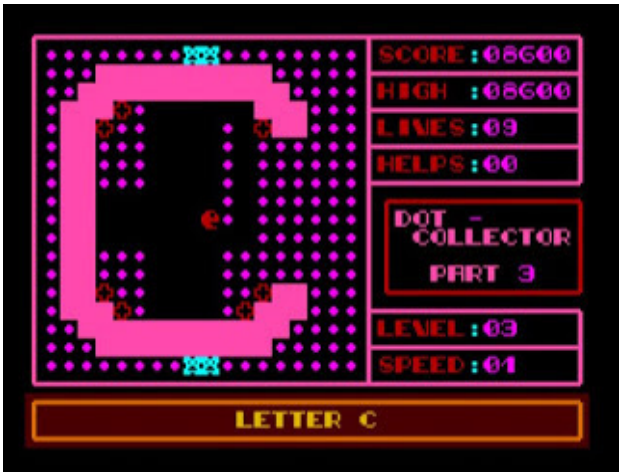
A Dot Collector is bejön a képbe!

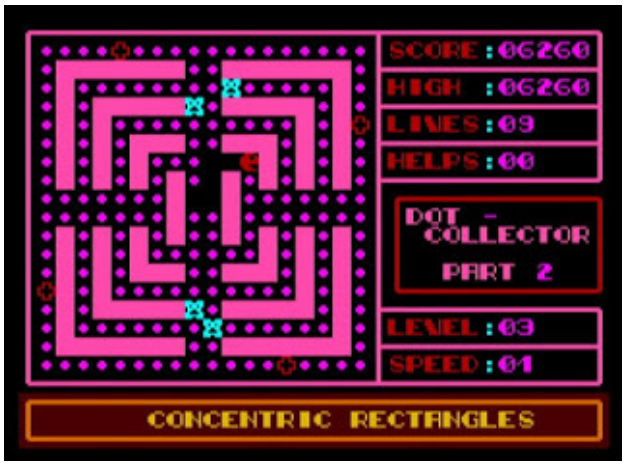
Egyszer valamikor, én se tudom, mikor, talán a 90-es évek második felében eszembe jutott a Dot Collector játék, hogy az milyen rövid, magnóról betöltődik két füttyre a fejléc után, és hogy az is valamiféle karakteres képernyőn fut, legalábbis karakterekből épül fel a képernyő, hiszen a mozgás is karakteres.

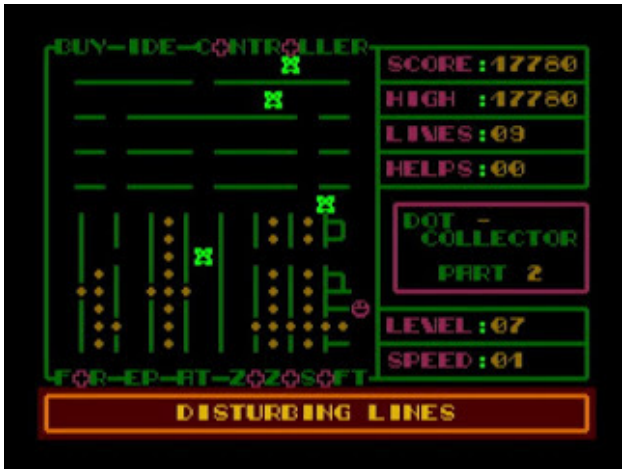


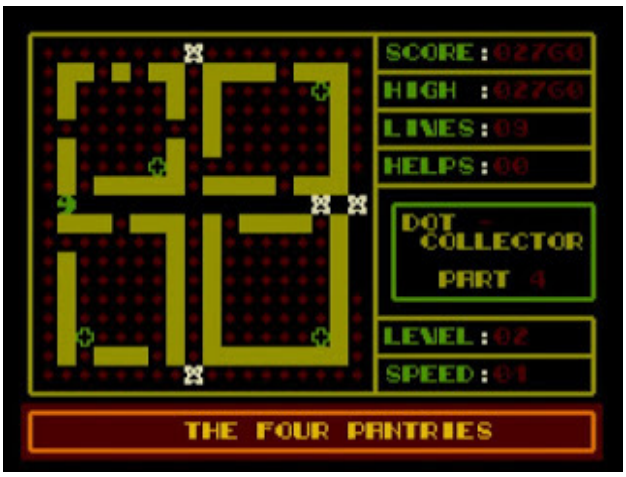
Arra gondoltam, mivel rövid program, talán nem lesz nehéz benne megtalálni azokat a részeket, ahol a pályák tárolódnak. Meg is találtam, ASMON-nal átírva a kód megfelelő részét a pálya is módosult, aminek ugyancsak nagyon megörültem. Persze így kissé nehéz lenne áttervezni a pályát.
Mivel a program rövid, arra gondoltam, egy basic programmal beolvasom magát az egész játékot, egy tömbben letárolom bájtonként, és onnan megjelenítem a pályákat és megcsinálom, hogy lehessen a pályákat szerkeszteni. Biztos eltartott egy darabig, mire ez meglett, de már nem emlékszem a részletekre.
Azonban sajnos tragédia történt. Nem sok floppylemezem ment tönkre, de amin a Dot Collector pályaszerkesztő volt, az pont tönkrement, így elveszett minden. Nem hagyott azonban nyugodni a dolog, és a 21. században újra nekiálltam megírni a Dot Collector pályaszerkesztőt, ami szép lassan kész is lett. Így aztán lett Dot Collector 2, Dot Collector 3, Dot Collector 4, többé-kevésbé újszerű pályákkal. (Az ep128.hu oldalról letölthető a Dot Collector 2., 3., 4. része, és a pályaszerkesztő is:
http://ep128.hu/Ep_Games/Leiras/Dot_Games.htm

A Dot Collector pályaszerkesztője









Szöget ütött a fejembe az is, hogy ha már ki tudom nyerni a Dot Collector pályáit, akár fel is használhatnám azokat basic programban, így akár a Hungry Creature-höz hasonló játékot lehetne írni úgy, hogy a szörnyek a Dot Collector pályáin mozognának. A Dot Collectorban a szörnyek mozgása meglehetősen egyszerűen van megoldva: mindig az emberkénk irányába próbálnak mozogni, arra, ahol éppen nincsen fal. Ezért a Dot Collector némiképp logikai jellegű játék, a szörnyek nem mozognak folyamatosan, ahogy általában a pacman típusú játékokban. Nem tudni, a játék készítői miért így oldották meg, hiszen a véletlenszerű, pacman játékokból ismert szörnymozgás megvalósítása sem lett volna annyira bonyolult, ha már nagy nehezen én is meg tudtam csinálni. De lehet, hogy direkt írták meg így a programot, bár dobhatott volna rajta, ha dinamikusabb a játék. A lényeg, hogy elhatároztam, összeeresztem a Dot Collector pályáit a dinamikusan mozgó szörnyekkel. Ki is cseréltem a Hungry Creature játékban a pályákat a Dot Collector pályáira. Sajnos sokáig nem tudtam folytatni a nagy művet, mert valami probléma volt valamelyik szörny koordinátáival, hiszen a játéktér kb. feleakkora, mint amekkora az egész képernyőn elférne, és a Hungry Creature pályáiban más volt a szörnyek startpozíciója. Sokáig nem jöttem rá, miért nem tudom átírni a szörnyek startpozícióját, hogy most már az új pályákon mozogjanak. De elhatároztam, hogy ha addig élek is, megcsinálom. Emlékszem, valamikor a 2010-es évek közepén lett laptopom, amire természetesen az EP128Emu-t is feltettem. Időnként Pestre utaztam, ami majdnem egy óra vonatozás, és kipróbáltam, milyen lehet utazás közben programozgatni Enterprise-on az emulátorral. A mai napig emlékszem, hogy ott a vonaton sikerült megvalósítani, hogy a szörnyek a Dot Collectorból kiszedett pályán mozogjanak. Nagyon boldog voltam, de a kalauznak nem számoltam be a sikeremről, hiszen nem értette volna, mit jövök én Enterprise emulátorokkal. (Igaz, ki tudja, akár neki is lehetett régen Enterprise-a.) A lényeg, hogy meglett végül a Dot Collector Turbo Edition becenévre keresztelt játék, ahol a szörnyek dinamikusan mozognak, bár a képernyőn még mindig csak két szín volt, az egyik a háttér színe. A Dot Collector pályáin varázstabletták is vannak, melyek egy kis időre sérthetetlenséget adnak, és egy számláló jelzi (a jobb oldali kijelzőn a HELP mellett), hogy meddig tart a hatása. Ezt elsőre bonyolultnak gondoltam megcsinálni. Helyette azt találtam ki, hogy plusz energiát adnak a varázstabletták, így a szörnyek nem nyiffantják ki egyből hősünket. Ahogy a program fejlesztésével haladtam, mindig feltöltöttem az újabb verziót az Enterprise Forever fórumra. Ott Ferro73 fórumtárs is elkezdett foglalkozni a programmal, javaslatokat tett a módosításokra, több részt át is írt benne. Tett bele többek között karakteres animációt, a varázstabletták már villogtak. Javaslatára már nem mentette el tömbbe a játékteret a program, hanem a képernyőről olvasta be a karaktereket, a szörnyek mozgásához. A karakterek beolvasásához már nem GET #102:A$, hanem SPEEK került bevetésre, valamint a szörnyek és az emberkénk képernyőre írása is már nem PRINT AT, hanem SPOKE segítségével történt:

Régi módszer:

```
PRINT #102,AT oszlop,sor:;
GET #102:A$
IF A$=blablabla
```

Új módszer:

```
IF SPEEK(255,VM+(sor*36)+oszlop)=blablabla
LET VM=(SPEEK(255,14644)+
((SPEEK(255,14645)-128)*256))-35
```

Ez persze még csak 128-as gépen működik, Enterprise 64-en nem. Az új módszer programozásánál arra is figyelni kell, hogy a kép videomemória-beli helye nagyon sok mindentől megváltozhat. Ezért fixre nem szabad állítani, ki kell olvasni az LPT táblából, csak úgy lesz jó mindenhol. Ehhez én persze nem értek, Ferro73-ék megoldották ezt is a programban:

```
5515    LET LPBADDL=SPEEK(255,16372):LET
        LPBADDH=SPEEK(255,16373)
5516    LET LPBADDH=LPBADDH-128:LET LPBSZ=255
5517    LET LPBADD=LPBADDL+(256*LPBADDH)
        ...
5530    OPEN #102:"VIDEO:"
        ...
5555    LET CIM=LPBADD+(3*16) !
5556    LET GETAH=SPEEK(LPBSZ,CIM+5)-128
5557    LET VM=SPEEK(LPBSZ,CIM+4)+
        (256*GETAH)-35
```

Így a Dot Collector Turbo Edition már egy feljavított változata lett a Hungry Creature-nek, azon túl, hogy új pályákat tartalmazott. Az új pályák többé-kevésbé eltértek a Dot Collector eredeti pályáitól, mivel az újfajta szörnymozgás bezavarhat ebbe: vagy pillanatok alatt végigságuld a szörny egy folyosón, vagy pedig egy nagyobb, üres részen szöszmötöl sokáig.

A képernyő jobb oldalát a kijelzők foglalják el (pontszám, életek száma, stb.) Ennek az alsó részén van egy kis hely, ahol eredetileg csak egy árva Dot Collector felirat virított. Pihenésképpen megpróbáltam ide más szöveget írni, vagy a játéktéren belülihez hasonló falakat rajzolni. Láttam, hogy ezek a falak akár a játéktér meghosszabbításai is lehetnének, és azt találtam ki, hogy a negyedik varázstabletta felvételekor megnyílik ott egy kapu, vagyis szóközt rak a gép a játéktér jobb alsó részén a falba, és át lehet menni a kijelzők alá. Ezt a lehetőséget SlashNet, ukrán fórumtársunk először bugnak nézte. A későbbi fejlesztéseknél nagy szerepe lesz még ennek a kapunak, ugyanis csak így juthatunk majd el a pálya bizonyos részeire, de ne rohanjunk még annyira előre!

