FALUDI ANDRÁS

a dBASE és a PROP-BASE adatbázis-kezelő rendszerek kézikönyve

SZÁMALK

Josef Havel
Max - Stromeyer - Str. 9
7750 Konstanz

A dBASE ÉS A PROP–BASE ADATBÁZIS–KEZELŐ RENDSZEREK KÉZIKÖNYVE

dBASE AND PROP–BASE RELATIONAL DATABASE MANAGEMENT SYSTEMS' MANUAL

ANDRÁS FALUDI

dBASE and PROP-BASE Relational Database Management Systems' Manual

COMPUTING APPLICATIONS AND SERVICE COMPANY
BUDAPEST, 1987

A dBASE és a PROP-BASE adatbázis-kezelő rendszerek kézikönyve

FALUDI ANDRÁS

SZÁMÍTÁSTECHNIKA–ALKALMAZÁSI VÁLLALAT
BUDAPEST, 1987

Lektorálta Pogány Csaba
Supervised by Csaba Pogány

ISBN 963 553 119 2

Tartalom

MÁSODIK RÉSZ

Az adatbázis-kezelő alkalmazása

HARMADIK RÉSZ

Az adatbázis-kezelő parancsai

NEGYEDIK RÉSZ

Függelék

Előszó

A dBASE II és a PROP-BASE adatbázis-kezelő rendszer mikroszámítógépre készült. Ezek az adatbázis-kezelők lehetővé teszik az adatok kényelmes kezelését — adatbázisok létrehozása, azok egyszerű módosíthatósága, továbbá nagyfokú programozhatóságuk és fejlett belső automatikájuk révén.

A kézirat elkészítésének idején hazánkban a dBASE II 2.3. számú változata és a PROP-BASE rendszer a legelterjedtebb, így ezek teljes ismertetését tartalmazza e könyv. (Az említetteknél előbbi változatok valamivel kevesebb, a későbbiek valamivel több szolgáltatást nyújtanak.) Mindkét adatbázis-kezelő használata középiskolai végzettség esetén minimális számítástechnikai ismeretekkel is elsajátítható. Természetesen ismerni kell hozzá az adott számítógép kezelését és tulajdonságait.

A kezdők aránylag gyorsan megtanulhatnak értékes és használható programokat készíteni a könyvben közölt ismeretek alapján. Azok, akik már programoztak más nyelveken, könnyedén elsajátíthatják az adatbázis-kezelő rendszer használatát.

A könyv három fő részből áll:

Az *első rész* ismerteti az előbbiekben említett adatbázis-kezelő rendszerek szerkezetét és nyelvi jellemzőit. Gyakorlott programozók ebben a részben megismerhetik a rendszer sajátosságait. Akik most tanulnak programozni, megfelelő ismereteket meríthetnek belőle a további anyagrészek megértéséhez is.

A *második rész* az adatbázisok létrehozásától a feldolgozott adatok megjelenítéséig a rendszer alkalmazásának igényei szerint mutatja be a parancsok adta lehetőségeket egy-egy adott felhasználási környezetben.

A *harmadik rész* a két adatbázis-kezelő rendszer parancsait tartalmazza alfabetikus sorrendben, részletes leírásukkal. Az első két részben megismert parancsok összes formáját megtalálhatja itt az olvasó.

A *függelék* tartalmazza a parancsok, műveletek, beépített függvények pár szavas magyarázatokat adó listáit és a hibaüzenetek felsorolását. A könyv anyagának tanulmányozását célszerű a 18. fejezettel kezdeni, ez ugyanis a legfontosabb fogalmakat és a téma tárgyalása során használt jelöléseket foglalja össze.

Az adatbázis-kezelő rendszer üzenetei a könyv összes példájában angol és magyar nyelven is szerepelnek, a valóságban természetesen csak az alkalmazott rendszerre jellemző nyelvűek lehetnek.

Ötleteivel hasznos segítséget nyújtott Styaszni Gyula és Sallai László. Köszönetet mondok Pogány Csabának a kötelező lektori feladatot jóval meghaladó lelkiismeretes, alapos munkájáért és számos javaslatáért, amelyeket igyekeztem értékesíteni; továbbá a kiadónak, hogy a szerkesztésben is értékes segítséget nyújtott, és könyvem kiadását lehetővé tette. Külön köszönet illeti *Pető Endrét,* akinek közreműködése nélkül ez a könyv nem készülhetett volna el.

Budapest, 1986. január 25.

A SZERZŐ

ELSŐ RÉSZ

AZ ADATBÁZIS-KEZELŐ ISMERTETÉSE

1. Bevezető

A dBASE II és PROP-BASE adatbázis-kezelő rendszer mikroszámítógépre készült, kis és közepes méretű adatbázisok egyszerű kezelésére.

Fontosabb szolgáltatásaik a következők:

— adatbázisrendszer létesítése,
— igen egyszerű adatbővítés és -törlés,
— adatstruktúra- és programszerkesztés,
— adatok megjelenítése képernyőn és nyomtatón,
— programok és adatok nagymértékű függetlenítése egymástól,
— jelentések, beszámolók készítése egy vagy több adatbázisból,
— automatikus szorzás, osztás, részösszegképzés, teljes összeg képzése és más adatmanipulációk,
— betekintés az adatbázisokba (egyidejűleg egy vagy több adatbázisba).

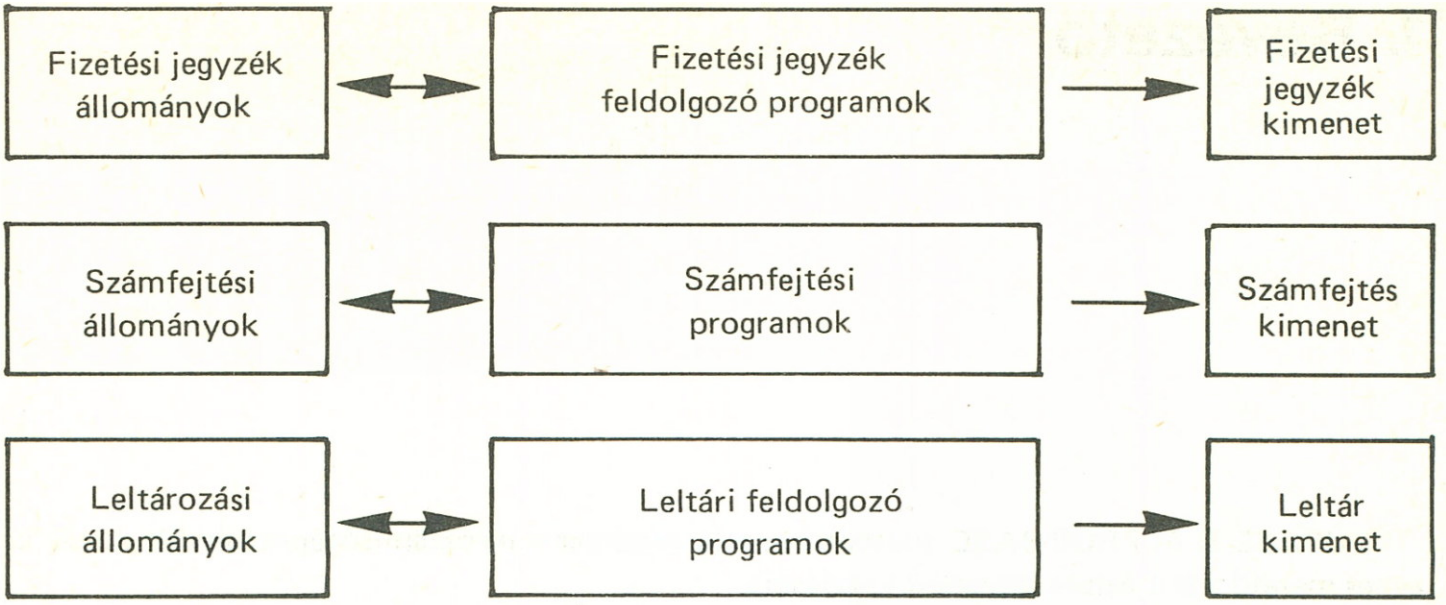
Egy adott feladat elvégzésére szolgáló program megírása dBASE II vagy PROP-BASE rendszerben körülbelül tizedannyi időt vesz igénybe, mint más magas szintű nyelveken, például BASIC-ben, PASCAL-ban, COBOL-ban, FORTRAN-ban vagy PL/1-ben.

A dBASE II és a PROP-BASE által létrehozott adatbázis információját átalakíthatjuk olyan formátumúra, amely kompatibilis más (BASIC, PASCAL, COBOL, FORTRAN, PL/1 stb.) rendszerekkel. Az adatbázis-kezelő rendszerrel feldolgozhatunk olyan adatokat, amelyek más — például az előbb említett — rendszerekből származnak.

1.1 AZ ADATBÁZIS-KEZELŐ RENDSZER

Az adatbázis-kezelő rendszerek, mint a dBASE II vagy a PROP-BASE is, lényegesen különböznek az állománykezelő rendszerektől. Egy *állománykezelő rendszer* szerkezetét az 1-es ábra mutatja.

A fizetésjegyzék-programok a fizetésjegyzék-állományokat, a számfejtési programok a számfejtési állományokat, a leltározási programok pedig a leltározási állományokat dolgozzák fel. Ahhoz, hogy egy olyan információt kapjunk, amely a különböző állományok adatainak kombinációjából származik, új program írására lenne szükségünk, amely azonban nem biztos, hogy működik, hiszen a különböző állományokból származó adatok inkompatibilisek lehetnek. Az adatokat pedig a programok olyan mértékben takarhatják (rejthetik el), hogy azok kinyerése már nem kifizetődő.

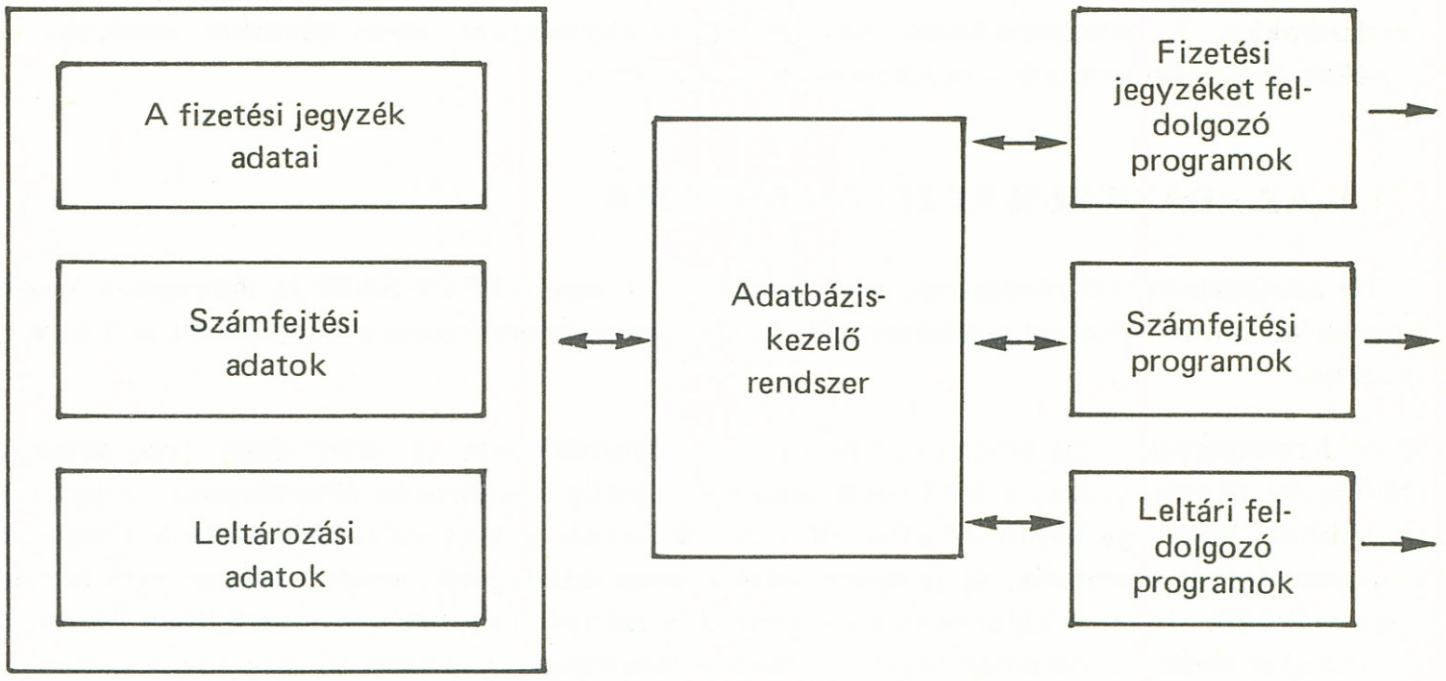


1. ábra. Állománykezelő rendszer szerkezete

Az *adatbázis-kezelő rendszer* az adatokat a saját rendezési elvei szerint rendszerezi, és ez lehetővé teszi, hogy az adatokból (rekordokból) könnyen vegyük ki a szükséges információt. Egy adatbázis-kezelő rendszer szerkezete a 2-es ábrán látható.

Az adatok megjelenítését és a velük végzett műveleteket az adatbázis-kezelő rendszer hajtja végre, nem pedig az egyedi felhasználói programok. Valamennyi felhasználói rendszer hozzáférhet minden adathoz. Egy állománykezelő rendszerben mindez az adatok nagymértékű többszörözését követelné meg. Eltekintve az adatbeviteli hibalehetőségektől, a jó adatok sértetlenségét meglehetősen nehéz fenntartani, ha ugyanazoknak az adatoknak a másolatai szükségesek a különböző állományokban. Az adatbázis-kezelő rendszer ezt a problémát elkerüli.

Egy állománykezelő rendszerben újfajta feldolgozáshoz új program megírására és új állományok készítésére van szükség. Az adatbázis-kezelő rendszerben is új programot kell írni, amely másként fér hozzá az adatokhoz, de az adatbázis-kezelő rendszer gondoskodik arról, hogy az adatokat ne kelljen más struktúrába átalakítani.



2. ábra. Adatbázis-kezelő rendszer szerkezete

Amikor egy rekordhoz új típusú adatokat kell hozzátenni (például személyi nyilvántartási rendszerben a fizetésemeléseket), az állománykezelő programokat módosítani kell. Az adatbázis-kezelő rendszerben ezzel szemben az adatok bővítése és megváltoztatása nem befolyásolja azokat a programokat, amelyek ezeket az új információkat nem használják; egyszerűen nem vesznek arról tudomást, hogy ezek az adatok is ott vannak.

Az adatbázis-kezelő rendszerek többnyire hierarchikusak vagy relációsak. (Ezek az elnevezések az adatok nyilvántartási rendszerére vonatkoznak.) A *hierarchikus* rendszer meglehetősen összetett és nehézkesen kezelhető, mert az adatok összefüggését halmazok, láncoló listák tartják nyilván, és mutatók (pointerek) adják meg a rendszernek a szükséges információt. Nagyon gyorsan kimerülhetnek azok a tárak, amelyek a listák listáinak listáit, illetve a mutatók mutatóinak mutatóit tartalmazzák. A *relációs* adatbázis-kezelő rendszer, amilyen például a dBASE II vagy a PROP-BASE is, sokkal egyszerűbb. Az adatelemek közötti viszonyok egy kétdimenziós táblázat segítségével írhatók le (lásd 1-es táblázat).

1. táblázat: A relációs adatbázis-kezelő rendszer szerkezete.

1. oszlop	2. oszlop	3. oszlop	4. oszlop	5. oszlop
Számlaszám	Szállító	Leírás	Összeg	Munkaszám

A táblázat sorai a rekordok. Minden oszlop egy-egy mező a rekordokon belül. A táblázat minden egyes bejegyzése egy önállóan kezelhető érték (nincsenek tömbök, halmazok stb.). Az egyes oszlopokban a bejegyzéseknek azonos típusúaknak kell lenniük. Minden rekord (sor) egy önálló egység, a rekordok (sorok) sorrendje lényegtelen.

A gyakorlati alkalmazás szempontjából döntő, hogy az adatok számának növekedésével az adatbázisok, illetve a rekordok nem válnak bonyolultabbakká, csak nagyobbakká.

A könyvben szereplő adatbázis-kezelő rendszerekben az állományoknak más rendszerektől eltérő szerkezetük és elnevezésük van. Esetünkben „adatbázis-kezelő állomány" a rendszer által létrehozható összes állomány neve (1.7. rész). Ezen belül az adatokat tartalmazó állományokat „adatbázis-állomány"-nak (1.7.1. rész) nevezzük.

1.2. AZ ADATBÁZIS-KEZELŐ HARDVER- ÉS SZOFTVERKÖRNYEZETE

Az adatbázis-kezelő rendszer a következő hardver- és szoftverkörnyezetet igényli:

- 8080, 8085 vagy Z80-as mikroprocesszoros rendszer,
- a legtöbb mikroszámítógépen minimálisan 48 kbyte memória; néhány gép — mint például a NORTH STAR — esetében 56 kbyte,
- CP/M (1.4 vagy 2.x változat), CDOS, CROMIX vagy PROPOS, illetve ezekkel kompatibilis operációs rendszer,
- háttértár (rendszerint mágneslemez),
- vezérelhető fénypontpozíciós megjelenítő, ha ki akarjuk használni az adatbázis-kezelő rendszer összes lehetőségét,
- nem kötelezően: nyomtató (ezt néhány parancs használja).

1.3 SPECIFIKÁCIÓK – MÉRETKORLÁTOZÁSOK

Az adatbázis-kezelő rendszer a következő méretkorlátozásokkal rendelkezik. (A felsorolásban nem szereplő jelölések a függelékben megtalálhatók.)

Egy időben nyitott állományok száma	max.	16
Rekordok száma egy adatbázis-állományban	max.	65535
Mezők száma egy rekordban	max.	32
Karakterek száma rekordonként	max.	1000
Karakterek száma egy mezőben	max.	254
Memóriaváltozók száma	max.	64
Függőben levő GET-ek száma	max.	64
Memóriaváltozók hossza	max.	254 karakter
Memóriaváltozók összterjedelme	max.	1536 karakter
Változók nevének hossza	max.	10 karakter
Állománynevek hossza (CP/M szerint)	max.	8 karakter
Karakterlánc hossza	max.	254 karakter
Parancssor hossza	max.	254 karakter
Fejléc hossza a jelentésben	max.	254 karakter
A második fejléc hossza a jelentésben	max.	60 karakter
Indexelési kulcs hossza	max.	100 karakter
Pontosság		10 számjegy
Számábrázolási tartományok		

$$+1 \times 10^{-63} < R < +1{,}8 \times 10^{63}$$
$$-1 \times 10^{-63} > R > -1{,}8 \times 10^{63}$$

A SUM és a REPLACE parancsban végrehajtandó feladatok száma	max.	5
A USE és a SET parancsban a munkaállományhoz használható indexállományok száma	max.	7

1.4 AZ ADATBÁZIS-KEZELŐ RENDEZÉSI ELVEI

Az adatbázisok adatait általában szabályosan, meghatározott sorrendben szeretnénk elérni. Néhány adatbázis esetében az adatok bevitelének sorrendje megegyezik azzal a sorrenddel, ahogyan az adatokat felhasználni szeretnénk. Sok esetben azonban különböző sorrendben bevitt adatokra vagyunk kíváncsiak. Az adatbázis-kezelő rendszerben az adatok rendezése a SORT és az INDEX parancs segítségével oldható meg. (Mindkét parancs leírását lásd a 8.1. és a 11.2. részben.)

A *SORT* parancs teljes rekordokat mozgat, az adatbázist újrarendezve. A rendezés előírható bármely megadott mező (pl. név, sorszám stb.) tartalma szerint, növekvő vagy csökkenő sorrendben. A rendezés alapját képező mezőt *kulcsnak* hívjuk. Az ilyen rendezésnek az a hátránya, hogy a különböző célú felhasználások különböző mezők (kulcsok) szerinti sorbarendezést kívánnak. További hátrány, hogy az újonnan bevitt rekordok nem a megfelelő helyre, hanem az állomány végére kerülnek, ezért ha a sorrendet meg akarjuk tartani, minden adatbevitel után újra sorba kell rendezni az adatokat. Az adatok megkeresése (kikeresése) is viszonylag lassú, mert az ilyen adatbázisban soros módon (lásd LOCATE, 84. és 190. old.) kell keresni.

Az *INDEX* a fenti problémákra ad megoldást. Ilyenkor egy új állományt hozunk létre, amely csak az érdekelt kulcsokat tartalmazza, s nem a teljes adatbázist. A *kulcs* ebben az esetben egy kifejezés, amely adatbázismezők, változók, állandók, függvények és műveletek kombinációja lehet. Egy leltári rendszerben az alkatrész száma, a készlet, a költség stb. lehetnek a jellemző kulcsok. Egy személyi nyilvántartási rendszerben a név, a FEOR szám, vagy annak utolsó három számjegye adhatja az indexelés alapját. Az indexelt adatbázisban csak a kulcsok rendezését végzi el az adatbázis-kezelő rendszer azokkal a mutatókkal (a pointerekkel) együtt, amelyek az indexállomány rekordjaihoz tartozó rekordokra mutatnak. Az adatbázis-kezelő rendszer az indexekhez az úgynevezett B*-fákat használja; ezek hasonlóak a bináris fákhoz, de sokkal gazdaságosabban szervezik a tárolást, és sokkal gyorsabb működést tesznek lehetővé. (Indexelésnél a FIND adatkereső parancs végrehajtása közepes és nagy adatbázisokon mintegy 2 másodpercet vesz igénybe.) Ha az alkalmazás megkívánja, több, különböző kulcs szerinti indexállományt hozhatunk létre. Készíthetünk egyetlen adatbázishoz is több indexállományt, a szállító neve, az ügyfél száma, a postai irányítószám, vagy bármely más kulcs szerint rendezve.

Ha új adatot viszünk az adatbázisba, akkor az éppen használatban levő indexállományba is bekerül a szükséges információ. Az indexelt adatbázis egyik előnye, hogy gyorsan rátalálunk arra a rekordra, amelyben a kívánt adatok vannak.

1.5 AZ ADATBÁZIS-KEZELŐ REKORDJAI ÉS ADATTÍPUSAI

Az általunk tárgyalt adatbázis-kezelő rendszerek személyi számítógépekre készültek, ezért számolni kell korlátaikkal. (De mint azt a felhasználó az alkalmazás során látni fogja — a gyakorlatban meglehetősen ritkán ütközünk ezekbe a korlátokba.) Ezek az adatbáziskezelő rendszerek egy állományban maximum 65536 rekordot engednek meg. Egy rekord maximum 32 mezőt, illetve 1000 karaktert foglal magában. A rekordot úgy kell elképzelni, mint egy 1000 karakter hosszú szalagot, amelyet legfeljebb 32 mezőre oszthatunk fel. A mezők egyenként legfeljebb 254 karaktert tartalmazhatnak. Használhatunk csak egy karakterből (és mezőből) álló rekordot is.

Példánkban minden rekord 5 mezőből áll, s a teljes hosszúság 58 karakter. (Lásd a 3-as ábrát.)

Adattípusok

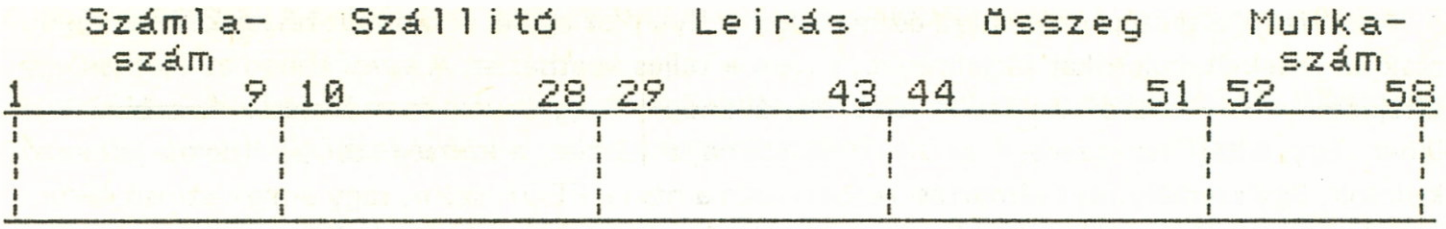
Mint már mondottuk, egy adatbázis-állományban minden azonos nevű mező csak azonos típusú adatokat tartalmazhat.

Az adatok a következő típusúak lehetnek:

Karakter: minden megjeleníthető ASCII karakter, beleértve a számjegyeket, a szimbólumokat és a szóközt.

Numerikus: pozitív és negatív számok; a méretkorlátozások listájában megadott határokon belül.

Logikai: igaz/hamis (true/false vagy yes/no) értékek, amelyek egy karakternyi mezőt foglalnak. **T**, **t**, **Y** és **y** igaznak; **F**, **f**, **N** és **n** hamisnak értékelődik.



3. ábra. A példában szereplő rekord felépítése.

1.6 AZ ADATMEZŐK

Minden mezőnek nevet kell adni, melyet az adatbázis-kezelő rendszer felismer, amikor a mezőt keressük. A mezőnév maximum 10 karakter hosszú lehet, betűvel *kell* kezdődnie, tartalmazhat számjegyeket és közrezárt kettőspontot is.

Példák:

A	helyes
A123456789	helyes
Haz:Szam	helyes, kis- és nagybetű is lehet benne
Haz Szam	nem helyes a szóköz miatt
A123,B456	nem helyes a vessző miatt
Varos:	nem helyes, hiszen csak a közrezárt kettőspont megengedett

Tanácsos hosszabb neveket használnunk, hogy az elnevezés kifejezze, tükrözze a tartalmat. A „Haz:Szam" ilyen szempontból sokkal jobb, mint a „Haz", és nyilvánvalóan jobb, mint a „H". Ha legfeljebb 9 karaktert használunk a mezők neveiben, a memóriaváltozók kezelése egyszerűbb, mert lehet a mezőnévhez tartozó memóriaváltozó neve ugyanaz, csak egy karakterrel (például egy kezdő „M"-mel) több.

A programok készítésekor célszerű az adatbázis-kezelő rendszernek szóló parancsokat nagybetűvel írni; nagy- és kisbetűvel pedig a mezőneveket, memóriaváltozókat és más, ellenőrizendő értékeket. Ennek hasznosságát akkor értjük meg igazán, amikor egy parancsállományhoz később visszatérünk, hogy azon valamilyen módosítást végezzünk.

1.7 AZ ADATBÁZIS-KEZELŐ ÁLLOMÁNYTÍPUSAI

Az adatbázis-kezelő rendszerben használt állománynevek maximum 8 karakter hosszúak lehetnek, és három karakternyi állománynév-kiterjesztést tartalmaznak egy — az állománynevet követő — pont után. Az állománynév-kiterjesztést az adatbázis-kezelő rendszer — ha nem adtuk meg másként — automatikusan az állományhoz rendeli. Az adatbázis-kezelő rendszerben az állománynév-kiterjesztés meghatározza az állomány típusát. Nem célszerű a kettőspont használata az állománynévben, mert bár az adatbázis-kezelő rendszer ezt megengedi, a CP/M operációs rendszer parancsai nem tudják az ilyen nevű állományt kezelni. A 10 karakteres állománynevet a CP/M levágja 8 karakter hosszúságig. A kis- és nagybetűvel írt állományneveket a CP/M nagybetűssé alakítja, de használhatjuk a kisbetűket is, ha azok jobban néznek ki a parancsállományban (pl. Nevek.DBF).

Az adatbázis-kezelő rendszer állománya nem más, mint egyetlen név alatt kezelt azonos típusú információk gyűjteménye, vagyis hasonló egy nagy iratgyűjtőhöz.

A dBASE II és a PROPBASE 7-féle állománytípussal dolgozik:

1.7.1 Adatbázis-állomány

Állománynév-kiterjesztése: **DBF**

Ezekben az állományokban vannak az adatok. A hárombetűs állománynév-kiterjesztést az adatbázis-kezelő rendszer automatikusan hozzáteszi az állománynévhez, amikor a CREATE paranccsal egy új állományt készítünk. Ne próbáljuk ezeket az állományokat szövegfeldolgozó program alatt kezelni. (Lásd a 43. oldalon a 4.1, és a 142. oldalon a 11.2. részt!)

1.7.2 Indexállomány

Állománynév-kiterjesztése: **NDX**

Az INDEX parancs végrehajtásakor automatikusan jön létre. Az indexelés nagy segítséget nyújt a gyors adatkereséshez a nagyobb adatbázisokban. (Lásd a 81. oldalon a 8.1—2, és a 175. oldalon a 11.2. részt!)

1.7.3 Parancsállomány

Állománynév-kiterjesztése: **CMD**

Parancssorozatot tartalmaz, amely egy-egy gyakran használt részfeladat, vagy egy összetett probléma megoldására szolgál. A parancsállomány megírása szövegfeldolgozó, szövegszerkesztő programokkal történhet, de az adatbázis-kezelő rendszerben is van rá lehetőség. (Lásd a 91. oldalon a 9.1, és a 153. oldalon a 11.2. részt!)

1.7.4 Jelentésformátum-állomány

Állománynév-kiterjesztése: **FRM**

Az ilyen típusú állományokat automatikusan készíti el az adatbázis-kezelő rendszer, amikor a REPORT parancsot bevezető párbeszédet hajtjuk végre. Tartalmazza a fejrészt, továbbá hogy mit kell összegezni, az egyes oszlopok tartalmát, stb. Az állomány szövegfeldolgozó, szövegszerkesztő program alatt módosítható. (Lásd a 63. oldalon a 6.2. és a 214. oldalon a 11.2. részt!)

1.7.5 Formátumállomány

Állománynév-kiterjesztése: **FMT**

@ jelű parancsokat és azok részeit — kivéve a READ parancsot —, illetve megjegyzéseket tartalmaz. Létrehozása, módosítása a parancsállományéhoz hasonlóan történhet, de vegyük figyelembe, hogy az adatbázis-kezelő rendszer ^MODIFY COMMAND^ paranccsal aktivizált szövegszerkesztője <.CMD> állománynév-kiterjesztést ad az állománynak, ha másként nem adjuk meg! A formátumállományt a ^SET FORMAT TO <állománynév>^ parancs teszi hozzáférhetővé, és a READ parancs aktivizálja. (Lásd a 235. oldalon a 11.2. részben).

1.7.6 Memóriaállomány

Állománynév-kiterjesztése: **MEM**

A memóriaállományt automatikusan hozza létre az adatbázis-kezelő rendszer, amikor a programozó SAVE utasítással mágneslemezre íratja a számítási folyamat eredményeit. Tehát a memóriaállomány azokból a változókból áll, amelyekre a későbbiekben szükség lesz. Maximum 64 érték őrizhető meg ilyen módon egy memóriaállományban; egyenként legfeljebb 254, együttesen legfeljebb 1536 karakter hosszúságban. Az így megőrzött memóriaváltozók és azok értéke szükség esetén majd a RESTORE paranccsal visszamásolható. (Lásd az 57. oldalon az 5.2.3—4, és a 226. oldalon a 11.2. részt!)

1.7.7 Szöveg típusú állomány

Állománynév-kiterjesztése: **TXT**

A szöveg típusú állomány kimeneti (output) szövegeket tartalmaz. Ez az eredményállomány akkor jön létre, amikor a SET ALTERNATE parancs segítségével mágneslemezen is szeretnénk megőrizni mindazt, amit a képernyőre írtunk. Ez a lehetőség jól felhasználható például ahhoz, hogy a rendszerünk működéséről naplót vezessünk, s ezt később szerkesszük, kinyomtassuk vagy megőrizzük. Szintén TXT típusú állományok jönnek létre, amikor a COPY . . . SDF parancsokat hajtjuk végre. (Lásd a 233. oldalon a 11.2. részben!)

2. Az adatbázis-kezelő nyelvi jellemzői

A felsorolásban nem szereplő jelölések és fogalmak a függelékben találhatók.

2.1 ÁLLANDÓK ÉS VÁLTOZÓK

Az adatbázis-kezelő rendszerekben szereplő mennyiségek lehetnek állandók vagy változók. Tartalmazhatnak számot (numerikus adat); logikai értéket, mint a T (True = igaz) és az F (Fals = hamis); továbbá karaktert (minden megjeleníthető betű, számjegy, jel és szóköz), illetve karakterláncot. A karakterlánc egyszerűen karakterek együttese (beleértve az üres helyeket, számjegyeket és egyéb megjeleníthető karaktereket). A karakterláncok módosíthatók, adatokként is használhatók, és műveleteket is végezhetünk velük.

A karakterláncok tartalmazhatnak bármilyen megjeleníthető karaktert (beleértve az üres közt is), de ha az & jelet karakterként kívánjuk alkalmazni, akkor két üres köz közé kell zárnunk, mert az adatbázis-kezelő rendszer az & karaktert a makró beépített függvény jelölésére használja. (Ennek leírását lásd a 2.7. részben, a 32. oldalon!) A „részkarakterlánc" (substring) egy adott karakterláncnak bizonyos részét jelenti.

2.2 ÁLLANDÓK

Egy állandó értéke nem változik, akár az adatbázisban, akár a számítógép memóriájában van. Úgynevezett literál értékű, azaz pontosan azt az értéket adja, amit mutat. Lehet számjegy, mint a leírt 3 (három), vagy logikai érték is. A karakterek és a karakterláncok is lehetnek állandók, de egy kissé eltérően kell őket kezelni.

Ha egy karaktert vagy karakterek együttesét karakterállandóként kezelünk, akkor határolójelek (aposztrófok, idézőjelek vagy szögletes zárójelek) közé kell zárnunk. Ha a karakterlánc valamelyik határolójelet is tartalmazza, akkor ettől eltérő határolójelet kell alkalmazni.

Például: "Ez az év '86".

2.3 VÁLTOZÓK

A változó értéke módosítható, neve bármilyen megengedett, az adatbázis-kezelő rendszer által elfogadott elnevezés lehet. (Betűvel kezdve, egészen 10 karakter hosszúságig, esetleg szabadon választott, közbeiktatott kettősponttal. Tartalmazhat számjegyeket, nem lehet benne üres köz.)

Amikor egy változónak nevet adunk, arra törekedjünk, hogy a változó neve közérthető legyen, bárki megérthesse annak jelentését. Ha csak 9 — vagy annál kevesebb — karaktert használunk az adatbázismezők nevének jelzésére, és ugyanezt a nevet akarjuk memóriaváltozónévként is felhasználni, akkor az utóbbi elé egy „M" betűt tegyünk. A későbbiekben világossá válik, hogy tulajdonképpen miért indokolt a majdnem azonos elnevezés. Sokkal egyszerűbb ilyenkor a helyzet, mintha egy teljesen új nevet vettünk volna föl a memóriaváltozó számára.

Amikor egy változóba adatot juttatunk, az adatbázis-kezelő rendszer azonnal felismeri, hogy milyen típusú az adat. Ezután az adat már csak típusának megfelelő műveletekben szerepelhet. Egy karakterláncot például ne próbáljunk meg osztani egy számmal, mert ez helytelen.

Az adatbázis-kezelő rendszer a parancsok végrehajtása során egy határolók nélküli karakterlánccal találkozva először ellenőrzi, hogy az parancsszó vagy paraméter-e. Ha egyik sem, akkor megvizsgálja, hogy változó nevéről van-e szó.

2.3.1 Adatbázismezők

A változók többsége adatbázismező. Az adatbázis-kezelő rendszer ellenőrzi, hogy az aktuális munkatérben levő adatbázis tartalmaz-e a megadott karakterláncnak megfelelő nevű mezőt. A mezőkről már szó volt az 1.5. és az 1.6. részekben.

2.3.2 Memóriaváltozók

A változók lehetnek memóriaváltozók is. Ha az adatbázis-kezelő rendszer nem talált a megadott karakterláncnak megfelelő nevű mezőt, megvizsgálja, hogy létezik-e ilyen elnevezésű memóriaváltozó.

Az adatbázis-kezelő rendszer fenntart egy területet a memóriában, maximum 64 memóriaváltozó tárolására. Ezek mindegyikének legnagyobb hosszúsága 254 karakter lehet, de a teljes terület, amely az összes memóriaváltozó részére rendelkezésre áll, 1536 karakternyi. Úgy is felfoghatjuk ezt, mint egy 64 elemből álló tárolóhely-sorozatot.

A memóriaváltozókat arra lehet használni, hogy vagy átmeneti adatokat tároljunk bennük, vagy pedig a beérkező adatokat ne közvetlenül az adatbázis-állományokba juttassuk. Például először ilyen tároló rekeszbe tehetjük a dátumot, MDatum néven. Ezután ezt bármikor megkapjuk, ha megkérdezzük, hogy mi az MDatum változó értéke. Ezt az értéket tehát bármelyik adatbázisunk <Datum> mezejébe másolhatjuk (anélkül, hogy ismét be kellene gépelni).

2.4 ADATBÁZIS-KEZELŐ MŰVELETEK

A *műveletek* segítségével az adatbázis-kezelő rendszer különböző vizsgálatokat végez az adatokon.

Az *aritmetikai* és a *logikai* műveletek esetében gömbölyű zárójeleket ( ) használhatunk a csoportosításhoz. A zárójelek használatával befolyásolhatjuk a műveletek végrehajtási sorrendjét, mert először a zárójelben szereplő művelet(ek) elvégzése történik meg.

A felsorolt műveletek kijelöléséhez tehát rendelkezésünkre állnak a

( ) : zárójelek a csoportosításhoz.

2.4.1 Aritmetikai műveletek

*	:	szorzás
/	:	osztás
+	:	összeadás
–	:	kivonás

Az aritmetikai kifejezések értékének kiszámításakor a *műveletek elvégzése során* „elsőbbségi" sorrendet kell betartani. Az aritmetikai műveletek végrehajtása balról jobbra halad az elsőbbségi sorrend figyelembevételével. Az *elsőbbségi sorrend* a következő: a zárójeleken belüli műveletek, majd a szorzás és osztás, majd pedig az összeadás és kivonás. Elsőbbségben egymást meg nem előző műveletek végrehajtási sorrendje: balról jobbra. Az alábbi példák mutatják mindezt:

17/33*72+8 = 45.09	(osztás, szorzás, majd összeadás)
17/(33*72+8) = 0.00644	(szorzás, összeadás, majd osztás)
17/33*(72+8) = 41.21	(osztás, összeadás, majd szorzás)

2.4.2 Összehasonlító műveletek (relációk)

Az összehasonlító műveletekben szereplő relációk:

<	:	kisebb, mint
>	:	nagyobb, mint
=	:	egyenlő
<>	:	nem egyenlő
<=	:	kisebb vagy egyenlő
>=	:	nagyobb vagy egyenlő

A két jellel (például <>) kijelölt összehasonlító műveletekben a jelek sorrendje meghatározott (például >< tilos).

Az összehasonlító műveletek aritmetikai, illetve karakterértékeket hasonlítanak össze, és logikai eredményt állítanak elő.

2.4.3 Logikai műveletek

.NOT.	:	logikai nem
.AND.	:	logikai és
.OR.	:	logikai megengedő vagy
$	:	részkarakterlánc logikai művelet (részkarakterlánc-keresés)

A logikai műveleteknél a .NOT., az .AND. és az .OR. kulcsszó előtt és mögött szereplő pont szerves része a jelnek, ezért mindig ki kell azt is írni.

A logikai *műveletek között* elsőbbségi sorrend van. A logikai műveletek végrehajtása balról jobbra halad az elsőbbségi sorrend figyelembevételével. Az *elsőbbségi sorrend* a következő: zárójelen belüli műveletek, .NOT., .AND., .OR., $.

A *részkarakterlánc logikai művelet* különösen fontos. Hasznosságát nagyon hatékony keresési képességének köszönheti. Formája a következő:

^<részkarakterlánc> $ <karakterlánc>^

Ez a művelet a $ jel jobb oldalán megadott karakterláncban azt a részkarakterláncot keresi, amelyet a bal oldalon megadtunk. Ha megtalálta a keresett karakterláncot, akkor a művelet eredménye T, egyébként F. A művelet karakterlánc-változókkal és karakterlánc-állandókkal dolgozik.

A logikai műveletek (kivéve a részkarakterlánc logikai műveletet) a logikai eredményt adó műveletek és beépített függvények eredményeit értékelve logikai eredményt állítanak elő (igaz, hamis). Az .AND. logikai művelet csak akkor ad igaz eredményt, ha mind az előtte, mind az utána álló érték igaz. Az .OR. igaz eredményt ad, ha az előtte és az utána álló érték közül bármelyik igaz. A .NOT. az utána következő érték ellentettjét adja. A zárójeles kifejezésekben előbb a zárójelben levő művelet elvégzése valósul meg.

Logikai művelet	Eredménye
F .AND. F	F
F .AND. T	F
T .AND. F	F
T .AND. T	T
F .OR. F	F
F .OR. T	T
T .OR. F	T
T .OR. T	T
.NOT. F	T
.NOT. T	F
(F .AND. T) .OR. (F .AND. T)	F
.NOT. ((F .OR. T) .AND. (F .OR. T))	F

Tegyük fel, hogy az adatok megjelenítését szolgáló egyik parancs tartalmazza az alábbi kifejezést:

```
... (MunkaSzam=730  .OR.  MunkaSzam=731);
        .AND.   (Datum >= 791001  .AND.;
                 Datum <= 791031)
```

A példa szerinti parancs megjeleníti a képernyőn az összes 1979 októberében érvényes rekordot, amely a 730-as és a 731-es munkaszámra költségterhelést tartalmaz. (A parancsot a pontosvessző használatával több sorba írtuk.)

Példánkban az adatbázis-kezelő rendszer minden egyes rekordra megvizsgálja, hogy

- a munkaszám (MunkaSzam) egyenlő-e 730-cal (T vagy F),
- a munkaszám egyenlő-e 731-gyel (T vagy F),
- az időpont (Datum) nagyobb vagy egyenlő-e, mint 791001 (T vagy F),
- az időpont kisebb vagy egyenlő-e, mint 791031 (T vagy F).

Az adatbázis-kezelő rendszer ezután három logikai műveletet (.OR., .AND., .AND.) végez el mielőtt eldöntené, hogy az illető rekordot ki kell-e írni a képernyőre, vagy sem.

Az első .AND. hatására az adatbázis-kezelő pusztán azokat a rekordokat fogja megjeleníteni, amelyeknél igaz a bal és jobb oldalon álló zárójeles kifejezések mindegyike. Az első zárójelben szereplő kifejezés kiértékelésekor először a MunkaSzam nevű mezőt vizsgálja meg. Abban az esetben, ha ebben a mezőben az érték 730 *vagy* 731, ez a részkifejezés igaz értéket kap. Ha a mező valamilyen más értéket tartalmaz, ez a részkifejezés hamis lesz, és természetesen ez a rekord már nem írható a képernyőre. Ha az első részkifejezés igaz, az adatbázis-kezelő rendszernek további ellenőrzést kell végeznie; a második részkifejezés szerint megvizsgálva a Datum nevű mező tartalmát. Ha a mező tartalma 791001 *és* 791031 értékek közé esik, a kifejezés igaz. Ekkor a rekord a képernyőre íratható. Ellenkező esetben a teljes kifejezés hamis, és az adatbázis-kezelő rendszer a következő rekordot vizsgálja meg az előbbi szempontok szerint.

2.4.4 Karakterlánc-műveletek

A *karakterlánc-műveletek* karakterlánc-eredményt adnak.

+	:	karakterláncok összeadása — üres helyekkel együtt
−	:	karakterláncok összeadása — az üres helyek átmozgatásával

A karakterláncok összeadása egyszerűen azt jelenti, hogy a karakterláncok a megadott helyen (pl. képernyőn, nyomtatón vagy egy változóban) a megnevezés sorrendjében egymás mögé kerülnek.

A + jel és a − jel egyaránt két karakterláncot egyesít. A „plusz"-jel úgy illeszti egymáshoz a két karakterláncot, ahogyan azokat találta. A „mínusz"-jel a karakterlánc mellett levő üres helyeket az egyesített karakterlánc végére teszi. Nem szünteti meg ezeket, de a legtöbb esetben elegendő az eltüntetésük, ugyanis a megjelenítésnél nem látszik az üres hely az egyesített karakterláncok mögött.

Ha a karakterlánc mögött levő üres helyeket valóban meg is akarjuk szüntetni, használjuk a ^TRIM^ beépített függvényt. (Lásd 2.5. rész,30. oldal.)

2.5 BEÉPÍTETT FÜGGVÉNYEK

A beépített függvények olyan speciális célú műveleteket végeznek el, amelyek végrehajtása a hagyományos matematikai, összehasonlító, logikai és karakterlánc-műveletekkel nehezen, vagy egyáltalán nem lennének megoldhatók. A beépített függvények eredménye megjeleníthető, például az interaktív ? jelű parancs segítségével:

? <beépített függvény neve { (<változó>) | (<karakterlánc>) | (<kifejezés>) } >

MEGJEGYZÉS

A beépített függvények megadásakor a zárójeleket és az idézőjeleket a fejezetben közölt módon *használni kell.*

A beépített függvények a következők

CHR (<szám>)

A megadott *számnak megfelelő ASCII karaktert* adja. Attól függően, hogy az alkalmazott terminál az általános ASCII kódokat hogyan használja, különböző műveleteket végeztethetünk el. Például a ^? CHR(12)^ törli a képernyőt; a ^? CHR(14)^ inverz videokaraktereket állít elő; stb. Más kódok a hardveregységek (például a nyomtató) beállítására szolgálhatnak.

EOF

Az (end of file) *állomány vége* beépített függvény. Igaz értéket ad, ha a használatban levő adatbázis-állomány végét már elértük; és hamis értéket ad, ha az állomány végét még nem értük el.

FILE ({ <változó> / "<állománynév>" / <kifejezés> })

Az *állománykereső beépített függvény.* Igaz értéket ad, ha a mágneslemezen a megadott állomány létezik; hamisat, ha nem létezik. Állománynév-kiterjesztés megadásának hiányában a beépített függvény automatikusan adatbázis-állományt értelmez. Ha ebben a beépített függvényben egy konkrét állománynevet adunk meg, azt idézőjelek közé kell tenni. Karakterlánc-változó esetében az idézőjelek nem szükségesek. Használhatjuk bármelyik érvényes karakterlánc-kifejezést is. A ^FILE("B:"+Adatbazis)^ megadja, hogy az <Adatbazis> nevű memória-változóban tárolt állománynév a B lemezegységen létezik-e.

INT ({ <változó> / <kifejezés> })

Az *egészrész* beépített függvény egy numerikus érték — a matematikaitól eltérő! — „egész-részét" adja oly módon, hogy egyszerűen elhagyja a tizedespont mögötti részt. A zárójelben lehet szám, lehet egy változó neve vagy egy kifejezés. Az utóbbi esetben először kiértékeli a kifejezést a rendszer, majd előállítja az eredmény egészrészét.

Az "INT(123.86)" 123-at ad eredményül, az "INT(—123.86)" pedig —123-at.

Ha a zárójelben változó van, akkor a változó értékének egészrésze lesz az eredmény.

Kerekítéskor, a legközelebbi egész számra való kerekítéshez használjuk a következő formátumot: "INT(<érték>+0.5)".

Az egészrész beépített függvény használható akkor is, ha *tizedes értékre* akarunk kerekíteni. Az "INT(<érték>*10+0.5)/10" kifejezés az értéket egy tizedesjegy pontosságra adja. Ha két tizedesjegyre kerekítünk, akkor tíz helyett százat, három tizedesjegy esetén ezret kell írnunk és így tovább.

LEN ({ <változó> / <karakterlánc> })

A *karakterlánc hosszát adó beépített függvény.* Megadja, hogy a megnevezett karakterláncban hány karakter van. Ez hasznos lehet például ahhoz, hogy a program a kezelő beavatkozása nélkül eldöntse: az információ elhelyezésére mekkora helyet kell biztosítania. Ha változót adunk meg, ez a függvény a memóriaváltozó vagy a mező méretét adja meg, nem pedig tartalmának a hosszát (mivel a nem használt pozíciókon üres helyek vannak).

Például:

```
? len("hossz")
. 5
```

PEEK(<cím>)

Ez a beépített függvény megadja, hogy a kérdezett *memóriacímen* milyen értéket találhatunk. Decimális címeket és értékeket használ. Alkalmazhatjuk például a ^CHR(PEEK(123))^ függvénykombinációban is, ez a 123-as memóriahely tartalmának megfelelő ASCII karaktert adja.

STR ({ <szám> / <változó> / <kifejezés> } , <hossz>[,<tizedesjegyek száma>])

Ez a beépített függvény *numerikus értéket alakít át karakterlánccá.* Egy számot vagy egy numerikus változó tartalmát átalakítja olyan karakterlánccá, amelyet megadtunk a hosszúság és a tizedesjegyek számának meghatározásával. A megadott hossz elég *kell,* hogy legyen legalább a számjegyek és egy tizedespont számára. Ha a numerikus érték rövidebb, mint a meghatározott hosszúság, a maradék rész üres helyekkel töltődik ki. Ha nem adjuk meg a tizedesjegyek számát, a rendszer 0-nak veszi.

Ez a beépített függvény többek között egyszerűbbé teszi a képernyőn való megjelenítést.

Például:

```
? STR(123.12,6,2)
```

TEST(<kifejezés>)

a felhasználó által bevitt kifejezés *szintaktikai ellenőrzésére* szolgál. Használata biztosítja, hogy ilyen hiba miatt a program futása ne álljon meg. Ha a kifejezés helyes, a TEST 0-tól eltérő értéket ad, ha a kifejezés helytelen, 0 a vizsgálat eredménye.

Például a 9.2. részben (92. old.) megismerhető IF parancs felhasználásával:

```
IF 0 = TEST(<kifejezés>)
   <Például egy üzenet a felhasználónak: 'Az adat hibás'>
   <programlépés: Újra kérni az adatot>
ELSE
   <továbblépés a programban>
ENDIF
```

TRIM ({ <változó> / <kifejezés> })

a karakterlánc-változót követő *üres helyek megszüntetésére* szolgál. A beépített függvényt használhatjuk az 5.2.1. részbeli (54. oldal) STORE parancs felhasználásával az alábbiak szerint is:

^STORE TRIM(<memóriaváltozó>) TO (<memóriaváltozó>)^.

Ebben az esetben ugyanabban a memóriaváltozóban is tárolhatjuk az új értéket.

TYPE(<kifejezés>)

az *adat típusának* meghatározására szolgál. Eredményül C, N vagy L karaktert ad attól függően, hogy az adat típusa karakter, numerikus vagy logikai.

VAL ({ <változó> / <karakterlánc> })

Ez a beépített függvény *számjegyekből álló karakterláncot számmá alakít át.* Ha a karakterlánc nem számjeggyel kezdődik, akkor 0 az eredmény; ha a kezdő számjegy(ek) után más karakter is található – kivéve a tizedespontot –, akkor csak a kezdő számjegy(ek)et veszi figyelembe. A VAL('123') egy 123-as számot, a VAL('tíz') egy 0-ás számot ad eredményül.

Használhatjuk ezt a beépített függvényt például a részkarakterlánc beépített függvénnyel együtt is.

Például:

^VAL($(<karakterlánc>,<kezdet>,<hossz>))^

!({ <változó> / <karakterlánc> })

Ez a beépített függvény *kisbetűről nagybetűre* állítja át a karakterláncot. Egy karakterláncban vagy egy karakterlánc-változóban a karaktereket a-tól z-ig megváltoztatja kisbetűről nagybetűre, minden más karakter változatlan marad. Nagyon célszerű például a bemeneti egységről (a billentyűzetről stb.) érkező adatok egységesítésére használni, mert így nagymértékben könnyebb a keresés és műveletvégzés.

Például:

STORE !(<memóriaváltozó>) TO <memóriaváltozó>

Ebben az esetben *is* tárolhatjuk ugyanabban a memóriaváltozóban az új értéket.

#

A *rekordszám* beépített függvény. A használatban levő adatbázis-állomány aktuális rekordjának sorszámát adja meg.

$ ({ <kifejezés> / <változó> / <karakterlánc> } , <kezdet>, <hossz>)

Ez a *részkarakterlánc* beépített függvény, amely egy karakterlánc-állandóból vagy egy karakterváltozóból a meghatározott pozíciótól kezdődően az adott hosszban karaktereket válogat ki. Például: legyen a Datum nevű változónk tartalma '841017' – ekkor a $(Datum,5,2) utasítás eredménye '17' lesz. Ahhoz, hogy ezeket a számjegyeket számmá alakítsuk át, használhatjuk a "VAL($(Dátum,5,2))" függvénykombinációt. Ha a <hossz> nagyobb a karakterlánc hosszánál, a beépített függvény – a kezdő karaktertől kezdve – minden karaktert figyelembe vesz.

Ne keverjük össze ezt a részkarakterlánc-beépített függvényt a részkarakterlánc-kereső logikai művelettel, amelyet a 2.4.3. részben (26. old.) ismertettünk.

& <változó>

A *makró helyettesítési* beépített függvény. Ha ezt a szimbólumot használjuk egy memóriaváltozó neve előtt, az adatbázis-kezelő rendszer a nevet a memóriaváltozó értékével helyettesíti. A memóriaváltozó értékének karakter típusú adatnak kell lennie. Alkalmazhatjuk, amikor egy összetett kifejezést gyakran kell használnunk; ha parancsállományok között adunk át előre nem ismert értékeket; vagy amikor egy parancsállományban a memóriaváltozók értéke a felhasználói program futása során töltődik fel. Arra is jó például, hogy egy – előzőleg bevitt – adatbázis nevét megkapjuk, vagy arra, hogy parancsainkat rövidítve adjuk meg a segítségével (^STORE 'Delete Record' TO D^; és a ^&D 5^ parancs a felhasználói program futásakor az ötödik rekordot fogja törlésre jelölni).

*

A *törölt rekordot jelző* beépített függvény. Igaz értéket ad, ha a rekord törlésre kijelölt állapotban van; hamis értéket, ha nem szerepel ilyen jelölés a rekordnál.

@({ <változó1> , <változó2> })
   { <karakterlánc1> , <karakterlánc2> }

Részkarakterlánc-kereső beépített függvény. Ezt így képzelhetjük el: „Hol van az 1-es karakterlánc a 2-es karakterláncban?" A művelet eredményeképpen azt a karakterpozíciót kapjuk meg, ahol az 1-es karakterállandó vagy karakterváltozó a 2-es karakterlánc-állandóban vagy karakterváltozóban kezdődik. Ha az 1-es karakterlánc a 2-esben nem fordul elő, „0" az eredmény.

E beépített függvény egyik alkalmazása: egy megadott karakterlánc kezdetének megállapítása. Ezután például használhatjuk az előbb ismertetett részkarakterlánc beépített függvényt.

Ha csak azt szeretnénk tudni, hogy egy karakterlánc megtalálható-e egy másik karakterláncban, célszerűbb a részkarakterlánc logikai műveletet használni (lásd 2.4.3. rész).

2.6 KIFEJEZÉSEK HASZNÁLATA BONYOLULTABB MŰVELETEK ELVÉGZÉSÉRE

A kifejezés nem más, mint műveleti jelekkel összekapcsolt állandók, változók, beépített függvények.

A kifejezések hossza 254 karakter lehet. Bármilyen bonyolult kifejezést megadhatunk. Gyakran előfordul – a hatásukat több rekordra kifejtő parancsoknál –, hogy feltétlenül szükséges mezőváltozó jelenléte (is) a kifejezésben.

Láthattunk már egy kifejezést a 2.4.3. részben, ahol megmutattuk, hogy az adatbázis-kezelő rendszer parancsaiban hogyan lehet azt alkalmazni. Amint láttuk, a kifejezés nagyon hatékony bővítési lehetőséget adott a parancshoz. Ha a 11.2. részben felsorolt parancsok listáját végignézzük, kiderül, hogy nagyon sok parancs az alábbi módosított formákban jelenhet meg.

<parancsszó> [FOR <kifejezés>]
<parancsszó> [WHILE <kifejezés>]

Ezek a kibővítések olyan lehetőségeket adnak a felhasználónak, amelyeket más rendszerek esetében nem mindig kapunk meg. Mint a bevezetőben már említettük, tapasztalatok szerint egy adott feladat elvégzésére szolgáló program megírása dBASE II vagy PROP-BASE rendszerben körülbelül tizedannyi időt vesz igénybe, mint más magas szintű nyelveken. De ehhez szükségünk van arra, hogy megismerjük a kifejezéseket, továbbá azt is, hogyan kell összeállítanunk a parancsokat egy parancsállományba, amely azután ugyanazt a műveletet ismételni tudja.

Ha egy parancsban FOR, illetve WHILE paramétert (v.ö.: 18.2. rész) és mellette egy kifejezést adunk meg, a parancs csak azokkal a rekordokkal kapcsolatosan kerül végrehajtásra, amelyeknél a kifejezés logikai értéke igaznak bizonyul.

2.6.1 A FOR paraméter

Ha egy parancsot FOR paramétert követő kifejezéssel adunk meg, akkor az a parancsot — az értelmezési körnek megfelelően — az összes olyan rekordra végrehajtja, amelynél a kifejezés értéke igaznak bizonyul.

Például:

```
... FOR (MunkaSzam=730 .OR. MunkaSzam=731);
        .AND. (Datum>= 791001 .AND.;
               Datum<= 791031)
```

2.6.2 A WHILE paraméter

Ha a parancsot WHILE paramétert követő kifejezéssel adjuk meg, akkor a parancs végrehajtása az aktuális rekordon kezdődik, és az első olyan rekordnál fejeződik be, amelyre a kifejezés értéke hamis. Ha az első rekord nem felel meg a feltételnek, akkor a parancs végrehajtása már ott véget ér. A WHILE paramétert minden olyan parancsban használhatjuk — a LOCATE parancs kivételével —, ahol a FOR paraméter használata megengedett.

Például:

```
... WHILE (MunkaSzam=730 .OR. MunkaSzam=731);
        .AND. (Datum>= 791001 .AND.;
               Datum<= 791031)
```

2.7 A MAKRÓ

Az adatbázis-kezelő rendszer megengedi azt is, hogy parancsból karakter típusú memóriaváltozó értékével adjunk ki parancsot.

Azt a műveletet, amikor egy memóriaváltozó megadásával annak tartalmát kapja meg a rendszer, *makróhívásnak* nevezzük. Ha egy karakter típusú memóriaváltozó előtt közvetlenül & jel

található, akkor az adatbázis-kezelő rendszer a memóriaváltozót — a & jellel együtt — a tartalmával helyettesíti. Ha a & után található karakterek nem adnak változónevet, akkor nincs helyettesítés. Ha pedig a változó nem karakter típusú, hibaüzenetet kapunk. (Lásd még a 2.5. részben, a 31. oldalon.)

A makróhívásokat gyakori összetett kifejezések rövidítésére, egymásba ágyazott parancsállományok közötti értékátadásra, automatikus változóelnevezésre használhatjuk.

2.8 ALAPBEÁLLÍTÁSOK

Lehetőségünk van a rendszer alapbeállításának megváltoztatására is. A SET parancs (231. old., 11.2. rész.) alkalmas arra, hogy megváltoztassuk az üzenetek megjelenését és megjelenési helyét, lassítsuk és ellenőrizzük a program futását, módosítsuk a karaktervizsgálat feltételét vagy a mágneslemezmeghajtó alapértelmezését, bizonyos fokig összekössünk két adatbázist, futás közben állományokat aktivizáljunk, felülírjuk a rendszermemóriában tárolt adatokat.

2.9 PROGRAMOZÁSI SZABÁLYOK

Az adatbázis-kezelő rendszer használata során be kell tartani néhány szabályt:

- a parancsok nem lehetnek 254 karakternél hosszabbak, beleértve az esetleges makrókifejezés tartalmát is,
- a parancsokat — ha szükséges — pontosvessző használatával a képernyő következő sorában folytathatjuk,
- a parancsoknak a 11.2. részben felsorolt parancsszavak valamelyikével kell kezdődniük, ezt követően a paraméterek (NEXT, FOR, WHILE, WITH stb.) tetszőleges sorrendben állhatnak,
- a parancsszavakat rövidítve is megadhatjuk, minimum 4 karakter hosszúságban (például DISP ≡ DISPL ≡ DISPLA ≡ DISPLAY),
- a parancsokban a parancsszavak és a paraméterek elválasztásához tetszőleges számú szóközkaraktert használhatunk, de így sem léphetjük túl a már említett 254 karakteres felső határt,
- a parancsok egymásba ágyazásánál különösen figyelni kell a

  DO WHILE — ENDDO,
  IF — ELSE — ENDIF,
  DO CASE — CASE . . . CASE — OTHERWISE — ENDCASE

  szerkezetekre. Csak teljes szerkezeteket szabad egymásba ágyazni. Az adatbázis-kezelő rendszer nem tudja kezelni az átlapolt szerkezeteket, és ilyenkor programunk kiszámíthatatlanul fog működni.
- DO CASE szerkezeteket egymásba ágyazni tilos!
- Egy időben legfeljebb 16 állomány lehet nyitott, beleértve bármilyen általunk használatba vett állománytípust és a COPY, az INSERT, a PACK, a REPORT, a RESTORE, a SAVE és a SORT parancsok által átmenetileg megnyitott, kiegészítő állományokat. (A SORT két átmeneti állományt nyit!)

2.10 KAPCSOLÓDÁS MÁS RENDSZEREKHEZ

Az adatbázis-kezelő rendszer képes kiolvasni más programok (pl. BASIC, COBOL, FORTRAN, PASCAL, PL/1 stb.) által készített állományokat is, illetve e rendszerek számára állományokat készíthet.

Az APPEND-del olvashatók az olyan általánosan használt állományok, ahol a sorokat a CP/M operációs rendszernek megfelelően egy kocsivissza és egy soremelés zárja, ha az SDF (System Data Format) szabadon választható paramétert is használjuk. Ugyanígy megfordítva, létrehozhatók más rendszerhez tartozó állományok a COPY paranccsal.

A DELIMITED paraméter használatával a különböző rendszerek által megkövetelt határolókat is kezelhetjük, illetve létrehozhatjuk. (Lásd még a 76. oldalon a 7.6 részben!)

2.11 A KÉPERNYŐ HASZNÁLATA

Az adatbázis-kezelő rendszer alapbeállításban sokféle lehetőséget nyújt a szerkesztéshez a képernyőn. Egy teljes rekord megjeleníthető egy időben — kivéve, ha egyszerre nem fér a képernyőre); mozoghatunk előre-hátra (mérettől függetlenül) a rekordban; javíthatjuk, illetve felülírhatjuk a rekord tartalmát. A szerkesztés kódjait a 4.6. rész tárgyalja.

Az alapbeállítás szerinti szerkesztési módban — numerikus adatok bevitele esetén — a számjegyeket 0-tól 9-ig, a tizedespontot, a „—" és „+" jeleket, valamint a szóközkaraktert használhatjuk.

A SET SCREEN OFF paranccsal kikapcsolhatjuk ezt a szerkesztési módot, s akkor a mezők egyenként jelennek meg a képernyő alján. Ilyenkor nincs mód a rekordon belüli hátramozgásra.

Változatos képernyőterveket készíthetünk, ha nem a ? jelű, hanem a @ jelű parancsot használjuk. A @ jelű paranccsal az adatbevitelt is az igényeknek megfelelően szabályozhatjuk.

Az adatbázis-kezelő rendszerben általános vezérlőkaraktereket is használhatunk. Ezeket a 3.1. részben (38. old.) ismertetjük.

MÁSODIK RÉSZ

AZ ADATBÁZIS-KEZELŐ ALKALMAZÁSA

3. A rendszer használatbavétele

A felsorolásban nem szereplő jelölések és fogalmak meghatározása a függelékben található.

3.1 AZ ADATBÁZIS-KEZELŐ RENDSZER MUNKÁBA ÁLLÍTÁSA (DBASE, PROPBASE)

A dBASE II adatbázis-kezelő rendszert az operációs rendszer betöltése és bejelentkezése (pl. az A> jel megjelenése) után a

^DBASE^

szó begépelésével indíthatjuk.

A PROP-BASE adatbázis-kezelő rendszer használatához előbb az ESCBIO60 programot kell elindítani, amely biztosítja a PROP-BASE számára a környezetet a 2.11. részben leírt teljes képernyős szerkesztéshez. Ezután a PROP-BASE névből a kötőjelet elhagyva a

^PROPBASE^

szöveget kell begépelni.

A ^DBASE^, illetve a ^PROPBASE^ kulcsszó megadásának hatására a kért rendszer betöltődik a memóriába, és dátumozási felhívással kezdi meg működését. A dátumot részenként, háromszor két karakterben kell megadni, a számokat törtvonallal (/) kell elválasztani egymástól. Az elválasztáshoz — a PROP-BASE esetében a pontot kivéve — más karaktert is használhatunk. Ha nem kívánunk dátumot megadni, elegendő a <CR> billentyű megnyomása is.

A dBASE II hónap—nap—év, a PROP-BASE év—hónap—nap sorrendű dátumot igényel.

A dátum ezután bekerül minden — az adott futásban — módosítandó adatbázis-állomány mágneslemezen őrzött jellemzői közé, valamint megjelenik a jelentések (REPORT) fejlécében is, hacsak a jelentés kérésekor nem módosítjuk ezt.

A dátum közlése után megjelenik a képernyő első oszlopában egy pont. Ez az adatbázis-kezelő rendszer választ váró üzenete. Ezután adhatjuk meg a parancsokat.

A parancsok nem lehetnek 254 karakternél hosszabbak, és a 11.2. részben felsorolt parancsszavak valamelyikével kell kezdődniük. A parancsszó után következhetnek a parancshoz tartozó paraméterek és a kifejezések.

Az adatbázis-kezelő rendszer parancsvégrehajtás előtt minden sort megvizsgál, és a talált hibát jelzi. A hibaüzenetben megjelenik a teljes parancs, de a zavart okozó parancsrész kezdete fölött kérdőjel látható. A hibaüzenet alatt megjelenik a

MIT JAVIT ? :

üzenet. Ide, a kettőspont mögé kell begépelnünk a parancs hibás részét. A

MIRE JAVIT ? :

kérdésre válaszolva pedig azt a szöveget írjuk, amelyre a hibásnak minősített részt ki kell cserélni. A rendszer ezután megkérdezi, hogy akarunk-e még javítani. Ha itt nemmel (N) válaszolunk, a parancssort újra megvizsgálja, és ha nem hibás, végrehajtja. Ha igennel (I) – a dBASE II esetében Y-nal – válaszolunk, újabb javítást végezhetünk.

Indíthatjuk az adatbázis-kezelő rendszert az operációs rendszer betöltése után

^DBASE <állománynév>^ vagy
^PROPBASE <állománynév>^

üzenettel is. Ilyenkor a rendszer nem kér dátumot, és azonnal az <állománynév>-nek megfelelő parancsállományt hívja meg és hajtja végre.

Az adatbázis-kezelő használata során folyamatosan alkalmazhatók a következő *vezérlőkarakterek.*

<ESC> (escape) lehetővé teszi a felhasználói program megállítását; hatására a rendszer mindig közvetlen módba kerül, és az adatbázis-kezelő rendszer választ váró jele (a pont) látható a képernyőn. Ez a működési mód kikapcsolható a SET ESCAPE OFF paranccsal, s ilyenkor az <ESC> kezelés hatástalan a parancsállomány futásakor.
Az (escape) hatására megszakad a hosszú végrehajtási idejű parancsok (COUNT, DELETE, DISPLAY, INPUT, LIST, LOCATE, RECALL, REPLACE, SKIP, SUM) végrehajtása.

