ENTERPRESS

Magazin az ENTERPRISE felhasználóknak

2019/5-6. szeptember–december

Egérkezelés a Lemmings-ben

ENTERPRISE pólók!

Egérkezelés a Lemmings-ben I.

Írta: Povázsay Zoltán (Povi)

Az EnterMice megjelenésével (köszönet érte lengyel barátunknak, Pear-nek) kedvenc számítógépünkhöz most már gond nélkül csatlakoztathatunk PS/2 egeret. Azonban mit sem ér a hardver megfelelő szoftver nélkül, kis túlzással sajnos szinte egy kézen meg lehet számolni az egér használatát támogató programokat, játékokat. A szoftverhiányt kétféle képpen lehet bővíteni: az ember vagy ír magának egyet, ami kezeli az egeret (ilyen lesz a készülő WallPipe játék), vagy egy már meglévő programba valahogy beleerőszakolja az egérkezelő rutint (ilyet műveltem Miklán Attila barátunk Turbó Amőba játékával).

Ebben a cikkben ez utóbbi módszert mutatom be a vállalkozó kedvű olvasónak, vagyis azt, hogy hogyan lehet egy régi játékot egérrel irányíthatóvá tenni. A cikk feltételezi a Z80 assembly ismeretét, szükség lesz a disassemblált kód értő olvasására; szóval vigyázat: mélyvíz következik!

A műtétre kiszemelt játék a DMA Design 1991-ben megjelent játéka, a **Lemmings**. Bizonyára sok olvasónk ismeri ezt a játékot, én is sokat játszottam vele anno PC-n, egérrel. A játéknak készült egy ZX Spectrum változata is (ami sajnos nem a legszebben sikerült port), aminek később EDC-nek (alias egzo, alias **Vincze Béla György**) köszönhetően készült el Enterprise-ra átírt változata. A játék értelemszerűen egérrel nem, csak joystickkal és billentyűzettel irányítható, viszont „üvölt" az egérkezelésért! Lássuk akkor a medvét!

Maga az egérkezelő rutin, ami olvassa a egérgombok állapotát, és az elmozdulást, IstvanV műve, ezt a rutint fogom én is felhasználni (lásd: EnterMice leírás a Wiki-n, „A still more condensed Mouse+all buttons+wheel reading routine, thanks by IstvanV" címen).

A munka megkezdéséhez az első, és megkerülhetetlen lépés a LEMMINGS.COM fájl visszafejtése. Erre több okból is szükségünk van: először is, ebből tudjuk meg, hogy a játék többi fájljai (az SCR, PR1, PR2 és PR3 kiterjesztésűek) hova töltődnek be; ezekre a címekre szükségünk lesz majd a többi fájl diszasszemblálásához. Másrészt, mivel a Spectrum-on az alsó 16kB-os tárterületen ROM található, ezért a saját rutinjainkat is erre a területre tudjuk majd elhelyezni, ezt praktikusan a LEMMINGS.COM fájl mögé hozzáfűzve tudjuk majd megtenni.

Az első fájlt visszafejtve (én a dZ80 programot szoktam használni – inkland.org.uk/dz80/) láthatjuk, hogy a fájlokat beolvasó rutin a 0x0222 címen található, ami az IX regiszter által mutatott táblázat alapján tölti be a file-okat a megfelelő helyre. Lássuk a táblázat egy bejegyzését (a példában a LEMMINGS.PR1 fájlhoz tartozó adatokat):

```
030D    db      „1"
        db      0xfa
        db      0xfb
        dw      0x5b00
        dw      17344
```

A táblázat összesen 28 byte-os, ami a betöltendő 4 file egyenként 7 byte-os adatait tartalmazza. Az első byte a file-kiterjesztés utolsó karakterét tartalmazza. A második és harmadik bájt (0xfa és 0xfb) azt mutatja meg, hogy melyik szegmenseket kell belapozni a 2-es és a 3-as lapra. A következő word (0x5b00) mutatja azt a memóriacímet, ahova a fájlt be kell tölteni. Az utolsó word pedig a betöltendő fájl méretét (17344 byte) adja meg. Ezen adatok birtokában már könnyen a helyes címekre tudjuk visszafejteni a PR1, PR2 és PR3 fájlokat. Itt természetesen nem törekszünk a teljes visszafejtésre, ami után a forrásfájl szerkeszthető és újrafordítható lesz, nekünk épp csak annyi elég, hogy a minket érdeklő részekre rátaláljunk.

Mik is azok a dolgok, amik minket érdekelnek? Alapvetően két területet kell feltárnunk: mivel az egérkezelő rutinunkat 50 Hz-es megszakításban szerertnénk hívni, jó lenne tudni, hogy a Lemmings hogyan kezeli a megszakításokat, majd ha ez meg van, akkor majd be tudjuk valahogy injektálni a saját kódunkat. A másik terület, amire kíváncsiak vagyunk, az maga az irányítás: meg kell keresni azt a részt a kódban, ahol a játékmenet közbeni billentyű figyelése van (Q, A, O és P), és felderíteni, hogy az hogyan működik.

Elsőként nézzük a megszakítást. Feltételezzük, hogy a játék saját megszakítási rutint alkalmaz, ezért valahol biztosan átírja a 0x0038 címen található megszakítási rutin elejét, hogy eltérítse azt a saját rutinjára. A LEMMINGS.PR2-ben találhatunk is egy gyanús részt, ami pont ezt teszi:

```
8F3C    ld      a, 0xc3         ; JP opcode
8F3E    ld      hl, 0xb6e2
8F41    ld      (0x0038), a
8F44    ld      (0x0039), hl
```

Nézzük csak, mi található a 0xb6e2 címen!

```
B6E2    di
B6E3    push    af
B6E4    push    bc
B6E5    push    de
B6E6    push    hl
B6E7    exx
B6E8    ex      af,af'
B6E9    push    af
B6EA    push    bc
    ...
```

Ez valóban úgy néz ki, mint egy megszakítási rutin: az elején tiltja a megszakításokat, majd elmenti a regiszterek értékét a verembe. Így nincs is más dolgunk, hogy a 0x8f3f címen lévő megszakítási rutin címét átírjuk a saját rutinunk címére. A saját megszakítási rutinunk pedig így néz ki:

```
MyInt:  di
        push af
        push bc
        push de
        push hl

        in   a, (0xb4)
        and  0xaa
        ld   b, a
        neg
        and  b
        bit  5, a
        call nz, MouseRoutine

        jp   0xb6e7
```

A rutin elején tiltjuk a megszakításokat, elmentjük a verembe az AF, BC, DE és HL regisztereket, majd megnézzük, hogy 50Hz-es megszakítás okozta-e a megszakítást, és ha igen, meghívjuk az egérkezelő rutinját. Végül az eredeti (a 0xb6e2 címen kezdődő) megszakítási rutin közepére ugrunk, a PUSH HL utáni EXX utasításra. A video-megszakítás detektálására valószínűleg van egyszerűbb módszer is, én az EXOS ROM-ból merítettem az ötletet (lásd: ROM0 0xc4c1-en kezdődő részt).

Nézzük, mi van a MouseRoutine függvényünkben:

```
MouseRoutine:           ; mouse routine
                        ; by IstvanV
        ld   c, 4
rLoop1: ld   h, l
        ld   l, d
        ld   d, e
        ld   e, 1
        ld   a, 2       ; RTS low
rLoop2: out  (0xb7), a
        ld   b, 17      ; 25.6 us at
                        ; 10 MHz without
                        ; wait states
mWLoop: djnz mWLoop
        ld   b, 4
rLoop3: ld   a, b
        out  (0xb5), a
        in   a, (0xb6)
        rra             ; data is read
                        ; from K column
        rra
        rl   e
        djnz rLoop3
        ld   a, b       ; RTS high
        jr   nc, rLoop2
        dec  c
        jr   nz, rLoop1
        out  (0xb5), a  ; A = 0
        in   a, (0xb6)
        bit  7, d       ; check if mouse
                        ; detected
        ret  nz         ; return if no
                        ; EnterMice detected

                      ; falls to HandleMouse
HandleMouse:
                        ; TODO
        ret
```

A függvény eleje, ami az egér gombokat és az elmozdulás irányát és nagyságát olvassa IsvanV műve. Ha a függvény EnterMice hardvert detektál, akkor futását folytatja a HandleMouse nevű eljárásban, (ami egyelőre még üres), egyébként visszatér a saját megszakítási rutinunk végére. A HandleMouse az alábbi paramétereket kapja:

A regiszter: a b1 törölt állapota jelenti a bal oldali egérgomb megnyomását

H regiszter: horizontális relatív elmozdulás (előjeles érték, pozitív irány balra)

L regisztrer: vertikális relatív elmozdulás (előjeles érték, pozitív irány felfelé)

Ahhoz, hogy a HandleMouse eljárásunkat megírhassuk, jobban bele kell ásnunk magunkat a játék lelkivilágába!
A játék közben a kurzort a Q, A, O és P billentyűkkel irányíthatjuk, és az M billentyűvel választhatjuk ki a lemminget munkára. Próbáljuk megkeresni azt a részt a programban, ami az imént felsorolt billentyűk lenyomását figyeli. Az Enterprise-on a a billentyűzet olvasása a 0B5H port írásával és olvasásával történik, mi is ilyen műveleteket keresünk most.

