ENTERPRESS

Magazin az ENTERPRISE felhasználóknak

2022/1-2. január–április

35.

JUBILEUM

Hazánkba 1987-ben jutott el az Enterprise 128 K-s változat: 1987. május 19-én a Magyar Divatcsarnok Lotz-termében mutatták be.

35 éve Magyarországon

Írta: Matusa István (Tutus)

Ahogy ünnepi címlapunkon is láthatjátok, 35 éve 1987. május 19-én a Magyar Divatcsarnok Lotz termében mutatták be először az Enterprise 128-as gépet hazánkban!

Nagy idő ez, és fantasztikus látni azt, hogy kis közösségünk nem csak hazánkban, hanem az egész világon igen aktív!

A kezdeti nehézségek ellenére felkaroltuk ennek a számítógépnek a supportját. Azt, hogy milyen sikerrel, szerintem kár taglalni, hiszen ez továbbra is (sőt egyre jobban!) mindenkinek egy hobbi. Elmondhatjuk, hogy a 35 év alatt sok hardver kiegészítő és szoftver is készült az Enterprise gépre, mely (igen, 35 év után is!) még mindig működik!

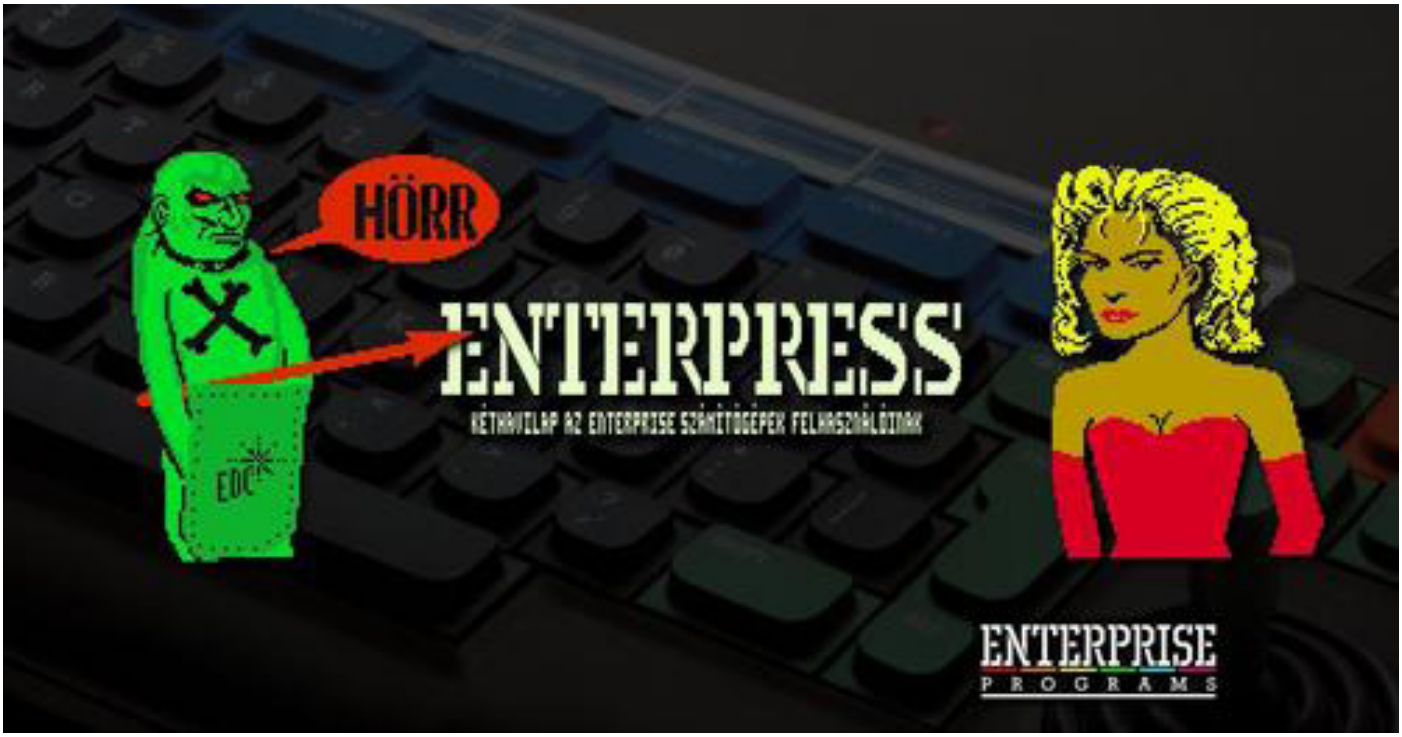
Ebből az alkalomból szerveztünk egy jubileumi klubtalálkozót 2022. május 21-én.
Sajnos nem úgy sikerült minden, ahogy elterveztem! A főállású munkám miatt például nem tudtam erre az időpontra megjelentetni a jubileumi Enterpress Magazint (amit most olvasol).
Sok más meglepetés és ötlet is volt, de sajnos ezek sem valósultak meg. A Nyugatinál mindkét terem a miénk volt erre az alkalomra. Sajnos nagyon kevesen voltunk, így csak egyik termet használtuk... Erre azért nem gondoltam volna, hogy csak annyien leszünk... Ketten is jelezték előtte, hogy fontos elfoglaltság miatt nem tudnak jönni, ez így rendben is van, érthető. De hol voltak a többiek?

Bízom benne, hogy kis közösségünk lelkesedése az elkövetkező években is töretlen lesz és sok örömünk lesz még kedvenc számítógépünkben, az Enterprise 128-ban!

2022 őszén folytatjuk, terveink szerint 3 klubtalálkozó lesz.

Addig is minden Kedves Olvasónknak jó nyaralást és pihenést kívánunk!

Enterprise Forever!



Interjú Peter Hinerrel

a ZZZIP alkotójával

- Örülünk, hogy az Enterprise csoport tagjai között láthatunk, és hogy kedvenc számítógépünk megjelenése után sok évvel Enterprise Forever fórumon üdvözölhettünk. Hogy találtál rá az Enterprise Forever fórumra?

Gondolkodtam a korai számítógépes éveken, és elkezdtem keresgélni a Google-on keresztül – hirtelen rábukkantam az Enterprise Forever oldalra. Bejelentkeztem és meleg fogadtatásban részesültem – ez nagyon kedves volt.

A következő néhány kérdésre most egy folyamatos történetként írok választ, mert szerintem egy folyamba illenek. Íme a kérdések:

- Mikor és hogyan találkoztál először az Enterprise számítógéppel?

- Volt más számítógéped az Enterprise előtt?

- Milyen számítógépeket ismertek az országodban, az Egyesült Királyságban a 80-as és 90-es években?

- Hogyan jött az ötlet, hogy megírd a Zzzip Compiler-t?

- Mennyi időbe telt a Zzzip megírása?

Nem az Enterprise volt az első számítógépem – az első számítógépem egy Compucolor II volt, amelyet az Intelligent Systems Corporation, egy egyesült államokbeli cég gyártott. Abban az időben az Egyesült Királyságban meglehetősen sok számítógép volt elérhető, az apró Sinclair ZX-től a közepes méretű TRS-80-ig és a Commodore Petig, de mindegyik fekete-fehér (vagy zöld-fehér) képernyővel rendelkezett. Szóba került, hogy hamarosan színes számítógépek jelennek meg, ezért úgy döntöttem, várok egy kicsit. Végül megérkeztek a színes számítógépek (az 1970-es évek végén), és választhattam: vagy a Compucolor II-t vagy az Apple II-t. Számomra kézenfekvőnek tűnt a Compucolor választása, mert sokkal jobb volt a kijelző minősége. Persze utólag visszagondolva nem biztos, hogy ez volt a jó választás, mert a Compucolor cég pár év múlva csődbe ment. Ez azonban nem volt teljesen rossz számomra, mert azt tapasztaltam, hogy az Egyesült Királyság felhasználói csoportja nagyon aktívvá vált, miután a gyártó részéről a támogatás megszűnt. Az USA-ban, Kanadában és Ausztráliában is működtek felhasználói csoportok, és hamarosan szoros kapcsolatot alakítottunk ki velük, hírleveleket és szoftvereket cseréltünk, stb.

A Compucolor monitorja egy beépített floppy lemezmeghajtót tartalmazott (egy speciálisan a Compucolor-hoz készített kijelző), és 5.25-ös floppy lemezt használt, körülbelül 50 KB tárolókapacitással. A számítógépnek volt egy verziója 16 KB RAM-mal, de emeltem a tétet, és a 32 KB RAM-ra bővített verziót vettem meg!

Tehát ez volt az a számítógép, amelyen megtanultam a BASIC-et, a szokásos módszerrel, amelyet akkoriban mindenki használt. Nyomtatott magazinokból egyszerű programokat, soronként gépeltünk be, ami sokáig tartott és sok hibát eredményezett. Aztán megpróbáltuk működésre bírni - ez részben a hibák kijavítását, részben a BASIC utasítások módosítását jelentette, hogy az megfeleljen az egyes számítógépek speciális követelményeinek (minden számítógépen volt némi eltérés a BASIC nyelvben). Így tanultunk meg mindannyian BASIC programokat írni.

Természetesen sok boldog órát is eltöltöttem olyan játékokkal, mint a Space Invaders, Pacman stb. Ezeket a játékokat assemblyben írták, majd gépi kódba fordították be, így elég gyorsan futottak. A BASIC nyelven írt programok sokkal lassabbak voltak - mint az Enterprise-on is. Így nemsokára elkezdtem tanulni, hogyan kell programot írni Assemblyben.

Az egyik korai próbálkozásom a Colossal Cave Adventure, az első klasszikus szöveges kalandjáték Compucolor

verziója volt. Sok szempontból ez egyszerűen egy már létező játék másolása volt, de számomra az volt a kihívás, hogy olyan verziót készítsek, amely nagyon hasonlít az eredeti játékhoz, de 16 KB RAM-mal rendelkező gépen is fusson (bár a gépemben 32 KB RAM volt, sok másnak csak 16 KB-os gépe volt). Úgy értem el, hogy listát készítettem a 255 leggyakoribb szóból a játékban, melyekre egyetlen 8 bites szám hivatkozhat. A fennmaradó számot (nulla) használtam a szólistára való hivatkozásról egy kevésbé gyakori szó kiírására való változás jelzésére (és ezt a nulla számjegyet egy másik számjegy követte, hogy jelezze, hány betű lesz, mielőtt visszatérnénk a szólistához). . Ha belegondolsz, rájössz, hogy nem kell mind a 8 bit az ábécé összes betűjének (csak a nagybetűk) kiírásához. Ezt 5 bittel tudtam megtenni, így tovább tudtam tömöríteni az írást, ez a 8 bites bájt határok átlépését jelentette. Ezt a tömörítési módszert még a 255 gyakori szót tartalmazó listához is alkalmaztam. Néhány évvel később, amikor megkaptam az Enterprise számítógépet, elkészítettem a Colossal Cave Adventure új verzióját, de addigra a technológia nagyot fejlődött, és már nem volt csodálatos ezt a játékot személyi számítógépen futtatni. Valójában voltak már jobb szöveges kalandjátékok a Level Nine-tól.

Hosszan beszéltem a Compucolorról, mert ez volt az utam az Enterprise felé, és mert ez volt a ZZZIP, a BASIC fordítóm eredete is. Már mondtam, hogy elkezdett érdekelni az Assembly nyelv, és kíváncsi voltam, hogyan működik a BASIC interpreter. Így hát visszafejtettem a BASIC ROM tartalmát, és kinyomtattam - ez sok órát vett igénybe, és körülbelül 10 cm vastag papírkupacot lett az eredménye.

Évekig tanulmányoztam ezt a listát, és végül azonosítottam a matematikai rutinok (például összeadás, kivonás stb.) belépési pontjait, valamint olyan függvényeket, mint a SIN, COS, TAN stb. Azon tűnődtem, hogy lehetséges lenne-e használni ezeket a függvényeket egy Assembly programban – és sok munka után készítettem egy egyszerű programot, amely elfogadja a numerikus bemenetet, feldolgozza azt a BASIC ROM-ban lévő függvényen keresztül, majd létrehoz egy numerikus kimenetet. Bemutattam ezt a programot a felhasználói csoport találkozóján, és az egyik barátomat nagyon érdekelte. Valójában becsületet kell tulajdonítanom neki, amiért arra késztetett, hogy többet és többet tegyek. Végül elkészítettem az első BASIC fordítómat, ami körülbelül 3-5-szörösére gyorsította fel a BASIC programokat - nem túl nagy gyorsulás, mert sok időt töltött a BASIC ROM-ban lévő rutinok futtatásával, de ennek ellenére gyorsabb volt, mint az eredeti BASIC program. Eladtam ezt a programot az Egyesült Királyságban, az Egyesült Államokban, Kanadában és Ausztráliában működő felhasználói csoportok tagjainak, akik nagyon örültek a gyorsulásnak. Nem kértem sokat a programért, de nagyon népszerű volt, így elég pénzt kerestem egy új nyomtató vásárlásához!

Hamar rájöttem azonban, hogy a BASIC ROM-ban a lebegőpontos matematikai rutinok használata megakadályozza a sebesség jelentős növekedését, ezért készítettem egy második verziót, amely csak egész számokat használt. Ez természetesen azt jelentette, hogy bizonyos matematikai függvények már nem lesznek megvalósíthatók (vagy nagyon pontatlanok), de az olyan programok esetében, mint a játékok, az egész számok használata elég jó volt – és 50-szeres sebességnövekedés volt szükséges egy játékhoz.

Körülbelül ekkoriban ismerkedtem meg az Enterprise-szal. Felhasználói csoportunk különböző tagok otthonában találkozott, egyikük egy fiatal fiú volt (aki később sikeres játékprogramozási vállalkozást épített fel). Ennek a fiatal fiúnak az apja nagy érdeklődést mutatott felhasználói csoportunk iránt, és számos találkozónkra eljött. Rádióműsor-producer volt, a BBC-nek dolgozott, és az általa készített sorozatok egyike a személyi számítógépekről szólt. Egy nap azt mondta nekem, hogy kapcsolatba lépett az Enterprise embereivel, és megemlítette a BASIC fordítómat. Érdekelte őket, és meg akarták beszélni velem – ezért felmentem az Enterprise londoni irodájába, és volt egy találkozóm a szoftvermenedzserrel. Felajánlott nekem egy Enterprise számítógépet féláron, ha a BASIC fordító egy verzióján dolgozom, amelyet esetleg minden eladott Enterprise számítógéphez adnak. Így hát elkezdtem, amiből idővel ZZZIP lett.