<EL> törli az utoljára beütött karaktert.

<CTRL>–H szintén törli az utoljára beütött karaktert.

<CTRL>–P be-, illetve kikapcsolja a nyomtatóra írást.
(Lásd még a SET PRINT parancsot a 233. oldalon.)

<CTRL>–U törli a pillanatnyi sort.

<CTRL>–X szintén törli a pillanatnyi sort.

MEGJEGYZÉS

A teljes képernyős szerkesztés (SET SCREEN ON) kódjai (lásd a 4.6. részben) az adott funkciókban ugyanezen vezérlőkaraktereknek egy részét más feladat elvégzésére használják.

3.2 ELJÁRÁS LEMEZCSERE ESETÉN (RESET)

Ha többlemezes programot készítünk, vagy egyéb okból lemezt cserélünk, a csere után a CP/M operációs rendszer nem engedi meg az írást új lemezre, amíg egy „melegindítás" nem következik be. A RESET parancsot használjuk arra, hogy helyreállítsuk a kapcsolatot az operációs rendszer és a lemez, illetve az adatbázis-kezelő rendszer között.

A RESET megpróbálja megnyitni mindazokat az állományokat, amelyek a lemezcsere előtt nyitva voltak. Ha egy korábban nyitott állomány egyik aktív lemezen sem található, akkor lezárja. Lemezcsere nélkül kiadott RESET parancsnak nincs semmilyen hatása.

MEGJEGYZÉS

Ha új lemezen van olyan nevű állomány, mint amilyen a csere előtti lemezen volt, akkor azt a RESET parancs nem zárja le. Ez a hiba elkerülhető, ha a lemezcsere előtt állományainkat USE, illetve CANCEL parancsokkal lezárjuk.

3.3 AZ ADATBÁZIS-KEZELŐ JELLEMZŐINEK MEGVÁLTOZTATÁSA ÉS AZ ALAPBEÁLLÍTÁSOK (SET)

Az adatbázis-kezelő rendszer számos olyan parancsot tartalmaz, amelyek segítségével meghatározható, hogyan működjön együtt az adatbázis-kezelő rendszer a számítógéppel. A paramétereket beállíthatjuk a parancsállomány elején (és így is hagyhatjuk), de átállíthatjuk, illetve vissza is állíthatjuk őket. A legtöbb alkalmazás esetében az alapbeállítások kielégítik igényeinket. A paramétereket a parancsállományban vagy közvetlen módban a SET parancs segítségével lehet átállítani. A következőkben leírt parancsokban általában csak az alapbeállítást írjuk le. Ahol különösen fontos, ott mindkét működést ismertetjük. Az alapbeállítást nem minden esetben szükséges megjegyezni, hiszen munka közben kiderül, hogy az adatbázis-kezelő indításakor hogyan állnak ezek a „kapcsolók".

A SET paraméterek a 11.2. részben a 231. oldalon is szerepelnek.

SET ALTERNATE TO <állománynév>

parancs kijelöl egy .TXT állománynév-kiterjesztéssel ellátott állományt annak érdekében, hogy mindazt, ami a képernyőre íródik — kivéve a @ jelű parancs eredményeit —, megőrizhesse.

Mindaddig nem kezdődik meg a képernyő tartalmának másolása a kijelölt állományba, amíg a ^SET ALTERNATE ON^ parancsot ki nem adjuk.

Ha a lemezre írást meg akarjuk szüntetni, ^SET ALTERNATE OFF^ parancsot kell adnunk. Újabb ^SET ALTERNATE ON^ parancs hatására a beírás folytatódik.

SET BELL

Hangjelzéssel figyelmeztet, amikor egy mező megtelik, vagy a felhasználó illegális adatbevitelre tesz kísérletet.

SET CALL TO <cím>

Számítógép-memóriacímet adhatunk meg vele, amelyet a CALL parancs aktivizál.

SET CARRY

Az APPEND módban egy üres rekordot mutat. Átállítás után az APPEND parancs teljes képernyős szerkesztési módban a megelőző rekordból átmásolja az adatokat az új rekordba.

SET COLON

Kettőspontot használ a képernyőn a bemenő információk elhatárolására.

SET CONFIRM

Ha az adatbevitel során egy – változóhoz rendelt – képernyőterület megtelik, azt elhagyja a fénypont. Ha átkapcsoljuk az alapbeállítást, akkor egy <enter> vagy <CR> kódot kell adni, mielőtt ezt a területet elhagyná a fénypont.

SET CONSOLE

A kimenő információkat a képernyőre írja.

SET DATE TO xx/xx/xx

Az adatbázis-kezelő dátumot őrző rendszerváltozójának beállítására szolgál.

SET DEBUG

Az ECHO és a STEP paraméterű SET parancsok kimenő információit képernyőre, átállítás után nyomtatóra írja a rendszer.

SET DEFAULT TO <lemezegység>

Kijelöli az „alap" lemezegységet.

SET ECHO

A parancsállomány parancsai végrehajtásukkor nem jelennek meg a kimeneti egységen.

SET EJECT

A REPORT parancs esetében minden új jelentés elején lapemelést hajt végre.

SET ESCAPE

Lehetővé teszi, hogy az <escape> (<ESC>) nyomógombbal a parancsállomány végrehajtását megszakítsuk.

SET EXACT

Megenged különböző hosszúságú karakterláncokat, például "ABCD"="AB" igaz lehet. (A FIND parancsot is befolyásolja.) Átállítás után megköveteli, hogy az összehasonlításban a két karakterlánc karakterről karakterre megegyezzen.

SET FORMAT TO SCREEN

A @ jelű parancs kimenő információit a képernyőre írja.

SET FORMAT TO PRINT

A @ jelű parancs kimenő információit a nyomtatóra írja.

SET FORMAT TO <formátumállomány neve>

Formátumállományt adhatunk meg vele, amit a READ parancs aktivizál.

SET HEADING TO <karakterlánc>

REPORT parancshoz második fejrészt adhatunk meg vele, maximum 60 karakter hosszban.

SET INTENSITY

A szerkesztési módban lehetővé teszi — ha van ilyen — a kettős intenzitást, azaz fokozott fényerővel dolgozhatunk.

SET LINKAGE

Átállítás után lehetővé teszi, hogy az adatbázisokat a megjelenítés érdekében (maximum 64 mezőig *és* kijelzett rekordonként 2000 byte-ig) összekössük. A P. és S. bevezető karaktereket a mezőnevek esetében akkor kell használni, ha azok mindkét adatbázisban ugyanazok. (Lásd a 80. oldalon a 7.8. részt és a 228. oldalon a SELECT parancsot.)

SET MARGIN TO <szám>

A nyomtatón a bal oldali margót állítja be (a megadható szám <= 254).

SET PRINT

Átállítás után kijelez minden kimenő információt (kimenetet) a listázó egységen (pl. nyomtatón).

SET RAW

A DISPLAY és a LIST parancsok végrehajtásakor a rekordokat úgy írja ki, hogy a mezők között üres helyet hagy.

SET SCREEN

Az APPEND, az EDIT, az INSERT és a CREATE parancsok számára szabad mozgást ad a teljes képernyőn.

SET STEP

Átállítás után a rendszer minden parancs végrehajtása után megáll annak érdekében, hogy a parancsállományt tesztelni lehessen (hibakeresés).

SET TALK

Minden egyes parancs gépi végrehajtásának eredménye, illetve visszajelzése megjelenik a képernyőn.

3.4 ÁLLOMÁNYOK ÁTNEVEZÉSE (RENAME)

Néha szükségünk lehet arra is, hogy állományainkat átnevezzük. Ez a probléma megoldható a RENAME parancs segítségével. Az átnevezés a CP/M katalógusában ilyenkor annak rendje és módja szerint megtörténik.

A parancsban először a régi nevet adjuk meg, majd az újat. Megnyitott állományt nem szabad átnevezni.

Az adatbázis-kezelő rendszer az állománynév-kiterjesztés elhagyásakor feltételezi, hogy adatbázis-állományról van szó, és a .DBF kiterjesztést rendeli az állományhoz. Például, ha a <Raktar.DBF> adatbázis-állományt <Kisrakt.DBF> névre akarjuk változtatni, a

^RENAME Raktar TO Kisrakt^

parancsot kell kiadnunk. Más esetekben az állománynév-kiterjesztést is meg kell adnunk. Például:

^RENAME Menu.BAK TO Menu.CMD^.

3.5 KILÉPÉS AZ ADATBÁZIS-KEZELŐ RENDSZERBŐL (QUIT)

Ha befejeztük a munkát, el kell hagynunk az adatbázis-kezelő rendszert. A ^QUIT^ parancscsal szabályosan járunk el: a QUIT ugyanis lezárja az összes adatbázis- és parancsállományt, majd visszaadja a vezérlést az operációs rendszernek. (Nem tanácsos az adatbázis-kezelő elhagyása más módon: például a reset billentyű használata vagy a lemez kivétele.)

A parancs a TO paraméterrel kibővíthető úgy, hogy a kilépés után CP/M rendszerbeli parancso(ka)t vagy közvetlenül hívható állomány(oka)t állítsunk munkába. Az így végrehajtandó programok és CP/M parancsok száma nem korlátozott, csupán az adatbázis-kezelő rendszer parancsaira vonatkozó 254 karakteres felső határt kell betartanunk. Az állományneveket, illetve a parancsokat aposztrófok közé kell tenni, és vesszővel kell elválasztani egymástól. Például a ^QUIT TO '<állománynév vagy rendszerparancs>'[,'<lista>']^ parancs lehetővé teszi, hogy az adatbázis-kezelő rendszerben végzett munkát úgy fejezzük be, hogy automatikusan elkezdjük a CP/M parancsok vagy a CP/M rendszerben hívható állományok végrehajtását.

Meghatározhatjuk például, hogy a kilépés után TM jelű szövegszerkesztőnkkel egy jelentést akarunk írni, majd VE jelű editorunkkal módosítani akarjuk a Menu.CMD parancsállományt. Ezután az operációs rendszer STAT parancsával először megnézzük, mekkora helyet foglal el a két állomány, majd megnézzük, mennyi hely van a B lemezen. Átmásoljuk az állományokat, visszatérünk az adatbázis-kezelőhöz, és végrehajtatjuk a Menu.CMD nevű parancsállományt. Mindezt egy parancsban megadhatjuk:

```
^QUIT TO 'tm jel.szf','ve menu.cmd','stat a:*.*','stat b:',;
'pip b:=a:jel.szf','pip b:=a:menu.cmd','dbase menu'^
```

vagy

```
^QUIT TO 'tm jel.szf','ve menu.cmd','stat a:*.*','stat b:',;
'pip b:=a:jel.szf','pip b:=a:menu.cmd','propbase menu'^
```

A fenti példa azt is szemlélteti, hogyan folytathatjuk parancsainkat a képernyő következő sorában a pontosvessző használatával. „Kocsivissza" kód előtt álló pontosvessző esetén ugyanis az adatbázis-kezelő rendszer a következő sort az előző parancs folytatásának tekinti.

4. Adatbázisok létrehozása, bővítése, bevezetés a parancsok alkalmazásába

4.1 ADATBÁZIS LÉTREHOZÁSA (CREATE)

A könnyebb megértés és a későbbi kipróbálás érdekében előállítunk egy egyszerű adatbázist nevek és címek számára. Az adatbázisunkban minden rekord a következő információkat fogja tartalmazni:

NÉV	14 karakterhosszban
CÍM	19 karakterhosszban
HELYSÉG	14 karakterhosszban
IRÁNYÍTÓSZÁM	4 karakterhosszban

Először írjuk be, hogy

^CREATE^

Az adatbázis-kezelő rendszer válaszolni fog, és kéri az állomány nevét.

Adjunk meg egy betűvel kezdődő, maximum 8 karakter hosszú állománynevet, kettőspont és szóköz nélkül. Mivel ez egy olyan állomány, amely nevek szerint épül fel, példánkban a ^NEVEK^ állománynevet adjuk neki.

Megtehetjük azt is, hogy a CREATE paranccsal együtt mindjárt megadjuk az állomány nevét is:

^CREATE Nevek^

Amikor a carriage return (kocsivissza) gombot leütöttük, az adatbázis-kezelő rendszer olyan állományt fog előállítani, amelynek a neve <Nevek.DBF>. A név utáni rész, amely a pontot követi, az állománynév CP/M kiterjesztése, a „database file" (adatbázis-állomány) szavak rövidítése.

Az adatbázis-kezelő rendszerben minden egyes részadatnak, amelyet a rendszerbe akarunk helyezni, *mező* a neve. Ezeknek a mezőknek a csoportosítását hívjuk *rekordnak.* Példánkban minden egyes rekord négy mezőt fog tartalmazni. Az adatbázis-kezelőnek ismernie kell minden egyes mező nevét. Azt is tudnia kell, hogy milyen típusú adatokat fognak tartalmazni az egyes mezők, milyen hosszúak, és ha az adatok numerikusak, hány tizedesjegy (decimális érték) van bennük.

Például:

```
. create
ENTER FILENAME: nevek
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD  NAME,TIPE,WIDTH,DECIMAL PLACES
001
. create
ALLOMANY NEVE : nevek
REKORD SZERKEZET AZ ALÁBBI SZERINT:
 MEZŐ   NEVE,TIPUSA,HOSSZA,TIZEDES HELYEK
001
```

A mezőnevek maximum 10 karakter hosszúak lehetnek, nagy- és kisbetűs változatban is írhatók. A mezőnevet betűvel kell kezdeni, és nem lehet benne üres hely, de tartalmazhat számjegyeket és kettőspontokat. Ne rövidítsünk többet, mint amennyi feltétlenül szükséges. A számítógép értené, de mások számára a túlzott rövidítéssel érthetetlen lenne az információ.

Az adat típusának meghatározásához egyetlen betűt használunk („C" karaktert jelent, „N" numerikust, „L" pedig logikai változót). Esetünkben minden mező karakter típusú adatokat tartalmaz. A mező 254 karakter hosszúságig terjedhet. Ha a mező numerikus típusú, és decimális helyeket is megadtunk, vigyázzunk arra, hogy a tizedespont is egy karakterpozíciónak felel meg, egy karakter helyet foglal.

Példánkban tudjuk azt, hogy milyen neveket akarunk adni az egyes mezőknek. Az adatok típusát is ismerjük — amelyeket a mezők tartalmazni fognak —, ismerjük a hosszúságukat is. Adjuk meg tehát a számítógépnek az információt!

Például:

```
. create nevek
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD  NAME,TIPE,WIDTH,DECIMAL PLACES
001     nev,c,14
002     cim,c,19      REKORD SZERKEZET AZ ALÁBBI SZERINT:
003     helyseg,c,14 MEZŐ   NEVE,TIPUSA,HOSSZA,TIZEDES HELYEK
004     ir szam,c,4
BAD NAME FIELD                                  HIBÁS MEZŐ NÉV
004     ir:szam,c,4
005     <cr>
```

A fenti példa bemutatja az adatstruktúra-képzés folyamatát. Figyeljük meg, hogy mi történt a 4. mezőnél! Hibát követtünk el, amikor egy üres közt írtunk be a mezőnévbe. Az adatbázis-kezelő rendszer közli, milyen típusú hibát követtünk el, és lehetőséget ad a javításra. Figyeljük meg azt is, hogy az irányítószám is karakter típusú. Nem volt szükségünk arra, hogy numerikusként adjuk meg, mert számolni nem akarunk vele. Így is tudjuk a kisebb, mint (<); nagyobb, mint (>); egyenlő (=); nem egyenlő <> összehasonlító műveleteket (relációkat) használni. Karakterláncok összehasonlíthatók az elemek ABC-ben elfoglalt helye szerint.

Amikor az adatbázis-kezelő rendszer azt kéri, hogy az 5. mező specifikációit is adjuk meg, egy <CR>-rel válaszolva jelezzük, hogy az adatok definiálását befejeztük. Az adatbázis-kezelő

rendszer megőrzi az adatstruktúrát, majd megkérdezi, hogy akarunk-e a struktúrába adatokat bevinni.

A <Nevek.DBF> adatbázis azonnal kész arra, hogy adatokat fogadjon, így válaszoljunk a gép kérdésére I-vel (PROP-BASE) vagy Y-nal (dBASE II).

4.2 AZ ADATBÁZIS FELTÖLTÉSE (USE)

A kérdésre adott válasz után a képernyő törlődik, és bal felső sarkában a RECORD feliratot és egy # jelet követően megjelenik a rekord száma, alatta pedig nevükkel együtt következnek a mezők. A fénypont az első rekord első mezejének első karakterpozícióján áll.

Példa:

```
RECORD#00001
NEV      :                  :
CIM      :                     :
HELYSEG  :                  :
IR:SZAM  :     :
```

A mezők hosszúságának jelzésére két kettőspont szolgál. Amikor egy mezőt teljesen kitöltöttünk, vagy kitöltés közben <*CR*>-t billentyűzünk, a fénypont eggyel lejjebb ugrik a következő mezőre. A fénypont visszamozgatható az előző mezőre, ^*E (ENQ)* kóddal. Amikor az utolsó mezőt is kitöltöttük, az adatbázis-kezelő a következő üres rekordot fogja a képernyőre hozni.

Példánkban a <Nevek.DBF> adatbázist a következő adatokkal töltjük meg:

```
ASZTALOS ANTAL  KACSA U. 16        BUDAPEST       1027
BOKA ISTVAN     FENYVES UT 32      BUDAPEST       1028
CINGLER MONIKA  PESTI U 8          BUDAKESZI      2020
DEAK GABOR      DEAK FERENC U 1    HATVAN         6060
ELEK FERENC     DOHANY U. 19/b     ZALAKAROS      9595
FEHER ELEK      MENTE TER 3        ELEKES         4346
GONDA ANDRAS    JOZSEF KRT. 46     BUDAPEST       1077
HEGEDUS PAL     BETLEHEMI U. 123   MARTONVASAR    2454
```

Ha olyan hibát követünk el, amely nem korrigálható visszalépéssel és felülírással, a következő fejezetben leírt módon kell azt kijavítani. Ha az adatbázis-kezelő rendszer alapállapotát jelző pont jelenne meg, akkor a következőt kell leírnunk:

. use Nevek
. append

és folytatni kell az adatbevitelt.

Az adatok bevitelét úgy kell lezárni, hogy a <CR> billentyűt nyomjuk meg, ha az irányítószámot is megadtuk már, és a következő rekord első mezejének első karakterén álluńk. Ha valamilyen adatot már begépeltünk, vagy a fénypontot tovább mozgattuk, akkor ^*Q-t (DC1-et)* (lásd az 50. oldalon a 4.5, ill. az 51. oldalon a 4.6. részt) kell betáplálni. Az adatbázis-kezelő rendszer elhagyja az adatbeviteli módot, és bejelentkezik a szokásos pont képével — mutatván

azt, hogy kész a parancsaink fogadására. Ha itt meg akarunk állni és nem akarjuk folytatni a munkát, QUIT paranccsal elhagyhatjuk az adatbázis-kezelő rendszert. A ^QUIT^-t kell megadni minden olyan esetben, amikor az adatbázis-kezelővel folytatott műveletet be akarjuk fejezni. Ez automatikusan lezárja minden állományunkat; ha ezt nem tesszük, előfordulhat, hogy megrongáljuk adatbázisunkat.

4.3 ÚJABB REKORDOK HOZZÁADÁSA AZ ADATBÁZISHOZ (APPEND, INSERT)

Bármilyen adatbázishoz gyorsan és egyszerűen adhatunk hozzá újabb rekordokat egyszavas paranccsal. Először a kívánt adatbázist ^USE <állománynév>^ paranccsal kell kiválasztani, azután az ^APPEND^ parancsot kell begépelni.

Példa

```
. use Nevek
. append

RECORD#00009

NEV       :                :
CIM       :                        :
HELYSEG   :                :
IR:SZAM   :     :
```

Az adatbázis-kezelő válaszként a képernyőre írja azt a rekordszámot, amely az állomány utolsó rekordja után következik. A képernyőn megjelennek a mezők nevei, kettőspont mutatja a mezők terjedelmét. A fénypont az első pozíción áll, ahol az adatbevitel kezdhető. A rekord kitöltése után hozzáadódik (hozzáfűződik) az állomány végéhez. Ha egy mezőt kitöltöttünk adatokkal, a fénypont automatikusan lejjebb mozog a következő mezőre. A <CR> lenyomásával is tovább lehet haladni. Ha egy mezőbe nem akarunk adatokat beírni, egyszerűen a <CR> gombot kell használni. Ilyenkor a karakter típusú mezők automatikusan üres helyekkel töltődnek fel, a numerikus mezők zérus, a logikai mezők hamis (fals) értéket fognak mutatni. Nem szükséges a tizedespont leütése, ha nincsenek számjegyek a tizedespont után, az adatbázis-kezelő rendszer automatikusan elhelyezi a tizedespontot és a szükséges mennyiségű, tizedespontot követő nulla számjegyet.

Rekordokat szúrhatunk be (inzertálhatunk) egy meghatározott helyre az adatbázisba (például annak érdekében, hogy az alfabetikus sorrendet megtartsuk) a következő utasítás segítségével:

^INSERT [BEFORE] [BLANK]^

Ha csak az ^INSERT^ parancsszót használjuk önmagában, a rekord az aktuális rekord utáni helyre kerül.

Ha megadjuk a ^BEFORE^ paramétert is, a rekord az aktuális rekord elé kerül. A képernyőn mindkét esetben ugyanaz — egy kitöltetlen rekord — jelenik meg. Ugyanolyan módon kell el-

járni, mint az ^APPEND^ és a ^CREATE^ parancsok esetében. Ha a parancsban a ^BLANK^ paramétert is megadtuk, egy üres rekord kerül beszúrásra, és ebben az esetben a képernyőn nincs kitöltendő információ.

Adjuk meg a következő neveket alfabetikus sorrendben a <Nevek.DBF> adatbázisunkhoz

```
EDENYESI GABOR  EDENY U. 5        EDELENY        6576
INKE LASZLO     KOMAROMI UT 45    GYOR           2676
JENEI PAL       PETOFI TER 13     KOSPALLAG      3556
```

A parancsok sorrendje a következő:

. **use nevek**
. **go 5**
. **insert before** **(adatok megadása az első névhez)**
. **append** **(adatok megadása a másik két névhez)**

Ha ^INSERT^ módban az utolsó mezőt is kitöltöttük, az adatbázis-kezelő rendszer visszatér a parancsmódhoz. (A jellegzetes pont jelenik meg.)

Az ^APPEND^ mód elhagyásához a fénypontot egy új mező kezdetére kell állítani, majd *<CR>* vagy *^Q (DC1)* kulcsot (lásd a 4.5., ill. 4.6. részt) kell megadni.

Bármelyik adatbeviteli módban vagyunk, az adatbázis-kezelő rendszer elhagyhatja ezt, ha *^W*-t *(ETB)* adunk (lásd 4.6. rész).

4.4 REKORDOK TÖRLÉSE AZ ADATBÁZISBÓL, AZ ADATBÁZIS MEGTISZTÍTÁSA (DELETE, RECALL, PACK)

Az adatbázis-kezelő rendszerben az ^EDIT^ szerkesztési módon kívül, melyet a 4.5. részben ismertetünk, közvetlenül is törölhetünk rekordokat.

Az aktuális rekord törlésére a ^DELETE^ parancs szolgál. Ha egynél több rekordot kívánunk törölni, a következő parancsalakzatot kell használni:

^DELETE <érvényességi kör>^, ahol az <érvényességi kör> ugyanaz, mint más adatbázis-kezelő parancsok esetében: *All, Record n* vagy *Next n* közül valamelyik.

A törlés feltételessé tételéhez a következő parancsot kell használni

^DELETE [<érvényességi kör>] [FOR <kifejezés>]^

vagy

^DELETE [<érvényességi kör>] [WHILE <kifejezés>]^

ahol a <kifejezés> vagy egy feltétel, vagy pedig feltételek együttese, amelyeknek teljesülniük kell ahhoz, hogy a parancs végrehajtódjék. (Bővebb kifejtését lásd a 2.6. részben, a 31. oldalon!)

Annak érdekében, hogy egy egész állományt töröljünk, a következő parancsot kell megadni:

^DELETE FILE [<lemezegység:>] <állománynév>^

Legyünk óvatosak, mert ha ezt a parancsot adjuk ki, a nevezett állományban levő adataink azonnal törlődnek!!

Az állományokkal ellentétben a rekordok, amelyeket törlésre jelölünk ki, visszaállíthatók. A ^DELETE^ parancs ugyanis ahelyett, hogy törölné az adatokat, csak megjelöli azokat egy csillaggal. Ez a csillag látható is, amikor ^LIST^ vagy ^DISPLAY^ paranccsal képernyőre írjuk rekordjainkat. Az adatbázis-kezelő rendszer az így megjelölt rekordokat mellőzi, és eljárásaiban, műveleteiben nem használja. Ezeket a rekordokat újra használhatjuk, ha törlési jelölésüket megszüntetjük a következő paranccsal:

^RECALL [<érvényességi kör>] [FOR <kifejezés>]^

vagy

^RECALL [<érvényességi kör>] [WHILE <kifejezés>]^

Ugyanúgy dolgozik ez a parancs is, mint a ^DELETE^ az <érvényességi körrel> és a szabadon választható feltétellel. Ha ilyenkor feltételes kifejezést használunk, nem kötelező ugyanolyannak lennie, mint a DELETE esetében, amikor a rekordokat törlésre jelöltük meg.

Bizonyos esetekben szükségünk van arra, hogy az adatbázisunkat megtisztítsuk a szükségtelen rekordoktól, hogy helyet teremthessünk további rekordok beléptetéséhez. Ennek érdekében gépeljük be:

^PACK^

A parancs végrehajtása alkalmával a törlésre jelölt rekordok ténylegesen megszűnnek, és a művelet végén megkapjuk, hogy hány rekord van az adatbázisban.

Vigyáznunk kell e parancs használatával, mert az így törölt rekordok véglegesen elvesznek.

A következő példában bemutatjuk, hogy a már létező <Nevek.DBF> adatbázisban milyen parancsokkal törölhetünk rekordokat. Először kilistázzuk a NEVEK adatbázist, majd kijelöljük törlésre a 2. és 4. rekordot. Listázással meggyőződünk a jelölések érvényéről, majd megszüntetjük a 4. rekord törlésre jelölését. Újabb listázás után töröljük a 2. rekordot.

```
. use nevek
. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST        1027
00002 BOKA ISTVAN     FENYVES UT 32       BUDAPEST        1028
00003 CINGLER MONIKA  PESTI U 8           BUDAKESZI       2020
00004 DEAK GABOR      DEAK FERENC U 1     HATVAN          6060
00005 ELEK FERENC     DOHANY U. 19/b      ZALAKAROS       9595

. delete record 2
00001 DELETION(S)                                  00001 TÖRLÉS

. delete record 4
00001 DELETION(S)                                  00001 TÖRLÉS
```

```
. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST        1027
00002*BOKA ISTVAN    FENYVES UT 32        BUDAPEST        1028
00003 CINGLER MONIKA PESTI U 8            BUDAKESZI       2020
00004*DEAK GABOR     DEAK FERENC U 1      HATVAN          6060
00005 ELEK FERENC    DOHANY U. 19/b       ZALAKAROS       9595

. recall record 4
00001 RECALL(S)                                 00001 VISSZAÁLLITÁS

. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST        1027
00002*BOKA ISTVAN    FENYVES UT 32        BUDAPEST        1028
00003 CINGLER MONIKA PESTI U 8            BUDAKESZI       2020
00004 DEAK GABOR     DEAK FERENC U 1      HATVAN          6060
00005 ELEK FERENC    DOHANY U. 19/b       ZALAKAROS       9595

. pack
PACK COMPLETE 00004 RECORDS COPIED
                                PACK VÉGE 00004 REKORD MÁSOLVA
. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST        1027
00002 CINGLER MONIKA PESTI U 8            BUDAKESZI       2020
00003 DEAK GABOR     DEAK FERENC U 1      HATVAN          6060
00004 FLEK FERENC    DOHANY U. 19/b       ZALAKAROS       9595
```

4.5 A REKORDOKBAN LEVŐ ADATOK JAVÍTÁSA (EDIT, BROWSE)

Ha az adatbevitel közben bármilyen hibát követtünk el, ezt a teljes képernyős szerkesztési módban gyorsan és könnyen kijavíthatjuk. Írjuk le, hogy

^USE Nevek^
^EDIT <szám>^

ahol a „szám" az adatbázis valamelyik rekordjának száma. Az adatbázis-kezelő rendszer a teljes rekordot behozza, és a teljes képernyős szerkesztési mód parancsaival a rekordban levő bármelyik adatot módosíthatjuk. A *^C (ETX)* kód megadásával a következő rekordra mehetünk. A *^R (DC2)* kóddal visszaléphetünk az előző rekordra.

Példa

```
. use nevek
. edit 3

RECORD#00003                                    DELETED
NEV       :CINGLER MONIKA :
CIM       :PESTI U. 8           :
HELYSEG   :BUDAKESZI        :
IR:SZAM   :2020:


. use nevek
. edit 3

REKORD#00003                                    TÖRÖLVE
NEV       :CINGLER MONIKA :
CIM       :PESTI U. 8           :
HELYSEG   :BUDAKESZI        :
IR:SZAM   :2020:
```

Ha a ^*U (NAK)* segítségével egy rekordot törlésre jelölünk, a képernyő tetején „DELETED" felirat jelenik meg. A ^*U (NAK)* ismételt lenyomására a felirat eltűnik, és a rekord törlési jelölése megszűnik. Ha majd a ^LIST^ vagy a ^DISPLAY^ parancsokat használjuk, minden egyes törlésre kijelölt rekord mellett egy csillagot (*) láthatunk.

Ha az EDIT szerkesztési módot a ^*Q (DC1)* lenyomásával hagyjuk el, a változtatások nem kerülnek a mágneslemezre; ^*W (ETB)* használatával az összes változtatások megőrzésével fejeződik be a szerkesztés.

Ha sok rekordon ugyanazt a mezőt szeretnénk módosítani, vagy egyéb okból szeretnénk adatbázisunkba egy „ablakon" betekinteni, használjuk a

^BROWSE^

parancsot. Ennek hatására egyszerre 19 rekord adattartalma jelenhet meg a képernyőn, illetve valamivel kevesebb, ha egy adatmező 80 karakternél hosszabb. A rekordok a képernyőn soronként, egymás alatt helyezkednek el. Minden sorban annyi adatmező látható, amennyi csak elfér. A rekordokon előre-hátra (azaz a képernyőn fel-le), a mezőkön (és egyúttal a képernyőn) jobbra-balra lehet mozogni.

Ezért hasonlíthatjuk tulajdonképpen a képernyőt egy ablakhoz, amely az adatbázisra nyílik, abból egy részt megmutat, és bármerre „mozgatható". Így ezen az „ablakon" keresztül adatbázisunk bármely részét megtekinthetjük, és a szokásos szerkesztő eljárásokkal módosíthatjuk.

4.6 A SZERKESZTÉS KÓDJAI

Az adatbázis-kezelő rendszer alapbeállításában (SET SCREEN ON) szabadon mozoghatunk a teljes képernyős szerkesztés közben a megfelelő kódok használatával. (SET SCREEN OFF paranccsal kikapcsolhatjuk ezt a szerkesztési módot.) A kódok egy-egy funkciót jelölnek. A funkciókat aszerint csoportosítottuk, hogy mely parancsnál mire használhatók a szerkesztés kódjai. Az általános funkciók mindig aktívak.

Általános funkciók

^X (CAN) a fénypontot lefelé mozgatja a következő mezőre.
Alkalmazható erre a célra a ^F (ACK) is.

^E (ENQ) a fénypontot visszafelé mozgatja ez előző mezőre.
Alkalmazható erre a célra a ^A (SOH) is.

^D (EOT) a fénypontot előre mozgatja a következő karakterre.

^S (DC3) egy karakternyit visszatolja a fénypontot.

^V (SYN) átkapcsoló a felülírással, illetve a beszúrással való adatbevitel számára.
Alapállapota: felülírással megvalósított adatbevitel.

^G (BEL) azt a karaktert törli, amelyen a fénypont áll.

<Rubout> (DEL) a fényponttól balra álló karaktert törli.

^Y (EM) törli az aktuális mező tartalmát.

^Q (DC1) elhagyja a szerkesztési módot, és visszatér alapállapotba anélkül, hogy a mágneslemezre írná a szerkesztés során végzett változtatást.

^W (ETB) a szerkesztésben végzett minden változtatást figyelembe vesz, és visszatér alapállapotba.
Alkalmazható erre a célra a ^O (SI) is.

APPEND funkciók

^R (DC2) a rekordot a mágneslemezre írja, és a következő rekordra áll.
Alkalmazható erre a célra a ^C (ETX) is.

<ENTER> vagy <CR>. Ha akkor adjuk meg ezt a kódot, amikor a fénypont egy új rekord kezdőpozícióján áll, a rendszer alapállapotba tér vissza.

^Q (DC1) törli a rekordot, és visszatér a normál műveletvégzéshez.

BROWSE funkciók

^X (CAN) a következő adatmezőre lép.
Alkalmazható erre a célra a ^F (ACK) is.

^E (ENQ) visszalép az előző adatmezőre.
Alkalmazható erre a célra a ^A (SOH) is.

^B (STX) a mezőket jobbra mozgatja.

^Z (SUB) a mezőket balra mozgatja.

^C (ETX) a következő rekordra lép.

^R (DC2) visszalép egy rekordot.

^U (NAK) az aktuális rekord törlésre való megjelölése, illetve a jelölés megszüntetése.

EDIT funkciók

(Ne használjuk ezeket az APPEND módban!)

^C (ETX) a rekordot mágneslemezre írja, és a következő rekordra áll.

^R (DC2) mágneslemezre írja a rekordot, és visszamegy az előző rekordra.

^U (NAK) átkapcsoló a rekord törlésre való megjelölése, és ennek ellenkezője számára.

^Q (DC1) nem veszi figyelembe a szerkesztésben az aktuális rekordon végzett változtatásokat, és visszatér alapállapotba.

GET funkciók

^X (CAN) a következő adatra lép.
Alkalmazható erre a célra a ^F (ACK) is.

^E (ENQ) visszalép az előző adatra.
Alkalmazható erre a célra a ^A (SOH) is.

MODIFY funkciók

^T (DC4) törli azt a sort, ahol a fénypont van, és az alatta levő sorokat eggyel feljebb mozgatja.

^Y (EM) törli az aktuális sor tartalmát, és üres helyekkel tölti fel.

^N (SO) a sorokat a fénypont helyétől kezdve egy pozícióval lejjebb mozgatja, és a sorok közé elhelyez egy új, üres sort ott, ahol a fénypont áll.

^C (ETX) lefelé mozog a sorokon.

^R (DC2) felfelé mozog a sorokon.

MEGJEGYZÉS

Ebben a szerkesztési módban a numerikus adatok bevitelénél a 0-tól 9-ig terjedő számjegyeket, a tizedespontot, a „–" és a „+" jeleket és a szóközt használhatjuk. A szerkesztés kódjaiként használt vezérlőkarakterek egy része – a felsorolt funkciókon kívüli üzemmódban – más feladatokat lát el (v.ö. a 38. oldalon a 3.1. résszel).

5. A változók és használatuk

Ha egy karaktert vagy karakterek együttesét úgy kezeljük, mint *karakterállandót,* aposztrófok, idézőjelek vagy szögletes zárójelek közé kell zárnunk. Így érti meg a számítógép, hogy ezzel a karaktersorral úgy kell bánnia, mint karakterekkel. Lássuk, hogy mit jelent ez!

```
. use nevek
. ? 'nev'
. ? nev
```

A ? jelű parancs jelentése: „Mi az, hogy . . .?" vagy „Mennyi az . . . értéke?". Az első „mi az, hogy 'nev' ?" kérdésre a számítógép válasza nem lesz más, mint az, hogy NEV, mert ez a ? jelű parancsban karakterhatárolókkal jelölt karakterállandó. De ha elhagyjuk az aposztrófot, az adatbázis-kezelő rendszer egy, a parancsban szereplő paraméternek vagy változónévnek tekinti a karakterláncot. Ebben az esetben ez a karakterlánc nem paraméter, hanem változó.

A *változók* értéke módosítható.

5.1 ADATBÁZISMEZŐK HASZNÁLATA (REPLACE)

A változók többsége adatbázismező. Esetünkben a NEVEK adatbázis tartalmaz egy <Nev> nevű mezőt, ezért az adatbázis-kezelő rendszer azt az értéket jelzi a képernyőn, amit az aktuális <Nev> mező tartalmaz. (Az alábbi példához használjuk a 4.2. részben a 45. oldalon található listát.)

Példa

```
. use nevek
. ? 'nev'
NEV
. ? nev
ASZTALOS ANTAL
```

Most gépeljük be azt, hogy ^USE^. Mivel nem adtunk meg állománynevet, a számítógép lezárja minden állományunkat. Ha most ismét azt gépeljük be, hogy ^? nev^, válaszként kapjuk, hogy hibát követtünk el. Igen, mert megpróbáltunk egy olyan változót használni, amely nem létezett. (Hiszen ekkor már nem volt használatban olyan állomány, amelyben ilyen mezőnév létezett volna.)

Mint mondottuk, a változók értéke módosítható. Az adatbázismezők esetében erre a REPLACE parancsot és a WITH paramétert használhatjuk. Ennek formája a következő.

```
. use nevek
. goto 2
. ? nev
BOKA ISTVAN

. replace nev with 'BERENDI HENRIK'
00001 REPLACEMENT(S)                              00001 HELYETTESITES

. ? nev
BERENDI HENRIK
```

Figyeljük meg, hogy a nevet (mivel karakterlánc) aposztrófok közé zártuk. A REPLACE paranccsal egyidejűleg maximum öt mező értékét változtathatjuk meg a következő módon.

```
^REPLACE nev WITH 'KOVACS', cim WITH 'ALVEGI UT 45.' ,helyseg;
WITH 'LEPENCE'^
```

Mint láthatjuk, az egyes mezőváltoztatások közé vesszőt kell tenni. (A parancsot pontosvesszővel lehet folytatni a következő sorban.)

5.2 MEMÓRIAVÁLTOZÓK HASZNÁLATA

Változók a *memóriaváltozók* is (a mezőnevek mellett).

5.2.1 Memóriaváltozók létrehozása (STORE)

Ahhoz, hogy egy memóiaváltozót létrehozzunk és abba — vagy egy már létező memóriaváltozóba — karakter, numerikus vagy logikai típusú adatot juttassunk, a ^STORE^ parancsot kell használnunk. A teljes formátum a következő:

^STORE <kifejezés> TO <memóriaváltozó> [,<lista>]^

A listában legfeljebb 5 memóriaváltozót sorolhatunk fel, melyek ugyanazt az értéket kapják.

Példák

```
. store "HOGY ERZI MAGAT '86-BAN ?" to kerdes
HOGY ERZI MAGAT '86-BAN ?

. store 10 to szorzo
10
```

```
 . store 17.35 to szorzando
17.35

 . ? szorzo*szorzando
       173.50

 . ? kerdes
HOGY ERZI MAGAT '86-BAN ?
```

Az első sorban megadott karakterláncot kettős idézőjellel határoltuk, mert a szövegen belül szükségünk volt az aposztrófra, és a számítógép számára egy másik karakterlánchatárolót kellett választanunk.

Most kezdjük az alábbi parancsok begépelésével a munkát!

```
 . store 99 to nagy
99

 . store 33 to kicsi
33

 . store nagy/kicsi to eredmeny
3

 . store '99' to forma
99

 . ? nagy/kicsi
         3

 . ? nagy/3
        33

 . ? forma/3
***SYNTAX ERROR***                          ***FORMAI HIBA***
 ?
? FORMA/3
```

Hibáztunk, amikor '99'-et akartunk 3-mal osztani. Amikor egy változóba adatot juttatunk, az adatbázis-kezelő rendszer azonnal felismeri, hogy milyen típusú az adat. Ezután az adat már csak a típusának megfelelően szerepelhet. Ne próbáljunk meg például egy karakterláncot aritmetikai műveletben felhasználni, mert ez helytelen, és nem is logikus.

Miután létrehoztunk memóriaváltozókat, ^DISPLAY MEMORY^ parancs segítségével nézzük meg, mi van a memóriában. Láthatjuk majd az előbbi példa memóriaváltozóinak nevét, tí-

pusát, tartalmát, és — az utolsó sorban — olvashatjuk a memóriaváltozók számát és az általuk elfoglalt hely nagyságát.

```
. display memory
KERDES                          (C)     HOGY ERZI MAGAT '86-BAN
SZORZO                          (N)          10
SZORZANDO                       (N)          17.35
NAGY                            (N)          99
KICSI                           (N)          33
EREDMENY                        (N)          3
FORMA                           (C)          99
**TOTAL** 07 VARIABLES USED     00054 BYTES USED
                **ÖSSZESEN** 07 MEMORIAVALTOZO    00054 BYTE
```

SZABÁLYOK

— Kifejezésben szereplő karakterláncok mindenképpen bezárandók aposztrófok, idézőjelek, illetve szögletes zárójelek közé.

— A karakterláncok bármilyen megjeleníthető karaktert tartalmazhatnak (beleértve az üres közt is).

— Ha az & jelet mint karaktert szerepeltetjük, két üres köz közé kell zárnunk, mert az adatbázis-kezelő rendszer ezt a karaktert a makró beépített függvény jelölésére használja (ennek a leírását lásd a 32. oldalon a 2.7. részben).

Az előbbi példa utolsó parancsa a 6.1. részben ismertetendő ^DISPLAY^ parancs egy nagyon hasznos formája. (Adhatjuk azt a parancsot is, hogy ^LIST MEMORY^.)

5.2.2 Memóriaváltozók törlése (RELEASE)

Egy memóriaváltozót megszüntethetünk a

^RELEASE <memóriaváltozó neve>^

paranccsal. Megszabadulhatunk az összes memóriaváltozótól, ha a parancs

^RELEASE ALL^.

Meghatározott memóriaváltozókat törölhetünk a

^RELEASE <memóriaváltozó, lista>^

paranccsal, a memóriaváltozókat a listában vesszővel elválasztva egymástól.

Gépeljük be a következő parancsszavakat és nézzük a választ! (Előtte célszerű ^ERASE^ paranccsal a képernyőt letörölni.)

```
. display memory
KERDES                    (C)     HOGY ERZI MAGAT '86-BAN
SZORZO                    (N)          10
SZORZANDO                 (N)          17.35
NAGY                      (N)          99
KICSI                     (N)          33
EREDMENY                  (N)          3
FORMA                     (C)          99
**TOTAL** 07 VARIABLES USED     00054 BYTES USED
                **ÖSSZESEN** 07 MEMÓRIAVÁLTOZÓ    00054 BYTE

. release kicsi
. display memory
KERDES                    (C)     HOGY ERZI MAGAT '86-BAN
SZORZO                    (N)          10
SZORZANDO                 (N)          17.35
NAGY                      (N)          99
EREDMENY                  (N)          3
FORMA                     (C)          99
**TOTAL** 06 VARIABLES USED     00048 BYTES USED
                **ÖSSZESEN** 06 MEMÓRIAVÁLTOZÓ    00048 BYTE

. release all
. display memory
**TOTAL** 00 VARIABLES USED     00000 BYTES USED
                **ÖSSZESEN** 00 MEMÓRIAVÁLTOZÓ    00000 BYTE
```

5.2.3 Memóriaváltozók megőrzése (SAVE)

A memóriaváltozókat mágneslemezen a

^SAVE TO <állománynév>^

paranccsal tárolhatjuk. A parancs hatására keletkező .MEM kiterjesztésű memóriaállomány tartalmazni fogja az összes pillanatnyilag meghatározott memóriaváltozót.

Próbaképpen gépeljük be a következőket!

```
. store 1 to egy
. store '1' to elso
. store 2 to ketto
. store '2' to masodik
. display memory
```

```
EGY                             (N)      1
ELSO                            (C)     1
KETTO                           (N)      2
MASODIK                         (C)     2
**TOTAL** 04 VARIABLES USED     00014 BYTES USED
                **ÖSSZESEN** 04 MEMORIAVALTOZÓ    00014 BYTE

. save to proba
```

5.2.4 Megőrzött memóriaváltozók munkába állítása (RESTORE)

^RESTORE FROM <állománynév>^

paranccsal kiolvashatjuk a ^SAVE TO <állománynév>^ paranccsal létrehozott állományokat, s így újra használhatjuk a mágneslemezen tárolt memóriaváltozókat. A parancs a kiadása előtt létrehozott összes memóriaváltozót törli.

Állítsuk munkába az 5.2.3. részben mágneslemezre írt memóriaváltozókat:

```
. release all
. display memory
**TOTAL** 00 VARIABLES USED     00000 BYTES USED
                **ÖSSZESEN** 00 MEMORIAVALTOZÓ    00000 BYTE

. restore from proba
. display memory
EGY                             (N)      1
ELSO                            (C)     1
KETTO                           (N)      2
MASODIK                         (C)     2
**TOTAL** 04 VARIABLES USED     00014 BYTES USED
                **ÖSSZESEN** 04 MEMORIAVALTOZÓ    00014 BYTE
```

6. Az adatok és a memóriaváltozók megjelenítése

6.1 AZ ADATOK MEGJELENÍTÉSE LISTÁZÓ PARANCSOKKAL (LIST, DISPLAY)

6.1.1 A listázó parancsok legegyszerűbb formái

Az adatbázis-kezelő rendszer parancsszavai angol nyelvű szavak. Megadhatjuk őket, amikor a rendszer a jellegzetes ponttal jelentkezik.

Amikor meghatározzuk, hogy melyik adatbázisállománnyal akarunk dolgozni,

^USE <állománynév>^ parancsot kell adnunk.

Ahhoz, hogy megnézzük azt a rekordot, amelyiken éppen állunk,

^DISPLAY^ -t kell beírni.

Az adatbázisban levő valamennyi rekord megjelenítése

^LIST^

paranccsal történhet. (A listázás megállítását, majd folytatását a ^S (DC3) kód megadásával lehet elérni.)

6.1.2 A listázó parancsok kibővítése az összehasonlító műveletekkel

Az adatbázis-kezelő rendszer tulajdonságai közül a hatékonyság növelését talán leginkább az szolgálja, hogy a parancsok kibővíthetők és különböző formára szabhatók. A legtöbb parancs kiegészíthető kifejezésekkel; ezáltal (segítségükkel) határozzuk meg az adatbázis-kezelő rendszer teendőit. A parancsok nagy- és/vagy kisbetűkkel is megadhatók, maximum 254 karakter hosszban. Ha elértük a parancs megadásakor a képernyő végét, pontosvesszőt kell írnunk utolsó karakterként a sorban. Ez után nem szabad üres helyet tenni. A rendszer a következő sort úgy fogja tekinteni, mint az előző parancs folytatását.

Az adatbázis-kezelő rendszer összehasonlító műveleteit (relációit) nagyon hatékonynak fogja találni a felhasználó.

Ezek a következők:

<	: kisebb, mint
>	: nagyobb, mint
=	: egyenlő
<=	: kisebb vagy egyenlő
>=	: nagyobb vagy egyenlő
<>	: nem egyenlő

Ezek a műveletek logikai értéket állítanak elő.

Korábban említettük, hogy a ^LIST^ parancs az adatbázisunk összes rekordját megjeleníti. A parancs egyik formája a következő:

^LIST [OFF] [FOR <kifejezés>]^

A szabadon választható OFF paraméterezéssel a parancs a rekordszámokat nem fogja megjeleníteni a képernyőn. Ha az ugyancsak szabadon választható FOR-ral paraméterezünk, csak azok a rekordok jelennek meg a képernyőn, amelyekre a FOR utáni kifejezés igaz.

Példa

```
. use nevek
. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST      1027
00002 CINGLER MONIKA PESTI U. 8           BUDAKESZI     2020
00003 DEAK GABOR      DEAK FERENC U. 1    HATVAN        6060
00004 ELEK FERENC     DOHANY U. 19/B      ZALAKAROS     9595

. list off
ASZTALOS ANTAL KACSA U. 16          BUDAPEST      1027
CINGLER MONIKA PESTI U. 8           BUDAKESZI     2020
DEAK GABOR      DEAK FERENC U. 1    HATVAN        6060
ELEK FERENC     DOHANY U. 19/B      ZALAKAROS     9595

. list for ir:szam = '9'
00004 ELEK FERENC     DOHANY U. 19/B      ZALAKAROS     9595

. list for ir:szam < '3'
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST      1027
00002 CINGLER MONIKA PESTI U. 8           BUDAKESZI     2020

. list off for nev = 'DEAK'
DEAK GABOR      DEAK FERENC U. 1    HATVAN        6060
```

MEGJEGYZÉS

Ha karakterlánc esetén a keresett érték egy részét adjuk csak meg, az adatbázis-kezelő ezt úgy értelmezi, mint a mező tartalmának a kezdetét, és ezt veszi figyelembe összehasonlításaiban. A példában nem volt szükségünk Deak Gabor nevének teljes megadására, de ha több Deak nevű lenne az adatbázisunkban, szükség lenne a teljes név megadására. (V. ö.: 232. old. SET EXACT)

A ^DISPLAY^ parancs hasonló a ^LIST^ parancshoz. Egyik gyakran használt formája a következő:

^DISPLAY { A ll / [Record] <szám> / Next <szám> } [OFF] [FOR <kifejezés>]^

A parancsban kiválasztható <érvényességi kör> meghatározása lehetővé teszi azt, hogy a ^DISPLAY^ parancs kívánt értelmezését érjük el. Ugyanis ha azt adjuk meg, hogy „record <szám>", akkor csak a <szám> által meghatározott rekordot jeleníti meg a képernyőn.

A „next <szám>" parancs értelemszerűen az aktuális rekordtól következő, <szám> darab rekordot jeleníti meg a képernyőn, beleértve az aktuális rekordot is.

A ^DISPLAY ALL^ parancs ugyanaz, mint a ^LIST^ parancs, de a ^LIST^ valamennyi rekordot folyamatosan jeleníti meg a képernyőn, a ^DISPLAY ALL^ pedig 15 rekordnyi csoportokban. Az első 15 rekordnyi adatcsoport megjelenítése után egy billentyű leütésére vár az adatbázis-kezelő rendszer. Bármelyik nyomógomb lenyomására a soron következő 15 rekordot fogjuk látni a képernyőn.

Akárcsak a ^LIST^ parancs esetében, itt is van egy szabadon választható FOR rész, amely arra alkalmas, hogy olyan adatokat válogassunk ki az adatbázisunkból, amelyek megfelelnek a logikai kifejezésben megadott feltételeknek.

6.1.3 Információk a rendszertől

Amennyiben személyesen szeretnénk az adatbázisról információkat szerezni, a LIST parancs alkalmazható arra is, hogy a rendszertől a szükséges adatokat megkapjuk.

^LIST STRUCTURE^

parancs megadása azt eredményezi, hogy a használatban levő adatbázis-állomány struktúrája jelenik meg a képernyőn.

Példa

```
. use nevsor
. list structure
STRUCTURE FOR FILE : NEVEK.DBF      AZ ÁLLOMÁNY SZERKEZETE
NUMBER OF RECORDS:   000010         REKORDOK SZÁMA
DATE OF LAST UPDATE: 85/10/19       UTOLSÓ HASZNÁLAT NAPJA
PRIMARY USE DATABASE                ELSŐDLEGES MUNKATER
```

```
FLD    NAME  TYPE   WIDTH  DEC     MEZŐ  NÉV  TIPUS  HOSSZ
001 NEV        C        14                            TIZEDESEK
002 CIM        C        19
003 VAROS      C        14
004 IR:SZAM    C        04
**TOTAL**              0052          **ÖSSZESEN**
```

^LIST FILES^

parancs megmutatja az éppen kiválasztott lemezegységen levő adatbázis-állományok neveit. (Az adatbázis-állományok állománynév-kiterjesztése pont utáni „DBF" rövidítés.)

```
. list files
DATABASE FILES   # RCDS   LAST UPDATE
NEVEK    DBF     00005     85/10/19
                           ADATBAZIS ALL. REKORDOK UTOLSO DATUM
```

A ^LIST FILES ON <lemezegység>^

parancs az általunk megjelölt mágneslemezen levő adatbázisállományok neveit jeleníti meg. A meghajtó megnevezésénél nem szabad használni a CP/M-ben alkalmazott kettőspontot!

A ^DISPLAY^ parancs, hasonlóan a ^LIST^ parancshoz, rendszerfunkciók betöltésére is alkalmas. A ^DISPLAY STRUCTURE^ parancs egyenértékű a ^LIST STRUCTURE^ paranccsal, a ^DISPLAY FILES^ parancs hatása ugyanaz, mint a ^LIST FILES^ parancsé.

A ^LIST^ és a ^DISPLAY^ parancs egyaránt képes a mágneslemezen elhelyezkedő, a CP/M rendszer szerint megnevezett állomány nevének megjelenítésére.

A ^DISPLAY FILES LIKE *.COM ON B^

parancs például a „b" mágneslemez összes „.COM" típusú állományának nevét képernyőre írja. A CP/M rendszer megnevezési elveit a CP/M kézikönyvéből lehet megismerni. A különböző típusú állományok megjelenítésére a következő minta szolgáljon például:

^DISPLAY FILES LIKE <rendszer szerinti megnevezés> [ON <meghajtó>]^

6.1.4 Parancssorok javítása

Az adatbázis-kezelő rendszer parancsszavait rövidített formában is megadhatjuk, legfeljebb négy betűre rövidítve a parancsszavakat. De ha több betűt használunk, akkor minden betűnek helyesnek kell lennie. (Például: ^DISPLAY^, ^DISP^ és a ^DISPLA^ helyes parancsszavak, de nem helyes a ^DISPRAY^-ként megadott formátum.)

A hibás parancsszót a rendszer átveszi, és lehetőséget ad korrekcióra úgy, hogy a teljes parancsot nem kell újból leírnunk.

```
***UNKNOWN COMMAND                          ***ISMERETLEN PARANCS
DISPRAY ALL FOR MUNKASZAM=730 .AND. DATUM=791015
CORRECT AND RETRY (Y/N)? y                  JAVIT RAJTA (Y/N)?
CHANGE FROM :PR                             MIT JAVIT :
CHANGE TO :Pl.                              MIRE JAVIT :
DISPLAY ALL FOR MUNKASZAM=730 .AND. DATUM=791015
MORE CORRECTIONS (Y/N)? n                   JAVIT MEG (Y/N)?
```

A rendszer megismétli az olyan parancsot, amelyet nem ismer. Ha ezt meg akarjuk változtatni, nem kell a teljes parancsot újra megadnunk. A gép által küldött „CHANGE FROM" üzenetre a parancsnak csak azt a részét kell megadnunk; <CR>-rel befejezve a választ; amelyet rossznak találunk. Ezután a gép megkérdezi, hogy a beírt részt mire akarjuk megváltoztatni („CHANGE TO"). Ilyenkor be kell írnunk azt, amivel a helytelen részt helyettesíteni akarjuk. (Az említett példában csak egyetlen betűt változtattunk meg.) Az adatbázis-kezelő rendszernek ez a képessége akkor használható előnyösen, ha hosszú parancsot akarunk javítani.

MEGJEGYZÉS

Az ^ERASE^ parancs a képernyőt letörli, és a bejelentkező pont a képernyő bal felső sarkában jelenik meg. Így elérhetjük, hogy az új parancsot teljesen új képernyővel kezdhetjük.

6.2 AZ ADATOK MEGJELENÍTÉSE MEGHATÁROZOTT FORMÁBAN (REPORT)

A 8.2. részben leírt ^FIND^ és ^LOCATE^ parancs alkalmas olyan rekord(ok) megkeresésére, amely(ek) valamilyen ismertetőjellel bír(nak). Sokszor azonban olyan adatösszegzést kérnénk, amely több – a feltételeknek megfelelő – rekordból származó adatokat tartalmaz. A ^REPORT^ parancsszóval kezdődő parancs könnyen és gyorsan eleget tesz az ilyen elvárásnak. (A ^SET EJECT OFF^ parancs alkalmazásával a jelentés – report – elején az automatikus lapemelést el lehet kerülni.) Válasszuk ki azt az adatbázist, amelyből a jelentést készítjük! Alkossuk meg a felhasználói jelentésformátumot a következők begépelésével!

```
^SET EJECT OFF^
^USE <adatbázis-állomány>^
^REPORT^
```

Ezután az adatbázis-kezelő rendszer kérdések sorozatát teszi fel, hogy alkalmunk legyen meghatározni a jelentésformátumot. Meghatározhatjuk, hogy milyen mezőkkel akarunk az adatbázisból dolgozni, megadhatjuk a jelentés címsorát, az egyes oszlopok szélességét és fejlécét, majd azt, hogy mely oszlopokat kell összegezni, és így tovább. Ha másképpen nem kérjük, akkor a jelentés formátuma automatikusan alakul: 8 karakternyi bal oldali margó, 56 sor egy oldalon és 80 karakteres lapszélesség. Kipróbálhatnánk mindezt azokkal az állományokkal, amelyeket eddig már előállítottunk, de a <Nevek.DBF> adatbázisunk nem tartalmaz annyi adatot, hogy kitetszhessék, mennyire hatékony ez a parancs. Ezért most egy <Forgalom.DBF> nevű állományt fogunk használni, amely része egy valódi alkalmazásnak, egy működő kereskedelmi rendszernek.

```
. use forgalom
. report
ENTER REPORT FORM NAME:vevok                    JELENTÉSFORMATUM NEVE
```
Mi a jelentésformátum-állomány neve?
Jelen esetben <Vevok.FRM>.

```
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH
      BEÁLLITHATOK,  M=BAL MARGO,    L=SOR/LAP,     W=SZÉLESSÉG
```
Beállítható a bal margó (pl. M=10), majd vesszővel elválasztva tőle laponkénti sorok száma (pl. L=55), és — újabb vessző után — a jelentés szélessége (pl. W=60). Bármelyik beállítás elhagyható. Most egyiket sem állítottuk be.

```
PAGE HEADING?(Y/N) y                          LESZ LAP FEJRÉSZ ?
```
Lesz-e fejrész? Igen.

```
ENTER PAGE HEADING: 'havi osszesites      MI A LAP FEJRÉSZE ?
```
A fejrész szövege: havi osszesites.

```
DOUBLE SPACE REPORT?(Y/N) n              ÜRES SOROK BEIKTATÁSA ?
```
Kell-e üres sor a rekordok közé? Nem.

```
ARE TOTALS REQUIRED?(Y/N) y                  KIVÁN ÖSSZEGZÉST ?
```
Kell-e majd összegzés? Igen.

```
SUBTOTALS IN REPORT?(Y/N) n                KÉR RÉSZÖSSZEGZÉST ?
```
Kellenek-e majd részösszegzések? Nem.

```
COL   WIDTH,CONTENTS                    OSZLOP SZÉLESSÉG,TARTALOM
```
Oszlop szélessége, tartalma?

```
001    8,dat
```
Az oszlop száma automatikusan jelenik meg. A többi adatot a felhasználó gépeli be. Példánkban az 1-es oszlop szélességét 8 karakterben határoztuk meg, és a <Dat> mező tartalmát íratjuk ki.

```
ENTER HEADING:datum                                  OSZLOP NEVE
002    22,nev
ENTER HEADING:vevo                                   OSZLOP NEVE
003    22,aru
ENTER HEADING:arucikk                                OSZLOP NEVE
004    12,ar
ENTER HEADING:osszeg                                 OSZLOP NEVE
ARE TOTALS REQUIRED?(Y/N) y                  KIVÁN ÖSSZEGZÉST ?

005    <cr>
```

Kell-e erre az oszlopra összegzés? Igen.
Ha be akarjuk fejezni a jelentésformátum meghatározását, kezeljük az <enter> vagy a <cr> nyomógombot, amikor a következő oszlop száma megjelenik.

Az adatbázis-kezelő rendszer azonnal megkezdi a jelentés adását, és az egész adatbázison végigmegy, ha hagyjuk. A jelentés megszakítására az *<ESC>* (*<escape>*) billentyű szolgál. A rendszer a jelentés formátumának megalkotásával egyidőben menti a jelentés formátumára vonatkozó utasításokat egy olyan állományba, amelynek .FRM lesz az állománynév-kiterjesztése. Ezt az állományt később használhatjuk vagy átalakíthatjuk anélkül, hogy még egyszer végig kellene menni a teljes jelentésformátum megfogalmazásán. A parancs egyik formátuma az alábbi:

^REPORT FORM <jelentésformátum-állomány neve> [<érvényességi kör>] [FOR<kifejezés>] [TO PRINT]^

Az 1985. májusi forgalmat tartalmazó listát a

^REPORT FORM Vevok FOR dat>'85/04/30' .AND. dat<'85/06/01'

eredményezi anélkül, hogy a formátumot újra meg kellett volna adnunk.

Példa

```
. use forgalom
. report form vevok for dat>'85/04/30' .and. dat<'85/06/01'

PAGE NO. 00001                                    LAP SZAM:  00001
85/05/30
                        HAVI OSSZESITES

 DATUM          VEVO                  ARUCIKK          OSSZEG

85/05/01     KISS PETER              FOTOPAPIR         177.00
85/05/10     NAGY ANASZTAZIA         SZINES NEGATIV    605.00
85/05/10     KISS PETER              FIXIR              37.10
85/05/15     KOVACS ISTVAN           DIA HIVAS         200.00
85/05/23     KISGERGELY PETER        NAGYITO           565.00
85/05/23     KISGERGELY PETER        VEGYSZEREK         56.00
**TOTAL**                                            1640.10
                                                     **OSSZES**
```

Az <érvényességi kör> rész automatikusan mindent figyelembe vesz, ha ezt másképpen nem adjuk meg. A <kifejezés> rész más feltételekkel is kibővíthető lett volna. Ha a parancs végén a „TO PRINT" paraméter is szerepel, a jelentés nyomtatóra kerül.

A fejlécben levő információt kibővíthetjük második fejléccel, ha a REPORT parancs kiadása előtt azt írjuk be, hogy

^SET HEADING TO <fejléc szövege>^

A szöveg 60 karakteres lehet, de nem használhatunk bennek idézőjelet (") vagy aposztrófot (').

Nem jelenik meg a dátum, a lapszámozás és a második fejléc, ha a PLAIN paramétert is megadjuk a parancsban.

6.3 POZICIONÁLÓ PARANCSOK (GO vagy GOTO és SKIP)

Ha már létrehoztunk egy adatbázist, az adatbázis-kezelő rendszer pozicionáló parancsai segítségével gyorsan és könnyen tudunk mozogni benne. A pozicionáló parancsok segítségével meghatározhatjuk, hogy melyik rekord legyen a munkaállomány aktuális rekordja, amelyre a rekordmutató mutat.

^**GO TOP**^ (vagy ^GOTO TOP^)
hatására adatbázisunk első rekordja,

^**GO BOTTOM**^
parancs hatására pedig az utolsó rekord iesz az aktuális.

Egy meghatározott rekord elérését a

^**GOTO** <szám>^ (vagy ^GO <szám>^)
parancsok megadásával valósíthatjuk meg. A „GO"-t el is hagyhatjuk; ilyenkor egyszerűen csak a rekorszámot kell közölnünk.

^**SKIP**^
a következő rekordra vezérli a rendszert.

^**SKIP**±<szám>^
előre vagy visszafelé mozgatja a vezérlést a <szám> által meghatározott mértékben.

^**SKIP** ±<változó vagy kifejezés>^
az előre vagy visszafelé mozgatás mértékét a változó vagy a kifejezés értéke adja meg.

Példák

```
. use nevek
. go top
. display
00001 ASZTALOS ANTAL  KACSA U. 16        BUDAPEST        1027
. go bottom
. display
00011 JENEI PAL       PETOFI TER 13      KOSPALLAG       3556
. goto 5
. display
00005 EDENYESI GABOR EDENY U. 5          EDFLENY         6576
```

```
. 8
. display
00008 GONDA ANDRAS    JOZSEF KRT. 46     BUDAPEST          1077

. skip -3
. display
00005 EDENYESI GABOR EDENY U. 5          EDELENY           6576

. skip
. display
00006 ELEK FERENC     DOHANY U. 19/b     ZALAKAROS         9595
```

6.4 AZ INTERAKTÍV (? JELŰ) PARANCS

A ? jelű parancs lehetővé teszi, hogy az adatbázis-kezelő rendszert kalkulátorként használjuk. Egyszerűen be kell gépelnünk a kérdőjelet és utána — egy üres közt hagyva — egy matematikai műveletet, beépített függvényt, kifejezést, amelynek az értékére kíváncsiak vagyunk. Ezt követő <CR> kezelés után a következő sorban a rendszer adja a választ.

Ha két kérdőjelet használunk (^??^), a válasz ugyanabba a sorba érkezik.

Példa

```
. ? 73/3.0000
        24.3333

. ? 73.00/3
        24.33

. ? 73/3
        24
```

A ? jelű parancs matematikai műveletek eredményét az általunk megadott számok közül a legtöbb tizedesjeggyel rendelkező szám tizedesjegyeinek száma szerint hozza. Ezt szemlélteti a fenti példa.

A ? jelű parancsot úgy is felfoghatjuk, mintha „Mennyi az, hogy . . . ?" kérdést tennénk fel. A ? jelű parancsban megadhatunk egy kifejezést, egy változót (mezőnevet vagy memóriaváltozót), egy beépített függvényt vagy ezek listáját, vesszővel elválasztva őket egymástól.

Példa

```
. use nevek
. 6
. ? ir:szam
9595

. ? nev+varos
ELEK FERENC    ZALAKAROS
```

```
. skip
. ? nev
FEHER ELEK

. go bottom
. ? varos
KOSPALLAG
```

A beépített függvények leírásánál (27. oldal, 2.5 rész) bemutattuk, hogy a ? jelű parancs milyen módon alkalmazható az adatbázis-kezelő rendszer beépített függvényeinek egyszerű alkalmazásához. A későbbiekben ismertetjük azt is, hogyan használható ez a parancs a gépkezelőnek vagy a felhasználónak szóló üzenetek képernyőn való megjelenítésére.