A LEMMINGS.PR2-ben rábukkanhatunk erre a függvényre, gyanús, hogy ez kell nekünk:

```
FE0D    OUT   (0B5H), A
FE0F    IN    A, (0B5H)
FE11    RET
```

A függvény az A regiszterben várja a billentyűzet-mátrix sorának a számát (0..9), majd visszatér a sor billentyűinek állapotával (a törölt bit jelenti a lenyomott billentyűt). Jó,

ha előttünk van az Enterprise billentyűzet-mátrixa, hogy lássuk, a számunkra fontos billentyűk melyik sor-oszlop pozicíót foglalják el a mátrixban.
Az imént megtalált, 0xfe0d címen lévő billentyűzet olvasó függvény hívása valóban sokszor előfordul a kódban, és minden egyes esetben egy „LD A, imm" utasítás előzi meg. Gyorsan keressünk is rá az „M" billentyű figyelésére, ami a mátrix 8. sorában található:

```
C7B9    LD    A, 8
C7BB    CALL  FE0D     ; out 0b5h, a / in a
C7BE    BIT   0, A       ; „M" pressed?
C7C0    JR    NZ, C7C7   ; jump if not
                         ; pressed
C7C2    LD    A, 255
C7C4    LD    (F56D), A
C7C7
```

Mit látunk? Ha lenyomjuk az „M" billentyűt (kijelölünk egy lemminget), akkor a F56D címen található változó értéke 255 lesz. Ugyanezt a működést várjuk el akkor is, ha a bal egérgombot nyomjuk le, egészítsük ki gyorsan a HandleMouse eljárásunkat!

```
HandleMouse:
; check left mouse button
        and  2
        jr   nz, noBtn
        dec  a           ; A = 255
        ld   (0xf56d), a
noBtn:
; TODO: check mouse cursor movement
        ret
```

Mint ahogy arról korábban volt róla szó, a HandleMouse eljárás az A regiszterben kapja ez egérbillentyű állapotát: a regiszter 1-es bitjének törölt állapota jelenti azt, hogy a bal egérgomb lenyomott állapotban van. Az „AND 2" utasítással az 1-es bit kivételével töröljük az A regiszter bitjeit. Ha az eredmény nulla, akkor az egérgombot lenyomták: az F56D címen lévő változó értékét 255-re állítjuk be. Ha az eredmény nem nulla, akkor az értékadást kihagyva, a noBtn címre ugorva folytatjuk a program futását.
Nézzük a kurzor mozgatását felfelé! A „Q" billentyűt figyelő rész itt található:

```
lc7e1:  ld    a,02h    ; select keyboard
                       ; matrix row 2
        call  lfe0d    ; out 0b5h, a / in a
        bit   1,a        ; „Q" pressed?
        jr    nz,lc82b   ; no -> jump to
                         ; check down
                         ; direction
lc7ea:
```

Majd így folytatódik:

```
lc7ea:  ld    a,(0f57ah)
        bit   0,a
        jr    nz,lc821
        ld    a,01h
        ld    (0f57ah),a
        xor   a
        ld    (0f57bh),a
lc7fa:  ld    a,(0f56bh)
        cp    00h
        jp    z,lc891
        dec   a
        ld    (0f56bh),a
        ld    a,01h
        ld    (0f56ch),a
        ld    a,(0f57bh)
        bit   2,a
        jp    z,lc891
        ld    a,(0f56bh)
        cp    08h
        jr    c,lc891
        sub   08h
        ld    (0f56bh),a
        jr    lc891
lc821:  ld    hl,0f57bh
        bit   2,(hl)
        jr    nz,lc7fa
        inc   (hl)
        jr    lc7fa
```

Ez így egy kicsit nehezen olvasható, alakítsuk át a fenti kódot C nyelvre:

```
if (varF57A & 1 == 0)
{
    varF57A = 1;
    varF57B = 0;
}
else if (varF57B < 4)
{
    varF57B += 1;
}
if (varF56B != 0)
{
    varF56B -= 1;
    varF56C = 1;
    if (varF57B == 4 && varF56B >= 8)
    {
        varF56B -= 8;
    }
}
```

Így se sokkal jobb, de már átlátható a logikája, és kikövetkeztethetők a változók jelentése.

A játékban a kurzor sebessége függ attól, hogy menynyi ideje tartjuk lenyomva a kurzormozgató billentyűket. Rövid idejű lenyomásra a kurzornak finom (pixelenkénti) mozgása van, de a billentyűt nyomva tartva a kurzor mozgása gyorsul, egészen a játékmező széléig, ahol nyomva tartott billentyű mellett is visszalassul a sebessége a pixelenkénti mozgásra.

Ha a fenti működési elvet ismerjük, akkor könnyebben értelmezhető a feni kód is. Mivel a billentyűzet figyelése minden videomegszakításban történik, valahogy könyvelni kell, hogy a lenyomott billentyűt vajon épp csak most nyomtuk le, vagy már az előző megszakítási ciklusban is le volt-e nyomva. Pontosan erre szolgál a varF57A változónk, ami valójában egy bitfield: értéke 1, ha a „Q" billentyű le volt már nyomva, értéke 2, ha az „A" billentyű volt lenyomva, és értéke 0, ha se nem az „A", se nem a „Q" billentyű nem volt lenyomva az előző megszakításkor.

Folytatjuk

DRVTEST (by Bruce Tanner)

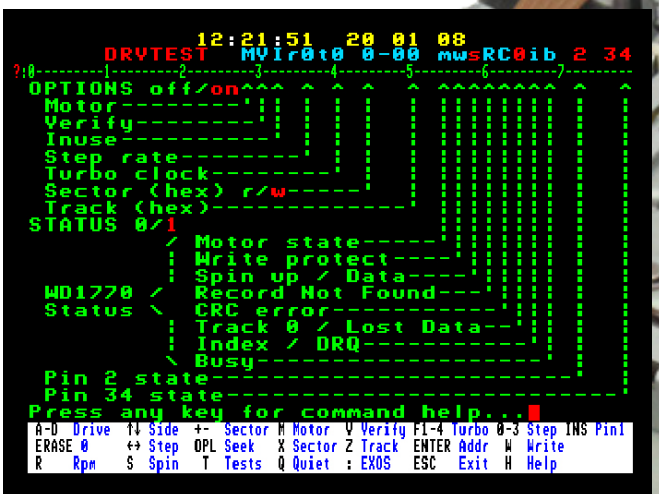
Írta: Németh Zoltán (Zozosoft)

Megjelent a DRVTEST! Ez egy kis alkalmazás hajlékonylemez-meghajtók és EXDOS kártyák teszteléséhez, diagnosztizálásához és vizsgálatához.

Azért készült, hogy segítsen az EXDOS 3 fejlesztésében, valamint segítsen a régi hardverek életben tartásában.

A DRVTEST.COM betölthető a BASIC-ből, vagy futtatható IS-DOS parancsként, vagy a DRVTEST.ROM behelyezhető egy ROM-ba, és a következővel indítható :DRVT.

Egyes helyeken meglehetősen technikai, de a H gombbal elérhető HELP leírás. Vagy csak válassz egy meghajtót A vagy B gombbal, majd nyomd meg a T gombot a tesztek futatásához és kezdődik a meghajtó vizsgálata!



Enterprise Forever

mobil alkalmazás Android készülékekre

Elkészült az Enterprise Klub mobil alkalmazása!

Első körben Android készülékeken lehet futtatni.
A Google Play alkalmazásból lehet letölteni.

Az **Enterprise Klub** mobil alkalmazásán keresztül megismerheted klubunk történetét, az Enterprise számítógéppel foglalkozó weboldalak linkjeit, valamint elérhetőségeinket.
Klubtagok számára friss hírekkel szolgálunk a klubbal és az Enterprise számítógéppel kapcsolatban, valamint láthatják a klubtalálkozók időpontjait és a találkozóval kapcsolatos fontos információkat.

Üzenőfal menüpontunk jelenleg még fejlesztés alatt áll. Itt a későbbiekben klubtagjaink üzenhetnek egymásnak, valamint a klub vezetőségének.
Az alkalmazás főoldalán közvetlen link található az Enterprise Klub weboldalára és az Enterpress Magazin elektronikus verziójára, melyet mobil készüléken is lapozgathatsz.

