MC
110.451

CLIPPER

kézikönyv

Molnár Lajos
Molnárné
Nagy Anikó

Molnár Lajos
Molnárné Nagy Anikó

CLIPPER KÉZIKÖNYV

Novotrade Rt.

Lektorálta: **BORS GYÖRGY**

Szerkesztette: **GYÖRKE TIBORNÉ**

Második, javított kiadás

MC 110.451

KONYV-TÁR

1990

R 2

Copyright © Molnár Lajos, Molnárné Nagy Anikó, 1989

ISBN 963 585 045 X Törölve

A kiadásért felel RÉNYI GÁBOR, a Novotrade Rt. vezérigazgatója
Budapest, 1990

Műszaki szerkesztő: Horváth Ferencné

Szedte: TÁRKI

Készült a Somogy Megyei Nyomdaipari Vállalat kaposvári üzemében
Megjelent 17,5 (A/5 ív terjedelemben
Felelős vezető: Mike Ferenc igazgató

ISBN 963 585 104 9

Bevezetés

Az IBM PC-n Magyarországon alkalmazott relációs-adatbázis-kezelők közül a Clipper a legrugalmasabb, ez nyújtja a legnagyobb szabadságot a programozónak. A legutóbbi, a '87 nyári változat képes a dBASE adatbázisok (.dbf) és indexállományok (.ndx) kezelésére. Előnyös tulajdonságai miatt a programozók igazán sokoldalú felhasználóbarát rendszereket készíthetnek ezen a nyelven.

Ennek a könyvnek nem célja a Clipper nyelv megtanítása. Egy részletes, jól áttekinthető referencia-kézikönyv ez, amelyben a Clipper minden létező parancsa és standard függvénye megtalálható. Van olyan utasítás, amelyet még az eredeti Nantucket kézikönyvben sem említenek, csak a fordítóprogram visszafejtése után derült fény létezésére, így kerülhetett bele a könyvbe is. Minden parancsnak és függvénynek kipróbált, pontos a szintaxisa.

A Clipper nyelv lehetővé teszi C és assembly nyelvű rutinok beillesztését a programokba. A könyv válaszol a probléma megoldására, megtudható, hogy az idegen nyelvű modulok írásának milyen módjai vannak, és milyen segítséget nyújt ehhez a Clipper. A technikai adatokból megismerhető többek között az egyes adattípusok ábrázolásmódja az adatbázisban, valamint a Clipper által használt adatbázisok és indexállományok fizikai szerkezete is.

Sok sikert és kevés problémát kívánunk minden Clipper programozónak!

Molnár Lajos
Molnárné Nagy Anikó

A Clipper és a Plink86 a Nantucket Corporation,
a Norton Guides és a Norton Commander a Peter Norton Computing,
a dBASE az Ashton-Tate,
a FoxBase a Fox Software,
a Kedit a Mansfield Software Group,
a Sidekick, Turbo-C és a TLINK a Borland Inc,
az MS-DOS pedig a Microsoft Corporation védjegye.

A parancsok és a funkciók leírásakor használt jelölések

A könyvben használt jelölések a hagyományokon alapszanak, új jelölést nem alkalmaztunk.

[ ] A szögletes zárójelek a parancsok szabadon választható részét fogják közre.
< > A csúcsos zárójelek közé írt adatot felhasználónak kell megadnia.
^ A Ctrl billentyű jele.
... A pontok helyére bármilyen Clipper kifejezés írható.
| A listából egyet (de csak egyet) kötelező választani.
'87 Csak a '87 nyári változattól alkalmazható.

Néhány tanács a programíráshoz

Az első és legfontosabb tanács: sohase tartsa fölöslegesnek a programok megtervezését, még akkor sem, ha látszólag csak egy nyúlfarknyi programról van szó. Tapasztalatunk szerint Jackson programtervezési módszere jól alkalmazható Clipper programokra is.

Gyakori az „ömlesztett" programlista. A tagolatlan program áttekinthetetlen, nagyon nehéz megtalálni benne az esetleges hibás részeket. A strukturált lista nemcsak szebb, hanem sokkal könnyebben is kezelhető. Már a program begépelésekor tartsa szem előtt a lista formátumát! A sorokat nem érdemes utólag rendezni, hiszen sokkal fáradságosabb, és gyakran el is marad.

A legjobb módszer a következő: ha elkezd begépelni egy ciklust vagy egy feltételt, akkor a nyitó utasítás után rögtön írja oda a záró utasítást is. A sorok beszúrása egy jó szövegszerkesztővel semmilyen problémát nem okozhat. Ez a módszer megóv minket attól a hibától, hogy több egymásba ágyazott ciklus vagy feltétel esetén kimaradnak az ezeket lezáró parancsok. A belső utasításokat néhány karakterrel beljebb kezdve áttekinthetőbbé válik a programstruktúra. Mindig nagy- vagy kisbetűvel írja a programokat! Ha keverve használja a betűtípust, csúnya lesz a lista, és nem alkalmazható az a javaslat sem, hogy a program módosításakor az újonnan begépelt sorokat mindig az ellenkező betűtípussal írja. A fordítót nem zavarja ez a kettősség, viszont könnyebb megtalálni a javított részeket. A fejlesztés és a módosítás befejezése után írja át ezeket a részeket az eredeti betűtípusra. Ezt a műveletet jobb szövegszerkesztők egyetlen utasítással megoldják. A Clipper utasításait négy karakterig lehet rövidíteni. (Lehet, de nem érdemes.) A listát más is kézbe veheti, számára – ha nem szokta meg a rövidítéseket, vagy éppen most próbálkozik ezen a nyelven programot írni – értelmetlen zagyvaság lesz az egész.

A szövegszerkesztő

Kis programokban, vagy akkor, ha várhatóan sokszor kell majd módosítani a programot, használjon gyorsan előhívható szövegszerkesztőt. Ilyenek a tárrezidens programok, pl. a Sidekick vagy a Norton Commander editorai. Legfőbb előnyük a gyors előhívhatóság (betöltésre nincs szükség, mindig rendelkezésre állnak). Hátrányuk, hogy csak korlátozott méretű programokat írhat velük. Nagyobb programokban, programrendszerekben érdemes egy jobb szövegszerkesztőt – pl. a Keditet – használni. A Ke-

dit utasításkészletét megismerve a saját igényeinek megfelelő rendszer állítható össze. Ez a szövegszerkesztő sokszor segített már bonyolult feladatok megoldásában. Egy 130-140 programból álló rendszerben pl. három függvény nevét kellett átírni. Egy jól sikerült Kedit utasítássorral negyedóra alatt megoldódott a feladat. Természetesen a Kedit csak ajánlás. Mindenkinek van egy olyan szövegszerkesztője, amire esküszik, amit szeret.

A Clipper technikai leírása

A Clipper 2046 memóriaváltozót tud kezelni. Egy tömb egy változónak felel meg. A Clipper tömb legfeljebb 4096 elemet tartalmazhat, az elemek típusa akár keveredhet is!

Mezők, változók

A Clipper 1 billió rekordot kezel egy állományban, amelyek csak fix hosszúságú rekordok lehetnek!

A mező-, változó- és tömbnevek megengedett hossza 10 karakter.

Karakteres	mező:	32 kbájt;
	változó:	64 kbájt.
Numerikus	mező:	19 bájt, ebből legfeljebb 15 lehet a tizedesek száma, karakteresen tárolja az adatot előjellel és tizedesponttal együtt;
	változó:	20 karakter, ebből legfeljebb 15 lehet a tizedesek száma, a számítások pontossága 18 jegyű.
Dátum:		mindig 8 bájtot foglal el, az adatot éééehhnn formában tárolja.
Logikai:		mindig 1 bájtot foglal el, értéke 'T' *igaz*; 'F' *hamis*.
Memo:		az adatbázisban egy 10 bájtos mutatót (pointert) foglal el, hossza max. 64 kbájt (.dbt állományban).

Állományok

A Clipper 255 állományt tud nyitva tartani, de a DOS a 20. állomány megnyitásakor hibát jelez. Ez az MS/PC-DOS3.30-ban már kiküszöbölhető a < SET CLIPPER = F255 > DOS változó megadásával. Az MS-DOS saját céljára foglal le öt állományt, ezért a számok a valóságban ennyivel kevesebbek.

Az MS-DOS 3.30 alatt a Clipper új lehetősége: egy állomány egyszerre több munkaterületen is megnyitható, akár mindegyik EXCLUSIV módban! Ha nem adott meg egyéni ALIAS nevet, akkor az eredeti fájlnévvel csak az utoljára megnyitott munkaterületét tudja kiválasztani.

Az adatbázisrekord hossza és az egy rekordon belüli mezők száma a gép memóriájától függ, a Clipper nem korlátozza.

Ez az adatbázis-szerkezet max. 2046 mezőt engedélyez.

Az indexkifejezés max. hossza 250 bájt.

Az indexkulcs max. hossza 1000 bájt, és nem lehet benne típuskeveredés.

Példa:

INDEX ON NEV+CIM Az indexkifejezés 7 karakter hosszúságú, az indexkulcs a NEV hossza és a CIM hossza.

A max. 250 állományon belül egyszerre egy katalógus (.cat) és egy áttekintő (.vue) állomány, ill. 250 adatbázis tartható nyitva.

Minden adatbázishoz egy címke- (.lbl), egy memo- (.dbt) egy lista- (.frm), egy szűrő- (.qry), és elvben korlátlan számú indexállomány tartozik, ezek természetesen csökkentik a nyitva tartható adatbázisok számát.

A Clipper a procedúra- és a képernyőformátum-állományokat lefordítja, és a programhoz szerkeszti. A képernyőformátum-állományokat (.fmt) vagy át kell nevezni programállománynak (.prg), vagy a hivatkozásban meg kell adni a teljes nevet (SET FORMAT).

A Clipper adatbázisainak szerkezete

Az adatbázisok (.dbf állományok) szerkezete egyetlen bájt különbséggel megegyezik a dBASE III Plus állományok szerkezetével. Az állományok tartalmazzák a mezők neveit, típusát és a hosszát is.

Az adatrekordok előtt az adatbázismezők számánál eggyel több 32 bájtos leíró rekord van, ezeket a Clipper programból nem érheti el közvetlenül. Az első 32 bájtos rekord az egész állomány szerkezetét mutatja, a többi adatbázismezők adatait tartalmazza. Ezek után következnek a tényleges adatrekordok.

A rekordok szerkezete

Fájlleíró rekord:

- azonosító (1 bájt)
  alaphelyzetben 03h és 83h, ha az állományhoz memofájl tartozik;
- az utolsó módosítás dátuma (3 bájt)
  év (évszázad nélkül) 1 bájt
  hó 1 bájt
  nap 1 bájt;

- rekordszám (4 bájt);
- adatterület kezdő ofszetje az állományban (2 bájt);
- rekordhossz (2 bájt);
- nem használt (20 bájt).

Mezőleíró rekordok numerikus, logikai, memo- és dátumtípusokban:

- név (10 bájt);
- nem használt (1 bájt);
- típus (1 bájt) (karakteresen, pl. L = logikai);
- nem használt (4 bájt);
- hossz (1 bájt);
- tizedesek (1 bájt);
- nem használt (22 bájt).

Mezőleíró rekordok karakteres típusokban:

- név (10 bájt);
- nem használt (1 bájt);
- típus (1 bájt) (C);
- nem használt (4 bájt);
- hossz (2 bájt).
- nem használt (22 bájt).

A mezőleíró rekordok után a 0Dh, 00h bájtsorozat következik. (A 00h bájt nem található meg a dBASE állományok szerkezetében!) Ez után következnek az adatrekordok. Minden rekordhoz egy 1 bájtosjelző tartozik. Ha a jelzőbájt „space , akkor a rekord érvényes, ha „*, akkor a rekord törlésre kijelölt. Ez a jelzőbájt a rekord legelső bájtja. A fájl végén egy 1Ah kódú fájlvégejel van.

A Clipper .ndx állományainak szerkezete

A Clipper is képes d*B*ASE szerkezetű indexállományokat használni, ezek .ndx kiterjesztésűek, szerkezetük a következő:

Az állomány 512 bájtos rekordokból (lapokból) áll. Az első Clipper, a többin pedig növekvő sorrendben a kulcsokat.

Az állományleíró rekord felépítése:

- azonosító (1 bájt);
- nem használt (3 bájt);
- az indexállomány hossza lapokban (4 bájt);
- nem használt (4 bájt);
- a kulcs logikai hossza (2 bájt);
- a kulcsok kezdőcíme a lapokon (4 bájt);
- a kulcs fizikai hossza (2 bájt);
- nem használt (4 bájt);
- a kulcskifejezés karakteresen (488 bájt).

A kulcsokat tartalmazó rekordok szerkezete:

- a kulcskifejezések száma a lapon (4 bájt);
- kulcsmező (n bájt).

Minden fizikai kulcs a következőképpen épül fel:

- a kulcshoz tartozó adatbázisrekord fizikai sorszáma (8 bájt).
- kulcsérték (x bájt).

A Clipper .ntx állományainak szerkezete

A Clipper saját – a dBASE-nél gyorsabb indexelést eredményező – indexállományai .ntx kiterjesztésűek. Az állomány 1024 bájtos rekordokból (lapokból) áll. Az első lapon az indexállományra vonatkozó információkat tárolja a Clipper, a továbbiakon pedig a kulcsértékeket. Az indexállományok szerkezete a következő:

Az állományleíró rekord felépítése:

- azonosító (1 bájt);
- nem használt (11 bájt);
- a kulcs fizikai hossza (2 bájt);
- a kulcs logikai hossza (2 bájt);
- nem használt (4 bájt);
- a kulcsokat tartalmazó lapok száma (2 bájt);
- a kulcskifejezés karakteresen (1002 bájt).

A kulcsokat tartalmazó rekordok szerkezete:

- hány kulcsot tartalmaz ez a lap (2 bájt);
- a kulcsok kezdő ofszetjei ezen a lapon, minden cím egy 2 bájtos érték (n bájt);
- kulcskifejezések (x bájt).

Minden kulcskifejezés két részből áll:

- a kulcshoz tartozó adatbázisrekord fizikai sorszáma (8 bájt),
- kulcsérték (y bájt).

A Clipper bővítő moduljai

NDX.obj

A modult a programhoz szerkesztve nem a Clipper .ntx típusú indexállományait használja, hanem a dBASE III Plus .ndx állományaival kompatibilis állományokat. Eddig nem volt lehetőség arra, hogy a Clipper a dBASE eltérő felépítésű indexállományaival dolgozzon; így viszont megválaszthatja, hogy egy kész program milyen típusú indexeket tartalmazzon.

PCBIOS.obj

A Clipper alapértelmezésben direkt képernyőkezelésű. Ha olyan gépen szeretné futtatni a megírt programot, amely nem teljesen IBM kompatibilis, csupán a BIOS szintjén, akkor ezt a modult kell a programhoz szerkeszteni.

ANSI.obj

A Clipper alapértelmezésben direkt képernyőkezelésű. Ha olyan gépen szeretné futtatni a megírt programot, amely nem teljesen IBM kompatibilis, csupán az ANSI szabvány szerinti terminálkezelést támogatja, akkor ezt a modult kell a programhoz szerkeszteni.

IBMANSI.obj

A Clipper alapértelmezésben direkt képernyőkezelésű. Ha olyan gépen szeretné futtatni a megírt programot, amely nem teljesen IBM kompatibilis, csupán olyan szabványos ANSI képernyőkezelést nyújt, amely ismeri az IBM bővítéseket (színek, funkcióbillentyűk), akkor ezt a modult kell a programhoz szerkeszteni.

DEBUG.obj

Ez a modul tartalmazza a Clipper nyomkövető rendszerét.

A Clipper operátorai

Aritmetrika

**	hatványozás
^	hatványozás
-	előjelváltozás
*	szorzás
/	osztás
+	összeadás
-	kivonás
%	az osztás maradéka (moduló)
=	értékadás

Sztringkezelő műveletek

'...'	a karakterlánc megadása
"..."	a karakterlánc megadása
[...]	a karakterlánc megadása
+	összefűzés
–	összefűzés (a köztes szóközöket az eredmény végére teszi)
$	részkarakterlánc keresése (mit $ miben)

Relációk

>	nagyobb
<	kisebb
> =	nagyobb egyenlő, nem kisebb
< =	kisebb egyenlő, nem nagyobb
< >	nem egyenlő
#	nem egyenlő
!=	nem egyenlő
=	egyenlő
= =	egyenlő (karakterláncban teljes egyezés!)

Logikai operátorok

.NOT	negáció (tagadás)
!	negáció (tagadás)
.AND	logikai ÉS
.OR.	logikai VAGY
->	átirányító operátor

Példa
? MEZŐ
Az aktuális adatbázis MEZŐ adatszavát írja ki.
? FAJL->MEZŐ
A FAJL alias nevű adatbázis MEZŐ adatszavát írja ki.
M->MEZŐ

Nem adatbázis adatszót, hanem egy MEZŐ nevű memóriaváltozót fog kiírni.

A Clipper érvényességi köre

RECORD <n>
Az n. rekord.
NEXT <n>
A következő n rekord, az aktuálist is beleértve.
ALL
A fájlban levő összes rekord.
REST
Az aktuálistól a fájl végéig az összes rekord.
FOR <feltétel>
A paraméter az utasítást azokon a rekordokon hajtja végre, amelyekre a *feltétel igaz.* Az érvényességi kör alapértelmezése az összes rekord (ALL).
WHILE <feltétel>
A paraméter az utasítást addig hajtra végre, amíg a *feltétel igaz.* Az első olyan rekordnál kilép, ahol nem teljesül a *feltétel.* Az érvényességi kör alapértelmezése a hátralévő összes rekord (REST).
Ha a két paramétert együtt használjuk, akkor a WHILE <feltétel> a nagyobb prioritású. A parancs akkor fejeződik be, ha a WHILE *nem teljesül.*
Alias-> (<kifejezés>)
A kifejezést a megadott adatbázis-területen értékeli ki.

Példa

USE ALLOMANY ALIAS TORZS
IF TORZS->(EOF( ))

A Clipper fordítóprogramjával felismerhető kulcsszavak listája

Elsődleges kulcsszavak

ACCEPT	EJECT	LABEL	REPORT
APPEND	ELSE	LIST	RESTORE
AVERAGE	ELSEIF	LOCATE	RETURN
BEGIN	END	LOOP	RUN
BREAK	ENDCASE	MENU	SAVE
CALL	ENDDO	NEXT	SEEK
CANCEL	ENDIF	NOTE	SELECT
CASE	ENDTEXT	ON	SET
CLEAR	ERASE	OTHERWISE	SKIP
CLOSE	EXIT	PACK	SORT
COMMIT	EXTERNAL	PARAMETERS	STORE
CONTINUE	FIND	PRIVATE	SUM
COPY	FOR	PROCEDURE	TEXT
COUNT	FUNCTION	PUBLIC	TOTAL
CREATE	GO	QUIT	TYPE
DECLARE	GOTO	READ	UNLOCK
DELETE	IF	RECALL	UPDATE
DIR	INDEX	REINDEX	USE
DISPLAY	INPUT	RELEASE	WAIT
DO	JOIN	RENAME	ZAP
EDIT	KEYBOARD	REPLACE	

Másodlagos kulcsszavak

ADDITIVE	CENTURY	DEVICE	FROM
ALIAS	CLEAR	DOUBLE	FUNCTION
ALL	COLOR	ECHO	GERMAN
ALTERNATE	COLOUR	ERROR	GET
AMERICAN	COMMAND	ESCAPE	GETS
ANSI	CONFIRM	EXACT	HEADING
BEFORE	CONSOLE	EXCEPT	HELP
BELL	CURSOR	EXCLUSIVE	INDEXES
BLANK	DARABASES	EXTENDED	INTENSITY
BOTTOM	DATE	FIELDS	INTO
BOX	DEBUG	FILE	ITALIAN
BRITISH	DECIMALS	FILTER	KEY
CARRY	DEFAULT	FIXED	LABEL
CASE	DELETED	FOR	LIKE
CENTER	DELIMITED	FORMAT	MARGIN
CENTRE	DELIMITERS	FRENCH	MEMORY

MESSAGE	PROCEDURE	SAVE	SUMMARY
NEXT	PROMPT	SAY	TALK
NOEJECT	RANDOM	SCOREBOARD	TO
OFF	RANGE	SCREEN	TOP
ON	RECORD	SDF	TYPEAHEAD
ORDER	RELATION	SELECTION	UNIQUE
PARAMETERS	REPLACE	SEQUENCE	VALID
PATH	REPORT	SOFTSEEK	WHILE
PICTURE	REST	STATUS	WITH
PLAIN	SAFETY	STEP	WRAP
PRINTER	SAMPLE	STRUCTURE	

Függvények

Ezekben a függvényekben a fordító ellenőrzi a paramétereket!

ABS	EXP	LOWER	ROW
ALIAS	FCOUNT	LTRIM	RTRIM
ASC	FIELDNAME	MAX	SECONDS
AT	FILE	MEMORY	SELECT
BOF	FLOCK	MIN	SPACE
CDOW	FOUND	MONTH	SQRT
CHR	IF	NETNAME	STR
CMONTH	IIF	PCOL	SUBSTR
COL	INDEXKEY	PCOUNT	TIME
CTOD	INKEY	PROCLINE	TRANSFORM
DATE	INT	PROCNAME	TRIM
DAY	ISCOLOR	PROW	TYPE
DELETED	ISCOLOUR	READVAR	UPDATED
DOW	LASTKEY	RECCOUNT	UPPER
DTOC	LASTREC	RECNO	USED
DTOS	LEN	REPLICATE	VAL
EMPTY	LOCK	RLOCK	WORD
EOF	LOG	ROUND	YEAR

A fordító

A fordítóprogram – Clipper.exe – a forrásnyelvű programokból egy átmeneti (ún. object) állományt hoz létre. Ez már nem egy forrásnyelvű, de még nem is futtatható gépi kódú program. Az átmeneti állományokból a szerkesztőprogram (linker) hozza létre a végső, futtatható .exe programot. A megadott fájlon kívül azokat a programokat is lefordítja, amelyekre közvetlenül hivatkozott, minden procedúraállományt (SET PROCEDURE), minden meghívott programfájlt (DO) és minden formátumállományt (SET FORMAT). Lehetőség van arra is, hogy felsorolja mindazon programfájlokat,

amelyeket le akar fordítani. Ilyenkor egy .clp kiterjesztésű állományban kell megadni a neveket, minden sorban csak egyet. Ha ilyen listaállományt használ, akkor a fordítási opciók minden állományra vonatkoznak. Nincs viszont lehetőség arra, hogy az így felsorolt állományokat más-más opcióval fordítsa le. Ilyenkor használjon annyi listafájlt, ahány fordítási opciót meg akar adni.

A fordító a következő paramétereket fogadja el:

- a fordítani kívánt fájl neve (.prg) (az .obj modul neve a megadott név lesz);
- @ + a fordítani kívánt fájlok neveit tartalmazó listafájl (.clp) (az .obj modul neve a listafájl neve lesz).
  Az alapértelmezés a -m opció.
  Pl. A>Clipper @lista -q
  (lista.obj !!!)

A következő opciókat csak kisbetűvel fogadja el a Clipper:

-l A sorszámokat nem tárolja a debugger számára. A program soronként 3 bájttal rövidebb lesz!

-m Csak a megadott modulban szereplő rutinokat fordítja le, a többit külső hivatkozásként kell definiálni, ezért azokat külön lehet fordítani.

-o A lefordított programot (.obj) a megadott lemezre, ill. könyvtárba helyezi el.
Pl. A>clipper program -oc:\sajat

-p A fordító betöltése után vár a forrásprogramo(ka)t tartalmazó floppyra.

-q Gyors fordítás. Nem írja ki a lefordított sor számát, csak a végeredményt.

-s Nem készít .obj modult, csak ellenőrzi a program helyességét.

-t A fordítás közben használt munkaállományt az itt megadott meghajtón hozza létre.
Pl. C>clipper program /td:

Ahol a D lemez lehet egy memóriadiszk is, így gyorsabb lesz a fordítás.

-v Az összes változónevet kiegészíti M-> jellel. Kivételek azok a változók, amelyeknél adatbázis-hivatkozást használt.

A fordító hibaüzenetei

A Clipper fordítóprogramjának hibaüzeneteit négy nagy csoportba sorolhatjuk. Az első a fordító paramétereire vonatkozik, a második a fordítandó program fizikai hibáit jelzi, a harmadik a fordítás során felfedezett hibákat, a negyedik pedig a Clipper belső hibaüzeneteit.

Paraméter hibák:

*** Invalid option:	Hibás opciót használt.
*** Invalid parameter:	Hibás paramétert használt.

A fordítandó program fizikai hibái:

can't open input	Nem tudta megnyitni a forrásprogramot.
can't open output	Nem tudta megnyitni az átmeneti állományt.
can't position input	Nem tudta használni a forrásprogramot.
cannot create temporary file	Fordítás közben nem tudott létrehozni egy segédállományt.
cannot open script	Nem tudta megnyitni a listaállományt.
cannot open, assumed external	Fordítás közben olyan modulra talált hivatkozást, amelyet nem tudott megnyitni, külső modulként jelölte be; külön lefordítva a szerkesztőprogram hozzáfűzheti a programhoz.
FATAL at xxxx	Fizikai hibát talált valamelyik állományban.
write error	írás közben hibát talált, valószínűleg az objectmodulban.

A fordítás során felfedezett hibák:

A fordító a hibás sort a standard outputra írja ki, ^ jellel jelölve azt a pontot, ahol a hibát feltételezi. A standard output alapértelmezése a képernyő, de a kiírás átirányítható egy lemezállományba is, ha az utasítássorban nagyobbjel (>) után megadja a fájlnevet.

A következő üzenetek az adott utasításban hibásan használt szintaxisok eredményei:

@ error
ACCEPT/INPUT error
APPEND error
AVERAGE error
CALL error
CLEAR error
CLOSE error
COPY error
COUNT error
CREATE error
DELETE error
DO error
FIND error
FOR error
IF () error
INDEX error
JOIN error
LABEL error
LIST error
LOCATE error
PARAMETER error
RECALL error
RELATION error
RELEASE error
REPLACE error
REPORT error
RESTORE error
SAVE error
SEEK error
SET error
SORT error
STORE error
SUM error
TOTAL error
UPDATE error
USE error
WAIT error

Ezek az üzenetek minden utasításban előfordulhatnak:

ASSIGNMENT error	Fölösleges paramétert adott meg egy függvényben.
bad command args	Hibás paraméterezés.

comma error	Hiányzó vagy fölösleges vessző van a szövegben.
dafault thing	A fordító szerint hiányzik valami.
ENDCASE expected	Hiányzik az ENDCASE.
ENDDO expected	Hiányzik az ENDDO.
ENDIF expected	Hiányzik az ENDIF.
expression expected	A rendszer kifejezést várt.
identifier expected	A rendszer azonosítót várt.
illegal bracket	Felesleges zárójel.
illegal device	Hibás egységmegadás.
illegal symbol mode	Hibás jelölésmód.
incorrect number of arguments	Hibás a paraméterek száma.
internal expr error	Belső kifejezéshiba.
invalid field	Nem létező terület.
invalid mode in lex	Hibás szóhasználat.
invalid procedure mode	Hibás procedúra-használat.
invalid variable	Hibás változónév.
missing 2nd quote	A záró aposztrófot is meg kell adni.
missing DO WHILE or FOR	EXIT utasítást használ cikluson kívül.
missing SEQUENCE	END utasítást használ cikluson kívül.
NEXT expected	Hiányzik a NEXT.
no return value	A függvény végén nem ad vissza értéket.
operand expected	A rendszer operandust várt.
phase error	Fordítás közbeni, nem szintaktikai hiba.
procedure exceeds max size	A procedúra túllépte a lehetséges legnagyobb méretet, több rutinra, függvényre kell szétdarabolni, és újra le kell fordítani.
Redefinition of predefined function	Már meglévő funkció nevét adta egy saját függvénynek.
rest of line ignored	A ^ utáni sorrészt a fordító figyelmen kívül hagyja.
segment exceeds max size	A program valamelyik szegmense túllépte a megengedett legnagyobb értéket, valószínűleg túl hosszú adatterületet próbált létrehozni.
SET not recognized	A megadott SET paramétert a rendszer nem ismeri.
structure error	Szerkezeti hiba.
symbol redefinition error	Egy nevet már másra használt fel (pl. megegyezik egy programfájl és egy procedúra neve).
symbol table exceeds max size	Több nevet nem tud már tárolni, túl sok a procedúra, ill. a függvény. Megoldása egyszerű: a programokat .clp állományokba kell csoportosítani, és úgy lefordítani.
too many external references	A rendszer túl sok külső hivatkozást talált.

too many fixups	Túl sok a konstans.
too many labels	Túl sok a változó. Két megoldás lehetséges: vagy kevesebb változót, vagy több tömböt használjon.
too many procs	Túl sok a procedúra, procedúrák helyett használjon függvényeket.
too many public declarations	Túl sok a PUBLIC változó. Két megoldás lehetséges: vagy kevesebb PUBLIC változót, vagy több tömböt használjon.
too many segments	Túl sok darabban fordította a programot.
too many symbols	Túl sok szimbólumot használ.
unbalanced brackets	Hiányzó zárójelek.
unbalanced parenthesis	Hiányzik a zárójel.
unexpected END	Nem várt END.
unexpected EOF	Nem várt fájlvége.
unrecognized dcode	Fel nem ismert kód.
variable expected	A rendszer változónevet várt.
verb not recognized	A rendszer ezt a kulcsszót nem használja, de felismeri.

A Clipper belső hibái:

floating-point error:	Lebegőpontos számolási hibák.
denormal	Nem normalizált alakú szám.
divide by 0	Osztás nullával.
explicitly generated	Függvényérték létrehozva.
inexact	Pontatlan eredmény.
invalid	Hibás eredmény.
overflow	Túlcsordulás.
square root	Négyzetgyök.
stack overflow	Veremtúlcsordulás.
stack underflow	Verem-alulcsordulás.
underflow	Alulcsordulás.
unemulated	Kiszámítatlan érték.
integer divide by 0	Egész osztása nullával.
not enough space for environment	Nincs elég memória a futási környezet létrehozásához.
run-time error	Futás közbeni hiba.
floating point not loaded	Megsérült a Clipper program.
null pointer asignment	Nem valós eredmény.
memory fault	Memóriahiba.
Packed file is corrupt	Megsérült Clipper program.