7. Az adatbázisok, illetve az adatok módosítása

7.1 ADATBÁZISOK ÉS STRUKTÚRÁK MÁSOLÁSA (COPY)

Az adatbázis-kezelő rendszerben egy adatbázis-állomány másolatát létrehozhatjuk anélkül, hogy visszatérnénk számítógépünk operációs rendszeréhez. Gépeljük be a következőket!

```
. use nevek
. copy to atmeneti
00011 RECORDS COPIED                          00011 REKORD MASOLVA

. use atmeneti
. display structure
STRUCTURE FOR FILE: ATMENETI.DBF              ALLOMANY SZERKEZETE:
NUMBER OF RECORDS: 00011                      REKORDOK SZAMA
DATE OF LAST UPDATE: 00/00/00                 UTOLSO DATUM
PRIMARY USE DATABASE                          ELSODLEGES MUNKATER
FLD        NAME  TYPE  WIDTH  DEC      MEZO NEV TIPUS HOSSZ
001  NEV             C    014                      TIZEDESEK
002  CIM             C    019
003  VAROS           C    014
004  IR:SZAM         C    004
**TOTAL**                 00052          **OSSZES**

. list
00001 ASZTALOS ANTAL KACSA U. 16          BUDAPEST        1027
00002 BERENDI HENRIK FENYVES UT 32        BUDAPEST        1028
00003 CINGLER MONIKA PESTI U 8            BUDAKESZI       2020
00004 DEAK GABOR     DEAK FERENC U 1      HATVAN          6060
00005 EDENYESI GABOR EDENY U. 5           EDELENY         6576
00006 ELEK FERENC    DOHANY U. 19/b       ZALAKAROS       9595
00007 FEHER ELEK     MENTE TER 3          FLEKES          4346
00008 GONDA ANDRAS   JOZSEF KRT. 46       BUDAPEST        1077
00009 HEGEDUS PAL.   BETLEHEMI U. 123     MARTONVASAR     2454
00010 INKE LASZLO    KOMAROMI UT 45       GYOR            2676
00011 JENEI PAL      PETOFI TER 13        KOSPALLAG       3556
```

MEGYJEGYZÉS

Ha COPY paranccsal egy már létező adatbázis-állomány nevét adjuk meg, az állomány felülíródik és a *régi adatok elvesznek.*

A ^COPY TO ATMENETI^ parancs egy új adatbázist hozott létre, amelynek a neve: <Atmeneti.DBF>. Ez az adatbázis-állomány teljesen megegyezik a <Nevek.DBF>-fel, ugyanaz a szerkezete (lásd CREATE, 43. old.) és ugyanazokkal az adatokkal van feltöltve.

A parancsot kibővíthettük volna a következőképpen:

^COPY TO <állománynév> [STRUCTURE] [FIELD <lista>]^

Ezzel a paranccsal *csak* a szerkezetet vagy pedig a szerkezet bizonyos *részeit* lehet egy másik állományba átmásolni. Adjuk meg a következő parancsokat:

```
. use nevek
. copy to atmeneti structure
. use atmeneti
. display structure
STRUCTURE FOR FILE: ATMENETI.DBF          ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS: 00000                  REKORDOK SZÁMA
DATE OF LAST UPDATE: 00/00/00             UTOLSÓ DÁTUM
PRIMARY USE DATABASE                      ELSŐDLEGES MUNKATÉR
FLD      NAME  TYPE  WIDTH  DEC         MEZŐ NÉV TÍPUS HOSSZ
001  NEV            C    014                      TIZEDESEK
002  CIM            C    019
003  VAROS          C    014
004  IR:SZAM        C    004
**TOTAL**                00052           **ÖSSZES**
```

A szerkezet egy részét úgy is átmásolhatjuk, hogy a parancsban csak azokat a mezőket soroljuk föl, amelyeket az új adatbázisban szeretnénk használni.

```
. use nevek
. copy to atmeneti structure fields nev, varos
. use atmeneti
. display structure
STRUCTURE FOR FILE: ATMENETI.DBF          ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS: 00000                  REKORDOK SZÁMA
DATE OF LAST UPDATE: 00/00/00             UTOLSÓ DÁTUM
PRIMARY USE DATABASE                      ELSŐDLEGES MUNKATÉR
FLD      NAME  TYPE  WIDTH  DEC         MEZŐ NÉV TÍPUS HOSSZ
001  NEV            C    014                      TIZEDESEK
003  VAROS          C    014
**TOTAL**                00029           **ÖSSZES**
```

7.2 EGY ÜRES ADATBÁZIS SZERKEZETÉNEK MEGVÁLTOZTATÁSA (MODIFY STRUCTURE)

Ha nincs az adatbázisunkban adat, a ^MODIFY^ utasítás a leggyorsabb és a legegyszerűbb parancs arra, hogy hozzáadjunk, töröljünk, átnevezzünk, új méretet adjunk vagy más változtatásokat végezzünk az adatbázis szerkezetében (struktúrájában). *Tudni kell azonban, hogy a* ^MODIFY^ *utasítás megsemmisít minden adatot, amely az adatbázisban van. Ezért ne használjuk, miután már adatokat vittünk be az adatbázisba.* (A későbbiekben be fogjuk mutatni, hogy bizonyos módon biztonságosan elvégezhető a művelet még ilyenkor is.)

A <kifizetes.DBF> jelen pillanatban még nem tartalmaz adatokat, így ezzel fogunk dolgozni. Hasznos változtatás lenne, ha átneveznénk a <B41725> mezőt <Munka:Díj>-ra. Ebben az esetben a rövidítés hasonló lenne az <Ir:Szam> mezőhöz. Adjuk meg a következőket:

```
. use Kifizetes
. list structure
STRUCTURE FOR FILE: KIFIZETE.DBF          ALLOMANY SZERKEZETE:
NUMBER OF RECORDS: 00000                  REKORDOK SZAMA:
DATE OF LAST UPDATE: 00/00/00             UTOLSO DATUM:
PRIMARY USE DATABASE                      ELSODLEGES MUNKATER
FLD       NAME    TYPE    WIDTH    DEC    MEZO NEV TIPUS HOSSZ
                                                    TIZEDESEK
001   NEV           C       014
002   CIM           C       019
003   VAROS         C       014
004   IR:SZAM       C       004
005   B41725        N       010     002
**TOTAL**                 00062             **OSSZES**

. modify structure
MODIFY ERASES ALL DATA RECORDS...PROCEED?(Y/N):y
      A MODIFY TORLI AZ OSSZES ADATREKORDOT...ELJARAS?(Y/N)
```

Az adatbázis-kezelő letörli a képernyőt és az első 16 mezőt listázza. A fénypont az első mezőnév első pozícióján áll.

^X *(CAN)* használatával mozoghatunk a mezőkön lefelé. Most adjuk meg az új mezőnevet, felülírva a régit.

A ^MODIFY^ szerkesztési módból való kilépés kétféleképpen lehetséges.

^W **(ETB)** kulcs használatával megváltoztatjuk a szerkezetet a mágneslemezen is, majd pedig az adatbázis-kezelő rendszer visszatér a parancsmódhoz.

^Q **(DC1)** kulcs alkalmazásával; anélkül, hogy a végrehajtott változtatásokat megőriznénk, az adatbázis-kezelő rendszer visszatér alapállapotba. Így tulajdonképpen visszatérhetünk az eredeti szerkezethez anélkül, hogy azt megváltoztattuk volna.

A változtatás után, ^W-vel kilépve:

```
. list structure
STRUCTURE FOR FILE: KIFIZETE.DBF          ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS: 00000                  REKORDOK SZÁMA:
DATE OF LAST UPDATE: 00/00/00             UTOLSÓ DÁTUM:
PRIMARY USE DATABASE                      ELSŐDLEGES MUNKATER
FLD      NAME     TYPE   WIDTH   DEC     MEZŐ NÉV TIPUS HOSSZ
001   NEV           C      014                      TIZEDESEK
002   CIM           C      019
003   VAROS         C      014
004   IR:SZAM       C      004
005   MUNKA:DIJ     N      010     002
**TOTAL**                 00062           **ÖSSZES**
```

7.3 AZ ADATBÁZISBAN LEVŐ MEZŐK BŐVÍTÉSE ÉS TÖRLÉSE (COPY, USE, MODIFY, APPEND)

Ahogy az adatbázis-kezelő rendszer felhasználásában előre haladunk, minden valószínűség szerint szükségünk lesz az adatbázisunk bővítésére vagy a mezők törlésére.

A ^MODIFY STRUCTURE^ utasítás önmagában megsemmisítené az adatbázis adatait. A ^COPY^ és az ^APPEND^ használatával azonban ez elkerülhető. Az eljárás a következő: A megváltoztatandó adatbázis-állomány struktúráját át kell másolni egy átmeneti állományba, azután ezen elvégezni a szükséges szerkezeti módosításokat. A megváltoztatandó állományból át kell másolni az adatokat az új, módosított szerkezetű állományba. Példaképpen az eddig is használt <Nevek.DBF> nevű állományt fogjuk használni.

Bizonyos esetekben hasznos lenne, ha úgy listázhatnánk, hogy egy általunk megadott kód szerinti adatok jelennének meg, például egy felhasználói számot tartalmazó mezőt tennénk hozzá a <Nevek.DBF> nevű állományunkhoz. Annak érdekében, hogy ezt végrehajtsuk — és természetesen ne töröljük le a már rendelkezésünkre álló rekordokat — a következő parancsokat adjuk meg.

```
. use nevek
. copy to atmeneti structure
. use atmeneti
. modify structure
```

Módosítsuk a szerkezetet úgy, hogy a fénypontot mozgassuk az első üres helyre, gépeljük be a változtatást, majd pedig adjunk ^W(ETB) kódot; így a változtatásokat megőrizve térhet vissza az adatbázis-kezelő rendszer az alapállapotba. A ^DISPLAY STRUCTURE^ parancs által megbizonyosodunk arról, hogy rendben végeztük-e a műveletet. Ha ez így van, akkor az adatokat a <Nevek.DBF> nevű állományból hozzáadhatjuk az új állományunkhoz. Adjuk meg, hogy

```
^APPEND FROM NEVEK^
```

A mező méreteit is megváltoztathattuk volna. Az ^APPEND^ parancs úgy viszi át az adatokat a mezőkbe, hogy a megfelelő nevekhez teszi őket.

```
. display structure
STRUCTURE FOR FILE: ATMANETI.DBF          ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS: 00011                  REKORDOK SZÁMA:
DATE OF LAST UPDATE: 00/00/00             UTOLSÓ DÁTUM:
PRIMARY USE DATABASE                      ELSŐDLEGES MUNKATÉR
FLD      NAME    TYPE  WIDTH   DEC      MEZŐ NÉV TÍPUS HOSSZ
001   NEV          C       014                      TIZEDESEK
002   CIM          C       019
003   VAROS        C       014
004   IR:SZAM      C       004
005   KOD          N       003
**TOTAL**                  00055          **ÖSSZES**
```

Az új állományunk, az <Atmeneti.DBF> tartalmazza már az új mezőt és a régi adatokat. ^DISPLAY STRUCTURE^, majd ^LIST^ parancsok után, ha megbizonyosodtuk róla, hogy jól sikerült a másolás és helyesnek találtuk az adatátvitelt, befejezhetjük műveletünket a következővel:

```
. copy to nevek
. use nevek
```

A ^COPY^ parancs felülírja a régi struktúrát és az adatokat. Miután a képernyőre írattuk és listáztattuk az új <Nevek.DBF> állományt, a ^DELETE FILE ATMENETI^ paranccsal törölhetjük az <Atmeneti.DBF> nevű állományt.

Összefoglalva az eddigieket: ez az eljárás arra alkalmas, hogy hozzáadjunk vagy töröljünk mezőket az adatbázisból a következő sorrend szerint végrehajtott parancsokkal:

```
. use nevek                                          (régi állomány)
. copy to atmeneti structure          (uj állomány szerkezete)
. use atmeneti                                       (uj állomány)
. modify structure    (uj állomány szerkezetének módosítása)
. append from nevek    (régi állomány adatainak hozzáfüzése)
00010 RECORDS ADDED                    00010 REKORD HOZZÁFŰZVE
. copy to nevek                                      (régi állomány)
00010 RECORDS COPIED                   00010 REKORD MÁSOLVA
. use nevek
```

7.4 ADATBÁZISMEZŐK ÁTNEVEZÉSE (COPY, APPEND)

Mint azt korábban már elmondottuk, az ^APPEND^ parancs az egyik állomány mezőiből a másik állomány megfelelő mezőibe másolja át az adatokat. Ha az APPEND FROM-mal meghívott állományban levő valamelyik mező nem létezik a USE-zal használt állományban, akkor a mező tartalma nem kerül átvitelre. Jóllehet, a parancs teljes formátuma csak arra ad lehető-

séget, hogy adatokat vigyünk át vele, de felhasználhatjuk arra is, hogy a mezőknek új nevet adjunk az adatbázisban. Például, ha át akarjuk nevezni a <Kifizetesek.DBF> állományunkban a <B41725> mezőt <Munka:Díj>-ra, és vannak már adataink az adatbázisban, a következőképpen kell ezt végrehajtanunk.

```
. use nevek                                    (régi állomány)
. copy to atmeneti sdf      (uj, SDF állomány: Atmeneti.TXT)
. modify structure
. append from atmeneti.txt sdf    (az adatok hozzáfüzése az
                                   Atmeneti.TXT állományból)
00010 RECORDS ADDED                  00010 REKORD HOZZAFUZVE
```

Ha most végrehajtatjuk a ^DISPLAY STRUCTURE^ utasítást, az utolsó mező neve már <Munka:Díj> lesz.

Amikor a

^COPY TO <célállomány> SDF^ vagy a
^COPY TO <célállomány> DELIMITED

parancsok segítségével az adatokat egy <.TXT> kiterjesztésű állományba írjuk, használhatjuk a teljes parancsformátumot, hogy megadjuk a részletes parancsértelmezést, az *<érvényességi kör>*-t, a *mezőnevek* felsorolását és a *<kifejezések>*-et.

Egy <.TXT> kiterjesztésű állományba, amelyet az ^SDF^ (vagy ^DELIMITED^) paraméterrel kiegészített paranccsal állítottunk elő, az adatok az eredeti állományban meghatározott mezőhosszak szerint kerülnek át.

MEGJEGYZÉS

Ne változtassuk meg a mező pozícióját vagy méretét!

Az adat, amit az adatbázisból a szöveg típusú állományba másoltunk, *pozíció* szerint kerül át, és nem név szerint.

Ha a szerkezet módosításával a mező méretét is megváltoztatjuk, az adatbázis-struktúra számára hozzáférhetetlenné tesszük adatbázisunkat, és az átmásolt adatokat nem tudjuk visszamásolni.

Ha egy, a nevében <.TXT> kiterjesztést tartalmazó állományt szerkesztünk a szövegszerkesztő program segítségével — például ideiglenes adatbevitelként —, vegyük figyelembe az előbb leírtakat, és úgy szerkesszük meg adatainkat, hogy azok hossza megegyezzen az adatbázis-kezelő rendszerbe másolásnál használandó adatbázis-állomány szerkezetében meghatározott mezőhosszakkal.

7.5 AZ ADATBÁZIS-SZERKEZETEK MEGŐRZÉSE ÉS MÓDOSÍTÁSA, HA A SZERKEZETET ADATKÉNT KEZELJÜK (COPY . . . EXTENDED, CREATE . . . FROM . . .)

A ^COPY^ parancs által az adatbázis szerkezetéhez felhasználói programból is hozzáférhetünk. Adjuk meg a következőket!

```
. use nevek
. list structure
STRUCTURE FOR FILE: NEVEK.DBF          ALLOMANY SZERKEZETE:
NUMBER OF RECORDS: 00011               REKORDOK SZAMA:
DATE OF LAST UPDATE: 00/00/00          UTOLSO DATUM:
PRIMARY USE DATABASE                   ELSŐDLEGES MUNKATER
FLD      NAME   TYPE  WIDTH   DEC     MEZŐ NEV TIPUS HOSSZ
001   NEV         C       014                    TIZEDESEK
002   CIM         C       019
003   VAROS       C       014
004   IR:SZAM     C       004
**TOTAL**                 00052        **ÖSSZES**

. copy to ujdonsag structure extended
. use ujdonsag
. display structure
STRUCTURE FOR FILE: UJDONSAG.DBF       ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS: 00004               REKORDOK SZÁMA:
DATE OF LAST UPDATE: 00/00/00          UTOLSÓ DÁTUM:
PRIMARY USE DATABASE                   ELSŐDLEGES MUNKATÉR
FLD    NAME       TYPE  WIDTH  DEC     MEZŐ NÉV TÍPUS·HOSSZ
001   FIELD:NAME   C      010                  TIZEDESEK
002   FIELD:TYPE   C      001
003   FIELD:LEN    N      003
004   FIELD:DEC    N      003
**TOTAL**              00018          **ÖSSZES**

. list
00001   NEV          C 14  0
00002   CIM          C 19  0
00003   HELYSEG      C 14  0
00004   IR:SZAM      C  4  0
```

Az <Ujdonsag.DBF> adatbázis rekordjai leírják a <Nevek.DBF> adatbázis szerkezetét, és egy felhasználói program közvetlenül hozzá tud férni ehhez az információhoz.

Azt is elérhetjük, hogy egy állományt, azzal a szerkezettel, amit az <UJDONSAG.DBF> a rekordjaiban leír, beágyazhassunk a programunkba. Így ennek kezelője megadhatja egy állomány szerkezetét anélkül, hogy az adatbázis-kezelő rendszer parancsait megtanulná. A felhasználói program pedig a következő paranccsal állítaná elő a felhasználó számára az adatbázist:

^CREATE <adatbázis-állomány> FROM <"struktúra"-állomány>^

7.6 MÁS RENDSZEREK ADATÁLLOMÁNYAINAK KEZELÉSE (COPY, APPEND)

A dBASE II, illetve a PROP-BASE által létrehozott adatbázis információját átalakíthatjuk olyan formátumra, amely kompatibilis más rendszerekével. Említettük már például a BASIC, a PASCAL, a FORTRAN, a PL/1 stb. magas szintű nyelvekben írt rendszereket. Az adatbázis-kezelő rendszerrel viszont feldolgozhatjuk az e rendszerekből származó adatokat. A CP/M rendszerben ugyanis az általános adatformátum (rövidítve SDF) tartalmaz egy carriage return (kocsi vissza) és line feed (soremelés) kódot minden egyes szövegsor végén. Hogy valamelyik adatbázisunkból egy kompatibilis adatállományt állítsunk elő, a ^COPY^ parancs következő formáját kell használnunk:

^COPY TO IDEGEN SDF^

Ez a parancs egy olyan állományt hoz létre, amelynek a neve <Idegen.TXT>. Hagyjuk el ^QUIT^-tel az adatbázis-kezelő rendszert, és használjuk például a szövegszerkesztőnket, hogy az állományt megnézzük. Azt fogjuk találni, hogy ugyanúgy dolgozhatunk vele, mint egy, a fentiekben felsorolt rendszerrel előállított állománnyal. Az általános adatformátum (SDF — System Data Format) lehetővé teszi azt is, hogy az adatbázis-kezelő olyan adatokkal dolgozzon, amelyek más rendszerekben létrehozott állományokból származnak, jóllehet *vigyázni kell arra, hogy az adatok hossza megegyezzen annak az adatbázisnak a struktúrájában megadott hosszal, amelyikben használni akarjuk őket.*

Tehát szövegszerkesztővel előállítottunk egy olyan állományt, amelynek a neve <MasNevek.TXT>, ezt hozzákapcsolhatjuk a <Nevek.DBF> állományunkhoz. Ha például a <MasNevek.TXT> állomány a következő információkat tartalmazza:

KOVACS PETER	NAGYAJTAI U 2/B	BUDAPEST	1022
FAZEKAS PÁL	HENGERSOR U 55	BUDAPEST	1118
STORNO BRUNO	BACHUS UT 45	SOPRON	3456
FRANK FERENC	ROTTENBILLER U 35	BUDAPEST	1132
(14)	(19)	(14)	(4)

akkor hozzáadhatjuk azt a <Nevek.DBF> nevű állományhoz, a következők megadásával:

^USE NEVEK^
^APPEND FROM MASNEVEK.TXT SDF^.

MEGJEGYZÉS

Az adatoknak ugyanakkora helyet kell elfoglalniuk, mint amekkorát az adatbázisunk szerkezetében mezőszélességként meghatároztunk.

Egy már létező adatbázis-állományhoz más rendszerbeli adatállományból adatok hozzáadása pusztán másodperceket vesz igénybe. Az eljárás hasonló az előbbihez abban az esetben is, ha az „idegen" rendszerbe tartozó állomány más határolókat tartalmaz. E rész elején felsorolt rendszereknél az adatállomány-formátumban a mezők között vesszőket használunk, és felülvesszőkkel (aposztrófokkal) határoljuk a karakterláncokat, hogy elválasszuk az állománybeli adatokat. Ahhoz, hogy előállítsuk és használjuk az ilyen típusú adatállományokat, a DELIMITED

kulcsszót kell csatolnunk az SDF helyett. Szemléltetésként, hogy ez miként módosíthatja a parancs végrehajtását, adjuk meg a következőt:

^COPY TO ATMENETI DELIMITED^

Ezután térjünk vissza az operációs rendszerünkhöz, megnézni az adatokat. Ha a mi rendszerünk az előbbitől eltérő határolót tartalmaz, a parancsban megadhatjuk azt is:

^COPY TO ATMENETI DELIMITED [WITH <határoló>]^.

(Nem lehet határoló a „<" vagy a „>" szimbólum.) Ha a rendszerünk csak vesszőt, és semmi mást nem tartalmaz a karakterláncok mellett, akkor

^COPY TO ATMENETI DELIMITED WITH , ^

gépelendő be.

A ^COPY^ és az ^APPEND^ parancsok teljes formátuma a *rendszer-adatállományokkal kapcsolatos* másolási műveletekben a következő:

^COPY [<érvényességi kör>] TO <állománynév> [FIELD <lista>]⇒

⇒ { SDF / DELIMITED [WITH <határoló>] } [STRUC⇒

⇒TURE] [{ FOR <kifejezés> / WHILE <kifejezés> }] ^

^APPEND FROM <állománynév.TXT> { SDF / DELIMITED [WITH <határoló>] } ⇒

⇒ [{ FOR <kifejezés> / WHILE <kifejezés> }] ^

Mint láthatjuk mindkét parancs végrehajtása módosítható a <kifejezések> használatával, és a ^COPY^-nál levő <érvényességi kör> paraméter ugyanúgy megadható, mint más parancsok esetében.

MEGJEGYZÉS

Az adatbázis-kezelő rendszer automatikusan generálja az állománynév-kiterjesztést az állomány létrehozásakor, a „.TXT" állománynév-kiterjesztést viszont meg *kell* adnunk akkor, amikor APPEND utasítást használunk, és más rendszerbe tartozó adatállományból hozzuk be az adatokat.

Az APPEND parancs esetében minden mezőnek, amely a <kifejezés> részben szerepel, léteznie *kell* abban az adatbázisban, amelyikbe az adatokat átmásoljuk.

7.7 GYORSABB ADATMÓDOSÍTÁSI LEHETŐSÉGEK (REPLACE, CHANGE)

Az alábbi parancs használatával a változtatások gyorsan elvégezhetők bármelyik, illetve az összes rekordra vonatkozóan is.

^REPLACE [<érvényességi kör>] <mező1> WITH <kifejezés1>⇒

⇒[,<mező2> WITH <kifejezés2>] . . .[,<mező5> WITH <kifejezés5>]⇒

⇒ [{FOR <kifejezés> / WHILE <kifejezés>}] ^

Ez különlegesen hasznos parancs, mert végrehajtja a helyettesítést vagy a cserét a megnevezett mezőben — azzal az adattal, amelyet megadunk a „WITH" kulcsszó után. A REPLACE parancsban egynél több — maximum 5 — mezőt is megjelölhetünk; úgy, hogy az első mező és adat után vesszővel elválasztva soroljuk fel a továbbit (a továbbiakat).

Az adat lehet egészen új információ (beleértve a szóközöket, tehát a törlést is) vagy pedig egy művelet eredménye. Például előfordulhat, hogy az adó csökkentését a számláinkon figyelembe kell venni. Ebben az esetben a következőt hajtathatjuk végre: ^(REPLACE all Fogy:Ar WITH Fogy:Ar/1.06)^. A helyettesítést feltételessé is tehetjük úgy, hogy használjuk a FOR vagy a WHILE paramétert, és megadunk valamilyen feltételt — kifejezés formájában.

Ahhoz, hogy megmutassuk, miként dolgozik ez a parancs, szükségünk van egy új adatbázisra. A példában az adatbázis létrehozását és feltöltését nem mutatjuk be.

```
. use koltseg
. list structure
STRUCTURE FOR FILE: KOLTSEG.DBF          ADATBAZIS SZERKEZETE:
NUMBER OF RECORDS: 00005                 REKORDOK SZAMA:
DATE OF LAST UPDATE: 00/00/00            UTOLSO DATUM:
PRIMARY USE DATABASE                     ELSODLEGES ADATBAZIS
FLD     NAME     TYPE  WITH   DEC       MEZO NEV TIPUS HOSSZ
                                                  TIZEDESEK
001   MEG:NEV      C     015
002   MENNYISEG    N     006
003   DARAB:AR     N     007    002
004   KIADAS       N     008    002
**TOTAL**                00037            **OSSZES**

. list
00001   FOTOPAPIR          191       1.57
00002   FORTE DIAPOSITIV  1236      80.00
00003   FORTE NEGATIV      153      69.50
00004   AGFA DIAPOSITIV    171      96.00
00005   AGFA NEGATIV        75      85.50

. replace all kiadas with mennyiseg*darab:ar
00005 REPLACEMENT(S)                     00005 HELYETTESITES
```

```
. list
00001  FOTOPAPIR            191      1.57       299.87
00002  FORTE DIAPOSITIV    1236     80.00     98880.00
00003  FORTE NEGATIV        153     69.50     10633.50
00004  AGFA DIAPOSITIV      171     96.00     16416.00
00005  AGFA NEGATIV          75     85.50      6412.50
```

A REPLACE parancsot kényelmesnek fogjuk találni a parancsállományokban, amikor egy olyan üres rekordot, amelyet hozzáfűztünk az állományhoz, fel akarunk tölteni. A memória-változókból származó adatok a programokban gyakran arra szolgálnak, hogy az üres mezőket feltöltsék.

Ha nagyon sok rekordunk van, és csak kevés számú mezőt kell megváltoztatnunk, az alábbi parancs segítségével ez könnyű lesz:

^CHANGE [<érvényességi kör>] FIELD <lista> [FOR<kifejezés>]^

Az <érvényességi kör> paraméter ugyanaz, mint más parancsoknál. A mezőlistában legalább egy mezőt meg kell nevezni, de több mezőnév is megadható, ha azokat vesszővel elválasztjuk egymástól.

Ez a parancs megkeresi az első olyan rekordot, amely kielégíti a kifejezés által meghatározott feltételeket, majd pedig kijelzi a képernyőre a rekord nevét és tartalmát. Ahhoz, hogy a mezőben az adatokat megváltoztathassuk, be kell írnunk az új információt. Ha nem szükséges változtatás, egyszerűen <enter> vagy <cr> kódot kell megadnunk. Ha a mező üres, és adatokat szándékozunk beleírni, szünet (space = űr) billentyűt kell ütnünk.

Ha a rekord valamennyi mezejét megmutatta már az adatbázis-kezelő rendszer, a következő olyan rekordot (illetve annak az első mezejét) fogja elővenni, amely az adott feltételeknek megfelel. Amikor vissza akarunk térni az adatbázis-kezelőhöz, az <ESC> (ESCAPE) (^[) billentyűt kell leütnünk.

```
. use koltseg
. change field darab:ar

REKORD#00001
DARAB:AR        1.57
CHANGE?         1.66  <cr>                                  CSERÉL?
                  (Cserél?  Adjon szóközt a cseréhez, ha a mező
                  üres. Uj érték: 1.66 <cr>.)

DARAB:AR        1.66
CHANGE? <cr>      (vagy <enter>)

RECORD#00002
DARAB:AR
CHANGE?
```

7.8 MUNKAVÉGZÉS TÖBB ADATBÁZISSAL (SELECT PRIMARY, SELECT SECONDARY)

^USE <állománynév>^ paranccsal kezdtük a munkát az adatbázis-kezelővel ahhoz, hogy megmondjuk a rendszernek, melyik az az állomány, amely érdekel bennünket. Ezzel az állománnyal végre tudtuk hajtani azokat a műveleteket, amelyekkel adatokat vittünk be, szerkesztettünk, átalakítottunk stb. Ahhoz, hogy egy másik adatbázissal dolgozzunk, a ^USE <új állománynév>^ parancsot kell megadnunk. Erre az adatbázis-kezelő rendszer lezárja az előző állományt, és megnyitja az újonnan megadottat. Így annyi állományt használhatunk, amennyit csak akarunk, mind közvetlen, mind közvetett módban. Ha csak a ^USE^ parancsot adjuk ki, egy állományt lezárhatunk anélkül, hogy egy újat megnyitnánk.

Amikor a USE parancs segítségével egy állományt munkába állítunk, az adatbázis-kezelő az első rekordra áll. A legtöbb esetben egyébként is ezt szeretnénk. De előfordulhat, hogy egy újabb állományhoz kívánunk hozzáférni anélkül, hogy a már meghívott és munkába vett állományban elmozdulnánk egy elért rekordról. Az adatbázis-kezelőnek van egy nagyon hatékony megoldása, amely lehetővé teszi, hogy két különálló aktív területen tudjunk dolgozni: az egyik a PRIMARY (első), a másik a SECONDARY (második) munkatér. A SELECT parancs segítségével lehet az egyik munkatérről a másikra átkapcsolni.

Automatikusan a PRIMARY területen vagyunk, amikor kezdjük a munkát. Azért, hogy egy újabb adatbázison tudjunk dolgozni anélkül, hogy az eredetiben a pozíciónkat elvesztenénk, adjunk a rendszernek

^SELECT SECONDARY^

parancsot, majd ezt követően ^USE <új állománynév>^ parancsot. Ahhoz, hogy az eredeti munkaterületre visszatérjünk,

^SELECT PRIMARY^

parancsot kell beírnunk, és folytathatjuk az ezzel az adatbázissal végzett munkánkat.

A két munkaterület egymástól függetlenül használható. Azok a parancsok, amelyek adatokat vagy rekordokat mozgatnak, csak a használatban levő adatterületre hatnak.

Információt vihetünk *át* az egyik területről a másik területre úgy, hogy a mezőváltozók neve előtt „P.", illetve „S." bevezető jelöléseket alkalmazunk. Ha a PRIMARY területen vagyunk, tegyük az „S." bevezető jelölést azokhoz a mezőváltozókhoz, amelyeket a SECONDARY (második) területről kívánunk használni. Ha a SECONDARY területen vagyunk, értelemszerűen a „P." bevezető jelölést használjuk. Például ^SELECT SECONDARY^ után a SECONDARY munkaterületről a (PRIMARY munkaterületen aktivizált) <Nevek.DBF> adatbázis <Nev> mezőjéhez a „P.NEV" hivatkozással férhetünk hozzá.

8. Műveletek az adatbázisokon

8.1 AZ ADATBÁZIS ÁTRENDEZÉSE (SORT, INDEX)

Az adatokat leggyakrabban rendezetlenül visszük be az adatbázisba, vagy mint ahogy a <Nevek.DBF> adatbázis esetében tettük, valamilyen sorrend szerint. Nem biztos, hogy ugyanebben a sorrendben kívánjuk megtartani az adatainkat (például szükségünk lehet rájuk a helységek alfabetikus sorrendjében). Ilyen célokat szolgál az adatbázis-kezelő rendszer néhány lehetősége, amelyek segítségével az adatbázisunk rendezhető. Felhasználhatjuk sorbarendező és indexelő műveletét, melyekről már az 1.4. részben is volt szó. Az indexelt állományok lehetővé teszik, hogy a rekordokat gyorsan megtaláljuk (ez a FIND parancs esetében 2 másodpercen belül van).

A SORT parancs alkalmazását *sorbarendezésnek* nevezzük. Az állományok rekordjait növekvő és csökkenő sorrend szerint rendezhetjük. A parancs egyik alakja a következő:

^SORT ON <kulcsmező> TO <állománynév> [DESCENDING]^

A mezőnév határozza meg azt a kulcsot, amelynek alapján az állomány rekordjait rendezni kell. Ez a kulcsmező lehet karakter típusú vagy numerikus (de nem lehet logikai). Karakter típusú mező esetén ASCII kód szerint, numerikus mezőnél értelemszerű a sorrend. A sorbarendezés — ha ezt nem adjuk meg másként — automatikusan növekvő sorrend szerinti lesz, de a parancsban — az utolsó, szabadon választható parancsrészben — ezt csökkenő sorrendre módosíthatjuk. A többkulcsos sorbarendezést kezdjük a legkevésbé fontos kulccsal, a SORT parancsok sorozatát megadván a legfontosabb kulcsig. (A sorbarendezés folyamán az adatbázis-kezelő rendszer ugyanis csak annyi rekordot mozgat meg, amennyire feltétlenül szükség van.) Ahhoz, hogy a példánkban szereplő <Nevek.DBF> nevű állományt a helységek alfabetikus sorrendjében kapjuk meg, a következőt gépeljük be:

```
. use nevek
. sort on helyseg to hely
. use hely
. list
. copy to nevek
```

(További példa a 11.2. részben, a 239. oldalon.)

MEGJEGYZÉS

Ne használjuk a SORT parancsot úgy, hogy célállományként a forrásállományt adjuk meg! Egy pillanatnyi zavar a tápfeszültségben tönkreteheti a teljes adatbázisunkat. A sorbarendezést inkább egy átmeneti állományba végezzük, s ha megbizonyosodtunk arról, hogy a sorbarendezés jól sikerült, rendezett adatainkat ezután másoljuk vissza az eredeti állománynév alá!

Egy adatbázis indexelhető is. Ilyenkor olyan, mintha sorba lenne rendezve. Az indexelő parancs formátuma a következő:

^INDEX ON { <kifejezés> / <kulcsmező> } TO <indexállomány-név>^

A parancs végrehajtásával egy új, <.NDX> állománynév-kiterjesztésű állomány jön létre. Csak a kulcs által jelölt adatok rendezése következik be, habár úgy tűnik, mintha a teljes adatbázist rendeztük volna. A kulcs lehet egy mezőváltozó neve, de egy összetett kifejezés is (amely mezőre is utal), 100 karakter hosszúságig. Az indexelés kulcsában nem lehet logikai típusú mező! Ahhoz, hogy adatbázisunkat rendezzük az irányítószám <Ir:Szam> alapján, a következőt adjuk meg:

```
. use nevek
. index on ir:szam to szamok
. use nevek index szamok
. list
```

(További példa a 11.2. részben, a 175. oldalon.)

Az indexelést elvégezhettük volna három mező alapján is, a következő kulccsal:

^INDEX Ir:Szam+Helyseg+Cím TO Irany^

Azokat a numerikus mezőket, amelyeket karakter típusú mezőkkel együtt használunk fel, át kell alakítanunk karakter típusúvá. Ha az irányítószám numerikus mező lenne — 5 hellyel és ezen belül 2 decimális hellyel —, az STR beépített függvény (2.5. rész, 29. oldal) elvégzi a szükséges átalakítást:

^INDEX STR(Ir:Szam,5,2)+Helyseg+Cím TO Irany^

Az indexelt állomány kezeléséből adódó gyorsaság kihasználása érdekében úgy kell a ^USE^ parancsot paraméterezni, hogy rögtön megadjuk, milyen indexállomány szerint akarjuk az adatbázisunkat munkába állítani. Az erre szolgáló parancs a következő:

^USE <adatbázis-állomány neve> INDEX <indexállomány neve>^

Egyes pozicionáló parancsok (GO TOP, GO BOTTOM, SKIP, EDIT módban a ^C (ETX) és a ^R (DC2) stb.), ha indexelt állománnyal dolgoznak, az index szerinti pozicionálásokat hajtják végre, és nem az eredeti adatbázisban mozognak. A ^GO BOTTOM^ parancs végrehajtása például azt eredményezi, hogy az index szerinti utolsó rekordra jutunk ahelyett, hogy az eredeti adatbázisban levő utolsó rekordot érnénk el.

Ha az APPEND, az EDIT, a REPLACE vagy a PACK parancsokkal a kulcsban szereplő mezőkön úgy változtatunk, hogy az indexállomány is használatban van, akkor az indexállomány az új adatoknak megfelelően fogja tartalmazni az indexelési sorrendet.

Más indexállományok is, amelyeket USE-zal nem hívtunk meg, meghívhatók annak érdekében, hogy az új adatoknak megfelelő sorrendet mutassák. Ennek módja:

^SET INDEX TO <indexállomány1> [,<indexállomány2>] . . .⇒

⇒[,<indexállomány7>]^

Ezután használjuk az APPEND, az EDIT stb. parancsot. Minden megnevezett indexállomány a pillanatnyi helyzetnek megfelelő lesz.

Az indexelt állományok nagyon nagy előnye az, hogy lehetővé teszik a ^FIND^ parancs alkalmazását (ezt a következő részben fogjuk leírni), amellyel még nagy adatbázisok esetében is rövid időn belül megtaláljuk a keresett rekordot.

8.2 REKORD MEGKERESÉSE (FIND, LOCATE)

Ha valamilyen adatot keresünk, használhatjuk a FIND parancsszót, de csak akkor, ha az adatbázis indexelt, és az indexállományt USE-zal vagy SET INDEX-szel használatba vettük. Egyszerűen gépeljük be azt, hogy

^FIND <kifejezés értéke>^

ahol a <kifejezés értéke> az az eredmény — vagy karakterlánc esetében annak *kezdete* —, amely az indexelési kulcsban megadott kifejezés alapján adódik. Ha az indexelési kulcs eredménye karakterlánc, a <kifejezés értéke> bármilyen rövid lehet, de legyen elegendő hosszú a keresett rekord megtalálásához. A „te" például nagyon sok szóban előfordul, a „terem" jobban, a „teremőr" sokkal jobban meghatározott.

Adjuk meg a következő parancsokat:

```
. use nevek index szamok          (Az indexállományt a 8.1.
. find 10                          részben hoztuk létre.)
. display
00001 ASZTALOS ANTAL KACSA U. 16        BUDAPEST        1027

. find 9
. display
00006 ELEK FERENC     DOHANY U. 19/b    ZALAKAROS       9595
```

Ha a megadott <kifejezés értéke> nemcsak egyszer fordul elő az adatok között, az adatbázis-kezelő rendszer az első olyan rekordra áll, amely a feltételnek megfelel. Ha a megadott <kifejezés értéke> alapján az adatbázis-kezelő rendszer nem talál egyetlenegy rekordot sem, akkor jelzi, hogy „NO FIND". A FIND parancs alkalmazható olyan állományok esetében is, amelyek kulcsa több mezőt tartalmaz. Az összetett kulcs hátránya ebben az esetben az, hogy

az adatot csak a kulcs szerinti sorrendben kereshetjük. Azaz (a 8.1. részben létrehozott) <Irany> indexállomány esetén úgy férhetünk hozzá az adatainkhoz, hogy a FIND parancsszó után az irányítószámot, vagy az irányítószámot és a helységet, vagy pedig mind a három mező tartalmát (a címet is) beírjuk. Viszont nem férhetünk hozzá az adatainkhoz, ha csak a helységet vagy a címet keressük.

Ilyen igénynél a FIND helyett inkább az itt következő LOCATE parancsot kell kiadnunk, vagy pedig egy másik indexállományt használnunk, amelyet például a helységmező — mint első kulcs — alapján indexeltünk.

Ha valamilyen speciális tulajdonságú adatunkhoz akarunk hozzáférni, az alábbi utasításra van szükség:

^LOCATE [<érvényességi kör>] FOR <kifejezés>^

Ez a parancs akkor hatékony, ha olyan adatot keresünk az állományban, amely szerint az állományt nem indexeltük. (Például, az állomány irányítószám szerint indexelt, de minket a cím érdekel stb.)

Az egész adatbázis átvizsgálásához nem szükséges az <érvényességi kör> paraméter megadása, mivel a LOCATE parancs az első rekorddal kezdi működését. Ha az állomány egy részét akarjuk csak átvizsgálni, alakítsuk

^LOCATE Next <szám> FOR <kifejezés>^

formájúra a parancsot. A keresés az aktuális rekordtól kezdődik, és a <szám>-ban megadott számú rekord átvizsgálása után ér véget. Ha ezzel a mozgatással az állomány végét túllépnénk, a ^LOCATE^ parancs minden egyes rekordot — az aktuális pozíciótól — az állomány végéig átvizsgál. Ha karakter típusú mezőben keresünk adatot, a keresett karakterláncot a feltételben aposztrófok közé kell zárni.

Ha a rendszer talált egy olyan rekordot, amely az adott feltételeknek megfelel, azon megáll. Alapbeállításban (SET TALK ON) az aktuális rekord megnevezésével (RECORD:n) a képernyőn kijelez, így amelyik rekordra az adatbázis-kezelő rendszer rááll, azt megjeleníthetjük és tartalmát módosíthatjuk.

Ha valószínű, hogy még további rekordok is megfelelnek az adott feltételnek, a

^CONTINUE^

parancsra az adatbázis-kezelő rendszer folytatja a keresést, és ha nem talál a feltételeknek megfelelő rekordot, ki fogja jelezni, hogy

„END OF LOCATE" vagy „END OF FILE ENCOUNTERED".

A LOCATE és a CONTINUE parancs között feltételes kifejezéseket tartalmazó parancsokat is el lehet helyezni. A LOCATE parancsban levő kifejezést az adatbázis-kezelő rendszer tárolja, s így a CONTINUE azt megkapja.

MEGJEGYZÉS

A LOCATE és a CONTINUE között végrehajtandó, feltételes kifejezéseket tartalmazó parancsban a kifejezés nem lehet 128 karakternél hosszabb.

Példák

```
. use nevek
. locate for nev='EDE'
RECORD#00005                                          REKORD#00005

. display
00005 EDENYESI GABOR EDENY U. 5          EDELENY          6576

. locate for ir:szam>'3' .and. nev<'F'
RECORD#00004                                          REKORD#00004

. display nev, ir:szam
00004 DEAK GABOR      6060

. continue
RECORD#00005                                          REKORD#00005

. continue
RECORD#00006                                          REKORD#00006

. continue
END OF FILE ENCOUNTERED
```

8.3 AUTOMATIKUS SZÁMLÁLÁS ÉS ÖSSZEADÁS (COUNT, SUM)

Néhány esetben – bár nincs szükségünk arra, hogy a rekordokat ténylegesen megnézzük – szeretnénk tudni, hány olyan rekord van, amely bizonyos feltételeknek megfelel; vagy érdekel bennünket valamilyen feltételeknek megfelelő adatok összege.

Az összeszámlálás végrehajtását eredményezi a következő parancs:

$$^{\Delta}\text{COUNT}\ [\langle\text{érvényességi kör}\rangle]\ \left[\left\{\begin{matrix}\text{WHILE}\ \langle\text{kifejezés}\rangle\\ \text{FOR}\ \langle\text{kifejezés}\rangle\end{matrix}\right\}\right]\ [\text{TO}\Rightarrow$$

$$\Rightarrow\langle\text{memóriaváltozó név}\rangle]^{\Delta}$$

Ha semmilyen paramétert sem közlünk, a COUNT az adatbázisunkban levő összes rekordot megszámolja. Az <érvényességi kör> megad egy meghatározott számú rekordot. A kifejezés bármilyen összetett logikai feltétel lehet. (Lásd a kifejezésekről szóló korábbi részt, a 31. oldalon.) Az összeszámlálás eredménye egy memóriaváltozóban tárolható, amely – ha a korábbiakban nem létezett – automatikusan létrejön.

MEGJEGYZÉS

A paraméter nélküli COUNT parancs az összeszámláláskor a törlésre kijelölt rekordokat is figyelembe veszi.

Példa
(a 7.1. részben a 69. oldalon szereplő lista szerint.)

```
 . use nevek
.. count
COUNT=00011                                          ÖSSZES
 . count for 'E' $ nev
COUNT=00008                                          ÖSSZES
```

Az összegzéshez a következő parancsformátumot használjuk.

$$^{\triangle}\text{SUM}\ \left\{\begin{array}{l}\langle\text{mezőnév1}\rangle\\ \langle\text{kifejezés1}\rangle\end{array}\right\}\ \left[,\left\{\begin{array}{l}\langle\text{mezőnév2}\rangle\\ \langle\text{kifejezés2}\rangle\end{array}\right\}\right]\ \ldots\ \left[,\left\{\begin{array}{l}\langle\text{mezőnév5}\rangle\Rightarrow\\ \langle\text{kifejezés5}\rangle\end{array}\right\}\right]$$

$$\Rightarrow[\langle\text{érvényességi kör}\rangle]\ \left[\left\{\begin{array}{l}\text{FOR}\ \langle\text{kifejezés}\rangle\\ \text{WHILE}\ \langle\text{kifejezés}\rangle\end{array}\right\}\right]\qquad[\text{TO}\ \langle\text{memóriaváltozó(k)}\rangle]^{\triangle}$$

Ebben a parancsban 5 numerikus mezőig vagy kifejezésig lehet megadni feladatokat ahhoz, hogy azok szerint az adatbázisban összegzéseket végezzünk. Ha egynél több mezőt vagy kifejezést szerepeltetünk, azokat vesszővel kell elválasztanunk egymástól. Az összegzésben részt vevő rekordok számát korlátozni lehet, ha az <érvényességi kör> részt alkalmazzuk, illetve feltételt határozunk meg a FOR vagy a WHILE paraméter után.

Ha a paraméterben (vesszővel ellátva) memóriaváltozókat is megnevezünk, figyelnünk kell arra, hogy a rendszer az összegeket pozíció alapján helyezi el a memóriaváltozókba. Ha az utolsó kifejezés(ek), illetve mező(k) eredményét nem akarjuk memóriaváltozó(k)ban tárolni, csupán tudni szeretnénk, mennyi az illető adatok összege, ez nem probléma: csak azt az első néhány változót kell megnevezni, amelyre szükségünk van. Ha egy szükségtelen összeg is előfordul (például öt közül az elsőre, a harmadikra és a negyedikre van szükségünk), nevezzünk meg memóriaváltozókat az utolsó tárolni kívánt kifejezésig, illetve mezőig (példánkban négyet), majd pedig miután a SUM parancsot a rendszer végrehajtotta, a RELEASE paranccsal szüntessük meg a felesleges memóriaváltozót (ebben az esetben a másodikat).
(A példát lásd a 11.2. részben, a 243. oldalon.)

8.4 AZ ADATOK ÖSSZEGZÉSE ÉS A RÉSZLETEZÉSEK ELHAGYÁSA (TOTAL)

A TOTAL parancs a REPORT parancs részösszegző működéséhez hasonlóan dolgozik. A különbség a kettő között az, hogy itt az eredmények egy új adatbázisba kerülnek, nem úgy, mint a REPORT-ban, ahol csak kiíródnak. Ez a parancs különösen alkalmas arra, hogy a szükségtelen részeket az összegzésből kihagyjuk. Formátuma a következő:

$$^{\triangle}\text{TOTAL ON}\ \langle\text{kulcsmező}\rangle\ \text{TO}\ \langle\text{adatbázis-állomány neve}\rangle\Rightarrow$$

$$\Rightarrow[\text{FIELDS}\ \langle\text{mezőlista}\rangle]\ \left[\left\{\begin{array}{l}\text{FOR}\ \langle\text{kifejezés}\rangle\\ \text{WHILE}\ \langle\text{kifejezés}\rangle\end{array}\right\}\right]\ ^{\triangle}$$

MEGJEGYZÉS

Az adott adatbázis, amelyből az információt kapjuk, mindenképp olyan kulcsmező szerint kell, hogy sorbarendezett (SORT) vagy indexelt (INDEX) legyen, amelyet ebben a parancsban használunk.

Az új adatbázis minden egyes azonos tartalmú kulcsmezőről egy rekordon egy bejegyzést tartalmaz. (A példát lásd a 11.2. részben, a 245. oldalon.)

A TOTAL parancs minden mezőt feldolgoz, ami az adatbázisban van. Ha az új adatbázis-állomány még nem létezik, akkor oly módon, hogy a régi adatbázis szerkezetét megtartja. Azaz egy rekordot nyit egyféle kulcsmezőtartalomra, ahol az összegzésre kijelölt mezőben az azonos kulcsmezőtartalomhoz tartozó mezők tartalmának összege jelenik meg, a többi mezőbe értékelhetetlen adatokat ír. (Többnyire az összegzésben részt vevő, sorrendben első rekord mezőtartalmát.) Ezt elkerülhetjük akkor, ha az új állomány mezőinek számát korlátozzuk előbb egy

^COPY TO <állománynév> STRUCTURE FIELDS <mezőlista>^

paranccsal.

Ha a TOTAL paranccsal ezután végeztetjük el az összegzést — most már a létező állományba —, az új adatbázis csak a kijelölt mezőket fogja tartalmazni.

Ugyanez a módszer hasznos lehet más összegzések esetében is.

MEGJEGYZÉS

Mindkét esetben azzal a nagyon komoly problémával kell szembenéznünk, hogy az e műveletre kijelölt adatbázisban a mezők terjedelme meghatározott. Ebből fakadóan a TOTAL vagy a COPY parancs végrehajtása során keletkező mezők sem lesznek hosszabbak a forrásállomány mezőinél. Majdnem biztos viszont, hogy az összeg több helyiértékű lesz, mint az őt képző mezők tartalma.

Feltétlenül tanácsos a ^COPY^ parancs után a másolással keletkezett állomány szerkezetében — ^MODIFY STRUCTURE^ parancs segítségével — az összegzésre kijelölt *mezők terjedelmét* a várható szükségletnek megfelelően akár *három-négy karakterhellyel is megnövelni.*

Ellenkező esetben a mező az összegzés során nagyon könnyen túlcsordulhat, s akkor egyszerűen „0.00"-t fog tartalmazni, sőt annak ellenére, hogy hibás ez az eredmény, nem számíthatunk hibajelzésre.

8.5 KÉT ADATBÁZIS REKORDJAINAK EGYMÁSBA OLVASZTÁSA (UPDATE)

Egy adatbázisból másikba vihetünk adatokat a következő paranccsal.

^UPDATE FROM <adatbázisállomány neve> ON <kulcsmező>⇒

⇒ { ADD <mezőlista> / REPLACE <mezőlista> } ^

Mindkét adatbázist előre kell *rendezni* (sorbarendezni vagy indexelni) a megadott kulcsmező alapján, vagyis mielőtt ezt a parancsot használjuk. Mindkét állományt az első rekordra kell állí-

tani (USE vagy GO TOP paranccsal). Az UPDATE parancs hatására a két adatbázisban szereplő (azonos nevű) kulcsmezők összehasonlítására kerül sor. Ha azonos a kulcsmezők tartalma, a FROM utasításrész utáni adatbázisból származó adatok ADD paraméter esetén hozzáadódnak numerikusan a USE utasítással használatba vett állományhoz, REPLACE paraméter esetén pedig ezekkel az adatokkal *helyettesítődnek* a USE-zal használt állomány adatai a parancs mezősávjában szereplő mezőkben. Ha a kulcsmezők tartalma nem egyezik meg, akkor az adatbázis-kezelő rendszer lép egy rekordot abban az adatbázisban, ahol az ASCII kód szerinti kisebb kulcsmező-tartalmat találta.

. Ez a parancs nagyon jól alkalmazható arra a célra, hogy az adatainkat naprakészen tartsuk. (A példát lásd a 11.2. részben, a 247. oldalon.)

8.6 TELJES ADATBÁZISOK ÖSSZEKAPCSOLÁSA (JOIN)

Az adatbázis-kezelő rendszer egyik hatékony parancsa a JOIN összekapcsolási, összekötési parancs. Két adatbázissal dolgozik (az elsődleges és a másodlagos munkaterületek állományaival), s ezzel a két adatbázissal egy újabb (harmadik) adatbázist állít elő. A parancs formátuma a következő:

$$\text{JOIN TO <új állomány neve>} \left\{ \begin{array}{l} \text{WHILE <kifejezés>} \\ \text{FOR <kifejezés>} \end{array} \right\} \text{[FIELD <mezőlista>]}$$

Ha a parancs kiadásakor az elsődleges munkatérben vagyunk, a másodlagos munkatérben levő változók neve elé az „S." betűjelzést kell tennünk. Ha a második munkaterületen vagyunk, az első munkaterületről használt változók neve elé a „P." jel szükséges. Működés közben a parancs először az elsődleges munkaterület első rekordjára áll, kiértékeli a feltételt, s ha az igaznak bizonyult, a másodlagos munkaterület minden egyes rekordját kiértékeli. Minden esetben, amikor a kifejezésnek megfelelő igaz értéket talál, egy rekordot helyez el az új állományban. Amikor a második munkaterület minden egyes rekordja sorra került — vagy az elsődleges munkatér rekordjára hamisnak bizonyult a feltétel —, a rendszer továbblép az elsődleges munkatérben levő állomány következő rekordjára, majd elvégzi a kiértékeléseket. Mindez addig ismétlődik, ameddig az állományok összes rekordja sorra nem kerül.

MEGJEGYZÉS

Előfordulhat, hogy ez a műveletvégzés nagyon sok időt vesz igénybe, ha az adatbázisaink túl nagyok. Előfordulhat az is, hogy az adatbázis-kezelő a műveletvégzést nem tudja befejezni, mert a megszorítások — az összehasonlításokat illetően — túl lazák. Két, egyenként 1000 rekordos állomány egy olyan új adatbázist képes létrehozni a JOIN paranccsal, amelyben 1 000 000 rekord lenne abban az esetben, ha az egyesítési <kifejezés> részben minden igaznak találtatott. Az adatbázis-kezelő rendszer viszont legfeljebb 65 535 rekorddal tud dolgozni egy állományban.

A parancsot az alábbi parancssorozattal használjuk.

```
. use egyik
. select secondary
. use masik
. join to harmadik for P.meg:nev=meg:nev field meg:nev, ;
mennyiseg, darab:ar, cikkszam
```

Ez a parancssorozat egy <Harmadik.DBF> nevű új adatbázist állít elő, amelynek négy mezője lesz. Ezek a <Meg:Nev>, a <Mennyiseg>, a <Darab:Ar> és a <Cikkszam>. Ezeknek a mezőknek a szerkezete (adattípus, méret) ugyanaz, mint az összekapcsolandó két állományban. (A példát lásd a 11.2. részben, a 184. oldalon.)

9. Programozás, parancsállományok készítése

9.1 A PARANCSÁLLOMÁNY ÖSSZEÁLLÍTÁSA (MODIFY COMMAND <állomány>)

Mindazok a parancsok, amelyeket eddig tárgyaltunk, nagyon hatékonyak, és sok mindent megoldhatunk velük. Az adatbázis-kezelő rendszerben rejlő lehetőségeknek azonban csak kis részét bontakoztatják ki. Az adatbázis-kezelő összes előnye akkor mutatkozik meg, amikor parancsállományt írunk.

A parancsokat eddig közvetlen módban, terminálról adtuk meg az adatbázis-kezelő rendszer számára. Ez nem elég gyors módszer, sok a hibalehetőség is. Nagyobb feladatok megoldása így már igen nehézkes. A rendszeresen visszatérő feladatok megoldásához szükség van arra, hogy a parancsokat megfelelő sorrendben, egy állományba rögzítsük. Erre szolgálnak a parancsállományok. Azok a parancsok, amelyeket parancsállományban egyszer megadunk, ismételhetően, újra és újra végrehajthatók.

Amikor egy parancsállományt hozunk létre, tulajdonképpen a számítógépet programozzuk. Az adatbázis-kezelő rendszerben használatos parancsokat tartalmazó parancsállományok lényegében felhasználói programok.

Parancsállomány létrehozása szövegszerkesztő vagy szövegfeldolgozó program segítségével lehetséges, amelybe a parancsainkat beírjuk. Ennek a CP/M rendszerbe tartozó állománynak a neve után <.CMD> állománynév-kiterjesztést kell tenni. Az adatbázis-kezelő rendszer a parancsállományban levő lista elején kezdi a parancsok feldolgozását, és folytatja mindaddig, amíg a listán végig nem haladt. BASIC-ben például ez a sorrend nagyon jól látható, mert a programsorok számozottak. Egyes nyelvekben (a dBASE II és a PROP-BASE is ide tartozik) a sorrend nem látható ilyen jól, de a számítógép a parancsokat az első sorral kezdve, majd a másodikkal folytatva, és így tovább, sorról sorra hajtja végre. Más nyelvek az egyes parancsok között elválasztókat (például kettőspontot) használnak. Az adatbázis-kezelő rendszer egyszerűen csak egy kocsivissza (carrige return, <CR>) kódot használ, hogy a parancsot lezárja. Csak abban az esetben szűnik meg a parancsok szekvenciális végrehajtása, ha külön parancs szól arról, hogy mást kell tenni. Ez rendszerint valamilyen feltételek alapján következik be: a számítógépnek egy kifejezés, feltétel alapján döntenie kell. Ezt a kifejezést vagy feltételt a parancsállományba kell helyeznünk. (Minderről a későbbiekben többet fogunk még mondani.)

Először is állítsunk elő egy parancsállományt, amelynek a <Teszt> állománynevet adjuk. Mint mondottuk, ezt egy szövegszerkesztő vagy szövegfeldolgozó program segítségével tehetjük meg, de egy sokkal egyszerűbb lehetőségünk is van az adatbázis-kezelő rendszerben. Éspedig a

^MODIFY COMMAND Teszt^

parancs hatására az A adatbázis-kezelő egy üres képernyőt ad elénk. Erre írhatjuk a parancsain-

kat a korábban bemutatott szerkesztési rendszer eszközeivel (4.6. rész). Használjuk ezeket az eszközöket, és gépeljük be a következő példát. (A sor vége a parancs végét is jelenti, hacsak nem használjuk a parancs új sorban folytatására lehetőséget adó pontosvesszőt.)

Példa

```
USE Nevek
COPY TO Atmeneti Structure FIELDS Nev, IrSzam
USE Atmeneti
APPEND FROM Nevek
COUNT FOR Nev='G' TO G
DISPLAY MEMORY
? 'Ezzel az elso parancsallomannyal el is keszultunk'
```

Amikor készen vagyunk, használjuk a ^*W (ETB)* kódot, hogy az adatbázis-kezelő az alaprendszerhez térjen vissza.

Ha ugyanúgy írtuk meg a parancsokat, mint ahogy a példában szerepel, felhasználói programunk nem fog működni. Gépeljük be még egyszer, hogy ^MODIFY COMMAND Teszt^, és szúrjuk be a kettőspontot, hogy kijavítsuk az <Ir:Szam> nevű mezőt.

Ha nagyobb parancsállományokat írunk majd, rá fogunk jönni, hogy a beépített szövegszerkesztő segítségével beírhatunk, javíthatunk, korrigálhatunk, azaz változtathatunk a parancsállományokon anélkül, hogy visszamennénk a rendszerszintre. Jelenleg ez a beépített szövegszerkesztő körülbelül 4000 sort képes feldolgozni, ezért ennél nagyobb parancsállományokat más szövegszerkesztővel állítsunk össze.

A parancsállomány önmagában természetes, de mégis érdekes, hogy egy parancssorozatot milyen egyszerűen lehet végrehajtani egyetlenegy kiadott parancs segítségével. Ez hasonló ahhoz, ahogy a CP/M rendszerben a .COM vagy .SUB állománynév-kiterjesztésű állományokat használjuk.

Ha CP/M operációs rendszerben dolgozunk, a kész parancsállományt lefuttathatjuk, ha a következőket adjuk meg:

```
A>^DBASE teszt^
```

illetve

```
A>^PROPBASE teszt^
```

Ha már az adatbázis-kezelő rendszerben vagyunk, a bejelentkező pontot láthatjuk, és egyszerűen azt kell megadnunk, hogy

```
. do teszt
```

9.2 DÖNTÉSEK ÉS VÁLASZTÁSOK (IF...ELSE...ENDIF), (DO CASE...OTHERWISE...ENDCASE)

A felhasználói programokban gyakori igény, hogy feltételektől függően lehessen végrehajtani a parancssorozatokat. Az adatbázis-kezelő rendszerben a feltételes végrehajtás megvalósítására két szerkezet is rendelkezésre áll.

Az IF. .ELSE (HA. .KÜLÖNBEN) szerkezetben az IF parancsszóhoz tartozó kifejezés logikai eredményétől függ, hogy az IF parancs és az azt követő ELSE parancsszó, vagy az ELSE és az azt követő ENDIF parancsszavak közé írt parancs(ok) végrehajtására kerül-e sor. Az IF szerkezetet az ENDIF parancsszó zárja le.

A DO CASE szerkezetben több feltételt is megadhatunk a DO CASE parancsot követő, tetszőleges számú CASE (ESET) parancsszóhoz tartozó kifejezésben. Abban a CASE ágban szereplő parancs(ok) végrehajtása történik meg, amelynél – egy DO CASE szerkezeten belül – a kifejezés sorrendben először ad logikai igaz értéket. Az utolsó CASE ág után lehet egy OTHERWISE (EGYÉBKÉNT) parancsszó. Az OTHERWISE ágban levő parancs(ok) akkor érvényesülnek, ha egyetlen CASE ág kifejezése sem adott logikai igaz értéket. Minden CASE ág a következő CASE, OTHERWISE vagy ENDCASE parancsszóig, az OTHERWISE ág az ENDCASE parancsszóig tart. A DO CASE szerkezetet az ENDCASE parancsszó zárja le.

9.2.1 Egyszerű döntés

Ha csak egyszerű döntést akarunk hozni, az IF szerkezet a megfelelő. Az ELSE ágat elhagyhatjuk, és a következő formátumot használjuk.

```
IF <feltétel> [.AND. <felt2> .OR. <felt3> . . . . .]
  <parancs>
  [<parancs2>]
  [<. . . . . >]
ENDIF
```

A feltétel logikailag igazra vagy hamisra értékelhető kifejezések sorozata lehet (254 karakter hosszúságig). A <koltseg.DBF> nevű állományunkat tekintve a következőt kérhetjük: ha a darabár kisebb, mint 70 (Ft), és a mennyiség kisebb, mint 20 (db), akkor készítsen az adatbáziskezelő rendszer – a <Rendelo.CMD> parancsállomány aktivizálásával – egy rendelést, és ezt a döntést a képernyőn is tudassa.

```
USE Koltseg.dbf
IF Darab:Ar<70 .AND. Mennyiseg<20
  ? 'Rendelek meg'!'
  do Rendelo.cmd
ENDIF
```

Ha minden feltétel logikai igazra értékelődik, a számítógép sorra végrehajtja azokat a parancsokat, amelyek az IF és az ENDIF között vannak, majd ezt követően az ENDIF utáni parancsot veszi sorra. Ha a feltételek logikai hamisra értékelődnek, egy ugrás következik az ENDIF mögötti első parancsra.

9.2.2 Kettős választás

Olyankor, amikor a feltételek logikai igaz, illetve hamis voltától függően kell egy-egy parancssorozatot végrehajtani, az IF. .ELSE szerkezetet használjuk:

```
IF <feltétel(ek)>
   <parancs(ok)>
ELSE
   <parancs(ok)>
ENDIF
```

A feltétel(ek) logikai igaz eredménye esetén a számítógép az IF ágban, logikai hamis eredménye esetén pedig az ELSE ágban levő parancsokat hajtja végre. A kiválasztott ágban szereplő parancsok végrehajtása után az ENDIF-et követő paranccsal folytatódik a parancsok végrehajtása.

9.2.3 Kettőnél több választás

Ha a feltételektől függően kettőnél több parancssorozatot akarunk elvégeztetni, a DO CASE szerkezet lesz a megfelelő:

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

Ebben a szerkezetben a számítógép a feltételek logikai értékétől függően választja ki valamelyik ágban a feladatot. Ha egy CASE ágat végrehajtott, akkor a következők — függetlenül attól, hogy a feltétel logikai értéke igaz-e — nem kerülnek sorra. Ha egyik CASE ág feltétele sem értékelődik logikai igazra, akkor az OTHERWISE ágban megfogalmazott feladat végrehajtására kerül sor. Ezután a vezérlés az ENDCASE utáni sorra ugrik.

Az OTHERWISE ág elhagyható. Ilyenkor is az első logikai igaznak talált CASE ágban hajtja végre a számítógép a parancsokat, majd az ENDCASE-t követő sorban folytatja a munkát, illetve ha egyik CASE ágnál sem teljesülnek a feltételek, akkor is az ENDCASE utáni soron folytatódik a végrehajtás.

9.2.4 Többszörös választási lehetőség

Gyakran olyan döntést kell hoznunk, amely több feltételsor kombinált eredményétől függ. Ebben az esetben az IF. .ELSE. .IF szerkezetet kell alkalmaznunk. Ez ugyanaz, mint az IF. .ELSE szerkezet, de most több egymásba illesztett szinten alkalmazzuk:

```
IF <feltétel(ek)1>
   <parancs(ok)1>
   IF <feltétel(ek)2>
      <parancs(ok)2>
   ENDIF 2
```

```
ELSE
   IF <feltétel(ek)3>
      <parancs(ok)3>
   ELSE
      IF <feltétel(ek)4>
         <parancs(ok)4>
      ELSE
      ENDIF 4
   ENDIF 3
ENDIF 1
```

A szintezés szükséges mélysége lehetővé teszi, hogy megvalósítsuk a helyes döntést.

9.2.5 Döntésekkel és választásokkal kapcsolatos programozási szabályok

a) Minden egyes IF-nek egy hozzá tartozó ENDIF-et; és minden egyes DO CASE-nek egy hozzá tartozó ENDCASE-t kell tartalmaznia, méghozzá úgy, hogy egy szerkezet lezárása előtt a benne megkezdett összes szerkezetet le kell zárni, illetve egy szerkezeten belül csak az abban megkezdett szerkezetek zárhatók le.

b) Az adatbázis-kezelő rendszer az ENDIF sorának ENDIF utáni részét nem olvassa be, ezért az ENDIF után különböző megjegyzéseket tehetünk, amelyek megkönnyíthetik a felhasználói program logikai felépítésének követését.

c) IF szerkezet bármelyik ágába lehet akár IF, akár DO CASE szerkezetet ágyazni. DO CASE szerkezet bármelyik ágába lehet IF szerkezetet ágyazni.

d) DO CASE szerkezetben sehol sem szabad DO CASE szerkezetet elhelyezni.

9.3 A PARANCSSOROZAT ISMÉTLÉSE (DO WHILE . . . LOOP . . . ENDDO)

A dBASE II és a PROP-BASE adatbázis-kezelő rendszerekben az ismétlést a DO WHILE szerkezet valósítja meg. Mivel a szerkezetben levő parancsok végrehajtása az utolsó parancs végrehajtása után többször is újra kezdődhet, a ciklikus végrehajtás miatt az ilyen szerkezeteket ciklusoknak is nevezhetjük.

A DO WHILE szerkezet felépítése a következő:

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

Amikor a feltétel(ek)ben szereplő kifejezés logikai igaz eredményt ad, a ciklusban szereplő parancsok végrehajtásra kerülnek.

MEGJEGYZÉS

Ne felejtsük el, hogy a DO WHILE parancs és az ENDDO parancsszó között levő parancsok valamelyikének hamisra kell változtatnia a feltételben szereplő kifejezés logikai értékét, különben az ismétlés örökké folytatódna.

Ha szeretnénk megszámolni, hogy egy tevékenység hányszor ismétlődik, a következő megoldást is lehet alkalmazni:

```
STORE 1 TO Szamolo                  — Az ismétlésszámláló (a „szamolo" nevű memóriaváltozó) 1-re állítása
DO WHILE Szamolo < 11               — A tevékenység 10-szeri elvégzésének ellenőrzése a DO WHILE szerkezet elején, a „szamolo" nevű memóriaváltozónak összehasonlító műveletben való szerepeltetésével.
                                      Ha a feltétel logikai igaz eredményt ad, a szerkezetben levő első parancsra, ellenkező esetben az ENDDO utáni parancsra kerül a vezérlés.
   IF Nev = '     '                 — Ha nincs adat,
      SKIP                          — lép egy rekordot.
      LOOP                          — Vissza a DO WHILE-hoz.
   ENDIF
   DO Eljaras1.cmd                  — A parancsállomány végrehajtása.
   STORE Szamolo+1 TO Szamolo       — Az ismétlésszámláló növelése.
ENDDO                               — Vissza a DO WHILE-hoz.
```

Ebben a példában, ha van adat a <Nev> mezőben, az adatbázis-kezelő rendszer azokat a parancsokat hajtatja végre, amelyeket az <Eljaras1.CMD> nevű parancsállományban írtunk le. Az <Eljaras1.CMD> végének elérésekor a vezérlés visszatér a „DO Eljaras1" parancs után levő sorba. Ezt követően a „Szamolo" nevű változó értékét eggyel növeli, majd egy vizsgálatot végez, hogy annak értéke kisebb-e, mint 11. Ha igen, akkor újra végrehajtja a DO WHILE szerkezetben foglaltakat. Amikor már tízszer ismétlődött a ciklus (a számláló értéke 11), az ENDDO utáni parancssorra kerül a vezérlés.