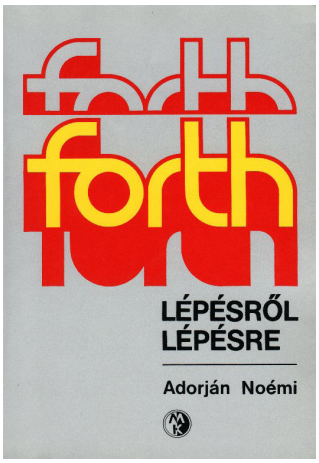
Kellemes böngészést kívánunk! Enterprise Forever!

Direkt link:
https://play.google.com/store/apps/details?id=hu.naturami.app.enterpriseforever

IS-FORTH

Intelligent Software - 1985
rendszerbővítő, FORTH programozási nyelv

Bevezetés



Az alábbi „rövid ismertető” a FORTH programozási nyelv, pontosabban annak - az ***Intelligent Software Ltd*** által megvalósított - IS-FORTH rejtelmeibe kívánja bevezetni az olvasót. Az ismertető nem teljes, mindössze arra vállalkozik, hogy ízelítőt adjon a FORTH rejtelmeiből. A leírás **Adorján Noémi - FORTH lépésről lépésre** című könyve (Műszaki könyvkiadó - 1990) alapján készült, az IS-FORTH „nyelvjárásához” igazítva.

Néha nem is gondolnánk, hogy programgyűjteményünk egyes darabjaiban micsoda hatalmas” erők rejlenek. Közéjük tartozik az IS-FORTH. Ha valaki legális úton jutott a rendszerhez, gyakran még az eredeti dokumentációval a kezében sem megy sokra: nehéz megtenni az első lépéseket. Dokumentáció nélkül pedig végképp kudarcba fulladhatnak a FORTH-szal kapcsolatos kísérletek. Így könnyen hajlunk arra, hogy az egészet a sarokba dobjuk, pedig a FORTH igen komoly nyelv.

A Forth nyelvet az amerikai **Charles Moore** fejlesztette ki a 70-es években. Akkoriban Kaliforniában egy rádió teleszkóp vezérlési és adatfeldolgozási munkáin dolgozott. Egy gyors, kompakt, rugalmas és kibővíthető programozási nyelvre lett volna szüksége. Miután nem talált alkalmas nyelvet, nekilátott egy úgynevezett negyedik (fourth) generációs programozási nyelv elkészítéséhez. Az általa használt gép fájlkezelője csak öt karakteres neveket engedett meg, ezért a „fourth” szócskát az ismert „forth”-ra rövidítette. A cél főképp a programozó eredményességének javítása volt a gép hatékonyságának feláldozása nélkül.

A legtöbb programozási nyelvben komoly megszorítások léteznek. Például ha gyors az assembler, akkor csak egy típusú processzorra használható, nehéz megtanulni és a programozás során ismerni kell az adott gép belső felépítését. Egy másik gépre való áttéréskor újra kell tanulni az adott gépet. A Basic-et könnyű megtanulni, elérhető a gépek nagyrészén, viszont a feladatok többségéhez nagyon lassú és sok dialektusa van. A Forth sokkal gyorsabb a Basic-nél, könnyebben megtanulható. Majdnem minden létező processzoron elérhető és nagy vonzerővel rendelkezik.

Moore eredményeinek híre szájhagyomány útján gyorsan terjedt először a rádió csillagászok körében, majd Amerika nyugati partjának tudományos köreiben. Moore és néhány kollégája megragadva a lehetőséget otthagyták régi munkájukat és megalakították a „Forth Inc.” céget, mely a Forth implementációk kereskedelmi változatainak és szoftver csomagoknak az előállításával foglalkozik. Nem sokkal később felhasználói közösségek alakultak ki az USA-ban és Európában. Ezek a csoportok bővítették a nyelvet, új változatokat hoztak létre. Ezek közül kettő terjed el szélesebb körben:

- **Fig-Forth**
  A Forth Interest Group Inc. egy nemzetközi felhasználókból álló csoport, amely abból a célból alakult, hogy kidolgozzák a saját verziójukat, amit Fig-Forth-nak neveztek el. Ez a társaság a „Forth Installation Manual” című leírást tekinti a nyelv alapjául, amely saját kiadványuk. A Fig-Forth-ban írt programok viszonylag könnyen átírhatók más interpretációknak megfelelő módon. Létezik néhány különbség, amiket itt nem taglalunk és nem is ajánljuk az IS-FORTH nyelvvel újonnan ismerkedőknek, hogy Fig-Forth leírásból tanuljanak.

- **Forth-79 és Forth-83**
  A Europian Forth User’s Group (EFUG) egy szabványt próbált elkészíteni, amelynek a Forth-77 nevet adta. Ahogy a nyelv fejlődött, létrejött egy köztes Forth-78 változat. 1980. első hónapjaiban létrejött a Forth-79. Ez utóbbi változatot fogadták el „hivatalos” Forth-ként. Az újságok és a szoftveripar széles körben támogatták a Forth-79-et. A Forth-83 kiterjesztett és modernizált változata a Forth-79-nek. Ezt a változatot az American National Standards Institute (ANSI) is saját szabványaként fogadta el.

Az IS-FORTH megfelel a Forth-83-nak, csak tartalmaz egy opcionális dupla pontosságú szókészletet és egy assembler szótárat. Korának legygyorsabb FORTH implementációja.
Az Enterprise implementáció több mint száz speciális szót tartalmaz, melyek lehetővé teszik a gép és az operációs rendszer által kínált lehetőségek kihasználását. Minden olyan esetben, ahol olyan szóra volt szükség, mely nem szerepel a szabványban, más Forth implementációkban már létező szavakat vett át az Intelligent Software.

A FORTH-ra legelfogultabb rajongója sem mondhatja, hogy könnyen tanulható. Előnyeit többek között annak köszönheti, hogy gépközeli nyelv (tehát használatához érteni kell egy kicsit a számítógép „lelkét"), hogy bővíthető, alakítható (tehát tudni kell valamennyire, hogyan működik maga a FORTH), és hogy sok benne az eredeti, szellemes, de nem feltétlenül könnyen érthető elgondolás. A FORTH megismerése tehát nem megy erőfeszítések nélkül. Mégis megéri, hiszen:

- a FORTH olyan eszköz, amellyel rövid idő alatt lehet gyors, kis tárigényű és megbízhatóan működő programot írni, sőt, ez a munka még élvezetes is;
- esetleg érdemes a számítógépről többet tudni, mint amennyi a BASIC, FORTRAN stb. programozáshoz szükséges;
- a FORTH-szal együtt néhány igen szellemes programozási trükköt is megismerünk, ami jól jöhet még;
- a FORTH teljesen más, mint a „megszokott" (magas szintű) programnyelvek (főleg, mint a BASIC). Edzhetjük vele egy fontos képességünket: hogy el tudunk fogadni a megszokottól eltérő megközelítést is. (A számítógéppel foglalkozóknak erre nagyobb szükségük van, mint másnak.)

Mindenkinek, aki elszánta magát a FORTH tanulására, sok sikert és jó szórakozást!

1. Az első lépések

Indítsuk el a FORTH-ot! Az IS-Forth rendszerbővítő, tehát ha a gép bekapcsolása után nem az indul el, ezt a :FORTH paranccsal tehetjük meg. A FORTH interpreter a forgalmazó cég, és a (BASIC-hez hasonlóan) rendszerben lévő és a programozó számára szabadon maradt memória kiírásával indul. Ezután várja, hogy adjunk neki egy parancssort. Addig vár, amíg a neki szóló sort be nem fejeztük.
Honnan tudja, mikor fejeztük be? Onnan, hogy lenyomjuk a sor végét jelző ENTER billentyűt. A begépelt sort az ENTER adja át a FORTH-nak; az ENTER lenyomásáig tehát más nem történik, mint az, hogy a FORTH ránk vár. Ezt érdemes megjegyezni, ha nem akarunk sok időt azzal tölteni, hogy az ENTER nélkül adott utasításaink végrehajtására várunk.

Az IS-FORTH-ban a BASIC-ben megszokott EXOS szerkesztő funkciók változatlanul működnek. Kezdjük úgy, hogy mindjárt be is fejezzük: adjunk egy üres parancssort: Nyomjuk meg az ENTER-t! A feleletként kapott OK azt jelenti, hogy a FORTH a mondottakat (esetünkben a nagy semmit) hiánytalanul végrehajtotta. Az OK után a FORTH a következő sor elejére várja újabb kívánságainkat.
Egy egyszerű szó, amit megért:

CR

Az OK ezúttal egy sorral lejjebb jelenik meg, a FORTH kiírt egy kocsivissza és egy soremelés karaktert. A Basic nyelvtől eltérően az IS-FORTH-ban az Interpreter megkülönbözteti a kis és nagybetűket! Minden parancsot nagy betűvel kell írnunk. Ha kis betű is szerepel egy szóban a Forth nem érti meg! A FORTH indításakor a billentyűzet nagybetűs-üzemmódba kerül. A „biztonság kedvéért" minden ENTER leütésekor, - ha esetleg megszűntettük a CAPS üzemmódot - a Forth ismét nagybetűs üzemmódba vált.
Hogy a hatás látványosabb legyen, írjuk most ugyanezt egy sorba többször! Ehhez azt kell tudnunk, hogy:

**a szavakat nagybetűvel kell írnunk,
az egyes szavakat egy soron belül (egy vagy több) szóközzel választjuk el egymástól.**

Tehát, ha azt írjuk:

CR CR CR CR

a FORTH kiírja a négy üres sort és OK-val díjazza a szabályos feladatkiírást. Ha viszont azt írjuk, hogy

CRCR CR CR

akkor a FORTH az érthetetlen CRCR láttára megsértődik, és OK-ra sem méltatva minket, abbahagyja az egészet. A két „jó" CR-t már el sem olvassa. Játsszunk még „kiírósdit":

42 EMIT

Az EMIT szó kiírja azt a karaktert, amelyiknek a kódját megadtuk neki; a 42 a csillag kódja.

