Angster Erzsébet

Kertész László

Turbo Pascal 6.0

'A' .. 'Z'

Angster Erzsébet – Kertész László

Turbo Pascal 6.0
'A'..'Z'

Első kiadás

Felelős kiadó: Angster Erzsébet, Kertész László

Felelős vezető: Wilpert Gábor
92/162

ISBN 963 450 081 1

Előszó

Könyvünk a Turbo Pascal programozási nyelv összefoglalója. E lapozgató a Turbo Pascal 6.0 könyvünk kivonata, s azok számára készült, akik lemezeik közé szeretnék rejteni kis helyen elférő „tudományukat", esetleg érettségire készülnek e programozási nyelvből. A Neumann János Számítástechnikai Szakközépiskola ajánlása alapján ugyanis a Munkaügyi Minisztérium e könyv használatát engedélyezte érettségiken.

Bár a Borland cég szinte évenként dobja piacra a különféle újdonságokat kínáló Turbo Pascal verziókat, ezen újdonságokat sokszor csak a „profik" használják. Könyvünk nagyobb része szinte verziótól független, így még sokáig használhatják azok a programozók, akik a Turbo Pascal alapjait tanulják. Ugyanilyen meggondolásból nyugodtan forgathatják kiadányunkat a Turbo Pascal korábbi verzióit (4.0-tól) használók is.

A lapozgatóba csak azokat az információkat tettük, melyeket egy programozó – véleményünk szerint – rendszeresen használ. Elhagytuk a példákat és a magyarázó szövegeket, és a környezettel foglalkozó fejezetekből csak a billentyűk használatára vonatkozó táblázatok maradtak meg. A könyv tartalmazza a Borland cég által kifejlesztett Turbo Access és Turbo Sort egységek leírását is, melyek nem tartoznak szorosan a nyelvhez.

Kívánjuk, hogy az Olvasó minél több örömét lelje könyveink használatában!

Kérjük, hogy észrevételeit, javaslatait írja meg az alábbi címre:

Angster-Kertész
Budapest, Pf 701/234
1399.

a szerzők

Szintaktika 1
A Pascal program elemei 1
Definíciók 5
Egyszerű utasítások 8
Struktúrált utasítások 9
A program felépítése 13
Az egység felépítése 14
Eljárás felépítése 15
Függvény felépítése 16
Deklarációk érvényességi köre 17

Adattípusok, műveletek 19
Definíciók 19
Karakter típus 21
Egész típusok 21
Logikai típus 23
Intervallum típus 24
Valós típusok 25
Karakterlánc típus 28
Tömb típus 28
Rekord típus 30
Halmaz típus 31
Szöveges állomány 32
Típusos állomány 33
Típusnélküli állomány 33

Típusos mutató 33
Típusnélküli mutató 34
Eljárás típus 35
Objektum típus 36
Típusazonosság 39
Típus-kompatibilitás 39
Értékadás-kompatibilitás 40
Műveletek 41
Kifejezések kiértékelése 46
Értékadás, értékparaméter átadás 47

Direktívák 49
Kapcsoló- és paraméter direktívák 49
Feltételes direktívák 54
Előre definiált feltételes szimbólumok 56
Speciális direktívák 56

System egység 61
Konstansok 61
Változók 61
Eljárások, függvények 62

Crt egység 93
Konstansok 93
Változók 93
Eljárások, függvények 94

Printer egység **101**
Változó 101

Dos egyég **103**
Típusok 103
Konstansok 105
Változó 105
Eljárások, függvények 106

Graph egység **115**
Típusok 115
Konstansok 116
Változók 120
Eljárások, függvények 120

Overlay egység **141**
Konstansok 141
Változók 141
Eljárások, függvények 142

TAccess egység **145**
Típusok 145
TABuild típusok 145
Konstansok 145
TAccess.Def konstansai 145
Változók 146
Eljárások, függvények 146

TAHigh egység **154**
Típus 154
Konstansok 154
Változó 154
Eljárások, függvények 154

TSort egység **159**
Eljárások, függvények 159

Assembler **163**
Direktívák 163
Szimbólumok 163
Prefixek 163
Assembly utasítások 163

Fordítási hibakódok **183**

Futási hibakódok **195**
DOS hibák 195
B/K hibák 196
Kritikus hibák 197
Fatális, mindig programleállást okozó hibák 198
Turbo Access futási hibakódok 200

Keretrendszer – forróbillentyűk **201**

A szövegszerkesztő parancsai **204**
Kurzormozgatás 204

Beszúrás/törlés 205
Blokk-kezelés 205
Egyéb funkciók 206

Memóriatérképek 209
Az IBM XT/AT vázlatos memóriafelosztása 209
A Turbo Pascal által használt memória térképe 210
A szubrutinverem felépítése 211

Grafikus üzemmódok jellemzői 212
Billentyűzet kódtábla 214
Nyomtató vezérlőkódok 217
ASCII tábla 218

Tárgymutató

A Pascal program elemei

Egy Pascal program a következő elemi egységekből épül fel:

- Szimbólumok
- Fenntartott szavak
- Azonosítók
- Címkék
- Számkonstansok
- Szövegkonstansok
- Elválasztó jelek: egy vagy több szóköz, kontrol karakterek (pl. soremelés illetve megjegyzés).

Elválasztó jel nem lehet elem része – kivétel a szövegkonstans.

A Turbo Pascal fordító az azonosítók és fenntartott szavak tekintetében a kis és nagybetűkre nem érzékeny – azok megválasztása csak a programozási stílusra jellemző.

Szimbólumok

- Nagybetűk, kisbetűk, számjegyek: **A..Z,a..z,0..9**
- Hexadecimális számjegyek – $ után: **0..9,A..F,a..f**
- Egykarakteres szimbólumok: **+ – * / = < > [ ] . , ( ) :; ^ @ { } $ #**
- Kétkarakteres szimbólumok: **<= >= <> := .. (* *) (. .)**

Fenntartott szavak

AND	GOTO	RECORD
ASM	IF	REPEAT
ARRAY	IMPLEMENTATION	SET
BEGIN	IN	SHL
CASE	INLINE	SHR
CONST	INTERFACE	STRING
CONSTRUCTOR	LABEL	THEN
DESTRUCTOR	MOD	TO
DIV	NIL	TYPE
DO	NOT	UNIT
DOWNTO	OBJECT	UNTIL
ELSE	OF	USES
END	OR	VAR
FILE	PACKED	WHILE
FOR	PROCEDURE	WITH
FUNCTION	PROGRAM	XOR

Szabványos direktívák

ABSOLUTE	FAR	NEAR
ASSEMBLER	FORWARD	PRIVATE
EXTERNAL	INTERRUPT	VIRTUAL

A fenntartott szavakkal ellentétben a szabványos direktívák felüldeklarálhatók, ez azonban nem ajánlatos.

Programsor

A program elemeit sorokba írjuk. Egy sor maximálisan 126 karakter hosszú lehet.

Megjegyzés

Korlátlan hosszúságú szöveg, bárhány programsoron keresztül { } vagy (* *) között.

Direktíva

Fordítónak szóló utasítás. Olyan speciális megjegyzés, mely első karaktere a $ jel. Lásd Direktívák fejezet.

Azonosító

Az alábbiak egyikét azonosítja:

- Program
- Egység
- Eljárás
- Függvény
- Metódus
- Címke
- Konstans
- Típus
- Változó
- Rekord- és objektummező

Az azonosító kisbetűből, nagybetűből, számjegyből és aláhúzáskarakterből állhat. Akármilyen hosszú lehet, de csak az első 63 karaktere szignifikáns. Számjeggyel nem kezdődhet, a kis- és nagybetűket nem különbözteti meg a fordító.

Minősített azonosító

programazonosító.azonosító | *egységazonosító.azonosító*

A program és egység ugyanolyan azonosítóinak megkülönböztetése.

Címke

Számjegysorozat 0 és 9999 között vagy azonosító. A program bármely utasítása megjelölhető címkével, mely mögé kettőspontot kell írni. A megadott címkére a Goto utasítással lehet ugrani. A címkét a Label kulcsszó után még abban a blokkban deklarálni kell, amelyben hivatkozunk rá. A Goto utasítás csak indokolt esetben használandó! Minden program megírható a struktúrált utasítások segítségével, a Goto használata a legtöbb esetben áttekinthetetlenné teszi a programot!

Számkonstans

- Decimális egész: *[+/–] számjegyek* (–2147483648..2147483647)
- Hexadecimális egész: *[+/–]* **$** *max. 8 hexadecimális számjegy*
- Valós ($m*10^e$): *[+/–] mantissza [***E/e** *[exponens]]*

  ahol mantissza: *[+/–] számjegyek [.számjegyek]*,

  exponens: *[+/–] számjegyek*

Szövegkonstans (karakterlánc)

ASCII karakterek sorozata. A karaktersorozatot megadhatjuk aposztrófok között felsorolva. A kettős aposztróf egyet reprezentál. Az aposztrófok közé írt szöveg(ek) elé, mögé és közé beágyazhatunk kontroll illetve ASCII karaktereket is. Kontroll karakter: ^*karakter*; ASCII karakter: #*ASCII kód*, ASCII kóddal megadott karakter. A megjeleníthető karaktereket általában aposztrófok közé írjuk.

Definíciók

Formális paraméterlista

*[***Var***] azonosító [,azonosító...]* **:** *típusazonosító*
[; ***[Var]*** *azonosító [,azonosító...]* **:** *típusazonosító ...]*

Aktuális paraméterlista

kifejezés / változó [, kifejezés / változó ...]

Függvényhívás

- *Függvényazonosító [(aktuális paraméterlista)]*
- *Függvényváltozó [(aktuális paraméterlista)]*
- *Objektum-azonosító.függvénymetódus [(aktuális paraméterlista)]*
- *Konstruktor-azonosító [(aktuális paraméterlista)]*

A kiterjesztett szintaktika használata esetén ($X+) a függvényeket lehet eljárásként is hívni. Ekkor a visszaadott függvényérték veszendőbe megy. Lásd Direktívák, $X-/+.

Típuskonvertált változó

Típusazonosító(változóhivatkozás)

A változóhivatkozás és típusazonosító által meghatározott memóriaméreteknek meg kell egyezniük. Pl. Byte(Karlanc[0]).

Típuskonvertált érték

Típusazonosító(kifejezés)

Típusazonosító és kifejezés típusa sorszámozott vagy mutató lehet. Pl. LongInt(2).

Változóhivatkozás

Változó (a típusos konstans is változó!), struktúrált vagy karakterlánc típusú változó egy komponense, illetve mutató típusú változó vagy függvényérték által mutatott dinamikus változó:

- *változó-azonosító [index / .mezőnév / ^ ...]*
- *típusos konstans azonosítója [index / .mezőnév / ^ ...]*
- *típuskonvertált változó [index / .mezőnév / ^ ...]*
- *függvényhívás^ [index / .mezőnév / ^ ...]*

ahol index: [*kifejezés [, kifejezés ...]*]

Halmazkonstruktor

[*kifejezés[..kifejezés][,kifejezés[..kifejezés]...]*]

ahol *kifejezés* típusa sorszámozott, sorszáma a 0..255 intervallumban lehet.
Pl. [1,5..8,N+5].

Kifejezés

Operandusok műveleti jelekkel (operátorokkal) összekapcsolt sorozata. Az operandusoknak a művelettől függően kompatibilis típusúaknak kell lenniük. A kifejezés

típusát az operandusok és a művelet(ek) együttesen határozzák meg. Lásd az Adattípusok, műveletek fejezetet.

Lehetséges operandusok:

- *[@]változóhivatkozás*
- *@eljárás-azonosító*
- *@függvényazonosító*
- *(kifejezés)*
- *függvényhívás*
- *halmazkonstruktor*
- *konstans*
- *típuskonvertált érték*

Konstans kifejezés

Olyan kifejezés, melyet a fordító ki tud értékelni. Konstans kifejezésben az alábbi függvények szerepelhetnek:

Abs	**Hi**	**Lo**	**Ord**	**Ptr**	**SizeOf**	**Swap**
Chr	**Length**	**Odd**	**Pred**	**Round**	**Succ**	**Trunc**

Konstans kifejezésben tilos változóra (és típusos konstansra) hivatkozni, valamint nem használható a @ operátor.

Feltétel

Logikai (Boolean típusú) kifejezés

Utasítás

[címke :] egyszerű utasítás / struktúrált utasítás

Egyszerű utasítások

Értékadás

változóhivatkozás := *kifejezés*
függvényazonosító := *kifejezés*

Eljáráshívás

eljárás-azonosító [(aktuális paraméterlista)]
eljárásváltozó [(aktuális paraméterlista)]

Metódushívás

objektum-azonosító.metódusazonosító [(aktuális paraméterlista)]
[objektumtípus.]metódusazonosító [(aktuális paraméterlista)]

ahol *metódusazonosító* eljárást, konstruktort, vagy destruktort azonosít

Vezérlésátadás

GOTO *címke*

Gépi kódú betét

INLINE(*elem [/ elem ...]*) mint az INLINE direktívánál

Assembler betét

ASM *utasítás [szeparátor utasítás ...]* **END**

ahol *utasítás*: Assembler utasítás; *szeparátor*: pontosvessző, új sor vagy Pascal megjegyzés

Struktúrált utasítások

Összetett utasítás – szekvencia

BEGIN
[utasítás [; utasítás ...]]
END

A BEGIN–END párost utasítás-zárójelnek is szokás nevezni.

Elágazás – szelekció

IF *feltétel* **THEN** *utasítás*
[**ELSE** *utasítás]*

- Ha ELSE-t elhagyjuk, akkor a szelekció egyágú.
- *Utasítás* lehet összetett utasítás, így egy ágon több utasítás is végrehajtható.
- *Utasítás* lehet újabb IF utasítás.
- ELSE mindig a legutolsó THEN párja.


Vigyázat! Ha a THEN utáni utasítást pontosvesszővel zárjuk, azzal a teljes IF utasítást lezárjuk.

Választás – többágú szelekció

CASE *szelektor* **OF**
állandó[..állandó][,állandó[..állandó]...] **:** *utasítás*;
[állandó[..állandó][,állandó[..állandó]...] **:** *utasítás*; ... *]*
[**ELSE** *utasítás]*
END

ahol *szelektor*: sorszámozott kifejezés; *állandó*: konstans kifejezés, mely sorszámának az Integer vagy Word intervallumába kell esnie. *Szelektor* és *állandó* kompatibilis típusúak.

- Az utasítások közül egy vagy nulla hajtódik végre. *Utasítás* lehet összetett, vagyis bármelyik ágon több utasítás is végrehajtható.
- Csak az első olyan ág hajtódik végre, melynél *szelektor* értéke beleesik a megadott tartományba. ELSE-re akkor kerül a vezérlés, ha egyik ág sem tudott végrehajtódni. Ha nincs ELSE, akkor lehetséges, hogy egy ág sem kerül végrehajtásra.

Hátultesztelő ciklus – iteráció

REPEAT
[utasítás [; *utasítás ...]]*
UNTIL *kilépési feltétel*

A REPEAT és UNTIL fenntartott szavak a ciklus utasításait összefogják. A ciklus utasításait ciklusmagnak is szokás nevezni. A ciklusmag egyszer mindenképpen végrehajtódik, és egészen addig ismételten végrehajtódik, amíg a kilépési feltétel nem teljesül. Olyan esetekben szokás használni, amikor a cik-

lust egyszer mindenképpen végrehajtjuk, és a ciklus újbóli végrehajtása attól függ, hogy mi a végrehajtás eredménye.

Elöltesztelő ciklus – iteráció

WHILE *belépési feltétel* **DO**
utasítás

Utasítás (ciklusmag) lehet összetett utasítás, így a ciklusban több utasítást is végre tudunk hajtani egyszerre. Ha a belépési feltétel már először nem teljesül, akkor a ciklusmag egyszer sem kerül végrehajtásra (üres ciklus). Olyan esetekben szokás használni, amikor a ciklust nem feltétlenül akarjuk végrehajtani, és a ciklus újbóli végrehajtása attól függ, hogy van-e még feldolgozandó adat.

Elöltesztelő léptető ciklus – iteráció

FOR *ciklusváltozó* **:=** *kezdőérték* **TO / DOWNTO** *végérték* **DO**
utasítás

ahol *ciklusváltozó* lokális sorszámozott típusú változóhivatkozás, *kezdőérték* és *végérték* azzal értékadás-kompatibilis kifejezések.

- TO esetén *ciklusváltozó* sorszám szerint eggyel növekszik, DOWNTO esetén eggyel csökken. A ciklusmag a ciklusváltozó minden egyes értékére egyszer végrehajtódik – először a kezdőértékre, utoljára a végértékre.
- A ciklusváltozó értékét felhasználhatjuk, de meg nem változtathatjuk!
- *Utasítás* (ciklusmag) lehet összetett utasítás, így a ciklusban egyszerre több utasítást is végre tudunk hajtani.

- A FOR ciklus elöltesztelő ciklus, így TO esetén ha *végérték* < *kezdőérték* illetve DOWNTO esetén ha *kezdőérték* < *végérték*, akkor a ciklusmag egyszer sem kerül végrehajtásra.

Minősítő utasítás

WITH *változó-azonosító [,változó-azonosító ...]* **DO**
utasítás

ahol *változó-azonosító* rekord vagy objektum azonosítója lehet. A felsorolt változók mezőire való hivatkozásnál *utasítás*ban elegendő csak a mező-azonosítókat illetve metódus-azonosítókat megadni. *Utasítás* lehet összetett utasítás, így a minősítés egy egész programrészre hatással lehet.


Ne használjuk dinamikus változóra, ha a mutatót az utasításon belül megváltoztatjuk!

A program felépítése

Programfej

[**PROGRAM** *azonosító [(periféria [,periféria ...])];]*
[**USES** *azonosító [,azonosító ...];]*

A System egység automatikusan a programhoz szerkesztődik.

Programblokk – deklarációs rész *(a deklarációk száma és sorrendje kötetlen)*

[**LABEL** *címke [,címke ...];]*
[**TYPE** *azonosító = típus; ...]* (*típus* = *típusazonosító* vagy *típusleírás*)
[**CONST** *azonosító [: típus] = konstans kifejezés / tömb konstans / rekord konstans / objektum konstans / halmaz konstans /* **NIL** *;]*
[**VAR** *azonosító: típus; [***ABSOLUTE** *szegmens:ofszet / változó-azonosító;]...]*
[**PROCEDURE** *azonosító [(formális paraméterlista)];*
[**ASSEMBLER; / INLINE / FORWARD; / EXTERNAL; / INTERRUPT;** *]*
eljárásblokk]
[**FUNCTION** *azonosító [(formális paraméterlista)] : típusazonosító;*
[**ASSEMBLER; / INLINE / FORWARD; / EXTERNAL;** *]*
függvényblokk]

Programblokk – végrehajtandó rész

BEGIN
[utasítás [; utasítás ...]]
END.

Az egység felépítése

Egységfej

UNIT ***azonosító*****;**

azonosítót kell majd megadnunk a Uses kulcsszó után abban a programban vagy egységben, ahol ezt az egységet használjuk. Ezt a nevet kell adnunk továbbá a lemezre mentett forrásnyelvű egységnek.

Egységblokk – deklarációs rész – illesztő rész

INTERFACE

[**USES** *azonosító [,azonosító ...];]*

Globális deklarációk – az egység és az őt használó program illetve más egység számára elérhető azonosítók: címkék, konstansok, típusok, változók valamint eljárás- és függvényfejek a kifejtő részből kiemelve.

Egységblokk – deklarációs rész – kifejtő rész

IMPLEMENTATION

[**USES** *azonosító [,azonosító ...];]*

Lokális deklarációk: címkék, változók, típusok, konstansok, eljárások, függvények valamint az illesztő részben deklarált eljárások és függvények kifejtése.

Egységblokk – inicializáló rész (a program elején egyszer végrehajtódik)

[**BEGIN**

utasítás [; *utasítás ...]]*

END.

Eljárás felépítése

Eljárásfej

PROCEDURE *eljárás-azonosító [(formális paraméterlista)]*;

Deklarációs rész

LABEL ...
CONST ...
TYPE ...
VAR ...
PROCEDURE ...
FUNCTION ...

Blokk

BEGIN
utasítások
END;

Formális paraméterlista

*[***VAR***] azonosító [,azonosító...]* **:** *típusazonosító*
[; ***[VAR]*** *azonosító [,azonosító...]* **:** *típusazonosító ...]*

Eljárás hívása – eljárás típusú utasítás

eljárás-azonosító [(aktuális paraméterlista)];

Aktuális paraméterlista

kifejezés/változó [, kifejezés/változó ...]

Függvény felépítése

Függvényfej

FUNCTION *függvényazonosító [(formális paraméterlista)]* **:** *típusazonosító*;

Deklarációs rész

LABEL ...
CONST ...
TYPE ...
VAR ...
PROCEDURE ...
FUNCTION ...

Blokk

BEGIN
utasítások
END;

Formális paraméterlista

*[***VAR***] azonosító [,azonosító...]* **:** *típusazonosító*
*[***;** *[***VAR***] azonosító [,azonosító...]* **:** *típusazonosító ...]*

Függvény hívása

Függvényazonosító [(aktuális paraméterlista)]

Aktuális paraméterlista

kifejezés/változó [, kifejezés/változó ...]

Deklarációk érvényességi köre

A program-, egység-, eljárás-, függvény és metódusblokk deklarációs részében deklarált azonosítók és címkék a blokk lokális azonosítói illetve címkéi. Egy blokk deklarációs részében egy azonosító vagy címke csak egyszer szerepelhet, és az a deklarációtól (a deklarációt nem beleértve) a blokk végéig van érvényben, beleértve a belső – a blokk által tartalmazott – blokkokat is, a következő kivételektől eltekintve:

- nem érvényes abban a belső blokkban, ahol ugyanilyen azonosító illetve címke van deklarálva
- mutató típus alaptípusa lehet olyan típus, melyet még nem deklaráltunk, de a deklarációnak még ugyanabban a típusdeklaráló részben szerepelnie kell
- rekordok különböző szintjein lehetnek ugyanolyan mezőnevek
- objektumok mezői, metódusainak lokális változói egy szinten szerepelnek, azoknak egyedinek kell lenniük
- metódusoknak csak a feje szerepel az objektumtípusban. Ettől a deklarációtól kezdve az objektum ismeri a metódust, de még ugyanabban a deklarációs részben azt ki kell fejteni
- az egység INTERFACE részében lévő deklarációk ismertek az egész egységben, attól függetlenül, hogy kifejtésüket csak később, a kifejtő részben adjuk meg.

Egységeket használó programban illetve egységben érvényben vannak a használt egységek illesztő részeinek deklarációi. Ha ugyanaz az azonosító több egységben is szerepel, akkor a minősítés nélküli azonosítóra való hivatkozás mindig a később

felsorolt egységben lévőt jelenti. Minősített azonosítóval a program és bármely egység bármely azonosítójára hivatkozhatunk.

A Turbo Pascal adattípusok a következő nagyobb csoportokba sorolhatók:

- egyszerű
- karakterlánc
- struktúrált
- mutató
- eljárás
- objektum

Definíciók

Sorszámozott típusok

Minden lehetséges értékhez hozzárendelhető egy sorszám (ORDinal number, rendszám), az értékek sorszámuk szerint rendezettek. Sorszámozott típusok.

- egészek
- logikai
- karakter
- felsorolt
- intervallum

A sorszámok: 0,1,2 ... – kivéve az egész típusokat, ahol a sorszám maga a szám.

Egyszerű típusok

- sorszámozott típusok
- valós típusok

Állomány típusok

- szöveges állomány
- típusos állomány
- típusnélküli állomány

Struktúrált (összetett) típusok

- tömb
- rekord
- objektum
- halmaz
- állomány típusok

A PACKED fentartott szó struktúrált típus előtt használható, de Turbo Pascal-ban hatástalan – az adatok tömörítése automatikus.

Mutató típusok

- típusos mutató
- típusnélküli mutató.

Karakter típus – CHAR

Értékei

Bármelyik kiterjesztett ASCII karakter.

Konstans

Egykarakteres szövegkonstans, például 'B', '5', #48, ^L.

Tárolás

1 bájt, a karakter ASCII kódja (rendszáma)

Műveletek

A karakterek összehasonlíthatók a <, <=, >, >=, =, <> műveletekkel. Az eredmény igaz vagy hamis voltát a karakterek ASCII kódjai határozzák meg. Pl. 'A' < 'B' igaz, 'A' > 'a' hamis, '0' <> #0 igaz.

Egész típusok

Típus	*Értékei*	*Tárolás*
Byte	0..255	Előjel nélküli, 8 bit
Word	0..65,535	Előjel nélküli, 16 bit
ShortInt	–128..127	Előjeles, 8 bit
Integer	–32,768..32,767	Előjeles, 16 bit
LongInt	–2,147,483,648..2,147,483,647	Előjeles, 32 bit

Először mindig a kisebb helyiértékű bájt kerül tárolásra.

Konstans

Decimális és hexadecimális egészek, pl. 13, $45, -3768, -$80000. Előredefiniált konstansok:

MaxInt = 32,767 az Integer típus legnagyobb értéke;
MaxLongInt = 2,147,483,647 a MaxLongInt típus legnagyobb értéke.

Az egész konstans típusát a következő táblázat határozza meg:

-2,147,483,648..-32,769	LongInt
-32,768..-1	Integer
0..255	Byte
256..32,767	Integer
32,768..65,535	Word
65,536..2,147,483,647	LongInt

2,147,483,647-nél nagyobb számot csak valósként adhatunk meg.

Műveletek

NOT, +, – (+ és – előjelek)
***, /, DIV, MOD, AND, SHL, SHR**
+, –, **OR, XOR**
<, <=, >, >=, =, <>, **IN**

Logikai típus – BOOLEAN

Értékei

False, True (speciális felsorolt típus)

Konstans

False, True – két előredefiniált logikai konstans.

Tárolás

1 bájt, False = 0, True = 1. A nullától eltérő érték mindig True-t jelent.

Műveletek

NOT
AND
OR, XOR
<, <=, >, >=, =, <>, IN

Felsorolt típus – (*konstansazonosító[,konstansazonosító...]*)

Értékei

A felsorolásban szereplő azonosítók.
Az azonosítók egyediek, Ord(*konstansazonosító*) = 0, 1, ...

Konstans

A felsorolásban szereplő azonosítók.

Tárolás

A tárolás az azonosító rendszáma alapján történik. 1 bájt, ha az azonosítók száma kisebb, mint 256, egyébként 2 bájt. Az első azonosító értéke 0, a másodiké 1, stb. lesz. Természetesen az esetlegesen eltárolt felsorolt típusú értékek más programban mást jelenthetnek.

Műveletek

<, <=, >, >=, =, <>, IN

Intervallum típus – *konstans1 .. konstans2*

Egy már létező sorszámozott típus (eredeti típus) egy intervalluma. *konstans1* és *konstans2* konstans kifejezések, mégpedig ugyanolyan típusúak, és *konstans1* ≤ *konstans2*. *konstans1* zárójellel nem kezdődhet, mert ekkor a fordító felsorolt típust feltételez. Az eredeti típus az a legkisebb típus, melybe ez az értékintervallum belefér.

Értékei

≥ *konstans1*, ≤ *konstans2*

Konstans

Mint az eredeti típus konstansa.

Tárolás

A tárolás a legkisebb olyan típus szerint történik, amelyikbe belefér az alsó és a felső határ is.

Műveletek

Mint az eredeti típusnál.

Valós típusok

A lebegőpontos alak: [*mantissza*][E[*exponens*]]

Jelentése: *mantissza**$10^{exponens}$, ahol

mantissza egy előjeles valós szám, $0 \leq \text{Abs}(mantissza) < 10$, *exponens* egy előjeles egész szám – ennyi helyiértékkel kell *mantissza* tizedespontját jobbra vagy balra vinni.

Ha az E után *exponenst* elhagyjuk, akkor az 0 lesz, vagyis *mantissza* eggyel szorzódik ($E^0 = 1$). A szám pontosságát *mantissza* tizedespontja utáni azon jegyek száma adja, amelyek még részt vesznek a számolásban.

A következő táblázatban a Comp-ot leszámítva az abszolút értékhatárokat adtuk meg: a nullához legközelebb és a nullától legtávolabb eső értéket. Az értékek természetesen lehetnek negatívak is.

Típus	*Értékei*	*Tárolás*	*Pontosság*
Real	2.9E-39..1.7E38	6 bájt	11-12 jegy
Single	1.5E45..3.4E38	4 bájt	7-8 jegy
Double	5.0E-324..1.7E308	8 bájt	15-16 jegy
Extended	3.4E-4932..1.1E4932	10 bájt	19-20 jegy
Comp	$-2^{63}+1..\ 2^{63}-1$ (-9.2E18..9.2E18)	8 bájt	19-20 jegy egész érték

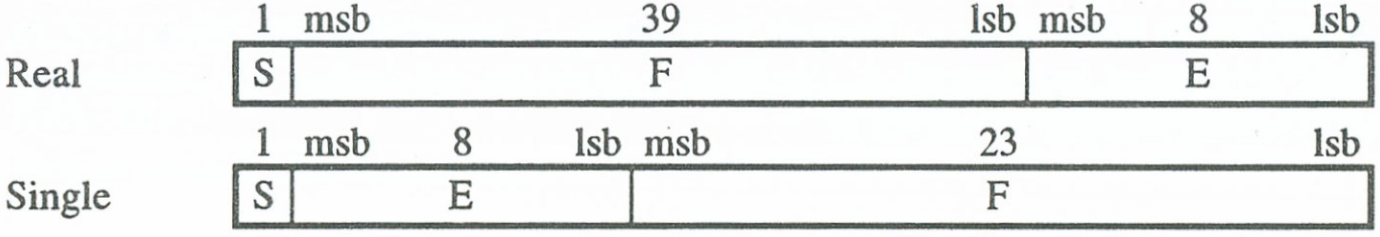
A Real típust leszámítva mindegyik használatához szükséges a matematikai társprocesszor, illetve annak emulálása. Lásd $N és $E direktívák.

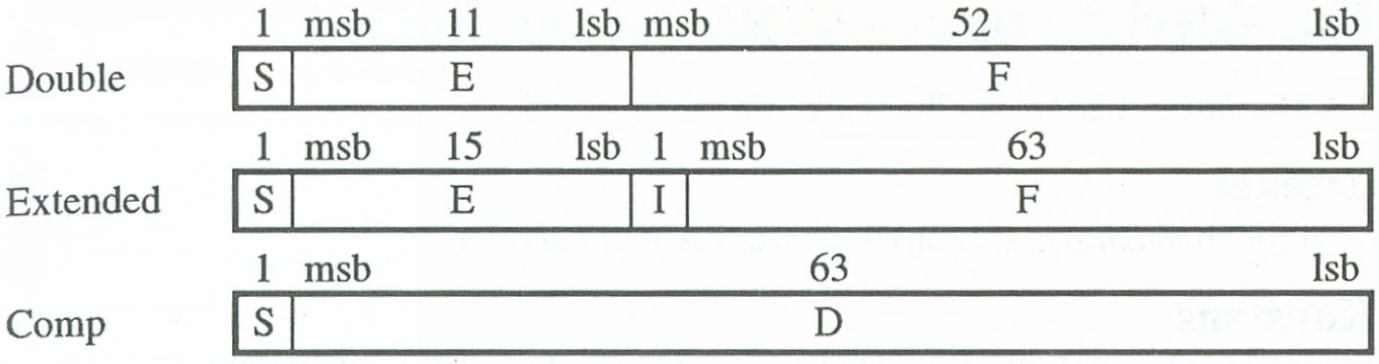
Konstans

Pl. 4.812, -789222.5, 45.8E7, 1E-300

Tárolás

A valós szám belső ábrázolása kettes számrendszerbeli lebegegőpontos formában történik: +/- Szignifikáns * 2^{Exponens}, ahol Szignifikáns 1 bitet tartalmaz a bináris pont előtt. $0 \leq \text{Szignifikáns} < 2$.





Jelölések:

S	Előjelbit (Sign)	D	Kettes komplemens ábrázolás
E	Exponens	msb	Legnagyobb helyiértékű bit
F	Szignifikáns tizedesei (0.F/1.F)	lsb	Legkisebb helyiértékű bit

A tárolás mindig a legkisebb helyiértékű bájttal kezdődik, legutoljára a legnagyobb helyiértékű. Az előjelbitet tartalmazó bájt tehát mindig az utolsó.

Műveletek

+, – (előjelek)
*, /
+, –
<, <=, >, >=, =, <>

Karakterlánc típus – STRING *[[maxhossz]]*

$1 \leq Maxhossz \leq 255$. Ha elhagyjuk, *Maxhossz* = 255.

Értékei

Legfeljebb *maxhossz* számú kiterjesztett ASCII karakter.

Konstans

Szövegkonstans, pl.: 'Hello', ^G'Ide figyelj', 'Ez "szép" dolog'#10#$0A

Hivatkozás a karakterlánc egy elemére

Azonosító[*Index*], ahol $0 \leq Index \leq MaxHossz$

Tárolás

Maxhossz+1 bájton, a 0. bájt az aktuális hossz.

Műveletek

+

<, <=, >, >=, =, <> *(az ASCII kódok alapján)*

Tömb típus – ARRAY[*indextípus[,indextípus...]*] OF *alaptípus*

Indextípus csak sorszámozott lehet – kivéve LONGINT.

Konstans

(*alaptípus típusú konstans [,alaptípus típusú konstans...]*)

Csak a CONST deklarációs részben adhatjuk meg típusos konstans kezdőértékeként.

Hivatkozás a tömb egy elemére

Tömbazonosító[*Index*][[*Index*] ...]
Tömbazonosító[*Index*[,*Index* ...]]

ahol *index indextípus* típusú kifejezés.

Tárolás

A tárolás sorfolytonosan történik, előbb a belső index növekszik. A foglalt memóriaterület méretét az alaptípus mérete és az egyes indextípusok számossága határozzák meg.

Speciális tömbök

Direkt memóriaelérést szolgáló előre deklarált tömbök

MEM[*szegmens*:*eltolás*]	hivatkozás egy Byte típusú elemre
MEMW[*szegmens*:*eltolás*]	hivatkozás egy Word típusú elemre
MEML[*szegmens*:*eltolás*]	hivatkozás egy LongInt típusú elemre

Direkt portelérést szolgáló előre deklarált tömbök

PORT[*portcím*]	hivatkozás egy Byte típusú elemre
PORTW[*portcím*]	hivatkozás egy Word típusú elemre

Rekord típus – RECORD

[mezőlista;]
[**CASE** *[szelektormező:] sorszámozott típus* **OF**
értékleírás **:** *(mezőlista)*
[; értékleírás **:** *(mezőlista) ...]]*
END

ahol
mezőlista:
mező*azonosító[* ,mező*azonosító...]* **:** *típus*
[; mező*azonosító[* ,mező*azonosító...]* **:** *típus...]*
értékleírás:
konstans kifejezés [,konstans kifejezés ...]

Egy rekordnak csak egy változó része lehet, a fix rész mögött. A változó rész elhagyható. Ha a szelektormező hiányzik, vagyis csak egy típust adunk meg, akkor annak csak szintaktikai szerepe van, a felsorolt értékeknek nincs jelentőségük.

Hivatkozás a rekord egy mezőjére

rekordazonosító.mezőazonosító
WITH *rekordazonosító [,rekordazonosító...]* **DO**
utasítás

Konstans

(*mezőazonosító* : *konstans*
[; *mezőazonosító* : *konstans ...]*)

Csak a CONST deklarációs részben adható meg típusos konstans kezdőértékeként.

Tárolás

A rekord teljes memóriaigényét a fix rész, plusz a legnagyobb változó rész határozza meg. A $A direktíva (Word Align Data) a rekord elemeire nincs hatással.

Halmaz típus – SET OF *alaptípus*

Alaptípus csak sorszámozott lehet, melyben a lehetséges elemek sorszámai 0 és 255 közöttiek lehetnek; egy halmaznak maximálisan 256 eleme lehet.

Halmazkonstruktor

[*kifejezés[..kifejezés][,kifejezés[..kifejezés]...]*]

ahol *kifejezés* típusa sorszámozott, sorszáma a 0..255 intervallumban lehet. Amennyiben egy értékintervallum felső határa kisebb, mint az alsó, a megadott elemek nem lesznek a halmazban. A halmaznak többek között halmazkonstruktorral adhatunk értéket.

Üres halmaz: [ ].

Konstans

[*kifejezés[..kifejezés][,kifejezés[..kifejezés]...]*]

ahol *kifejezés* alaptípus típusú konstans kifejezés, tehát változókat nem tartalmazhat. Így lehet megadni a típussal rendelkező konstans kezdőértékét is.

Tárolás

1 elem = 1 bit (0 - nem eleme, 1 - eleme a halmaznak). Ha alaptípus Min..Max, és E egy elem rendszáma, akkor: Méret = (Max Div 8) - (Min Div 8) + 1 bájt. Az elem az (E Div 8) - (Min Div 8) -dik bájt (E Mod 8)-dik bitje.

Műveletek

*

+, –

=, <>, <=, >=, IN

Szöveges állomány – TEXT

Soros szervezés, a sorok változó hosszúságúak. A sorokat a sorvégjel (CR/LF), az állományt állományvégjel (^Z) zárja. Elérés csak szekvenciálisan (sorosan) történhet, az állományt vagy csak olvasni, vagy csak írni lehet.

Tárolás

256 bájton (TextRec típussal – lásd DOS egység Típusok).

Típusos állomány – FILE OF *alaptípus*

Komponensei egyforma méretűek és *alaptípus* típusúak. A komponensek nem lehetnek állomány illetve objektum típusúak, valamint ilyen típusokat tartalmazó struktúrált típusúak. A beolvasás illetve kiírás egysége a komponens. A komponensek sorszáma 0, 1, ... Az elérés szekvenciálisan, vagy a komponens sorszáma szerint direkt történhet. A típusos állomány írható és olvasható is.

Tárolás

128 bájton (FileRec típussal – lásd DOS egység Típusok).

Típusnélküli állomány – FILE

A Bevitel/Kivitel alacsony szintű, puffer nélküli. A komponenshossz az állomány nyitásakor megadható (alapértéke 128). Elérés szekvenciálisan vagy a komponens sorszáma szerint direkt történhet. A komponensek sorszáma (pozíciója) = 0,1,... Az állomány írható és olvasható is.

