Az eljárásokat és függvényeket célszerű egyesíteni és egy közös PRG kiterjesztésű (PROCEDURE) állományban tárolni. Ilyen állományt a

SET PROCEDURE TO p

paranccsal lehet megnyitni. *p* – az állomány neve.
A *p* állomány megnyitása után hozzá lehet férni a benne tárolt eljárásához és a függvényekhez. Egyszerre csak egy Procedure-állományt lehet megnyitni. A már megnyitott állományt a

SET PROCEDURE TO

paranccsal kell bezárni.

Függvények
Ha rutin eredményként csak egy értéket ad vissza, akkor célszerű függvényt szerkeszteni. A függvény (mint az eljárás is) külső vagy belső lehet. A függvény első parancsa a

FUNCTION p

parancs. *p* – a függvény neve.
Ha a függvénynek paramétereket kell átadni, akkor a FUNCTION parancs után PARAMETERS parancsot kell elhelyezni. A függvény utolsó parancsa a

RETURN <kifejezés>

parancs, amely a <kifejezés> értékét adja vissza. Ha a függvényben nincs RETURN parancs, akkor az alapértelmezés szerint a

RETURN .T.

parancs kerül végrehajtásra. (A függvény visszaadott értéke .T. lesz).
A függvényeket a kifejezésekben lehet alkalmazni:
P ( ) – ha a függvénynek nincsenek paraméterei, vagy
P (x, y, ...) – ha a függvény paramétereket tartalmaz (x, y,... az aktuális paraméterek).
A másik lehetőség: a függvény aktivizálása a DO paranccsal. Ebben az esetben a RETURN parancs <kifejezést> nem tartalmazhat.

Példa

```
SET TALK OFF
? negyzet(2)
WAIT WINDOWS "Rendben"

FUNCTION negyzet
PARAMETERS ennek
RETURN ennek*ennek   && visszatér a szám négyzetével
```

A változók hatóköre
Az eljárásokba és függvényekbe az adatokat nemcsak paraméterekkel lehet átadni. A FoxPro lehetőség ad **globális** változók alkalmazására, amelyek érvényesek a program összes rutinjában. A globális változókat a

PUBLIC <változók> / <tömbök>

paranccsal lehet deklarálni. Ezeknek a változóknak a kezdőértéke -. F..
A főprogramban deklarált változó globális lesz (minden eljárásból hozzá lehet férni). A parancsban létrehozott és alkalmazott változó alapértelmezésként *public* tulajdonságú lesz.
Lokális változókat is lehet létrehozni (deklarálni):

PRIVATE <változók> / <tömbök>

A lokális változók csak abban a rutinban érvényesek, amelyben deklarálva vannak. Ha az eljárás egy másik eljáráshoz fordul, akkor a lokális változó abban is érvényes. A változókat csoportosítva is lehet deklarálni (maszkokat alkalmazva):

PRIVATE ALL / LIKE <mask> / **EXCEPT** <mask>.

A program ideiglenes leállítása

WAIT [<kifejezés>] **[TO** <változó>] **[WINDOW] [NOWAIT] [CLEAR]** [**TIMEOUT** <másodperc>]

A <kifejezés> megjelenik a képernyőn. Ha nincs megadva a <kifejezés>, akkor a képernyőn a

Press any key to continue ...

szöveg jelenik meg.
A <kifejezést> ajánlatos megadni, mivel ezzel bővebb információt lehet közölni a felhasználónak;
TO <változó> - a felhasználó által bevitt karakter a <változóba> kerül. Ez a <változó> csak egy karaktert képes befogadni.
WINDOW – a <kifejezés> a rendszerüzenet-ablakba kerül (a képernyő jobb felső sarkában),
NOWAIT – az üzenet a billentyű lenyomása nélkül eltűnik a rendszer-üzenetablakból,
CLEAR - a rendszer-üzenetablak eltávolítása,
TIMEOUT <másodperc> - a program <másodpercig> vár a billentyű lenyomására. Ha nincs megadva ez az opció, akkor a várakozási idő végtelen.
Ha a képernyő tartalmát nem akarjuk megváltoztatni, akkor a

WAIT ''

parancsot kell alkalmazni, amely semmit nem ír ki a képernyőre. A WAIT parancsot lekérdezésekre is lehet alkalmazni.

Memóriaváltozók

Adatok bevitele és értékadás

A változóknak a

<változó> = <kifejezés>

paranccsal értékeket lehet adni.
Ha a *<változó>* nem létezik, akkor a parancs azt létrehozza. A *<változó>* a *<kifejezés>* értékét és típusát kapja. Ha egy értéket több változónak akarunk adni, akkor a

STORE <kifejezés> **TO** <változók listája>

parancsot alkalmazhatjuk. A változók megkapják a kifejezés kiértékelt értékét.

INPUT [<szöveg>] **TO** <változó>
ACCEPT <szöveg> **TO** <változó>

Az ACCEPT paranccsal csak karakter típusú adatokat lehet bevinni, és ebben az esetben a karaktersort nem kell idézőjelek közé tenni.

?, ?? és ??? parancsok

A **? / ??** parancsokkal kifejezéseket lehet kiértékelni és megjeleníteni a képernyőn:

? <kifejezés-1> [**PICTURE** ...] [**FUNCTION** ...] [**AT** N] [**FONT** C1 [, N2]]
[**STYLE** C2] [, <kifejezés – 2> ...]
parancs a kiértékelt eredményt a következő sor elejére helyezi el,

?? <kifejezés-1> [**PICTURE** ...] [**FUNCTION** ...] [**AT** N] [**FONT** C1 [, N2]]
[**STYLE** C2] [, <kifejezés – 2> ...]
parancs a kurzor pozíciójától kezdve írja ki az eredményt;
AT N – az N-dik oszloptól kezdve írja ki,
A ?/ ?? parancsok vezérlő karaktereket is tartalmazhatnak.