A szerkesztőprogram (linker)

A Clipper rendszerhez a gyártó a Plink86 szerkesztőprogramot adja, amely létre tudja hozni azt az overlay szerkezetet, amelyet a Clipper használ. Ezt semmilyen más szerkesztőprogram nem tudja. Bevált módszer, hogy a felhasználók a Borland cég TLINK szerkesztőprogramjával fűzik össze a programokat. Tapasztalatunk szerint csak az 1.x verzió használható a Clipperhez. A Plink86 hátránya a lassúság. A DOS Link programja kétszer-háromszor gyorsabban dolgozik. Ha nincs szükség overlay ágakra, használjon inkább más szerkesztőprogramot! A Plink86 parancsait megadhatja a DOS parancssorból vagy interaktív üzemmódban is. Ezek mellett az előkészített utasítássort egy .lnk kiterjesztésű állományban is rögzítheti. Ilyenkor a programot a következő módon kell indítani:

Plink86 @<név>.lnk

A Plink86 vezérlőutasításai

#	Megjegyzés.
;	Az utasítások lezáró karaktere.
BATCH	Ha a szerkesztő nem talál egy .obj vagy egy .lib állományt, akkor egy prompt után bekéri az elérési útvonalat, ill. a helyes nevet (a Plink86 e funkcióját lehet így letiltani.)
BEGINAREA	Az overlay terület kezdetének jelzése. Használható helyette a BEGIN is.
DEBUG	Hibakeresési lehetőséget nyújt (az éppen betöltött overlay ág nevét kiírja a képernyőre).
DEFINE	Belső változó megadása.
ENDAREA	Az overlay terület végének jelzése. Használható helyette az END is.
FILE	Az objectmodul nevének megadása (ha több van, akkor vesszővel kell elválasztani őket).
HEIGHT	A laphossz megadása sorokban a memóriatérkép számára. Az alapértelmezés 65.
LIBRARY	A Clipper könyvtári rutinjait tartalmazó .lib állomány megadása (ha ez csak Clipper.lib, akkor nem kell megadni).
LOWERCASE	Minden változónév kisbetűsre konvertálása.

MAP Memóriatérkép készítése a programról.
Használata:

MAP = fájlnév [A] [,G] [,M] [,S]

- A minden szegmenst felsorol, a kezdőcím szerinti rendezettségben (alapértelmezés);
- G a PUBLIC változók listáját is előkészíti;
- M részletes információt ad a modulokról a kezdőcím és a hossz felsorolásával;
- S minden programrészt kilistáz a névsorban.

NOBELL A rendszer ne sípoljon minden Plink86 üzenetnél.

OUTPUT A kész .exe program nevének megadása (ha semmit nem ad meg, akkor a név az első FILE utasításban megadott .obj modul neve lesz).

OVERLAY CODE Az overlay területeket leíró rész kezdősorának megadása.

SEARCH A saját felhasználói könyvtár megadása (.lib).

SECTION A felsorolt .obj modulok egy belső overlay szerkezetet alkotnak.

SECTION INTO A megadott .ovl modul a felsorolt .obj modulokból áll.

UPPERCASE Minden változónév nagybetűsre konvertálása.

VERBOSE A Plink86 működése közben a felhasználó nyomon követheti, mivel foglalkozik a szerkesztő.

WIDTH A memóriatérkép számára a sorhossz megadása karakterben. Az alapértelmezés 80.

WORKFILE A Plink86 munkaállományának átirányítása egy másik lemezre.

Példa (.lnk):

```
BATCH FILE elso, debug
SEARCH SAJAT
DEBUG
OVERLAY CODE
BEGINAREA
    SECTION INTO OVL1 FILE LIST1, LIST2
    SECTION INTO OVL2 FILE MODO1, MODO2
ENDAREA
;
```

A Plink86 figyelmeztető üzenetei (Warning)

A szerkesztés folytatódik, a hibákat a belső hibakezelő rutinok javítják.

1 Hibás címzés.
2 Nics kijelölt veremterület.
3 A szegmenscsoport túllépte a 64 kbájtot. (Ez a leggyakrabban akkor fordul elő, ha assembly modult használ a programhoz; ilyenkor célszerű átírni a modulban használt szegmensneveket).
4 A modul nevét nem találta az inputfájlban.
5 Hibás címzés.
6 Csak a CP/M-86 operációs rendszerben használt üzenet.
7 Nem object formátumú állomány.
8 Hibás az objectfájl ellenőrző száma.
9 Hibás az objectfájl rekordhossza.
10 Hibás címzés.
11 Többszörös PUBLIC definíció ugyanarra a változóra.
12 Egy adatszegmenst több csoporthoz rendelt.
13 Egy szegmenst többször, különböző típusúra definiált.
14 Több veremszegmenst definiált.
15 Nem használt.
16 Nem használt.
17 Nem használt.
18 Nem használt.
19 Nincs elég memória egy állomány megkereséséhez.
20 Nem egyezik a számított és a tényleges modulhossz.
21 Az első menetben feljegyzett változók száma nem egyezik a második menetben talált változók számával.

A Plink86 hibaüzenetei (Error)

A szerkesztő megáll, a hibákat a felhasználónak kell kijavítani.

1 Túl sok .lnk állományt használ, csak három szint engedélyezett.
2 Lemezhiba az .lnk állomány olvasása közben.
3 Nem találja az .lnk állományt.
4 Nem használt.
5 Az inputsor túl hosszú (max. 64 karakter lehet).
6 Illegális számjegyet talált; alapértelmezésben a hexadecimális, egyébként a decimális számrendszert kell használni a címkékben.
7 Nem használt.
8 Nem használt.
9 Nem használt.
10 A DOS számára illegális karakter van egy fájlnévben.
11 A rendszer utasítást várt a sorban.
12 Nem használt.
13 Nem használt.

14 A rendszer változónevet várt a sorban.
15 A rendszer egyenlőségjelet várt.
16 A rendszer 16 bites egész értéket várt.
17 Nincs FILE utasítás, legalább egynek azonban lennie kell.
18 A jobb oldali határolójelet a rendszer nem találja.
19 Nem használt.
20 Nem használt.
21 Nem használt.
22 Nem használt.
23 Nem használt.
24 Nem használt.
25 Nem használt.
26 Nem használt.
27 Nem használt.
28 Nem használt.
29 Nem használt.
30 A munkaállományt nem tudta létrehozni.
31 Fizikai hiba a munkállomány írása közben.
32 Fizikai hiba a munkállomány olvasása közben.
33 Fizikai hiba a munkállomány pozicionálása közben.
34 Túl sok szimbólumot használt (max. 35000 lehet).
35 Nem használt.
36 Nem használt.
37 Nem használt.
38 Nem használt.
39 Nem használt.
40 Nem használt.
41 Hiányzik az objectmodul vége.
42 Fizikai hiba az objectmodul olvasása közben.
43 Nem találja a megadott objectmodult vagy a modulkönyvtárat.
44 Nem használt.
45 Nem tudta létrehozni az outputállományt.
46 Hibás az outputállomány típusa (.exe vagy .cmd lehet).
47 Fizikai hiba az outputállomány írása közben.
48 Fizikai hiba az outputállomány olvasása közben.
49 Nem tudta lezárni az outputállományt.
50 Nem tudta létrehozni a memóriatérképet leíró állományt.
51 Nem definiált szimbólumnevet talált; azt az objectmodult, amelyben a procedúrát vagy függvényt definiálta, valószínűleg nem szerkesztette hozzá a programhoz.
52 Nem használt.
53 Nem használt.
54 Nincs elég memória a Plink86 futásához (legalább 256 kbájt szabad memóriára van szükség).
55 Nem használt.
56 Nem használt.

57	Nem használt.
58	Túl hosszú a veremszegmens.
59	Nem használt.
60	Nem használt.
61	Hibás az objectmodul felépítése (LTL szegmenst tartalmaz); valószínűleg nem Clipper kompatibilis fordítót használt valamelyik C, ill. assembly nyelvű modulhoz. Megfelelő objectmodult a MASM 5.00, ill. a Microsoft C 5.00 kompatibilis fordítók állítanak elő.
62	Hibás az objectmodul felépítése.
63	Hibás az objectmodul felépítése.
64	Hibás a modulkönyvtár felépítése.
65	Hibás az objectmodul felépítése.
66	Hibás az objectmodul felépítése.
67	Nem használt.
68	Nem használt.
69	Nem használt.
70	Hibás az objectmodul felépítése.
71	Hibás az objectmodul felépítése.
72	Nem használt.
73	Hibás az objectmodul felépítése.
74	Nem használt.
75	Nem használt.
76	Nem használt.
77	Nem használt.
78	Nem használt.
79	Nem használt.
80	Túl sok overlay területet határozott meg.
81	Több ENDAREA utasítás van, mint BEGINAREA.
82	Több BEGINAREA utasítás van, mint ENDAREA.
83	A kész program túllépné az 1 Mbájtos memóriaterületet, használjon overlay területeket.
84 – 199	Nem használt.
200 – 226	A Plink86 belső hibájára utaló, nem publikált hibaüzenetek. Próbálja meg az eredeti lemezről újra átmásolni, ha nem sikerül, akkor forduljon ahhoz, akitől a program származik.

A Clipper hibakezelő rendszere

A Clipper futásidejű, helyreállítható hibái a következő csoportokba sorolhatók:

- adatbázis;
- kifejezés;
- keveredés;
- nyomtató;
- nyitás;
- nem definiált;
- egyéb.

Minden csoportnak egy hibakezelő funkciója van. A funkciók a saját csoportjukban bekövetkező hiba esetén kapják meg a vezérlést. A felhasználó a hibakezelést saját programjából végezheti. A hibakezelő funkciót az Errorsys.prg programban célszerű definiálni. A lefordított rutinokat azután a saját programhoz kell szerkeszteni, így az eredeti függvények helyett a saját függvények kapják meg a vezérlést. Csak arra kell vigyázni, hogy szerkesztéskor az Errorsys.obj vagy a Debug.obj ne előzzön meg saját modulokat!

Három paramétert mindig megkap a funkció. Az első a hívó program neve, a második a hívó program sorszáma, a harmadik pedig egy információs paraméter. Az információs paraméter szöveges formában utal a hiba jellegére.

A funkció a hiba jellegétől függően még négy paramétert kaphat. A negyedik paraméter a hibát „modellezi". (A Clipper forrásnyelvű szövegként adja vissza a hibás sort!) Az ötödik, a hatodik és a hetedik paraméter egy hibás kifejezés minden egyes operandusát tartalmazza. Az ötödik a bal oldalit, a hatodik a jobb oldalit, a hetedik az eredményt. A negyedik paraméter ezekkel a változókkal „modellezi" a hibát.

A 12/0 pl. hibát okoz. A három paraméter legyen _1, _2, _3 (szabályos változónevek). A modellparaméter '_1 / _2' lesz. A _1 = 12, a _2 = 0 a _3 pedig 0 lesz.

Adatbázishibák

Ezek a hibák az adatbázist használó utasítások végrehajtása közben keletkeznek.

Db_error (<procedúranév>, <sorszám>, <info>)

Az *<info>* paraméter öt értéket vehet fel:

- Database required (nincs nyitott adatbázis);
- Lock required (a művelethez zárolni kell az állományt);
- Exlusive required (a művelethez az állományt EXLUSIVE módban kell megnyitni);
- Field numeric overflow (REPLACE utasítás esetén a numerikus mezőben nem fér el a megadott szám);
- Index file corrupted (a rendszer hibás indexállományt talált, amely valószínűelg nem a megadott adatbázishoz tartozik).

Ha a hibakezelő funkció *igaz* értéket ad vissza, akkor a Clipper megkísérli újból végrehajtani a műveletet; ha *hamis*-at, akkor a Clipper automatikusan egy QUIT utasítást hajt végre, és visszatér a DOS-ba, ha pedig egy BREAK utasítást ad ki, akkor a végrehajtás az aktuális BEGIN SEQUENCE végén folytatódik.

Kifejezéshibák

Ezek a hibák a kifejezések kiértékelése közben keletkeznek.

Expr_error (<procedúranév>,<sorszám>,<info>, <modell> _1, _2, _3)
Az *<info>* paraméter négy értéket vehet fel:
- Type mismatch (típuskeveredés);
- Subscript range (nem kijelezhető érték, vagy a maximálisnál nagyobb tömbelemre hivatkozik);
- Zero divide (nullával próbált osztani);
- Expression error (kifejezéshiba makrókifejtés közben).

A *<modell>* paraméter a hibát okozó utasítás modelljét tartalmazza. A paraméterek helyett az _1, _2, _3 változónevek használhatók. (Ezeket a hibakezelő függvény is megkapja.)

Hibás lesz pl. az

ELSO=1+2'

kifejezés. A *<modell>* paraméter a következő sztringet fogja tartalmazni: '_1+_2'. Ilyenkor egyszerű a hibajavítás, csak a hibás típusú paramétert kell megváltoztatni.

```
_2=VAL(_2)
RETURN &modell
```

Keveredéshibák

Ezek a hibák típuskeveredésből származnak.

Misc_error (<procedúranév>,<sorszám>,<info>,<modell>)

Az *<info>* paraméter a következő két üzenetet tartalmazhatja:

- Type mismatch (egy REPLACE utasításban nem megfelelő típusú kifejezést használt);
- RUN error (RUN utasítást használt, de nincs elég szabad memória a COMMAND.COM program elindításához. A hiba pontos okát a Doserror() függvénnyel kérdezheti le.

Az Expr_error() függvénnyel ellentétben a *<modell>* paramétert nem lehet makróként végrehajtani.

Ha a hibakezelő funkció *igaz* értéket ad vissza, vagy BREAK utasítást ad ki, akkor a Clipper az aktuális SEQUENCE-hez tartozó END utasításon folytatja a programfutást. Ha viszont *hamis* értéket ad vissza, akkor a program egy QUIT utasítást hajt végre.

Nyomtatóhibák

Ezek a hibák akkor keletkeznek, ha a nyomtatóegység nem „üzemkész".
Print_error (<procedúranév>,<sorszám>,<info>)

Az *<info>* paraméter csak egy értéket vehet fel:

- Printer error.

Ha a hiba elhárult – pl. bekapcsolta a nyomtatót vagy befűzte a papírt –, egy *igaz* érték visszaadásakor a program megismétli az írásműveletet, *hamis* érték visszaadásakor pedig a következő utasítást hajtja végre.

Nyitáshibák

Ezek a hibák állománynyitáskor keletkeznek, kivéve az alacsony szintű állománykezelő függvényeket.

Open_error (<procedúranév>,<sorszám>,<info>,<modell>,_1)

Az *<info>* paraméter csak egy értéket vehet fel:

- Open error.

Az _1 paraméter azt a fájlnevet tartalmazza, amelyet nem tudott megnyitni. A program *igaz* érték visszaadásakor újra megpróbálkozik a hibát eredményező utasítás végrehajtásával, *hamis* érték visszaadásakor pedig a következő utasításon folytatódik a programvégrehajtás.

„Nem definiált" hibák

Ezek a hibák akkor keletkeznek, ha olyan memóriaváltozóra hivatkozik, amelyet még nem definiált.

Undef_error(<procedúranév>,<sorszám>,<info>,<modell>,_1)

Az _1 paraméter tartalmazza azt az azonosítót, amelyet a rendszer nem talált.

Az *<info>* paraméter három értéket vehet fel:

- Undefined identifier (olyan változóra vagy adatbázismezőre hivatkozik, amely még nem létezik);
- Not an array (olyan tömbre hivatkozik, amelyik még nem létezik);
- Missing EXTERNAL (olyan függvényre vagy procedúrára hivatkozik, amelyet nem szerkesztett a programhoz; általában makrókifejezésben fordul elő).

Igaz érték visszaadásakor vagy a BREAK utasítás kiadásakor a program az aktuális SEQUENCE végére ugrik, *hamis* érték visszaadásakor pedig egy QUIT utasítást hajt végre.

Egyéb hibák

A következő hibákat nem lehet programból kezelni, mert minden esetben megszakítják a programot:

Internal error n

Hibás indexállományban keletkezik. A program egy billentyű lenyomása után véget ér. Az *n* értékei a következők:

0	Hibás a hibakezelő rendszer.
1	Aritmetikai túlcsordulás.
2	Memóriahiba.
3	Memóriahiba.
4	Memóriahiba.
5	Memóriahiba.
6	Állománypufferhiba.
7	Állománypufferhiba.
8	Állománypufferhiba.
9	Állománypufferhiba.
10	Túl sok egymásba ágyazott BEGIN SEQUENCE...END.
11	A BEGIN SEQUENCE...END parancs belső hibája.
12	Aritmetikai veremtúlcsordulás a BEGIN SEQUENCE...END után.
14	Hiba a SORT parancsban.
15	Hiba a SORT parancsban.
16	Az adatbázis nincs nyitva.
17	Hibás az .ntx állomány.
18	Hibás az .ntx állomány.
19	Hibás az .ntx állomány.
20	Hibás az .ndx állomány kulcskifejezése.
21	Hibás az .ndx állomány kulcskifejezése.
22	Hibás az .ntx állomány kulcskifejezése.
92	Hiba a SORT vagy az INDEX ON parancsban.

Disk Full

Az állományműveletek során megtelt a lemez vagy a főkönyvtári katalógus. Ha a képernyőn megjelenő kérdésre Y-t válaszol, akkor a Clipper megkísérli újra végrehajtani a lemezműveletet, N válaszra pedig kilép a programból.

Multiple Error

A hibakezelő rutinban van hiba! Egy billentyű lenyomása után kiléphet a programból.

Out of Memory

A megkezdett művelet végrehajtásához nincs elég szabad memóriaterület. A SET CLIPPER megváltoztatásával esetleg fel lehet szabadítani a szükséges memóriaterületet, vagy az előzőleg indított memóriarezidens programokat kell eltávolítani a memóriából. Egy billentyű lenyomása után kiléphet a programból.

Not Enough Memory

A művelet elkezdéséhez sincs elég szabad memóriaterület. A SET CLIPPER megváltoztatásával esetleg fel lehet szabadítani a szükséges memóriaterületet, vagy az előzőleg indított memóriarezidens programokat kell eltávolítani a memóriából. Egy billentyű lenyomása után kiléphet a programból.

Kapcsolat más programnyelvekkel (C, assembly függvények írása)

Előbb-utóbb előfordul, hogy a Clipper függvények nem elég gyorsak, nem elég hatékonyak. Ilyenkor más megoldást kell keresni. A leghasznosabb assembly nyelvű rutint írni. A Clipper '87 nyári változatát a Microsoft C 5.00-val írták, ezért a saját függvényeknek is e nyelv paraméterátadási procedúráját kell követniük. Szerencsés ötlet volt a fejlesztőktől, hogy egy illesztőmodult írtak, amely elsősorban a C nyelvű modulokat illeszti a Clipper rendszerhez. Az illesztőfüggvényeket természetesen assembly nyelvből is használhatjuk. Ezek a függvények lehetővé teszik a paraméterátadást, -átvételt a Clipper és a saját modul között. Egy-egy illesztőmodult felhasználói függvényként is használhat. Használat közben a következőket kell figyelembe venni:

C nyelv

A Clipper illesztőfüggvényeihez az EXTEND.h és a NANDEF.h include állományokat használhatja. Eredményül a Microsoft C 5.00-val készített objectmodullal megegyező objectmodult kell kapni. A lebegőpontos számításokat a Clipper ún. alternatív módszerrel használja. A saját funkciókat 'void pascal'-ként kell definiálni.

A következő fordítási opciókat kell megadni:
- Használjon large (nagy) memóriamodellt!
- Az objectmodulban ne szerepeljen az alapértelmezett run-time könyvtár!
- A belső funkcióhívások előtt és után a C ne ellenőrizze a belső vermet!

A Microsoft C 5.00 változathoz a következő fordítási opciókat javasoljuk:

C1 /c /AL /Z1 /Oalt /FPa /Gs <név>.c

/c	Csak fordítás szerkesztés nélkül.
/AL	Large memóriamodell.
/Z1	Nincs alapértelmezett könyvtár.
/Oalt	Optimalizált kódot állít elő:
	a elhagyja az alias ellenőrzést;
	1 ciklusokat optimalizál;
	t sebességre optimalizál;
/FPa	Használjon alternatív lebegőpontos aritmetikát!
/Gs	A veremellenőrzés tiltása.

A Microsoft MASM 5.00-val készített objectmodullal megegyező objectmodult kell kapni eredményül. A Clipper illesztőfüggvényeihez az EXTENDA.MAC és az EXTENDA.INC include állományokat használhatja.

Far típusúnak kell definiálni a saját funkciót és az összes illesztőrutint.

Turbo-C 2.0 változathoz a következő opciókat javasoljuk:

-ml	Large memória modell
-G	Sebességre optimalizál
-Ixxxx	Include (.h) állományok könyvtára
-Lyyyy	Könyvtár (.lib) állományok könyvtára
-zACODE	Kódszegmens neve: CODE
-zSDGROUP	Adatszegmens csoportneve: DGROUP
-c	Csak fordítás

Ezeket az opciókat a Turboc.cfg állományba kell beíni, így külön-külön sorba.

Az illesztőfüggvények

Ugyanazon függvényeket kell használni C és assembly nyelvből, csupán egy eltérés van.

A Microsoft C a függvényei elé automatikusan elhelyez egy aláhúzásjelet, így a függvények alkalmazásakor ezt el kell hagyni a név elől!

Az assembly nyelvben azt kell tudni, hogy az illesztőfüggvények a paramétereket a verembe kérik, és ott is hagyják azokat, a „rendrakás" tehát a programozó feladata. A rendszer csak az SP, BP, SS, CS és IP regisztereket tárolja automatikusan.

Paraméterként egy tömbnevet is megadhat, ilyenkor a függvények a tömb elemeit is elérik.

A leírásokban az a példa szerepel, amelyben tömbelemet vesz át a függvény. Ha nem tömbnevet használ paraméterként, akkor a tömbelem-hivatkozásokat teljes egészében el kell hagyni!

A használható függvények (rutinok):

__parinfo **_parinfo( )**

Paraméterteszt.
A megadott sorszámú paraméterről közöl információt.
A 0. paraméter a paraméterek száma.

Hívása:

```
ASM                     C
mov   ax, sorszám       int típus,sorszám;
push  ax                típus = _parinfo(sorszám);
call  __parinfo
add   sp,2
; az eredmény az AX-ben van
```

__parinfa **_parinfa( )**

Tömbelemteszt.
A megadott sorszámú elemről közöl információt.

A 0. tömbelem az elemek számát adja vissza.

Hívása:

```
ASM                     C
mov   ax, tömbelem      int típus,sorszám,tömbelem;
push  ax                típus = _parinfa(sorszám,tömbelem);
mov   ax, sorszám
push  ax
call  __parinfa
add   sp,4
; az eredmény az AX-ben van
```

__parc **_parc( )**

Karakterlánc átvétele a paramétersorból.

Egy olyan mutatót (pointert) ad vissza, amely a karakterláncra mutat (DX : AX). Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

Hívása:

```
ASM                     C
mov   ax,tömbelem       char *karakterek;
push  ax                int tömbelem,sorszám;
mov   ax,sorszám        karakterek = _parc(sorszám,tömbelem);
push  ax
call  __parc
add   sp,4
```

__pards **_pards( )**

Dátumsztring átvétele a paramétersorból.

Egy olyan mutatót (pointert) ad vissza, amely a karakterláncra mutat (DX : AX). Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

A dátumsztring minden esetben 'ééééHHNN' formátumú.

Hívása:

```
ASM                     C
mov   ax,tömbelem       char *dátum;
push  ax                int tömbelem,sorszám;
mov   ax,sorszám        dátum = _pards(sorszám,tömbelem);
push  ax
call  __pards
add   sp,4
```

__parl **_parl( )**

Logikai érték átvétele a paramétersorból.

Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

A 0 visszaadott érték *hamis* értéket jelent, egyéb érték – általában 1 – pedig *igaz* értéket.

Hívása:

ASM	C
mov ax,tömbelem	int van;
push ax	int tömbelem,sorszám;
mov ax,sorszám	van = _parl(sorszám,tömbelem);
push ax	
call __parl	
add sp,4	
; az eredmény az AX-ben van	

__parni **_parni( )**

Egész szám átvétele a paramétersorból.

Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelemek számát is!

Hívása:

ASM	C
mov ax,tömbelem	int szám;
push ax	int tömbelem,sorszám;
mov ax,sorszám	szám = _parni(sorszám,tömbelem);
push ax	
call __parni	
add sp,4	
; az eredmény az AX-ben van	

__parnl **_parnl( )**

Hosszú egész szám átvétele a paramétersorból.

Egy 32 bites egész számot ad vissza (DX : AX). Ha a paraméter több, akkor második paraméterként meg kell adni a tömbelem számát is!

Hívása:

ASM	C
mov ax,tömbelem	long szám;
push ax	int tömbelem,sorszám;
mov ax,sorszám	szám = _parnl(sorszám,tömbelem);
push ax	
call __parnl	
add sp,4	

__parnd **_parnd( )**

Dupla hosszúságú lebegőpontos szám átvétele a paramétersorból.

Egy olyan mutatót (pointert) ad vissza (DX : AX), amely a 8 bájtos számra mutat.

Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

Hívása:

```
ASM                     C
mov  ax,tömbelem        double szám;
push ax                 int tömbelem,sorszám;
mov  ax,sorszám         szám = _parnd(sorszám,tömbelem);
push ax
call __parnd
add  sp,4
```

__parclen **_parclen( )**

A paraméterként megadott sztring hossza a záró 0 bájt nélkül.

Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

Hívása:

```
ASM                     C
mov  ax,tömbelem        int,hossz;
push ax                 int tömbelem,sorszám;
mov  ax,sorszám         hossz = _parclen(sorszám,tömbelem);
push ax
call __parclen
add  sp,4
```

__parcsiz **_parcsiz( )**

A paraméterként megadott sztringgel lefoglalt memóriaterület hosszát adja vissza.

Ha a paraméter tömb, akkor második paraméterként meg kell adni a tömbelem számát is!

Hívása:

```
ASM                     C
mov  ax,tömbelem        int hossz;
push ax                 int tömbelem,sorszám;
mov  ax,sorszám         hossz = _parcsiz(sorszám,tömbelem);
push ax
```

```
call    __parcsiz
add     sp,4
```

__ret **_ret( )**

Clipper verem rendbetétele.

Nem ad vissza paramétert. Az e függvénnyel befejeződő rutinokat csak DO paranccsal szabad hívni, különben a függvény összekeveri a Clipper belső adatvermét!

Hívása:

```
ASM                 C
call    __ret       _ret( );
```

__retc **_retc( )**

Egy karakterláncot ad vissza a Clipper számára.

Minden függvény cak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatvermét!

Hívása:

```
ASM                 C
; a DS : SI         char *puffer;
; mutat a nullával  _retc(puffer);
; lezárt sztringre
push    ds
push    si
call    __retc
add     sp,4
```

__retds **_retds( )**

Dátum formátumú (ÉÉÉÉHHNN) karakterlánc átadása a Clippernek.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
ASM                 C
; a DS : SI         char *dátum;
; mutat a nullával  _retds(dátum);
; lezárt dátumra
push    ds
push    si
call    __retds
add     sp,4
```

__retl **_retl( )**

Logikai érték átadása a Clippernek.

Ha az érték 0, akkor *hamis*-nak, ellenkező esetben *igaz*-nak veszi a Clipper.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
ASM                 C
push  ax            int logért;
call  __retl        _retl(logért);
add   sp,2
```

__retni **_retni( )**

16 bites, előjeles, egész szám átadása a Clippernek.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
ASM                 C
push  ax            int szám;
call  __retni       _retni(szám);
add   sp,2
```

__retnl **_retnl( )**

32 bites, előjeles, egész szám átadása a Clippernek.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
ASM                 C
push  ax            long szám;
push  bx            _retnl(szám);
call  __retnl
add   sp,4
; BX : AX
; a szám
```

__retnd **_retnd( )**

64 bites, előjeles, lebegőpontos szám átadása a Clippernek.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
    ASM                 C
push  ax                double szám;
push  bx                _retnd(szám);
push  cx
push  dx
call  __retnd
add   sp,8
; DX : CX: BX : AX
; a szám, C
; lebegőpontos
; ábrázolási
; formában
```

__retclen **_retclen( )**

A megadott *sztringet* adja vissza a Clippernek, a megadott *hossz*-szal. Megadható az a kezdőpont, ahonnan a rutin elkezdheti a számolást.

Minden függvény csak egy értéket adhat vissza, különben összekeveredik a Clipper belső adatverme!

Hívása:

```
    ASM                 C
; AX - méret           char *puffer;
; a DS : SI            int hossz;
; mutat                _retclen(puffer,hossz);
; a sztringre
push  ax
push  ds
push  si
call  __retclen
add   sp,6
```

__exmgrab **_exmgrab( )**

Legfeljebb 64 kbájt memória lefoglalására használható.

A mutatót (pointert) a DX : AX regiszterek alkalmazzák. Ha a regiszterek értéke 0 (NULL), akkor a lefoglalás nem sikerült.

Hívása:

```
    ASM                 C
mov   ax,méret          unsigned char *terület;
push  ax                unsigned int méret;
call  __exmgrab         terület = _exmgrab(méret);
add   sp,2
```

__exmback **_exmback( )**

Az __exmgrab rutinnal lefoglalt memóriaterület szabadítható fel vele.
Fontos, hogy a paraméterek megegyezzenek az __exmgrab által használt értékekkel!

Hívása:

```
ASM                     C
mov   ax,méret          unsigned char *terület;
push  ax                unsigned int méret;
; a DS : SI pointer a   _exmback(terület,méret);
; lefoglalt memóriára
; mutat
push  ds
push  si
call  __exmback
add   sp,6
```

_strlen **strlen( )**

Az _strlen rutin egy 0 bájttal lezárt sztring hosszát adja meg.

Hívása:

```
Asm                     C
; a DS : SI pinter      unsigned char *puffer;
; a sztringre           unsigned int méret;
; mutat                 méret = strlen(puffer);
push  ds
push  si
call  _strlen
add   sp,4
; a hosszúság az AX-ben van
```

__dbfhead **_dbfhead( )**

Az aktuális munkaterületen lévő adatbázis leíró rekordjára mutató mutatót (pointert) adja vissza.
Figyelem! Nem szabad módosítani a területen, csak olvasni lehet onnan!

Azzal is segíti a C nyelven programozókat, hogy előre definiál egy adatbázis-leíró struktúrát, amelyet a _dbfhead( ) függvénnyel együtt érdemes használni.