Tárolás

128 bájton (FileRec típussal – lásd DOS egység Típusok).

Típusos mutató – ^*alaptípus azonosítója*

A mutató mindig egy 4 bájtos memóriacímet tartalmaz, arra a memóriacímre (változóra) mutat. Típusos mutató esetén a mutatott változó típusa: *alaptípus*.

Értékadás
New, @, Ptr, GetMem, :=

Hivatkozás a mutatott változóra
mutató-azonosító^

Konstans
Nil (a memória 0:0 címére mutat)

Tárolás
4 bájton (szegmens:eltolás)

Műveletek
@

=, <>

Típusnélküli mutató – POINTER

4 bájtos memóriacímet tartalmaz, arra mutat. A mutatott változónak nincs típusa.

Értékadás
@, Ptr, GetMem, :=

Hivatkozás a mutatott változóra
mutató-azonosító^

Konstans
Nil (a memória 0:0 címére mutat)

Tárolás
4 bájton (szegmens:eltolás)

Műveletek
@
=, <>

Eljárás típus

PROCEDURE *[(formális paraméterlista)]*
FUNCTION *[(formális paraméterlista)]* **:** *típusazonosító*

- A formális paraméterlista azonosítóinak csak szintaktikai szerepük van.
- Azon eljárások és függvények, melyek azonosítóját ilyen típusú változó kapja, távoli hívással fordítandók ($F+ vagy Far); nem lehetnek szabványosak, beágyazottak, valamint nem deklarálhatók INLINE és INTERRUPT direktívával.

Értékadás
azonosító **:=** *eljárásazonosító/függvényazonosító*

ahol az azonosított eljárás/függvény formális paramétereinek típusai rendre megegyeznek az eljárás típusú *azonosító* deklarálásakor megadott formális paraméterek típusaival. Függvény esetében az eredmény típusoknak is egyezniük kell.

Hivatkozás

azonosító[(aktuális paraméterlista)]

Tárolás

4 bájton (a rutin címe)

A @ operátor (illetve az Addr függvény) a változó értékét, vagyis a rutin címét adja vissza. Ha az eljárás típusú változó memóriacímét akarjuk megkapni, akkor a @ operátort illetve Addr függvényt kétszer kell alkalmaznunk.

Objektum típus – **OBJECT(*objektumtípus*)**

[mezőlista]
[metóduslista]
PRIVATE
[mezőlista]
[metóduslista]
END

ahol

objektum típus:
örökölt objektumtípus

mezőlista:
mezőazonosító[,mezőazonosító...] : mezőtípus
[; mezőazonosító[,mezőazonosító...] : mezőtípus...]

metóduslista:
Az objektum típushoz tartozó metódusok fejei. A lista elemeinek száma és sorrendje kötetlen:

Procedure *azonosító [(formális paraméterlista)]*; *[* **Virtual;** *]*
Function *azonosító [(formális paraméterlista)]* **:** *típusazonosító; [* **Virtual;** *]*
Constructor* *azonosító [(formális paraméterlista)]* **;**
Destructor* *azonosító [(formális paraméterlista)]* **;** *[* **Virtual;** *]*

- Objektum típus csak a program vagy egység legfelső szintjén deklarálható. A metódusok kifejtését még ugyanabban a deklarációs részben meg kell adni, ott a formális paraméterlista elhagyható.
- A PRIVATE kulcsszó utáni deklarációk más egységből illetve programból nem érhetők el.
- A metódusok paraméterei és lokális változói, valamint az objektum adatmezői – beleértve az örökölt adatmezőket is – ugyanazon a programszinten vannak. Az azonosítókat egyedinek kell deklarálni. A metódusok blokkjaiban hivatkozhatunk az objektumtípusban deklarált adatmezőkre.
- Virtuális metódust tartalmazó objektumot konstruktor hívásával inicializálni kell. Ekkor kap értéket a VMT mező, mely a megfelelő VMT táblára fog mutatni.
- Ha egy metódus virtuális, akkor az objektum-hierarchián lefelé attól kezdve az csak virtuálisként deklarálható felül – a formális paraméterek típusainak rendre egyezniük kell.

* A konstruktor és destruktor speciális szerepét lásd a New, Fail és Dispose leírásánál.

Hivatkozás

Objektumváltozóból (előfordulásból):
Objektumazonosító.Mezőazonosító
Objektumazonosító.Metódusazonosító[(aktuális paraméterlista)]

Metódusból:
Mezőazonosító
Objektumtípus.Metódusazonosító[(aktuális paraméterlista)]

- Metódushívás mindig előfordulásból indul.
- Statikus metódus esetén az objektum-hierarchián lefelé haladva az első ilyen nevű metódus hajtódik végre (fordításkor dől el).
- Virtuális metódus esetén a végrehajtás a hívó objektum által mutatott VMT alapján történik (futáskor dől el).
- Használható a WITH utasítás.

Konstans objektum

(*mezőazonosító*:*mezőtípusú konstans*
[;*mezőazonosító*:*mezőtípusú konstans ...*])

Csak a CONST deklarációs részben adható meg típusos konstans kezdőértékeként. A felsorolás bármely ponttól kezdve elhagyható.

Self paraméter

Metódusok rejtett, utolsó paramétere: a hívó objektum címe – 4 bájt.

Tárolás

Minden virtuális metódust, konstruktort vagy destruktort tartalmazó (polimorfikus) objektumtípushoz létezik az adatszegmensben pontosan egy VMT (Virtuális Metódusok Táblázata).

Cím ↑

n. virtuális metódus címe	DWORD
2.. n-1. virtuális metódus címe	n-2 db. DWORD
1. virtuális metódus címe	DWORD
objektum negatív mérete (bájtokban)	WORD
objektum mérete (bájtokban)	WORD

Az objektum előfordulásban először az örökölt objektumtípus adatmezői kerülnek tárolásra, majd sorban az objektumtípusban deklarált adatmezők. Polimorfikus objektum tartalmaz egy VMT-re mutató mezőt is az első örökölt polimorfikus objektumtípus adatmezői után (2 bájt).

Két egymásból származó polimorfikus objektumtípus VMT-jében ugyanazon nevű virtuális metódusok ugyanazon a relatív címen helyezkednek el. Ezen objektumtípusok előfordulásaiban a VMT mezők azonos relatív címen találhatók.

Típusazonosság

Változó paraméter esetén az aktuális és formális paraméter típusának azonosnak kell lennie.

Két típus akkor, és csak akkor azonos, ha:

- azonosítójuk ugyanaz
- az egyik típust a másik azonosítójával deklaráltuk

Típus-kompatibilitás

A típus-kompatibilitás kifejezések kiértékelésénél szükséges.

Két típus kompatibilis, ha az alábbiak egyike igaz:

- a két típus azonos vagy leírása ugyanaz
- mindkettő valós vagy mindkettő egész
- az egyik a másik, vagy mindkettő egy harmadik intervalluma
- mindkettő halmaz, kompatibilis alaptípusokkal
- mindkettő karaktertömb, egyenlő számú komponensekkel
- az egyik karakterlánc, a másik karakter, karakterlánc vagy karaktertömb típusú
- az egyik típusnélküli, a másik bármilyen típusú mutató
- mindkettő eljárás típusú, azonos paraméterszámmal, paramétertípusokkal és függvények esetén azonos eredménytípussal

Értékadás-kompatibilitás

Értékadásnál a jobb oldali kifejezésnek értékadás-kompatibilisnek kell lennie a változó-hivatkozással illetve függvény-azonosítóval; értékparaméter átadásánál az aktuális paraméter kifejezésnek értékadás-kompatibilisnek kell lennie a formális paraméterrel.

T2 értékadási szempontból kompatibilis T1-el (vagyis megengedett a *T1 típusú változó* := *T2 típusú kifejezés*), ha az alábbiak egyike igaz:

- T1 és T2 azonos típusok, de egyik sem állomány, illetve azt tartalmazó struktúrált típus
- T1 és T2 kompatibilis sorszámozott vagy valós típusok, és T2 értékkészlete beleesik T1 értékkészletébe
- T1 valós, T2 egész típus
- T1 karakterlánc, T2 karakter vagy karakterlánc típus
- T1 karakterlánc, T2 karaktertömb
- T1 és T2 kompatibilis karaktertömbök
- T1 és T2 kompatibilis halmaz típusok, és T2 minden eleme beleesik T1 lehetséges elemeinek tartományába
- T1 és T2 kompatibilis mutató típusok
- T1 és T2 kompatibilis eljárás típusok
- T1 eljárás típus, T2 eljárás vagy függvény azonos paraméterszámmal, paramétertípusokkal és függvények esetén azonos eredménytípussal
- T1 objektum típusú, T2 típusa pedig T1 típusának egy leszármazottja.

- P1 := P2 megengedett, ha P1 a T1 objektumra, P2 a T2 objektumra mutat, ahol T2 típusa T1 típusának egy leszármazottja.

Műveletek *(a jobboldali oszlopban lévő betűk jelentését lásd később)*

Jel	*Művelet*	*Operandusok*	*Eredmény*	
NOT	logikai NEM	Boolean	Boolean	L
	bitenkénti NEM	egész	egész	B
+, -	előjel	egész	egész	A
		valós	valós	A
@	cím	változó vagy szubrutin	Pointer	C
*	szorzás	egész	egész	A
		bármelyik valós	valós	A
	metszet	halmaz	halmaz	H
/	valós osztás	egész	valós	A
		bármelyik valós	valós	A
DIV	egész osztás	egész	egész	A
MOD	maradékképzés	egész	egész	A
AND	logikai ÉS	Boolean	Boolean	L
	bitenkénti ÉS	egész	egész	B

x **SHL** *n*	eltolás balra n bittel	egész, n ≥ 0	x típusa	B
x **SHR** *n*	eltolás jobbra n bittel	egész, n ≥ 0	x típusa	B
+	összeadás	egész	egész	A
		bármelyik valós	valós	A
	összefűzés	String[n], Char	String	K
	unió	halmaz	halmaz	H
-	kivonás	egész	egész	A
		bármelyik valós	valós	A
	különbség	halmaz	halmaz	H
OR	logikai VAGY	Boolean	Boolean	L
	bitenkénti VAGY	egész	egész	B
XOR	logikai KIZÁRÓ VAGY	Boolean	Boolean	L
	bitenkénti KIZÁRÓ VAGY	egész	egész	B
<, <=, >, >=, =, <>	összehasonlítás, tartalmazásvizsgálat	kompatibilis típusok	Boolean	R
x **IN** y	elemvizsgálat	y halmaz, x típusa: y alaptípusa	Boolean	R

Magyarázatok a MÜVELETEK táblázathoz:

A – aritmetikai művelet

- ha az operandus egész számkonstans, akkor a következő típusú lesz:

-2,147,483,648..-32,769	LongInt
-32,768..-1	Integer
0..255	Byte
256..32,767	Integer
32,768..65535	Word
65,536..2,147,483,647	LongInt

- ha az operandus valós számkonstans, akkor az $N- esetén Real, $N+ esetén Extended típusú lesz
- előjel megadása a típust nem változtatja meg
- ha a *, MOD, DIV, + vagy - műveletek mindkét operandusa egész, akkor az eredmény típusa a következő táblázat szerinti lesz:

	ShortInt	**Integer**	**LongInt**	**Byte**	**Word**
ShortInt	Integer	Integer	LongInt	Integer	LongInt
Integer	Integer	Integer	LongInt	Integer	LongInt
LongInt	LongInt	LongInt	LongInt	LongInt	LongInt
Byte	Integer	Integer	LongInt	Integer	Word
Word	LongInt	LongInt	LongInt	Word	Word

Intervallum típus esetén az eredmény az eredeti típus szerint alakul.

- ha a *, + vagy - műveletek valamelyik operandusa valós, illetve / művelet esetén az eredmény $N- mellett Real, $N+ mellett Extended típusú lesz

B – bitenkénti művelet

- a NOT művelet a típust nem változtatja meg
- AND, OR és XOR esetén lásd az előző táblázatot

C – @ (cím) művelet

- a művelet eredménye Pointer (szegmens:eltolás) típusú lesz

H – halmaz művelet

- halmaz műveletek (+, -, *) bal és jobb oldalán kompatibilis halmazoknak kell szerepelniük
- ha az eredményhalmaz – illetve halmazkonstans – legkisebb eleme e1, legnagyobb eleme e2, akkor annak típusa SET OF e1..e2 lesz

K – karakterlánc művelet

- a + (összefűző) művelet mindkét oldalán karakter vagy karakterlánc szerepelhet
- az eredmény bármely karakterlánc típussal kompatibilis – de a Char típussal nem
- ha az eredmény 255 karakternél hosszabb lenne, csonkul

L – logikai művelet

- a kiértékelési módokat lásd a $B+/- direktíva leírásánál

R – reláció (összehasonlító művelet)

- egyszerű és karakterlánc típusok esetén:
  - a <, <=, >, >=, =, <> műveletek használhatók;
  - az operandusoknak kompatibiliseknek kell lenniük, de lehet az egyik egész, a másik valós;
  - a karakterláncok összehasonlítása az ASCII kódok alapján történik, a Char típus mint String[1] viselkedik. Az első nem egyenlő karakter eldönti a sorrendet, ha addig egyenlő, akkor a rövidebb karakterlánc a kisebb

- mutató típus esetén:
  - csak az = és a <> műveletek használhatók;
  - két mutató akkor egyenlő, ha szegmens- és eltolás részeik egyenlők;
  - a New és GetMem eljárások mindig normalizálnak, azaz a mutató eltolás része 0 és $F között lesz
  - két mutató, mely ugyanazon címre mutat, nem feltétlenül egyenlő

- halmaz típusok esetén:
  a <=, >=, =, <>, IN műveletek használhatók

Kifejezések kiértékelése

- először a zárójelben szereplő kifejezések kerülnek kiértékelésre
- a kifejezés fordításkor kiértékelhető részeit a fordító értékeli ki, és ezen részeredményeket mint konstansokat helyezi el a programban – az így létrejött részeredmények típusát azok értékei határozzák meg
- minden részeredmény, valamint az egész kifejezés meghatározott típussal rendelkezik, mely az adott művelettől és az operandusok típusától függ
- a műveletek kiértékelési sorrendjét azok prioritása szabja meg:
  - NOT, +, -, @
  - *, /, DIV, MOD, AND, SHL, SHR
  - +, –, OR, XOR
  - <, >, <=, >=, <>, =, IN

Az 1. csoportba tartozó műveletek egy operandussal rendelkeznek – ezek prioritása a legnagyobb –, míg a többi műveletnek két operandusa van. Azonos prioritás esetén a balról jobbra szabály lép érvénybe, melyet a fordító felülbírálhat optimális kód készítése érdekében.

- Az egyébként azonos prioritású műveletek különbözőképpen való csoportosítása a kifejezés típusát befolyásolhatja; az esetleges értékcsonkulásra a programozónak kell figyelnie – célszerű a konstans kifejezéseket előbbre venni.

Értékadás, értékparaméter átadás

A kifejezésnek értékadás-kompatibilisnek kell lennie a változó-hivatkozással illetve függvény-azonosítóval. Az esetleges túlcsordulások és túlindexelések – amennyiben a fordító ki tudja szűrni – szintaktikai hibát eredményeznek. Ezek futás közbeni ellenőrzését a $R+ direktívával tehetjük meg. Az alapértelmezés $R-, tehát a fogadómezőbe kerülő érték csonkulhat, illetve akaratlanul is felülírhatjuk a memória különböző területeit.

A direktívák leírásánál használt jelölések jelentése:

G – globális direktíva: hatása az egész programra kiterjed, a deklarációs rész előtt kell állnia

L – lokális direktíva: a program bármely pontján elhelyezhető, be- és kikapcsolható

Kapcsoló– és paraméter direktívák

A kapcsoló direktíváknál az alapértelmezés szerinti állapotot írtuk előre. Utána zárójelben a keretrendszerbeli megfelelőjét adtuk meg.

$A+/- (O/C/Word align data) G

Bekapcsolt állapotban ($A+) minden egy bájtnál nagyobb méretű változó és típusos konstans szóhatárra (kettővel osztható memóriacímre), kikapcsolt állapotban ($A-) bájthatárra kerül. Rekordok, objekumok és tömbök elemei mindig bájt-folytonosan helyezkednek el a memóriában. 80x88-as processzor esetén a direktíva hatástalan, 80x86-nál a direktíva aktív állapota gyorsítja a futást.

$B-/+ (O/C/Complete boolean eval) L

Bekapcsolt állapotban ($B+) a feltételeket teljes egészében kiértékeli a program, függetlenül attól, hogy az egész feltétel szempontjából szükséges-e vagy sem. Kikapcsolt állapotban ($B-) csak addig értékeli az egyes részfeltételeket, míg egyértelműen el nem dönthető, hogy a teljes feltétel igaz vagy hamis lesz. Inaktív álla-

potában a futás gyorsabb, de a nem kiértékelt feltételrész esetleges hibái nem derülnek ki, függvényei nem kerülnek végrehajtásra.

$D+/- (O/C/Debug information) G

Bekapcsolt állapotban ($D+) információs tábla készül, így lehetővé válik az illető egység vagy program keretrendszerből vagy külső nyomkövetővel történő hibakeresése/nyomkövetése, valamint a futási hiba helyének meghatározása.
Az O/D/Debugging, az O/L/Map file kapcsolói, valamint az O/C/Local symbols kapcsolója ($L+/- direktíva) csak akkor hatásosak, ha ez aktív ($D+). Ha ki van kapcsolva, több memóriánk marad a program futtatásához.

$E+/- (O/C/Emulation) G

Bekapcsolt állapotban ($E+) a fordító a programhoz szerkeszti a 80x87-es aritmetikai segédprocesszort emuláló rutincsomagot. Futáskor, ha a 80x87 jelen van, a számításokat az végzi, egyébként a rutincsomag. Egységekben a kapcsolónak nincs hatása. A kapcsoló hatástalan $N- mellett.

$F-/+ (O/C/Force far calls) L

Bekapcsolt állapotban ($F+) FAR hívási módot használ a fordító: a címek duplaszavasak, így az aktuális kódszegmensen kívülre is lehet hivatkozni. Kikapcsolt állapotban ($F-) a szükséges – FAR vagy NEAR – hívási módot használja. Overlay egységeket használó programok esetében ajánlott minden egység és a program elejére $F+ direktívát elhelyezni. Eljárás típusú változóknak értékül adott rutinok $F+ mellett fordítandók. Lásd FAR direktíva.

$G+/- (O/C/286 instructions) L

Bekapcsolt állapotban ($G+) a fordító 80286-os kódot generál (a következő 80286-os utasítások is bekerülhetnek a futó program kódjába: ENTER, LEAVE, PUSH *érték*, kiterjesztett IMUL - SHL - SHR). Kikapcsolt állapotban csak a 8086-os processzor utasításkészletét használja, és így a program bármely 80x86-os gépen futni tud. A G+ mellett fordított programok gyorsabbak és jobbak, de a 8086 és 8088-as processzorokon nem tudnak majd futni. A fordító nem generál ellenőrző rutint, így magunknak kell gondoskodni arról, ha 80286/80287 utasításokat használunk.

$I+/- (O/C/I/O checking) L

Bekapcsolt állapotban ($I+) hibás B/K műveletek esetén a program futása hibaüzenet kíséretében leáll. Kikapcsolt állapotban ($I-) a program B/K hiba miatt nem áll le. Ilyenkor a műveletek hibakódja az IOResult függvénnyel minden esetben lekérdezendő, különben a hibát követő további B/K műveleteket a program nem hajtja végre. A függvény a System egység InOutRes változójának értékét adja vissza, és egyben nullázza is azt.

$I *állománynév* L

A fordító a megadott forrásállomány tartalmát befoglalja a direktíva helyére. Ha *állománynév* nem rendelkezik kiterjesztéssel, a fordító automatikusan .PAS-t feltételez; ha nem rendelkezik lemezegységnévvel és/vagy útvonalleírással, akkor a fordító előbb az aktuális, majd a menürendszer O/D/Include directories pontjában meghatározott katalógusokban keresi. Ezen állományok maximálisan 15 szint mélységig ágyazhatók egymásba. Begin és End között nem szerepelhet.

$L+/- (O/C/Local symbols) **G**

Bekapcsolt állapotban ($L+) engedélyezi a lokális szimbólumokra vonatkozó információk generálását, és hibakeresés/nyomkövetés során azok kiértékelését. A kapcsoló hatástalan $D- mellett.

$L ***állománynév*** **L**

Hozzászerkeszti az egységhez vagy programhoz a megadott tárgykódú (object) állomány tartalmát. Ha *állománynév* nem rendelkezik kiterjesztéssel, a fordító automatikusan .OBJ-t feltételez; ha nincs lemezegységnév és/vagy útvonalleírás, akkor a fordító előbb az aktuális, majd a menürendszer O/D/Object directories pontjában meghatározott katalógusokban keresi. A tárgykódú állomány minden PUBLIC szubrutinjának fejét EXTERNAL direktívával a Pascal forrásállományban deklarálni kell. A direktíva a deklarációs részben bárhol állhat.

$M ***veremméret, heapmin, heapmax*** (O/Memory sizes) **G**

A direktíva a program memóriafoglalását határozza meg. A verem méretét (1024.. 65520) *veremméret*, a heapméret alsó határának minimumát (0.. 655360) *heapmin*, felső határának maximumát (heapmin.. 655360) *heapmax* állítja be. Egységekben hatástalan.

$N-/+ (O/C/8087/80287) **G**

Bekapcsolt állapotban ($N+) a valós számításokat a 80x87-es aritmetikai segédprocesszorral végezteti a program, míg kikapcsolt állapotban ($N-) könyvtári rutinokat használ, és a valós típusok közül csak a Real alkalmazható. Ha a direktíva

aktív állapota mellett nincs a gépben 80x87-es segédprocesszor, akkor azt emulálnunk kell ($E+).

$O-/+ (O/C/Overlays allowed) **G**

Bekapcsolt állapotban ($O+) az egység későbbi overlay-struktúrába szervezését engedélyezi.

$O ***egységnév*** **G**

Overlay-struktúrába szervezi a program által használt, itt megadott egységet, melyet előzőleg $O+ mellett kell fordítani. A direktívának a programfej után kell állnia. Az overlay-struktúrába szervezett egysége(ke)t a fordító a programot tartalmazó forrásállomány nevével megegyező, .OVR kiterjesztésű fizikai állományban helyezi el. Egységekben nincs hatása.

$R-/+ (O/C/Range checking) **L**

Bekapcsolt állapotban ($R+) engedélyezi értékhatár-ellenőrző kód generálását. Az ellenőrzés tömbök és karakterláncok indexelésére, sorszámozott típusú változók értékadásaira, valamint virtuális metódus hívásakor az objektum előkészítettségének vizsgálatára terjed ki . Ha $R+ mellett hiba lép fel, a program futása hibaüzenet kíséretében leáll. Mivel a program sebességét csökkenti és a kód méretét növeli, csak a program fejlesztésekor, tesztelésekor célszerű bekapcsolni.

$S+/- (O/C/Stack checking) **L**

Bekapcsolt állapotban ($S+) minden szubrutinhíváshoz kódot generál a verem túlcsordulásának ellenőrzésére. Ha egy szubrutin hívásakor nincs elegendő hely a veremben a visszatérési cím, a paraméterek és a lokális változók számára, $S+ esetén

a program futása hibaüzenet kíséretében leáll – egyébként a rendszer összeomolhat.

$V+/- (O/C/Strict var-strings) **L**

Bekapcsolt állapotban ($V+) a karakterlánc típusú formális és aktuális változó (Var) paraméter típusának egyeznie kell. Kikapcsolt állapotban ($V-) bármilyen karakterlánc típusú aktuális paraméter átadható – természetesen ekkor a programozónak kell gondoskodnia az esetleges szemantikai hibák kivédéséről.

$X+/- (O/C/Extended syntax) **G**

Bekapcsolt állapotban a függvények eljárásként is hívhatók. Ez esetben a függvény visszatérési értéke figyelmen kívül marad (Pl.: ReadKey ;). A direktíva a System egység függvényeire nem alkalmazható.

Feltételes direktívák

$DEFINE *xxx* (O/C/Conditional defines) **L**

Feltételes fordításhoz definiálja *xxx* feltételes szimbólumot. Ha *xxx* már definiált, a direktívának nincs hatása. A szimbólumok hossza nem korlátozott, de a fordító csak az első 63 karakterüket veszi figyelembe. Kis- és nagybetűket, számjegyeket és _ karaktert tartalmazhatnak; számjeggyel nem kezdődhetnek.

$UNDEF *xxx* L

Az előzőleg $DEFINE direktívával vagy a keretrendszer O/C/Conditional defines pontjával definiált *xxx* feltételes szimbólumot megszünteti. Ha *xxx* nem definiált – vagy már megszüntetett – nincs hatása a direktívának.

$IFDEF *xxx* L

Ha *xxx* feltételes szimbólum definiált, a fordító a következő forrássorokat ($ELSE-ig illetve $ENDIF-ig) lefordítja.

$IFNDEF *xxx* L

Ha *xxx* feltételes szimbólum nem definiált – vagy a feltételes szimbólum már megszüntetett –, a fordító a következő forrássorokat ($ELSE-ig illetve $ENDIF-ig) lefordítja.

$IFOPT *d+* | *d-* L

Ha *d* kapcsoló típusú direktíva a megadott állapotban van (+ vagy -), a fordító a következő forrássorokat ($ELSE-ig illetve $ENDIF-ig) lefordítja.

$ELSE

$IFDEF | $IFNDEF | $IFOPT és $ENDIF között alkalmazható. Ha a feltételes fordítást indító direktíva nem teljesül, a $ELSE és $ENDIF közötti forrássorokat fordítja le a fordító.

$ENDIF

Határolja a feltételes fordításba eső forrássorokat.

Előre definiált feltételes szimbólumok

VER60

Jelzi, hogy Turbo Pascal fordító verziószáma 6.0. Minden verzió a neki megfelelő szimbólumot definiálja. Az 5.5-ös verzióban a VER55 szimbólum van érvényben.

MSDOS

Jelzi, hogy a Turbo Pascal fordító MS-DOS illetve PC-DOS operációs rendszer alatt működik.

CPU86

Jelzi, hogy a CPU a 80x86 processzor családhoz tartozik.

CPU87

Akkor definiált, ha a gépben van fordításkor 80x87-es matematikai társprocesszor.

Speciális direktívák

ASSEMBLER ;

Assembly nyelven írt eljárások és függvények kaphatják ezt a direktívát. Ekkor a rutin végrehajtandó része csak **Asm** és **End** közé írható (a Begin–End elhagyan-

dó). A fordító az alábbi optimalizálásokat végzi a direktívával deklarált rutinok esetén:

- az értékparaméterek nem kerülnek át lokális változókba, a karakterlánc és 1, 2 vagy 4 bájttól eltérő méretű értékparamétereket változó paraméterekként kell kezelni;
- csak karakterlánc típusú függvények esetén használható a @Result (függvény értékének címe) mutató;
- paraméter és lokális változók hiányában nem generál veremfoglaló kódot.

EXTERNAL ;

A szubrutin blokkja egy tárgykódot tartalmazó (.OBJ) állományban található, melyet $L direktívával a programhoz vagy egységhez kell szerkeszteni bárhol a deklarációs részben. A direktívával deklarált szubrutinhoz a tárgykódú állományban ugyanolyan nevű, PUBLIC típusú rutinnak kell tartoznia. A tárgykódú (object) állományban nem állhat olyan PUBLIC címke, melyhez a programban nincs EXTERNAL deklaráció. (Lásd $L *állománynév* direktíva.)

FAR ;

Rutinok hívási módjának jelzésére szolgál, ha a rutin fejének végén szerepel (lásd $F+/- direktíva). Használható Asm utasításban is (... FAR PTR ...).

FORWARD ;

A direktívával ellátott szubrutin blokkját nem kell feltétlenül a fej után leírni, megadható a deklarációs rész bármely későbbi pontján is. Ott a blokk előtt csak az

esetleges paraméterlista nélküli fejet kell megismételni. Ezzel lehetővé válik, hogy szubrutinok kölcsönösen hívják egymást. Az egység Interface részében nem lehet megadni!

INLINE (*elem*[/ *elem* ...]);

ahol *elem* egy kódolt processzor utasítás, Pascal állandó vagy változó azonosítója lehet.
Gépi kódban megírt szubrutin beágyazása a programba. Az utasításokban a program változóinak és konstansainak azonosítói is szerepelhetnek. A < operátor az utána álló kifejezés alsó bájtját adja, míg > két bájtot jelent (ha a kifejezés egy bájt, a felső 00h lesz). Változók előtt + vagy - jel is állhat, jelezve a változó kezdőcímétől való eltolást. A direktíva helyettesíti a szubrutin blokkját. (Az InLine utasítás szerkezete ehhez hasonló.)

INTERRUPT ;

Megszakítási eljárásokhoz használandó direktíva. Megszakítási eljárást – speciális be- és kilépési utasítássorozata miatt – a programból közvetlenül meghívni tilos. Bármely megszakítási vektor (kellő körültekintéssel) az eljárás címére átállítható. A megszakítási eljárásokat távoli hívással kell deklarálni ($F+ vagy FAR direktíva)!
Az eljárás blokkja a szokásos; formális paraméterlistája a következő: (a lista tagjai tetszőleges ponttól kezdve elhagyhatók) :

(Flags, CS, IP, AX, BX, CX, DX, SI, DI, DS, ES, BP : Word);

NEAR ;

Rutinok hívási módjának jelzésére szolgál, ha a rutin fejének végén szerepel (lásd $F+/- direktíva). Használható Asm utasításban is (... NEAR PTR ...).

Konstansok

MaxInt = 32767;

MaxLongInt = 2147483647;

Változók

Szabványos B/K
Input : Text;
Output : Text;

Állománykezelés
FileMode : Byte = 2;

B/K hibakezelés
InOutRes : Integer = 0;

Heap-kezelés
HeapOrg : Pointer = Nil;
HeapEnd : Pointer = Nil;
HeapPtr : Pointer = Nil;
FreeList : Pointer = Nil;
HeapError : Pointer = Nil;

Overlay-kezelés
OvrHeapSize : Word = 0;
OvrDebugPtr : Pointer = Nil;

Memóriacímek
PrefixSeg : Word = 0;
StackLimit : Word = 0;

Véletlenszám generálás
RandSeed : LongInt = 0;

Matematikai társprocesszor
Test8087 : Byte = 0;

Programbefejezés
ExitCode : Integer = 0;
ErrorAddr : Pointer = Nil;
ExitProc : Pointer = Nil;

OvrLoadList : Word = 0;
OvrCodeList : Word = 0;

```
OvrHeapOrg   : Word = 0;        OvrDosHandle : Word = 0;
OvrHeapPtr   : Word = 0;        OvrEmsHandle : Word = 0;
OvrHeapEnd   : Word = 0;
```

Elmentett megszakítási vektorok

```
SaveInt00  : Pointer;   SaveInt35  : Pointer;   SaveInt3C  : Pointer;
SaveInt02  : Pointer;   SaveInt36  : Pointer;   SaveInt3D  : Pointer;
SaveInt1B  : Pointer;   SaveInt37  : Pointer;   SaveInt3E  : Pointer;
SaveInt21  : Pointer;   SaveInt38  : Pointer;   SaveInt3F  : Pointer;
SaveInt23  : Pointer;   SaveInt39  : Pointer;   SaveInt75  : Pointer;
SaveInt24  : Pointer;   SaveInt3A  : Pointer;
SaveInt34  : Pointer;   SaveInt3B  : Pointer;
```

Eljárások, függvények

Abs (X) : X típusa

Visszaadja *X* egész vagy valós kifejezés abszolútértékét.

Addr (X) : Pointer

Visszaadja az *X* változó, típusos állandó, eljárás, vagy függvény memóriacímét. Kivételt képez az eljárás típusú változó. Ha *X* eljárás típusú változó, akkor Addr(*X*) az *X* változóban tárolt eljáráscímet adja vissza. Az eljárás típusú változó memóriacímét úgy kaphatjuk meg, hogy a függvényt kétszer alkalmazzuk. Az Addr függvény által visszaadott érték értékadás szempontjából kompatibilis az összes mutató típussal. Az Addr függvény helyettesíthető a @ operátorral.

Append (Var F:Text)

Megnyitja írásra az *F* szöveges állományt, és az állomány aktuális pozícióját az állomány végére állítja – a következő írások bővítik az állományt. Ha az állomány nyitva volt, akkor az eljárás lezárja, majd újra nyitja azt. Amennyiben az utolsó 128 bájtos blokkban volt állományvégjel (^Z, vagyis #26 karakter), akkor az aktuális pozíció az első ilyenre áll. *F* állományváltozót előzőleg az Assign eljárással egy létező fizikai állományhoz kell rendelni. Ha *F*-hez az üres karakterláncot rendeljük, akkor az írás a szabványos B/K eszközre (CON) történik. $I- esetén az IOResult függvénnyel a nyitás eredményessége lekérdezhető.

ArcTan (X:Real) : Real

X valós kifejezés arkusztangensét adja vissza radiánban.

Assign (Var F; Name:String)

F állomány típusú változóhoz (logikai állományhoz) hozzárendeli *Name* fizikai állományt. A logikai állományra vonatkozó minden további művelet *Name* fizikai állományra vonatkozik. A kapcsolat a program végéig tart, hacsak egy újabb Assign eljárással nem rendelünk a logikai változóhoz egy másik fizikai állományt. *Name* tartalmazhat teljes útvonalleírást is, hossza maximálisan 79 karakter lehet. *Name* azonosíthat DOS perifériát is: CON, LPT1 (=PRN), LPT2, LPT3, COM1 (=AUX), COM2, NUL. Ha *Name* üres karakterlánc, az eljárás *F*-hez a CON perifériát rendeli.

Nyitott állományra nem szabad használni.

BlockRead (Var F:File; Var Buf; Count:Word [; Var Result:Word])

F típusnélküli állományból beolvas *Count* rekordot, és elhelyezi *Buf* változó memóriacímétől kezdődően. Csak nyitott állományra alkalmazható, a rekord méretét nyitáskor adtuk meg. *Result* tartalmazza visszatéréskor, hogy hány rekordot olvasott be az eljárás. Beolvasás után az állomány pozíciója *Result* rekorddal előrébb áll. *Result* csak a nem csonka rekordok számát adja vissza, vagyis ha a rekordméret > 1, akkor az utolsó beolvasáskor a (*Result*+1)-edik rekordban is lehet az állományhoz tartozó adat. Ha *Result* hiányzik, és nem tudta a megadott számú rekordot beolvasni, B/K hiba lép fel. A rekordméret 128 bájt, ha az *F* állomány megnyitásakor azt külön nem adtuk meg. Count * rekordméret maximálisan 65535 bájt lehet. A művelet eredményessége $I- esetén az IOResult függvénnyel lekérdezhető.


Ügyeljünk arra, hogy *Buf* változó nagysága elegendő legyen a kért adatok tárolására, mert a *Buf* után deklarált változókat könnyűszerrel felülírhatjuk!

BlockWrite (Var F:File; Var Buf; Count:Word [; Var Result:Word])

F típusnélküli állományba kiír *Count* rekordot *Buf* változó kezdőcímétől. Csak nyitott állományra alkalmazható, a rekord méretét nyitáskor adtuk meg. *Result* tartalmazza visszatéréskor, hogy hány rekordot írt ki az eljárás. Kiírás után az állomány pozíciója *Result* rekorddal előrébb áll. Ha *Result* hiányzik, és nem tudta a megadott számú rekordot kiírni (pl. betelt a lemez), B/K hiba lép fel. A rekordméret 128 bájt, ha az *F* állomány megnyitásakor azt külön nem adtuk meg. *Count* * re-

kordméret nem lehet 65535 bájtnál nagyobb. Az eljárás csak teljes rekordokat ír ki. A művelet eredményessége $I- esetén az IOResult függvénnyel lekérdezhető.

ChDir (S:String)

Az aktuális katalógust az *S* paraméterben megadottra változtatja. *S* teljes útvonalleírás is lehet, tartalmazhat lemezegységnevet is. Az eljárás annyiban tér el az operációs rendszer CD parancsától, hogy lemezegység megadása esetén át is vált az illető egységre. A DOS az itt megadott katalógust – és lemezegységet – tekinti majd aktuálisnak a program lefutása után. A művelet eredményessége $I- esetén az IOResult függvénnyel lekérdezhető.

Chr (X:Byte) : Char

Az egy bájton tárolható karakterek mindegyikének megfelel egy kód. Az ASCII kódtábla szerint az 'A' betű kódja például 65. Ez a függvény a megadott kódot karakterré alakítja, vagyis az *X* Byte típusú kifejezésnek, mint ASCII kódnak (karakter sorszámának) megfelelő kiterjesztett ASCII karaktert adja vissza.

Close (Var F)

Lezárja az előzőleg Reset, Rewrite vagy Append eljárással megnyitott *F* logikai állományt. Lezárás előtt a még pufferben maradt adatokat kiírja az állományba, a katalógust aktualizálja. Végül felszabadítja az állományhoz tartozó kezelőszámot (DOS handle) további felhasználásra. A művelet eredményessége $I- mellett az IOResult függvénnyel lekérdezhető.

Concat (S1 [,S2 ...] :String) : String

A paraméterként átadott karakterláncokat összefűzi. Ha a függvény által visszaadott karakterlánc hossza 255-nél nagyobb lenne, akkor a 255. karakter utániakat levágja. A függvény helyettesíthető a + művelettel.

Copy (S:String; Index,Count:Integer) : String

S karakterlánc *Index* pozíciójától számított *Count* karakterét mint részláncot adja vissza. Ha *Index* nagyobb, mint az *S* karakterlánc aktuális hossza, a függvény értéke üres karakterlánc lesz. Ha *Count* nagyobb, mint ahány karakter az *Index*-edik pozíció után van, akkor csak a karakterlánc hátralévő karaktereit adja vissza. Ha *Index* < 1, akkor az 1-nek, ha *Count* < 0, akkor az 0-nak tekintendő.

Cos (X:Real) : Real

X valós kifejezés (radián) koszinuszát adja vissza. A függvény értékkészlete -1 és 1 közé esik.

CSeg : Word

A függvény visszaadja a CS regiszter tartalmát, vagyis annak a kódszegmensnek (code segment) a szegmenscímét, amelyből a függvény meghívásra került. Ha a függvényt a főprogramból hívjuk, akkor a főprogram kódszegmensét, míg egységből történő hívás esetén annak az egységnek a kódszegmensét kapjuk meg.