Példa
? 'Vigyáz! ', CHR(7) && 7- a hangjelzés kódja

Tömbök és tömbkezelő parancsok
Tömb definiálását a
DECLARE <változó> (N1 [, N2])
(vagy **DIMENSION** <változó> (N1 [, N2]))

Adatbázis-állomány mezőinek másolása tömbbe (vagy memóriaváltozókba)

SCATTER [FIELDS <mezőlista>] **[MEMO] TO** <tömb> / **TO** <tömb>
BLANK/ MEMVAR/ MEMVAR BLANK

A SCATTER parancs az aktuális rekord összes vagy a megnevezett mezők értékeit egy tömbbe (TO <tömb>), vagy memóriaváltozókba (MEMVAR) másolja át. A tömböt (memóriaváltozókat) a SCATTER parancs hozza létre. A létrehozott memóriaváltozók neve megegyezik a mezők nevével. Ezért szükséges az összetett nevek alkalmazása: *M.<változó>* vagy *M->változó* (M - Memory). Ez fontos lehet, ha a program egyidejűleg azonos nevű mező és memóriaváltozót tartalmaz.
A memo-mezők csak akkor vesznek részt a másolásban, ha megadtuk a **MEMO** paramétert.
A **BLANK** paraméter memóriaváltozókat (tömböt) hoz létre, hozzárendeli a megfelelő mezők típusát, de nem tölti fel adatokkal.

A tömb (memóriaváltozók) másolása az adatbázis mezőibe

GATHER FROM <tömb> / **MEMVAR [FIELDS** <mezőlista>] **[MEMO]**

FROM <tömb> - a tömb elemeinek értékeit az aktuális rekord mezőibe másolja át.

Adatbázis rekordjainak másolása egy kétdimenziós tömbbe

COPY TO ARRAY <tömb> **[FIELDS** <mezők> [<hatókör>]]
[FOR <feltétel- 1>] **[WHILE** <feltétel – 2>]

Tömb hozzáfűzése az adatbázis-állományhoz

APPEND FROM ARRAY <tömb> **[FOR** <feltétel>] **[FIELDS** <mezőlista>]

Tömb másolása az adatbázis rekordjaiba (a már létező adatok helyébe)

REPLACE FROM ARRAY <tömb> [<hatókör>] **[FIELDS** <mezők>]
[FOR <feltétel-1>] **[WHILE** <feltétel-2>]

Tömbelem törlése

ADEL (<tömb>, <elem(sor)szám> [, 2]**).**

Az ADEL függvény törli a <tömb> elemét (ha a tömb kétdimenziós, akkor törli a tömb sorát vagy oszlopát). Az *<elem (sor)szám>* a tömbelem (sor vagy oszlop) sorszáma, amelyet törölni kell. Ha megadjuk a harmadik paramétert, és annak az értéke 2, akkor a függvény törli az oszlopot. A függvény nem változtatja a tömb elemeinek számát. A törölt elem (sor, oszlop) a tömb végére kerül és. F. értéket kap. Visszaadott érték: 1 (ha az elem törölve volt).

Elem vagy sor (oszlop) beszúrása a tömbbe

AINS (<tömb>, <elem(sor)szám>, [, 2])

A függvény paraméterei megegyeznek az ADEL függvény paramétereivel. Az új elem (sor/oszlop) értéke: .F., a következő elemek nagyobb index-értéket kapnak. Az utolsó elem a tömbből kicsordul és megsemmisül.
Visszaadott érték: 1 (ha az elem törölve volt).

Tömb elemek másolása egy másik tömbbe

ACOPY (<tömb1>,<tömb2>**[,** N1 **[,** N2 **[,** N3 **]]])**

A függvény a <tömb1> (forrástömb) elemeit a <tömb2>-be (céltömbbe) másolja át.
N1- kezdőindex (<tömb1> eleme, amelytől a másolás kezdődik), alapértelmezése: 1,
N2 – az átmásolandó elemek száma (alapértelmezés: az első elemtől az utolsó elemig,
N3- kezdőindex a <tömb2>-ben, ahová a másolt tömb első eleme kerül (alapértelmezése: 1).

Keresés tömbben

ASCAN (<tömb>, <keresett érték> [,<kezdőindex> [,<elemszám>]])

A függvény a <tömbben> keresi a <keresett értéket>.
Ha adott a <kezdőindex>, akkor a keresés a megadott elemtől kezdődik (alapértelmezés – az első elemtől),
<elemszám> - a függvény ennyi elemet fog átvizsgálni (alapértelmezés: az utolsó elemig).
A függvény visszaadott értéke: a keresett értékkel megegyező első elem indexe, vagy 0 (ha a tömb nem tartalmaz ilyen elemet).

Tömb elemek rendezése

ASORT (<tömb> [,<kezdőindex> [,<elemszám] [, N]]])

Ha adott a <kezdőindex>, akkor a rendezés az adott elemtől kezdődik (alapértelmezés – az első elemtől), <elemszám> - elemek száma, amelyek részt vesznek a rendezésben (alapértelmezés: az utolsó elemig).
N – megadja a rendezés sorrendjét: 0 – növekvő sorrend, 1 - csökkenő sorrend.

Tömb elemeinek (sorainak, oszlopainak) száma

ALEN (<tömb> [, N]**)**

A függvény megállapítja a tömb elemeinek, sorainak vagy oszlopainak számát. Ha N=1, akkor a függvény a sorok, ha N=2 akkor az oszlopok számát adja vissza.

Memóriaváltozók mentése, visszatöltése, megjelenítése

SAVE TO <MEM-állomány>**/TO MEMO** <memo-mező> **[ALL LIKE/ EXCEPT** <maszk>**]**

ALL LIKE – menti a maszknak megfelelő változókat.
Az ALL EXCEPT opció esetén azok a változók kerülnek mentésre, amelyek a maszknak nem felelnek meg. A parancs alapértelmezése - az összes memóriaváltozó.

Biztonsági szempontok miatt hasznos lehet a következő parancs:

SET SAFETY ON/OFF

Ha az állomány, amelybe menteni akarjuk a változókat, már létezik, és a SET SAFETY parancs ON opciója van beállítva, akkor az állomány felülírása előtt a rendszer figyelmeztet:

<állomány-név> **already exists, overwrite it?**
(az állomány már létezik, felülírja azt?)

Memóriaváltozók visszatöltése

RESTORE FROM <állomány> **/FROM MEMO** <memo- mező> **[ADDITIVE]**

A memóriaváltozók törlését a memóriából a

RELEASE <memóriaváltozó-lista> / **ALL [LIKE/EXCEPT** <maszk>**]**

parancsal lehet végrehajtani. A

CLEAR MEMORY

parancsal az összes változót lehet törölni.