Definiáljunk egy csillagíró szót!

```
: CS 42 EMIT ;
```

Itt a kettőspont azt jelenti, hogy új szót definiálunk. A kettőspont is szó!! Így szóköz kell utána.
A kettőspont utáni CS (vagy bármi egyéb) egy név; így fogják hívni az új szót. A név bármilyen karaktert tartalmazhat, kivéve a szóköz, kocsivissza és backspace karaktert (ezeket a szavak elválasztására, a sor végének jelölésére, javításra használjuk).

Ami ezután jön (42 EMIT), az a cselekmény; itt írjuk le, hogyan fog működni az új szó.
A pontosvessző (ami szintén szó, tehát nem szabad „ráírni" az előtte állóra, sem közvetlenül „utána ragasztani" más szavakat) zárja a definíciót.

Ezzel megírtuk első FORTH programunkat. Most már ez is ugyanolyan FORTH szó, mint akármelyik másik, tehát nevének leírásával futtatható:

```
CS
```

Sőt, új szavak alkotására is felhasználhatjuk:

```
: PONT CR CS ;
: VONAL CR CS CS CS
```

Az új szavakkal pedig tovább építkezhetünk:

```
: F CR VONAL PONT VONAL PONT PONT PONT CR ;
```

Az új szavakat érdemes azonnal ki is próbálni. Így lehet (és ajánlatos) „biztosra menni". A FORTH egyik legvonzóbb tulajdonsága éppen ez: az építőkövek, amelyekből a program végül összeáll, megírásuk után azonnal, külön-külön is kipróbálhatók.

A FORTH alapszavak nagy része ugyanígy FORTH-ban íródott, más alapszavak felhasználásával. Például a

```
SPACE
```

szó, amely egy szóközt ír a képernyőre, így épül fel:

```
: SPACE 32 EMIT ;
```

A már ismert CR pedig így:

```
: CR 13 EMIT 10 EMIT ;
```

Ha a képernyőt már teljesen „összefirkáltuk", a

```
CLS
```

paranccsal törölhetjük le.

1.1. A szótárról

Mitől lett a CS, az F stb. végrehajtható szó? Mi történik, amikor ilyen „kettőspontos szódefiníciót" írunk?

A FORTH a számára értelmezhető szavakat egy szótárban tartja. Betöltés után a szótárban a FORTH alapszavak vannak. Új szavak létrehozásával a szótárat - vagy, ha úgy tetszik, magát a FORTH nyelvet - bővítjük. A szótári szavak neveit a

```
VLIST
```

(Vocabulary List; a vocabulary, ejtsd: vokébjulöri szó, jelentése: szótár) szóval írathatjuk ki a képernyőre. A VLIST hatására meginduló „szóáradat" a STOP billentyű leütésével megállítható. Ha saját szavakat definiálunk és utána VLIST-tel szemügyre vesszük a szótárunkat, látjuk, hogy a legutoljára definiált szavak jelennek meg legelőször; előbbi működésünk után például a szótárlista valahogy így kezdődik:

```
F VONAL PONT CS
```

A FORTH interpreter, mikor egy szót értelmezni akar, először is elkezdi azt a szótárban keresni. Mégpedig az útoljára definiált szónál; ebben talál adatot arról, hogy hol kezdődik az utolsó előttinek írt szó, így ha kell, ott folytatja a keresést, és így tovább.
Ha tehát példánkban az F definíciója után írunk egy újabb F szót:

```
: F 70 EMIT ;
```

akkor az F szót „átdefiniáltuk"; a FORTH figyelmeztető hibajelzést ad és beírja az új szót a szótárba.

Álljunk meg egy pillanatra!
Jó, jó, hogy a CS, F stb. attól végrehajtható, hogy benne van a szótárban. Beletettük, mikor definiáltuk őket. Az is igaz lehet, hogy az EMIT benne van, hiszen alapszó. De mitől van benne a 42 meg a 70? Csak nincs benne az összes szám? Ha pedig valami nincs a szótárban, akkor miért nem szól miatta az interpreter, miért tesz úgy, mintha minden a legnagyobb rendben volna?

Elvből. Az elv az, hogy ami nem szótári szó, az biztosan szám, tehát a FORTH interpreter a szótárban való sikertelen keresés után megpróbálja számnak értelmezni a kapott karaktersorozatot. Ha nem megy („számszerűtlen" karakterek vannak benne), akkor az tényleg hiba. Ha viszont igen, számról van szó, akkor ez a szám a verembe kerül.

Mi legyen, ha ráuntunk a definícióinkra, nem akarjuk őket tovább használni? Például átdefiniáltunk egy szót, de megbántuk. A megoldás a FORGET (felejts) szó. A FORGET után (még ugyanabban a sorban) kell megadni az elfelejtendő szó nevét. Pl. ha a második F szavunkat vissza szeretnénk vonni:

FORGET F

A FORGET elfelejti a megadott szót, ezenkívül az utána definiált (tehát a szótárban „fölötte levő") szavakat. Micsoda??? Mindent, amit utána definiáltunk? Ez így van. Elvileg ugyanis bármelyik szóban, amelyet az elfelejtendő után írtunk, használhattuk ezt az éppen törölni kívánt szót! A FORTH szótár szavai egymásra épül(het)nek, nem lehet belőle csak úgy, „középről" törölni. (Meg lehet viszont őrizni szavaink forrásszövegét, hogy hogyan, arról lesz még szó.) Most csak meg szeretnék nyugtatni mindenkit: nem kell majd egy hiba miatt mindig mindent újra begépelni!)

Melyik F szótól kezdve fog a FORGET felejteni, ha kettő is van? Szinte látatlanban meg lehet mondani: a „felsőtől", az utoljára definiálttól. A szavak keresése a szótárban, bármi célból történjen is, mindig felülről halad, ilyen irányban lehet a szótárat gyorsan végignézni. Ezzel példánkban kiássuk a régi, a csillagos F szót, és újra ez lesz az érvényes.

1.2. Mi a verem?

A verem (angol neve stack, ejtsd: sztek) igen fontos része a FORTH-nak. Ebben „levelezneknek" egymással az egyes szavak. Például az EMIT a veremben keresi annak a karakternek a kódját, amelyet ki kell írnia a képernyőre; miután kiírta, le is pusztítja a veremről.
Azért hívják veremnek, mert több dolgot (esetünkben több számot) lehet benne tartani; ezek közül mindig csak egyhez férünk hozzá: ahhoz, amelyik legutoljára került oda, vagyis „legfelül van". Hogy ezt kitapasztalhassuk, egy új FORTH alapszót tanulunk.

- Neve: . (azaz egyetlen pont karakter).
- Működése: kiírja a képernyőre a verem tetején levő számot és egy szóközt tesz utána.
- Hatása a veremre: a kiírt számot törli a verem tetejéről.

Próbáljuk ki!

```
65
.
```

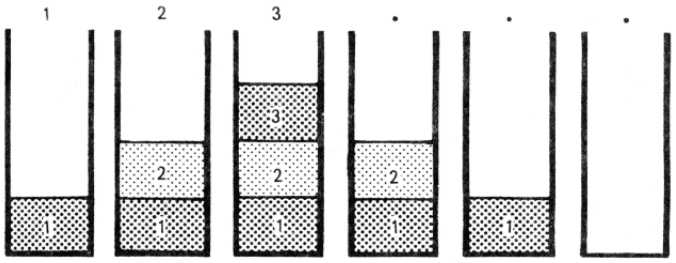
A veremre tettük a 65-öt (első sor). -tal „rákérdeztünk" (második sor). Vissza is írta! Egyúttal törölte is. Győződjünk meg erről. Írjunk még egy pontot. Hibajelzést kapunk, amely azt jelenti, hogy több elemet akartunk a veremből elhasználni, mint amennyi volt benne.
És ha nem? Könnyen előfordulhat, hogy a Kísérletező Olvasó már egy csomó mindent művelt, mire ide eljut. Esetleg már volt a veremben valami. A verem kiürítésének legegyszerűbb módja: begépelünk egy szót, amelyről tudjuk, hogy a FORTH nem ismeri. A „feldühödött" interpreter kiüríti a vermet; ha ezután próbálja ki valaki a fentieket, meglátja, hogy így igaz. A módszer hasznos lehet, mikor véletlenül rakjuk tele a vermet „szeméttel". (Mondjuk ciklusban felejtünk el valami fölöslegeset törölni.)
Próbáljuk ki ugyanezt több számmal:

```
1 2 3 . . .
```

Melyik számot fogja először kiírni? Azt, amelyik a verem tetején van, tehát amelyik utoljára került a verembe. Ezt egyúttal törli is; a következő tehát az alatta levő elemet írja ki és törli. A kapott válasz:

```
3 2 1
```

Az egyes lépések után a verem a következő ábra szerint néz ki.