Dec (Var X [; N:LongInt])

X sorszámozott típusú változó értékét csökkenti eggyel, illetve *N*-nel. Az eljárás megfelel az *X* := Pred(*X*) egyszeri illetve *N*-szeri végrehajtásának. Az eljáráshoz a fordító optimalizált (gyors) kódot generál.

Delete (Var S:String; Index,Count:Integer)

S karakterlánc típusú változóból törli az *Index*-edik karaktertől számított *Count* karaktert. Ha *Index* nagyobb, mint az *S* karakterlánc aktuális hossza, nem töröl. Ha *Count* nagyobb, mint ahány karakter az *Index*-edik pozíció után van, akkor csak a hátralévő karaktereket vágja le.

Dispose (Var P:Pointer [; destruktor])

Felszabadítja a *P* típusos mutatóval mutatott heap-beli területet. A felszabadított terület nagysága az aktuális paraméter típusától függ. A felszabadítás után minden további hivatkozás *P*^-ra hibát eredményezhet. Futási hibát okoz, ha hívásakor *P* nem a heap-be mutat. Ha a felszabadítás a heap tetejéről történik, akkor a HeapPtr lejjebb mozog, egyébként a heap-ben egy szabad terület, „lyuk" keletkezik, melyet szabad blokknak nevezünk. A szabad blokkokat a későbbiekben a New illetve GetMem eljárások újra felhasználhatják. A heap szabad területeit a rendszer az un. szabad területeket nyilvántartó listában (free list) jegyzi, és tartja karban. Az első szabad területre a FreeList változó mutat. A mutatott terület első 4 bájtja mutat a következő szabad blokkra stb., az utolsó szabad blokk a heap tetejére mutat.
Amennyiben *P* egy objektum mutatója, második paraméterként az objektum destruktora is átadható. Ilyen esetben a felszabadítás előtt végrehajtásra kerül a destruktor. A felszabadítást maga a destruktor végzi, annak utolsó tevékenységeként.

Ha az objektum rendelkezik VMT-vel (Virtuális Metódusok Tálázata), és az objektumot inicializáltuk (ekkor az objektum VMT mezője a megfelelő VMT-re mutat), a destruktor a VMT-ben található méret alapján végzi a felszabadítást. Ha az objektumot előzőleg nem inicializáltuk, nem történik felszabadítás. Ha tehát a Dispose első paramétere objektum mutatója, akkor a felszabadított terület nagysága nem feltétlenül a mutatott objektum típusának megfelelő. Polimorfikus objektumok esetén a felszabadított terület attól függ, hogy azt a bizonyos objektumot milyen VMT-hez rendeltük inicializáláskor. Címszerinti paraméterátadásnál ugyanis egy objektum átadható bármelyik őstípusnak. A Dispose ilyen esetben is az aktuális, megfelelő méretű területet szabadítja fel.

Nem ajánlatos a Dispose és FreeMem eljárásokat a Mark és Release eljárásokkal együtt használni!

DSeg : Word

A függvény visszaadja a DS regiszter tartalmát, vagyis az adatszegmens szegmenscímét. Programunkhoz egy adatszegmens (data segment) tartozik, mely tartalmazza az összes főprogramban illetve egységben deklarált globális változót, valamint a program összes globális és lokális típusos konstansát. A DS regiszter a program futása során nem változik.

Eof [(Var F:Text)] : Boolean *(szöveges állományokra)*

A függvény értéke True, ha az aktuális pozíció az állomány logikai végén (^Z karakteren) vagy fizikai végén áll, egyébként False. Ha *F* hiányzik, a függvény az Input állományt vizsgálja. Amennyiben Eof(*F*) True, Eoln(*F*) is True lesz. $I- esetén az IOResult függvénnyel a művelet eredményessége lekérdezhető.

Eof (Var F) : Boolean *(típusos és típusnélküli állományokra)*

A függvény értéke True, ha az aktuális pozíció az *F* állomány utolsó komponense mögé került (például az utolsó komponens beolvasása után) vagy üres az állomány, egyébként False. $I- esetén az IOResult függvénnyel a művelet eredményessége lekérdezhető.

Eoln [(Var F:Text)] : Boolean

A függvény értéke True, ha az aktuális pozíció a szöveges *F* állomány sorvégjelén (CR/LF karakterpár) vagy állományvégjelén (^Z karakter) áll, egyébként False. Ha *F* hiányzik, a függvény az Input állományt vizsgálja. A művelet eredményessége $I- esetén az IOResult függvénnyel lekérdezhető.

Erase (Var F)

Törli az *F* állományváltozóhoz rendelt fizikai állományt. Megfelel a DOS DEL illetve ERASE parancsának. Nyitott állományra nem szabad használni. $I- esetén a törlés eredményessége lekérdezhető az IOResult függvénnyel.

Exit

Az eljárás hatására a vezérlés egy programszinttel feljebb kerül: kilép az aktuális szubrutinból illetve a főprogramból. Másképpen: Goto utasítás az Exit-et tartalmazó blokk záró End; illetve End. sorára.

Exp (X:Real) : Real

X valós kifejezés exponenciálisát adja vissza (e^X).

Fail

Csak konstruktorból hívható eljárás. Hatása részben megegyezik az Exit eljárásával, tehát a programvezérlés a konstruktor záró End-jére kerül.
Ha a konstruktort a kiterjesztett New második paramétereként adjuk meg, akkor a heapfoglalást a dinamikus objektum számára maga a konstruktor bevezető kódja végzi. Ha ilyenkor nincs elegendő heap, akkor alapértelmezésben a program futási hibával „elszáll". Lehetőségünk van arra, hogy a heap-hibát saját magunk kezeljük le a HeapError mutató átállításával. Ha a heap-hiba lekezelő függvény visszatérési értéke 1, akkor a program nem áll le, hanem a New Nil értéket ad vissza, mely a sikertelen heap-foglalási kísérletet mutatja. Ekkor a konstruktor blokkja már nem kerül végrehajtásra. Ha a konstruktor bevezető kódjának sikerült lefoglalnia a heap-et a dinamikus objektum számára, akkor elvégzi az objektum VMT mezőjének inicializálását is.
Előfordulhat azonban, hogy az objektum további helyfoglalásai sikertelenek, de bármely más okból is meghátrálhatunk (bocsánat, mégsem kell ez az objektum!). Ekkor hívjuk meg a Fail eljárást. A Fail eljárás felszabadítja a lefoglalt objektumot, – amelynek még volt ugyan helye a heap-ben, de így már nincs rá szükségünk –, kilép az eljárásból és a New eljárás vagy függvény Nil értékkel tér vissza.
Dinamikus objektumok esetén a New visszaadott Nil értéke jelzi a sikertelenséget. Azonban ha egy statikus objektum konstruktorát, vagy egy örökölt konstruktort hívunk meg, nincs olyan mutató, mely ezt jelezné nekünk. A Turbo Pascal megengedi a konstruktor függvényként való hívását. A konstruktor False értéket ad vissza, ha a vezérlés Fail eljárásra került, egyébként True.

FilePos (Var F) : LongInt

F típusos vagy típusnélküli állomány azon komponensének sorszámát adja vissza, melyre az állománymutató mutat. Ha az állománymutató a legelső (nulladik) komponensre mutat vagy üres az állomány, a függvény értéke 0 lesz. Szöveges állományokra nem alkalmazható, az állománynak nyitva kell lennie. Amennyiben Eof(*F*)= True, akkor FilePos(*F*)= FileSize (*F*). Kereséskor figyelembe kell venni, hogy a Read eljárás az állománypozíciót eggyel előrébb lépteti, vagyis ha a Read-el beolvasott elem a keresett, akkor annak pozíciója a függvény által visszaadott érték-1. $I- esetén az IOResult függvénnyel a művelet eredményessége lekérdezhető.

FileSize (Var F) : LongInt

F állomány méretét – azaz komponenseinek számát – adja vissza. Ha az állomány üres, akkor a függvény értéke 0 lesz. Az állomány mérete bájtokban: FileSize(*F*) * Komponens mérete. A komponens mérete típusos állomány esetén SizeOf(Komponenstípus), típusnélküli állomány esetén a Reset által megadott méret. Szöveges állományokra nem alkalmazható, és az állománynak nyitva kell lennie. $I- esetén az IOResult függvénnyel a művelet eredményessége lekérdezhető.

FillChar (Var X; Count:Word; Value)

Feltölti *X* változó kezdőcímétől indulva a memória *Count* bájt hosszúságú területét *Value* sorszámozott típusú kifejezés értékével. Ord(Value) a 0..255 intervallumba eshet.


Memória-ellenőrzés nincs, ezért óvatosan használandó!

Flush (Var F:Text)

A Rewrite vagy Append eljárással írásra megnyitott *F* szöveges állományba kiírja az állomány átmeneti pufferének tartalmát. Olvasásra megnyitott állományok esetében nincs hatása. $I- esetén az IOResult függvénnyel a nyitás eredményessége lekérdezhető. Amikor egy állományba írunk illetve onnan olvasunk, a rendszer az adatokat egy átmeneti tárolóban – pufferben – helyezi el. Erre azért van szükség, mert a memória és háttértár közötti adatmozgatás mindig sokkal lassúbb, mint ha az adatokat csak memóriából memóriába írjuk. Íráskor a rendszer megvárja, míg megtelik a puffer, s az azt követő írás alkalmával az egész puffer tartalmát kiírja lemezre. Ha olvasni is tudunk az állományból, akkor olvasás előtt is kiírja a puffert. Ha csak olvasunk, akkor egyszerre több adatot olvas be a rendszer a pufferbe, s csak akkor olvas újra, ha a kért adat nincs a pufferben. A puffer-mechanizmus működését szöveges állományok esetén programból részben befolyásolhatjuk. A SetTextBuf eljárással megadhatjuk a puffer méretét, valamint írásra nyitott szöveges állományoknál a Flush eljárással fizikai írásra kényszeríthetjük a rendszert.

Frac (X:Real) : Real

X valós kifejezés tizedes részét adja vissza: Frac(*X*) = *X* - Int(*X*).

FreeMem (Var P:Pointer; Size:Word)

Felszabadítja a heap-ből a *P* által mutatott *Size* méretű területet. A felszabadítás után minden további hivatkozás *P*^-ra hibát eredményezhet. Futási hibát okoz, ha hívásakor *P* nem a heap-be mutat.

Nem ajánlatos a Dispose és FreeMem eljárásokat a Mark és Release eljárásokkal együtt használni! Amennyiben *Size* nem egyezik az előzőleg lefoglalt terület méretével, a heap megsérülhet.

GetDir (D:Byte; Var S:String)

A *D* paraméter által meghatározott lemezegységre vonatkozó aktuális katalógus teljes útvonalát (d:útvonal) adja vissza S-ben. *D*=0 az aktuális, 1 az A:, 2 a B:, stb. lemezegységet jelenti. B/K ellenőrzés nem történik, ha a *D* által megadott lemezegység érvénytelen, vagy nincs bent lemez, akkor az érvénytelen lemezegység főkatalógusát adja vissza, mintha minden rendben lenne (Pl. 10 esetén 'J:\').

GetMem (Var P:Pointer; Size:Word)

Lefoglal a heap-ben egy *Size* méretű területet, és kezdőcímét elhelyezi *P*-ben. A *P* aktuális típusától függő memóriaterületre vagy dinamikus változóra *P*^-al lehet hivatkozni. Az átadott mutató általában típus nélküli (Pointer), és a programozónak kell gondoskodnia arról, hogy a lefoglalt méretű heap-területet használja. Futási hiba lép fel, ha nincs a heap-ben Size méretű, egybefüggő szabad terület – kivéve, ha a HeapError által mutatott heap-hiba lekezelő függvény visszatérési értéke nem 0. Maximálisan 64 K (egy szegmensnyi) heap foglalható le egyszerre az eljárással. *P*^ mindig 8-al osztható bájthatáron kezdődik.

Halt [(ExitCode:Word)]

Leállítja a program futását és a vezérlést visszaadja az operációs rendszernek illetve a hívó programnak. A Halt eljárás még a program végleges befejezése előtt inicializálja az összes egység kilépési eljárását. *ExitCode* opcionális paraméterrel ki-

szállási kódot lehet megadni. Ez a paraméter lesz az egység *ExitCode* változójának az értéke, melyet kilépési eljárásban még használhatunk – ilyenkor ErrorAddr Nil értékű lesz. A hívó programban ez a kilépési kód a DOS egység DosExitCode függvényével, Batch állományban pedig az ERRORLEVEL-lel lekérdezhető. A paraméter nélküli Halt megfelel a Halt(0)-nak.

Hi (X) : Byte

X Integer vagy Word típusú kifejezés felső (high) bájtját adja vissza. A két bájton tárolható egészek belső ábrázolása alsó+felső bájt – a memória kisebb címén szerepel az alsó bájt. *X* = 256 * Hi(X) + Lo(X).

Inc (Var X [; N:LongInt])

X sorszámozott típusú változó értékét növeli eggyel, illetve *N*-nel. Az eljárás megfelel az *X* := Succ(*X*) egyszeri illetve *N*-szeri végrehajtásának. Az eljáráshoz a fordító optimalizált (gyors) kódot generál.

Insert (Source:String; Var S:String; Index:Integer)

Source karakterláncot beszúrja *S* karakterlánc változóba az *Index*-edik karakterpozíciótól úgy, hogy az ottlévő karaktereket jobbra tolja. Ha a beszúrás utáni karakterlánc 255 karakternél hosszabb lenne, a 255. utáni karaktereket levágja. Ha Index nagyobb, mint *S* aktuális hossza, akkor *S* végéhez fűzi hozzá *Source* karaktereit. Ha *Index* < 1, akkor a beszúrás az első karaktertől történik.

Int (X:Real) : Real

X valós kifejezés egész részét adja vissza. A függvény 0 felé kerekít, vagyis Int(*X*) = *X* - Frac(*X*). E függvény csak a visszaadott érték típusában tér el a Trunc függvénytől.

IOResult : Word

Az utolsó Beviteli/Kiviteli művelet hibakódját adja vissza, amennyiben a periféria-műveletek ellenőrzése inaktív ($I-). A függvény a System egység InOutRes változójának értékét veszi fel – ami egy DOS B/K hibakód –, majd nullázza azt. Ha az InOutRes változó értéke nem 0, a B/K műveletek nem kerülnek végrehajtásra.

Length (S:String) : Integer

S karakterlánc aktuális hosszát adja vissza. Ha az aktuális paraméter egy változó, akkor a visszaadott érték nem más, mint *S*[0] rendszáma, hiszen az aktuális hosszt mindig a 0. bájt tartalmazza: Length(*S*) = Ord(*S*[0]).

Ln (X:Real) : Real

X valós kifejezés természetes alapú logaritmusát adja vissza.

Lo (X) : Byte

X Integer vagy Word típusú kifejezés alsó (low) bájtját adja vissza. A két bájton tárolható egészek belső ábrázolása alsó + felső bájt – a memória kisebb címén szerepel az alsó bájt. Pl. 543 = 2*256+31 esetén az alsó címen lesz a 31, melyet a 2 követ. *X* = 256 * Hi(*X*) + Lo(*X*).

Mark (Var P:Pointer)

Az eljárás megjegyzi a heap-mutató értékét, vagyis a HeapPtr aktuális értékét átmásolja *P*-be. A heap-mutató később a Release eljárással visszaállítható.

MaxAvail : LongInt

Visszaadja a heap legnagyobb szabad, egybefüggő területének méretét bájtokban. Ennél nagyobb méretű dinamikus változónak már nincs hely a dinamikus tárban. Egybefüggő, szabad területnek számít a szabad listára felfűzött összes blokk, valamint a HeapPtr feletti szabad memória. A szabad lista első blokkjára a FreeList változó mutat. A minimális és maximális heap-igényt a program kezdetekor a $M fordítási direktívával adhatjuk meg.

MemAvail : LongInt

Visszaadja a heap összes szabad területének méretét bájtokban. Ez MaxAvail értékénél nem kisebb, hiszen ez a szabad listára felfűzött területek, plusz a HeapPtr feletti szabad memória összege. Ha a program során a Dispose illetve FreeMem eljárásokkal nem szabadítottunk fel memóriát, akkor a szabad lista üres, a heap tömör, nincsenek lyukak benne. Ekkor MaxAvail = MemAvail. A szabad lista első blokkjára a FreeList változó mutat. A minimális és maximális heap-igényt a program kezdetekor a $M fordítási direktívával adhatjuk meg.

MkDir (S:String)

Létrehozza az *S* paraméterben megadott katalógust. *S* teljes útvonalleírás is lehet, tartalmazhat lemezegységnevet is. Az eljárás megfelel a DOS MD parancsának. A művelet eredményessége $I- esetében az IOResult függvénnyel lekérdezhető.

Move (Var Source,Dest; Count:Word)

Source változó kezdőcímétől *Count* bájtot átmásol *Dest* változó első bájtjától kezdődően. Az eljárás *Source* és *Dest* esetleges átfedését automatikusan lekezeli. Nem végez típus- illetve memória-ellenőrzést, ezért óvatosan használandó!

New (Var P:Pointer [; konstruktor])

Létrehoz a heap-ben egy dinamikus változót – mely típusát *P* határozza meg –, és címét elhelyezi *P*-ben. Az aktuális paraméter csak típusos mutató lehet. A dinamikus változóra *P*^-al lehet hivatkozni. Futási hiba lép fel, ha a heap-ben nincs elegendő egybefüggő terület. A futási hiba úgy kerülhető el, ha a HeapError változót egy olyan saját heap-hiba lekezelő függvényre állítjuk, amelynek visszatérési értéke nem 0. Ha nem sikerül a területfoglalás, akkor a heap-hiba lekezelő visszatérési értékétől függően futási hiba áll elő, a New által visszaadott érték Nil lesz illetve a helyfoglalás megismétlődik. A foglalt terület nagysága mindig 8-al osztható. Egy különálló Byte típusú változó pl. 8 bájtot foglal el a heap-ből! A New által visszaadott mutató normalizált, azaz az ofszet része csak 0 vagy 8 lehet.
A New függvényként is hívható – ekkor *P* egy típusos mutató típusazonosítója. Ez esetben létrejön a heap-ben a megadott típusú dinamikus változó, s annak mutatója lesz a függvény értéke. Amennyiben *P* egy objektum mutatója, második paraméterként az objektum konstruktora is átadható. Ilyenkor a New meghívja a konstruktort. A heap foglalását az objektum számára tulajdonképpen a konstruktor bevezető kódja végzi. A heap foglalása után végrehajtódik a konstruktor blokkja. Ha Fail utasításra kerül a vezérlés, akkor a New által most lefoglalt terület felszabadul, és a vezérlés a konstruktor záró End-jére kerül.

Odd (X:LongInt) : Boolean

A függvény értéke True, ha *X* egész típusú kifejezés értéke páratlan, egyébként False.

Ofs (X) : Word

A függvény *X* kezdőcímének eltolási (ofszet) értékét adja vissza. *X* egy változó vagy szubrutin azonosítója lehet.

Ord (X) : LongInt

X sorszámozott típusú kifejezés sorszámát adja vissza. A sorszámozott típusra az jellemző, hogy minden lehetséges értékéhez hozzárendelhető egy egész szám, a sorszám. A karakterhez annak ASCII kódja, egészhez maga a szám, felsoroltho z a felsorolás sorszáma 0-tól számozva. Speciális felsorolt típus a logikai, vagyis Ord(False) = 0 és Ord(True) = 1.

ParamCount : Word

Visszaadja, hogy a program hány parancssor-paraméterrel indult. Futtatáskor, a lemezre fordított .EXE kiterjesztésű programnak DOS alól indítva paraméterek adhatók át. A paraméterek a program neve után következnek, és azokat szóközök és/vagy tabulátorok választják el egymástól (Pl. Prog Param1 Param2). Az egyes paraméterek a ParamStr függvénnyel kérdezhetők le. Parancssor-paraméterek a környezetben a **R**/Parameters menüpontban szimulálhatók.

ParamStr (Index:Word) : String

Visszaadja az *Index*-edik parancssor-paramétert, melyet a program indításakor adtak meg. A program paramétereit a szóköz vagy TAB karakterek választják el egymástól. (Pl. Proba.Exe program indítása 3 paraméterrel: Proba P1 P2 P3). Ha *Index* nulla vagy nagyobb, mint a ParamCount függvény értéke, akkor üres karakterláncot ad vissza. 3.0 és későbbi DOS verziók alatt ParamStr(0) értéke a futó program állományspecifikációja, illetve ha a keretrendszerben nem lemezre fordítunk, akkor a Turbo.Exe állományspecifikációja.

Pi : Real

Pi értékét adja vissza a függvény. Értéke 3.1415926536, matematikai társprocesszor (8087, 80287 vagy 80387) használata esetén {$N+} 3.14159265358979323 85.

Pos (SubStr,S:String) : Byte

Visszaadja, hogy *SubStr* részlánc az *S* karakterlánc hányadik karakterpozíciójától kezdődően található meg először. Ha *S* nem tartalmazza *SubStr*-t, akkor a függvény értéke 0 lesz.

Pred (X) : X típusa

X sorszámozott kifejezés sorszám szerinti előző értékét adja vissza. Pl. Pred('B') = 'A', Pred(1) = 0, Pred(True) = False stb. A programozónak kell figyelnie arra, hogy az eggyel kisebb sorszámhoz tartozzék érték a megfelelő típusból.

Ptr (Seg,Ofs:Word) : Pointer

Mutató típusúvá konvertálja a *Seg* és *Ofs* paraméterekben megadott szegmens:eltolás címet. A függvény a mutatót nem normálja. A függvény típusnélküli mutatót ad vissza, de típuskonverzióval bármilyen típusú változóra mutathatunk vele.

Random [(Range:Word)] : Real vagy Word

Ha *Range* hiányzik: Real típusú és $0 \leq X < 1$, egyébként Word típusú és $0 \leq X < Range$ véletlenszámot generál. Ha *Range* értéke 0, akkor a függvény is 0 értékű lesz. A függvény az egységben definiált RandSeed változó egy következő értékét számítja, és ez alapján ad vissza egy véletlen számot. Használat előtt ajánlatos a véletlenszám generátort inicializálni a Randomize eljárással, vagy a RandSeed változó értékadásával. Ez utóbbi esetben ismételten ugyanazt a véletlenszám sorozatot kapjuk.

Randomize

Az operációs rendszer szerinti idő alapján véletlen értékkel inicializálja a véletlenszám generátort: beállítja RandSeed értékét. A Random függvény a RandSeed értéke alapján számítja a véletlen értéket, majd újra állítja azt.

Read ([Var F:Text;] V1 [,V2 ...]) (szöveges állományok)

Az eljárás beolvassa *F* szöveges állományból az aktuális pozíción kezdődő értéke(ke)t a paraméter(ek) típusától függően, majd elhelyezi *V1* [*V2*...] változó(k)ban. A változók egész, valós, karakteres vagy karakterlánc típusúak lehetnek. Az állománymutató az utolsó beolvasott karakter mögé áll.

- Numerikus változók esetében a bevezető TAB, szóköz és sorvégjel (CR/LF) karaktereket átugorja, és beolvas egy karaktersorozatot a legközelebbi Tab, szóköz, sorvégjel illetve állományvégjelig (^Z). Ha a beolvasott karaktersorozat nem numerikus formátumú, futási hiba lép fel.
- Karakter változó esetén az aktuális karaktert olvassa. Ha az állomány mutatója sorvégjelen áll (Eoln(F) = True), akkor a karakter értéke #13 lesz, ha állományvégjelen (Eof(F) = True), akkor #26.
- Karakterláncok esetén sorvégjelig (illetve állományvégjelig) olvas. Ha a változó rövidebb, mint a beolvasott karaktersorozat, akkor a szöveg csonkul.

Ha *F* hiányzik, akkor az eljárás az Input állományból olvas, alapértelmezés szerint a CON perifériáról. Ekkor a bevitel billentyűzetről történik, a leütött karakterek a képernyőn is megjelennek. Ha nincs a billentyűzet pufferben CR/LF-el lezárt karaktersorozat, akkor a puffer tartalma szerkeszthető a legközelebbi Enter leütéséig. Az Enter hatására a pufferbe kerül a CR/LF karakterpár. Ezután a Read eljárás a puffer elejéből vesz ki karaktereket a soronkövetkező paraméter típusának megfelelően.

A Read eljárás csak nyitott állományra alkalmazható. Szöveges állomány puffere a SetTextBuf eljárással megváltoztatható. $I- esetén az IOResult függvénnyel lekérdezhető, hogy sikerült-e a beolvasás.

Képernyőről történő olvasás esetén ajánlatos a ReadLn eljárást használni, mert ha nem töröljük az esetlegesen pufferben maradt karaktereket, akkor a legközelebbi Read illetve ReadLn eljárás nem várakozik, hanem a maradék karaktereket olvassa be. Ezáltal a bevitel időben követhetetlenné válhat. Ajánlatos továbbá, hogy képernyőről egy Read eljárással csak egy változóba olvassunk be, mert ellenkező eset-

ben a program ki van téve a programot használó egyén szeszélyeinek, s gyakorlatilag ellenőrizhetetlen, hogy melyik változóba milyen adat kerül. Legbiztosabb módszer, ha minden bevitelnél egy ReadLn eljárással egy karakterláncot olvasunk be (ez nem okozhat B/K hibát), s a beolvasott karakterláncot alakítjuk át igényeinknek megfelelően.

Read (F; V1 [,V2 ...]) (típusos állományok)

F típusos állományból beolvas egy(-egy) komponenst *V1* [*V2* ...] változó(k)ba. Az állománymutató minden olvasás után automatikusan a következő komponensre lép. A változó(k) típusának és *F* alaptípusának egyezniük kell. Ha az állománymutató az utolsó komponens után áll, az olvasási kísérlet B/K hibát eredményez. Csak nyitott állományra alkalmazható. $I- esetén az IOResult függvénnyel lekérdezhető, hogy sikerült-e a beolvasás.

ReadLn ([Var F:Text;] V1 [, V2 ...])

Csak szöveges állományokra használható. Megfelel a Read eljárásnak, de a változó(k)ba történő beolvasás után a következő sor elejére állítja az állomány mutatóját. Ha nem adunk meg változót, akkor olvasás nem történik, csak a mutató áll a következő sor elejére. Ha *F* hiányzik, az Input állományból történik az olvasás, majd a puffer tartalma törlődik.

Release (Var P:Pointer)

A megadott memóriacím feletti dinamikus tárat felszabadítja: *P* mutató értékét átmásolja a heap-mutatóba (HeapPtr-be). A memóriacímet előzőleg a Mark eljárás-

sal szokás megjegyezni. Az eljárás törli a szabad listát, azaz a FreeList változó értékét Nil-re állítja.

Nem ajánlatos a Mark és Release eljárásokat a Dispose és FreeMem eljárásokkal együtt használni!

Rename (Var F; NewName:String)

Átnevezi *NewName* névre az előzőleg *F*-hez rendelt fizikai állományt. Minden további hivatkozás *F*-re az új fizikai állományra vonatkozik. $I- esetén az IOResult függvénnyel lekérdezhető, hogy sikerült-e az átnevezés. Megfelel a DOS REN parancsának az alábbi eltérésekkel:

- ha *F*-hez katalógusnevet rendeltünk, azt is átnevezi (a katalógus struktúrabeli helyét nem lehet megváltoztatni, csak a nevét);
- ha F-hez egy állomány nevét rendeltük, és az új névben egy másik katalógus vagy útvonal is szerepel, akkor az állományt az átnevezéssel egyidőben a megadott katalógusba mozgatja (lemezek között nem lehet átnevezni).

Nyitott állományra nem szabad használni!

Reset (Var F [:File; RecSize:Word])

Megnyitja az Assign eljárással előzőleg *F*-hez rendelt, létező fizikai állományt, és az állománymutatót az állomány elejére állítja (FilePos(*F*)=0). Ha az állomány üres (0 hosszúságú illetve szöveges állomány esetén az első karakter a ^Z), akkor Eof(*F*) = True. Ha *F* szöveges típusú, akkor csak olvasni lehet az állományból, míg típusos és típusnélküli állományok írhatók és olvashatók is lesznek az eljárás hívása után. Típusnélküli állományok esetében megadható a rekordméret is *RecSize* pa-

raméterben, ha nem adjuk meg, 128 bájt lesz. Ha *F* állományváltozóhoz üres karakterláncot rendeltünk, akkor az Input állományt nyitja meg – ez az inicializáló részben automatikusan megtörténik a program indulásakor. Ha az eljárást nyitott állományra alkalmazzuk, akkor azt automatikusan lezárja újranyitás előtt. $I- esetén az IOResult függvénnyel a nyitás eredményessége lekérdezhető.

Rewrite (Var F [:File; Recsize:Word])

Ha az Assign eljárással előzőleg *F* állományváltozóhoz rendelt fizikai állomány még nem létezik, létrehozza azt, és szöveges állomány esetében írásra, típusos és típusnélküli állomány esetében írásra és olvasásra megnyitja. Ha a fizikai állomány már létezik, akkor előbb törli annak tartalmát. Az állománymutató az állomány elejére áll (FilePos(F)=0), és az állomány üres lesz (FileSize(F)=0). Ha *F* állományváltozóhoz üres karakterláncot rendeltünk, akkor az Output állományt nyitja meg írásra – ez az inicializáló részben automatikusan megtörténik a program indulásakor.
Típusnélküli állományok esetén a *RecSize* paraméterrel a rekordméret adható meg, egyébként az 128 bájt lesz. $I- esetén az IOResult függvénnyel a nyitás eredményessége lekérdezhető.

RmDir (S:String)

Megszünteti az *S* paraméterben megadott alkatalógust. *S* tartalmazhat teljes útvonalleírást is. Megfelel a DOS RD parancsának. Ha a kitörlendő katalógus nem létezik, éppen aktuális, vagy nem üres, B/K hiba lép fel. $I- esetén az IOResult függvénnyel a nyitás eredményessége lekérdezhető.

Round (X:Real) : LongInt

X valós kifejezést a legközelebbi egészre kerekíti. Futási hiba lép fel, ha *X* kerekített értéke nem esik bele a LongInt típus értékkészletébe. A .5-re végződő számokat a nagyobb abszolút értékű egészre kerekíti.

RunError [(ErrorCode:Word)]

Programtesztelésnél használatos eljárás, mely leállítja a program futását és futási hibát generál, mintha az a hiba ténylegesen előfordult volna (Lásd Futási hibakódok, B/K hibák). Az egység ErrorAddr változója a RunError utasításra fog mutatni. *ErrorCode* opcionális paraméterrel a futási hiba kódja adható meg. Ha hiányzik, a kód 0 lesz (Runtime error). A RunError eljárás még a program végleges befejezése előtt inicializálja az összes egység kilépési eljárását. *ErrorCode* lesz az egység ExitCode változójának az értéke, melyet kilépési eljárásban még használhatunk. A hívó programban ez a kilépési kód a DOS egység DosExitCode függvényével, batch állományban pedig az ERRORLEVEL-lel lekérdezhető.

Seek (Var F; N:LongInt)

F típusos vagy típusnélküli állomány mutatóját az *N*-edik komponensre állítja. Az első komponens sorszáma 0. Ha *N* > FileSize(*F*), akkor az állomány a legközelebbi íráskor a megadott sorszámú komponensig bővül, olvasáskor B/K hiba lép fel. Az állománynak nyitva kell lennie. Típusnélküli állományok esetén a komponens mérete 128 bájt, vagy a nyitáskor megadott érték. $I- esetén az IOResult függvénnyel a művelet sikeressége lekérdezhető.

SeekEof [(Var F:Text)] : Boolean

A függvény az *F* szöveges állomány következő TAB, szóköz és sorvégjel karaktereit átugorja, és értéke True lesz, ha a mutató az állományvégjelre került, egyébként False. Ha *F* hiányzik, az Input állományról van szó. Az állománynak nyitva kell lennie. $I- esetén az IOResult függvénnyel a művelet sikeressége lekérdezhető.

SeekEoln [(Var F:Text)] : Boolean

A függvény az *F* szöveges állomány következő TAB és szóköz karaktereit átugorja, és értéke True lesz, ha a mutató sorvégjelre került, egyébként False. Ha *F* hiányzik, az Input állományról van szó. Az állománynak nyitva kell lennie. $I- esetén az IOResult függvénnyel a művelet sikeressége lekérdezhető.

Seg (X) : Word

Visszaadja *X* kezdőcímének szegmenscímét. *X* egy változó vagy szubrutin azonosítója.

SetTextBuf (Var F:Text; Var Buf [; Size:Word])

Hozzárendeli *F* szöveges állományváltozóhoz az eredeti 128 bájtos B/K puffer helyett *Buf* átmeneti puffert. *Size* opcionális paraméterrel lehet megadni a puffer méretét; ha hiányzik, akkor *Buf* mérete határozza meg a puffer hosszát. Az eljárás hatása a következő *F*-re vonatkozó Assign-ig, illetve a program végéig tart.
CON perifériáról történő bevitel során az eljárással korlátozhatjuk a bevihető karakterek számát. Ekkor *Buf* méretét illetve *Size* értékét vegyük két bájttal nagyobbra, mint a bevihető karakterek száma, mert a sorvégjel (CR/LF) is bekerül a puffer-

be! Tapasztalataink szerint csak a Crt egység használata mellett van hatása a *Size* paraméternek.


Nyitott állományra csak közvetlenül a Reset, Rewrite vagy Append eljárás után ajánlatos használni, hogy ne történjék adatvesztés.

Sin (X:Real) : Real

X valós kifejezés (radián) szinuszát adja vissza. A függvény értékkészlete -1 és 1 közé esik.

SizeOf (X) : Word

X típus vagy változó fizikai méretét adja vissza bájtokban.

Ha a SizeOf paramétere objektumtípus, akkor SizeOf a típusban deklarált adatok össz méretét adja. Virtuális metódussal rendelkező objektumtípusok esetén + 2 bájt (a VMT táblára mutató VMT mező mérete). Ha a paraméter egy VMT-vel rendelkező (polimorfikus) objektum előfordulás, akkor a visszaadott érték a VMT-ben tárolt méret, ami azonban nem mindig az objektum típusa által deklarált, hanem az aktuális méret. (A VMT mező inicializáláskor kap értéket, mely akkor is megmarad, ha azt cím szerint paraméterként átadjuk.)
Ha a virtuális metódust tartalmazó objektumot konstruktorral nem inicializáljuk, akkor a méret definiálatlan marad.

SPtr : Word

A függvény visszaadja az SP (stack pointer) regiszter tartalmát, vagyis a veremmutató eltolási (ofszet) címét. A veremmutató szegmenscímét az SSeg függvénnyel kérdezhetjük le. Az SP regiszter induláskor a maximális értéket veszi fel, mert a

stack egy fejére állított verem: lefelé növekszik, s felfelé csökken. Egy eljárásba való belépéskor SPtr értéke csökken, majd az eljárásból való kilépéskor visszaáll az eredeti érték. Ahogy a verem fogy, az SPtr közelít a nullához. Ha nincs elég verem, túlcsordulási hiba áll elő (stack overflow error).

Sqr (X) : X típusa

X egész vagy valós típusú kifejezés négyzetét adja vissza.

Sqrt (X:Real) : Real

X valós kifejezés négyzetgyökét adja vissza.

SSeg : Word

A veremmutató szegmenscímét (SS regiszter tartalmát) adja vissza. A veremszegmens (stack segment) a memóriának egy speciális része, melyet a rendszer arra használ, hogy ott ideiglenesen adatokat tároljon. Adatot csak a verem tetejére lehet tenni, és csak a tetejéről lehet levenni. Ide kerülnek többek között az eljárások paraméterei, lokális változói, valamint visszatérési címei. A verem maximális mérete 64 kB, s a memória SSeg*16. bájtján kezdődik. Méretét a $M fordítási direktívával szabályozhatjuk. Az SS regiszter a program futása során nem változik. A verem relatív mutatóját az SP regiszter tartalmazza, melyet az SPtr függvénnyel kérdezhetünk le.

Str (X [: Width [: Decimals]]; Var S:String)

X egész vagy valós típusú kifejezés értékét az őt reprezentáló karakterlánccá alakítja és azt visszaadja az *S* paraméterben. *Width* a mezőszélességet, *Decimals* a tizedesek számát határozza meg. A karaktereket a megadott karakterláncban jobbra

ütköztetve helyezi el, bevezető szóközök után. Ha *Decimals* hiányzik, akkor az egész vagy lebegőpontos forma kerül *S*-be. Ha *Decimals* értéke kisebb, mint a tizedesek száma, akkor kerekítés történik. Ha *Width* hiányzik, vagy értéke akkora, hogy annyi karakteren nem férne el a numerikus formátum, akkor *X*-et annyi karakteren ábrázolja, amennyi szükséges. Ha a karakterlánc maximális hossza nem elég nagy, az eredmény csonkul.

Succ (X) : X típusa

X sorszámozott típusú kifejezés sorszám szerinti következő értékét adja vissza. Pl. Succ('A') = 'B', Succ(0) = 1, Succ(False) = True stb. A programozónak kell figyelnie arra, hogy az eggyel nagyobb sorszámhoz tartozzék érték a megfelelő típusból.

Swap (X) : X típusa

X Integer vagy Word típusú kifejezés értékét az alsó és felső bájt felcserélésével adja vissza. (*X* := Lo(*X*) * 256 + Hi(*X*))

Trunc (X:Real) : LongInt

X valós kifejezés nulla felé kerekített egész részét adja vissza. Futási hibát okoz, ha az érték nem esik bele a LongInt típus értékkészletébe. E függvény csak a visszaadott érték típusában tér el az Int függvénytől.

Truncate (Var F)

Levágja az *F* típusos vagy típusnélküli állomány komponenseit az aktuális pozíciótól (Eof(*F*) True lesz). Az állománynak nyitva kell lennie. Ha az állomány vége után pozícionálunk, akkor az állomány bővül. $I- esetén az IOResult függvénnyel a csonkítás sikeressége lekérdezhető.

