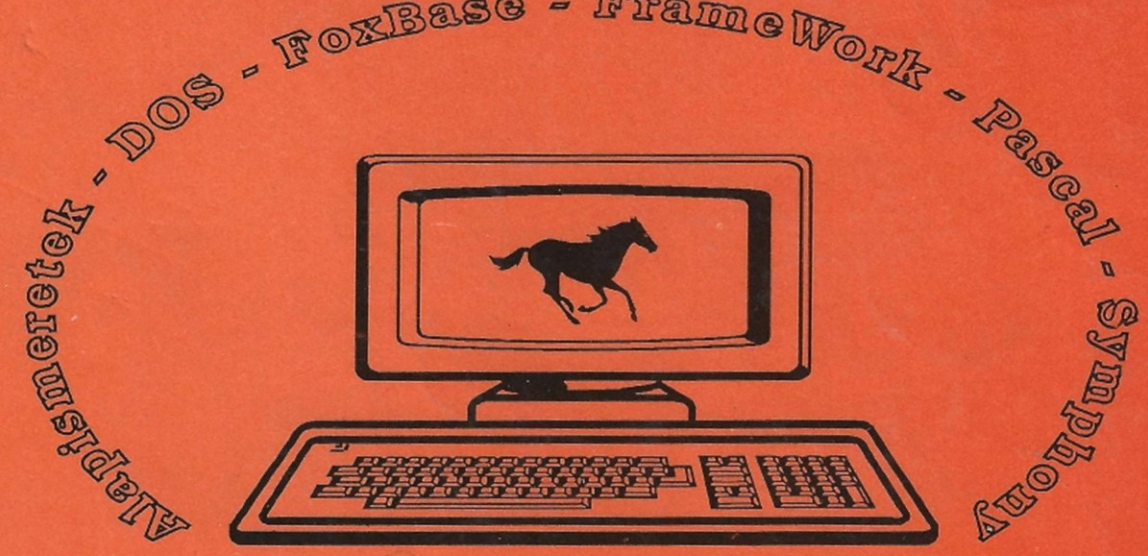
IBM PC XT/AT

kezelése

ALAPISMERETEK ÉS ALKALMAZÓI PROGRAMOK



BINOM Szervezői, Számítástecnikai és Kereskedelmi BT.

IBM PC XT/AT

kezelése

ALAPISMERETEK ÉS ALKALMAZÓI PROGRAMOK

BINOM Szervezői, Számítástecnikai és Kereskedelmi BT.

Készült a
KÖZPONTI STATISZTIKAI HIVATAL
irányelveinek megfelelő
személyi számítógép kezelő és személyi számítógép szoftverüzemeltető
szaktanfolyamokhoz és szakmai képesítő vizsgákhoz

Szerzők

Bartha Mihály
Bosnyák László
Cserny Péter
Madar László
Misnyovszki Magdolna
Varga Zoltán

Alkotó szerkesztő

Dr Imecs László

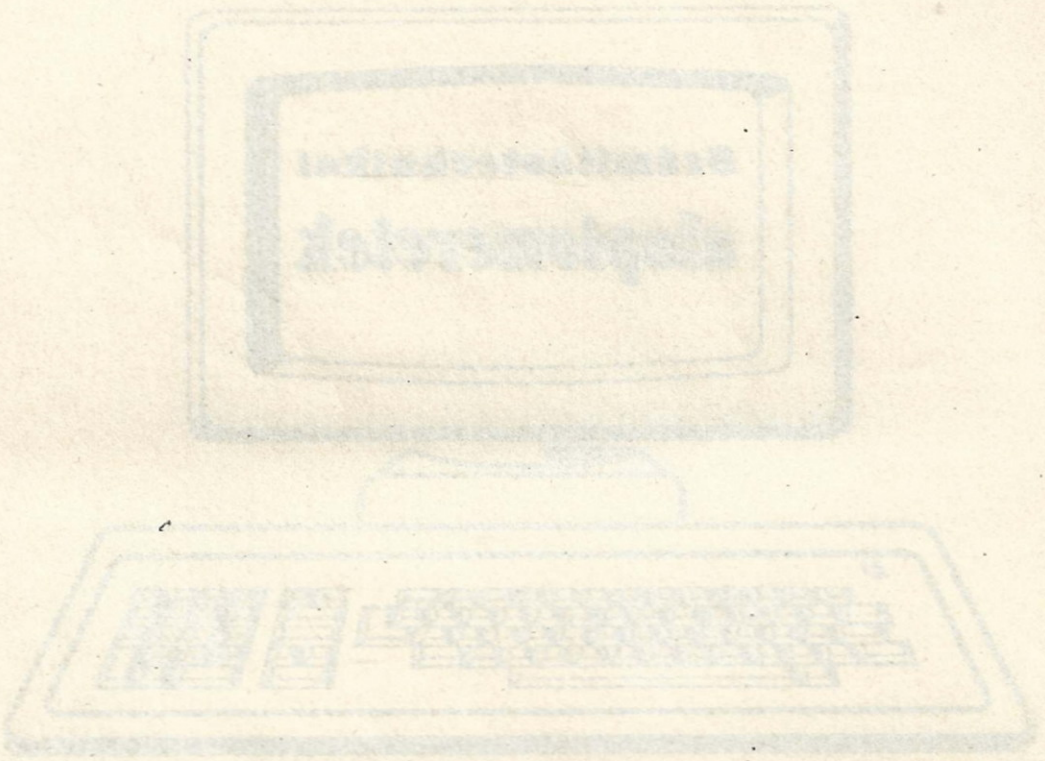
Lezárva: 1992. július 31.