Tervben volt, hogy attribútum képernyősre átírjuk ezt a játékot, ez azonban elmaradt, mert nagyon sok mindent másképp kellene megcsinálni benne, ráadásul attribútum képernyőn a karakteres animációtól is érzékeny búcsút kellene venni.

Egy kis kitérő: a Hamika kígyós játék is berobban a színpadra

Valamikor a 2010-es évek második felében előkerült a téma a fórumon a karakteres-grafikus (Endi által GraCha-nak nevezett) videomódokról. Eltartott egy darabig, mire felfogtam, hogy ez mi is tulajdonképpen, amiről a fórumban a profik írnak. (Erről az Enterpress 2019/3-4. számában már volt egy cikk.) Tehát, a háttér színével együtt egyszerre 4 szín lehet a karakteres képernyőn, egyetlen karakteren belül is lehet variálni a színeket, viszont a karakterek 8 pixel helyett csak 4 pixel szélesek. Ha még karaktersoronként külön videolapokat nyitunk, mindegyiken külön-külön beállíthatunk más palettát, így látványos programokat lehet írni, hiszen a 256 színből van mit válogatni. Születtek is ilyen játékok, a Bricky Prise és a Treasure Cave (by Geco), profi, gépi kódban írt játékok. Engem pedig nem hagyott nyugodni a gondolat, hogy ha már kiderült, hogy van ilyen karakteres-grafikus üzemmód, meg lehetne csinálni a Hamikát is így és akár sok más játékot is. A Hamikát végül megcsináltam, ez lett az Entersnake, erről már volt szó, ugyancsak az Enterpress 2019/3-4 számában.

Volt egyszer egy ismeretlen szerzőtől származó, fel-le scrollozó, karakteres képernyőn futó basic játék, ahol egy labirintusból kellett kitalálni. A neten sehol nem találtam, csak nekem volt meg, még anno egy cserepartnerem küldte. Előástam nagy nehezen valahonnan a süllyesztőből ezt a játékot, de érdemben nem is foglalkoztam vele. Inkább az ütött szöget a fejembe, hogy milyen érdekes ez a függőleges scroll basicben, ilyet még máshol nem igazán láttam (kivéve Endi Mega Pac Man, Zzzippel lefordított játéka), pedig egészen egyszerű megcsinálni, és ehhez négyszínű karakteres lapot is lehetne használni. Elég nagy videolapot létre lehet hozni basicből, többszörösét annak, mint ami a képernyőre egyszerre kifér, és könnyen, gyorsan lehet állítani, hogy ebből mi látsszon éppen. Így egészen egyszerű megcsinálni, hogy egy függőlegesen scrollozó pályán kóválygojon az emberkénk. Az emberkének középen kell lennie, és mindig fölötte és és alatta is kb. 10 karakternyit kell mutatni a képernyőből a DISPLAY utasítás segítségével, akkor is, ha fel-le mozgunk. Ezen az egyszerű ötleten felbuzdulva megcsináltam a Hamikának a scrollos változatát, amikoris a kígyóval egy nagyobb pályán kell bolyonganunk, mint ami a képernyőn egyszerre látszik. Pontosabban csak az első három pályát csináltam meg ebből, a többi még várat magára, ez lenne majd az Entersnake 2.

Folytatjuk!

SymbiFace 3 újdonságok

A ScoreTrack és az SF3 Midi Sinth opciójánál Hans megtalálta a hibát és kijavította.

Azok számára, akik szeretnék kipróbálni a ScoreTrack ROM alkalmazást, és nem akarnak várni a következő DFU kiadásra, frissíthetnek egy béta DFU-val.

Ne felejtsd el, töltsd be a ROM-ot valahová, töltsd be a dalokat és az ENV fájlt egy könyvtárba vagy hajlékonylemezre, majd hajtsd végre:

: SF3 AMODE 7
:ST

Hans kiadta a SymbiFace3 új frissítését.

- Új RAM / ROM menedzsment.

A CPLD chip fejlesztésének helyhiánya miatt a RAM engedélyezett zónák ezentúl 256 KB szélesek lesznek, X0-XF, kivéve néhány kisebb zónát (08-0F, F0-F7 és F8-FB).

A lehetséges írásvédett (ROM) zónák néhány tartományra korlátozódnak: 04-07, 08-0F, 60-6F és 70-7F. Ez csak korlátozza a valódi ROM-ok használatát az SF3-on, de a felhasználó továbbra is elhelyezheti a ROM-okat a RAM zónákban, amelyeket az EXOS helyesen fog megtalálni emulált romként.

Ez nem éppen korlátozás, mert továbbra is ugyanúgy használhatjuk az SF3-at, mint a frissítés előtt, de kevesebb szabadsággal. Most jobban tudatában kell lennünk annak, hogy ne helyezzük a Ram / Rom zónákba, amelyek ütköznek a más eszközök által telepített memóriával. Például Saint memóriabővítései általában egy 64KB-os memória darabot tesznek az alsó sorába, általában XC-XF. Ha engedélyezzük azt a teljes 256 KB-os sort az SF3-on, akkor a két memória összeütközik, és az Enterprise meghibásodik, semmi karcsú. Tehát el kell hagynunk azt a 192 KB-os zónát, (X0-XB).

Mostantól az USB pendrive-ot hatalmas tárhelyként használhatja majd az Enterprise-on ...

Geconak köszönhetően mostantól képesek vagyunk kezelni a FAT32-re formázott pendrive-ot a régi FAT12 rendszerünkön. Olyan rendszert készített, amely közvetlenül az SF3-mal kommunikál és lefordítja a parancsokat a két különböző rendszer között. Valójában ez átlátható a felhasználó számára, mert akár több fájlos programok is használhatók.

A trükköt az SF3Boot.ROM új verziójára tartalmazza, amelyet a TMTLogic linken találsz a Roms könyvtárban.

A használat egyszerű:

Geco egy új DEF_DEV_ parancsot hozott létre, amelyet a munkamenet elején kell végrehajtani.

:def_dev_XXXX
file - sf3 usb
disk - exdos
tape - tape loading

Ahogy a neve is jelzi, ha a def_dev_file parancsot hajtja végre, akkor a Mentés és betöltés alapértelmezett elérési útját hozzárendeli az SF3-hoz, de a trükk nem használja az EXDOS.ROM .... parancsot, : DIR, hibát fog okozni egy only-Tape Enterprise-on, vagy más meghajtótartalmat jelenít meg az SD, IDE vagy Floppy alapú Enterprise-on ...

Ennek megoldására a Geco néhány egyszerű fájlkezelő parancsot hozott létre az SF3Boot.ROM-on belül, amelyek ugyanazokat a műveleteket hajtják végre, mint az EXDOS.ROM-ban ismertek:

:sf3 cd
:sf3 dir
:sf3 md
:sf3 rd
:sf3 del

Természetesen nem veszítjük el a többi tárhelyet sem, mert a műveletet továbbra is átirányíthatja rájuk, ha csak a meghajtó betűjét tartalmazza a névfájlban, például:

töltse be az „f: example.bas" fájlt, vagy ugyanezt magnóval.

Béta-tesztelőként rengeteg programot és fájlt kipróbáltam, és a rendszer nagyon megbízható .... a SymbOS kivételével.

A Symbos ugyanezt tölti majd be az USB pendrive-ról, de miután betöltötte, nem kezeli, csak a szokásos SD vagy hajlékonylemezeket. De ez nem a Geco illesztőprogramjának hibája, csak az, hogy az Enterprise SymbOS verzió még mindig nem beszél az SF3 USB-vel ...

Ezt már régen rögzítették a CPC-verzióban, ezért már közel állunk ahhoz, de már csak arra várunk, hogy Prodatron ezt megvalósítsa.

Élvezd, és nyugodtan tedd fel kérdéseid az EP fórumban!

gflorez

Egyszerre két szövegszerkesztő! Hogy van ez?

Írta: Bodnár Tamás (Szipucsu)

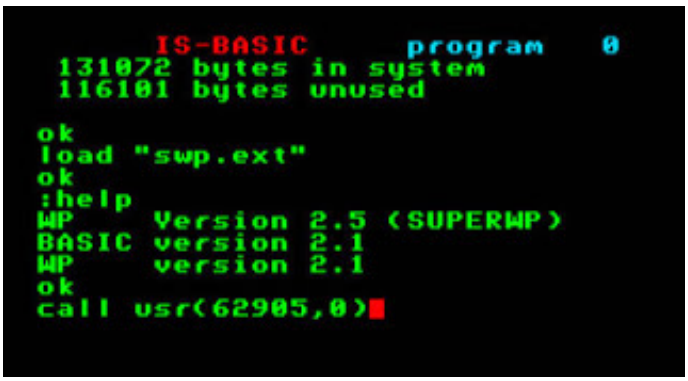
A 20. század vége felé talán több Enterprise tulajdonoshoz is eljutott a Super WP bővítés. Ezt betöltve egy okosabb szövegszerkesztőt kapunk, amely többek között nyomtatóhoz vezérlőkódokat is kezelni tud. Ezen kívül az újabb EXOS verziók is tartalmazzák a szuper szövegszerkesztőt. Ha kiadjuk a :HELP parancsot, meg is győződhetünk erről.

A program messzemenően kompatibilis a számítógépbe beépített WP szövegszerkesztővel. Betöltés után több szembetűnő változást tapasztalhatunk: más színekkel, és 80 karakteres szöveg-megjelenítéssel jelentkezik be.

Igen ám, de a bővítők között ott van a régi beépített szövegszerkesztő is. Azt is a :WP paranccsal tudnánk meghívni, de ilyenkor mindig az új indul el. Nem mintha akkora szükség lenne rá, hiszen a szuper szövegszerkesztő mindent tud, amit a régi, de ha már ott van, el lehet indítani azt is valahogy? Ha nem, akkor miért van ott? A TYPE parancs is az újat hívja meg. Lehet, hogy a szuper szövegszerkesztő felülírja a régit, és a régi még sincs ott? Mi hát az igazság?

Bizonyára minden felhasználó oldalát furdalta a kíváncsiság, hogy mi is a helyzet. Az igazság az, hogy ott van a régi szövegszerkesztő is. Tehát, ha valaki esetleg nem bírja ki máshogy, és a régi színeit akarja látni szövegszerkesztés közben, annak jó hírrel szolgálhatunk! Bár az elindításához némi rafinériára van szükség, ami a következő:

CALL USR(62905,0)



Így már a régi szövegszerkesztő indul el.

A hivatalos kezdési pontja a WP-nek az EXOS ROM 2. 16K-ján (épp az van belapozva a 3. lapra) 62902, ezen a címen van egy rutin, amely ellenőrzi, hogy :WP-t írtunk-e be, ezt kihagyja a CALL USR(62905,0).
Ha Zozotools van a rendszerben, lehetőségünk van a :RL 02h paranccsal teljesen kilőni a régi szövegszerkentyűt a ROM listából. De még az is lehet, hogy egy későbbi verzióban nem is lesz benne a régi, csak az új WP, vagy esetleg külön paranccsal lehet majd indítani? Meglátjuk!

(Megjegyzés: Wolfgang fórumtársunknak ütött szöget a fejébe a nagy kérdés, hogy a régi szövegszerkesztőt elő lehet-e hívni. A problémára a megoldást Dangerman fórumtárs ismertette.)

Super WP történelem

Gépünk beépített szövegszerkesztője (WP 2.1-es verzió) meglehetősen szerény képességgel bír. Talán ezt érezték a készítők is, ezért a német cég később kiadta a Super WP 2.5 változatot. (A Super WP 2.6-ot már ZozoSoft követte el, ez került az EXOS 2.3-ba.) Ez eredetileg német nyelvű változat, de készült belőle „maszek" magyarított verzió is.

(ep128.hu)

Egy régi emlék

Emlékszem, régen egyik EP-hétvégén Zozoéknál voltam és segítségemet kérte (mint nyomdásznak). Találjunk ki valami normális színeket a Super WP-nek! Ezt szerintem sikerült megoldanunk, abban biztos vagyok, hogy az eredeti színeknél sokkal szebbek az általunk kitalált színek! :)

Matusa István - Tutus

:WP21 :WP25

Zozo tervezi, hogy az EXOS-ba bekerülnek majd az alábbi parancsok: :WP21 :WP25. Így kényelmesen elérhető mindkét WP.