TypeOf (objektum) : Pointer

Visszaadja az *objektum* típushoz vagy változóhoz tartozó virtuális metódusok táblázatának (VMT) címét. Minden objektum típushoz, mely rendelkezik virtuális metódussal, konstruktorral vagy destruktorral, tartozik pontosan egy VMT az adatszegmensben. Egy ilyen polimorfikus objektum előfordulás inicializálásakor az objektum rejtett VMT mezője értéket kap, vagyis az a megfelelő VMT-re fog mutatni. A TypeOf függvény ezt a mutatót adja vissza. Csak VMT-vel rendelkező, azaz virtuális metódust, konstruktort vagy destruktort tartalmazó objektumokra lehet alkalmazni. TypeOf paramétere lehet akár objektum típus, akár objektum típusú változó. Objektum típus esetén TypeOf az ahhoz a típushoz tartozó VMT táblázat címét adja vissza. Objektum típusú változó esetén a változó rejtett VMT mezője határozza meg a függvény értékét, mely az objektum változó inicializálásakor – a konstruktor hívásakor – kap értéket. Ezzel a függvénnyel lehet eldönteni egy objektum előfordulás aktuális típusát.

UpCase (Ch:Char) : Char

A *Ch* karakternek megfelelő nagybetűt adja vissza. Ha *Ch* értéke nem esik az 'a'..'z' intervallumba, akkor a függvény értéke megegyezik *Ch*-val.

Val (S:String; Var V; Var Code:Integer)

S karakterlánc típusú kifejezést *V* típusának megfelelően numerikussá alakítva elhelyezi *V* egész vagy valós típusú változóban. Ha sikerült az átalakítás, akkor *Code* értéke 0, egyébként az átalakítás során talált első hibás karakter pozíciója lesz. Az eljárás a bevezető szóközöket és TAB karaktereket átugorja. Ha az átalakított érték nem esik bele a *V* típusa szerinti értékkészletbe, akkor $R+ esetén ez futási hibát

okoz, míg $R- mellett a valós vagy LongInt típusú *V* értéke definiálatlan és *Code* nullától eltérő lesz; egyéb típusoknál *V* túlcsordul és *Code* értéke 0 lesz.

Write ([Var F:Text;] V1 [,V2 ...]) *(szöveges állomány)*

Az eljárás kiírja *F* szöveges állományba az aktuális pozíciótól *V1* [*V2* ...] kifejezés(ek) értékét. Ha *F* hiányzik, akkor az Output állományba ír, alapértelmezésben a CON perifériára, azaz a képernyőre. Az állománymutató (képernyő esetén a kurzor) az utolsó kiírt karakter mögé áll. Az eljárással egész, valós, karakteres, karakterlánc és logikai típusú értékek írhatók ki. Mezőszélesség és számok esetén tizedesek száma megadható az Str eljáráshoz hasonlóan: Write(*V* [:Szel [:Tized]]). A kiírandó változó összesen Szel karakteren kerül kiírásra, jobbra ütköztetve. Ha a változó hosszabb, mint Szel, akkor a kiírás annyi karakteren történik, amennyi szükséges. A tizedesek száma csak akkor adható meg, ha *V* szám. A teljes szélességbe a tizedespont és a tizedesek száma is beletartozik. Egész esetén természetesen a tizedespont után nulla következik, valós esetben, ha a Tized kisebb, mint a tizedesek száma, akkor kerekítés történik. $I- esetén az IOResult függvénnyel az írás eredményessége lekérdezhető.

Write (F, V1 [,V2 ...]) *(típusos állomány)*

F típusos állományba kiírja *V1* [*V2* ...] változó(k) értékét. Az állománymutató minden írás után automatikusan a következő komponensre lép. A változó(k) típusának és *F* alaptípusának egyezniük kell (típusazonosítójuk ugyanaz). Ha az állománymutató az utolsó komponens után áll, bővül az állomány. $I- esetén az IOResult függvénnyel az írás eredményessége lekérdezhető.

WriteLn ([Var F:Text;] V1 [,V2 ...])

Csak szöveges állományokra használható. Megfelel a Write eljárásnak, de a kifejezés(ek) után még sorvégjelet (CR/LF) is kiír – képernyőre íráskor a kurzor a következő sor elejére kerül. Kifejezés(ek) hiányában csak sorvégjelet ír ki.

Konstansok

Szöveges üzemmódok

BW40 = 0;
BW80 = 2;
Mono = 7;
C40 = 1;
CO40 = 1;
CO80 = 3;
Font8x8 = 256;
C80 = 3;

Háttér- és tintaszínek

Black = 0;
Blue = 1;
Green = 2;
Cyan = 3;
Red = 4;
Magenta = 5;
Brown = 6;
LightGray = 7;
Blink = 128;

További tintaszínek

DarkGray = 8;
LightBlue = 9;
LightGreen = 10;
LightCyan = 11;
LightRed = 12;
LightMagenta = 13;
Yellow = 14;
White = 15;

Változók

CheckBreak : Boolean;
CheckEof : Boolean;
CheckSnow : Boolean;
DirectVideo : Boolean;
LastMode : Word;
TextAttr : Byte;
WindMin, WindMax : Word;

Eljárások, függvények

AssignCrt (Var F:Text)

Az eljárás az *F* Text típusú állományváltozót hozzárendeli a Turbo Pascal-ban definiált CRT perifériához. Használatával gyorsabb képernyőkezelést lehet elérni. A szabványos Input és Output állományokat az egység inicializáló része automatikusan hozzárendeli a CRT-hez és megnyitja.

ClrEol

Az aktuális háttérszínnel törli a kurzor és a sor vége közötti karaktereket. Az eljárás ablakrelatív, a kurzor pozíciója változatlan marad.

ClrScr

Az eljárás az aktuális háttérszínnel törli az aktuális szöveges ablakot, és a kurzort az ablak bal felső sarkába, az (1,1) pozícióba viszi.

Delay (Ms:Word)

Az eljárás *Ms* milliszekundum késleltetést vált ki.

DelLine

Az aktuális szöveges ablakban a kurzor sorát törli, és a törölt sor alatti sorokat feljebb pergeti. Az ablak utolsó sorát az aktuális háttérszínnel törli. Az eljárás ablakrelatív, a kurzor helye nem változik.

GotoXY (X,Y:Byte)

A kurzort az aktuális szöveges ablak (*X*,*Y*) karakter-pozíciójába viszi. *X* a vízszintes (oszlop), *Y* a függőleges (sor) koordináta. Az eljárás ablakrelatív, az ablak bal felső sarka az (1,1) pont. Ha az (*X*,*Y*) koordináta nem esik az ablakba, akkor a kurzor helye nem változik.

HighVideo

Nagy intenzitásúra (fényesre) változtatja az aktuális tintaszínt szöveges üzemmódban. Az eljárás a TextAttr változó 3. bitjét állítja 1-re, vagyis a kiírás a továbbiakban mindenképpen egy 8 és 15 közé eső színnel fog történni.

InsLine

Az aktuális szöveges ablakban a kurzor sorába beszúr egy üres sort az aktuális háttérszínnel. A beszúrt sor alatti összes sort lejjebb pergeti, az utolsó sor elvész. Az eljárás ablakrelatív, a kurzor helye nem változik.

KeyPressed : Boolean

A függvény értéke True, ha a billentyűzetpufferben van kiolvasásra várakozó karakter, egyébként False. A puffer tartalma nem változik. A függvény nem érzékeli az ALT, SHIFT, CTRL, NUM LOCK és CAPS LOCK módosító billentyűket önmagukban. Bizonyos billentyűk és billentyűkombinációk kettős kódot generálnak (lásd Billentyűzet kódtábla). A puffer tartalmát a ReadKey függvénnyel olvashatjuk ki.

LowVideo

Kis intenzitásúra változtatja az aktuális tintaszínt szöveges üzemmódban. Az eljárás a TextAttr változó 3. bitjét állítja 0-ra, vagyis a kiírás a továbbiakban mindenképpen egy 0 és 7 közé eső színnel fog történni.

NormVideo

Az eljárás visszaállítja a program indításakor a kurzor pozíciójában detektált szöveges háttér- és tintaszínt.

NoSound

Az eljárás kikapcsolja a beépített hangszórót.

ReadKey : Char

A függvény segítségével a billentyűzetet olvashatjuk úgy, hogy a képernyőn nem történik visszajelzés. A leütött billentyű kódja (scan kód) a billentyűzet pufferébe kerül, a legutoljára eltárolt kód mögé – ha van még számára hely. Bizonyos billentyűk illetve kombinációk esetében két karakter kerül a pufferbe (kiterjesztett scan kód). Ilyenkor az első karakter #0, mely önmagában sosem kerülhet a pufferbe (lásd Billentyűzet kódtábla). A puffer sorként működik, azaz kiolvasni mindig a legelőször bekerült karaktert lehet. Ha a függvény hívásakor a billentyűzetpufferben van kiolvasásra várakozó karakter (KeyPressed=True), akkor a függvény rögtön visszatér, egyébként billentyűleütésre vár. A függvény értéke a pufferbe legelőször bekerült karakter lesz. A pufferben lévő karakterek száma eggyel csökken. Figyelnünk kell arra, hogy az éppen soron következő kód kettős-e, mert a függvény mindig csak egy karaktert vesz ki a pufferből! Ha CheckBreak értéke False,

akkor a Ctrl-Break leütése #3-as karaktert generál, egyébként a program futása megszakad.

Sound (Hz:Word)

Az eljárás bekapcsolja a belső hangszórót, és *Hz* frekvenciájú hangot bocsát ki. A Hz (Hertz) a frekvencia mértékegysége, és az 1 mp alatti rezgések számát adja meg. A hangszóró csak a NoSound eljárással kapcsolható ki. Egyszerre csak egy hangot lehet megszólaltatni, egy újabb Sound eljárás módosítja a hangmagasságot. Az énekelhető hangtartomány: kb. 55 Hz-tól 1300 Hz-ig terjed, a hallható hangtartomány pedig kb. 20 Hz-tól 20000 Hz-ig. A normál A (a1) hang frekvenciája 440 Hz. Minden félhang az előző $\sqrt[12]{2}$ -szöröse, egy hang oktávja a frekvencia kétszerese. A hangszínt és a hangerőt sajnos nem lehet változtatni.

TextBackground (Color : Byte)

Az eljárás szöveges üzemmódban a háttérszínt *Color*-ra állítja. Tulajdonképpen a TextAttr változó 4..6 bitjeit állítja be a megadott értékre. A Write és WriteLn utasítások ettől kezdve ilyen háttérszínnel írnak a képernyőre. Mivel a háttérszín tárolására karakterhelyenként csak 3 bit áll rendelkezésre (mint TextAttr változónál), a szöveges képernyőn karakterhelyenként 8 színből választhatunk hátteret. Ha *Color*-nak 7-nél nagyobb számot adunk, akkor az *Color* MOD 8 lesz. (Színkonstansokat lásd Konstansok, Háttér- és tintaszínek.)

TextColor (Color : Byte)

Az eljárás szöveges üzemmódban a tintaszínt *Color*-ra állítja. Tulajdonképpen a TextAttr változó 0..3 bitjeit állítja be a megadott értékre. A Write és WriteLn uta-

sítások ettől kezdve ilyen tinta-, vagyis karakterszínnel írnak a képernyőre. A tintaszín tárolására karakterhelyenként 4 bit áll rendelkezésre (mint TextAttr változónál), így a szöveges képernyőn karakterenként 16 színből választhatunk (0..15). Ha *Color*-hoz 128-at adunk, a kiírt szöveg villogni fog (TextAttr 7. bitje). (Színkonstansokat lásd Konstansok, Háttér- és tintaszínek, További tintaszínek.)

TextMode (Mode : Word)

Az eljárás *Mode* szöveges üzemmódba állítja a képernyőt, a szöveges ablak az egész képernyő lesz. DirectVideo változót True értékűre állítja, színes üzemmód esetén CheckSnow is True lesz. Meghívja a NormVideo eljárást, az aktuális üzemmódot eltárolja LastMode változóban. TextMode(LastMode) eljárás a legutóbb aktív szöveges üzemmódot állítja vissza, például grafikus üzemmódból való visszatérés után (lásd Graph egység, RestoreCrtMode eljárás.)

WhereX : Byte

Szöveges üzemmódban a kurzor X (vízszintes) karakter-pozícióját adja vissza. A függvény értéke ablakrelatív, a bal felső sarok az (1,1) koordináta.

WhereY : Byte

Szöveges üzemmódban a kurzor Y (függőleges) karakter-pozícióját adja vissza. A függvény értéke ablakrelatív, a bal felső sarok az (1,1) koordináta.

Window (X1,Y1,X2,Y2:Byte)

Az eljárás szöveges üzemmódban az ablakot állítja be (*X1*,*Y1*) bal felső, (*X2*,*Y2*) jobb alsó sarokpozíciókkal. Minden további képernyőművelet erre az ablakra fog vonatkozni mindaddig, amíg az ablakot meg nem változtatjuk. Az aktuális ablak

alapértelmezésben az egész képernyő, pl. CO80 üzemmód esetén (1,1,80,25). A megadott koordináták abszolútak, vagyis függetlenek az aktuális ablakbeállítástól. A bal felső sarok az (1,1), a beállított koordinátákat a WindMin és WindMax változók őrzik.

Változó

Lst : Text ;

Az egység inicializáló része automatikusan hozzárendeli az első nyomtatót (LPT1-et, vagy másképpen PRN-t) Lst-hez, és megnyitja azt írásra. Így az egységet használó program írhat az Lst állományba – azaz a nyomtatóra.

Típusok

CPU regiszterek

```
Registers = Record
    Case Integer Of
        0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Word);
        1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
End;
```

Dátumrekord

```
DateTime = Record
    Year : Word;
    Month : Word;
    Day : Word;
    Hour : Word;
    Min : Word;
    Sec : Word;
End;
```

Keresőrekord *(FindFirst és FindNext)*

```
SearchRec = Record
    Fill : Array[1..21] Of Byte;
    Attr : Byte;
    Time : LongInt;
    Size : LongInt;
    Name : String[12];
End;
```

Parancs-sor karakterlánc

```
ComStr = String[127];
```

Útvonal karakterlánc

```
PathStr = String[79];
```

Állománynév karakterlánc

```
NameStr = String[8];
```

Katalógus karakterlánc

```
DirStr = String[67];
```

Kiterjesztés karakterlánc

```
ExtStr = String[4];
```

Szöveges B/K puffer

```
TextBuf : Array[0..127] Of Char;
```

Szöveges állomány adatai

```
TextRec = Record
    Handle : Word;
    Mode : Word;
    BufSize : Word;
    Private : Word;
    BufPos : Word;
    BufEnd : Word;
    BufPtr : ^TextBuf;
    OpenFunc : Pointer;
    InOutFunc : Pointer;
    FlushFunc : Pointer;
    CloseFunc : Pointer;
    UserData : Array[1..16] Of Byte;
    Name : Array[0..79] Of Char;
    Buffer : TextBuf;
End;
```

Típusos, típusnélküli állomány adatai

```
FileRec = Record
    Handle : Word;
    Mode : Word;
    RecSize : Word;
    Private : Array[1..26] Of Byte;
    UserData : Array[1..16] Of Byte;
    Name : Array[0..79] Of Char;
End;
```

Konstansok

Állományok állapotai
ReadOnly = $01;
Hidden = $02;
SysFile = $04;
VolumeID = $08;
Directory = $10;
Archive = $20;
AnyFile = $3F;

Megnyitási módok
fmClosed = $D7B0;
fmOutput = $D7B2;
fmInput = $D7B1;
fmInOut = $D7B3;

Flag konstansok
FCarry = $0001;
FZero = $0040;
FParity = $0004;
FSign = $0080;
FAuxiliary = $0010;
FOverflow = $0800;

Változó

DosError : Integer;

2	Az állomány nem található
3	Az útvonal nem található
5	Tiltott hozzáférés
6	Érvénytelen kezelőszám (handle)
8	Kevés a memória
10	Érvénytelen környezet
11	Érvénytelen formátum
18	Nincs több állomány

Eljárások, függvények

DiskFree (Drive:Word) : LongInt

Visszaadja a *Drive* lemezegységben lévő lemez szabad területének méretét. Ha *Drive* értéke 0, akkor az aktuális, ha 1 az A:, ha 2 a B:, stb. lemezegységről van szó. Ha a lemezegység száma (*Drive* paraméter) hibás, vagy a lekérdezés sikertelen, akkor a függvény értéke -1 lesz.

DiskSize (Drive:Word) : LongInt

Visszaadja a *Drive* lemezegységben lévő lemez teljes kapacitását. Ha *Drive* értéke 0, akkor az aktuális, ha 1 az A:, ha 2 a B:, stb. lemezegységről van szó. Ha a lemezegység száma (*Drive* paraméter) hibás, vagy a lekérdezés sikertelen, akkor a függvény értéke -1 lesz.

DosExitCode : Word

Az Exec eljárással hívott program visszatérési kódját adja vissza. Az alsó bájt tartalmazza a program által küldött kódot: futási hibakódot, vagy a Halt illetve Keep eljárás paraméterét. A felső bájt értéke:

0 ha normálisan lefutott az alprogram
1 ha Ctrl-C -vel megszakították
2 ha eszközhiba miatt állt le
3 ha Keep eljárással fejeződött be futása

A függvény csak egyszer adja vissza a kért értéket, utána nullázódik.

DosVersion : Word

Visszaadja a DOS verziószámát. Az alsó bájt tartalmazza a fő-, a felső bájt az alverziószámot. DOS 3.31 esetén például az alsó bájt 3, a felső 31.

EnvCount : Integer

A DOS környezeti változók számát adja vissza.

EnvStr (Index:Integer) : String

Az *Index*-edik DOS környezeti változót és annak értékét adja vissza. Az első DOS környezeti változó száma 1. Ha *Index* értéke kisebb, mint 1, vagy nagyobb, mint EnvCount értéke, akkor a függvény értéke üres karakterlánc lesz. A visszaadott karakterlánc formája: *Környezeti változó=Érték.*

Például: PATH=C:\DOS;C:\TP6; VER=DOS 3.31 BITFAX=C:\BITFAX

Exec (Path,CmdLine:String)

A *Path* paraméterben megadott programot *CmdLine* parancssorral futtatja. *Path* tartalmazhat lemezegységnevet és útvonalleírást is. A heap maximális felső határát akkorára kell lecsökkenteni, hogy a futtatandó külső program beférjen a memóriába (lásd $M direktíva). Hívása előtt és után használjuk a SwapVectors eljárást! Az eljárás eredményességét DOSError változó tartalmazza a hívás után – ha értéke 0, a hívott program visszatérési kódja a DosExitCode függvénnyel lekérdezhető.

A COMMAND.COM futtatásakor az első parancssor-paraméter '/C ' kell legyen!
Pl.: Exec('\Command.Com','/C Dir *.Pas');

FExpand (Path:PathStr) : PathStr

A paraméterként átadott *Path* fizikai állománynevet kiegészíti teljes állományspecifikációvá. A visszaadott karakterlánc csupa nagybetűből áll, és tartalmaz egy lemezazonosítót, egy kettőspontot, valamint egy gyökérkönyvtárból induló útvonalat. A függvény nem keresi az állományt, s szintaktikai ellenőrzés sem történik. A PathStr típust az egység definiálja. A függvény feloldja a '.' és '..' hivatkozásokat.

FindFirst (Path:String; Attr:Word; Var S:SearchRec)

A *Path* paraméterben meghatározott, megadott attribútumú katalógus-bejegyzések közül kikeresi az elsőt, és elhelyezi az *S* változóban. Az *Attr* paraméterben megadott állományokon kívül mindig visszaadja a normál (Attr=0) és az arhív (Attr= $20) állományokat is, ezért azokat szükség esetén le kell választani a beolvasott bejegyzésekről (lásd Konstansok, Állományok állapotai). *Path* tartalmazhat lemezegységnevet és útvonalleírást is. *S*.Name 12 hosszú, az csak az útvonal nélküli nevet és kiterjesztést tartalmazza. Az eljárás eredményességét a DOSError változó tartalmazza hívás után, értékei általában a következők:

0: van ilyen állomány;
3: nincs ilyen útvonal;
18: nincs ilyen állomány a megadott katalógusban.

A SearchRec típust az egység definiálja.

FindNext (Var S:SearchRec)

Az előzőleg FindFirst eljárással definiált katalógus-bejegyzések közül a következőt adja vissza az *S* paraméterben. *S*.Name 12 hosszú, az csak az útvonal nélküli nevet

és kiterjesztést tartalmazza. Az eljárás eredményességét a DOSError változó adja meg hívás után, értékei általában a következők:

0: van következő állomány;
18: nincs több ilyen állomány a megadott katalógusban.

A SearchRec típust az egység definiálja.

FSearch (Path:PathStr; DirList:String) : PathStr

A *Path* paraméterben megadott fizikai állományt megkeresi az aktuális lemezegység aktuális katalógusában, majd a *DirList* paraméterrel meghatározott katalógusokban. A *DirList* paraméterben az egyes útvonalakat egymástól pontosvesszővel elválasztva kell megadni, mint a DOS PATH környezeti változójánál. Ha a kért állományt megtalálta, akkor a visszaadott karakterlánc a megadott útvonalak egyike és az állománynév konkatenációja. Ha nem találta meg, akkor a függvény értéke üres karakterlánc lesz. A PathStr típust az egység definiálja.

FSplit (Path:PathStr; Var Dir:DirStr; Var Name:NameStr; Var Ext:ExtStr)

A *Path* paraméterben megadott állományspecifikációt komponenseire bontva adja vissza. *Dir* tartalmazza a lemezegységkódot és az útvonalat az esetleges kezdő és záró \ jellel együtt, *Name* az állomány nevét, *Ext* pedig a kiterjesztést az elválasztó ponttal együtt. A szétbontásnál szintaktikai ellenőrzés nem történik. Ha *Path*-ban nem adunk meg egy komponenst, akkor az annak megfelelő változó üres karakterlánc lesz. A PathStr, DirStr, NameStr és ExtStr típusokat az egység definiálja.

GetCBreak (Var Break:Boolean)

Break paraméterében visszaadja, hogy a DOS Ctrl-Break ellenőrzése aktív-e, vagy sem. Ha értéke False – vagyis az ellenőrzés inaktív –, a DOS csak periféria-műveletek hívásakor (konzol, printer, kommunikációs eszközök) fogadja el a program Ctrl-Break-kel történő megszakítását, míg True esetén minden rendszer-híváskor.

GetDate (Var Year,Month,Day,DayOfWeek:Word)

Az operációs rendszer szerinti aktuális dátumot adja vissza paramétereiben: *Year* az évszámot (1980..2099), *Month* a hónap számát (1..12), *Day* a nap számát (1..31) és *DayOfWeek* a hét napjának sorszámát (0 a vasárnap, 1 a hétfő, stb.) tartalmazza visszatéréskor.

GetEnv (EnvVar:String) : String

Az *EnvVar* paraméterben meghatározott DOS környezeti változó értékét adja vissza. Ha az nem létezik, a függvény értéke üres karakterlánc lesz.

GetFAttr (Var F; Var Attr:Word)

Az *F*-hez rendelt fizikai állomány attribútumát adja vissza az *Attr* paraméterben (lásd Konstansok, Állományok állapotai). Az állománynak nem szabad nyitva lennie az eljárás hívásakor. *F* bármilyen típusú állományváltozó lehet.

GetFTime (Var F; Var Time:LongInt)

Az *F*-hez rendelt fizikai állomány utolsó írási dátumát és idejét adja vissza pakolt formában a *Time* paraméterben. Ez az UnpackTime eljárással szétpakolható. Az állománynak nyitva kell lennie. *F* bármilyen típusú állományváltozó lehet.

GetIntVec (IntNo:Byte; Var Vector:Pointer)

Az *IntNo* paraméterrel meghatározott megszakítási rutin címét adja vissza a *Vector* paraméterben.

GetTime (Var Hour,Minute,Second,Sec100:Word)

Paramétereiben visszaadja az operációs rendszer szerinti aktuális időt. *Hour* tartalmazza az órákat (0..23), *Minute* a perceket (0..59), *Second* a másodperceket (0..59), *Sec100* a századmásodperceket (0..99).

GetVerify (Var Verify:Boolean)

Visszaadja a DOS VERIFY jelzőjének aktuális értékét. Ha *Verify* True értékű, akkor minden lemez-íráskor külön ellenőrzés is történik, egyébként nem.

Intr (IntNo:Byte; Var Regs:Registers)

Meghívja az *IntNo* paraméterrel meghatározott megszakítást. A megszakítás hívása előtt a processzor AX, BX, CX, DX, BP, SI, DI, DS és ES regisztereit feltölti a *Regs* rekord megfelelő mezőivel, majd visszatéréskor ezeket, valamint a Flags regisztert visszatölti *Regs*-be. A Registers típust az egység definiálja. Azon megszakításokat, melyek függenek SP és SS értékétől, illetve ezeket megváltoztatják, nem szabad meghívni!

Keep (ExitCode:Word)

Leállítja a program futását, a vezérlést visszaadja a hívó programnak, illetve az operációs rendszernek, de a program a kód-, adat- és verem szegmensével, valamint a heap-pel együtt a memóriában marad (rezidens kiszállás). A kiszállási kódot

ExitCode-dal kell megadni, melyet a hívó programból a DosExitCode függvénnyel vagy batch állományban ERRORLEVEL-lel lekérdezhetünk.
Csak nagy körültekintéssel használjuk! A TSR (Terminate and Stay Resident) program vagy állándóan működik kilépés után – például óramegszakítással –, vagy azt aktivizálnunk kell. Ennek megoldása azonban egyáltalán nem könnyű feladat. A Borland cég által forgalmazott Turbo Toolbox tartalmazza a TSR Tools szoftvercsomagot, mely segítségével könnyen írhatunk rezidens programokat.

Ha a heap méretét nem korlátozzuk rezidens programokban, a rendszer – szabad memória hiányában – lefagy. Rezidens programot csak DOS alól indítsunk!

MsDos (Var Regs:Registers)

A $21-es megszakítást hívja meg (DOS belépési pont). A megszakítás hívása előtt a processzor AX, BX, CX, DX, BP, SI, DI, DS és ES regisztereit feltölti a *Regs* rekord megfelelő mezőivel, majd visszatéréskor ezeket, valamint a Flags regisztert visszatölti *Regs*-be. A Registers típust az egység definiálja.

Azon funkciókat, melyek függenek SP és SS értékétől, illetve ezeket megváltoztatják, nem szabad meghívni!

PackTime (Var DT:DateTime; Var Time:LongInt)

DT rekord tartalmát pakoltan elhelyezi *Time* paraméterben. A DateTime típust az egység definiálja. Lásd SetFTime.

SetCBreak (Break:Boolean)

Az eljárással a DOS Ctrl-Break ellenőrzése be- vagy kikapcsolható. Ha *Break* értéke False, a programok csak periféria-műveletek hívásakor szakíthatók meg Ctrl-Break-kel, míg True esetén minden rendszer-híváskor.

SetDate (Year,Month,Day:Word)

Beállítja az operációs rendszer dátumát. *Year* tartalmazza az évszámot (1980..2099), *Month* a hónap (1..12) és *Day* a nap számát (1..31). Ha a megadott dátum érvénytelen, akkor az eljárás hatástalan.

SetFAttr (Var F; Attr:Word)

Az *F*-hez rendelt fizikai állomány attribútumát megváltoztatja az *Attr* paraméterben megadottra (lásd Konstansok, Állományok állapotai). Az állománynak zárva kell lennie. *F* bármilyen típusú állományváltozó lehet.

SetFTime (Var F; Time:LongInt)

Beállítja az *F*-hez rendelt fizikai állomány utolsó írásának dátumát és idejét, melyet a *Time* paraméterben pakolt formában adunk meg. Az állománynak nyitva kell lennie. *Time* értéke a PackTime eljárással állítható elő. *F* bármilyen típusú állományváltozó lehet.

SetIntVec (IntNo:Byte; Vector:Pointer)

Beállítja az *IntNo* (0..255) sorszámú megszakítási vektort a *Vector* paraméterrel meghatározott címre. A *Vector* értékét sokszor úgy határozzuk meg, hogy a @

operátorral rámutatunk a megszakítási rutinra. A megszakítási rutint magunk is megírhatjuk.

SetTime (Hour,Minute,Second,Sec100:Word)

Beállítja az operációs rendszer szerinti időt. *Hour* az órák (0..23), *Minute* a percek (0..59), *Second* a másodpercek (0..59) és *Sec100* a századmásodpercek (0..99) értékét tartalmazza híváskor. Ha a megadott idő érvénytelen, az eljárás hatástalan.

SetVerify (Verify:Boolean)

Beállítja a DOS VERIFY jelzőjének aktuális értékét. Ha *Verify* True értékű, akkor minden lemez-íráskor külön ellenőrzés is történik, egyébként nem.

SwapVectors

A System egységben deklarált SaveIntXX – megszakítási vektorokat tartalmazó – változók értékét felcseréli a hozzájuk tartozó megszakítási vektorokkal. Tipikusan az Exec eljárás hívása előtt és után használatos. A fordító a főprogramhoz olyan kódot generál, mely a deklarált megszakítási vektorokat induláskor elmenti, kilépéskor pedig visszaállítja.

UnpackTime (Time:LongInt; Var DT:DateTime)

A *DT* rekordváltozóba szétpakolja a *Time* paraméterrel meghatározott pakolt dátumot és időt. A DateTime típust az egység definiálja.

Típusok

Grafikus ablakjellemzők
bal felső és jobb alsó sarok, vágás

```
ViewPortType = Record
    X1, Y1 : Integer;
    X2, Y2 : Integer;
    Clip : Boolean;
End;
```

Szövegjellemzők
betűtípus, szövegállás, karakterméret
vízszintes és függőleges igazítás

```
TextSettingsType = Record
    Font : Word;
    Direction : Word;
    CharSize : Word;
    Horiz : Word;
    Vert : Word;
End;
```

Kitöltési minta
felhasználói minta

```
FillPatternType = Array[1..8] Of Byte;
```

Vonaljellemzők
stílus, felhasználói minta, vastagság

```
LineSettingsType = Record
    LineStyle : Word;
    Pattern : Word;
    Thickness : Word;
End;
```

Ívjellemzők
középpont, kezdő és végpont

```
ArcCoordsType = Record
    X, Y : Start;
    XStart, YStart : Integer;
    XEnd, YEnd : Integer;
End;
```

Kitöltési jellemzők
stílus, szín

```
FillSettingsType = Record
    Pattern : Word;
    Color : Word;
End;
```

Palettajellemzők
méret, színek
```
PaletteType = Record
    Size : Byte;
    Colors : Array[0..MaxColors] Of ShortInt;
End;
```

Pontkoordináta
X és Y pozíció
```
PointType = Record
    X, Y : Integer;
End;
```

Konstansok

Grafikus hibakódok *(GraphResult lehetséges értékei)*

grOk	=	0 ;			
grNoInitGraph	=	-1 ;	grFontNotFound	=	-8 ;
grNotDetected	=	-2 ;	grNoFontMem	=	-9 ;
grFileNotFound	=	-3 ;	grInvalidMode	=	-10 ;
grInvalidDriver	=	-4 ;	grError	=	-11 ;
grNoLoadMem	=	-5 ;	grIOError	=	-12 ;
grNoScanMem	=	-6 ;	grInavlidFont	=	-13 ;
grNoFloodMem	=	-7 ;	grInvalidFontNum	=	-14 ;

Grafikus meghajtók

Detect	=	0 ;	EGAMono	=	5 ;
CGA	=	1 ;	IBM8514	=	6 ;
MCGA	=	2 ;	HercMono	=	7 ;
EGA	=	3 ;	ATT400	=	8 ;
EGA64	=	4 ;	VGA	=	9 ;

PC3270	=	10 ;	CurrentDriver	=	-128 ;

Grafikus üzemmódok

CGAC0	=	0 ;	EGALo	=	0 ;
CGAC1	=	1 ;	EGAHi	=	1 ;
CGAC2	=	2 ;	EGA64Lo	=	0 ;
CGAC3	=	3 ;	EGA64Hi	=	1 ;
CGAHi	=	4 ;	EGAMonoHi	=	3 ;
MCGAC0	=	0 ;	ATT400C0	=	0 ;
MCGAC1	=	1 ;	ATT400C1	=	1 ;
MCGAC2	=	2 ;	ATT400C2	=	2 ;
MCGAC3	=	3 ;	ATT400C3	=	3 ;
MCGAMed	=	4 ;	ATT400Med	=	4 ;
MCGAHi	=	5 ;	ATT400Hi	=	5 ;
VGALo	=	0 ;	IBM8514Lo	=	0 ;
VGAMed	=	1 ;	IBM8514Hi	=	1 ;
VGAHi	=	2 ;			
HercMonoHi	=	0 ;	PC3270Hi	=	0 ;

Normál színek

Black	=	0 ;	DarkGray	=	8 ;
Blue	=	1 ;	LightBlue	=	9 ;
Green	=	2 ;	LightGreen	=	10 ;
Cyan	=	3 ;	LightCyan	=	11 ;
Red	=	4 ;	LightRed	=	12 ;
Magenta	=	5 ;	LightMagenta	=	13 ;

Brown	=	6 ;	Yellow	=	14 ;
LightGray	=	7 ;	White	=	15 ;

EGA színek

EGABlack	=	0 ;	EGADarkGray	=	56 ;
EGABlue	=	1 ;	EGALightBlue	=	57 ;
EGAGreen	=	2 ;	EGALightGreen	=	58 ;
EGACyan	=	3 ;	EGALightCyan	=	59 ;
EGARed	=	4 ;	EGALightRed	=	60 ;
EGAMagenta	=	5 ;	EGALightMagenta	=	61 ;
EGABrown	=	20 ;	EGAYellow	=	62 ;
EGALightGray	=	7 ;	EGAWhite	=	63 ;

Vonalstílus

SolidLn	=	0 ;	———			
DottedLn	=	1 ;	- - - -	DashedLn	=	3 ; — — —
CenterLn	=	2 ;	— - —	UserBitLn	=	4 ; *felhaszn.*

Vonalvastagság

NormWidth	=	1 ;	———	ThickWidth	=	3 ; ▬▬▬

Betűtípus

DefaultFont	=	0 ;	*bittérkép*			
TriplexFont	=	1 ;		SansSerifFont	=	3 ;
SmallFont	=	2 ;		GothicFont	=	4 ;

Szövegállás

HorizDir	=	0 ;	VertDir	=	1 ;

Felhasználói karakterméret

UserCharSize	=	0 ;

Szövegigazítás *(a grafikus kurzor pozícióját a példákban · jelzi)*

LeftText	=	0 ;	·AB	BottomText	=	0 ; .AB
CenterText	=	1 ;	A·B	*CenterText*	=	*1 ;* ·AB
RightText	=	2 ;	AB·	TopText	=	2 ; ˙AB

Grafika vágás

ClipOn	=	True ;	ClipOff	=	False ;

Tetőrajzolás

TopOn	=	True ;	TopOff	=	False ;

Kitöltési stílus

EmptyFill	=	0 ;				
SolidFill	=	1 ;	HatchFill	=	7 ;	
LineFill	=	2 ;	XHatchFill	=	8 ;	
LtSlashFill	=	3 ;	InterleaveFill	=	9 ;	
SlashFill	=	4 ;	WideDotFill	=	10 ;	
BkSlashFill	=	5 ;	CloseDotFill	=	11 ;	
LtBkSlashFill	=	6 ;	UserDotFill	=	12 ;	*felhaszn.*

Legnagyobb színkód érték

```
MaxColors       =    15 ;
```

Rajzolási mód *(a * jelű állandók csak a PutImage eljárásnál használhatók)*

```
CopyPut         =    0 ;        ORPut       =    2 ;*
XORPut          =    1 ;        ANDPut      =    3 ;*
                                NOTPut      =    4 ;*
```

Változók

```
GraphGetMemPtr : Pointer;      GraphFreeMemPtr : Pointer;
```

Eljárások, függvények

A külön nem jelölt szubrutinok (funkciójuktól függően) mindig az aktuális beállításokkal dolgoznak (alapértelmezéseket lásd GraphDefaults eljárásnál):

- háttérszín (lásd SetBkColor)
- tintaszín (lásd SetColor,)
- vonaljellemzők (lásd SetLineStyle)
- kitöltési jellemzők (lásd SetFillStyle, SetFillPattern)
- rajzolási mód (lásd SetWriteMode) – csak egyenes vonalak esetében
- szövegjellemzők (lásd SetTextStyle, SetTextJustify)
- A szögek fokokban értendők, az óra járásával ellentétesen növekszenek. (0: , 90: , 180: , 270: , ... periodikus.)

A koordináták mindig ablakrelatívak, kivéve a SetViewPort-ban.

A hibakódok a GraphResult függvénnyel lekérdezhetők.

A ✎-val jelölt szubrutinok hívásakor grafikus üzemmódban kell lennie a gépnek.

Arc (X,Y:Integer; StAngle, EndAngle, Radius:Word) ✎

Körívet rajzol az (*X*,*Y*) középpont köré *Radius* (vízszintes) sugárral az aktuális rajzoló színnel. A körív *StAngle* szögtől *EndAngle* szögig rajzolódik. A szögbejárás az óra járásával ellentétes irányú – 3 óránál van a 0 fok; 12 óránál a 90 fok; stb. Az eljárás a grafikus kártyára jellemző arányossági tényezőt felhasználva igazi körívet próbál rajzolni, vagyis a függőleges sugarat ez alapján számítja (alapértelmezésben az arányossági tényező úgy van beállítva, hogy a kör körnek látszódjék – lásd GetAspectRatio eljárás). A legutóbb húzott ív adatai a GetArcCoords eljárással lekérdezhetők.

Bar (X1,Y1,X2,Y2:Integer) ✎

Keretvonal nélküli kitöltött négyszöget rajzol. (*X1*,*Y1*) és (*X2*,*Y2*) a négyszöget definiáló két átellenes sarok koordinátái. A kitöltés mintáját és színét a SetFillStyle illetve SetFillPattern eljárásokkal lehet beállítani. Keretes kitöltött négyszöget a Bar3D eljárással lehet rajzolni 0 mélység mellett.

Bar3D (X1,Y1,X2,Y2:Integer; Depth:Word; Top:Boolean) ✎

Háromdimenziós, keretvonallal határolt téglatestet rajzol. (*X1*,*Y1*) és (*X2*,*Y2*) az elülső lap két átellenes sarkának koordinátái; *Depth* a mélység; ha *Top* értéke True, akkor a téglatest tetejét is megrajzolja, egyébként nem (lásd Konstansok, Tetőrajzolás). Az eljárás a téglatest elülső lapját kitölti. A kitöltés mintáját és színét a Set-

FillStyle illetve SetFillPattern eljárásokkal lehet beállítani. A keret az aktuális vonalstílussal és színnel rajzolódik, mely adatokat a SetLineStyle és SetColor eljárásokkal lehet beállítani.