Természetesen a ZZZIP korai verziói meglehetősen egyszerűek voltak, de amikor megmutattam első erőfeszítéseimet az Enterprise embereinek, a reakciójuk biztató volt. Ha jól emlékszem, a két ember, akivel leggyakrabban találkoztam, Keith Elliott és Steve Groves volt. Mivel az Enterprise irodákban dolgoztak (úgy emlékszem, hogy a londoni Goodge Street közelében valahol), feltételezem, hogy marketingesek voltak – az egyik a hardverért, a másik a szoftverért volt felelős, bár nem emlékszem, melyik volt melyik. Rendszeresen jártam az Enterprise irodáiban, és megismerkedtem a programtesztelő csapat tagjaival is – különösen emlékszem, hogy gyakran adtak nekem kipróbálásra új játékok korai verzióit. Nem emlékszem pontosan, mennyi időbe telt, amíg a ZZZIP az elejétől a végső verzióig eljutot - talán néhány év -, de végül készen állt a kiadásra - és ugyanabban a pillanatban az Enterprise csődöt jelentett.

Az Egyesült Királyságban már létrehoztunk egy Enterprise felhasználói csoportot, amely most sokkal aktívabbá vált, mivel a gyártó támogatása megszűnt. Akkor még nem tudtuk, mi történt az eladatlan számítógépek, szoftverek és kiegészítők készletével (bár az előzményeket természetesen jóval később olvastam). Csak tovább dolgoztunk, és mint felhasználói csoport támogattuk egymást. Nem hiszem, hogy eladtam bármennyi szoftvert az Egyesült Királyság felhasználói csoportjának (mert a britek túl szűkmarkúak voltak ahhoz, hogy pénzt költsenek ilyesmire), és az Egyesült Királyságon kívüli felhasználói csoportokkal sem volt kapcsolatunk. Amúgy soha nem számítottam arra, hogy sok pénzt keressek azzal, hogy megírom a fordítóprogramomat – ez pusztán a szeretet eredménye volt. Nagy örömömre szolgált, hogy megírtam az egykor a Compucolor-on élvezett játékok BASIC verzióit – majd ZZZIP-pel befordítottam, hogy gyorsan futhassanak. Ezek közé a játékok közé tartozott a Pacman és a Space Orbs – plusz a Tetris egy verziója (amely a Compucolor után jött). És azt hiszem, elérkeztünk az Enterprise történetem jelenéhez.

- Mivel foglalkoztál az Enterprise után?
Az Enterprise után vettem egy IBM PC-t – a pontos modellre nem emlékszem (a 486-os szám ismerősen hangzik), de sokkal gyorsabb volt, mint az Enterprise, sokkal több RAM-mal és sokkal nagyobb lemezmeghajtóval. Természetesen az egyik első dolog, amit megpróbáltam megvizsgálni, az operációs rendszer volt. Az első lépés a visszafejtés volt – nem próbáltam azonnal kinyomtatni egy listát, mert azt gondoltam, hogy elég nagy, és rengeteg papírt igényel. Szóval csak egy kisebb részt fejtettem vissza, és az eredményt a lemezmeghajtón lévő fájlba írtam. Arra számítottam, hogy ez a művelet egy órát vesz igénybe, de úgy tűnt, nem történt semmi. A számítógép csak rövid, zúgó hangot adott, és ennyi. Szóval megpróbáltam újra - ugyanaz. Ezután ellenőriztem, hogy mi van a lemezmeghajtón, és két példányt találtam egy nagyon nagy fájlból – a számítógépnek mindössze néhány másodpercre volt szüksége minden példány megírásához. Ekkor rájöttem, hogy a számítástechnika nagyon messze került az Enterprise-tól – a világ egészen más lett.
Elhatároztam, hogy programot írok a PC-re. Korábban írtam egy FishLife nevű Assembly programot (úgy hiszem az Enterprise verzió nálatok is megvan), és úgy döntöttem, reprodukálom PC-re. DOS alatt futott, tehát csak natív gépi kód volt, a Windows-val való együttműködéssel kapcsolatos problémák nélkül. Nos, remekül ment, egy dolgot leszámítva – olyan gyors volt, hogy nem lehetett látni, mi történik. Hatalmas késleltetési hurkokat kellett beillesztenem, hogy a sebességet legalább századára, esetleg ezredére csökkentsem, hogy látható eredményt kapjak!

Később más nyelvekkel is eljátszottam, mint például a Delphi és a Visual Basic, de valójában a programozás felett lejárt az időm. Az egyetlen hasznos dolog, amit létrehoztam, egy adatbáziskezelő program volt, amely a számítógépes magazinok előlapjára csatolt ingyenes CD-kre menti az összes programot. Emlékszel azokra a CD-kre – Magyarországon is voltak? Nos, ez voltak a legjobb ok arra, hogy az Egyesült Királyságban minden hónapban számítógépes magazint vásároljanak. Csak be kellett helyezni az új CD-t a számítógépembe, és le kellett futtatnom az adatbázis-programomat - az automatikusan beolvasta a CD-ről a könyvtárat és berakta az adatbázisba -, majd később kereshettem a kívánt programot.

- Jelenleg mi a civil munkád?
Ó, drágám, nem gondolod, hogy még mindig dolgozom, igaz? Valójában közel 78 éves vagyok, és 58 évesen mentem nyugdíjba, tehát közel 20 éve vagyok nyugdíjas. Már nem írok programokat, de van egy hobbim, ami felkelti az érdeklődésemet és elfoglal – szeretek videófilmeket készíteni, és egy filmes klubhoz tartozom – ez most a hobbim.

- Úgy tudjuk, hogy alapprogramokat (pl. Tetris, Orbs) írtál a Zzziphez. Írtál más programokat is Enterprise-ra?
Írtam még néhány programot az Enterprise-ra – semmi izgalmasat – a legtöbb programom segédprogram volt. Meg voltam elégedve az IDEA programommal (Integrated Disassembler Editor & Assembler), és magam is használtam mindenre, amire szükségem volt – nem hiszem, hogy sokan (sőt senki?) használták. Arra is emlékszem, hogy írtam egy hasznos Fájl-összehasonlító programot, hogy jelezze a változásokat a program egyik verziója és a következő között. Mindenesetre megvan minden, amit írtam a könyvtáratokban – szóval ne remélj többet!

- Tudtál olyan emberekről (az Enterprise Forever közösség tagjain kívül), akik Basic programokat készítettek a Zzziphez (akkor amikor a Zzzip-et írtad)?
Biztos vagyok benne, hogy számos más programot is írtak az Egyesült Királyság felhasználói csoportjának tagjai. Az egyik tag Andy Burnham volt, egy nagyon tehetséges szoftveríró, aki megjelent az Enterprise Forumon. Azonban nem tudok olyan névvel szolgálni, akit még nem ismersz.

- Kit ismertél, aki az Enterprise-al is foglalkozott? Tudtad, hogy pl. Bruce Tanner, az IS-BASIC megalkotója?
Ahogy korábban is mondtam, az Enterprise céggel – más szóval a Marketing csoporttal – tartottam a kapcsolatot. Ezért nem ismertem meg Bruce Tannert vagy a többi fejlesztőt. A fejlesztőkhöz legközelebbi emberek a programtesztelők voltak, akikkel találkoztam – nagyon barátságosnak találtam őket, de nem emlékszem nevekre.

- Mi a véleményed az Enterprise számítógépről és az IS-BASIC-ről?
Az Enterprise hardvere nagyon lenyűgöző volt, hatalmas képességekkel – szem előtt kell tartani a technológia akkori állapotát. A szoftver nem volt annyira lenyűgöző. Az operációs rendszer jó volt - gyorsan futott, megbízható volt, és nem volt benne semmi felesleges, időt és helyet foglaló kiegészítés (a mai Windowshoz képest!). Az IS-BASIC vegyes csomag volt - egyrészt megbízható és meglehetősen rugalmas is (a visszafejtés során azt tapasztaltam, hogy gyakran többféle bemenetet fogadott el, nem egyetlen formátumot). Másrészt nagyon lassú volt – gyanítom, hogy az egyik oka a BCD használata volt, nem pedig a lebegőpontos aritmetika. Természetesen a többi Enterprise szoftver nagy része zseniális volt. Inkább a játékokat választottam más programozási nyelvek helyett – mindig is szerettem a játékokat, és szerettem volna egy kis szünetet a Basictől és a fordítóktól, így nem töltöttem sok időt a Forth, LISP, Pascal stb. felfedezésével.

- Nem gondoltál arra, hogy továbbra is programokat írj az Enterprise-ra? Milyen programokat írnál, ha továbbra is Enterprise gépen programoznál?
Elnézést, de a válasz „Nem" - nem írok többet programokat - csak videózok. Több száz videót készítettem (valóban több százat). Némelyikük nyilvános a YouTube-on, és ha érdekel, megtalálhatod őket, ha rákeresel a nevemre (Hiner Péter). Egyébként van egy velem azonos nevű fiatal cseh srác is, akinek van néhány számítógépes játék videója a YouTube-on - az nem én vagyok!

Az interjút Bognár Tamás (Szipucsu) készítette.
Fordítás: Noel Persa (Geco)

Zzzzip

A BASIC compiler - 1. rész

Peter Hiner - 1986.

Egy programkatalógusban a Zzzip 2.0-ról írnak, de nálunk csak az 1.1-es verziót lehetett megszerezni. Peter Hiner, a Zzzip írója, aki 2010-ben talált rá az Enterprise fórumra, szintén nem tud 2.0-ás verzióról. Azonban létezett 1.2-es verzió, amit akkor meg is osztott velünk a forráskóddal együtt, kifejezve aggodalmát, hogy hosszú ideig egy régi verziót használtunk, ugyanakkor örült, hogy még mindig van érdeklődés a programja iránt. Arra már nem emlékezett, hogy miben tér el egymástól az 1.1-es és az 1.2-es verzió. Utánanézett, és kiderült, Tim Box-szal való levelezése során említette, hogy probléma volt a gyökvonás függvényével, ezt javította ki az 1.2-es verzióban. Emlékei szerint egy olyan verzióba is belekezdett, amely sprite-okat is kezel, ezt azonban sajnos nem találta meg. Létezik egy ZIPX verzió is, mely a Boxsoft basic bővítéshez készült, ezt szintén a rendelkezésünkre bocsátotta.
Peter maga is írt basic játékokat, melyek Zzzippel lefordítva jól használhatók. Ő írta a Boxsoft (Screenplay) kazettákon megjelent Pacman, Orbs, Galaxians játékokat.
A Zzzip néhány, játékokat tartalmazó kazetta B oldalán volt megtalálható hazánkban (Gratis Soft), leginkább így lehetett hozzájutni – vagy természetesen cserepartnertől. Az Egyesült Királyságban külön kazettás programként került forgalomba. Még az Enterface anno hirdetett egy pályázatot, ahova gyorsítást igénylő basic programokkal lehetett nevezni, és az első helyezett nyereménye a Zzzip lett volna. Endi Miner című játéka lett a győztes, a program listáját le is közölték az újságban, de a nyereményt Endi azóta sem kapta meg.
A Zzzip magyar nyelvű leírása régóta rendelkezésünkre áll. Az angol nyelvű leírás létezéséről azonban 2007-ig nem tudtunk, akkor John Fante fórumtársunk tette közzé.

1. Bevezetés

A ZZZIP egy Egész értékű BASIC Fordító. A BASIC-ben írt programot átalakítja gépi kódúra a gyorsabb futtatás érdekében. A ZZZIP a nagyobb sebességnövekedést azáltal éri el és az „Egész értékű" elnevezés is arra utal, hogy a BASIC programban szereplő konstansok és változók értékeit egész értékeknek tekinti (azaz a tizedespont utáni számjegyeket nem veszi figyelembe). Ennek következtében van néhány megkötés a ZZZIP használatára vonatkozólag, de ha gondosan követjük a kézikönyv utasításait, a ZZZIP-et könnyen fogjuk kezelni és meglepően jó eredményeket érhetünk el. Az első nyilvánvaló kérdés: „Hányszor lesz gyorsabb a lefordított programok futása?" A válasz attól függ, hogy milyen típusú BASIC program került lefordításra. Egy olyan program, amely véletlenszerűen megadott számokat rendez sorba, körülbelül 50-szer lenne gyorsabb, mint a BASIC, míg egy string-eket sorbarendező program esetleg csak(!) 12-szer. Másrészt egy olyan program, amely különösebb számolás nélkül ábrázol pontokat és vonalakat, valószínűleg csak kétszer lenne gyorsabb a BASIC-nél (ami még mindig bizonyos haladást jelent). Ha BASIC programunkat úgy módosítjuk, hogy a gépi kódú programoknál alkalmazott módszereket használja, sokkal gyorsabb képernyőkezelést kaphatunk (a szalagon található BENCH. BAS program a gyors képernyőkezelésre ad példát). Szalagunkon a ZZZIP (mely három részből áll -ZIP, ZIPA, ZIPB) és a BENCH. BAS programok találhatók, ez utóbbi néhány egyszerű rutint tartalmaz, s ezekkel szemlélteti a ZZZIP által biztosított sebességnövekedést. Javasoljuk, hogy először a BENCH.BAS-t fordítsa le, mert ezáltal jobban megismerheti a ZZZIP használatához szükséges eljárást.

Mi is az a compiler?

A compiler egy fordítóprogram, amely egy magasszintű nyelven (esetünkben BASIC-ben) megírt programot fordít gépi kódú programmá (pontosabban pszeudó-kóddá). Így lehetőségünk van gyorsabb programok készítésére, anélkül, hogy más programozási nyelvet kellene választanunk. A compiler-eknek két csoportja létezik: integer és real fordítók. Az integer fordítók csak egész számokkal tudnak dolgozni, cserébe viszont a lefordított program gyorsabban fut mint a real fordítókkal készített program. A real fordítók nem csak egész számokkal tudnak dolgozni, ezért azonban a lassabb programfutásssal fizetünk.

2. A ZZZIP futtatása

A ZZZIP egyformán jól működtethető szalagon és lemezen. A 2.1 fejezet a szalagon történő használatot írja le, de a lemezes használatra is alkalmazható a nyilvánvaló különbségek (és a 2. 2 fejezet pontjainak) figyelembevételével.