ISBN:
963 04 2450 9

Kiadja a BINOM BT.
Tipográfia: Cserny Péter
Készült a Váci Ofszet Kft. Nyomdájában
Felelős vezető: Szilva István

TARTALOMJEGYZÉK


Számítástechnikai
alapismeretek



1. SZÁMÍTÁSTECHNIKAI ALAPISMERETEK

1.1 A számítógép fogalma, kialakulása

Számítógép minden olyan berendezés, amely képes adatok tárolására, visszakeresésére, feldolgozására emberi beavatkozás nélkül a gépben korábban elhelyezett utasítássorozat segítségével.

Régóta szeretnék az emberek, hogy a számolási, mechanikus műveleteket valamilyen géppel végeztessék el. Az első ilyen eszközt már az ókorban használták, ez volt az ún. abacus. (A japánok, és a kínaiak is használtak már ehhez hasonló eszközt, a soroban-t.)

Később, a középkorban többen is foglalkoztak számológépek készítésével. De bizonyíthatóan az első ténylegesen elkészült és működő számológépet Pascal készítette el. A következő igen fontos állomás volt az automatikus vezérlés (program) megteremtése. (A program utasítások sorozatából épül fel.) Az első ilyen számológép Charles Babbage nevéhez fűződik (a XIX. század közepe). Az ötletet a szövőszékeknél használt lyukasztott kártyáktól kölcsönözte. Ez lehetővé tette, hogy ne csak egy műveletet, hanem többet tudjon egy gép végrehajtani emberi beavatkozás nélkül. Később már nemcsak az elvégzendő műveleteket, hanem az adatokat is ilyen lyukkártyákon, lyukszalagokon tárolták.

Az 1940-es években jött a forradalmi változás. Neumann Jánosnak köszönhetően nyerték el a mai számítógépek jelenlegi formájukat.

A Neumann gép legfontosabb tulajdonságai:

- programvezérelt (hasonlóan Babbage gépéhez),
- tárolta az adatokat és az elvégzendő feladatokat (programot),
- elektronikus (a korábbi gépek mechanikus elemekkel működtek; a mennyiségeket pl. különböző hosszúságú lécekkel jelölték),
- bináris számrendszert használt.

A Neumann gép létrejöttétől kezdődően a számítógépek fejlődését szakaszokra, ún. generációkra osztjuk.

1.2 Számítógép generációk

1. Első generációs számítógépek (1946-58):

- műveletek elvégzésére elektroncsöveket (zárt, légüres térben elhelyezett elektródákból álló egyenirányító, vagy erősítő eszközöket) használtak,
- adatok tárolása lyukkártyán, lyukszalagon történt
- alacsony megbízhatóság, gyakori hibázás,
- néhány ezer művelet másodpercenként.

2. Második generációs számítógépek (1958-65):

- műveletvégzéshez félvezetős tranzisztorokat (félvezető rétegekből felépített kapcsoló, vagy erősítő eszközt) alkalmaztak,
- tároláshoz ferrittárakat használtak,
- 1 millió műv/sec.,
- operációs rendszerek megjelenése (op. rendsz.: olyan programrendszer, amely lehetővé teszi a felhasználó számára a számítógép hatékony és kényelmes használatát. Vezérli a programok végrehajtását, kapcsolatot teremt a felhasználó és a gép között.)

3. Harmadik generációs számítógépek (1965-72):

- integrált áramkörök (IC),
- 10-15 millió műv/sec.,
- a szoftver súlyának növekedése a hardverhez képest, (hardver (hardware): a számítógép műszakilag megalkotott, megépített elemeinek összesége; szoftver (software): a számítógép működése során felhasznált programok, program rendszerek összesége.)