A LOOP parancsszó hatására a végrehajtási sorrend megváltozik: annak a DO WHILE ciklusnak az elején folytatódik, amely ezt a parancsszót tartalmazza.

Példánkban, ha a <Nev> nevű mező üres, az illető rekordot nem kell feldolgozni, ezért a LOOP parancsszó után a vezérlés visszatér a „DO WHILE Szamolo < 11" sorra. Az üres rekord a számláló tartalmát nem módosítja, mivel ilyen esetben kikerüljük azt a parancsot, amely a számláló tartalmát növeli.

9.4 PARANCSÁLLOMÁNYOK AKTIVIZÁLÁSA (DO <állomány>)

Egy számítógépi nyelvnek az a képessége, hogy algoritmusokat, feltételek szerint akár többször is alkalmazható programrészleteket, eljárásokat lehet segítségével létrehozni, nagymértékben egyszerűsíti a számítógépek programozását. A BASIC nyelvben ezeket az eljárásokat szubrutinoknak hívjuk. A Pascal és PL/1 programnyelvekben eljárás a nevük. Az adatbázis-kezelő

rendszerben eljárásnak is nevezhető *parancsállományokat* használunk erre a célra, amelyek a következő paranccsal hozhatók működésbe:

^DO <parancsállomány neve>[.CMD]^

Az előző oldalon levő példában egy parancsállományt akkor aktivizáltunk, amikor a „DO Eljaras1.cmd" parancsot adtuk, de adhattuk volna a „DO Eljaras1" parancsot is. Az Eljaras1-nek mint parancsállománynak a tartalma például a következő lehet:

```
IF Varos = 'BUDAPEST'
  DO Fovaros.cmd
ELSE
  IF Varos = 'HATVAN'
    DO Alfold.cmd
  ELSE
    IF Varos = 'SOPRON'
      DO Dunantul.cmd
    ELSE
      DO Azonosito.cmd
    ENDIF
  ENDIF
ENDIF
RETURN
```

Dolgozhatunk olyan parancsállományokkal is, amelyek önmaguk is más parancsállományokat aktivizálnak. Az adatbázis-kezelő rendszerben egyszerre 16 tetszőleges típusú állomány lehet nyitott. Ha például egy adatbázis-állomány USE paranccsal használatban van, 15 másik tetszőleges állományt nyithatunk meg mellette, figyelembe véve azt, hogy néhány parancs kiegészítő állományokat igényel. (A REPORT, az INSERT, a COPY, a SAVE, a RESTORE és a PACK parancs egyet, a SORT kettőt.) Mindez a programozási szabadság gyengítésének tűnhet, de valójában az állományok száma tulajdonképpen korlátlan, mert ha a használt állományokat lezárjuk, újabbakat nyithatunk meg. Az a lényeg, hogy egyidőben legfeljebb 16 legyen nyitva. Egy parancsállomány akkor van lezárva, ha a vezérlés elérte a parancsállomány végét, vagy pedig RETURN parancsot kellett végrehajtani.

A RETURN parancs visszaadja a vezérlést annak a parancsállománynak, amelyikből az őt tartalmazó parancsállományt aktivizálták, vagy pedig a közvetlen módhoz, ha az állomány aktivizálása onnan történt. A RETURN parancs nem feltétlenül szükséges ahhoz, hogy a vezérlés az állomány végét elérve az aktivizáló állományhoz térjen vissza.

9.5 ADATOK INTERAKTÍV BEVITELE FUTÁS ALATT (WAIT, INPUT, ACCEPT)

Számtalan esetben a gépkezelőtől kérnek a parancsállományok adatokat. A következő parancsok teszik az ilyen adatbevitelt lehetővé.

^WAIT [TO <memóriaváltozó>]^

A parancs felfüggeszti a parancsállomány végrehajtását, és egy *karaktert* vár a billentyűzetről, miközben a képernyőn „WAITING" („VÁROK") felirat jelenik meg. A végrehajtás egy tetszőleges billentyű lenyomása után folytatódik. Ha egy memóriaváltozót is megadtunk, akkor a karakter abban tárolódik is. Ha a karakter nem jeleníthető meg (pl. <CR>, <enter>, kontrolkarakter stb.), egy üres szóköz kerül a változóba.

^INPUT ['<üzenet>'] TO <memóriaváltozó>^

A parancs — a parancsállomány végrehajtását felfüggesztve — *bármilyen típusú* adatot elfogad a billentyűzetről, és azt a megadott nevű memóriaváltozóba helyezi. Amennyiben a memóriaváltozó nem létezik még, akkor előállítja. Ha a szabadon választható <üzenet> részt is közöljük, a képernyőn ez az üzenet megjelenik egy kettősponttal a végén, jelezve, hogy a gép a kívánt adatot várja. A változó adattípusa (karakter, numerikus vagy logikai) az éppen megadott adat típusa szerint állítódik be. *Karakterláncokat* idézőjelek vagy szögletes zárójelek közé *kell* zárni. (Példa a 179. oldalon a 11.2. részben.)

^ACCEPT ['<üzenet>'] TO <memóriaváltozó>^

Ez a parancs — a parancsállomány végrehajtását felfüggesztve — *karakter típusú* adatot vár, anélkül, hogy a karaktertípus esetében szükséges határolókat meg kéne adni. Ez nagyon hasznos hosszú karakterláncok bevitelénél.

Tanácsok az alkalmazáshoz

WAIT
abban az esetben előnyös, ha gyors adatbevitelt akarunk megvalósítani, de ne használjuk olyankor, amikor egy rossz adatbevitel súlyos károsodást okozhat adatbázisunkban.

ACCEPT
nagyon hasznos hosszú karakterláncok beviteléhez, mivel nem igényel karakterlánc-határolókat. Kényelmes akkor is, amikor csak egy karaktert akarunk bevinni.

INPUT
segítségével a rendszer akár numerikus, akár logikai, akár karakter típusú adatokat is fogad.

9.6 MEGJEGYZÉSEK A PARANCSÁLLOMÁNYBAN (REMARK, NOTE, *)

Sokszor lesz szükségünk a parancsállományokban arra, hogy üzeneteket helyezzünk el akár a magunk, akár egy másik programozó, akár a felhasználó számára. Üzeneteket a 6.4-es és a 9.5-ös részben ismertetett parancsok segítségével is lehet küldeni, és mint majd a későbbiekben kiderül (lásd a 10. fejezetet), számos más lehetőségünk is nyílik erre. Azok a parancsok azonban az üzeneten kívül más paramétereket is tartalmaznak, illetve tartalmazhatnak, némelyek közülük pedig felhasználói tevékenységet is igényelnek.

Az adatbázis-kezelő rendszer azonban a közvetlen üzenetküldést is támogatja. A

^NOTE [<karakterlánc>]^

parancs olyan üzenet elhelyezését teszi lehetővé a parancsállományban, amely nem jelenik meg egyik kimeneti eszközön sem. A NOTE helyett választhatjuk a

^* [<karakterlánc>]^ parancsot is.

Ezekkel a parancsokkal (NOTE, *) megjegyzéseket szúrhatunk be a parancsállományba, például annak könnyebb áttekinthetősége érdekében. A parancs a természetéből fakadóan alkalmat ad arra is, hogy ideiglenesen nem használt parancsaink elejére írva, azokat üzenetté minősítve, mintegy kiiktassa a felhasználói programból. Így nem kell elvesztenünk ezeket a sorokat, és a későbbiekben a * törlésével aktívvá tehetjük őket.

Ha viszont olyan üzenetet akarunk elhelyezni, amely a felhasználónak (is) szól, akkor részesítsük előnyben a

^REMARK <karakterlánc>^ parancsot.

(V. ö. 210. old., 11.2.rész, REMARK)

MEGJEGYZÉS

A felsorolt három parancsnál a karakterláncot nem kell az egyébként szokásos karakter-határolók közé tenni.

9.7 KILÉPÉS A PARANCSÁLLOMÁNYBÓL; AZ ADATBÁZIS-KEZELŐ RENDSZER ALAPÁLLAPOTBA ÁLLÍTÁSA (RETURN, CANCEL, CLEAR, QUIT)

A parancsállomány futtatását — annak belsejéből is — többféle módon lehet megszakítani, befejezni. A

^RETURN^

parancs hatására a vezérlés visszatér oda, ahonnan a parancsállományt ^DO <parancsállomány neve>^ paranccsal aktivizáltuk. Tehát vagy egy parancsállománynak a hívást követő sorába, vagy közvetlen módba, a billentyűzetre. A parancs az őt tartalmazó parancsállományt lezárja. A

^CANCEL^

parancsra véglegesen abbahagyja az adatbázis-kezelő rendszer a parancsállomány végrehajtását, lezárja azt, és visszatér közvetlen módba a billentyűzethez. Megjelenik a jellegzetes pont a képernyőn. A

^CLEAR^

a dBASE II-t és a PROP-BASE-t alapállapotba hozza, a ^USE^-zal meghívott összes adatbázis-

állományt lezárja, törli a memóriaváltozókat, újra kiválasztja az elsődleges munkaterületet. A

^QUIT^

parancs (részletesen lásd a 3.5. részben) lezárja az összes állományt, és visszaadja a vezérlést az operációs rendszernek (vagy azon keresztül más programoknak).

9.8 KAPCSOLAT A SZÁMÍTÓGÉP MEMÓRIÁJÁVAL (POKE, CALL)

Előfordulhat, hogy kapcsolatot kívánunk teremteni a számítógép memóriájával abból a célból, hogy információkat helyezzünk el ott, illetve, hogy a számítógép memóriájában elhelyezett információkhoz az adatbázis-kezelő rendszerből hozzájussunk. Lehetőség nyílik erre az adatok és címek megadásával decimális formában, a

^POKE <cím>, <byte lista>^

parancs segítségével. A POKE parancs a megadott memóriacímtől kezdődően elhelyezi a <byte lista> elemeit a számítógép memóriájában. A

^POKE 3500,0,1,10,11,12,13,32^

parancs a 3500-as memóriacímtől kezdve sorban elhelyezi a szintén decimális formában megadott értékeket (a 0-val kezdve) a számítógép memóriájában.

A számítógép memóriájában levő értékekhez a PEEK beépített függvény segítségével férhetünk hozzá, amely megadja, hogy a kért memóriacímen milyen értéket találhatunk. Természetesen a PEEK függvény a POKE parancshoz hasonlóan decimális címeket és értékeket használ. A

^? PEEK(3500)^

parancs a kimeneti egységre írja a 3500-as memóriacím tartalmát, a

^STORE CHR(PEEK(3506)) TO Szok^

parancs a „Szok" nevű memóriaváltozóban tárolja a 3506-os memóriacímen levő értéket ASCII kódban.

Lehetőség van arra is, hogy gépi kódú programokat hajtsunk végre úgy, hogy közben nem hagyjuk el az adatbázis-kezelő rendszert. A

^CALL [<memóriaváltozó>]^

parancs hatására a ^SET CALL <cím>^ parancs (lásd 3.3. rész) <cím> paraméterében megadott helyen kezdődő gépi kódú programra kerül a vezérlés. (A címet itt is decimális értékként kell megadni.)

A megadott cím elérésekor a H–L regiszterpár a memóriaváltozó első byte-jára, a hossz byte-ra mutat. A memóriaváltozónak karakterláncnak kell lennie, és nem lehet hosszabb, mint amekkora adat számára van hely.

A gépi kódú programból való visszatérés egy gépi kódú *return* végrehajtásával történik.

Az A400H feletti címtartomány a CP/M BDOS-ig rendelkezésre áll, de vegyük figyelembe, hogy a SORT parancs is ezen a területen dolgozik.

9.9 TANÁCSOK A PARANCSÁLLOMÁNYOK MEGÍRÁSÁHOZ

A dBASE II és PROP-BASE – lévén igazi adatbázis-kezelő rendszerek – számos parancsot és technikai megoldást alkalmaznak, amelyekkel sokkal több információt (és sokkal könnyebben) kaphatunk meg az adatbázisunkból, mint más – jelenleg mikroszámítógépeken rendelkezésre álló – állománykezelő rendszerekkel.

Ahhoz, hogy az adatbázis-kezelő rendszerben működő felhasználói programok szerkesztését megtanuljuk, célszerű a következő példákat átnéznünk, illetve alkalmaznunk saját igényeinknek megfelelő adatokkal.

Elsőként a ^CREATE^ paranccsal hoztunk létre adatbázist. A <Nevek.DBF> adatbázis-állományt többször módosítottuk, mezőneveket és méreteket változtattunk – adatokkal és adatok nélkül is.

Adatbázisaink szerkezetét ellenőrizhetjük. Célszerű az adatbázisok mezőneveit, a mezők típusát, hosszát a lehetőségekhez képest azonosra tervezni, hogy az állományok egybeolvasztása és más parancsok könnyen használhatók legyenek. Az egyik adatbázisból származó adatok a másik adatbázis – megegyező vagy más nevű – mezőibe beilleszthetők, de közös nevek használatával egyszerűbb az adatátmásolás.

A parancsállományok megírásakor – a példánkban is – a következő módszert használtuk:

– felmértük és körvonalaztuk a feladatot,
– konkrétan megfogalmaztuk a feladat megoldását,
– kidolgoztuk az algoritmusokat,
– az algoritmusokat az adatbázis-kezelő parancsainak alkalmazásával programoztuk.

Amikor valamilyen részprobléma felvetődik, de nem tudjuk azonnal megoldani, csak egy parancsállomány nevét adjuk meg, amelyet azután később dolgozunk ki. Tartsuk szem előtt, hogy az elnevezések az algoritmus-megfogalmazás szintjéről úgy kerüljenek át a parancsállományokba, hogy 10 karakterben elférjenek, de egyértelmű jelentésüket megtartsák.

A következő példában szereplő parancsállomány futtatására két lehetőség van: CP/M szintről ^dbase fizetes^, illetve ^propbase fizetes^; adatbázis-kezelőben ^do fizetes^ parancs megadásával. A példa összefoglalja az eddig tanult ismeretek nagy részét, és bőséges magyarázatokat ad.

Ezt a kis felhasználói programot újra és újra futtathatjuk, kipróbálhatjuk az összes változatot, majd rossz válaszok megadásával is próbálkozhatunk. Így megismerhetjük működését, és azt, hogy az adatbázis-kezelő rendszer hogyan kezeli a hibákat.

```
********************  FIZETES.CMD  ********************
*       Ez a parancsállomány üzenetet ad a felhasználónak a
* képernyőre, es választ vár tőle. Az adatot egy memóriaváltozóba olvassa be, majd a kiválasztott feladatot végrehajtja.
```

```
*       Ez a parancsállomány egy felhasználói program rész-
* lete, de müködőképes. A menüben megadott  feladatok végre-
* hajtására hivatott  parancsállományok nincsenek kidolgoz-
* va,  de néhány  müveletet elvégeztetünk,  amelyből megtud-
* hatjuk, hogy mi is történik.
*       Alap müködési módban az adatbázis-kezelő a képernyő-
* re irja az egyes parancsok végrehajtásának eredményeit. Ez
* zavaró lehet,  ezért SET TALK OFF parancsot adunk,  hogy a
* feliratok ne jelenjenek meg.
SET TALK OFF
USE KiFizetes.dbf
ERASE
*       Jó szokás letörölni a képernyőt mielőtt bármilyen uj
* adatot irnánk rá.
*       A ^?^ parancs alkalmazható az információk képernyőre
* irására:
?
?
?
?
?'              PENZTARI KIFIZETESEK MENUJE'
?
?
?'              0 = Kilepek'
?'              1 = Kifizetheto osszegek'
?'              2 = Uj szamlak bevitele'
?'              3 = Fizetesek nyugtazasa'
?
?'              Kerem valasszon'
WAIT TO Valtozat
ERASE
*       Mivel a három választási lehetőséget még nem dolgoz-
* tuk ki, a számitógéppel különböző magyarázatokat iratunk
* ki, attól függően, hogy a menüből mit választottunk.
IF Valtozat = '1'
* A @ jelü parancs leirását lásd a következő részben.
  @ 0,20 SAY 'Egy'
ELSE
  IF Valtozat = '2'
    @ 1,20 SAY 'Ketto'
```

```
  ELSE
    IF Valtozat = '3'
      @ 2,20 SAY 'Harom'
    ELSE
      @ 7,20
      @ 8,20 SAY 'Az 1, 2 vagy 3 karaktereken kivul'
      @ 9,20 SAY 'megadott barmely karakter hatasara'
      @ 10,20 SAY 'ennek az uzenetnek a kiirasa  utan'
      @ 11,20 SAY 'a parancsallomany befejezi mukodeset.'
      @ 12,20 SAY 'Megjegyzendo, hogy a szamjegyeknek,'
      @ 13,20 SAY 'amelyek az IF szerkezetben vannak,'
      @ 14,20 SAY 'idezojelek kozott kell lenniuk, mert'
      @ 15,20 SAY 'a WAIT parancs csak karaktertipusu'
      @ 16,20 SAY 'adatot fogad el.'
      @ 17,20 SAY '
    ENDIF 3
  ENDIF 2
ENDIF 1
*       Minden IF-hez egy-egy ENDIF-nek kell tartoznia.
* Az ENDIF-ek után cimkéket tettünk, amelyek azt jelzik,
* hogy melyik ENDIF melyik IF-hez tartozik.
?
?
?
?
INPUT 'Akarja folytatni (Igen vagy Nem)?' TO Folytatas
ERASE
IF Folytatas
  INPUT "Adjon egy szamot." TO Szam
ELSE
  @ 10,20 SAY "  MIERT NEM?  "
  WAIT
ENDIF
ERASE
@ 10,20 SAY " NEM  ALL  MODOMBAN  FOLYTATNI. "
@ 12,20 SAY "  V I S Z O N T L A T A S R A. "
* A következő  DO WHILE  szerkezet  arra  szolgál,  hogy  az
* utolsó  üzenetet  a képernyő törlése előtt  egy  ideig  a
* képernyőn  tartsa,  hogy  el lehessen olvasni.  Ön  is
* használhatja ezt a megoldást parancsállományaiban. A fel-
```

```
* irat képernyőn való tartásának ideje a felső  határ (100),
* vagy a lépésköz (+1) módosításával változtatható meg.
STORE 1 TO X
DO WHILE X < 100
  STORE X + 1 TO X
ENDDO
ERASE
RETURN
********************  FIZETES.CMD  vege  **********************
```

Az imént használt „pénztári kifizetések menüje", amely a <Fizetes.CMD> parancsállományban található, bővíthető egy újabb választási lehetőséggel. Legyen ez „4 = Számlák írása".

Első lépésként kérjük a dátum megadását. Az alábbi parancssorokban az MDatum nevű változóba olvassuk be a dátumot, majd ellenőrizzük, hogy helyes-e.

```
ERASE
SET TALK OFF
STORE "          " TO MDatum
STORE T TO RosszDatum
DO WHILE RosszDatum
* A @ jelü és a READ parancsok leirását lásd a következő
* részben.
  @ 5,5 SAY "Mai datum EE/HH/NN" GET MDatum PICTURE;
  "99/99/99"
  READ
  IF VAL($(MDatum,1,2)) < 1   .OR. VAL($(MDatum,1,2)) > 12;
  .OR. VAL($(MDatum,4,2)) < 1   .OR. VAL($(MDatum,4,2)) > 31;
  .OR. VAL($(MDatum,7,2)) <> 85
    STORE "          " TO MDatum
    @  7,5  SAY "**HIBAS DATUM,  KEREM  MEGISMETELNI.**"
    STORE T TO RosszDatum
  ELSE
    STORE F TO RosszDatum
  ENDIF
ENDDO
ERASE
```

Az adatbázis-kezelő rendszer, a parancsállomány tartalmának megfelelően, először üres helyekkel tölti fel az MDatum változót, majd a @...SAY...GET...PICTURE parancs hatására (10.1. rész) a képernyőn megjelenik a

```
Mai datum EE/HH/NN:  /  /  :
```

felirat.

A dátum megadása után az IF paranccsal ellenőrizzük, hogy a hónap 1 és 12, a nap 1 és 31 közé esik-e, az év egyenlő-e 85-tel. Ez három lépésben valósul meg.

– A $ (részkarakterlánc) beépített függvény kiválasztja az ellenőrizendő 2–2 karaktert,
– a VAL beépített függvény egész számmá alakítja,
– majd ezt összehasonlítjuk a még megengedett értékkel.

Ha a megengedetten kívüli értékeket talál a program, az MDatum ismét üres karakterekkel töltődik fel, és a képernyőn hibajelzést kapunk. Ha az ellenőrzés során az értékek a megadott határokon belül vannak, a felhasználói program végrehajtása folytatódik.

Következő feladatunk a tényleges számlakiírás. Az általunk használt számlák formáját figyelembe véve a parancsok a következők.

```
@ 8,3 SAY Osszeg
* Ez karakter tipusu változó, amely az összeget tartalmazza.
@ 11,38 SAY Vevo:Szama
@ 11,50 SAY MDatum
@ 11,65 SAY Fogy:Ar
@ 13,10 SAY Meg:Nev
@ 14,10 SAY Cim
@ 15,10 SAY Helyseg
@ 15,35 SAY Ir:Szam
@ 17,10 SAY Kod
* A változókat egy másik eljárás  tölti fel,
* itt  értéket adhatunk nekik egy-egy STORE-ral.
```

Nyomtatás előtt ellenőrizhetjük a fenti felhasználói program működését (a kiírandókat először a képernyőre küldve), majd a SET parancs segítségével a SCREEN-ről PRINT módra áttérve nyomtassunk ki egy számlát. (A változók értékadása egy másik parancsállományban van.)

Egy *nyomtatott oldal* legfeljebb *255 sort* tartalmazhat. A sorszámláló és a nyomtató alapállapotba hozatala PRINT módban küldött ^EJECT^ paranccsal lehetséges.

Az előbbi példák jól szemléltettek néhány lényeges dolgot:

1. Jó szokás a képernyő gyakori törlése az ERASE parancs alkalmazásával.
2. A kis- és nagybetűk használata áttekinthetőbbé teszi a programot. Az adatbázis-kezelő rendszer nagybetűnek tekint minden betűt, de az emberi szem számára előnyös a kétféle betű használata.
3. A ? jelű parancs üres sorok és szövegek képernyőre írására alkalmas.
4. A WAIT parancs megállítja a parancssorozat végrehajtását, és egy karaktert vár a billentyűzetről. Ezt a várt jelet szigorúan karakterként kell kezelni. Így tettük ezt mi is az IF szerkezetben, ahol a feltételrészben a kódot aposztrófok közé zártuk, jelezve, hogy karaktert várunk.
5. Az INPUT parancs bármilyen adattípust elfogad, de a karaktereket és karakterláncokat idézőjelbe, aposztrófok vagy szögletes zárójelek közé kell tenni. Ha az üzenet aposztrófot tartalmaz, idézőjelek vagy szögletes zárójelek használandók határolójelként.
6. A változókat nem kell előre definiálni. Ha egy újabb változóra van szükségünk, csak új változónevet kell használnunk (egyidejűleg legfeljebb 64 aktív változó lehet).

7. A logikai változók egyszerűen kezelhetők.
   Az IF Folytatas parancs jelentése IF Folytatas = T
   (azaz: ha a Folytatas nevű logikai típusú memóriaváltozó aktuális értéke logikai igaz).
8. A parancsállomány végén a RETURN nem feltétlenül szükséges; itt azért áll, mert a parancsállományt úgy használjuk, mintha egy másik parancsállomány aktivizálta volna. Így ismeri fel az adatbázis-kezelő rendszer, hogy abba a parancsállományba kell vissztérnie, amely ennek a működését indította.

10. Az adatbevitel és az adatszolgáltatás formátumának megtervezése

10.1 SZERKESZTÉS ÉS FORMÁZÁS A KÉPERNYŐN (SET FORMAT TO SCREEN) (@. .SAY. .GET. .PICTURE. .READ) (ERASE, CLEAR GETS)

A ?, az ACCEPT, az INPUT és a REMARK parancs alkalmas arra, hogy üzeneteket írjunk ki velük a képernyőre a felhasználó számára. Mindegyiknek hátránya azonban, hogy az üzenet a képernyőn az éppen aktuális sor utáni sorban fog megjelenni.

Van azonban egy másik módszer is. Ha a képernyőnk X—Y fénypont-pozicionálós, akkor egy további adatbázis-kezelő parancs lehetővé teszi, hogy az üzenetek és a bevitt adatok ott helyezkedjenek el, ahol szeretnénk. A

^@ <koordináták> [SAY '<üzenet>']^

parancs az üzenetet (amelyet aposztrófok, idézőjelek vagy szögletes zárójelek közé kell zárni) a képernyőnek arra a részére helyezi, amelyet a <koordináták> által határoztunk meg. A koordináták a képernyő *sor* és *oszlop* helyeit jelölik. A 0,0 koordináták a képernyő kiindulási helyét (home — otthon — pozícióját), azaz a bal felső sarkát jelentik. A kiindulási hely 0,0 koordinátaszámaiból az is következik, hogy ha például a parancsban a 9,34 koordinátákat adtunk meg, akkor üzenetünk a 10. sor 35. oszlopában fog kezdődni.

MEGJEGYZÉS

Az adatbázis-kezelő rendszer a 0-ás sort a felhasználónak szóló üzenetek számára tartja fenn, ezért azt ne használjuk.

A parancs SAY része szabadon választható, mert ez a parancs arra is alkalmas, hogy az egyik sort (vagy annak egy részét) töröljük a képernyőn. Írjuk be a következő parancssorozatot egy parancsállományba, és futtassuk le.

Példa

```
ERASE
@ 20,30 SAY 'lentre is.'
@ 5,67 SAY 'Fentre is'
@ 11,11 SAY "irhatunk, meg"
@ 22,0
```

```
WAIT
@ 20,0
@ 5,0
@ 11,16
```

Az eddigiekben ezzel a paranccsal üzeneteket jelenítettünk meg, most pedig bemutatjuk egy olyan kifejezés értékének megjelenítését, amelyben egy vagy több változó szerepel. Ennek formája:

^@ <koordináták> [SAY <kifejezés>]^

Adjuk ki a következő parancsokat!

```
USE Nevek
@ 13,14 SAY Cim
@ 13,6 SAY Ir:Szam
SKIP 3
@ 17,5 SAY Nev + ', ' + Cim
```

Mint látjuk, a parancs tovább bővíthető annak érdekében, hogy megjeleníthessük a használatban levő változók (memóriaváltozók vagy egy adatbázis mezőneveinek) értékét, és mindezeket azon a képernyőpozíción, amelyiken akarjuk. (A változóknak, amelyeket a GET és a SAY paraméterek után megadunk, használatuk előtt létezniük kell, egyébként hibaüzenetet kapunk.)

A parancs még tovább is bővíthető:

^@ <koordináták> [SAY { '<üzenet>' / <kifejezés> }] [GET <változó>]^

Próbaképpen adjuk meg a következő parancssorozatot!

```
ERASE
USE Nevek
@ 15,5 SAY 'Cim' GET Cim
@ 9,14 SAY 'Iranyitoszam'
@ 10,18 GET Ir:Szam
@ 5,0 SAY 'Nev' GET Nev
```

Az előbbi parancsok eredményeként a változók értékeit (üzenettel vagy anélkül) a képernyő különböző helyein jeleníti meg az adatbázis-kezelő rendszer. Ezzel a lehetőséggel megtervezheti a programozó az adatbeviteli formátumot; például elérheti, hogy a képernyő hasonlítson ahhoz a formához, ahogy a felhasználó az adatfeldolgozást korábban papíron végezte. A

^@ <koordináták> SAY ['üzenet'] GET <változó>^

paranccsal elérhetjük, hogy az üzenetek és a változók (azok értéke) bármelyik, általunk kívánt helyre kerüljön a képernyőn. Felsorolhatunk parancsokat, melyeket egy ^READ^ parancs követ. Így a teljes képernyő formátumának a vezérlésére lehetőségünk nyílik.

Annak érdekében, hogy az előző parancssorozatban megadott helyen levő változókba adatot juttathassunk, adjuk meg a

^READ^

parancsot. A parancs hatására a fénypont automatikusan arra a változóra ugrik, amelyiket a ^READ^ parancs kiadása előtt elsőnek adtuk meg egy @ jelű parancsban. Most lehetőségünk van rá, hogy új adatot vigyünk be, vagy pedig elhagyjuk ezt a változót egy <CR> (<enter>) billentyű lenyomásával. Ha elhagytuk a változót, a következőhöz ugrik a fénypont. Változtassuk meg az adatokat a még hátralevő változókban. Amikor az utolsót is befejeztük, az adatbáziskezelő rendszer alapállapotba kerül. (Megjelenik a jellegzetes bejelentkező pont.) Most kérjük: DISPLAY. A rekordunk az újonnan megadott adatokat fogja tartalmazni.

A GET hasonlóképpen dolgozik, mint az INPUT és az ACCEPT parancs, de sokkal hatásosabb, mert lehetővé teszi, hogy több változót adjunk meg egyidejűleg.

Egy adatbázis akár 32 mezővel is rendelkezhet. Az eddig ismert adatbeviteli rendszerben nem volt módunk néhány kiválasztott mezővel foglalkozni. Most ahelyett, hogy az APPEND parancsot használnánk, amely az adatbázisunk összes mezejét listázza a képernyőn, alkalmazhatjuk az APPEND BLANK parancsot. Az APPEND BLANK létrehoz egy üres mezőkből álló rekordot, amelyet a GET parancs segítségével úgy tölthetünk fel, hogy csak azokat a mezőket jelenítjük meg a képernyőn, amelyekkel foglalkozni akarunk.

Az eddigi <Nevek.DBF> állományunk nem a legjobb példa, de mégis szemlélteti, hogy egy nagy struktúrába hogyan juttathatunk be adatokat úgy, hogy csak az érintett mezők jelenjenek meg a képernyőn.

Ahhoz, hogy nagyobb gyakorlatot szerezzünk a parancsállományok előállításában, hozzunk létre egy <Bevivo.CMD> nevű állományt, amely az alábbi parancsokat tartalmazza:

```
************************  Bevivo.CMD  ************************
  ERASE
  ? '   Ez a felhasznaloi program lehetove teszi azt,'
  ? '           hogy ujabb rekordokat adjunk a'
  ? 'Nevek.DBF adatbazisunkhoz, a mezok kivalogatasaval.'
  ?
  ? 'Csak a Nev es az Iranyitoszam mezoket adjuk meg'
  ?
  ?
  ? ' S leutesere az adatbevitelnek vege'
  ? '   <CR> leutesere folytatodik.'
  WAIT TO Folytato
  USE Nevek.dbf
  DO WHILE Folytato <> 'S' .AND. Folytato <> 's'
    APPEND BLANK
    ERASE
    @ 10,0 SAY "Nev" GET Nev
    @ 10,30 SAY "Iranyitoszam" GET Ir:Szam
    READ
```

```
    ? ' S leutesere az eljaras megall'
    ? '  <CR> leutesere folytatodik.'
    WAIT TO Folytato
  ENDDO
  RETURN
********************  Bevivo.CMD  vege  ********************
```

Az üzenetek kiírását és az adatbevitelt itt a képernyő általunk választott helyén tudjuk megvalósítani. A parancsállománnyal vigyünk adatokat néhány rekordba. Ha ezt befejeztük, listáztassuk ki LIST paranccsal az állományt, hogy lássuk mit adtunk hozzá.

Amikor később visszatérünk a CP/M operációs rendszerbe, és szükségünk lesz erre a parancsállományra, gépeljük be:

. dbase bevivo

illetve

. propbase bevivo

MEGJEGYZÉS

Az ^ERASE^ vagy a ^CLEAR GETS^ parancsot kell kiadni minden 64. GET után. Válasszuk a ^CLEAR GETS^ parancsot, ha azt szeretnénk, hogy a képernyő változatlan maradjon.

A @ jelű parancs használható a SAY rész nélkül is,

^@ <koordináták> GET <változó>^

alakban. (Egy későbbi READ utasítás természetesen kell még a parancsállományba.) Ilyenkor nincs üzenet, de a számítógép ekkor is kijelzi azokat a kettőspontokat a képernyőn, amelyekkel a változó részére fenntartott helyet határolja.

MEGJEGYZÉS

Ha a képernyős szerkesztést és formázást alkalmazzuk (SCREEN módban vagyunk), a sorok és oszlopok számai nem szükségszerűen vannak sorrendben, de mégis jobb, ha sorrendben írjuk őket, mert majd a PRINT utasításhoz sorrendben *kell* lenniük.

Ugyanez a parancs kibővíthető speciális formázási lehetőséggel is, mint ez az alábbi példában látható.

^@ <koordináták> SAY ['<üzenet>'] GET <változó> [PICTURE '<formátum>']^

A szabadon választható (opcionális) PICTURE paraméter az alább felsorolt, a formátumot megadó szimbólumokkal tölthető ki. Az adatbázis-kezelő rendszer a szimbólumok hatására a szimbólum mellett leírt tevékenységet végzi.

9 vagy #	válaszként csak számjegyeket fogad el.
A	csak alfabetikus karaktereket fogad el.
!	a bemenő karaktereket nagybetűssé alakítja.

X	bármilyen karaktert elfogad.
$	'$' jelet jelez ki a képernyőn.
*	'*' jelet jelez ki a képernyőn.

Lássunk egy példát a PICTURE paraméter alkalmazására!

^@ 5,1 SAY "Mai datum" GET Datum PICTURE '99/99/99'^

parancs az alábbit fogja kijelezni:

```
Mai datum:  /  /  :
```

jelezvén azt, hogy a Datum nevű változó üres volt. Ebben a példában csak számjegyek írhatók be válaszként.

10.2 A NYOMTATANDÓ INFORMÁCIÓ FORMÁZÁSA (SET FORMAT TO PRINT) (@. .SAY. .USING, EJECT)

Ha a SET FORMAT TO PRINT parancsot használjuk, a @ jelű parancs hatására az információ a képernyő helyett a nyomtatón jelenik meg. Mivel adatot csak képernyőn lehet bevinni, a parancs GET és PICTURE része mellőzendő, és a READ parancs nem használható.

Azokat az adatokat, amelyeket valamilyen formátumra akarunk írni, tanácsos előbb a képernyő segítségével megtervezni. Ha a formátum már megfelelő, SET FORMAT TO PRINT-ben írhatunk a nyomtatóra, a következő parancs segítségével.

^@ <koordináták> SAY { <változó> / <állandó> / <kifejezés> } [USING <formátum>]^

Nyomtatás céljára a koordinátákat sorrendben kell megadnunk!

A sorok számainak növekvő sorrendben kell szerepelniük (a 7-es sort előbb kell megadni, mint a 9-est stb.). Minden sorban az oszlopoknak is sorrendben kell lenniük (előbb a 15-ös, azután a 63-as oszlop gépelhető stb.).

A parancs kiírhatja egy változó éppen aktuális értékét, egy kifejezés értékét, egy karakterláncot vagy pedig egy üzenetet.

Ha a USING részt is tartalmazza a parancs, a USING paraméter után megadott szimbólum meghatározza, hogy milyen karaktereket ír ki az adatbázis-kezelő rendszer, és hogy ez hol jelenik meg a lapon.

A USING szimbólumok az alábbiak:

9 vagy #	csak számjegyeket ír.
A	csak alfabetikus karaktereket ír.
X	minden nyomtatható karaktert kiír.
$	csak számjegyeket ír, illetve '$' jelet a bevezető zérus(ok) helyén.
*	csak számjegyeket ír, illetve '*' jelet a bevezető zérus(ok) helyén.

MEGJEGYZÉS

Mivel a ^@ 10,5 SAY Óra×órabér USING '××××××××.99'^ parancs nem tartalmaz GET részt, egyformán használható a képernyőn és a nyomtatón való megjelenítésre is. Az Óra = 8 és órabér = 12.73 értékek esetén a parancs kiírná (megjelenítené) a „××××× 101.84"-et, ami bizonyos esetekben nagyon hasznos nyomtatási formátumot ad.

HARMADIK RÉSZ

AZ ADATBÁZIS-KEZELŐ PARANCSAI

11. A parancsok részletes ismertetése

Az adatbázis-kezelő rendszer parancsai mindig parancsszóval kezdődnek. Ezt követhetik a paraméterek és a kifejezések. A paraméterek tetszőleges sorrendben állhatnak.

Az eddigiekben nem foglalkoztunk a parancsok minden lehetséges alkalmazási formájával. A parancsszavak felsorolásánál a belőlük alkotható parancsok összes megjelenési formáját ismertetni fogjuk. A jelölések a függelék végén találhatók.

A parancsszavak négy csoportba oszthatók.

1. Önmagukban is parancsot alkotó parancsszavak.
   (Például ERASE, RESET)

2. Önmagukban is, és paraméterekkel is alkalmazható parancsszavak.
   (Például APPEND, LIST)

3. Értelmezhető és végrehajtható parancsot csak paraméterekkel együtt alkotó parancsszavak.
   (Például COPY, LOCATE)

4. Több parancsszóból álló szerkezetek részeiként szereplő parancsszavak.
   a) Szerkezetet nyitó (definiáló) parancsszavak.
      (Például DO CASE, IF)
   b) Szerkezetbeli parancsszavak.
      (Például CASE, LOOP)
   c) Szerkezetet záró parancsszavak.
      (Például ENDDO, ENDIF)

11.1 MEGJEGYZÉSEK AZ ISMERTETÉSHEZ MELLÉKELT PÉLDÁKKAL KAPCSOLATBAN

A parancsok ismertetésénél egy nem túlzott igénnyel létrehozott, ám az összes parancs hatását jól bemutató raktári rendszert alkalmazunk. Raktárunkban különböző színű, spray kiszerelésű Neolux és Wernodux festékeket tárolunk. A példákban legsűrűbben előforduló adatbázis-állományok szerkezete a 2–4. táblázatokban található. (A rekordok mérete a törlésre való megjelölés számára fenntartott hely miatt nagyobb 1 byte-tal a mezők hosszának összegénél.)

A példákban nem kizárólagosan az éppen tárgyalt parancsszó szerepel, hiszen már mindegyik parancsszót megismerhettük. Ha az Olvasó bármelyik parancsra, parancsszóra nem emlé-

kezne, lapozzon vissza a *második* részbe, vagy itt, a harmadik részben nézze meg az alfabetikus felsorolás szerint. A parancsszavakat lapszélre emeltük ki, így azok könnyen megtalálhatók.

2. Táblázat. A <Raktar.DBF> és az <Ujanyag.DBF> szerkezete.

Tárolt adat	Mezőnév	Mezőtípus	Mezőhossz	Tizedesjegyek száma
Anyag neve	ANYAGNEV	Karakter	15 byte	
Érkezett mennyiség	ERKEZETT	Numerikus	4 byte	
Raktárból kiadott mennyiség	KIADVA	Numerikus	4 byte	
Kódszám	ANYAGKOD	Karakter	3 byte	
Darabár	EGYSEGAR	Numerikus	6 byte	2
Összérték	OSSZERTEK	Numerikus	9 byte	2

Egy rekord terjedelme 42 byte.

3. Táblázat. A <Vetelezok.DBF> szerkezete.

Tárolt adat	Mezőnév	Mezőtípus	Mezőhossz	Tizedesjegyek száma
Anyagfelvételező neve	NEV	Numerikus	15 byte	
Kódszám	ANYAGKOD	Karakter	3 byte	
Felvett mennyiség	MENNYISEGE	Numerikus	6 byte	
Dátum	DATUM	Karakter	8 byte	

Egy rekord terjedelme 33 byte.

4. táblázat. A <Bovites.DBF> szerkezete.

Tárolt adat	Mezőnév	Mezőtípus	Mezőhossz	Tizedesjegyek száma
Kódszám	ANYAGKOD	Karakter	3 byte	
Darabár	EGYSEGAR	Numerikus	5 byte	2

Egy rekord terjedelme 9 byte.

11.2 PARANCSSZAVAK

ACCEPT	(elfogad)
APPEND	(hozzáfűz)
BROWSE	(böngészik)
CALL	(hív)
CANCEL	(érvénytelen)
CASE	(eset)
CHANGE	(csere)
CLEAR	(kitisztít)
CONTINUE	(folytat)
COPY	(másol)
COUNT	(számlál)
CREATE	(létrehoz)
DELETE	(töröl)
DISPLAY	(kijelez)
DO	(csinál)
EDIT	(szerkeszt)
EJECT	(a következő lapra állít)
ELSE	(különben)
ENDCASE	(DO CASE vége)
ENDDO	(DO WHILE vége)
ENDIF	(IF vége)
ERASE	(képernyőtörlés)
FIND	(megtalál)
GO . . GOTO	(menj)
IF	(ha)
INDEX	(indexel)
INPUT	(beolvas)
INSERT	(beilleszt)
JOIN	(egyesít)
LIST	(listáz)
LOCATE	(helyet megállapít)
LOOP	(hurok)
MODIFY	(módosít)
NOTE	(magyarázat)
OTHERWISE	(egyébként)
PACK	(összecsomagol)
POKE	(bejuttat)
QUIT	(elhagy)
READ	(olvas)
RECALL	(visszavon)
RELEASE	(felszabadít)
REMARK	(megjegyzés)
RENAME	(átnevez)
REPLACE	(kicserél)
REPORT	(jelent)

RESET (utánállít)
RESTORE (visszahelyez)
RETURN (visszatér)
SAVE (félretesz)
SELECT (kiválaszt)
SET (beállít)
SKIP (ugrik)
SORT (átrendez)
STORE (tárol)
SUM (összegez)
TOTAL (összeget képez)
UPDATE (frissít)
USE (használ)
WAIT (várakozik)
?
@
*

Alakja:

ACCEPT ["<üzenet>"] TO <memóriaváltozó>

Hatása:

A billentyűzetről válaszként érkező karaktert vagy karakterláncot elhelyezi egy memóriaváltozóban.

Ismertetése:

A parancs hatására az adatbázis-kezelő rendszer a billentyűzetről érkező válaszra vár. Ha a parancs parancsállományban szerepel, a felhasználói program futása megáll, és csak a válaszadás után folytatódik.

A parancs sajátos tulajdonsága, hogy korábban nem használt memóriaváltozót is megnevezhetünk benne, mert ha az nem létezik még, akkor az adatbázis-kezelő rendszer automatikusan létrehozza.

A parancs hatására keletkező memóriaváltozó — a beérkező karakterektől függetlenül — mindenkor karaktertípusú, akkor is, ha korábban nem az volt. A válaszban nem szükséges a máskor szokásos karakterhatároló jelöléseket alkalmazni. Ha a billentyűzetről érkező válasz csak egy egyedülálló <CR> kód, a memóriaváltozóba egy szóközkarakter kerül.

A parancsban az üzenet szabadon választható, azaz nem kötelező üzenetet is elhelyezni. Az üzenet a képernyőn kettőspont kíséretében jelenik meg. Itt fogalmazhatjuk meg közlendőinket, melyekre a választ várjuk. Üzenetünk a parancsban a szokásos határolók bármelyike között elhelyezhető.

Példák

```
. accept "Kerem irja be a nevet" to nev1
Kerem irja be a nevet:Elso Jozsef

. accept "Kerem irja be a nevet" to nev2
Kerem irja be a nevet:Masodik Janos

. display memory                                    (Lásd DISPLAY)
NEV1          (C)  Elso Jozsef
NEV2          (C)  Masodik Janos
**TOTAL**  02 VARIABLES USED   00023 BYTES USED
                ** OSSZESEN** 02 MEMORIAVALTOZO 00023 BYTE

. accept to harmi
:Harmadik Peter

. accept "Kerem irja be a nevet" to nev1
Kerem irja be a nevet:Masodik Janos
```

ACCEPT

```
  . display memory
NEV1          (C)  Masodik Janos
NEV2          (C)  Masodik Janos
BARMI         (C)  Harmadik Peter
**TOTAL**  03 VARIABLES USED   00039 BYTES USED
                    ** ÖSSZESEN** 03 MEMORIAVALTOZÓ 00039 BYTE
```

APPEND

Alakja:

a) APPEND

b) APPEND BLANK

c) APPEND FROM <állomány> [{WHILE <kifejezés> / FOR <kifejezés>}] ⇒

⇒ [{DELIMITED [WITH <határoló>] / SDF}]

Hatása:

Új rekordok hozzáfűzése az aktuális munkatérben levő adatbázis-állományhoz.

Ismertetése:

Az APPEND (továbbá a CREATE és az INSERT) parancs szolgál arra, hogy egy adatbázisba új adatokat vigyünk be, azaz egy adatbázis-állományhoz rekordokat adjunk hozzá. Az APPEND (és a CREATE) használatával egy alkalommal több rekorddal bővíthetjük az adatbázist (az INSERT esetében csak eggyel).

Különösen hasznos az APPEND parancs akkor, ha felvetődik az adatbázis-mezők bővítésének, rövidítésének vagy mezők felvételének, illetve megszüntetésének igénye. A CREATE parancs alkalmazásával létre kell hozni egy kívánt szerkezetű adatbázis-állományt, s ebbe az APPEND parancs végrehajtásával átvinni a régi adatbázis adatait. Azok a mezők, amelyek csak az új adatbázisban szerepelnek, üresek maradnak.

a) A rövid „APPEND" parancs hatására a munkaállomány mezőnevei sorban megjelennek, és a billentyűzetről adatokkal tölthetjük fel őket. Több rekorddal bővíthetjük ily módon az adatbázis-állományt, mindaddig, míg egy rekord első mezőjének első karakterpozícióján állva a <CR> billentyűt nem nyomjuk meg.

Rendezett (INDEX-elt) adatbázis-állományok esetében a USE parancsban meghatározott indexállomány is automatikusan megváltozik az új információknak megfelelően (kivéve az „APPEND BLANK" esetében).

Az adatbázishoz tartozó többi indexállományt ismételt rendezéssel kell felújítani.

A „SET CARRY ON" parancs kiadása azt eredményezi, hogy az APPEND végrehajtásakor az aktuális rekord minden esetben feltöltődik az előző rekord tartalmával, amit azután módosíthatunk. Kitűnően alkalmazható ez akkor, amikor egymást követő rekordok sok azonos adatot tartalmaznak.

b) Az APPEND BLANK parancs esetében egy üres helyekkel kitöltött rekorddal bővül a munkaállomány, melyet az EDIT vagy a REPLACE parancs alkalmazásával tölthetünk fel adatokkal.

c) Az adatbázisba az <állomány> parancsrész által meghatározott állományból kerülnek át a rekordok. Ha az új – átkerülő – rekordok rövidebbek, mint az aktuális munkatérben levő adatbázis-állomány rekordjai, az új rekord üres helyekkel (szóközökkel) bővül; a hosszabb rekordok viszont csonkulnak.

APPEND

A rövid „APPEND FROM <állomány>" parancs esetében (az SDF és a DELIMITED paraméterek nélkül) a forrásként szolgáló adatbázis-állományról feltételezi a parancs, hogy az adatbázis-kezelő rendszer struktúrájának megfelelő felépítésű.
Az aktuális munkatérben levő állomány és a forrásállomány szerkezetének összehasonlítása után azokat a mezőket másolja át a parancs, amelyek mindkét állományban szerepelnek. Az aktuális munkatérben levő adatbázis-állomány szerkezetének megfelelően bővülnek vagy csonkulnak az egyes mezők.

A FOR paraméter alkalmazásával az adatbázis-kezelő rendszer a forrásállomány összes rekordját ellenőrzi a <kifejezés> adta feltételek szerint, és csak azokat másolja át, melyekre a feltétel igaznak bizonyul.
A WHILE paraméter használata esetén a forrásállomány aktuális rekordjától kezdve addig tart a parancs végrehajtása, amíg a soron következő rekordokra a <kifejezés> igaznak bizonyul.
Az első rekord átmásolása után következik a célállományban a feltételek vizsgálata. Ha a kifejezés „értéke" igaz, a rekord az adatbázis-állományban marad, ellenkező esetben törlődik. Ezután az adatbázis-kezelő rendszer a forrásállomány következő rekordjával végzi el az előző tevékenységsorozatot mindaddig, amíg — FOR paraméter esetén — a forrásállomány végére nem ér, illetve — WHILE paraméter esetén — a kifejezés az adott rekordnál hamis értéket nem ad. Ennek a parancsformának a hátránya az, hogy a feltételt tartalmazó <kifejezés>-ben csak azok a mezők szerepelhetnek, amelyek az aktuális munkatérben levő célállományban megtalálhatók.
Amikor az SDF szabadon választható paramétert megadjuk, a parancs feltételezi, hogy az adatok általános adatformátumban (System Data Format; 2.11. és 7.6. rész) vannak. Ahhoz, hogy az adatok megfelelő helyre kerüljenek, minden mezőnek az adatbázis-állomány adatstruktúrájában meghatározott hosszúságúnak kell lennie. A rekordok átvitele mindaddig folytatódik, amíg az <állomány> véget nem ér.
A DELIMITED paraméter megadása esetén a parancs feltételezi, hogy a forrásállomány rekordjaiban határolók között vannak az adatok.
A legtöbb számítógépes program olyan adatállományokat készít, amelyekben a karakterláncok idézőjelek vagy aposztrófok közé kerülnek, és a mezőket vessző választja el egymástól. Az APPEND parancs csak az ilyen határolókkal ellátott állományokból képes adatok helyes átvitelére. Ugyanakkor helyesen dolgozik akkor is, ha a mezők adatai határoló nélküliek, de a mezőket vessző választja el egymástól. Az adatok a határolóktól „megtisztítva", az adatbázis-állomány szerkezetének megfelelő formában kerülnek át az adatbázisba.

* * *

Az APPEND és SET SCREEN ON végrehajtási ideje alatt érvényes szerkesztési kódok

$^{\Delta}$X (CAN) a fénypontot lefelé mozgatja a következő mezőre.
(Alkalmazható e célra a $^{\Delta}$F (ACK) is.)

^E (ENQ) a fénypontot visszafelé mozgatja az előző mezőre.
(Alkalmazható e célra a ^A (SOH) is.)

^D (EOT) a fénypontot előre viszi a következő karakterre.

^S (DC3) egy karakternyit visszalépteti a fénypontot.

^V (SYN) kapcsoló a felülírási és a közéírási mód között.

^G (BEL) törli azt a karaktert, amelyen a fénypont áll.

<Rubout> (DEL) a fényponttól balra álló karaktert törli.

^Y (EM) törli az aktuális mező tartalmát.

^R (DC2) a rekordot a mágneslemezre írja, és a következő rekordra áll.
(Alkalmazható e célra a ^C (ETX) is.)

<ENTER> kulcs <CR>-nek felel meg. Ha akkor adjuk meg ezt a kódot, amikor a fénypont egy új rekord kezdő pozícióján áll, a rendszer alapállapotba tér vissza.

^Q (DC1) törli a rekordot, és visszatér a normál műveletvégzéshez.

^W (ETB) minden változtatást, amit a szerkesztésben végeztünk, figyelembe vesz, és visszatér alapállapotba.
(Alkalmazható e célra a ^O (SI) is.)

MEGJEGYZÉS

APPEND szerkesztési módban a 0-tól 9-ig terjedő számjegyek, a tizedespont, a „–" és a „+" jel mellett a szóközt is használhatjuk numerikus adatok bevitelekor.
A vezérlőkarakterek – az adatbázis-kezelő rendszer alapállapotában – ugyanezeknek a kódoknak egy részét más feladatok elvégzésére használják (v.ö. a 3.1. résszel).

Példák

```
. use raktar
. display all
00001   NEOLUX PIROS        300    200 123    38.50          0.00

. append

RECORD#00002                                          REKORD#00002
ANYAGNEV   : NEOLUX KEK
ERKEZETT   : 300
```

APPEND

```
KIADVA     : 100
ANYAGKOD   : 124
EGYSEGAR   : 30
OSSZERTEK  : <CR>

RECORD#00003                                        REKORD#00003
ANYAGNEV   : VERNODUX KEK
ERKEZETT   : 500
KIADVA     : 300
ANYAGKOD   : 789
EGYSEGAR   : 96.1
OSSZERTEK  : <CR>

RECORD#00004                                        REKORD#00004
ANYAGNEV   : <CR>

 . display all
00001  NEOLUX PIROS        300    200 123    38.50          0.00
00002  NEOLUX KEK          300    100 124    30.00          0.00
00003  VERNODUX KEK        500    300 789    96.10          0.00

 . use ujanyag
 . display all
00001  NEOLUX ZOLD         350    150 125    28.60          0.00
00002  VERNODUX SARGA      300    300 798    91.10          0.00

 . use raktar

 . append from ujanyag
00002 RECORD ADDED                          00002 REKORD HOZZAADVA

 . displ all
00001  NEOLUX PIROS        300    200 123    38.50          0.00
00002  NEOLUX KEK          300    100 124    30.00          0.00
00003  VERNODUX KEK        500    300 789    96.10          0.00
00004  NEOLUX ZOLD         350    150 125    28.60          0.00
00005  VERNODUX SARGA      300    300 798    91.10          0.00
```

```
. append blank
. display all
00001   NEOLUX PIROS        300    200 123    38.50         0.00
00002   NEOLUX KEK          300    100 124    30.00         0.00

00003   VERNODUX KEK        500    300 789    96.10         0.00
00004   NEOLUX ZOLD         350    150 125    28.60         0.00
00005   VERNODUX SARGA      300    300 798    91.10         0.00
00006                         0      0         0.00         0.00

. go 6
. replace anyagnev with 'VERNODUX PIROS'
000001 REPLACEMENT(S)                          00001 HELYETTESITES

. display all
00001   NEOLUX PIROS        300    200 123    38.50         0.00
00002   NEOLUX KEK          300    100 124    30.00         0.00
00003   VERNODUX KEK        500    300 789    96.10         0.00
00004   NEOLUX ZOLD         350    150 125    28.60         0.00
00005   VERNODUX SARGA      300    300 798    91.10         0.00
00006   VERNODUX PIROS        0      0         0.00         0.00
```

Adatbevitel lehetséges más rendszerben létrehozott állományból is. Nézzünk példát a BASIC-ben meghatározott, két rekordot tartalmazó <festek.adt> állományból megvalósítandó adatbevitelre!

```
'NEOLUX SARGA',400,200,'145',35.90
'NEOLUX FEHER',800,700,'100',34.00
```

```
. append from festek.adt delimited
00002 RECORD ADDED                             00002 REKORD HOZZÁADVA

. display all
00001   NEOLUX PIROS        300    200 123    38.50         0.00
00002   NEOLUX KEK          300    100 124    30.00         0.00
00003   VERNODUX KEK        500    300 789    96.10         0.00
00004   NEOLUX ZOLD         350    150 125    28.60         0.00
00005   VERNODUX SARGA      300    300 798    91.10         0.00
00006   VERNODUX PIROS        0      0         0.00         0.00
00007   NEOLUX SARGA        400    200 145    35.90         0.00
00008   NEOLUX FEHER        800    700 100    34.00         0.00
```

BROWSE

Alakja:

BROWSE

Hatása:

Az adatbázis-állomány adatainak szerkesztése és megjelenítése a képernyőn.

Ismertetése:

Egy időben legfeljebb 19 rekord adatait írja a képernyőre. (Ha a mezők több mint 80 karaktert tartalmaznak, akkor kevesebbet.) Minden képernyősor annyi mezőt tartalmaz, amennyi ott elfér. A képernyő első sorában az aktuális rekord száma látható. Úgy kell tekinteni a képernyőt, mint egy ablakot, amelyen keresztül belenézünk az adatbázisba. A sorokon le- és felfelé mozogva érhetünk el újabb rekordokat, jobbra vagy balra mozogva pedig a rekordokon belül más-más mezőket.

A teljes képernyős szerkesztési mód kódjai érvényesek, bármely adat szerkesztése lehetséges.

A BROWSE végrehajtási ideje alatt érvényes szerkesztési kódok

^X (CAN) a következő adatmezőre, azaz egy mezővel jobbra vezérli a fénypontot. (Erre a célra a ^F (ACK) vezérlőkaraktert is alkalmazhatjuk.)

^E (ENQ) a megelőző adatmezőre, azaz egy mezővel balra vezérli a fénypontot. (Erre a célra a ^A (SOH) vezérlőkaraktert is alkalmazhatjuk.)

^D (EOT) a következő karakterre, azaz egy karakterrel jobbra viszi a fénypontot.

^S (DC3) a megelőző karakterre, azaz egy karakterrel balra tolja a fénypontot.

^C (ETX) a következő rekordra, azaz egy sorral lejjebb mozdítja a fénypontot.

^R (DC2) a megelőző rekordra, azaz egy sorral feljebb viszi a fénypontot.

^G (BEL) törli azt a karaktert, amelyen a fénypont áll.

RUBOUT törli a fénypont előtti, azaz a tőle balra álló karaktert.

^B (STX) a mezőket jobbra mozgatja.

^Z (SUB) a mezőket balra mozgatja.

^U (NAK) az aktuális rekord jelölése törlésre, illetve a jelölés megszüntetése.

^W (ETB) kilépés alapállapotba a változtatások megőrzésével.

^Q (DC1) kilépés alapállapotba a változtatások megőrzése nélkül.

MEGJEGYZÉS

BROWSE szerkesztési módban a 0-tól 9-ig terjedő számjegyek, a tizedespont, a „–" és a „+" jel mellett a szóközt is használhatjuk numerikus adatok bevitelére.

A vezérlőkarakterek – az adatbázis-kezelő rendszer alapállapotában – ugyanezeknek a kódoknak egy részét más feladatok elvégzésére használják (v.ö. a 3.1. résszel).

CALL

Alakja:

CALL [<memóriaváltozó>]

Hatása:

Gépi kódú programok végrehajtása.

Ismertetése:

A parancs hatására gépi kódú programok végrehajtása közben nem hagyjuk el az adatbázis-kezelő rendszert. A parancs kiadásakor a ^SET CALL <cím>^ parancs <cím> paraméterében megadott helyen kezdődő gépi kódú rutinnak adja át a vezérlést az adatbázis-kezelő rendszer. (A címet itt is decimális értékként kell megadni.) A megadott cím elérésekor a H—L regiszterpár a memóriaváltozó első byte-jára, a hossz byte-ra mutat. A memóriaváltozónak karakterláncnak kell lennie, és nem lehet hosszabb, mint amekkora adat számára a gépi kódú programban meghatározott hely van.

A rutinból való visszatérés egy gépi kódú *return* végrehajtásával történik.

Az A400H feletti címtartomány a CP/M BDOS-ig rendelkezésre áll, de vegyük figyelembe, hogy a SORT parancs is ezt a területet használja.

CANCEL

Alakja:

CANCEL

Hatása:

Parancsállomány végrehajtásának befejezése, közvetett mód elhagyása.

Ismertetése:

A közvetett módból való kilépésre, azaz a parancsállományban rögzített felhasználói program befejezésére szolgál. A parancs hatására az adatbázis-kezelő rendszer közvetlen módba, azaz alapállapotba kerül.

Példa:

```
ACCEPT  ' Ki akar lepni a felhasznaloi programbol?  ;
(I)gen/(N)em ' TO X
IF X=I
  CANCEL
ENDIF
```

CASE

Alakja:

CASE

Szerepe:

A ^DO CASE^ szerkezet egyik ága.

Ismertetése:

Az egyes CASE ágak vizsgálata során az első igaz eredményt adó kifejezés által meghatározott CASE ágra kerül a vezérlés. Több „igaz" értékű ág közül csakis az első végrehajtására kerül sor.

(Lásd a DO parancsot!)

CHANGE

Alakja:

CHANGE [<érvényességi kör>] FIELD <mezőlista> {[FOR <kifejezés> / WHILE <kifejezés>]}

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül az aktuális rekord,
- FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamis rekordig minden rekord.

Hatása:

Az adatbázis-állományban levő adatok egyszerű módosítása.

Ismertetése:

A FIELD <lista>-ban felsorolt, egymástól vesszővel elválasztott mezők tartalma megjelenik a képernyőn, majd dönthetünk arról, hogy módosítjuk-e az adatot. A mezők megjelenése a listában meghatározott sorrendű. A lista kimerülésével az <érvényességi kör> értelmének megfelelően ajánl fel még rekordokat javításra az adatbázis-kezelő rendszer.

A FOR és a WHILE paraméter használatával egy kifejezésben meghatározhatjuk, hogy mely feltételeknek megfelelő rekordokat akarjuk módosítani.

Egy mező teljes törlése a CHANGE? (VÁLTOZTATJA?) üzenetre adott ^Y majd <CR> kódokkal történik.

Ha nem akarunk már — a feltételeknek egyébként megfelelő — rekordokon javítani, akkor ESCAPE karakter használatával „soronkívül" visszatérhetünk az adatbázis-kezelő rendszer alapállapotához.

Példa

```
. use Naptár
. change field Kelt

RECORD:#00001                                   REKORD:#00001
KELT: 02/17/84
CHANGE? 84                                          CSERÉL?
TO          85

KELT: 02/17/85
CHANGE?  <CR>                                       CSERÉL?
```

CLEAR

Alakja:

CLEAR [GET[S]]

Hatása:

Az adatbázis-kezelő rendszer alaphelyzetbe állítása, illetve az aktív „GET" adatbevitel módjára vonatkozó információk törlése.

Ismertetése:

A CLEAR parancs alaphelyzetbe állítja az adatbázis-kezelő rendszert. Az összes munkábaállított adatbázis-állományt lezárja, és használaton kívül helyezi. Megszünteti a memóriaváltozókat, és az elsődleges munkatérre vezérli az adatbázis-kezelő rendszert.

Ez a parancs az adatbázis-kezelő rendszert érintetlen állapotba hozza. Például, ha egy parancsállomány végrehajtása során a rendszer a másodlagos munkatérben dolgozik, és a parancsállomány elhagyása után abban is marad; akkor egy új parancsállomány végrehajtásában zavart okozhat, ha azt feltételezi, hogy a rendszer az elsődleges munkatérben dolgozik. Ezért a parancsállományok kezdetén ajánlatos alkalmazni a CLEAR utasítást, ha ezt más szempont nem zárja ki. (V. ö. a SELECT paranccsal, 228. old.)

A GET (s) paraméter használatánál viszont csak a @jelű parancsokban levő GET adatbeviteli módra vonatkozó információt törli. Ez alatt azokat a GET-eket kell érteni, amelyek a rendszer indulása, illetve a legutolsó CLEAR GET(S) vagy ERASE után léptek életbe. (V. ö. a @jelű paranccsal, 255. old.)

A ^CLEAR GET[S]^ a képernyőt érintetlenül hagyja, szemben az ERASE paranccsal, amely a képernyőt is letörli. Legalább minden 64. GET után CLEAR GET(S) vagy ERASE utasítást kell kiadni.

CONTINUE

Alakja:

CONTINUE

Szerepe:

A LOCATE parancs végrehajtásának folytatása.

Ismertetése:

Az adatbázis-állományban a LOCATE parancsban szereplő feltételeket kielégítő rekordok keresésének folytatása. Amikor a LOCATE parancs segítségével a rendszer megtalálta a feltételeknek megfelelő, sorrendben első rekordot, a szükséges műveletek elvégzése után folytathatjuk a feltételeknek megfelelő következő rekord keresését a CONTINUE parancs kiadásával. Az így keresett rekord megtalálása után újra kiadhatjuk a CONTINUE parancsot.

A LOCATE és a CONTINUE közé több parancs ékelődhet, bár ezek mérete korlátozott. (Bővebb információ és példák a LOCATE parancsnál.)

COPY

Alakja:

COPY TO <állomány> [<érvényességi kör>] [FIELD <mezőlista>]⇒

⇒ [{DELIMITED [WITH <határoló>] / STRUCTURE [EXTENDED] / SDF}] [{WHILE <kifejezés> / FOR <kifejezés>}]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül vagy FOR paraméterrel minden rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamisnak ítélt rekordig minden rekord.

Hatása:

A munkába állított adatbázis-állomány másolása.

Ismertetése:

Az <állomány> lehet az adatbázis-kezelő rendszer formátumának, vagy más rendszer adatformátumának (SDF, 2.10. és 7.6. rész) megfelelő is.

A STRUCTURE paraméter megadásával csak az adatbázis-állomány szerkezete kerül másolásra.

Ha az EXTENDED paramétert is megadjuk, az adatbázis-kezelő rendszer egy olyan adatbázis-állományt hoz létre, mely a forrásállomány szerkezetét mint adatokat rekordjaiban tartalmazza.

FIELD paraméterezésnél az adatbázis-kezelő rendszer csak a mezőlistában felsorolt mezőkre értelmezi a parancsot, akkor is, ha csak a szerkezet (struktúrát) akarjuk másolni.

Az SDF paraméter hatására az állomány struktúra nélkül másolódik egy hagyományos ASCII formátumú állományba. Így állíthatunk elő olyan állományokat, melyeket más rendszerben kívánunk felhasználni.

A STRUCTURE és az SDF paraméter kölcsönösen kizárja egymást.

DELIMITED paraméterezésnél a célállományban a karakter típusú mezők tartalma aposztrófok közé kerül, a mezőket pedig vesszők választják el egymástól. (Ez a fordítottja az azonos alakú APPEND parancs hatásának.)

Alaphelyzetben a DELIMITED paraméter hatására a karakter típusú mezők határolója az aposztróf ('). A WITH paraméter lehetővé teszi, hogy bármelyik karaktert határolóként jelöljük ki. Ha vessző (,) szerepel határolóként, a karakter típusú mezők végéről és a numerikus mezők elejéről az üres helyeket levágja a rendszer. A karakterláncok nem kerülnek idézőjelek közé. (Az APPEND parancs, mint láttuk, más rendszer által létrehozott állományból való hozzáfűzésnél csak idézőjelet ('') vagy aposztrófot (') fogad el karakterhatárolóként.)

A DELIMITED vagy az SDF paraméterek használatakor a célállomány nevének állománynév-kiterjesztése alapértelmezés szerint <.TXT>. Más esetekben az alapértelmezés <.DBF>.

Ha a célállomány még nem létezett, akkor létrejön és a mágneslemezen levő nyilvántartásba is bejegyződik. Ha a célállomány létezik, az adatbázis-kezelő rendszer azt felülírja.