2.1 Szalagos rendszerek

Az indulás a BASIC képernyő-editorából történjék, miután törölt belőle minden más programot. A ZZZIP futtatásá-

hoz nyomja le a START billentyűt, s az három részletben automatikusan betöltődik. Ha ez megtörtént, a ZZZIP egy szöveg kiírásával jelzi, hogy annak a kazettának a bekészítésére vár, amely a fordítani kívánt BASIC programot tartalmazza. A ZZZIP feltételezi a magnó távvezérlésének lehetőségét (ha nincs ilyen, a PAUSE billentyűt kell használnia). Gépelje be a BASIC program nevét (vagy egyszerűen nyomja meg az ENTER billentyűt, s ezáltal az első szóba jöhető program betöltődik). A ZZZIP a BASIC programot a memória egy lefoglalt területére tölti be. Megjegyezzük, hogy az ENTERPRISE 64 gépen ennek a nagysága 16K-ra korlátozódik, s ha a BASIC program túl nagy, a ZZZIP a „Memory Full" (a memória megtelt) üzenetet írja ki. Ezután a ZZZIP egy szöveg kiírásával jelzi, hogy annak a kazettának a bekészítésére vár, amelyre programjának lefordított változatát ki akarja menteni, továbbá a magnó RECORD billentyűjének lenyomására. Gépelje be azt a nevet, amelyet a lefordított programnak kíván adni, ezután a ZZZIP működésbe lép. (A programnevekről a 2. 3 fejezet ír bővebben). A ZZZIP négy menetben dolgozza fel a BASIC programot, s kiírja a BASIC sorok sorszámait is, így teszi lehetővé a program fordításának nyomonkövetését. Ha az első két menet alatt a ZZZIP szembekerül valamilyen problémával (például egy BASIC paranccsal amelyet nem tud kezelni), egy üzenetet hagy a kritikus sorszámnál, majd rátér a következő sorra egészen addig, míg ennek a menetnek a végére nem ér. Ily módon rendelkezésre áll azoknak a sorszámoknak a listája, amelyekhez az adott menetben problémás sorok tartoztak, s a mellettük feltüntetett hibaüzenetek jelentését a 4. fejezetben található útmutatások alapján értelmezhetjük. Rendszerint ebben a részben a ZZZIP probléma nélkül fut, s a harmadik és negyedik menettel folytatódik, amikor is kiír a szalagra egy kis BASIC betöltő programot, majd magát a lefordított programot. Ez utóbbi két menet alatt is van egyfajta ellenőrzés, s ha előfordul valamilyen hiba, a ZZZIP kiír egy üzenetet és azonnal leáll. Végezetül a ZZZIP megkérdezi, hogy akarja-e más egyéb programok fordítását. Ha igen, a ZZZIP visszatér ahhoz a ponthoz, amelynél a BASIC forrásprogram betöltődik.

2.2 Lemezes rendszerek

Mindenekelőtt át kell másolnia a ZIP, ZIPA, ZIPB és BENCH.BAS programokat szalagról lemezre. A BASIC forrásprogram vagy a lefordított program nevének bevitele során beírhatók olyan többletinformációk, mint pl. a lemezegység azonosítója vagy a directory elérési út. A forrás és a lefordított program lehet különböző lemezen is, ha szükséges, s ha csak egy lemezegység áll rendelkezésre, a szokásos lemezcserélésre szólító üzenetek jelennek meg.

2.3 Programnevek

A különböző lemezrendszerek kompatibilitását néhány programnévre vonatkozó szabály biztosítja. Ha a lefordított programot nem látja el névvel (csak megnyomja az ENTER billentyűt), a ZZZIP a default, azaz alapértelmezésű nevet fogja használni: „Z". A megkívánt név formátuma a következő: egy legfeljebb 8 karakteres főrész, melyet egy esetleges kiterjesztés követ (a pont írásjel és még legfeljebb 3 karakter), mely a program típusát jelzi. A ZZZIP a lefordított program nevét mindig 12 karakterre korlátozza. Amennyiben talál kiterjesztést, akkor ezt a BASIC betöltőhöz csatolja, a lefordított programnak a „.Z" kiterjesztést adja. Ha nincs kiterjesztés, és a név főrésze nem hosszabb 8 karakternél, a ZZZIP egyszerűen hozzácsatolja a lefordított program nevéhez a „.Z" kiterjesztést. A programok elnevezésére a következő konvenciót javasoljuk:

PROGRNEV.BAS (BASIC változat)
PROGRNEV.ZIP (Betöltő program)
PROGRNEV.Z (Lefordított változat)

3. A lefordított program futtatása

A lefordított programok majdnem ugyanúgy kezelhetők, mintha BASIC programok volnának. A legfontosabb különbség az, hogy egyszerre csak egy lefordított program tartható a memóriában és ez a 0. program kell, hogy legyen. A programon belül a RUN parancs más programok automatikus betöltésére és futtatására használható (szabadon keverve a BASIC és a lefordított programokat). Minden lefordított programot egy kis BASIC betöltő program előz meg, ez automatikusan betölti a lefordított programot és ellenőrzi a program méretét, beleértve a változók által igényelt helyet, stb. Előfordulhat, hogy egy nagyon nagy program lefordítása után megjelenik az „Insufficient memory" (nem elegendő memória) üzenet. Ilyenkor meg lehet próbálni a gép teljes újraindítását és a program újratöltését, de ha ez nem segít, az eredeti program módosításával kell csökkenteni az igényelt memória mennyiségét. Ezek után a lefordított programnak ugyanazokat az eredményeket kell szolgáltatnia futtatáskor, mint az eredeti BASIC programnak, csak gyorsabban. A STOP billentyűvel megszakítható a program futtatása, a START billentyűvel pedig újraindítható. Megjegyezzük, hogy a CONTINUE nem működik, valamint a SAVE és a LIST sem használható lefordított programnál. Ha valamilyen okból a lefordított program futása mégsem lenne tökéletes, nézzen utána a lehetséges okoknak a 4. fejezetben. Mindamellett javasoljuk, hogy futtassa le mégegyszer a lefordított programot és próbálja megkeresni azt a helyet ahol elszáll a program, ily módon leszűkítheti a BASIC programban történő hibakeresés területét.

4. Problémák kezelése

A felmerülő problémák kezelésének egyetlen járható útja az eredeti BASIC programba való visszatérés és annak módosítása. Jóllehet, a ZZZIP-et alapos vizsgálatoknak vetették alá, nem lehetetlen, hogy esetleg hibát talál benne (egy még ismeretlen tulajdonságát!), de ezekben az esetekben is a BASIC programot kell módosítania úgy, hogy a ZZZIP azt pontosan le tudja fordítani. Ne feledje el annak a lehetőségét sem, hogy hiba következhet be a BASIC program betöltése során is. A ZZZIP a következő hibaüzeneteket adhatja:

Memory full (memória megtelt)

A BASIC forrásprogram betöltése közben azt tudatja, hogy a BASIC program túl hosszú (az Enterprise 64 gépen a határ 16 Kbyte). Ha az üzenet a fordítás egy későbbi szakaszában jelenik meg, akkor azt mutatja, hogy a ZZZIP kifogyott a változónevek tárolására fenntartott területből. Próbálkozzon a változók számának, vagy a változónevek hosszának csökkentésével.

Too many labels (túl sok címke)

Ez azt jelenti, hogy a ZZZIP kifogyott a címkék (főleg a sorszámok) tárolására fenntartott helyből. Próbálkozzon a sorok számának csökkentésével, kitörölve a REM sorokat, többszörös utasítások használatával, vagy a program két részre bontásával.

„...” not supported (nem elfogadható)

A ZZZIP jelzi a kezelhetetlen elemet és annak BASIC sorszámát. Az elem lehet egy parancs (lásd az 5. fejezetet) vagy a REF függvény, vagy lehet egy olyan változó, amely egy konstans érték helyett határozza meg egy vektor méretét vagy egy karaktersorozat maximális hosszát (a ZZZIP ezeknek rögzített memória területeket tart fenn, s ezért nem engedi meg a „dinamikus dimenzionálást”).

Syntax?

Ez valószínűleg azt jelöli, hogy egy változó ugyanolyan néven szerepel, mint egy beépített függvény, pl. a SIZE vagy a VAL. Ez a BASIC-ben bizonyos esetekben megengedett (habár a megfelelő beépített függvény használatát akadályozza), de a ZZZIP minden esetben a függvény értelmezést feltételezi. Ezért egy zárójelben lévő operandust vár, pl. a SIZE(X)-et. Akkor is problémák merülhetnek fel, ha a kulcsszavakat, úgymint a COLOUR vagy a SCREEN szavakat változónévként használjuk.

Garbage (hulladék)

Majdnem biztos, hogy ez a program pontos betöltésének sikertelenségét jelzi.
Non-integer / out of range (nem egész / tartományon kívül)
A ZZZIP jelezni fogja, ha egy konstans nem elfogadható, vagy azért, mert nem egész, vagy mert kívül esik a megengedett -32767-től +32767-ig terjedő megengedett intervallumon.
Loop / block termination (ciklus / blokk lezárás)
A ZZZIP ellenőrzi a ciklusok és blokkok pontos lezárását. Ne felejtsük el, hogy a BASIC programot a ZZZIP a sorszámok szerint sorban fordítja le, s ez a sorrend nem biztos, hogy megegyezik a futtatás alatti végrehajtási sorrenddel. Mindenfajta ciklus és blokk egymásba skatulyázható, de a ZZZIP elvárja a sorrendben összeillő nyitó és záró utasításokat.
Reference not found (nem találja a hivatkozást)
Olyan helyre történő hivatkozást jelöl, amelyet a ZZZIP nem címkézett meg. Ez a BASIC egy fölösleges sorának tulajdonítható, mely GOTO-t vagy más nem létező sorra hivatkozó parancsot tartalmaz. Másrészt olyan utasítás is oka lehet, ahol pl. a 200. sor létezik, a RESTORE 200 számára viszont nem tartalmaz DATA-t.
Identifier declared twice (kétszer deklarált azonosító)
Azt jelzi, hogy a BASIC sorban szereplő egy vagy több változó vagy vektor határozott deklarálása már megtörtént valamelyik előző NUMERIC, STRING vagy DIM utasításban. Ott a legvalószínűbb az előfordulása, ahol lokális változókat vagy vektorokat DEF blokkokon belül deklaráltunk. Vizsgálja át azokat a változókat vagy vektorokat, amelyeket előzőleg már egyszer deklarált és válasszon nekik egyedi nevet.
Warning - global variable (figyelmeztetés - globális változó)
Azon problémák megoldása érdekében, amelyek a DEF sorokban vagy blokkokban előforduló lokális változókkal kapcsolatosak, a ZZZIP ellenőrzi az átadódó paraméterként használatos változókat, úgymint az X-et és Y-t a DEF FUNC(X,Y)-ban. Abban az esetben, ha egy változónevet már használtunk valamelyik megelőző sorban, vagy ha már határozottan deklaráltuk egy tetszőleges sorban, a ZZZIP egy figyelmeztető üzenetet ad. Ez arra emlékeztet, hogy a ZZZIP a változót globálisként fogja kezelni, de mivel ez nincs hatással a lefordított program futására, az üzenet csak figyelmeztetés és a fordítás folytatódik.

Compiled program does not run correctly
(a lefordított program hibásan fut)

A két legvalószínűbb ok a DEF blokkokban előforduló lokális/globális változók eltérő kezelése és az egész aritmetika használatának hatásai. Két egyszerű szabály van: egyrészt bármely lokális változónévnek egyedinek kell lennie, másrészt kerülni kell az olyan helyzeteket, amelyekben egy definiált függvény önmagát hívó CALL utasítást tartalmaz. A használt egész aritmetikában a tizedespont után álló számjegyek nincsenek figyelembe véve, és minden értéknek a -32767 és +32767 számtartományba kell esni. Ezek a szabályok egyaránt vonatkoznak a számítások közbeeső eredményeire és a végső eredményre. Ily módon az aritmetikai függvényekben alkalmazott műveleti sorrend nagyon fontos. Például, a LET X=2/4*10 nulla eredményt fog adni, s ahhoz, hogy a pontos 5 értékű eredményt adja, a LET X=2*10/4 alakra kell módosítanunk. Hasonlóan a LET X=5*10000/2 kifejezést át kell írnunk a LET X=10000/2*5 formára. Ezek a példák nyilvánvalóak, mert konstans értékeket használnak. A következő LET X=A/B*C utasítás már nem ilyen magától értetődő. Ilyen esetekben szükség lehet az A, B és C lehetséges értékeinek átgondolására, azt a szabályt alkalmazva, hogy jobb osztás

előtt szorozni, hacsak az értékek nem olyan nagyok, hogy már a közbeeső eredmények is túlléphetik a 32767-et. Bizonyos körülmények között megengedhető, hogy egy változó érték 32768 és 65535 közé essen, mégpedig akkor, ha összeadás vagy kivonás eredménye. Mindamellett óvatosan kell élnünk ezzel az engedménnyel, mivel arra épül, hogy a legnagyobb helyértékű bit szokásos előjelbitként való használatát figyelmen kívül hagyja. Leírható például a LET X=25000+25000, s ezekután X a következő utasítástípusok bármelyikében használható:

LET X=X+Y vagy LET X=X-Y
POKE X,Y vagy LET Y=PEEK(X)
SPOKE S,X,Y vagy LET Y=SPEEK(S,X)
CALL FUNCTION(X,Y,Z)
LET A=FUNCTION(X,Y,Z)
IF X=Y THEN... (de nem < vagy >)

Ennek a tulajdonságnak a gyakorlati haszna a memória címek kezelésében nyilvánul meg, de ügyelni kell arra, hogy csak a fent említett utasítástípusokat használjuk.

5. BASIC parancsok

A ZZZIP megtart csaknem minden olyan parancsot, amely egy BASIC programon belül általában végrehajtható. A Későbbiekben látni fogja, hogy néhány parancs használatakor óvatosnak kell lennie vagy némelyeket egyáltalán nem használhat, a többi viszont korlátozás nélkül rendelkezésre áll.