```
typedef struct
{
        byte signatrure;
        byte ymd[3];
```

```
        long last_rec;
        quant data_off;
        quant rec_size;
        byte pad[20]<
{ DBF_HEADER;
```

Hívása:

```
ASM                 C
call  __dbfhead     DBF_HEADER *leíró;
; a DX : AX         leíró = _dbfhead( );
; a leíró;
; a területre
; mutat
```

__dbfopen **_dbfopen( )**

Ha az aktuális munkaterületen nincs megnyitva adatbázis, akkor 0 értéket ad vissza, ha van, akkor 1-et.

Hívása:

```
ASM                 C
call  __dbfopen     int van;
; az érték AX-ben   van = _dbfopen( );
; van
```

A Clipper támogatása a C nyelvű függvények használatához

A programozó munkáját segítő definíciók az EXTEND.h és a NANDEF.h állományban találhatók.

A NANDEF.h a következő hét definíciót tartalmazza:

```
#define   FALSE   0
#define   TRUE    1

#define   NIL           '\0'
#define   NULL    0L

typedef   unsigned char byte;
typedef   unsigned int quant;
typedef   int Boolean;
```

A többi definíció az EXTEND.h állományban van.

Elsőként a CLIPPER makró, amelyet a függvények definiálására használhat:

```
#define   CLIPPER  void pascal
```

Ezután a parinfo által visszaadott típusdefiníciók következnnek:

```
#define   UNDEF       0
#define   CHARACTER   1
#define   NUMERIC     2
#define   LOGICAL     4
#define   DATE        8
#define   ALIAS       16
#define   MPTR        32
#define   MEMO        65
#define   WORD        128
#define   ARRAY       512
```

Az MPTR az egyetlen olyan érték, amelyet nem közvetlenül használ. Mindig ak-

kor fordul elő, amikor egy paraméter nem érték, hanem egy változó kezdőcíme. Ilyenkor a típust jelző érték és az MPTR értéke egy kivonással szétválasztható.

Példa:

```
VÁLTOZÓ =DATE ( )
MÁSIK = 'SZIA'
SAJÁTF(123,VÁLTOZÓ,@MÁSIK)
```

Az első paraméter típusa 2 (NUMERIC), a másodiké 8 (DATE), az utolsóé pedig 33 (CHARACTER + MPTR) lesz.

Az állomány többi része az illesztőfüggvények definícióit tartalmazza.

A paraméterek típusellenőrzésére kilenc makrót definiál:

```
#define   PCOUNT        (_parinfo(0))
#define   ISCHAR (n)    (_parinfo(n)   &  CHARACTER)
#define   ISNUM (n)     (_parinfo(n)   &  NUMERIC)
#define   ISLOG (n)     (_parinfo(n)   &  LOGICAL)
#define   ISDATE (n)    (_parinfo(n)   &  DATE)
#define   ISMEMO (n)    (_parinfo(n)   &  MEMO)
#define   ISBYREF (n)   (_parinfo(n)   &  MPTR)
#define   ISARRAY (n)   (_parinfo(n)   &  ARRAY)
#define   ALENGTH (n)   (_parinfo(n, 0))
```

A Clipper támogatása az assembly nyelvű függvények használatához

A Clipper adatszegmenseinek neve általában _DATA. Ezt a nevet lehetőelg ne használja, ill. csak akkor, ha a Clipper belső adataival dolgozó függvényt ír. Az adatszegmensek a DGROUP csoportnéven érhetők el. Az illesztőfüggvények alkalmazásához az szükséges, hogy a DS regiszter erre az adatszegmenscsoportra mutasson. A Clipper kódszegmenseinek általában CODE a neve.

Az EXTEDNDA.mac és az EXTENDA.inc állományban találhatók azok a definíciók és makrók, amelyekkel assembly nyelvű funkciókat írhat a Clipperhez.

Az EXTENDA.mac állomány az előző Clipper változatokkal dolgozók számára lehet hasznos. Egyszerű definíciókat tartalmaz a paraméterek teszteléséhez, átvételéhez és visszaadásához. A függvény ezekkel a makrókkal nem használhat tömböket paraméterként.

Az INCLUDE állomány csak a következő rutinok definiálását végzi el:

_PARINFO _PARC _PARNI _PARNL _PARND _PARDS _PARL
_RETC _RETNI _RETNL _RETND _RETDS _RETL

A definiált makrók rövid leírása (az értékek csak 16 bitesek lehetnek):

GET_PCOUNT	az AX-be kerül az átadott paraméterek száma;
GET_PTYPE n	az n. paraméter típusa az AX-ben;
GET_CHAR n	a karakteres paraméter átvétele a DX:AX-be;
GET_INT n	egy 16 bites egész szám átvétele az AX-be;
GET_LONG n	egy 32 bites egész szám átvétele a DX:AX-be
GET_DBL n	egy lebegőpontos szám átvétele; az ES:SI a számra mutat, a DX:CX:BX:AX tartalmazza is;
GET_DATESTR n	egy logikai érték átvétele az AX-be;
RET_CHAR seg,offset	egy karakterlánc visszaadása a Clippernek;
RET_INT érték	egy 16 bites egész szám visszaadása a Clippernek;
RET_LONG	kisebb helyi érték, nagyobb helyi érték egy 32 bites szám visszaadása a Clippernek;
RET_DBL	négy helyi érték növekvő sorrendben egy lebegőpontos szám visszaadása a Clippernek;

RET_DATESTR seg,offset	egy dátumsztring visszaadása a Clippernek;
RET_LOGICAL érték	egy logikai érték visszaadása a Clippernek.

Az EXTENDA.inc állomány a '87-es Clipperhez, a MASM 5.00 változatához készült. Ez az állomány már fejlettebb makrókkal segíti a programírást, de csak a következő illesztőfüggvényeket definiálja:

__parinfo __parinfa __parc __parcsiz __parclen
__parni __parnl __parnd __parl __pards

__retc __retclen __retni __retnl
__retnd __retl __retds __ret

A következő makrók használhatók:

CLpublic < név1,név2, ... >

A < > jelek között felsorolt rutinneveket PUBLIC típusúra definiálja.

CLfunc típus név < paramétertípus paraméternév, ... >

A saját függvényeket definiálhatja vele, a *típus*, a *név* és a paraméterek megadásával. A további makrók az így definiált függvények paramétereinek ellenőrzését és átvételét automatikusan elvégzik, ellenőrzik a visszaadott érték típusát, és előkészítik a program végrehajtását is.

A megadható típusok:

void	nincs	
char	4 bájtos	(pointer)
int	2 bájtos	
long	4 bájtos	
double	8 bájtos	(lebegőpontos formában)
log	2 bájtos	
date	4 bájtos	

CLcode

Elkezdi a rutin „törzsét", és beolvassa a paramétereket. Ezután írható a saját funkció assembly nyelvű része.

CLret érték

Érték-et ad vissza a Clippernek.
A típusnak meg kell egyeznie a CLfunc makróval definiált típussal. Megadhat regisztert is, de annyit, hogy kiadják az aktuális típus hosszát.

PCOUNT

Az átadott paraméterek száma, a CLcode makró állítja be.

TESTNUL változó

Beállítja 1-re a Zéró bitet, ha a *változó* értéke 0.
Csak a Clipper makrókkal definiált változókat képes tesztelni!

CODESEG megnevezés

A használt kódszegmens nevet adhatja meg vele. Ha semmit nem ad meg, akkor a programfájl neve, pontosabban a MASM @fileName változójának tartalma + _TEXT lesz a szegmensnév. Ha sehol sem adja meg, akkor a CLfunc vagy a WORKFUNCS makró beállítja a szegmensnév alapértelmezését.

DATASEG megnevezés

Az adatszegmens nevét adhatja meg vele. Ha nem ad meg nevet, akkor a _DATA lesz az adatszegmens neve.
Figyelem! A DATASEG makrót mindenképpen használni kell, nem hagyható el úgy, mint a CODESEG!

CLstatic < típus változó érték, ... >

A változókat definiálja az adatszegmensben.
A megadható típusok:

byte	1 bájt;
len	EQU érték, az előzőleg definiált változó hossza;
int	2 bájt;
log	2 bájt;
long	4 bájt;
double	8 bájt;
cptr	4 bájt, egy név_OFF és egy név_SEG nevű azonosítót (EQU) hoz létre;
string	0-val lezárt karakterlánc.

CLlocal < típus név, ... >

A változókat definiálja a veremben.
A megadható *típus*-ok:

void	0 bájt
char	4 bájt
int	2 bájt

long	4 bájt
double	8 bájt
log	2 bájt
date	4 bájt

A makrót a CLfunc és a CLcode előtt kell használni!

CLextgern < típus név, ... >

Külső szimbólumokat definiál. A nevek elé egy aláhúzásjelet tesz (Microsoft C konvenció). Természetesen gondoskodik arról, hogy a függvényen belül aláhúzásjel nélkül is hivatkozhasson a *név-re.*

A megadható *típus*-ok:

byte, int, log, long, double, cptr, far.

CLlabel < típus címke, ... >

Változódefiniálás közben használható. A következő változóra más típusúként hivatkozhat.

A megadható *típus*-ok:

byte	1 bájt
string	1 bájt
int	2 bájt
log	2 bájt
long	4 bájt
double	8 bájt
cptre	4 bájt

Cglobal < név1, ... >

A megadott nevet PUBLIC-ként definiálja.

Figyelem! A *név* elé egy aláhúzásjelet tesz!

Ccall funkció < param1, ... >

C nyelvű függvényt hív meg, és elvégzi a paraméterek átadását is. Csak a Clipper makrókkal definiált változókat képes helyesen átadni

WORKFUNCS

Kódszegmens-deklarációt végez el, ezután írhatja meg a saját assembly nyelvű rutinokat.

ENDWORK

A munkakódszegmens vége. Egy ENDS utasítást ad ki.

SES regiszter, változó

A megadott 4 bájtos *változó*-ba elmenti az ES és a *regiszter* tartalmát. A LES assembly utasítás fordítottja.

SDS regiszter, változó

A megadott 4 bájtos *változó*-ba elmenti a DS és a *regiszter* tartalmát. Az LDS assemlby utasítás fordítottja.

DOSREQ funkciószám

Az INT 21h DOS funkciókérést hajtja végre. Az Ah-ba tölti be a *funkciószám*-ot.

$define név érték

Az EQU utasítást hajtja végre (név EQU érték).

OFFPART változó

Egy szegmens: offszet típusú pointerváltozóban az offszet érték.

SEGPART változó

Egy szegmens: offszet típusú pointerváltozóban a szegmensérték.

LSW változó

EGy 4 bájtos, szám típusú változóban a kisebb helyi értékű 2 bájtot jelöli.

MSW változó

Egy 4 bájtos, szám típusú változóban a nagyobb helyi értékű 2 bájtot jelöli.

TAR

A Clipper parancsok

A Clipper előző verzióiban a fájlkezelő parancsok többsége változókkal nem paraméterezhető, csak makrókkal. Ez a megoldás nagyon lelassítja a programot és nagy a tárigénye. Ebben az új verzióban a Clipper készítői már felkínáltak egy másik megoldást is. A parancsokat „ál függvényként" is lehet használni. A paraméterlista így kissé öszvérszerű lett. A legfontosabb paramétereket zárójelek között lehet megadni, mint egy függvény esetében, ilyenkor már kötelező a változónév használata! Ez a forma gyorsabb, és sokkal kevesebb helyet foglal el, mint a makrós forma.

=

<változónév> = <kifejezés>

Egy memóriaváltozónak ad értéket, ha kell létre is hozza azt, A változó típusa a *kifejezés* típusától függ. Az alapértelmezés a PRIVATE változó, hacsak nem definiálta már másnak.

Figyelem! Ez az értékadás nem használható adatbázismezők feltöltésére! Még akkor sem, ha az ALIAS nevet kiírja a változó neve elé.

Nem a kívánt eredményt kapja, ha a következő formában használja:

```
USE PROBA
PROBA->MEZO = 'ÉRTÉKE'
```

Ilyenkor létrehoz egy MEZO nevű PRIVATE változót.
Egy változó típusa értékadással megváltoztatható, nem okoz hibát.

```
VALT = '12.45'          && Karakteres típusú.
VALT = VAL(VALT)        && Numerikus típusú.
```

Lásd még: STORE.

;

;

A programsor elválasztó karaktere. Karakterlánc belsejében nem használható, ilyenkor válassza ketté a sztringet.

Példa:

```
? 'Kérem helyezze az A egységbe a CL0102 lemezt,' + ;
  'és nyomjon le egy billentyűt!'

LOCATE FOR NEV = UGYFEL .AND. DATUM = BEERKDAT . AND. ;
KOD= MITKER
```

!

! <parancs> | (<változónév>)

Lehetővé teszi egy MS-DOS parancs kiadását a Clipperből. Memóriarezidens programot *ne* indítson el!

Különleges eset:

! COMMAND

Ilyenkor a DOS indul el, és bármilyen parancsot kiadhat, ha van még elegendő memória. A saját programba a DOS EXIT utasításával térhet vissza!

Példa:

! FORMAT A: /S/V

Az MS-DOS lemez előkészítő programjának elindítása.

Lásd még: INDEX, PACK, RUN, SET INDEX, SET UNIQUE, USE.

&

& <karakteres változó>

Makró helyettesítésére használható, a *változó* értéke kerül a parancssorba. Olyan utasításban érdemes alkalmazni, amelyik változóval nem paraméterezhető.

A makrók nem tartalmazhatnak parancsszavakat vagy azok rövidítését, valamint vesszőkaraktert, de parancsparaméterként felhasználhatók. Ha függvényre hivatkozik így, akkor azt fel kell sorolni a program elején, az EXTERNAL rutinok között. Kifejezésekben a makrónév végét a . (pont) karakterrel lehet jelezni.

&&

&& <megjegyzés>

Megjegyzés írása a parancssorba.

Példa:

villogo = 'W*/N,,,,' && A villogó szín beállításához.

Lásd még: *, NOTE.

*

*** <megjegyzés>**

Megjegyzéssor írása a szövegbe.

Példa:

* A villogó szín beállításához.
villogó = 'W*/N,,,,'

Lásd még: &&, NOTE.

?

? <kifejezéslista>

A képernyő következő sorába írja a kifejezés(ek) értékét. A lista elemeit vesszővel kell elválasztani.

Lásd még: ??, @...SAY, TEXT.

??

?? <kifejezéslista>

A képernyő aktuális sorába írja a kifejezés(ek) értékét. A lista elemeit vesszővel kell elválasztani.

Lásd még: ?, @...SAY, TEXT.

@

@ <sor, oszlop>

A *sor*-ban az *oszlop*-tól kezdve a *sor* végéig <space> karakterket ír. Ha olyan koordinátákat ad meg, amelyek kívül esnek a képernyőn, akkor nem csinál semmit.

Az érvényes sorkoordináták: 0-24, az érvényes oszlopkoordináták: 0-79.

Lásd még: @...CLEAR, CLEAR, SCROLL( ).

@ BOX

@ <bal felső sor>, <bal felső oszlop>;
<jobb alsó sor>,<jobb alsó oszlop>;
BOX <karakterlánc>

Keretet rajzol a kijelölt „ablak" köré. A keretet a *karakterlánc*-ban megadott karakterekből építi fel, az óramutató járásával megegyező irányban. Ehhez 8 karakterre van szükség. Megadható a 9. karakter is, ilyenkor ezzel tölti ki az ablakot. Ha nem adja meg ezt a karaktert, akkor semmivel sem tölti ki az ablakot, tehát nincs alapértelmezett karakter. Ha hibás koordinátákat ad meg, vagy ha a sarkokat felcseréli, akkor sem csinál semmit. Nem kötelező mind a 8 keretkaraktert megadni, ilyenkor az utolsó karakterrel helyettesíti a hiányzókat, és nem tölti ki az ablakot.

Példa:

```
KERET = '12345678s'
@  10,10,15,15 BOX KERET
*
*  Keretet rajzol a 10,10-től a 15,15-ig.
*
*  122223
*  8ssss4
*  8ssss4
*  8ssss4
*  8ssss4
*  766665
*
*  formában.
```

Lásd még: @...CLEAR, @...TO.

@ CLEAR

@ <bal felső sor>, <bal felső oszlop> CLEAR;
[TO <jobb alsó sor>, <jobb alsó oszlop>]

A bal felső saroktól törli a képernyőt úgy, mint egy ablakot. Ha nem adja meg a jobb alsó sarok koordinátáit, akkor a 24, 79 értékekkel dolgozik. Helytelen koordináták megadásakor, vagy ha a sarkok koordinátáit felcseréli, a parancs nem hajtódik végre.

Lásd még: @...BOX, CLEAR, SCROLL( ).

@ GET

@ <sor, oszlop> [SAY <kifejezés>;
[PICTURE <karakteres kifejezés>]];
GET <változónév> | <mezőnév>;
[PICTURE <karakteres kifejezés>];
[RANGE <alsó határ>, <felső határ>];
[VALID <logikai kifejezés>]

Adatbekérés kijelölése a képernyőn. Ha hibás sor- vagy oszlopkoordinátát ad meg, a parancs nem hajtódik végre. A *mezőnév* egy másik munkaterületen lévő adatbázismező is lehet!

A PICTURE értékei:

Funkciók

A funkciók elé minden esetben @-ot kell írni!

!	Nagybetűre konvertál.
A	Csak betűket fogad el.
B	A numerikus adatot balra igazítja.
D	SET DATE dátumforma
E	Dátum típusú változóban felcseréli az első két összetevőt. Számok esetén felcserélődik a tizedespont és a helyi értéket jelző vessző funkciója, de a maszkkaraktereket az eredeti formában kell használni!

Példa:

```
maszk1='999,999,999.99'          &&  tizedespontot használ
maszk2='@E 999,999,999.99'       &&  tizedesvesszőt használ,
                                 &&  a vesszők helyett
                                 &&  ponttal tagolja a
                                 &&  számokat
```

K Ha az első lenyomott billentyű nem a kurzormozgató billentyű volt, akkor törli a mezőt. A számokban ez az alapértelmezés.
R A PICTURE paraméternél megadott nemfunkció – és nemmaszkkaraktereket nem viszi ki a mezőbe, csak akkor, ha nem adja meg ezt az opciót.
Sn Karakterlánc bekérésekor egy egysoros „ablakban" dolgozik. Ha a karakterlánc hosszabb, mint a megadott hossz, akkor elgörgeti azt.
Z Zéruselnyomás.

Maszkok

Ha együtt használja a funkciókkal, akkor egy <space> karakter kell közéjük!

! Nagybetűre konvertál.
Hasonló a 9-es maszkhoz, de elfogadja a <space> karaktert is.
$ A szám elején lévő nullákat $-ral tölti fel.
* A szám elején lévő nullákat *-gal tölti fel.
, Számok tagolására használt jel.
. Tizedespont.
9 Csak számjegyek (numerikus adatban előjel is).
A Csak betűket fogad el.
L Csak logikai adatot fogad el.
N Alfanumerikus adatot jelöl.
X Bármilyen karaktert elfogad.
Y A logikai adatok közül is csak az Y vagy az N lehet.

Bármilyen más karakter megjelenik a mezőben, azt már nem lehet felülírni!

A RANGE paraméter számokban és dátum típusú adatokban használható. Megadható egy zárt intervallum, ezen belül kell lennie az adatnak.

A VALID lehetőséget ad az input ellenőrzésére. A mező addig nem hagyható el, amíg a *kifejezés igaz* nem lesz (egy saját függvénnyel pl. bármilyen bonyolult feltételt megszabhat).

Példa:

```
AKAR = 'I'
@    5,0 SAY 'Folytatja a munkát (I/N) ? : ' GET AKAR ;
     PICTURE '@!' VALID AKAR $ 'IN'
     READ
*
* Addig nem lép tovább, amíg I-t vagy N-et nem válaszol.
*
```

A következő kurzormozgató és különleges billentyűk használhatók:

Home	A kurzort az aktuális mező elejére állítja.
^Home	A kurzort az első mező elejére állítja.
End	A kurzort az aktuális mező végére állítja.
^End	A kurzort az utolsó mező elejére állítja.
Felfelé nyíl	A kurzort az előző mező elejére állítja.
Lefelé nyíl	A kurzort a következő mező elejére állítja.
^Jobbra nyíl	A kurzort a következő szó elejére állítja.
^Balra nyíl	A kurzort az előző szó elejére állítja.
^Y	A mező végéig töröl.
PgDn, PgUp	Befejezi a változó és a mező szerkesztését.
Esc	Eldobja az eddig bevitt adatokat, majd kilép.
^U, Alt-U	Visszaállítja a szerkesztés előtti állapotot.
Return	A következő mező elejére áll, majd befejezi a változó és a mező szerkesztését.

Lásd még: ?, ??, @, @...CLEAR, @...TO, CLEAR, CLEAR GETS, READ, SET BELL, SET CONFIRM, SET DELIMITERS, SET DEVICE, SET FORMAT, SET INTENSITY, TEXT, COL( ), PCOL( ), PROW( ), ROW( ), SETPRC( ).

@ PROMPT

@ <sor, oszlop> PROMPT <karakterlánc1>;
[MESSAGE <karakterlánc2>]

A menügenerláló utasításcsoportba tartozik. Egy menüpont helyét és rövid tartalmát adhatja meg, a menüpontot részletesebben a MESSAGE paraméterrel írhatjuk le. Ez az üzenet a SET MESSAGE utasítással beállított sorban fog megjelenni. Egy menühöz legfeljebb 32 menüpont adható meg. Ha hibás sor- vagy oszlopkoordinátát adott meg, akkor a parancs nem hajtódik végre.

Lásd még: MENU TO, SET COLOR, SET COLOUR, SET MESSAGE, SET WRAP, ACHOICE( ), SETCOLOR( ), SETCOLOUR( ).

@ SAY

@ <sor, oszlop> SAY <kifejezés>;
[PICTURE <karakteres kifejezés>]

A kifejezéseket formázottan jeleníti meg. Hibás koordináták megadásakor a parancs nem hajtódik végre.

A PICTURE értékei:

Funkciók

A funkciók elé minden esetben @-ot kell írni!

! Nagybetűre konvertál.
(A negatív számot zárójelbe teszi (a teljes hosszt!).
) A negatív számot zárójelbe teszi (az aktuális hosszt!).
A Csak betűket fogad el.
B Numerikus adatot balra igazítja.
C Negatív szám után CR, „KÖVETEL" szöveget ír.
D SET DATE dátumforma.
E Dátum típusú változóban felcseréli az első két összetevőt. Számok esetén felcserélődik a tizedespont és a helyi értéket jelző vessző funkciója, de a maszkkaraktereket az eredeti formában kell használni!

Példa:

```
maszk1='999,999,999.99'          &&  tizedespontot használ
maszk2='@E 999,999,999.99'       &&  tizedesvesszőt használ,
                                 &&  a vesszők helyett
                                 &&  ponttal tagolja
                                 &&  számokat
```

X Negatív szám után DB, „TARTOZIK" szöveget ír, de nem ír előjelet!
Z Zéruselnyomás.

Maszkok

Ha együtt használja a funkciókkal, akkor egy <space> karakter kell közéjük!

!	Nagybetűre konvertál.
#	Hasonló a 9-es maszkhoz, de elfogadja a <space> karaktert is.
$	A szám elején lévő nullákat $-ral tölti fel.
*	A szám elején lévő nullákat *-gal tölti fel.
,	Számok tagolására használt jel.
.	Tizedespont.
9	Csak számjegyek (numerikus adatban előjel is).
A	Csak betűket fogad el.
L	Csak logikai adatot fogad el.
N	Alfanumerikus adatot jelöl.
X	Bármilyen karaktert elfogad.
Y	A logikai adatok közül is csak az Y vagy az N lehet.

Lásd még: ?, ??, @, @...CLEAR, @...TO, CLEAR, CLEAR GETS, READ, SET BELL, SET CONFIRM, SET DELIMITERS, SET DEVICE, SET FORMAT, SET INTENSITY, TEXT, COL( ), PCOL( ), PROW( ), ROW( ), SETPRC( ).

@ TO

@ <bal felső sor>, <bal felső oszlop> TO;
<jobb alsó sor>, <jobb alsó oszlop> [DOUBLE]

Kettős keretet rajzol a kijelölt „ablak" köré, a DOUBLE megadásával. Ha hibás sor- vagy oszlopkoordinátákat ad meg, vagy felcseréli a sarkokat, a parancs nem hajtódik végre.

Lásd még: @...BOX, @...CLEAR, ACHOICE( ), DBEDIT( ), SCROLL( ).

ACEPT

ACCEPT [<karakteres kifejezés>] TO <változónév>

Megjeleníti a képernyőn a *karakteres kifejezés* értékét, majd a kurzor aktuális pozíciójától egy karakterláncot olvas a változóba a billentyűzetről.

Lásd még: @...GET, @...SAY, INPUT, WAIT, INKEY( ).

APPEND BLANK

APPEND BLANK

Üres rekordot hoz létre az adatbázis végén. (Hálózatban az új rekord lezárt lesz!). A rekordmutató erre a rekordra fog mutatni. Ha nem sikerül a hálózatban az új rekordot létrehozni vagy lezárni, akkor a NETERR( ) függvény *igaz* értéket ad vissza.

Lásd még: APPEND FROM, SET CARRY, SET CONFIRM, SET FORMAT.

APPEND FROM

APPEND [<érvényességi kör>] FROM <fájlnév> | (<változónév1>);
[FIELDS <mezőlista>] [FOR <feltétel1>] ;
[WHILE <feltétel2> [DELIMITED WITH [BLANK | ;
<határolójel> | (<változónév2>)]] | SDF

Egy másik állományból rekordokat fűz az aktuális adatbázishoz. Hálózatban az APPEND ... FROM közös használatra nyitja meg az input állományt. Ebben a parancsban az *érvényességi kör* az input állományra vonatkozik. Az alapértelmezés az ALL.

Lásd még: COPY, FREAD( ).

AVERAGE

AVERAGE [érvényességi kör] <mezőlista> TO <változólista> ;
[FOR <feltétel1>] [WHILE <feltétel2>]

Segítségével egy adatbázismező aktuális értékeinek átlagát egyszerűen kiszámíthatja. A megadott tartományon belül kiszámítja az adatszó átlagértékét, és azt egy változóban tárolja. Egyszerre nemcsak egy mezőt lehet megadni, hanem többet is.

Példa:

```
*
*  Kiszámítja az ELADAR és a NYILVAR átlagait.
*  Az ELADAR átlagát az ELADAS, a NYILVAR átlagát pedig az
*  ERTEK fogja tartalmazni.
*
```

```
AVERAGE ELADAR, NYILVAR TO ELADAS,;
ERTEK FOR VEVO = 'Kovács'
```

Lásd még: SUM, TOTAL.

BEGIN SEQUENCE '87

```
BEGIN   SEQUENCE
        -
        -
        -
        [BREAK]
END
```

A szorosan egymáshoz tartozó utasításokat egy csoportba – szekvenciába – fogja össze. A legtöbb hibakezelő rutin már feltételezi ezt a csoportosítást. Ha a BREAK utasítást egy ilyen szekvencia közben adja ki, akkor az aktuális szekvencia utáni utasításon folytatódik a programvégrehajtás.

Lásd még: BREAK.

BREAK '87

```
BREAK
```

Ha a BREAK utasítást egy BEGIN SEQUENCE és egy END utasítás között adja ki, akkor az aktuális szekvencia utáni utasításon folytatódik a programvégrehajtás.

Lásd még: BEGIN SEQUENCE.

CALL

CALL <rutin> [WITH <paraméterlista>]

A CALL paranccsal olyan nem Clipper nyelvű rutinokat hívhat meg, amelyek nem adnak vissza értéket. Ennél a rutinhívási formánál nem a Clipper paraméterátadási procedúráját kell követni, hanem a Microsoft C 5.00-ét. Használata a dBASE kompatibilitás miatt érdekes. A CALL paranccsal csak 7 paramétert lehet átadni a rutinnak. Használja helyette a sokkal több támogatást adó függvényhívási formát! A paramétereket kétféleképpen lehet átadni. Az érték szerinti átadáskor a hívott rutin a paramétereket tetszés szerint módosíthatja. A másik lehetőség, ha a paraméter nem konkrét érték, hanem egy memóriacím, méghozzá annak a változónak a címe, amely az értéket tartalmazza. Ha a rutin bármit változtat az értéken, az a változóban is módosulni fog! Természetesen ilyen formában csak változókat lehet paraméterként átadni. Az ilyen típusú paraméterátadáshoz a változó elé @ jelet kell tenni.

Lásd még: DO, FUNCTION, PROCEDURE.

CANCEL

CANCEL

Megállítja a program végrehajtását, majd visszatér a DOS-hoz, és lezárja az összes nyitott állományt.

Lásd még: QUIT, RETURN.

CLEAR

CLEAR

Törli a képernyőt és az összes olyan aktív GET-et, amelyekre még nem adott ki READ-et, majd az 0,0 pozícióra állítja a kurzort. A képernyő törlésére használja inkább a CLEAR SCREEN utasítást!

Lásd még: @, @...CLEAR, CLEAR GETS, SCROLL( ).

CLEAR ALL

CLEAR ALL

Lezárja az összes adatbázist, a memo- és indexállományokat, valamint azokat a kapcsolatokat, amelyeket a SET RELATION-nel jelölt ki. Törli az aktuális memóriaváltozókat, az összes aktív GET-et. Az 1 munkaterület lesz *az aktuális.*

Lásd még: CLEAR MEMORY, CLOSE, RELEASE.

CLEAR GETS

CLEAR GETS

Törli a még be nem olvasott (aktív) GET parancsokat. Ezután a READ parancs *csak a formátumállományban megadott GET parancsokat hajtja végre.*

Lásd még: @...CLEAR, CLEAR.

CLEAR MEMORY

CLEAR MEMORY

Törli az összes memóriaváltozót, akár PUBLIC, akár PRIVATE típusúnak definiálta őket.

Lásd még: CLEAR ALL, RELEASE ALL.

CLEAR SCREEN '87

CLEAR SCREEN

A Clipper képernyőtörlő utasítása. Csak a képernyőt törli, a GET-eket nem.

Lásd még: @, @...CLEAR, CLEAR, CLEAR GETS, SCROLL( ).

CLEAR TYPEAHEAD

CLEAR TYPEAHEAD

Kiüríti a billentyűzetpuffert. Akkor használható, ha a felhasználó számára nem engedi meg, hogy megszokásból (vakon) adjon választ valamely kérdésre.

Lásd még: KEYBOARD, SET TYPEAHEAD, ACHOICE( ), DBEDIT( ), LASTKEY( ), NEXTKEY( ).

CLOSE

CLOSE

Lezárja az aktuális .dbf fájlt és a hozzá tartozó indexfájlokat.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

CLOSE ALTERNATE

CLOSE ALTERNATE

Lezárja az aktuális kimeneti fájlt.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

CLOSE DATABASES

CLOSE DATABASES

Lezárja az összes adatbázisfájlt és a hozzájuk tartozó valamennyi állományt.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

CLOSE FORMAT

CLOSE FORMAT

Lezárja a képernyőformátum-fájlt. Mivel a Clipper ezeket az állományokat is lefordítja és a programhoz szerkeszti, a hatása csak annyi, hogy ezentúl a READ utasítás nem fogja végrehajtani a formátumállományban leírt @... utasításokat.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

CLOSE INDEXES

CLOSE INDEXES

Minden munkaterületen lezárja az indexállományokat.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

CLOSE PROCEDURES

CLOSE PROCEDURES

A Clipper parancsok közé csak dBASE kompatibilitása miatt került be. Az a funkciója, hogy lezárja a szubrutingyűjteményt tartalmazó fájlt. Mivel a Clipper programok futása közben ilyen állomány nincs nyitva, használata felesleges.

Lásd még: CANCEL, CLEAR ALL, CLEAR MEMORY, QUIT, RETURN, SET ALTERNATE TO, USE.

COMMIT '87

COMMIT [ALL]

Felírja a lemezre a Clipper belső adatpuffereinek tartalmát. Így egy módosítás után biztos lehet benne, hogy az adatok egy esetleges áramkimaradáskor sem vesznek el.

Lásd még: SKIP.

CONTINUE

CONTINUE

Az előzőleg végrehajtott LOCATE parancsban előírt feltételeknek megfelelően megkeresi a következő rekordot. Ha talál, akkor a FOUND( ) függvény *igaz* értéket ad vissza. Minden egyes munkaterületen használhatók egymástól független LOCATE utasítások.

Lásd még: LOCATE, FOUND( ).

COPY FILE

COPY FILE <fájlnév1> | (<változó 1> TO ;
<fájlnév2> | (<változó2>)

A megadott állományt egy másikba másolja át. A fájlneveket teljesen ki kell írni. Nyitott adatbázist nem lehet így másolni, használja helyette a COPY TO ... parancsot. A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, COPY, SET DEFAULT, USE.

COPY STRUCTURE

COPY STUCTURE TO <állomány> | (<változónév>) ;
[FIELDS mezőnév1, mezőnév2, ...]

Egy nyitott állomány szerkezetéből egy másikat hoz létre. Az alapértelmezés a teljes szerkezet átmásolása. Hálózatban a COPY kizárólagos használatra nyitja meg a fájlt. A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: COPY STRUCTURE EXTENDED, CREATE.

COPY STRUCTURE EXTENDED

COPY TO <új fájl> | (<változónév>) STRUCTURE EXTENDED;
[FIELDS <mezőnév1> [,<mezőnév2>...]]

Egy különleges szerkezetű adatbázist hoz létre, amelyet az aktuális adatbázis mezőinek leírásával tölt fel. Az új állománynak 4 mezője lesz:

FIELD_NAME	karakteres,	a hosszúsága	10	(mező neve)
FIELD_TYPE	karakteres,	a hosszúsága	1	(mező típusa)
FIELD_LEN	numerikus,	a hosszúsága	3	(mező hossza)
FIELD_DEC	numerikus,	a hosszúsága	3	(tizedesjegyek száma)

Az aktuális állomány szerkezetét így lehet lemásolni, ill. megváltoztatni, és egy új állományt létrehozni a programból. Hálózatban a COPY TO kizárólagos használatra nyitja meg a fájlt. A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CREATE, CREATE FROM, AFIELDS( ), FIELD( ), TYPE( ).

COPY TO

COPY TO [<érvényességi kör>] <fájlnév> | (<változónév1>) ;
[FIELD <mezőlista>] [FOR <feltétel1>] ;
[WHILE <feltétel2>] [SDF | DELIMITED ;
[WITH BLANK | <határolójel> | (<változónév2>)]]

Az aktuális adatbázist a megadott fájlba másolja át.
Az SDF hatására a képzett fájl a következő formátumú lesz:
– a kiterjesztés alapértelmezése .txt;

– állandó hosszúságú rekordok:
– a rekordokat CR, LF, (0Dh, 0Ah) karakterek zárják;
– a mezők is állandó hosszúságúak, nincs köztük határolójel;
– a karakteres mezők balra igazítottak;
– a számok karakteres formájúak, jobbra igazítva;
– a dátum típusú mezők ÉÉÉÉHHNN formájúak;
– a logikai mezők T vagy F tartalmúak;
– a fájlt az 1Ah karakter zárja le.

A DELIMITED opció hatása hasonlít az SDF-ére. Az így készített állomány formátuma a következő:

– – a kiterjesztés alapértelmezése .txt;
– – változó hosszúságú rekordok;
– – a rekordokat CR, LF (0Dh, 0Ah) karakterek zárják;
– – a mezők is változó hosszúságúak, a határolójelet a felhasználó adhatja meg, az alapértelmezés a vessző (,);
– – a karakteres mezőket balra igazítva, a záró <space> karaktereket levágva ábrázolja, a mezőket alapértelmezésben " (idézőjel) határolja;
– – a számokat karakteres formában, balra igazítva, a záró <space> karaktereket levágva ábrázolja;
– – a dátum típusú mezők ÉÉÉÉHHNN formájúak;
– – a logikai mezők T vagy F tartalmúak;
– – a fájlt az 1Ah karakter zárja le.

Figyelem! Hálózatban a COPY TO kizárólagos használatra nyitja meg a fájlt. A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: APPEND FROM, COPY FILE, COPY STRUCTURE, SET DELETED.

COUNT

COUNT [<érvényességi kör>] [FOR <feltétel1>] ;
[WHILE <feltétel2>] TO <memóriaváltozó>

A megadott feltételnek megfelelő rekordok számát a változóban tárolja. Az *érvényességi kör* alapértelmezése ALL.

Lásd még: AVERAGE, SUM, TOTAL, LASTREC( ).

CREATE

CREATE <adatbázisnév> | (<változónév>)

Üres, bővített szerkezetű fájlt hoz létre, négy mezővel. Hálózatban a CREATE kizárólagos használatra nyitja meg a fájlt. Egy új adatbázis kialakításához ezt a parancsot a CREATE FROM paranccsal együtt használja!

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Az új állománynak 4 mezője lesz:

FIELD_NAME	karakteres,	a hosszúsága	10	(mező neve)
FIELD_TYPE	karakteres,	a hosszúsága	1	(mező típusa)
FIELD_LEN	numerikus,	a hosszúsága	3	(mező hossza)
FIELD_DEC	numerikus,	a hosszúsága	3	(tizedesjegyek száma)

Lásd még: COPY STRUCTURE EXTENDED, CREATE FROM.

CREATE FROM

CREATE <új fájl> | (<változónév1>) FROM ;
<leírófájl> | (<változónév2>)

Olyan új adatbázisfájlt hoz létre, amelyben a szerkezetet a CREATE paranccsal létrehozott fájl tartalma határozza meg. Hálózatban a CREATE ... FROM ... közös használatra nyitja meg az input fájlt.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Karakteres mezők használatakor 999-nél hosszabbakat is megadhat, ugyanis a FIELD_DEC és a FIELD_LEN a következőképpen határozza meg a hosszúságot: hossz = FIELD_DEC * 256 + FIELD_LEN.

Lásd még: COPY STRUCTURE EXTENDED, CREATE.

DECLARE

DECLARE <tömbnév1>[<méret1>] [,<tömbnév2>[<méret2>]]

PRIVATE típusú tömböt hoz létre. (A méretet határoló [] jelek a parancs szintaxisához tartoznak!) Egy tömb legfeljebb 4096 elemet tartalmazhat, és a tömb elemei nemcsak egyféle típusúak lehetnek. Minden tömb egy változónak felel meg, max. 2048 tömböt lehet használni. Már a parancs formájából is kiderül, hogy a Clipper tömbjei csak „egydimenziósak" lehetnek. A tömböket a SAVE parancs nem tudja lemezre írni.

A tömbelemeket memóriaváltozóként is használhatja. Csak akkor nem használhat tömbelemet memóriaváltozó helyett, ha makróval akar egy tömböt létrehozni.

Hibásak tehát a következő utasítások:

```
TOMBNEV=    'ELSO [2]'
DECLARE     &TOMBNEV
PUBLIC      &TOMBNEV
PRIVATE     &TOMBNEV
```

Érvényes viszont a következő forma:

```
TOMBNEV=    'ELSO'
DECLARE     &TOMBNEV[2]
PUBLIC      &TOMBNEV[2]
PRIVATE     &TOMBNEV[2]
```

Lásd még: ADEL( ), ADIR( ), AFILL( ), AINS( ), ASCAN( ), ASORT( ).

DELETE

DELETE [<érvényességi kör>] [FOR <feltétel1>] ;
[WHILE <feltétel2>]

Logikailag törli a megadott rekordokat, alapértelmezésben az aktuálisat. A parancs hálózatban történő használata előtt az állományt az FLOCK( ) függvénnyel le kell foglalni.

Lásd még: PACK, RECALL, SET DELETED.

DELETE FILE

DELETE FILE <fájlnév> | (<változónév>)

Egy állományt töröl a lemezről. Nyitott állomány *nem* törölhető!
A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, ERASE, FILE( ).

DIR

DIR [<meghajtó> : [<útvonal> [<fájlnévmaszk>]] | ;
(<változónév>)]

A megadott lemezmeghajtóban lévő fájlok nevét írja ki. Alapértelmezésben az adatbázisok nevét, az utolsó módosítás idejét és a rekordszámot, minden más esetben a fájlnevet, a kiterjesztést, a fájl hosszát és az utolsó módosítás idejét.
A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: ADIR( ).

DISPLAY

DISPLAY [OFF] [érvényességi kör] <kifejezéslista> ;
[FOR <feltétel>] [WHILE <feltétel>] ;
[TO PRINT | FILE <fájlnév> | (<változónév>)]

Rekordokat és adatokat ír ki az aktív adatbázisból. A *kifejezéslista* nemcsak mezőket, hanem bármilyen más kifejezést is tartalmazhat. Ha nem ad meg *feltétel*-t, akkor csak az aktuális rekordot írja ki, különben az alapértelmezés, az ALL lép érvénybe.

Az OFF paraméter a rekordok sorszámának kiíratását tiltja le.

Lásd még: LIST.

DO

DO <rutin> [WITH <paraméterlista>]

A megadott *rutin*-t indítja el, amely különálló program vagy procedúra lehet. Függvényt is meg lehet így hívni. A *rutin*-t Clipper, C vagy assembly nyelven írhatja meg. A paramétereket kétféleképpen adhatja át. Az érték szerinti átadáskor a hívott rutin a paramétereket tetszés szerint módosíthatja. A másik lehetőség az, ha a paraméter nem konkrét érték, hanem egy memóriacím, méghozzá annak a változónak a címe, amely az értéket tartalmazza. Ha a rutin bármit megváltoztat az értéken, az a változóban is módosulni fog! Természetesen így csak változókat lehet paraméterként átadni. Az ilyen típusú paraméterátadáskor a változó elé @ jelet kell tenni.

Lásd még: CALL, FUNCTION, PARAMETERS, PRIVATE, PROCEDURE, PUBLIC,RETURN, SET PROCEDURE.

DO CASE

```
DO    CASE
      CASE <feltétel1>
      -
      <utasítás>
      -
      [CASE <feltétel2>]
      [CASE <feltételn>]
      [OTHERWISE]
ENDCASE
```

Többszörös elágaztatási lehetőség. Ha a megadott feltétel *igaz*, akkor a CASE utáni utasítások hajtódnak végre. A program csak az első igaz ágat hajtja végre, utána az ENDCASE-t követő utasításokkal folytatja a végrehajtást.

Lásd még: DO, DO WHILE, IF, IF( ), IIF( ).

DO WHILE

```
DO    WHILE <feltétel>
      -
      -
      -
      [LOOP]
      [EXIT]
ENDDO
```

Strukturált ciklus a programban. A ciklusnak akkor van vége, ha a *feltétel hamis* lesz. A DO WHILE .T. végtelen ciklust okoz. A LOOP visszaugrik a ciklus elejére, és újra kiértékeli a *feltétel*-t. Az EXIT a *feltétel* vizsgálata nélkül kiugrik a ciklusból, majd az ENDDO utáni parancs hajtódik végre. Több ciklust egymásba is ágyazhat.

Lásd még: FOR, IF, LIST, RETURN.

EDIT

EDIT

Ha használni próbálja, a fordító felismeri és hibaüzenetet is ad. Csak dBASE kompatibilitása miatt került be a kulcsszavak közé.

EJECT

EJECT

A SET DEVICE parancs aktuális állapotától függetlenül lapdobáskaraktert (12, 0Ch) küld a nyomtatóhoz, majd a PROW( ) és a PCOL( ) értékeit nullára állítja.

Lásd még: PCOL( ), PROW( ), SETPRC( ).

ELSE

ELSE

Egy IF vagy egy ELSEIF utasítás *hamis* ágának kezdetét jelzi.

Lásd még: ELSEIF, END, ENDIF, IF.

ELSEIF '87

ELSEIF

Egy IF vagy egy másik ELSEIF hamis ágának kezdetét jelzi, egy új feltételt adhat meg vele. Ezt a feltételágat nem kell külön ENDIF-fel lezárni!

Tulajdonképpen egy
```
IF
.
.
ELSE
      IF
      .
      .
      ENDIF
ENDIF
```
szerkezet rövidítése.

Lásd még: ELSE, END, ENDIF, IF.

END '87

END

Alapértelmezésben a BEGIN SEQUENCE-szel kezdett utasításcsoport végét jelzi. Használható még az összes END-del kezdődő utasítás *helyett* is!

Lásd még: BEGIN SEQUENCE, ENDCASE, ENDDO, ENDIF, ENDTEXT.

ENDCASE

ENDCASE

A többszörös elágazást lehetővé tevő DO CASE utasítás végét jelzi, használható helyette az END is.

Lásd még: DO CASE, END.

ENDDO

ENDDO

A DO WHILE ciklus záró utasítása, használható helyette az END is.

Lásd még: DO WHILE, END.

ENDIF

ENDIF

Az IF feltételes elágazás végét jelző utasítás, használható helyette az END is.

Lásd még: ELSE, ELSEIF, END, IF, IF( ), IIF( ).

ENDTEXT

ENDTEXT

A TEXT utasítással kezdett szövegblokk végét jelzi, használható helyette az END is.

Lásd még: END, TEXT.

ERASE

ERASE <fájlnév> | (<változónév>)

Törli a megadott fájlt a katalógusból. Nyitott állomány nem törölhető!
A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, DELETE FILE, USE, FILE( ).

EXIT

EXIT

A feltétel vizsgálata nélkül kilép a DO WHILE és a FOR ciklusból.

Lásd még: DO WHILE, END, ENDDO, LOOP.

EXTERNAL

EXTERNAL <külső rutinok listája>

Azoknak a rutinoknak a neveit kell megadni, amelyeket másik objectmodulban definiált. Ha egy rutinnév csak makrókifejezésben fordul elő, még a szerkesztőprogram (linker) sem fogja magától a programhoz szerkeszteni a megfelelő modult!

Lásd még: PUBLIC.

FIND

FIND <kulcskonstans> | (<változónév>)

A megadott kulccsal megegyező indexű első rekordot keresi meg, tehát csak indexelt állományon használható. Ha talál, akkor a FOUND( ) függvény *igaz*értéketadvissza. *Kulcskonstans*-ként csak karakteres kifejezést, változóként pedig numerikus és dátum típusút adhat meg.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: INDEX, LOCATE, SEEK, SET INDEX, SET ORDER, SET SOFTSEEK, EOF( ), FOUND( ).

FOR

```
FOR <memóriaváltozó> = <kezdőérték> TO <végérték> ;
    [STEP <lépésköz>]
    -
    <parancs>
    -
    [EXIT]
NEXT
```

Olyan ciklus szervezése, amelyet előre meghatározott számú alkalommal hajt végre. Alapértelmezésben a *kezdőérték* és a *végérték* különbsége – +1 – adja a végrehajtás számát. Ha az első végrehajtás előtt sem teljesül a végfeltétel, akkor egyszer sem hajtódik végre a ciklusmag. A *memóriaváltozó* a ciklus belsejében mindig az aktuális ciklusértéket tartalmazza, a ciklus végén pedig a *végérték* + *lépésköz* lesz az értéke. A *lépésköz* alapértelmezése 1, de lehet negatív is.

Lásd még: DO CASE, DO WHILE, IF.

FUNCTION

FUNCTION <név>
-
-
-
RETURN <kifejezés>

Felhasználói funkciót deklarál. Az eljárásokhoz hasonlóan a felhasználói funkciók is átvehetnek paramétereket. A funkciók bármely megfelelő típusú kifejezésben, valamint utasításokban és @...SAY parancsokban használhatók. A paramétereket kétféleképpen adhatja át. Az érték szerinti átadáskor a hívott rutin a paramétereket tetszés szerint módosíthatja. A másik lehetőség az, ha a paraméter nem konkrét érték, hanem egy memóriacím, méghozzá annak a változónak a címe, amely az értéket tartalmazza. Ha a rutin bármit megváltoztat az értéken, az a változóban is módosulni fog! Természetesen így csak változókat lehet paraméterként átadni. Az ilyen típusú paraméterátadáskor a változó elé @ jelet kell tenni.

Lásd még: PARAMETERS, PROCEDURE, RETURN.

GO

GO BOTTOM | TOP | <numerikus kifejezés>

A rekordmutatót közvetlenül a megadott rekordra helyezi. A BOTTOM az állomány vége, a TOP az állomány eleje (logikai sorrendben).

Lásd még: GOTO, SET DELETED, SET FILTER TO, SKIP, BOF( ), EOF( ), LASTREC( ), RECNO( ).

GOTO

GOTO BOTTOM | TOP | <numerikus kifejezés>

A rekordmutatót közvetlenül a megadott rekordra helyezi. A BOTTOM az állomány vége, a TOP az állomány eleje (logikai sorrendben).

Lásd még: GO, SET DELETED, SET FILTER TO, SKIP, BOF( ), EOF( ), LASTREC( ), RECNO( ).

IF

IF <feltétel1>

.

<utasítás>

.

[ELSEIF <feltétel2>]

.

<utasítás>

.

[ELSE]

.

ENDIF

Egy parancs feltételes végrehajtását teszi lehetővé a programban, egy másik útvonal választhatóságával. Az IF-eket egymásba is ágyazhatja. Ha ELSEIF utasítást használ, akkor azt nem kell külön ENDIF utasítással lezárni!

Lásd még: DO CASE, IF( ), IIF( ).

INDEX

INDEX ON <kifejezés> TO <fájlnév> | (<változónév>)

Az aktuális adatbázist a megadott módon logikailag rendezi. A képzett indexállománnyal a megadott sorrendben érhetők el az adatbázis rekordjai. Karakteres, numerikus és dátum típusú kulcsokat adhat meg. A kulcskifejezésben csak egyforma típusú mezők állhatnak. Ha különböznek az adatok, akkor egyforma típusúra (pl. karakteresre) kell konvertálni őket, ehhez a Clipper belső függvényeit használhatja. Ne használja a TRIM( ) függvényt az indexállomány létrehozásakor, mert később hibás hosszúságra utaló hibaüzenetet ad a Clipper. Ha mégis használja, akkor a kulcskifejezést annyi szóközkarakterrel kell kiegészíteni, amennyi a mezők hosszának megfelel. Ezt legkönnyebben a PAD( ) függvénnyel teheti meg.

Hálózatban az INDEX kizárólagos használatra nyitja meg a fájlt. A max. kulcshossz 1000 karakter. A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal. A parancs a létrehozott kulcsállományt meg is nyitja.

Lásd még: CLOSE, FIND, REINDEX, SEEK, SET INDEX, SET ORDER, SET UNIQUE, USE, DTOS( ), INDEXEXT( ), INDEXKEY( ), INDEXORD( ), PAD( ).

INPUT

INPUT [<karakterlánc>] TO <memóriaváltozó>

A kurzor aktuális pozíciójától tetszőleges típusú értéket gépelhet be egy *memóriaváltozó*-ba. A begépelt szöveg bármilyen kifejezés, akár függvény is lehet. Ha *karakterlánc*-ot gépel be, akkor azt idézőjelek között kell megadni! A gép először kiírja a *karakterlánc*-ot a képernyőre, és csak utána kéri be az adatot. Az ESC nem szakítja meg a begépelést.

Példa:

* Programsor:
SET DATE ANSI
INPUT 'Kérem a mai dátumot:' TO DATUM

* Képernyőn:
Kérem a mai dátumot: DATE( )

* A DATUM tartalma:
89.03.13.

Lásd még: ACCEPT, WAIT.

JOIN

JOIN WITH <aliasnév> | (<változónév1>) ;
TO <új fájl> | (<változónév2>) [FOR <feltétel1>] ;
[WHILE <feltétel2>] [FIELDS <mezőlista>]

A megadott rekordokat és mezőket egyesíti az aktuális, ill. a megadott adatbázisfájlból. Hálózatban a JOIN kizárólagos használatra nyitja meg a fájlt.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: APPEND FROM, REPLACE, SET RELATION.

KEYBOARD

KEYBOARD <karakterlánc>

A billentyűzetpuffert feltölti a megadott *karakterlánc*-cal. A következőkben a program úgy tesz, mintha a billentyűzetről gépelte volna be. A funkcióbillentyűket nem lehet így szimulálni! A parancs minden egyes használata kiüríti a billentyűzetpuffert, a pufferben tehát nem lehet összefűzni egy karakterláncot. A rendszer billentyűzetpufferét a SET TYPEAHEAD utasítással állíthatja be.

Lásd még: SET KEY, CHR( ), LASTKEY( ), NEXTKEY( ).

LABEL

LABEL FORM <fájlnév1> | (<változónév1>) [SAMPLE] ;
[<érvényességi kör>] [TO PRINT | FILE <fájlnév2> | ;
(<változónév2>)] [FOR <feltétel>] [WHILE <feltétel>]

Címzéseket nyomtat a megadott címzésmintafájllal. A SAMPLE paraméter a próbanyomtatás elvégzését jelenti. Hálózatban a LABEL FORM... közös használatra nyitja meg az inputfájlt, az alapértelmezés az ALL.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: REPORT FORM.

LIST

LIST [OFF] [<érvényességi kör>] <kifejezéslista>;
[FOR <feltétel1>] [WHILE <feltétel2>];
[TO PRINT | FILE <fájlnév> | (<változónév>)]

Rekordokat és adatokat ír ki az aktív adatbázisból. A *kifejezéslista* nemcsak mezőket, hanem bármilyen más kifejezést is tartalmazhat. Ha nem ad meg feltételt, akkor csak az aktuális rekordot írja ki, különben az alapértelmezés, az ALL lép érvénybe. Az OFF paraméter a rekordok sorszámának kiíratását tiltja le.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: DISPLAY.

LOCATE

LOCATE [<érvényességi kör>] [FOR <feltétel1>];
[WHILE <feltétel2>]

Megkeresi azt a rekordot, amelyik megfelel a feltételnek. Ha talál, akkor a FOUND( ) függvény *igaz* értéket ad vissza. A helyes működéshez vagy a FOR, vagy a WHILE feltételnek szerepelnie kell, bár a fordító szerint mindkettő elhagyható. Minden munkaterületen egy LOCATE parancs lehet érvényben. Ez a keresési mód index nélküli állományon is használható, ellentétben a FIND és a SEEK paranccsal. A fordító a WHILE paramétert is elfogadja, de futás közben mindig tipushibát jelez.

Lásd még: CONTINUE, FIND, SEEK, FOUND( ), STRTRAN( ).

LOOP

LOOP

A legutoljára végrehajtott DO WHILE utasításhoz tér vissza, majd újra kiértékeli a feltételt, és a végrehajtást ettől a ponttól folytatja.

Lásd még: DO WHILE.

MENU

MENU TO <memóriaváltozó>

Az előzőleg kiadott @...PROMT és SET MESSAGES utasításokkal meghatározott menüt indítja el. A kiválasztás eredményét a *memóriaváltozó*-ban tárolja. Ez az eredmény a @...PROMT utasítások fizikai sorszáma, ill. 0, ha nem választott ki semmit. A menüket egymásba is ágyazhatja. A *memóriaváltozó* kezdeti értéke határozza meg, hogy a megjelenéskor melyik menüpont lesz az aktuális.

A következő kurzormozgató billentyűket használhatja:

Felfelé nyíl	az előző pont;
Lefelé nyíl	a következő pont;
Balra nyíl	az előző pont;
Jobbra nyíl	a következő pont;
Home	a legelső pont;
End	a legutolsó pont;
PgUp	a menüpont kiválasztása;
PgDn	a menüpont kiválasztása;
Return	a menüpont kiválasztása;
Bármilyen betű	az első olyan menüpont kiválasztása, amely a lenyomott billentyűvel kezdődik;
Esc	megszakítás, amely nullát ad vissza.

Lásd még: @...PROMPT, SET MESSAGE, SET WRAP, ACHOICE( ).

NEXT

NEXT

A FOR ciklus záró utasítása.

Lásd még: FOR.

NOTE

NOTE <megjegyzéssor>

A programba nem végrehajtandó megjegyzéssort illeszt.

Lásd még: *, &&.

ON ERROR

ON ERROR

A Cipper egy felhasználói hibakezelő funkciórendszert alkalmaz helyette, csak dBASE kompatibilitása miatt említjük meg. Használatakor a fordító felismeri, és hibát jelez!

ON ESCAPE

ON ESCAPE

A Clipper a SET KEY-t alkalmazza helyette, csak dBASE kompatibilitása miatt említjük meg. Használatakor a fordító felismeri, és hibát jelez!

ON KEY

ON KEY

A Clipper a SET KEY-t alkalmazza helyette, csak dBASE kompatibilitása miatt említjük meg. Használatakor a fordító felismeri, és hibát jelez!

OTHERWISE

OTHERWISE

A többszörös elágazás – a DO CASE – „egyéb eset" ágának kezdetét jelzi. Az OTHERWISE és az ENDCASE közötti utasítások akkor hajtódnak végre, ha egyetlen CASE feltétel sem teljesül. Az utasítás el is hagyható. Ha most sem teljesül egyetlen CASE feltétel sem, a program futása az ENDCASE utasítás után folytatódik.

Lásd még: DO CASE.

PACK

PACK

Fizikailag törli a törlésre megjelölt adatbázisrekordokat. Hálózatban a törlés előtt zárolni kell az állományt!

Lásd még: DELETE, RECALL, REINDEX, SET DELETED, ZAP, DELETED( ).

PARAMETERS

PARAMETERS <memóriaváltozó-lista>

Egy Clipper nyelven írt rutinban azokat a változókat jelöli ki, amelyekben a megadott paramétereket tárolni fogja. Ha az átadott paraméter referencia típusú (tömbnév vagy @ volt előtte), akkor a változó minden módosítása az „ős" változóba is bekerül, különben nem! A tömbnév átadása után a tömbelemek már elérhetők. A megadott tömbre a rutin belsejében, a PARAMETERS utasításban megadott névvel hivatkozhat. A rutinok a tömböket mindig referenciaként kapják, a tömbelemeket, a változókat, a kifejezéseket és az adatbázismezőket pedig értékként.

Figyelem! A tömbelemeket csak értékként lehet átadni!

Ez a parancs lehetővé teszi azt is, hogy a kész program paramétereket vegyen át a DOS parancssorból. A parancssor paraméterlistája bármilyen karakterekből állhat. A paraméter a programnév utáni első nemszóközkarakterrel kezdődik, és minden paramétert szóközzel kell elválasztani. Az idézőjelek között megadott karakterláncokat – ha szóközkaraktert is tartalmaznak – a rendszer nem tudja kezelni. Minden rutinban csak egy PARAMETERS utasítás lehet.

A DOS parancssorból mindig karakteres típusú paramétert kap a program.

Lásd még: DO, PRIVATE, PROCEDURE, PUBLIC, SET PROCEDURE, PCOUNT( ).

PRIVATE

PRIVATE <memóriaváltozó-lista>

A felsorolt memóriaváltozókat csak ez a program és az ebből hívott alprogramok „ismerik". A „felsőbb" programok nem használják. Ha a hívó program ugyanazt a változót használja, akkor annak az értéke elmentődik, és amikor a hívó visszakapja a vezérlést, az érték visszaíródik.

Figyelem! A két változó típusa eltérhet egymástól, ez azonban nem okoz hibát. Ezzel az utasítással tömböket is definiálhat, amelyekre a változóknál elmondottak természetesen érvényesek. (Ezek a változók a veremben jönnek létre.) A tömbelemeket memóriaváltozóként használhatjuk. Csak akkor nem használhat tömbelemet memóriaváltozó helyett, ha makróval akar egy tömböt létrehozni!

Hibásak tehát a következő utasítások:

```
TOMBNEV=   'ELSO[2]'
DECLARE    &TOMBNEV
PUBLIC     &TOMBNEV
PRIVATE    &TOMBNEV
```

Érvényes viszont a következő forma:

```
TOMBNEV=   'ELSO'
DECLARE    &TOMBNEV[2]
PUBLIC     &TOMBNEV[2]
PRIVATE    &TOMBNEV[2]
```

Lásd még: DECLARE, PARAMETERS, PUBLIC.

PROCEDURE

PROCEDURE <eljárásnév>

Az eljárásfájlban a rutinok kezdetét határozza meg. Az eljárásnevek legfeljebb 10 karakteresek lehetnek, a tartalmuknak a DOS fájlnévkoncepcióinak kell megfelelniük. Érdemesebb inkább függvényszerű rutinokat írni, mivel a procedúrák (eljárások) – a függvényekkel ellentétben – nem adhatnak vissza értéket. Az eljárások egy programfájl (.PRG) végére is írhatók. A paramétereket kétféleképpen adhatja át. Az érték szerinti átadáskor a hívott rutin a paramétereket tetszés szerint módosíthatja. A másik lehetőség az, amikor a paraméter nem konkrét érték, hanem egy memóriacím, méghozzá annak a változónak a címe, amely az értéket tartalmazza. Ha a rutin bármit megvál-

toztat az értéken, az a változóban is módosulni fog! Természetesen így csak változókat lehet paraméterként átadni. Az ilyen típusú paraméterátadáskor a változó elé @ jelet kell tenni.

Lásd még: DO, SET PROCEDURE.

PUBLIC

PUBLIC <memóriaváltozó-lista>

A megadott memóriaváltozókat az összes rutin használhatja. Alapértelmezésben a fordító számára minden változó PRIVATE típusú. Tehát minden olyan modulban, amelyet nem azzal a rutinnal együtt fordít le, amelyben PUBLIC típusúnak definiálta a változót, de nem adta meg, hogy az PUBLIC változó, a változók PRIVATE típusúak lesznek! Természetesen máshol megadhat egy ugyanilyen nevű PRIVATE változót is, a kettő nem zavarja egymást. Ezzel az utasítással tömböket is definiálhat, ezek ugyanúgy fognak viselkedni, mint a többi változó. (Ezek a változók az adatszegmensben jönnek létre.)

A tömbelemeket memóriaváltozóként használhatja. Csak akkor nem használható tömbelem memóriaváltozó helyett, ha makróval akar egy tömböt létrehozni!

Hibásak tehát a következő utasítások:

```
TOMBNEV=   'ELSO[2]'
DECLARE    &TOMBNEV
PUBLIC     &TOMBNEV
PRIVATE    &TOMBNEV
```

Érvényes viszont a következő forma:

```
TOMBNEV=   'ELSO'
DECLARE    &TOMBNEV[2]
PUBLIC     &TOMBNEV[2]
PRIVATE    &TOMBNEV[2]
```

Van egy különleges PUBLIC változó, a CLIPPER. Ha ilyen névvel PUBLIC változót definiál, akkor az logikai változó lesz, *igaz* értékkel. A dBASE kompatibilitás rendszerének azt a sajátosságát használja ki ez a megoldás, hogy alapértelmezésben minden PUBLIC változó *hamis* értékű logikai változóként használható, egészen addig, amíg egy értékadással ezt meg nem változtatja. Ennek segítségével olyan „öszvér" programot lehet írni, amely nemcsak Clipper kódot tartalmaz, hanem pl. eredeti dBASE vagy FoxBase kódot is. (A FoxBase a FOX nevű PUBLIC változót használja erre a célra.)

Lásd még: DECLARE, PARAMETERS, PRIVATE.

QUIT

QUIT

Lezárja az összes fájlt, és kilép a Clipperből.

Lásd még: CANCEL, RETURN.

READ

READ [SAVE]

Az előzőleg kiadott @...GET utasításokkal kijelölt adatbekéréseket aktiválja. Ha a SAVE paramétert használja, akkor a következő READ újra aktiválja a kiadott GET parancsokat, különben ezek törlődnek, így nem keverednek össze az egymás utáni GET-READ utasítássorok. A Clipper – a dBASE-zel ellentétben – csak egyképernyős READ utasításokat képes használni.

A READ parancs kurzormozgató billentyűi:

Balra nyíl Ctrl-S	Egy karakterpozíciót balra lép, de nem lép át az előző GET mezőbe, ha eléri a mező határát.
Jobbra nyíl Ctrl-D	Egy karakterpozíciót jobbra lép, és átlép a következő GET mezőbe, ha eléri a mezőhatárt, és a SET Confirm OFF az érvényes beállítás
Ctrl-balra nyíl Ctrl-A	Az előző szó elejére áll.
Ctrl-jobbra nyíl Ctrl-F	A következő szó elejére áll.
Felfelé nyíl Ctrl-E	Az előző GET mező elejére áll.
Lefelé nyíl Ctrl-X Return Ctrl-M	A következő GET mező elejére áll.

Home	Az aktuális mező elejére áll.
End	Az aktuális mező végére áll.
Ctrl-Home	A legelső GET mező elejére áll.
Ctrl-End	A legutolsó GET mező elejére áll.

READ parancs szerkesztőbillentyűi:

Del Ctrl-G	Törli a kurzor pozíciójában lévő karaktereket.
Backspace Ctrl-H	Törli a kurzor előtti karaktert.
Ctrl-T	Törli a jobbra lévő szót.
Ctrl-Y	A kurzortól a mező végéig töröl.
Ctrl-U	Visszaállítja a mező eredeti tartalmát.
Ins Ctrl-V	Átvált beszúrás üzemmódba.
Ctrl-W Ctrl-C PgUp PgDn Return Ctrl-M	Befejezi a szerkesztést, és elmenti a változtatásokat.
Esc	Befejezi a szerkesztést, de nem menti el a változtatásokat.

Lásd még: @..GET, CLEAR GETS, SET FORMAT, LASTKEY( ).

RECALL

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

A törlésre kijelölt rekordokat „visszahozza", azaz megszünteti a kijelölést. Az alapértelmezés az ALL. Hálózatban való használata előtt az állományt a FLOCK( ) függvénnyel le kell foglalni.

Lásd még: DELETE, PACK, SET DELETED, ZAP, DELETED( ).

REINDEX

REINDEX

Újra felépíti a nyitott indexfájlokat. Hálózatban való használata előtt zárolni kell az állományt! Használata után a rekordmutató az első rekordra mutat. Nem működik helyesen a program akkor, ha a kulcsmezőn valamilyen változtatást végzett, pl. növelte a hosszát. Ilyenkor az INDEX paranccsal újra kell indexelni az állományt. Nincs szükség a használatára akkor, ha az indexállományok minden módosítás közben nyitva voltak, mert a Clipper automatikusan aktualizálja az indexállományokat.

Lásd még: INDEX, PACK, SET INDEX, SET UNIQUE, USE.

RELEASE

RELEASE [<memóriaváltozó-lista> | [ALL [LIKE | EXCEPT ;
<memóriaváltozó>]]

Törli az aktuális memóriaváltozókat. Azokat nem, amelyeket valamiért „nem lát". (Ha pl. egy változó a legfelső szinten van, és egy alatta lévő szinten PRIVATE-ként egy ugyanolyan nevű változót definiált, majd ezt törli, akkor a felette lévő változó megmarad.)

A LIKE paraméter a megadott azonosítójú változókat törli. Az EXCEPT paraméter azokat hagyja meg, amelyeket kijelölt. A többszörös kijelölést a DOS szűrőkaraktereivel (?, *) végezheti el. Ha csak az ALL opciót használja, akkor csak az aktuális rutinban lévő változók törlődnek.

Lásd még: CLEAR ALL, CLEAR MEMORY, QUIT.

RENAME

RENAME <régi fájlnév> | (<változónév1>) ;
TO <új fájlnév> | (<változónév2>)

Új nevet ad egy fájlnak. A teljes fájlneveket kell megadni, nyitott állomány nem nevezhető át!

Figyelem! Ha egy adatbázisállományt nevezett át, akkor a hozzá tartozó memofájlt is át kell nevezni!

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: COPY FILE, DELETE FILE, ERASE, RUN, FILE( ).

REPLACE

REPLACE [<érvényességi kör>] <mező1> WITH <kifejezés1> ;
[<mező2> WITH <kifejezés2>...] [FOR <feltétel1>] ;
[WHILE <feltétel2>]

Az adatmezők tartalmát a megadott értékre változtatja. Hálózatban való használata előtt az állományt az FLOCK( ) függvénnyel le kell foglalni. Az alapértelmezés az aktuális rekord.

Lásd még: APPEND, JOIN, UPDATE, STRTRAN( ).

REPORT

REPORT FORM <fájlnév1> | (<változónév1>) ;
[<érvényességi kör>] [FOR <feltétel1>] ;
[WHILE <feltétel2>] [TO PRINT] [TO FILE <fájlnév2> | ;
(<változónév2>)] [SUMMARY] [PLAIN] ;
[HEADING <karakterlánc>] [NOEJECT]

Adatok előre definiált formájú kiiratását teszi lehetővé. A PLAIN paraméter hatására a fejléc csak az első oldalon jelenik meg, nincs a fejlécben dátum és oldalszám. A HEADING paraméterben megadott *karakterlánc*-ot fejlécként írja ki. A kiírást azelőtt végzi el, hogy a rekordmutatót elmozdítaná a következő rekordra! A NOEJECT hatására a nyomtatás előtt nem dob lapot. A SUMMARY hatására csak összegsorokat ír.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal. Hálózatban a REPORT FORM ... közös használatra nyitja meg az inputfájlt.

Lásd még: LABEL FORM, LIST.

RESTORE FROM

RESTORE FROM <fájlnév> | (<változónév>) [ADDITIVE]

Visszahozza a megadott állományból az elmentett memóriaváltozókat. Az alapértelmezés a .mem fájl. Ha nem adja meg az ADDITIVE paramétert, akkor visszatöltés előtt törlődnek a változók. A változókat PRIVATE típusúnak kell definiálni. Hálózatban a RESTORE FROM ... közös használatra nyitja meg az inputfájlt.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: RESTORE SCREEN, SAVE, SAVE SCREEN.

RESTORE SCREEN

RESTORE SCREEN [FROM <memóriaváltozó>]

Visszaírja a SAVE SCREEN paranccsal elmentett képernyőtartalmat. Ez a megoldás gyorsabb, mintha a programozó jegyezné meg, hogy milyen képernyőt kell újrarajzolnia. Nem használhatja viszont a SAVESCREEN( ) függvénnyel elmentett képernyőket.

Lásd még: SAVE SCREEN, RESTSCREEN( ), SAVESCREEN( ).

RETURN

RETURN

Befejez egy programot vagy egy procedúrát, és törli az abban létrehozott PRIVATE változókat. Mindig ez a logikailag utolsó végrehajtható sor a procedúrában.

Lásd még: CANCEL, PRIVATE, PROCEDURE, PUBLIC.

RETURN

RETURN <kifejezés>

A függvények befejező parancsa, amely törli az azokban létrehozott PRIVATE változókat. A függvény a *kifejezés* értékét fogja „visszaadni".

Lásd még: CANCEL, FUNCTION, PRIVATE, PUBLIC.

RUN

RUN <parancs> | (<változónév>)

Lehetővé tesz egy MS-DOS parancs kiadását a Clipperből. Memóriarezidens programot *ne* indítson el!

Különleges eset:

RUN COMMAND

Ilyenkor a DOS indul el, és bármilyen parancsot kiadhat, ha van még elegendő memória. A saját programba a DOS EXIT utasításával térhet vissza!

Példa:

RUN FORMAT A: /S /V

Az MS-DOS lemet előkészítő programjának elindítása.

Lásd még: !, INDEX, PACK, SET INDEX, SET UNIQUE, USE.

SAVE

SAVE TO <fájlnév> | (<változónév1>) ;
[ALL [LIKE | EXCEPT <változónév2>]]

Az aktuális memóriaváltozókat egy memóriafájlba másolja. A LIKE paraméter a megadott azonosítójú változókat menti. Az EXCEPT paraméter azokat nem menti, amelyeket kijelölt. A többszörös kijelölést a DOS szűrőkaraktereivel (?, *) végezheti

el. Hálózatban a SAVE kizárólagos használatra nyitja meg a fájlt. Tömböket nem lehet így kimenteni! Az alapértelmezés a .memfájl.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: RESTORE, RESTORE SCREEN, SAVE SCREEN.

SAVE SCREEN

SAVE SCREEN [TO <memóriaváltozó>]

Az aktuális képernyőt egy karakteres memóriaváltozó-ba írja, amely akár tömbelem is lehet.

Lásd még: RESTORE SCREEN, SAVE.

SEEK

SEEK <kifejezés>

Az aktuális indexelt adatbázisban megkeresi az első olyan rekordot, amelynek kulcsa megegyezik a *kifejezés* értékével. Ha talált, akkor a FOUND( ) függvény *igaz* értéket ad vissza.

Lásd még: FIND, INDEX, LOCATE, SET DELETED, SET EXACT, SET INDEX, SET SOFTSEEK, USE, EOF( ), FOUND( ), RECNO( ).

SELECT

SELECT <munkaterület sorszám> | <ALIAS név>

A megadott munkaterületet választja ki az aktuálisnak. A munkaterület sorszáma helyett egy ALIAS *név* is megadható.

Ha a 0. munkaterületet adjuk meg, akkor a következő szabad munkaterületre áll.

Lásd még: SET INDEX, USE, ALIAS( ), SELECT( ).

SET ALTERNATE

SET ALTERNATE TO [<fájlnév> | (<változónév>)]

Minden, ami a képernyőre kiíródik, ebbe a fájlba is bekerül. Az alapértelmezés a .txt kiterjesztés. Hálózatban az ALTERNATE kizárólagos használatra nyitja meg a fájlt. Az állomány előző tartalma elveszik, a @... utasítások eredményei nem kerülnek a fájlba! (Erre a SET PRINTER TO és S SET DEVICE TO utasítások használhatók.) A *fájlnév* nélkül kiadott utasítás lezárja az előzőleg megnyitott állományt.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, DISPLAY TO FILE, LABEL FROM TO FILE, LIST TO FILE, REPORT FROM TO FILE, SET ALTERNATE, SET PRINTER TO, TEXT TO FILE, TYPE TO FILE, FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

SET ALTERNATE

SET ALTERNATE ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására az előzőleg megnyitott (SET ALTERNATE TO) állományba is kiírja a képernyőre kerülő adatokat, kivéve azokat, amelyeket a @... utasításokkal már kiírt. Az OFF paraméter csak felfüggeszti a kiírásokat, de nem zárja le a fájlt. Az alapértelmezés az OFF.

Lásd még :CLOSE, DISPLAY TO FILE, LABEL FORM TO FILE, LIST TO FILE, REPORT, REPORT FORM TO FILE, SET ALTERNATE TO, SET PRINTER TO, TEXT TO FILE, TYPE TO FILE, FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

SET BELL

SET BELL ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására az adatbekérő utasításokban a mező feltöltése után egy rövid, éles hangot ad a program.

Az alapértelmezés az OFF.

Lásd még: SET CONFIRM, CHR( ).

SET CARRY

SET CARRY ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET CENTURY

SET CENTURY ON | OFF | (<logikai kifejezés>)

Az évszázadot jelzi ki a dátumokban. Az érvényes dátumforma: 0100.01.01-től 2999.12.31-ig. Az alapértelmezés az OFF.

Lásd még: SET DATE, CTOD( ), DATE( ), DTOC( ), DTOS( ), YEAR( ).

SET CLIPPER=

SET CLIPPER= [Vnnn;] [Rnnn;] [Ennn;] [Xnnn;] [Fnnn;] [;Sn]

Beállítja a kész felhasználói program DOS környezetét. Nem végrehajtható Clipper utasítás, DOS parancsként kell használni. Minden Clipper nyelven írt program a következő memóriafelosztást használja:

A közös munkaterület mérete 24 kbájt. A fennmaradó tárterület 20 %-át (legfeljebb 44 kbájt) a változónak, 33 %-át (de legkevesebb 16 kbájt) pedig az indexpufferek-nek és a külső programoknak foglalja le, az esetleg megmaradó területet a közös munkaterülethez kerül.

Ezen a felosztáson a következőképpen változtathat:

V A változók területe
Minden változó 22 bájtot foglal el. *Figyelem!* Minden tömbelem egy változónak számít. A karakterlánc típusúak a közös munkaterületen dinamikusan helyezkedik el.
Példa: területmegadás 100 változó számra
(100*22)/1024 = 2.1484
SET CLIPPER= V003;

R A külső programok és az indexpufferek területe

E Helyfoglalás a memóriabővítésben
A Clipper az Intel/Lotus szerinti „above board" bővítést támogatja. A paramétert legalább 016-ra kell beállítani ahhoz, hogy a Clipper használni tudja. Ekkor az indexpufferek ide helyeződnek át.

X A memória egy részének elrejtése a Clipper elől
Az itt megadott méretű memóriát a Clipper alkalmazói program nem használja. Így lehet kipróbálni, hogyan viselkedik majd a program egy kisebb memóriájú gépben. Ez az elzárás csak a Clipper programokra érvényes, ha egy külső programot indít el, akkor az ehhez az elzárt részhez is hozzáfér!

F A program számára nyitva tartható állományok max. száma
A DOS 3.30-as változatától kezdve már 255 az egyszerre nyitva tartható állományok száma.

S A 0 érték megadásakor gyors lesz a képernyőkezelés, ez az alapértelmezés is. Ez a képernyőhasználat néhány színes képernyőn „hóeséshez" hasonló vibrálást okoz, amely az 1 érték megadásával elkerülhető, de lassabbá teszi a képernyőkezelést.

SET COLOR

SET COLOR TO [<normál> [,<kiemelt>] [,<keret>] [,<háttér>];
[,<nem kiválasztott>] | (<karakteres kifejezés>)]

Beállítja a programban használt színeket. Minden színt két kód határoz meg. Elsőként azt a színkódot kell megadni, amivel írni fog, másodiknak azt, hogy milyen háttérre. A kódokat perjellel (/) kell elválasztani.
A paraméterek jelentése a következő:

normál	Ezt a színt használja minden adatkiíró utasítás.
kiemelt	Ezt a színt használják a @...GET utasítások az aktuális mező jelzésére.
keret	A képernyő keretének színe. A rendszer csak az első kódot veszi figyelembe, nem használható a háttérként megadott kód.
háttér	Jelenleg nem használja a Clipper (egy későbbi fejlesztés számára lefoglalt).
nem kiválasztott	Ezt a színt használja a @...GET utasítás azoknak a mezőknek a jelzésére, amelyek nem aktuálisak. Ha ezt a para-

métert elhagyja, akkor a rendszer csak a *kiemelt* színnel dolgozik.

A SET COLOR utasítást a Clipper figyelmen kívül hagyja, ha a programhoz szerkeszti az ANSI.OBJ modult.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Ha a színkód után „*"-ot ír, akkor az a szín villogni fog.

A Clipperben nemcsak színkódot használhat, hanem színszámot is.

Szín	Szám	Betű
Fekete	0	N
Sötétkék	1	B
Sötétzöld	2	G
Sötét cián	3	BG
Sötétvörös	4	R
Sötétlila	5	RB
Barna	6	GR
Szürke	7	W
Szürke	8	N+
Világoskék	9	B+
Világoszöld	10	G+
Világos cián	11	BG+
Világosvörös	12	R+
Világoslila	13	RB+
Sárga	14	GR+
Fehér	15	W+

csak monokrom képernyőn

Aláhúzott	U
Nem látható	X
Inverz	I

\+ fényes

Lásd még: SET COLOUR, ISCOLOR( ), SETCOLOR( ).

SET COLOUR

SET COLOUR TO [<normál> [,<kiemelt>] [,<keret>] [,<háttér>];
[,<nem kiválasztott] |(<karakteres kifejezés>)]

Beállítja a programban használt színeket. Minden színt két kód határoz meg. Elsőként azt a színkódot kell megadni, amivel írni fog, másodiknak azt, hogy milyen háttérre. A kódokat perjellel (/) kell elválasztani.

A paraméterek jelentése a következő:

normál	Ezt a színt használja minden adatkiíró utasítás.
kiemelt	Ezt a színt használják a @...GET utasítások az aktuális mező jelzésére.
keret	A képernyő keretének színe. A rendszer csak az első kódot veszi figyelembe, nem használható a háttérként megadott kód.
háttér	Jelenleg nem használja a Clipper (egy későbbi fejlesztés számára lefoglalt).
nem kiválasztott	Ezt a színt használja a @...GET utasítás azoknak a mezőknek a jelzésére, amelyek nem aktuálisak. Ha ezt a paramétert elhagyja, akkor a rendszer csak a *kiemelt* színnel dolgozik.

A SET COLOUR utasítást a Clipper figyelmen kívül hagyja, ha a programhoz szerkeszti az ANSI.OBJ modult.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Ha a színkód után „*”-ot ír, akkor az a szín villogni fog.

A Clipperben nemcsak színkódokat használhat, hanem színszámokat is.

Szín	Szám	Betű
Fekete	0	N
Sötétkék	1	B
Sötétzöld	2	G
Sötét cián	3	BG
Sötétvörös	4	R
Sötétlila	5	RB
Barna	6	GR
Szürke	7	W
Szürke	8	N+
Világoskék	9	B+
Világoszöld	10	G+
Világos cián	11	BG+
Világosvörös	12	R+
Világoslila	13	RB+
Sárga	14	GR+
Fehér	15	W+

csak monokrom képernyőn

Aláhúzott	U
Nem látható	X
Inverz	I

\+ fényes

Lásd még: SET COLOR, ISCOLOR( ), SETCOLOR( ).

SET CONFIRM

SET CONFIRM ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására a @...GET parancs csak a kilépőbillentyűk lenyomása után fejezi be a szerkesztést.

A kilépőbillentyűk a következők:

Ctrl-C,
PgUp,
Ctrl-W,
PgDn,
Esc,
Return.

Lásd még: @...GET, READ, SET BELL.

SET CONSOLE

SET CONSOLE ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására az ACCEPT, INPUT, WAIT, ?, ??, xxxx TO PRINT parancsokban az „eredményt" a képernyőre írja.
A teljes képernyős parancsok (@..., CLEAR SCREEN) ettől az utasítástól függetlenül is a képernyőre fognak írni.

Lásd még: SET DEVICE.

SET CURSOR

SET CURSOR ON | OFF | (<logikai kifejezés>)

A kurzor jelenlétét engedélyezi vagy tíltja. Az alapértelmezés az ON.

Lásd még: SET CONSOLE.

SET DATE

SET DATE AMERICAN | ANSI | BRITISH | FRENCH | GERMAN | ; ITALIAN

A dátumkiíratás formáját határozza meg. Az alapértelmezés az AMERICAN.

AMERICAN	HH/NN/ÉÉ
ANSI	ÉÉ.HH.NN
BRITISH	NN/HH/ÉÉ
FRENCH	NN/HH/ÉÉ
GERMAN	NN.HH.ÉÉ
ITALIAN	NN-HH-ÉÉ

Lásd még: SET CENTURY, CTOD( ), DATE( ), DTOC( ).

SET DEBUG

SET DEBUG ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismer, nem jelez hibát, de a parancsnak nincs hatása.

SET DECIMALS

SET DECIMALS TO <numerikus kifejezés>

Beállítja a tizedesjegyek számát a kiíratási műveletek számára.

Lásd még: SET FIXED.

SET DEFAULT

SET DEFAULT TO [<meghajtó>[<:elérési útvonal>]] | [(<változónév>)]

Meghatározza az aktuális *meghajtó*-t a lemezműveletek számára. Nem kell megadni a „:"-ot, de megadható az aktuális könyvtárnév. Paraméter nélkül az lesz az aktuális meghajtó és könyvtár, ahonnan a programot elindította.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: SET PATH.

SET DELETED

SET DELETED ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására elrejti a törlésre kijelölt rekordokat, ilyenkor a RECALL ALL is hatástalan. Az INDEX és a REINDEX – a beállítástól függetlenül – mindig feldolgozza az összes rekordot. Ha a sorszáma alapján közvetlenül rááll egy rekordra, akkor a rekord az ON ellenére feldolgozható. Az alapértelmezés az OFF.

Lásd még: DELETE, INDEX, RECALL, SET INDEX USE, DELETED( ).

SET DELIMITERS

SET DELIMITERS ON | OFF | (<logikai kifejezés>)

Az On hatására engedélyezi a képernyős megjelenítésekben a mezőhatároló jelek használatát. Az OFF hatására a határolójelek nem jelennek meg. Az alapértelmezés az OFF.

Lásd még: @...GET.

SET DELIMITERS

SET DELIMITERS TO [<karakteres kifejezés> | DEFAULT]

A mezőket határoló karaktereket adhatja meg vele. Az első karakter a mező elejét, a második a mező végét jelzi. A DEFAULT hatására a mezők elején és végén kettőspont jelenik meg. Ezek a határolójelek nem kerülnek be az adatbázisba.

Lásd még: @...GET.

SET DEVICE

SET DEVICE TO SCREEN | PRINT

A @ parancs eredményeit a képernyőre vagy a nyomtatóra küldi. A PRINT paramétert használva a SET PRINTER-rel meghatározott eszközre küldi az adatokat. Az alapértelmezés a SCREEN.

Lásd még: @..SAY, EJECT, SET PRINTER TO, PCOL( ), PROW( ), SETPRC( ).

SET ECHO

SET ECHO ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET ESCAPE

SET ESCAPE ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására az Alt-C billentyűk együttes lenyomásakor megszakad a program végrehajtása. Ekkor a rendszer felteszi a kérdést: Be akarja-e fejezni a programfutást vagy folytatja tovább (Continue?). Ha Y-t válaszol, akkor folytatódik a program. Minden más billentyű hatására befejeződik a működés. A parancs nemcsak az előbb leírt esetet szabályozza. Az ON paraméter hatására a @...GET parancsból az Esc billentyűvel is kiléphet, különben nem. Az alapértelmezés az ON.

Lásd még: @...GET, CANCEL, QUIT, READ, RETURN.

SET EXACT

SET EXACT ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására a karakteres összehasonlításban először levágja az operandusokról a határoló <space> karaktereket és csak ezután hasonlít. Az alapértelmezés az OFF.

Példa:

```
SET EXACT OFF
? „abc" = „abc"          && Eredménye .F.
? „abc" = „abc"          && Eredménye .T.

SET EXACT ON
? „abc" = „abc"          && Eredménye .T.
? „abc" = „abc"          && Eredménye .T.
```

Lásd még: DISPLAY, FIND, LIST, LOCATE, SEEK, = =.

SET EXCLUSIVE

SET EXCLUSIVE ON | OFF | (<logikai kifejezés>)

Az ezután megnyitott adatbázis-állományok (.dbf, .dbt, .ntx, .ndx) kizárólagos (ON) vagy közös (OFF) használatúak lesznek. Az alapértelmezés az ON.

Lásd még: UNLOCK, USE...EXCLUSIVE, FLOCK( ), RLOCK( ).

SET FILTER

SET FILTER TO [<feltétel>]

Az adatbázist úgy tünteti fel, mintha az csak olyan rekordokat tartalmazna, amelyek kielégítik a megadott *feltétel*-t. Minden munkaterülethez tartozhat egy ilyen szűrő. Az INDEX és a REINDEX – a *feltétel*-től függetlenül – mindig feldolgozza az összes rekordot. Kijelölés után ki kell adni egy rekordmutatót állító parancsot (pl. GO TOP), hogy a Clipper a *feltétel*-nek megfelelő rekordra álljon.

Lásd még: SET DELETED.

SET FIXED

SET FIXED ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására a SET DECIMALS paranccsal meghatározott számú tizedesjegyet ír ki. Az alapértelmezés az OFF.

Lásd még: SET DECIMALS, EXP( ), LOG( ), SQRT( ), VAL( ).

SET FORMAT

SET FORMAT TO <fájlnév>

Formátumfájlt nyit meg az adatbegépeléshez. A *fájlnév* .fmt vagy .prg kiterjesztésű lehet. Ha .clp állománnyal fordít, de nem ad meg kiterjesztést, akkor a fájlnak .prg kiterjesztésűnek kell lennie. Ezután minden READ utasítás az ebben az állományban leírt @... utasításokat is végrehajtja. A Clipper nem törli a képernyőt, ha egy formátumállományból hajt végre parancsokat.

Lásd még: @...GET, @...SAY, READ

SET FUNCTION

SET FUNCTION <numerikus kifejezés> TO <karakterlánc>

A 39 (2-40) funkcióbillentyűhöz lehet tetszőleges *karakterlánc*-ot rendelni. A *karakterlánc*-ban a ";" karakter a Return billentyűt jelöli, de tartalmazhat vezérlőkaraktereket is. Ha egy funkcióbillentyűhöz a SET FUNCTION és a SET KEY paranccsal is hozzárendelt valamit, akkor a Set KEY parancs lesz az „erősebb". Az F1 billentyűt ebben a parancsban nem használhatja, mert alapértelmezésben a Clipper a HELP rutint rendeli hozzá.

A 1-10-es funkcióbillentyűk a megfelelő funkcióbillentyűk (F1-F10-ig) lenyomásával aktiválhatók.

A 11-20-as funkcióbillentyűk a megfelelő funkcióbillentyűk (F1-F10-ig) és a SHIFT billentyű lenyomásával aktiválhatók.

A 21-30-as funkcióbillentyűk a megfelelő funkcióbillentyűk (F1-F10-ig) és a Ctrl (control) billentyű lenyomásával aktiválhatók.

A 31-40-es funkcióbillentyűk a megfelelő funkcióbillentyűk (F1-F10-ig) és az Alt billentyű lenyomásával aktiválhatók.

Lásd még: SET KEY

SET HEADING

SET HEADING ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET INDEXES

SET INDEXES TO [<indexfájlnévlista> | (<változónévlista>)]

Megnyitja a megadott indexfájlt. Az indexfájl használati módját (közös vagy kizárólagos) a hozzá tartozó állomány határozza meg. Ha változóneveket használ, akkor minden egyes állománynevet külön változóban kell tárolni.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, INDEX, REINDEX, SET ORDER, USE.

SET INTENSITY

SET INTENSITY ON | OFF | (<logikai kifejezés>)

Az ON paraméter hatására a @...GET parancs a színbeállításban megadott <kiemelt> és <nem kiválasztott> színeket használja, különben csak a <normál> színt.

Lásd még: @..GET, @...SAY, SET COLOR, SET COLOUR, SETCOLOR( ).

SET KEY

SET KEY <numerikus kifejezés> TO [<rutinnév>]

A megadott kódú billentyűhöz egy rutint rendel. A billentyűkód megegyezik az INKEY( ) függvény által visszaadott értékkel. Ha egy billentyűhöz a SET FUNCTION és a SET KEY paranccsal is hozzárendel valamit, akkor a SET KEY a magasabb prioritású. A Clipper az induláskor automatikusan elvégzi a következő hozzárendelést:

SET KEY 28 TO help

Általában a következő utasítások közben hívható a billentyűhöz rendelt rutin:

WAIT,
READ,
ACCEPT,
INPUT,
MENU TO.

A rutin három paramétert kap:

- a hívó rutin nevét;
- a hívó rutin aktuális sorszámát és
- annak a változónak a nevét, amelyet éppen kitölt.

Lásd még: CLEAR SCREEN, KEYBOARD, SET FUNCTION, LASTKEY( ), NEXTKEY( ).

SET MARGIN

SET MARGIN TO <numerikus kifejezés>

Beállítja a bal oldali margót a nyomtatón. Az alapértelmezés a 0.

Lásd még: @...GET, @...SAY, SET DEVICE, SET PRINT.

SET MENUS

SET MENUS ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET MESSAGE

SET MESSAGE TO [<numerikus kifejezés>] [CENTER | CENTRE]

A menügeneráló utasítások közé tartozik. Beállíthatja vele, hogy az egyes menüpontokhoz tartozó bővebb magyarázat melyik sorban jelenjen meg. A sorokat itt 1-től 24-ig számozhatja. A CENTER vagy a CENTRE paraméter a 0. oszlophoz képest középre igazítja a szöveget. Ha nem ad meg értéket, vagy 0-t ad meg, akkor az üzenetek nem kerülnek a képernyőre.

Lásd még: @...PROMPT, MENU, SET WRAP.

SET ORDER

SET ORDER TO [<numerikus kifejezés>]

A megnyitott indexfájlok közül kijelöli az aktuálisat. A 0 paraméter azt eredményei, mintha az állomány nem lenne indexelve.

Példa:

```
SET INDEX TO NEV,CIM,SZDAT
.
.
.
SET ORDER 3
* Eredménye: az SZDAT állomány lesz az aktuális
* Kulcsállomány.
```

Lásd még: INDEX, REINDEX, SET INDEX, USE, INDEXEXT( ), INDEXKEY( ), INDEXORD( ).

SET PATH

SET PATH TO [<elérési út> | (<változónév>)]

Megadhatók azok a könyvtárak, amelyekben a program az adatállományokat megkeresheti, ha az aktuálisban nem találja meg. Természetesen akár másik lemezen lévő könyvtárakat is megadhat. Ez az egyetlen eset, amikor a sorelválasztó pontosvessző nem használható! Az elemeket vesszővel és pontosvesszővel is el lehet választani.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

```
SET PATH TO C:\MUNKA\ADATOK1,;                    &&nem jó
C:\MUNKA\ADATOK2
SET PATH TO C:\MUNKA\ADATOK1;C:\MUNKA\ADATOK2 &&helyes
```

Lásd még: DIR, SET DEFAULT, FILE( ).

SET PRINTER

SET PRINTER ON | OFF | (<logikai kifejezés>)

Az On paraméter hatására a nyomtatóra küldött adatokat a SET PRINTER TO utasítással meghatározott egységre küldi. Különben „elnyeli" az adatokat.

Lásd még: EJECT, SET CONSOLE, SET DEVICE, SET PRINTER.

SET PRINTER

SET PRINTER TO [<eszköznév> | <fájlnév> | (<karakteres kifejezés>)]

A nyomtatóra írt szöveget átirányítja. A *fájlnév* alapértelmezése a .prn. Ha olyan eszközre akar írni, amelyik nem érhető el, akkor az *eszköznév*-et *fájlnév*-nek tekinti. Az eszköz alapértelmezése a PRN.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: @...SAY, SET DEVICE, SET PRINT.

SET PROCEDURE

SET PROCEDURE TO [<eljárásfájlnév>]

Fordításkor megnyitja a megnevezett eljárásfájlt. A kiterjesztés alapértelmezése a .prg. Futáskor figyelmen kívül marad ez a parancs, ugyanis a Clipper fordító a programba befordítja az így megadott procedúraállományt.

Lásd még: DO, FUNCTION, PROCEDURE, RETURN.

SET RELATION

SET RELATION [ADDITIVE] TO<kulcskifejezés1> | RECNO( ) | ;
<numerikus kifejezés1> INTO <név> ;
[, TO <kulcskifejezés2> | RECNO( ) | ;
<numerikus kifejezés2> INTO <név>...]

A kulcskifejezésnek megfelelően több adatbázist fűz össze. A második stb. állománynak a kifejezés szerint kell indexelve lennie. Ha az első állományban mozog (pl. keres), akkor a hozzá kapcsolódó állományokban is megtörténik a keresés. Az ADDITIVE paraméter hatására a parancs nem „rombolja le" az eddig felépített összeköttetéseket, hanem az újat is a régihez kapcsolja.

Lásd még: INDEX, REPLACE, SET INDEX, SET ORDER, UPDATE, USE.

SET SAFETY

SET SAFETY ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET SCOREBOARD

SET SCOREBOARD ON | OFF | (<logikai kifejezés>)

A 0. sorban megjelenő Clipper üzeneteket engedélyez, ill. letiltja, a READ és a MEMOEDIT( ) végrehajtása közben. Az Insert és a Range is hibaüzenetnek számít.

Lásd még: @...GET, READ, MEMOEDIT( ).

SET SOFTSEEK '87

SET SOFTSEEK ON | OFF | (<logikai kifejezés>)

Az alapértelmezés az OFF. Ha az ON állapotban SEEK, FIND vagy LOCATE utasítást ad ki, és nem talál egyező kulcsú rekordot, akkor a FOUND( ) és az EOF( ) függvény *hamis* értéket ad vissza. A rekordmutató arra a rekordra fog mutatni, amelyik a megadott kulcsnál nagyobb ugyan, de a legkisebb köztük az eltérés. Ilyenkor az EOF( ) függvény csak akkor ad *igazi* értéket, ha nincs a megadottnál nagyobb kulcsú rekord. Ha az OFF állapotban nem talál, akkor a rekordmutató az utolsó utáni rekordra mutat, az EOF( ) függvény *igaz* lesz, a FOUND( ) pedig *hamis*. A SET EXACT nincs hatással erre az utasításra. A SET RELATION nem veszi figyelembe a SOFTSEEK állapotát és OFF-ként viselkedik.

Lásd még: INDEX, SEEK, SET EXACT, SET RELATION.

SET STATUS

SET STATUS ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET STEP

SET STEP ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET TALK

SET TALK ON | OFF

Csak dBASE kompatibilitása miatt említjük meg. A fordító felismeri, nem jelez hibát, de a parancsnak nincs hatása.

SET TYPEAHEAD

SET TYPEAHEAD TO <pufferméret>

A billentyűzetpuffer méretét adhatja meg vele, ez a szám 0-tól 32768-ig terjedhet. Ha a puffer méretét 0-nak adja meg, akkor ezzel letiltja az Alt-C és az Alt-D billentyűk használatát, ez viszont nagyon gyors billentyűzetkezelést okoz.

Lásd még: ACCEPT, INPUT, KEYBOARD, READ, SET KEY, LASTKEY( ), NEXTKEY( ).

SET UNIQUE

SET UNIQUE ON | OFF | (<logikai kifejezés>)

Az azonos kulcsú rekordok közül az első (ON) vagy az összes (OFF) kerül be az aktuális indexállományba. Az alapértelmezés az OFF.

Lásd még: FIND, INDEX, REINDEX, SEEK, SET INDEX, USE.

SET WRAP

SET WRAP ON | OFF | (<logikai kifejezés>)

Az On paraméter hatására a menükezelő utasításban (MENU) a menüpontok „végtelenítését" okozza. Az alapértelmezés az OFF.

Lásd még: @...PROMPT, MENU TO, SET MESSAGE, ACHOICE( ).

SKIP

SKIP <numerikus kifejezés> [ALIAS <szám> | <aliasnév> | ;
(<változónév>)]

Az aktuális rekordtól kezdve a *numerikus kifejezés*-nek megfelelően lépteti a rekordmutatót. Negatív érték hatására visszafelé lép, ha van nyitott indexállomány, akkor indexsorrendben lép vissza. A SKIP 0 a memóriabeli állománypuffert felírja a lemezre, de az indexpuffereket nem! Használja helyette a COMMIT parancsot!

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: COMMIT, CONTINUE, FIND, GO, GOTO, LOCATE, SEEK, BOF( ), EOF( ), RECNO( ).

SORT

SORT [<érvényességi kör>] TO [<új fájl> | (<változónév>)] ;
ON <mező1> [/A | /D] [/C] ;
[,<mező2> [/A| [/D] [/C] ... ;
[FOR <feltétel1>] [WHILE <feltétel2>]

Új adatbázist hoz létre, amelyben az aktuális adatbázis rekordjait az ön által megadott mezők és feltételek újrarendezik. A /A növekvő, a /D pedig csökkenő sorrendbe rendez. Az alapértelmezés a /A. A /C hatására nem tesz különbséget a kisbetűk és a nagybetűk között. A karakteres mezőket ASCII sorrendbe, a numerikus mezőket nagyság szerint, a dátummezőket pedig idő sorrendbe rendezi. Logikai és memomezők szerint nem tud rendezni.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Ha lehet, használjon inkább indexállományokat! Az utasítás három fájlhandle-t foglal le, ezek közül a harmadik egy átmeneti állomány, az ún. Clipsort. tmp.

Hálózatban a SORT kizárólagos használatra nyitja meg a fájlt.

Lásd még: INDEX, ASORT( ).

STORE

STORE <kifejezés> TO <memóriaváltozó-lista>

Egy vagy több memóriaváltozónak értéket ad, ha kell, létre is hozza azokat. A változók típusa a *kifejezés* típusától függ. A változók típusának alapértelmezése a PRIVATE, hacsak nem definiálta már másnak.

Figyelem! Ez az értékadás nem használható adatbázismezők feltöltésére; még akkor sem, ha kiírja az ALIAS nevet a változók neve elé.

Nem a kívánt eredményt kapja, ha a parancsot a következő formában használja:

```
USE PROBA
STORE 'ÉRTÉKE' TO PROBA->MEZO
```

Ilyenkor létrehoz egy MEZO nevű PRIVATE változót!

Egy változó típusa értékadással megváltoztatható, nem okoz hibát.