ENTERPRISE WORLD

A fontos dolgok egy helyen!

EXOS ROM-ok

EXDOS, IDE ROM-ok

SD kártya szoftverek

MIDI és zeneszerkesztők

3D nyomtató fájlok

SF3 kártya driverek stb.

ENTERPRISE COMPUTERS

https://enterprise-world.net

Pótoljuk ami pótolható!

- első szempont a minőség

2020. június 15.

Egy érdekes poszt jelent meg az enterpriseforever.com fórumon, melyet CS484 nevű fórumtársunk közölt:

„Sziasztok!
Első posztom lévén megpróbálom rövidre fogni a dolgot. SzörG-vel beszélgettem az SD adapterről, és feljött egy esetleges új cartridge ház készítése hozzá. Az is feljött hogy nemigen vannak az Enterhez utánpótlás műanyag alkatrészek. Ezen felbuzdulva leszaladtam a laborba és egy hetes szívás után (néhány vegyszerem már 3x-an túllépte a lejárati időket), úgy néz ki hogy tudok gyártani pót alkatrészeket az Enter-hez.

Sajnos nem sikerült olyan cartridge-et szereznem ami gyári állapotban van. Majd ha beszereztem SzörG-től egy SD adaptert, akkor tervezek hozzá egy cartridge házat és jó minőségben le is tudom gyártani.

Mivel ezen kívül ott van még másik 47 project, plusz család is, ezért ez nem instant fog megtörténni, ennek a postnak főleg igényfelmérő célja van.

Mivel az alapanyagok/gyártástechnológia valamint az időm nem az etherből táplálkozik, a legolcsóbb alkatrész (mondjuk egy joystick gomb) úgy 1e-ről indul, es a cartridge-ot úgy 5e körülre saccolom. Aztán majd ha lesz igény rá, és a technológia adott, akár teljes gépház nyomtatására is sor kerülhet. No de ne szaladjunk ennyire előre, várom visszajelzéseteket...”

CS484 hozzászólásai ebben a témában már elérték a közel 200-at. Leírásai alapján - persze viccesen fogalmazva - mi úgy látjuk, hogy egy komplett gyár van otthonában :-), olyan felszerelésekkel, melyek nekünk ismeretlenek :-)

Félretéve a viccet, CS484 kitartó és maximalista hozzáállásának már meglett a gyümölcse! Több műanyag és gumi alkatrész is elkészült már, nagyon profi minőségben!

Például:
- két féle joystick (eredeti szürke és zöld színben)
- joystick belső alkatrészei
- gumiharang (!!!). (Eredeti szürke és kék színben)
- billentyűk pótlása, melyekre a nevük is rákerül
- LED-sapka, mely szebben világít mint az eredeti

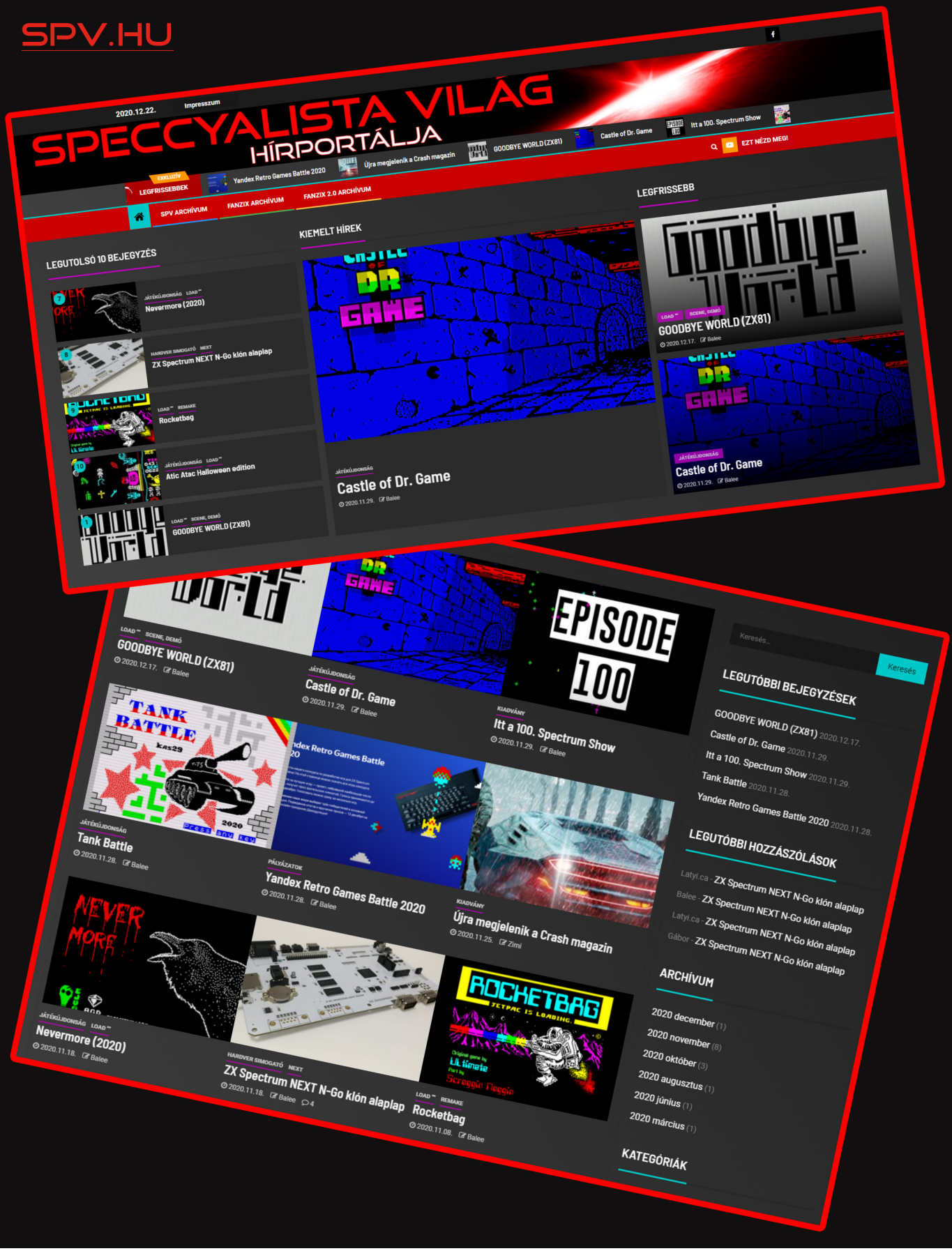
A további alkatrészek tervezése, készítése folyamatban van. Kísérjétek figyelemmel az enterpriseforever.com fórumot!
A színeket majd online magazinunkban láthatjátok :-)

Tutus

ENTERPRISE pólók!

ÚTJÁRA INDÍTOTTUK A SPECCYALISTA VILÁG HÍRPORTÁLJÁT, AHOL IGYEKSZÜNK MAJD A LEGFRISSEBB HÍREKKEL SZOLGÁLNI A SINCLAIR VILÁGBÓL ÉS A MAGAZINUNK HÁZA TÁJÁRÓL.

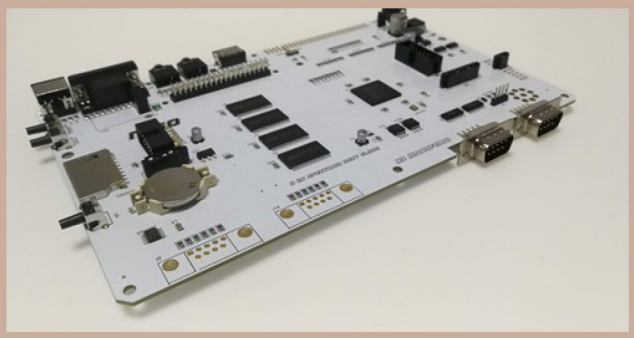
SPV.HU

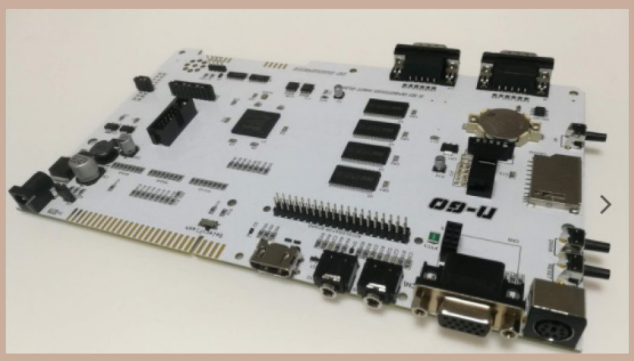


ZX SPECTRUM N-GO KLÓN ALAPLAP

Már van NEXT klón alaplap is, ráadásul ezt még kapni is lehet vagy legalábbis rövidebb határidővel hozzá lehet jutni. Teljesen azonos méretekkel a 2B alaplap klónja. Javításra kerültek a NEXT első szériájának HDMI hibái. Helyet kapott rajta egy kapcsolón keresztül választható dual SPI flash, az egyik a ZX Spectrum Next rendszert, a másodlagos pedig a ZXDOS rendszert tartalmazza.

Állítólag a FREESNGO kuponkóddal még ingyenes szállításban is részesülhetünk.





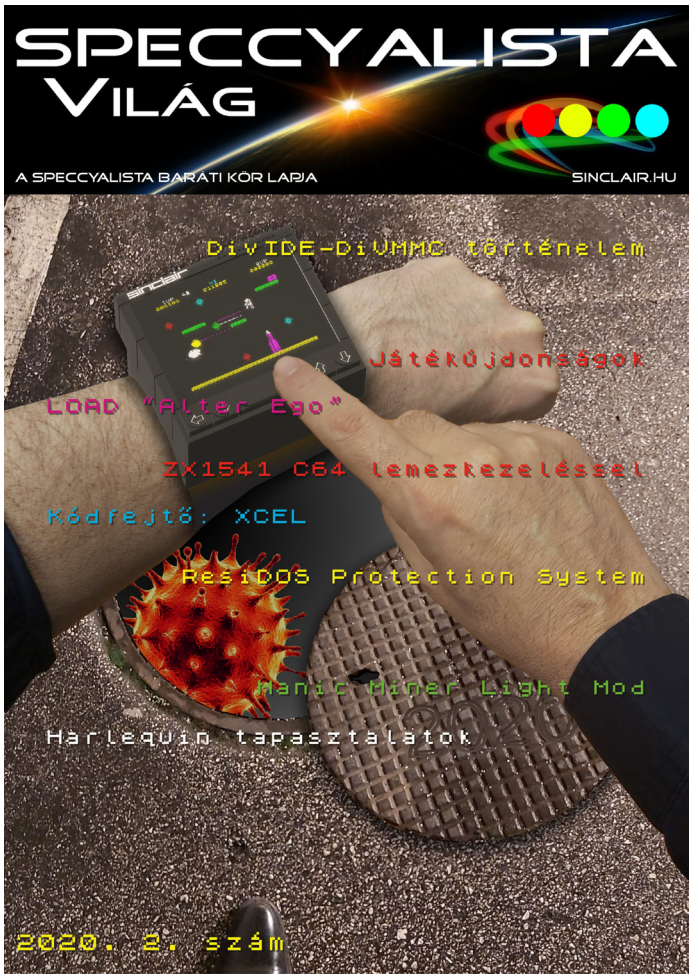
MANUFERHI oldala, ahol megvásárolhatod, akár egy plexi védővel és táppal egyetemben.