ALLOCATE

Az ALLOCATE parancs felmozgat a memóriában egy BASIC programot, s ily módon helyet biztosit a gépi kódú beszúrásoknak. A lefordított programok nem mozgathatók el, ezért esetükben a beszúrásoknak egy 255 byte-nyi terület (az 1300H-tól fölfelé) van fenntartva. Megjegyezzük, hogy ennek a területnek a méretét az ALLOCATE parancs nem befolyásolja. Valójában 255 byte-nál több is rendelkezésre állhat annak a kockázatával, hogy esetleg átnyúlik a 17FDH címtől lefelé terjeszkedő string-kezelő területre. Így tulajdonképpen körülbelül 1Kbyte-nyi rész tartalmazhat beszúrásokat egy programban, s ide nem értendő bele a hosszú stringek kezelése. A legokosabb, ha lefordítja a programot és megnézi, hogy működik-e! Megjegyezzük, hogy a lefordított programoknak nincs szüksége arra a rutinra, amelyet rendszerint az EXOS 2.0 verziójában, az ALLOCATE parancsban előforduló hiba kiküszöbölésére használunk. Ha mégis szerepel a programban, akkor a rutinnak meg kell vizsgálnia, hogy VERNUM=2 igaz-e. Így a lefordított programban ez letiltódik (mivel ekkor a VERNUM függvény értéke 1).

DEF

A ZZZIP a legtöbb DEF parancsot elfogadja, mind a különálló sorok, mind pedig a blokkok függvényként történő definiálásánál, de néhány korlátozás azért fennáll. Lényeg az, hogy a ZZZIP minden változót globálisként kezel. Ennek az az egyik oka, hogy a ZZZIP egy BASIC programot a sorok számozásának sorrendjében dolgoz fel, s ez valószínűleg nem fog megegyezni azzal a sorrenddel, amelyben a sorok a futás során végrehajtódnak. Ezért visszakövetkeztetéssel nem tudja mindig meghatározni egy változó globális vagy lokális voltát. Ha egy DEF blokkon belül a változókat lokálissá akarjuk tenni, ezt egyszerűen úgy érhetjük el, hogy egyedi változóneveket használunk. Mindamellett ne felejtsük el, hogy ennek hatására sem fogja a ZZZIP a DEF blokkok futása során a változókat lokálisnak tekinteni, ezért a DEF blokkok nem használhatók rekurzívan. Kissé bővebben kifejtve, tegyük fel, hogy egy blokk, melynek fejléce a DEF FUNC(X) utasítás, tartalmazza a CALL FUNC(Y) utasítást. A BASIC minden alkalommal, amikor a függvény meghívja önmagát, megőrzi az X lokális értéket és a későbbiekben, mikor a függvény „legombolyítja" önmagát, visszaadja ezeket az egymást követő X értékeket. A ZZZIP nem őrzi meg ezeket a lokális értékeket, s így egy ilyen helyzetben nem szolgáltat pontos eredményt. Ezért ne engedje meg, hogy egy függvény a CALL utasítással önmagát hívhassa. A „csak globális" szabály alól van egy kivétel, ha egy DEF blokkban lévő változónak a neve megegyezik magának a blokknak a nevével, ez a változó felhasználható egy LET utasításban és ezek után a ZZZIP lokális változónak fogja tekinteni. Ez képessé tesz egy DEF blokkot arra, hogy visszaadjon egy értéket a főprogramnak. Példaként nézzük a következő függvényt:

```
1000 DEF TEN
1010   LET TEN=10
1020 END DEF
```

A főprogramban lévő PRINT TEN utasítás hatására a 10-es szám kerül majd nyomtatásra. Az 1010-es sor egy egyszerű LET utasítást tartalmaz, s a ZZZIP az ott lévő TEN változót lokálisnak tekinti. Ellenben, ha a TEN nevet az 1010-es sorban valamilyen más típusú utasításban használnánk, ez a TEN függvény rekurzív CALL hívását okozná és a lefordított programban hibásan működne. Hasonlóan, ha olyan függvényből akarunk értéket visszaadni, amely segédváltozókat tartalmaz paraméterátadás céljából, szintén használhatunk lokális változót. Például a LET DOUBLE=X*2 megengedett egy olyan blokkban, melynek a fejléce DEF DOUBLE(X), s a főprogramban lévő PRINT D0UBLE(3) hatására a 6 kerül majd nyomtatásra. Egyszerű függvénynevek is átadódhatnak egy másik függvényen belül felhasználandó paraméterként, így a főprogramban lévő PRINT DOUBLE(TEN) a 20 kinyomtatását eredményezi. Ezek a példák hibátlanul működnek egy lefordított programban. Megjegyezzük, hogy a ZZZIP nem fogadja el a REF parancsot és hibaüzenetet ír ki. Valami más, a BASIC programéhoz hasonló eredményeket szolgáltató módszert kell találnia.

PROGRAM

Ez a parancs nincs hatással a lefordított programra és a ZZZIP a REM-hez hasonlóan kezeli.

RESTORE

Folytatjuk!

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

Sötét évszázadok 2.

Írta: Bodnár Tamás (Szipucsu)

Hihetetlen, mi került elő még 2021-ben... A Facebook csoportban felbukkant egy eddig ismeretlen, eredeti EP játék!

Budavölgyi László, a szerző írta:

30 év után sikerült betöltenem ezt a kalandjátékot, amit még 17 évesen írtam anno. Teljesen assembly-ben készült, saját LPT-t építettem fel benne, és a képeket tömörítenem kellett, így is 128KB lett a program. Van benne néhány bug, amit már nem javítottam ki, de egész jól játszható. Az új Exdos és floppy drive szerzeményemmel sikerült olvasni az eredeti lemezt, pedig azt hittem, már végleg elveszett. Végig is játszottam, bár a 80%-át már elfelejtettem.

Az első részt is megtaláltam, az szöveges kalandjáték, és még basic-ben készült.

Ha Manthan lovag vagy a varázsló a helyszínen van, belassul sajnos, többször kell nyomni a space vagy nyíl billentyuket, de le nem fagy.

A kód nagy részét a grafika teszi ki. A szerző joystick-kal rajzolt mindent a Paintbrushban.

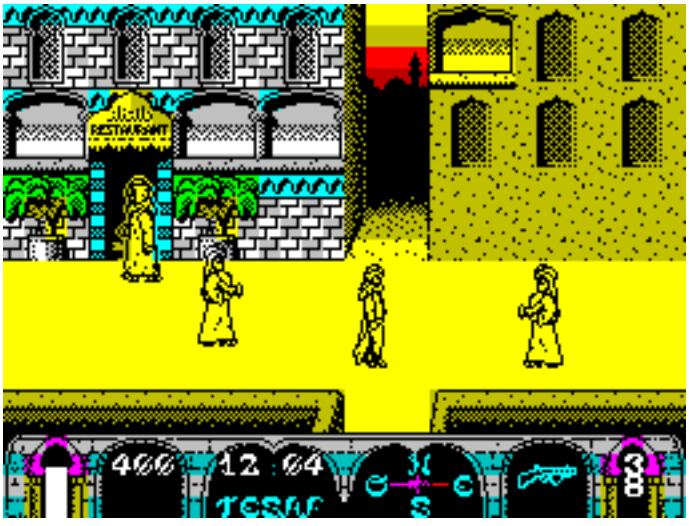
A szerző az első részt eddig még nem tette közzé.

TUAREG

Írta: Kiss László (Lacika)

Marrakech, 1990. január 23. Futótűzként terjedt a hír: Ait-Amart, Abdul Aziz szultán lányát berber gerillák rabolták el és váltságdíjat követelnek érte. A követelt váltságdíj tetemes: a szultán vagyonának fele! Csak annyi bizonyos, hogy a lány a gyorsan lezárt városban raboskodik valahol. Ben-Yussef, a szultáni hadsereg kapitánya - ismertebb nevén a Fekete Tigris - egyszemélyes mentőakcióra indul. Feladata, hogy három napon belül megtalálja és kiszabadítsa a szultán lányát fogvatartói karmaiból! A feladat egyáltalán nem könnyű: a hatalmas arab város a sikátorok és szűk utcák szinte átláthatatlan szövevénye, ráadásul „nyüzsög" (költői túlzás) az emberektől, nem tudhatjuk bizonyosan, ki a barát, ki ellenség. A város 12 - névvel ellátott - kerületre tagolódik, valamelyikben barátságosak az emberek, valamelyikben ellenségesek. A városban ötféle karakterrel találkozhatunk:

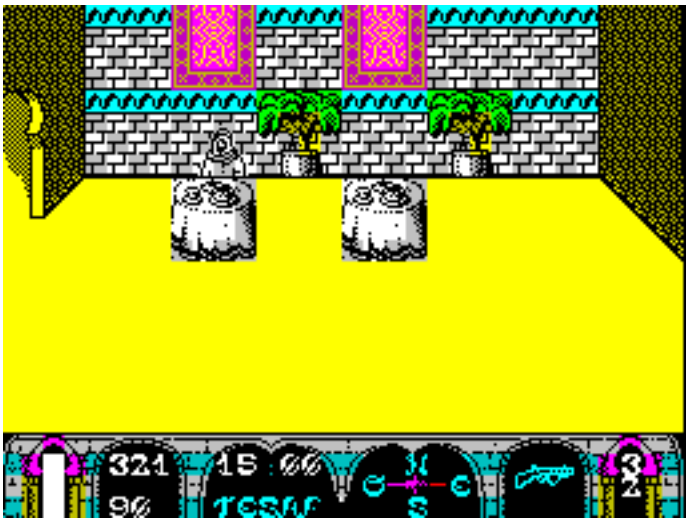
- Mór férfi: fel-alá sétálnak az utcákon. Amíg barátságosak, odamehetünk hozzájuk (szemből kell eléjük állni), némi pénzért cserébe információt szolgáltatnak, merre raboskodik Ait-Amari: ESTE - tőlünk keletre, OESTE - nyugatra, NORTE - északra, SUR - délre. Ha nincs pénzünk, nem segít. Ha véletlenül lelőjük, a kerület lakói ellenségesek lesznek, és lőni fognak ránk. Ha ellenséghez sétálunk oda, rossz információt ad, és a pénzt is elveszi. (Ezt onnan is lehet tudni, hogy ha egymás után többször megkérdezzük, össze-vissza beszél.)
- Az asszonyoknak talán valami vízhordó alkalmatosság van a hónuk alatt. Némi energiaveszteségért cserébe pénzt lophatunk tőlük. Már attól, akinek van egyáltalán. Persze ne csodálkozzunk, ha ezután nem leszünk népszerűek a kerületben...
- Rendőr: fegyverrel a kezében rohangászik. Roppant kellemetlen alak! Messziről észreveszi hogy fegyver van nálunk és ránk ront (gyorsabb, mint mi). Ha hozzánk ér, elveszi az összes lőszerünket. Nincs jobb megoldás, mint hogy lelőjük. Pontot úgysem számol a program...
- Haramiák: nagydarab szakálas fazonok. Minden olyan ajtó előtt strázsál egy, ami a hercegnő rejtekhelye lehet. Bent még több van belőlük. 10 találattól gyengülnek csak el. Addig az ajtó közelébe se tudunk jutni, amíg az őrtől meg nem szabadulunk.
- Az ablakokból leselkedő alakok. Ha átsétálunk alattuk, habozás nélkül megpróbálnak hozzánk vágni egy kezükbe akadó edényt.



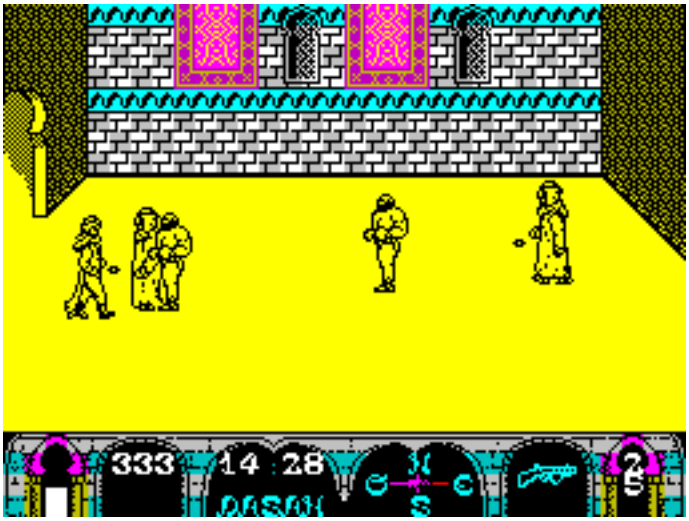
A tájékozódás kicsit bonyolult, mert ha befordulunk a sarkon, a nézőpont is változik. Az iránytűn az égtájak helyzete tehát ennek megfelelően váltakozik. Azért nem olyan „durva a helyzet" minta Tir-Na-Nog-

ban, itt mindössze két nézet van: vagy az utca északi vagy a nyugati oldalát látjuk a háttérben. Az iránytűtől balra egy óra van, alatta a kerület neve, ahol éppen vagyunk. Az időnek fontos szerepe van a játékban, mert a szolgáltatást nyújtó helyek csak megadott időben vannak nyitva. Este 8-kor besötétedik, de sajnos a forgalom ilyenkor sem csökken az utcákon. A város utcáin sok ajtót látunk ezek közül a leggyakoribbak a jelöletlen ajtók. Ezek bármelyike lehet a hercegnő rejtekhelye, minden játékban máshova „rejti" a program. Emellett többféle jelzéssel ellátott ajtó mögött várja valamely kereskedő, hogy szolgálatainkra álljon (jó pénzért):

- Armeria (fegyverkereskedés): Ide betérve többféle (olcsóbb - drágább) fegyver közül válogathatunk. Egyetlen fegyver lehet nálunk, csak akkor érdemes újat venni, ha a régi kiürült. A kés egyszer használatos hajítófegyver, a pisztolyban 10, a géppisztolyban 40 lőszer van. A legpraktikusabb a géppisztoly használata. A boltban az ‚E' nyomkodásával válogathatunk, a ‚C' megnyomásával vásárolhatunk. Reggel 10-től este 20 óráig van nyitva.
- Posada / Restaurant (fogadó / étterem): Délután 14 és 15 óra között vannak nyitva. Az elfogyasztott ebéd pótolja energiánkat. Az étterem drágább, mint a fogadó. Gondolva arra, hogy nem sikerül a hercegnőt az első nap kiszabadítani, napi egy étkezés szükséges.
- Meson / Hotel (motel / szálloda): Éjféltől térhetünk be egy kiadós alvásra, ami visszatölti az energiánkat. A szálloda drágább, mint a motel.
- Házak (jelzés nélkül): Ezek bármelyike lehet a hercegnő rejtekhelye. Mindegyik előtt egy haramia őrködik, bent még több vár, érdemes tehát tele tárral berontani a szobákba... Amelyik szobát kipucoltuk, az már üres is marad.