A változók és értékük kiírhatok a képernyőre:

DISPLAY/LIST MEMORY [LIKE <maszk>**]**

vagy a nyomtatóra (állományba):

DISPLAY/LIST MEMORY [LIKE <maszk>**] TO PRINTER/ FILE** <állomány>

Konstansok definiálása

Ha egy konstanst a programban többször akarunk alkalmazni, akkor a program elején a

#DEFINE <K> <kifejezés>

parancsal lehet deklarálni.
A kifejezés kiértékelődik és a *K* nevű konstans megkapja ezt az értéket.
Pl.: #DEFINE maxertek 100

Vezérlési utasítások (parancsok)

IF utasítás

IF <feltétel>
<parancsok – 1>
[ELSE
<parancsok –2>]
ENDIF

Ha a *<feltétel>* igaz (.T.), akkor a *<parancsok-1>* kerül végrehajtásra, ha - nem, akkor - a *<parancsok-2>*. Ha az ELSE rész hiányzik, akkor a .F. esetén a vezérlés az ENDIF utáni parancsra kerül.

DO CASE utasítás

DO CASE
CASE <feltétel – 1>
<parancsok - 1>
CASE <feltétel – 2>
<parancsok - 2>
...
[OTHERWISE
<parancsok-N>]
ENDCASE

Ciklusok

Parancsok feltételes ciklikus ismétlése

DO WHILE <feltétel>
<Parancsok>
[LOOP]
[EXIT]
ENDDO

A ciklus törzsében EXIT és LOOP parancsokat lehet elhelyezni.
LOOP - a vezérlést a ciklus elejére adja vissza (a DO WHILE parancsra).
EXIT – függetlenül a <feltétel> értékétől a vezérlést átadja az ENDDO utáni parancsra (kilép a ciklusból).

Ciklus végrehajtása meghatározott számú lépésben

FOR <ciklus-paraméter> = N1 TO N3 **[STEP** N2**]**
<parancsok>
[LOOP]
[EXIT]
ENDFOR (vagy **NEXT**)

a <ciklus-paraméter> - számláló, amely a ciklus végrehajtását vezérli. A FOR utasítás minden alkalommal a <parancsok> végrehajtása előtt ellenőrzi a <ciklus-paraméter> értékét. Ha a számláló nem lépte át a végértéket, akkor a ciklus törzsét hajtja végre, ha átlépte – akkor kilép a ciklusból. N1, N2, N3 – numerikus kifejezések.
N1- a <ciklus-paraméter> kezdő értéke,
N3- a <ciklus-paraméter> végértéke,
N2- lépésköz, amely pozitív vagy negatív lehet. Alapértelmezése: 1.
A <ciklus-paraméter> a ciklus végrehajtásakor a következő értékeket vesz fel:

N1, N1+N2, N1+2*N2, N1+3*N2, ... , N3.

A <parancsok> a ciklus-paraméternek a felsorolt értékeivel valósul meg.
Bizonyos N1, N2, N3 értékeknél előfordulhat, hogy a számláló pontosan nem kapja meg az N3 értéket. Pl., ha N2>0 és N1+R*N2<N3, de N1+(R+1)*N2>N3.
Ha a lépésköz =1, akkor nem szükséges a STEP 1 opciót megadni.

Rekordok ciklikus vizsgálata

SCAN [<hatókör>] [**FOR** <feltétel-1>] [**WHILE** <feltétel –2>]
<parancsok>
[LOOP]
[EXIT]
ENDSCAN

Helyettesítési függvények

Makro-helyettesítés

&C – makro-helyettesítési függvény.
C – karakter típusú változó (a hossza maximálisan 255 lehet). A parancsban a *C* tartalma a &C függvény helyén lesz elhelyezve. A program futtatása alatt a &C függvény lehetőséget ad a parancsok paramétereinek megváltoztatására, ami rugalmasabbá teszi a programot.

Példa

```
C='adatok'
USE &C
```

Az *USE adatok* parancsot hoznak létre.

Példa

```
Db= 'dolgozok'
Tag=Dolgrend'
USE &db INDEX &tag
```

Makro-helyettesítés más formában is kiírható: a & karakter helyett zárójeleket lehet alkalmazni. Ebben az esetben a makro-helyettesítést gyorsabban hajtja végre.

Példa

```
USE (db) INDEX (tag)
```

EVALUTE (C)
függvény kiértékeli a C karakter típusú kifejezést.

51. Példa

? EVALUTE('3+4') – az eredmény 7.
? EVALUTE('nev') - az eredmény: SZABÓ A.J., ha a *nev* változó (mező) ezt az értéket tartalmazza.

Ha nincs szükség a függvény visszaadott értékére, akkor a függvényt értékadás nélkül lehet alkalmazni:

= f (<paraméterek>**)**

f - a függvény neve.

Hibakezelés

A FoxPro eszközöket tartalmaz a hibák elemzésére és kezelésére. A hibák feldolgozására olyan rutinokat lehet előkészíteni, amelyek csak hiba esetén futnak. Ezzel azt lehet elérni, hogy egy hiba miatt a program ne álljon le. A rutin elemzi a hibának az okát, és ha lehet, kiküszöböli a hibát. A FoxPro-ban mindegyik hibához egy kód (sorszám) tartozik, amelynek az értékét az

ERROR( )

függvény segítségével lehet megkapni. Ismertetjük a fontosabb hibakódokat:

1 –az állomány nem létezik.
3 – az állomány meg van nyitva.
4 – az állomány már létezik.
12 – a változót nem lehet megtalálni.
24 – nem numerikus kifejezés
36 – numerikus túlcsordulás.
41 – a kifejezés nem karakter típusú.
46 – nem megengedett érték.
47 – nincs elég memória a meghajtón.
58 – a LOG () függvény argumentuma zéró vagy negatív.
61 – hiba a SQRT () függvény paraméterében.
62 – hiba a ** (^) művelet paraméterében.
108 – az állományt más program használja.
109 – a rekordot más program használja.
110 – az állomány kizárólagos használata tiltott.
111 – nem lehet beírni az írásvédett (read-only) állományba.
125 – a nyomtató nincs kész állapotban.
130 – a rekord nincs zárolva.
291 – hiba az ASIN () függvény paraméterében.
292 – hiba a LOG10 () függvény paraméterében.
293 – hiba az ACOS () függvény paraméterében.
1002 – hiba az I/O műveletben.
1101 – az állományt nem lehet megnyitni.
1104 – hiba az állomány olvasásában.
1105 – hiba az állomány beírásában.
1149 –a puffer megnyitására nincs elegendő memória.
1162 – a Procedure nem található.
1307 – osztás nullával.
1502 – a rekord zárolva van, nem lehet módosítani.
1503 –nem lehet zárolni (locked) az állományt.