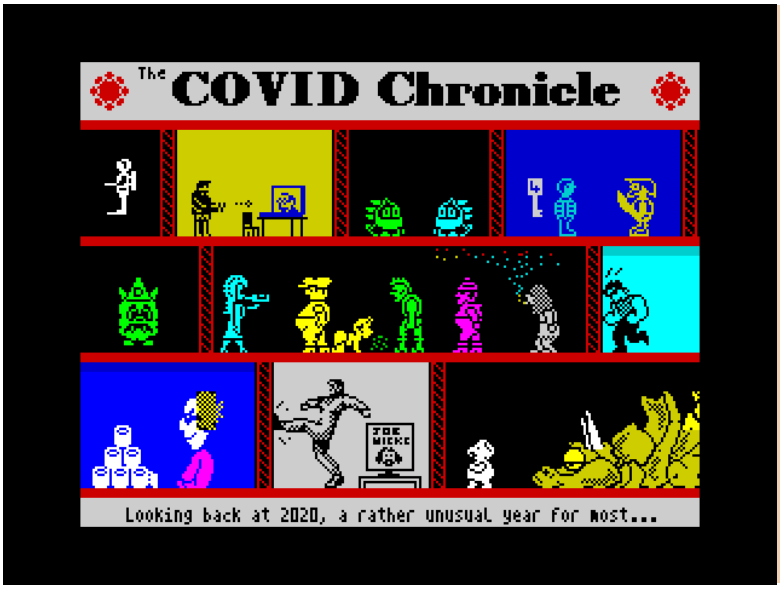
Megjelent a SPECCYALISTA VILÁG legújabb száma

A 2020/2. számban ismét sok édekes cikk található.

Többek között:

- játékújdonságok
- Manic Miner Light Mod
- hardver ötletek
- hardver simogató





The COVID Chronicle 2020

Egy kis visszatekintés. 2020 kicsit másképp. :)
Kiadó : Stonechat Productions (UK)
Szerzők :
Dave Hughes (UK),
Sergio Vaquer Montes (Spain),
Jim Waterman (UK)

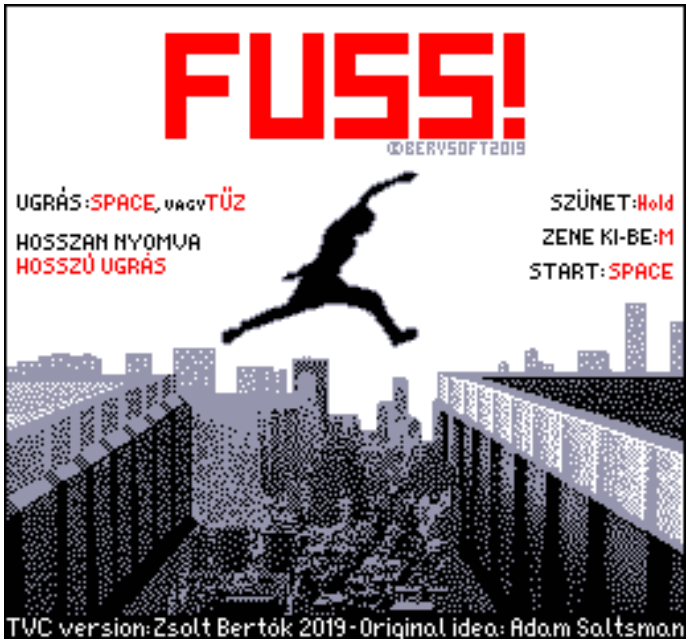
Balee
(Speccyalista Világ hírportál)

2020 terméséből:

Szüretelés a TVC fájáról

Írta: Bodnár Tamás
(Szipucsu)

A 2020-as év kissé Inside Outing volt. Ez azt jelenti, hogy a jól ismert helyzet miatt a kültéri helyeket kevésbé látogathattuk, viszont a lakás minden rejtett bugyrát bőséges lehetősége volt mindenkinek felkutatni, megismerni, átrendezni, vagy akár kristályokat keresni benne, még ha nem is lehetett találni ilyeneket. Természetesen nem volt kötelező senkinek sem otthoni elfoglaltságai keretében bútorokat arrébbpakolászni, felmászni rájuk, levenni a tetejüket, stb., intellektuális elfoglaltságot is lehetett keresni. Ezt tette Geco is, ugyanis CPC és TVC programokat is rábírt arra, hogy Enterprise-on fussanak. Az egyik, TVC-ről átírt program a **Fuss!** című volt, ami nem a programfutásra vonatkozik, hanem a játékosra: A városban tomboló vírus elől meg kell szöknünk, ki kell jutnunk a városból – természetesen csak a játékban. A játék főhőse nem tartozik a tériszonyos emberek közé, ugyanis a háztömb tetejére mászott fel, hogy egyik tetőről a másikra ugrabugrálva, a sok fertőzött embertől távol, egy szál szájmaszk nélkül fusson ki a városból. Ezt mindenkinek érdemes kipróbálnia! – természetesen csak a játékot, a valóságban senkinek nem javaslunk ilyet, de ha valaki mégis megpróbálja, azért a szerkesztőség felelősséget nem vállal! Ez egyébként egy runner típusú játék, és elég egyetlen gombot használni játék során, az ugrást, a képernyő folyamatosan scrolloz jobbról balra. Az eredeti TVC játékot Bertók Zsolt készítette, ez a 2009-es Canabalt című játék feldolgozása. A játék kifejezetten szép grafikával rendelkezik a TVC lehetőségeihez képest, de még Enterprise-on is jól mutat. Érdemes megfigyelnünk a házakon a feliratokat, rögtön a Videoton cég épületének tetejéről indulunk (talán az A Stúdió egyik munkatársát irányítjuk?), és játék során előfordul Centrum felirat is, sőt a Skála Áruház és az Orion is szaladó lábunk alá kerül, mint holmi talp alá való. A Novotrade logó csak azért maradt le, mert cirkalmassága miatt a szerző ebben a felbontásban nem tudta volna elhelyezni egyik épületen sem. A színek és a grafika nagyon el vannak találva, igazi nagyvárosi hangulatot teremtenek a háttérben látszó építkezési daruk is. Különlegesség még az 50 Hz-es (50 fps-es képernyőfrissítést használó) oldalirányú scroll, ami TVC-n szintén nem sok játékra jellemző. Az 50 Hz miatt a scroll nem mindig mutat jól PC-s emulátoron, de eredeti gépen természetesen tökéletes. A játékban lévő város 60 képernyőnyi grafikából áll, a város 104 különböző képdarab kombinációjából van összeállítva. A fejlesztésből az idő több mint felét a grafika és a pálya elkészítése vitte el. A játék könnyűnek



talán nem mondható, sokan pár perc után kidobják a számítógépet az ablakon, mivel nem mindig könnyű eltalálni, honnan kell pontosan ugrani. Bosszúságunkra minden elvétett ugrás után elölről kell kezdenünk a pályát, akkor is, ha már csak pár méter lett volna hátra. TVC-n eredetileg egycsatornás zene volt, ahol a két szólamot úgy oldotta meg a szerző, hogy a hangok igen gyorsan váltogatják egymást. Geco a zenét két csatornára szétszedte, de kedvünk szerint átkapcsolgathatunk egy- és kétcsatornás zene között. A grafikát is kellett konvertálni, és ami még érdekesség, TVC-n a hardver scroll szinte teljesen lehetetlenné teszi a videó memória eloszlását, megszünteti annak teljes folytonosságát, és soron belül teljesen más videócímtől kell a grafikus adatot eltárolni. Enterprise-on ez a rész egyszerűbb lett, mert a scroll nem okoz ilyen törést, megmarad a folyamatos videócímzés.
A játékhoz pályaszerkesztő és zeneszerkesztő is készült Windows alá, ezt azonban sajnos senki nem használta eddig, pedig jó kis pályákat lehetne tervezni.

Ha valakinek hiányérzete lenne, hogy a Fuss! című játék irányításához csak egyetlen billentyű kell, a **Crossfire** játékban kompenzálhatja ezt. Ugyanis itt nem csak négy irányba mozoghatunk, hanem mind a négy irányba lőhetünk is, így a 8 billentyű használata mindenki számára

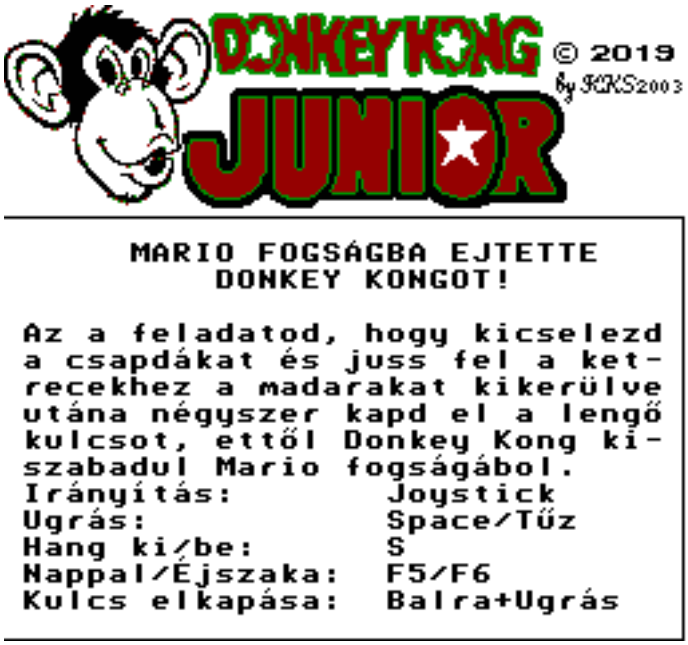
kiváló ujjgyakorlatot és koncentrációt fejlesztő agytornát biztosít. A pályán mind a négy irányból jönnek az ellenséges űrlények is. Ezért a játék leginkább a pacman és az Invaders játékstílust egyesíti magában. A pályán néha megjelenik egy rejtélyes pixelhalmaz, ez muníció. Ha sokat lövünk, elfogy a lőszer, azonban semmilyen kijelző nem mutatja, hány lövésünk van még, így erről lövésünk sincs, amíg el nem fogy mind, és csak nézünk. A dobozukból néha kipottyannak a behajtani tilos táblák, ezeket felvenni nem tilos, sőt ajánlott, bár csak pontszámunkat gyarapítják.

A játék először 1981-ben jelent meg Apple II-re, majd elkészült többek között C64-re, Atarira is. Kollár Zoltán 2020-ban megírta TVC-re a játékot. Az ötletet és a grafikát C64-ről vette, maga a program saját fejlesztés. A szerző eredetileg HT1080Z-re írta a játékot. Ez a gép csak fekete-fehér képet tud megjeleníteni, azonban (elvileg) létezik egy Micolor nevű bővítőkártya (bár jelenleg nem tudni létező példányról). Ezzel a kártyával jobb felbontás és több szín is elérhető. Zoltán a saját Real80 Pro emulátorával a Micolor kártyát is tudja emulálni, ehhez készítette eredetileg a játékot, amiből végül a TVC-s változatot a játékfejlesztő versenyre írta meg.

A TVC-s játékot digitális beszéd színesíti, mely nagyrészt a Galaxy Plus című HT játékból származik. A hangeffektekhez DAC konverterként működik a hanggenerátor (PWM kimenet lesz belőle). Ezek a hangok Enterprise-on megszakításból mennek, így a lejátszás idejére nem áll meg a játék. A játék eredetileg négy színt használ, amivel Geco nem elégedett meg, kihasználta a soronként váltogatható palettát színátmenetekre. A digi hang átalakítása Enterprise-ra nem volt különösebben nehéz.

Ki ne emlékezne a kvarcjátékokra gyerekkorából? Egyetlen hátrányuk az volt, hogy a tanítónéni elvette, ha óra alatt játszottunk vele. Ha valakinek esetleg azóta sem adta vissza a tanítónéni, annak sem kell szomorkodnia, ugyanis Kiss Károly TVC-re elkészített egy kvarcjáték adaptációt: **Donkey Kong Junior.** A program tényleg a kvarcjáték játékstílusát követi, a statikus pályán a kvarckijelzőknél megszokott darabos mozgással mozognak az ellenfelek és az általunk irányított figura.