Circle (X,Y:Integer; Radius:Word)

(*X,Y*) középpont köré *Radius* sugárral kört rajzol az aktuális színnel, melyet a SetColor eljárással állíthatunk be. Az eljárás a grafikus kártyára jellemző arányossági tényezőt felhasználva igazi kört próbál rajzolni, vagyis a függőleges sugarat ez alapján számítja (alapértelmezésben a arányossági tényező úgy van beállítva, hogy a kör körnek látszódjék – lásd GetAspectRatio).

ClearDevice

Törli a teljes képernyőt az aktuális háttérszínnel, és a grafikus kurzort az aktuális ablak (0,0) pontjába viszi.

ClearViewPort

A kitöltési színt beállítja az aktuális háttérszínre (nulladik palettaszín), és a Bar eljárás hívásával törli az aktuális grafikus ablakot. A grafikus kurzort az ablak (0,0) pontjába viszi.

CloseGraph

Visszaállítja a grafikus rendszer indítása előtti szöveges üzemmódot és felszabadítja a heap-ben a grafikus rendszer számára lefoglalt területet (lásd InitGraph). Az eljárás azokat a heap-területeket is felszabadítja, amelyeket a GraphGetMem és GraphFreeMem eljárásokkal allokáltunk grafikus meghajtók és fontok számára.

DetectGraph (Var GraphDriver,GraphMode:Integer)

Ezzel az eljárással lehet a gépben levő grafikus kártyát lekérdezni anélkül, hogy grafikus üzemmódba váltanánk. *GraphDriver* paraméterben kapjuk meg a detektált grafikus meghajtó kódját, *GraphMode*-ban pedig annak legnagyobb felbontású üzemmódját. Ezek a paraméterek adhatók át aztán például az InitGraph eljárásnak. Ha a gépben nincs grafikus meghajtó, akkor mind *GraphDriver*, mind a GraphResult függvény -2 értékű lesz (grNotDetected).

DrawPoly (NumPoints:Word; Var PolyPoints)

Az eljárás sokszöget rajzol az aktuális vonalstílussal és színnel: összeköti a *NumPoints* paraméterben megadott számú, *PolyPoints* változóban sorban megadott töréspontokat. Minden töréspont két Word típusú értékből áll: az X és Y koordinátából. *PolyPoints* típus nélküli paraméter: a programozónak kell gondoskodnia arról, hogy a megfelelő számú töréspont-pár ezen a memóriahelyen legyen elhelyezve. A töréspontok megadásához használható az egység által definiált PointType típus. Zárt sokszög rajzolása esetén az első és utolsó töréspontnak meg kell egyeznie. A rajzolási mód a SetWriteMode eljárással állítható be.

Ellipse(X,Y:Integer; StAngle,EndAngle,XRadius,YRadius:Word)

(*X*,*Y*) középpont köré *XRadius* vízszintes és *YRadius* függőleges sugárral ellipszis ívet rajzol *StAngle* szögtől *EndAngle* szögig. A szögbejárás az óra járásával ellentétes irányú – 3 óránál van a 0 fok; 12 óránál a 90 fok; stb. A legutóbb húzott ív adatai a GetArcCoords eljárással lekérdezhetők.

FillEllipse (X,Y:Integer; XRadius,YRadius:Word)

Kitöltött ellipszist rajzol (*X,Y*) középpont köré *XRadius* vízszintes, *YRadius* függőleges sugárral. Az eljárás egyben a körvonalat is megrajzolja az aktuális vonalstílussal. A kitöltés stílusa és színe a SetFillStyle eljárással állítható.

FillPoly (NumPoints:Word; Var PolyPoints)

Megfelel a DrawPoly eljárásnak, de a sokszöget ki is tölti. Ha a töréspontok által definiált vonalak keresztezik egymást, vagy a sokszög nem zárt, a kitöltés lehet hiányos, illetve túlcsordulhat. A kitöltés stílusa és színe a SetFillStyle eljárással állítható.

FloodFill (X,Y:Integer; Border:Word)

A képernyő azon *Border* színnel határolt területét tölti ki, melynek egyik pontját az (*X,Y*) koordinátákkal adjuk meg. Ha a határolószín nem folytonos, a kitöltés a területen kívül is folytatódik. A kitöltés stílusa és színe a SetFillStyle eljárással állítható. A kitöltés befejeződik, ha két üres vonal kerül kirajzolásra (pl. ha ritka a minta és kicsi a kitöltendő terület. Ha kitöltés közben elfogy a memória, akkor GraphResult -7 értéket ad vissza (grNoFloodMem). IBM8514 meghajtóval nem működik.

GetArcCoords (Var ArcCoords:ArcCoordsType)

Az utolsó Arc vagy Ellipse eljárással rajzolt ív adatait (közép-, kezdő- és végpont koordináták) elhelyezi az *ArcCoords* paraméterben. Az ArcCoordsType típust az egység definiálja.

GetAspectRatio (Var XAsp,YAsp:Word)

Minden grafikus meghajtó és üzemmód rendelkezik bizonyos torzítással, melyet az eljárás által visszaadott értékekből lehet kiszámítani. Az arányossági tényező (*XAsp* Div *YAsp*) felhasználható a torzítás korrigálására.

GetBkColor : Word

A függvény visszaadja a paletta azon indexét, amely az aktuális háttérszínt tartalmazza. Ez az index grafikus meghajtótól és módtól függően 0 és 15 közé eshet (a palettán max. 16 szín fér el). A grafika indítása illetve a SetPalette és SetAllPalette eljárások után ez az érték nulla, SetBkColor(Color) végrehajtása után pedig Color az értéke. Lásd SetBkColor eljárás.

GetColor : Word

Visszaadja a paletta azon indexét, amely szerint a rajzolószínt beállítottuk. A rajzolószín alapértelmezésben a paletta utolsó színe, de azt a SetColor eljárással megváltoztathatjuk.

GetDefaultPalette (Var Pal:PaletteType)

Visszaadja *Pal* paraméterben azt a palettabeállítást, mely a grafikus rendszer indításakor volt érvényes. A PaletteType típust az egység definiálja.

GetDriverName : String

Visszaadja az aktív grafikus meghajtó nevét, mint karakterlánc típusú értéket.

GetFillPattern (Var FillPattern:FillPatternType)

Az előzőleg SetFillPattern eljárással beállított kitöltési mintát adja vissza a *FillPattern* paraméterben. A FillPatternType típust az egység definiálja. Ha a grafikus üzemmód indítása óta nem hívtuk a SetFillPattern eljárást, akkor $FF értékkel feltöltött tömböt kapunk.

GetFillSettings (Var FillInfo:FillSettingsType)

Az előzőleg SetFillStyle eljárással beállított kitöltési jellemzőket (stílus és szín) adja vissza a *FillInfo* paraméterben. A FillSettingsType típust az egység definiálja. Ha a *FillInfo*.Pattern = UserFill, akkor a felhasználói bitmintát a GetFillPattern eljárással kérdezhetjük le. Ha a grafikus üzemmód hívása óta még nem hívtuk a SetFillStyle eljárást, akkor a visszaadott információ az alapértelmezett lesz: Pattern=SolidFill; Color=GetMaxColor.

GetGraphMode : Integer

Visszaadja az előzőleg InitGraph vagy SetGraphMode eljárással beállított grafikus üzemmódot. A függvény értéke 0 és 5 között lehet az aktív meghajtó függvényében.

GetImage (X1,Y1,X2,Y2:Integer; Var BitMap)

A grafikus képernyő (*X1*,*Y1*) és (*X2*,*Y2*) pontok által meghatározott négyszögletes területét elmenti *BitMap* változóba. *BitMap* első négy bájtja az elmentett terület vízszintes és függőleges méretét tartalmazza. A szükséges memóriaméret az ImageSize függvénnyel lekérdezhető, és nem haladhatja meg a 64K-t. Lásd PutImage eljárás.

GetLineSettings (Var LineInfo:LineSettingsType)

Az előzőleg SetLineStyle eljárással beállított vonaljellemzőket (stílus, felhasználói minta és vastagság) adja vissza a *LineInfo* paraméterben. A LineSettingsType típust az egység definiálja. Ha *LineInfo*.LineStyle <> UserBitLn, akkor *LineInfo*.Pattern értéke definiálatlan.

GetMaxColor : Word

Visszaadja a legnagyobb palettaindexet, melyet a SetColor eljárásnak át lehet adni.

GetMaxMode : Word

Visszaadja az aktuális grafikus meghajtó legnagyobb értékű üzemmódját. Lásd GetGraphMode eljárás.

GetMaxX : Integer

Visszaadja az aktuális grafikus meghajtón és üzemmódon érvényes legnagyobb X pozíció értékét.

GetMaxY : Integer

Visszaadja az aktuális grafikus meghajtón és üzemmódon érvényes legnagyobb Y pozíció értékét.

GetModeName (ModeNumber:Word) : String

Visszaadja az aktív meghajtóra érvényes, *ModeNumber* paraméterrel meghatározott grafikus üzemmód nevét, mint karakterlánc típusú értéket.

GetModeRange (GraphDriver:Integer; Var LoMode,HiMode:Integer)

A *GraphDriver* paraméterrel meghatározott grafikus meghajtón érvényes legkisebb és legnagyobb üzemmód értékét adja vissza a *LoMode* illetve *HiMode* paraméterekben. Ha a *GraphDriver* paraméter értéke érvénytelen, mind *LoMode*, mind *HiMode* -1 értékű lesz.

GetPalette (Var Palette:PaletteType)

Palette paraméterben visszaadja a paletta méretét (azaz a színek számát) és a palettaindexekhez tartozó színeket. A PaletteType típust az egység definiálja.

GetPaletteSize : Word

Visszaadja az aktuális grafikus meghajtó és üzemmód esetében, hogy a paletta hány színt tartalmaz (azaz a paletta méretét).

GetPixel (X,Y:Integer) : Word

Visszaadja az (*X*,*Y*) pozícióban lévő képpont palettaszínét.

GetTextSettings (Var TextInfo:TextSettingsType)

Visszaadja *TextInfo* paraméterben az aktuális szövegjellemzőket (betűtípus, szövegállás, karakterméret, vízszintes és függőleges igazítás), melyeket előzőleg a SetTextStyle, SetTextJustify, SetUserCharSize eljárásokkal állíthattunk be. A TextSettingsType típust az egység definiálja.

GetViewSettings (Var ViewPort:ViewPortType)

ViewPort paraméterben visszaadja az aktuális grafikus ablak jellemzőit, (bal felső, jobb alsó sarok koordinátája, grafika vágás) melyeket a SetViewPort eljárással állíthattunk be. A ViewPortType típust az egység definiálja.

GetX : Integer

A grafikus kurzor X koordinátáját adja vissza.

GetY : Integer

A grafikus kurzor Y koordinátáját adja vissza.

GraphDefaults

Alaphelyzetbe állítja a grafikus rendszer következő jellemzőit:

- grafikus ablak:
  X1=0, Y1=0,
  X2=GetMaxX, Y2=GetMaxY,
  Clip=ClipOn
- paletta:
  méret=GetMaxColor+1,
  színei: grafikus módtól függ
- háttérszín: Palette[0]
- tintaszín: Palette[GetMaxColor]
- kurzor:
  X=0, Y=0
- kitöltési jellemzők:
  stílus = SolidFill
  szín = GetMaxColor
  minta = $FF tömb
- vonaljellemzők:
  stílus = SolidLn
  felhasználói minta = $FF
  vastagság = NormWidth

- szöveges jellemzők:
  betűtípus = DefaultFont
  szövegállás = HorizDir
  karakterméret = 1
  vízszintes igazítás = LeftText
  függőleges igazítás = TopText

GraphErrorMsg (ErrorCode:Integer) : String

Visszaadja az *ErrorCode* paraméterhez tartozó grafikus hiba szövegét.

GraphResult : Integer

Az alábbi grafikus műveletek hibakódját adja vissza:

Bar	FloodFill	RegisterBGIdriver	SetLineStyle
Bar3D	GetGraphMode	RegisterBGIfont	SetPalette
ClearViewPort	ImageSize	SetAllPalette	SetTextJustify
CloseGraph	InitGraph	SetFillPattern	SetTextStyle
DetectGraph	InstallUserDriver	SetFillStyle	
DrawPoly	InstallUserFont	SetGraphBufSize	
FillPoly	PieSlice	SetGraphMode	

Hívása után a grafikus hibakód törlődik!

A grafikus hibakódok jelentése:

0 : Nincs hiba
-1 : A grafikus rendszer nem installált
-2 : Grafikus kártya nincs a gépben
-3 : A grafikus meghajtó (.BGI állomány) nem található
-4 : A grafikus meghajtó (.BGI állomány) hibás
-5 : Nincs elég memória a meghajtó (.BGI állomány) betöltéséhez
-6 : Kitöltés közben elfogyott a memória (scan fill)
-7 : Kitöltés közben elfogyott a memória (flood fill)
-8 : A betűtípus (.CHR állomány) nem található
-9 : Nincs elég memória a betűtípus (.CHR állomány) betöltéséhez
-10 : Érvénytelen üzemmód a megadott grafikus kártyán
-11 : Grafikus hiba
-12 : Grafikus B/K hiba
-13 : A betűtípus (.CHR fájl) hibás
-14 : Érvénytelen betűtípus-szám

ImageSize (X1,Y1,X2,Y2:Integer) : Word

Visszaadja, hogy mekkora memória szükséges az (*X1*,*Y1*) és (*X2*,*Y2*) koordináták által meghatározott grafikus képernyőterület GetImage eljárással történő elmentéséhez – beleértve a kép méreteit tartalmazó első négy bájtot is. Ha a képernyőterület elmentéséhez 64 kilobájtnál nagyobb memóriára lenne szükség, a függvény értéke 0 lesz. Lásd PutImage eljárás.

InitGraph (Var GraphDriver,GraphMode:Integer; DrivePath:String) ;

Grafikus üzemmódba állítja a gépet. *GraphDriver* a grafikus meghajtó, *GraphMode* az üzemmód kódját tartalmazza híváskor. A grafikus meghajtórutint tartalmazó (.BGI) állomány útvonalleírását (aktuális katalógus = '') a *DrivePath* paraméterben kell megadni, amennyiben a meghajtót nem töltöttük be a RegisterBGIDriver eljárással. Ha *GraphDriver* változó értéke 0 (Detect), akkor az eljárás a grafikus kártya függvényében indítja az üzemmódot – a legnagyobb felbontással. Az alapbeállításokat lásd a GraphDefaults eljárásnál.

InstallUserDriver (Name:String; AutoDetectPtr:Pointer) :Word;

Felhasználói grafikus meghajtó rutint installál, és visszaadja a hozzátartozó meghajtószámot. *Name* a meghajtó rutin fizikai állománynevét, *AutoDetectPtr* az opcionális detektáló függvény címét tartalmazza híváskor. Ha nincs detektáló függvény, akkor Nil értéket kell átadni *AutoDetectPtr*-nek. Ha a belső grafikus meghajtó tábla betelt, a függvény értéke -11 lesz.

InstallUserFont (FontFileName:String) : Integer ;

Felhasználói grafikus betűtípust installál, és visszaadja a hozzátartozó betűtípusszámot. *FontFileName* a betűképeket tartalmazó fizikai állomány nevét kapja meg híváskor. Ha a belső betűtípus tábla betelt, a függvény értéke 0 lesz.

Line (X1,Y1,X2,Y2:Integer)

(*X1*,*Y1*) és (*X2*,*Y2*) pontok közé egyenest húz. Az eljárás hívása után a grafikus kurzor koordinátája változatlan marad. A vonal színét a SetColor, stílusát és vastagságát a SetLineStyle eljárással állíthatjuk. A SetWriteMode eljárással a rajzolási módot adhatjuk meg.

LineRel (DX,DY:Integer)

Egyenest húz a grafikus kurzor aktuális pozíciójából az attól (*DX*,*DY*) relatív távolságra lévő pontba. A grafikus kurzor a vonal végére kerül.

LineTo (X,Y:Integer)

Egyenest húz a grafikus kurzor pozíciójából az (*X*,*Y*) pontba. A grafikus kurzor az (*X*,*Y*) pontba kerül.

MoveRel (DX,DY:Integer)

A grafikus kurzort az attól (*DX*,*DY*) relatív távolságra lévő pontba viszi, rajzolás nélkül.

MoveTo (X,Y:Integer)

A grafikus kurzort az (*X*,*Y*) pontba viszi, rajzolás nélkül.

OutText (TextString:String)

TextString szöveget megjeleníti az aktuális szövegjellemzőkkel a grafikus kurzor pozíciójában. A grafikus ablakból esetlegesen kieső szövegrészt levágja. A grafikus kurzor csak akkor kerül a szöveg végére, ha vízszintesen (Direction = HorizDir) és balra igazítva (Horiz = LeftText) írattunk ki.

OutTextXY (X,Y:Integer; TextString:String)

Megfelel az OutText eljárásnak, de a szöveget az (*X*,*Y*) pozícióban jeleníti meg.

PieSlice (X,Y:Integer; StAngle,EndAngle,Radius:Word)

Körcikket rajzol (*X*,*Y*) középponttal, *Radius* sugárral *StAngle* szögtől *EndAngle* szögig, és kitölti azt. A szögbejárás az óra járásával ellentétes irányú – 3 óránál van a 0 fok; 12 óránál a 90 fok; stb. Az eljárás a grafikus kártyára jellemző arányossági tényezőt felhasználva igazi körívet próbál rajzolni, vagyis a függőleges sugarat ez alapján számítja (alapértelmezésben a arányossági tényező úgy van beállítva, hogy a kör körnek látszódjék – lásd GetAspectRatio). Ha kitöltés közben nincs elég memória, akkor GraphResult -6 értéket ad vissza.

PutImage (X,Y:Integer; Var BitMap; BitBlt:Word)

Megjeleníti a képernyő (*X*,*Y*) pozíciójában (bal felső sarok) a *BitMap* által meghatározott képet, melyet előzőleg a GetImage eljárással mentettünk el. *BitBlt* határozza meg a rajzolási módot – CopyPut: direkt másolás (MOV); XORPut, ORPut, ANDPut: logikai műveletek a másolandó és eredeti képpont között; NOTPut: inverz kép.

PutPixel (X,Y:Integer; Pixel:Word)

Pixel színűre állítja az (*X*,*Y*) koordinátájú képernyőpontot. A grafikus kurzort nem állítja, rajzolószín marad az eredeti.

Rectangle (X1,Y1,X2,Y2:Integer)

Négyszöget rajzol. (*X1*,*Y1*), (*X2*,*Y2*) két átellenes sarok koordinátái. A vonal színét a SetColor, stílusát és vastagságát a SetLineStyle eljárással állíthatjuk. A SetWriteMode eljárással a rajzolási módot adhatjuk meg.

RegisterBGIDriver (Driver:Pointer) : Integer

Regisztrálja a felhasználó által betöltött vagy a programhoz szerkesztett gyári grafikus meghajtót (.BGI), és visszaadja a hozzárendelt meghajtószámot. *Driver* paraméternek a meghajtórutin memóriabeli címét kell átadni.

RegisterBGIFont (Font:Pointer) : Integer

Regisztrálja a felhasználó által betöltött vagy a programhoz szerkesztett gyári betűtípust (.CHR), és visszaadja a hozzárendelt betűtípus-számot. *Font* paraméternek a betűtípust leíró bájtsorozat memóriabeli címét kell átadni.

RestoreCrtMode

Visszaállítja a képernyőt az InitGraph eljárás által detektált szöveges üzemmódba. A grafikus rendszer számára lefoglalt heap-területet nem szabadítja fel – a SetGraphMode eljárással a grafikus üzemmód visszaállítható.

Sector (X,Y:Integer; StAngle,EndAngle,XRadius,YRadius:Word)

(*X*,*Y*) középpont köré *XRadius* vízszintes és *YRadius* függőleges sugárral ellipszis cikket rajzol *StAngle* szögtől *EndAngle* szögig, majd az aktuális kitöltési stílussal és színnel kitölti azt.

SetActivePage (Page:Word)

Az aktuális grafikus lapot *Page*-re (0 .. az üzemmódban érvényes maximális lapszám) állítja. Minden további grafikus utasítás az új lapra vonatkozik majd. A lapot a SetVisualPage eljárással tehetjük láthatóvá. Érvénytelen lapszám esetén az eljárás hatástalan. Több lap használatát csak az EGA (256K), VGA és Hercules grafikus kártyák támogatják. Ezzel lehetőség nyílik arra, hogy a képernyő elkészítését (mely időnként meglehetősen lassú is lehet) egy láthatatlan lapon csináljuk, majd egy hirtelen átkapcsolással láthatóvá tesszük.

SetAllPalette (Var Palette)

Megváltoztatja a színpaletta színeit. A *Palette* változó első bájtja a hossz (2 .. az üzemmódban érvényes színek száma). Ezt követik az egyes színértékek (-1 .. az üzemmódban érvényes legnagyobb szín száma - lásd Konstansok, Normál és EGA színek). Ha a színérték -1, akkor az illető palettaszín nem változik. A változás a képernyőn azonnal látható lesz. *Palette* változó lehet PaletteType típusú, melyet az egység definiál.

SetAspectRatio (XAsp,YAsp:Word)

XAsp és YAsp megadásával korrigálható a képernyő torzítása. XAsp DIV YAsp az arányossági tényező.

SetBkColor (Color:Word)

Beállíja a grafikus háttérszínt az aktuális paletta *Color* sorszámú színére (0 .. az üzemmódban érvényes legnagyobb palettaindex). Az eljárás mellékesen a nulladik palettaszínt felülírja a megadottal. Ha *Color* értéke 0, a háttér színe fekete lesz.

SetColor (Color:Word)

Beállítja a grafikus tintaszínt a paletta *Color* sorszámú színére (0 .. az üzemmódban érvényes legnagyobb palettaindex).

SetFillPattern (Pattern:FillPatternType; Color:Word)

Beállítja a *Pattern* paraméterrel megadott felhasználói kitöltési mintát és annak *Color* paraméterrel meghatározott színét (0 .. az üzemmódban érvényes legnagyobb palettaindex), melyet a FillPoly, FloodFill, Bar, Bar3D és PieSlice eljárások használnak fel. A FillPatternType típust az egység definiálja. 8 bájtban kell megadni a kitöltési mintát, ahol 1 bájt a minta 1 sorának felel meg. A sor 8 pontját a bájt 8 bitje reprezentálja. Az eljárás lehetővé teszi, hogy az előre definiált mintákon kívül – melyeket a SetFillStyle eljárás segítségével állíthatunk be – saját mintákat is definiálhassunk. Beállítás után nem kell külön meghívni a SetFillStyle eljárást.

SetFillStyle (Pattern:Word; Color:Word)

Beállítja az előre definiált kitöltési minták közül a *Pattern* paraméterrel meghatározottat (0 .. 12 – lásd Konstansok, Kitöltési stílus) és annak *Color* paraméterrel megadott színét (0 .. az üzemmódban érvényes legnagyobb palettaindex), melyet a FillPoly, Bar, Bar3D és PieSlice eljárások használnak fel. Felhasználói minta (Pattern = 12) a SetFillPattern eljárással definiálható (alapértelmezés $FF tömb).

SetGraphBufSize (BufSize:Word)

Beállítja a kitöltéseknél használt grafikus puffer méretét *BufSize*-ra. Az eljárásnak csak az InitGraph hívása előtt van hatása. Az alapméret 4 kB, mely egy kb. 650 töréssponttal rendelkező sokszög kitöltéséhez elegendő.

SetGraphMode (Mode:Integer)

Grafikus üzemmódba állítja vissza a gépet és törli a grafikus képernyőt. *Mode* paraméter (0 .. a meghajtón érvényes legnagyobb üzemmód – lásd Konstansok, Grafikus üzemmódok) határozza meg a már inicializált grafikus meghajtó üzemmódját. Az eljárás alaphelyzetbe állítja az összes grafikus jellemzőt (lásd GraphDefaults).

SetLineStyle (LineStyle:Word; Pattern:Word; Thickness:Word)

Beállítja a *LineStyle* paraméterben (0 .. 4 – lásd Konstansok, Vonal stílusok) megadott vonalstílust és a *Thickness* paraméterrel (1,3 – lásd Konstansok, Vonalvastagságok) meghatározott vonalvastagságot. Amennyiben *LineStyle* értéke 4 (UserBitLn), *Pattern*-ben kell megadni a felhasználói mintát (a szó 1 értékű bitjei rajzolódnak ki), egyébként az eljárás figyelmen kívül hagyja a paramétert.

SetPalette (ColorNum:Word; Color:ShortInt)

Megváltoztatja a színpaletta *ColorNum* (0 .. maximális palettaindex) szerinti színét *Color*-ra (0 .. az üzemmódban érvényes legnagyobb szín száma – lásd Konstanok, Normál és EGA színek). A változás a képernyőn azonnal látható lesz.

SetRGBPalette (ColorNum,Red,Green,Blue:Integer)

IBM8514 és VGA grafikus meghajtók esetében a színpaletta *ColorNum* szerinti színének alapszín értékeit állítja be. *Red* a vörös, *Green* a zöld és *Blue* a kék komponens intenzitását határozza meg. Az eljárás *Red*, *Green* és *Blue* alsó bájtjának felső hat bitjét használja fel.

SetTextJustify (Horiz,Vert:Word)

Grafikus üzemmódban *Horiz* a vízszintes (0 .. 2), *Vert* a függőleges (0 .. 2 – lásd Konstansok, Szövegigazítás) szövegigazítást állítja be. Az OutText és az OutTextXY ezután a szöveget ilyen igazításban írja ki.

SetTextStyle (Font,Direction,CharSize:Word)

Beállítja a grafikus szöveg jellemzőit. *Font* (0 .. 4 – lásd Konstansok, Betűtípusok) határozza meg a betűtípust, *Direction* (0,1 – lásd Konstansok, Szövegállás) a szövegállást és *CharSize* a méretszorzót (0 .. 10) – ez utóbbi alaphelyzetben bittérképes betűtípus esetén 1, egyéb betűtípusoknál 4. Ha *CharSize* 0 (lásd Konstansok, Felhasználói karakterméret), akkor a SetUserCharSize eljárással megadott méret lesz érvényes.

SetUserCharSize (MultX,DivX,MultY,DivY:Word)

Beállítja a nem bittérképes karakterek vízszintes és függőleges méretét. *MultX*:*DivX* hányadossal a vízszintes, *MultY*:*DivY* hányadossal a függőleges alapméretet szorozza meg.

SetViewPort (X1,Y1,X2,Y2:Integer; Clip:Boolean)

Beállítja az aktív grafikus ablakot. (*X1*,*Y1*) az ablak bal felső, (*X2*,*Y2*) a jobb alsó sarkát határozza meg. Az ablak bal felső sarka lesz ezután a (0,0) pont. Ha *Clip* értéke False (lásd Konsansok, Grafika vágás), akkor az ablakon kívülre eső rajzolatok is megjelennek, egyébként nem.

SetVisualPage (Page:Word)

A látható grafikus képernyőlapot *Page*-re (0 .. az üzemmódban érvényes legnagyobb lap száma) állítja. Érvénytelen lapszám esetén az eljárás hatástalan. Az aktuális lapot – amelyen az eljárások rajzolnak – a SetActivePage állítja be. Lásd SetActivePage eljárás.

SetWriteMode (WriteMode:Integer)

Beállítja a vonalak rajzolási módját. (0:MOV, 1:XOR művelet – lásd Konstansok, Rajzolási módok), melyet a DrawPoly, Line, LineRel, LineTo és Rectangle eljárások használnak fel. CopyPut esetén a kirajzolásnál minden pont az aktuális rajzolószínre lesz állítva, függetlenül attól, hogy mi volt a képernyőn. XORPut esetén az XOR művelet határozza meg a kirajzolás színét a már képernyőn lévő pontoktól függően. Ha az XORPut rajzolást kétszer hajtjuk végre egymás után, akkor az eredeti képernyőt kapjuk vissza.

TextHeight (TextString:String) : Word

Visszaadja a *TextString* szöveg magasságát – képpontokban.

TextWidth (TextString:String) : Word

Visszaadja a *TextString* szöveg szélességét – képpontokban.

Típus

OvrReadFunc = Function(OvrSeg:Word) : Integer;

Konstansok

Overlay hibakódok

ovrOk = 0;
ovrError = -1;
ovrNotFound = -2;
ovrNoMemory = -3;
ovrIOError = -4;
ovrNoEmsDriver = -5;
ovrNoEmsMemory = -6;

Változók

OvrFileMode : Byte;
OvrLoadCount : Word;
OvrTrapCount :Word;
OvrReadBuf : OvrReadFunc;
OvrEMSPages : Word;

OvrResult : Integer;

Az OvrResult értékeinek jelentése:

- **0** Nincs hiba
- **-1** Overlay kezelési hiba
- **-2** Az overlay állomány nem található
- **-3** Kevés a memória az overlay puffer számára
- **-4** Overlay állomány B/K hiba
- **-5** EMS meghajtó nincs installálva
- **-6** Nincs elegendő EMS memória

Eljárások, függvények

Az Overlay egység minden eljárása és függvénye állítja az OvrResult változót – lásd értékeinek jelentését.

OvrClearBuf

Törli az overlay puffer tartalmát. Ezzel arra kényszerítjük az overlay-kezelőt, hogy minden további overlay-rutinra való hivatkozáskor az egységet töltse be lemezről vagy az EMS-ből. Ha az eljárást overlay egységből hívjuk, akkor a puffer törlése után ismét betölti az egységet. Akkor szokás használni például, ha ideiglenesen szükségünk van az overley puffer által foglalt memóriaterületre.

OvrGetBuf : LongInt

Visszaadja az overlay puffer aktuális méretét. Alapértelmezés: a legnagyobb overlay egység mérete (a puffer lehet 64K-nál nagyobb).

OvrGetRetry : LongInt

Visszaadja az overlay puffer próbálkozási területének aktuális méretét. Lásd OvrSetRetry.

OvrInit (FileName:String)

Alaphelyzetbe állítja az overlay-kezelőt és megnyitja a *FileName*, overlay egységeket tartalmazó fizikai állományt. Ha *FileName* nem tartalmaz lemezegységnevet és/vagy útvonalleírást, az állományt először az aktuális, majd az .EXE állományt tartalmazó katalógusban keresi (DOS 3.x verziótól), végül a DOS PATH környeze-

ti változójával meghatározott katalógusokban. Az eljárást meg kell hívni még az első lapozás, illetve bármely egyéb heap-foglaló művelet végrehajtása előtt.
Az overlay-állomány hozzámásolható a programhoz (Copy DOS parancs /B opcióval). Ekkor overlay-állományként a programot kell megadni, de fordításkor a **D/Standalone debugging** opciónak Off állapotban kell lennie!

OvrInitEMS

Ha az EMS meghajtó aktív és elegendő az EMS memória, betölti abba az overlay állományt. Ezután a lemezről való betöltést a gyors memóriamásolás váltja fel. A program végén az EMS automatikusan felszabadul.

OvrSetBuf (BufSize:LongInt)

Beállítja az overlay puffer méretét. *BufSize* értékének nagyobb vagy egyenlőnek kell lennie, mint a kezdeti pufferméret, és kisebb vagy egyenlőnek, mint MemAvail + OvrGetBuf. Az eljárás hívásakor a heap-nek üresnek kell lennie: szüntessük meg az összes dinamikus változót, zárjuk le a grafikus üzemmódot. Hiba esetén (ovrError illetve ovrNoMemory) az overlay-kezelő tovább működik, azonban a puffer mérete változatlan marad. A puffer mérete meghaladhatja a 64 kB-ot, hiszen annak tartalmaznia kell a legnagyobb átlapolható egység kódját és inicializált változóit is.

OvrSetRetry (Size:Longint)

Beállítja *Size* méretűre az overlay puffer próbálkozási területét. Ha egy overlay egység a próbálkozási területre kerül, vagyis a körpuffer végétől a megadott területen belül helyezkedik el, az egység tesztelve lesz. Ha ez alatt az idő alatt az egység

valamely rutinja meghívásra kerül, az egység a körpuffer elejére másolódik. A próbálkozási terület alapértelmezésben 0.

Típusok

TABuild típusok

MaxDataType *(leghosszabb adatrekord típusa)*
MaxKeyType *(leghosszabb kulcs típusa)*

Turbo Access adatállomány

```
DataFile = Record
    F : File;
    FirstFree,
    NumberFree,
    Int1 : LongInt;
    ItemSize : Word;
    NumRec : LongInt;
End;
```

```
FileName = String[66];
```

Turbo Access indexállomány

```
IndexFile = Record
    DataF : DataFile;
    AllowDuplKeys : Boolean;
    KeyL : Byte;
    RR,PP : LongInt;
    Path : TaPath;
End;
```

Konstansok

```
NoDuplicates = 0;
```

```
Duplicates = 1;
```

TAccess.Def konstansai

MaxDataRecSize *(max. rekordméret)*
MaxKeyLen *(max. kulcshossz)*
MaxHeight *(B-fa magassága)*

PageSize *(4..254)*
PageStackSize *(3..254, ➔16..32)*
Order *(PageSize Div 2)*

Változók

Ok : Boolean;

TAErrorProc : ProcPtr;

Eljárások, függvények

AddKey (Var IndexF:IndexFile; Var DataRef:LongInt; Var Key)

Kulcs hozzáadása az indexállományhoz. *IndexF* az indexállomány logikai neve, amelyhez a kulcsot hozzá akarjuk adni. *DataRef* a rekord sorszáma, ahová a kulcshoz tartozó adatokat az adatállományba felírtuk. Ez általában az AddRec rutin által visszaadott rekordszám. *Key* a felírandó kulcs. Mivel típus nélküli paraméter, elvileg bármit átadhatunk paraméterként. A programozó felelőssége, hogy az átadott paraméter String típusú legyen. Ha a karakterlánc hosszabb, mint a megengedett maximális kulcshossz, csonkul. Ha sikerült a kulcs hozzáadása az indexállományhoz, Ok True lesz. Ok False, ha már létező kulcsot akarunk felírni, és a duplikált kulcs nincs megengedve (MakeIndex és OpenIndex rutinok hívásánál Status=0-t adtunk meg).

AddRec (Var DatF:DataFile; Var DataRef: LongInt; Var Buffer)

Új rekord hozzáadása az adatállományhoz. *DatF* a Turbo Access adatállomány logikai neve. Az eljárás a *DataRef* változóba adja vissza azt a rekordsorszámot, ahová *Buffer* tartalmát felírta. Az AddKey eljárást *DataRef*-el kell majd meghívni.

Buffer a felírandó adatokat tartalmazza. A rekord méretét nyitáskor adtuk meg, a helyes típusmegadásról a programozónak kell gondoskodnia. A rutin az Ok változót nem állítja, B/K hiba esetén leáll a program. AddRec hívása előtt ajánlatos megvizsgálni, hogy van-e elég hely a lemezen. Ha van törölt rekord az adatállományban, akkor az AddRec rutin azok helyére ír, s csak szükség esetén bővíti fizikailag az állományt. Ajánlatos a rekord első 4 bájtját a rendszer céljaira átengedni, s nullázni azt. Lásd DeleteRec eljárás.

ClearKey(Var IndexF:IndexFile)

Pozícionálás az indexállomány elejére. Ezt az utasítást kell kiadni, ha az indexállományt elölről akarjuk feldolgozni szekvenciálisan (pl. rendezett lista készítése esetén). Az indexállomány egy körkörös állomány, azt oda-vissza lehet szekvenciálisan olvasgatni. Az első és az utolsó index között van egy üres index, erre áll rá a ClearKey. Ezután NextKey fog ráállni az első indexre, PrevKey viszont az utolsóra.

CloseFile (Var DatF:DataFile)

DatF logikai adatállomány lezárása. Az adatállomány típusa kötelezően DataFile. Az állományt a MakeFile vagy az OpenFile eljárással nyitottuk meg. A program befejezése előtt az adatállományokat ajánlatos lezárni.

CloseIndex (Var IndexF:IndexFile)

IndexF indexállomány zárása. Az indexállományt a MakeIndex illetve az OpenIndex eljárással nyitottuk meg. A program befejezése előtt az indexállományokat ajánlatos lezárni.

DeleteKey (Var IndexF:IndexFile; DataRef:LongInt; Var Key)

Key-vel megadott kulcs törlése *IndexF* indexállományból. A *DataRef* paramétert csak duplikált kulcs megengedése esetén kell megadnunk (lásd MakeIndex; OpenIndex), amikor is az egyforma kulcsok közül a *DataRef* rekordhivatkozású kulcsot törli. Ilyenkor a megfelelő rekordsorszám kiválasztásához használjuk a SearchKey, NextKey, PrevKey rutinokat. Ha a kulcs hosszabb a megengedettnél, a maximális hosszra csonkul. Ok True lesz, ha a törlés sikerült. Amennyiben a megadott kulcs, vagy duplikáció esetén a megadott rekordhivatkozású kulcs nincs az indexállományban, akkor törlés nem történik, és Ok False értékű lesz. Sikeres törlés esetén *DataRef* a törölt kulcs rekordpozíciója lesz.

DeleteRec (Var DatF:DataFile; DataRef:LongInt)

DataRef pozíciójú rekord törlése a *DatF* adatállományból. *DataRef* paraméter értékét a DeleteKey, NextKey, PrevKey, SearchKey és FindKey eljárások állítják. DeleteRec az Ok-t nem állítja. A *DataRef* sorszámú rekord a törölt rekordok listájára kerül: a rekord első 4 bájtja egy mutató lesz a következő törölt rekordra, vagyis az eredeti információ felülíródik. Ez egészen addig nem okoz problémát, amíg például meg nem sérül az indexállomány – ilyenkor ugyanis az adatállományból kellene felépítenünk azt. Ajánlatos ezért a törzsrekord első 4 bájtját erre a célra fenntartani, és új rekord felírásakor annak nulla értéket adni. A törölt mutató sohasem lehet nulla, mert a nulladik rekord rendszercélokat szolgál, így azt törölni nem szabad. Az adatállomány szekvenciális olvasásánál a 4 bájtos státuszról tudjuk eldönteni, hogy törölt-e a rekord, vagy sem: ha a státusz nem nulla, akkor a rekord törölt.