ON ERROR [<parancs>]

A <parancs> akkor lesz végrehajtva, ha a program futtatása alatt egy hiba keletkezet. Leggyakrabban a DO parancsot szokták alkalmazni.

ON READERROR [<parancs>]

A <parancs> akkor lesz végrehajtva, ha az input (beviteli) művelet alatt hiba jött létre. A rutinban a hiba feldolgozása nem csak RETURN paranccsal fejeződhet be. Lehet a

RETRY

parancsot is alkalmazni. A RETRY a vezérlést ugyanarra a parancsra adja át, amely a rutinhoz fordult. A RETRY lehetőséget ad a műveletek ismétlésére, amíg a hibát nem küszöböljük ki.

Példa

```
SET TALK OFF
ON ERROR DO hibak  && hiba esetén végrehajtódik a hibak eljárás.
USE nincs

PROCEDURE hibak
hiba=ERROR()
?CHR(7) && hangjelzés
* mindegyik CASE elágazás egy-egy hibát fog feldolgozni
DO CASE
CASE hiba=1
? "hiányzik a dbf állomány"
* itt lehet létrehozni
CASE hiba=9
    * nem dbf az állomány
ENDCASE
RETURN
```

Az állományokkal végezhető általános műveletek

Lemeztartalom megjelenítése:

DIR [[**ON**] <meghajtó>] [[**LIKE**] <útvonal>][<maszk>] [**TO PRINTER/ TO FILE** <f>]

<maszk> - az állományok maszkja, amely a * és ? karaktereket tartalmaz,
az eredmény a nyomtatóra (**TO PRINTER**) vagy állományba (**TO FILE** <f>) irányítható.
Az alapértelmezés szerint a parancs a DBF kiterjesztésű állományokat listázza.
A nem aktív állományokat a

ERASE <filenév> (vagy **DELETE FILE** <filenév>)

parancsokkal lehet megsemmisíteni.
Ezekben a parancsokban az állomány kiterjesztését kell megadni.

RENAME <f> **TO** <f1>

parancs az állományt átnevezi.
<f1> – az állomány új neve.
A nem aktív állományt egy másik állományba a

COPY FILE <fájl1> **TO** <fájl2>

parancsal lehet másolni.
A <fájl1> és <fájl2> állományok kiterjesztéseit kötelező megadni.

Az adatbázis állomány másolása

Az aktív adatbázis állományt egy új állományba (f1) lehet átmásolni. Ezt a műveletet a

COPY TO <f1> [<hatókör>] [**FIELDS** <mezőlista>] [**FOR** <feltétel1>]
[**WHILE** <feltétel2>] [**TYPE** <állomány típusa>] [**WITH CDX**]

parancs hajtja végre.
FIELDS <mezőlista> – ha a mezőlista nincs megadva, akkor át lesz másolva az összes mező. A FIELDS paraméter azokat a mezőket választja ki, amelyeket akarunk átmásolni.
a <hatókör> alapértelmezése **ALL,**
WITH CDX – ha létezik strukturált indexállomány, akkor azt is átmásolja.
Ha a **TYPE** paraméter nincs megadva, akkor a létrehozott állomány kiterjesztése - **dbf.** A COPY TO parancsban az <f1> állománynak olyan kiterjesztést lehet adni, amely más alkalmazásoknak is megfelel.
Az <állomány típusa> fontosabb lehetséges értékei:
SDF (System Data Format) – az állomány ASCII formátumú, a rekordjai fix hosszúságúak, amelyeket CR és LF (kocsi vissza - soremelés) karakterek zárják. A mezők fix hosszúságúak, határoló jel nélkül. Az állomány kiterjesztésének alapértelmezése: TXT,
DELIMITED [WITH <c> / WITH BLANK/ WITH TAB] – az állomány ASCII formátumú, rekordjai változó hosszúságúak, amelyeket CR és LF (kocsi vissza - soremelés) karakterek zárnak. A mezők változó hosszúságúak és köztük határoló jel van elhelyezve.
A határoló jel alapértelmezése - vessző (**DELIMITED**),
DELIMITED WITH BLANK – a mezők szóközzel vannak elkülönítve,
DELIMITED WITH TAB – a mezőket tabulátorjel különíti el,
DELIMITED WITH <c> – a mezők **c** karakterrel vannak elkülönítve.
Az állomány kiterjesztésének alapértelmezése: .TXT.

Példa

```
USE adatok
COPY NEXT 3 TO temp TYPE DELIMITED && létrehoz egy szöveges *fájlt a
mezők vesszővel elválasztva
MODIFY FILE temp.txt
DELETE FILE temp.txt
```

Rekordok hozzáfűzése az adatbázis-állományhoz

Egy DBF vagy TXT kiterjesztésű állományból az aktív állomány végére rekordokat lehet fűzni:

APPEND FROM <f> [**FIELDS** <mezőlista>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>] [**TYPE** <állomány típusa>]

Ha a **TYPE** paraméter nincs megadva, akkor az <f> állomány kiterjesztése - DBF. Az <f> állományból csak azok a mezők lesznek átmásolva, amelyeknek a neve megegyezik az

aktív állomány mezőinek nevével.
FIELDS <mezőlista> - <f> állományból csak az adott mezők másolódnak át.
Az APPEND FROM parancs akkor is alkalmazható, ha az <f> állomány más alkalmazásokban jött létre. Ebben az esetben a TYPE paramétert kell alkalmazni (mint a COPY TO parancsban).
PL., új rekordok szöveges állományokból jöhetnek létre. Az APPEND FROM parancs kényelmes eszköz lehet az adatok konvertálására az ASCII- állományokból a DBF állományokba. Például, az

APPEND FROM <f.TXT> DELIMETED WITH BLANK

parancs a f.TXT állományból szöveges rekordokat visz át az aktív állományba. A szöveges állományban tárolt mezők közt szóköz van elhelyezve.