Az ajtó elé állva a program a pénztárcánkban (a „műszerfalon" balról a második kijelző) kijelzi, mennyibe kerül a szolgáltatás odabent. A fegyverkereskedésben a fegyver képe alatt látszik a vételár. A bal szélső sáv az energiánkat jelzi. Ez csökken, ha tűzharcban eltalálnak, egy haramia hozzánk ér, eltalál egy ránk dobott edény, pénzt lopunk, vagy az idő múlásával (kb. óránként egy sávot). Emiatt a napi étkezés, és az éjszakai pihenés elengedhetetlen, ha következő nap fittek akarunk lenni a



harcra. Egy életünk van, ha elfogy az energiánk, vége a játéknak! A jobb szélső számláló muníciónkat jelzi. A játék addig tart, amíg el nem fogy az energiánk, le nem telik az emberrablók által szabott határidő, vagy ki nem szabadítjuk a hercegnőt. Sajnos a hercegnő nem látszik az emberrablók rejtekhelyén, de ha lelőttük az összes bandität, a gratulációként kapott - külön betöltött - képen azért csak megcsodálhatjuk.

A Tuareg a felejthető tucatjátékokat forgalmazó TopoSoft egyik legérdekesebb próbálkozása. A készítők szemmel láthatóan egy „hagyományos" arcade játéknál összetettebb játékmenetben gondolkoztak, viszont kalandjátéknak sem nevezhető a produktum. A játék tempója is ezt az érzést erősíti: főhősünk megfontoltan sétálgat, a város méretéhez mérten talán túl komótosan is. A játék akár 5 perc alatt teljesíthető(!), de esélyesebb, hogy reménytelenül eltévedünk...

Az Enterprise verzió tetszőleges külső- vagy beépített botkormánnyal irányítható.

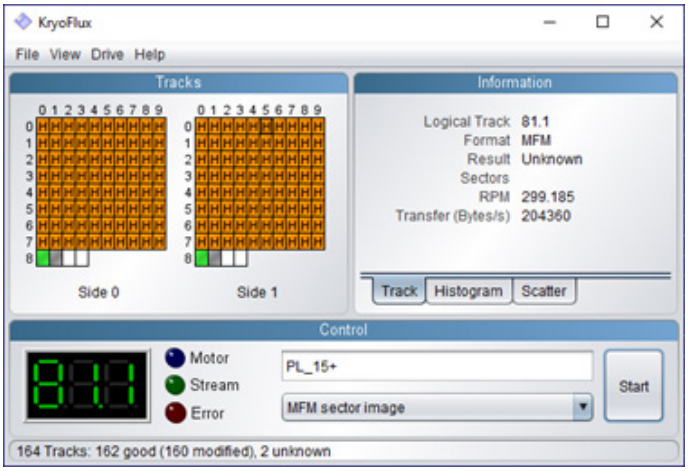
A billentyűzetkiosztás eltér: ‚1' - választás, ‚2' - vásárlás. HOLD / PAUSE - játék szüneteltetése / folytatása.

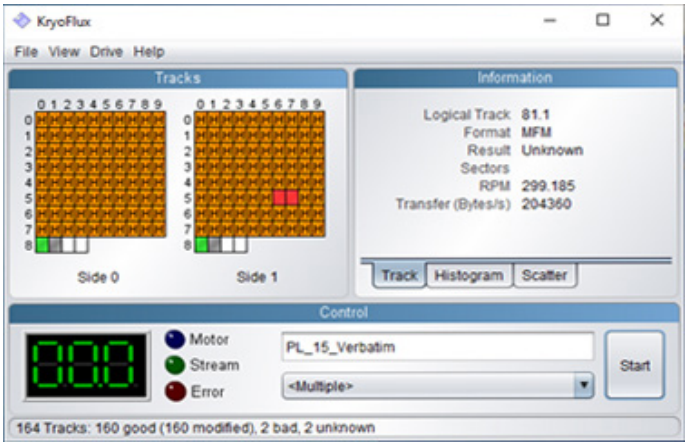
A nagy KryoFlux ismertető

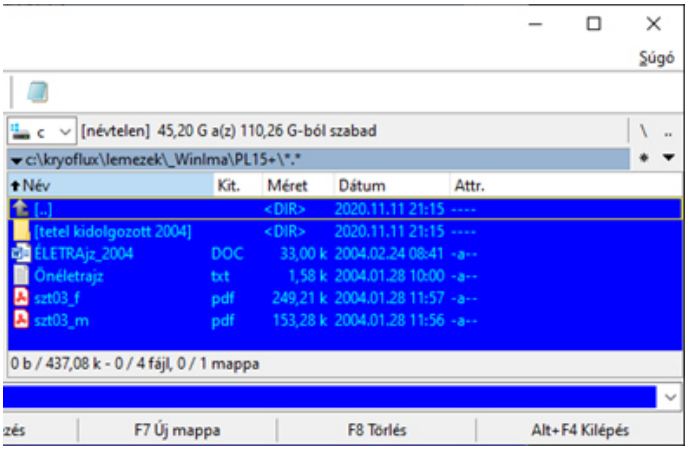
2. rész

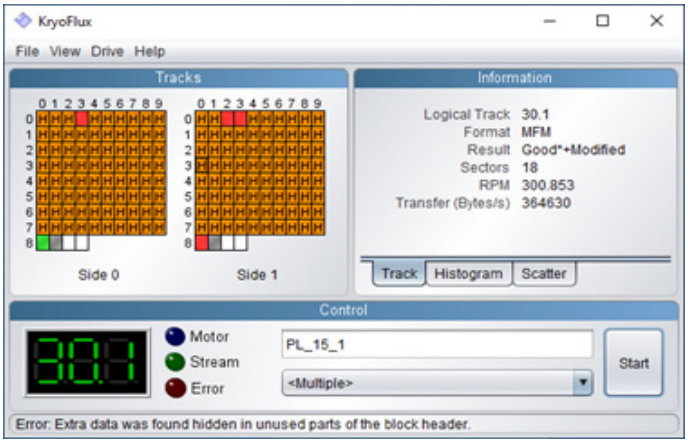
Írta: Orvos Gábor (Dr.OG)

Tapasztalatom az, hogy a HD-s lemezek sokkal sérülékenyebbek, nehezebben olvashatók, mint az alacsonyabb írássűrűséggel (pl. DD-s) rögzített diszkek. Érdemes többféle meghajtóval is próbálkozni, ha van rá lehetőség, illetve ha hozzáférünk azon drive-hoz, amin a lemez írva lett, az a legjobb. Sokszor egymás utáni beolvasásoknál változik a hibásnak ítélt sávok helyzete, így érdemes lehet többször próbálkozni, majd akár több, részben sérült image-ből összerakni egy épet. Utóbbihoz némi manuális másolgatás szükséges, a hibátlan tackeket magunknak kell összerakni egy mappába, majd a DTC GUI-t „deviceless" módban futtatva (»Drive« fül, »Stream Files« opció) egy hibátlan .img fájlt tudunk generálni, ebből már kinyerhetők a lemezen lévő fájlok.

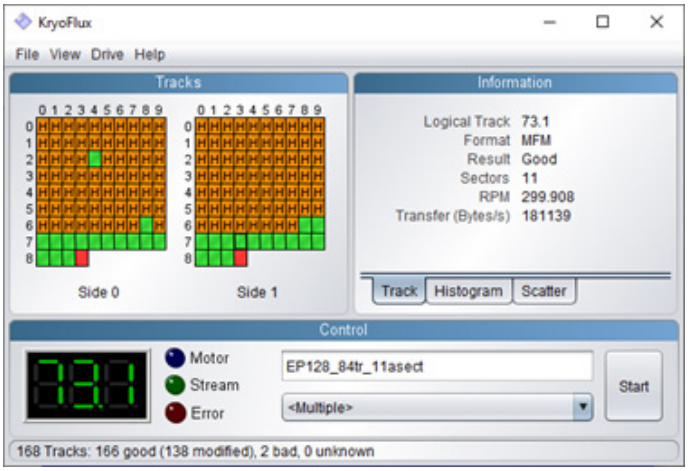


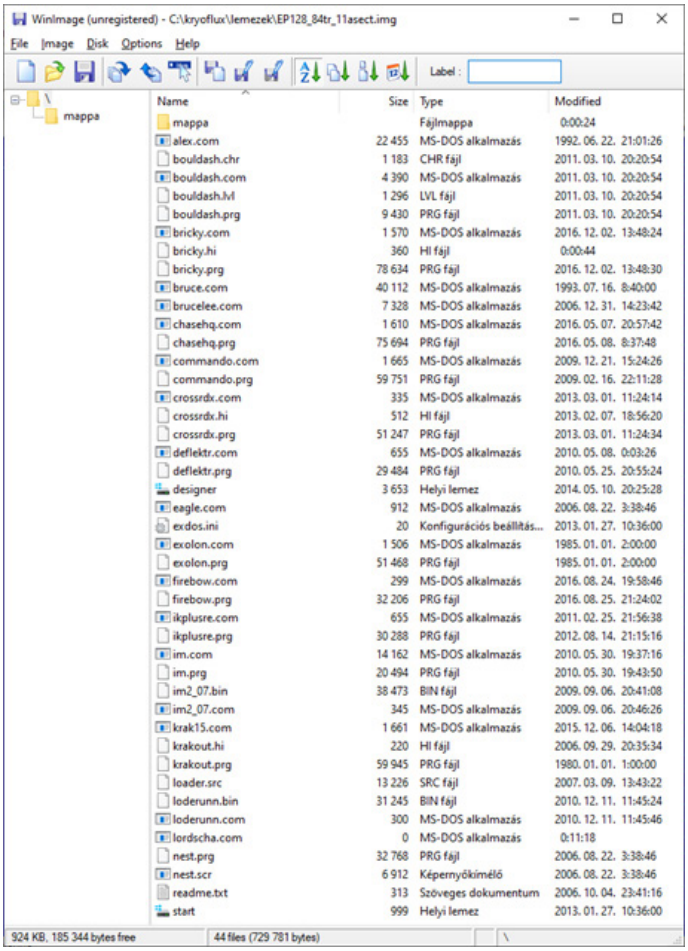






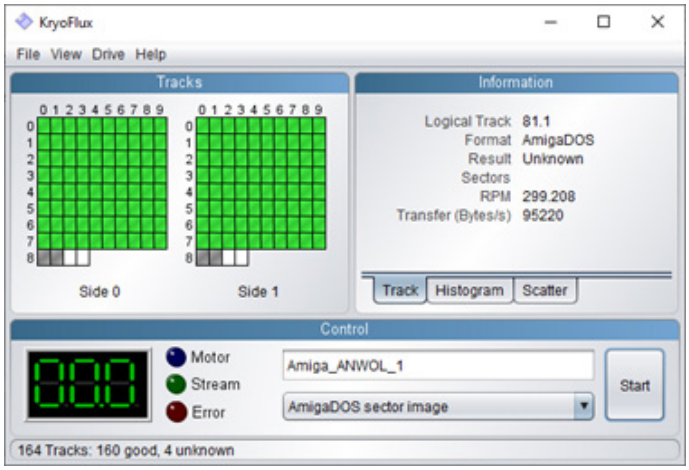
A 720k-s, DD-s 3,5"-ös lemezek még hosszú idő után is nagy valószínűséggel beolvashatóak, akár PC-n, akár más, fájlrendszer szempontjából kompatibilis rendszeren, pl. Enterprise-on írtuk azokat. Még egy érdekesség: a különféle „egzotikus" sáv- és szektorszámokkal is elboldogul a szerkezet, pl. egy 84-sávos, sávonként 11 szektort tartalmazó DD-s lemezzel próbáltam, a lemez utolsó sávjára már nem jutott mágnesezhető bevonat, emiatt hibás, de az összes adat szépen lementhető róla.





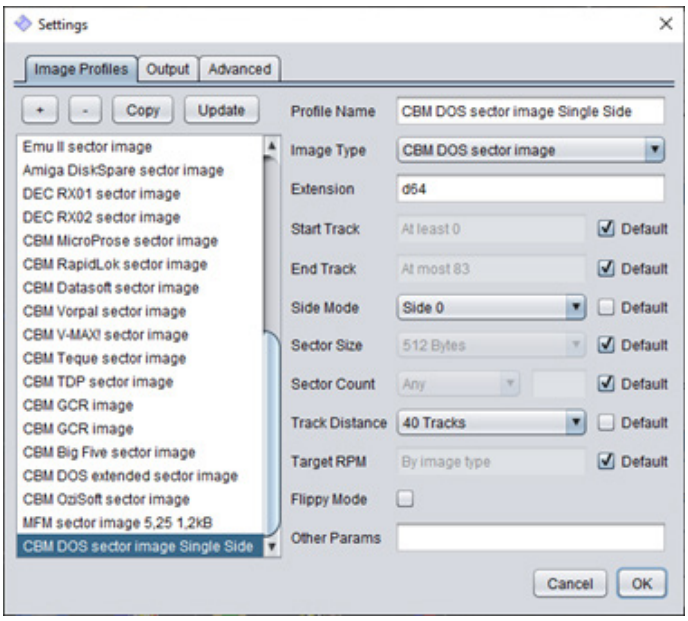
2. Amigás diszkek

Ezek messze jobban tartják magukat, mint a PC-s adathordozók. Ennek oka valószínűleg az alacsonyabb írássűrűség. Gyakorlatilag az összes lemezt, amit próbáltam, be tudtam olvasni hibátlanul. Ha a profilok közül az AmigaDOS sector image-t választjuk, akkor egy szabványos, 880kB-s .adf-et is kapunk rögtön. Azt hozzátenném, hogy először egy nullkilométeres NEC FD1231H meghajtóval próbáltam beolvasni a lemezeket, de ez időnként jelzett hibás sávokat. Aztán elővettem egy régebbi, használt SONY MPF520-1 FDD-t, azzal első pöccre hibátlanul végigment minden. Sok esetben nem csak zöld, de szürke blokkok is vannak, ne essünk kétségbe, ez itt nem zajt, hanem ismeretlen kódolási sémát jelöl (trackloaderes cucc vagy másolásvédelem). Utóbbi esetben az .adf-ünk hibás lesz, de a nyers sávinformációkat visszaírva egy másik lemezre az ugyanúgy fog működni, mint az eredeti igazi vason. Szerencsére a törött másolásvédelmű és natív AmigaDOS-os cuccokkal nem lesz gond, sőt, a vége felé már a műsoros lemezekre sem raktak védelmet.