A legtöbb szó a veremben legfelül található számmal / számokkal dolgozik. Ezeket a számokat eközben gyakran kiveszik onnan, holott a későbbiekben még szükségünk lenne rájuk. Ezért szükség lehet a veremben látható legfelső szám „lemásolására", így a műveletet a másolaton hajtjuk végre, az eredeti a veremben marad. Ez a szó a:

```
DUP ( n - - - - n n )
```

A nyelv hagyományai szerint az utasítások működését (hatását az adatveremre) megjegyzésként (azaz zárójelek között), az alábbi formában szokás dokumentálni: (bemenő adatok - - -kimenő adatok)

Egyszerre akár két elemet (vagy egy dupla pontosságú elemet) is megduplázhatunk:

```
2DUP (n m - - - - n m n m )
```

A verem teljes tartalmát egyetlen lépésben törölhetjük az

```
SP! (n...m - - - - )
```

szó használatával.
Igen hasznos szó, a

```
.S ( - - - - )
```

mely kiírja a veremben lévő számokat, de nem távolítja el azokat. A kiírást a legalsó elemmel kezdi, és sorban kiírja valamennyi a veremben lévő számot. E parancs segítségével bármikor ellenőrizhetjük a verem tartalmát, anélkül, hogy bármit módosítanánk benne. Nagy segítség a tanuláshoz!

Folytatjuk!

ENTERPRISE COMPUTERS

2020. május 16.

13-19 óráig

Zuglói Civil Ház
1144 Budapest, Csertő park 12.

Információ:
www.enterpriseklub.hu

ENTERPRISE KLUB

Telefon: 30-322-3017 • e-mail: info@enterpriseklub.hu

2019 évi játékok

Írta: Kiss László
(Lacika)

Elérkezett az év vége - és mivel idén is jónéhány látványos játékkal gazdagodtunk - ideje összefoglalni az ide játéktermést! (Vállalva a kockázatot, hogy a Karácsonyra elkészülő meglepetések kimaradnak a felsorolásból...). Az első két cím rögtön eredeti Enterprise-fejlesztés:

Az év első játéka, a **Treasure Cave** nem teljesen „új": a Treasure Cave 4K bővített változata. Ez egy „egyszerű" gyűjtögetős platform játék. A pályán fellelhető valamennyi érmét össze kell gyűjtenünk, miközben

véletlenszerűen mozgó ellenfelek kergetnek. Az ellenfelek számától és kezdő szinttől függően kapunk sérthetetlenséget, ami tűzre aktiválható, és 5 másodpercig tart. A program Geco, Endi és Szipucsu munkája.

Kotasoft a **Banana** második részével lepett meg minket. A többképernyős (25 terem van a programban), gyűjtögetős játékban főhőseinknek banánokat kell összegyűjteni. Ténykedésünket zárt ajtók, és ellenfelek nehezítik. A játékstílustól meglepő módon kétszemélyes játékra is van lehetőség, amely egyedi játékélményt nyújt.

Geco megkapta a spanyol 4MHz programozó csapat valamennyi CPC-s játékának forráskódját, Így sorban elkészültek azok konverziói. Az Adios A La Casta első és második része szintén egy gyűjtögetős platformjáték (ahogy a csapat összes játéka is az). A spanyol nyelvű történettől most diszkréten eltekintek, a játék lényege: összegyűjteni mindent... Az első rész 4 színes-, a második 16 színes üzemmódban készült remek lehetőséget adva a kétféle látványvilág összehasonlítására.

Az **Operation Alexandra** erőssége viszont pont az ötletes történet! 1942-ben, egy évvel azután,

hogy Németország megtámadta a Szovjetuniót, az Észak-sarktól 1000 kilométerre fekvő Alexandra Land nevű szigeten - szó szerint a semmi közepén - a németek egy titkos időjárási kutatóállomást létesítettek. Az állomás működése a Második Világháború idején igen fontos volt, mivel az általa készített méréstani jelentések nélkülözhetetlenek voltak a csapatok, tengeralattjárók és hajók mozgásának tervezéséhez. Az állomást 1944-ben kiürítették, miután a személyzet a nem megfelelően átsütött jegesmedve hús fogyasztása következményeként Trichinózissal (a trichinák által okozott betegség) fertőződött. Az állomás létezését fennmaradt hiteles bizonyíték hiányában sokáig mítosznak gondolták (melyet egy 1954-ben megjelent könyv táplált), pontos elhelyezkedése a feledés homályába veszett. Az állomást 2016-ban(!) találta meg egy orosz kutatócsoport, ahol számos, jó állapotban lévő leletet találtak. A valóságot színezte ki a 4MHz programozó csapata a játékuk történetében: 1976-ban járunk, amikor a sarkvidéken állomásozó katonai erők ismeretlen jeleket fognak az említett szigetről. Ki is küldenek egyetlen felderítőt Mihail Mashkov-ot, hogy nézzen utána a forrásnak. A megadott pozícióban Mihail egy látszólag elhagyatott bázist talál, ám egy váratlan hóvihar megnehezíti az állomáshelyre való visszatérést, ezért Mihail menedéket keres a földalatti komplexumban. Hamarosan rádöbben, hogy nincs egyedül... Egy idegen - cseppet sem barátságos civilizáció - rendezkedett be. Főhősünk elindul, hogy egyes-egyedül rendet - és újra kihalt állomást - teremtsen...

A spanyol csapat legkiforrottabb, legélvezetesebb játéka a **El Tesoro Perdido De Cuauhtemoc.**

A történet szerint Cuauhtémoc azték uralkodó legendás kincsét keressük. A játék főhőse és kivitelezése a Rick Dangerous-re emlékeztet, a játékmenet viszont valamivel összetettebb. A küldetést - némileg meggondolatlanul - minden felszerelés nélkül kezdjük. Márpedig célunk eléréséhez mindenképpen szükségünk lesz több rúd dinamitra, és egy revolver is nagymértékben javítja túlélési esélyeinket (amihez persze lőszert is kell szereznünk). Felszerelésünket ezen kívül búvárfelszereléssel is kiegészíthetjük, amivel a vízzel elárasztott járatokat kereshetjük fel. A porral oltóval pedig az utunkat álló lángnyelvet olthatjuk el. Az eszközhasználat automatikus, ahol erre szükség van. A porral oltó ikonja villog is abban a teremben, ahol használatos. A játékmenet nem teljesen lineáris, így főhősünknek az eszét is kell használni: a kapcsolókat megnyomva megnyithatunk addig lezárt átjárókat („valahol"), így a helyszíneket többször oda-vissza be kell járnunk. A játék azért segít, általában átjárók közelében (legalábbis légvonalban...) található a hozzá tartozó kapcsoló. A narancssárga kődarabokat dinamittal takaríthatjuk el az utunkból (már ha szerezünk dinamitot, és még nem puffogtattuk el az összeset). Végcélunk elérése érdekében tett erőfeszítéseink közben a zöld és piros drágakövek megszerzésével gyarapíthatjuk vagyonunkat és pontszámunkat. A legleleményesebbek titkos kincseskamrát is találhatnak. Az ajtók nem a háttér részét alkotják: rajtuk keresztül tudunk másik terembe belépni (‚fel' billentyű). Az egyik ajtó egyirányú. Az utolsó - utunk végcéljához vehető - ajtó zárva van, csak négy aranyrúd megszerzésével tudjuk kinyitni. Az aranyrudak megszerzéséért energiát / plusz életet is kapunk. A kincseskamra bejáratát egy nagy, tűzlabdákat köpködő fej őrzi. Ennek elpusztításához 5 rúd dinamitot kell tartalékolnunk (közvetlenül a szobor elé kell rakni, egyesével). Az F jelzésű ajtó egyébként szintén egyirányú, mögötte találunk elegendő dinamitot. Találhatunk még itt egy doboz energiaitalt, ami pótolja esetleg megcsappant energiánkat. Ellenfeleink az ilyen játékoknál „szokásos" vadállatok (piranhák, óriás darazsak, sündisznók, denevérek, pókok, rákok, kígyók) és zöld csontvázak. Ezeket egy lövéssel lelőhetjük, ha nagyon útban vannak, de amikor legközelebb visszajövünk újra ott mászkálnak. A dinamit elvileg szintén használható az ellenfelek ellen, de ebből mégannyi sincs, mint lőszerből, be kell osztani! Természetesen a csapdák sem hiányozhatnak: karók, falból kilövellő dárdák, tűzgolyók, alattunk szétmáló platók.