Mario új vállalkozásba vág, állatkertet akar nyitni, amihez a nagy majmot, Donkey Kongot már el is kapta és ketrecbe zárta. Azonban ifj. Donkey Kong (másik nevén: Donkey Kong Junior) nem hagyja annyiban a dolgot, elindul id. Donkey Kong kiszabadítására. A dzsungelben egyik liánról a másikra kell ugrálnunk, hogy eljussunk a ketrecet nyitó kulcsig, melyet Mario szórakozottsága miatt kiejtett a zsebéből. A dzsungel állatvilága jelen esetben egy fura madárfajból, és egy rendszertanilag nehezen besorolható protézisből áll (talán Mario hagyta ott a fogát Donkey Konggal való küzdelemben?), melyek természetesen ránk nézve harapós kedvükben vannak.



A szerzőnek nagyon tetszett gyerekkori barátjának a Game&Watch Donkeykong Junior kvarcjátéka. Adta magát az ötlet, hogy TVC-re megírja. A játék képeit a netről szerezte be, hiszen PC változat akkor már létezett. Ezeket a képeket aztán át kellett alakítani a TVC-nek megfelelő méretre, színre és formátumra. Ez volt a szerző első gépi kódú programja, amit Z80-ra írt, ezért a megjelenítést háromszor kellett újraírnia az elejétől kezdve, mire jó lett. Károly megkereste a gyerekkori barátját is, akié a gép volt, ő írt hozzá a Donkeykong játékok zenéit összemixelve egy intro és egy game over zenét. A 2019-es TVC játékíró versenyre készült a játék, így a végét gyorsan kellett befejezni. A játékmenet így nem teljesen olyan, mint az eredeti, nem annyi az ellenség és nem olyan gyorsan jönnek, de így is élvezhető a játék. Sokat segítettek Károlynak a programozásban a TVC Facebook csoportban közzétett leírások, segédprogramok.

Bár Enterprise-on a szoftveres TVC emulátorral is fut a játék (néhány kisebb hiányossággal), Geco teljesen átírta Enterprise-ra, és most 64k-s gépen is játszható. Az időzítés megoldása okozott kisebb problémát, mert a TVC verzió a késleltetést Z80 ciklusokkal oldotta meg, így EP-n ez alapból rövidebb ideig tartott. Geco úgy akarta, hogy turbós gépen se gyorsuljon fel, ezért áttért az 50Hz-es időzítésre, amit nem lehetett teljesen pontosan megfeleltetni a Z80 ciklusos időzítésnek, de nagyjából sikerült.

IS-FORTH - 3. rész

Intelligent Software - 1985
rendszerbővítő, FORTH programozási nyelv

1.6. Gyorsműveletek

A legtöbb számítógépnek gyorsan működő gépi utasítása van arra, hogy valamit 1-gyel növeljen vagy csökkentsen, 2-vel szorozzon vagy osszon, megvizsgálja az előjelét. Ehhez képest az a sorozat, hogy 1 + (tegyél a veremre 1-et, hívd a + szót) lassú és nehézkes. Az ún. „gyorsműveletek" levágják a felesleges kanyarokat, és körülményeskedés nélkül elindítják a megfelelő gépi utasításokat. A gyorsműveletek:

1+	(n - - - - n1)	eggyel növeli n értékét;
1-	(n - - - - n1)	eggyel csökkenti n értékét;
2+	(n - - - - n1)	kettővel növeli n értékét;
2-	(n - - - - n1)	kettővel csökkenti n értékét;
2*	(n - - - - n1)	megduplázza n értékét;
2/	(n - - - - n1)	megfelezi n értékét;
0=	(n - - - - f)	f akkor igaz, ha n = 0;
D0=	(nn - - - - f)	A 0= dupla pontosságú alakja;
0<	(n - - - - f)	f akkor igaz, ha n < 0;
0>	(n - - - - f)	f akkor igaz, ha n > 0;

Láthatóan a gyorsműveleteket végző szavak hasonlóan néznek ki, mint az ugyanúgy működő lépésenkénti parancsok, csak egy szóba írjuk az operandust a műveleti jellel; az 1+ szó ugyanazt a műveletet végzi, mint az 1 + sorozat, csak gyorsabban.

1.7. A verem átrendezése

A FORTH szavak elvárják, hogy a vermen a megfelelő sorrendben kapják a működésükhöz szükséges paramétereket. Ez nem mindig egyszerű. Időnként a paraméterek a veremben rossz sorrendben keletkeznek, lehet köztük felesleges, de az is előfordulhat, hogy valamelyikre még egyszer szükség lenne. Az ilyen gondok megoldására szolgálnak a következő szavak:

SWAP (a b - - - - b a) megcseréli a két felső elemet;
2SWAP (a b c d - - - - c d a b) párban megcseréli a négy (vagy két dupla pontosságú) legfelső elemet
DUP (a - - - - a a) megduplázza a legfelső elemet;
2DUP (a b - - - - a b a b) megduplázza a két (vagy egy dupla pontosságú) legfelső elemet;
?DUP (a - - - - a a) megduplázza a legfelső elemet, ha az nem nulla.
OVER (a b - - - - a b a) a második elemről készít egy másolatot a verem tetején;
2OVER (a b c d - - - - a b c d a b) a harmadik, negyedik (vagy a második dupla pontosságú) elemről készít másolatot a verem tetején;
ROT (a b c - - - - b c a) a harmadik elemet kiszedi alulról, és feldobja a tetőre;
2ROT (a b c d e f - - - - c d e f a b) az ötödik, hatodik (vagy a 3. dupla pontosságú) elemet kiszedi alulról, és feldobja a verem tetejére;
-ROT (a b c - - - - c a b) a ROT-al ellenkező irányba forgatja a verem legfelső három elemét.
-2ROT a két első elemet berakja az ötödik, hatodik helyre
DROP (a - - - -) eltávolítja a legfelső elemet;
2DROP (a b - - - -) eltávolítja a két (vagy egy dupla pontosságú) legfelső elemet.
NOT (- - - - 0) logikai hamis, azaz 0 értéket tesz a verembe

Írjunk például egy olyan szót, amelynek hatása a veremre: (X Y - - - - Z), ahol Z = X * Y - (X + Y)

Nem kezdhetjük a dolgot aritmetikai művelettel, hiszen akkor elveszítenénk az x-et meg az y-t a veremről. Valamilyen módon konzerválnunk kell őket. Jó fogás erre az OVER kétszeri alkalmazása. Az egyes lépések mellett feltüntettük, hogy a lépés után mi lesz a veremben; ez a felírási mód igen hasznos, amíg nem válunk a verem rutinos bűvészévé. (Senkit ne zavarjon, hogy a definíciót több sorba írtuk!)

```
: XY
OVER
OVER
*
ROT
ROT
+
-          ( X Y ) (ez van a legelején a veremben)
( X Y X )
( X Y X Y )
( X Y szorzat )
( X szorzat Y )
( szorzat X Y )
( szorzat összeg )
( Z )
```

Hasznos, de nem szabványos szavak:
Van néhány veremkezelő FORTH szó, amely nincs benne a FIG FORTH alapszókészletben, az IS-FORTH-ban azonban szerepel:
DEPTH (- - - - n)
A szó (jelentése: mélység) a verem tetejére teszi a verem elemeinek (a DEPTH végrehajtása előtti) számát.
PICK (n1 - - - - n2)
A verem bármely elemét a verem tetejére másolja. Használatakor a 0 jelenti a verem első elemét, az 2 a másodikat és így tovább. Ha például a negyedik elemet akarjuk legfelülre másolni és a verem tartalma a következő: 1 2 3 4 5, akkor adjuk írjuk be: 3 PICK
Ennek hatására a verem tartalma a következő lesz: 1 2 3 4 5 2
Érdemes megjegyezni, hogy a 0 PICK megegyezik a DUP-pal, az 1 PICK pedig az OVER-rel.
ROLL
Kiszedi a verem n-edik elemét és a verem tetejére teszi. Hasonló a ROT-hoz, csak itt tetszőleges számú elem megforgatható (rotálható). Tegyük fel, hogy a verem tartalma a következő: 1 2 3 4 5 6 7
A 4 ROLL hatására az eredmény a következő: 1 2 4 5 6 7 3
A 3 ROLL a ROT, a 2 ROLL a SWAP szóval egyenlő hatású.
A veremhatás szokásos jelölésével a ROLL működését csak pontatlanul írhatjuk le.

Verem mélység
Egy helyesen futó program esetén nagyon valószínűtlen, hogy a verem kitöltené a rendszer memóriáját. Ennek ellenére, ha egy ciklusban maradék elemet hagyunk a veremben, előfordulhat a verem túlcsordulása. Az ilyen hibák elkerülésére tanácsos felkészülni. Ezt nagyban segíti a „DEPTH" (mélység) parancsszó. Ha használjuk, akkor a verem tetejére kerül a DEPTH végrehajtása előtti veremmélység. Ezután ez az érték kiíratható a „." operátorral. Ez szintén használható összetettebb ciklusok ellenőrzésére. Ha a ciklusmélység meghalad egy irreális értéket (pl. 2000), akkor ezt a hibát egy adott programrésszel lekezeljük. Ez könnyen megvalósítható egy feltételes vezérlésátadó (IF) utasítással. A jó programozási módszerek elkerülik ezt a típusú problémát.

1.8. Még egy szó a kiírásról

A

."

szó kiírja az utána megadott szöveget egészen a legközelebbi idézőjelig („) A záró idézőjelnek és a ." szónak egy sorban kell lennie! A ." szó csak szódefinícióban alkalmazva működik! Parancsmódban a .(szó használható. Két egyszerű példa:

: LOCSI-FECSI CR ." En vagyok az ENTERPRISE" CR ;

.(En vagyok az ENTERPRISE) CR

Nem szabad elfeledkezni róla, hogy a ." és .(után szóközt kell hagyni! Így jelezzük, hogy a ." és .(külön szó. A szóköz nem számít bele a kiírandó szövegbe.

Formázott kiírásra ad lehetőséget a
.R (n m - - - -)

A szó n-et egy m szélességű mezőben jobbra igazítva írja ki.

1.9. Hogyan tároljuk programjainkat?

Eddigi próbálkozásainkban az a bosszantó, hogy a programok szövegei nem maradnak meg, nem lehet őket kijavítva vagy változatlanul újra felhasználni. Megtehetjük, hogy a programokat nem közvetlenül adjuk át az interpreternek, hanem valamilyen adathordozóra, az ún. screen-ekbe írjuk őket; itt bármikor javíthatók vagy elolvastathatók az interpreterrel. A screen (ejtsd: szkrín) szó magyarul képernyőt jelent, amit mi rövidítve kernyőnek fogunk nevezni, de a szakirodalom blokként is említi.
A kernyő a szöveges információ tárolásának eszköze. Egy kernyőben annyi szövegnek van hely, amennyit egyszerre kezelhetünk a képernyőn.; ez a FORTH szabvány szerint 16 sor, egy sorban 64 karakterrel. Az IS-FORTH-ban egy kernyő 1024 byte (16*64). A szabvány FORTH egy lemezt egy szektorhalmaznak tekint, amelyet feloszt magának kernyőkre.

Az IS-FORTH ún. kernyőfile-okat használ, ezzel lehetővé téve, hogy ugyanazon lemezen más file-okat is tárolhassunk. A file neve a kernyő száma lesz (a kiterjesztése pedig 4TH). Az IS-FORTH nem csak a hagományos blokk-formátumot kezeli, mert azt magnós rendszerben nem kényelmes használni.
Lemezes rendszerben ezek a blokkok a lemezen helyezkednek el. A megfelelő részek szerkesztéskor programíráskor bekerülnek a memóriába (bufferekbe), majd később kiíródnak. Mindez láthatatlan a felhasználó számára. Amikor egy blokkot a BLOCK-kal el akarunk érni, akkor esetleg már az előbbi műveletek miatt bent van memóriában. Ebben az esetben a memóriabufferek tartalma jelenítődik meg szerkesztésre. Ha a kért blokk nincs a memóriában, akkor az egy üres bufferba másolódik. Amennyiben már az összes buffer foglalt, akkor a legrégebben használt visszakerül lemezre, és így felszabadul egy az új blokk számára.
Egy kazettás rendszerben, a blokkoknak mindenképp a bufferekben kell lenniük. A blokkok folyamatos mozgatása, a memória és a szalag közt hosszadalmas és kényelmetlen lenne. Ha nincs elég memória a blokkoknak, hibajelzést kapunk. A CREATE-BUFFERS hívás további helyeket szabadít fel (lásd referencia rész).