Példák

```
. use raktar  (A rekordok szerkezetét lásd a 11.1. részben.)
. display all
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  NEOLUX ZOLD         350    150 125    28.60        0.00
00005  VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS        0      0         0.00        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00

. copy to raktar2
00008 RECORDS COPIED                          00008 REKORD MASOLVA

. use raktar2
. display all
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  NEOLUX ZOLD         350    150 125    28.60        0.00
00005  VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS        0      0         0.00        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00

. copy to raktar3 for erkezett>300
00004 RECORDS COPIED                          00004 REKORD MASOLVA

. use raktar 3
. disp all
00001  VERNODUX KEK        500    300 789    96.10        0.00
00002  NEOLUX ZOLD         350    150 125    28.60        0.00
00003  NEOLUX SARGA        400    200 145    35.90        0.00
00004  NEOLUX FEHER        800    700 100    34.00        0.00

. copy to raktar4 field anyagnev, anyagkod, egysegar
00004 RECORDS COPIED                          00004 REKORD MASOLVA
```

COPY

```
 . use raktar 4
 . disp all
00001  VERNODUX KEK     789    96.10
00002  NEOLUX ZOLD      125    28.60
00003  NEOLUX SARGA     145    35.90
00004  NEOLUX FEHER     100    34.00

 . use raktar
 . disp all
00001  NEOLUX PIROS       300    200 123    38.50         0.00
00002  NEOLUX KEK         300    100 124    30.00         0.00
00003  VERNODUX KEK       500    300 789    96.10         0.00
00004  NEOLUX ZOLD        350    150 125    28.60         0.00
00005  VERNODUX SARGA     300    300 798    91.10         0.00
00006  VERNODUX PIROS       0      0          0.00         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00

 . go 3
 . copy next 3 to raktar5
00003 RECORDS COPIED                          00003 REKORD MÁSOLVA

 . use raktar5
 . display all
00001  VERNODUX KEK       500    300 789    96.10         0.00
00002  NEOLUX ZOLD        350    150 125    28.60         0.00
00003  VERNODUX SARGA     300    300 798    91.10         0.00

 . use raktar
 . display structure
STRUCTURE FOR FILE:  RAKTAR.DBF          ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS :  00008               A REKORDOK SZÁMA
DATE OF LAST UPDATE: 85/04/15            UTOLSÓ DÁTUM
PRYMARY USE DATABASE                     ELSŐDLEGES MUNKATÉR
FLD       NAME      TYPE  WIDTH   DEC   MEZŐ NÉV TÍPUS HOSSZ
001      ANYAGNEV     C     015                    TIZEDESEK
002      ERKEZETT     N     004
003      KIADVA       N     004
```

```
004      ANYAGKOD      C     003
005      EGYSEGAR      N     006     002
006      OSSZERTEK     N     009     002
**TOTAL**                   00042           **ÖSSZES**

 . copy to szerkeze structure
 . use szerkeze
 . display structure
STRUCTURE FOR FILE:  SZERKEZE.DBF        ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS :  00000               A REKORDOK SZÁMA
DATE OF LAST UPDATE: 85/04/15            UTOLSÓ DÁTUM
PRYMARY USE DATABASE                     ELSŐDLEGES MUNKATÉR
FLD        NAME      TYPE  WIDTH   DEC   MEZŐ NÉV TÍPUS HOSSZ
001      ANYAGNEV      C     015                    TIZEDESEK
002      ERKEZETT      N     004
003      KIADVA        N     004
004      ANYAGKOD      C     003
005      EGYSEGAR      N     006     002
006      OSSZERTEK     N     009     002
**TOTAL**                   00042           **ÖSSZES**

 . use raktar
 . copy to szerkeszt structure extended
00006 RECORDS COPIED                     00006 REKORD MÁSOLVA

 . use szerkeszt
 . display structure
STRUCTURE FOR FILE:  SZERKESZ.DBF        ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS :  00006               A REKORDOK SZÁMA
DATE OF LAST UPDATE: 85/04/15            UTOLSÓ DÁTUM
PRYMARY USE DATABASE                     ELSŐDLEGES MUNKATÉR
FLD        NAME      TYPE  WIDTH   DEC   MEZŐ NÉV TÍPUS HOSSZ
001      FIELD:NAME    C     010                    TIZEDESEK
002      FIELD:TYPE    C     001
003      FIELD:LEN     N     003
004      FIELD:DEC     N     003
**TOTAL**                   00018           **ÖSSZES**
```

COPY

```
. display all
00001   ANYAGNEV    C    15     0
00002   ERKEZETT    N     4     0
00003   KIADVA      N     4     0
00004   ANYAGKOD    C     3     0
00005   EGYSEGAR    N     6     2
00006   OSSZERTEK   N     9     2
```

Idegen rendszerek számára állománynév-kiterjesztéssel és a DELIMITED paraméter használatával adhatunk át adatokat.

(Ha nem vesszőt akarunk határolónak, akkor a parancsban a WITH paramétert követően a karakterhatárolót is meg kell nevezni.)

A következő példában a<Masmilyen.STR>nevű állományba másolunk adatokat.

```
. use raktar
. display all
00001   NEOLUX PIROS        300    200 123    38.50        0.00
00002   NEOLUX KEK          300    100 124    30.00        0.00
00003   VERNODUX KEK        500    300 789    96.10        0.00
00004   NEOLUX ZOLD         350    150 125    28.60        0.00
00005   VERNODUX SARGA      300    300 798    91.10        0.00
00006   VERNODUX PIROS        0      0         0.00        0.00
00007   NEOLUX SARGA        400    200 145    35.90        0.00
00008   NEOLUX FEHER        800    700 100    34.00        0.00

. copy to masmilyen.str delimited
00008 RECORDS COPIED                          00008 REKORD MÁSOLVA
```

Az adatbázis-kezelő rendszerből kilépve adatainkat egy szövegszerkesztő segítségével megtekinthetjük.

Ime az eredmény.

```
'NEOLUX PIROS   ',  300, 200,'123', 38.50,   0.00
'NEOLUX KEK     ',  300, 100,'124', 30.00,   0.00
'VERNODUX KEK   ',  500, 300,'789', 96.10,   0.00
'NEOLUX ZOLD    ',  350, 150,'125', 28.60,   0.00
'VERNODUX SARGA ',  300, 300,'789', 91.10,   0.00
'VERNODUX PIROS ',    0,   0,'   ',  0.00,   0.00
'NEOLUX SARGA   ',  400, 200,'145', 35.90,   0.00
'NEOLUX FEHER   ',  800, 700,'100', 34.00,   0.00
```

Alakja:

COUNT [<érvényességi kör>] [{FOR <kifejezés> / WHILE <kifejezés>}] [TO <memória-⇒ ⇒változó>]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül vagy FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamisra értékelt rekordig minden rekord.

Hatása:

A munkába állított adatbázis-állomány rekordjainak összeszámlálása.

Ismertetése:

A COUNT parancs a FOR vagy a WHILE paraméter megadásakor csak a <kifejezés>-ben szereplő feltételeket kielégítő rekordokat számolja meg az <érvényességi körben>. Paraméter nélkül a törlésre kijelölt rekordok is részt vesznek az összeszámolásban.

A művelet eredményét elhelyezhetjük memóriaváltozóban, melyet a TO paraméter után kell megadnunk. A parancs a végrehajtás során a memóriaváltozót létrehozza, ha az korábban nem létezett.

Példák

```
. use raktar
. display all
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  NEOLUX ZOLD         350   150 125   28.60        0.00
00005  VERNODUX SARGA      300   300 798   91.10        0.00
00006  VERNODIJX PIROS       0     0        0.00        0.00
00007  NEOLUX SARGA        400   200 145   35.90        0.00
00008  NEOLUX FEHER        800   700 100   34.00        0.00

. count
COUNT = 00008                                          ÖSSZES

. count for erkezett>300
COUNT = 00004                                          ÖSSZES

. count for 'NEOLUX' $ anyagnev
COUNT = 00005                                          ÖSSZES
```

COUNT

```
 . count for anyagnev='VERNODUX'
COUNT = 00003                                                  ÖSSZES

 . count for 'KEK' $ anyagnev
COUNT = 00002                                                  ÖSSZES

. count for anyagnev='KEK'
COUNT = 00000                                                  ÖSSZES

 . go top
 . count for erkezett>300 next 6
COUNT = 00002                                                  ÖSSZES

 . go top
 . count next 6 for erkezett>300
COUNT = 00002                                                  ÖSSZES

 . display memory
**TOTAL**  00 VARIABLES USED  00005 BYTES USED
               **ÖSSZESEN**  00 MEMÓRIAVÁLTOZÓ  00000  BYTE

 . count to tetel
COUNT = 00008                                                  ÖSSZES

 . ? tetel
          8

 . display memory
TETEL          (N)     8
**TOTAL**  01 VARIABLES USED  00006 BYTES USED
               **ÖSSZESEN**  01 MEMÓRIAVÁLTOZÓ  00006  BYTE
```

CREATE

Alakja:

a) CREATE [<állománynév>]

b) CREATE <állománynév> FROM <adatbázis-állomány neve>

Hatása:

Új, adatbázis-kezelő rendszer szerkezetű adatbázis-állomány létrehozása.

Ismertetése:

a) A CREATE [<állománynév>] parancs kiadása után a felhasználó határozhatja meg az állomány szerkezetét, a mezők neveit és — ha a parancs nem tartalmazza — az adatbázis-állomány nevét is.

Ha az állomány nevét nem adjuk meg a parancsban, „FILENAME" üzenet jelenik meg a képernyőn. A felhasználónak érvényes állománynevet kell megadnia. Az állománynév csak CP/M rendszerben megengedett speciális karaktereket tartalmazhat. Például a „B:" a B jelű mágneslemez meghajtóegységet jelenti.

Külön figyelmeztetést kapunk a rendszertől, ha már bejegyzett állományra hivatkozunk:

DESTROY EXISTING FILE?

azaz: törölhető a talált adatállomány? A kérdésre az állomány törlésének engedélyezése esetén Y-nal (dBASE II), illetve I-vel (PROP-BASE), ellenkező esetben N-nel kell válaszolni.

Ha az adatbázis-kezelő rendszer számára újnak találtatott az állománynév, vagy az adatállomány törlését engedélyezzük, a következő üzenetre adott válasszal meghatározhatjuk az adatbázis-állomány szerkezetét.

```
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD     NAME, TYPE, WIDTH, DECIMAL PLACES
 001
```

```
REK.SZERKEZET AZ ALABBI SZERINT:
 MEZŐ      NEVE,TIPUSA,HOSSZA,TIZEDES HELYEK
 001
```

Közöljük a mezők nevét és a szerkezetre vonatkozó információkat az üzenet utolsó sorában kezdve!

A mezőnév legfeljebb 10 karakterből álló karakterlánc lehet, melyben betűk, számjegyek és közrezárt kettőspontok szerepelhetnek, de mindig betűvel kell kezdődnie.

A mező típusa egyetlen karakterrel határozható meg. Ezek:

C : karakter,
N : numerikus,
L : logikai.

Ezután a mező hosszát kell megadni. Például egy 55 karakterből álló karakterlánc 55 byte hosszú mezőben fér el.

CREATE

A numerikus adatok vagy egész, vagy tizedes számok. Az egész számokat olyan hosszú mezőben kell elhelyezni, mint a várható legnagyobb szám számjegyeinek száma. A tizedes számokhoz kétféle értéket kell megadni. Az első érték — a hossz megállapításához — az előforduló legtöbb számjegyet tartalmazó szám hossza, beleértve a tizedespontot is. A második érték a tizedespont utáni számjegyek száma.

A logikai típusú adatok 1 byte hosszúságúak.

Az adatokat egymástól vesszővel elválasztva kell megadni, az üzenetben meghatározott sorrendben. A tizedeshelyek számát csak numerikus mező esetén szabad kitölteni, de ott sem kötelező, ha egész számokkal kívánunk csak dolgozni. A mezőt meghatározó adatok bevitelét a <CR> billentyű lenyomásával kell befejezni.

Ha nem szándékozunk több mezőt megnevezni az adatbázisban, akkor az automatikusan megjelenő mezőszámozást követően a <CR> billentyűt kell lenyomni.

Ezután az adatbázis-kezelő rendszer megkérdezi, hogy akarunk-e az adatbázisba adatokat bevinni. Ha azonnal szeretnénk ezt, válaszoljunk Y-nal (dBASE II), illetve I-vel (PROP-BASE), és rögtön megkezdhetjük az adatbevitelt a szerkesztés kódjainak (lásd a 4.6. részt) használatával. Ha nem kívánunk rögtön adatokat bevinni, válaszoljunk N-nel, ekkor az adatbázis-kezelő rendszer alapállapotba kerül.

b) Ha létezik olyan — COPY TO <állománynév> STRUCTURE EXTENDED paranccsal létrehozott — adatbázis-állományunk, amely egy adatbázis struktúráját adatokként tartalmazza rekordjaiban, akkor létrehozhatunk CREATE <állománynév> FROM <adatbázis-állomány neve> paranccsal egy új adatbázis-állományt. Az új adatbázis struktúráját a FROM paraméter után megadott állomány rekordjainak tartalma határozza meg.

Példa

```
 . create raktar
ENTER RECORD STRUCTURE AS FOLLOWS
 FIELD     NAME,TYPE,WIDTH,DECIMAL PLACES
 001       anyagnev,c,15
 002       erkezett,n,4
 003       kiadva,n,4
 004       anyagkod,c,3
 005       egysegar,n,6,2
 006       osszertek,n,9,2
 007       <CR>
INPUT DATA NOW? n

 . create vetelezok
REKORD SZERKEZET AZ ALÁBBI SZERINT :
 MEZŐ NEVE,TIPUSA,HOSSZA,TIZEDES HELYEK
 001       nev,n,15
 002       anyagkod,c,3
```

```
 003      mennyiseg,n,4
 004      datum,c,8
 005      <CR>
ADATBEVITEL    ? n

 .list stru
STRUCTURE FOR FILE:  VETELEZO.DBF     AZ ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS:   00000            A REKORDOK SZÁMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATÉR
FLD       NAME      TYPE  WITH  DEC   MEZŐ NÉV TIPUS HOSSZ
001      NEV            N     015                    TIZEDESEK
002      ANYAGKOD       C     003
003      MENNYISEG      N     004
004      DATUM          C     008
**TOTAL**                    00031     **ÖSSZESEN**
```

Adatfeltöltés (lásd APPEND) után:

```
. display all
00001   KIS PAL          123     20 85/01/11
00002   NAGY ANTAL       100    100 85/01/11
00003   MOLNAR JOZSEF    777     20 85/01/11
00004   KIS PAL          124     40 85/01/12
00005   NAGY ANTAL       789     20 85/01/13
00006   MOLNAR JOZSEF    125     10 85/01/20
00007   KIS PAL          145     20 85/01/13
00008   NAGY ANTAL       100     20 85/01/15
00009   MOLNAR JOZSEF    125     10 85/01/12
```

DELETE

Alakja:

a) DELETE [<érvényességi kör>] {[FOR <kifejezés> / WHILE <kifejezés>]}

b) DELETE FILE <állománynév>

Az <érvényességi kör> alapértelmezése:
(az *a)* formátumnál)
- paraméter nélkül az aktuális rekord,
- FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamisra értékelt rekordig minden rekord.

Hatása:
a) Rekordok megjelölése az adatbázis-állományban későbbi törlés céljából.
b) A megadott állomány törlése a nyilvántartásból.

Ismertetése:
a) Az <érvényességi kör>-nek és a FOR, illetve a WHILE paraméter mögötti kifejezésben adott feltételeknek megfelelő összes rekordot megjelöli. (Alapértelmezés szerint csak az aktuális rekordot.) A tényleges fizikai törlést csak a PACK parancs végzi el.
A törlendő rekordokat a másoló (pl. COPY stb.), a bővítő (pl. APPEND, JOIN, TOTAL stb.), a sorbarendező (SORT stb.) parancsok, a jelentést készítő parancs (REPORT) és a műveletek figyelmen kívül hagyják.
A későbbi törlésre vonatkozó megjelölés megszüntethető a RECALL parancs kiadásával.
A megjelölt rekordok a képernyőre, illetve a nyomtatóra írathatók. Az ilyen rekordoknál a rekordszám mögött egy „*" jelenik meg. A „*" jelenléte vagy hiánya a parancsok FOR és WHILE részeinél külön logikai mezőként kezelhető.
(Pl. „LIST FOR *" , "COUNT WHILE .NOT. *")

b) A megnevezett állományt a mágneslemez nyilvántartásából törli, és az általa foglalt mágneslemez-területet az operációs rendszer számára felszabadítja. Nem törli azonban az állományt, ha azt éppen munkaállományként kezeljük.

Példák

```
. use raktar
. display all
00001   NEOLUX PIROS        300    200 123    38.50        0.00
00002   NEOLUX KEK          300    100 124    30.00        0.00
00003   VERNODUX KEK        500    300 789    96.10        0.00
00004   NEOLUX ZOLD         350    150 125    28.60        0.00
00005   VERNODUX SARGA      300    300 798    91.10        0.00
```

```
00006  VERNODUX PIROS       0      0          0.00          0.00
00007  NEOLUX SARGA       400    200 145     35.90          0.00
00008  NEOLUX FEHER       800    700 100     34.00          0.00

 . go 2
 . delete
00001 DELETION(S)                                    00001 TÖRLÉS

 . go 6
 . delete next 2
00002 DELETION(S)                                    00002 TÖRLÉS

 . display all
00001  NEOLUX PIROS       300    200 123     38.50          0.00
00002 *NEOLUX KEK         300    100 124     30.00          0.00
00003  VERNODUX KEK       500    300 789     96.10          0.00
00004  NEOLUX ZOLD        350    150 125     28.60          0.00
00005 *VERNODUX SARGA     300    300 798     91.10          0.00
00006 *VERNODUX PIROS       0      0          0.00          0.00
00007  NEOLUX SARGA       400    200 145     35.90          0.00
00008  NEOLUX FEHER       800    700 100     34.00          0.00

 . recall all
00003 RECALL(S)                                  00003 VISSZAÁLLÍTÁS

 . display all
00001  NEOLUX PIROS       300    200 123     38.50          0.00
00002  NEOLUX KEK         300    100 124     30.00          0.00
00003  VERNODUX KEK       500    300 789     96.10          0.00
00004  NEOLUX ZOLD        350    150 125     28.60          0.00
00005  VERNODUX SARGA     300    300 798     91.10          0.00
00006  VERNODUX PIROS       0      0          0.00          0.00
00007  NEOLUX SARGA       400    200 145     35.90          0.00
00008  NEOLUX FEHER       800    700 100     34.00          0.00

 . display files
DATABASE FILES    # RCDS     LAST UPDATE
                        ADATBÁZISOK     REKORDOK     UTOLSÓ DÁTUM
RAKTAR    DBF       00008      85/04/15
UJANYAG   DBF       00002      85/04/15
```

DELETE

```
VETELEZO DBF      00009      85/04/15
BOVITES  DBF      00000      85/04/15
RAKTAR2  DBF      00008      85/04/15
RAKTAR3  DBF      00004      85/04/15
RAKTAR4  DBF      00004      85/04/15
RAKTAR5  DBF      00004      85/04/15
SZERKEZE DBF      00000      85/04/15
SZERKESZ DBF      00006      85/04/15

 . delete file raktar5
FILE HAS BEEN DELETED                          AZ ALLOMANY TÖRÖLVE

 . delete file raktar4
FILE HAS BEEN DELETED                          AZ ALLOMANY TÖRÖLVE

 . delete file raktar3
FILE HAS BEEN DELETED                          AZ ALLOMANY TÖRÖLVE

 . delete file raktar2
FILE HAS BEEN DELETED                          AZ ALLOMANY TÖRÖLVE

 . display files
DATABASE FILES    # RCDS     LAST UPDATE
                       ADATBAZISOK     REKORDOK     UTOLSÓ DATUM
RAKTAR    DBF     00008      85/04/15
UJANYAG   DBF     00002      85/04/15
VETELEZO  DBF     00009      85/04/15
BOVITES   DBF     00000      85/04/15
SZERKEZE  DBF     00000      85/04/15
SZERKESZ  DBF     00006      85/04/15
```

Alakja:

a) DISPLAY [<érvényességi kör>] [{FOR <kifejezés> / WHILE <kifejezés>}] [<kifejezéslista>] [OFF]

b) DISPLAY STRUCTURE

c) DISPLAY MEMORY

d) DISPLAY FILES [ON <lemezegység>] [LIKE <rendszer szerinti elnevezés>]

Az <érvényességi kör> alapértelmezése:
(az *a)* formátumnál)
- paraméter nélkül egy rekord,
- FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamisra értékelt rekordig minden rekord.

Hatása:

a) A munkaállomány rekordjainak megjelenítése.

b) A munkaállomány szerkezetének megjelenítése.

c) A memóriaváltozók kiírása.

d) A mágneslemezen levő állománynyilvántartás tartalmának közlése.

Ismertetése:

a) A munkaállomány teljes vagy részleges megjelenítésére szolgál. Az <érvényességi kör> és a FOR <kifejezés> parancsrészek mindegyikének hiánya esetén csak az aktuális rekord adataival dolgozik. Az <érvényességi kör> nélkül, de a FOR <kifejezés> parancsrésszel az adatbázis-állomány összes rekordjának adatait tekintetbe veszi.
Az összes mező megjelenik, hacsak a <kifejezéslista> részt nem adjuk meg. Megengedett a lista kifejezéseiben adatmezőket, memóriaváltozókat vagy bármilyen — egyébként is megengedett — numerikus karakter vagy logikai értékeket (literált, állandót), beépített függvényeket, műveleteket használni.
Az OFF parancsrész megadásával a rekordok tartalma előtt megjelenő rekordszám kiírása megszűnik.
A FOR, illetve a WHILE paraméter megadásakor csak a <kifejezés>-ben adott feltételt teljesítő rekordokat veszi figyelembe a parancs.
Minden 15 rekordnyi kiírás után megáll a megjelenítés. A folytatáshoz bármelyik billentyű leütése elegendő.
A LIST parancs azonos a DISPLAY paranccsal, kivéve, hogy az előbbi nem áll meg 15 rekordonként, s alapértelmezésben minden rekordot tekintetbe vesz. Az <ESC> billentyű leütése mindkét parancs azonnali megszakítását eredményezi, és az adatbázis-kezelő rendszer alapállapotba tér vissza.

b) Csak az aktuális munkatérben levő adatbázis-állomány szerkezetéről közöl információt.

c) Az éppen érvényben levő memóriaváltozókat név és hozzátartozó érték megadásával listázza.

DISPLAY

d) Az alap- vagy a megjelölt mágneslemez ([ON <lemezegység>]) nyilvántartásában levő adatbázis-állományok név szerinti megjelenítésére szolgál. Néhány statisztikai adatot is megad.

A LIKE <rendszer szerinti elnevezés> megadásával más, nem csak adatbázis-állomány felsorolását kapjuk. A <rendszer szerinti elnevezés> leggyakoribb formája a „*.típus", ahol a típus a TXT, az FRM, a MEM vagy bármely más, legfeljebb három karakter. Ezeket az állományokat a parancs úgy listázza, mint ahogy a CP/M DIR parancsa.

Példák

```
. use raktar
. display structure
STRUCTURE FOR FILE:  RAKTAR.DBF       AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00008            REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATÉR
FLD      NAME       TYPE  WIDTH  DEC  MEZŐ NÉV TIPUS HOSSZ
001      ANYAGNEV     C     015                    TIZEDESEK
002      ERKEZETT     N     004
003      KIADVA       N     004
004      ANYAGKOD     C     003
005      EGYSEGAR     N     006   002
006      OSSZERTEK    N     009   002
** TOTAL**                 00042       **ÖSSZESEN**

. list structure
STRUCTURE FOR FILE:  RAKTAR.DBF       AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00008            REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATÉR
FLD      NAME       TYPE  WIDTH  DEC  MEZŐ NÉV TIPUS HOSSZ
001      ANYAGNEV     C     015                    TIZEDESEK
002      ERKEZETT     N     004
003      KIADVA       N     004
004      ANYAGKOD     C     003
005      EGYSEGAR     N     006   002
006      OSSZERTEK    N     009   002
** TOTAL**                 00042       **ÖSSZESEN**
```

```
   display all
00001  NEOLUX PIROS       300    200 123    38.50        0.00
00002  NEOLUX KEK         300    100 124    30.00        0.00
00003  VERNODUX KEK       500    300 789    96.10        0.00
00004  NEOLUX ZOLD        350    150 125    28.60        0.00
00005  VERNODUX SARGA     300    300 798    91.10        0.00
00006  VERNODUX PIROS       0      0         0.00        0.00
00007  NEOLUX SARGA       400    200 145    35.90        0.00
00008  NEOLUX FEHER       800    700 100    34.00        0.00

 . list
00001  NEOLUX PIROS       300    200 123    38.50        0.00
00002  NEOLUX KEK         300    100 124    30.00        0.00
00003  VERNODUX KEK       500    300 789    96.10        0.00
00004  NEOLUX ZOLD        350    150 125    28.60        0.00
00005  VERNODUX SARGA     300    300 798    91.10        0.00
00006  VERNODUX PIROS       0      0         0.00        0.00
00007  NEOLUX SARGA       400    200 145    35.90        0.00
00008  NEOLUX FEHER       800    700 100    34.00        0.00

 . go top
 . display
00001  NEOLUX PIROS       300    200 123    38.50        0.00

 . list next 1
00001  NEOLUX PIROS       300    200 123    38.50        0.00

 . go top
 . display next 3
00001  NEOLUX PIROS       300    200 123    38.50        0.00
00002  NEOLUX KEK         300    100 124    30.00        0.00
00003  VERNODUX KEK       500    300 789    96.10        0.00

 . list anyagnev
00001  NEOLUX PIROS
00002  NEOLUX KEK
00003  VERNODUX KEK
00004  NEOLUX ZOLD
```

DISPLAY

```
00005  VERNODUX SARGA
00006  VERNODUX PIROS
00007  NEOLUX SARGA
00008  NEOLUX FEHER

 . go 4
 . display anyagnev
00004  NEOLUX ZOLD

 . go 5
 . list anyagnev, anyagkod
00001  NEOLUX PIROS     123
00002  NEOLUX KEK       124
00003  VERNODUX KEK     789
00004  NEOLUX ZOLD      125
00005  VERNODUX SARGA   798
00006  VERNODUX PIROS
00007  NEOLUX SARGA     145
00008  NEOLUX FEHER     100

 . list anyagnev for erkezett>300
00003  VERNODUX KEK
00004  NEOLUX ZOLD
00007  NEOLUX SARGA
00008  NEOLUX FEHER

 . display for erkezett>300
00003  VERNODUX KEK       500    300 789    96.10        0.00
00004  NEOLUX ZOLD        350    150 125    28.60        0.00
00007  NEOLUX SARGA       400    200 145    35.90        0.00
00008  NEOLUX FEHER       800    700 100    34.00        0.00

 . display anyagnev for erkezett>300
00003  VERNODUX KEK
00004  NEOLUX ZOLD
00007  NEOLUX SARGA
00008  NEOLUX FEHER
```

```
 . list for erkezett>300 off
VERNODUX KEK        500    300 789    96.10         0.00
NEOLUX ZOLD         350    150 125    28.60         0.00
NEOLUX SARGA        400    200 145    35.90         0.00
NEOLUX FEHER        800    700 100    34.00         0.00

 . use raktar index keszleten
 . list
00005  VERNODUX SARGA     300    300 798    91.10         0.00
00001  NEOLUX PIROS       300    200 123    38.50         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00
00002  NEOLUX KEK         300    100 124    30.00         0.00
00003  VERNODUX KEK       500    300 789    96.10         0.00
00004  NEOLUX ZOLD        350    150 125    28.60         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00006  VERNODUX PIROS     500    200 777    81.10         0.00

 . list anyagnev, erkezett-kiadva
00005  VERNODUX SARGA        0
00001  NEOLUX PIROS        100
00008  NEOLUX FEHER        100
00002  NEOLUX KEK          200
00003  VERNODUX KEK        200
00004  NEOLUX ZOLD         200
00007  NEOLUX SARGA        200
00006  VERNODUX PIROS      300

 . use  raktar
 . list for anyagnev='NEOLUX'
00001  NEOLUX PIROS       300    200 123    38.50         0.00
00002  NEOLUX KEK         300    100 124    30.00         0.00
00004  NEOLUX ZOLD        350    150 125    28.60         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00

 . store 'VERNODUX' to festek
VERNODUX
```

DISPLAY

```
. list for anyagnev=festek
00003  VERNODUX KEK       500    300 789    96.10        0.00
00005  VERNODUX SARGA     300    300 798    91.10        0.00
00006  VERNODUX PIROS       0      0          0.00        0.00

. display memory
FESTEK        (C)  VERNODUX
**TOTAL**  01 VARIABLES USED  00008 BYTES USED
                    **ÖSSZESEN  01 MEMÓRIAVÁLTOZÓ  00008 BYTE

. list memory
FESTEK        (C)  VERNODUX
**TOTAL**  01 VARIABLES USED  00008 BYTES USED
                    **ÖSSZESEN  01 MEMÓRIAVÁLTOZÓ  00008 BYTE

. display files
DATABASE FILES    # RCDS     LAST UPDATE
                       ADATBÁZIS-ÁLL.  REKORDOK   UTOLSÓ DÁTUM
RAKTAR    DBF     00008      85/04/15
UJANYAG   DBF     00002      85/04/15
VETELEZO  DBF     00009      85/04/15
BOVITES   DBF     00000      85/04/15
SZERKEZE  DBF     00000      85/04/15
SZERKESZ  DBF     00006      85/04/15

. list files
DATABASE FILES    # RCDS     LAST UPDATE
                       ADATBÁZIS-ÁLL.  REKORDOK   UTOLSÓ DÁTUM
RAKTAR    DBF     00008      85/04/15
UJANYAG   DBF     00002      85/04/15
VETELEZO  DBF     00009      85/04/15
BOVITES   DBF     00000      85/04/15
SZERKEZE  DBF     00000      85/04/15
SZFRKESZ  DBF     00006      85/04/15

. list files like *.*
RAKTAR   .DBF     UJANYAG .DBF    VETELEZO.DBF    BOVITES .DBF
PIP      .COM     STAT    .COM    MUNKA   .TXT    RAKTARAM.NDX
SZERKEZE.DBF      SZERKESZ.DBF    LEVEL   .SZF    ADAT    .SDF
Z80TRNS .PAS      PAR     .PAR
```

Alakja:

a) DO <állomány>

b) DO WHILE <kifejezés>
    parancssor(ok)
ENDDO

c) DO CASE
    CASE <kifejezés>
        parancssor(ok)
    [CASE <kifejezés>
        parancssor(ok)]
    [CASE
        .
        .
        .]
    [OTHERWISE
        parancssor(ok)]
ENDCASE

Hatása:

a) Parancsállományban rögzített felhasználói program végrehajtásának indítása, áttérés közvetett módba.

b) Parancssorozat feltételes, ismételt végrehajtása.

c) Feltételes végrehajtást eredményező összetett szerkezet.

Ismertetése:

a) A DO <állomány> parancs az <állomány>-t megnyitja, és parancsállományként <.CMD> használja.
Az állomány tartalma adatbázis-kezelő rendszerparancsok sorozata, melyeket az adatbázis-kezelő úgy értelmez és hajt végre, mintha közvetlenül a billentyűzetről érkeznének.
A DO <állomány> parancsok 16 szintig egymásbaágyazhatók. (A parancsállományok tartalmazhatnak DO parancsokat, melyekkel újabb parancsállományok végrehajtását indítják.)
Egy parancsállomány végrehajtása befejeződik az állomány végének elérésekor vagy a RETURN parancs hatására. Ha egy végrehajtás alatt álló parancsállományba egy másik állományból — aktivizálással — jutottunk, a végrehajtás befejezésekor a RETURN parancsra az adatbázis-kezelő rendszer a vezérlést az aktivizáló állománynak adja vissza, a „DO <állománynév>" parancsot követő sorra.
Bármely parancsállomány végrehajtása során előforduló CANCEL parancs hatására az összes parancsállomány lezárásával közvetlen módba jutunk.

b) A DO és az ENDDO közötti parancssorozat végrehajtása mindaddig ismétlődik, amíg a <kifejezés> logikai értéke igaz. Hamis érték esetén a vezérlés az ENDDO után következő parancssorra lép. Ügyeljünk a parancsok egymásba ágyazásának szabályaira!
(V. ö. 2.9. rész)

c) A DO CASE szerkezetben a CASE ágak száma nem korlátozott, az OTHERWISE ág nem kötelező.

A CASE ágak tekinthetők úgy, mint az OTHERWISE ágban végrehajtandók alóli kivételek. Ha bizonyos feladatok speciális feltétel szerinti végrehajtást igényelnek, ezek egy-egy CASE ágba kerülnek, míg minden más feltétel esetén az OTHERWISE ág szerinti feladatot kell elvégeznie az adatbázis-kezelő rendszernek.

A DO CASE szerkezet végrehajtása legjobban IF feltételes elágazások sorozatával szemléltethető. Az adatbázis-kezelő rendszer úgy hajtja végre a DO CASE szerkezetet, mintha IF—ENDIF feltételes elágazások sorozata lenne.

```
DO CASE
   CASE VALASZ = 'igen'
      parancssor(ok)
   CASE VALASZ = 'nem'
      parancssor(ok)
   OTHERWISE
      parancssor(ok)
ENDCASE
```

```
IF VALASZ = 'igen'
   parancssor(ok)
ELSE
   IF VALASZ = 'nem'
      parancssor(ok)
   ELSE
      (parancssor(ok)
   ENDIF
ENDIF
```

Tehát az egyes CASE ágak vizsgálata során elsőként igaz eredményt adó kifejezés által meghatározott ág végrehajtása után az ENDCASE-re ugrik a vezérlés. Több *igaz* ág közül is csak az *első* lesz végrehajtva.

Az OTHERWISE ág végrehajtására akkor jut a vezérlés, ha nincs egyetlen logikai igaz CASE ág sem.

Ha nincs egyetlen logikai igaz CASE ág sem, és nincs OTHERWISE ág sem, akkor a rendszer a DO CASE szerkezet egyetlen parancsát sem hajtja végre.

A DO CASE és az első CASE közé írt parancsok végrehajtása soha nem történik meg.

DO CASE szerkezeteket egymásba ágyazni nem szabad.

Példák

1. A következő rövid, *b)* parancsformátum szerinti példában az adatbázis-kezelő rendszer kiírja a kimeneti egységre az aktuális rekord tartalmát, majd a következő rekordra lépteti a rekordmutatót. Ez a tevékenység az adatbázis-állomány végéig ismétlődik (.not. eof).

```
DO WHILE .NOT. EOF
   DISPLAY ANYAGNEV
   SKIP
ENDDO
```

2. A *c)* parancsformátum szerinti első példában az aktuális rekord tartalmától függően más és más információk kerülnek a kimeneti egységre. A második példában az adatbázis-állomány végének elérésekor a memóriaváltozók értéke szabja meg a döntést.

```
DO CASE
  CASE ANYAGNEV='NEOLUX'
    DISPLAY ANYAGNEV, EGYSEGAR OFF
  CASE ANYAGNEV='VERNODUX'
    DISPLAY ANYAGNEV, ANYAGKOD OFF
  CASE 'PIROS' $ ANYAGNEV
    DISPLAY OFF
  OTHERWISE
    ? 'Nincs ilyen'
ENDCASE
```

```
DO CASE
  CASE .NOT. EOF
    ? 'Van meg adat.'
  CASE 3 = 2 + SZAM
    DO SZAMOLAS
  CASE 'A' $ SZOVEG
    ? 'Igaz'
ENDCASE
```

EDIT

Alakja:

EDIT [n]

Hatása:

Az adatbázismezőkben levő adatok módosítása.

Ismertetése:

A parancs lehetőséget nyújt az adatmezők tartalmának közvetlen megváltoztatására.
Ha SET SCREEN állapotban (alapbeállításban) van az adatbázis-kezelő rendszer, akkor az ^EDIT n^ parancsra megjelenik a képernyőn az n. rekord az ^APPEND^ parancs hatásához hasonlóan. A javításokat a 4.6. részben ismertetett kódok szerint végezhetjük el.
Ha SET SCREEN OFF állapotban vagyunk már, nem használhatjuk ki a teljes képernyős szerkesztés lehetőségeit. ^EDIT^ parancs után a gép kérdésére meg kell adni a rekord számát, a mező nevét vagy számát és az új értéket. Ha az új értéket is közöljük, akkor az adatbázis-kezelő kijelzi az aktuális értéket, majd megkérdezi, akarunk-e még ezen változtatni. Ha a javítás így már megfelel, a <CR> billentyűt kell leütni. Miután a koordinátákat egyszer megadtuk, bármelyiket elhagyhatjuk, az adatbázis-kezelő rendszer a korábban közölt értéket veszi alapul. Minden rekord javítása után megismétlődik a koordinátákra vonatkozó kérdés. Ahhoz, hogy az EDIT módot elhagyhassuk, a koordináták helyett <CR>-t kell begépelni.
Ha USE-zal vagy SET INDEX TO paranccsal aktivizáljuk az adatbázis-állományhoz tartozó indexállományt, akkor az indexállomány a javításoknak megfelelő sorrendet fogja mutatni. Egyéb indexállományokat újabb INDEX-paranccsal kell aktualizálni.

* * *

Az EDIT és SET SCREEN ON végrehajtási ideje alatt érvényes szerkesztési kódok

^X (CAN) a fénypontot lefelé mozgatja a következő mezőre.
(Alkalmazható erre a célra a ^F (ACK) is.)

^E (ENQ) a fénypontot visszafelé mozgatja az előző mezőre.
(Alkalmazható erre a célra a ^A (SOH) is.)

^D (EOT) a fénypontot előre tolja a következő karakterre.

^S (DC3) egy karakternyit visszalépteti a fénypontot.

^V (SYN) kapcsoló a felülírási és a beszúrási mód között.

^G (BEL) a fénypont alatti karaktert törli.

<Rubout> (DEL) a fényponttól balra álló karaktert törli.

^Y (EM) törli az aktuális mező tartalmát.

^U (NAK) átkapcsoló az aktuális rekord törlésre jelölésére, illetve a jelölés megszüntetésére.

^C (ETX) a rekordot mágneslemezre írja, és — indexelt adatbázis esetén az index szerinti — következő rekordra áll.

^R (DC2) mágneslemezre írja a rekordot, és visszamegy — indexelt adatbázis esetén az index szerinti — előző rekordra.

^Q (DC1) nem veszi figyelembe az aktuális rekordon végzett változtatásokat, és visszatér alapállapotba.

^W (ETB) az adatbázis-kezelő rendszer a szerkesztésben végzett minden változtatást figyelembe vesz, és visszatér alapállapotba.
(Alkalmazható erre a célra a ^O (SI) is.)

MEGJEGYZÉS

Az EDIT szerkesztési módban a 0-tól 9-ig terjedő számjegyek, a tizedespont, a „–" és a „+" jel mellett a szóközt is használhatjuk numerikus adatok bevitelénél.
A vezérlőkarakterek — az adatbázis-kezelő rendszer alapállapotában — ugyanezeknek a kódoknak egy részét más feladatok elvégzésére használják. (V. ö. a 3.1. résszel.)

Példák

```
. use raktar
. list
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  NEOLUX ZOLD         350    150 125    28.60        0.00
00005  VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS        0      0         0.00        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00

. edit
RECORD#00006                                    REKORD#00006
ANYAGNEV:VERNODUX PIROS   :
ERKEZETT :     0:
KIADVA   :     0:
```

EDIT

```
ANYAGKOD :      :
EGYSEGAR :  0.00:
OSSZERTEK:      0.00:
```

(A szerkesztési módban írtuk be:

```
                500
                200
              777
                  81.10
          ^W          )
```

```
 . list
00001  NEOLUX PIROS       300   200 123   38.50        0.00
00002  NEOLUX KEK         300   100 124   30.00        0.00
00003  VERNODUX KEK       500   300 789   96.10        0.00
00004  NEOLUX ZOLD        350   150 125   28.60        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00008  NEOLUX FEHER       800   700 100   34.00        0.00
```

EJECT

Alakja:

EJECT

Hatása:

Új lap kezdése a nyomtatón.

Ismertetése:

Ha SET PRINT ON (vagy ^P), illetve SET FORMAT TO PRINT paranccsal úgy rendelkeztünk, hogy a nyomtatót használjuk, az EJECT parancs lapemelést hajt végre, azaz a következő lap elejére állítja be a papírt a nyomtatáshoz. Ezen túl — a @ jelű parancs számára — a sor- és az oszlopszámlálók értékét is zérusra állítja.

ELSE

Alakja:

ELSE

Szerepe:

Az IF szerkezet egyik ága.

Ismertetése:

Az IF utáni kifejezés hamis logikai értéke esetén az ELSE ágban meghatározott parancsokat hajtja végre a rendszer. Az ELSE ág elhagyható.

(Lásd az IF parancsot.)

ENDCASE

Alakja:

ENDCASE

Szerepe:

A DO CASE szerkezet lezárása.

Ismertetése:

Az első igaz logikai értékű CASE ágban vagy — ha létezik — az OTHERWISE ágban meghatározott műveletek elvégzése után a felhasználói program futása az ENDCASE-t követő parancssorban folytatódik.

(Lásd a DO parancsot.)

ENDDO

Alakja:

ENDDO

Szerepe:

A DO WHILE ciklus lezárása.

Ismertetése:

Hatására a felhasználói program végrehajtása a DO WHILE ciklus első sorára tér vissza, ahol a <kifejezés> logikai értékét ismételten meghatározza az adatbázis-kezelő rendszer.
(Lásd a DO parancsot.)

ENDIF

Alakja:

ENDIF

Szerepe:

Az IF szerkezet lezárása.

Ismertetése:

A feltételek szerint az IF ágban vagy — ha létezik — az ELSE ágban meghatározott műveletek elvégzése után a felhasználói program futása az ENDIF-et követő sorban folytatódik. (Lásd az IF parancsot.)

ERASE

Alakja:

ERASE

Hatása:

A képernyő törlése.

Ismertetése:

A képernyő törlése után a fénypontot a képernyő bal felső sarkába állítja.

Ha SET SCREEN ON beállítás van érvényben, és @ jelű parancsokat használtunk, az ERASE utasítás e @ jelű parancsok GET és PICTURE részeire vonatkozó információkat törli az adatbázis-kezelő rendszer belső memóriájából.

Alakja:

FIND <kifejezés értéke>

Hatása:

INDEX paranccsal rendezett adatbázisban keresi azt az első rekordot, amelynél az indexelési kulcs alapján adódó érték megegyezik a <kifejezés értéke>-vel.

Ismertetése:

A FIND parancs végrehajtása nagyon gyors. Hajlékony mágneslemezzel dolgozó rendszerekben átlagosan 2 másodperc keresési idővel dolgozik.

Segítségével csak indexelt állományokban kereshetünk.

A FIND az első olyan rekordot keresi meg, amelynél az indexelési kulcs alapján adódó pillanatnyi érték megegyezik a <kifejezés értéke>-vel. Mindig az *első* ilyen rekordot keresi, még akkor is, ha a parancs kiadásakor nem az állomány elején áll az adatbázis-kezelő rendszer!

Ha az INDEX parancsban használt kulcs eredménye – rekordonként persze más és más – karakterlánc; a FIND akkor is helyesen dolgozik, ha a keresett <kifejezés értéke> csak eleje az indexelési kulcs alapján adódó pillanatnyi értéknek.

MEGJEGYZÉS

Ha az indexelési kulcs alapján adódó karakterlánc elején üres helyek vannak, akkor a <kifejezés értéke> is ugyanennyi üres hellyel (szóközzel) kell hogy kezdődjön. Ilyenkor az üres helyek definiálásának érdekében a <kifejezés értéke>-nek idézőjelek vagy aposztrófok között kell lennie.

Numerikus eredményt adó kulcs szerinti indexelésnél a FIND azt a rekordot keresi, melynél az indexelési kulcs alapján adódó numerikus érték megegyezik a <kifejezés értéke>-vel.

A keresendő értéket memóriaváltozó is tartalmazhatja. Ekkor makró-helyettesítés (&) szükséges. Karaktertípusú memóriaváltozók közvetlenül, numerikus típusúak a STR függvény által átalakítva használhatók.

Egy „megtalált'' rekord ugyanúgy kezelhető, mint bármely más adatbázisrekord.

A sikertelen keresést a képernyőre írt „NO FIND'' üzenet jelzi. Ilyenkor a rekordszám beépített függvény (#), mivel nincs aktuális rekord, 0 értéket ad.

Több, az indexelési kulcs szerint azonos értéket tartalmazó rekordot a SKIP vagy a ^LOCATE FOR <kifejezés>^ parancs használatával kereshetünk meg. A SKIP hátránya, hogy nem ismeri fel az eltérő értéket.

A SET EXACT ON beállítás hatására a FIND csak akkor jelez sikeres keresést, ha a keresett érték karakterről karakterre megegyezik az indexelési kulcs szerinti karakterlánccal. Az esetleges záró üres helyeket a rendszer figyelmen kívül hagyja (v. ö. a SET paranccsal, 232. old.).

FIND

Példák

```
. use raktar
. list stru
STRUCTURE FOR FILE:  RAKTAR.DBF       AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00008            A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSÖDLEGES MUNKATER:
FLD      NAME      TYPE  WIDTH  DEC  MEZÖ NÉV TIPUS HOSSZ
001      ANYAGNEV    C     015                      TIZEDESEK
002      ERKEZETT    N     004
003      KIADVA      N     004
004      ANYAGKOD    C     003
005      EGYSEGAR    N     006   002
006      OSSZERTEK   N     009   002
**TOTAL**                 00042        **ÖSSZESEN**

.list
00001  NEOLUX PIROS       300   200 123   38.50        0.00
00002  NEOLUX KEK         300   100 124   30.00        0.00
00003  VERNODUX KEK       500   300 789   96.10        0.00
00004  NEOLUX ZOLD        350   150 125   28.60        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00008  NEOLUX FEHER       800   700 100   34.00        0.00

. index on anyagnev to raktaram
00008 RECORDS INDEXED                  00008 REKORD INDEXELVE

. list
00008  NEOLUX FEHER       800   700 100   34.00        0.00
00002  NEOLUX KEK         300   100 124   30.00        0.00
00001  NEOLUX PIROS       300   200 123   38.50        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00004  NEOLUX ZOLD        350   150 125   28.60        0.00
00003  VERNODUX KEK       500   300 789   96.10        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
```

FIND

```
 . find NEOLUX P
 . disp
00001  NEOLUX PIROS       300    200 123    38.50         0.00

 . find vernodux k
NO FIND                                                  NINCS

 . find VERNODUX K
 . disp
00003  VERNODUX KEK       500    300 789    96.10         0.00

 . ? #
     3

 . ? eof
 .F.

 . find neolux
NO FIND                                                  NINCS

 . ? #
     0

 . ? eof
 .F.

 . use raktar
 . index on len(trim(anyagnev)) to hossz
00008 RECORDS INDEXED                    00008 REKORD INDEXELVE

 . list
00002  NEOLUX KEK         300    100 124    30.00         0.00
00004  NEOLUX ZOLD        350    150 125    28.60         0.00
00001  NEOLUX PIROS       300    200 123    38.50         0.00
00003  VERNODUX KEK       500    300 789    96.10         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00
00005  VERNODUX SARGA     300    300 798    91.10         0.00
00006  VERNODUX PIROS     500    200 777    81.10         0.00
```

FIND

```
 . find 12
 . disp
00001  NEOLUX PIROS       300    200 123    38.50         0.00

 . find 13
NO FIND                                                  NINCS

 . disp
 . find 14
 . disp
00005  VERNODUX SARGA     300    300 798    91.10         0.00
 . use raktar
 . index on erkezett-kiadva to keszleten
00008 RECORDS INDEXED                      00008 REKORD INDEXELVE

 . list
00005  VERNODUX SARGA     300    300 798    91.10         0.00
00001  NEOLUX PIROS       300    200 123    38.50         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00
00002  NEOLUX KEK         300    100 124    30.00         0.00
00003  VERNODUX KEK       500    300 789    96.10         0.00
00004  NEOLUX ZOLD        350    150 125    28.60         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00006  VERNODUX PIROS     500    200 777    81.10         0.00

 . find 100
 . disp
00001  NEOLUX PIROS       300    200 123    38.50         0.00

 . list anyagnev, erkezett-kiadva
00005  VERNODUX SARGA        0
00001  NEOLUX PIROS        100
00008  NEOLUX FEHER        100
00002  NEOLUX KEK          200
00003  VERNODUX KEK        200
00004  NEOLUX ZOLD         200
00007  NEOLUX SARGA        200
00006  VERNODUX PIROS      300
```

```
 . use raktar
 . index on $(anyagnev,(@(' ',anyagnev))+1,10) to szin
00008 RECORDS INDEXED                    00008 REKORD INDEXELVE

 . list
00008  NEOLUX FEHER       800   700 100   34.00        0.00
00002  NEOLUX KEK         300   100 124   30.00        0.00
00003  VERNODUX KEK       500   300 789   96.10        0.00
00001  NEOLUX PIROS       300   200 123   38.50        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00004  NEOLUX ZOLD        350   150 125   28.60        0.00

 . find PIROS
 . disp
00001  NEOLUX PIROS       300   200 123   38.50        0.00

 . store 'SARGA' to szine
SARGA

 . find &szine
 . disp
00005  VERNODUX SARGA     300   300 798   91.10        0.00
```

GO vagy GOTO

Alakja:

a) GO[TO] TOP
b) GO[TO] BOTTOM
c) GO[TO] [RECORD] <szám>
d) <szám>
e) GO[TO] [RECORD] <memóriaváltozó>

Hatása:

a) A rekordmutató mozgatása az adatbázis *első elemére.*
b) A rekordmutató mozgatása az adatbázis *utolsó elemére.*
c) d) e) A rekordmutató mozgatása az adatbázis jelölt számú rekordjára.

Ismertetése:

Az adatbázis rekordmutatójának mozgatására, beállítására használjuk.

a) b) A GO[TO] TOP és a GO[TO] BOTTOM parancsnál, ha az állomány indexelt adatbázis, az első, illetve az utolsó rekord nem feltétlenül jelenti fizikailag is az első, illetve az utolsó helyen szereplő rekordot. Ilyenkor a sorrendet az indexelési kulcsnak megfelelően kell értelmeznünk.

c) d) A GO[TO] [RECORD] <szám> és a csak <szám> parancsok esetében közvetlenül,

e) a GO[TO] [RECORD] <memóriaváltozó> parancs esetében egy memóriaváltozó közvetítésével adjuk meg az elérni kívánt rekord számát.

Példák

```
. use raktar
. list
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  NEOLUX ZOLD         350    150 125    28.60        0.00
00005  VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS      500    200 777    81.10        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00

. go top
. disp
00001  NEOLUX PIROS        300    200 123    38.50        0.00

. go bottom
. display
00008  NEOLUX FEHER        800    700 100    34.00        0.00
```

```
 . go7
*** UNKNOWN COMMAND                    *** ISMERETLEN PARANCS
go7
CORRECT AND RETRY (Y/N)? y             JAVIT ÉS ISMÉTEL (I/N)
CHANGE FROM : o7                                MIT   JAVIT :
CHANGE TO   : o 7                               MIRE JAVITJA:
go 7
MORE CORRECTIONS (Y/N)? n              UJABB JAVITÁSOK  (I/N)?

 . disp
00007  NEOLUX SARGA       400   200 145   35.90        0.00

 . go 4
 . list next 10
00004  NEOLUX ZOLD        350   150 125   28.60        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00008  NEOLUX FEHER       800   700 100   34.00        0.00

 . go 4
 . disp
00004  NEOLUX ZOLD        350   150 125   28.60        0.00

 . list
00001  NEOLUX PIROS       300   200 123   38.50        0.00
00002  NEOLUX KEK         300   100 124   30.00        0.00
00003  VERNODUX KEK       500   300 789   96.10        0.00
00004  NEOLUX ZOLD        350   150 125   28.60        0.00
00005  VERNODUX SARGA     300   300 798   91.10        0.00
00006  VERNODUX PIROS     500   200 777   81.10        0.00
00007  NEOLUX SARGA       400   200 145   35.90        0.00
00008  NEOLUX FEHER       800   700 100   34.00        0.00

 . go bottom
 . display
00008  NEOLUX FEHER       800   700 100   34.00        0.00
```

GO vagy GOTO

```
 . list
00001  NEOLUX PIROS        300    200 123    38.50         0.00
00002  NEOLUX KEK          300    100 124    30.00         0.00
00003  VERNODUX KEK        500    300 789    96.10         0.00
00004  NEOLUX ZOLD         350    150 125    28.60         0.00
00005  VERNODUX SARGA      300    300 798    91.10         0.00
00006  VERNODUX PIROS      500    200 777    81.10         0.00
00007  NEOLUX SARGA        400    200 145    35.90         0.00
00008  NEOLUX FEHER        800    700 100    34.00         0.00

 . 4
 . display
00004  NEOLUX ZOLD         350    150 125    28.60         0.00

 . store 6 to szam
     6
 . go szam

 . display
00006  VERNODUX PIROS      500    200 777    81.10         0.00

 . szam
*** UNKNOWN COMMAND                          *** ISMERETLEN PARANCS
szam
CORRECT AND RETRY (Y/N)? n                   JAVIT ÉS ISMÉTEL (I/N)

 . &szam
MAKRO IS NOT A CHARACTER STRING      A MAKRO NEM KARAKTERLANC
&SZAM
CORRECT AND RETRY (Y/N)? n                   JAVIT ÉS ISMÉTEL (I/N)

 . index on anyagnev to raktaram
00008 RECORDS INDEXED                        00008 REKORD INDEXELVE

 . list
00008  NEOLUX FEHER        800    700 100    34.00         0.00
00002  NEOLUX KEK          300    100 124    30.00         0.00
00001  NEOLUX PIROS        300    200 123    38.50         0.00
```

```
00007  NEOLUX SARGA       400    200 145    35.90        0.00
00004  NEOLUX ZOLD        350    150 125    28.60        0.00
00003  VERNODUX KEK       500    300 789    96.10        0.00
00006  VERNODUX PIROS     500    200 777    81.10        0.00
00005  VERNODUX SARGA     300    300 798    91.10        0.00

 . go top
 . display
00008  NEOLUX FEHER       800    700 100    34.00        0.00

 . go bottom
 . display
00005  VERNODUX SARGA     300    300 798    91.10        0.00

 . go 7
 . display
00007  NEOlUX SARGA       400    200 145    35.90        0.00
```

IF

Alakja:

```
IF <kifejezés>
   <parancssor(ok)>
[ELSE
   <parancssor(ok)>]
ENDIF
```

Hatása:

Feltételes elágazás a parancsok végrehajtásában.

Ismertetése:

Ha a <kifejezés> igaz logikai értékű, az IF utáni parancsokat az adatbázis-kezelő rendszer végrehajtja. Hamis érték esetén az ELSE ágban levő parancsok kerülnek sorra. Az ELSE ág elhagyható. Ilyenkor – hamis eredmény esetén – az IF és az ENDIF közé írt parancsok végrehajtása elmarad és a felhasználói program végrehajtása az ENDIF-et követő parancscsal folytatódik.

Az IF parancsok bármilyen mélységig egymásba ágyazhatók. Ügyeljünk a parancsok egymásba ágyazásának szabályaira (2.9. rész)!

Példák

```
IF VALASZ = 'igen'
   DO KIIRO.CMD
ELSE
   DO FOMENU.CMD
ENDIF
```

```
IF NYOMTAT = 'igen'
   SET PRINT ON
ENDIF
```

Alakja:

INDEX ON <kifejezés> TO <indexállomány neve>

Hatása:

A munkaállomány indexelése a <kifejezés>-ben megadott kulcs alapján.

Ismertetése:

A parancs végrehajtása során egy új állomány (indexállomány) jön létre, mely a munkaállomány rekordjaira vonatkozó mutatókat tartalmazza. A munkaállomány ezután látszólagosan — a <kifejezés>-ben meghatározott kulcs szerint — sorba rendezett formát mutat, de fizikailag változatlan marad.

A sorrend növekvő, de numerikus értéket adó kifejezés esetén csökkenő is lehet. (Pl. ,,100—kifejezés'' vagy ,,1/kifejezés''.)

A látszólagosan sorbaállított adatbázis rekordjai között a FIND parancs segítségével nagyon gyorsan kereshetünk a kifejezés eredményének teljes vagy részleges megadásával. (Lásd a FIND parancsot, 165. old.)

Nem minden alkalmazás kívánja meg a munkaállomány rendezett állapotát. Lehetőség van az indexelt vagy az eredeti formájú adatbázis felhasználására.

Egy adatbázis-állományhoz több indexállományt készíthetünk. Az indexállományok közül — egy futtatáson belül — a legkésőbb készített határozza meg a látszólagos sorrendet. E sorrend megszüntetése, illetve megváltoztatása a USE és a SET INDEX TO parancsok megfelelő használatával érhető el. Ezekkel a parancsokkal egyrészt kijelölhető az elsődleges indexállomány (amely a látszólagos rendezett állapotot megszabja), másrészt azok az indexállományok, amelyeket szeretnénk az adatbázissal együtt naprakészen tartani. Az adatbázisállományban végzett módosítások (APPEND, EDIT, REPLACE, READ vagy BROWSE parancs) hatása ilyenkor az indexállományokra is kiterjednek. (Nem érvényes ez a PACK parancsra, mely csak egy indexállomány megfelelő módosítását hajtja végre, a többi indexállomány módosítása érdekében az adott kulcs szerinti indexelést ismételten el kell végezni.)

MEGJEGYZÉS

Mivel az indexelés kulcsát egy <kifejezés> is adhatja, az indexelés bármilyen összetett, bonyolult kulcs alapján is elvégezhető.

Példák

```
. use raktar
. list stru
STRUCTURE FOR FILE : RAKTAR.DBF          AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS :  00008               REKORDOK SZAMA:
DATE LAST UPDATE :   85/04/15            UTOLSO HASZNALAT NAPJA:
PRIMARY USE DATABASE                     ELSODLEGES MUNKATER
FLD   NAME      TYPE  WIDTH  DEC        MEZO NEV TIPUS HOSSZ
                                                    TIZEDESEK
001   ANYAGNEV    C     015
002   ERKEZETT    N     004
```

INDEX

```
003  KIADVA      N     004
004  ANYAGKOD    C     003
005  EGYSEGAR    N     006    002
006  OSSZERTEK   N     009    002
**TOTAL**              00042                **ÖSSZESEN**
```

```
 . list
00001  NEOLUX PIROS          300    200 123    38.50          0.00
00002  NEOLUX KEK            300    100 124    30.00          0.00
00003  VERNODUX KEK          500    300 789    96.10          0.00
00004  NEOLUX ZOLD           350    150 125    28.60          0.00
00005  VERNODUX SARGA        300    300 798    91.10          0.00
00006  VERNODUX PIROS        500    200 777    81.10          0.00
00007  NEOLUX SARGA          400    200 145    35.90          0.00
00008  NEOLUX FEHER          800    700 100    34.00          0.00

 . index on kiadva to elvitt
00008 RECORDS INDEXED                          00008 REKORD INDEXELVE

 . list
00002  NEOLUX KEK            300    100 124    30.00          0.00
00004  NEOLUX ZOLD           350    150 125    28.60          0.00
00001  NEOLUX PIROS          300    200 123    38.50          0.00
00006  VERNODUX PIROS        500    200 777    81.10          0.00
00007  NEOLUX SARGA          400    200 145    35.90          0.00
00003  VERNODUX KEK          500    300 789    96.10          0.00
00005  VERNODUX SARGA        300    300 798    91.10          0.00
00008  NEOLUX FEHER          800    700 100    34.00          0.00

 . go top
 . disp
00002  NEOLUX KEK            300    100 124    30.00          0.00

 . go bottom
 . disp
00008  NEOLUX FEHER          800    700 100    34.00          0.00

 . go 7
 . disp
00007  NEOLUX SARGA          400    200 145    35.90          0.00
```

```
 . skip
REKORD: 00003

 . disp
00003  VERNODUX KEK       500    300 789    96.10         0.00

 . skip -2
REKORD: 00006

 . disp
00006  VERNODUX PIROS     500    200 777    81.10         0.00

 . use raktar
 . index on erkezett-kiadva to keszleten
00008 RECORDS INDEXED                      00008 REKORD INDEXELVE

 . list anyagnev, erkezett, kiadva, anyagkod, ;
egysegar, erkezett-kiadva
00005  VERNODUX SARGA     300    300 798    91.10  0
00001  NEOLUX PIROS       300    200 123    38.50  100
00008  NEOLUX FEHER       800    700 100    34.00  100
00002  NEOLUX KEK         300    100 124    30.00  200
00003  VERNODUX KEK       500    300 789    96.10  200
00004  NEOLUX ZOLD        350    150 125    28.60  200
00007  NEOLUX SARGA       400    200 145    35.90  200
00006  VERNODUX PIROS     500    200 777    81.10  300

 . go bottom
 . disp
00006  VERNODUX PIROS     500    200 777    81.10         0.00

 . go top
 . disp
00005  VERNODUX SARGA     300    300 789    91.10         0.00

 . use raktar
 . index on str(erkezett,4,0)+anyagnev to karakter
00008 RECORDS INDEXED                      00008 REKORD INDEXELVE
```

INDEX

```
 . list
00002  NEOLUX KEK          300    100 124    30.00         0.00
00001  NEOLUX PIROS        300    200 123    38.50         0.00
00005  VERNODUX SARGA      300    300 798    91.10         0.00
00004  NEOLUX ZOLD         350    150 125    28.60         0.00
00007  NEOLUX SARGA        400    200 145    35.90         0.00
00003  VERNODUX KEK        500    300 789    96.10         0.00
00006  VERNODUX PIROS      500    200 777    81.10         0.00
00008  NEOLUX FEHER        800    700 100    34.00         0.00

 . use raktar
 . index on $(anyagnev,(@(' ',anyagnev))+1,10) to szin
00008 RECORDS INDEXED                    00008 REKORD INDEXELVE

 . list
00008  NEOLUX FEHER        800    700 100    34.00         0.00
00002  NEOLUX KEK          300    100 124    30.00         0.00
00003  VERNODUX KEK        500    300 789    96.10         0.00
00001  NEOLUX PIROS        300    200 123    38.50         0.00
00006  VERNODUX PIROS      500    200 777    81.10         0.00
00005  VERNODUX SARGA      300    300 798    91.10         0.00
00007  NEOLUX SARGA        400    200 145    35.90         0.00
00004  NEOLUX ZOLD         350    150 125    28.60         0.00

 . find PIROS
 . disp
00001  NEOLUX PIROS        300    200 123    38.50         0.00

 . set index to keszleten
 . list anyagnev, erkezett, kiadva, erkezett-kiadva,;
 anyagkod, egysegar off
VERNODUX SARGA     300    300       0 798    91.10
NEOLUX PIROS       300    200     100 123    38.50
NEOLUX FEHER       800    700     100 100    34.00
NEOLUX KEK         300    100     200 124    30.00
VERNODUX KEK       500    300     200 789    96.10
NEOLUX ZOLD        350    150     200 125    28.60
NEOLUX SARGA       400    200     200 145    35.90
VERNODUX PIROS     500    200     300 777    81.10
```

Alakja:

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

Hatása:

A billentyűzetről válaszként beérkező értéket elhelyezi egy memóriaváltozóban.

Ismertetése:

A parancs alkalmas arra, hogy a parancsállományok végrehajtásakor a felhasználónak megadjuk a lehetőséget adatok bevitelére. Ha szükséges, a parancs automatikusan létrehozza a memóriaváltozót. Az "<üzenet>" megadása esetén az adatbevitel előtt megjelenik az üzenet a képernyőn.

A memóriaváltozó típusát a válaszként bevitt adat típusa egyértelműen meghatározza. Határolók közt megadott karakterlánc esetében karakter típusú, numerikus kifejezés esetén numerikus típusú lesz. T és Y logikai igaz, F és N logikai hamis memóriaváltozó értéket ad. A TYPE függvény szolgálhat az adat típusának megállapítására.

Az üzenet aposztrófok vagy idézőjelek közé helyezendő.

Az INPUT parancsot numerikus és logikai típusú adatok beolvasására ajánlatos használni. Karakterláncokhoz jobb az ACCEPT parancs alkalmazása.

Példák

```
 . input to elso
:25
  25

 . input to masodik
:100/4+elso
  50

 . input 'Kerek egy szamot' to harmadik
Kerek egy szamot:9876
  9876

 . input 'Kerek egy logikai valtozot' to logikai
Kerek egy logikai valtozot:T
 .T.

 . input 'Kerek egy karakterlancot' to szoveg
Kerek egy karakterlancot:Jo lesz igy?
SYNTAX ERROR, RE-ENTER                    SZINTAKTIKAI HIBA,ISMET
:'Ez igy talan jobb'
Ez igy talan jobb
```

INPUT

```
 . input 'Kerek meg egy logikai valtozot' to log
Kerek meg egy logikai valtozot:y
 .T.

 . list memo
SZAM            (N)     7
FESTEK          (C)   VERNODUX
LOGIKAI         (L)   .T.
ELSO            (N)     25
MASODIK         (N)     50
HARMADIK        (N)     9876
SZOVEG          (C)   Ez igy talan jobb
LOG             (L)   .T.
**TOTAL**       09 VARIABLES USED  00050 BYTES USED
                    **ÖSSZESEN  09 MEMÓRIAVÁLTOZÓ  00050 BYTE
```

Alakja:

INSERT [BEFORE] [BLANK]

Hatása:

Újabb rekord elhelyezése az adatbázis rekordjai közé.

Ismertetése:

Az INSERT parancs segítségével egyszerre csak egy rekord építhető be az adatbázisba.

Az új rekord az aktuális rekord mögé kerül, hacsak nem alkalmazzuk a BEFORE paramétert, melynek hatására az új rekord az aktuális rekord elé épül be.

A BLANK paraméter megadása egy üres rekord elhelyezését eredményezi. Ha a BLANK paraméter hiányzik, a parancs – az APPEND parancs működéséhez hasonlóan – kéri a rekord feltöltését adatokkal.

Lehetőleg ne építsünk be adatokat az indexelt adatbázisokba, mert az indexállomány nem mindig követi ezt a változást. E helyett USE paranccsal indexállomány(ok) nélkül állítsuk munkába az adatbázis-állományt, és a beépítések után – az indexállomány(ok) módosítása érdekében – az adott kulcs(ok) szerinti indexelés(eket) ismételten végezzük el.