Az **R-Type** az egyik legismertebb, legszebb, legnehezebb, stb., shoot-em-up arcade játék. A 8 bites konverziók többsége is a lehetőségekhez képest nagyon szépen sikerült. A kiadó természetesen CPC verziót is szeretett volna (hisz a C64 és Spectrum mellett ez volt Nyugat-Európában még elterjedtebb 8-bites gép). Az idő viszont szoros volt, a fejlesztők mindössze egy hónap határidőt kaptak a CPC verzióra. A fejlesztők

ezért a leggyorsabb és esetünkben a legkiábrándítóbb megoldást választották: a Spectrum-verzió portolását. A CPC azonban nem tud attribútum képernyőt, így azonos felbontás (és azonos képernyőméret) mellett, 4 színes üzemmódban fut a hivatalos konverzió. Az eredmény - egy eredetileg színpompás játék esetében - természetesen csapnivaló lett, amely sem a sajtónak, sem a játékosoknak nem tetszett. Valószínűleg ez az elégedetlenség inspirálta az Easter Egg csapatát, amikor elkészítették a Spectrum verzió alapján újraírt változatukat. Ezt konvertálta Geco, így az Ep tulajdonosok összehasonlíthatják a Spectrum-féle attribútum verziót, a 16 színes üzemmódú CPC-s változattal.

A Martech - a maga korában - professzionálisnak számító **Nigel Mansell's Grand Prix** c. játéka az Enterprise tulajdonosoknak is ismerős lehet. Kétféle Spectrum-konverzió is közkézen forgott anno, de mindkettő hibás volt (csak az első pályán lehetett játszani). Geco ennek javítgatása helyett inkább a sokkal látványosabb (színes) CPC verziót írta át, néhány extrával „felturbózva". A program kb. 60%-kal gyorsabb lett, mint a CPC verzió: Meg lett szüntetve a másolgatás, helyette az ég animációja LPT címcserével mozog, a versenytér pedig két területet használ, míg az egyiket rajzolja a program, a másikat jeleníti meg a CPC verzió a memória egyik területére rajzolta, és ha befejezte, akkor bemásolta a videómemóriába. A gyorsulás miatt egy picit nehezebb lett a játék, ezért, és mert turbós gépen játszhatatlan lenne a sebesség, F1-F6 billentyűkkel állítható, F1: nincs lassítás, F6: 5 frame lassítás. A játékban az 1987-es Forma-1 es világbajnokságban vehetünk részt, amikor ugyan nem Nigel Mansell lett a világbajnok (a pontszámtáblán második lett Nelson Piquet mögött), de a Spectrum szülőhazája Anglia...

Az **International Karate** multiplatformos fejlesztés volt. A C64, a viszonylag jól sikerült Spectrum verzió mellett létezik többek között hivatalos C16-os (ebben azért sok köszönet nincs...) és CPC verzió is. Ez utóbbi elég csúnya: 4 színes üzemmódban fut, így Geco nem ezt konvertálta: a C64-es grafikát (némi raszter-színezéssel feltupírozva) felhasználva írta újra a programot. A kezelés és a kunsztok természetesen változatlanok, de a látvány szebb. A játékban 8 helyszín van: ezek C64-en két külön programban voltak, az Enterprise verzióban az összes egyszerre a memóriában fért (a 8 kép tömörítve 18 Kbyte). Emellett az eredeti 4 bites digi hangok is elférték a memóriában. A program EXOS kompatibilis, reset után elmenti a pontszámtáblát, és be is tölti kivéve magnós konfigurációban. Turbós gépen a digi effekt és a zene a gép sebességéhez állító-

dik, a digi SID lejátszás sebessége változik: 4MHz-en 9 KHz, 5MHz-en 10 Khz, 6MHz-en 12 KHz, 8MHz, és fölötte 16KHz.

A **Mag the Magician** egy 2017-es (első pillantásra RPG-nek tűnő) gyűjtögetős játék. Geco egy konverzióba gyúrta az MSX és a ZX Uno verziót, betöltés után választhatunk, melyiket akarjuk „látni". A nyugdíjas varázslóknak otthont adó kis falucskában,

Telerinben a mágia és varázslatok kutatásával, csendben múlatják mindennapjaikat az agg varázslók. Egy napon azonban nagy izgalom támad a faluban: a Nagy Varázskönyv öt lapja - melyek oly varázslatokat, titkokat tartalmaznak, melyek rossz kezekbe kerülve veszélyesek is lehetnek - eltűnt! Mag a lelkes tanonc vállalja a feladatot, hogy átkutatja a környéket és megkeresi az eltűnt könyvlapokat. (Igencsak mágikus lapokról van szó, piros alapon fehér firkálmányokat tartalmaznak...) A küldetés később veszélyesebb lesz, mint amilyennek kezdetben ígérkezik: igaz ugyan, hogy két lap valamely szórakozott varázslónál kallódik egy pedig elveszett, de kettőt az orkok loptak el!

dBase II. 2.43 (IS-DOS) – V. rész

Több rekord „egyidejű” javítását teszi lehetővé a BROWSE parancs. A rekordok táblázatos formában jelennek meg, az aktuális rekordtól kezdődően. Tetszőlegesen módosíthatjuk a mezők tartalmát, de nincs lehetőségünk új rekordok hozzáfűzésére.

5.6.2. Rekordok törlése az adatbázisból

Egy rekord törlése nem csupán azt jelenti, hogy tartalmát töröljük hanem azt is, hogy maga a rekord megszűnik, eggyel kevesebb rekordja lesz az adatbázis-állománynak. A rekordok emelkedő sorszámozása helyreáll, minden rekord, amely a törölt után helyezkedett el, eggyel „előbbre lép” (eggyel csökken a sorszáma). Rekordok törlése a DELETE utasítás szolgál, melynek a következő az alakja:

DELETE [<érvényességi kör>] [WHILE <feltétel>] [FOR <feltétel>]

Bizonyos körbe tartozó (valamilyen feltételt kielégítő) rekordok törölhetők, de csak két lépésben. Először meg kell jelölnünk azokat a rekordokat, amelyeket törölni szándékozunk („törlésre jelölés”) majd egy külön paranccsal lehet elvégezni a megjelöltek tényleges törlését. A parancs törlésre jelöli az érvényességi körbe tartozó, az esetleges feltételeket kielégítő rekordokat. (Az érvényességi kör alapértelmezés szerint az aktuális rekord.) A parancs végrehajtása után a program közli velünk, hány rekordot jelöltünk törlésre. A törlésre jelölt állapotot a LIST és DISPLAY parancs eredményében (a kiírt listában) a rekordsorszám és a rekord tényleges, tartalma között megjelenő csillag („*”) jelzi. Példaképpen jelöljük törlésre a „KARTON” állományból a száznál kisebb készlettel rendelkező árucikkeket, és nézzük meg a rekordok listáját:

```
                    IS-DOS

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

***  dBASE II     Ver 2.3B   22 FEB 82
. use karton
. delete for keszlet<100
00005 DELETION(S)
. list
00001  001101 golyostoll        23.50    125      5 12.34
00002 *000213 kek tinta         34.60     50      5 12.33
00003  012677 cimke              0.90   1340    100 11.23
00004 *001321 piros tinta       43.50     76      5 11.23
00005 *045677 toltotoll        231.00     34      1 11.22
00006 *026267 tuzogep          489.00     12      1 12.33
00007  000333 tuzokapocs         1.50    500     50 11.24
00008  002631 miltonkapocs       1.50    500     50 11.24
00009 *000012 ragaszto          12.70     77     10 13.45
00010  000001 geppapir           0.50   2550    100 12.45
.
```

Egy környezeti paraméter segítségével a törlésre jelölt rekordok elrejthetők a további parancsok elől, mintha már megtörtént volna a tényleges törlés. Ha a

SET DELETED ON

parancsot adjuk ki, a törlésre jelölt rekordokat nem veszik figyelembe az parancsok, amelyek érvényességi köre több rekordot ölel fel (ALL, illetve FOR vagy WHILE paraméter szerepel bennük), nem jelennek meg a képernyőn. (Azok a parancsok, amelyeknek ilyen paramétere nem lehet, minden rekordot figyelembe vesznek az előírt tevékenység elvégzésekor, tartalmuktól és tulajdonságaiktól függetlenül.) A törlésre jelölt rekordok elrejtése után ezek nem jelennek meg például az adatbázisról készíttetett listában, azonban a rekordsorszámok nem folyamatosan követik egymást: a törlésre jelölt rekordjaink fizikailag még léteznek. A törlésre jelölt adatokhoz ekkor csak konkrét formában lehet hozzáférni (DISPAY RECORD x).