Az editor
A programok szerkesztéséhez a beépített szövegszerkesztő használható. A szövegszerkesztőt az

EDIT (n - - - -)

paranccsal indíthatjuk. Az n szám a szerkeszteni kívánt kernyő sorszáma lesz. (Pl. 1 EDIT parancs kiadása után az 1. kernyőt (blokkot) szerkeszthetjük.) A blokkok sorszáma 1-től 32767-ig terjedhet. A szerkesztés közben a szabványos Exos szerkesztő-funkciókat használhatjuk.

Amennyiben az adott kernyő-ben nem fejeződik be a program, a következő blokkra a

-->

szóval hivatkozhatunk a kernyő végén. Az így összefűzött blokkoknak egymás utáni, növekvő sorrendben kell lenniük.
Amikor a blokk szerkesztését befejeztük, az ESC megnyomásával léphetünk ki a szerkesztőből. Egy rövid szünet után a FORTH visszatér a normál szöveges vagy grafikus képernyőhöz. A begépelt szöveg most a bufferekben foglal helyet, és újra behívható a szerkesztés céljából. Kilépéskor a FORTH megadja a kernyőben felhasznált byteok számát. A szerkesztéshez használt buffer 2 K méretű, így előfordulhat, hogy 1 K-nál nagyobb programot írtunk be. Ilyenkor a következő üzenetet kapjuk az ESC lenyomása után:

Block too large: ABORT (Y/N)?

A felhasználónak négy lehetősége van ilyenkor:
Y Az editálás megszakad, és az editor buffer tartalma nem másolódik át egy Forth bufferba. A szerkesztett szöveg elvész.

N A szerkesztés nem fejeződik be, folytathatjuk mintha semmi sem történt volna. (Kitörölhetjük a program egy részét.)

ENTER

A szerkesztés befejeződik, az editor buffer tartalma nem másolódik át a Forth blokk bufferébe. Az editor buffer érintetlen marad, és ugyanannak a blokknak az újbóli editálásakor meglesz a régi szövegünk.

ESC A szerkesztés befejeződik és csak az editor buffer első 1 K-ja másolódik a Forth bufferébe. Az editor buffer érintetlen marad.

Ha bármely esetben leütjük az ESC billentyűt, a blokk buffer megjelölésre kerül mint módosított, és így lemezes rendszeren újraíródhat más blokkok elérése esetén.
A fenti esetekben amikor az editor buffer érintetlen marad, az azt jelenti, hogy ugyanazt a blokkot újra szerkesztve a blokk nem töltődik be az editor bufferba, hanem az utoljára ott lévő szöveget szerkeszthetjük.
A szerkesztés alatt néhány szó lezárja a csatornát és ezzel elvész a bent lévő szöveg. (Pl. TEXT, LORES, stb.)

A szerkesztést bármikor megszakíthatjuk a STOP billentyű megnyomásával. Ez a következő üzenetet eredményezi:

STOP key pressed: ABORT (Y/N)?

A Y, N és az ESC válaszok hatása ugyanaz, mint) szerkesztés befejezésénél (ESC) láttuk.

Az ENTER billentyű használható arra, hogy átmenetileg a Forth-hoz térjünk vissza szerkesztés közben, úgy, hogy a beírt szöveg megmaradjon.

Programok kezelése
A kazettára mentéshez / betöltéshez a programnak nevet kell adni. Ezt a nevet a

NAME (addr - - - -)

Változó tartalmazza. Ennek legegyszerűbb formája:

„ név” NAME

Magnós rendszerben, ha nem használjuk a NAME változót, név nélkül is menthetünk.
Ha több buffert akarunk kezelni (így több blokk mentése válik lehetővé egy adott néven - kazettás rendszerben ezt a módot javasolt használni.), akkor a

BUFFERS ON

Szavakat használjuk. Ilyenkor az össze használt blokk egy file-ba kerül mentésre.
Amennyiben minden blokkot külön kívánunk kezelni (lemezes rendszerben) úgy a

BUFFERS OFF

használandó. Ez utóbbi esetben a használt blokkok külön-külön file-ba kerülnek a lemezre. A NAME változót ilyenkor figyelmen kívül hagyja a rendszer, a file-ok nevei a blokkszám lesz, a kiterjesztés pedig .4TH. (Tehát ha pl az 1-2. blokkokat használtuk, 1.4TH és 2.4TH file-ok jönnek létre. A .4TH kiterjesztésű blokk-file-ok 1024 byte-os szöveges állományok, míg a több blokkot tartalmazó formátum 01h típusbájtú EXOS modul.
Ha a FORTH lemezes rendszert detektál induláskor, automatikusan a BUFFERS OFF módit állítja be alapértelmezésnek, magnós konfigurációban pedig a BUFFERS ON módot.
SAVE-BUFFERS (- - - -) Adott néven szalagra / lemezre ment minden blokkot.
LOAD-BUFFERS (- - - -) Beolvassa szalagról / lemezről a SAVE-BUFFERS utasítással kimentett blokkokat.
EMPTY-BUFFERS (- - - -) Felszabadítja az összes buffert, de NEM menti el azokat!

Ha megszerkesztettünk egy kernyőt, akkor a

LOAD (n - - - -)

szóval átadhatjuk az interpreternek, ha még nincs a memóriában, betölti lemezről a rendszer. (Lefuttathatjuk a programot.) A vermen a kernyő számát kell megadni. A LOAD hatására pontosan ugyanaz történik, mintha begépelnénk a megadott számú kernyőn található szöveget. Ha, mint többnyire, a kernyő definíciókat tartalmaz, akkor a betöltés, vagyis a LOAD hatására megjelennek a szótárban a definiált szavak. Ha több egymás utáni kernyőt akarunk betölteni, akkor a --> szót írjuk a kernyők végére. Betöltéskor ennek hatására az adott kernyő interpretálása abbamarad és a következő kezd el betöltődni. A

;S

szó hatására a kernyő interpretálása megszakad, az interpreter ott folytatja, ahol a LOAD volt.
Egyszerre több egymást utáni blokkot adhatunk át az interpreternek a

THRU (n m - - - -)

szóval, ami az n és m közöti blokkokat tölti be. A 2 5 THRU utasítás a kettes blokktól az ötösig tölt.
Az interpreternek szinte mindegy, hogy az adott pillanatban billentyűzetről vagy kernyőről kapja-e a vezérlést. Azt a karakterfolyamot, amelyet az interpreter értelmez, az angol nyelvű irodalom input stream-nek nevezi, ezt itt befolyamnak fordítjuk.
Egy kernyő szövegét a

LIST (n - - - -)

szóval írathatjuk ki a képernyőre.
Megállapodás szerint minden képernyő legfelső sora tartalmaz valamilyen utalást a kernyő tartalmára. Ezt használja fel az

INDEX (tól ig - - - -)

szó, amely nem szabvány ugyan, de az IS-FORTH alapszókészlet tartalmazza. Az INDEX kilistázza a megadott két szám közötti számú kernyők legfelső sorát és ezzel mintegy tartalomjegyzéket ad a kernyőinkről.

2. Vezérlési Szerkezetek

2.1. A feltételes utasítás

Tudjuk már, hogyan kaphatunk jelzőt egy feltétel igaz vagy hamis voltáról. Most megtanuljuk, hogyan lehet a jelzőket használni.

Az IF ... THEN
(A szabvány FORTH-ban IF ... ENDIF szerkezetnek hívják). Szerkezete:

(feltétel vizsgálat) IF (az igaz ág szavai) THEN

Az IF ... THEN szerkezet működése: az IF megeszi a verem tetején levő jelzőt. Ha a jelző igaz, az IF és THEN közötti programrész végrehajtódik, ha nem, nem. Mielőtt példát mutatnánk rá, gyorsan szögezzük le:

Valamennyi szerkezet, amely vezérlésátadást tartalmaz (tehát a feltételes és ciklus
képző utasítások), csakis definícióban használható!
Ha legegyszerűbb példának okáért meg szeretnénk vizsgálni, hogy a verem tetején 6 van-e, és ha igen, kiíratjuk a „Helyes" szöveget, akkor a következő szót kell elkészíteni:

: HAT? 6 = IF ." Helyes" THEN ;

Ebben a definícióban a verem kiértékelődik, mint azt már láttuk. Ha a feltétel igaz, akkora a „Helyes" felirat megjelenik. Ha a feltétel hamis, semmilyen tevékenység nem történik.

Írjunk egy olyan szót, amely ha 0 és 9 közötti számot talál a vermen, kiírja a képernyőre: EGYJEGYU . Természetesen felhasználjuk az előző fejezetben definiált 1JEGY (n - - - - f) szavunkat. Az új szó veremhatása: (n - - - -)

: EGYJ 1JEGY IF ." EGYJEGYU" THEN CR ;

Más magas szintű programnyelvek a THEN-t teljesen másképp használják; mielőtt hagynánk magunkat megkeverni, legjobb, ha az eszünkbe véssük: ez más, a THEN itt ENDIF-et jelent! Az IF-et és THEN-et nem használhatjuk egymás nélkül.

Az ELSE
Ha nemcsak egy adott feltétel teljesülésekor vannak teendőink, hanem az ellenkező esetben is, akkor ilyesféleképpen építjük a programunkat.

IF (itt van, amit akkor kell tenni, ha a jelző igaz);
ELSE (itt van, amit akkor kell tenni, ha nem)
THEN

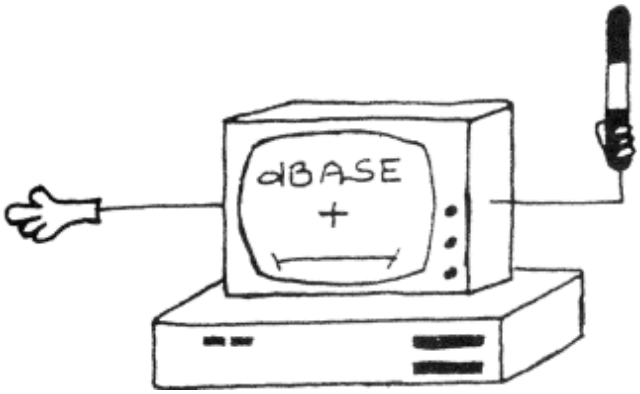
Például:

: HAT? 6 = IF ." Helyes" ELSE ." Nem helyes" THEN ;

: EGYJ 1JEGY IF ." EGYJEGYU" ELSE ." NEM EGYJEGYU" THEN CR ;

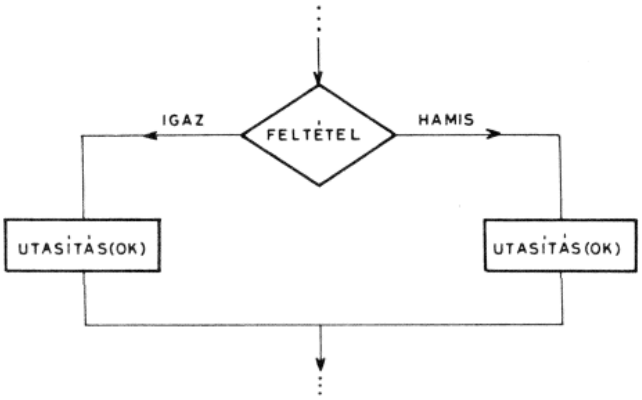
Folytatjuk!