Példák

```
. use raktar
. list
00001  NEOLUX PIROS        300   200 123   38.50      0.00
00002  NEOLUX KEK          300   100 124   30.00      0.00
00003  VERNODUX KEK        500   300 789   96.10      0.00
00004  NEOLUX ZOLD         350   150 125   28.60      0.00
00005  VERNODUX SARGA      300   300 798   91.10      0.00
00006  VERNODUX PIROS      500   200 777   81.10      0.00
00007  NEOLUX SARGA        400   200 145   35.90      0.00
00008  NEOLUX FEHER        800   700 100   34.00      0.00

. goto 4
. insert

RECORD#00005                                    REKORD#00005
ANYAGNEV   : NEOLUX FEKETE
ERKEZETT   : 100
KIADVA     : 0
ANYAGKOD   : 128
EGYSEGAR   : 65.00
OSSZERTEK  : <CR>
```

INSERT

```
. list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  NEOLUX ZOLD         350   150 125   28.60        0.00
00005  NEOLUX FEKETE       100     0 128   65.00        0.00
00006  VERNODUX SARGA      300   300 798   91.10        0.00
00007  VERNODUX PIROS      500   200 777   81.10        0.00
00008  NEOLUX SARGA        400   200 145   35.90        0.00
00009  NEOLUX FEHER        800   700 100   34.00        0.00

. goto 4
. insert before

RECORD#00004                                        REKORD#00004
ANYAGNEV   : VERNODUX FEHER
ERKEZETT   : 100
KIADVA     : 50
ANYAGKOD   : 867
EGYSEGAR   : 78.80
OSSZERTEK  : <CR>

. list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  VERNODUX FEHER      100    50 867   78.80        0.00
00005  NEOLUX ZOLD         350   150 125   28.60        0.00
00006  NEOLUX FEKETE       100     0 128   65.00        0.00
00007  VERNODUX SARGA      300   300 798   91.10        0.00
00008  VERNODUX PIROS      500   200 777   81.10        0.00
00009  NEOLUX SARGA        400   200 145   35.90        0.00
00010  NEOLUX FEHER        800   700 100   34.00        0.00

. goto record 4
. insert blank
. list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
```

INSERT

```
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  VERNODUX FEHER      100     50 867    78.80        0.00
00005                        0      0         0.00        0.00
00006  NEOLUX ZOLD         350    150 125    28.60        0.00
00007  NEOLUX FEKETE       100      0 128    65.00        0.00
00008  VERNODUX SARGA      300    300 798    91.10        0.00
00009  VERNODUX PIROS      500    200 777    81.10        0.00
00010  NEOLUX SARGA        400    200 145    35.90        0.00
00011  NEOLUX FEHER        800    700 100    34.00        0.00

 . goto record 5
 . disp
00005                        0      0         0.00        0.00

 . replace anyagnev with 'VERNODUX ZOLD', erkezett with 550,;
kiadva with 200, anyagkod with '989', egysegar with 60.60
00001 REPLACEMENT(S)                                 00001 HELYETTESITES

 . list
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  VERNODUX FEHER      100     50 867    78.80        0.00
00005  VERNODUX ZOLD       550    200 989    60.60        0.00
00006  NEOLUX ZOLD         350    150 125    28.60        0.00
00007  NEOLUX FEKETE       100      0 128    65.00        0.00
00008  VERNODUX SARGA      300    300 798    91.10        0.00
00009  VERNODUX PIROS      500    200 777    81.10        0.00
00010  NEOLUX SARGA        400    200 145    35.90        0.00
00011  NEOLUX FEHER        800    700 100    34.00        0.00
```

JOIN

Alakja:

JOIN TO <állomány> { FOR <kifejezés> / WHILE <kifejezés> } FIELDS <mezőlista>

Hatása:

Két adatbázisból — rekordjaikat feltételek alapján összekapcsolva — új adatbázist képez.

Ismertetése:

Az összekapcsolásban részt vevő adatbázis-állományokat az elsődleges, illetve a másodlagos munkatérbe kell helyezni. A JOIN parancsot az elsődleges munkaterület kijelölése (SELECT PRIMARY) után adhatjuk ki. Az adatbázis-kezelő rendszer kiválasztja az elsődleges munkatérben levő adatbázis-állomány első rekordját, majd kiértékeli a <kifejezés>-t a másodlagos munkatérben levő adatbázis-állomány minden rekordjára. A kifejezés igaz értéke esetén egy rekorddal bővül az új állomány.

A másodlagos munkatérben levő adatbázis-állomány végének elérésekor az elsődleges munkatérben a következő rekordra lép az adatbázis-kezelő, majd a másodlagos munkatérben levő adatbázis-állomány első rekordjára ugrás után az eljárás megismétlődik.

A folyamat FOR paraméter esetén az elsődleges munkatérben levő adatbázis-állomány végének elérésekor, WHILE paraméter esetén pedig az első hamis logikai értéket adó rekordnál fejeződik be.

A FIELDS paraméter elhagyása esetén az új adatbázis-állomány az elsődleges munkatérben levő állományból minden mezőt, a másodlagos munkatérben levő állományból pedig annyi mezőt tartalmaz, amennyi még belefér az adatbázis-állományba, a 32 mezőben megadott felső korlátig. (Lásd a méretkorlátozások listáját.)

A FIELDS paraméter megadásakor csak a <mezőlista>-ban felsorolt mezők kerülnek az új állományba.

Túl hosszadalmas lehet a parancs végrehajtása, ha nagyok az adatbázisok, vagy a <kifejezés>-ben foglalt feltételeknek sok rekordpárosítás tesz eleget.

Vegyük figyelembe a JOIN parancs <kifejezés>-beli feltételeinek megfogalmazásakor a rendelkezésre álló háttérkapacitást. Gondoljunk arra is, hogy az adatbázis-kezelő rendszer legfeljebb 65 535 rekordot enged meg egy adatbázis-állományon belül!

Példák

```
. select secondary
. use meret
. list structure
STRUCTURE FOR FILE:  MERET.DBF        AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00003            A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSO HASZNALAT NAPJA:
SECUNDARY USE DATABASE                MASODLAGOS MUNKATER
FLD     NAME        TYPE  WIDTH  DEC  MEZO NEV TIPUS HOSSZ
001     MERETE       N     003                    TIZEDESEK
002     KOD          N     001
** TOTAL **              00005        **OSSZES**
```

```
 . list
00001     10  1
00002     20  2
00003     30  3

 . select primary
 . use anyag
 . list stru
STRUCTURE FOR FILE:  ANYAG.DBF        AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00003            A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSÖDLEGES MUNKATÉR
FLD     NAME        TYPE  WIDTH  DEC MEZÖ NÉV TIPUS HOSSZ
001     ANYAG         C     010                    TIZEDESEK
002     KOD           N     001
** TOTAL **             00012        **ÖSSZES**

. list
00001  vas          1
00002  rez          2
00003  aluminium    3

 . join to osszevont for kod<5
 . use osszevont
 . list

00001  vas          3    10   1
00002  vas          3    20   2
00003  vas          3    30   3
00004  rez          2    10   1
00005  rez          2    20   2
00006  rez          2    30   3
00007  aluminium    1    10   1
00008  aluminium    1    20   2
00009  aluminium    1    30   3

 . use anyag
 . join to osszevont for kod=s.kod
 . use osszevont
 . list
00001  vas          3    30   3
00002  rez          2    20   2
00003  aluminium    1    10   1
```

JOIN

```
. use anyag
. join to osszevont for kod<5 field anyag, merete
. use osszevont
. list off
vas             10
vas             20
vas             30
rez             10
rez             20
rez             30
aluminium       10
aluminium       20
aluminium       30

. select secondary
. use vetelezok
. list

00001   KIS PAL           123     20 85/01/11
00002   NAGY ANTAL        100    100 85/01/11
00003   MOLNAR JOZSEF     777     20 85/01/11
00004   KIS PAL           124     40 85/01/12
00005   NAGY ANTAL        798     20 85/01/13
00006   MOLNAR JOZSEF     125     10 85/01/20
00007   KIS PAL           145     20 85/01/13
00008   NAGY ANTAL        100     20 85/01/15
00009   MOLNAR JOZSEF     125     10 85/01/12

. select primary
. use raktar
. list
00001   NEOLUX PIROS      300    200 123    38.50        0.00
00002   NEOLUX KEK        300    100 124    30.00        0.00
00003   VERNODUX KEK      500    300 789    96.10        0.00
00004   VERNODUX FEHER    100     50 867    78.80        0.00
00005   VERNODUX ZOLD     550    200 989    60.60        0.00
00006   NEOLUX ZOLD       350    150 125    28.60        0.00
00007   NEOLUX FEKETE     100      0 128    65.00        0.00
```

```
00008  VERNODUX SARGA     300    300 798    91.10        0.00
00009  VERNODUX PIROS     500    200 777    81.10        0.00
00010  NEOLUX SARGA       400    200 145    35.90        0.00
00011  NEOLUX FEHER       800    700 100    34.00        0.00

 . join to mitkert for anyagkod=s.anyagkod field nev, ;
anyagnev, mennyiseg, egysegar, anyagkod
 . use mitkert
 . list stru
STRUCTURE FOR FILE:  MITKERT.DBF      AZ ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS:   00010            A REKORDOK SZÁMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATÉR
FLD     NAME        TYPE  WIDTH  DEC  MEZŐ NÉV TIPUS HOSSZ
001     NEV            C     015                      TIZEDESEK
002     ANYAGNEV       C     015
003     MENNYISEG      N     004
004     EGYSEGAR       N     006    002
005     ANYAGKOD       C     003
** TOTAL **              00044        **ÖSSZES**

. list
00001  KIS PAL          NEOLUX PIROS       20   38.50 123
00002  KIS PAL          NEOLUX KEK         40   30.00 124
00003  MOLNAR JOZSEF    NEOLUX ZOLD        10   28.60 125
00004  MOLNAR JOZSEF    NEOLUX ZOLD        10   28.60 125
00005  NAGY ANTAL       VERNODUX SARGA     20   91.10 798
00006  MOLNAR JOZSEF    VERNODUX PIROS     20   81.10 777
00007  KIS PAL          NEOLUX SARGA       20   35.90 145
00008  NAGY ANTAL       NEOLUX FEHER      100   34.00 100
00009  NAGY ANTAL       NEOLUX FEHER       20   34.00 100
```

LIST

Alakja:

a) LIST [<érvényességi kör>] [{WHILE <kifejezés> / FOR <kifejezés>}] [<kifejezéslista>] [OFF]

b) LIST STRUCTURE

c) LIST MEMORY

d) LIST FILES [ON <lemezegység>] [LIKE <rendszer szerinti elnevezés>]

Az <érvényességi kör> alapértelmezése
(az *a)* formánál)
- paraméter nélkül vagy FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamisra értékelt rekordig minden rekord.

Hatása:

a) A munkaállomány rekordjainak megjelenítése.

b) A munkaállomány szerkezetének megjelenítése.

c) A memóriaváltozók kiírása.

d) A lemezen levő állományok felsorolása, vagyis a nyilvántartás kiírása.

Ismertetése:

a) A munkaállomány teljes vagy részleges megjelenítésére szolgál. Az <érvényességi kör> és a FOR vagy WHILE paraméter hiánya esetén az adatbázis-állomány összes rekordját megjeleníti. Ha az <érvényességi kör> hiányzik, de a FOR <kifejezés> paraméter megvan, az állomány összes rekordjának adatait tekintetbe veszi, bár csak a FOR paraméter <kifejezés>-ében adott feltételt teljesítő rekordokat jeleníti meg. Amikor az <érvényességi kör> ugyan hiányzik, de a WHILE <kifejezés> paraméter szerepel, az aktuális rekordtól a <kifejezés>-ben megadott feltételeknek még eleget nem tevő utolsó rekordig jelennek meg az adatok.

Szabad a lista kifejezéseiben adatmezőket, memóriaváltozókat, illetőleg bármilyen, egyébként is megengedett numerikus, karakter vagy logikai értékeket (literált, állandót), beépített függvényeket, kifejezéseket, műveleteket is használni.

Az OFF parancsrész megadásával a rekordok tartalma előtt megjelenő rekordszám kiírása megszűnik.

A LIST parancs azonos a DISPLAY paranccsal, kivéve, hogy az előbbi nem áll meg 15 rekordonként, s alapértelmezésben — a WHILE paraméter használatát kivéve — *minden* rekordot tekintetbe vesz. Az ESC karakter beütése mindkét parancs végrehajtásának azonnali megszakításával jár, és az adatbázis-kezelő rendszer alapállapotba tér vissza.

b) Csak az aktuális munkatérben elhelyezkedő adatbázis-állomány szerkezetéről közöl információt.

c) Az éppen érvényben levő memóriaváltozókat a nevük és a hozzátartozó értékek megadásával listázza.

d) Elsősorban az alap- vagy a megjelölt mágneslemezen ([ON <lemezegység>]) levő adatbázis-állományok név szerinti megjelenítésére használatos. Néhány statisztikai adatot is megad.
A LIKE <rendszer szerinti elnevezés> megadásával más állományok felsorolását is megkaphatjuk. A <rendszer szerinti elnevezés> leggyakoribb formája a „*.típus", ahol a típus a TXT, FRM, MEM vagy bármely más legfeljebb három karakter. Ezeket az állományokat úgy listázza, mint ahogy a CP/M DIR parancsa.
(Példákat lásd a DISPLAY parancsnál.)

LOCATE

Alakja:

LOCATE [<érvényességi kör>] FOR <kifejezés>
.
.
.
[CONTINUE]

Az <érvényességi kör> alapértelmezése:

— az összes rekord.

Hatása:

Az adatbázis-állományban keresi azt az első rekordot, amely a <kifejezés> által adott feltételeknek eleget tesz.

Ismertetése:

Sikeres keresésre a parancs a megtalált rekord számával válaszol. Ha sikertelen a keresés, az END OF FILE üzenetet kapjuk, és a rekordmutató az utolsó rekordon marad.

Ha az <érvényességi kör>-ben a NEXT-tel megadott számú rekordon belül nem található a feltételeknek eleget tevő rekord, az END OF LOCATE választ kapjuk, a rekordmutató pedig az utolsóként megvizsgált rekordra mutat.

A CONTINUE hatására folytatódik a keresés a feltételt kielégítő rekordok megtalálása érdekében.

A LOCATE és a CONTINUE között parancsokat is megadhatunk. Ezekben a parancsokban azonban a FOR paraméter és a <kifejezés> hossza 128 karakterre korlátozódik 254 helyett.

Gyorsabb a LOCATE parancs végrehajtása, ha az adatbázist indexállomány nélkül használjuk.

Példák

```
. use raktar
. list
00001  NEOLUX PIROS        300   200 123    38.50        0.00
00002  NEOLUX KEK          300   100 124    30.00        0.00
00003  VERNODUX KEK        500   300 789    96.10        0.00
00004  VERNODUX FEHER      100    50 867    78.80        0.00
00005  VERNODUX ZOLD       550   200 989    60.60        0.00
00006  NEOLUX ZOLD         350   150 125    28.60        0.00
00007  NEOLUX FEKETE       100     0 128    65.00        0.00
00008  VERNODUX SARGA      300   300 798    91.10        0.00
00009  VERNODUX PIROS      500   200 777    81.10        0.00
00010  NEOLUX SARGA        400   200 145    35.90        0.00
00011  NEOLUX FEHER        800   700 100    34.00        0.00
```

```
 . locate for 'SARGA' $ anyagnev
RECORD: 00008                                        REKORD: 00008

 . disp
00008  VERNODUX SARGA     300    300 789    91.10         0.00

 . continue
RECORD: 00010                                        REKORD: 00010

 . disp
00010  NEOLUX SARGA       400    200 145    35.90         0.00

 . continue
END OF FILE ENCOUNTERED                AZ ALLOMANY VEGERE JUTOTT

 . locate for erkezett=500
RECORD: 00003                                        REKORD: 00003

 . ? #
     3

 . ? eof
 .F.

 . continue
RECORD: 00009                                        REKORD: 00009

 . ? #
     9

 . ? eof
 .F.

 . continue
END OF FILE ENCOUNTERED                AZ ALLOMANY VEGERE JUTOTT

 . ? #
    11
```

LOCATE

```
. ? eof
.T.

. disp
00011  NEOLUX FEHER          800    700 100    34.00          0.00
```

Alakja:

LOOP

Szerepe:

A DO WHILE szerkezeten belüli parancsok átugrása, majd a feltétel ismételt értékelése.

Ismertetése:

A parancsszó a DO WHILE szerkezeten belül helyezkedik el. Hatására az ENDDO parancsszóig, azaz a ciklus végéig található parancsokat átugorva a vezérlés ismét a DO WHILE-hoz kerül. A felhasználói program futása a <kifejezés> kiértékelésével és a parancsoknak e kiértékeléstől függő végrehajtásával folytatódik. A LOOP működése hasonlítható az ENDDO-éhoz, ugyanúgy a DO WHILE parancssorra – természetesen egymásba ágyazott szerkezeteknél a megfelelő szintre – adja a vezérlést.

Akkor alkalmazzuk, amikor egy nagyméretű DO WHILE szerkezet végrehajtása időigényessé válik, vagy olyan parancsokat tartalmaz a szerkezet, amelyek bizonyos feltételeknél feleslegessé válnak.

Ha lehet, kerüljük el a LOOP alkalmazását. Oldjuk meg a feladatot más módon, például úgy, ahogy a következő példában bemutatjuk.

Példa

```
STORE 1 TO SZAMOL                    * Az ismétlésszámláló (a
                                       "szamol" nevü memóriaváltozó) 1-re állitása.
DO WHILE SZAMOL<11                   * A tevékenység 10-szeri elvégzésének ellenörzése a DO WHILE szerkezet elején, a "szamol" nevü memóriaváltozó összehasonlitó müveletben való szerepeltetésével.
  STORE SZAMOL+1 TO SZAMOL           * Az ismétlésszámláló növelése.
  IF ANYAGNEV=''                     * Ha nincs adat,
    SKIP                             * lép egy rekordot.
    LOOP                             * Vissza a DO WHILE-hoz.
  ENDIF
  DO ELJARAS                         * Parancsállomány aktivizálása.
ENDDO                                * Vissza a DO WHILE-hoz.
```

LOOP

Ugyanez LOOP nélkül:

```
STORE 1 TO SZAMOL
DO WHILE SZAMOL<11
  STORE SZAMOL+1 TO SZAMOL
  IF ANYAGNEV='                          * Ha nincs adat,
    SKIP                                   lép egy rekordot.
  ELSE                                   * Különben (ha van adat)
    DO ELJARAS                             inditja a parancsállományt.
  ENDIF
ENDDO
```

Alakja:

a) MODIFY STRUCTURE

b) MODIFY COMMAND [<parancsállomány neve>]

Hatása:

a) Az adatbázis szerkezetének módosítása.

b) Parancsállomány megírása, tartalmának módosítása.

Ismertetése:

a) Bármilyen módosítás megengedett az adatbázis szerkezetében. A mezők jellemzői (név, típus, hossz, tizedesjegyek száma) megváltoztathatók, mezők törölhetők, vagy új mezők vehetők fel. Csak a munkaállományok módosíthatók.

A képernyőn megjelenő struktúra közvetlenül megváltoztatható a szerkesztés kódjainak (lásd a 4.6. részt) alkalmazásával.

A parancs — mivel a szerkezet módosítása az adatok megsemmisítésével jár együtt — egy figyelmeztetés megjelenésére kapott (igenlő) válasz esetén adja csak meg a módosítás lehetőségét.

Az adatok megőrzésének módja: a struktúrát átmásoljuk egy új állományba, ezt módosítjuk, majd az adatokat ehhez az új struktúrájú állományhoz fűzzük. (Ezáltal a struktúra úgy változik, hogy közben az adatok változatlanul megmaradnak.) Ha szükséges, az új állománynak az eredeti állomány nevét is adhatjuk, de ekkor előbb törölni kell a régi állományt.

A következő példában két lehetőséget mutatunk be. Kis adatbázis-állományoknál a kevesebb parancsot igénylő, nagy adatbázis-állományoknál a kevesebb másolással járó módszert érdemes használni.

Kis adatbázisnál:

(Az eljárás két teljes másolást tartalmaz.)

```
. use raktar
. copy to atmeneti
. modify structure
. append from atmeneti
. delete file atmeneti
```

Nagy adatbázisnál:

(Az eljárás egy teljes másolást tartalmaz.)

```
. use raktar
. copy to atmeneti structure
. use atmeneti
. modify structure
. append from raktar
. delete file raktar
. use
. rename atmeneti to raktar
```

b) A MODIFY COMMAND — bár korlátozott mértékű szerkesztési lehetőséget ad — helyettesíti az idegen (nem az adatbázis-kezelő rendszerhez tartozó) szövegszerkesztő programokat.

Ha a <parancsállomány> megadását elmulasztjuk, a parancs külön kéri az állomány nevét. Hivatkozhatunk nem <.CMD> kiterjesztésű szöveges állományokra is. A megadott állo-

mányt, ha még nem szerepel a nyilvántartásban, bejegyzi a parancs. A szerkesztés befejezése után a régi állomány <.BAK> állománynév-kiterjesztést kap, az új információt tartalmazó állományt pedig vagy a paraméterben megadott kiterjesztéssel, vagy — annak hiányában — <.CMD>-vel látja el a rendszer.
A parancs szövegszerkesztésbeli alkalmazhatóságát a következők korlátozzák:

1. Az állomány hossza legfeljebb 4000 sor lehet.
2. 77 karakternél nem lehet több egy sorban.
3. A TAB karaktereket szóközkarakterekké alakítja.
4. Nem tud keresni és blokkokat mozgatni.

* * *

A MODIFY és SET SCREEN ON végrehajtási ideje alatt érvényes szerkesztési kódok

^X (CAN) a fénypontot lefelé viszi a képernyőn a következő sorra.
(Alkalmazható erre a célra a ^F (ACK) is.)

^E (ENQ) a fénypontot felfelé mozgatja a képernyőn az előző sorra.
(Alkalmazható erre a célra a ^A (SOH) is.)

^D (EOT) a fénypontot balra tolja a következő karakterre.

^S (DC3) egy karakterrel jobbra lépteti a fénypontot.

^V (SYN) kapcsoló a felülírási és az inzertálási mód között.

^G (BEL) törli azt a karaktert, amelyen a fénypont áll.

<Rubout> a fényponttól balra álló karaktert törli.

^T (DC4) törli azt a sort, ahol a fénypont van, és az alatta levő sorokat eggyel feljebb mozgatja.

^Y (EM) törli annak a sornak a tartalmát, ahol a fénypont van, és a sort üres helyekkel tölti fel.

^N (SO) a fényponttól számítva egy pozícióval lejjebb viszi a sorokat, és elhelyez egy új üres sort ott, ahol a fénypont áll.

^C (ETX) a képernyőn levő információktól függően 5—10 sort lefelé mozog a fénypont.

^R (DC2) a képernyőn levő információktól függően 5—10 sort felfelé megy a fénypont.

^Q (DC1) elhagyja a szerkesztési módot, és visszatér alapállapotba anélkül, hogy a változtatásokat rögzítené a mágneslemezen.

^W (ETB) minden változtatást rögzít a mágneslemezen, és visszatér alapállapotba. (Alkalmazható erre a célra a ^O (SI) is.)

MEGJEGYZÉS

MODIFY szerkesztési módban a 0-tól 9-ig terjedő számjegyek, a tizedespont, a „–" és a „+" jel mellett a szóközt is használhatjuk numerikus adatok bevitelénél.

A vezérlőkarakterek — az adatbázis-kezelő rendszer alapállapotában — e kódok egy részét más feladatok elvégzésére használják. (V. ö. a 3.1. résszel.)

NOTE

Alakja:

a) NOTE [tetszőleges karakterek]

b) * [tetszőleges karakterek]

Hatása:

Megjegyzések, magyarázatok elhelyezése a parancsállományokban.

Ismertetése:

Az így elhelyezett magyarázatok a felhasználói program áttekinthetőségét segítik, nem jelennek meg a felhasználói program futása alatt.

Példák:

```
NOTE a program adatfeltöltö része

* a program adatszolgáltató része
```

Alakja:

OTHERWISE

Szerepe:

A DO CASE szerkezet egyik ága.

Ismertetése:

Ha egyik CASE ág <kifejezés>-e sem bizonyul igaznak, akkor az OTHERWISE ágban megfogalmazott feladatokat hajtja végre az adatbázis-kezelő rendszer. Az OTHERWISE ág elhagyható.

(Lásd a DO parancsot!)

PACK

Alakja:

PACK

Hatása:

A DELETE parancsban törlésre kijelölt rekordok tényleges törlése.

Ismertetése:

A rekordok törlésével tartalmuk véglegesen elveszik.

Indexelt adatbázisokhoz tartozó indexállományok közül csak az elsőnek kijelölt indexállomány aktualizálódik a PACK parancs végrehajtása során. A többi indexállomány aktualizálása érdekében ismételt indexelést kell végezni az egyes kulcsok szerint. Időkímélő, ha nagyobb méretű indexelt adatbázisban a parancsot előbb az indexállomány kijelölése nélkül hajtjuk végre, majd az indexelést újra végezzük.

Nincs szükség mindig a PACK parancs használatára: sokszor elegendő a DELETE önmagában. Sőt jobb is, ha a rekordokat meg akarjuk tartani az adatbázisban. Mint a DELETE parancsnál leírtakból kitűnik, a törlésre kijelölt rekordokat például a COPY, az APPEND és a SORT parancs nem veszi figyelembe. A COUNT figyelembe veszi a törölt rekordokat, de ez a probléma megoldható a törölt rekordot jelző beépített függvény alkalmazásával (pl.: COUNT FOR .NOT. * .AND. ...).

Sokszor fontos tudnunk, hogy a vizsgált rekord törlésre jelölt-e vagy sem. Ebben is segít a törölt rekordot jelző beépített függvény.

A PACK parancs helyettesíthető a COPY paranccsal. Az adatbázist egy új állományba másoljuk, ennek során a törlésre jelölt rekordok nem kerülnek át. Ezután a régi állományt töröljük vagy dokumentumként megőrizzük, az új adatbázis-állományt pedig átnevezzük.

Példák

```
. list
00001  NEOLUX PIROS        300   200 123    38.50        0.00
00002  NEOLUX KEK          300   100 124    30.00        0.00
00003  VERNODUX KEK        500   300 789    96.10        0.00
00004  VERNODUX FEHER       100    50 867    78.80        0.00
00005  VERNODUX ZOLD        550   200 989    60.60        0.00
00006  NEOLUX ZOLD          350   150 125    28.60        0.00
00007  NEOLUX FEKETE        100     0 128    65.00        0.00
00008  VERNODUX SARGA       300   300 798    91.10        0.00
00009  VERNODUX PIROS       500   200 777    81.10        0.00
00010  NEOLUX SARGA         400   200 145    35.90        0.00
00011  NEOLUX FEHER         800   700 100    34.00        0.00

. delete record 5
00001 DELETION(S)                                  00001 TÖRLÉS
```

```
 . list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  VERNODUX FEHER      100    50 867   78.80        0.00
00005 *VERNODUX ZOLD       550   200 989   60.60        0.00
00006  NEOLUX ZOLD         350   150 125   28.60        0.00
00007  NEOLUX FEKETE       100     0 128   65.00        0.00
00008  VERNODUX SARGA      300   300 798   91.10        0.00
00009  VERNODUX PIROS      500   200 777   81.10        0.00
00010  NEOLUX SARGA        400   200 145   35.90        0.00
00011  NEOLUX FEHER        800   700 100   34.00        0.00

 . pack
PACK COMPLETE,  000010 RECORDS COPIED
                       CSOMAG VÉGE   , 00010 REKORD MÁSOLVA

 . list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  VERNODUX FEHER      100    50 867   78.80        0.00
00005  NEOLUX ZOLD         350   150 125   28.60        0.00
00006  NEOLUX FEKETE       100     0 128   65.00        0.00
00007  VERNODUX SARGA      300   300 798   91.10        0.00
00008  VERNODUX PIROS      500   200 777   81.10        0.00
00009  NEOLUX SARGA        400   200 145   35.90        0.00
00010  NEOLUX FEHER        800   700 100   34.00        0.00

 . go 4
 . delete next 3
00003 DELETION(S)                              00003 TÖRLÉS

 . list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004 *VERNODUX FEHER      100    50 867   78.80        0.00
```

PACK

```
00005 *NEOLUX ZOLD          350   150 125   28.60          0.00
00006 *NEOLUX FEKETE        100     0 128   65.00          0.00
00007  VERNODUX SARGA       300   300 798   91.10          0.00
00008  VERNODUX PIROS       500   200 777   81.10          0.00
00009  NEOLUX SARGA         400   200 145   35.90          0.00
00010  NEOLUX FEHER         800   700 100   34.00          0.00

 . go 5
 . recall
00001 RECALL(S)                                  00001 VISSZAALLITAS

 . list
00001  NEOLUX PIROS         300   200 123   38.50          0.00
00002  NEOLUX KEK           300   100 124   30.00          0.00
00003  VERNODUX KEK         500   300 789   96.10          0.00
00004 *VERNODUX FEHER       100    50 867   78.80          0.00
00005  NEOLUX ZOLD          350   150 125   28.60          0.00
00006 *NEOLUX FEKETE        100     0 128   65.00          0.00
00007  VERNODUX SARGA       300   300 798   91.10          0.00
00008  VERNODUX PIROS       500   200 777   81.10          0.00
00009  NEOLUX SARGA         400   200 145   35.90          0.00
00010  NEOLUX FEHER         800   700 100   34.00          0.00

 . pack
PACK COMPLETE,  000008 RECORDS COPIED
                       CSOMAG VEGE   , 00008 REKORD MASOLVA

 . list
00001  NEOLUX PIROS         300   200 123   38.50          0.00
00002  NEOLUX KEK           300   100 124   30.00          0.00
00003  VERNODUX KEK         500   300 789   96.10          0.00
00004  NEOLUX ZOLD          350   150 125   28.60          0.00
00005  VERNODUX SARGA       300   300 798   91.10          0.00
00006  VERNODUX PIROS       500   200 777   81.10          0.00
00007  NEOLUX SARGA         400   200 145   35.90          0.00
00008  NEOLUX FEHER         800   700 100   34.00          0.00
```

Alakja:

POKE <cím>, <byte-lista>

Hatása:

Adatok elhelyezése a számítógép memóriájában.

Ismertetése:

A parancs a megadott <cím>-től kezdődően elhelyezi a <byte-lista> elemeit a számítógép memóriájában.

A byte-lista – decimális értékként megadott – elemeit vesszővel kell elválasztani egymástól és a <cím>-től.

MEGJEGYZÉS

A PEEK(<cím>) függvény adja meg, hogy egy adott memóriacímen milyen értéket találunk.

Mind a POKE parancs, mind a PEEK függvény decimális címeket és értékeket használ.

Az A400H tartomány a CP/M BDOS-ig rendelkezésre áll, de a SORT parancs is használja ezt a területet.

Példák

```
. poke 30499, 8
. ? peek(30499)
8

. store 30500 to tarolo
30500

. poke tarolo+2, 0,1,10,11,12,13,32
. ? peek(tarolo+2)
0

. store peek(tarolo+8) to szokozkod
32

. store chr(peek(tarolo+8)) to szokoz
```

QUIT

Alakja:

QUIT [TO '<állománynév vagy rendszerparancs>' [,'<lista>']]

Hatása:

Az összes állomány lezárása után átadja a vezérlést az operációs rendszernek, illetve az operációs rendszerben használható programoknak és rendszerparancsoknak.

Ismertetése:

Az adatbázis-kezelő rendszer elhagyását a képernyőre küldött üzenet jelzi.

A TO <'parancs'-lista> paraméter megadásakor az operációs rendszer a listában felsorolt rendszerparancsokat és programokat a felsorolás sorrendjében hajtja végre. A lista elemei vesszővel (,) választandók el egymástól és külön-külön aposztrófok (') közé kell őket tenni. Ez a megoldás lehetővé teszi az adatbázis-kezelő rendszer elhagyása után más programok indítását, rendszerparancsok végrehajtását.

A lista hossza a szokásos 254 karakter lehet. Több CP/M operációs rendszerbeli parancs, illetve program is indítható. A lista utolsó eleme lehet egy olyan parancs is, amely az adatbázis-kezelő rendszert aktivizálja. Ilyenkor az adatbázis-kezelő rendszer kapja meg újra a vezérlést. Minden más esetben a listában felsorolt utasítások végrehajtása után a CP/M operációs rendszer kapja meg a vezérlést.

Példa

```
. quit to 'dir b:','PIP LPT:=QUIT.TXT','dbase eljaras'
```

illetve

```
. quit to 'dir b:','PIP LPT:=QUIT.TXT','propbase eljaras'
```

Példánkban a kilépés után jegyzéket kapunk a b mágneslemez-meghajtón levő állományokról, majd kinyomtatódik a quit.txt nevű állomány, végül visszatérünk az adatbázis-kezelő rendszerbe az „eljárás" nevű parancsállomány megnyitásával.

Alakja:

READ

Hatása:

a) Lehetővé teszi a @ jelű parancs(ok)ban GET paraméterrel szereplő változók beolvasását.

b) Aktivizálja a formátumállományt.

Ismertetése:

a) A parancs hatására a felhasználói program visszalép az előző READ vagy CLEAR GET utáni első @ jelű parancsban meghatározott GET <változó>-ra, és kezdeményezi a beolvasást. Ha előzőleg READ vagy CLEAR GET még nem szerepelt, akkor a READ hatására a parancsállományban levő első @ jelű parancsban meghatározott GET <változó>-nál kezdődik a beolvasás. A beolvasás után a következő GET <változó>-ra vezérlődik a felhasználói program. Ha így újból elértük a READ-et, a felhasználói program futása a READ utáni sorban folytatódik. A READ előtti első változó első karaktere és az utolsó változó utolsó előtti karaktere között szabadon mozoghatunk a teljes képernyős szerkesztés kódjaival (lásd a 4.6. részt).

A @. . . SAY . . . GET és a READ parancsban használt változók vagy mezőnevek, vagy karakter, numerikus, illetve logikai típusú memóriaváltozók lehetnek. E változóknak a parancsok kiadása előtt létezniük kell. A memóriaváltozókat szükség esetén létrehozhatjuk például a megfelelő számú üres hely betöltésével:

STORE ' ' TO Ures, STORE 0 TO Szam.

b) Ha a @ . . . SAY . . . GET parancsot formátumállományban helyeztük el, és a felhasználói programban kiadtuk a ^SET FORMAT TO <formátumállomány neve>^ parancsot, akkor a formátumállományt a kiadását követő READ-ek aktivizálják; a formátumállományban levő @. . . SAY . . . GET parancs végrehajtódik — megformázva a képernyőt, és lehetővé téve az összes GET <változó> szerkesztését.

A további részleteket illetően lásd a @ jelű parancsot a 255. oldalon.

Példák

```
STORE ' ' to milyen
STORE '          ' to datum
ERASE
@ 10,5 SAY 'Milyen adatbazis-kezeloje van ?'
@ 12,7 SAY 'D = dBASE II'
@ 13,7 SAY 'P = PROP-BASE'
@ 15,11 GET milyen PICTURE '!'
READ
@ 20,2 SAY 'Kerem a datumot'
IF milyen = 'D'
  @ 20,18 SAY 'HH/NN/EE formaban' GET datum PICTURE ;
  '99/99/99'
  READ
```

```
   STORE $(datum,7,2)+'/'+$(datum,1,2)+'/'+$(datum,4,2);
   TO datum
ELSE
   @ 20,18 SAY 'EE/HH/NN formaban' GET datum PICTURE ;
   '99/99/99'
   READ
ENDIF



STORE '       ' TO keresztnev
STORE '      ' TO nevnap
ERASE
SET FORMAT TO notesz
READ
DO eljaras2.CMD
ERASE
READ
```

RECALL

Alakja:

RECALL [<érvényességi kör>] [FOR <kifejezés>]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül egy rekord,
- FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamis rekordig minden rekord.

Hatása:

A törlésre jelölt rekordok kijelölésének feloldása.

Ismertetése:

A DELETE paranccsal törlésre jelölt rekordok kijelölését megszünteti, e rekordok ismét „teljes értékűvé" válnak az adatbázisban.

Példák

```
. use raktar.dbf
. list
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003  VERNODUX KEK        500    300 789    96.10        0.00
00004  NEOLUX ZOLD         350    150 125    28.60        0.00
00005  VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS      500    200 777    81.10        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00

. go 3
  delete next 3
00003 DELETION(S)                                  00003 TÖRLÉS

. list
00001  NEOLUX PIROS        300    200 123    38.50        0.00
00002  NEOLUX KEK          300    100 124    30.00        0.00
00003 *VERNODUX KEK        500    300 789    96.10        0.00
00004 *NEOLUX ZOLD         350    150 125    28.60        0.00
00005 *VERNODUX SARGA      300    300 798    91.10        0.00
00006  VERNODUX PIROS      500    200 777    81.10        0.00
00007  NEOLUX SARGA        400    200 145    35.90        0.00
00008  NEOLUX FEHER        800    700 100    34.00        0.00
```

RECALL

```
 . recall record 4
00001 RECALL(S)                                  00001 VISSZAALLITAS
 . list
00001  NEOLUX PIROS        300    200 123    38.50         0.00
00002  NEOLUX KEK          300    100 124    30.00         0.00
00003 *VERNODUX KEK        500.   300 789    96.10         0.00
00004  NEOLUX ZOLD         350    150 125    28.60         0.00
00005 *VERNODUX SARGA      300    300 798    91.10         0.00
00006  VERNODUX PIROS      500    200 777    81.10         0.00
00007  NEOLUX SARGA        400    200 145    35.90         0.00
00008  NEOLUX FEHER        800    700 100    34.00         0.00

 . recall all
00002 RECALL(S)                                  00002 VISSZAALLITAS

 . list
00001  NEOLUX PIROS        300    200 123    38.50         0.00
00002  NEOLUX KEK          300    100 124    30.00         0.00
00003  VERNODUX KEK        500    300 789    96.10         0.00
00004  NEOLUX ZOLD         350    150 125    28.60         0.00
00005  VERNODUX SARGA      300    300 798    91.10         0.00
00006  VERNODUX PIROS      500    200 777    81.10         0.00
00007  NEOLUX SARGA        400    200 145    35.90         0.00
00008  NEOLUX FEHER        800    700 100    34.00         0.00
```

RELEASE

Alakja:

a) RELEASE ALL

b) RELEASE <memóriaváltozó> [,<memóriaváltozó-lista>]

Hatása:

Memóriaváltozók megszüntetése.

Ismertetése:

A <memóriaváltozó-lista> által meghatározott (egymástól vesszővel elválasztott) változókat vagy az összes (ALL) memóriaváltozót megszünteti, és az általuk lefoglalt helyet új memóriaváltozók rendelkezésére bocsátja.

Példák

```
. list memory
SZAM            (N)      1000
LOGIKAI         (L)  .F.
BETU            (C)  A
KISABC          (C)  abcdefghijklmnoprstuvz
NAGYABC         (C)  ABCDEFGHIJKLMNOPRSTUVZ
**TOTAL**  05 VARIABLES USED  00054 BYTES USED
                **ÖSSZESEN**  05 MEMORIAVALTOZÓ  00054 BYTE

. release betu
. list memory
SZAM            (N)      1000
LOGIKAI         (L)  .F.
KISABC          (C)  abcdefghijklmnoprstuvz
NAGYABC         (C)  ABCDEFGHIJKLMNOPRSTUVZ
**TOTAL**  04 VARIABLES USED  00053 BYTES USED
                **ÖSSZESEN**  04 MEMORIAVALTOZÓ  00053 BYTE

. release szam, kisabc
. list memory
LOGIKAI         (L)  .F.
NAGYABC         (C)  ABCDEFGHIJKLMNOPRSTUVZ
**TOTAL**  02 VARIABLES USED  00025 BYTES USED
                **ÖSSZESEN**  02 MEMORIAVALTOZÓ  00025 BYTE

. release all
. list memory
**TOTAL**  00 VARIABLES USED  00000 BYTES USED
                **ÖSSZESEN**  00 MEMORIAVALTOZÓ  00000 BYTE
```

REMARK

Alakja:

REMARK [<karakterlánc>]

Hatása:

A <karakterlánc>-ban szereplő szöveg a felhasználói program futása közben megjelenik.

Ismertetése:

A parancsban megadott karakterek a felhasználói program futása során a REMARK parancssor elérésekor a kiválasztott kimeneti egységen megjelennek.

Példa

```
. remark****** remark proba ******
****** remark proba ******
```

Alakja:

RENAME <megváltoztatandó állománynév> TO <új állománynév>

Hatása:

Állomány nevének megváltoztatása.

Ismertetése:

A mágneslemez nyilvántartásában az állomány nevét megváltoztatja.

Az állomány nevét követő, annak típusát meghatározó karakterek (az állománynév-kiterjesztés) elhagyása esetén az adatbázis-kezelő rendszer az állományt adatbázisnak tekinti, s így a típust megadó karaktereket <.DBF>-re állítja. (Lásd az 1.7. részben az állományok típusáról írtakat!)

Megnyitott állományt nem szabad átnevezni!

Példák

```
. list files like
DATABASE FILES    # RCDS     LAST UPDATE
                       ADATBAZIS-ALL.  REKORDOK    UTOLSO DATUM
RAKTAR    DBF     00008      85/04/13
UJANYAG   DBF     00002      85/04/13
VETELEZO  DBF     00005      85/04/13
BOVITES   DBF     00000      85/04/13
NVS       DBF     00002      85/04/13

. rename nvs to nevsor
. list files like
DATABASE FILES    # RCDS     LAST UPDATE
                       ADATBAZIS-ALL.  REKORDOK    UTOLSO DATUM
RAKTAR    DBF     00008      85/04/13
UJANYAG   DBF     00002      85/04/13
VETELEZO  DBF     00005      85/04/13
BOVITES   DBF     00000      85/04/13
NEVSOR    DBF     00002      85/04/13

. rename rename.txt to atnevez.szf
. rename z80trans.mac to z80trans.pas
```

REPLACE

Alakja:

REPLACE [<érvényességi kör>] <mezőnév1> WITH <kifejezés1> [,<mezőnév2>⇒

⇒ WITH <kifejezés2>] . . .[,<mezőnév5> WITH <kifejezés5>] [{FOR <kifejezés> / WHILE <kifejezés>}]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül egy rekord,
- FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamis rekordig minden rekord.

Hatása:

Adatbázismezők tartalmának új értékre cserélése.

Ismertetése:

A parancs hasonló a STORE parancshoz azzal a különbséggel, hogy a STORE csak memóriaváltozók értékét képes megváltoztatni, a REPLACE pedig csak adatbázismezőkre van hatással.

Az <érvényességi kör> paraméter és a FOR, illetve WHILE paraméter nélkül a parancs csak az aktuális rekorddal dolgozik.

A „<mezőnév> WITH <kifejezés>" formulával határozzuk meg a mezőt, amelybe a <kifejezés> értékét bevisszük. Ezután vesszővel elválasztva ugyanígy megadhatunk újabb mezőt a hozzá tartozó értékkel. Egy REPLACE parancsban legfeljebb 5 mező tartalmát változtathatjuk meg.

Egy indexelt adatbázis-állomány aktivizálásakor a kiválasztott indexállomány(ok) az aktualitásoknak megfelelően módosulni fog(nak), ha a hozzá(juk) tartozó kulcs tartalmazza a REPLACE parancsban megváltoztatott mezőt. Azok az indexállományok, amelyeket nem aktivizáltunk (a USE vagy a SET INDEX parancs kiadásakor), a módosításokat nem tartalmazzák.

Amikor olyan mező tartalmán végzünk változtatást, amely az indexelési kulcsban szerepel, az indexelési sorrend azonnal megváltozik. Megváltozik a rekordok indexelés szerinti sorrendje és a „következő rekord" nem az lesz, amelyik eddig volt. Ebben az esetben tehát nem használhatjuk <érvényességi kör>-ként a NEXT n paramétert.

Példák

```
. use raktar.dbf
. list
00001  NEOLUX PIROS        300   200 123   38.50        0.00
00002  NEOLUX KEK          300   100 124   30.00        0.00
00003  VERNODUX KEK        500   300 789   96.10        0.00
00004  NEOLUX ZOLD         350   150 125   28.60        0.00
00005  VERNODUX SARGA      300   300 798   91.10        0.00
```

```
00006  VERNODUX PIROS     500    200 777    81.10         0.00
00007  NEOLUX SARGA       400    200 145    35.90         0.00
00008  NEOLUX FEHER       800    700 100    34.00         0.00

. replace all osszertek with erkezett*egysegar
00008 REPLACEMENT(S)                              00008 HELYETTESITES

 . list
00001  NEOLUX PIROS       300    200 123    38.50     11550.00
00002  NEOLUX KEK         300    100 124    30.00      9000.00
00003  VERNODUX KEK       500    300 789    96.10     48050.00
00004  NEOLUX ZOLD        350    150 125    28.60     10010.00
00005  VERNODUX SARGA     300    300 798    91.10     27330.00
00006  VERNODUX PIROS     500    200 777    81.10     40550.00
00007  NEOLUX SARGA       400    200 145    35.90     14360.00
00008  NEOLUX FEHER       800    700 100    34.00     27200.00

 . replace egysegar with egysegar*1.15, osszertek with;
osszertek*1.15 for trim($(anyagnev,(@(' ',anyagnev))+1,10));
='PIROS'
00002 REPLACEMENT(S)                              00002 HELYETTESITES

 . list
00001  NEOLUX PIROS       300    200 123    44.27     13282.50
00002  NEOLUX KEK         300    100 124    30.00      9000.00
00003  VERNODUX KEK       500    300 789    96.10     48050.00
00004  NEOLUX ZOLD        350    150 125    28.60     10010.00
00005  VERNODUX SARGA     300    300 798    91.10     27330.00
00006  VERNODUX PIROS     500    200 777    93.26     46632.50
00007  NEOLUX SARGA       400    200 145    35.90     14360.00
00008  NEOLUX FEHER       800    700 100    34.00     27200.00
```

REPORT

Alakja:

REPORT [FORM <jelentésformátum-állomány>] [<érvényességi kör>]⇒

⇒ [{FOR <kifejezés> / WHILE <kifejezés>}] [TO PRINT] [PLAIN]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül vagy FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamis rekordig minden rekord.

Hatása:

A munkaállomány kért adatait a kívánt formátumban jeleníti meg.

Ismertetése:

Minden kiadott REPORT parancs egy jelentést készít a USE-zal már munkába állított állományokról. A jelentésben fejléceket helyezhetünk el az oszlopok felett. Az oszlopok tartalmának meghatározásakor minden adatbázis-kezelő művelet, beépített függvény, változó és állandó, valamint ezek kombinációja szerepelhet, ugyanúgy, mint egy kifejezésben.

A FOR és a WHILE paraméterek mellett csak az <érvényességi kör>-nek és a <kifejezés>-ben megadott feltételeknek megfelelő rekordok szerepelnek a jelentésben. A TO PRINT paraméter hatására a jelentést a gép ki is nyomtatja.

PLAIN paraméterezéssel – az egyébként a lap tetején megjelenő – lapszámozás, dátumozás és a második fejléc elmarad, és így a jelentés beilleszthető egy szövegszerkesztővel készített anyagba.

Más parancsok is hatással vannak a jelentés formátumára. A jelentés – alapbeállításban – lapemeléssel kezdődik, ezt kikapcsolhatjuk SET EJECT OFF paranccsal.

A ^SET HEADING TO <karakterlánc>^ paranccsal újabb fejlécet határozhatunk meg. Ez a második fejléc azonban nem lehet hosszabb 60 karakternél. A külön fejléc nem tárolódik, ezért minden rendszerfuttatáskor újra meg kell adni. Ha egy rendszerfuttatáson belül már adtunk ki ^SET HEADING TO <karakterlánc>^ parancsot, és egy újabb jelentésnél nincs szükségünk külön fejlécre, akkor a <karakterlánc>-ot töltsük fel szóközökkel!

Ha a rendszerdátum (SET DATE TO) szóközökkel van feltöltve, a jelentés fejlécében nullák lesznek a dátumban.

A REPORT parancs változatos jelentéseket készít, van azonban az adatbázis-kezelő rendszerben speciálisabb formátumok előállítására is lehetőség. Összetettebb adatfeldolgozási műveletekhez a SET FORMAT TO PRINT és a @ jelű parancsok gazdagabb választékot nyújtanak.

Amikor először adjuk a REPORT parancsot, azaz egy még nem létező jelentésformátum-állományt nevezünk meg, ez az állomány automatikusan létrejön, és az adatbázis-kezelő rendszer által feltett kérdésekre adott válaszainkkal töltődik fel. A válaszok befejezése után azonnal készül egy jelentés. A későbbiekben – jelentéskészítés céljából – már hivatkozhatunk erre a jelentésformátum-állományra.

A REPORT parancs kiadása utáni párbeszéd a következő:

```
. use raktar.dbf index szin.ndx
. list
00008  NEOLUX FEHER        900   700 100    45.50    27200.00
00002  NEOLUX KEK          500   100 124    30.00     9000.00
00003  VERNODUX KEK        500   300 789    96.70    48050.00
00001  NEOLUX PIROS        300   200 123    44.27    13282.50
00006  VERNODUX PIROS      500   200 777   100.10    46632.50
00005  VERNODUX SARGA      450   300 798   112.00    27330.00
00007  NEOLUX SARGA        400   200 145    35.90    14360.00
00004  NEOLUX ZOLD         350   150 125    28.60    10010.00

. list stru
STRUCTURE FOR FILE:  RAKTAR.DBF       AZ ÁLLOMÁNY SZERKEZETE:
NUMBER OF RECORDS:   00008            REKORDOK SZÁMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATER
FLD      NAME      TYPE  WIDTH  DEC  MEZŐ NÉV TIPUS HOSSZ
001      ANYAGNEV    C     015                      TIZEDESEK
002      ERKEZETT    N     004
003      KIADVA      N     004
004      ANYAGKOD    C     003
005      EGYSEGAR    N     006   002
006      OSSZERTEK   N     009   002
**TOTAL**                 00042        **ÖSSZESEN**

. report form jelentes

ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH
BEÁLLITHATO: M=BAL MARGÓ, L=SOR/LAP, W=SZÉLESSÉG  m=10,w=55
```

Az első sorban beállíthatjuk a margó szélességét (m=10) (az alapbeállítás 8 szóköz), a laponkénti sorok számát (1) (alapbeállítás 57 sor) és a jelentés szélességét (w=55) (az alapbeállítás 80 karakter). Az utóbbi beállítása a fejléc középre helyezéséhez fontos.

```
PAGE HEADING? (Y/N) y                 LESZ LAP FEJRÉSZ? (I/N)
```

Ha itt n-nel válaszolunk, az adatbázis-kezelő rendszer a következő kérdést nem teszi fel.

```
ENTER PAGE HEADING: Raktari keszletek     MI A LAP FEJRÉSZE :
```

A fejléc maximum 254 karakter hosszúságú lehet, egyébként a kérdésekre adott válaszok „elcsúsznak" a helyükről, és az adatbázis-kezelő nem tudja értelmezni azokat.

A fejlécben (és később az egész jelentésben) a pontosvesszőt vezérlőkarakterként kétféle értelmezésben használhatjuk.

a) ha a pontosvesszőt azonnal (szóköz nélkül) <CR> követi, az eddig megszokott módon terjeszthetjük ki parancssorunkat a képernyő következő sorára.

b) ha a pontosvesszőt nem <CR> karakter követi, akkor a jelentésben a pontosvessző utáni információ új sorba kerül.

Ha a fejlécben megadott szöveg nem fér ki egy sorban, akkor a még elférő utolsó szóköznél választódik el. Ha egy szó netán hosszabb, mint a rendelkezésre álló hely, akkor ennek teljes kitöltése szerint választ el az adatbázis-kezelő rendszer.

```
DOUBLE SPACE REPORT? (Y/N) n          ÜRES SOROK BEIKTATÁSA (I/N)
```

Akkor hasznos, ha egy-egy rekorddal kapcsolatos jelentés több sorba kerül. Így elválaszthatjuk az információkat egymástól.

```
ARE TOTALS REQUIRED? (Y/N) y          KIVAN ÖSSZEGZÉST? (I/N)
```

A kérdésre adott igen (y) válaszra a jelentés végén megjelenik a későbbiekben meghatározandó numerikus mezők tartalmának összege. Nemleges (n) válasz esetén az adatbázis-kezelő rendszer a következő öt kérdést már nem is teszi fel.

```
SUBTOTALS IN REPORT? (Y/N) y          KÉR RÉSZÖSSZEGZÉST? (I/N)
```

A kérdést követő négy kérdésre adott válasszal meghatározandó szempontok szerint kérhetjük részösszegzések elvégzését is, ha itt igennel (y) válaszolunk. Ha nemmel (n) válaszolunk, akkor a következő négy kérdés elmarad.

```
ENTER SUBTOTALS FIELD:  $(anyagnev,@(' ',anyagnev)+1,5)
                                      RÉSZÖSSZEGZÉS MEZEJE :
```

Esetünkben a részösszegképzés szempontja az anyag nevében található szóköz utáni rész (vagyis a festék színe).

```
SUMMARY REPORT ONLY? (Y/N) n          ÖSSZEFOGLALÓ JELENTÉS? (I/N)
```

Ha erre a kérdésre igennel (y) válaszolunk, akkor nem jelenik meg a jelentésben az egyes rekordok tartalma, csak az előző kérdésre adott válasz alapján kialakuló részösszeg.

```
EJECT PAGE AFTER SUBTOTALS? (Y/N) n
                                      LAPEMELÉS RÉSZÖSSZEG UTÁN ? (I/N)
```

Erre a kérdésre akkor válaszoljunk igennel (y), ha azt szeretnénk, hogy minden részösszegzés után új lapot kezdjen a jelentés.

```
ENTER SUBTOTAL HEADING: Szin:         A RÉSZÖSSZEG FEJRÉSZE :
```

Itt határozzuk meg a részösszeg fejrészét, de vegyük azt is figyelembe, hogy az általunk beírt szöveghez az adatbáziskezelő rendszer hozzáteszi a részösszegképzés mezejéhez megadott kulcs aktuális értékét.

```
WIDTH,CONTENTS                        OSZLOP SZÉLESSÉG,TARTALOM
```

A REPORT parancs itt rákérdez az első oszlop szélességére és tartalmára. Ha a válasz egy mező neve (pl. 001 15,anyagnev), akkor sincs köze az itt megadott oszlopszélességnek a mező

adatbázisban szereplő szélességéhez. Ha az adat hosszabb, mint az oszlopszélesség, akkor több sorba kerül a „Mi a lap fejrésze:" kérdésnél leírt módon. Mivel az oszlop tartalma karakter típusú, használhatjuk a karakterlánc-műveletek valamelyikét.
(Pl.

```
001 25,anyagnev+' FESTÉK'
```

vagy

```
002 25,anyagnev+';'+'----------'+';'+anyagkod)
001      15,$(anyagnev,1,@(' ',anyagnev))
```

Mivel a részösszegképzés mezejének az anyagnévben található szóköz utáni részt (szín) adtunk meg, ide a szóköz előtti részt (a festék fajtája) íratjuk ki.

```
ENTER HEADING: ---------------;Anyag neve; ;---------------
                                          OSZLOP NEVE   :
```

Az oszlop fejléceként megadott karakterláncban hasznosítjuk a pontosvessző soremelő tulajdonságát. (Figyeljük meg a példában szereplő eredménylistán, hogy a kezdő vonalkák száma megegyezik az oszlopszélességgel, ezért az utánuk szereplő pontosvessző egy üres sort eredményez!)

Az oszlop fejléceiben a pontosvessző mellett további karaktereket használhatunk a formátum meghatározására. Ha az oszlop fejlécében a címet egy „<" karakter előzi meg, akkor a cím balra lesz igazítva az oszlopban. A „>" jel pedig jobbra igazítja a címet.

```
002      4,erkezett-kiadva
```

Itt két numerikus mező különbségét jelenítjük meg.

```
ENTER HEADING: ----Rak-ta-;ron;----          OSZLOP NEVE:
```

Mivel a kezdő négy vonalka, illetve a „rak" szótag az elválasztó-jellel kitölti az oszlopszélességet, és nem is akartunk feleslegesen sort emelni, nem használtuk a pontosvesszőt.

```
ARE TOTALS REQUIRED? (Y/N) y          KIVAN ÖSSZEGZÉST? (Y/N)
```

Itt határozhatjuk meg, hogy erre a numerikus oszlopra vonatkozóan kérünk-e összegzést. Az adatbáziskezelő-rendszer ezt a kérdést csak akkor teszi fel, ha az oszlop numerikus értékű, és a kérdéssorozat elején feltett „kíván összegzést?" kérdésre igennel válaszoltunk. Ha részösszegzést is kértünk, az itt adott választól függően az is létrejön.

```
003      8,egysegar
ENTER HEADING: --------;Darabar; ;--------      OSZLOP NEVE:
ARE TOTALS REQUIRED? (Y/N) n          KIVAN ÖSSZEGZÉST? (Y/N)
```

A darabár egységárat tartalmazó oszlopában értelmetlen lett volna összegzéseket kérni.

```
004     10,(erkezett-kiadva)*egysegar
ENTER HEADING: ----------;Raktari ertek;----------
OSZLOP NEVE:
ARE TOTALS REQUIRED? (Y/N) y          KIVAN ÖSSZEGZÉST? (Y/N)
005     <cr>
```

Ha több oszlopra nincs szükségünk, az oszlopszám jelentkezésekor a <CR> billentyűt kell leütni. Ezután kezdődik a jelentéskészítés.

Példák

Az első példában összefoglalva megismételjük az ismertetőben részletezett jelentésmeghatározás folyamatát, a képernyőn megjelenő üzeneteket — az egyszerűség kedvéért már csak magyarul — és az általunk begépelt válaszokat.

```
. use raktar index szin
. report form jelentes
BEÁLLITHATÓ: M=BAL MARGÓ, L=SOR/LAP, W=SZÉLESSÉG  m=10,w=55
LAP FEJRÉSZE? (Y/N) y
MI A LAP FEJRÉSZE : Raktari keszletek
ÜRES SOROK BEIKTATÁSA? (Y/N) n
KIVAN ÖSSZEGZÉST   ? (Y/N) y
KÉR RÉSZÖSSZEGZÉST ? (Y/N) y
RÉSZÖSSZEGZÉS MEZEJE : $(anyagnev,@(' ',anyagnev)+1,5)
ÖSSZEFOGLALÓ JELENTÉS? (Y/N) n
LAPEMELÉS RÉSZÖSSZEG UTÁN ? (Y/N) n
A RÉSZÖSSZEG FEJRÉSZE : Szin:
OSZLOP SZÉLESSÉG,TARTALOM
001     15,$(anyagnev,1,@(' ',anyagnev))
OSZLOP NEVE  : ---------------;Anyag neve; ;---------------
002     4,erkezett-kiadva
OSZLOP NEVE  : ----Rak-ta-;ron;----
KIVAN ÖSSZEGZÉST   ? (Y/N) y
003     8,egysegar
OSZLOP NEVE  : --------;Darabar; ;--------
KIVAN ÖSSZEGZÉST   ? (Y/N) n
004     10,(erkezett-kiadva)*egysegar
OSZLOP NEVE  : ----------;Raktari ertek;----------
KIVAN ÖSSZEGZÉST   ? (Y/N) y
005     <cr>
```

Az előbbi tevékenységsorozat hatására a jelentést a képernyőn a következő formában kapjuk meg:

```
LAPSZAM: 00001
06/04/85

                Raktari keszletek
---------------- ---- -------- ----------
                 Rak-
   Anyag neve     ta-  Darabar   Raktari
                  ron             ertek
---------------- ---- -------- ----------
* Szin: FEHER
NEOLUX            200    45.50    9100.00
**OSSZESITVE**
                  200             9100.00

* Szin: KEK
NEOLUX            400    30.00   12000.00
VERNODUX          200    96.70   19340.00
**OSSZESITVE**
                  600            31340.00

* Szin: PIROS
NEOLUX            100    44.27    4427.00
VERNODUX          300   100.10   30030.00
**OSSZESITVE**
                  400            34457.00

* Szin: SARGA
VERNODUX          150   112.00   16800.00
NEOLUX            200    35.90    7180.00
**OSSZESITVE**
                  350            23980.00

* Szin: ZOLD
NEOLUX            200    28.60    5720.00
**OSSZESITVE**
                  200             5720.00
** OSSZES**
                 1750           104597.00
```

REPORT

A következő példában egy egyszerűbb jelentést készítettünk a <Raktar.DBF> adatbázisról. A képernyőn megjelenő üzeneteket itt is csak magyarul szerepeltettük.

```
. use raktar
. report
REPORT FORM. FILE NEVE: lista
BEÁLLITHATO: , M=BAL MARGÓ   , L=SOR/LAP    , W=SZÉLESSÉG
LAP FEJRÉSZE? (Y/N) n
URES SOROK BEIKTATÁSA? (Y/N) n
KIVAN ÖSSZEGZÉST   ? (Y/N) n
OSZLOP SZÉLESSÉG,TARTALOM
001      15,anyagnev
OSZLOP NEVE   : Anyag neve
002      4,erkezett
OSZLOP NEVE. : Erkezett
003      4,kiadva
OSZLOP NEVE   : Kiadva
004      6,egysegar
OSZLOP NEVE   : Darabar
005      9,osszertek
OSZLOP NEVE   : Össz- ertek
006
```

A fenti tevékenységsorozat hatására a következő jelentés jelenik meg a képernyőn. (Figyeljük meg a fejlécekben a hibás elválasztásokat ott, ahol az oszlop szélessége kisebb, mint ahány karakterből a fejléc szövege áll. Az előbbiekben leírt elválasztási szabály alapján a „Darabar" oszlopot választotta el a legcsúnyábban az adatbázis-kezelő rendszer az „r" betű átvitelével. Az „Osszertek" oszlopnál már irányítottuk a helyes elválasztást a kötőjel és a szóköz beiktatásával.)

```
LAPSZAM: 00001
85/04/06
```

Anyag neve	Erkezett	Kiadva	Darabar	Össz-ertek
NEOLUX PIROS	300	200	44.27	13282.50
NEOLUX KEK	500	100	30.00	9000.00
VERNODUX KEK	500	300	96.70	48050.00
NEOLUX ZOLD	350	150	28.60	10010.00
VERNODUX SARGA	450	300	112.00	27330.00
VERNODUX PIROS	500	200	100.10	46632.50
NEOLUX SARGA	400	200	35.90	14360.00
NEOLUX FEHER	900	700	45.50	27200.00

A következő példában a <Vetelezok.DBF> adatbázisról készítettünk jelentést a nyomtatóra az adatok összegzésével. A jelentéskészítés során kapott üzenetek szövege itt is magyarul szerepel.

```
. use vetelezok.dbf
. list stru
STRUCTURE FOR FILE:  VETELEZO.DBF     AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00009            REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSO HASZNALAT NAPJA:
PRIMARY USE DATABASE                  ELSODLEGES MUNKATER
FLD       NAME      TYPE  WIDTH  DEC  MEZO NEV TIPUS HOSSZ
001       NEV          C     015                    TIZEDESEK
002       ANYAGKOD     C     003
003       MENNYISEG    N     004
004       DATUM        C     008
**TOTAL**                    00031      **OSSZESEN**

. index on nev to vetelezok.ndx
. list
00001   KIS PAL          123     20 85/01/11
00004   KIS PAL          124     40 85/01/12
00007   KIS PAL          145     20 85/01/13
00003   MOLNAR JOZSEF    777     20 85/01/11
00006   MOLNAR JOZSEF    125     10 85/01/20
00009   MOLNAR JOZSEF    125     10 85/01/12
00002   NAGY ANTAL       100    100 85/01/11
00005   NAGY ANTAL       789     20 85/01/13
00008   NAGY ANTAL       100     20 85/01/15

. report form vetelezok to print
BEALLITHATO:,M=BAL MARGO,L=SOR/LAP,W=SZELESSEG  m=10,w=37
LAP FEJRESZE? (Y/N) y
MI A LAP FEJRESZE : Kivetelezett festekek
URES SOROK BEIKTATASA? (Y/N) n
KIVAN OSSZEGZEST   ? (Y/N) y
KER RESZOSSZEGZEST ? (Y/N) y
RESZOSSZEGZES MEZEJE : nev
OSSZEFOGLALO REPORT? (Y/N) n
LAPEMELES RESZOSSZEG UTAN ? (Y/N) n
```

REPORT

```
A RÉSZÖSSZEG FEJRÉSZE : Kivetelezo :
OSZLOP SZÉLESSÉG,TARTALOM
001      8, mennyiseg
OSZLOP NEVE   : Darab
KIVÁN ÖSSZEGZÉST   ? (Y/N) y
002      6,anyagkod
OSZLOP NEVE   : Kod
003      12,datum
OSZLOP NEVE   : Kivetelezes napja
004
```

Az előbbi jelentésformátum meghatározása után az alábbi jelentést nyomtatja ki az adatbázis-kezelő rendszer.

```
LAPSZAM: 00001
85/04/06

    Kivetelezett festekek
   Darab     Kod  Kivetelezes
                      napja

* Kivetelezo :   KIS PAL
       20 123     85/01/11
       40 124     85/01/12
       20 145     85/01/13
**OSSZESITVE**
       80

* Kivetelezo :   MOLNAR JOZSEF
       20 777     85/01/11
       10 125     85/01/20
       10 125     85/01/12
**OSSZESITVE**
       40

* Kivetelezo :   NAGY ANTAL
      100 100     85/01/11
       20 789     85/01/13
       20 100     85/01/15
**OSSZESITVE**
      140

** OSSZES**
      260
```

RESET

Alakja:

RESET

Hatása:

Lemezcsere esetén aktivizálja az új lemezeket.

Ismertetése:

Egy lemez behelyezése után a CP/M operációs rendszer nem engedi meg, hogy a lemezre írjunk, amíg nincs egy „melegindítás". Ezt elvégzi a RESET parancs, és újra megnyitja mindazokat az állományokat, amelyek a csere előtt nyitva voltak. Ha egy korábban nyitott állomány nem található az új lemezen, akkor azt lezárja. Lemezcsere nélkül kiadott RESET parancsnak nincs semmi értelme.

RESTORE

Alakja:

RESTORE FROM <memóriaállomány neve>

Hatása:

Memóriaváltozókat őrző állomány beolvasása.

Ismertetése:

A ^SAVE TO <memóriaállomány neve>^ paranccsal egy állományban megőrzött memóriaváltozók a ^RESTORE FROM <memóriaállomány neve>^ parancs hatására ismét aktívvá válnak. A RESTORE parancsot megelőzően definiált memóriaváltozókat azonban a parancs hatására törli a gép.

Példa

(Az itt következő példákban a később sorrakerülő SAVE parancs példáiban megőrzött memóriaváltozókat aktivizáljuk. (V. ö. 226. old.)

```
. display memory
**TOTAL**  00 VARIABLES USED      00000 BYTES USED
                **ÖSSZESEN**  00 MEMÓRIAVÁLTOZÓ  00000 BYTE

. restore from emlek
. display memory
EGY                         (N)          1
BETUK                       (C)        SDFGHJKL
**TOTAL**  02 VARIABLES USED      00009 BYTES USED
                **ÖSSZESEN**  02 MEMÓRIAVÁLTOZÓ  00009 BYTE

. restore from memoriter
. disp memo
SZAM                        (N)         19
FELIRAT                     (C)        felirat
LOGIKAI                     (L)        .F.
**TOTAL**  03 VARIABLES USED      00011 BYTES USED
                **ÖSSZESEN**  03 MEMÓRIAVÁLTOZÓ  00011 BYTE
```

RETURN

Alakja:

RETURN

Hatása:

A parancsállomány lezárása, a vezérlés átadása az aktivizálás helyére.

Ismertetése:

A parancsállományokon belül vagy végén elhelyezett RETURN hatására a vezérlés visszakerül ahhoz a parancsállományhoz, amely a parancsot tartalmazó állományt hívta. (Lehet ez közvetlen mód is, ha a kezdeményezés onnan történt.) Egyúttal lezáródik a RETURN parancsot tartalmazó parancsállomány. (A parancsállományok végén nem feltétlenül szükséges a RETURN parancs elhelyezése, ugyanezt a hatást váltja ki a parancsállomány végének elérése is.)

SAVE

Alakja:

SAVE TO <memóriaállomány neve>

Hatása:

Memóriaváltozók értékének tárolása egy állományban.

Ismertetése:

A parancs kiadása előtt definiált memóriaváltozók és értékük — későbbi felhasználás céljából — a <memóriaállomány>-ba kerülnek. A memóriaváltozók munkába állítására a RESTORE parancs alkalmas.