```
STORE '12.45'        TO VALT      && Karakteres típusú.
STORE VAL(VALT)      TO VALT      && Numerikus típusú.
```

Lásd még: CLEAR MEMORY, PRIVATE, PUBLIC, RELEASE, RESTORE, SAVE.

SUM

SUM [<érvényességi kör>] <mezőlista> TO ;
<memóriaváltozó-lista> [FOR <feltétel1>] ;
[While <feltétel2>]

A megadott mezők tartalmát összegzi az aktív adatbázisban. Az alapértelmezés az ALL. Nemcsak mezőket adhat meg, hanem bármilyen numerikus kifejezést is.

Példa:

```
*
* Kiszámítja az ELADAR és a NYILVAR összegeit.
* Az ELADAR összegzését az ELADAS, a NYILVAR összegzését
* pedig az ERTEK fogja tartalmazni.
*

SUM ELADAR,NYILVAR TO ELADAS,ERTEK FOR VEVO='Kovács'
```

Lásd még: AVERAGE, TOTAL.

TEXT

TEXT [TO PRINT | FILE <fájlnév> | (<változónév>)]
.
.
.
ENDTEXT

Kiír egy szövegtömböt a képernyőre, és ha szükséges, megadja egy másik outputra is. A *fájlnév* kiterjesztésének alapértelmezése a .txt. Ha a szövegben makrót talál, akkor a makró kifejtését írja ki.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: ?, ??, @...SAY, MEMOLINE( ), MLCOUNT( ).

TOTAL

TOTAL ON <kulcs> [<érvényességi kör>] [FIELDS <mezőlista>] ;
TO <fájlnév> | (<változónév>) [FOR <feltétel1>] ;
[WHILE <feltétel2>]

Olyan adatbázist hoz létre, amelyben az azonos kulcsú rekordok megadott mezőit összegzi és egyetlen rekordban tárolja. Az állománynak kulcs szerint rendezettnek vagy indexeltnek kell lennie. Hálózatban a TOTAL kizárólagos használatra nyitja meg a fájlt. Az alapértelmezés a .dbf, ALL.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal. A fordító szerint a FIELDS paraméter elhagyható, de ilyenkor a parancs nem összegez, hanem az azonos kulcsú rekordok közül az elsőt hagyja meg.

Lásd még: AVERAGE, SUM.

TYPE

TYPE <fájlnév1> | (<változónév>) ;
[TO PRINT | FILE <fájlnév2> | (<változónév2>)]

Egy szövegfájl tartalmát a képernyőre és a megadott outputra írja ki. A listázást nem lehet megszakítani. Hálózatban a TYPE közös használatra nyitja meg az input-fájlt. A fájlnevek alapértelmezése a .txt.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: COPY FILE.

UNLOCK

UNLOCK [ALL]

A lezárt rekordokat és állományokat szabadítja fel. Az ALL hatására az összes olyan lezárt állomány és rekord felszabadul, amelyet a program zárt le.

Lásd még: SET EXCLUSIVE, USE...EXCLUSIVE, FLOCK( ), LOCK( ), RLOCK( ).

UPDATE

UPDATE ON <kulcsmező> FROM <aliasnév> REPLACE <mező1> WITH ;
<kifejezés1> [,<mező2> WITH <kifejezés2>...] [RANDOM]

Az adatbázist módosíthatja vele. A módosító állománynak egy másik munkaterületen nyitva kell lennie. Mindkét állományt a kulcskifejezés szerint rendezni, ill. indexelni kell. Hálózatban az UPDATE... FROM...közös használatra nyitja meg a módosító fájlt. Ha a RANDOM paramétert használja, akkor csak az aktuális állománynak kell rendezettnek lennie.

Lásd még: APPEND, REPLACE, SET UNIQUE.

USE

USE [<fájlnév> | (<változónév>)] ;
[INDEX <fájllista> | (<változónévlista>)] ;
[EXCLUSIVE] [ALIAS <név> | (<karakteres kifejezés>)]

Megnyitja a megadott adatbázis-állományt. A következő USE parancsig meghatározza az összes művelethez használt adatbázisfájlt. Ha hálózatban használja az állományt, akkor az EXCLUSIVE paraméter hatására csak az a program fér hozzá, amelyik megnyitotta. Az ALIAS paraméterrel megadhat egy belső nevet, ez akár más is lehet, mint a fájl fizikai neve. Ennek akkor van kiemelkedő szerepe, ha egy állományt több munkaterületen is megnyit. Az alapértelmezésben a külső és a belső (ALIAS) név megegyezik.

Az állományokra ajánlatos az ALIAS *név*-vel hivatkozni, nem pedig a terület számával! Ha változóneveket használ az indexállományok megadására, akkor minden egyes állománynevet külön változóban kell tárolni.

A zárójelek közötti változókkal gyorsabb futást és rövidebb programot kap, mint a makrós formátummal.

Lásd még: CLOSE, INDEX, SELECT, SET EXCLUSIVE, SET INDEX.

WAIT

WAIT [<karakterlánc>] [TO <memóraváltozó>]

Felfüggeszti a program futását egy billentyű lenyomásáig. Ha nem ad meg *karakterlánc*-ot, akkor a 'Press any key to continue...' szöveg fog megjelenni. A funkcióbillentyűket nem fogadja el.

Lásd még: ACCEPT, INPUT, INKEY( ).

ZAP

ZAP

Fizikailag törli az összes rekordot az aktív adatbázisból. Hálózatban való használata előtt az állományt le kell foglalni!

Lásd még: CLEAR, DELETE, PACK, FLOCK( ).

A Clipper függvények

A Clipper rendszer függvényeinek két nagy csoportja van.

Az első csoportba a belső, csak a Clippert kifejlesztő programozók által ismert függvények tartoznak. Ezeket a függvényeket (és a belső változókat) a Clipper nyelvből nem lehet elérni.

A másik a felhasználói függvények csoportja. Ezek a programozók számára is elérhető függvények a Clipper.lib és az Extend.lib állományban találhatók. (Célszerű ezt a két könyvtárat összemásolni Clipper.lib néven.) A függvények közül néhánynak a forrásszövegét is közreadták, lefordítva és egy Extend.lib állományba összegyűjtve a programok nagy részét.

A Clipper rendszerhez adott forrásprogramok a következők:

APNDXJ.PRG	Clipper nyelven írt függvények;
EXAMPLEP.PRG	Clipper nyelven írt függvények;
LOCKS.PRG	Clipper nyelven írt függvények;
EXAMPLEC.C	C nyelven írt függvények;
EXAMPLEA.ASM	assembly nyelven írt függvények.

ABS

ABS (<numerikus kifejezés>)

A numerikus kifejezés abszolút értékét adja vissza.

ACCEPTAT '87

ACCEPTAT(<sor>,<oszlop>,<karakterlánc>)

Egy *karakterlánc*-ot kér be a képernyőről. Csak a standard IBM karakterkészlet kódjait fogadja el. A magyar ékezetes karakterek csak akkor rögzíthetők vele, ha átírja a forrásprogramot. Ehhez az eredeti Clipper lemezen található APNDXJ.PRG fájlt kell módosítani. A 234. sorba a

CASE char > 31 .AND. char < 127

helyett a

CASE char > 31

sort kell írni. Fordítsa le és vigye be a LIB programmal a Clipper könyvtárba!

Használhatja a BackSpace-t és a balra nyílt. Mindegyik törli az utolsó karaktert! A sor-, ill. oszlopkoordinátákkal megadott helyre kiírja a *karakterlánc*-ot, amely után begépelheti a szöveget.

Lásd még: @...GET, ACCEPT, INPUT.

ACHOICE '87

ACHOICE (<bal felső sor>,<bal felső oszlop>,;
<jobb alsó sor>,<jobb alsó oszlop>,;
<tömbnév1>;
[,<tömbnév2> | <logikai kifejezés>];
[,'<saját funkció>';
[,<tömbindex>[,<relatív sorszám>]]])

A függvény lehetővé teszi, hogy egy tömbelemet rámutatással válasszon ki. A tömb elemeit egy „ablakban" jeleníti meg, az első elemet inverz színben (a kurzort „elnyeli"). Ezután az elemek közül kiválaszthat egyet vagy többet, ezt már a programozó döntheti el. A függvény önállóan kezeli a felfelé, lefelé nyilat, a PgUp-PgDn és a Home-End billentyűket.

Az első négy paraméter egy ablakot határoz meg a képernyőn, ezután annak a tömbnek a neve következik, amelyben a kiválasztandó szövegek vannak. A függvény az első nemdefiniált típusú elemig jeleníti meg az adatokat.

A második tömb mutató jellegű. Minden eleme egy-egy logikai érték. Ha az elem *igaz*, akkor a másik tömb megfelelő eleme kiválasztható, különben nem. Ekkor a saját függvény a módparaméterben négyes értéket kap (nem választható ki). Ha csak egy logikai értéket ad meg, akkor a függvény a tömb minden elemét ilyen értékűnek feltételezi.

A DBEDIT funkcióhoz hasonlóan itt is egy saját függvény gondoskodik a rugalmasságról, amely minden egyes billentyű lenyomása után meghívódik. A lenyomott billentyű kódját a LASTKEY függvényből tudhatja meg.

A saját függvény három paramétert kap: a módot, a kurzor sorát és a tömbindexet.

A mód értékei a következők lehetnek:

0 a kurzormozgató billentyűt nyomta le;
1 a felfelé nyilat vagy a PgUp billentyűt nyomta le, és a tömb elején állt;
2 a lefelé nyilat vagy a PgDn billentyű nyomta le, és a tömb végén állt;
3 billentyűt nyomott le;
4 nem kiválasztható.

A saját függvény visszaadható értékei:

0 az ACHOICE befejezi a működést, és 0 értéket ad vissza;
1 normál befejezés, az ACHOICE a *tömbindex*-et adja vissza;
2 folytatás;
3 a saját függvény kapja vissza a vezérlést, és a *tömbindex* arra a következő elemre áll, amelynek első betűje megegyezik a lenyomott billentyűvel; ha nincs ilyen, nem csinál semmit.

Ha pl. a kurzor az „ablak" és a tömb végén áll, akkor a KEYBOARD paranccsal a Home billentyű (1) értékét töltheti a billentyűzetpufferba. Ezzel a trükkel a tömb „végtelennek" fog látszani!

Ha nem ad meg függvényt, akkor az első <Return>-re visszaadja a tömbindexet, az <Esc>-re pedig a nullát.

A használható billentyűk:

Felfelé nyíl	*	egy elemet fel;
Lefelé nyíl	*	egy elemet le;
Home		az első elem;
Ctrl-PgUp	*	az első elem;
End		az utolsó elem;
Ctrl-PgDn	*	az utolsó elem;
PgUp	*	egy ablakkal felfelé;
PgDn	*	egy ablakkal lefelé;
Return		az elem kiválasztása, befejezés;
Esc,		a függvény befejezése, kiválasztás nélkül;
Jobbra nyíl;		
Balra nyíl		
Bármi más		a kurzor a következő azonos betűvel kezdődő elemre áll.

A *-gal megjelölt billentyűk akkor is „élnek", ha megadott egy saját függvényt.

Még két paramétert adhat meg. A *tömbindex* arra az elemre mutat, amelyik az alapértelmezés lesz. (Kezdetben az aktuális elem.) Ha nem adja meg, akkor ez az első elem. A *relatív sorszám* a kijelölt „ablakon" belüli aktuális sorszám lesz. A függvény észreveszi a helytelen értékeket, és az alapértelmezést használja.

Ha pl. egy fájlnevet kell kiválasztania, akkor az ADIR függvénnyel olvassa be a megfelelő fájlnévcsoportot, és az ACHOICE függvénnyel egyet kiválaszthat belőle.

```
.
.
.
DARAB=ADIR('*.DBF')                    && A max. elemszám.
PRIVATE TOMB[DARAB]                    && A TOMB létrehozása,
ADIR('*.DBF',TOMB)                     && feltöltése és
INDEX=ACHOICE(10,10,15,30,TOMB)        && kiíratása.
.
.
.
```

Lásd még: @...PROMPT, MENU TO, SET MESSAGE, DBEDIT( ).

ACOPY '87

ACOPY (<inputtömb>,<outputtömb>[,<honnantól>[,<hossz>;
[,<hová>]]])

Egy tömböt egy másikba másol, az alapértelmezésben az egész tömböt. Megadható honnan mennyit másoljon, és az *outputtömb*-be honnan írjon. Az eredménytől függetlenül logikai *hamis* értéket ad vissza. Nem figyelmeztet, ha a *hossz* nagyobb, mint az *outputtömb*, de nem okoz hibát sem.

Lásd még: ACHOICE( ), ADEL( ), ADIR( ), AFIELDS( ), AFILL( ), AINS( ), ASCAN( ), ASORT( ), LEN( ).

ADD_REC '87

ADD_REC ([<időtartam>])

Hálózatban egy üres rekordot ad a fájl végéhez. Ezt a rekordot le is foglalja más felhasználó elől, és a rekordmutatót ráállítja. Ha a művelet sikeres volt, logikai *igaz* értékkel tér vissza.

Paraméterként megadható egy *időtartam*. Ekkor a függvény a megadott ideig próbálkozik. Ha nem ad meg paramétert, vagy nullát ad meg, akkor az *időtartam*-ot végtelennek veszi, és csak akkor tér vissza, ha sikeres volt a művelet.

Forrása a Locks.prg állományban található. *Figyelem!* Ezt a modult egyik .lib könyvtár sem tartalmazza. Le kell fordítani és a DOS Lib programjával a Clipper.lib-be kell vinni.

Lásd még: APPEND BLANK, FLOCK( ), LOCK( ), RLOCK( ).

ADEL

ADEL (<tömbnév>,<index>)

Egy tömb egyik elemét törli. Az *index* „alatti" elemek eggyel „feljebb" kerülnek, az utolsó (új) elem definiálatlan típusú lesz. A művelet sikerességétől függetlenül mindig logikai *hamis* értéket ad vissza.

Lásd még: ACHOICE( ), ACOPY( ), ADIR( ), AFIELDS( ), AFILL( ), AINS( ), ASCAN( ), ASORT( ), LEN( ).

ADIR

ADIR (<fájlnévmaszk>[,<fájlnévtömb>[,<mérettömb>;
[,<az utolsó módosítás napja tömb>;
[,<az utolsó módosítás ideje tömb>;
[,<attribútumtömb>]]]]])

Megszámolja a megadott *fájlnévmaszk*-kal kijelölt fájlokat (ezt adja vissza), és ha kell, egy tömbben tárolja is azokat. Ha nincs elég hely a tömbben, akkor csak annyi fájlnevet tölt be, amennyi elfér, és nem ad hibaüzenetet.

Ha egy tömböt nem akar megadni, de a többi paraméterértékre szüksége van, akkor egy üres karakteres változót adjon meg a tömb helyett!

Ha a tömböket megadja, akkor az tartalmazni fogja a fájlok adatait. Ha az attribútumtömböt is megadja, akkor a függvény minden kijelölt katalógusbejegyzést beolvas, a könyvtárneveket és a titkosított neveket különben nem olvasná be.

A visszaadott attribútumok:

R írásvédett;
H titkos;
S rendszer;
D könyvtár;
A archív.

Egy állománynak természetesen több attribútuma is lehet, ilyenkor egy karakterlánc formájában adja meg a Clipper.

Ha pl. egy fájlnevet kell kiválasztania, akkor az ADIR függvénnyel olvassa be a megfelelő fájlnévcsoportot, és az ACHOICE függvénnyel egyet kiválaszthat belőle.

```
.
.
.
URES=''
DARAB=ADIR('*.DBF')                          && A maximális
                                             && elemszám.
PRIVATE TOMB[DARAB],HOSSZ[DARAB]             && A tömbök
PRIVATE TIPUS[DARAB]                         && létrehozása,
ADIR('*.DBF',TOMB,HOSSZ,URES,URES,TIPUS)     && feltöltése és
INDEX=ACHOICE(10,10,15,30,TOMB)              && kiíratása.
.
.
.
```

Lásd még: DIR, ACHOICE( ), ACOPY( ), ADEL( ), AFIELDS( ), AFILL( ), AINS( ), ASCAN( ), ASORT( ), LEN( ).

AFIELDS '87

AFIELDS (<mezőnévtömb>[,<típustömb>[,<hossztömb>;
[,<tizedesek száma tömb>]])

A tömböket az aktuális nyitott fájl mezőadataival tölti fel. A töltést az első elemmel kezdi! Ha a tömbök kisebbek, mint a mezők száma, akkor nem ad hibaüzenetet. Ez elkerülhető az FCOUNT( ) függvény használatával, amely az adatbázismezők számát adja vissza.

Ha egy tömböt nem akar megadni, de a következő értékre szüksége van, akkor egy üres karakteres változót adjon meg a tömb helyett!

```
.
.
.
PRIVATE MEZOK[FCOUNT( )]          && A tömb létrehozása.
AFIELDS(MEZOK)                    && A tömb feltöltése.
.
.
.
```

Lásd még: ACHOICE( ), ACOPY( ), ADEL( ), ADIR( ), AFILL( ), AINS( ), ASCAN( ), ASORT( ), FCOUNT( ), FIELD( ), LEN( ).

AFILL

AFILL (<tömbnév>,<érték>[,<kezdőpozíció>[,<hossz>]])

Egy tömböt a megadott *érték*-kel tölt fel, az alapértelmezés szerint az első elemtől az utolsóig. Amikor létrehoz egy tömböt, akkor az elemek mind definiálatlan típusúak lesznek. Ezzel a függvénnyel gyorsan kezdőértéket adhat egy tömb minden elemének. Mindig logikai *hamis* értéket ad vissza.

Lásd még: ACHOICE( ), ACOPY( ), ADEL( ), ADIR( ), AFIELDS( ), AINS( ), ASCAN( ), ASORT( ), LEN( ).

AINS

AINS (<tömbnév>,<index>)

Egy elemet szúr be a megadott tömbbe. Az *index*-szel megadott elem és az „alatta" lévő elemek eggyel „lejjebb" kerülnek. Az *index*-edik elem típusa nemdefiniált lesz, az utolsó elem pedig elveszik! Mindig logikai *hamis* értéket ad vissza.

Lásd még: ACHOICE( ), ACOPY( ), ADEL( ), AFIELDS( ), AFILL( ), ADIR( ), AINS( ), ASCAN( ), ASORT( ), LEN( ).

ALIAS

ALIAS ([<munkaterületszám>])

A kiválasztott munkaterületen lévő fájl ALIAS nevével tér vissza. Ha a paraméter 0 vagy nincs, akkor az aktuális fájl ALIAS nevet adja. Ha a megadott területen nincs megnyitva állomány, akkor egy „üres" sztringet ad.

Lásd még: SELECT, USE, SELECT( ).

ALLTRIM

ALLTRIM(<karakteres kifejezés>)

A karakterlánc elejéről és végéről levágja a <space> karaktereket, és az így kapott karakterláncot adja vissza. Gyorsabb és szebb megoldás, mint az LTRIM(RTRIM(<sztring>)) forma.

Lásd még: LTRIM( ), RTRIM( ), TRIM( ).

ALTD '87

ALTD ([<mód>])

A Clipper Debuggerének használatát szabályozza. Ha nem ad meg paramétert, akkor a Debugger elindul. Mindig logikai *hamis* értéket ad vissza.

A mód értékei a következők lehetnek:

nincs	a Debugger indítása a főmenü megjelenítésével;
0	nem engedi meg az Alt-D használatát;
1	megengedi az Alt-D használatát;
2	a Debugger indítása a Private változók listájának megjelenítésével.

Lásd még: SET ESCAPE.

AMPM

AMPM (<idősztring>)

Egy időformátumú karakterláncból (óó:pp:mm), amely 24 órás kijelzésű, 12 órás kijelzésű, 11 bájt hosszúságú karakterlánccal ('óó:pp:mm am') tér vissza. A délelőttöket „am", a délutánokat „pm" jelöli. A forrása az Examplep.prg.

Lásd még: TIME( ).

ASC

ASC (<karakteres kifejezés>)

A karakterlánc első karakterének ASCII kódjával tér vissza.

Lásd még: CHR( ), INKEY( ).

ASCAN

ASCAN (<tömbnév>,<érték>[,<kezdőcím>[,<hossz>]])

Megkeresi a tömbben a megadott *érték*-et. Alapértelmezésben a tömb elejétől a végéig keres, de ezt felülbírálhatja. A visszaadott érték 0, ha nem talál, különben a keresett elem indexe. A SET EXACT értéke befolyásolhatja a keresés eredményét! Ha a *hossz* túlmutat a tömbön, akkor típushiba keletkezik. Ha a *kezdőcím* mutat túl a tömbön, akkor a visszaadott érték logikai *hamis* lesz.

Lásd még: ACHOICE( ), ACOPY( ), ADEL( ), ADIR( ), AFIELDS( ), AFILL( ), AINS( ), ASORT( ), LEN( ).

ASORT '87

ASORT (<tömbnév>[,<honnantól>[,<milyen hosszan>]])

A tömböt növekvő sorrendbe rendezi, az alapértelmezés szerint az elejétől a végéig. Megadható a rendezés kezdete és hossza is. A hosszúság túlmutathat a tömbön, ez nem okoz hibát. Minden esetben logikai *hamis* eredményt ad vissza. Csak azonos típusú adatokat tud rendezni!

Lásd még: ACHOICE( ), ACOPY( ), ADEL( ), ADIR( ), AFIELDS( ), AFILL( ), AINS( ), ASCAN( ), LEN( ).

AT

AT (<keresett karakterlánc>,<hol keresse>)

Megadja, hogy a *keresett karakterlánc* melyik pozíciótól kezdve található meg egy másik karakterláncban. Ha a visszaadott érték 0, akkor nincs benne.

Lásd még: RAT( ), STRTRAN( ), SUBSTR( ), $.

BIN2I '87

BIN2I (<karakteres kifejezés>)

A *karakteres kifejezés* első két bájtját adja vissza, 16 bites egész számként értelmezve. (Binárisan tárolt számok konverziója.) A Clipper a karakterláncok végét egy bináris 0 bájttal jelzi, ezért a további részek (ha vannak), a Clipperből nem érhetők el! A forrása az Examplea.asm.

Lásd még: BIN2L( ), BIN2W( ).

BIN2L '87

BIN2L (<karakteres kifejezés>)

A *karakteres kifejezés* első négy bájtját adja vissza, 32 bites egész számként értelmezve. (Binárisan tárolt számok konverziója.) A Clipper a karakterláncok végét egy bináris 0 bájttal jelzi, ezért a további részek (ha vannak), a Clipperből nem érhetők el! A forrása az Examplea.asm.

Lásd még: BIN2I( ), BIN2W( ).

BIN2W '87

BIN2W (<karakteres kifejezés>)

A *karakteres kifejezés* első két bájtját adja vissza, 32 bites egész számként értelmezve. (Binárisan tárolt számok konverziója.) A Clipper a karakterláncok végét egy bináris 0 bájttal jelzi, ezért a további részek (ha vannak), a Clipperből nem érhetők el! A forrása az Examplea.asm.

Lásd még: BIN2I( ), BIN2L( ).

BOF

BOF ( )

Igaz értéket ad vissza, ha a rekordmutató a fájl logikailag legelső rekordja elé mutat. Ez két esetben lehetséges, vagy egy SKIP paranccsal, ha az első rekordon ad ki egy SKIP -1 parancsot, vagy ha az állomány üres. Ilyenkor az EOF függvény is *igaz* értéket ad vissza.

Lásd még: SKIP, EOF( ), LASTREC( ), RECCOUNT( ), RECNO( ).

BROWSE

BROWSE ([<bal felső sor>,<bal felső oszlop>;
<jobb alsó sor>,<jobb alsó oszlop>])

Az adatbázismezőket szerkesztő funkció, amely a SAVE SCREEN-nel elmenti a képernyőt. Ha SAVE SCREEN-nel mentett ki egy előző képernyőt, akkor az elveszik, használja inkább a SAVE SCREEN TO ... formát! Ha nem ad meg paramétereket, akkor az alapértelmezés: 1, 0, 23, 79.
A forrása az Examplep.prg.

Lásd még: DBEDIT( ).

CDOW

CDOW (<dátumkifejezés>)

A hét egyik napjának amerikai jelölésmód szerinti nevével tér vissza a dátumváltozóból.

Sunday	(vasárnap)
Monday	(hétfő)
Tuesday	(kedd)
Wednesday	(szerda)
Thursday	(csütörtök)
Friday	(péntek)
Saturday	(szombat)

Lásd még: SET DATE, CMONTH( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

CHR

CHR (<numerikus kifejezés>)

Azt a karaktert adja vissza, amelynek az ASCII kódja a *numerikus kifejezés* értékével egyenlő. A CHR(0) az egyetlen olyan lehetőség, amellyel 0 bájtot küldhet bárhová, pl. a nyomtatónak egy vezérlő karaktersorban. Ha egy karakterlánchoz fűzi hozzá, akkor az utána következő részek a Clipper számára elvesznek. A Clipper ugyanis egy 0 bájttal zárja le a karakterláncokat.

Lásd még: ?, @...SAY, KEYBOARD, ASC( ), INKEY( ), LEN( ).

CMONTH

CMONTH (<dátumkifejezés>)

A dátumváltozóból a hónap amerikai jelölésmód szerinti nevével tér vissza.

January	(január)
February	(február)
March	(március)
April	(április)
May	(május)
June	(június)
July	(július)
August	(augusztus)
September	(szeptember)
October	(október)
November	(november)
December	(december)

Lásd még: CDOW( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

COL

COL ( )

A kurzor aktuális oszloppozícióját adja vissza.

Lásd még: @, @...BOX, @...CLEAR, @...GET, @...SAY, @...TO, PCOL( ), PROW( ), ROW( ).

CTOD

CTOD (<karakteres kifejezés>)

A karakterláncként megadott dátumot dátum típusú változóvá alakítja át. A forma a SET DATE kifejezés szerinti, a karakterláncban az elválasztójel bármi lehet!

Lásd még: SET DATE, CMONTH( ), DATE( ), DAY( ), DOW( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

CURDIR '87

CURDIR ([<lemeznév>])

A megadott (az alapértelmezés az aktuális) lemezen az aktuális könyvtárnevet adja vissza. Ha a lemez nem elérhető, vagy a főkönyvtár az aktuális, akkor egy 0 hosszúságú karakterlánccal tér vissza. A forrása az Examplea.asm.

Lásd még: SET DEFAULT.

CURRENCY '87

CURRENCY (<numerikus kifejezés>)

A megadott számot '$999,999,999,999.99' formátumú karakterláncként adja vissza. A forrása az Apndxj.prg.

DATE

DATE ( )

A rendszerdátummal tér vissza. A visszaadott formát a SET DATE és a SET CENTURY paranccsal állíthatjuk be.

SET DATE	Dátumforma
AMERICAN	HH/NN/ÉÉ
ANSI	ÉÉ.HH.NN
BRITISH	NN/HH/ÉÉ
FRENCH	NN/HH/ÉÉ
GERMAN	NN.HH.ÉÉ
ITALIAN	NN-HH-ÉÉ

Lásd még: SET CENTURY, SET DATE, CDOW( ), CMONTH( ), CTOD( ), DAY( ), DOW( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

DAY

DAY (<dátumkifejezés>)

A nap értékét adja vissza a kifejezésből. Ha nem szökőévben február 29. szerepel a dátumban, akkor nullát ad vissza, és nem jelez hibát.

Lásd még: CDOW( ), CMONTH( ), CTOD( ), DATE( ), DOW( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

DAYS

DAYS (<másodpercek>)

A *másodpercek*-ben megadott időből kiszámítja a napok számát, 24 órás napokat feltételezve. A forrása az Examplep.prg.

Lásd még: SECONDS( ), TIME( ).

DBEDIT

DBEDIT ([<bal felső sor>[,<bal felső oszlop>;
[,<jobb alsó sor>[,<jobb alsó oszlop>]]];
[,<mezőnévtömb>];
[,'<sajátfüggvény-név>'];
[,<PICTURE tömb>];
[,<fejléctömb>];
[,<fejlécaláhúzás-tömb> | <aláhúzáskarakter>];
[,<lábjegyzet-elválasztó tömb> |;
<elválasztó karakter>];
[,<lábjegyzettömb> | <lábjegyzet-kifejezés>]

Az aktuális adatbázis mezőinek egyszerű szerkesztését végzi el egy megadott ablakban. A *mezőnévtömb* különleges felépítést igényel: az elemek a szükséges mezők neveit tartalmazzák! A tömbnek csak annyi eleme lehet, amennyit szerkeszteni akarunk! A DBEDIT egy ablakot képez a képernyőn, itt aktualizálhatja az állományt.

A saját függvény paraméterei a funkció és az index.

A funkció egy numerikus érték, lehetséges értékei a következők:

0 egy kurzormozgató billentyűt nyomott le;
1 egy kurzormozgató billentyűt nyomott le, és elérte az állomány elejét;
2 egy kurzormozgató billentyűt nyomott le, és elérte az állomány végét;
3 az állomány üres;
4 egy tetszőleges billentyűt nyomott le, értékét a LASTKEY( ) függvénnyel tudhatja meg.

Az index egy numerikus érték, amely a használt tömb egyik elemére mutat. Innen tudhatja meg, melyik mezőt szerkeszti a DBEDIT.

A függvény három értéket adhat vissza:

0 a szerkesztés befejezése;
1 a szerkesztés folytatása;
2 a szerkesztés folytatása, de a képernyő újraépítésével.

Lásd még: @...GET, @...SAY, READ, ACHOICE( ).

DB_ERROR '87

DB_ERROR (<prodecúranév>,<sorszám>,<hibaüzenet>)

A Clipper hibakezelő függvényeinek egyike. Akkor hívja meg a rendszer, ha adatbázishiba lép fel. Az átadott paraméterek a Help rendszerhez hasonlók. Ha az alapértelmezés nem felel meg, akkor egy saját függvénnyel felülírhatja.

DBF

DBF ( )

Az aktuális fájl ALIAS nevét adja vissza. A Clipper nem jegyzi meg a nyitott adatbázisok neveit, ezért csak az ALIAS nevet tudja visszaadni. Használja inkább az ALIAS függvényt! A forrása az Examplep.prg.

Lásd még: ALIAS( ).

DBFILTER '87

DBFILTER ( )

Az aktuális munkaterülethez tartozó SET FILTER utasítással beállított feltétellel tér vissza. Ha nem ad ki SET FILTER utasítást, akkor egy 0 hosszúságú karakterlánccal tér vissza.

Lásd még: DBRELATION( ).

DBRELATION '87

DBRELATION (<numerikus kifejezés>)

A kifejezést sorszámnak tekinti, és a SET RELATION-ben megadott kulcskifejezések közül ezt a sorszámút adja vissza. Ha nem ad meg SET RELATION-t, akkor egy 0 hosszúságú karakterlánccal tér vissza.

Példa:

```
SELECT 1
USE készlet     INDEX cikkszám
SELECT 2
USE törzs       INDEX azon
SELECT 3
USE kódok       INDEX kódszám
SELECT 1
SET RELATION TO cikkszám INTO törzs ;
      TO színe           INTO kódok

*
? DBRELATION(2)      && értéke 'színe'
```

Lásd még: DBRSELECT( ).

DBRSELECT '87

DBRSELECT (<numerikus kifejezés>)

A kifejezést sorszámnak tekinti, és a SET RELATION-ben megadott fájlok közül ennek a munkaterületnek a számát adja vissza. Ha nem ad meg SET RELATION-t, akkor 0 ad vissza.

Példa:

```
SELECT 1
USE készlet     INDEX cikkszám
SELECT 2
USE törzs       INDEX azon
SELECT 3
USE kódok       INDEX kódszám
SELECT 1
SET RELATION TO cikkszám INTO törzs ;
     TO színe          INTO kódok

*
? DBRSELECT(2)        && értéke 3
```

Lásd még: DBRELATION( ).

DELETED

DELETED ( )

Igaz értéket ad vissza, ha az aktuális rekord törlésre kijelölt. Ha SET DELETED ON utasítást ad ki, akkor törölt rekordra csak közvetlen GO, ill. GOTO utasítással léphet.

Lásd még: DELETE, PACK, RECALL, SET DELETED.

DESCEND '87

DESCEND (<kulcskifejezés>)

A Clipper indexeket eddig csak növekvő sorrendben használhatta. Ezzel a függvénnyel csökkenő sorrendű indexfájlokat is létrehozhat. Ha használja, akkor az INDEX és a SEEK utasításban is használnia kell!

Lásd még: CONTINUE, FIND, INDEX, LOCATE, SEEK.

DIM2 '87

DIM2 (<sor>,<oszlop>)

Egy kétdimenziós tömb címzését segíti. A ROWS nevű változó tartalmazza a max. oszlopszámot egy sorban. Erről a programozónak kell gondoskodnia! A forrása az Apndxj.prg.

Lásd még: DECLARE, PRIVATE, PUBLIC.

DISKSPACE

DISKSPACE ([<lemezszám>])

A megadott számú lemezen lévő üres hely méretét bájtokban adja vissza. A meghajtók jelölése:

0 vagy nincs	az aktuális meghajtó;
1	az A meghajtó;
2	a B meghajtó;
3	a C meghajtó;
.	
.	
.	

A forrása az Examplec.c.

DOSERROR '87

DOSERROR ( )

Az utolsó DOS hiba számát adja vissza, eredményesen használhatja a Clipper hibakezelő funkcióiban.

Lásd még: FERROR( ).

DOW

DOW (< dátumkifejezés >)

A *dátumkifejezés*-ből visszaadja, hogy a hétnek melyik napja van. A visszaadott szám az amerikai naptár szerinti sorszám.

1 vasárnap
2 hétfő
3 kedd
4 szerda
5 csütörtök
6 péntek
7 szombat

Lásd még: CDOW( ), CMONTH( ), CTOD( ), DATE( ), DAY( ), DTOC( ), DTOS( ), MONTH( ), YEAR( ).

DTOC

DTOC (< dátumkifejezés >)

A *dátumkifejezés*-t karakteresre alakítja át. A formát a SET DATE és a SET CENTURY utasítás befolyásolja. Ha indexkifejezésként használ egy dátum típusú változót, akkor ne a DTOC függvénnyel alakítsa át karakteres formára, hanem a DTOS-szal, mert így a dátumformától teljesen független formájú karakterláncot kap.

SET DATE	Dátumforma
AMERICAN	HH/NN/ÉÉ
ANSI	ÉÉ.HH.NN
BRITISH	NN/HH/ÉÉ
FRENCH	NN/HH/ÉÉ
GERMAN	NN.HH.ÉÉ
ITALIAN	NN-HH-ÉÉ

Lásd még: SET CENTURY, SET DATE, CDOW( ), CMONTH( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOS( ), MONTH( ), YEAR( ).

DUP_CHK '87

DUP_CHK (<'változónév'>,<ALIAS név>)

Igaz értéket ad vissza, ha a kulcsértéknek megfelelő rekord még nincs felvéve a fájlba. A fájlnak indexelve kell lennie erre a kulcskifejezésre! A függvény eredeti forrásában (APNDXJ.PRG) két hiba van, az első a 131. sorban. Ha a megadott változó üres, akkor is *igaz* értékkel tér vissza, ezt .F.-re kell javítani. A másik hiba a 142. sorban található, a SELECT után – a változónév elé – & jelet kell tenni! A forrása az Apndxj.prg.

DTOS

DTOS (<dátumkifejezés>)

A *dátumkifejezés*-nek megfelelő karakterlánccal tér vissza éééhhnn formában. Indexállományok létrehozásakor is használható, mert a visszaadott érték független a SET DATE-tel beállított dátumkijelzéstől.

Lásd még: INDEX, CDOW( ), CMONTh( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOC( ), MONTH( ), YEAR( ).

ELAPTIME

ELAPTIME (<kezdet>,<vég>)

A megadott időpontok között eltelt időt adja vissza óó:pp:mm formában. Az időpontokat óó:pp:mm formában kell megadni!
A forrása az Examplep.prg.

Lásd még: TIME( ).

EMPTY

EMPTY (<bármilyen kifejezés>)

Igaz értéket ad vissza, ha a kifejezés értéke „üres". A különböző típusú kifejezések „üres" értékei:

karakterlánc	ha csak space karakterek vannnak benne, vagy a hossza 0;
numerikus	0;
dátum	0000.00.00 (ANSI formában);
logikai	.F..

EOF

EOF ( )

Igaz értéket ad vissza, ha a rekordmutató a fájl végén áll. A Clipper kereső utasítással – LOCATE, FIND, SEEK – is beállítják. *Igaz* értéket ad vissza, ha *nem* talált.

Lásd még: FIND, LOCATE, SEEK, SKIP, BOF( ), FOUND( ), LASTREC( ), RECNO( ).

ERR_MSG '87

ERR_MSG (<hibaüzenet>)

A *hibaüzenet*-et a képernyő közepén jeleníti meg. Egy billentyű lenyomására vár, a képernyőt a SAVE SCREEN paranccsal menti el. Használja inkább a SAVE SCREEN TO formát!

A forrása az Apndxj.prg.

ERRORLEVEL '87

ERRORLEVEL ([numerikus kifejezés>])

Ha nem ad meg értéket, akkor az előző program hibakódját adja vissza, különben a saját programét állítja be. A program után használhatja a DOS ERRORLEVEL utasítását is.

Lásd még: CANCEL, QUIT, RETURN.

EXP

EXP (<numerikus kifejezés>)

A kifejezés exponenciális függvényértékét adja vissza. A LOG függvény inverze!

Lásd még: SET DECIMALS, SET FIXED, LOG( ).

EXPR_ERROR '87

EXPR_ERROR (<procedúranév>,<sorszám>,<hibaüzenet>,<modell>,;
<bal oldali érték>,<jobb oldali érték>,;
<eredmény>)

A Clipper hibakezelő rendszerének része. Ha valamelyik kifejezésben hiba van, akkor megkapja a vezérlést. A *modell* a hibás kifejezést karakterlánc formában tartalmazza.

Lásd még: @...PROMPT, MENU TO, SET MESSAGE, DBEDIT( ).

FCLOSE '87

FCLOSE (<fájlsorszám>)

A Clipper egy alacsony szintű fájlkezelő rendszert is nyújt a felhasználónak. Ez a függvény egy nyitott fájlt zár le. Ha a művelet jó volt, akkor *igaz* értéket ad vissza.

Lásd még: FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

FCOUNT

FCOUNT( )

Az aktuális állomány mezőinek számával tér vissza. Ha nincs nyitva állomány, akkor a visszaadott érték 0.

Lásd még: FIELD( ), FIELDNAME( ), TYPE( ).

FCREATE '87

FCREATE (<fájlnév>[,<attribútum>])

Egyike a Clipper alacsony szintű fájlkezelő függvényeinek. A megadott nevű fájlt létrehozza, megnyitja és visszaadja a fájl sorszámát is. Minden esetben tárolja el!

Ha az *attribútum*-ot nem adja meg, akkor a függvény még nem archivált típusú fájlt hoz létre, és azt írásra, ill. olvasásra nyitja meg. Ha hiba van, akkor a visszaadott érték -1 lesz. A hiba okát az FERROR függvénnyel kérdezheti le.

Attribútum	Jelentése
1	Csak olvasható
2	Titkos
4	Rendszer
8	Lemeznév
16	Alkönyvtár
32	Még nem archivált
64	Nem használt
128	Nem használt

Lásd még: SET DEFAULT, SET PATH, FCLOSE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

FERROR '87

FERROR ( )

A legutóbbi alacsony szintű fájlkezelő függvény utáni DOS hiba számát adja vissza. Ha nem volt hiba, akkor nullát ad vissza. Mivel a Clipper alacsony szintű állománykezelő funkciói a belső Clipper funkciókat használják, ezzel a függvénnyel minden fájlművelet ellenőrizhető!

Lásd még: FCLOSE( ), FCREATE( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

FIELD

FIELD (<numerikus kifejezés>)

Az aktuális adatbázis olyan mezőjének nevével tér vissza, amelyik a kifejezés értéke szerinti sorszámú. Ha nincs ennyi mező a fájlban, akkor egy 0 hosszúságú karakterlánccal tér vissza.

Lásd még: COPY STRUCTURE EXTENDED, AFIELDS( ), FCOUNT( ), FIELDNAME( ), LASTREC( ), TYPE( ).

FIELDNAME

FIELDNAME (<numerikus kifejezés>)

Az aktuális adatbázis olyan mezőjének nevével tér vissza, amelyik a kifejezés értéke szerinti sorszámú. Ha nincs ennyi mező a fájlban, akkor egy 0 hosszúságú karakterlánccal tér vissza.

Lásd még: COPY STRUCTURE EXTENDED, AFIELDS( ), FCOUNT( ), FIELDNAME( ), LASTREC( ), TYPE( ).

FILE

FILE (<fájlnév>)

a megadott nevű fájl létezését ellenőrzi. A teljes nevet meg kell adni karakterlánc formájában.

Lásd még: SET DEFAULT, SET PATH.

FIL_LOCK '87

FIL_LOCK (<várakozási idő>)

Zárolja az aktuális fájlt. A megadott másodpercig próbálkozik, ha ezalatt az idő alatt nem sikerül zárolni, akkor *hamis* (.F.) értéket ad vissza. 0 paraméter esetén addig vár, amíg zárolni nem tudja a fájlt, fél másodpercenként ismét próbálkozik. A forrása az Locks.prg.

Lásd még: FLOCK( ), LOCK( ), NETERR( ), RLOCK( ).

FKLABEL

FKLABEL (<funkcióbillentyű sorszám>)

A megadott sorszámú funkcióbillentyű karakteres nevét (F1, F2, ..., F40) adja vissza. A forrása az Examplep.prg.

Lásd még: SET KEY.

FKMAX

FKMAX( )

A rendszerben használt funkcióbillentyűk max. számát adja vissza, konstans értékként 40-et. A forrása az Examplep.prg.

Lásd még: FKLABEL( ).

FLOCK

FLOCK ( )

Az aktuális állományt lezárja a program számára. Ha ez sikerült, *igaz* (.T.) értéket ad vissza. Ha nem, *hamis*-at (.F.). Ez akkor lehetséges, ha valaki ezt már megtette, vagy a fájl egyik rekordját lezárta. Ha ezután egy RLOCK, ill. LOCK utasítást ad ki, akkor az FLOCK hatása megszűnik!

Lásd még: @...PROMPT, MENU TO, SET MESSAGE, DBEDIT( ).

FOPEN '87

FOPEN (<fájlnév>[,<nyitásmód>])

A megadott nevű fájlt nyitja meg. Ha a művelet sikerült, visszaadja a fájl sorszámát. Ezentúl ezzel a sorszámmal hivatkozhat erre az állományra. A nyitásmód alapértelmezése a csak olvasásra nyitott.

Nyitásmód	Jelentése
0	írás
1	olvasás
2	írás – olvasás

Lásd még: SET DEFAULT, SET PATH, FCLOSE( ), FCREATE( ), FERROR( ), FREAD( ), FREADSTR( ), FSEEK( ), FWRITE( ).

FOUND

FOUND( )

Ha a legutóbbi SEEK, FIND, LOCATE vagy CONTINUE parancs megfelelő rekordot talál, akkor *igaz* értéket ad vissza. Ez csak közvetlenül a parancs után érvényesül. Ha bármilyen utasítás van a keresés és a függvény hívása között, akkor az eredmény *hamis* lesz! Használja helyette az EOF( ) függvényt!

Lásd még: CONTINUE, FIND, LOCATE, SEEK, SET RELATION, SET SOFTSEEK, EOF( ).

FREAD '87

FREAD (<fájlsorszám>,@<hova>,<mennyit>)

Olvas a fájlból. Ez az egyetlen olyan függvény, amelyben használni kell a referenciakénti paraméterátadást! Visszaadja, hogy hány karaktert olvasott be valójában.

Lásd még: FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREADSTR( ), FWRITE( ).

FREADSTR '87

FREADSTR (<fájlsorszám>,<mennyit>)

A beolvasott karakterláncot adja vissza. Az olvasás a megadott hosszig, vagy az első 0 bájtig tart.

Lásd még: FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FSEEK( ), FWRITE( ).

FSEEK '87

FSEEK (<fájlsorszám>,<pozíció>[,<kezdőpozíció>])

A megadott bájtra pozicionál. A függvény a jelenlegi pozíciót adja vissza.

Kezdőpozíció	Jelentése
0	a fájl elejétől (alapértelmezés)
1	az aktuálistól
2	a fájl végétől

Lásd még: FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FWRITE( ).

FWRITE '87

FWRITE (<fájlsorszám>,<honnan>[,<mennyit>])

A függvény karakterláncot ír egy fájlba. Az alapértelmezés a változó hossza. A pozíciómutatót is lépteti.

Lásd még: FCLOSE( ), FCREATE( ), FERROR( ), FOPEN( ), FREAD( ), FREADSTR( ), FSEEK( ).

GETE

GETE (<DOS változó>)

A *DOS változó* SET-tel beállított értékét adja vissza.

GET_PIC '87

GET_PIC (<mező | változónév>)

Előre definiált (bekérő) maszkokat rendel hozzá a karakteres és a numerikus típusú változókhoz. Ez a PICTURE utasításban használható fel. Az előre definiált maszkok:

Karakteres	'@KSn' (ahol az *n* a mező vagy a változó hossza);
Numerikus	'999999.999'.

A forrása az Examplep.prg.

HARDCR

HARDCR (<karakteres kifejezés>)

Egy karakterláncban a CHR(141)-eket CHR(13)-ra cseréli.

Lásd még: ?, ??, @...SAY, REPORT FORM, MEMOEDIT( ), MEMOLINE( ), MEMOREAD( ), MEMOTRAN( ), MEMOWRIT( ), MLCOUNT( ).

HEADER

HEADER( )

Az aktuális adatbázis fejlécrekordjának hosszát adja vissza. Ezt a függvényt egy másolóprogram használhatja a másolandó állomány teljes hosszának kiszámításához.

Példa:

```
.
.
.
HOSSZ=HEADER( )+(RECSIZE( )*RECCOUNT( ))
IF DISKSPACE( ) >= HOSSZ
    COPY TO MASOLAT
ELSE
    ? 'Nincs elég hely a lemezen !'
ENDIF
```

Lásd mág: DISKSPACE( ), RECCOUNT( ), RECSIZE( ).

I2BIN '87

I2BIN (<16 bites szám>)

Egy *16 bites szám*-ot 2 bájtos sztringként ad vissza. A forrása az Examplea.asm.

Lásd még: BIN2I, BIN2L, BIN2W, L2BIN.

IF

IF (<feltétel>,<igaz válasz>,<hamis válasz>)

Az IF ... ELSE ... ENDIF utasítássort egyszerűsíti. Ha az első paraméterben megadott logikai kifejezés értéke *igaz*, akkor a második paraméterben megadott kifejezést hajtja végre, különben a harmadik paramétert. A válaszkifejezéseket csak akkor ellenőrzi, ha a *feltétel* szerint azt kell visszaadni! A kifejezések helyességét a TYPE függvénnyel ellenőrizheti.

Lásd még: IF, IIF( ), TYPE( ).

IIF

IIF (<feltétel>,<igaz válasz>,<hamis válasz>)

Az IF ... ELSE ... ENDIF utasítássort egyszerűsíti. Ha az első paraméterben megadott logikai kifejezés értéke *igaz*, akkor a második paraméterben megadott kifejezést hajtja végre, különben a harmadik paramétert. A válaszkifejezéseket csak akkor ellenőrzi, ha a *feltétel* szerint azt kell visszaadni! A kifejezések helyességét a TYPE függvénynyel ellenőrizheti.

Lásd még: IF, IF( ), TYPE( ).

INDEXEXT '87

INDEXEXT( )

Visszaadja, hogy '.NTX' vagy '.NDX' indexszel dolgozik-e a Clipper.

Lásd még: INDEXKEY( ), INDEXORD( ).

INDEXKEY

INDEXKEY (<sorszám>)

A kiválasztott indexállomány indexkifejezésével tér vissza. A paraméter a SET INDEX, ill. a USE ... INDEX parancson belüli *sorszám*-ot jelöli. Ha ennek értéke 0, akkor az aktuális indexfájl kulcsát adja.

Lásd még: SET INDEX, SET ORDER, USE, INDEXEXT( ), INDEXORD( ).

INDEXORD '87

INDEXORD( )

Azt a sorszámot adja vissza, amely jelzi az éppen használt indexfájlt.

Lásd még: SET INDEX, SET ORDER, USE, INDEXEXT( ), INDEXKEY( ).

INKEY

INKEY ([<várakozási idő>])

A lenyomott billentyű ASCII kódjának numerikus értékével tér vissza.

A visszakapott értékek a következők:

Billentyű	Ctrl-billentyű	érték
Home	^A	1
^Jobbra nyíl	^B	2
PgDn	^C	3
Jobbra nyíl	^D	4
Felfelé nyíl	^E	5
End	^F	6
Del	^G	7
Backspace	^H	8
Tab	^I	9
^Enter	^J	10
	^K	11
	^L	12
Enter	^M	13
	^N	14
	^O	15
	^P	16
	^Q	17
PgUp	^R	18
Balra nyíl	^S	19
	^T	20
	^U	21
Ins	^V	22
^End	^W	23
Lefelé nyíl	^X	24
	^Y	25
^Balra nyíl	^Z	26
Esc	^[	27
F1	^\	28
^Home	^]	29
^PgDn	^^	30
^PgUp	^_	31

Billentyű	Ctrl-billentyű	érték
F2 ... F10		-1 – -9
Shift-F1 ... F10		-10 – -19
Ctrl-F1 ... F10		-20 – -29
Alt-F1 ... F10		-30 – -39

A többi esetben a karakter ASCII kódját adja vissza.

Megadhatja a várakozási időt is. Ha nem ad meg semmit, akkor az éppen lenyomott billentyű kódját, ill. 0-t ad vissza. Ha megad egy értéket, akkor annyi másodpercig vár egy billentyű lenyomására. Ha ezalatt az idő alatt lenyom egy billentyűt, akkor annak kódja, ha nem nyom le semmit, akkor 0 az eredmény. Ha nullát ad meg paraméterként, akkor addig vár, amíg – az eltelt időtől függetlenül – le nem nyom egy billentyűt.

Lásd még: SET KEY, CHR( ), LASTKEY( ), NEXTKEY( ).

INT

INT (<szám>)

Egész szám (integer) funkció, amely bármely numerikus kifejezést egész számmá alakít át, a tizedesponttól jobbra eső számjegyeket levágva.

Lásd még: ROUND( ).

ISALPHA

ISALPHA (<karakterlánc>)

Ha a *karakterlánc* első karaktere betű, akkor *igaz* értéket ad vissza.

Lásd még: ISLOWER( ), ISUPPER( ).

ISCOLOR

ISCOLOR ( )

Az éppen használt monitor típusát adja meg. Ha a monitor színes, akkor a függvény értéke *igaz*.

Lásd még: SET COLOR, ISCOLOUR( ).

ISCOLOUR '87

ISCOLOUR ( )

Az éppen használt monitor típusát adja meg. Ha a monitor színes, akkor a függvény értéke *igaz*.

Lásd még: SET COLOR, ISCOLOR( ).

ISLOWER

ISLOWER (<karakterlánc>)

Ha a megadott *karakterlánc* első karaktere kisbetű, akkor *igaz* értéket ad vissza.

Lásd még: ISALPHA( ), ISUPPER( ).

ISPRINTER

ISPRINTER ( )

Ha az LPT1-hez rendelt nyonmtató készen áll, akkor *igaz* értéket ad vissza. A forrása az Examplea.asm.

Lásd még: SET DEVICE, SET PRINTER.

ISUPPER

ISUPPER (<karakterlánc>)

Ha a megadott *karakterlánc* első karaktere nagybetű, akkor *igaz* értéket ad vissza.

Lásd még: ISALPHA( ), ISLOWER( ).

L2BIN '87

L2BIN (<32 bites szám>)

Egy *32 bites szám*-ot 4 bájtos sztringként ad vissza. A forrása az Examplea.asm.

Lásd még: BIN2I, BIN2L, BIN2W, I2BIN.

LASTKEY

LASTKEY ( )

Az utoljára lenyomott billentyű ASCII kódjának numerikus értékével tér vissza. Ez az érték megegyezik az INKEY( ) által visszaadott értékkel.

A visszakapott értékek a következők:

Billentyű	Ctrl-billentyű	érték
Home	^A	1
^Jobbra nyíl	^B	2
PgDn	^C	3
Jobbra nyíl	^D	4
Felfelé nyíl	^E	5
End	^F	6
Del	^G	7
Backspace	^H	8
Tab	^I	9

Billentyű	Ctrl-billentyű	érték
^Enter	^J	10
	^K	11
	^L	12
Enter	^M	13
	^N	14
	^O	15
	^P	16
	^Q	17
PgUp	^R	18
Balra nyíl	^S	19
	^T	20
	^U	21
Ins	^V	22
^End	^W	23
Lefelé nyíl	^X	24
	^Y	25
^Balra nyíl	^Z	26
Esc	^[	27
F1	^\	28
^Home	^]	29
^PgDn	^^	30
^PgUp	^_	31
F2 ... F10		-1 – -9
Shift-F1 ... F10		-10 – -19
Ctrl-F1 ... F10		-20 – -29
Alt-F1 ... F10		-30 – -39

A többi esetben a karakter ASCII kódját adja vissza.

Lásd még: KEYBOARD, CHR( ), INKEY( ).

LASTREC

LASTREC ( )

Az éppen aktív adatbázis rekordjainak számával tér vissza.

Lásd még: SET DELETED, SET FILTER, FIELD( ), RECCOUNT( ).

LEFT

LEFT (<karakterlánc>,<hossz>)

A *karakterlánc* kezdetétől számítva a paraméterben megadott hosszúságú *karakterlánc*-ot adja vissza.

Lásd még: AT( ), RIGHT( ).

LEN

LEN (<karakterlánc> | <tömbnév>)

A függvény egy numerikus értékkel tér vissza, ez a megadott karakteres kifejezésben lévő karakterek száma. Ha a paraméter egy tömb neve, akkor a tömb definiált méretét adja vissza.

Lásd még: DECLARE, ALLTRIM( ), LTRIM( ), RTRIM( ), TRIM( ).

LENNUM

LENNUM (<szám>)

A *szám* hosszát karakterekben adja meg. A hosszúságba a negatív előjel és a tizedespont is beleszámít! A forrása az Examplep.prg.

Lásd még: LEN( ), STR( ).

LOCK

LOCK ( )

Az aktuális állományban lezár egy rekordot. Egyszerre csak egy rekord lehet lezárva egy fájlban. Ha előtte lezárta az egész állományt, akkor az állomány felszabadul, és csak a kijelölt rekord lesz lezárva.

Lásd még: RLOCK( ).

LOG

LOG (<szám>)

A megadott *szám* természetes logaritmusával tér vissza. A tízes alapú logaritmus kiszámításához a következő függvényt kell használni:

LOG 10 (N) = LOGe (N) / LOGe (10)

Lásd még: EXP( ).

LOWER

LOWER (<karakterlánc>)

A nagybetűkkel írt karaktereket kisbetűkké alakítja át. Csak a standard karakterkészlet kódjait képes átalakítani!

Lásd még: UPPER( ).

LTRIM

LTRIM (<karakterlánc>)

A függvény a bal oldali kezdő szóközöket levágja a *karakterlánc*-ról.

Lásd még: ALLTRIM( ), RTRIM( ), STR( ), SUBSTR( ), TRIM( ).

LUPDATE

LUPDATE ( )

Az aktuális adatbázis utolsó módosításának idejét adja vissza. A visszaadott dátumformát a SET DATE utasítással állíthatja be.

Lásd még: SET CENTURY, SET DATE, DATE( ).

MAX

MAX (<kifejezés1>,<kifejezés2>)

A két kifejezés értéke közül a nagyobbat adja vissza. Paraméterként dátum, sőt logikai típusú kifejezést is megadhat. Ha logikai típusú paramétert ad meg, akkor az *igaz* (.T.) lesz a nagyobb.

Lásd még: MIN( ).

MEMOEDIT

MEMOEDIT (<szerkesztendő karakterlánc>);
[,<bal felső sor>,<bal felső oszlop>,;
<jobb alsó sor>,<jobb alsó oszlop>];
[,<megjelenítésjelző>];
[,<sajátfunkció-név>];
[,<sorhossz>];
[,<tabulátorhossz>];
[,<kezdősor>];
[,<kezdőoszlop>];
[,<ablakkezdő sorpozíció>];
[,<ablakkezdő oszloppozíció>]

A függvénnyel megjelenítheti a karakterláncokat, egyszerű szerkesztési lehetőségeket használva.

A képernyőn kijelölhet egy ablakot, ahol a szerkesztendő szöveget látni akarja. Ha nem ad meg semmit, akkor a teljes képernyőn dolgozhat. A függvény nem ír az ablak vagy a képernyő utolsó oszlopára!

Egy logikai kifejezés megadásával jelezheti, hogy szerkeszteni vagy csak megjeleníteni akar-e? *Igaz* érték esetén szerkesztőmódba lép a függvény, *hamis* érték esetén pedig csak megjelenít. Ha nem adja meg a kifejezést, akkor szerkesztőmódban dolgozhat.

A MEMOEDIT felhasználói funkciót is használhat, amelyet minden billentyű lenyomása után meghív. A lenyomott billentyű kódját a LASTKEY függvényből tudhatja meg. A felhasználói funkció három paramétere a következő: mód, szövegsor, szövegoszlop.

A módparaméter jelentése:

Mód	Funkció	Értelmezése
0	Szerkesztés	Bármilyen billentyű lenyomásakor
1	A szöveg értelmezése	A kurzormozgató vagy a szöveget újraformázó billentyű lenyomásakor
2	A szöveg változik	Egy különleges vezérlőbillentyű lenyomásakor
3	Beállítás	A MEMOEDIT elindulásakor

Mielőtt a MEMOEDIT ténylegesen elindulna, meghívja a felhasználói funkció beállítási módját. Itt azokat a paramétereket állíthatja be, amelyeket használni fog majd (pl. beszúrásüzemmód). A funkciót mindaddig ebben a módban hívja meg a MEMOEDIT, amíg a visszaadott paraméter 0 nem lesz!

A felhasználói funkció visszaadott értékei:

Érték	Értelmezése
0	A MEMOEDIT standard válasza
1-31	A kapott kódnak megfelelő billentyű funkciójának elvégzése (lásd INKEY( ))
32	A lenyomott billentyűt nem veszi figyelembe (a kurzormozgató, a Return, a Backspace, a Tab és a Del billentyűt, valamint az összes normál karaktert nem lehet így figyelmen kívül hagyni)
33	A lenyomott billentyűt tárolja a szövegben
34	Sortörésváltás Ha a MEMOEDIT szerkesztés közben a sor végéhez ér, és a sortörés be van kapcsolva, akkor egy „lágy" sorvégejelet tesz a sor végére, és a következő sor elejére ugrik. Ha a sortörés nincs bekapcsolva, akkor a sor végéhez érve addig vár, amíg a Returnt le nem nyomja. A „lágy" sorvégejelek hibát okozhatnak a Clipper más parancsaiban, ezért a szerkesztés befejeztével érdemes ezeket átalakítani a HARDCR( ) funkcióval. Az alapértelmezés a bekapcsolt sortörésüzemmód. Bizonyos esetben hibásan működik ez a funkció. Ha egy szó hosszabb a kijelölt ablak vagy a képernyő szélességénél, és abba

akar beszúrni, akkor a begépeléskor a függvény a szó utolsó betűit átviszi a következő sorba, de közéjük <space> karaktert szúr!

35 Scrollváltás

100 A következő szóra áll
(a 2-es kód funkciója -> újraformázás)

101 Az ablak elejére áll
(a 23-as kód funkciója -> elmenti a változásokat és kilép)

Megadhatja a használt *sorhossz*-t is. A megadott érték akár nagyobb is lehet, mint a megadott ablak szélessége, ilyenkor automatikus scrollt hajt végre a MEMOEDIT, ha eléri az ablak határát. Az ablak szélességének alapértelmezése az 1.

Ezután a *tabulátorhossz*-t adhatja meg. A függvény a paraméterben megadott számú <space> karaktert szúr be minden egyes TAB, ill. Ctrl-J lenyomáskor a képernyőn megjelenő szövegbe. A fájlban viszont a 9-es kódú karaktert rögzíti! A paraméter értékének alapértelmezése a 4.

A következő paraméterrel megadhatja azt a sort, amelyre a szerkesztés előtt a MEMOEDIT-nek rá kell állnia. A paraméter alapértelmezése az 1.

Megadhatja azt az oszlopot is, amelyre a szerkesztés előtt a MEMOEDIT-nek rá kell állnia. A paraméter alapértelmezése az 1.

A következő két paraméterrel azt a relatív helyet adhatja meg, ahol a szerkesztés előtt a kurzor áll majd. (Ez a cím a megadott ablakhoz képest relatív.) A paraméter értékének alapértelmezése 0,0.

A szerkesztendő szöveg kivételével minden paraméter elhagyható. Sőt, ha csak néhány paramétert szeretne megadni, akkor arra is van lehetőség. Ilyenkor a be nem állított paraméterek helyett egy üres változót kell megadni (pl. DUMMY).

A szerkesztés közben használható billentyűk:

Billentyű	**Kód**	**Funkció**
Tab Ctrl-J	9	Megadott számú <space> karaktert szúr a szövegbe
Felfelé nyíl Ctrl-E	5	Egy sorral feljebb áll

Billentyű	Kód	Funkció
Lefelé nyíl Ctrl-X	24	Egy sorral lejjebb áll
Balra nyíl Ctrl-S	19	Egy karakterrel balra
Jobbra nyíl Ctrl-D	4	Egy karakterrel jobbra
Ctrl-balra nyíl Ctrl-A	26 1	Az előző szó elejére áll
Ctrl-jobbra nyíl Ctrl-F	2 6	A következő szó elejére áll
Home	1	A sor elejére áll
End	6	A sor végére áll
Ctrl-Home	29	Az ablak elejére áll
Ctrl-End	23	Az ablak végére áll
PgUp	18	Az előző ablak
PgDn	3	A következő ablak
Ctrl-PgUp	31	A szöveg elejére áll
Ctrl-PgDn	30	A szöveg végére áll
Ctrl-Y	25	Törli az aktuális sort
Ctrl-T	20	Törli a jobbra lévő szót
Ctrl-B	2	A szöveget újraformázza
Ctrl-W	23	Elmenti a változásokat
Esc	27	Eldobja a változásokat

A kurzormozgató nyilak a csak megjelenítésmódban is használhatók.

Lásd még: ACHOICE( ), DBEDIT( ), HARDCR( ), MEMOLINE( ), MEMOREAD( ), MEMOTRAN( ), MEMOWRIT( ), MLCOUNT( ).

MEMOLINE '87

MEMOLINE (<amit darabol>,<sorhossz>.<melyik sort>;
[,<tabulátorhossz>[,<jelző>]])

Visszaadja az így kijelölt sort.

Paraméter	**értéke**
sorhossz	4-240
tabulátorhossz	2-(sorhossz-1)
jelző	.T. (nem veszi figyelembe a sztring CR és LF karaktereit)
	.F. (észreveszi és igazít)

Lásd még: HARDCR( ), MEMOEDIT( ), MEMOREAD( ), MEMOTRAN( ), MEMOWRIT( ), MLCOUNT( ).

MEMOREAD

MEMOREAD (<fájlnév>)

Szöveges állományt olvas be egy változóba.

Lásd még: REPLACE, HARDCR( ), MEMOEDIT( ), MEMOLINE( ), MEMOTRAN( ), MEMOWRIT( ), MLCOUNT( ).

MEMORY

MEMORY (0)

A szabad memória méretét adja vissza kbájtban. Itt kötelezően csak a 0-t adhatja meg paraméterként!

MEMOTRAN

MEMOTRAN (<karakterlánc>[,<mire>[,<mit cseréljen>]])

A megadott *karakterlánc*-ban egy megadott részsztring összes előfordulását kicseréli. A *mit* alapértelmezése a CHR(141)+CHR(10), a *mire*-é pedig a CHR(13)+CHR(10).

A szövegszerkesztők általában kétféle sorvégejelet használnak. A „kemény" sorvégejelet – CHR(13)+CHR(10) – és a „lágy" sorvégejelet – CHR(141)+CHR(10). A függvénnyel a lágy sorvégejeleket kemény sorvégejelekre cserélheti ki.

Lásd még: HARDCR( ), LCOUNT( ), MEMOEDIT( ), MEMOLINE( ), MEMOREAD( ), MEMOWRIT( ).

MEMOWRIT

MEMOWRIT (<fájlnév>,<karakterlánc>)

A *karakterlánc*-ot lemezre menti. Ha a mentés sikerült, *igaz* értéket ad. A *fájlnév*-et a karakterlánc első <space>-szel lezárt szaváből képzi.

Lásd még: HERDCR( ), MEMOEDIT( ), MEMOLINE( ), MEMOREAD( ), MEMOTRAN( ), MLCOUNT( ).

MIN

MIN (<kifejezés1>,<kifejezés2>)

A két kifejezés értéke közül a kisebbet adja vissza. Paraméterként dátumot, sőt logikai típusú kifejezést is megadhat. Ha logikai típusú paramétert ad meg, akkor a *hamis* (.F.) lesz a kisebb.

Lásd még: MAX( ).

MISC_ERROR '87

MISC_ERROR (<procedúranév>,<sorszám>,<hibaüzenet>,<modell>)

A hibakezelő rendszer függvénye. Ha a műveletekben eltérő típusú adatokkal dolgozik, akkor átveszi a vezérlést.

MLCOUNT '87

MLCOUNT (<mit>[,<sorhossz>])

Visszaadja, hány sorból áll egy karakterlánc az adott *sorhossz*-szal számolva. Hosszparaméter nélkül a karakterláncban lévő CR-rel és LF-fel lezárt sorok számát adja vissza.

Lásd még: HARDCR( ), MEMOEDIT( ), MEMOLINE( ), MEMOREAD( ), MEMOTRAN( ), MEMOWRIT( ).

MLPOS '87

MLPOS (<karakteres kifejezés>[,<sorhossz>[,<melyik sort>;
[,<tabulátorhossz>[,<jelző>]]]])

A megadott sor kezdőpozícióját adja vissza.

Paraméter	**értéke**
sorhossz1	4-240
tabulátorhossz	2-(sorhossz-1)
jelző	.T. (nem veszi figyelembe a sztring CR és LF karaktereit)
	.F. (észreveszi és igazít)

Lásd még: MEMOLINE( ).

MOD

MOD (<osztandó>,<osztó>)

A két szám osztási maradékát adja vissza. *Figyelem!* Az eredmény eltérhet a % operátor eredményétől! A forrása az Examplep.prg.

Példa:

Operátor		Funkció
3 % 3 ::= 0.00		MOD(3, 3) ::= 0
3 % 2 ::= 1.00		MOD(3, 2) ::= 1
3 % 1 ::= 0.00		MOD(3, 1) ::= 0
3 % 0 ::= 0.00	<->	MOD(3, 0) ::= 3
3 % -1 ::= 0.00		MOD(3,-1) ::= 0
3 % -2 ::= 1.00	<->	MOD(3,-2) ::= -1
3 % -3 ::= 0.00		MOD(3,-3) ::= 0
-3 % 3 ::= 0.00		MOD(-3, 3) ::= 0
-3 % 2 ::= -1.00	<->	MOD(-3, 2) ::= 1
-3 % 1 ::= 0.00		MOD(-3, 1) ::= 0
-3 % 0 ::= 0.00	<->	MOD(-3, 0) ::= -3
-3 % -1 ::= 0.00		MOD(-3,-1) ::= 0
-3 % -2 ::= -1.00		MOD(-3,-2) ::= -1
-3 % -3 ::= 0.00		MOD(-3,-3) ::= 0
3 % 3 ::= 0.00		MOD(3, 3) ::= 0
2 % 3 ::= 2.00		MOD(2, 3) ::= 2
1 % 3 ::= 1.00		MOD(1, 3) ::= 1
0 % 3 ::= 0.00		MOD(0, 3) ::= 0
-1 % 3 ::= -1.00	<->	MOD(-1, 3) ::= 2
-2 % 3 ::= -2.00	<->	MOD(-2, 3) ::= 1
-3 % 3 ::= 0.00		MOD(-3, 3) ::= 0
3 % -3 ::= 0.00		MOD(3,-3) ::= 0
2 % -3 ::= 2.00	<->	MOD(2,-3) ::= -1
1 % -3 ::= 1.00	<->	MOD(1,-3) ::= -2
0 % -3 ::= 0.00		MOD(0,-3) ::= 0
-1 % -3 ::= -1.00		MOD(-1,-3) ::= -1
-2 % -3 ::= -2.00		MOD(-2,-3) ::= -2
-3 % -3 ::= 0.00		MOD(-3,-3) ::= 0

Lásd még: %.

MONTH

MONTH (<dátumkifejezés>)

A dátumváltozóból a hónap számával tér vissza.

Lásd még: CDOW( ), CMONTH( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOC( ), DTOS( ), YEAR( ).

MULTI_FORM '87

MULTI_FORM (<tömbnév>)

A Clipper nem támogatja a többoldalas formátumállományok írását, erre ezt a függvényt használhatja. Egy tömböt fel kell tölteni az előre elkészített formátumállományok neveivel. A tömb nem tartalmazhat üres elemet, mert az hibát okoz! A képernyők váltására a PgUp-PgDn billentyűkombinációt használhatja. A függvény az aktuális képernyő sorszámát adja vissza.

Figyelem! A függvény három esetben fejezi be a működését:

- ha ESC-et nyomott le;
- ha az első képernyőn a PgUp billentyűt nyomta le;
- ha az utolsó képernyőt is befejezte (akár a PgDn-nal).

A forrása az Apndxj.prg.

Lásd még: SET FORMAT.

NDX

NDX (<sorszám>)

Egy indexfájlnevet generál. A Clipper nem jegyzi meg a megnyitott állományok nevét, nem úgy, mint a dBASE. Ezt a függvényt is csak kompatibilitása miatt említjük meg. A generált indexnév az 'NTXn' lesz.

NETERR

NETERR ( )

Hálózatban a hálózatot érintő utasítások végrehajtásának sikerességét ellenőrizheti vele. Ha hiba történt, akkor *igaz* értéket ad.

Lásd még: APPEND BLANK, USE...EXCLUSIVE, FLOCK( ), RLOCK( ).

NETNAME

NETNAME ( )

A felhasználó hálózati nevét adja vissza. Csak IBM PC Network hálózatban működik!

NET_USE '87

NET_USE (<fájlnév>,<nyitásmód>,<várakozási idő>)

Hálózatban egy állományt nyit meg. Ha a *nyitásmód* .T., akkor az állományt kizárólagos használatra nyitja meg. Mindig a megadott ideig próbálkozik. Ha itt 0-t ad meg, akkor addig vár, amíg sikerül megnyitnia az állományt. A NET_USE a NETERR( )-rel ellenőrzi a megnyitás sikerességét. Ha a függvénynek sikerült megnyitnia az állományt, akkor *igaz* értéket ad vissza.

Figyelem! A NET_USE nem nyitja meg az indexállományokat! A forrása a Locks.prg.

Lásd még: SET INDEX, USE, NETERR( ).

NEXTKEY '87

NEXTKEY ( )

A következő – még nem beolvasott – karaktert adja vissza a billentyűpufferből anélkül, hogy onnan törölné.

Lásd még: INKEY( ), LASTKEY( ).

OPEN_ERROR '87

OPEN_ERROR (<procedúranév>,<sorszám>,<hibaüzenet>,;
<modell>,<fájlnév>)

Hibakezelő függvény. Ha nem sikerül megnyitni egy állományt, akkor elindul.

OS

OS ( )

A használt operációs rendszer nevét adja vissza. Konstans értékként az 'MS/PC-DOS'-t. A forrása az Examplep.prg.

PAD

PAD (<karakterlánc>,<hossz>)

A *karakterlánc*-ot a megadott *hossz*-ra egészíti ki úgy, hogy a végéhez <space> karaktereket fűz. A forrása az Examplep.prg.

PCOL

PCOL ( )

A nyomtatófej oszloppozícióját adja vissza.

Lásd még: COL( ), PROW( ), ROW( ), SETPRC( ).

PCOUNT

PCOUNT ( )

A rutin a program kapott paramétereinek számát adja vissza.

Lásd még: DO..WITH, PARAMETERS.

PRINT_ERRO '87

PRINT_ERRO (<procedúranév>,<sorszám>)

Hibakezelő függvény. Ha a nyomtató még nem üzemkész, akkor elindul.

PROCFILE '87

PROCFILE ( )

A futó program vagy eljárás forráskódját tartalmazó állomány nevét adja vissza.

Lásd még: PROCLINE( ), PROCNAME( ).

PROCLINE

PROCLINE ( )

A futó rutin éppen végrehajtott sorának számát adja vissza. Ha a rutint -l opcióval fordítja, akkor a függvény által visszaadott érték 0.

Lásd még: PROCFILE( ), PROCNAME( ).

PROCNAME

PROCNAME ( )

Az éppen végrehajtott rutin nevét adja vissza.

Lásd még: PROCFILE( ), PROCLINE( ).

PROW

PROW ( )

A nyomtatófej sorpozícióját adja vissza.

Lásd még: SET DEVICE, SET PRINT, SET PRINTER, COL( ), PCOL( ), ROW( ), SETPRC( ).

RAT

RAT (<mit>,<hol>)

A karakterlánc utolsó előfordulását keresi meg.

Lásd még: AT( ), STRTRAN( ), SUBSTR( ), $.

READEXIT '87

READEXIT ([<logikai érték>])

Beállíthatja, hogy a felfelé, ill. lefelé nyíl megszakítsa-e a GET utasítást, vagy sem. Ha a logikai kifejezés *igaz*, akkor befejezi a szerkesztést, különben a GET utasítás nem veszi figyelembe kilépő karakterként. Az alapértelmezés a *hamis*. Az előző értéket adja vissza.

Lásd még: @...GET, @...SAY, READ, SET KEY, READINSERT( ).

READINSERT '87

READINSERT ([<logikai érték>])

Ha a *logikai érték* .T., akkor a READ és a MEMOEDIT beszúrásmódban, ha .F., akkor felülírásmódban dolgozik majd. Az alapértelmezés a *hamis*. Az előző értéket adja vissza.

READKEY

READKEY ( )

Csak dBASE kompatibilitása miatt említjük meg. A READ utasítást több billentyűvel is befejezheti, ezeknek a billentyűknek az értékét adja vissza ez a függvény, és jelzi, ha a READ közben megváltozik a mező tartalma. A Clipper ez utóbbit az UPDATED( ) függvénnyel oldja meg, a dBASE pedig a befejező billentyű kódjához 256-ot ad. Ez az eltérés a Clipper és a dBASE között. A Clipper kevesebb billentyűnek engedi befejezni a READ utasítást, csak a PgUp-nak, a PgDn-nak, az Esc-nek, a ^W-nek, az Enter-nek, a ^PgUp-nak, a ^PgDn-nak, ill. akkor, ha az utolsó mezőt kitöltötte. A READKEY függvény is csak ezeket jelzi, minden más esetben 0 értéket ad vissza. A forrása az Examplep.prg.

A befejezés módja	értéke
PgUp	6
PgDn	7
Ctrl-PgUp	34
Ctrl-PgDn	35
Esc	12
Ctrl-W	14
Enter	15
Kitöltött mező	15

Lásd még: LASTKEY( ), UPDATED( ).

READVAR

READVAR ( )

Az éppen módosított változó vagy mező nevét adja vissza. A GET vagy a MENU utasítást megszakító függvény használhatja.

Lásd még: @...GET, READ, SET KEY.

RECCOUNT

RECCOUNT ( )

Az éppen aktív adatbázis rekordjainak számával tér vissza.

Lásd még: LASTREC( ).

REC_LOCK '87

REC_LOCK (<várakozási idő>)

A megadott ideig próbálkozik az aktuális rekord lezárásával. Ha sikerül, akkor *igaz* értéket ad vissza, különben *hamis*-at. Ha a várakozási idő 0, akkor addig vár, amíg a lezárás sikeres lesz. A forrása a Locks.prg.

RECNO

RECNO( )

Az aktuális adatbázis aktív rekordjának fizikai sorszámát adja vissza.

Lásd még: BOF( ), EOF( ).

RECSIZE

RECSIZE ( )

Az aktuális állomány rekordhosszát adja vissza. A hossz a rekord fizikai hossza, a törlést jelző bájttal együtt.

REPLICATE

REPLICATE (<ismétlődő karakter>,<darab>)

Azonos karakterekből álló sztringet lehet előállítani vele.

Lásd még: SPACE( ).

RESTSCREEN '87

RESTSCREEN (<bal felső sor>,<bal felső oszlop>,;
<jobb alsó sor>,<jobb alsó oszlop>,;
<változónév>)

A SAVESCREEN( ) függvénnyel elmentett ablakot visszaírja az itt megadott ablakba. A két ablak helye különbözhet, de a területüknek meg kell egyeznie!

Figyelem! A SAVE SCREEN utasítással elmentett képernyőt ne próbálja meg így helyreállítani!

Lásd még: RESTORE SCREEN, SAVE SCREEN, SAVESCREEN( ).

RIGHT

RIGHT (<karakterlánc>,<hossz>)

Levágja a megadott *karakterlánc* végéről a *hossz* hosszúságú darabot. Ezt az új karakterláncot adja vissza.

Lásd még: LEFT( ), SUBSTR( ).

RLOCK

RLOCK ( )

Az aktuális állományban lezár egy rekordot. Ha volt előzőleg lezárt rekord, akkor az most felszabadul. Egyszerre csak egy rekord lehet lezárva egy fájlban.

Lásd még: SET EXCLUSIVE, UNLOCK, USE...EXCLUSIVE, FLOCK( ), LOCK( ).

ROUND

ROUND (<kerekítendő szám>,<tizedesek>)

A számokat a megadott számú tizedes helyi értékig kerekíti.

Lásd még: INT( ).

ROW

ROW ( )

A kurzor sorpozícióját adja vissza. Ez a függvény által visszaadott értéket a @... utasítások nem állítják át!

Lásd még: @...GET, @...SAY, COL( ), PCOL( ), PROW( ).

RTRIM

RTRIM (<karakterlánc>)

A jobb oldali befejező szóközkaraktereket vágja le a *karakterlánc*-ból.

Lásd még: ALLTRIM( ), LTRIM( ), TRIM( ).

SAVESCREEN '87

SAVESCREEN (<bal felső sor>,<bal felső oszlop>,;
<jobb alsó sor>,<jobb alsó oszlop>)

A megadott ablakot egy változóba menti el. Ezt csak a RESTSCREEN( ) függvénnyel ajánlatos visszaírni. Az így kimentett ablakot a RESTORE SCREEN FROM paranccsal csak akkor tudja visszamenteni, ha a teljes képernyőt elmentette.

Lásd még: RESTORE SCREEN, SAVE SCREEN, RESTSCREEN( ).

SCROLL

SCROLL (<bal felső sor>,<bal felső oszlop>,;
<jobb alsó sor>,<jobb alsó oszlop>,;
<sorszám>)

A képernyőt görgeti (scrollozza) a megadott ablakban. Ha a *sorszám* 0, akkor törli az ablakot. Ha a *sorszám* pozitív, akkor felfelé, különben lefelé görget.

Lásd még: @...BOX, @...CLEAR, @...TO...DOUBLE.

SECONDS

SECONDS ( )

A rendszeridőt másodpercek.századmásodpercek formájában adja vissza.

Lásd még: SEC( ), TIME( ).

SECS

SECS (<idősztring>)

A karakteresen (óó:pp:mm) megadott időt másodpercekre számítja át. Az elválasztó karakterek bármilyenek lehetnek! A forrása az Examplep.prg.

Lásd még: TIME( ).

SELECT

SELECT ([<munkaterület>])

Ha a *munkaterület*-nek ALIAS nevet ad meg a karakterláncban, akkor ellenőrzi, van-e ilyen névvel megnyitott állomány. Ha van, akkor visszaadja a munkaterület számát, különben 0-t ad vissza. Amennyiben egy állományt több munkaterületen is megnyitott, akkor a függvény a később megnyitott állomány munkaterületének számát adja vissza. Ha számot ad meg, és létezik az a munkaterület, akkor arra áll rá.

Példa:

```
TORZS='UTORZS'
SELECT(SELECT(TORZS))
* Csak így lehet egy névvel munkaterületre pozicionálni!
```

Lásd még: SELECT, USE, ALIAS( ).

SETCANCEL '87

SETCANCEL ([<logikai érték>])

Az Alt+C billentyűk működését állíthatja be és kérdezheti le vele. Ha a paraméter *igaz*, akkor „él", különben nem. Az alapértelmezés az *igaz*. A függvény az előző beállítás értékét adja vissza.

Lásd még: SET ESCAPE, ALTD( ).

SETCOLOR '87

SETCOLOR ([<karakterlánc>])

A színek beállítását végezheti el vele. Paraméterei megegyeznek a SET COLOR utasításával. Minden esetben a betűs rövidítéseket, nem pedig a színszámokat adja vissza. Az előző beállítás értékét is visszaadja, ezek az értékek azonban néhány esetben eltérnek!

Megadott érték	Visszaadott érték
'RB'	'BR'
'W'	'BGR'
0	" (semmi)

Lásd még: SET COLOR, SET COLOUR.

SETPRC

SETPRC (<sor>,<oszlop>)

A nyomtató sor-, ill. oszlopértéket állítja be. A *sor* a PROW, az *oszlop* pedig a PCOL értékeit állítja be. Akkor érdemes használni, ha a nyomtatóra vezérlő jelsorozatot küld. Ilyenkor ezek az értékek megváltoznak, de a fej nem mozdul el a helyéről.

Lásd még: SET DEVICE, PCOL( ), PROW( ).

SOUNDEX

SOUNDEX (<karakterlánc>)

A megadott *karakterlánc*-ot úgy kódolja át, hogy a hasonló hangzású szavak azonos kódot kapnak. Ezzel a függvénnyel az indexelt állományokban már akkor is meg lehet találni egy keresett nevet, ha nem emlékszik rá pontosan. A *karakterlánc*-nak négy karakternél hosszabbnak kell lennie! *Figyelem!* Csak betűk lehetnek a *karakterlánc*-ban, különben a funkció hibásan működik, és 0 hosszúságú *karakterlánc*-ot ad vissza! A forrása az Examplec.c.

Lásd még: INDEX, DESCEND( ).

SPACE

SPACE (<darabszám>)

Olyan karakterláncot hoz létre, amely a megadott számú szóközből áll.

Lásd még: REPLICATE( ).

SQRT

SQRT (<szám>)

A megadott pozitív *szám* négyzetgyökével tér vissza.

Lásd még: SET DECIMAL, SET FIXED.

STR

STR (<szám>[,<hossz>[,<tizedesek száma>]])

A megadott *szám*-ot karakterláncca alakítja át. A *hossz* alapértelmezése a 10.

Lásd még: SUBSTR( ), VAL( ).

STRTRAN '87

STRTRAN (<hol>,<mit>[,<mire>][,<honnantól>][,<mennyit>]

Törli a karakterláncot, vagy egy megadottra cseréli ki. Az alapértelmezés a törlés, az elsőtől kezdve minden karaktert töröl.

Lásd még: SET EXACT, AT( ), RAT( ), SUBSTR( ), $.

STRZERO

STRZERO (<szám>,[<hossz>[,<tizedesek száma>]])

A megadott *szám*-ot karakteres formában adja vissza, vezető nullákkal kiegészítve. A forrása az Examplep.prg.

STUFF

STUFF (<karakterlánc>,<honnantól>,<mennyit>,<mire>)

Részkarakterláncot helyettesíthet vele. A első *karakterlánc*-ban – az első számtól kezdve – a második szám hosszúságú részt kicseréli a második *karakterlánc*-cal. A forrása az Examplec.c.

SUBSTR

SUBSTR (<karakterlánc>,<honnantól>[<,mennyit>])

A *karakterlánc* meghatározott részét vágja le. Ha a kezdő- vagy a befejező pozíció túlmutat a *karakterlánc*-on, akkor a függvény 0 hosszúságú karakterlánc-ot ad vissza.

Lásd még: AT( ), RAT( ).

TIME

TIME ( )

A rendszeridővel tér vissza, óra:perc:másodperc alakban.

Lásd még: DATE( ), SECONDS( ).

TONE '87

TONE (<frekvencia>,<időtartam>)

Az IBM PC belső hangszóróját szólaltatja meg. Az *időtartam* 1/18 másodpercben értendő.

Néhány hang hozzávetőleges frekvenciaértéke:

C'	130.80	C	261.70	
C'#	138.60	C#	277.20	
D'	146.80	D	293.70	
D'#	155.60	D#	311.10	
E'	164.80	E	329.60	
F'	174.60	F	349.20	
F'#	185.00	F#	370.00	
G'	196.00	G	392.00	
G'#	207.70	G#	415.30	
A'	220.00	A	440.00	(normál A hang)
A'#	233.10	A#	466.20	
B'	246.90	B	493.90	

Lásd még: SET BELL.

TRANSFORM

TRANSFORM(<kifejezés>,<formátum>)

A megadott *kifejezés*-t a megadott *formátum*-ra alakítja át. A *formátum*-ok megegyeznek a SAY utasítás PICTURE paraméterével.

Lásd még: @...PICTURE, LOWER( ), STR( ), UPPER( ).

TRIM

TRIM (<karakterlánc>)

Levágja a szóközöket a *karakterlánc* végéről.

Lásd még: ALLTRIM( ), LTRIM( ), RTRIM( ), SUBSTR( ).

TSTRING

TSTRING (<másodpercek>)

Másodpercről óó:pp:mm formájú sztringgé alakítja a megadott időt. A forrása az Examplep.prg.

Lásd még: SECS( ).

TYPE

TYPE (<karakteres kifejezés>)

A memóriaváltozó vagy az adatbázismező adatának típusával tér vissza. A *karakteres kifejezés* változónevet, mezőnevet, ill. bármilyen kifejezést tartalmazhat. A tömb elemei külön-külön is vizsgálhatók.

Típus	Megnevezés
A	tömb
C	karakter
D	dátum
L	logikai
M	memo
N	numerikus
U	definiálatlan
UE	szintaktikai hiba
UI	nem definiált függvény

UE az eredmény akkor, ha egy IF vagy egy IIF függvényt vizsgál, és a végrehajtandó műveletben szintaktikai hiba van.

UI az eredmény akkor, ha egy függvény visszatérítési értékére kíváncsi, de a függvény nem elérhető, nincs a programhoz szerkesztve.

Példa:

```
DECLARE TOMB[100]
TOMB[1] = 'ABCD'
TOMB[2] = 2
TOMB[3] = .F.
? TYPE('TOMB')                              && 'A'
? TYPE('TOMB[2]')                           && 'N'
? TYPE('IF(TOMB[3],,,JO",TOMB[10])')        && 'UI'
```

UNDEF_ERRO '87

UNDEF_ERRO (<procedúranév>,<sorszám>,<hibaüzenet>,<modell>,<'változónév'>)

Hibakezelő függvény. Akkor kapja meg a vezérlést, ha egy olyan változóra hivatkozik, amelyet még nem hozott létre.

UPDATED

UPDATED ( )

Ha az utolsó READ utasításban bármely GET mező értéke megváltozik, akkor *igaz* értéket ad vissza.

Lásd még: @...GET, READ.

UPPER

UPPER (<karakterlánc>)

A kisbetűkkel írt karaktereket nagybetűkké alakítja. Csak a standard karakterkészlet betűivel dolgozik!

Lásd még: LOWER( ).

USED '87

USED ( )

Ha az aktuális munkaterületen már megnyitott egy állományt, akkor *igaz* értéket ad vissza.

Lásd még: SELECT, USE, NETERR( ), SELECT( ).

VAL

VAL (<karakterlánc>)

A *karakterlánc* formában megadott számot szám típusú adatként adja vissza.

Lásd még: SET DECIMALS, STORE, STR( ), SUBSTR( ), TYPE( ).

VALIDTIME '87

VALIDTIME (<idősztring>)

Az óó:pp:mm formában megadott időről eldönti, hogy logikailag helyes-e. Ha helyes, akkor *igaz* értéket ad. Ha az óra kisebb, mint 24, ha a perc és a másodperc kisebb, mint 60, akkor, helyes. Az elválasztójel bármi lehet. A forrása az Apndxj.prg.

Lásd még: TIME( ).

VERSION

VERSION ( )

A Clipper rendszer éppen használt változatának nevét adja vissza szöveges formában:

„Clipper, Summer '87"

A forrása az Exampleg.prg.

WORD

WORD (<maximum 16 bites szám>)

A megadott *szám*-ot a Clipper belső formátumáról 16 bites számra alakítja át.

Lásd még: CALL.

YEAR

YEAR (<dátum>)

A megadott dátumból az évet adja vissza.

Lásd még: SET CENTURY( ), CDOW( ), CMONTH( ), CTOD( ), DATE( ), DAY( ), DOW( ), DTOC( ), DTOS( ), MONTH( ).

BSZ SZÉCHENYI KÖNYVTÁR

Tartalomjegyzék

350,– Ft

Az adatbázis-kezelő nyelvek közül talán a leggazdagabb kihasználási lehetőséget a Clipper nyújtja. Lehetővé teszi a tömbök a képernyőkezelő függvények használatát és még számos más programozást segítő, egyszerűsítő utasítások, függvények alkalmazását.

Néhány cím a tartalomból:

A Clipper technikai leírása
A Clipper állományainak szerkezete
Clipper parancsok

CLIPPER