A rekordok törlésre jelölt tulajdonságát programból is tudjuk ellenőrizni a „*” függvény segítségével. Ez logikai értéket ad vissza, és akkor „igaz”, ha az aktuális rekord törlésre van jelölve. E függvény felhasználásával készíthetünk az adatbázisról olyan listát is, amelyben csak a törlésre jelölt rekordok szerepelnek:

```
SET DELETED OFF
LIST FOR *
```

A * függvény visszatérési értékét nem kell semmivel összehasonlítani, az már önmagában igaz vagy hamis értékű.
A dBASE a rekordok törlésre jelöltségét az adatbázisban tára (azaz a lemezes állományba is felírja), mégpedig abban a „felesleges” byte-ban, amellyel hosszabb egy rekord, mint a mezők hosszának összege.

A törlésre jelöltséget meg lehet szüntetni (a rekordokat „vissza lehet hívni”) a

RECALL [<érvényességi kör>] [WHILE <feltétel>] [FOR <feltétel>]

paranccsal. Az érvényességi kör alapértelmezése itt is az aktuális rekord. Az „elrejtett” rekordokhoz ez a parancs is csak akkor fér hozzá, ha konkrétan hivatkozunk rájuk. Végrehajtáskor a DELETE parancshoz hasonlóan közli, hány rekordot „hívtunk vissza”.

A törlésre jelölt rekordok végleges megszűntetése (tényleges törlése) a

PACK

paranccsal történik. Ez a kiadás pillanatában törlésre jelölt összes rekordot törli, és helyreállítja az egyesével növekvő sorszámozást. Az így megszüntetett rekordok semmilyen módon nem nyerhetők vissza.

6. Adatbázisok rendezése

Adataink rendezése több szempontból szükséges lehet. Például az adatbázisról készített listákat mi magunk is szeretjük rendezve látni, ezekben hamarabb megtaláljuk a keresett rekordot. Ez utóbbi szempontból a dBASE is jobban kedveli a rendezett állományokat. Néhány parancs működése közben feltételezi, hogy a rekordok rendezve vannak mások nem is működnek, ha az adatbázis rekordjai nem rendezettek. A dBASE II kétféle módon tud rendezni. Az egyik fizikailag átrendezi a rekordokat, a másik csak a logikai sorrendjüket tartja nyilván egy segédállományban, ezt nevezzük indexállománynak.

6.1. Adatrekordok fizikai rendezése

A SORT parancs egy másolatot készít az aktív adatbázisról, a rekordok a logikai rendnek megfelelő sorrendben kerülnek az új állományba. Az így készült állomány tehát fizikailag rendezett (a fizikai rekordsorrend megegyezik a logikai renddel).
A parancs alakja:

SORT ON <mezőnév> TO <új állománynév> [DESCENDING]

Az új állomány közönséges adatbázis-állomány, típusjele alapértelmezés szerint „.DBF" lesz. A rendezés után az eredeti állomány marad nyitva, ha az újat akarjuk a továbbiakban használni, azt előbb meg kell nyitni. A rendezés alapértelmezés szerint növekvő sorrend szerint történik. Numerikus mezők esetében ez a rendezés egyértelmű. Karakteres mezőn az ASCII kódtáblázat alapján végzi a rendezést, szerencsére ebben ABC rendben vannak a betűk, de minden nagybetű megelőzi a kisbetűket. (De senki ne várja, hogy a magyar ékezetes betűket tartalmazó szavak a megfelelő helyre kerüljenek...) Ebben a rendben például a „anita" a „Zita" mögé kerül. Ez persze csak az angol abc betűire vonatkozik, a magyar karakterkészlettel ez a program még nem tud tökéletesen hozni). A DESCENDIG opció használatával kérhetünk csökkenő sorrendbe történő rendezést is. A rendezett állomány a lemezen jön létre, használatához először meg kelt nyitni. Ettől kezdve ugyanúgy kezelhető, mint bármely más adatbázis-állomány, ezért amikor módosítjuk a rekordok tartalmát, a rendezettség megszűnhet (ha új rekordot fűzünk hozzá, akkor nagy valószínűséggel meg is szűnik), és újra kell rendezni.

6.2. Adatbázis logikai rendezése

A logikai rendezés - indexelés - egy segédállományt hoz létre a rekordok sorrendjének (és a rendezési kulcs) tárolására. Ilyenkor a rekordok fizikai sorrendje nem változik, és az indexállomány - ha meg van nyitva - minden módosításkor aktualizálódik. Ez az utóbbi tulajdonsága is indokolja, hogy sokkal gyakrabban használjuk, mint a SORT parancsot. A dBASE is ezt a módszert támogatja inkább, az indexelt állományokban ugyanis néhány fontos műveletet gyorsabban tud elvégezni.
Az indexállományok elkészítésére egyetlen parancsot használhatunk:

INDEX ON <kulcskifejezés> TO <indexállomány neve>

Indexelni csak növekvő sorrendben lehet. Vigyázni kell

azonban arra, hogy a karakteres kulcskifejezések értéke nem lehet hosszabb 99 karakternél (más típusnál ezt a korlátot nem lehet megközelíteni sem). Logikai típusú mező itt sem szerepelhet. Az indexelési parancsban érvényességi kör nem adható meg, az egész állományt indexeli (még a törlésre jelölt, elrejtett rekordok is). Az indexállomány típusjele „.NDX". Az indexállomány létrejötte után azonnal aktivizálódik, és ettől kezdve minden parancs az ebben előírt sorrendben „látja" az adatbázist. Ezt nevezzük logikai sorrendnek, a rekordok a parancsok által meghatározott műveletekben mindig ebben a sorrendben vesznek részt. Az INDEX parancs segítségével egy adatbázishoz tetszőleges számú indexállomány készíthető. Ez a parancs végrehajtása előtt lezárja az aktuális adatbázis jelenleg nyitott indexállományait - ha vannak ilyenek -, csak az éppen elkészült marad aktív állapotban.

A szükséges indexelések elvégzése (az indexállományok után az adott adatbázishoz egyszerre több, de legfeljebb 7 indexállományt nyithatunk meg a

SET INDEX TO <indexállomány-lista>

paranccsal. Rögtön az adatbázis megnyitásakor is aktivizálhatók az állományok a

USE <adatbázisnév> INDEX <indexállomány-lista>

parancs formával. A felsorolásban elsőként megadott indexállomány lesz a főindex (master index), amely meghatározza a logikai sorrendet, minden további megnyitott indexállomány is aktualizálódik az adatbázisban végzett módosítások nyomán.

Ha egy indexállomány nincs nyitva, amikor módosítani kellene (például új rekordok hozzáfűzésekor), akkor megnyitása után a módosításokról nem tud, rossz sorrendben listáz az új rekordokat egyáltalán nem veszi figyelembe. (Ezért javasolt az adatbázishoz tartozó összes indexállományt is megnyitni, akkor is, ha nincs rájuk szükségünk.) Ilyenkor nekünk kell külön gondoskodnunk ennek kijavításáról. Az indexállományok aktualizálását, „újjáépítését" a

REINDEX

Paranccsal is kiválthatjuk, ez az aktív adatbázishoz megnyitott összes indexállományt újra elkészíti.

Az indexállományok csak ahhoz az adatbázishoz nyithatók meg, amelyhez készítettük (ezért lényeges megjegyezni, hogy egy-egy indexállomány hova tartozik). Az adatbázis lezárásakor az összes indexállománya is lezáródik. A munkaterületek tartalmáról kapott információk között (DISPLAY STATUS parancs) azt is elolvashatjuk, hogy egy-egy adatbázishoz mely indexállományok vannak nyitva, és ezek közül melyik a főindex.

6.3. A rekordmutató mozgatása indexelt adatbázisban
Az indexelt állományok listázásakor a rekordsorszámok már nem folyamatosan emelkedő rendben követik egymást, hanem meglehetős összevisszaságban, ahogy azt a logikai sorrend előírja. Lényeges, hogy ebben a fizikai sorrendtől független logikai rendben is tudjunk mozogni (például most a GOTO 1 paranccsal egyáltalán nem biztos, hogy az első rekordra kerülünk).
Az adatbázis két - logikai - végére a következő parancsok viszik a rekordmutatót:

GO TOP - a logikailag első rekordra áll
GO BOTTOM - a logikailag utolsó rekordra áll

Szükség van arra is, hogy a rekordsorszámok ismerete nélkül logikai rendben „a következő" rekordra tudjunk menni. Ezt teszi az alábbi parancs:

SKIP

E parancs ilyen formában eggyel „lejjebb" lépteti a rekordmutatót a sorbann. Megadhatunk azonban egy számot is, hogy annyival menjünk előbbre, esetleg (negatív szám esetén) visszafelé:

SKIP <n>
elmondottakból következik, hogy a következő két parancs teljesen egyenértékű:

SKIP
SKIP 1

Valamink a következő parancsok is helyesek: SKIP 5, SKIP -2
A SKIP parancs - éppen azért, mert nem lehet tudni melyik rekordra helyezi a rekordmutatót, végrehajtás után üzenetet küld. A „Record 0009" üzenet például arról tájékoztat bennünket, hogy a 9. sorszámú rekord lett az aktuális. Az üzenet megjelenése a SET TALK paranccsal tiltható le. Ha az utasítással „kilépnénk" az adatbázisból, a parancs az első vagy utolsó rekordra ugrik.