Példa

```
SET SAFE OFF
USE bikini
COPY STRUCTURE TO backup
APPEND FROM bikini FOR album=5
COPY TO temp TYPE DELIMITED
MODIFY FILE temp.txt
USE
DELETE FILE backup.dbf
DELETE FILE temp.txt
```

Műveletek az adatbázis struktúrájával

Ha új adatbázis állományt kell létrehozni, amelynek a struktúrája azonos vagy hasonlít az aktív állomány struktúrájára, akkor a

COPY STRUCTURE TO <f1> **[FIELDS** <mezők>**]**

parancs segítségével lehet létrehozni új üres állományt (f1), amelynek a mezői azonosak az aktív állomány mezőivel vagy a mezői a
FIELDS <mezők> paraméter által lesznek kiválasztva.
Leggyakrabban az adatbázis-állományt CREATE paranccsal szokták létrehozni, amely interaktív módban működik. Előfordulhat, hogy a program futtatása alatt is szükséges új állományt létrehozni. E célból két parancsot kell alkalmazni. A

COPY TO <f1> STRUCTURE EXTENDED

parancs az aktív állomány struktúráját átírja a <f1> állományba, (a <f1> rekordjaiban az aktív állomány mezőjének paraméterei tárolódnak).
<f1> négy mezőt tartalmaz:
FIELD_NAME (a mező neve),
FIELD_TYPE (a mező típusa),
FIELD_LEN (a mező hossza),
FIELD_DEC (a numerikus mező tizedes számjegyeinek száma).
Szükség esetén az <f1> állomány módosítható és ezzel az állomány struktúrája megváltozható. Az új állományt az <f1> állomány alapján a

CREATE <f> FROM <f1>

paranccsal lehet létrehozni.
<f> - az új állomány neve.

Példa

```
CLEAR
USE adatok
COPY TO temp STRUCTURE EXTENDED
USE temp
WAIT WINDOW 'az adatok.dbf felépítése' NOWAIT
BROWSE
CREATE backup FROM temp
USE backup
DISPLAY STRUCTURE
```

Állományok összeolvasztása

Az aktív állományt egy másik állománnyal össze lehet olvasztani. Ezt a műveletet a
JOIN WITH <munkaterület> **TO** <f1> **[FIELDS** <mezőlista>**] FOR**<feltétel>**]**

parancs hajtja végre.
Az új állomány neve – f1 és a mezőit a **FIELDS** <mezőlista> adja.
WITH <munkaterület> - a nem aktív állomány munkaterülete.

Példa

```
USE kader IN a
SELECT b
USE brig
JOIN WITH a TO kader2 FOR num=kader.num FIELDS a.nev, num, ora
```

Aktív állomány módosítása egy másik állomány alapján

UPDATE ON <kulcs-mező> **FROM** <munkaterület> **REPLACE** <mező1>
WITH <kifejezés1>, <mező2> **WITH** <kifejezés2>, ... **[RANDOM]**

Az UPDATE parancs lehetőséget ad az aktív adatbázis állomány mezőinek módosítására egy másik állomány alapján (amelynek a munkaterületét kell megadni). Az aktív és a módosító állományok rekordjai között *'egy-egy'* vagy *'egy-több'* reláció lehet. Az aktív állománynak csak azok a rekordjai lesznek módosítva, amelyeknek a kulcsmező értéke egyenlő a másik állomány kulcsmező értékével. Az aktív állomány mezői a WITH paraméterben megadott kifejezések értékeit kapják. A kifejezések a másik állomány adatait tartalmazhatják. A parancs végrehajtása előtt az állományokat a <kulcs-mező> szerint ajánlatos rendezni (indexelni). Ha az aktív állomány nincs rendezve, akkor a RANDOM paramétert kell megadni.

Példa
Minden tranzakció (banki művelet) a *BankNap.DBF* állományban tárolódik. A bevételek a *Be* nevű mezőben, a kiadások a *Ki* nevű mezőben vannak elhelyezve. Ezek az adatok alapján a *Bank.DBF* állomány mindegyik *nev* mezőre nézve változik a megfelelő *Összeg* mező értéke:

```
USE BankNap
INDEX ON nev TO banknev
SELECT b
```

```
USE Bank
UPDATE ON nev FROM a REPLACE Osszeg WITH Osszeg+ ;
        BankNap.Be – BankNap.Ki
```

Rekordok feltételes számlálása

COUNT [<hatókör>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>] [**TO** <változó>]

parancs megállapítja az aktív állomány azok rekordjainak számát, amelyek megfelelnek a feltételnek.
A <hatókör> alapértelmezése – ALL,
FOR <feltétel1> - amelyet a parancs mindegyik rekordban ellenőriz,
WHILE <feltétel2> - a számlálás addig folytatódik, amíg a feltétel igaz,
TO <változó> - az eredmény (a rekordok száma) a <változóba> kerül.

Példa

```
USE emberek
COUNT ALL FOR irszam=3300 TO egriek
WAIT WINDOW 'Egri előfizetők: '+ LTRIM(STR(egriek))
```

Numerikus kifejezések összegezése és átlaga

SUM [<kifejezés-lista>] [<hatókör>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>]
[**TO** <változólista> / **TO ARRAY** <tömb>]

<kifejezés-lista> - az aktív adatbázis-állomány numerikus típusú mezői vagy kifejezések, amelyeknek az összegét a parancs kiszámítja. Ha nincs megadva a <kifejezés-lista>, akkor az aktív állomány összes numerikus mezői részt vesznek a műveletben. Csak azok rekordok vesznek részt a műveletben, amelyek megfelelnek a **FOR** <feltétel1>-nek. Az összegezés addig tart, amíg a **WHILE** <feltétel2> igaz. A <hatókör> alapértelmezése – ALL. Az eredmény a <változólistába> vagy a <tömbbe> lesz elhelyezve. Ha a tömb még nem létezett, akkor a parancs létrehozza. Ha a tömb mérete eltér a szükséges mérettől, akkor a tömb megkapja a szükséges méretet.