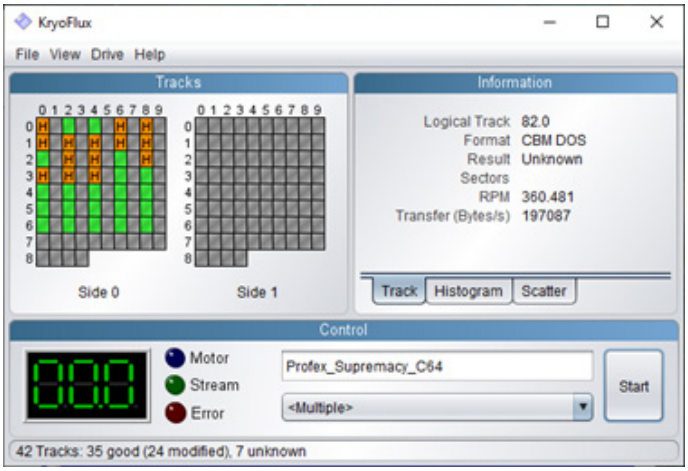


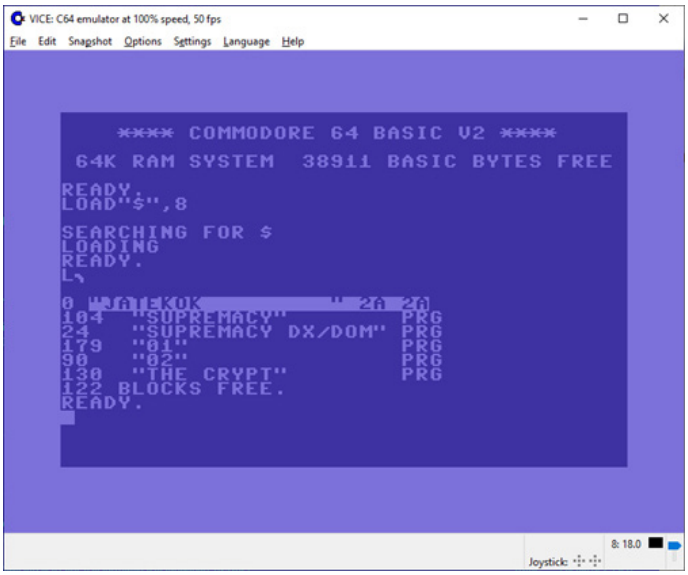
3. C64-es (vagy azzal kompatibilis, pl. VIC-20, C16/Plus4, C128) lemezek:

CBM DOS sector image-t érdemes választani, így a nyers adatok beolvasása mellett .d64 fájl is generálódik. Tapasztalatom, hogy ezek is eléggé hibatűrők, ha nem is első nekifutásra, de többszöri próbálkozásra szinte az összes beolvasható. Mivel a drive-ot egyelőre nem moddoltam, csak az első oldalakkal próbálkoztam. Érdemes ilyen esetben a flippy módot (Flippy Mode) kikapcsolni, a sávszámot pedig 40-re állítani. Az 1-es oldal vizsgálatát is kapcsoljuk le, elég a 0. Én ehhez csináltam, illetve módosítottam egy már meglévő profilt (»File«→»Settings«→»Image Profiles«→kijelölés, majd »Copy«, lásd képen).

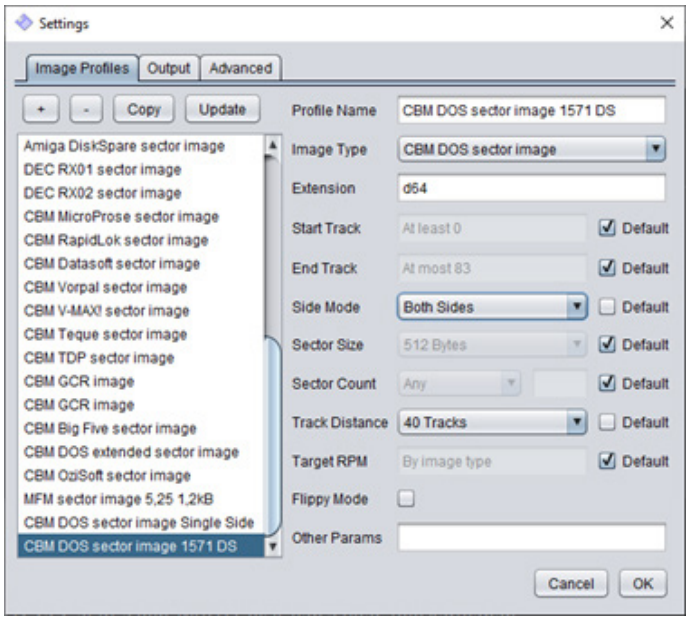


Érdekesség, hogy a HD-s lemezre írt adatokat is vissza tudta olvasni a rendszer, pedig ez nem mindig szokott sikerülni még eredeti hardveren (C64 és 1541 kombó) sem. Ha jól megfigyeljük, akkor a fizikailag 80 sávot kezelő meghajtó minden 2. track-et szürkével jelöli, ahogy az 1. oldalt és a 70 fölötti sávokat is (a CBM gépek hack nélkül csak 35 sávot kezelnek a 40-ből).





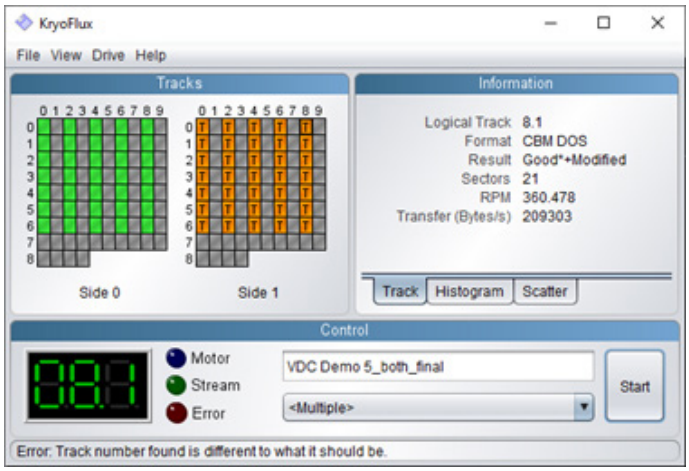
Még egy kuriózum: a C1571-es lemezmeghajtóval C128-on írt kétoldalas lemezeket is szépen beolvassa, a megfelelő beállítások és az eredmény az ábrán láthatók.



Az eredmény 2db .d64 kiterjesztésű fájl lesz (_s0 és _s1 végű), ezeket pl. Windows alatt az alábbi paranccsal tudjuk .d71-é összefűzni:

copy /b lemeznév_s0.d64+lemeznév_s1.d64 lemeznév.d71

Az eredmény bitre megegyezik az eredeti lemez tartalmával.



A beolvasás sebessége:

Hardvertől és operációs rendszertől függően összeállításonként változik, én egy 64-bites Win 10-et futtató, SSD-s kb. 3 éves gamer PC-n és egy szintén 64-bit Win 10-es SSD-s 7 éves laptopon próbáltam, az előzővel 2,5, az utóbbival kb. 4 percig tartott egy-egy lemez teljes beolvasása, ami nem olyan rossz. A művelet ideje tapasztalatom szerint független a kódolási formátumtól, tehát a 35 sávos, egyoldalas, DD-s, tehát kb. 170kB-ot tartalmazó C64-es lemezt ugyanannyi idő alatt viszi be, mint a 80

sávos, kétoldalas, HD-s 1,44MB-os PC-s floppyt. Persze ha lemezhibák nehezítik a beolvasást, az újrapróbálkozások számától függően többszörösére nőhet az időtartam, extrém esetben akár 20-30 percre is.

Néhány szó a floppy lemezekről:

Nagyjából a ‚80-as évek elejétől a ‚90-es évek közepéig ez volt a fő hordozható tárolóeszköz a 8 és 16-bites gépek világában. Méret szempontjából megkülönböztetünk 3", 3,5", 5,25" és 8"-es lemezeket (vannak egyéb egzotikus értékek is, de ezek a leggyakoribbak), bár igazán közülük csak a két középső terjedt el szélesebb körben. Lehet a lemez egy- vagy kétoldalas, adatsűrűség (density) szempontjából egyszeres (szimpla-SD), kétszeres (dupla-DD), négyszeres (quad-QD) vagy magas (high-HD). A gyakorlatban főleg DD-s és HD-s lemezekkel találkozunk, bár néhány számítógép és floppyvezérlő, pl. az Enterprise 128 EXDOS kártyája a WD1772-es chippel képes volt az 5,25"-os DD-s lemezeket 80+ sávosra formázni, sávonként 9-10, vagy akár 11 szektorral, ami lényegében a QD-nek felel meg. Ehhez persze speciális FDD (jumperrel állítható írási sűrűség és fordulatszám [300 vagy 360 RPM]) és média (96 TPI [track-per-inch]) is kellett (volna).
A sávonkénti szektorszám is formátumfüggő: a DD-s PC-s lemezek általában 9 szektort tartalmaznak sávonként, kivéve a nagyon régi 320kB-os XT-s lemezek, amelyek csak 8-at. Az Amiga pl. 11 szektort ír a 3,5"-ös DD-s lemezekre sávonként. HD-s PC diszkek esetén 18 szektor a szokásos, de fel lehet menni 22-ig, ami viszont nem szabványos. Érdekes módon a sávonkénti 21 szektorral viszont a „modern" Windowsok elboldogulnak, 1680kB-os lesz így az eredetileg 1,44-es lemez.

A lemezen tárolt adatok kódolása

Az információ alapvetően 2-féle kódolással kerülhet a lemezre: vagy MFM (Módosított Frekvencia Moduláció), vagy GCR (Group Code Recording). Előbbit az IBM PC és a vele kompatibilis rendszerek használták (lemezformátum szempontjából a Videoton TVC és az Enterprise is ez a kategória), utóbbit pl. az Apple és a Commodore (1541-es termékvonal). Amennyiben „flippy", vagyis „fordigatós" lemezzel van dolgunk, az tovább bonyolítja a helyzetet. Akinek volt dolga 8-bites Commodore számítógéppel és hozzá dukáló FDD-vel, az tudja, hogy az 5,25"-ös lemezek másik oldalát is lehetett használni, mindössze az eredetivel azonos magasságban kellett még egy írásvédő ablakot lyukasztani a lemez túlsó szélére, mintegy az eredeti tükörképeként. Erre azért volt lehetőség, mert a 1541-es meghajtó az indexlyukkal nem foglalkozott, így fordítva berakva a lemezt már tudtuk is használni. Sajnos a modern (1985 után gyártott) 5,25"-ös PC-s drive-ok fel sem pörgetik a lemezt, ha nincs indexlyuk a megfelelő oldalon. Tehát megfordítva nem fogjuk tudni használni a lemezt, hacsak nem alkalmazunk valamilyen trükköt az indexlyuk figyelő rendszer kijátszására. Kézenfekvő a megoldás, hogy mivel a PC-s FDD-knek alapból 2 feje van, használjuk beolvasáshoz a túlsó oldalit! Igaz, hogy ebben az esetben a lemez forgási iránya éppen ellentétes lesz az eredetivel, de erre (a bitsorrend megfordítására) van megoldás a DTC-n belül (pl. a GUI-ban »Flippy Mode« kipipálása). Ami nagyobb gond, hogy eltolás van a lemez 2 oldala között a sávok vonatkozásában: a túloldalon 8 HD (vagy 4 SD) track-kel visszább vannak a sávok, de elvileg negatív tartományba nem mehetünk, csak ha a drive-ot fizikailag és elektronikusan is moddoljuk. Erről a használati útmutató csak fél mondatban tesz említést, de tartalmaz linket egy 3 részes, összesen félórás Youtube videóhoz (https://www.youtube.com/playlist?list=PLecGtGq1QOG_g9TFv-hmFRME4FsqFiz2ir&feature=view_all), ahol részletesen elmondják a mikéntjét. Még nem szántam rá magam az átalakításra, megelégedtem a C64-es lemezek „A" oldalának beolvasásával. A Commodore-os floppyjaimat már úgyis archiváltam az RR-Net MK3-asommal, csak kíváncsi voltam, hogy mivel tud többet a KryoFlux.

Nem mindig jön össze...

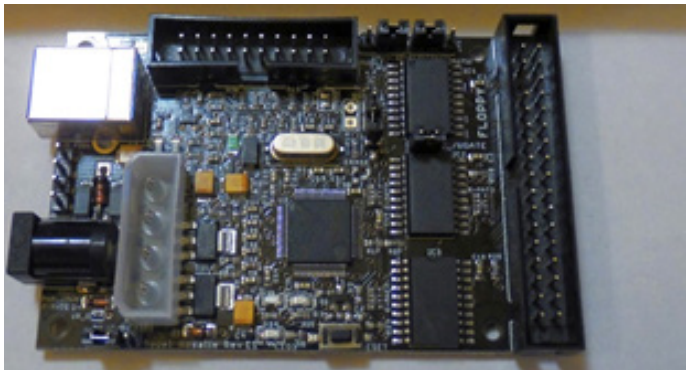
Sajnos a floppykat sem kíméli az idő vasfoga: mind a média hardveres része degradálódik (pl. elporlik a lemez felszíne, ahogy megadja magát a mágnesezhető réteget az alapkoronghoz rögzítő ragasztó), mind a rajta rögzített

mágneses jel gyengül, a tárolási körülményektől (hőmérséklet, páratartalom, por, penész, elektromágneses zajforrás) erősen függő mértékben.