Esetenként fontos lehet tudni, hogy az aktuális rekord után van e további rekord, vagy esetleg az adatbázis végén állunk. Ezt egy tesztelő függvény ellenőrzi:

EOF - igaz értéket szolgáltat, ha az állomány végén állunk (end of file)

A függvénynek nincs argumentuma.

7. Keresés az adatbázisban

Keresés alatt azt a folyamatot értjük, amikor a rekordmutatót egy adott tulajdonságú rekordra szeretnénk beállítani, hogy utána valamilyen parancso(ka)t hajthassunk végre a rekordon. Kétféle keresési módszer áll rendelkezésünkre:

Soros (szekvenciális) keresés - a feltétel vizsgálata az első rekordon indul, és a megfelelő rekordon áll meg. Tetszőleges adatbázisra alkalmazható.
Gyorskeresés - kihasználja, hogy az állomány a keresési szempint szerint rendezett. Csak indexelt állomány esetén, a főindex keresésre használható.

7.1 Soros keresés

A soros keresést megvalósító parancs formája:

LOCATE [FOR <feltétel>] [WHILE <feltétel>] [<érvényességi kör>]

Leggyakrabban FOR paraméterrel használjuk (ilyenkor az érvényességi kör az „ALL" alapértelmezést kapja), ha elhagyjuk mindhárom paraméterét, az aktuális rekordot „tatalja meg". Bármilyen - dBASE által értelmezhető - feltétel szerint tud keresni az aktív adatbázisban.
Ezzel a paranccsal lehetőségünk van az összes adott feltételnek eleget tevő rekord megkeresésére. Az utolsó LOCATE parancs által kijelölt keresés folytatható (nincs minden munkaterületnek külön LOCATE lehetősége, ezért a folytatási parancs kiadása előtt mindig gondoskodjunk arról, hogy ugyanazon a munkaterületen legyünk, ahol eredetileg kiadtuk a LOCATE parancsot). A továbbfolytatás a

CONTINUE

Paranccsal (continue = folytat) kérhető.
Ez a parancs-pár (a LOCATE és a CONTINUE) is üzenetet küld a végrehajtás eredményéről („Record = 00002" azt jelenti, hogy ráállt a rekordmutató a második rekordra, amint ezt a példában szereplő DISPLAY parancs is bizonyítja).
Ha az érvényességi körben nincs több ilyen tulajdonságú rekord, ezt „END OF FILE ENCOUNTERED" üzenettel jelzi. „ALL" érvényességi kör esetén a rekordmutató ilyenkor az állomány végén van. A LOCATE parancs rendezetlen és rendezett adatbázisban egyaránt tud keresni, indexelt adatbázisban a logikai sorrendben veszi sorra a rekordokat (ezért előfordulhat, hogy a tizedik rekordot hamarabb találja meg, mint az ötödiket).

```
                    IS-DOS

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

***  dBASE II     Ver 2.3B   22 FEB 82
. use karton
. locate for keszlet<50
RECORD: 00005
. display
00005  045677 toltotoll       231.00     34      1 11.22
. continue
RECORD: 00006
. display
00006  026267 tuzogep         409.00     12      1 12.33
. continue
END OF FILE ENCOUNTERED
.
```

7.2.Gyorskeresés

A gyorskeresés működési elve miatt lényegesen gyorsabb mint a soros keresés (főleg nagyobb adatbázisoknál), de csak indexelt bázisban, a főindex szerinti keresésre alkalmazhatjuk. Nem adható meg paraméterként érvényességi kör, Mindenképpen az egész állományban keresünk. Ezt a keresési formát nem lehet továbbfolytatni, a feltételt kielégítő rekordok közül mindig csak az elsőt találhatjuk meg. Megjegyzendő azonban, hogy amennyiben vannak megfelelő rekordok, akkor azok a megtalált rekordot követik a logikai rendben. A parancs formája:

FIND <karaktersorozat> vagy <kifejezés>

Karakteres vagy numerikus indexkulcs mellett használható. A parancsszó után felírt karaktereket vagy számjegyeket hasonlítja össze indexkulccsal. Például - „CIKKNEV" szerint indexelt „KARTON" esetén - keressük „toltotoll" rekordját:

FIND toltotoll

A FIND előnye, hogy a karaktereket nem kell határolójelek közé tenni, hátránya, hogy ettől beszűkül a felhasználási lehetőségek köre. Például memóriaváltozók nem írhatók a parancsba csak úgy, önmagukban. Ha szóközzel kezdődő karakterláncot akarunk keresni, idézőjelek közé kell rakni a karaktersorozatot.
A parancs nem küld üzenetet, ha talált megfelelő rekordot, csak amikor nem talál, akkor üzen: „NO FIND" Mivel a rekordkeresésnek elsősorban programok készítésekor van jelentősége, szükségünk lehet annak eldöntésére, hogy találtunk-e megfelelő rekordot. Itt is használhatjuk az EOF függvényt, mert „nemtalálás" esetén biztosan az állomány végére kerül a rekordmutató.

7.2.1 A makróhelyettesítés funkciója
Többek között azért, hogy a FIND paranccsal memóriaváltozó értéke szerint is tudjunk keresni, meg kell ismerkednünk (és barátkoznunk) az makró-helyettesítéses funkcióval. Tegyük fel például, hogy van egy „ARUCIKK" nevű memóriaváltozónk, amiben elhelyeztük a „toltotoll" karaktersorozatot; keressünk most a változó segítségével:

FIND &arucikk

A „&" funkció hatására a dBASE még a parancs végrehajtása előtt a „&arucikk" helyére beírja a változó tartalmát, és ezek után gyakorlatilag végrehajtja a „FIND toltotoll" parancsot (a & jel és a változó neve között lehet szóköz).

A „&" funkció karakterláncok belsejében, határoló-jelek között is működik:

? „Ezek &arucikk.ak"

A parancs hatására „ezek toltotollak" felirat jelenik meg a képernyőn. A változónév mögötti pont jelzi a „&" funkció részére a memóriaváltozó nevének végét, utána újra konstans karakterek következnek.

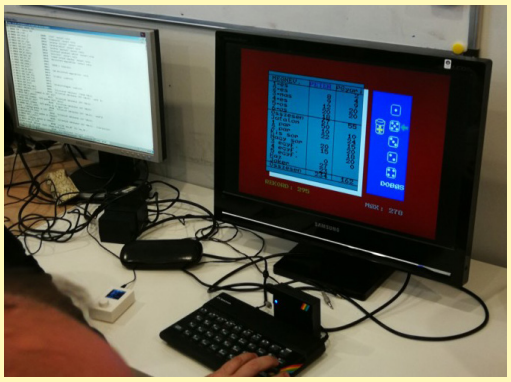
Előnyös tulajdonsága a makró-helyettesítésnek, hogy tetszőleges parancsot is tárolhatunk makrókban, amit bármikor végrehajthatunk, akár parancsmódban, akár program futása közben:

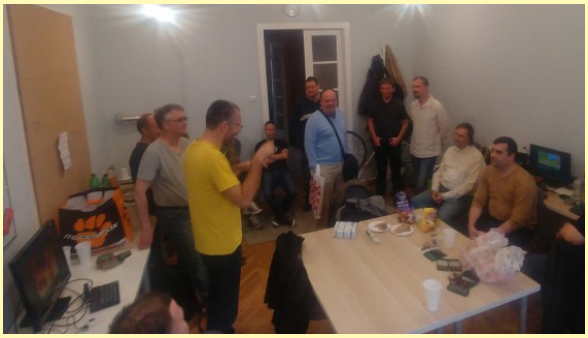
STORE „DELETE RECORD" TO D
&D 5

A fenti két utasítás hatása: DELETE RECORD 5
Az elmondottak miatt, vagy ezek ellenére a „&" funkcióval bánjunk mindig óvatosan, egyrészt mert lelassítja a programvégrehajtást, másrészt a változó tartalmától függően megzavarhatja a tényleges parancsvégrehajtást.

Folytatjuk!

Enterprise + Spectrum Klub 2019. április 13.

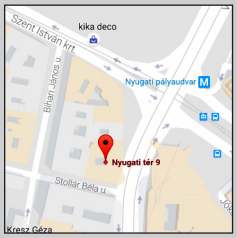






ENTERPRISE KLUB

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 - 2019/5-6. szeptember - 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: http://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
http://enterpress.news.hu/index.php/magazin