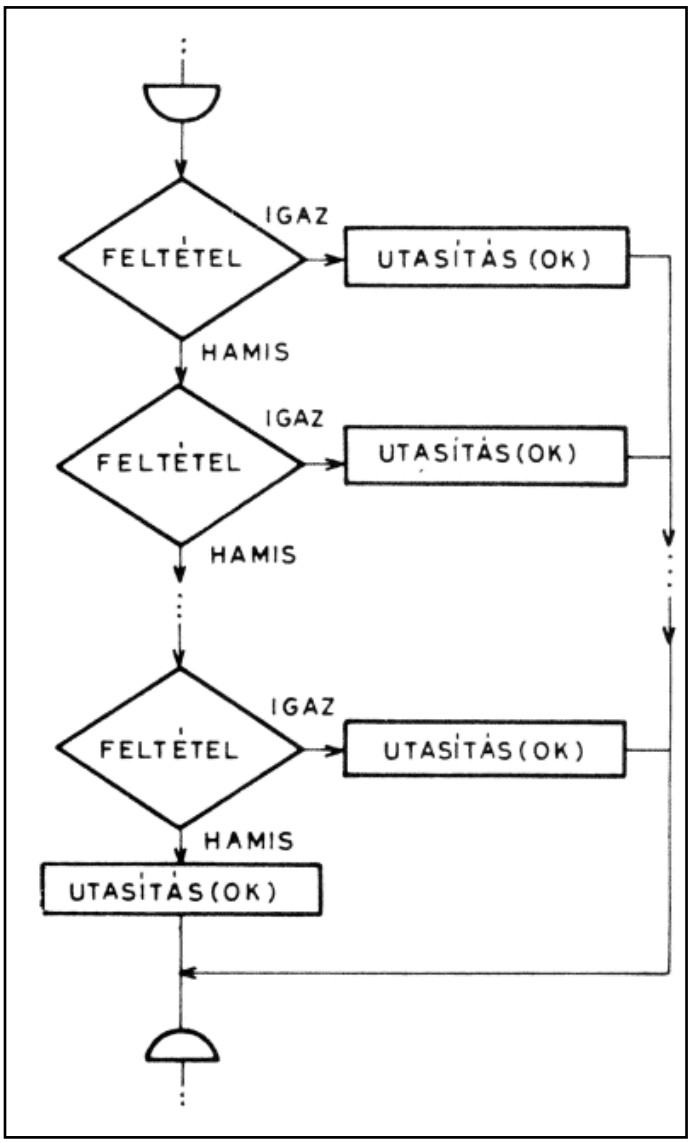
dBase II. 2.43 (IS-DOS) – VI. rész



10.3. Vezérlési szerkezetek

Az eljárásokban fontos szerepet kapnak az ún. vezérlési szerkezetek, amelyek használata minőségi különbséget jelent egy „program" és a párbeszédes parancsok egymás utáni végrehajtása között. Ezekkel a programnyelvi utasításokkal elérhető, hogy a különböző lehetőségek bekövetkezésekor, futás közben dőljön el, hogy pontosan





milyen parancsokat hajtson végre a rendszer.
A dBASE II magasszintű programozási nyelvnek tekinthető, ezt bizonyítja az is, hogy rendelkezik olyan utasításokkal, amelyek ezeket a szerkezeteket létrehozzák és irányítják. (Az ilyen jellegű utasítások általában több részből állnak, így párbeszédes módban nem is használhatók.)

10.3.1. Döntési szerkezetek

Ezekben a programozási szerkezetekben valamilyen feltétel teljesülésétől függnek a továbbiakban végrehajtandó utasítások.

Egyszerű feltételes szerkezet:
A végrehajtás menete jól szemléltethető a blokkdiagramján. A szerkezetet létrehozó parancs formája:

```
IF<feltétel>
   <parancsok>
ELSE
   <parancsok>
ENDIF
```

IF ág:
a feltétel igaz volta esetén végrehajtandó parancsok

ELSE ág:
a feltétel igaz volta esetén végrehajtandó parancsok

Ha az „ELSE" ágat elhagyjuk, hamis feltétel esetén azonnal az ENDIF utasítást követő parancsra lép a dBASE, és azt próbálja meg végrehajtani. Mindkét ágra tetszőleges utasítások írhatók, akár újabb IF szerkezet is, de

vigyázzunk arra, hogy minden megnyitott IF-hez tartozzék egy ENDIF is!

Esetszétválasztás:
Néhány programnyelvben ezt „kapcsolószerkezetnek" is nevezik; több felsorolt feltételből kikeresi az első igaz értékűt, és a hozzá tartozó parancso(ka)t hajtja végre, majd kilép a szerkezetből, az azt követő utasításra. A szerkezet dBASE nyelvű megvalósítása:

```
DO CASE
  CASE <feltétel>
    <parancs(ok)>
  [CASE <feltétel>
    <parancs(ok)>]
  [OTHERWISE
    parancs]
ENDCASE
```

Csak egyetlen (az első igaz feltételű) CASE parancsait hajtja végre, utána kilép az ENDCASE utasítást követő parancsra. (A CASE ágak száma tetszőleges.) Az OTHERWISE után felsorolt parancsokat akkor hajtja végre, ha egyetlen feltételt sem talált igaznak. Ha nem használunk OTHERWISE részt, az ENDCASE után folytatódik a végrehajtás.

10.3.2. Ciklusszervezés
A ciklus egy utasításcsoport ismételt végrehajtása, amíg egy adott ún. ciklusfeltétel teljesül. Amikor a ciklusfeltétel hamissá válik, befejeződik a ciklus, és a ciklust követő paranccsal folytatódik a programvégrehajtás. A ciklust létrehozó utasításokat a következő formában kell a programszövegbe írni:

```
DO WHILE <feltétel>
  <parancs(ok)>
ENDDO
```

Ez egy ún. elöltesztelt ciklus; a parancsok első végrehajtása előtt is ellenőrzi a feltételt. Ha már ekkor hamisnak találja, egyszer sem végzi el a ciklusmag utasításait. A ciklusok is „egymásba ágyazhatók", azaz a ciklusmag tartalmazhat egy másik ciklust, de azt is be kell fejezni az ENDDO utasítással. Elméleti szempontból fontos, hogy ha a programvégrehajtás belépett egy ciklusba, akkor a ciklusvégen is szigorúan át kell haladnia. Soha ne írjunk ciklusmag belsejébe RETURN parancsot. Programunkat egy program is elindíthatja, ilyenkor a RETURN hatására a hívott program befejeződik, de lezáratlanul marad a ciklus, és a hívó program ciklusai ettől zavarba jöhetnek.

A dBASE nyelve rendelkezik egy paranccsal, amelyek lehető ad számunkra, hogy a megkezdett ciklus befejezését a normális végrehajtás menetétől függetlenül is kezdeményezhessük. A

```
LOOP
```

parancs hatására a ciklusmag további parancsait figyelmen kívül hagyva, azonnal a ciklusfeltétel újbóli kiértékelését végzi el a rendszer. Ha előtte a ciklusfeltételben szereplő változó(k) értéke megváltozott, akkor ez a ciklus befejezését is jelentheti. Gyakran használjuk olyan helyzetekben is, amikor szeretnénk elölről elkezdeni a ciklusmag végrehajtását. Az utasítás az utoljára elkezdett - legbelsőciklusra vonatkozik, azaz annak feltételét tudjuk újra kiértékeltetni. A külső, ezt tartalmazó ciklusokat az utasítás nem befolyásolja.

10.4. Változók a programokban
Programírás közben állandóan szükségünk van különböző tárolóhelyekre, ahol adatainkat ideiglenesen el tudjuk helyezni; mint már az alapfogalmaknál említettük, ezeket a tárolóhelyeket szokás változóknak nevezni. Programot írni változók használata nélkül szinte lehetetlen, mert így csak állandó értékű adatok szerepelhetnének az utasításokban, azaz minden alkalommal ugyanazt a tevékenységet végezné el a program, minden futtatás alkalmával pontosan ugyanazt az eredményt kapnánk. Az ilyen utasítássorozatot csak erős túlzással lehet számítógépes programnak tekinteni.
Programjainkban ugyanolyan változókat használhatunk, mint párbeszédes módban, azaz memóriaváltozókat és adatbázismezőket. Az eddig ismertetett, változókat kezelő parancsokon kívül rendelkezésünkre áll még néhány utasítás, amelyek egy részét nincs értelme programon kívül alkalmazni, de amelyek léte ékesen bizonyítja, hogy a változóknak (és itt most kivételesen elsősorban a memóriaváltozókat értjük ezen) a programírásban van elsődleges szerepük.

10.4.1. Memóriaváltozók használata programban
Először foglalkozzunk a memóriaváltozók programon belüli alkalmazásával! Ez több olyan új parancsot jelent, amelyek használatával nagyon kényelmesen és „egyszerűen" programozhatjuk adatbázis-kezelőnket. A memóriaváltozók definícióját- programon belül is az értékadó utasítások végzik. Az eddig megismert parancs (a STORE) mellett azonban használhatunk olyan utasításokat is, amelyek a program felhasználójától fogadnak adatot, és azt egy megadott memóriaváltozóban helyezik el. Az ilyen változók pontos értéke minden futtatásnál más és más lehet, értéküket a billentyűzeten begépelt kifejezés határozza meg.

Tetszőleges típusú adat bekérésére alkalmas a következő, interaktív dódban is használható (de legalább kipróbálható) parancs:

```
INPUT [<üzenet>] TO <memóriaváltozó>
```

Az „üzenet" egy karakteres kifejezés lehet, általában valamilyen magyarázószöveg, mellyel emlékeztetjük a felhasználót arra, hogy milyen jellegű adatot vár a program (ha ez egy állandó karaktersorozat, akkor a határolójelek közé kell tenni). Az „üzenet" a parancs végrehajtásának első lépéseként megjelenik a képernyőn (az utoljára használt sort követő sor első pozíciójától kezdve), és utána (a kurzortól kezdődően) gépelhetünk be egy tetszőleges, dBASE által értelmezhető kifejezést, a adott nevű memóriaváltozó ennek típusát és értékét veszi fel. Tekintettel arra, hogy kifejezést vár a parancs, a karakteres típu-

sú adatok határolójelek között gépelve értékelődnek ki helyesen.
Próbaként gépeljük be a következő parancsot:

INPUT „Kerek begepelni egy szot: „ TO szoveg

Hatására a képernyő következő sorában megjelenik az, általunk megadott üzenet, és mögötte a kurzor; ide gépelhetjük be például „PITE" szócskát mint karakteres adatot (idézőjelekkel együtt). A kifejezés begépelését az ‚ENTER' billentyű megnyomásával kell befejezni. Egy egyszerű karakterlánc-állandó helyett begépelhető kifejezést is:
INPUT ‚Kerek egy kifejezest:' to szoveg2
Kerek egy kifejezest:"ALMAS"+szoveg
? szoveg2 ALMASPITE

Kifejezetten csak karakteres adatok bevitelére alkalmas az

ACCEPT [<üzenet>] TO <memóriaváltozó>

parancs. Míg az INPUT parancs által létrehozott memóriaváltozóról csak futási időben dől el, hogy milyen típusú, addig az ACCEPT parancs a memóriaváltozót mindenképpen karakteres típusúként hozza létre, és begépelt karaktereket „betű szerint" helyezi el benne. Működési módja miatt nem kell határolójeleket használni a begépeléskor, illetve ha mégis kiírjuk ezeket, akkor a szöveg határolójelekkel együtt kerül a változóba.
Mindkét parancs a végrehajtáskor egy új memóriaváltozót hoz létre, illetve ha előzőleg már létezett ilyen nevű memóriaváltozó, akkor azt felülírja. A memóriaváltozó a típusát, hosszát, értékét mind az utasítás végrehajtásakor kapja. Az adat, illetve kifejezés begépelésének végét mindkét parancs esetén az ‚ENTER' billentyű kell jeleznünk. Amíg ez nem történt meg, a program „áll".

A következő parancs is adatbevitelt valósít meg, de speciálisan egyetlen billentyű megnyomására vár, és amikor ezt megtettük, azonnal engedi tovább a program futását:

WAIT [TO <memóriaváltozó>]

Paraméter nélkül használva a parancs felfüggeszti a programvégrehajtást (a képernyőn erről a WAITING felirat tájékoztat), amíg a felhasználó egy tetszőleges billentyű megnyomásával nem jelzi, hogy folytatható az utasítások kiértékelése. A folytatáshoz a vezérlőbillentyűk (pl. a ‚SHIFT') kivételével bármelyik billentyűt megnyomhatjuk.
Ha beírjuk a parancsba a „TO <memóriaváltozó>" paramétert is, a lenyomott billentyűnek megfelelő karakter bekerül a megadott nevű, egy byte hosszúságú memóriaváltozóba. Ha ezt a paramétert nem használjuk, a karakter nem tárolódik, utólag nem lehet lekérdezni, hogy milyen billentyű hatására folytatja a dBASE a programvégrehajtást.
Még egyszer hangsúlyozzuk, hogy míg az előző két parancs végrehajtásakor az ‚ENTER' billentyű megnyomása jelzi az adatbevitel befejezését, addig a WAIT parancsát az első leütött billentyű hatására már befejeződik az utasítás végrehajtása.