AVERAGE [<kifejezés-lista>] [<hatókör>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>]
[**TO** <változólista> / **TO ARRAY** <tömb>]

AVEGAGE parancs paraméterei megegyeznek a **SUM** parancs paramétereivel. Az eredmény – a kifejezések átlaga.

Példa

```
USE varos
COUNT FOR neme='F' TO x
SUM Gyerek TO gyerekek
AVERAGE FOR neme='N' Gyerek TO y
```

Statisztikai számítások

CALCULATE [<kifejezés-lista>] [<hatókör>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>]
[**TO** <változólista> / **TO ARRAY** <tömb>]

A **CALCULATE** parancs felépítése a SUM parancs struktúrájára hasonlít. A <kifejezés-lista> pénzügyi és statisztikai függvényeket tartalmazhat:
AVG (.) – átlag (számtani közép),
CNT (.) – a számításban résztvevő rekordok száma,
MAX (.) – a maximum (a karakteres és dátum típussal is alkalmazható),
MIN (.) – a minimum (a karakteres és dátum típussal is alkalmazható),
STD (.)– a szórás kiszámítása,
SUM (.) – összeg,
VAR (.) – szórásnégyzet számítása,
NPV (N1, N2 [, N3]) – pénzügyi függvény.

Példa

```
USE szamla
CALCULATE AVG(bevetel), MIN(bevetel), MAX(bevetel)
CALCULATE STD(bevetel), VAR(bevetel) TO mstd, mvar
? mstd, mvar
```

A mezők kulcsérték szerinti összegezése

TOTAL ON <kifejezés> [<hatókör>] [**FOR** <feltétel1>] [**WHILE** <feltétel2>] **FIELDS**
<mezőlista>] [**TO** <új állomány>]

Az aktív állomány a <kifejezés> szerint van indexelve. A TOTAL parancs úgy számítja ki a numerikus mezők összegeit, hogy az összegezésekben csak azok a rekordok vesznek részt, amelyekben a <kifejezés> értékei azonosak. Az <új állományba> a <kifejezés> mindegyik lehetséges értékének a megfelelő összege kerül.

Példa

```
USE Kader
INDEX ON Osztag TO Oszt
TOTAL ON Osztag TO SumFiz FIELDS Fizetes
* A SumFiz.DBF állomány osztagonként tartalmazza a fizetéseknek az
*összegét.
```

Műveletek memo- és general-mezőkkel

A FoxPro olyan parancsokat is tartalmaz, amelyek a műveleteket kizárólag a memo-mezőkkel hajtják végre.

APPEND MEMO <memo-mező> **FROM** <f> [**OVERWRITE**]

Az <f> állományból az adatok átmásolódnak a <memo-mezőbe>. Az OVERWRITE paraméter törli a memo-mező előző értékét.

70. Példa

```
SET SAFETY OFF
USE dolgozok
GO 1
COPY MEMO memmezo TO test.txt
MODIFY FILE test.txt
APPEND MEMO memmezo FROM test.txt OVERWRITE
```

CLOSE ALL

COPY MEMO <memo-mező> **TO** <f> **[ADDITIVE]**

A parancs a memo-mező tartalmát a *f.txt* állományba másolja át. ADDITIVE paraméter esetén a másolás a létező állomány végén történik. Ha az ADDITIVE paraméter nincs megadva, akkor a <f> állomány felülíródik.

MODIFY MEMO m1 [, m2,] **[NOEDIT] [NOWAIT] [RANGE** n1, n2]
[SAVE] [WINDOW w]

APPEND GENERAL <general mező> **FROM** <állomány>

parancs az állomány tartalmát a *generál* típusú mezőbe helyezi el. Ha a general-mező már tartalmaz OLE-objektumot, akkor a régi tartalom megsemmisül. Az állomány teljes nevét kötelező megadni.

Példa
APPEND GENERAL tabla FROM "C:\EXCEL\CHART1.XLS" CLASS EXCELCHART
A *TABLA* nevű general-mezőbe egy EXCEL tábla lesz importálva.

Rendszerváltozók és a rendszer konfigurálása

A FoxPro rendszerváltozókat is tartalmaz, amelyeknek a neve _ karakterrel kezdődik. A program futtatása alatt ezek a változók automatikusan megkapják a megfelelő értékeket. A változokat a programban használhatjuk.
Például, a _TALLY rendszerváltozó azoknak a rekordoknak a számát tartalmazza, amelyek részt vettek a következő parancsokban: APPEND FORM, AVERAGE, CALCULATE, COPY TO, COUNT, DELETE, INDEX, JOIN, PACK, REINDEX, REPLACE, SELECT (SQL), SORT, SUM, TOTAL, UPDATE.
A FoxPro a program futtatása előtt a _TALLY változónak 0 értéket ad.
A rendszer konfigurálása a környezet paramétereit tartalmazza, amelyek befolyásolják a FoxPro program futtatását. Külső és belső konfigurálás létezik.
A **külső konfigurálást** a CONFIG.FPW konfigurációs állomány tartalmazza. A CONFIG.FPW - szöveges állomány, amelyet a programozó módosíthat. A CONFIG.FPW mindegyik sora tartalmazza a paraméter nevét és értékét.
Pl. SORTWORK paraméter azt a meghajtó/katalógust adja, amelyben tárolódnak az ideiglenes állományok a rendezési és indexelési műveleteknél,
TEDIT – tartalmazza a külső szövegszerkesztő nevét. Ha ezt a paramétert megadjuk, akkor a MODIFY COMMAND parancs nem a beépített szövegszerkesztőt fogja alkalmazni,
WP – tartalmazza a külső szövegszerkesztő nevét a memo- mezők módosítására.
A CONFIG.FPW állomány SET parancsokat is tartalmazhat.
Pl. DELETED = ON.
A **belső konfigurálás** a FOXUSER.DBF és FOXUSER.FPT állományok által történik és tartalmazzák a belső szövegszerkesztő beállításait, a BROWSE-ablakok paramétereit, színek beállításait stb.
A CONFIG.FPW egyes paramétereinek az alapértelmezése a következő:

ALTERNATE = OFF
ANSI = OFF
AUTOSAVE = OFF

BELL = ON
BLINK = ON
BLOCKSIZE = 64
BORDER = SINGLE
CARRY = OFF
CENTURY = OFF
CLEAR = ON
CLOCK = OFF
CONFIRM = OFF
CONSOLE = ON
CURSOR = ON
DATE = AMERICAN
DECIMALS = 2
DELETED = OFF
DEVICE = SCREEN
ECHO = OFF
ESCAPE = OFF
EXACT = OFF
EXCLUSIVE = ON
FULLPATH = ON
HEADING = ON
HOURS = 12
INTENSITY = ON
MARGIN = 0
MARK = " / "
MEMOWIDTH = 50
MOUSE = 5
MOUSE = ON
NEAR = OFF
OPTIMIZE = ON
PRINT = OFF
SAFETY = ON
SCOREBOARD = OFF
SPACE = ON
STATUS = OFF
STEP = OFF
SYSMENU = ON
TALK = ON
TYPEAHEAD = 20
UNIQUE = OFF

FoxPro függvényei

Matematikai függvények

Függvény	Művelet (eredmény)
N1=**ABS**(N)	N abszolút értéke
L=**BETWEEN**(V, Vmin, Vmax)	L= .T., ha Vmin <=V <=Vmax
N1=**CEILING**(N)	A legközelebbi egész szám, N1>=N
N1=**FLOOR**(N)	A legközelebbi egész szám, N1<=N
N1=**INT**(N)	N egész része

V=**MAX**(V1, V2,...)	Maximum
V=**MIN**(V1, V2,...)	Minimum
N=**MOD**(N1, N2)	N1/N2 egész osztás maradéka
N=**ROUND**(N1, N2)	N1 kerekítése N2 tizedes számjegyig.
N1=RAND	valós típusú véletlen szám előállítása, 0<=N1<1
N1=**RAND**(N)	egész típusú véletlen szám előállítása, 0<=N1<N
N1=**SIGN**(N)	Előjelfüggvény (sign): N1=1, ha N>0, N1=0, ha N=0, N1=-1, ha N<0
N1=**EXP**(N)	e alapú exponenciális függvény
N1=**LOG**(N)	Természetes alapú logaritmus
N1=**LOG10**(N)	Tízes alapú logaritmus
N1=**SQRT**(N)	Négyzetgyökvonás
N1=**SIN**(N)	Szinusz (N radián)
N1=**COS**(N)	Cosinus (N radián)
N1=**TAN**(N)	Tangens (N radián)
N1=**ASIN**(N)	Arcus sinus
N1=**ATAN**(N)	Arcus tangens
N1=**ATN2**(N1, N2)	Arcus tangens (N1/N2)
N=**PI()**	π
N1=**DTOR**(N)	Fokokban adott szögérték radiánba konvertálása
N1=**RTOD**(N)	Radiánban adott érték szögre konvertálása

Karakteres függvények

L=C1 **$** C2	L=.T., ha C2 tartalmazza a C1-et
N1=**AT**(C1,C2,[N])	C1 karaktersorozat helyének keresése a C2 karaktersorozatban. Ha N adott, akkor az N-dik előfordulását keresi. Alapértelmezés: 1. Visszaadott érték: a C2 a C1-ben hányadik pozícióban kezdődik, vagy – 0, ha nem fordul elő.
N1=**ATC**(C1,C2,[N])	Az AT függvényhez hasonlóan működik, de nagy- és kisbetűk között nem tesz különbséget.
N1=**RAT**(C1,C2,[N])	Az AT függvényhez hasonlóan működik, de a C2-ben jobboldalról kezdi a C1 keresését
L=**INLIST**(V,V1,V2,...)	A V=V1 **OR** V=V2 **OR** ... műveletet végzi. *Megjegyzés:* A ! INLIST(V,V1,V2,...) - a V#V1 AND V#V2 AND .... műveletet hajtja végre.
N=**LEN**(C)	C hossza
L=**LIKE**(C1,C2)	C1-maszk szerint keres a C2-ben. A C1 '*' és '?' helyettesítő karaktereket is tartalmazhat. Ha C2 a C1 maszkot tartalmazza, akkor L=.T., másként-L=.F.. Kis és nagy betűket nem különbözteti meg.
N=**OCCURS**(C1,C2)	C1 előfordulása száma a C2-ben.
C1=**LEFT**(C,N)	C bal oldali részéből N karaktert választ ki
C1=**RIGTH**(C,N)	C jobb oldali részéből N karaktert választ ki
C1=**SUBSTR**(C,N1,[N2])	C karaktersorozatból N1-dik karaktertől kezdve N2 karaktert másol át a C1-be. Ha N2 nincs megadva, akkor a másolás a C utolsó karakteréig történik.
C1=**LTRIM**(C)	A bal oldali szóközök törlése

C1=**RTRIM/TRIM**(C)	A jobb oldali szóközök törlése
C1=**ALLTRIM**(C)	A bal és jobb oldali szóközök törlése
C1=**REPLICATE**(C,N)	C karaktersorozat ismétlése N-szer
C=**SPACE**(N)	C feltöltése N szóközzel
C=**STUFF**(C1,N1N2,C2)	C1 karaktersorozatban karakterek törlése vagy beszúrása. N1- kezdőpozíció, N2 – karaktert kell törölni (ha N2=0, akkor csak beszúrást végez), C2 – a beszúrandó karaktersorozat.
C=**PADR**(C1,N,[C2])	Az eredmény hossza N, C tartalmazza a C1-et és jobb oldaláról C2 karakterekkel lesz kiegészítve
C=**PADL**(C1,N,[C2])	Az eredmény hossza N, C tartalmazza a C1-et és bal oldaláról C2 karakterekkel lesz kiegészítve.
C=**PADC**(C1,N,[C2])	Az eredmény hossza N, C a C1-et középen tartalmazza, jobb és bal oldalról kiegészíti C2 karakterekkel.
N=**ATLINE**(C1,M)	M-ben C1-et keres. Annak a sornak a sorszámát adja vissza, amelyik tartalmazza a C1-et. Ha nem talált, akkor N=0.
N=**ATCLINE**(C1,M)	Hasonlít az ATLINE-hoz, nem különbözteti meg a nagy- és kisbetűket.
N=**RATLINE**(C1,M)	Keresi a C1 utolsó előfordulását az M-ben és annak a sornak a sorszámát adja vissza, amely a C1-et tartalmazza. Ha nem talált, akkor N=0.
C=**MLINE**(M,N)	Visszaadja a memo-mező N-edik sorának a tartalmát
L=**ISALPHA**(C)	Ha a C első karaktere betű, akkor L=.T., másként L=.F.
L=**ISLOWER**(C)	Ha C első karaktere kis betű, akkor L=.T., másként L=.F.
L=**ISUPPER**(C)	Ha C nagybetűvel kezdődik, akkor L=.T., másként L=.F.
C1=**LOWER**(C)	C sorozat betűit kisbetűkre konvertálja
C1=**PROPER**(C)	C sorozatból nagy kezdőbetűs szavakat alakít (a szó nagybetűvel kezdődik, a többi betű – kicsi).
C1=**UPPER**(C)	C karaktersorozat nagybetűre konvertálása