EraseFile (Var DatF:DataFile);

Lezárja a *DatF* adatállományt, és fizikailag törli azt. Az állományt előzőleg meg kell nyitni.

EraseIndex (Var IndexF:IndexFile)

Lezárja az *IndexF* indexállományt, és fizikailag törli azt. Az állományt előzőleg meg kell nyitni.

FileLen (Var DatF:DataFile) : LongInt

A függvény a *DatF* adatállomány fizikai hosszát adja vissza, beleértve az összes törölt, valamint a nulladik rekordot is. Az állománynak nyitva kell lennie.

FindKey (Var IndexF:IndexFile; Var DataRef:LongInt; Var Key)

Pozícionálás az *IndexF* indexállomány *Key*-vel megadott kulcsára. Ok akkor lesz True, ha van a megadottal pontosan egyező kulcs. Ha a kulcsot megtalálja, akkor annak rekordhivatkozását *DataRef*-ben helyezi el. Ha duplikált kulcsokat is megengedtünk, akkor az első ilyen kulcsra pozícionál.

FlushFile (Var DatF:DataFile)

A *DatF* legutóbbi változtatásait lemezre írja a pufferből. Az AddRec, DeleteRec, MakeFile és PutRec eljárások után szokás meghívni.

FlushIndex (Var IndexF:IndexFile)

Az *IndexF* legutóbbi változtatásait lemezre írja a pufferből. Az AddKey, DeleteKey és MakeIndex eljárások után szokás meghívni.

GetRec (Var DatF:DataFile; DataRef: LongInt; Var Buffer)

A *DatF* adatállomány *DataRef* pozíciójú rekordját beolvassa *Buffer* változóba. *Buffer* egy típus nélküli paraméter, a programozó felelőssége a megfelelő méretű memóriaterületről gondoskodni. Ha *Buffer* kisebb, mint a beolvasott rekord, akkor a *Buffer* utáni területek felülíródnak. Ok-t nem állítja.

MakeFile (Var DatF:DataFile; FileN:FileName; RecLen:Word)

Létrehoz egy Turbo Access adatállományt, melynek fizikai neve *FileN*, és hozzárendeli a *DatF* logikai állományhoz. *RecLen* az adatállomány egy rekordjának hossza bájtokban, 18 ≤ *RecLen*, melynek megadásánál a SizeOf függvényt szokás használni. Ha már létezett ilyen adatállomány a lemezen, akkor először törli azt. Az üres állomány hossza 1 rekordnyi lesz. Ok True értékű lesz, ha sikerült a létrehozás, egyébként False.

MakeIndex (Var IndexF:IndexFile; FileN:FileName; KeyLen, Status: Integer)

Létrehoz egy Turbo Access indexállományt, melynek fizikai neve *FileN*, és hozzárendeli az *IndexF* logikai állományhoz. *KeyLen* az indexállományba felírható maximális kulcshossz: SizeOf(Kulcs)-1. Ha már létezett ilyen indexállomány a lemezen, akkor először törli azt. Ha *Status* értéke Duplicates, akkor egyforma kulcsok is felvihetők az indexállományba, NoDuplicates esetén nem. Ha sikerült a létrehozás, Ok True értékű lesz, egyébként False.

NextKey (Var IndexF:IndexFile; Var DataRef:LongInt; Var Key)

IndexF mutatója a következő kulcsra áll. Általában akkor használjuk, ha az adat- vagy indexállományt szekvenciálisan, kulcs szerint növekvőleg akarjuk feldolgozni. Előzőleg a ClearKey, FindKey, vagy SearchKey rutinok egyikével rá kell állni egy adott kulcsra. (AddKey és DeleteKey eljárások hívása után a mutató definiálatlan lesz). Ok True, ha van következő kulcs, indexállomány vége esetén Ok False lesz. Az indexállomány körkörös állomány: ha Ok False, akkor a következő NextKey az indexállomány első kulcsára áll. Ha van következő kulcs, akkor *Key* fogja tartalmazni a kulcsot, *DataRef* pedig a hozzátartozó rekordhivatkozást. *DataRef* értékével hívható meg például a GetRec eljárás, mellyel a kulcshoz tartozó adatokat olvashatjuk be.

OpenFile (Var DatF:DataFile; FileN:FileName; RecLen:Word)

Megnyitja a már létező, *FileN* fizikai nevű adatállományt, és hozzárendeli *DatF* logikai állományhoz. *RecLen* az adatállomány egy rekordjának hossza bájtokban: $18 \leq RecLen$, melynek megadásánál a SizeOf függvényt szokás használni. Ha az állomány nem létezik, vagy nem sikerült a megnyitása, Ok értéke False lesz, egyébként True.

Figyelem! Az adatállományt ugyanazzal a rekordhosszal kell megnyitni, mint amivel létrehoztuk azt, egyébként futási hiba áll elő.

OpenIndex (Var IndexF:IndexFile; FileN:FileName; KeyLen,Status:Integer)

Megnyitja a már létező, *FileN* fizikai nevű indexállományt, és hozzárendeli *IndexF* logikai állományhoz. *KeyLen* az indexállományba felírható maximális kulcshossz: SizeOf(Kulcs)-1. Ha *Status* értéke Duplicates, akkor egyforma kulcsok is felvihetők az indexállományba, NoDuplicates esetén nem. Ha az indexállomány nem létezik, vagy nem sikerült a megnyitása, Ok értéke False lesz, egyébként True.

Figyelem! Az indexállományt ugyanazzal a kulcshosszal kell megnyitni, mint amivel létrehoztuk azt, egyébként futási hiba áll elő.

PrevKey (Var IndexF:IndexFile; Var DataRef:LongInt; Var Key)

IndexF mutatója az előző kulcsra áll. Általában akkor használjuk, ha az adat- vagy indexállományt szekvenciálisan, kulcs szerint csökkenőleg, visszafelé akarjuk feldolgozni. Előzőleg a ClearKey, FindKey, vagy SearchKey rutinok egyikével rá kell állni egy adott kulcsra. (AddKey és DeleteKey eljárások hívása után a mutató definiálatlan lesz). Ok True, ha van előző kulcs, indexállomány eleje esetén Ok False lesz. Az indexállomány körkörös állomány: ha Ok False, akkor a következő PrevKey az indexállomány utolsó kulcsára áll. Ha van előző kulcs, akkor *Key* fogja tartalmazni a kulcsot, *DataRef* pedig a hozzátartozó rekordhivatkozást. *DataRef* értékével hívható meg például a GetRec eljárás, mellyel a kulcshoz tartozó adatokat olvashatjuk be.

PutRec (Var DatF:DataFile; DataRef:LongInt; Var Buffer)

A *Rekpoz* sorszámú rekord felírása a *DatF* adatállományba. *Buffer* a felírandó adatokat tartalmazza. A rekord méretét nyitáskor adtuk meg, a helyes típusmegadásról a programozónak kell gondoskodnia. A rutin az Ok változót nem állítja.

SearchKey (Var IndexF:IndexFile; Var DataRef:LongInt; Var Key)

Pozícionálás az adott, vagy az első annál nagyobb kulcsra (generikus keresés). *IndexF* az indexállomány logikai neve. A SearchKey eljárást általában akkor használjuk, ha a kulcsnak csak az eleje ismert. Ha az indexállományban duplikált kulcsok is előfordulnak, akkor az első ilyenre pozícionál. Ok True, ha van ilyen kulcs, egyébként False. Ez utóbbi eset csak akkor állhat elő, ha a legnagyobb kulcsnál is nagyobbat adtunk meg, vagy üres az állomány. *Key* változóban adjuk meg a keresett kulcsot vagy annak kezdeti részláncát. Ok esetén *Key* tartalmazza a megtalált kulcsot, *DataRef* pedig a rekordhivatkozást. Ha Ok False, *Key* értéke elromlik.

UsedRecs (Var DatF:DataFile) : LongInt

A függvény visszaadja a *DatF* adatállományban lévő érvényes rekordok számát. Az állománynak nyitva kell lennie. A teljes adatállományt a nulladik rekord, az érvényes valamint a törölt rekordok teszik ki. Az adatállományban lévő törölt rekordok száma: FileLen(*DatF*) - UsedRecs(*DatF*) - 1.

Típus

```
DataSet = Record
    Data     : DataFile;
    Index    : IndexFile;
End;
```

Konstansok

ExactMatch = True; PartialMatch = False;

Változó

TARecNum : LongInt;

Eljárások, függvények

TAClose (Var DatSet:DataSet);

Lezárja az előzőleg TACreate vagy TAOpen rutinok valamelyikével megnyitott *DatSet* indexelt adatállományt.

TACreate (Var DatSet:DataSet; DatFName:FileName; RecordLen:Word;IndexFName:FileName; KeyLen:Integer)

Létrehoz egy indexelt adatállományt, vagyis egy adatállományt és egy indexállományt, és hozzárendeli ezt az együttest a *DatSet* logikai névhez. Az adatállomány fizikai neve *FileName*, az indexállományé *IndexFName*. *RecordLen* az adatállomány egy rekordjának hossza bájtokban, 18 ≤ *RecordLen*, *KeyLen* pedig az indexállomány maximális kucshossza. Ezek megadásánál a SizeOf függvényt szokás használni. Duplikált kulcsokat nem lehet felvinni. Ha már létezett ilyen adatállomány illetve indexállomány a lemezen, akkor először törli azokat. Az eljárás az Ok-t nem állítja.

TADelete (Var DatSet:DataSet; Var Key)

Törli a *DatSet* indexelt adatállományból a megadott kulcshoz tartozó adatokat, azaz az indexet az indexállományból és a hozzá tartozó adatrekordot az adatállományból. Ha nincs ilyen kulcs, Ok False lesz.

TAErase (Var DatSet:DataSet)

Lezárja és fizikailag törli a *DatSet*-hez előzőleg hozzárendelt és megnyitott indexelt adatállományt. Ok-t nem állítja.

TAFlush (Var DatSet:DataSet)

A *DatSet* legutóbbi változtatásait lemezre írja a pufferből. A TACreate, TADelete, TAInsert, TAUpdate és TAWrite rutinok után szokás meghívni.

TAInsert (Var DatSet:DataSet; Var CurRec, Key)

A *DatSet* indexelt adatállományt bővíti a megadott kulcsú rekorddal. Ha ilyen kulcs már szerepelt az adatok között, akkor a bővítés nem történik meg, és Ok False lesz.

TANext (Var DatSet:DataSet; Var CurRec, Key);

Következő elemre lépés a *DatSet* indexelt adatállományban. Ha nincs több adat, akkor Ok False lesz. Ha ilyenkor a TANext rutint mégegyszer meghívjuk, akkor a pozíció az állomány első elemére kerül. A következő adat kulcsa a *Key* változóban, míg a hozzá tartozó adatrekord a *CurRec*-ben lesz megtalálható.

TAOpen (Var DatSet:DataSet; DatFName:FileName; RecordLen:Word; IndexFName:FileName; KeyLen:Integer)

Megnyitja a már létező *DatFName* nevű adatállományt és a hozzátartozó *IndexFName* nevű indexállományt, és hozzárendeli ezt az együttest a *DatSet* logikai állományhoz. *RecordLen* az adatállomány egy rekordjának hossza bájtokban, $18 \leq$ *RecordLen*, *KeyLen* pedig az indexállomány maximális kucshossza. Ezek megadásánál a SizeOf függvényt szokás használni. Duplikált kulcsok nincsenek engedélyezve. Ha sikerült mindkét állomány nyitása, akkor Ok True lesz, egyébként False.

TAPrev (Var DatSet:DataSet; Var CurRec, Key)

Előző elemre lépés a *DatSet* indexelt adatállományban. A legkisebb kulcsú adat előtt Ok False lesz. Ha ilyenkor a TAPrev rutint mégegyszer meghívjuk, akkor a

pozíció az állomány utolsó elemére kerül. Az előző adat kulcsa a *Key* változóban, míg a hozzá tartozó adatrekord a *CurRec*-ben lesz megtalálható.

TARead (Var DatSet:DataSet; Var CurRec, Key; FindExact:Boolean)

Megadott kulcsú rekord olvasása a *DatSet* indexelt állományból. A kulcsot *Key* változóban kell megadni. Ha *FindExact* értéke True (ExactMatch), akkor csak a megadottal pontosan egyező kulcsú adatot olvassa be, False (PartialMatch) esetén *Key* egy generikus kulcs, és az első olyan, vagy annál nagyobb kulcsú rekordot olvassa be. *CurRec* tartalmazza a kért rekordot. Ha sikerült a beolvasás, Ok True lesz, egyébként False.

TAReset (Var DatSet:DataSet)

Pozícionálás a *DatSet* indexelt adatállomány elejére. Ezt az utasítást kell kiadni, ha az indexelt adatállományt elölről, szekvenciálisan akarjuk feldolgozni akár kulcsok szerint növekvő, akár csökkenő sorrendben (pl. rendezett lista készítése esetén). Az ezt követő TANext az indexelt adatállomány első, míg TAPrev az utolsó elemére áll.

TAUpdate (Var DatSet:DataSet; Var CurRec, Key)

A *DatSet* indexelt adatállomány *Key* kulcsú elemét felülírja *CurRec* tartalmával. Ha a megadott kulcsú elem nem létezik, akkor felülírás nem történik, és Ok False lesz. Ügyeljünk arra, hogy ha a rekordban a kulcsnak egy biztonsági másolatát tároljuk, akkor azt ne változtassuk meg!

TAWrite (Var DatSet:DataSet; Var CurRec, Key)

A *DatSet* indexelt adatállományba felírja a megadott kulcsú rekordot. Ha ilyen kulcs már szerepelt az adatok között, akkor azt felülírja, ha eddig még nem létezett, akkor bővíti az állományt.

Eljárások, függvények

SortEOS : Boolean;

A függvényt a TurboSort függvény Output eljárásából lehet meghívni. True értéket ad vissza, ha nincs több rendezett adat (SortReturn-el az összeset visszaolvastuk).

SortRelease (Var ReleaseRecord)

Az eljárást a TurboSort függvény Input eljárásából lehet meghívni. Az eljárással adatokat küldhetünk rendezésre. *ReleaseRecord* tartalmazza a rendezésre bocsátandó adatot. Minthogy *ReleaseRecord* típus nélküli paraméter, a programozó feladata megállapítani, hogy az átadott adat helyes-e. A rendezendő adat *ReleaseRecord* címétől kezdve annyi bájt lesz, amennyit a TurboSort első paramétereként megadtunk.

SortReturn (Var ReturnRecord)

Az eljárást a TurboSort függvény Output eljárásából lehet meghívni. Az eljárással a rendezett adatokból olvashatunk szekvenciálisan. A beolvasott adatot *ReturnRecord* fogja tartalmazni. Minthogy *ReturnRecord* típus nélküli paraméter, a programozó feladata a megfelelő nagyságú memóriaterületet biztosítani. A beolvasott adat *ReturnRecord* címétől kezdve annyi bájt lesz, amennyit a TurboSort első paramétereként megadtunk.

TurboSort (ItemLength: Integer; InpPtr, LessPtr, OutPtr: ProcPtr) : Integer;

A függvény hívásával egy QuickSort rendezés hajtódik végre a megadott paraméterektől függően. *ItemLength* a rendezendő adategység mérete bájtokban, *InpPtr* a rendezés Input eljárásának címe, *LessPtr* a rendezési függvény címe, *OutPtr* pedig a rendezés Output eljárásának címe. ProcPtr típust a Sort egység deklarálja (=Pointer). TurboSort visszaadott értéke 0, ha a rendezés rendben lezajlott, egyébként az a hiba kódját tartalmazza.

LSortEOS : Boolean;

A függvényt az LTurboSort függvény Output eljárásából lehet meghívni. True értéket ad vissza, ha nincs több rendezett adat (LSortReturn-el az összeset visszaolvastuk).

LSortRelease (Var ReleaseRecord)

Az eljárást az LTurboSort függvény Input eljárásából lehet meghívni. Az eljárással adatokat küldhetünk rendezésre. A *ReleaseRecord* tartalmazza a rendezésre bocsátandó adatot. Minthogy *ReleaseRecord* típus nélküli paraméter, a programozó feladata megállapítani, hogy az átadott adat helyes-e. A rendezendő adat *ReleaseRecord* címétől kezdve annyi bájt lesz, amennyit az LTurboSort első paramétereként megadtunk.

LSortReturn (Var ReturnRecord)

Az eljárást az LTurboSort függvény Output eljárásából lehet meghívni. Az eljárással a rendezett adatokból olvashatunk szekvenciálisan. A beolvasott adatot *Return-*

Record fogja tartalmazni. Minthogy *ReturnRecord* típus nélküli paraméter, a programozó feladata a megfelelő nagyságú memóriaterületet biztosítani. A beolvasott adat *ReturnRecord* címétől kezdve annyi bájt lesz, amennyit az LTurboSort első paramétereként megadtunk.

LTurboSort (ItemLength: Integer; InpPtr, LessPtr, OutPtr: ProcPtr) : Integer;

A függvény hívásával egy QuickSort rendezés hajtódik végre a megadott paraméterektől függően. *ItemLength* a rendezendő adategység mérete bájtokban, *InpPtr* a rendezés Input eljárásának címe, *LessPtr* a rendezési függvény címe, *OutPtr* pedig a rendezés Output eljárásának címe. ProcPtr típust az LSort egység deklarálja (=Pointer). LTurboSort visszaadott értéke 0, ha a rendezés rendben lezajlott, egyébként az a hiba kódját tartalmazza.

A TurboSort illetve LTurboSort által visszaadott értékek:

- **0** A rendezés hiba nélkül lezajlott
- **3** Nincs elég memória a rendezés számára. A minimális munkaterület az adatméret háromszorosa.
- **8** Rossz adatméret. 2 és 32767 közé kell esnie.
- **9** Sort egység használatánál 32767-nél több az adatok száma. Használja az LSort egységet!
- **10** Íráshiba rendezés közben. Betelt a lemez, vagy hibás.
- **11** Lemezolvasási hiba. Valószínűleg hibás a lemez.
- **12** A rendezőprogram nem tudott munkaállományt létrehozni. Tele a lemez, vagy lockolva van.

Direktívák
DB
DW
DD

Szimbólumok
@Code
@Data
@Result

Prefixek
LOCK
REP
REPE / REPZ
REPNE / REPNZ

SEGCS
SEGDS
SEGES
SEGSS

Assembly utasítások

A jelzőbiteknél használt jelölések:

O	túlcsorulás (overflow)	**T**	lépésenként fut (trap)	**A**	fél átvitel (aux carry)
D	irányjelző (direction)	**S**	előjel (sign)	**P**	paritás (parity)
I	megszakítás (interrupt)	**Z**	nulla (zero)	**C**	átvitel (carry)

definiálatlan **változik** ↓0 lesz ↑1 lesz

AAA

O S Z **A** *P* **C**

ASCII korrekció összeadás után. Csak olyan nem pakolt BCD számokat összeadó ADD utasítás után használjuk, mely az AL regiszterben hozza létre az eredményt. Ha a megelőző összeadás eredménye egy 9-nél nagyobb szám, az AH regiszter ér-

téke növekszik, az A és C jelzőbitek bebillennek; egyébként AH változatlan marad, a két jelzőbit pedig törlődik. Az AL regiszter felső négy bitje nulla lesz.

AAD

O **S Z** *A* **P** *C*

ASCII korrekció osztás előtt. Kétszámjegyű nem pakolt BCD szám korrigálását végzi osztás előtt úgy, hogy az AL-ben lévő alsó, és az AH-ban lévő felső helyértéket binárissá alakítva (AL + AH*10) elhelyezi az AL-ben, majd AH-t nullázza.

AAM

O **S Z** *A* **P** *C*

ASCII korrekció szorzás után. Két nem pakolt BCD számot összeszorzó, az eredményt az AX regiszterben hagyó MUL utasítás után használjuk csak. Az utasítás az AL-ben lévő (száznál biztosan kisebb) bináris számot úgy pakolja szét, hogy a magasabb helyértékű az AH-ba, az alacsonyabb az AL-be kerül (AL ⇐ AL MOD 10, AH ⇐ AL / 10).

AAS

O S Z **A** *P* **C**

ASCII korrekció kivonás után. Olyan nem pakolt BCD számokat egymásból kivonó SUB utasítás után használatos, mely az eredményt az AL regiszterbe teszi. Ha az AL-ben lévő érték 9-nél nagyobb, AH-t csökkenti, az A és C jelzőbitek bebillennek; egyébként AH változatlan marad és a jelzőbitek törlődnek. AL felső négy bitje az utasítás után nulla lesz.

ADC

O S Z A P C

Összeadás átvitellel. Összeadja a két operandust, és az eredményt növeli eggyel, ha a C jelzőbit 1, majd az eredményt elhelyezi az elsőben. Tipikusan több bájtos/szavas összeadások láncolatában használjuk.

ADD

O S Z A P C

Összeadás. Összeadja a két operandust, majd az eredményt az elsőben elhelyezi.

AND

O↓ S Z *A* P C↓

ÉS kapcsolat. A két operandust bitről bitre ÉS kapcsolatba hozza, majd az eredményt leteszi az elsőbe.

BOUND[286]

Tömb-index ellenőrzése. Az utasítás ellenőrzi, hogy az első operandusként megadott regiszter értéke (index) a második operandussal címzett két egymást követő szó által meghatározott intervallumban van-e. Ha nem, 5-ös megszakítás (INT 5) generálódik.

CALL

Szubrutinhívás. Az operandusként megadott szubrutinra ugrik a processzor, ahonnan RET utasítással lehet visszatérni. A szubrutin egy általunk deklarált Pascal eljárás vagy függvény is lehet.

CBW

Bájt konvertálása szóvá. Az AL regiszterben lévő előjeles értéket 16 bitesre kiterjesztve elhelyezi az AX regiszterben (AH minden bitje felveszi AL legfelső bitjének értékét).

CLC

C↓

Átvitel jelzőbit törlése. Törli az átvitelt jelző (C) bitet; más jelzőbitekre és regiszterekre nincs hatással. Lásd STC, CMC.

CLD D↓

Irány jelzőbit törlése. Törli a string-műveletek irányát meghatározó (D) jelzőbitet. A következő CMPSx, INSx, LODSx, MOVSx, OUTSx, SCASx és STOSx utasításokban az index (SI és/ vagy DI) regiszter(ek) értéke növekszik majd. Lásd STD.

CLI I↓

A megszakítás (I) jelzőbit törlése. Az utasítás végrehajtása után a megszakításokat nem szolgálja ki a processzor, míg a bit ismét 1 nem lesz. Lásd STI.

CMC C

Az átvitel (C) jelzőbit negálása. Az utasítás megfordítja az átvitelt jelző (C) bit értékét, más bitekre és regiszterekre nincs hatással. Lásd CLC, STC.

CMP O S Z A P C

Összehasonlítás. Hasonlóan a SUB utasításhoz, kivonja az első operandusból a másodikat, de csak a jelzőbiteket állítja be, az operandusok változatlanok maradnak.

Viszony	*Átvitel bit (C)*	*Zéró bit (Z)*
Op1 < Op2	1	0
Op1 = Op2	0	1
Op1 > Op2	0	0

CMPSB, CMPSW O S Z A P C

Assembly string-ek összehasonlítása. Az utasítás összehasonlítja az ES:DI és a DS:SI által mutatott két bájtot (CMPSB) illetve szót (CMPSW). Az összehasonlítás során a jelzőbitek beállnak (lásd CMP utasítás), majd az irányt meghatározó

(D) jelző bit állapotától függően az indexregiszterek értéke növekszik (D=0) vagy csökken (D=1) eggyel (CMPSB) illetve kettővel (CMPSW). A címzett bájtok/ szavak változatlanok maradnak. Az utasítás előtt használható a REPx prefix, ekkor az iterációk számát CX (és REPZ, REPE, REPNZ, REPNE mellett a Z jelzőbit) határozza meg.

DAA

O **S Z A P C**

Összeadás utáni pakolt BCD korrekció. Olyan pakolt BCD számokat összeadó ADD utasítás után használjuk, mely az AL regiszterben helyezte el a szintén pakolt kétszámjegyű BCD eredményt. Az utasítás AL-t úgy korrigálja, hogy az helyesen tartalmazza összeadás után a pakolt BCD számokat.

DAS

O **S Z A P C**

Kivonás utáni pakolt BCD korrekció. Olyan kivonás után használjuk, mely az AL regiszterben hozza létre a kétszámjegyű pakolt BCD értéket. Az utasítás AL-t úgy korrigálja, hogy az helyesen tartalmazza kivonás után a pakolt BCD számokat.

DEC

O S Z A P

Csökkentés eggyel. Az operandusként megadott regiszter vagy memóriarekesz értékét csökkenti eggyel. Ez az utasítás nincs hatással az átvitel (C) jelzőbitre – ha az alulcsordulást érzékelni szeretnénk, használjuk a SUB *x*,1 utasítást.

DIV

O S Z A P C

Előjel nélküli osztás. Az AX (a hányados az AL-be, a maradék az AH-ba kerül majd) vagy DX:AX (a hányados az AX-be, a maradék a DX-be kerül majd) regisz-

tert osztja az operandusként megadott regiszter vagy memóriarekesz értékével. Konstanssal osztani nem lehet.

ENTER[286]

Belépés magasszintű szubrutinba. Első paramétere a veremben lefoglalandó bájtok számát határozza meg, a második a rutin beágyazási mélységét (0..31) a magasszintű nyelven írt programban. Ez utóbbival tulajdonképpen azt adjuk meg, hogy hány keretmutatót kell az illető rutin lokális területére átmásolni az előzőből. Az utasítás után a BP regiszterrel és negatív eltolással ([BP–*x*]) címezhetjük a változókat. Ha a második paraméter nulla, az utasítás BP-t leteszi a verembe, egyenlővé teszi SP-vel, végül kivonja az első paramétert SP-ből. A szubrutin végén a RET utasítás előtt a LEAVE-vel lehet helyreállítani a mutatókat.

HLT

A processzor leállítása a következő megszakításig vagy reset-ig. Az utasítás a processzort HALT állapotba viszi, amelyből csak megszakítással, nem maszkolt megszakítással (NMI) vagy reset-tel lehet kibillenteni. IBM PC környezetben a CLI utáni HLT a gép lefagyását eredményezi, mert az NMI nem aktivizálható kívülről.

IDIV

O S Z A P C

Előjeles osztás. Az AX (a hányados az AL-be, a maradék az AH-ba kerül majd) vagy DX:AX (a hányados az AX-be, a maradék a DX-be kerül majd) regisztert osztja az operandusként megadott regiszter vagy memóriarekesz értékével. Konstanssal osztani nem lehet. Ha a hányados nem fér bele AL-be illetve AX-be, vagy

az osztó értéke nulla, automatikusan nullás megszakítás generálódik (INT 0). A maradék előjele megegyezik majd az osztandó előjelével.

IMUL

O *S Z A P* C

Előjeles szorzás. Az AL (eredmény AX-ben lesz) illetve AX (eredmény AX:DX-ben lesz) regisztert megszorozza az operandussal, ami 8 vagy 16 bites regiszter vagy memóriarekesz lehet. 80286-on három paraméterrel is használható: az elsőben (16 bites regiszter) elhelyezi a második (16 bites regiszter vagy memóriarekesz) és harmadik (8 vagy 16 bites konstans) szorzatát.

Törli az átvitel (C) és túlcsordulás (O) jelzőbiteket, ha

- AL szorzása után AH megegyezik AL előjeles kiterjesztésével;
- AX szorzása után DX megegyezik AX előjeles kiterjesztésével;
- 80286-on három paraméterrel használjuk és az eredmény elfér a célregiszterben, tehát értéke -32768 és 32767 között van.

IN

Olvasás B/K port-ról. Az első paraméterként meghatározott AL vagy AX regiszterbe beolvas egy bájtot/szót a második paraméterként megadott port-ról. A második paraméter egy 8 bites konstans vagy a DX regiszter lehet (a 256..65535 közötti portokat csak a DX segítségével tudjuk elérni).

INC

O S Z A P

Növelés eggyel. Az operandusként megadott regiszter vagy memóriarekesz értékét növeli eggyel. Ez az utasítás nincs hatással az átvitel (C) jelzőbitre – ha a túlcsordulást érzékelni szeretnénk, használjuk az ADD *x*,1 utasítást.

INSB[286], INSW[286]

Assembly string-ek beolvasása portról. A DX regiszterrel címzett portról beolvas egy bájtot .(INSB) illetve szót (INSW) az ES:DI által megadott címre, majd az irányt meghatározó (D) jelzőbittől függően DI regisztert növeli (D = 0) vagy csökkenti (D = 1) eggyel INSB, kettővel INSW esetén.
Az utasítás előtt használható a REPx prefix, ekkor az iterációk számát CX határozza meg.

INT, INTO

I↓ T↓

Szoftver megszakítás generálása. Az INT*x* utasítás szoftver úton adja át a vezérlést a megszakítást kiszolgáló rutinnak. *x* 0 és 255 között lehet – jelezve, hogy mely megszakítási vektor címzi a kiszolgáló rutint (a megszakítási vektorok – 32 bites távoli címek – a memória alján helyezkednek el a megszakítási vektor táblában).
Az INTO utasítás gyakorlailag INT4-nek felel meg, mely akkor hajtódik végre, ha a túlcsordulást jelző (O) bit értéke 1.
A megszakítás hívása a fentieken kívül abban is különbözik a normál szubrutin-hívástól, hogy elmenti a jelzőbiteket, és a kiszolgáló rutinból IRET utasítással kell visszatérni.

IRET

O D I T S Z A P C

Visszatérés megszakítást kiszolgáló rutinból. Kiemeli a veremből a visszatérési címet és a jelzőbiteket, majd a rutint hívó INT*x* utáni utasításra viszi a processzort.

Jx

Ugrás feltétel függvényében. Az utasítás átadja a vezérlést az operandusként jelzett címkére, ha az *x* feltétel teljesül. A címkével jelzett utasításnak ugyanebben a szegmensben kell lennie, a feltételes ugrást követő utasítás első bájtjától -128..127 bájtra.

Utasítás	*Ugrás, ha*	*Utasítás*	*Ugrás, ha*
JA, JNBE *címke*	>	**JG, JNLE** *címke*	előjelesen >
JAE, JNB *címke*	≥	**JGE, JNL** *címke*	előjelesen ≥
JB, JNAE *címke*	<	**JL, JNGE** *címke*	előjelesen <
JBE, JNA *címke*	≤	**JLE, JNG** *címke*	előjelesen ≤
JO/JNO *címke*	az O jelzőbit 1/0	**JS/JNS** *címke*	az S jelzőbit 1/0
JZ/JNZ *címke*	a Z jelzőbit 1/0	**JP /JNP***címke*	a P jelzőbit 1/0
JC/JNC *címke*	a C jelzőbit 1/0	**JE/JNE** *címke*	=/≠
JCXZ *címke*	a CX regiszter 0	**JPE/JPO** *címke*	a paritás páros/páratlan

A JCXZ utasítás abban különbözik a többitől, hogy a jelzőbitek helyett a CX regisztert vizsgálja.

Tipikusan arra használjuk, hogy a következő iterációt átugorjuk, ha CX nulla. Ha ezt nem tennénk meg, CX=0 mellett a ciklus 65536-szor futna le (a ciklusokat szervező LOOPx utasítások előbb csökkentik CX-et, majd ezután vizsgálják!).

JMP

Feltétel nélküli vezérlésátadás. Távoli (szegmensek közötti) ugrásra is alkalmas. Az utasítás és az operandus közé beírhatjuk a NEAR PTR és FAR PTR jelzéseket is.

LAHF

Jelzőbitek betöltése az AH regiszterbe. Az utasítással a jelzőbiteket tartalmazó szavas regiszter alsó bájtját tölthetjük át az AH regiszterbe. AH jelzőbit-tartalma az utasítás után: **SZxAxPxC**.

LEA

Effektív (eltolási) cím betöltése. Az első paraméterként megadott 16 bites regiszterbe betölti a második (memóriarekesz) eltolási címét.

LEAVE^{286}

Kilépés magasszintű szubrutinból. Az utasítás az ENTER^{286} párja. A BP regiszter SP-be történő átmásolásával felszabadítja a szubrutin lokális változóihoz rendelt verem területet, majd visszatölti az elmentett BP-tartalmat. A paraméterek területe a RET *x* utasítással szabadítható fel visszatéréskor.

LDS, LES

Távoli mutató betöltése regiszterekbe. Az LDS utasítás betölti a DS és az első paraméterként megadott 16 bites regiszterbe a második – duplaszavas – paramétert, mely egy memóriarekesz távoli mutatója. A LES utasítás ugyanez, de az ES-be

tölti a szegmenscímet. Tipikusan mutatók és szubrutinok cím-paramétereinek kezelésére használatos.

LOCK

Memóriahasználat tiltása. Többprocesszoros környezetben használatos prefix, mely-lyel az utasítás idejére a megosztott memória más processzor által történő használatát letilthatjuk.

LODSB, LODSW

Assembly string-elem betöltése. A DS:SI regiszterekkel címzett memóriarekesz tartalmát betölti az AL (LODSB) illetve AX (LODSW) regiszterbe, majd az irányt megadó jelzőbit (D) függvényében növeli (D=0) vagy csökkenti (D=1) SI-t eggyel (LODSB) / kettővel (LODSW).
Az utasítás iterálható a REP prefix segítségével, bár inkább LOOP ciklusban használjuk, hiszen AL/AX értékét fel kell dolgoznunk a következő betöltés előtt.

LOOP*x*

Ciklus. Az utasítással hátultesztelő ciklust szervezhetünk. Csökkenti CX értékét, majd ha az nem nulla és az esetleges feltétel teljesül, a címke operandussal meghatározott címre ugrik. A címkével jelzett utasításnak ugyanebben a szegmensben kell lennie, a LOOP-ot követő utasítás első bájtjától -128..127 bájtra. A jelzőbiteket nem állítja.

LOOP *címke*	*ugrás, ha* **CX** <> 0
LOOPE, LOOPZ *címke*	*ugrás, ha* **CX** <> 0 *és* **Z**=1
LOOPNE, LOOPNZ *címke*	*ugrás, ha* **CX** <> 0 *és* **Z**=0

MOV

Értékadás. Az első operandusba átmásolja a második értékét. A jelzőbitek változatlanok maradnak.

MOVSB, MOVSW

Assembly string-ek másolása. A DS:[SI] címről átmásolja a bájtos (MOVSB) vagy szavas (MOVSW) értéket az ES:[DI] címre, majd az irányt meghatározó (D) jelzőbit állapotától függően mindkét index regisztert növeli (D=0) vagy csökkenti (D=1) eggyel (MOVSB) illetve kettővel (MOVSW). Ebben az utasításban csak a forrás szegmens bírálható felül.
Az utasítás előtt szerepelhet REP prefix, ha több bájtot/szót kívánunk mozgatni.

MUL

O *S Z A P* C

Előjel nélküli szorzás. AL-t illetve AX-et megszorozza az operandussal, az eredmény az AX illetve DX:AX regiszterben jön létre.

NEG

O S Z A P C

Kettes komplemens. Az 8 vagy 16 bites regiszter vagy memóriarekesz operandus kettes komplensét (0-*x*) helyezi vissza az operandusba. Ha az operandus nulla, az átvitel (C) jelzőbit törlődik, egyébként 1 lesz.

NOP

Üres utasítás

NOT

Egyes komplemens. Az 8 vagy 16 bites regiszter vagy memóriarekesz operandus egyes komplensét (bitenkénti negáltját) helyezi vissza az operandusba.

OR

O↓ S Z *A* P C↓

Bitenkénti VAGY. Bitenként VAGY kapcsolatba hozza a két operandust, majd az eredményt az elsőben elhelyezi.

OUT

Írás B/K port-ra. Az AL vagy AX regiszter tartalmát kiküldi a konstanssal vagy a DX regiszterrel jelzett portra.

OUTSB[286], OUTSW[286]

Assembly string kiküldése portra. A DX regiszterben meghatározott portra kiküldi a DS:[SI]-vel címzett bájtot (OUTSB) vagy szót (OUTSW), majd az irányjelző bit (D) függvényében növeli (D=0) illetve csökkenti (D=1) SI-t eggyel (OUTSB) vagy kettővel (OUTSW).
Az utasítás iterálható a REP prefix segítségével, az iterációk számát CX határozza meg ilyenkor.

POP

Szó kiemelése a veremből. Kiemeli a veremből az SS:SP által mutatott szót, elhelyezi a megadott 16 bites regiszterben vagy memóriarekeszben, majd növeli SP-t kettővel. CS és IP nem lehet operandusa az utasításnak.

POPA[286]

Az általános regiszterek visszatöltése a veremből. Az utasítás a PUSHA fordítottja: visszatölti az általános célú regisztereket a veremből. A sorrend: DI, SI, BP, *SP*, BX, DX, CX, AX. Az utasítás csak kiemeli az elmentéskor letett SP tartalmat, de SP-t nem módosítja.

POPF

O D I T S Z A P C

Jelzőbitek visszatöltése a veremből. Az SS:SP által mutatott szót betölti a jelzőbiteket tartalmazó regiszterbe, majd növeli SP-t kettővel. Lásd PUSHF.

PUSH

Szó letétele a verembe. Csökkenti SP-t kettővel, majd az operandusként megadott 16 bites regiszter vagy memóriarekesz tartalmát elhelyezi az SS:SP címre. 80286 esetén 16 bites konstans is letehető a verembe.

PUSHA[286]

Az általános regiszterek letétele a verembe. Az utasítás leteszi a verembe az általános célú regisztereket a következő sorrendben: AX, CX, DX, BX, SP, BP, SI, DI. A POPA utasítás az itt elmentett SP értéket csak kiemelni fogja, nem módosítja majd SP-t.

PUSHF

A jelzőbitek letétele a verembe. Csökkenti SP-t kettővel, majd az SS:SP címre leteszi a jezőbiteket tartalmazó regiszter tartalmát. Lásd POPF.

RCL, RCR O C

Forgatás a C jelzőbittel együtt. Az elsőként megadott operandus bitjeit és kilencedik bitként az átvitel (C) jezőbitet forgatják balra (RCL) illetve jobbra (RCR) úgy, hogy a kieső bit a sorozat végére visszakerül. A forgatások számát a második operandus adja, ami 1 vagy a CL regiszter lehet. 286-ostól a második operandus 8 bites konstans is lehet.