Vannak olyan lemezek, amiket egyszerűen nem lehet hibátlanul beolvasni, és olyanok is akadnak, amiken elsőre végigszalad a folyamat hiba nélkül. Persze a lemezek többsége valahol a két véglet között helyezkedik el, ha nem is elsőre, de néhány nekifutás után sikerülni szokott. Ha tehetjük, próbálkozzunk többször, akár másik meghajtóval is. Arra is van lehetőség, hogy több, részlegesen hibás kísérlet eredményeit összefésülve akár hibátlan lemezképet kapjunk, én ehhez elmentettem a beolvasás eredményéről egy screenshot-ot, és később manuálisan összemásolgattam a legjobb állapotú sávokat, aztán erre egy következő körben engedtem rá a DTC-t deviceless módban (lásd „PC-s (vagy más, FAT-12 fájlrendszerű) lemezek beolvasása" című rész).

Összességében elmondható, hogy ha van reális esély az adott floppy beolvasására, akkor a KryoFlux-szal sikerülni fog!

Végül hálámat szeretném kifejezni ***Óré Andrásnak*** a 3D-nyomtatott házért és ***Petrovszki Lászlónak*** a floppyk és drive-ok egy részéért.



English Fun

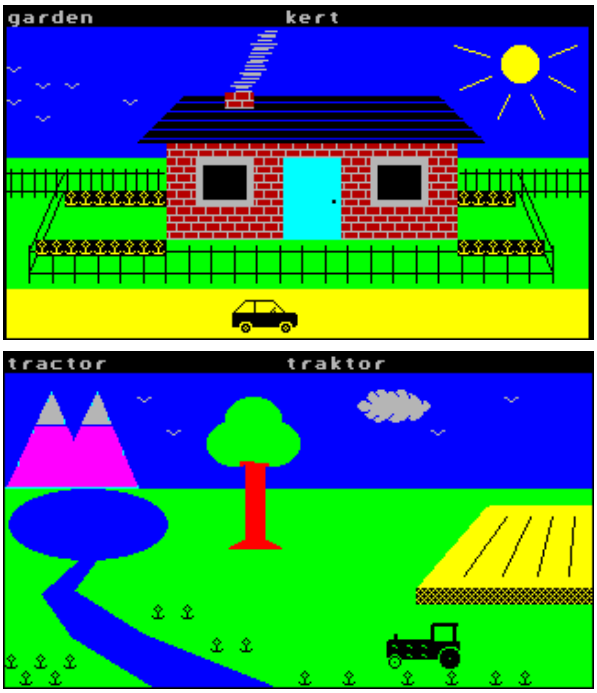
Grafikus-szöveges angol nyelvtanító BASIC program - kicsiknek és nagyoknak. A program indulása után kiválaszthatjuk, hogy melyik témakörbe tartozó szavakat / mondatokat gyakoroljuk:

1. Ház
2. Vidék
3. Strand
4. Mindegyik (az előző három kategória egymás után).
5. Számok
6. Program befejezése

A témakör kijelölése után az alábbiakból választhatunk:

1. Csak képek
A témához tartozó tárgyak képeit, angol elnevezéseit és magyar megfelelőit láthatjuk egymás után. A SPACE megnyomásával gyorsíthatjuk a felsorolást.
2. A képek teszttel
Kiválasztható, hogy a képek felrajzolása után a gép a megfelelő angol, vagy magyar szavakat „kérdezze ki". Rossz választ nem fogad el a program, nem megy tovább, újra próbálkozhatunk. Ha segítségre van szükségünk, nyomjuk meg a „?" billentyűt, a gép megadja a következő betűt.
3. Csak beszédgyakorlatok
A megjelölt témához kapcsolódó néhány alapvető mondat elsajátítására szolgál. Magyar mondatokat ad meg, melynek angol megfelelőit gombnyomásra tekinthetjük meg.
4. Képek és gyakorlatok
Az előző két üzemmód kombinációja. Először a szavakat, utána a mondatokat gyakorolhatjuk.

A számok gyakorlásánál először felsorolja a program néhány számot, angol megfelelőikkel, majd mi írhatunk be tetszőleges kétjegyű számot. A képen a leggyakrabban használt amerikai (nem SI) mértékegységgel megnevezését, átváltását (SI mértékegységbe) láthatjuk.
Bármelyik menüpontból a STOP megnyomásával visszalépethetünk a főmenübe.



Iview

Az Iview egy képnézegető program Enterprise-ra, ***Zozosoft*** munkája. A program szükségessége akkor vetődött fel, amikor 2008-ban az EPimgconv PC-s alkalmazással (IstvanV műve) megnyílt a lehetőség a PC-s formátumú képek Enterprise formátumúvá alakítására. Ezeket a PC-s képeket először önmagában működő Enterprise programmá alakította az EPimgconv (mely a képet egyből megjeleníti, külön képnézegető nélkül), ez azonban nem tette még lehetővé a képek szélesebb körű (pl. egyéb programokban történő) felhasználását. A program alapvetően ezen váltott-soros LPT-t használó képekhez készült. Ezek a képek 2, 4, 16, 256 színű vagy attribútum üzemmódúak is lehetnek, akár Interlace-ben is. A program maximálisan 46 karakter x 300 soros képméretet enged meg, ez két szín módban 736x300 pixel, interlace esetén 736x600 felbontást jelent, 4 szín vagy attributum mód esetén pedig 368x300 / 368x600 pixel. A képformátumban rejlő lehetőségekből az Enterprise Slideshow 2, -3 mutat egy kis ízelítőt (a képekhez kommentár talán nem szükséges...)

A program valamennyi változatot meg tudja jeleníteni, de emellett a VLOAD formátumú képeket is betölti. Használata egyszerű, így nem kell minden képgyűjteményből álló „demóprogramot" saját kezelőprogrammal ellátni. A programnak három változata van: EXT, ROM és COM formátumban is használható. A képek megjelenítésére három módszer lehetséges:

• A program lekezeli a modultöltés funkcióhívást, vagyis amikor a rendszerben van, akkor ha kiadunk LOAD parancsot egy RAW formátumú képre, az betöltődik. Billentyűnyomásra kilép a bejelentkező képhez.
• A Hsoft féle FILE bővítés jelenléte (megtalálható pl. a ZT 1.8 mellékleteként) esetén, ezzel tudunk a fájlok között böngészni. A kiválasztott képet betölti. Billentyűnyomásra visszatér a file-választáshoz, STOP-ra kilép. A FILE-s kiválasztáskor lehet szöveges listafájlt is indítani (formátuma mint az IVIEW.INI, erről később részletesen olvashatunk).
• Mielőtt a FILE-hez fordul, megpróbál megnyitni egy IVIEW.INI nevű fájlt. Ennek sima szöveges fájlnak kell lenni, amiben felsoroljuk a képfájlok neveit. Ha van INI fájl, akkor ezeket a képeket tölti be sorban, amíg el nem fogynak, és csak a végén jön a FILE-s fájl kiválasztás. Ha a listafájlban nem létező file-ra van hivatkozás, nem lép ki, hanem folytatja a következő képpel.

Ilyen listát könnyen létrehozhatunk:
• PC-n: DIR /B >IVIEW.INI
• EP-n: WP-ben DIR /W, majd a szövegből töröljük ki a felesleges sorokat, és F3-al mentsük el.
Mindkét esetben a felesleges fájlokat nem árt törölni a listából vagy a DIR paraméterezéssel csak a szükségeseket kilistázni, pl. ha RAW kiterjesztéssel mentettük el a képeket, akkor DIR *.RAW. (Ha kézzel kívánjuk szerkeszteni a listát, vegyük figyelembe, hogy a listában 32 vagy kisebb ASCII kódú karakterek minősülnek elválasztó karakternek, így lehet SPACE, CR, LF, TAB.)

A program kezelése nagyon egyszerű:
• Slideshow-nál ha ESC-t vagy STOP-ot nyomunk, befejezi a lista végrehajtását, visszalép a FILE kiválasztáshoz. Egyéb billentyű megnyomásával a képeket léptethetjük.
• Szintén slideshow üzemmódban, a funkció billentyűkkel lehet a képváltás sebességét állítani, F1 a leggyorsabb, F8 a leglassabb.
• A FILE-kiválasztás képernyőn STOP-ot nyomva kiléphetünk a programból.

A program működik magnós konfigurációban és EP64-en is, itt .COM verziót célszerű a kazetta elejére venni, és aztán mehetnek a képek sorban. 64K memóriával természetesen a ROM verziót kell berakni a konfigba, vagy a .COM-ot futtatni. Jelenleg nincs bántva a rendszer, EXOS határig tölt, így nagy nagy interlace kép estén a második félkép alján szmötyi látszik. Normál képekkel természetesen nincs gond.

IS-FORTH - 6. rész

Intelligent Software - 1985.
rendszerbővítő, FORTH programozási nyelv

A nyomtató programnak a következő dolgokat kell 60 sor kiírása után tenni:

```
: LAPDOB ( - - - - )
12 EMIT ( lapdobás )
0 SORSZ ! ( a sorszámlálás újrakezdése )
LAPSZ @ . (lapszám kiírása )
." lap" CR
LAPSZ INC ( a lapszámláló növelése )
;
```

A kiírást mindjárt lapdobással fogjuk kezdeni; ez a megfelelő pillanat a LAPSZ kezdőértékének beállítására is:

```
: ELSO-LAPDOB ( - - - - )
1 LAPSZ !
LAPDOB ;
```

A listázóprogram dolga lesz, hogy a listázást az ELSO-LAPDOB hívásával kezdje. Az átdefiniált CR:

```
: CR ( - - - - )
SORSZ INC (növeljük a sorszámlálót )
SORSZ @ 60 = ( lap vége? )
IF LAPDOB
ELSE CR
THEN
;
```

A FORTH interpreter maga is használ változókat, ezek az ún. rendszerváltozók.

Példa a rendszerváltozóra: A BASE
A BASE (ejtsd: béz, jelentése: alap) változó a kiíráskor, beolvasáskor használt számrendszer alapszámát tartalmazza. Eddig ez mindig 10 volt. Ha mondjuk a 2-es számrendszerben akarunk dolgozni, akkor ennek a változónak az értékét 2-re állítjuk:

```
2 BASE !
```

Próbálgassuk, milyen az élet kettes számrendszerben:

```
1 1 + = 10
```

Nem, a 9-es számjegyet most nem veszi be az interpreter „gyomra", hiszen kettes számrendszerben vagyunk, ahol ennek semmi értelme. Meglepetten tapasztalja viszont a kísérletező Olvasó, hogy a 2, 3 számok - amelyek szintén „értelmetlenek", működnek:

```
2 . = 10
```

Ennek az az oka, hogy az első néhány szám benne van a szótárban; ezeket gyakran használjuk. Az interpreter munkáját gyorsítja, hogy „készen találja" őket, nem kell konvertálnia. A leggyakrabban használt két számrendszer a 10-es és a 16-os; ezek beállítására vannak FORTH alapszavak. Az is IS-FORTH-ban ezen kívül a 8-as és a 2-es számrendszerre is van szava. Működésük a forrásszövegükből világos:

```
: DECIMAL 10 BASE ! ;
: HEX 16 BASE ! ;
```

Hogyan tudhatjuk meg, hogy az adott pillanatban mi a konverzió alapszáma? Egyszerű, mondhatnánk meggondolatlanul, csak elő kell venni a változóból az alapszámot és kiíratni:

```
BASE @ .
```

Mit tudtunk meg ebből? Hogy a számrendszer alapszámát az adott rendszerben egy 1 és egy 0 jegy ábrázolja, vagyis hogy megegyezik a számrendszer alapszámával. Többre megyünk, ha az eggyel kisebb számot íratjuk ki:

```
BASE @ 1- .
```

Ebből a válaszból tudhatjuk, hogy pl. F, azaz 15 a legkisebb egyjegyű szám, ezek szerint a 16-os számrendszerben vagyunk.
Állítsuk most 5-re az alapszámot, és definiáljunk új szót az 5-ös számrendszerben:

```
5 BASE !
: KIIR 10 .
KIIR
```

Mi történik, ha a KIIR-t a 10-es számrendszerben próbáljuk ki?

```
DECIMAL KIIR
```

A kapott válasz. 5. A szótári szavakban binárisan vannak a számok, a KIIR definíciójakor

0000 0000 0000 0101
került oda, ez a tízes számrendszerben 5.

Írjunk egy .BASE (- - - -) szót, amely a BASE tartalmát decimálisan írja a képernyőre, de nem rontja el! (vagy pontosabban: elrontás után visszaállítja.)

```
: .BASE ( - - - - )
BASE @ DUP
DECIMAL
.
BASE !
;
```

Példa a rendszerváltozóra: Az OUT
Tudjuk már, hogy a képernyőre író FORTH alapszavak mind az EMIT-tel írják ki az egyes karaktereket. Az EMIT eggyel növeli az OUT (ejtsd: aut, jelentése: ki) változó tartalmát. Így, ha az OUT-ot nullázzuk, utána bármikor megtudhatjuk, hogy a nullázás óta hány karakter került a képernyőre.
Képzeljük el például, hogy egyforma sorokat akarunk egymás alá írni, amelyek két (előre nem tudni, milyen hosszú) számból állnak; azt kívánjuk, hogy az első szám a képernyősor 3., a második pedig a 13. pozíciójában kezdődjék. Feltesszük, hogy a két szám a vermen van. Megírjuk azt a szót, amely a fentiek szerint írja ki őket:

```
: TABULAL
  CR
  0 OUT !
  3 SPACES .
  13 OUT @ -
  SPACES
  .
;
        ( n1 n2 - - - - )

( „ nullázzuk” az OUT változót )
( 3 szóköz, kiírjuk az első számot )
( 13-ra egészítjük ki a kiírt karakterek számát)

( kiírjuk a második számot )
```

Konstansok
Ha egy értéket úgy akarunk megőrizni a szótárban, hogy soha nem változtatjuk, akkor egyszerűbb olyan szavakat használni, amelyek az értéknek nem a címét, hanem magát az értéket adják a vermen. Ezek a konstansok. Ilyenek például az 1, 2, 3 szavak, amelyek benne vannak a szótárban és az 1, 2, 3 értéket teszik a veremre.
A Forth kétféle konstanst ismer: 16 és 32 bitest. Leggyakrabban az egyszeres pontosság (16 bit) használatos. A konstansokat használat előtt deklarálni kell. A definíció a következő szóval történik:

```
CONSTANT ( n - - - - )
```

A dolog meglehetősen hasonlít a változók definiálására: (mennyiség) CONSTANT (név). Például:

```
42 CONSTANT CSIL
```

ezzel a CSIL nevű konstans szót definiáltuk. A CSIL 42-t tesz a veremre. A következő példa ezek után már egyértelmű:

```
365 CONSTANT NAPOK
```

Miután ez bekerül a Forth szótárába (például úgy, hogy begépeljük), bármikor hivatkozunk a NAPOK szóra, a rendszer a verembe 365-öt helyez el későbbi használat céljára.