Az ismertetett három parancs egyike sem ad lehetőséget arra, hogy adatbevitel képi formáját befolyásoljuk, ennek módjaira a „Formátumozott adatbevitel" című fejezetben térünk ki.

10.4.2. Adatbázismezők tartalmának módosítása programból

Ha nem is hangsúlyoztuk, eddigi példáinkból érezhető, hogy, az adatbázismezők és a memóriaváltozók egy lényeges tevékenység szempontjából nagyon különböznek egymástól. Ez pedig az értékadás. Ahhoz, hogy programjainkban tudjuk módosítani a mezők tattalmát, mégpedig anélkül, hogy a felhasználó kezébe helyeznénk minden ellenőrzési lehetőséget az EDIT, BROWSE vagy hasonló parancs segítségével, szükségünk lesz egy olyan utasításra, amely valamilyen változó vagy kifejezés értékét tudja elhelyezni egy adott mezőben. Ezt valósítja meg a következő parancs:

REPLACE [<érvényességi kör>] <mezőnév> WITH <kifejezés> [,<mezőnév2> WITH <kifejezés2> ...] [WHILE <feltétel>] (FOR <feltétel>]

Ez a parancs az érvényességi körébe eső rekordok (alapértelmezés szerint az aktuális rekord) megadott mezőinek tartalmát cseréli ki a megadott kifejezések értékével. Az összetartozó mezők és kifejezések típusának egyezni kell. Mezőnévként bármely nyitott adatbázis mezőjét megadhatjuk. Indexelt állományban az indexállomány minden egyes rekordon elvégzett módosítás után aktualizálódik. Ezért az indexkulcs mezőjében történő módosítás alkalmával a FOR és WHILE paraméter használata nem ajánlatos, mert az azonnali átrendeződés miatt bizonyos rekordok kimaradhatnak a műveletből.

Ha ezzel a paranccsal egy új rekordot szeretnénk kitölteni, akkor azt a rekordot előzőleg létre kell hozni. Erre szolgál az

APPEND BLANK

parancs, mely az adatbázishoz fűz egy új rekordot, és a rekordmutatót erre állítja, ezért az ezt követően kiadott REPLACE parancs, érvényességi kör nélkül megadva, ennek mezőit tudja módosítani.

Lehetőségünk van arra is, hogy egy teljes adatbázist aktualizáljunk másik, adatbázis-állomány segítségével, felhasználva a két állomány azonos tartalmú mezőit. A kulcs szerint megegyező rekordokban történik meg a módosítás. A parancsot az alábbi formában kel megadni:

UPDATE FROM <adat-állomány> ON <kulcsmező> ADD <mezőlista> vagy REPLACE <mezőlista>

Az adatbázis-állományok mezői is mint változók alkalmasak a felhasználói adat fogadására, erre azonban csak a formátumozott adatbevitelt megvalósító parancs használható (az „interaktív" APPEND, BROWSE és társaiktól eltekintve), ennek alkalmazását ismertetjük` következő fejezetben.

10.5. Formátumozott adatbevitel, adatmegjelenítés

Egyik eddig ismertetett adatbeviteli parancs sem adott lehetősé arra, hogy az adatbeviteli képernyő megjelenési formáját befolyásolhassuk. Az ún. formátumozott adatbevitelt megvalósító parancsok használatával egy egyszerű APPEND parancs „semmitmondó megjelenési formáját is megváltoztathatjuk, ily módon a program „felhasználó-barátsága" nő. A képernyőn nem csupán a kurzor vagy az adatbázismezők puszta neve jeleníthető meg, hanem általunk meghatározott magyarázó és segítségnyújtó szövegek is.
A formátumozott adatbevitel megvalósításához végeredményben egyetlen parancs ismerete elegendő. A parancs, amely a felhasználó által definiált formában fogadja billentyűzetről vagy jeleníti meg a képernyőn az adatokat, az adatbekérés vagy -megjelenítés pontos helyét is befolyása. A parancs teljes felépítése:

@ <oszlop>,<sor> [SAY <kifejezés> [USING <formátum>]]
@ <oszlop>,<sor> [SAY <kifejezés>] [GET <változó> [PICTURE <formátum>]]

A „sor" és „oszlop" paraméterek a kurzort pozícionálják, a „SAY" paraméter megjeleníti a kifejezés értékét, a „GET" paraméter az adott változó tartalmát ajánlja fel módosításra, a „PICTURE" paraméterrel lehet befolyásolni a megjelenítendő vagy tárolandó adat formáját és tartalmát.
A parancs összetettségére való tekintettel egyesével, példákkal illusztrálva tárgyaljuk a paramétereket; hogy ezek közül melyek hagyhatók el, illetve melyek használhatók együtt, azt a parancs szintaktikájából leolvashatjuk (a [] zárójeleket figyelembe véve). Érdemes kipróbálni interaktív módban a parancs formáit (közben néha le kell törölni a teljes képernyőt az ERASE paranccsal).

A „sor" és „oszlop" paraméterek helyezik a kurzort a képernyő megadott sorába, illetve oszlopába. Mindkét paraméter megadható numerikus változóval is, de ügyeljünk a korlátokra. A képernyőn 25 sorban és 80 oszlopban helyezhetők el karakterek. Példa a pozícionálásra:

@ 4,5

A „SAY" paraméter használatával egy kifejezés értékét tudjuk megjeleníteni, esetleg más formában, mint ahogy a kifejezésben található adatokat tároljuk (például a kisbetűs szavakat megjeleníthet nagybetűsként, anélkül hogy valójában módosítanánk a tárolt adatot). A kifejezésbe bármely nyitott adatbázis mezője beírható, de ez csak aktuális rekord mezőjét jelenti. Hangsúlyozzuk, hogy egyetlen kifejezés írható a paraméterbe, így több adat megjelenítéséhez különböző „trükkökre" lesz szükségünk. Példaként tekintsük a következő parancsot:

@ 4,5 SAY ‚Az aru ara forintban: ‚+STR(ar,7,2)

Hatására a rendszer a megadott pozíciótól kezdve kiírja a határolójelek közé tett szöveget, és mögéje az aktuális rekord „ar" nevű mezőjének, tartalmát. (Az STR() konvertáló függvény a típusegyezés miatt kell, ennek második és harmadik argumentuma a keletkező karakterlánc hosszát, illetve a numerikus adatból az átalakítandó tizedesjegyek számát adja meg.) Ha egyetlen numerikus típusú kifejezés érti szeretnénk megjeleníteni, nem kell használni a különböző konvertáló függvényeket, de amikor ezeket karakteres adattal keverve íratjuk akkor már igen.

A „GET" paraméter segítségével adatokat tudunk a billentyűzetről bekérni. Mégpedig olyan kényelmes formában, hogy a megadott változó jelenlegi értéke megjelenik a képernyőn és - karakteres típusú adat esetén - azt a megszokott szerkesztőbillentyűkkel javíthatjuk olyanra, amilyent tárolni szeretnénk. A „PICTURE" paraméter használatával bizonyos mértékben korlátozhatjuk a bevihető adat formáját és tartalmát. A „GET" paraméterben megadott változónév egyaránt lehet memóriaváltozó és adatbázismező neve. A szerepeltetett változónak léteznie kell, ez memóriaváltozó esetében azt jelenti, hogy a beviteli parancs előtt valamilyen értékadó utasításban kell szerepeltetni, adatbázismező esetében pedig gondoskodnunk kell a parancs kiadása előtt, hogy a rekordmutató az aktív adatbázisnak egy rekordjára mutasson (az EOF() függvény ne legyen igaz értékű), és az adott táblázatban legyen olyan mező, melynek a megadott neve van.
A „@ ... GET" parancs az adatbevitelt még csak előkészíti; definiálja, hogy hol és mely változókba szeretnénk adatot bekérni. A tényleges adatbevitel engedélyezését (vagy ahogy másképpen szoktuk nevezni, a változók" aktivizálását) egy másik parancs, a

READ

végzi. Hatására a kurzor megjelenik az első változó kezdő pozíciójában. A parancs az előtte szereplő összes olyan „get változót" aktivizálja, amelyeket egy előző READ parancs még nem olvasott be. Az „összes" nem értendő szó szerint, ennek is van egy korlátja, mely számszerint 64. Ezt a 64 darab „get változót" célszerű egy oldalra zsúfolni, különben egyesek magyarázó szövegei már letörlődtek, és csak a puszta változó jelenik meg jelenlegi értékével. Az adatbevitel akkor ér véget, ha az utolsó változót is kitöltöttük, és leléptünk róla valamilyen módon (a kurzormozgató, illetve az ‚ENTER' billentyű megnyomásával). Ameddig a kurzor valamelyik változó „mezőjében" látható, a szerkesztőbillentyűkkel bármilyen irányban mozoghatunk, például egy előzőleg kitöltött változót javíthatunk.
A READ parancs nélkül a „@ ... GET" parancsok hatására ugyan megjelennek a változók a képernyőn, de a program továbbfut, nem ad lehetőséget arra, hogy adatainkat begépeljük.
Numerikus változóba való adatbekérés esetén a dBASE csak számjegyeket, előjelet és/vagy tizedespontot fogad el. Minden egyéb billentyű megnyomásakor figyelmeztető hangjelzést ad a rendszer, és képernyőre nem ír ki újabb karaktert. Példa két változó beolvasására:

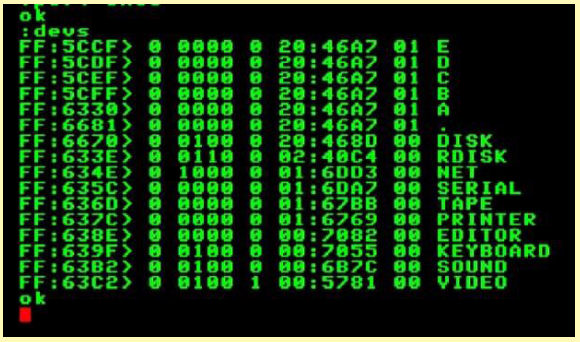
```
@ 2,2 SAY „Csaladnev:" GET CSNEV
@ 3,2 SAY „Keresztnev:" GET KNEV
READ
```

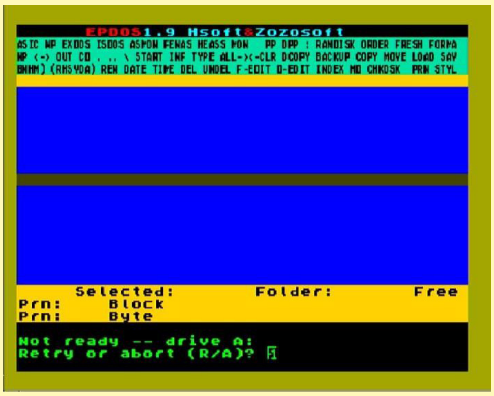
Folytatjuk!

Az Enterprise 128 megvalósítása FPGA-n keresztül

Jó híreink vannak! A fejlesztés nagyon jól halad! A kis fejlesztő csapat már a lemezkezelésnél tart!

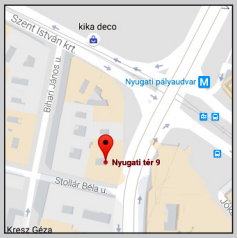
Néhány képernyő kép, melyek már az új rendszeren készültek.





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

https://enterpriseforever.com

ENTERPRESS Magazin - 2020/3-6. május-december
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: https://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
https://enterpress.news.hu/index.php/magazin