Ha 1-es konstanssal forgatunk balra (RCL) és forgatás előtt a bájt/szó legfelső bitje megegyezik az átvitel (C) jelzőbittel, a túlcsordulás (O) jelzőbit nulla lesz, egyébként 1. RCR esetén ugyanez történik, de a vizsgálat a forgatás utáni állapotra vonatkozik.

REP, REPE, REPZ, REPNE, REPNZ Z

Assembly string-műveletet iteráló prefixek. Ezek a prefixek a CMPSx, INS*x*, LODS*x*, MOVS*x*, OUTSx, SCASx és STOS*x* assembly string-utasítások elé írhatók. Az iterációk számát CX és a jelzőbitek határozzák meg (ha CX nulla, az utasítás 65536-szor fut le!). A REPE és REPZ prefixek iterálnak, ha CX nem nulla és a zéró (Z) jelzőbit 1 (ismétlés, ha egyezik); a REPNE és REPNZ ha CX nem nulla és a jelzőbit 0 (ismétlés, ha nem egyezik).

RET

Visszatérés szubrutinból. Az utasítás kiemeli a veremből a visszatérési címet a CS és IP regiszterekbe, így a következő végrehajtott utasítás a szubrutint hívó CALL utáni lesz.

Adhatunk konstans operandust is az utasításnak, ezzel a szubrutin paramétereinek területét is felszabadíthatjuk a veremben.

ROL, ROR

O C

Forgatás. Az elsőként megadott operandus bitjeit forgatják balra (ROL) illetve jobbra (ROR) úgy, hogy a kieső bit a sorozat végére visszakerül. A forgatások számát a második operandus adja, ami 1 vagy a CL regiszter lehet. 286-ostól a második operandus 8 bites konstans is lehet.

Ha 1-es konstanssal forgatunk balra vagy jobbra és forgatás után a bájt/szó legfelső bitje megegyezik az átvitel (C) jelzőbittel, a túlcsordulás (O) jelzőbit nulla lesz, egyébként 1.

SAHF

S Z A P C

AH áttöltése a jelzőbitek regiszterbe. Az utasítás áttölti AH értékét a jezőbitek regiszter alsó bájtjába. AH-ban a jelzőbitek: **SZxAxPxC**.

SAL, SAR

O S Z *A* P C

Eltolás az átvitel jelzőbitbe is. Az első operandus – 8 vagy 16 bites regiszter vagy memóriarekesz – bitjeit tolják balra (SAL) vagy jobbra (SAR) úgy, hogy a kieső bit az átvitel (C) jelzőbitbe kerül, és 0 lép be az operandus nulladik (SAL) illetve hetedik (SAR) bitjébe. Az eltolások száma a második operandus értékétől függ, mely 31 lehet maximum (ha nagyobb, az utasítás csak az érték alsó 5 bitjét használja fel). A második operandus 1 vagy a CL regiszter lehet. 286-ostól a második operandus 8 bites konstans is lehet.

Ha 1-es konstanssal tolunk el balra (SAL) és eltolás után a bájt/szó legfelső bitje megegyezik az átvitel (C) jelzőbittel, a túlcsordulás (O) jelzőbit nulla lesz; ha különböznek, a jelzőbit 1 lesz. SAR esetében a túlcsorduláa (O) jelzőbit mindig törlődik, ha a második paraméter 1-es állandó.

SHL, SHR

O S Z A P C

Eltolás. Az első operandus – 8 vagy 16 bites regiszter vagy memóriarekesz – bitjeit tolják balra (SHL) vagy jobbra (SHR) úgy, hogy 0 lép be az operandus nulladik (SHL) illetve hetedik (SHR) bitjébe. Az eltolások száma a második operandus értékétől függ, mely 31 lehet maximum (ha nagyobb, az utasítás csak az érték alsó 5 bitjét használja fel). A második operandus 1 vagy a CL regiszter lehet. 286-ostól a második operandus 8 bites konstans is lehet.

Ha 1-es konstanssal tolunk el balra (SHL) és eltolás után a bájt/szó legfelső bitje megegyezik az átvitel (C) jelzőbittel, a túlcsordulás (O) jelzőbit nulla lesz; ha különböznek, a jelzőbit 1 lesz. SHR esetén, ha 1-es konstans a második operandus, a túlcsordulás (O) jelzőbit felveszi az eredeti eltolandó bájt/szó legfelső bitjének értékét.

SBB

O S Z A P C

Előjeles kivonás áthozattal. A második operandushoz hozzáadja az átvitel (C) jelzőbitet, majd az eredményt kivonja az elsőből. A kivonás utáni értéket az első operandusban helyezi el. Ha konstanst vonunk ki szavas regiszterből vagy memóriarekeszből, a konstanst – ha szükséges – előjeles szóvá konvertálja.

SCASB, SCASW

O S Z A P C

Keresés assembly string-ben. Összehasonlítja AL (SCASB) illetve AX (SCASW) regisztert az ES:[DI] memóriarekesszel, majd az irány (D) jelzőbittől függően növeli (D=0) vagy csökkenti (D=1) DI-t eggyel (SCASB) illetve kettővel (SCASW). A jelzőbitek tartalmazzák az összehasonlítás eredményét (lásd CMP utasítás). Az utasításban nem lehet az alapértelmezett ES szegmensregisztert felülbírálni.

Használhatók a REP*x* prefixek, ekkor CX-ben kell lennie a maximális keresési hossznak.

STC

C↑

Az átvitel jelzőbit beállítása. Az átvitel (C) jelzőbitet 1 értékűre állítja. Lásd CLC, CMC.

STD

D↑

Az irány jelzőbit beállítása. Az irány (D) jelzőbitet 1 értékűre állítja. A következő assembly string-műveletek során az indexregiszterek csökkenni fognak. Lásd CLD.

STI

I↑

A megszakítás jelzőbit beállítása. A megszakításokat engedélyező/tiltó bitet beállítja, így a további megszakítások engedélyezettek. Lásd CLI.

STOSB, STOSW

Assembly string-elem letétele a memóriába. Az AL (STOSB) vagy AX (STOSW) regiszter tartalmát leteszi az ES:[DI] memóriarekeszbe, majd az irány (D) jelzőbittől függően növeli (D=0) vagy csökkenti (D=1) DI-t eggyel (STOSB) illetve kettővel (STOSW). Használható a REP prefix, ekkor CX-ben kell lennie a feltöltendő hossznak.

SUB

O S Z A P C

Előjeles kivonás. Kivonja a második operandust az elsőből, majd az eredményt elhelyezi az elsőben. Ha konstanst vonunk ki szavas regiszterből vagy memóriarekeszből, a konstanst – ha szükséges – előjeles szóvá konvertálja.

TEST

O↓ S Z *A* P C↓

Logikai összehasonlítás. Bitenkénti ÉS kapcsolatba hozza a két operandust, és a jelzőbiteket az eredménynek megfelelően beállítja. Az operandusok nem változnak meg. Az első operandus 8 vagy 16 bites regiszter vagy memóriarekesz lehet, a második hasonló bitszámú regiszter vagy konstans.

WAIT

Várakozás. Az utasítás hatására a processzor addig várakozik, amíg a BUSY vonal inaktív (tehát logikai magas) nem lesz. Ezt a vonalat a 80x87-es vezérli.

XCHG

Csere. Megcseréli a két – 8 vagy 16 bites regiszter vagy memórarekesz – operandus tartalmát (természetesen csak az egyik lehet memóriarekesz). Függetlenül attól hogy szerepel-e az utasításban a LOCK prefix, a csere idejére tiltja a memóriahozzáférést a többi processzor számára.

XLAT

Konvertálás táblázat alapján. A DS:BX által mutatott bájtos tömb AL regiszterrel meghatározott elemét betölti az AL regiszterbe.

XOR

O↓ S Z *A* P C↓

Bitenkénti KIZÁRÓ VAGY. A két operandust KIZÁRÓ VAGY kapcsolatba hozza, majd az eredményt az elsőben elhelyezi.

Fordítási hibakódok

1 Elfogyott a memória
- ! a memóriában lévő extra programot szüntesse meg (talán kétszer hívta a keretrendszert ?)
- ! a programot fordítsa lemezre (C/Destination ⇨DISK)
- ! a programot szerkessze lemezen (O/L/Link buffer ⇨DISK)

2 A fordító azonosítót vár

3 Az azonosító nincs deklarálva

4 Az azonosító már deklarálva van ebben a blokkban

5 Szintaktikai hiba: illegális karakter

6 Hiba a valós típusú konstansban

7 Hiba az egész típusú konstansban
- ! nagy valós egészeket tizedessel kell megadni (pl. 65540.0)

8 A karakterlánc konstans túlnyúlik a soron
- ? esetleg hiányzik a záró aposztróf

9 Túl sok az egymásba ágyazott (Include) forrásállomány
- ! maximálisan 15 lehet

10 A forrásállomány nem várt vége
- ? END. előtt vége van a programnak (pl. Begin End párhiba)
- ? egy Include állomány nem teljes
- ? esetleg nem zárt le egy megjegyzést

11 Túl hosszú a sor
- ! maximum 126 karakteres lehet

12 A fordító típusazonosítót vár

13 Túl sok a nyitott állomány

? az egyszerre megnyitható állományok maximális száma a DOS CONFIG.SYS állományában adható meg

14 Érvénytelen állománynév vagy útvonalleírás

15 Az állomány nincs az aktuális/keresési katalógusban

16 Tele van a lemez

! töröljön ki állományokat

! használjon új lemezt

17 Érvénytelen fordítási direktíva

? szintaktikai hiba

? globális direktívát talált a fordító a programblokkban

18 Túl sok az állomány

! Csökkentse az egymásba ágyazott állományok számát, nevük hosszúságát

19 Definiálatlan típus Pointer deklarációban

20 A fordító változó-azonosítót vár

21 Rossz típusdeklaráció

22 Túl nagy a struktúrált (összetett) típus

! maximum 65520 bájt lehet a mérete

23 A halmaz alaptípusa hibás

! az elemek rendszámai 0 és 255 között lehetnek

24 Állomány komponense nem lehet állomány illetve objektum

25 Érvénytelen karakterlánc-hossz

! a String deklarált maximális hossza 1 és 255 közé eshet

26 Típusegyeztetési hiba
- ? értékadás során a változó és a kifejezés típusa nem kompatibilis
- ? a formális és az aktuális paraméterek típusa nem kompatibilis
- ? az indextípus és az indexkifejezés nem kompatibilis
- ? a kifejezés operandusai nem kompatibilisek

27 Az intervallum alaptípusa nem sorszámozott

28 Az intervallum alsó határa nagyobb, mint a felső

29 A fordító sorszámozott típust vár

30 A fordító egész konstanst vár

31 A fordító konstanst vár

32 A fordító egész vagy valós konstanst vár

33 A fordító mutató típusazonosítót vár

34 Érvénytelen függvénytípus
- ! a függvény típusa csak egyszerű, String és Pointer lehet

35 A fordító címkét vár

36 A fordító a BEGIN kulcsszót várja

37 A fordító az END kulcsszót várja

38 A fordító egész típusú kifejezést vár

39 A fordító sorszámozott típusú kifejezést vár

40 A fordító Boolean típusú kifejezést vár

41 A műveletnek nem lehetnek ilyen operandusai

42 Szintaktikai hiba a kifejezésben

43 Hibás értékadás
- ! állomány- és típus nélküli paraméter nem szerepelhet értékadó utasítás bal oldalán

! függvény-azonosítónak csak a függvény blokkjában lehet értéket adni

44 A fordító a rekordváltozó egy mező-azonosítóját várja

45 A tárgykódot tartalmazó (object) állomány 64K-nál nagyobb

46 Hibás EXTERNAL definíció

? az EXTERNAL eljárás definíciójának nem felel meg PUBLIC definíció a tárgykódot tartalmazó állomány(ok)ban

47 A tárgykódot tartalmazó állomány érvénytelen rekordot tartalmaz

48 Túl nagy a kódszegmens

! a program és az egységek kódja egyenként maximum 65520 bájt lehet

49 Túl nagy az adatszegmens

! a program és az egységek globális változóinak együttes mérete maximum 65520 bájt lehet

50 A fordító a DO kulcsszót várja

51 Érvénytelen PUBLIC deklaráció

52 Érvénytelen EXTRN deklaráció

53 Túl sok az EXTRN deklaráció

! maximum 256 lehet

54 A fordító az OF kulcsszót várja

55 A fordító az INTERFACE kulcsszót várja

56 Érvénytelen memóriacím

? a tárgykódot tartalmazó állományban érvénytelen memóriacímre való hivatkozás szerepel

57 A fordító a THEN kulcsszót várja

58 A fordító a TO vagy DOWNTO kulcsszót várja

59 A deklaráció hiányzik

? FORWARD direktívával deklarált szubrutin nem található
? az egység Interface részében deklarált szubrutin nincs kifejtve az Implementation részben

60 Túl sok a programban vagy egységben deklarált szubrutin
! maximum 512 lehet

61 Hibás típuskonverzió
! itt csak változó típusát lehet konvertálni
! a változó és a céltípus méretének meg kell egyeznie

62 Osztás nullával

63 Érvénytelen állománytípus

64 Ilyen típusú változó nem szerepelhet Read/Write eljárásban
! szöveges állományból beolvasni csak karakter, egész, valós és karakterlánc típusú változóba lehet
! szöveges állományba kiírni karakter, egész, valós, karakterlánc és Boolean típusú kifejezéseket lehet

65 A fordító mutató típusú változót vár

66 A fordító karakterlánc típusú változót vár

67 A fordító karakterlánc típusú kifejezést vár

68 Az egységek egymásra hivatkozása nem megengedett

69 Egységnév ütközés
? a .TPU állományban található egységnév nem egyezik a USES kulcsszó után megadottal
! adja ugyanazt a nevet az egységnek és az őt tartalmazó forrásállománynak

70 Az egység módosult

? a program vagy egység által használt egy vagy több egységben változtatás történt a legutóbbi fordítás óta
! fordítsa le programját a C/Make vagy C/Build menüponttal

71 Ilyen egységnév már szerepelt a USES kulcsszó után

72 A .TPU állomány hibás
! győződjön meg róla, hogy az valóban .TPU állomány-e

73 A fordító az IMPLEMENTATION kulcsszót várja

74 A konstans típusa nem egyezik a CASE szelektorának típusával

75 A fordító rekord típusú változót vár

76 Intervallumon kívüli konstans
? az állandó értéke kiesik a tartományból értékadásnál, paraméterátadásnál vagy tömbindexelésénél

77 A fordító állomány típusú változót vár

78 A fordító mutató típusú kifejezést vár

79 A fordító egész vagy valós típusú kifejezést vár

80 A címke nincs a blokkban
! GOTO utasítással nem lehet kiugrani az aktuális blokkból

81 Ez a címke már létezik

82 A címke nincs definiálva

83 Hibás @ művelet
! a @ művelet operandusa csak változó, típusos konstans, eljárás és függvény azonosítója lehet

84 A fordító a UNIT kulcsszót várja

85 A fordító a ; jelet várja

86 A fordító a : jelet várja

87 A fordító a , jelet várja
88 A fordító a (jelet várja
89 A fordító a) jelet várja
90 A fordító az = jelet várja
91 A fordító a := jelet várja.
92 A fordító a [vagy a (. jelet várja
93 A fordító a] vagy a .) jelet várja
94 A fordító a . jelet várja
95 A fordító a .. jelet várja
96 Túl sok a változó
! a program vagy egység globális változóinak, valamint egy szubrutin lokális változóinak összmérete nem haladhatja meg a 64K-t
97 A FOR utasítás cilusváltozója érvénytelen
? nem egyszerű típusú
? nincs az aktuális szubrutin deklarációs részében deklarálva
98 A fordító egész típusú változót vár
99 Állomány illetve eljárás típus nem megengedett
! Típusos konstans nem lehet állomány illetve eljárás típusú
100 A karakterlánc és a karaktertömb hossza nem egyezik
101 A rekord típusú állandó mezősorrendje nem egyezik a deklarációval
102 A fordító szövegkonstanst vár
103 A fordító egész vagy valós típusú változót vár
104 A fordító sorszámozott típusú változót vár
105 Hiba az INLINE direktíva blokkjában
! változó hivatkozásnál a < operátor nem megengedett

106 A fordító karakter típusú kifejezést vár

107 Túl sok az adatáthelyezés

? a futtatható kód relokációs táblája meghaladja a 64K-t

? a program túl nagy a szerkesztő (linker) számára

! bontsa programját több részre, és hívja az alprogramokat az EXEC eljárással

112 A CASE utasítás konstansa kiesik a megengedett tartományból

! Egész típusú CASE esetében a konstans -32768 és 32767 közé eshet csak

113 Hibás utasítás

! Így nem kezdődhet egy utasítás

114 Az INTERRUPT típusú eljárást nem lehet direkt módon hívni

116 A 80x87 valós típusú változókat $N+ mellett kell fordítani

117 Az adott címnek nem feleltethető meg forrásnyelvű utasítás

118 Az Include direktíva blokkon belül nem megengedett

120 A fordító a NIL kulcsszót várja

121 Érvénytelen minősítés

? nem tömböt indexel

? nem rekord illetve objektum típusú változó mezőjére hivatkozik

? nem mutatóval mutat

122 Érvénytelen változóhivatkozás

123 Túl sok a szimbólum

? A program vagy egység 64K-nál több szimbólumot deklarál ⇨ Fordítsa programját $D- mellett, vagy tördelje kisebb részekre

124 Túl hosszú végrehajtandó rész

! maximum 24K lehet
! írjon eljárásokat!

126 Állomány típusú paraméter csak változó paraméter lehet

127 Túl sok a feltételes szimbólum, vagy túl hosszú a nevük

128 A feltételes direktívának nincs párja

129 Hiányzik a $ENDIF direktíva

130 Az O/C/Conditional defines menüpont szimbólumai helytelenek

! a Turbo Pascal nulla vagy több azonosítót vár szóközzel, vesszővel vagy kettősponttal elválasztva

131 A szubrutinfejek ütköznek

? Az itt deklarált szubrutin feje nem egyezik az INTERFACE részben vagy a FORWARD direktívánál megadottal

132 Kritikus lemezhiba fordítás közben

? például nincs a meghajtóban lemez

133 A kifejezés kiértékelhetetlen

134 A kifejezés helytelen

135 Érvénytelen formátum-megadás

136 Érvénytelen indirekt hivatkozás

137 Struktúrált (összetett) típusú változó itt nem használható

138 A System egység nélkül a kifejezés kiértékelhetetlen

? Esetleg hiányzik a Turbo.Tpl állomány, vagy nincs benne a System egység

139 A szimbólum csak a program futásakor válik elérhetővé

140 Túlcsordulás vagy nullával való osztás valós számításnál

141 Overlay szubrutint tartalmazó program lemezre fordítandó

142 Itt a @ operátor csak eljárás típusú változóval szerepelhet

143 Érvénytelen eljárás vagy függvény hivatkozás

? eljárás hívása kifejezésben

! a szubrutin $F+ mellett fordítandó és nem rendelkezhet INLINE vagy INTERRUPT direktívával, ha eljárás típusú változó kapja meg, mint értéket

144 Az egységet nem lehet overlay struktúrába szervezni

? valószínű, hogy fordítása nem $O+ mellett történt

147 A fordító objektum típust vár

148 Lokális objektum típus nem megengedett

149 A fordító a VIRTUAL kulcsszót várja

150 A fordító metódus azonosítóját várja

151 Konstruktor nem lehet virtuális

152 A fordító konstruktor azonosítóját várja

153 A fordító destruktor azonosítóját várja

154 A FAIL eljárás csak konstruktorból hívható

155 Érvénytelen utasítás-operandus kombináció

? túl kevés vagy túl sok az operandus az illető assembly utasításban

? az operandusok típusa vagy sorrendje helytelen

156 A fordító memória-hivatkozást vár

? esetleg hiányzik a regiszter indirekt címzést jelző [ ] pár

157 Relokálható szimbólumok nem adhatók össze és nem vonhatók ki egymásból

! változók, eljárások, függvények és címkék nem adhatók össze egymással és nem vonhatók ki egymásból assembly utasítás operandusában

158 Érvénytelen regiszter kombináció

! regiszter indirekt címzés csak a következő kombinációkkal használható: [BX], [BP], [SI], [DI], [BX+SI], [BX+DI], [BP+SI], [BP+DI]

⇨ lokális változók regiszter indirekt címzésekor a BP+ prefixet nem kell használni, mert a fordító automatikusan a BP-t használja bázis regiszterként

159 286/287 utasítások nem használhatók

! a 286/287 utasítások csak $G+ direktíva után használhatók

⇨ ezen utasítások sorrakerülése előtt ellenőrizni kell a processzor típusát, hiszen ezeket a 8086-os és a 8088-as nem ismeri

160 Érvénytelen szimbólum-hivatkozás

! standard eljárásokra, standard függvényekre és a speciális Mem, MemW, MemL, Port, PortW tömbökre nem lehet assembly utasításokban hivatkozni

! karakterlánc, lebegőpontos valós és halmaz állandókra nem lehet assembly utasításokban hivatkozni

! inline eljárásokra és függvényekre nem lehet assembly utasításokban hivatkozni

! a függvényen kívül nem lehet a @Result szimbólumra hivatkozni

! közeli ugrással (short JMP) csak címkére lehet ugrani

161 Kódgenerálási hiba

! a megelőző utasítás rész olyan LOOPNE, LOOPE, LOOP vagy JCXZ utasítást tartalmaz, melynek címkéje túl messze van

162 A fordító az ASM kulcsszót várja

DOS hibák

1 Érvénytelen DOS funkció

2 Az állomány nem található
- ? Reset, Append, Rename vagy Erase hívásakor az állományváltozóhoz rendelt fizikai állomány nem létezik, vagy neve érvénytelen

3 Az útvonal/katalógus nem található
- ? Reset, Rewrite, Append, Rename vagy Erase hívásakor az állományváltozóhoz rendelt fizikai állomány katalógusa nem létezik
- ? ChDir, MkDir vagy RmDir hívásakor az útvonal érvénytelen, vagy a katalógus nem létezik

4 Túl sok állomány van egyszerre nyitva
- ? Reset, Rewrite vagy Append hívásakor a nyitott állományok a DOS által megengedett számot túllépnék
- ! állítsa nagyobbra a CONFIG.SYS FILES értékét

5 Az állományhozzáférés tiltott
- ? Reset vagy Append hívásakor a FileMode változó csak írást engedélyez, és az állomány írásvédett, vagy katalógus
- ? Rewrite hívásakor a katalógus betelt, az állomány írásvédett vagy katalógus
- ? Rename hívásakor az új néven már létezik állomány, vagy katalógus
- ? Erase hívásakor az állomány írásvédett, vagy katalógus
- ? MkDir hívásakor a katalógus betelt, a név már létezik, vagy eszközt specifikál a név

? RmDir hívásakor a katalógus nem üres, a név nem katalógust specifikál, vagy a főkatalógust specifikálja
? Read, BlockRead hívásakor az állomány nem nyitott olvasásra
? Write, BlockWrite hívásakor az állomány nem nyitott írásra

6 Érvénytelen kezelőszám (handle)
? az állományváltozó valamilyen oknál fogva megsérült

12 Érvénytelen hozzáférési kód
? Reset vagy Append hívásakor típusos vagy típusnélküli állományra a FileMode változó értéke érvénytelen

15 Érvénytelen meghajtószám
? GetDir hívásakor a meghajtószám érvénytelen

16 Az aktuális katalógus nem szüntethető meg
? RmDir hívásakor a paraméter az aktuális katalógust specifikálja

17 Lemezek között nem lehet átnevezni
? Rename hívásakor az új név más meghajtót specifikál

B/K hibák *($I- esetén az IOResult függvénnyel lekérdezhetők)*

100 Lemezolvasási hiba
? Read hívásakor a típusos állomány rekordmutatója az állomány vége mögé mutat

101 Lemezírási hiba
? Close, Write, WriteLn vagy Flush hívásakor a lemez betelt

102 Az állomány nincs hozzárendelve fizikai állományhoz
? Reset, Rewrite, Append, Rename vagy Erase hívása előtt elmaradt az Assign

103 Az állomány nincs nyitva
? Close, Read, Write, Seek, Eof, FilePos, FileSize, Flush, BlockRead vagy BlockWrite hívásakor az állomány nincs nyitva

104 Az állomány nincs megnyitva olvasásra
? Read, ReadLn, Eof, Eoln, SeekEof vagy SeekEoln hívásakor a szöveges állomány nincs megnyitva olvasásra

105 Az állomány nincs megnyitva írásra
? Write, Writeln hívásakor a szöveges állomány nincs megnyitva írásra

106 Érvénytelen numerikus formátum
? Read, ReadLn hívásakor a numerikus forma hibás

Kritikus hibák

150 A lemez írásvédett
151 Ismeretlen egység
152 A lemezegység üzemképtelen
153 Ismeretlen parancs
154 CRC hiba
155 Hibás struktúrahossz lemezegységművelet hívásakor
156 Lemez-pozícionálási hiba
157 Ismeretlen adathordozó típus
158 A szektor nem található

159 A nyomtatóból kifogyott a papír
160 Hiba eszközre történő írás közben
161 Hiba eszközről történő olvasás közben
162 Hardver hiba

Fatális, mindig programleállást okozó hibák

200 Osztás nullával
201 Értékhatár-túllépés
- ? $R+ mellett fordított programok esetében
- ? a tömb valamely indexe kiesik a deklarált intervallumból
- ? értékadásban a kifejezés értéke kiesik a változó értékkészletéből
- ? az eljárás vagy függvény valamely aktuális paraméterének értéke kiesik a formális paraméter értékkészletéből

202 Verem túlcsordulás
- ? $S+ mellett fordított programok esetében eljárás vagy függvény hívásakor
- ? nincs elég hely a veremben
- ? a rutin önmagát hívja (rekurzív hívás), és a visszatérés nincs biztosítva
- ! növelje meg a verem méretét a $M direktívával

203 Heap túlcsordulási hiba
- ? New vagy GetMem hívásakor nincs elég hely a heap-ben

204 Érvénytelen mutató-művelet
- ? Dispose vagy FreeMem hívásakor
- ? a mutató értéke Nil

? a mutató nem a heap–be mutat
? a szabad lista nem bővíthető
? HeapPtr túl közel van a szabad lista elejéhez

205 Lebegőpontos túlcsordulás

206 Lebegőpontos alulcsordulás (csak 80x87 használata esetén)

207 Érvénytelen lebegőpontos művelet

? a Trunc vagy Round által kapott valós érték nem alakítható át egésszé a LongInt intervallumában
? az Sqrt paramétere negatív
? az Ln paramétere nulla vagy negatív
? a 80x87-es saját veremtára túlcsordult

208 Az overlay-kezelő nincs installálva

? a program overlay szubrutint hív, installált overlay-kezelő nélkül
! hívja az OvrInit eljárást a program elején

209 Overlay-állomány olvasási hiba

210 Az objektum nincs inicializálva

? R+ mellett fordított program esetén egy olyan objektum virtuális metódusa került meghívásra, amely előzőleg nem lett inicializálva konstruktor hívásával, és így annak VMT mezője nem egy VMT táblára mutat.

211 Absztrakt metódus hívása

? az öröklött absztrakt metódust nem deklarálja újra a leszármazott objektum
⇨ a Turbo Vision rendszer Objects egységében számos absztrakt metódus található, melyeket – funkciójukból következően – az öröklőnek felül kell deklarálnia

212 Stream regisztrációs hiba
- ? a regisztrációs rekord nem az adatszegmensben foglal helyet
- ? a regisztrációs rekord ObjType mezőjének értéke nulla
- ? az objektum típust már regisztrálta a program
- ? már regisztrált a program egy objektum típust ugyanilyen ObjType mező értékkel

213 A kollekció indexe kívülesik a lehetséges intervallumon

214 Kollekció túlcsordulás
- ! a kollekció tovább nem bővíthető, így újabb elem nem adható hozzá

Turbo Access futási hibakódok

1000 A rekord mérete nagyobb, mint MaxDataRecSize

1001 Rekord mérete 8 bájtnál kisebb

1002 A kulcs hossza nagyobb, mint MaxKeyLen

1003 Az adatállományt eltérő rekordmérettel hozták létre

1004 Az indexállományt eltérő kulcs vagy lapmérettel hozták létre

1005 Nincs elég memória a lappuffer részére

Billentyű	*Funkció*
Alt-szóköz	System menü (≡)
Alt-C	**C**ompile menü
Alt-D	**D**ebug menü
Alt-E	**E**dit menü
Alt-F	**F**ile menü
Alt-H	**H**elp menü
Alt-O	**O**ptions menü
Alt-R	**R**un menü
Alt-S	**S**earch menü
Alt-W	**W**indow menü
Alt-X	kilépés a keretrendszerből (**F**/Exit)
F1	Segítség (**H**/Contents)
F2	Mentés (**F**/Save)
F3	Betöltés (**F**/Open)
F4	Futtatás a kurzor soráig (**R**/Go to cursor)
F5	Ablak kicsi nagy (**W**/Zoom)
F6	Következő ablak (**W**/Next)
F7	Lépésenkénti futtatás, a szubrutinokba belemegy (**R**/Trace into)
F8	Lépésenkénti futtatás, szubrutinok egyben (**R**/Step over)
F9	Make típusú fordítás (**C**/Make)
F10	Főmenü
Alt-F1	Előző segítség (**H**/Previous topic)
Alt-F3	Az aktuális szövegszerkesztő ablak lezárása (**W**/Close)

Alt-F5	Kimeneti képernyő (**W**/User screen)
Alt-F9	Az aktuális szövegszerkesztő ablakban lévő program/egység fordítása (**C**/Compile)
Ctrl-F1	Segítség a kurzor feletti szóra (**H**/Topic search)
Ctrl-F2	A futó program inicializálása (**R**/Program reset)
Ctrl-F3	Szubrutinverem ablak (**W**/Call stack)
Ctrl-F4	Kifejezés kiértékelése (**D**/Evalute/modify)
Ctrl-F5	Az aktulális szövegszerkesztő ablak mozgatása/méretezése (**W**/Size/move)
Ctrl-F7	Új értékkövető megadása (**D**/Watches/Add watch)
Ctrl-F8	Töréspont be/ki (**D**/Toggle breakpoint)
Ctrl-F9	A program (tovább-)futtatása (**R**/Run)
Shift-F1	Segítség tárgymutató (**H**/Index)
Shift-F6	Előző ablak (**W**/Previous)
F1 F1	Segítség a segítség használatához (**H**/Help on help)
Alt-0	Aktív szövegszerkesztő ablakok listája (**W**/List)
Alt-1..9	Az illető számú szövegszerkesztő ablak kiválasztása
Ctrl-Del	Blokk törlése (**E**/Clear)
Shift-Del	Blokk mozgatása a vágólapra (**E**/Cut)
Ctrl-Ins	Blokk kimásolása a vágólapra (**E**/Copy)
Shift-Ins	Blokk bemásolása a vágólapról (**E**/Paste)
Ctrl-L	Előző keresés/keresés és csere ismétlése (**S**/Search again)

Ctrl-Break

A futó program megszakítása. Amennyiben az esetlegesen használt Crt egység CheckBreak változója True értékű (ez az alaphelyzet), a program megszakítható, egyébként nem. A keretrendszerben futtatott programok – a változó értékétől függetlenül – mindig megszakíthatók.

ESC

Visszatérés az előző menüszintre, a főmenüből és a dialógus ablakokból az aktuális szövegszerkesztő ablakba.

Kurzorvezérlők

Kimeneti ablakban:	az ablak tartalmának pásztázása
Figyelő-ablakban:	aktuális értékkövető kiválasztása illetve az aktuális sor pásztázása jobbra/balra

Kurzormozgatás

Ctrl-S, ←	előző karakterre
Ctrl-D, →	következő karakterre
Ctrl-A, Ctrl- ←	előző szóra*
Ctrl-F, Ctrl- →	következő szóra*
Ctrl-E, ↑	előző sorra
Ctrl-X, ↓	következő sorra
Ctrl-W	pergetés fel
Ctrl-Z	pergetés le
Ctrl-R, PgUp	előző lapra
Ctrl-C, PgDn	következő lapra
Ctrl-Q S, Home	sor elejére
Ctrl-Q D, End	sor végére
Ctrl-Q E, Ctrl-Home	ablak elejére
Ctrl-Q X, Ctrl-End	ablak végére
Ctrl-Q R, Ctrl-PgUp	szöveg elejére
Ctrl-Q C, Ctrl-PgDn	szöveg végére
Ctrl-Q B	blokk elejére
Ctrl-Q K	blokk végére

Ctrl-Q P	a legutolsó ezt megelőző kurzorpozícióra
Ctrl-Q W	előző hibahelyre (a hibaüzenet is megjelenik fent)
Ctrl-Q 0..9	megadott számú jelzőre, melyet a Ctrl-K 0..9 kombinációval állítottunk be

* Szóhatároló karakterek:
szóköz < > , ; . ( ) [ ] { } ^ ' * + - / $ # _ = | ~ ? ! " % & ` : @ \
valamint minden vezérlő és grafikus (127-nél nagyobb kódú) karakter.

Beszúrás/törlés

Ctrl-V, Ins	beszúró/felülíró mód váltása (**O/E/E/**Insert mode)
Ctrl-N	sor beszúrása
Ctrl-Y	sor törlése
Ctrl-Q Y	törlés a sor végéig
Ctrl-H, Backspace	kurzor előtti karakter törlése
Ctrl-G, Del	kurzor pozíciójában lévő karakter törlése
Ctrl-T	szó törlése a kurzortól jobbra
Ctrl-K Y	blokk törlése

Blokk-kezelés

Ctrl-K B	blokk elejének kiválasztása
Ctrl-K K	blokk végének kiválasztása

Shift-kurzorvezérlők	blokk kiválasztása (egérrel: nyomvatartott bal gomb mellett az egér mozgatása)
Ctrl-K T	szó kiválasztása blokk-ként
Ctrl-K P	blokk nyomtatása
Ctrl-K C	blokk másolása a kurzor pozíciójába
Ctrl-K Y, Ctrl-Del	blokk törlése
Ctrl-K H	blokk látható/rejtett
Ctrl-K V	blokk átmozgatása a kurzor pozíciójába
Ctrl-K R	blokk beolvasása lemezről a kurzor pozíciójába
Ctrl-K W	blokk kiírása lemezre
Ctrl-K I	blokk egy karakterrel jobbra mozgatása
Ctrl-K U	blokk egy karakterrel balra mozgatása
Ctrl-Ins	blokk másolása a vágólapra
Shift-Ins	blokk bemásolása a vágólapról a kurzor pozíciójába
Shift-Del	blokk mozgatása a vágólapra
Ctrl-Del	blokk törlése (a vágólapra nem másolódik be)

Egyéb funkciók

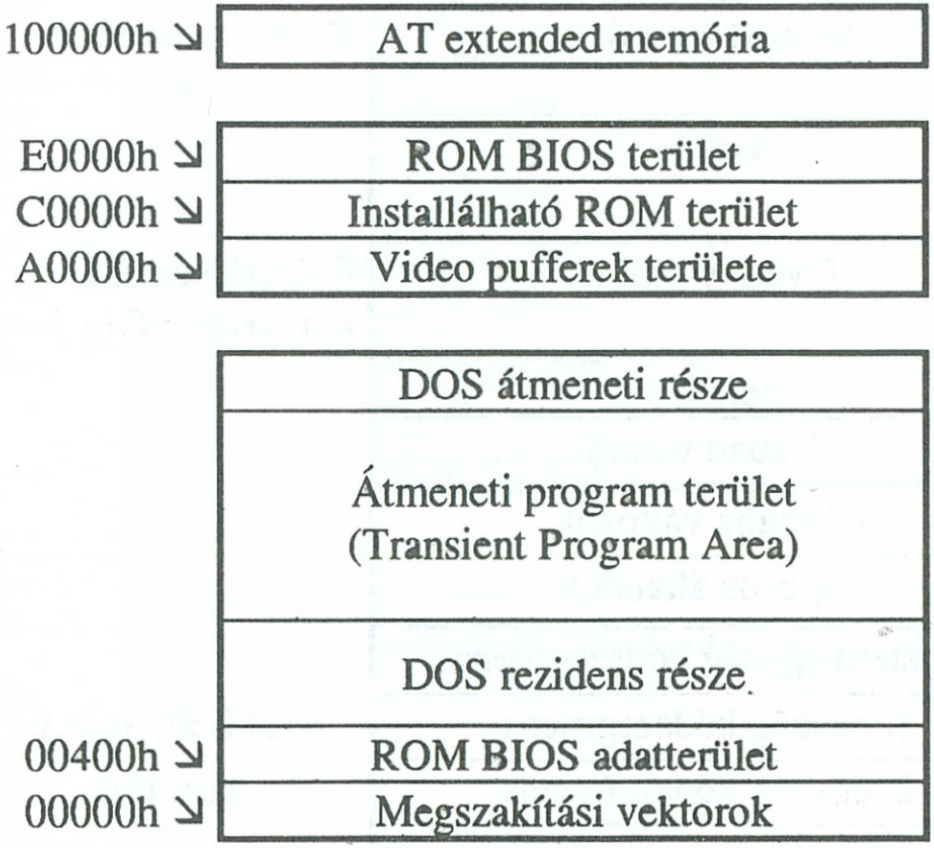
Ctrl-O I	eltolás soremelésnél (autoindent) be/ki (**O/E/E/A**utoindent mode)

Ctrl-O U	eltolás visszatörlésnél (unindent) be/ki (**O/E/E/B**ackspace unindents)
Ctrl-O F	optimális feltöltés (optimal fill) be/ki (**O/E/E/O**ptimal fill)
Ctrl-O T	tabulációs mód váltása (**O/E/E/U**se tab characters)
Ctrl-O R	kurzormozgatási mód TAB karaktereken (**O/E/E/C**ursor through tabs)
Ctrl-P Ctrl-*x*	*x* vezérlőkarakter beírása
Ctrl-K S, F2	mentés (**F/S**ave)
Ctrl-Q F	keresés (**S/F**ind...)
Ctrl-Q A	keresés és csere (**S/R**eplace...)
Ctrl-L	előző keresés/keresés és csere ismétlése (**S/S**earch again)
Ctrl-O O	az **O/C**ompiler és **O/M**emory sizes ablak opcióinak beszúrása a szöveg elejére
Ctrl-K 0..9	adott számú jelző elhelyezése
Ctrl-I, TAB	tabulálás
Ctrl-Q [	a kurzor pozíciójában lévő (, [, {, <, ', " jel párjának keresése előrefelé (a szöveg vége felé)*

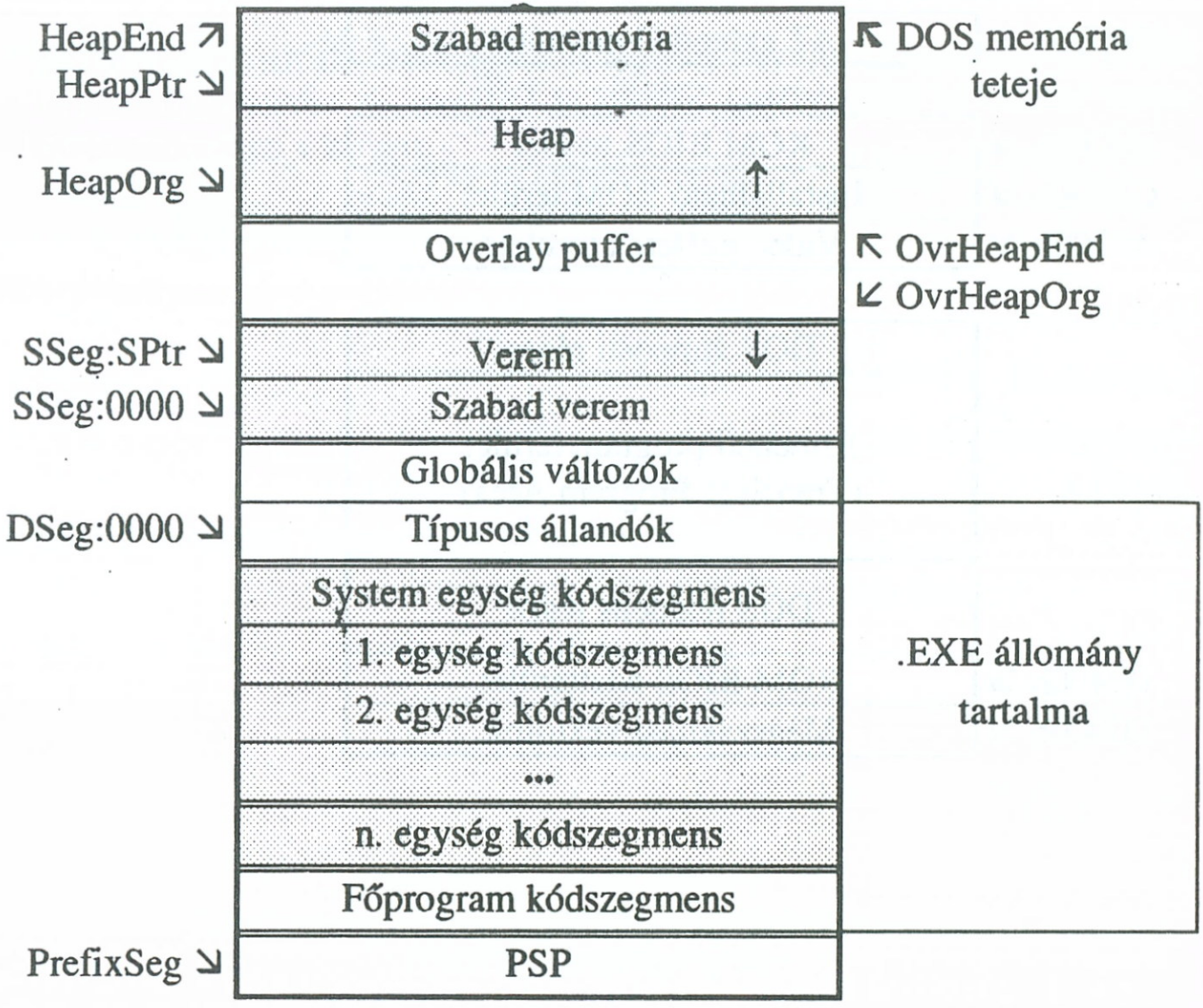
Ctrl-Q]	a kurzor pozíciójában lévő),], }, >, ', " jel párjának megkeresése visszafelé (a szöveg eleje felé)*

* A zárójelek párjait akkor is megtalálja a Ctrl-Q [és Ctrl-Q] parancs, ha többet ágyaztunk egymásba, de aposztróf és idézőjel esetében csak az előzőre illetve következőre viszi a kurzort.