4. Negyedik generációs számítógépek (1972-90-es évekig):

— LSI (nagy bonyolultságú) áramkörök, a számítógép alapvető elemeit néhány integrált áramkör tartalmazza,

— 100 millió műv/sec.,

5. Ötödik generációs számítógépek (1990-es évektől):

— VLSI (nagyon nagy bonyolultságú),

— mesterséges intelligencia megjelenése,

— 1000 millió műv/sec.

1.3 A számítógép felépítése, működése

Mit kell tudnia egy számítógépnek?

— be kell tudnia olvasni a végrehajtandó műveleteket és az adatokat,

— végre kell tudni hajtani a műveleteket az adatokkal,

— ki kell jeleznie, ill. tárolnia kell az eredményeket,

— de legfőképpen vezérelnie kell az egész folyamatot.

Ennek megfelelően a számítógépnek 5 fő egysége van

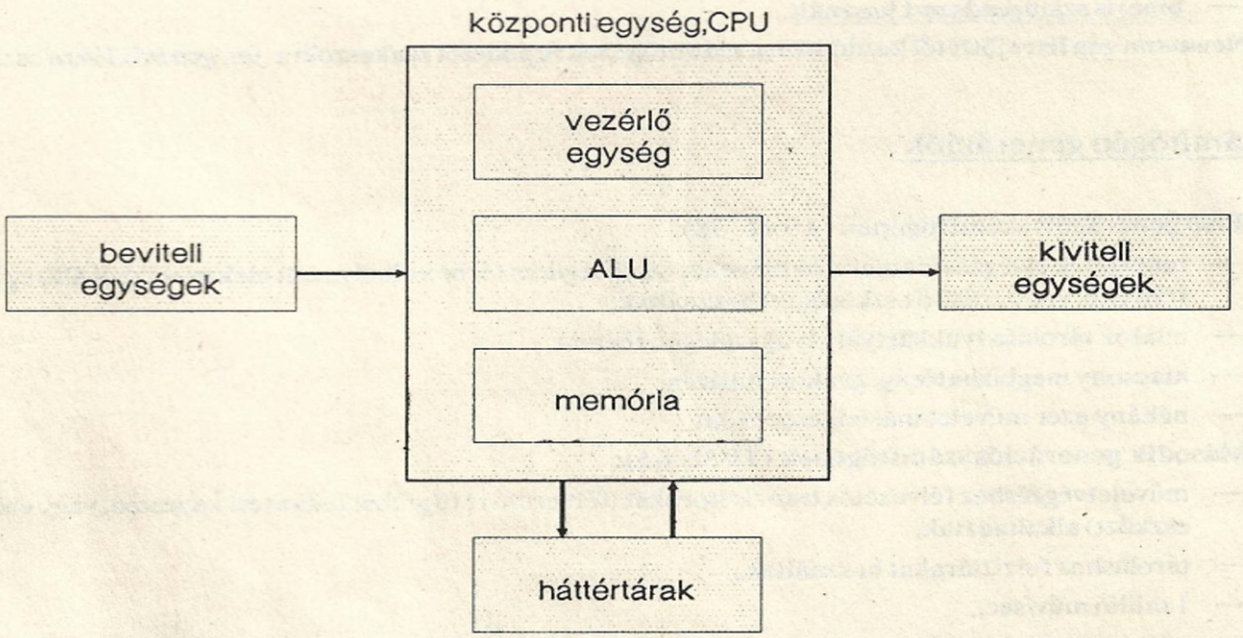
— bemenő (input) egység,

— kimenő (output) egység,

— vezérlő egység,

— aritmetikai (műveletvégző) egység,

— tároló egység.



1. vezérlő egység: Végrehajtja az utasításokat, vezérli a számítógép működését.

2. memória (operatív tár): Az adatok tárolására szolgál. Két fajtáját különböztetjük meg:

— RAM (Random Access Memory): írható-olvasható memória.

— ROM (Read Only Memory): csak olvasható memória.

A RAM és a ROM között az alapvető különbség, hogy a RAM a gép kikapcsolása után elveszti tartalmát, míg a ROM nem. (Létezik olyan ROM is (EPROM), amelynek tartalma külön erre létrehozott eszköz segítségével írható.)

3. aritmetikai-logikai egység (ALU): elemi aritmetikai, számolási műveleteket tud végrehajtani.

4. input és output egység: az ember és a gép közötti kapcsolatot biztosítsák.

Hogyan is működik egy számítógép?