Példa:
Ez az egyszerű program pénznemeket konvertál. Jelenleg mondjuk egy angol font 67 forint. Ha fontból forintot akarunk számolni, egyszerűen az átváltási számmal meg kell szorozni a fontok számát. Az átváltási arány konstans, ugyanis a program futása alatt nem változik azért annyira nem gyors az infláció). Az alkalmazásunk a következőképp fog kinézni:

```
67 CONSTANT FORINT
: FONT CR FORINT * . .” forint” CR ;
```

Ha 15 fontot akarunk forinttá számolni, a következő: kell beütni:

```
15 FONT
```

A válasz a következő lesz: 1085 forint.
Ha egy nagyon nagy alkalmazásunk van, ami sok helyen használja az átváltást és megváltozik időközben az árfolyam, akkor egyszerűen a deklarációt kell megváltoztatni a programban. Ha szükséges, egy konstans egy alkalmazáson belül is újradeklarálható.

Dupla pontosságú konstansok
32 bites konstanst egy másik szóval deklarálhatunk, melynek működése pontosan ugyanolyan, mint az előbb megismert CONSTANT. A következőképp használatos:

```
234000. 2CONSTANT NAGYSZAM
```

Ezután NAGYSZAM végrehajtása esetén 234000 kerül a verem tetejére duplapontosságú formában. A szám után lévő pont jelzi, hogy duplapontosságú számról van szó.

3.5. Hol változik a változó? Ismerkedés a szótárral

A szótármutató
A FORTH programozás során a szótár mérete állandóan változik. Hogy éppen hol van a szótár „teteje”, a szótár utáni első szabad hely, ahova a következő szótári szó kerül majd, azt az interpreter a DP (Dictionary Pointer, ejtsd: diksöneri pointer, jelentése: szótármutató) rendszerváltozóban tartja. A DP változó értékét sokat használjuk, ezért kiolvasására külön alapszó van:

```
HERE ( - - - - cím )
```

A szó forrásszövege:

```
: HERE ( - - - - cím ) DP @ ;
```

Már korábban volt róla szó, hogy a pad a szótár tetejétől állandó távolságra van. A jelenséget megmagyarázza a PAD szó forrásszövege:

```
: PAD ( - - - - cím ) HERE 68 + ;
```

Mi van a szótárban?
Egy szótárelem a következő részekből áll:

névmező: a szó neve;
láncmező: mutató a szótár előző szavára (ha nincs előző szó, 0);
kódmező: mutató a kódra, amely a szó futásakor végrehajtódik;
paramétermező: a szó végrehajtásához szükséges többi adat (pl. változóknál a változó értéke).

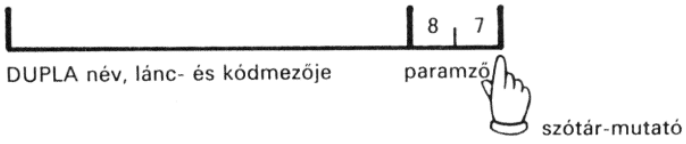
A paramétermező --- teljesen önkényesen ‚paramzőnek" rövidítjük - fizikailag a szó legvégén van. Így változó definiálása után a szótármutató pontosan az új változó paramzője mögé mutat. Ha ekkor a szótármutató értékét 1-gyel, 2-vel stb. növeljük, akkor ezzel megtoldjuk a változó paramzőjét 1, 2 stb. byte-tal. A szótármutató növelésére szolgáló alapszó, az

```
ALLOT ( n - - - - )
```

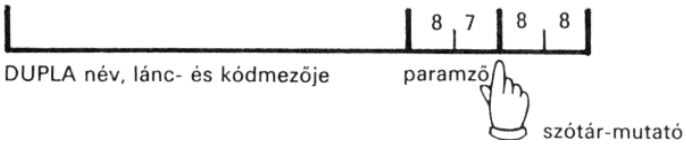
forrásszövege:

```
: ALLOT ( n - - - - ) DP +! ;
```

Hosszú változók
Ezzel a kezünkben van a lehetőség arra, hogy duplaszavas vagy hosszabb változókat definiáljunk.

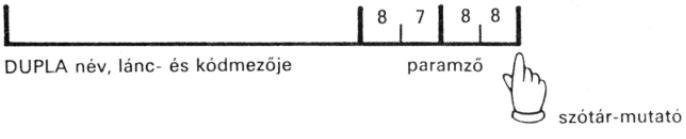



```
87 VARIABLE DUPLA
```




```
88 HERE !

2 ALLOT
```



A DUPLA tehát két szavas, az első szó kezdőértéke 87, a másodiké 88. A paramzőbe való kezdőérték-tárolásra, egyúttal a paramző „tágítására" valók a következő alapszavak:

```
, ( n - - - - )
```

(vesző) Az n-et a HERE címen, egy szón tárolja, a szótármutató (DP) értékét 2-vel növeli.

```
C, ( C - - - - )
```

Az n-et a HERE címen, 1 byte-on tárolja, a szótármutató (DP) értékét 1-gyel növeli.

A két szó forrásszövege:

```
: , ( n - - - - )
HERE !
2 ALLOT;

: C, ( c - - - - )
HERE C!
1 ALLOT ;
```

Most már tudunk vektorokat, hosszabb adatsorokat definiálni. Fontos tudni, hogy erre vannak a FORTH-nak sokkal kényelmesebb eszközei is, itt csak azért definiálunk vektort, hogy jobban értsük a FORTH működését.
Végül következzen négy, a változók használatát segítő szó:

?	(addr - - - -)	A változó címéről kiírja a változó értékét.
0!	(addr - - - -)	A változót nullázza.
1+!	(addr - - - -)	A változó értékét eggyel növeli.
1-!	(addr - - - -)	A változó értékét eggyel csökkenti.

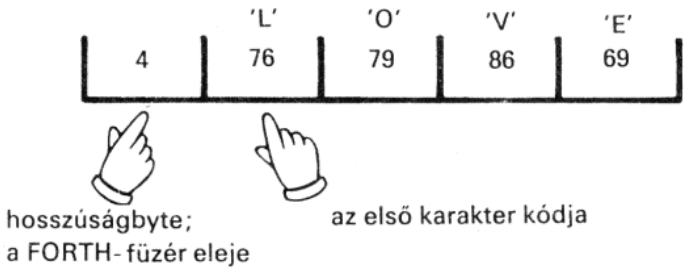
3.6. A FORTH -füzér és a WORD

A FORTH-füzérek (karakterláncok)
Kezdetben a szövegek kezelésének megismerése és megszokása egy kis türelmet vesz igénybe a Forthnál. Nincs például az INKEY$-nak megfelelő operátor. A legtöbb szöveg konstansok és változók formájában van jelen. A szövegek, számok, amelyeket beolvasunk vagy kiírunk, a memóriában karakterfüzéreket alkotnak. (A karakterfüzér angol elnevezése string, ejtsd: sztring.) Egy ilyen füzérrel akkor tudunk dolgozni, ha ismerjük a kezdőcímét és a hosszát. A FORTH-ban gyakori tárolási mód: a füzért az ún. hosszúságbyte-tal kezdjük, amely a füzér hosszát tartalmazza, és ezután jönnek a karakterkódok. Az így tárolt füzért FORTH-füzérnek nevezzük. Ismerkedjünk meg a

```
COUNT ( ff-cím - - - - cím hossz )
```

alapszóval, amely a FORTH-füzér címéből előállítja az első karakter címét és a füzér hosszát (azaz a TYPE kiíró

szó hívásához szükséges paramétereket)! A „LOVE” szót tartalmazó FORTH füzér byte-jai:



A COUNT szó forrása:

```
: COUNT ( ff-cím - - - - cím hossz )
DUP 1+
SWAP C@
;
```

FORTH-füzér alakjában szolgáltatja a szövegeket a FORTH interpreter egyik lelke, a WORD szó is.
Egy string operátorral már találkoztunk:

```
." szöveg"
```

Ennek korlátozott használhatósága miatt általánosabb szövegtárolási és kezelési formákra is szükség van. A string kezelő szavak a PAD-et használják. Ez csak átmeneti célokra használható, terület, mert az új szavak átírhatják a PAD-et.

A string változó
Ahogy a numerikus változóknál szó volt róla, a változó nevének végrehajtása azt eredményezi, hogy a változó címe bekerül a verembe, készen arra, hogy a további operátorok dolgozzanak vele. Ez ugyanígy megy a string változóknál is, annyi különbséggel, hogy a string első byte-ja a szöveg hosszát tartalmazza. Ez azt jelenti, hogy egy karakterlánc maximum 255 byte hosszú lehet.
A string változó nevét és maximális hosszát deklarálni kell, innen tudja a fordító, mennyi helyet kell lefoglalni az adott szöveg változónak. Példa:

```
80 $VARIABLE FRED$
```

Ez egy string változót deklarál, maximum 80 byte hosszt FRED$ néven. Jó ötlet a karakterlánc változók végé dollár jelet tenni (de nem kötelező), mert a programban így könnyű megkülönböztethetők a string változók a numerikus változóktól. A FRED$ végrehajtásakor a string kezdőcíme bekerül a verembe.

A string konstansok
Ez főleg olyan esetekben használatos, amikor egy változatlan szövegre a programban több helyen akarunk hivatkozni. A deklaráció a következőképpen néz ki:

```
„ Hello” $CONSTANT UDVOZLET$
```

Ez letárolja a „Hello” szöveget a szótárban, és amikor az UDVOZLET$ végrehajtódik, akkor a string címe bekerül a verembe.

String változók kiírására a már említett

```
TYPE
```

használhatjuk. A formája a következő: (cím) (a byte-ok száma) TYPE
pl:

```
UDVOZLET$ 5 TYPE CR
```

Nem egészen azt kapjuk amit vártunk, az „o” lemaradt a végéről. Megemlítjük újra, hogy a string első byte-ja a hossz-byte (azaz a példában 6 karaktert kellene kiiratnunk). Egy olyan operátorra van szükségünk, ami leveszi a számlálót a szöveg elejéről és egyet hozzáad a címhez, hogy az valóban a szöveg kezdetére mutasson. Így olyan formához jutunk, amit a TYPE vár. Ez az amit a COUNT megtesz. Például:

```
UDVOZLET$ COUNT TYPE CR
```

Ezek alapján definiáljunk egy PRINT szót, ami kiír egy stringet:

```
: PRINT COUNT TYPE ;
```

Ez egy nagyon hasznos definíció. Használatakor mindössze a stringváltozó vagy konstans nevét és az PRINT-et kell leírni:

```
UDVOZLET$ PRINT
```

A fenti példa a COUNT és a TYPE bemutatására szolgáltak. Valójában a Forth-ban létezik egy olyan string operátor, ami pontosan a PRINT feladatát látja el. Ez a szó a

```
$
```

(string kiírás). Példa:

```
UDVOZLET$ $.
```

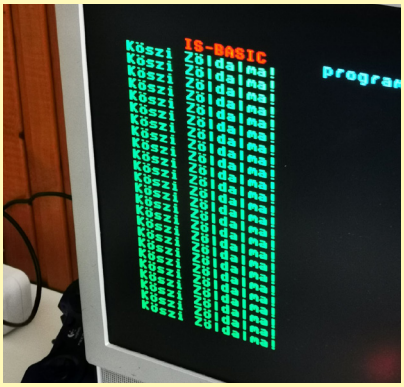
A String változók inputja
Erre többféle lehetőség van. Az EXPECT operátor egy olyan karaktersorozatot vesz be a billentyűzetről, ami egy kocsi vissza (RETURN, CR) karakterrel lett lezárva, és kevesebb karaktert tartalmaz, mint amennyit megadtunk. (de Maximum 255 karaktert).
Példa:

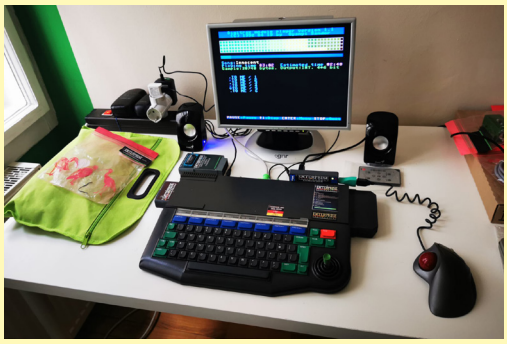
```
PAD 255 EXPECT
```

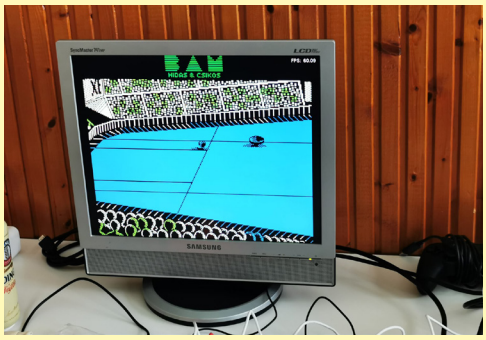
Ez maximum 255 karakteres szöveget vár, ami az ENTER leütésével lett lezárva, és tárolja a PAD-ben. Nem használhatjuk ezt közvetlenül string változók fel töltésére, mert EXPECT nem adja meg a szöveg hosszát. A SPAN szó az EXPECT után kiadva a verembe teszi azt a címet ahol a Forth a bevett szöveg hosszát tárolja.

Folytatjuk!

ENTERPRISE Klub - 2022. 04. 09.



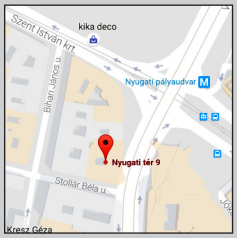




ENTERPRISE KLUB

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 - 2022/1-2. január–április
Főszerkesztő: Matusa István
Szerkesztőségi főmunkatárs: Németh Zoltán (Zozosoft)
A csapat: geco, Povi, Kiss László, SzörG, szipucsu, lgb, Bakó Róbert, Tamási Istvánné, Virág Attila, Dr.OG
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/magazin