Példa

```
. display memory
**TOTAL**  00 VARIABLES USED      00000 BYTES USED
                **ÖSSZESEN**  00 MEMORIAVALTOZÓ  00000 BYTE

. store 1 to egy
. store 'SDFGHJKL' to betuk
. displ memor
EGY                          (N)          1
BETUK                        (C)        SDFGHJKL
**TOTAL**  02 VARIABLES USED      00009 BYTES USED
                **ÖSSZESEN**  02 MEMORIAVALTOZÓ  00009 BYTE

. save to emlek
. release all
. disp memo
**TOTAL**  00 VARIARLES USED      00000 BYTES USED
                **ÖSSZESEN**  00 MEMORIAVALTOZÓ  00000 BYTE

. restore from emlek
. display memory
EGY                          (N)          1
BETUK                        (C)        SDFGHJKL
**TOTAL**  02 VARIABLES USED      00009 BYTES USED
                **ÖSSZESEN**  02 MEMORIAVALTOZÓ  00009 BYTE

. release all
. disp memo
**TOTAL**  00 VARIABLES USED      00000 BYTES USED
                **ÖSSZESEN**  00 MEMORIAVALTOZÓ  00000 BYTE
```

SAVE

```
. store 19 to szam
. store 'felirat' to felirat
. store F to logikai
. display memory
SZAM                            (N)          19
FELIRAT                         (C)        felirat
LOGIKAI                         (L)        .F.
**TOTAL**  03 VARIABLES USED      00011 BYTES USED
                **ÖSSZESEN**  03 MEMÓRIAVÁLTOZÓ  00011 BYTE

. save to memoriter.mem
```

SELECT

Alakja:

a) SELECT PRIMARY
b) SELECT SECONDARY

Hatása:

a) Az elsődleges munkatér kijelölése.
b) A másodlagos munkatér kijelölése.

Ismertetése:

Az adatbázis-kezelő rendszer lehetőséget ad két adatbázis egyidejű kezelésére; egyszerre két munkatérrel dolgozik. E munkaterek kijelölésére szolgál a SELECT parancs. A két munkaterületben levő adatbázisok között műveleteket végezhetünk, adatokat vihetünk át egyikből a másikba, összehasonlíthatjuk a bennük levő adatokat stb.

Az adatbázis-kezelő rendszer indításával automatikusan az elsődleges munkaterületet vesszük használatba, amely mindaddig aktív marad, míg a SELECT SECONDARY paranccsal a másodlagos munkateret ki nem jelöljük. A kijelölés után kell megadni, hogy melyik adatbázist kívánjuk használni e munkatérben.

Mindkét munkatérben változtathatjuk az adatbázisokat. (Megtehetjük azt is, hogy mindkét munkatérben ugyanazt az adatbázis-állományt helyezzük el.)

A munkaterületek közül mindig a legutóbb kijelölt az aktív.

Mindkét munkatér adatbázisának mezőváltozóira hivatkozhatunk, azaz a kifejezésekben szerepelhetnek változók mindkét adatbázisból. Ha a két adatbázisban azonos nevű mezők találhatók és ezekre kell hivatkoznunk, a mezőváltozó előtt a „P." (PRIMARY) vagy az „S." (SECONDARY) előtagot kell elhelyezni, hogy a hivatkozást egyértelművé tegyük.

Az adatbázisban mozgást előidéző parancsok (pl. GOTO, SKIP, REPORT, SORT, COPY, LIST, DISPLAY) csak a kiválasztott munkatérben levő adatbázisra vannak hatással.

A hatásukat több rekordra kifejtő parancsok (melyek <érvényességi kör> paraméterrel rendelkeznek), a SET LINKAGE ON parancs hatására az elsődleges és a másodlagos munkatérben egyaránt vezérlik a mozgást a rekordokon. (Lásd még a SET parancsot.)

A REPLACE parancs csak a kiválasztott munkatérben levő adatbázis változóira hat.

A DISPLAY STRUCTURE parancs szintén csak a kiválasztott munkatérben levő adatbázis struktúráját jeleníti meg.

Példák

```
. use raktar.dbf
. list
00001   NEOLUX PIROS        300    200 123    44.27    13282.50
00002   NEOLUX KEK          300    100 124    30.00     9000.00
00003   VERNODUX KEK        500    300 789    96.10    48050.00
00004   NEOLUX ZOLD         350    150 125    28.60    10010.00
00005   VERNODUX SARGA      300    300 798    91.10    27330.00
00006   VERNODUX PIROS      500    200 777    93.26    46632.50
00007   NEOLUX SARGA        400    200 145    35.90    14360.00
00008   NEOLUX FEHER        800    700 100    34.00    27200.00
```

```
. select secondary
. use vetelezok.dbf
. list
00001   KIS PAL           123     20 85/01/11
00002   NAGY ANTAL        100    100 85/01/11
00003   MOLNAR JOZSEF     777     20 85/01/11
00004   KIS PAL           124     40 85/01/12
00005   NAGY ANTAL        789     20 85/01/13
00006   MOLNAR JOZSEF     125     10 85/01/20
00007   KIS PAL           145     20 85/01/13
00008   NAGY ANTAL        100     20 85/01/15
00009   MOLNAR JOZSEF     125     10 85/01/12

. select primary
. list
00001   NEOLUX PIROS        300    200 123    44.27    13282.50
00002   NEOLUX KEK          300    100 124    30.00     9000.00
00003   VERNODUX KEK        500    300 789    96.10    48050.00
00004   NEOLUX ZOLD         350    150 125    28.60    10010.00
00005   VERNODUX SARGA      300    300 789    91.10    27330.00
00006   VERNODUX PIROS      500    200 777    93.26    46632.50
00007   NEOLUX SARGA        400    200 145    35.90    14360.00
00008   NEOLUX FEHER        800    700 100    34.00    27200.00

. select secondary
. list
00001   KIS PAL           123     20 85/01/11
00002   NAGY ANTAL        100    100 85/01/11
00003   MOLNAR JOZSEF     777     20 85/01/11
00004   KIS PAL           124     40 85/01/12
00005   NAGY ANTAL        789     20 85/01/13
00006   MOLNAR JOZSEF     125     10 85/01/20
00007   KIS PAL           145     20 85/01/13
00008   NAGY ANTAL        100     20 85/01/15
00009   MOLNAR JOZSEF     125     10 85/01/12

. go 5
. disp
00005   NAGY ANTAL        789     20 85/01/13
```

SELECT

```
. sele prim
. go 8
. disp
00008  NEOLUX FEHER          800    700 100    34.00    27200.00
```

Alakja:

a) SET <paraméter> {ON / OFF}

b) SET <paraméter> TO <karakterlánc>

Hatása:

Az adatbázis-kezelő rendszer működését meghatározó beállítások megváltoztatása.

Ismertetése:

a) A parancs a <paraméter>-ben megadott működést befolyásoló kapcsolók be-, illetve kikapcsolását végzi el.

A következőkben a kapcsolókat és hatásukat ismertetjük, csillaggal (*) jelölve az alapbeállítást.

Paraméter	Beállítás	Hatás
1. **ALTERNATE**	ON	A kimeneti információ mágneslemezre, egy – előzőleg SET ALTERNATE TO <állománynév> paranccsal – kijelölt és megnyitott állományba íródik.
	*OFF	A kimeneti információ nem íródik állományba.
2. **BELL**	*ON	Hangjelzéssel figyelmeztet az adat rendelkezésére álló hely (hossz) túllépésére, vagy illegális adatbeviteli kísérletre.
	OFF	A hangjelzés kikapcsolva.
3. **CARRY**	ON	Az APPEND parancs teljes képernyős üzemmódjában az előző rekord adataival feltölti a rekordot. (Ezután szerkeszthető.)
	*OFF	Nem hozza át az előző rekord tartalmát.
4. **COLON**	*ON	A @ jelű parancs GET részében szereplő változó a képernyőn kettőspontokkal határolt.
	OFF	Nincs határolás.
5. **CONFIRM**	ON	Teljes képernyős szerkesztéskor egy változóhoz tartozó területet csak kontrollkarakter (pl. <CR>) leütésére hagy el.
	*OFF	A képernyőn kijelölt területet megtelésekor elhagyja.

SET

Paraméter	Beállítás	Hatás
6. CONSOLE	*ON	A kimeneti információ a képernyőre kerül.
	OFF	A kimeneti információ nem jut a képernyőre.
7. DEBUG	ON	Az ECHO és a STEP paraméterű SET parancsok kimenő információja a nyomtatóra íródik. Fenti parancsok így nem zavarják a képernyő használatát.
	*OFF	A fenti parancsok kimenő információja a képernyőre kerül.
8. ECHO	ON	A végrehajtandó parancsokat kiírja a DEBUG-gal kiválasztott kimeneti egységre.
	*OFF	Nem írja ki a parancsokat.
9. EJECT	*ON	Minden új jelentést egy lapemeléssel kezd a REPORT parancs.
	OFF	Nincs automatikus lapemelés a jelentés kezdetén.
10. ESCAPE	*ON	Az <ESC> karakter (1B H) a parancsállomány végrehajtását azonnal megszakítja, és az adatbázis-kezelő rendszer alapállapotba tér vissza.
	OFF	Az <ESC> karakter nem szakítja meg a parancsállomány végrehajtását.
11. EXACT	ON	A kifejezésekben és a FIND parancsban szereplő karakterláncok teljes megegyezését megköveteli. (Kivételek a karakterláncot követő üres helyek.)
	*OFF	Az egyezés vizsgálata csak a második karakterlánc hosszának megfelelő számú karakterre terjed ki. (Például az "EGYEZIK"="EGY" kifejezés logikai értéke igaz).
12. INTENSITY	*ON	A teljes képernyős műveletek kettős intenzitású karaktereket használnak. (Néhány képernyőnél normál vagy inverz – fordított színezésű – megjelenítés.)
	OFF	Nincs kettős intenzitás.
13. LINKAGE	ON	Az olyan parancsok, amelyek hatásukat több rekordra fejtik ki (LIST, REPORT, SUM és mások, melyeknek <érvényességi kör> paraméterük van) az elsődleges és a másodlagos munkatérben egyidőben, párhuzamosan hajtódnak végre.
	*OFF	A fenti parancsok csak a kijelölt munkatérben lesznek végrehajtva.

Paraméter	Beállítás	Hatás
14. PRINT	ON	A kimeneti információ nyomtatón is megjelenik.
	*OFF	A kimeneti információ csak a képernyőn jelenik meg.
15. RAW	ON	A <mezőlista> paraméter nélküli DISPLAY és LIST parancsok a mezők közé nem tesznek üres helyeket.
	*OFF	A mezők közé üres helyek kerülnek.
16. SCREEN	*ON	Az APPEND, az INSERT, az EDIT és a CREATE paranccsal a teljes képernyőn dolgozhatunk.
	OFF	A képernyőterület használata korlátozott.
17. STEP	ON	A parancsállományok tesztelésére, hibakeresésre szolgál. A program futása minden parancs végrehajtása után megszakad, és lehetőség adódik — a következő parancs végrehajtására, — a parancsállomány elhagyására, — a parancs megadására a billentyűzetről.
	*OFF	Parancsállományok folyamatos végrehajtása.
18. TALK	*ON	A parancsok végrehajtásának eredménye megjelenik a képernyőn.
	OFF	A parancsok végrehajtásának eredménye nem jelenik meg a képernyőn.

b) A <paraméter> után szereplő értéket karakterláncban adjuk meg. *Karakterhatárolók használata tilos.*

1. SET ALTERNATE TO <állománynév>

A parancs kijelöli azt az állományt, melyben a (képernyőre, nyomtatóra) kiírt valamennyi információ a kiírással egyidejűleg rögzítődik. (Beleértendő az adatbázis-kezelő rendszernek adott és az általa küldött információ is.) Elvész viszont a @ jelű parancs hatására kiírt információ, mert ezt az adatbázis-kezelő rendszer így nem tudja kezelni.

A parancs hatására az állomány nyilvántartásba kerül és megnyitódik. Az állomány automatikusan a <.TXT> állománynév-kiterjesztést kapja. A korábban már létező <állomány> felülírás által elveszti tartalmát.

A kijelölt <állomány>-ba ténylegesen a

SET ALTERNATE ON

parancs hatására kezdi meg a beírást az adatbázis-kezelő rendszer, és a

SET ALTERNATE OFF

parancsra lezárja.

SET

Az <állomány> szövegszerkesztő programmal feldolgozható, nyomtatható; szövegállományként kezelhető.

Példa

```
SET ALTERNATE TO B:SZOVEGFL
SET ALTERNATE ON
        parancsok
SET ALTERNATE TO <egy állomány>
SET ALTERNATE ON
        parancsok
SET ALTERNATE OFF
```

2. SET CALL <cím>

A CALL parancs számára memóriacímet adhatunk meg vele. A CALL parancs kiadásakor a <cím>-ben megadott helyen kezdődő gépi kódú rutin kapja a vezérlést. A <cím>-et decimális értékként kell megadni.

3. SET DATE TO xx/xx/xx

Az adatbázis-kezelő dátumot őrző rendszerváltozójának értéke tetszés szerint beállítható. A dátum naptár szerinti érvényessége azonban nincs vizsgálva, ellentétben az adatbázis-kezelő rendszer bejelentkezésekor lejátszódó folyamattal.

Példa

```
. ? date()

. set date to 03/06/85
. ? date()
03/06/85
```

4. SET DEFAULT TO <lemezegység>

Kijelöli az alap lemezegységet, amelyen a lemezegység megadása nélkül kért állományokat keresi. Ez lehetővé teszi, hogy a parancsállományok bármely, más lemezegységen előforduló állományra hivatkozzanak. A & beépített függvény (makró) ezt a lehetőséget még kibővíti (v. ö. a 2.7. résszel).

A <lemezegység>-nél lehetséges, de nem szükséges a kettőspont használata. Tehát például a „C" és a „C:" egyaránt megfelelő a C lemezegységre hivatkozáskor.

MEGJEGYZÉS

A parancs csak az adatbázis-kezelő működéséhez jelöli ki az alap lemezegységet, nem változtatja meg az operációs rendszer szintjén érvényes kijelölést.

Példa

```
. set default to b:          Az adatbázis-kezelő rendszer a B lemezegységen
. use adatok                 keresi az adatbázis-állományt.
```

5. SET FORMAT TO SCREEN

E parancs hatására a @ jelű parancs a képernyőre írja ki az információkat. Ez egyben az adatbázis-kezelő rendszer *alapbeállítása.* Megváltoztatása a

SET FORMAT TO PRINT

paranccsal lehetséges, mire a @ jelű parancs a nyomtatóra írja ki az információkat. Az alapbeállítást, vagyis a @ jelű parancsok információinak képernyőre íratását ismét a SET FORMAT TO SCREEN parancs állítja vissza.

6. SET FORMAT TO [<formátumállomány neve>]

A parancsállomány futása során az így kijelölt formátumállományból olvassa be a @ jelű parancsokat az adatbázis-kezelő rendszer. A beolvasást a READ parancs indítja.

7. SET HEADING TO <karakterlánc>

A REPORT parancs végrehajtásakor felhasználjuk ezt a <karakterlánc>-ot, mint a jelentés második fejrészét. A <karakterlánc> legfeljebb 60 karakter hosszú lehet.

8. SET INDEX TO [<indexállomány1>]. . .[,<indexállomány7>]

A parancsban — egymástól vesszővel elválasztva — fel kell sorolni azokat a már létező indexállományokat, melyekkel együtt akarjuk kezelni a munkaállományokat. Legfeljebb 7 indexállomány jelölhető ki.

Az adatbázis-kezelő rendszer csak a parancsban felsorolt indexállományokat állítja munkába, a korábban nyitott indexállományokat lezárja. A rendszer a felsorolásban első helyen álló indexállományt tekinti az elsődleges indexállománynak, mely meghatározza az adatbázis (látszólagos) rendezett állapotát. Új elsődleges indexállomány kijelöléséhez a parancs megismétlése szükséges, a kívánt indexállományt első helyre állítva a listában.

A SET INDEX TO parancs — indexállomány megnevezése nélkül — megszünteti az adatbázis és az indexállományok összerendelését, és az adatbázis ismét szekvenciális állomány lesz.

MEGJEGYZÉS

Új indexállomány(ok) kijelölése után az indexmutató (pointer) beállításához a FIND vagy a GOTO parancsnak kell megelőznie az olyan parancsokat, amelyek NEXT <érvényességi kör> paramétert tartalmaznak.

9. SET MARGIN TO <szám>

Kijelölhetjük, hogy a jelentés (report) készítésekor a nyomtatón melyik oszlop koordinátája legyen a nyomtatási kép bal oldali széle. A bal margót meghatározó <szám> értéke csak egy 1 és 254 közötti állandó (konstans) lehet.

SKIP

Alakja:

SKIP [±] [<kifejezés>]

Hatása:

A rekordmutató mozgatása.

Ismertetése:

Az aktuális rekord számát tartalmazó rekordmutató értéke növelhető, illetve csökkenthető. A SKIP parancs a <kifejezés>-ben megadott számmal – kifejezés hiányában 1-gyel – növeli vagy csökkenti a rekordmutató értékét.

Indexállománnyal együtt kezelt adatbázisban a rekordmutató mozgása az indexelés sorrendjének megfelelő.

Ha a SKIP parancsban az adatbázis végénél távolabbi célt jelölünk meg, az adatbázis-kezelő rendszer az állomány *utolsó rekordjára* vezérli a rekordmutatót. A rekordmutató a további hasonló irányú SKIP parancsok hatására már nem mozdul el, és az *utolsó rekordon marad.* Amikor ez problémát okoz, az EOF függvényt hívhatjuk segítségül.

```
use nevsor.dbf
locate for szul:ido='1962.08.27'
skip
if eof
  append blank
else
  skip -1
  insert blank
endif
```

Ha a SKIP parancsban az állomány kezdeténél előbbi célt jelöltünk meg, a rendszer az első rekordra áll.

Példa

```
. use raktar
. list
00001  NEOLUX PIROS       300   200 123   44.27    13282.50
00002  NEOLUX KEK         300   100 124   30.00     9000.00
00003  VERNODUX KEK       500   300 789   96.10    48050.00
00004  NEOLUX ZOLD        350   150 125   28.60    10010.00
00005  VERNODUX SARGA     300   300 798   91.10    27330.00
00006  VERNODUX PIROS     500   200 777   93.26    46632.50
00007  NEOLUX SARGA       400   200 145   35.90    14360.00
00008  NEOLUX FEHER       800   700 100   34.00    27200.00
```

```
 . goto 6
 . disp
00006  VERNODUX PIROS     500    200 777    93.26    46632.50

 . skip
RECORD: 00007                                    REKORD: 00007

 . skip -4
RECORD: 00003                                    REKORD: 00003

 . skip -1
RECORD: 00002                                    REKORD: 00002

 . skip 2*2
RECORD: 00006                                    REKORD: 00006

 . store 3 to szam
  3

 . skip -szam
RECORD: 00003                                    REKORD: 00003

 . skip
RECORD: 00004                                    REKORD: 00004

 . skip szam
RECORD: 00007                                    REKORD: 00007

 . ? #
     7

 . ? eof
 .F.

 . skip
RECORD: 00008                                    REKORD: 00008

 . ? #
     8
```

SKIP

```
 . ? eof
 .F.

 . skip
RECORD: 00008                                        REKORD: 00008

 . ? #
      8

 . ? eof
 .T.
 . disp
00008  NEOLUX FEHER       800    700 100    34.00    27200.00
```

Alakja:

SORT ON <mezőnév> TO <állomány> [{ASCENDING / DESCENDING}]

Hatása:

Az adatbázis új állományt készít, amelyben a rekordok elhelyezkedési sorrendjét a parancs által kijelölt mező tartalma (kifejezést nem fogad el) határozza meg.

Ismertetése:

A sorrend meghatározásához a parancsban bármelyik mező kijelölhető (de mindig csak egy). A parancs végrehajtása után a munkaállomány változatlan marad.

A sorrendet természetesen meghatározhatjuk több mező tartalmának figyelembevételével is. Ilyenkor újabb és újabb parancsok megadásával kell a sorbarendezett állományok más-más mező szerinti rendezését elvégeznünk egymás után az összes kiválasztott mezőre. A rendezési folyamatot a legkevésbé fontos mező megnevezésével kell kezdeni, s a sorrend kialakításában legfontosabbnak ítélt mezővel kell zárni. Az adatbázis-kezelő rendszer ugyanis csak akkor változtatja meg a rekordok sorrendjét, ha az a sorbarendezéshez feltétlenül szükséges.

A sorrend lehet növekvő (ascendig) vagy csökkenő (descending). Az adatbázis-kezelő rendszer alapértelmezése a növekvő sorrend.

Példa

```
. use raktar.dbf
. list
00001   NEOLUX PIROS        300    200 123    44.27    13282.50
00002   NEOLUX KEK          300    100 124    30.00     9000.00
00003   VERNODUX KEK        500    300 789    96.10    48050.00
00004   NEOLUX ZOLD         350    150 125    28.60    10010.00
00005   VERNODUX SARGA      300    300 798    91.10    27330.00
00006   VERNODUX PIROS      500    200 777    93.26    46632.50
00007   NEOLUX SARGA        400    200 145    35.90    14360.00
00008   NEOLUX FEHER        800    700 100    34.00    27200.00

. sort on osszertek to rafordit
SORT COMPLETE                                          SORT VEGE

. use rafordit.dbf
. list
00001   NEOLUX KEK          300    100 124    30.00     9000.00
00002   NEOLUX ZOLD         350    150 125    28.60    10010.00
00003   NEOLUX PIROS        300    200 123    44.27    13282.50
```

SORT

00004	NEOLUX SARGA	400	200	145	35.90	**14360.00**
00005	NEOLUX FEHER	800	700	100	34.00	**27200.00**
00006	VERNODUX SARGA	300	300	798	91.10	**27330.00**
00007	VERNODUX PIROS	500	200	777	93.26	**46632.50**
00008	VERNODUX KEK	500	300	789	96.10	**48050.00**

Alakja:

STORE <kifejezés> TO <memóriaváltozó>

Hatása:

A memóriaváltozó a kifejezés értékével töltődik fel.

Ismertetése:

A parancs előbb kiértékeli a kifejezést, majd az eredményt elhelyezi a memóriaváltozóban.

A parancs új memóriaváltozó létrehozására és értékadására, valamint korábban használt memóriaváltozó új értékkel történő feltöltésére (ez is értékadás) alkalmas.

A STORE csak memóriaváltozók értékadására alkalmazható, adatbázisok mezőváltozóinak új értéket a REPLACE paranccsal adhatunk.

Példa

```
 . release all
 . list memo
** TOTAL**  00 VARIABLES USED  00000 BYTES USED
                  **ÖSSZESEN  00 MEMÓRIAVÁLTOZÓ  00000 BYTE

 . store 1000 to szam
  1000

 . store F to logikai
 .F.

 . store 'A' to betu
A

 . store 'abcdefghijklmnoprstuvz' to ABC
abcdefghijklmnoprstuvz

 . store szam-szam/100 to resz
  990

 . store betu+ABC to karakt
Aabcdefghijklmnoprstuvz

 . list memo
SZAM          (N)     1000
LOGIKAI       (L)   .F.
```

STORE

```
BETU          (C)  A
ABC           (C)  abcdefghijklmnoprstuvz
RESZ          (N)    990
KARAKT        (C)  Aabcdefghijklmnoprstuvz
** TOTAL**  06 VARIABLES USED  00058 BYTES USED
                  **ÖSSZESEN  06 MEMÓRIAVÁLTOZÓ  00058 BYTE
```

SUM

Alakja:

SUM { <kifejezés1> / <mezőnév1> } [, { <kifejezés2> / <mezőnév2> }] ... [, { <kifejezés5> / <mezőnév5> }] ⇒

⇒[TO <memóriaváltozó [,lista]>] [<érvényességi kör>] [{ FOR <kifejezés> / WHILE <kifejezés> }]

Az <érvényességi kör> alapértelmezése:

- paraméter nélkül vagy FOR paraméterrel az összes rekord,
- WHILE paraméterrel az aktuális rekordtól az első hamis értékű rekordig minden rekord.

Hatása:

Kifejezések, illetve mezők tartalmának összeadása.

Ismertetése:

A parancs maximum 5 megnevezett (és egymástól vesszővel elválasztott) numerikus típusú kifejezés (mező) tartalmát adja össze kifejezésenként (mezőnként) az <érvényességi kör> és – ha van ilyen – a FOR, illetve a WHILE paraméter <kifejezés>-ében megfogalmazott feltétel szerint.

Az eredményeket meg is őrizhetjük memóriaváltozó(k)ban, ha a ^TO <memóriaváltozó-lista>^ parancsrészt is használjuk. A parancs az elsőként megadott kifejezés (mező) összegét az első memóriaváltozóba, a másodikként felsoroltét a második memóriaváltozóba helyezi (és így tovább, ötig). Nem kell minden kifejezéshez (mezőhöz) memóriaváltozót megadni, de vegyük figyelembe a parancsban megadott sorrendet.

A memóriaváltozók a SUM parancs hatására létrejönnek, ha addig nem léteztek.

Példák

```
. use raktar
. list
00001  NEOLUX PIROS       300   200 123   44.27   13282.50
00002  NEOLUX KEK         300   100 124   30.00    9000.00
00003  VERNODUX KEK       500   300 789   96.10   48050.00
00004  NEOLUX ZOLD        350   150 125   28.60   10010.00
00005  VERNODUX SARGA     300   300 789   91.10   27330.00
00006  VERNODUX PIROS     500   200 777   93.26   46632.50
00007  NEOLUX SARGA       400   200 145   35.90   14360.00
00008  NEOLUX FEHER       800   700 100   34.00   27200.00

. sum erkezett
 3450

. sum erkezett for anyagnev='NEOLUX'
 2150
```

SUM

```
 . list memo
**TOTAL**    00 VARIABLES USED  00000 BYTES USED
                  **ÖSSZESEN**  00 MEMÓRIAVÁLTOZÓ 00000 BYTE

 . sum erkezett for anyagnev='VERNODUX' to verno
  1300

 . sum erkezett, kiadva, erkezett-kiadva, ;
(erkezett-kiadva)*egysegar, osszertek ;
to Merkezett, MKiadva, Keszlet, Kiadas, Mosszertek
  3450   2150   1750 104597.00 195865.00

 . list memo
VERNO          (N)     1300
MERKEZETT      (N)     3450
MKIADVA        (N)     2150
KESZLET        (N)     1750
KIADAS         (N)   104597.00
OSSZERT        (N)   195865.00
**TOTAL**    06 VARIABLES USED  00051 BYTES USED
**ÖSSZESEN**  06 MEMÓRIAVÁLTOZÓ 00051 BYTE
```

TOTAL

Alakja:

TOTAL ON <kulcsmező> TO <adatbázis-állomány> [FIELDS <mezőlista>]⇒

⇒ [{ FOR <kifejezés> / WHILE <kifejezés> }]

Hatása:

Az adatok összegzése új adatbázis-állományba.

Ismertetése:

A TOTAL parancs hasonlít a REPORT részösszegképző tulajdonságára. A rekordokon végighaladva összeadja az azonos nevű numerikus mezők tartalmát mindaddig, míg a <kulcsmező> tartalma egy rekordnál meg nem változik. Az így kapott eredményt egy új adatbázisba írja, azonos tartalmú kulcsmezőnként egy rekordba töltve az összegzett adatokat.

A forrásállománynak a TOTAL parancsban használt mezőre indexeltnek vagy sorbarendezettnek kell lennie. A TOTAL mindig új rekordot nyit a célállományban, ha a forrásállományban a <kulcsmező> tartalma megváltozik.

Ha a célállomány már létezik — és csak olyan nevű (de nem feltétlenül azonos hosszúságú) mezőket tartalmaz, mint a forrásállomány (lehet a mezők száma kevesebb) —, akkor adatainak felülírásával létrejön az összegezett adatbázis, hiszen ilyenkor a TOTAL parancs a struktúrát arra használja, hogy eldöntse, mely mezők vehetnek részt (numerikus típusuk alapján) az összegzésben. A nem numerikus mezők esetében csak a <kulcsmező> szerinti sorrendben első rekord tartalma kerül át.

Ha új adatbázis-állományba végezzük az összegzést, akkor a parancs vagy egy azonos, vagy a ^FIELDS <mezőlista>^-val szűkített struktúrájú adatbázis-állományt hozza létre.

Példa

```
. use vetelezok.dbf
. list
00001   KIS PAL          123    20 85/01/11
00002   NAGY ANTAL       100   100 85/01/11
00003   MOLNAR JOZSEF    777    20 85/01/11
00004   KIS PAL          124    40 85/01/12
00005   NAGY ANTAL       789    20 85/01/13
00006   MOLNAR JOZSEF    125    10 85/01/20
00007   KIS PAL          145    20 85/01/13
00008   NAGY ANTAL       100    20 85/01/15
00009   MOLNAR JOZSEF    125    10 85/01/12

. index on nev to nevsor
00009 RECORDS INDEXED                    00009 REKORD INDEXELVE

. total on nev to mennyit
00003 RECORDS COPIED                     00003 REKORD MASOLVA
```

TOTAL

```
 . use mennyit.dbf
 . list
00001   KIS PAL          123     80 85/01/11
00002   MOLNAR JOZSEF    777     40 85/01/11
00003   NAGY ANTAL       100    140 85/01/11

 . copy to mennyit stru
 . use mennyit
 . modi stru

 . list stru
STRUCTURE FOR FILE:  MENNYIT.DBF      AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00000  .         A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNÁLAT NAPJA:
PRIMARY USE DATABASE                  ELSŐDLEGES MUNKATÉR
FLD     NAME       TYPE  WIDTH  DEC  MEZŐ NÉV TIPUS HOSSZ
001     NEV           C     015                    TIZEDESEK
002     MENNYISEG     N     008
**TOTAL**                   00020     **ÖSSZESEN**

 . use vetelezok.dbf index nevsor.ndx
 . list
00001   KIS PAL          123     20 85/01/11
00004   KIS PAL          124     40 85/01/12
00007   KIS PAL          145     20 85/01/13
00003   MOLNAR JOZSEF    777     20 85/01/11
00006   MOLNAR JOZSEF    125     10 85/01/20
00009   MOLNAR JOZSEF    125     10 85/01/12
00002   NAGY ANTAL       100    100 85/01/11
00005   NAGY ANTAL       789     20 85/01/13
00008   NAGY ANTAL       100     20 85/01/15

 . total on nev to mennyit
00003 RECORD COPIED                   00003 REKORD MÁSOLVA

 . use mennyit.dbf
 . list
00001   KIS PAL           80
00002   MOLNAR JOZSEF     40
00003   NAGY ANTAL       140
```

UPDATE

Alakja:

UPDATE FROM <adatbázis-állomány> ON <kulcsmező>⇒

⇒ { REPLACE <mezőlista> / ADD <mezőlista> }

Hatása:

A munkaállomány módosítása másik adatbázisból.

Ismertetése:

A USE-zal aktivizált, az UPDATE parancsban is használt <kulcsmező> alapján indexelt vagy sorbarendezett adatbázis felsorolt mezőit módosítja a FROM paraméter után megadott — szintén ugyanilyen elv alapján indexelt vagy sorbarendezett — adatbázis-állomány adatainak alapján. A munkaállomány rekordjának módosítása akkor következik be, ha a <kulcsmező> tartalma megegyezik a FROM <adatbázis-állomány> valamelyik rekordjában levő ugyanilyen nevű mező tartalmával.

Kétféle módosítás paraméterezhető. A REPLACE áttölti az adatokat, a mező eredeti tartalmát felülírva. Az ADD pedig hozzáadja a munkaállománybeli mező tartalmához a FROM-mal meghívott állomány megfelelő adatát.

Mindkét adatbázisban az első rekordon — pl. GO TOP parancs kiadásával — kell megkezdeni a feldolgozást, ami azután rekordonként halad. Ha a <kulcsmező>-k tartalma megegyezik, akkor a helyettesítés, illetve összegzés után újabb rekordra lép a rendszer. Ha a <kulcsmező>-k tartalma nem egyezik meg, akkor az adatbázis-kezelő rendszer lép egy rekordot, mégpedig abban az adatbázisban, ahol az ASC II kód szerint kisebb a <kulcsmező> tartalma.

Példa

```
. use ujarak
. list stru
STRUCTURE FOR FILE:  UJARAK.DBF        AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00003             A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15          UTOLSÓ HASZNALAT NAPJA:
PRIMARY USE DATABASE                   ELSÖDLEGES MUNKATER
FLD     NAME       TYPE  WIDTH  DEC   MEZÖ NEV TIPUS HOSSZ
001     ANYAGKOD     C     003                      TIZEDESEK
002     EGYSEGAR     N     006   002
**TOTAL**                  00010        **ÖSSZESEN**

. list
00001   100    45.50
00002   777   100.10
00003   789    96.70
00004   798   112.00
```

UPDATE

```
. use raktar index Kodok
. list
00008  NEOLUX FEHER        800    700 100    34.00    27200.00
00001  NEOLUX PIROS        300    200 123    44.27    13282.50
00002  NEOLUX KEK          300    100 124    30.00     9000.00
00004  NEOLUX ZOLD         350    150 125    28.60    10010.00
00007  NEOLUX SARGA        400    200 145    35.90    14360.00
00006  VERNODUX PIROS      500    200 777    93.26    46632.50
00003  VERNODUX KEK        500    300 789    96.10    48050.00
00005  VERNODUX SARGA      300    300 798    91.10    27330.00

. update on anyagkod from ujarak replace egysegar
. list
00008  NEOLUX FEHER        800    700 100    45.50    27200.00
00001  NEOLUX PIROS        300    200 123    44.27    13282.50
00002  NEOLUX KEK          300    100 124    30.00     9000.00
00004  NEOLUX ZOLD         350    150 125    28.60    10010.00
00007  NEOLUX SARGA        400    200 145    35.90    14360.00
00006  VERNODUX PIROS      500    200 777   100.10    46632.50
00003  VERNODUX KEK        500    300 789    96.70    48050.00
00005  VERNODUX SARGA      300    300 798   112.00    27330.00

. use beerkezo.dbf
. list stru
STRUCTURE FOR FILE:  BFERKEZO.DBF     AZ ALLOMANY SZERKEZETE:
NUMBER OF RECORDS:   00004            A REKORDOK SZAMA :
DATE OF LAST UPDATE: 85/04/15         UTOLSÓ HASZNALAT NAPJA:
PRIMARY USE DATABASE                  ELSÖDLEGES MUNKATER
FLD     NAME        TYPE  WIDTH  DEC  MEZÖ NEV TIPUS HOSSZ
001     ANYAGKOD      C     003                      TIZEDESEK
002     ERKEZETT      N     004
** TOTAL **              00007         **ÖSSZESEN**

. list
00001   100    100
00002   124    200
00003   789      0
00004   798    150
```

```
. use raktar index kodok
. list
00008  NEOLUX FEHER        800   700 100    45.50    27200.00
00001  NEOLUX PIROS        300   200 123    44.27    13282.50
00002  NEOLUX KEK          300   100 124    30.00     9000.00
00004  NEOLUX ZOLD         350   150 125    28.60    10010.00
00007  NEOLUX SARGA        400   200 145    35.90    14360.00
00006  VERNODUX PIROS      500   200 777   100.10    46632.50
00003  VERNODUX KEK        500   300 789    96.70    48050.00
00005  VERNODUX SARGA      300   300 798   112.00    27330.00

. update on anyagkod from beerkezo add erkezett
. list
00008  NEOLUX FEHER        900   700 100    45.50    27200.00
00001  NEOLUX PIROS        300   200 123    44.27    13282.50
00002  NEOLUX KEK          500   100 124    30.00     9000.00
00004  NEOLUX ZOLD         350   150 125    28.60    10010.00
00007  NEOLUX SARGA        400   200 145    35.90    14360.00
00006  VERNODUX PIROS      500   200 777   100.10    46632.50
00003  VERNODUX KEK        500   300 789    96.70    48050.00
00005  VERNODUX SARGA      450   300 798   112.00    27330.00

. replace all osszertek with erkezett*egysegar
00008 REPLACEMENT(S)                          00008 HELYETTESITÉS

. list
00008  NEOLUX FEHER        900   700 100    45.50    40950.00
00001  NEOLUX PIROS        300   200 123    44.27    13282.50
00002  NEOLUX KEK          500   100 124    30.00    15000.00
00004  NEOLUX ZOLD         350   150 125    28.60    10010.00
00007  NEOLUX SARGA        400   200 145    35.90    14360.00
00006  VERNODUX PIROS      500   200 777   100.10    50050.00
00003  VERNODUX KEK        500   300 789    96.70    48350.00
00005  VERNODUX SARGA      450   300 798   112.00    50400.00
```

USE

Alakja:

USE [<adatbázis-állomány> [INDEX <indexállomány1> [,<indexállomány2> . . .⇒
⇒ ,<indexállomány7>]]]

Hatása:

Adatbázis-állomány megnyitása.

Ismertetése:

A USE paranccsal határozhatjuk meg, hogy az aktuális munkatérben melyik — létező — adatbázis-állománnyal kívánunk dolgozni. Egyúttal az ebben a munkatérben eddig használt adatbázis-állomány — ha volt ilyen — lezáródik. Ebben a parancsban sem szükséges az állománynév-kiterjesztés megadása, mert a parancsszó alapján az adatbázis-kezelő rendszer a jelzett állományt adatbázis-állományként keresi és kezeli.

A parancsban — már létező — indexállományokat is megnevezhetünk, az adatbázis-állományhoz hozzárendelve. Egyszerre maximum 7 indexállomány aktivizálható ily módon. Az adatbázis-kezelő rendszer az elsőként megnevezett indexállomány szerinti sorrendet fogja mutatni. A többi indexállomány csak néhány parancs (pl. APPEND, BROWSE, EDIT, READ, REPLACE) automatikus indexállomány-módosító műveletében vesz részt. (Ezek a parancsok automatikusan módosítják az adatbázishoz USE-zal vagy SET INDEX TO-val csatolt összes indexállományt, ha a rekordok tartalma megváltozik.) Ha a látszólagos sorrendet meg akarjuk változtatni, alkalmazhatjuk a SET INDEX TO parancsot.

Ha valamelyik munkatérben az önálló USE parancsot használjuk — a szabadon választható paraméterek nélkül —, akkor az éppen aktív adatbázis-állomány lezáródik.

Példák

```
. use
. list
. use raktar
. list
00001  NEOLUX PIROS       300   200 123    44.27    13282.50
00002  NEOLUX KEK         500   100 124    30.00    15000.00
00003  VERNODUX KEK       500   300 789    96.70    49450.00
00004  NEOLUX ZOLD        350   150 125    28.60    10010.00
00005  VERNODUX SARGA     450   300 798   112.00    50400.00
00006  VERNODUX PIROS     500   200 777   100.10    50050.00
00007  NEOLUX SARGA       400   200 145    35.90    14360.00
00008  NEOLUX FEHER       900   700 100    45.50    40950.00

. use raktar index kodok, keszlet, szin
. list
00008  NEOLUX FEHER       900   700 100    45.50    40950.00
00001  NEOLUX PIROS       300   200 123    44.27    13282.50
```

USE

00002	NEOLUX KEK	500	100	124	30.00	15000.00
00004	NEOLUX ZOLD	350	150	125	28.60	10010.00
00007	NEOLUX SARGA	400	200	145	35.90	14360.00
00006	VERNODUX PIROS	500	200	777	100.10	50050.00
00003	VERNODUX KEK	500	300	789	96.70	49450.00
00005	VERNODUX SARGA	450	300	798	112.00	50400.00

WAIT

Alakja:

WAIT [TO <memóriaváltozó>]

Hatása:

Választ vár a billentyűzetről.

Ismertetése:

A WAIT parancs felfüggeszti a parancsállomány végrehajtását, és *egy karaktert* vár a billentyűzetről, miközben a képernyőn a WAITING (VÁROK) felirat jelenik meg. Azonnal reagál a bemenő információra — bármelyik billentyű lenyomására —, és folytatódik a felhasználói program végrehajtása.

Ha a ^TO <memóriaváltozó>^ paramétert is megadtuk, akkor a beütött karaktert a rendszer abban tárolja. Ha ez nem írható (pl. <CR>, <enter>, kontrollkarakter stb.), akkor egy szóköz kerül a változóba.

Példa

```
   list memo
**TOTAL**    00 VARIABLES USED  00000 BYTES USED
                  **ÖSSZESEN**  00 MEMÓRIAVÁLTOZÓ 00000 BYTE

 . wait to elso
WAITING    e                                          VÁROK

 . wait to masodik
WAITING    5

 . wait to harmadik
WAITING    f                                          VÁROK

 . disp memo
ELSO          (C)   e
MASODIK       (C)   5
HARMADIK      (C)   f
**TOTAL**    03 VARIABLES USED  00003 BYTES USED
                  **ÖSSZESEN**  03 MEMÓRIAVÁLTOZÓ 00003 BYTE
```

Alakja:

a) ? [<kifejezés [, lista]>]

b) ?? [<kifejezés [, lista]>]

Hatása:

a) Közvetlen válasz a következő sorban.

b) Közvetlen válasz ugyanabban a sorban.

Ismertetése:

A ? jelű parancs a kimeneti egységre írja a kifejezés tartalmát, illetve annak eredményét. A kifejezéslistában több kifejezést is megadhatunk egymástól vesszővel elválasztva. A kifejezések — mint általában — tartalmazhatnak matematikai, logikai, karakterlánc-műveleteket, beépített függvényeket, változókat és ezek kombinációit. Matematikai művelet eredményét annyi tizedesjeggyel adja meg a ? jelű parancs, mint az általunk megadott számok közül a legtöbb tizedesjeggyel rendelkező szám tizedesjegyeinek száma.

A kifejezés nélkül megadott ? jelű parancs egy üres sort eredményez.

A ^??^ parancs soremelés nélkül, ugyanabba a sorba írja a választ, ahol a kérdést feltettük.

Példa

```
. use raktar
. list
00001  NEOLUX PIROS       300   200 123    44.27    13282.50
00002  NEOLUX KEK         500   100 124    30.00    15000.00
00003  VERNODUX KEK       500   300 789    96.70    49450.00
00004  NEOLUX ZOLD        350   150 125    28.60    10010.00
00005  VERNODUX SARGA     450   300 798   112.00    50400.00
00006  VERNODUX PIROS     500   200 777   100.10    50050.00
00007  NEOLUX SARGA       400   200 145    35.90    14360.00
00008  NEOLUX FEHER       900   700 100    45.50    40950.00

. ? anyagnev
NEOLUX PIROS

. ? 'anyagnev'
anyagnev

. ? 'anyagnev'+'e'
anyagneve

. store 'anyagnev' to szoveg
anyagnev
```

```
?
```

```
 . ? $(szoveg,1,5)+' '+$(szoveg,6,3)+'e'
anyag neve

 . go 5
 . ? egysegar
  112.00

 . ? erkezett
   450

 . ? (erkezett*10)-(erkezett*9)
      450

 . ? erkezett*egysegar
    50400.00

 . ? 173/3
    57

 . ? 173/3.0
    57.6

 . ? 173.00/3
    57.66

 . ? 173.00/3.00
      57.66

 . ? 173/3.00000000
      57.66666666
```

Alakja:

a) @ <koordináták> [SAY <kifejezés>] [GET <változó> [PICTURE <formátum>]]
b) @ <koordináták> [SAY <kifejezés> [USING <formátum>]]

Hatása:

a) Szerkesztés és formázás a képernyőn.
b) Formázás a nyomtatón.

Ismertetése:

A @ jelű parancs különösen hatékony eszköze a speciális formátumba rendezett információk megjelenítésének. Ez a parancs két nagyon eltérő formában alkalmazható. Először a közös vonásokat ismertetjük, s utána térünk rá az egyedi tulajdonságokra.

A <koordináták>-ban olyan értékpárt kell megadni — az értékpár elemeit egymástól vesszővel elválasztva —, amely a kimeneti egység *sor,* illetve *oszlop* helyeit határozza meg. (Az adatbázis-kezelő rendszerben a koordináta értékpár mindkét eleme a 0-tól 254-ig terjedő számok közé kell hogy essen.) Az értékpár elemei bármilyen alfanumerikus memóriaváltozók vagy numerikus kifejezések is lehetnek.

A <kifejezés> tartalma (ami lehet karakterlánc-határolók közé helyezett „üzenet" is) a <koordináták> által megadott helyen fog kezdődni.

A kifejezés belsejében nem lehet karakterhatároló, és nem tartalmazhat változót és karakterláncot egyszerre. Ilyenkor előbb STORE parancsot kell kiadni. Például:

```
^@ 10,10 SAY $(anyagnev,1,@('',anyagnev))^  vagy
^@ 10,8  SAY anyagnev+'FESTEK'^  (V. ö. 260. old.)
```

A ^@2,0^ parancs egy üres sort fog eredményezni.

A ^@ 10,20^ parancs törli a 11. sort a 21. oszloptól kezdve.

A SET FORMAT TO parancs határozza meg, hogy a @ jelű parancs mely formáját használhatjuk. (Az adatbázis-kezelő rendszer alapbeállításban a SET FORMAT TO SCREEN-nek megfelelően működik.)

a) SET FORMAT TO SCREEN-ben a @ jelű parancs a képernyőre írja az információkat. A leggyakrabban használatos képernyők koordinátáinál az értékpár első eleme 0 és 23, második eleme pedig 0 és 79 között változhat, a 24 sornak és a 80 oszlopnak megfelelően. (A 0,0 értékpár a képernyő bal felső sarkát (home pozíció), s például a 15,10 értékpár a képernyő 16. sorának és 11. oszlopának találkozási helyét jelöli ki.)

Az első sor az adatbázis-kezelő rendszer üzenetei számára van fenntartva, ezért a felhasználó ennek használatától el kell hogy tekintsen.

SET FORMAT TO SCREEN-ben a @ jelű parancsok bármilyen sorrendben megadhatók, például a ^@ 14,10^ parancs után következhet a ^@ 10,14^ is. Ugyanígy az oszlopok is bármilyen sorrendben következhetnek egymás után.

A @ parancsok kiadása előtt a képernyő az ERASE paranccsal törölhető. Erre a parancsra a gép eltüntet a képernyőről minden információt, érvényteleníti az összes GET-et, és a fénypontot a képernyő bal felső sarkába vezérli.

A SAY paraméter használatával olyan kifejezések tartalma, illetve olyan üzenetek jeleníthetők meg, amelyeket ebben a parancssorban nem kívánunk szerkeszteni (javítani, felülírni) a READ paranccsal.
A GET egy mezőváltozó vagy egy memóriaváltozó pillanatnyi értékének megjelenítésére szolgál. Természetesen a GET-tel szereplő változónak már léteznie kell. (Létrehozhattuk például STORE paranccsal.)
Ezután az adatbázis-kezelő a felhasználói programban továbbhaladva az első READ parancsnál megáll, és visszatér az e sorban levő GET utáni változóra, hogy ezt módosíthassuk. Ebből következik, hogy a változók „betöltéséhez" a későbbiekben egy READ parancsot is meg kell adni.

A változó módosítása után (a <CR> billentyű lenyomására a változó értéke nem változik) a felhasználói program megkeresi a sorrendben következő ^GET <változó>^-t, hogy azt is módosíthassuk. A változók során így végighaladva jut el a felhasználói program újra a READ-ig. SET SCREEN ON-ban (alapbeállítás) szabadon mozoghatunk a teljes képernyős szerkesztés kódjaival (lásd a 4.6. részt) mindaddig, míg az utolsó változót – <CR>-rel vagy teleírással – el nem hagytuk. Ezután a felhasználói program futása a READ-et követő parancssortól folytatódik.
Egy időben legfeljebb 64 GET lehet aktív. A GET-eket érvényteleníti az ERASE (ilyenkor a képernyő is törlődik) vagy a CLEAR GET[S] parancs.
Az adatok a változó adattípusának megfelelően (karakter, numerikus vagy logikai) formálódnak. Ha a GET-be foglalt változó logikai típusú, akkor csak Y, y, T, t, N, n, F, f karakterek vihetők be.
A GET-tel használatos ^PICTURE <formátum>^ paraméter speciális formátum képzésére és a bevitt adatok ellenőrzésére nyújt lehetőséget; egyrészt a képernyőn megjelenő karaktereket vezérli, másrészt korlátozza az adatbevitel során használható karakterek típusát, illetve átalakítja azokat. Ha a „PICTURE <formátum>"-ban megadott hossz rövidebb a változónál, akkor csak a <formátum> által meghatározott hosszúságú adat vihető be.
A PICTURE paraméter <formátum> része a következő – aposztrófok közé helyezendő! – szimbólumokkal tölthető ki. (Az adatbázis-kezelő rendszer a beolvasáskor – a szimbólumok hatására – a szimbólum mellett leírt műveletet végzi el):

9 vagy #	válaszként csak számjegyeket fogad el.
A	csak alfabetikus karaktereket fogad el.
!	a bemenő karaktereket nagybetűssé alakítja.
X	bármilyen karaktert elfogad.
$ vagy *	mint szimbólum jelenik meg.

A PICTURE <formátum> részében szereplő, szimbólumként fel nem sorolt karakterek mindegyike beillesztődik a változóba.

Például a ^@ 10,10 SAY 'Kerem a datumot' GET datum PICTURE '99/99/99' ^ parancsot követő READ parancs végrehajtása eredményeképpen a tizenegyedik sor tizenegyedik oszlopában kezdve megjelenik a következő:

```
Kerem a datumot :  /  /  :
```

jelezve, hogy a változó még szóközökkel van feltöltve. Adatot csak a kilencesek helyére lehet bevinni, és oda is csak számjegyeket. (Ha most sorban leütjük a 8, 5, 0, 6, 1, 2 számjegyek billentyűit, a változó tartalma 85/06/12 lesz.)

A ^SET FORMAT TO <formátumállomány neve>^ paranccsal aktivizálhatunk egy formátumállományt, amit aztán a programban következő READ parancs(ok) aktivizál(nak). A @ jelű parancsok a READ hatására beolvasódnak az állományból. Így több felhasználásra lehet definiálni egy formátumot. (A @ jelű parancs használata nem követeli meg a formátumállományt, mivel annak tartalmát a parancsállományban is leírhatjuk. V. ö. a SET paranccsal.)

SET FORMAT TO SCREEN-ben a SET BELL, a SET COLON és a SET INTENSITY parancs is befolyásolja a @ jelű parancs végrehajtását.

b) SET FORMAT TO PRINT-ben a @ jelű parancs nyomtatóra írja az információkat. A koordináta-értékpár nagyságát elsősorban a nyomtató szélessége és — ha nem akarunk a perforációra nyomtatni — a használatos papír hossza korlátozza. A 0,0 koordináta a papír bal felső sarkát jelenti.

A @ jelű parancsoknak azonban a nyomtatóra való íráshoz *sorrendben kell* lenniük, mert különben (például, ha ^@ 10,30^ után ^@ 10,25^ következik) a nyomtató lapot emel, és a következő oldalra ír. (Hasonló a helyzet a sorok esetében is.) Ha a koordináták lehetővé tennék az információ kinyomtatását ugyanabban a sorban, de a kezdő pont egy már kitöltött koordinátára esik, akkor a nyomtató nem lesz képes a megfelelő nyomtatásra. Előfordulhat, hogy az írófej a jobb oldali legszélső pontra fut, és ott egyetlen karakterpozícióra írja ki a sor hátralevő részébe szánt információt. Nincs akadálya azonban, hogy a @ jelű parancsok kiadása előtt alapállapotba hozzuk a nyomtatót az EJECT paranccsal (lapemelés; és a koordináták 0,0-ra állnak vissza).

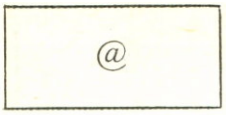
A SAY paramétert használjuk a kifejezések tartalmának, illetve az üzenetek megjelenítésére.

A SET FORMAT TO PRINT-ben használatos ^USING <formátum>^ paraméter speciális megjelenítésre nyújt lehetőséget. Ha a „USING <formátum>"-ban meghatározott hossz rövidebb az adat hosszánál, akkor az adat megcsonkul. A <formátum> rész a következő — aposztrófok közé helyezendő — szimbólumokkal tölthető ki. (Az adatbázis-kezelő rendszer a szimbólumok hatására a szimbólum mellett leírt műveletet végzi el.)

9 vagy #	csak számjegyeket ír.
A	csak alfabetikus karakterket ír.
!	nincs hatása.
X	minden nyomtatható karaktert ír.
$ vagy *	számjegyeket ír, és az elöl álló nullák helyére $ vagy * kerül.

Például a ^@ 10,10 SAY ora*oraber USING '*******.99' ^ parancs eredményeképpen a tizenegyedik sor tizenegyedik oszlopában kezdve jelenik meg a következő információ (ha az óra = 8, az órabér = 12,73):

****101.84

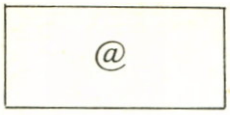


Ez a formátum főleg akkor hasznos, ha olyan adatokat nyomtatunk ki, amelyekkel kapcsolatban követelmény, hogy ne lehessen könnyen módosítani a kész nyomtatást.

Példák

```
**********************  Anyagmegn.FMT  ***********************
@ 3,10 SAY 'UJ ANYAGOK FELVETELE'
@ 7,5 SAY 'Az anyag megnevezese' GET anyagnev
@ 9,5 SAY 'Az anyag kodja' GET anyagkod PICTURE '999'
@ 11,5 SAY 'Beerkezett mennyiseg' GET erkezett
@ 13,5 SAY 'Darabar'
@ 13,24 GET egysegar PICTURE '999.99'
**********************  Anyagmegn.FMT  ***********************
************************  Bevivo.CMD  ************************
USE Raktar
STORE T TO Ism
SET FORMAT TO Anyagmegn
DO WHILE Ism
  APPEND BLANK
  ERASE
* Mint láthatjuk  ez az adatbevitel erősen eltér az APPEND
* által nyujtott formátumtól.
* Ugyanez  a @ parancssorozat  -amely  az  anyagmegn.FMT
* állományban van- alkalmas az adatok javitására  is,  ha az
* APPEND BLANK helyett FIND vagy LOCATE paranccsal
* ráállunk a módositandó rekordra.
  READ
  @ 21,1 SAY 'FOLYTATJUK AZ UJ ANYAGOK FELVETELET?' GET Ism
  READ
ENDDO
USE
RETURN
*********************  Bevivo.CMD vege  **********************
************************  Nyomtato.CMD  ************************
SET FORMAT TO PRINT
USE Raktar INDEX Szin
GO TOP
SET TALK OFF
```

```
STORE 55 TO Sor
STORE '      ' TO Szin
STORE STR(0,5) TO Lapszam
DO WHILE .NOT. EOF
* A nyomtató 54 sor után uj lapot kezd.
  IF Sor >54
    EJECT
* A kinyomtatandó lapszámozás hosszának beállitása.
    DO CASE
      CASE VAL(Lapszam)+1 < 10
        STORE 1 TO Hossz
      CASE VAL(Lapszam)+1 < 100
        STORE 2 TO Hossz
      CASE VAL(Lapszam)+1 > 999
        STORE 5 TO Hossz
    ENDCASE
* A fejrész megszerkesztése.
    STORE STR(VAL(Lapszam)+1,Hossz) TO Lapszam
    @ 2,10 SAY 'LAPSZAM:'
    @ 2,19 SAY Lapszam
    @ 2,30 SAY 'DATUM:'
    @ 2,39 SAY date()
    @ 4,24 SAY 'Raktari keszletek'
    @ 6,10 SAY '---------------- ---- -------- ----------'
    @ 7,26 SAY 'Rak-'
    @ 8,15 SAY 'Festek'
    @ 8,27 SAY 'ta-'
    @ 8,31 SAY 'Darabar'
    @ 8,41 SAY 'Raktari'
    @ 9,27 SAY 'ron'
    @ 9,42 SAY 'ertek'
    @ 10,10 SAY '---------------- ---- -------- ----------'
    STORE 11 TO Sor
  ENDIF
* Uj szin esetén az összegzések elvégzése,
  IF $(anyagnev,@(' ',anyagnev)+1,5) <> Szin
    STORE $(anyagnev,@(' ',anyagnev)+1,5) TO Szin
    STORE STR(#,5) TO Rekord
    SUM (erkezett-kiadva), ((erkezett-kiadva)*egysegar) TO;
```




```
Mkeszlet, Mertek WHILE $(anyagnev,@(' ',anyagnev)+1,5)=Szin
    GO &Rekord
* és a szin fejléc elkészitése.
     @ Sor,0
     @ Sor+1,11 SAY 'Szin: '
     @ Sor+1,17 SAY Szin
     STORE Sor+2 TO Sor
   ENDIF
* A rekord adatainak kiirása.
   STORE $(anyagnev,1,@(' ',anyagnev)) TO Fajta
   @ Sor,10 SAY Fajta
   @ Sor,26 SAY erkezett-kiadva USING '***9'
   @ Sor,30 SAY egysegar USING '99999.99'
   @ Sor,39 SAY (erkezett-kiadva)*egysegar USING '******9.99'
   STORE Sor+1 TO Sor
   SKIP
* Ha a következő rekordban uj szin van  vagy vége az
* állománynak, az összegzések kinyomtatása.
     IF $(anyagnev,@(' ',anyagnev)+1,5) <> Szin .OR. EOF
     @ Sor,10 SAY 'Reszosszeg:'
     @ Sor,26 SAY Mkeszlet USING '***9'
     @ Sor,39 SAY Mertek USING '******9.99'
     RELEASE Mkeszlet, Mertek
     STORE Sor+1 TO Sor
   ENDIF
ENDDO
* A teljes adatbázis összegzése és az eredmények
* kinyomtatása.
SUM (erkezett-kiadva), ((erkezett-kiadva)*egysegar) TO ;
Mkeszlet, Mertek
@ Sor,10 SAY 'Mindosszesen:'
@ Sor,26 SAY Mkeszlet USING '***9'
@ Sor,39 SAY Mertek USING '******9.99'
@ SOR+1,0
SET TALK ON
SET FORMAT TO SCREEN
USE
RETURN
******************  Nyomtato.CMD vege **********************
```

A <Nyomtato.CMD> parancsállomány (szándékosan!) a REPORT parancsnál bemutatott jelentéshez hasonló formátumot hoz létre azért, hogy a két megoldási módot összevethessük.

```
do nyomtato
```

A <Nyomtato.CMD> parancsállomány működtetése után kapott eredménylista a következő:

```
LAPSZAM: 1              DATUM:    85/06/06

                Raktari keszletek

-------------- ---- -------- ----------
                Rak-
     Festek      ta- Darabar    Raktari
                 ron             ertek
-------------- ---- -------- ----------
 Szin: FEHER
NEOLUX          *200    45.50 ***9100.00
Reszosszeg:     *200          ***9100.00

 Szin: KEK
NEOLUX          *400    30.00 **12000.00
VERNODUX        *200    96.70 **19340.00
Reszosszeg:     *600          **31340.00

 Szin: PIROS
NEOLUX          *100    44.27 ***4427.00
VERNODUX        *300   100.10 **30030.00
Reszosszeg:     *400          **34457.00

 Szin: SARGA
VERNODUX        *150   112.00 **16800.00
NEOLUX          *200    35.90 ***7180.00
Reszosszeg:     *350          **23980.00

 Szin: ZOLD
NEOLUX          *200    28.60 ***5720.00
Reszosszeg:     *200          ***5720.00
Mindosszesen:   1750          *104597.00
```

Alakja:

a) * [Bármilyen karakterek]
b) NOTE [Bármilyen karakterek]

Hatása:

Megjegyzések, magyarázatok elhelyezése a parancsállományokban.

Ismertetése:

Az így elhelyezett magyarázatok a felhasználói program áttekinthetőségét segítik, de nem jelennek meg a futás közben. (V. ö. a NOTE paranccsal.)

NEGYEDIK RÉSZ

FÜGGELÉK

12. Műveletek, beépített függvények, kifejezések jellemzői

12.1 A MŰVELETEK ÖSSZEFOGLALÁSA

Az aritmetikai és a logikai műveletekhez alkalmazható a

( ) : zárójel a csoportosításokhoz.

Aritmetikai műveletek

Aritmetikai értékekkel operálnak, az eredmény is aritmetikai érték.

Aritmetikai műveleti jelek:

*	:	szorzás
/	:	osztás
+	:	összeadás
–	:	kivonás

Összehasonlító műveletek (relációk)

Aritmetikai, illetve karakter értékekkel operálnak; az eredmény logikai érték.

Összehasonlító műveletekben szereplő relációs jelek:

<	:	kisebb, mint
>	:	nagyobb, mint
=	:	egyenlő
<>	:	nem egyenlő
<=	:	kisebb vagy egyenlő
>=	:	nagyobb vagy egyenlő

Logikai műveletek

Logikai értékekkel — illetve részkarakterlánc-kereső műveleteknél karakterláncokkal — operálnak; az eredmény logikai érték.

Logikai műveleti jelek:

.NOT.	:	logikai nem
.AND.	:	logikai és
.OR.	:	logikai megengedő vagy
$	:	részkarakterlánc előfordulásának jelzése (arra ad választ, hogy szerepel-e a bal oldali karakterlánc a jobb oldali karakterláncban.)

Karakterlánc-műveletek

Karakterláncokkal operálnak, és az eredmény is karakterlánc.

Karakterlánc-műveleti jelek:

+	:	karakterlánc-összeadás az üres helyekkel együtt
–	:	karakterlánc-összeadás az üres helyek átmozgatásával

12.2 A BEÉPÍTETT FÜGGVÉNYEK ÖSSZEFOGLALÁSA

CHR(<szám>)	a „szám" ASCII kóddá alakítása
EOF	állományvége állapot vizsgálata
FILE({ <változó> / <"állománynév"> / <kifejezés> })	állomány létezésének vizsgálata
INT({ <változó> / <kifejezés> })	egészrész-képzés
LEN({ <karakterlánc> / <változó> })	karakterlánc hosszának megadása
PEEK(<cím>)	érték kimásolása gépi memóriacímről
STR({ <szám> / <kifejezés> / <változó> } ,<hossz>,<tizedesjegyek>)	számból karakterlánc képzése
TEST(<kifejezés>)	kifejezés formai (szintaktikai) ellenőrzése
TRIM({ <változó> / <karakterlánc> / <kifejezés> })	az üres helyek levágása a karakterlánc végéről
TYPE(<kifejezés>)	adat típuskódjának megadása
VAL({ <karakterlánc> / <változó> })	karakterlánc egész számmá alakítása
!({ <karakterlánc> / <változó> })	karakterlánc betűinek kisbetűről nagybetűre alakítása
#	az aktuális rekord számának megadása

$({ <kifejezés> / <karakterlánc> / <változó> } ,<kezdet helye>,<hossz>)	részkarakterlánc kivágása, kiemelése karakterláncból
&<változó>	makró helyettesítése
*	törlendő rekord jelzése
@ ({ <részkarakterlánc> / <változó> } , { <karakterlánc> / <változó> })	részkarakterlánc kezdő pozíciójának megadása a karakterláncban

12.3 A KIFEJEZÉSEK JELLEMZŐI

A kifejezések műveleti jelekkel összekapcsolt operandusokból állnak.

A kifejezések építőelemei:

- műveleti jelek (lásd 12.1. rész)
- állandók (numerikus, karakterlánc és logikai típusú)
- változók (numerikus, karakterlánc és logikai típusú adatbázismezők és memóriaváltozók)
- beépített függvények (lásd 12.2. rész)

A kifejezések kiértékelése balról jobbra halad az elsőbbségi sorrend (prioritás) figyelembevételével. A kiértékelés sorrendje zárójelek alkalmazásával változtatható meg.

13. A parancsszavak és parancsformátumok listája

Az alábbi rövidítéseket használjuk ebben az összefoglalóban.

<kif> = kifejezés
<vált> = változó
<karlc> = karakterlánc
<koord> = koordináta

A <. . . .> jelek közötti szöveg helyett a megnevezett információt kell a felhasználónak begépelnie.

A [. . . .] szögletes zárójelek között a szabadon választható, illetve elhagyható részek vannak.

Az <állománynév> helyére mindig az adott parancs által értelemszerűen megkövetelt típusú állomány nevét kell beírni. Az állománynév-kiterjesztés az 1.7. részben leírt kivételektől eltekintve elhagyható.