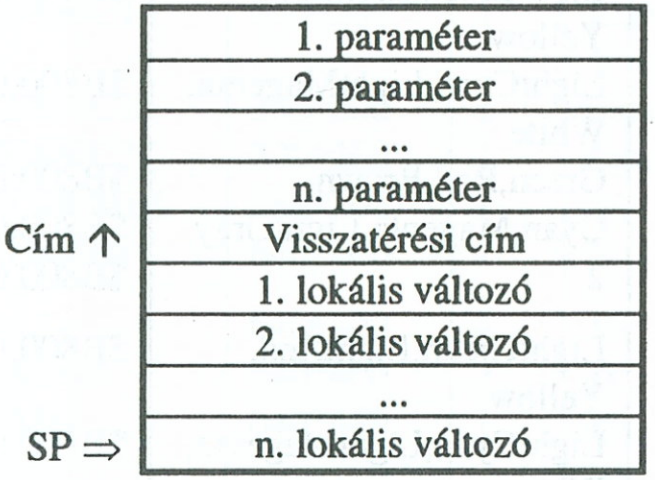
Az IBM XT/AT vázlatos memóriafelosztása



A Turbo Pascal által használt memória térképe



A szubrutinverem felépítése



A visszatérési cím NEAR model esetén WORD, FAR model esetén DWORD. A paraméterek mérete típusfüggő, de minimum 2 bájt (WORD).

GRAFIKUS ÜZEMMÓDOK JELLEMZŐI

Üzemmód	*Felbontás*	*Lap*	*Színek*	*0. lap címe*
CGAC0	320*200	1	LightGreen,LightRed, Yellow	$B800:0
CGAC1	320*200	1	LightCyan,LightMagenta, White	$B800:0
CGAC2	320*200	1	Green,Red,Brown	$B800:0
CGAC3	320*200	1	Cyan,Magenta,LightGray	$B800:0
CGAHi	640*200	1	2	$B800:0
MCGAC0	320*200	1	LightGreen,LightRed, Yellow	$B800:0
MCGAC1	320*200	1	LightCyan,LightMagenta, White	$B800:0
MCGAC2	320*200	1	Green,Red,Brown	$B800:0
MCGAC3	320*200	1	Cyan,Magenta,LightGray	$B800:0
MCGAMed	640*200	1	2	$B800:0
MCGAHi	640*480	1	2	$A000:0
EGALo	640*200	4	16	$A000:0
EGAHi	640*350	2	16	$A000:0
EGA64Lo	640*200	1	16	$A000:0
EGA64Hi	640*350	1	4	$A000:0
EGAMonoHi	640*350	1/2	2	$A000:0
HercMonoHi	720*348	2	2	$B000:0

ATT400C0	320*200	1	LightGreen,LightRed, Yellow	$B800:0
ATT400C1	320*200	1	LightCyan,LightMagenta, White	$B800:0
ATT400C2	320*200	1	Green,Red,Brown	$B800:0
ATT400C3	320*200	1	Cyan,Magenta,LightGray	$B800:0
ATT400Med	640*200	1	2	$B800:0
ATT400Hi	640*400	1	2	$A000:0
VGALo	640*200	4	16	$A000:0
VGAMed	640*400	2	16	$A000:0
VGAHi	640*480	1	16	$A000:0
PC3270Hi	720*350	1	2	$B000:0
IBM8514Lo	640*480		256	$A000:0
IBM8514Hi	1024*768		256	$A000:0

BILLENTYŰZET KÓDTÁBLA

Bill.	*Normál*		*Shift*		*Alt*		*Ctrl*	
	dec	*hex*	*dec*	*hex*	*dec*	*hex*	*dec*	*hex*
a A	97	61	65	41	0, 30	00, 1E	1	01
b B	98	62	66	42	0, 48	00, 30	2	02
c C	99	63	67	43	0, 46	00, 2E	3	03
d D	100	64	68	44	0, 32	00, 20	4	04
e E	101	65	69	45	0, 18	00, 12	5	05
f F	102	66	70	46	0, 33	00, 21	6	06
g G	103	67	71	47	0, 34	00, 22	7	07
h H	104	68	72	48	0, 35	00, 23	8	08
i I	105	69	73	49	0, 23	00, 17	9	09
j J	106	6A	74	4A	0,.36	00, 24	10	0A
k K	107	6B	75	4B	0, 37	00, 25	11	0B
l L	108	6C	76	4C	0, 38	00, 26	12	0C
m M	109	6D	77	4D	0, 50	00, 32	13	0D
n N	110	6E	78	4E	0, 49	00, 31	14	0E
o O	111	6F	79	4F	0, 24	00, 18	15	0F
p P	112	70	80	50	0, 25	00, 19	16	10
q Q	113	71	81	51	0, 16	00, 10	17	11
r R	114	72	82	52	0, 19	00, 13	18	12
s S	115	73	83	53	0, 31	00, 1F	19	13
t T	116	74	84	54	0, 20	00, 14	20	14
u U	117	75	85	55	0, 22	00, 16	21	15
v V	118	76	86	56	0, 47	00, 2F	22	16
w W	119	77	87	57	0, 17	00, 11	23	17
x X	120	78	88	58	0, 45	00, 2D	24	18
y Y	121	79	89	59	0, 21	00, 15	25	19
z Z	122	7A	90	5A	0, 44	00, 2C	26	1A

Bill.	*Normál*		*Shift*		*Alt*		*Ctrl*	
	dec	*hex*	*dec*	*hex*	*dec*	*hex*	*dec*	*hex*
0)	48	30	41	29	0, 129	00, 81	—	—
1 !	49	31	33	21	0, 120	00, 78	—	—
2 @	50	32	64	40	0, 121	00, 79	0, 3	00, 03
3 #	51	33	35	23	0, 122	00, 7A	—	—
4 $	52	34	36	24	0, 123	00, 7B	—	—
5 %	53	35	37	25	0, 124	00, 7C	—	—
6 ^	54	36	94	5E	0, 125	00, 7D	30	1E
7 &	55	37	38	26	0, 126	00, 7E	—	—
8 *	56	38	42	2A	0, 127	00, 7F	—	—
9 (	57	39	40	2B	0, 128	00, 80	—	—
space	32	20	32	20	0, 2	00, 02	32	20
Enter	13	ÖD	13	0D	—	—	10	0A
*	42	2A	42	2A	—	—	—	—
+	43	2B	43	2B	—	—	—	—
- _	45	2D	95	5F	—	—	—	—
= +	61	3D	43	2B	0, 131	00, 83	—	—
, <	44	2C	60	3C	—	—	—	—
. >	46	2E	62	3E	—	—	—	—
; :	59	3B	58	3A	—	—	—	—
/ ?	47	2F	63	3F	—	—	—	—
[{	91	5B	123	7B	—	—	27	1B
\ |	92	5C	124	7C	—	—	—	—
] }	93	5D	125	7D	—	—	29	1D
' "	39	27	34	22	—	—	—	—
` ~	96	60	126	7E	—	—	—	—

BILLENTYŰZET KÓDTÁBLA

Bill	*Normál*		*Shift*		*Alt*		*Ctrl*	
	dec	*hex*	*dec*	*hex*	*dec*	*hex*	*dec*	*hex*
F1	0, 59	00, 3B	0, 84	00, 54	0, 104	00, 68	0, 94	00, 5E
F2	0, 60	00, 3C	0, 85	00, 55	0, 105	00, 69	0, 95	00, 5F
F3	0, 61	00, 3D	0, 86	00, 56	0, 106	00, 6A	0, 96	00, 60
F4	0, 62	00, 3E	0, 87	00, 57	0, 107	00, 6B	0, 97	00, 61
F5	0, 63	00, 3F	0, 88	00, 58	0, 108	00, 6C	0, 98	00, 62
F6	0, 64	00, 40	0, 89	00, 59	0, 109	00, 6D	0, 99	00, 63
F7	0, 65	00, 41	0, 90	00, 5A	0, 110	00, 6E	0, 100	00, 64
F8	0, 66	00, 42	0, 91	00, 5B	0, 111	00, 6F	0, 101	00, 65
F9	0, 67	00, 43	0, 92	00, 5C	0, 112	00, 70	0, 102	00, 66
F10	0, 68	00, 44	0, 93	00, 5D	0, 113	00, 71	0, 103	00, 67
F11	0, 133	00, 85	0, 135	00, 87	0, 139	00, 8B	0, 137	00, 89
F12	0, 134	00, 86	0, 136	00, 88	0, 140	00, 8C	0, 138	00, 8A
←	0, 75	00, 4B	0, 75	00, 4B	—	—	0, 115	00, 73
→	0, 77	00, 4D	0, 77	00, 4D	—	—	0, 116	00, 74
↑	0, 72	00, 48	0, 72	00, 48	—	—	—	—
↓	0, 80	00, 50	0, 80	00, 50	—	—	—	—
Home	0, 71	00, 47	0, 71	00, 47	—	—	0, 119	00, 77
End	0, 79	00, 4F	0, 79	00, 4F	—	—	0, 117	00, 75
PgUp	0, 73	00, 49	0, 73	00, 49	—	—	0, 132	00, 84
PgDn	0, 81	00, 51	0, 81	00, 51	—	—	0, 118	00, 76
Ins	0, 82	00, 52	0, 82	00, 52	—	—	0, 4	00, 04
Del	0, 83	00, 53	0, 83	00, 53	—	—	0, 72	00, 48
ESC	27	1B	27	1B	—	—	27	1B
BS	8	08	8	08	—	—	127	7F
TAB	9	09	0, 15	00, 0F	—	—	—	—
PrtSc	—	—	—	—	—	—	0, 114	00, 72

Funkció	*EPSON FX-1000*		*Saját kód*	
	be	*ki*	*be*	*ki*
Nyomtató inicializálása	ESC @			
Levélminőségű (NLQ) nyomtatás	ESC x 1	ESC x 0		
Proporcionális (PS) nyomtatás	ESC p 1	ESC p 0		
Sűrített (condensed) nyomtatás	^O	^R		
Duplaszéles (double width) nyomtatás	ESC W 1	ESC W 0		
Duplamagas (double height) nyomtatás	ESC w 1	ESC w 0		
Kiemelt (emphasized) nyomtatás	ESC E	ESC F		
Duplánütés (double strike)	ESC G	ESC H		
Felső index	ESC S 0	ESC T		
Alsó index	ESC S 1	ESC T		
Aláhúzott (underlined) nyomtatás	ESC – 1	ESC – 0		
Döntött (italic) nyomtatás	ESC 4	ESC 5		
Puffer ürítése, lapdobás	^L			
Puffer ürítése, kocsivissza	^M			
Puffer ürítése, soremelés	^J			

ASCII TÁBLA

dec	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*
0	00	*NUL*	16	10	*DLE*	32	20		48	30	**0**
1	01	*SOH*	17	11	*DC1*	33	21	!	49	31	**1**
2	02	*STX*	18	12	*DC2*	34	22	"	50	32	**2**
3	03	*ETX*	19	13	*DC3*	35	23	#	51	33	**3**
4	04	*EOT*	20	14	*DC4*	36	24	$	52	34	**4**
5	05	*ENQ*	21	15	*NAK*	37	25	%	53	35	**5**
6	06	*ACK*	22	16	*SYN*	38	26	&	54	36	**6**
7	07	*BEL*	23	17	*ETB*	39	27	'	55	37	**7**
8	08	*BS*	24	18	CAN	40	28	(	56	38	**8**
9	09	*TAB*	25	19	*EM*	41	29	)	57	39	**9**
10	0A	*LF*	26	1A	*SUB*	42	2A	*	58	3A	:
11	0B	*VT*	27	1B	*ESC*	43	2B	+	59	3B	;
12	0C	*FF*	28	1C	*FS*	44	2C	,	60	3C	<
13	0D	*CR*	29	1D	*GS*	45	2D	-	61	3D	=
14	0E	*SO*	30	1E	*RS*	46	2E	.	62	3E	>
15	0F	*SI*	31	1F	*US*	47	2F	/	63	3F	?

dec	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	⌂

dec	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*
128	80	Ç	144	90	É	160	A0	á	176	B0	▒
129	81	ü	145	91	æ	161	A1	í	177	B1	▒
130	82	é	146	92	Æ	162	A2	ó	178	B2	▓
131	83	â	147	93	ô/ő†	163	A3	ú	179	B3	│
132	84	ä	148	94	ö	164	A4	ñ	180	B4	┤
133	85	à	149	95	ò/Ó†	165	A5	Ñ	181	B5	╡
134	86	å	150	96	û/ű†	166	A6	ª	182	B6	╢
135	87	ç	151	97	ù/Ú†	167	A7	º/Ő†	183	B7	╖
136	88	ê	152	98	ÿ/Ű†	168	A8	¿	184	B8	╕
137	89	ë	153	99	Ö	169	A9	⌐	185	B9	╣
138	8A	è	154	9A	Ü	170	AA	¬	186	BA	║
139	8B	ï	155	9B	¢	171	AB	½	187	BB	╗
140	8C	î	156	9C	£	172	AC	¼	188	BC	╝
141	8D	ì/Í†	157	9D	¥	173	AD	¡	189	BD	╜
142	8E	Ä	158	9E	₧	174	AE	»	190	BE	╛
143	8F	Å/Á†	159	9F	ƒ	175	AF	«	191	BF	┐

†CWI kód

dec	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*	*dec*	*hex*	*char*
192	C0		208	D0		224	E0	α	240	F0	≡
193	C1		209	D1		225	E1	β	241	F1	±
194	C2		210	D2		226	E2	Γ	242	F2	≥
195	C3		211	D3		227	E3	Π	243	F3	≤
196	C4		212	D4		228	E4	Σ	244	F4	⌠
197	C5		213	D5		229	E5	σ	245	F5	⌡
198	C6		214	D6		230	E6	µ	246	F6	÷
199	C7		215	D7		231	E7	τ	247	F7	≈
200	C8		216	D8		232	E8	ϕ	248	F8	°
201	C9		217	D9		233	E9	θ	249	F9	•
202	CA		218	DA		234	EA	Ω	250	FA	·
203	CB		219	DB		235	EB	δ	251	FB	√
204	CC		220	DC		236	EC	∞	252	FC	n
205	CD		221	DD		237	ED	∅	253	FD	2
206	CE		222	DE		238	EE	∈	254	FE	■
207	CF		223	DF		239	EF	∩	255	FF	

– 42, 43
$A+/- 49
$B+/- 49
$D+/- 50
$DEFINE 54
$E+/- 50
$ELSE 55
$ENDIF 55
$F+/- 50
$G+/- 51
$I 51
$I+/- 51
$IFDEF 55
$IFNDEF 55
$IFOPT 55
$L 52
$L+/- 52
$M 52
$N+/- 52
$O 53
$O+/- 53
$R+/- 53
$S+/- 53
$UNDEF 55
$V+/- 54
$X+ 5
$X+/- 54

* 42
+ 42, 43
/ 42
< 43
<= 43
<> 43
= 43
> 43
>= 43

@ 42
@Code 163
@Data 163
@Result 163

AAA 163
AAD 164
AAM 164
AAS 164
Abs 62
Abs 7
Absolute 2
adattípusok 19
ADC 164
ADD 165
AddKey 146
Addr 62

AddRec 146
aktuális paraméterlista 5,15, 16
állomány
  állapotai 105
  -kezelés változói 61
  -név karakterlánc 103
  szöveges 32
  típusnélküli 33
  típusok 20
  típusos 33
AND
  assembly 165
  pascal 2,42
ANDPut 120
AnyFile 105
Append 63
Arc 121
ArcCoordsType 115
Archive 105
ArcTan 63
Array 2,28
ASCII tábla 218
Asm 2,8
assembler 2
  betét 8
  direktíva 56

utasítások 163
Assign 63
AssignCrt 94
ATT400 116
ATT400x 117
azonosító 3

B/K
hibakezelés változói 61
hibák 194
Bar 121
Bar3D 121
Begin 2
Begin – End 9,13-14
beszúrás/törlés 205
betűtípus 118
billentyűzet kódtábla 214
BkSlashFill 119
Black 93,117
Blink 93
BlockRead 64
BlockWrite 64
blokk-kezelés 205
Blue 93,117
Boolean 23
BottomText 119
BOUND 165
Brown 93,118
BW40 93
BW80 93
Byte 21

C40 93
C80 93
CALL 165
Case 2,10
CBW 165
CenterLn 118
CenterText 119
CGA 116
CGACx 117
Char 21
ChDir 65
CheckBreak 93
CheckEof 93
CheckSnow 93
Chr 7,65
ciklus
elöltesztelő 10
elöltesztelő léptető 11
hátultesztelő 10
címke 4
Circle 122
CLC 165
CLD 166
ClearDevice 122
ClearKey 147
ClearViewPort 122
CLI 166
ClipOff 119
ClipOn 119
Close 65
CloseDotFill 119
CloseFile 147
CloseGraph 122
CloseIndex 147
ClrEol 94
ClrScr 94
CMC 166
CMP 166
CMPSB 166
CMPSW 166
CO40 93
CO80 93
Comp 26
ComStr 103
Concat 66
Const 2,15
Constructor 2,37
Copy 66
CopyPut 120

Cos 66
CPU regiszterek 103
CPU86 56
CPU87 56
Crt egység 93
CSeg 66
Ctrl–Break 203
CurrentDriver 117
Cyan 93,117

DAA 167
DarkGray 93,117
DAS 167
DashedLn 118
DataFile 145
DataSet 154
DateTime 103
dátumrekord 103
DB 163
DD 163
Dec
  assembly 167
  pascal 67
DefaultFont 118
definíciók 5
deklarációk érvényességi
  köre 17
Delay 94
Delete 67
DeleteKey 148
DeleteRec 148
DelLine 94
Destructor 2,37
Detect 116
DetectGraph 123
Directory 105
DirectVideo 93
direktíva 3
direktívák 49
  assembler 163
  feltételes 54
  kapcsoló 49
  paraméter 49
  speciális 56
DirStr 103
DiskFree 106
DiskSize 106
Dispose 67
Div
  assembly 167
  pascal 2,42
Do 2
Dos
  egység 103
  hibák 195
DosError 105
DosExitCode 106
DosVersion 107
DottedLn 118
Double 26
DownTo 2
DrawPoly 123
DSeg 68
Duplicates 145
DW 163

EGA 116
  színek 118
EGA64 116
EGA64x 117
EGABlack 118
EGABlue 118
EGABrown 118
EGACyan 118
EGADarkGray 118
EGAGreen 118
EGALightBlue 118
EGALightCyan 118
EGALightGray 118
EGALightGreen 118
EGALightMagenta 118

EGALightRed 118
EGAMagenta 118
EGAMono 116
EGARed 118
EGAWhite 118
EGAx 117
EGAYellow 118
egész típusok 21
egyéb szövegszerkesztő funkciók 206
egység
  Crt 93
  felépítése 14
  Graph 115
  Overlay 141
  Printer 101
  System 61
  TAccess 145
  TAHigh 154
  TSort 159
egyszerű
  típusok 19
  utasítások 8
elágazás 9
eljárás
  felépítése 15
  hívása 8,15
  típus 35
  típusú utasítás 15
eljárások, függvények
  Crt egység 94
  Dos egység 106
  Graph egység 120
  Overlay egység 142
  System egység 62
  TAccess egység 146
  TAHigh egység 154
  TSort egység 159
Ellipse 123
előjelváltás 42
elöltesztelő
  ciklus 11
  léptető ciklus 11
előre definiált szimbólumok 56
Else 2
EmptyFill 119
End 2
ENTER 168
EnvCount 107
EnvStr 107
Eof 68, 69
Eoln 69
Erase 69
EraseFile 149
EraseIndex 149
ErrorAddr 61
értékadás 8,48
  -kompatibilitás 41
értékparaméter átadása 48
érvényességi kör, deklarációk 17
ESC 203
ExactMatch 154
Exec 107
Exit 69
ExitCode 61
ExitProc 61
Exp 69
Extended 26
External 2,57
ExtStr 103

Fail 70
False 23
Far 2,57
fatális hibák 198
FAuxiliary 105
FCarry 105
felépítés
  egység 14

eljárás 15
függvény 16
program 13
felhasználói karakterméret 119
felsorolt típus 23
feltétel 7
feltételes direktívák 54
fenntartott szavak 2
FExpand 108
File 2,33
File Of 33
FileLen 149
FileMode 61
FileName 145
FilePos 71
FileRec 104
FileSize 71
FillChar 71
FillEllipse 124
FillPatternType 115
FillPoly 124
FillSettingsType 115
FindFirst 108
FindKey 149
FindNext 108
flag konstansok 105
FloodFill 124
Flush 72
FlushFile 149
FlushIndex 149
fmClosed 105
fmInOut 105
fmInput 105
fmOutput 105
Font8x8 93
For 2,11
fordítási hibakódok 183
formális paraméterlista 5,15, 16
formábillentyűk 201
Forward 2,57
FOverflow 105
FParity 105
Frac 72
FreeList 61
FreeMem 72
FSearch 109
FSign 105
FSplit 109
függvény
felépítése 16
hívása 5,16
Function 2,13,15,35,37
futási hibakódok 195
FZero 105

gépi kódú betét 8
GetArcCoords 124
GetAspectRatio 125
GetBkColor 125
GetCBreak 110
GetColor 125
GetDate 110
GetDefaultPalette 125
GetDir 73
GetDriverName 125
GetEnv 110
GetFAttr 110
GetFillPattern 126
GetFillSettings 126
GetFTime 110
GetGraphMode 126
GetImage 126
GetIntVec 111
GetLineSettings 127
GetMaxColors 127
GetMaxMode 127
GetMaxX 127
GetMaxY 127
GetMem 73

GetModeName 127
GetModeRange 128
GetPalette 128
GetPaletteSize 128
GetPixel 128
GetRec 150
GetTextSettings 128
GetTime 111
GetVerify 111
GetViewSettings 129
GetX 129
GetY 129
GothicFont 118
Goto 2,4,8
GotoXY 95
grafika vágás 119
grafikus
  ablakjellemzők 115
  hibakódok 116
  meghajtók 116
  üzemmódok jellemzői 212
Graph egység 115
GraphDefaults 129
GraphErrorMsg 130
GraphFreeMemPtr 120
GraphGetMemPtr 120
GraphResult 130
Green 93,117
grXXX konstansok 116

halmaz típus 31
halmazkonstruktor 6,31
Halt 73
HatchFill 119
háttérszínek, Crt egység 93
hátultesztelő ciklus 10
heap-kezelés változói 61
HeapEnd 61
HeapError 61
HeapOrg 61
HeapPtr 61
HercMono 116
HercMonoHi 117
Hi 7, 74
hibakódok
  fordítási 183
  futási 195
  grafikus 116
Hidden 105
HighVideo 95
hívás
  eljárás 8,15
  függvény 8,16
  metódus 8
  statikus 38
  virtuális 38
hivatkozás
  rekordmezőre 30
  tömb elemére 29
HLT 168
HorizDir 119

IBM8514 116
IBM8514x 117
IDIV 168
If 2,9
ImageSize 131
Implementation 2,14
IMUL 169
In
  assembly 169
  pascal 2,43
Inc
  assembly 169
  pascal 74
IndexFile 145
InitGraph 132
Inline, direktíva 58
InLine 2,8
InOutRes 61
Input 61

INSB 170
Insert 74
InsLine 95
InstallUserDriver 132
InstallUserFont 132
INSW 170
Int
  assembly 170
  pascal 75
Integer 21
Interface 2,14
InterleaveFill 119
Interrupt 2,58
intervallum típus 24
INTO 170
Intr 111
IOResult 75
IRET 170
iteráció 10-11
ívjellemzők 115

JMP 172
Jx 171

kapcsoló direktívák 49
karakter típus 21
karakterlánc típus 28
katalógus karakterlánc 103
Keep 111
keresőrekord 103
KeyPressed 95
kifejezés 6
  kiértékelése 47
  konstans 7
kiterjesztés karakterlánc 103
kitöltési
  jellemzők 115
  minta 115
  stílus 119
kompatibilitás
  értékadás 41
  típus 40
konstans
  kifejezés 7
  objektum 38
konstansok
  Crt egység 94
  Dos egység 105
  Graph egység 116
  Overlay egység 141
  System egység 61
  TAccess egység 145
  TAccess.Def 145
  TAHigh egység 154
kritikus hibák 197
kurzormozgatás 204
kurzorvezérlők 203

Label 2,4,13,15
LAHF 172
LastMode 93
LDS 172
LEA 172
LEAVE 172
LeftText 119
legnagyobb színkód 120
Length 7,75
LES 172
LightBlue 93,117
LightCyan 93,117
LightGray 93,118
LightGreen 93,117
LightMagenta 93,117
LightRed 93,117
Line 133
LineFill 119
LineRel 133
LineSettingsType 115
LineTo 133
Ln 75
Lo 7,75

LOCK 163,173
LODSB 173
LODSW 173
logikai típus 23
LongInt 21
LOOPx 173
LowVideo 96
LPT1 101
LSortEOS 160
LSortRelease 160
LSortReturn 160
Lst 101
LtBkSlashFill 119
LtSlashFill 119
LTurboSort 161

Magenta 93,117
MakeFile 150
MakeIndex 150
Mark 76
MaxAvail 76
MaxColors 120
MaxDataRecSize 145
MaxDataType 145
MaxHeight 145
MaxKeyLen 145
MaxKeyType 145
MCGA 116
MCGACx 117
megjegyzés 3
megnyitási módok 105
Mem 29
MemAvail 76
MemL 29
memóriacímek változói 61
memóriatérképek 209
MemW 29
metódus
  -hívás 8
  statikus hívása 38
  virtuális hívása 38
minősített azonosító 4
minősítő utasítás 12
MkDir 76
Mod 2, 42
Mono 93
MOV 174
Move 77
MoveRel 133
MoveTo 133
MOVSB 174
MOVSW 174
MsDos 56,112
MUL 174
mutató
  típusnélküli 34
  típusok 20
  típusos 33
műveletek 42
  precedenciája 47

NameStr 103
Near 2,59
NEG 174
New 77
NextKey 151
Nil 2
NoDuplicates 145
NOP 174
normál színek 117
NormVideo 96
NormWidth 118
NoSound 96
Not
  assembly 175
  pascal 2,42
NOTPut 120
nyomtató vezérlőkódok 217

Object 2,36
objektum

konstans 38
típus 36
Odd 7,78
Of 2
Ofs 78
Ok 147
OpenFile 151
OpenIndex 152
Or
assembly 175
pascal 2,43
Ord 7,78
Order 145
ORPut 120
összetett utasítás 9
OUT 175
Output 61
OUTSB 175
OUTSW 175
OutText 133
OutTextXY 134
overlay
egység 141
-kezelés változói 61
OvrClearBuf 142
OvrCodeList 61
OvrDebugPtr 61
OvrDosHandle 62
OvrEmsHandle 62
OvrEMSPages 141
OvrFileMode 141
OvrGetBuf 142
OvrGetRetry 142
OvrHeapEnd 62
OvrHeapOrg 62
OvrHeapPtr 62
OvrHeapSize 61
OvrInit 142
OvrInitEMS 143
OvrLoadCount 141
OvrLoadList 61
OvrReadBuf 141
OvrReadFunc 141
OvrResult 141
OvrSetBuf 143
OvrSetRetry 143
OvrTrapCount 141
ovrXXX konstansok 141

Packed 2,20
PackTime 112
PageSize 145
PageStackSize 145
palettajellemzők 116
PaletteType 116
ParamCount 78
paraméter
direktívák 49
Self 38
paraméterlista
aktuális 5,15-16
formális 5,15-16
ParamStr 79
parancs-sor karakterlánc 103
PartialMatch 154
PathStr 103
PC3270 117
PC3270Hi 117
Pi 79
PieSlice 134
Pointer 34
PointType 116
pontkoordináta 116
POP 175
POPA 176
POPF 176
Port 29
PortW 29
Pos 79
precedenciasor 47
Pred 7,79

prefixek 163
PrefixSeg 61
PrevKey 152
printer egység 101
Private 2,36
PRN 101
Procedure 2,13,15,35,37
Program 2,13
  -befejezés változói 61
  elemei 1
  felépítése 13
  -sor 3
Ptr 7,80
PUSH 176
PUSHA 176
PUSHF 176
PutImage 134
PutPixel 134
PutRec 153

rajzolási mód 120
Random 80
Randomize 80
RandSeed 61
RCL 177
RCR 177
Read 80, 82
ReadKey 96
ReadLn 82
ReadOnly 105
Real 26
Record 2,30
Rectangle 135
Red 93,117
RegisterBGIDriver 135
RegisterBGIFont 135
Registers 103
rekord típus 30
Release 82
Rename 83
REP 163
REPE 163
Repeat 2,10
REPNE 163
REPNZ 163
REPx 177
REPZ 163
Reset 83
RestoreCrtMode 135
RET 177
Rewrite 84
RightText 119
RmDir 84
ROL 178
Round 7,85
RunError 85

SAHF 178
SAL 178
SansSerifFont 118
SAR 178
SaveIntXX 62
SBB 179
SCASB 179
SCASW 179
SearchKey 153
SearchRec 103
Sector 135
Seek 85
SeekEof 86
SeekEoln 86
Seg 86
SEGCS 163
SEGDS 163
SEGES 163
SEGSS 163
Self 38
Set 2,31
SetActivePage 136
SetAllPalette 136
SetAspectRatio 136

SetBkColor 136
SetCBreak 113
SetColor 137
SetDate 113
SetFAttr 113
SetFillPattern 137
SetFillStyle 137
SetFTime 113
SetGraphBufSize 138
SetGraphMode 138
SetIntVec 113
SetLineStyle 138
SetPalette 138
SetRGBPalette 139
SetTextBuf 86
SetTextJustify 139
SetTextStyle 139
SetTime 114
SetUserCharSize 139
SetVerify 114
SetViewPort 140
SetVisualPage 140
SetWriteMode 140
Shl
  assembly 179
  pascal 2,43
ShortInt 21
Shr
  assembly 179
  pascal 2,43
Sin 87
Single 26
SizeOf 7,87
SlashFill 119
SmallFont 118
SolidFill 119
SolidLn 118
sorszámozott típusok 19
SortEOS 159
SortRelease 159
SortReturn 159
Sound 97
speciális
  direktívák 56
  tömbök 29
SPtr 87
Sqr 88
Sqrt 88
SSeg 88
StackLimit 61
statikus metódus hívása 38
STC 180
STD 180
STI 180
STOSB 180
STOSW 180
Str 88
String 2,28
struktúrált
  típusok 20
  utasítások 9
SUB 180
Succ 7,89
Swap 7,89
SwapVectors 114
SysFile 105
System egység 61
szabványos B/K változói 61
számkonstans 4
szekvencia 9
szelekció 9
  többágú 10
szimbólumok 1
  assembler 163
  előre definiált 56
szövegállás 119
szöveges
  állomány 32
    adatai 104
  B/K puffer 104
  üzemmódok 93

szöveg
-igazítás 119
-jellemzők 115
-konstans 4
-szerkesztő parancsai 204

TAccess egység 145
TAClose 154
TACreate 155
TADelete 155
TAErase 155
TAErrorProc 146
TAFlush 155
TAHigh egység 154
TAInsert 156
TANext 156
TAOpen 156
TAPrev 156
TARead 157
TARecNum 154
TAReset 157
TAUpdate 157
TAWrite 158
TEST 181
Test8087 61
tetőrajzolás 119
Text 32
TextAttr 93
TextBackground 97
TextBuf 104
TextColor 97
TextHeight 140
TextMode 98
TextRec 104
TextSettingsType 115
TextWidth 140
Then 2
ThickWidth 118
tintaszínek, Crt egység 93
típus
-azonosság 40
eljárás 35
felsorolt 23
halmaz 31
intervallum 24
karakter 21
karakterlánc 28
-kompatibilitás 40
-konvertált, változó 5
-konvertált, érték 6
logikai 23
objektum 36
overlay egység 141
rekord 30
TAHigh egység 154
tömb 28
típusnélküli
állomány adatai 104
mutató 34
állomány 33
típusok
állomány 20
Dos egység 103
egész 21
egyszerű 19
mutató 20
sorszámozott 19
struktúrált 20
TABuild 145
TAccess egység 145
valós 25
típusos
állomány 33
adatai 104
mutató 33
To 2
többágú szelekció 10
tömb
speciális 29
típus 28
TopOff 119

TopOn 119
TopText 119
TriplexFont 118
True 23
Trunc 7,89
Truncate 89
TSort egység 159
Turbo Access 145
  hibák 200
TurboSort 160
Type 2,13, 15
TypeOf 90

Unit 2,14
UnpackTime 114
Until 2,10
UpCase 90
UsedRecs 153
UserBitLn 118
UserCharSize 119
UserDotFill 119
Uses 2,13
utasítás 7
  eljárás típusú 15
  minősítő 12
utasítások
  assembly 163
  egyszerű 8
  struktúrált 9
útvonal karakterlánc 103

Val 90
választás 10
valós típusok 25
változó
  Dos egység 105
  -hivatkozás 6
  TAHigh egység 154
  típuskonvertált 5
változók
  Crt egység 93
  Graph egység 120
  Overlay egység 141
  System egység 61
  TAccess egység 146
Var 2,13, 15
véletlenszám generálás
változói 61
VER60 56
VertDir 119
vezérlésátadás 8
VGA 116
VGAx 117
ViewPortType 115
Virtual 2,37
virtuális metódus hívása 38
VMT 37,39
VolumeID 105
vonal
  -jellemzők 115
  -stílus 118
  -vastagság 118

WAIT 181
WhereX 98
WhereY 98
While 2,11
White 93,118
WideDotFill 119
WindMax 93
WindMin 93
Window 98
With 2,12,30,37
Word 21
Write 91
WriteLn 92

XCHG 181
XHatchFill 119
XLAT 181
XOR

assembly 181
pascal 2,43
XORPut 120

Yellow 93,118


495 Ft.