Dátum és idő függvények

D=**CTOD**(C)	Karakteres dátumformát dátumtípusra konvertálja
D=(C)	Lásd – CTOD függvényt, C – csak konstans lehet
D=**DATE()**	A rendszerdátum lekérdezése
N=**DAY**(D)	A dátumból a nap értékét adja vissza (1.. 31).
N=**DOW**(D)	D dátum héten belüli nap sorszáma (1.. 7), (vasárnap-1)
C=**DTOC**(D)	Dátum konvertálása karakterré
C=**DTOS**(D)	A dátum karakterekbe konvertálása az *éééehhnn* formátumban (év, hónap, nap)
D=**GOMONTH**(D,N)	A dátumhoz N hónapot ad hozzá
C=**CMONTH (D)**	A hónap szöveges kiírása angolul
N=**MONTH**(D)	A dátumból a hónap sorszám
N=**YEAR**(D)	Dátumból az év értékét adja vissza (numerikus

	formátumban)

Adatkonverzió

N=**ASC**(C)	A C karakter ASCII kódja
C=**CHR**(N)	Az ASCII kódnak megfelelő karakter
C=**STR**(N1,[N2],[N3])	N1 numerikus kifejezést karakterré alakítja. N2- az eredmény hossza, N3- a tizedes számjegyek száma
N=**VAL**(C)	Karakter formátumú számot numerikus típusra konvertál.

Állomány műveletek

L=**BOF**([mt])	ha a rekordmutató kilép az állományból az első rekordon keresztül, akkor L=.T., másként, L=.F.
C=**DBF**([mt])	Az állomány teljes neve
N=**DISKCPACE()**	A lemezen lévő szabad memória mérete (byte-ban)
L=**DELETED**([mt])	Az aktuális rekord logikai törlésének állapota. Ha a rekord törlésre van jelölve, akkor L=.T.
L=**EOF**([mt])	Ha a rekordmutató kilép az állományból az utolsó rekordon keresztül, akkor L=.T., másként, L=.F.
L=**FILE**(F)	Ha az F állomány létezik, akkor L=.T.
C=**FIELD**(N,[mt])	N sorszámú mező neve
L=**FOUND**([mt])	L=True, ha a keresési parancs (LOCATE, CONTINUE, SEEK) rekordot talált
N=**FCOUNT**([mt])	Az állomány mezőinek száma
C=**FILTER**([mt])	Szűrő-kifejezés tartalma
Mező1=**LOOKUP** (mező1, V, mező2)	Az adatbázis mező2-ben a V kifejezés első előfordulását keresi. Ha a keresés sikeres volt, akkor a mező1 értékét adja vissza.
D=**LUPDATE**([mt])	Az adatbázis állomány utolsó módosításának időpontja
C=**ORDER**([mt])	A főindex neve
N=**RECNO**([mt])	Az aktív rekord sorszáma
N=**RECCOUNT**([mt])	Az adatbázis állomány rekordjainak száma
N=**RECSIZE**([mt])	A rekord hossza (byte)
L=**SEEK**(V,[mt])	Ellenőrzi a keresés eredményességét. V – a kulcs értéke. Ha a keresés sikeresen ért véget, akkor L=.T.
L=**UPDATE**()	Ha a READ parancs vezérlése alatt egy GET – mező volt módosítva, akkor L=.T.

Egyéb függvények

N=**COL**()	A kurzor oszloppozíciója
N=**ROW**()	A kurzor sorpozíciója
N=**PCOL**()	A nyomtatófej aktuális oszloppozíciója
N=**PROW**()	A nyomtatófej aktuális sorpozíciója
N=**MCOL**([ablak])	Az egér helyzete (oszlop)
N=**MROW**([ablak])	Az egér helyzete (sor)
L=**MDOWN**()	Ha az egér bal oldali billentyűje van lenyomva, akkor L=.T.
N1=**INKEY**([N]) /**LASTKEY**( )	Egy karakter kivitele a billentyűzet pufferből (ASCII-kód). A függvény a billentyű lenyomására N másodpercig vár. Ha N=0, akkor addig vár, amíg egy billentyű nem lesz lenyomva. Ha N nincs megadva, akkor nem vár és az utolsó lenyomott billentyű kódját adja vissza.
READKEY()	visszaad egy kódot, amely annak a billentyűnek felel meg, amely befejezte a READ parancsot. Egy billentyűnek két kód felelhet meg. Az ASCII - ha az adatok a parancsban nem voltak módosítva, a másik, amely 256-tal nagyobb, ha az adatokat megváltoztattuk. (Pl. Esc - 12 vagy 268).
CAPSLOCK([L])	akkor bekapcsolja a CAPSLOCK Ha L=.T. és kikapcsolja.
N=**MEMORY**()	A rendelkezésre álló szabad memóriaterületet adja vissza (Kb-ban).
C=**TIME**( )	Rendszeridő.
V=**IIF**(L,V1,V2)	ha L=.T., akkor V=V1, másként, V=V2.
L=**EMPTY**(V)	Ha a V kifejezés üres, akkor L=.T., másként L=.F..
N=**SECONDS**( )	Rendszeridő másodpercekben (ezredmásodperc pontossággal).
C1=**TYPE**(C)	C változó típusnak C1 lehetséges értékei - {C, L, N, M, D, U}, U – definiálatlan típusú kifejezés.