ACCEPT ['üzenet'] TO <vált>
adatok elfogadása a billentyűzetről

APPEND [BLANK]

APPEND FROM <állománynév> [{SDF / DELIMITED [WITH <határoló>]}] [{WHILE <kif> / FOR <kif>}]
adatok hozzáfűzése az adatbázishoz

BROWSE
adatok közvetlen módosítása az adatbázisban

CALL <memóriaváltozó>
gépi kódú rutin indítása

CANCEL
a parancsállomány végrehajtásának megszakítása

CASE
a DO CASE szerkezet egyik ága

CHANGE [<érvényességi kör>] FIELD <lista> [{WHILE <kif> / FOR <kif>}]

adatbázismezők tartalmának módosítása

CLEAR

adatbázis-állomány és memóriaváltozók alapállapotba állítása

CONTINUE

a LOCATE parancs folytatása

COPY [<érvényességi kör>] TO <állománynév> [{SDF / DELIMITED [WITH <határoló>] / STRUCTURE [FIELD <lista>]}] ⇒

⇒ [{WHILE <kif> / FOR <kif>}]

a munkaállomány adatainak másolása másik állományba

COPY TO <állománynév> STRUCTURE EXTENDED

a munkaállomány szerkezetének átmásolása (adatként) a megadott új állomány rekordjaiba

COUNT [<érvényességi kör>] [TO <vált>] [{FOR <kif> / WHILE <kif>}]

a munkaállomány rekordjainak összeszámlálása

CREATE

új adatbázis-állomány létrehozása

CREATE <új állomány> FROM <régi állomány>

egy adatbázis-állományt alkot, amelynek struktúráját a régi állományban levő rekordok adatai határozzák meg. (Lásd COPY . . . EXTENDED)

DELETE [<érvényességi kör>] [{FOR <kif> / WHILE <kif>}]

törlendő rekordok kijelölése

DELETE FILE <állománynév>

állomány törlése a mágneslemez nyilvántartásából

DISPLAY [<érvényességi kör>] [{FOR <kif> / WHILE <kif>}] [OFF]

a munkaállomány adatainak megjelenítése a kiválasztott kimeneti egységen

DISPLAY [<érvényességi kör>] [{<kif> / <mező>}] [<,lista>] [{FOR <kif> / WHILE <kif>}]

a munkaállomány kijelölt mezőinek megjelenítése a kiválasztott kimeneti egységen

DISPLAY STRUCTURE

a munkaállomány szerkezetének megjelenítése a kiválasztott kimeneti egységen

DISPLAY MEMORY

a memóriaváltozók tartalmának megjelenítése a kiválasztott kimeneti egységen

DISPLAY FILES [ON <lemezegység>]

az adatbázis-kezelő rendszer állományait tartalmazó lemeznyilvántartás megjelenítése a kiválasztott kimeneti egységen

DO <állománynév>

egy parancsállomány végrehajtásának indítása

DO CASE

parancsok feltételes végrehajtását eredményező parancs

DO WHILE <kif>

egy parancssorozat ismételt végrehajtása

EDIT

adatok közvetlen módosítása az adatbázisban

EJECT

új lap tetejére állás a nyomtatón

ELSE

az IF szerkezet egyik ága

ENDCASE

a DO CASE szerkezet lezárása

ENDDO

a DO WHILE parancs lezárása

ENDIF

az IF parancs lezárása

ERASE

a képernyő törlése

FIND <kifejezés értéke>

a <kifejezés értéke> keresése egy indexelt munkaállomány rekordjai között, az indexelési kulcs rekordonként adódó értéke szerint

GO[TO] { TOP / [RECORD] / BOTTOM } <szám>

a rekordmutató megadott helyre állítása a munkaállományban

IF <kif>

parancssorozatok feltételes végrehajtása

INDEX ON <kif> TO <állománynév>

a kifejezés értéke alapján indexállomány készítése a munkaállományhoz

INPUT ['üzenet'] TO <vált>

a felhasználó által begépelt érték másolása memóriaváltozóba

INSERT [BEFORE] [BLANK]

új rekord beszúrása a munkaállomány két rekordja közé

JOIN TO <állománynév> { WHILE <kif> / FOR <kif> } [FIELDS <lista>]

olyan adatbázist készít, amelyet két másik adatbázis-állománynak – a kifejezésben megfogalmazott feltételek szerint – egybevágó rekordjaiból épít egybe

LIST

adatrekordokat és azokkal kapcsolatos kifejezéseket jelenít meg ugyanúgy, mint a DISPLAY

LOCATE [<érvényességi kör>] FOR <kif>

a kifejezés értékének megfelelő rekordot keres a munkaállományban

LOOP

a DO WHILE szerkezetben levők figyelmen kívül hagyása

MODIFY COMMAND <állománynév>

lehetővé teszi az adatbázis-kezelő rendszerben a parancs- és egyéb állományok írását és módosítását

MODIFY STRUCTURE

a munkaállomány szerkezetének megváltoztatása, és az adatok törlése

NOTE vagy *

futás közben nem látható magyarázó szövegek elhelyezése a parancsállományban

OTHERWISE

a DO CASE szerkezet egyik ága

PACK

a kijelölt rekordok törlése a munkaállományból

POKE

adatok elhelyezése a számítógép memóriájában

QUIT [TO <állománynév vagy rendszerparancs>]

az adatbázis-kezelő rendszer elhagyása és más CP/M szintű parancsok vagy .COM típusú állományok végrehajtatása

READ

a @ jelű parancsokban meghatározott adatbevitel indítása

RECALL [<érvényességi kör>] [{WHILE <kif> / FOR <kif>}]

megszünteti a rekordok törlésre jelölését a munkaállományban

RELEASE {<vált> [<,lista>] / ALL}

a nem kívánt memóriaváltozók törlése

REMARK

futás közben megjelenő megjegyzések elhelyezése a parancsállományban

RENAME <régi állománynév> TO <új állománynév>

a régi állománynevet megváltoztatja az új állománynévre

REPLACE [<érvényességi kör>] <mezőnév1> WITH <kif1> [,<mezőnév2> WITH ⇒
⇒<kif2>] . . .[,<mezőnév5> WITH <kif5>]

a munkaállományban levő mezők tartalmának megváltoztatása

REPORT [<érvényességi kör>] [FORM <állománynév>] [{FOR <kif> / WHILE <kif>}] ⇒
⇒[TO PRINT] [PLAIN]

jelentés, beszámoló (táblázat) készítése

RESET

a lemezek aktivizálása lemezcsere után

RESTORE FROM <állománynév>

memóriaállomány adatainak beolvasása

RETURN

parancsállomány lezárása, a vezérlés visszaadása az aktivizálás helyére

SAVE TO <állománynév>

memóriaváltozók állományba írása

SELECT { PRIMARY / SECONDARY }

átkapcsolás munkaterületek között

SET <paraméter> { OFF / ON / TO <karlc> }

az adatbázis-kezelő rendszer működését jellemző alapbeállítások megváltoztatása

SKIP [±] [<kif/szám>]

az aktuális rekordhoz képest előre vagy hátramozgás a munkaállományban

SORT ON <kulcsmező> TO <állománynév> [{ ASCENDING / DESCENDING }]

a munkaállomány rekordjainak átmásolása egy új állományba, egy mező tartalma szerint növekvő vagy csökkenő sorrendben

STORE { <szám> / <kif> / <karlc> } TO <vált> [, <lista>]

értékadás memóriaváltozó(k)nak

SUM [<érvényességi kör>] { <mező> [, <lista>] / <kifejezés> [, <lista>] } [TO <vált [<,lista>]> ⇒
⇒ [{ FOR <kif> / WHILE <kif> }]

az adatbázismezők tartalmának összegzése

TOTAL ON <kulcsmező> TO <állománynév> [FIELDS <mező [<,lista>]>]

a munkaállomány adatainak csoportosított összegzése és az összegek átmásolása egy új adatbázis-állományba

UPDATE FROM <állománynév> ON <kulcsmező> { ADD <mező [<,lista>]> / REPLACE <mező [<,lista>]> }

a munkaállomány módosítása másik adatbázis adataival

USE <állománynév> [INDEX <állomány [<,lista>]>]

munkaállományként megnyitja az adatbázis-állományt, a megadott indexállomány(ok)kal együtt

USE

lezár minden korábban megnyitott adatbázis-állományt

WAIT [TO <vált>]

megállítja a felhasználói program futását, a billentyűzetről jövő bemeneti adatra vár

? <kif [<,lista>]>

kifejezés (vagy kifejezések) megjelenítése a kiválasztott kimeneti egységen

@ <koord> [{ [SAY <kif>] [GET <vált> [PICTURE '<formátum>']] }]
{ SAY <kif> [USING '<formátum>'] }

a képernyőre vagy a nyomtatásra szánt információ formázása

* vagy NOTE

a parancsállományban (futás közben nem látható) magyarázó szövegek elhelyezése

14. A parancsok feladat szerinti funkcionális csoportosítása

14.1 AZ ÁLLOMÁNYSZERKEZETTEL KAPCSOLATOS PARANCSOK

CREATE

új adatbázis-állomány létrehozása

CREATE <új állomány> FROM <régi állomány>

egy új adatbázis-állományt alkot, amelynek struktúráját a régi állományban levő rekordok adatai határozzák meg. (Lásd 7.5. rész)

COPY TO <új állomány> STRUCTURE

a munkaállomány szerkezetének átmásolása a megadott új állományba

COPY TO <állománynév> STRUCTURE EXTENDED

a munkaállomány szerkezetének átmásolása a megadott új állományba, melynek rekordjai a régi állomány szerkezetét mint adatokat tartalmazzák

DISPLAY STRUCTURE
LIST STRUCTURE

a munkaállomány szerkezetének megjelenítése a kiválasztott kimeneti egységen

MODIFY STRUCTURE

megváltoztatja az állomány nevét, a méreteket, a teljes struktúrát, az adatbázisban levő adatok elvesznek

Az adatbázis struktúrájának megváltoztatása az adatok megőrzésével:

```
USE <régi állomány>
COPY TO <új állomány>
USE <új állomány>
MODIFY STRUCTURE
APPEND FROM <régi állomány>
COPY TO <régi állomány>
USE <régi állomány>
DELETE FILE <új állomány>
```

Mezők átnevezése az adatok megőrzésével:

```
USE <régi állomány>
COPY TO <új állomány> SDF
MODIFY STRUCTURE
APPEND FROM <régi állomány>.TXT SDF
DELETE FILE <új állomány>
```

14.2 ÁLLOMÁNYMŰVELETEK

USE <állománynév> [<indexállomány(ok)>]
munkaállományként megnyitja az adatbázis-állományt

USE <új állomány> [<indexállomány(ok)>]
munkaállományként megnyit egy új adatbázis-állományt és lezárja a régit

USE
lezárja a korábban megnyitott adatbázis-állományt

RENAME <régi állománynév> TO <új állománynév>
a régi állománynevet megváltoztatja az új állománynévre
Megnyitott állományt nem szabad átnevezni!

COPY TO <új állomány>
másolatot készít a munkaállományról a megnevezett állományba

CLEAR
valamennyi állományt lezárja és törli a memóriaváltozókat

SELECT { PRIMARY / SECONDARY }
munkatér kijelölése. A két munkatérben egy időben két állomány lehet nyitva egymástól függetlenül. Adatok átadása az egyik munkatérből a másikba a P. és az S. bevezető karakterekkel lehetséges

DISPLAY FILES [ON<lemezegység>]
az adatbázis-kezelő rendszer állományait tartalmazó lemeznyilvántartás megjelenítése a kiválasztott kimeneti egységen

DISPLAY FILES LIKE <rendszer szerinti azonosító> [ON <lemezegység>]
más típusú állományok listázása a mágneslemezről

QUIT
mindkét munkaterület állományainak lezárása, és az adatbázis-kezelő rendszer elhagyása

14.3 AZ ADATBÁZIS RENDEZÉSE

INDEX ON <kifejezés> TO <indexállomány>

a kifejezés értéke alapján indexállomány készítése a munkaállományhoz. A parancs többszörös kulcsokat is használhat.

SORT ON <kulcsmező> TO <új állomány>

a munkaállomány rekordjainak átmásolása egy új állományba, egy mező tartalma szerint növekvő vagy csökkenő sorrendben. A parancsot más-más kulcssal többször lehet ismételni.

14.4 ADATBÁZISOK KOMBINÁLÁSA

COPY TO <új állomány>

a munkaállományról másolatot készít a megnevezett új állományba

APPEND FROM <állománynév>

a munkaállományhoz rekordokat fűz a megnevezett állományból

UPDATE FROM <állománynév> ON <kulcsmező> { ADD / REPLACE } <mező [, <lista>]>

a munkaállomány módosítása másik adatbázis adataival.
Az adatbázis-kezelő rendszer a munkaállomány adataihoz vagy hozzáad, vagy helyettesíti azokat a megadott állományból vett adatokkal.

JOIN TO <állománynév> { WHILE <kif> / FOR <kif> } [FIELDS <lista>]

olyan új adatbázist készít, amelyet két másik adatbázis-állománynak a feltételek szerint megegyező rekordjaiból épít egybe

14.5 SZERKESZTÉS, FELÜLÍRÁS, ADATOK CSERÉJE

DISPLAY, LIST, BROWSE

lehetővé teszik a rekordok megtekintését, vizsgálatát

DELETE [<érvényességi kör>] [{ FOR <kif> / WHILE <kif> }]

bizonyos rekordokat törlésre jelöl meg a munkaállományban

RECALL [<érvényességi kör>] [{ FOR <kif> / WHILE <kif> }]

megszünteti a munkaállományban a rekordok törlésre való megjelölését

PACK

a törlésre megjelölt rekordok törlése a munkaállományból

EDIT

a kívánt rekordok közvetlen módosítását teszi lehetővé a munkaállományban

REPLACE [<érvényességi kör>] <mezőnév1> WITH <kif1> [,<mezőnév2> WITH⇒
⇒<kif2>] . . . [,<mezőnév5> WITH <kif5>]

mezők tartalmának megváltoztatása a munkaállományban

CHANGE [<érvényességi kör>] FIELD <mező[,lista]> [{WHILE <kif> / FOR <kif>}]

a kijelölt mezők tartalmának megváltoztatása a munkaállományban

@ <koord> . . . GET <vált> [PICTURE '<formátum>']

.
.
.

READ

a változó értékét módosítja

INSERT [BEFORE] [BLANK]

a munkaállomány két rekordja közé egy új beszúrása

UPDATE FROM <állománynév> ON <kulcsmező> {ADD / REPLACE} <mező [<,lista>]>

a munkaállomány módosítása másik adatbázis adataival

MODIFY COMMAND <állománynév>

parancs- és egyéb szöveg típusú állományok írását és módosítását teszi lehetővé az adatbázis-kezelő rendszerben

14.6 VÁLTOZÓK HASZNÁLATA (64 memóriaváltozó és bármennyi mezőnév)

LIST MEMORY, DISPLAY MEMORY

mindkettő kijelzi a memóriaváltozókat, típusukat és tartalmukat is

STORE {<szám> / <karlc> / <kif>} TO <vált> [<,lista>]

létrehozza vagy megváltoztatja a memóriaváltozó(ka)t

RELEASE { <vált> [<,lista>] | ALL }

nem kívánt memóriaváltozók törlése

SAVE TO <állománynév>

a memóriaváltozók beírása az állományba

RESTORE FROM <állománynév>

memóriaállomány adatainak beolvasása

14.7 INTERAKTÍV ADATBEVITEL

WAIT

megállítja a felhasználói program futását, és a billentyűzetről jövő bemeneti adatra vár

WAIT [TO <vált>]

megállítja a felhasználói program futását, és a billentyűzetről érkező adatot egy memóriaváltozóban tárolja

INPUT ['üzenet'] TO <vált>

a felhasználó által beütött érték másolása memóriaváltozóba. Bármilyen típusú adatot elfogad, a karakter típusú adatok idézőjelbe teendők.

ACCEPT ['üzenet'] TO <vált>

karakterláncot fogad el a billentyűzetről, melyet memóriaváltozóban tárol

@ <koord> [[SAY <kif>] [GET <vált> [PICTURE' <formátum>']]]]

.
.
.

READ

a képernyőre menő információ formázása, a változó értékének módosítása

14.8 KERESÉS

SKIP [±] [<kif/szám>]

mozgás a munkaállományban; az aktuális rekordhoz képest előre vagy hátra

GO[TO] { TOP | [RECORD] | BOTTOM } <szám>

a rekordmutató megadott helyre állítása a munkaállományban

FIND <kifejezés értéke>

a <kifejezés értéke>-t keresi egy indexelt munkaállományban, az indexelési kulcs szerint a rekordokban. Nagyon gyorsan dolgozik az indexállománnyal.

LOCATE [<érvényességi kör>] FOR <kif>

.
.
.

CONTINUE

a feltételeknek megfelelő rekord keresése a munkaállományban

14.9 ADATOK MEGJELENÍTÉSE

?, DISPLAY, LIST

kifejezések, rekordok, változók, struktúrák megjelenítése a kiválasztott kimeneti egységen

REPORT [<érvényességi kör>] [FORM <állománynév>] { FOR <kif> / WHILE <kif> } ⇒

⇒ [TO PRINT] [PLAIN]

szabályos, tervezett formátumot készít az adatok megjelenítéséhez, és így írja ki az adatokat minden híváskor

@ <koord> [SAY <kif> [{ USING '<formátum>' / PICTURE '<formátum>' }]]

képernyőre vagy nyomtatóra juttatja a kívánt formájú információt

14.10 PROGRAMOZÁS

A programokat a .CMD kiterjesztésű parancsállományok tárolják.

DO <állománynév>

egy parancsállományban tárolt felhasználói program végrehajtását indítja

DO CASE

CASE <kif>
parancsok
CASE <kif>
parancsok
CASE <kif>
parancsok
.
.
.

* Feltételes végrehajtást eredményező parancs, melyben csak egy ág végrehajtására kerülhet sor.

* A DO CASE szerkezet feltételektől függő egyik ága.

```
   OTHERWISE                    * A szerkezetnek a CASE ágak nem teljesülése esetén
      parancsok                   végrehajtandó ága.
ENDCASE                         * A DO CASE szerkezet lezárása.

DO WHILE <kif>                  * Egy parancssorozat ismételt végrehajtása, a feltétel-
   parancsok                      ként megadott kifejezés logikai értékét a ciklus szer-
   .                              kezetén belül kell megváltoztatni.
   .
   .
   LOOP                         * A szerkezetben levők figyelmen kívül hagyása.
ENDDO                           * A DO WHILE parancs lezárása.

IF <kif>                        * Feltételes végrehajtást eredményező parancs.
   parancsok
ELSE                            * Az IF szerkezet másik ága.
   más parancsok
ENDIF                           * Az IF szerkezet lezárása.
```

15. A méretkorlátozások listája

Egy időben nyitott állományok száma	max.	16
Rekordok száma egy adatbázis-állományban	max.	65535
Mezők száma egy rekordban	max.	32
Karakterek száma rekordonként	max.	1000
Karakterek száma egy mezőben	max.	254
Memóriaváltozók száma	max.	64
Függőben levő GET-ek száma	max.	64
Memóriaváltozók hossza	max.	254 karakter
Memóriaváltozók összterjedelme	max.	1536 karakter
Változók nevének hossza	max.	10 karakter
Állománynevek hossza (CP/M szerint)	max.	8 karakter
Karakterlánc hossza	max.	254 karakter
Parancssor hossza	max.	254 karakter
Fejléc hossza a jelentésben	max.	254 karakter
A második fejléc hossza a jelentésben	max.	60 karakter
Indexelési kulcs hossza	max.	100 karakter
Pontosság (lásd a 2.1.2)		10 számjegy
Számábrázolási tartományok		
		$+1 \times 10^{-63} < R < +1{,}8 \times 10^{63}$
		$-1 \times 10^{-63} > R > -1{,}8 \times 10^{63}$
A SUM és a REPLACE parancsban végrehajtható feladatok száma	max.	5
A USE és a SET parancsban használható indexállományok száma	max.	7

16. A hibaüzenetek listája

BAD DECIMAL WIDTH FIELD	Hibás a tizedesjegyek darabszáma.
BAD FILE NAME	Hibás állománynév.
BAD NAME FIELD	Hibás mezőnév.
BAD TYPE FIELD	Hibás mezőtípus. (Csak C, L vagy N lehet.)
BAD WIDTH FIELD	Hibás mezőszélesség.
CANNOT INSERT – THERE ARE NO RECORDS IN DATABASE FILE	Nem lehet beszúrni, az adatbázisban nincsenek rekordok.
CANNOT OPEN FILE	Nem nyitható meg az állomány.
COMMAND FILE CANNOT BE FOUND	A parancsállomány nem található.
DATA ITEM NOT FOUND	Az adat nem található.
DATABASE IN USE IS NOT INDEXED	Az adatbázis nem indexelt.
DIRECTORY IS FULL	A CP/M állománynyilvántartása (a lemez tartalomjegyzéke) megtelt.
DISK IS FULL	A lemez megtelt.
END OF FILE FOUND UNEXPECTEDLY	Váratlan állományvég elérése. (PACK és újraindexelés lehet kívánatos.)
"FIELD" PHRASE NOT FOUND	Hiányzik a FIELD paraméter.
FILE ALREADY EXISTS	Az állomány már létezik.
FILE DOES NOT EXIST	Az állomány nem létezik.

FILE IS CURRENTLY OPEN	Az állomány jelenleg nyitott.
FORMAT FILE CANNOT BE OPENED	A formátumállományt nem lehet megnyitni.
FORMAT FILE HAS NOT BEEN SET	Nincs formátumállomány SET-tel kijelölve.
ILLEGAL DATE TYPE	Nem engedélyezett adattípus.
ILLEGAL GOTO VALUE	Nem engedélyezett GOTO érték.
ILLEGAL VARIABLE NAME	Nem engedélyezett változónév. (Csak alfanumerikus karakter és közrezárt kettőspont engedélyezett.)
INDEX DOES NOT MATCH DATABASE	Az indexállomány nem az adatbázishoz (a munkaállományhoz) tartozik.
INDEX FILE CANNOT BE OPENED	Az indexállományt nem lehet megnyitni.
JOIN ATTEMPTED TO GENERATE MORE THAN 65,536 RECORDS	A JOIN 65 536-nál több rekord generálását kísérli meg.
KEYS ARE NOT THE SAME LENGTH	A kulcsok hossza nem egyenlő.
MACRO IS NOT A CHARACTER STRING	A makró-ban használt változó nem karakterlánc.
MORE THAN 5 FIELDS TO SUM	Öt mezőnél több szerepel a SUM parancsban.
NESTING LIMIT VIOLATION EXCEEDED	Túllépte az egymásbaágyazás korlátait. (Például DO CASE; IF esetén.)
NO EXPRESSION TO SUM	Nincs kifejezés a SUM parancsban.
NO "FOR" PHRASE	Hiányzik a FOR paraméter.
NO "FROM" PHRASE	Hiányzik a FROM paraméter.
NO FIND	A FIND parancs nem talált olyan rekordot, amenynek indexelési kulcsa megegyezne a keresett értékkel.
NON-NUMERIC EXPRESSION	A kifejezés nem numerikus.
"ON" PHRASE NOT FOUND	Hiányzik az ON paraméter.
OUT OF MEMORY FOR MEMORY VARIABLES	A memóriaváltozók összhossza túllépte a megengedett 1536 byte méretet.

```
*    *** DO 6
     do while ujdat
*       **** IF 10
        if oszlop>60
*         Ha az adatbeviteli-hely mutato elérte a képernyő
*         szélét, kezdeti értékre állitás.
          store 4 to oszlop
*         Az adatbevitelre használt sorok megtisztitása a
*         feliratozás meghagyásával.
          @ 6,10
          @ 7,10
          @ 8,10
        endif **** IF 10
*       Dátum növelés.
        store str(val(mnap)+1,2) to mnap
*       Adatbeviteli hely (a fénypont) jobbra léptetése a
*       soron belül 6 hellyel.
        store oszlop+6 to oszlop
*       Dátum kérése  a feltételezett dátum  megadásával,
*       amely felülirható.
        @ 6,oszlop get mnap picture [!!]
        read
        clear gets
*       **** IF 11
        if mnap=[V]
*         Kilépés a <Bevivo.CMD> parancsállományból.
          return
        endif **** IF 11
*       **** IF 12
        if mnap=[N] .or. mnap>[31]
*         Ha az mnap nevü memóriaváltozó értéke "N" vagy "31"
*         akkor a *** DO 6 és az **** IF 14 feltételének
*         logikai hamis értékre állitásával további végrehaj-
*         tás nélkül kilép a vezérlés a *** DO 6 ciklusból. A
*         ** DO 4 ciklus feltétele még logikai igaz, az adat-
*         bázis-kezelő rendszer uj nevet kér.
          store F to ujdat
        endif **** IF 12
*       **** IF 13
        if mnap=[F]
```

```
*       Ha az mnap memóriaváltozó érteke "F", akkor a ** DO
*       4, a *** DO 6 és az **** IF 14 feltételének logikai
*       hamis értékre állitásaval további végrehajtás nél-
*       kül kilép a vezérlés a *** DO 6 és ** DO 4 ciklu-
*       sokból. A * DO 1 ciklus feltétele még logikai igaz,
*       az adatbázis-kezelő rendszer uj  költségviselőt,
*       munkavégzés helyét, és nevet kér.
        store F to ujdat
        store F to ujnev
      endif **** IF 13
*     **** IF 14
      if ujdat
        store F to jo
*       ***** DO 7
        do while .not. jo
*         Kéri a menüben megadott két karakteres müszakkódot.
          @ 7,oszlop get mmusz picture [!!]
          read
          clear gets
*         ****** IF 15
          if mmusz=[T] .and. mmusz<>[TS]
*           Távollét esetén ujabb menü,  és uj adat a mmusz
*           nevü memóriaváltozóba. A távollét menü ujabb kó-
*           dokat  tartalmaz.  Ezt  a  távollét  menüt  a
*           <Tavol.CMD> parancsállomány irja a képernyőre.
            do tavol
          endif ****** IF 15
*         A bevitt karakterek elemzése következik arra
*         nézve, hogy teljes vagy tört munkanapot ér-
*         telmezzen-e az adatbázis-kezelő rendszer.
*         ****** IF 16
          if mmusz<>[80] .and. mmusz<>[85] .and. mmusz<>[BA];
 .and. mmusz<>[SZ] .and. mmusz<>[SG] .and. mmusz<>[ST] .and.;
mmusz<>[FM] .and. mmusz<>[FU] .and. mmusz<>[TS]
*           Szükség eséten -az elemzés alapjan- a munkában
*           töltött idő bevitele.
            @ 8,oszlop get mledo picture [9.9]
            read
            clear gets
          else
```

```
*           Ellenkező esetben az mledo nevü memóriaváltozó
*           nullázása.
            store 0 to mledo
          endif ****** IF 16
*         A bevitt karakter kombinációk ellenörzése követke-
          zik arra vonatkozólag, hogy a menükben szerepel-
*         tek-e. A jó érték jelzésére a jó nevü memóriaváltozónak .T. érték (igaz) adása.
*         ****** DC 1
          do case
            case mmusz=[80] .or. mmusz=[85] .or. mmusz=[K ];
 .or. mmusz=[NK] .or. mmusz=[NO]
              store T to jo
            case $(mmusz,1,1)=[N] .or. $(mmusz,1,1)=[D] .or.;
 $(mmusz,1,1)=[E] .or. $(mmusz,1,1)=[A] .or. $(mmusz,1,1)=[P];
 .or. $(mmusz,1,1)=[J]
*             ******* IF 17
              if $(mmusz,2,1)=[ ] .or. $(mmusz,2,1)=[H];
 .or. $(mmusz,2,1)=[V] .or. $(mmusz,2,1)=[G]
                store T to jo
              endif ******* IF 17
            case mmusz=[BA] .or. mmusz=[TS] .or. mmusz=[IT];
 .or. mmusz=[LT]
              store T to jo
            case mmusz=[SZ] .or. mmusz=[SG] .or. mmusz=[ST]
              store T to jo
            case mmusz=[FI] .or. mmusz=[FT] .or. mmusz=[FM];
 .or. mmusz=[FU]
               store T to jo
          endcase ****** DC 1
*         ****** IF 18
          if $(mnap,2,1)=[ ]
*           Annak ellenörzése, hogy jól van-e elhelyezve a
*           két karakterből álló változóban a dátum értéke.
*           Ha az elhelyezés nem jó, a jo nevü memóriaváltozóba .F. (hamis) érték kerül.
            store F to jo
          endif ****** IF 18
*         ****** IF 19
          if jo
*           Ha minden megfelel az előirásainknak, vagyis a
```

```
*             jo nevü memóriaváltozó értéke .T. (igaz), a be-
*             billentyüzött adatok a memóriaváltozókbol a re-
*             kordba másolódnak.
              append blank
              repl kod with mkhely+mhely+mmusz, nevkod with;
 mnevKod, elemzo with S.feorkod+S.alkalm, nap with mnap,;
 ledo with mledo
*             ******* IF 20
              if mmusz=[80]
*               A röviditett adatbevitel adatainak átalakitá-
*               sa szükséges formájura.
                repl ledo with 8, kod with $(kod,1,12)+[N ]
              endif ******* IF 20
*             ******* IF 21
              if mmusz=[85]
                repl ledo with 8.5, kod with $(kod,1,12)+[N ]
              endif ******* IF 21
            endif ****** IF 19
          enddo ***** DO 7
        endif **** IF 14
      enddo *** DO 6
*     Képernyő törlése.
      erase
*     *** IF 22
      if ujnev
*     Ha csak uj nevet kértünk, a "fejlec" visszairása.
        @ 1,3 say [Koltseghely]
        @ 1,55 say [Munkavegzes helye]
        @ 2,6 say mkhely
        @ 2,60 say mhely
      endif *** IF 22
    enddo ** DO 4
    erase
enddo * DO 1
* Visszatérés az aktivizáló parancsállományhoz.
return
*****************************************  BEVIV0.cmd
```

```
****************************************  SZEMALL.cmd
erase
* A  <Bevivo.CMD>  használatával  feltöltött  <Munjegy.DBF>
* adatainak  lefejtése  dolgozónként a <Szemall.DBF>  nevü
* átmeneti állományba, kinyomtatás céljából. A <Szemall.DBF>
* adatainak  összegzése  dolgozónként  egy  rekordba,  a
* <Dolgozo.DBF> nevü állományba adatgöngyölités céljából.
@ 5,20 say [MUNKANAPLOK KESZITESE]
release all
store [ ] to atm
@ 7,10 say [Munkanaplok keszitesenek megkezdese
       = K]
@ 8,10 say [Munkanaplok keszitesenek folytatasa  (megszakit;
as utan) = F]
@ 9,10 say [Egy, megnevezett dolgozo munkanaplojanak elkesz;
itese    = E]
@ 11,2 say [Varom a valasztott valtozat betujelet] get atm;
 picture '!'
read
* * IF 1
if atm<>[F] .and. atm<>[K] .and. atm<>[E]
* Nem létező kód esetén kilépés a felhasználói programból.
  return
else
* Létező  kód  esetén a ciklus nevü memóriaváltozó feltöl-
* tése a .not. eof karakterlánccal, DO WHILE ciklus felté-
* tel részébe történő makró-hivás céljából.
  store [.not. eof] to ciklus
endif * IF 1
* Képernyő törlés és türelemre intő felirat kiirása, mert
* a végrehajtás érzékelhető időt vesz igénybe.
erase
@ 4,20 say [Kerem varjon, dolgozom !]
* * IF 2
if atm=[F] .and. .not. file("b:dolgozo")
* Ha folytatást kértek, de nem létezik a b lemezegységben
* levő  mágneslemezen a  <Dolgozo.DBF>  adatbázis-állomány
* (ezt a FILE beépitett függvénnyel vizsgáljuk meg), azaz
* nem volt még feldolgozás, a gép átállitja a változó érté-
* két kezdésre.
  store [K] to atm
endif * IF 2
```

```
* * IF 3
if atm=[K] .and. file("b:dolgozo")
* Ha kezdést kértek,  és létezik a <Dolgozo.DBF> adatbázis
* állomány, akkor az törlésre kerül.
  dele file b:dolgozo
endif * IF 3
* Az egesz felhasználoi program "lelke" a munkaterek közötti
* mozgás.
sele PRIM
use
sele SECO
* * IF 4
if .not. file("b:munjegy.ndx") .or. atm=[K]
* Ha a <Munjegy.DBF> nincs sorba rendezve a  <Munjegy.NDX>
* állományban -ami azt jelenti, hogy ez az állomány még nem
* létezik-; vagy kezdést kértek, megtörténik a rendezés.
  use
  sele PRIM
  use b:munjegy
  index on nevkod to b:munjegy
  use
* A feltételes végrehajtásu ágakból azzal a munkatérrel kell
* kijönni,  amelyikkel abba  beléptünk.  Igy érhető el, hogy
* a ciklus végrehajtása, illetve kikerülése után is ugyanugy
* folytatódhasson a felhasználói program futása. Ha DO WHILE
* ciklus feltételében ".not. eof" szerepel, ez az állomány
* vége vizsgálat mindig az aktualis munkaterületen történik!
  sele SECO
endif * IF 4
use b:munjegy index b:munjegy
go top
* * IF 5
if file("b:dolgozo") .and. atm<>[E]
* Ha  létezik a <Dolgozo.DBF>,  és nem egy  dolgozó  adatait
* kérték, a <Dolgozo.DBF>-ben megkeresi az adatbázis-kezelő
* rendszer az utolsó bejegyzést, és ez alapján FIND pa-
* rancsal megkeresi a névkód szerint indexelt <Munjegy.DBF>-
* ben a dolgozót
  sele PRIM
  use b:dolgozo
  go bottom
```

```
* A keresés kulcsát tartalmazó nevkod mezőváltozó tartalmát
* átmásoljuk az mnevkod nevü memóriaváltozóba, mert makró-
* hivás csak karaktertipusu memóriaváltozóval lehetséges.
  store nevkod to mnevkod
  sele SECO
  find &mnevkod
  skip
endif * IF 5
* * IF 6
if atm=[E]
* Ha egy dolgozó adatait kérték.
  store T to ujra
* ** DO 1
  do while ujra
    store [                                ] to mnev
    @ 14,15 say [Kilepes = BEFEJEZTEM]
    @ 13,2 say [A dolgozo neve(nek kezdete)] get mnev
    read
    clear gets
*   *** IF 7
    if len(trim(mnev))<2
      loop
    endif *** IF 7
*   *** IF 8
    if !(trim(mnev))=[BEFEJEZTEM]
      return
    endif *** IF 8
    sele PRIM
    use nevsor
    store !(trim(mnev)) to mnev
    loca for nev=mnev
*   *** IF 9
    if eof
      @ 16,2 say [&mnev nev(kezdet)u dolgozo nincs a;
 nyilvantartasban !]
    else
      store str(#,3) to mnevkod
      cont
*     **** IF 10
      if eof
        go &mnevkod
        store F to ujra
```

```
      else
        @ 16,2 say [Ket &mnev nevkezdetu dolgozo van a;
nyilvantartasban !]
        @ 17,2 say [Irjon be tobb betut !]
      endif *** IF 10
    endif *** IF 9
  enddo ** DO 1
* ** IF 11
  if file("b:dolgozo")
*   Ha létezik a <Dolgozo.DBF>, és megtalálható benne a
*   dolgozó adatait tartalmazó rekord, töröljük azt.
    use b:dolgozo
    loca for nevkod=mnevkod
*   *** IF 12
    if .not. eof
      dele next 1
      pack
    endif *** IF 12
  endif ** IF 11
* A ciklus nevü memóriaváltozóba bemásoljuk a "nevkod=mnevkod
* .and. .not. eof" karakterláncot.
  store [nevkod=mnevkod .and. .not. eof] to ciklus
  sele SECO
* Megkerestetjük a géppel a <Munjegy.DBF>-ben a dolgozót.
  find &mnevkod
* ** IF 13
  if #<1
*   Ha nem találta, ami azt jelenti hogy az aktuális rekord
*   száma kisebb mint 1;  akkor átkapcsolunk az  elsődleges
*   munkaterületre, azért, hogy a <Nevsor.DBF> alapján jelez-
*   hessük, hogy melyik dolgozó adatai hiányoznak a <Munjegy.
*   DBF>-bol.
    sele PRIM
    use nevsor
    go mnevkod
    store nev to mnev
    @ 18,2 say [&mnev dolgozorol nincs teljesitmeny-adat;
 nyilvantartva !]
    @ 19,2 say [Nyomjon meg egy tetszoleges billentyut !!]
    set consol off
    wait
```

```
    set consol on
    retur.
  endif ** IF 13
endif * IF 6
erase
@ 4,20 say [Kerem varjon, dolgozom !]
sele PRIM
* * IF 14
if .not. file("b:dolgozo")
* Ha nem létezik a <Dolgozo.DBF> adatbázis-állomány,
* létrehozzuk.
  create b:dolgozo from szerkdol
  use b:dolgozo
endif * IF 14
sele SECO
* Felhasználjuk a makró-hivás számára létrehozott  karakter-
* tipusu memóriaváltozót.
* * DO 2
do while &ciklus
  sele PRIM
* ** IF 15
  if file("szemall")
* Ha létezik a <Szemall.DBF>, akkor töröljük ezt az átmeneti
* állományt.
    use
    dele file szemall
    use
  endif ** IF 15
* Létrehozzuk  az uj átmeneti állományt, melynek szerkezetét
* meghatározó rekordokat a <Szerksze.DBF> tartalmazza.
  create szemall from szerksze
  use szemall
* Hozzafüzzük a <Szemall.DBF> állományunkhoz a <Naptar.DBF>
* állományból a hét napjainak nevét (hétfő, kedd, stb).
  append from b:naptar
  sele SECO
* Két munkatér van, és egy harmadik adatbázissal is dolgozni
* kell. Ezért az ahol nevü memóriaváltozóba másoljuk az ak-
* tuális rekord sorszámát, karakteres formában, hogy majd
* egy makró-hivással visszatérhessünk hozzá.
  store str(#,3) to ahol
```

```
* Az üzemszerü,  ciklikus végrehajtást megkezdve, mely a "do
* while &ciklus" parancssorral kezdődik; adatokat, mező-
* tartalmakat másolunk a <Nevsor.DBF> nevü adatbázis-állo-
* mányból memóriaváltozókba.
  store nevkod to mnevkod
  use nevsor
  go &mnevkod
  store nev to mnev
  store feor to mfeor
  store havi:mi to mhavi:mi
  store napi:mi to mnapi:mi
* A dolgozó beosztásának száma 10-el nagyobb,  mint a rekord
* száma, amelyben a beosztást tároljuk.
  store szolg:beo-10 to beosz
  use b:beosztas
  go bottom
* Átmásoljuk az aktuális rekord sorszámát az atm nevü memó-
* riaváltozóba.
  store # to atm
* Megvizsgáljuk, hogy létezik-e ilyen számu beosztás.
* ** IF 16
  if atm<beosz
    store F to atm
  else
*   Ha létezik, megkeressük.
*   *** IF 17
  if beosz<10
      store 1 to hossz
    else
      store 2 to hossz
    endif *** IF 17
    store str(beosz,hossz) to atm
    go &atm
    store T to atm
*   Megvizsgáljuk , hogy a beosztást tartalmazó rekordban a
*   beosztáshoz tartozó első  napra van-e valami bejegyezve.
    *** IF 18
    if b1=[ ]
      store F to atm
    endif *** IF 18
  endif ** IF 16
```

```
* ** IF 19 '
  if .not. atm
    store beosz+10 to beosz
    @ 6,2 say [&beosz szamu beosztas erre a honapra
 nem kerult bevitelre !]
    @ 8,2 say [Ha nem adja meg a beosztast,]
    @ 9,2 say [&mnev dolgozot egymuszakosnak szamolom el !]
    @ 11,10 say [Megadja a beosztast ? Igen=*Y*, Nem=*N*];
 get atm
    read
*   *** IF 20
    if atm
      do beosztbe
    else
*     Az egymüszakos beosztás az első rekordban van.
      go top
    endif *** IF 20
  endif ** IF 19
* Most már kiválasztottunk egy beosztást, a másodlagos munka-
* térben, visszatérhetünk az elsődleges munkatérbe.
  sele PRIM
  use szemall
  go top
* A hét napjai már megvannak, most átmásoljuk a beosztást.
* Az elsődleges munkatérben rekordonként vannak a napok, a
* dátum a rekordszám lesz. A másodlagos munkatérben egy
* rekordban, mezőnként vannak a napok, a mezők neve B1, B2,
* ....Bn.
* ** DO 3
  do while .not. eof
*   *** DC 1
    do case
      case #<10
        store str(#,1) to atm
      othe
        store str(#,2) to atm
    endcase *** DC 1
    repl beoszt with s.b&atm
    skip
  enddo ** DO 3
* A beosztás átmásolása napról napra megtörtént, a beoszt
```

```
* nevü mezőváltozókból a B1,B2,...,Bn nevü mezőváltozók-
* ba.
  sele SECO
* Visszaállitjuk pozicionkat az elhagyott <Munjegy.DBF>
* indexelt állományban, az ahol nevü memóriaváltozó értékét
* felhasználva.
  use b:munjegy index b:munjegy
  go &ahol
* Itt kezdődik a dolgozó teljesitményének elemzése napról
* napra. A <Munjegy DBF> -egyéb feldolgozások érdekében-
* egy-egy rekordban egy dolgozónak ugyanazon a napon, egy-
* etlen munkahelyen, változatlan munkakörülmények között
* ledolgozott idejét tartalmazza, 17 byte terjedelemben.
* Igy egy dolgozó egy napja a munkahely és a munkakörülmények
* alapján részletekre bontva több rekordban is szerepelhet.
* Az adatok napi csoportositásban kerülnek az átmeneti
* <Szemall.DBF> adatbázisba, ahol egy rekord egyetlen nap
* adatait tartalmazza, 32 mező illetve 170 byte terjedelemben.
* A mezők a munkahely és a munkavégzés körülményei alapján
* lesznek kitöltve.
* A feladat megoldásában résztvevő két adatbázis szerkezete:

*          Szemall.CMD                          Dolgozo.CMD
* 001   NAPTIP      C    1    0        NEVKOD      C    3    0
* 002   BEOSZT      C    1    0        ELEMZO      C    2    0
* 003   TAVOL       C    2    0        PEKSZEK     C    8    0
* 004   LEDO        N    4    1        LEDO        N    5    1
* 005   TOSUM       N    4    1        FOLY:TAV    C   17    0
* 006   TO25        N    4    1        TO25        N    5    1
* 007   TO50        N    4    1        TO50        N    5    1
* 008   TO100       N    4    1        TO100       N    5    1
* 009   DPO         N    4    1        DPO         N    5    1
* 010   EPO         N    4    1        EPO         N    5    1
* 011   APO         N    4    1        APO         N    5    1
* 012   MAGPO       N    4    1        MAGPO       N    5    1
* 013   MELYPO      N    4    1        MELYPO      N    5    1
* 014   SZEPO       N    4    1        SZEPO       N    5    1
* 015   AFIO        N    4    1        AFIO        N    5    1
* 016   AFTO        N    4    1        AFTO        N    5    1
* 017   KESZLO      N    4    1        KESZLO      N    5    1
* 018   HOPO        N    4    1        HOPO        N    5    1
```

```
* 019  VEZPO      N   4   1        VEZPO      N   5   1
* 020  GEPPO      N   4   1        GEPPO      N   5   1
* 021  ORAKIEG    N   4   1        ORAKIEG    N   5   1
* 022  KOMMUSZO   N   4   1        KOMMUSZO   N   5   1
* A további 10 mezőt nem ez a parancsállomány használja fel.
* A következő ciklus egyszeri végrehajtása a <Munjegy.DBF>
* egy rekordjának elemzését és az elemzéstől függően mező-
* tartalmak másolását végzi el. A ciklusfeltétel egy dol-
* gozó adatainak elöbb leirt feldolgozását határozza meg.
* A továbbiakról a * DO 2 ciklusnak  a ciklus nevü memória-
* változóban tárolt feltétele alapjan születik döntés.
* ** DO '4
  do while nevkod=mnevkod .and. .not. eof
    store nap to mnap
    sele PRIM
    go &mnap
    store $(s.kod,5,2) to atm
*   *** DC 2
    do case
      case atm=[BA] .or. atm=[SZ] .or. atm=[SG] .or. atm=[ST]
        repl tavol with atm, s.ledo with 0
* Az atm nevü memóriaváltozót nem minden CASE ágban töltjük
* fel csillagokkal; ennek a következő DO CASE szerkezetek-
* ben lesz jelentősége.
        store [**] to atm
      case atm=[FM] .or. atm=[FU] .or. atm=[TS] .or. atm=[IT];
 .or. atm=[LT]
        repl tavol with atm, s.ledo with 0
        store [**] to atm
      case atm=[FI]
        repl afio with afio+s.ledo, s.ledo with 0
        store [**] to atm
      case atm=[FT]
        repl afto with afto+s.ledo, s.ledo with 0
        store [**] to atm
      case $(atm,1,1)=[N]
        repl ledo with ledo+s.ledo
      case $(atm,1,1)=[D]
        repl ledo with ledo+s.ledo, dpo with dpo+s.ledo
      case $(atm,1,1)=[E]
        repl ledo with ledo+s.ledo, epo with epo+s.ledo
```

```
      case $(atm,1,1)=[A]
        repl tosum with tosum+s.ledo
      case $(atm,1,1)=[P]
        repl tosum with tosum+s.ledo, dpo with dpo+s.ledo
      case $(atm,1,1)=[J]
        repl tosum with tosum+s.ledo, epo with epo+s.ledo
      case $(atm,1,1)=[K]
        repl kommuszo with kommuszo+s.ledo
        store [**] to atm
    endcase *** DC 2
*   *** DC 3
    do case
* Itt a csillaggal való feltöltésre a magyarázat. Mivel a DO
* CASE szerkezet csak az első logikai igaz értéket adó ágat
* hajtja végre, ujabb ellenörzések nélkül elérhetjük, hogy
* csak a munkaügyi rendeleteknek megfelelő esetekben legye-
* nek az adatok átmásolva, vagyis csak akkor, ha az atm
* nevü memóriaváltozó nem csillagokat tartalmaz. A következő
* felhasználói programsorban szereplő első CASE ág nem fel-
* tétlenül szükséges, de igy gyorsabb a felhasznalói program
* végrehajtása.
      case atm=[**]
        store [**] to atm
      case $(atm,2,1)=[K]
        repl ledo with ledo-s.ledo, keszlo with keszlo+s.ledo
      case $(atm,2,1)=[O]
        repl ledo with ledo-s.ledo, orakieg with orakieg+;
s.ledo
      case $(atm,2,1)=[H]
        repl hopo with hopo+s.ledo
      case $(atm,2,1)=[V]
        repl vezpo with vezpo+s.ledo
      case $(atm,2,1)=[G]
        repl geppo with geppo+s.ledo
    endcase *** DC 3
*   *** IF 21
    if atm<>[**]
      store $(s.kod,12,1) to atm
    endif *** IF 21
*   A kód nevü mezőváltozó tartalma karakterek sorozata, az
*   atm nevü memóriaváltozóba azért kerül e karakterek egy
```

```
*    része, hogy ne kelljen minden CASE ág értékelésekor a
*    gépnek előállitania a részkarakterlánc-képző beépitett
*    függvény értékét.
*    *** DC 4
     do case
       case atm=[**]
         store [**[ to atm
       case atm=[A]
         repl apo with apo+s.ledo
       case atm=[M]
         repl magpo with magpo+s.ledo
       case atm=[E]
         repl melypo with melypo+s.ledo
       case atm=[S]
         repl szepo with szepo+s.ledo
       case atm=[U]
         repl apo with apo+s.ledo, szepo with szepo+s.ledo
       case atm=[G]
         repl magpo with magpo+s.ledo, szepo with szepo+s.ledo
       case atm=[L]
         repl melypo with melypo+s.ledo, szepo with szepo+;
s.ledo
     endcase *** DC 4
     sele SECO
     skip
   enddo ** DO 4
* Egy a dolgozóra jellemző adat (amely az elemző nevü mező-
* változóban található) tárolása az atm nevü memóriaválto-
* zóban.
   store elemzo to atm
* Megint el kell hagynunk átmenetileg az adatbázist, de most
* már azt is vizsgálnunk kell, hogy feldolgoztuk-e az utolsó
* rekordot. Ha már az utolsó rekordot is feldolgoztuk, akkor
* az ahol nevü memóriaváltozónak a 000 karakterlánc értéket
* adjuk, ellenkező esetben pedig az aktuális rekord számát
* karakter formában (az STR beépitett függvény segitségével).
* ** IF 22
   if .not. eof
     store str(#,3) to ahol
   else
   store [000] to ahol
   endif ** IF 22
```

```
  sele PRIM
  go top
* Ujra végigmegyünk a <Szemall.DBF> adatbázison,  és elszá-
* moltatjuk a tulórákat a munkatörvénykönyv rendelkezései
* szerint; azaz adott szempontok szerinti vizsgálatok ered-
* ményétől függően bizonyos mezőtartalmakat átmásolunk.
* ** DO 5
  do while .not. eof
*   *** DC 5
    do case
      case tosum>4
        repl to25 with 2, to50 with 2, to100 with tosum-4
      case tosum>2
        repl to25 with 2, to50 with tosum-2
      othe
        repl to25 with tosum
    endcase *** DC 5
    skip
  enddo ** DO 5
  go top
* A tosum nevü mezőben levő adatot már felhasználtuk az e-
* löbbi  ciklusban,  nincs  már  az  itt  található  adatra
* szükségünk. Az adatbázis összes tosum nevü mezőjébe 0 érté-
* ket másolunk, mert ezt -vagyis, hogy az összes tusum nevü
* mezőváltozó értéke egyforma (0)- majd további összegzések-
* hez használjuk fel.
  repl all tosum with 0
* ** IF 23
  if file ("egyke")
*   Egy rekordos átmeneti adatbázis-állomány az összegzett
*   adatok számára, melyeket majd egy uj szerkezet szerint
*   akarunk egy már létező, másik adatbázis ujabb rekord-
*   jaként tárolni.
    dele file egyke
  endif ** IF 23
* Az azonos nevü mezők értékeinek összegzése, és az eredmé-
* nyek bemásolása az <Egyke.DBF> nevü, egy rekordos adatbá-
* zis-állományba.
  total on tosum to egyke
  use b:dolgozo
* Az <Egyke.DBF> nevü adatbázisban tárolt adatok átmásolása
```

```
* az eddigi állományoktól eltérő szerkezetü, dolgozónként
* egy rekordot tartalmazó <Dolgozo.DBF> nevü adatbázisba.
  append from egyke
  go bottom
  sele SECO.
  use szemall
  go top
  sele PRIM
* A dolgozóra jellemző adatok átmásolása az elsődleges
* munkatérben lévő <Dolgozo.DBF> nevü adatbázis nevkod és
* elemzo nevü mezőibe az mnevkod és az atm nevü memóriaváltozókból.
  repl p.nevkod with mnevkod, p.elemzo with atm
  sele SECO
* Különböző tipusu napok megszámlálása.
* Hogy  ne kelljen tul sok memóriaváltazót használni  - ezek
* száma  is korlátozott - a megszámlálást két részre bontva
* végezzük el, az első és a második alkalommal is ugyanazokat a memóriaváltozókat használjuk majd.
  count for s.tavol=[BA] to els
  count for s.tavol=[SZ] to mas
  count for s.tavol=[SG] to har
  count for s.tavol=[ST] to neg
  sele PRIM
* A megszámlálási eredményeket az adatbázis-állományokban
* használható mezők korlátozott száma miatt -az STR be-
* épitett függvény és a karakterlánc összeadó müvelet al-
* kalmazásával- karakter formában, egy mezőben (a foly:tav
* nevü mezőben) tároljuk.
  repl p.foly:tav with str(els,2)+str(mas,2)+str(har,2)+;
str(neg,2)
  rele els, mas, har, neg
  sele SECO
  count for s.tavol=[FM] to els
  count for s.tavol=[FU] to mas
  count for s.tavol=[IT] to har
  count for s.tavol=[LT] to neg
  sele PRIM
  repl p.foly:tav with $(p.foly:tav,1,8)+str(els,2)+;
str(mas,2)+str(har,2)+str(neg,2)
  rele els, mas, har, neg
```

```
* Kinyomtatjuk az eredményeket, a <Nyomtato.CMD> parancs-
* állománnyal. Egy ilyen eredménylap formája látható a 4.
* 5. ábrán, jelen parancsállományt követő oldalakon.
  DO NYOMTATO
  sele SECO
  use b:munjegy
* Visszaállitjuk pozicionkat a <Munjegy.DBF> adatbázis-ál-
* lományban.
 ** IF 24
  if ahol=[000]
    go bottom
    skip
  else
    go &ahol
  endif ** IF 24
enddo * DO 2
* Visszatérés az aktivizáló állományhoz.
return
************************************************  SZEMALL.cmd
```

Nev: KOVACS PETER FEOR: 12345678 1985 SZEPTEMBER

Teljesitmenyelszamolasi eredmenylap

D A T	N A P	B E O	TELJ. IDO	TULORA 25%	TULORA 50%	TULORA 100%	EJSZ. ORAK	DELUT ORAK	ALAGU POTL.	MELYS POTL.	MAGAS POTL.	KESZL ORAK	AFI	AFT
1	C	M	8.5	-	-	-	-	-	-	-	-	-	-	-
2	P	M	8.0	-	-	-	-	-	-	-	-	-	-	-
3	O	X		-	-	-	-	-	-	-	-	-	-	-
4	V	O		-	-	-	-	-	-	-	-	-	-	-
5	H	M	8.5	-	-	-	-	8.5	-	-	-	-	-	-
6	K	M	8.5	-	-	-	-	8.5	8.5	-	-	-	-	-
7	S	M	8.5	-	-	-	8.5	-	8.5	-	-	-	-	-
8	C	M	8.5	-	-	-	-	-	8.5	-	-	-	-	-
9	P	M	8.0	-	-	-	-	-	14.0	-	-	6.0	-	-
10	O	X		2.0	2.0	4.0	-	-	8.0	-	-	-	-	-
11	V	O		-	-	-	-	-	-	-	-	-	-	-
12	H	M	8.5	2.0	2.0	-	-	4.0	-	12.5	-	-	-	-
13	K	M	8.5	2.0	2.0	-	-	4.0	-	12.5	-	-	-	-
14	S	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
15	C	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
16	P	M	8.0	-	-	-	-	-	-	8.0	-	-	-	-
17	O	X		-	-	-	-	-	-	-	-	-	-	-
18	V	O		-	-	-	-	-	-	-	-	-	-	-
19	H	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
20	K	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
21	S	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
22	C	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
23	P	M	8.0	-	-	-	-	-	-	8.0	-	-	-	-
24	O	X		-	-	-	-	-	-	-	-	-	-	-
25	V	O		-	-	-	-	-	-	-	-	-	-	-
26	H	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
27	K	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
28	S	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
29	C	M	8.5	-	-	-	-	-	-	8.5	-	-	-	-
30	P	M	8.0	-	-	-	-	-	-	8.0	-	-	-	-
31	O	X		-	-	-	-	-	-	-	-	-	-	-
OSSZ:			184.5	6.0	6.0	4.0	8.5	25.0	47.5	134.0	-	6.0	-	-

Ho potl.: 85.0 Kisgep.p: 12.5

Kotelezo munkaido: 22 nap, 184 ora. Ledolg.nap:22 Szabsn: 0 Betegn: 0 Egyeb: 0
Napi munkaido: 8.4 ora.

Elvont pihenonap

D A T	N A P	B E O	TELJ. IDO	EJSZ. ORAK	DELUT ORAK	ALAGU POTL.	MELYS POTL.	MAGAS POTL.	FOLYT POTL.	HO POTL.	PO SZER.	VEZ.P	KISG.
11	V	O	8.0	-	-	8.0	-	-	-	-	-	-	-
18	V	O	4.0	-	-	-	4.0	-	-	-	-	-	-
OSSZ:			12.0	-	-	8.0	4.0	-	-	-	-	-	-

JEGYZET:

Nyomtatva: 1985. 10. 02.

K O Z P O N T I	Haviber	Napiber	Atl.napiber	1985 SZEPTEMBER
Berelszamolasi Vallalat				Nev:
4. Foosztaly 2. csoport		Oraber	Atl. oraber	KOVACS PETER
				FEOR: 12345678

	Orak	Megnevezes	Forint
Kotelezo munkaora: 184			
Kotelezo munkanap: 22	22 nap	1. Alapber	
Napi munkaido: 8.4	16.0	2. Tulora torzs	
Teljesitett ido: 184.5	8.50	3. Tulora potlek	
Ledolgozott nap: 22		4. Elvont pihenonap	
Szabadsagnap: 0		5. Munkakori potlek	
Beteg nap: 0		Mankopenz, beveteli jutalek	
Egyeb nap: 0		Premium	
		6. Csoportvezetoi potlek	
		Tuzrend. Munkavedelmi potlek	
JEGYZET:		Emelogep biztonsagi megbizott	
	8.5	7. Ejszakai potlek	
	47.5	Alaguti potlek	
		8. Mellekfoglalkozas	
		9. Delutani muszak potlek	
		Ejszakai muszak potlek	
		Folyamatos muszak potlek	
	2 2	10. Fizetett unnep	
		11. AFI	
		AFT	
		12. Fizetett szabadsag	
		Felmondasi ber	
	6.0	13. Keszenlet	
	85.0	Hopotlek	
		Magassagi potlek	
	134.0	Melysegi potlek	
		Szerelesi potlek	
	12.5	Kisgep kezeloi potlek	
		Vezetesi potlek	
		14. Altalanos berpotlek	310
		15. Csaladi potlek	
		16. OSSZES JARANDOSAG	
		17. Nyugdijjarulek	
		18.	
		Levonas	
		19. Levonas	
		20.	
		Levonas	
		21. Reszfizetes	
		22. KIFIZETENDO	

Elvont pihenonap elszamolas			
Megnevezes	Ora	Elsz alap	Ft
Telj. ido	12.0	24.0	
Ejsz.potl			
Delut.mp.			
Ejsz. mp.			
Folyt.mp.			
Csopvez p			
Prem.Bevj			
Alaguti p	8.0	16.0	
Hopotlek.			
Magassagi			
Melysegi	4.0	8.0	
Szerelesi			
Kisgepkez			
Vezet. p.			
OSSZESEN:	XXXX	XXXX	

Munkaruha	
Premium	
Jutalom	
Jarulekalap	
Munkasszallo dij	
Eloleg	
Karterites	
Utazasi koltseg terites	
Jogtalan ber	
Jogtalan TB.	
Tuzelo	
Allami Biztosito	
KST	
Idegen vall. tart.	
Letiltas	
OTP	
Honvedelmi hozzajarulas	
Javito-nevelo munka	

Teljesitmenyelszamolasi eredmenylappal egyutt beallitotta: igazolta:

Belyegzo.

Elszamolta: Ellenorizte:

Nyomtatva: 1985. 10. 02.

18. Jelölések

A könyvben szereplő **példákban** a *kisbetűk* azt jelölik, hogy azt a szöveget a *felhasználó* írta be, a *nagybetűk* pedig azt, amit az *adatbázis-kezelő* kijelez, illetve válaszként megad.

A **szövegben** a *nagybetűket* elsősorban az adatbázis-kezelő *parancsainak megjelenítésére* használjuk.

A ^ ^ jelek közötti szövegek a leírásban azokat a parancsokat és szövegeket tartalmazzák, amelyeket a felhasználó ír be. *A ^ szimbólumot a valóságban nem kell begépelni!*

A [ ] jelek közötti szövegek a dBASE II, illetve a PROP-BASE parancsainak azt a részét mutatják, amelyek szabadon választhatók („opcionálisak"), szükség lehet rá(juk) vagy elhagyhatjuk (őket).

A { } jelek közötti egymás fölött elhelyezkedő parancsrészek közül az egyik megadása kötelező.

A [{ }] jelek közötti egymás fölött elhelyezkedő parancsrészek közül az egyik alkalmazható, de az egész el is hagyható.

. . . jel azt mutatja, hogy a [ ] jelek közötti parancsrész többször ismételhető.

A < > jelek közé (az adatbázis-kezelő rendszer **parancsaiban**) a megnevezésnek megfelelő információt kell beírni. Például: <állománynév> azt jelenti, hogy itt egy állomány nevét kell beírni.
Arra is használjuk a < > jeleket a szövegben, hogy mezőneveket és állományneveket, illetve számítógép-billentyűket jelölünk velük.

A ⇒ jellel azt jelöltük, hogy a parancs teljes formája nem fért el egy sorban, ezért két (vagy több) sorba írtuk. Ezzel csak a folytatólagosságot mutatjuk, alkalmazáskor ezt nem kell begépelni!

Az <enter> és a <CR> kulcsszavak azt jelentik, hogy az „enter" vagy „carriage return" (kocsivissza) nyomógombot kell kezelni, ha a szövegben vagy egy példában e jelek valamelyike szerepel.

A ^ vagy a <CTRL> jel, melyet nagybetű követ, egy kontrollkarakter jelent. A kontrollkarakter a számítógép vezérlésére szolgál, mint például a 4.6. részben is láthatjuk. Nyomjuk le

a <CTRL> (kontroll)billentyűt, és közben üssük le a megfelelő betű billentyűjét. (Az elengedés sorrendje fordított.) A kontrollkarakter jelét követő *zárójeles betűcsoport* az adott kontrollkarakter ASCII jele. Erre azoknál a — nem nagy számban fellelhető — számítógépeknél van szükség, amelyeken nincs kontroll billentyű. Az ilyen billentyűzeneten viszont megtalálhatók az ASCII jelek.

18.1 A PARANCSOK LEÍRÁSÁBAN HASZNÁLT SZIMBÓLUMOK

Mint mondottuk, a < > jelek közé zárt szövegekkel nevezzük meg a parancsokban az olyan részek tartalmát, melyek kitöltése kötelező. Az így zárójelezett szöveg pontosan meghatározza a beírandó tartalmat.

Tekintsük végig a leggyakrabban előforduló megnevezéseket, a hozzájuk fűzött magyarázatokkal együtt!

<cím>

decimális szám — számítógép-memóriacímet jelöl.

<érvényességi kör>

amikor a parancs hatását több rekordra fejti ki, meghatározhatjuk, hogy az adott esetben hány rekord vegyen részt a műveletben.

Az <érvényességi kör> szimbólum helyére az alábbi lehetőségek valmelyikét kell beírni:

– ^ALL^	:	minden rekord
– ^RECORD <szám>^	:	a <szám>-ban megadott rekordszámú rekord
– ^NEXT <szám>^	:	az aktuális rekordtól kezdve annyi rekord, amennyit a <szám> meghatároz.

Az <érvényességi kör> sok esetben szabadon választható. Ha nem adjuk meg külön, akkor az adatbázis-kezelő rendszer az egyes parancsoknál felsorolt alapértelmezés szerint jár el.

<formátum>

a PICTURE és a USING paraméterekben (lásd a 10. részben a 110. oldalon) itt kell megadni a @ jelű parancsnál leírt formátumszimbólumokat.

<határoló>

más rendszerben (pl. BASIC, PASCAL, COBOL, FORTRAN, PL/1 stb.) használatos adat-, illetve karakterhatárolót (például ,) kell megadni.

<hossz>

a kijelölt kezdőponttól (azt is beleértve) a karakterlánc tagjainak száma.

<karakterlánc>

megjeleníthető karakterek bármilyen sorozata, beleértve a szóközt is. Nem megjeleníthető karakter például a kontrollkarakter.

<kezdet>

egy karakterláncban valamilyen művelethez (függvényhez) kijelölt rész első karakterének sorszáma.

<kifejezés>

műveleti jelekkel összekötött, zárójelekkel csoportosított állandók, változók és beépített függvények együttese.

<kifejezés értéke>

indexelt adatbázis indexelési kulcsában megadott kifejezés szerint adódó érték, mindig az aktuális rekord adatai alapján.

<koordináták>

két, egymástól vesszővel elválasztott számjegy, illetve numerikus típusú memóriaváltozó vagy kifejezés, amelyek a képernyő valamely pontját adják eredményül. Az első szám a sort, a második az oszlopot határozza meg. A képernyő bal felső sarka a 0,0 pozíció.
(A 10,20 koordináták a képernyő 11. sorának 21. pozícióját határozzák meg; az 5,8 koordinátapár a 6. sor 9. pozícióját jelenti; stb.)

<kulcs>

az a kifejezés (pl. INDEX stb.) vagy mező (pl. SORT stb.), amelynek figyelembevételével hajtja végre az adatbázis-kezelő rendszer a parancsot.

<lemezegység>

a mágneslemez-meghajtó (driver) és a benne levő hajlékony mágneslemez (floppy disk).

<részkarakterlánc>

egy adott karakterlánc bizonyos része.

<üzenet>

bármilyen karakter(lánc).

<'üzenet'>

bármilyen karakter(lánc), karakterhatárolók között.

<változó>

mező-, illetve memóriaváltozó.

18.2 NÉHÁNY FOGALOM

Az itt felsoroltak jórészt általános érvényűek a számítástechnikában, de egyes meghatározásokat e könyv tárgyköréhez szorosabban illesztettünk.

Adat

elemzés, illetve számítás céljaira szervezett információ.
Mindazok az értékek, számok, szavak, szövegek és más információk, amelyek a számítógépes feldolgozáshoz szükségesek, vagy amelyek a feldolgozás eredményeképpen megjelennek.

Adatfeldolgozás

az a folyamat, amelyben az adatokkal műveleteket végzünk.

Részei:

- adatelőkészítés,
- adatbevitel,
- adatértékelés,
- adatmódosítás,
- adatmásolás,
- adatösszegzés,
- új adatok szolgáltatása,
- adatkivitel,
- stb.

Adathordozó

minden olyan eszköz, amelyre adatokat rögzítünk.

Aktuális rekord

a meghívott adatbázisban a munka folyamán mindig van egy kitüntetett rekord, amelyre a rekordmutató mutat, azaz amelynek sorszámát az adatbázis-kezelő rendszer megjegyzi, meghatározva ezzel helyzetét az adatbázisban. Aktuális rekord az a rekord, amelyen az adatbázis-kezelő éppen áll. Az adatbázis-állomány meghívása, illetve GO TOP parancs után az első rekord; GO BOTTOM parancs kiadása, illetve az állomány végén való túllépés esetén az utolsó rekord az aktuális rekord.

Alapállapot

az adatbázis-kezelő rendszer bejelentkezési állapota; a képernyő első oszlopában egy pont jelenik meg, és a rendszer parancsra vár.

Alapbeállítás

az adatbázis-kezelő rendszer bejelentkezésekor a SET paranccsal állítható rendszerfunkciók helyzete.

Alapértelmezés

az <érvényességi kör> paraméterrel rendelkező parancsok hatásának köre, ha a paraméterben nem adtuk meg másként.

Algoritmus

egy feladat megoldására szolgáló eljárás, melyben az egyes lépések az előző lépések eredményeit hasznosítják.

Állománynév-hozzárendelés

az adatbázis-kezelő rendszer a maximum 8 betűs állománynévhez automatikusan hozzáteszi a pontot és a funkciónak megfelelő 3 betűs CP/M állománynév-kiterjesztést.

Fénypont (cursor)

a képernyőn jól feltűnő — általában villogó — fény, amely megmutatja, hogy hová kerülnek a képernyőre jutó információk.

Információ

minden, ami tudomásunkra hoz valamit, ami egy adott kérdésben a bizonytalanságunkat csökkenti.

Karakter

betű, számjegy, egyéb jel (írásjel, speciális jel).

Karakterlánc

karakterek együttese (például szöveg).

Kódolás

áttérés valamely más jelkészlet, szabályrendszer használatára.

Közvetett mód

amelyben az adatbázis-kezelő rendszer parancsállományból kiolvasott parancsokat hajt végre.

Közvetlen mód

amelyben az adatbázis-kezelő rendszer terminálról (billentyűzetről) érkező parancsokat hajt végre.

Munkaállomány

a USE paranccsal megnyitott adatbázis-állomány.

Munkatér

az adatbázis-kezelő rendszer egyidejűleg két, egymástól független adatbázis-állománnyal is tud dolgozni. A rendszer ilyenkor úgy működik, mintha fizikailag két teljesen különálló egységben folyna az állománykezelés, de bizonyos összeköttetésekre van lehetőség.

Paraméter

a parancsoknak a parancsszót követő olyan része, amely módosítja (többnyire korlátozza) a parancs működését.

Parancs

adott formájú karakterlánc, amely a számítógép számára értelmezhető és végrehajtható műveletet jelöl ki. Az adatbázis-kezelő rendszer parancsai mindig parancsszóval kezdődnek, ezt követhetik a paraméterek és a kifejezések. A paraméterek tetszőleges sorrendben állhatnak.

Parancsszó

az adatbázis-kezelő rendszer parancsai mindig parancsszóval kezdődnek.
Vannak olyan parancsszavak, amelyek
— önmagukban,
— önmagukban is, és paraméterekkel is,
— csak paraméterekkel együtt
alkotnak parancsot.

Pont

az adatbázis-kezelő rendszer alapállapotban (közvetlen mód) egy ponttal jelentkezik be, várva parancsainkat.

Pontosság

az adatbázis-kezelő rendszer pontossága 10 számjegy. Ez azt jelenti, hogy a tíz számjegynél hosszabb számok esetében az elölről számított 10. számjegy után már csak 0-ák szerepelnek, függetlenül attól, hogy a számítás eredményeképpen (vagy más módon) a rendszerbe kerülő szám a 10. számjegy után milyen számjegyeket tartalmazott.

Program

az adatokon végzett műveletek egymásutánját meghatározó parancsok sorozata.

Teljes képernyős szerkesztés

a szerkesztés kódjainak (lásd 4.6. rész) alkalmazásával lehetővé teszi a képernyőn a mozgást.

* * *

Kiadja: Számítástechnika-alkalmazási Vállalat
Felelős kiadó: Havass Miklós vezérigazgató
Felelős szerkesztő: Lukács Erzsébet
Műszaki vezető: Molnár Zoltán
Műszaki szerkesztő: G. Müller Zsuzsa
A fedelet tervezte: Németh Mihály
Megjelent: 29 (A/5) ív terjedelemben
Készült a SZÜV nyomdaüzemében
A nyomdai megrendelés törzsszáma:
Budapest, 1987

Ára: 197,– Ft

dB