A bemenő egységről a vezérlő egység utasítására a végrehajtandó program ill. az ehhez szükséges adatok bekerülnek a tárba. A vezérlő egység a memóriából sorban előveszi a végrehajtandó utasításokat és az adatokat, s az aritmetikai egységgel elvégezteti a szükséges műveleteket. Ha valamely adat, amely szükséges a feladat végrehajtásához nincs a tárban, akkor a vezérlő egység utasítja az input egységet annak tárba való továbbítására. Az elvégzett műveletek eredményei, ill. a már nem kellő adatok visszakerülnek a memóriába. A memória az eredményeket az output egységre irányítja át. A számolás közben létrejött részeredmények, ill. ideiglenesen nem kellő adatok háttértárolóra kerülnek.

Most miután vázlatosan megnéztük a számítógép belső működését, nézzük meg milyen külső egységei vannak egy számítógépnek.

1.4 A számítógép perifériái

I. Input egységek:

— billentyűzet (klaviatúra, keyboard): Két alapvető típusa ismert: a 83/84 és a 101/102 gombos változat. A legnagyobb része az írógépekre hasonlít. Itt találhatjuk meg az összes betűt és írásjelet.

— egér (mouse): Megfelelő program használatával, az "egeret" az asztalon tologatva, a képernyőn valamilyen alakzat (nyíl, pont stb.) követi elmozdulásait.

— scanner: Ez az eszköz grafikák, fényképek számítógépbe történő bevitelére, ill. feldolgozására szolgál.

II. Output egységek:

— monitor (**display**): Az egy sorban, ill. oszlopban megjeleníthető képpontok száma, ill. a használható színek alapján több csoportba oszthatjuk.

a. **monochrom** (egyszínű): A háttér színén kívül még egy színt tud. Olyan feladatoknál használják, ahol nincs szükség színek használatára, pl: adatrögzítésnél. Gyakran használják az ún. **Hercules** monitort, amely grafikus megjelenítésre is alkalmas. Ennek felbontása 720*348 pont.

b. **CGA**: Már négy szín megjelenítésére alkalmas, de igen rossz felbontással (320*200 v. 640*200). Fárasztja a szemet, alkalmazása nem ajánlott.

c. **EGA**: 16 színt használhatunk, felbontása: 640*350.

d. **VGA**: 256 szín használható, felbontása: 320*200-tól, akár 1024*768-ig is terjedhet. Grafikus tervezési munkáknál, vagy olyan helyen, ahol a színek használata fontos, akár az EGA, akár a VGA használható.

— nyomtató (**printer**): Olyan kiviteli berendezés, amely papíron jeleníti meg a feldolgozás eredményét. Fajtái:

a. **mechanikus** nyomtató: ahol a nyomtatás mechanikus elven alapul. Pl:

i. **mátrixnyomtató**: Minden karaktert pontokból tesz ki (pl: 7*5 v. 9*7). A nyomtatást 7 ill. 9 egymás alatti tű végzi, amely együtt mozog a papír előtt, vízszintes irányban. A megfelelő helyen a tűk kiugranak, és a festékszalagon keresztül nyomot hagynak a papíron.

ii. **sornyomtató**:

- láncos nyomtató: a betűk egy láncon találhatók meg;

- hengeres nyomtató: a betűk egy hengeren helyezkednek el.

b. nem mechanikus: A nyomtatás nem mechanikai, hanem pl: elektronikus, kémiai, vagy hőhatás segítségével történik. Pl:

i. **tintasugaras**: Apró tintacseppeket állít elő, és azt elektrosztatikusan viszi fel a papírra.

ii. **lézernyomtató**: Lézersugár segítségével alakítják ki a képet. Ezekkel a nyomtatókkal a nyomdai minőségű íráskép elérése sem lehetetlen.

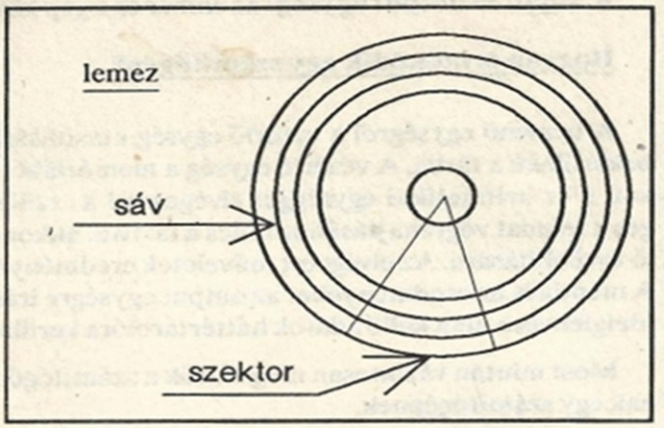
— rajzgép (**plotter**): Ez az eszköz vonalas ábrák rajzolására képes; elsősorban a mérnöki, tervezői munkában használják.

III. I/O (input-output) egységek:

— floppy lemez (hajlékony lemezes tároló)

— winchester (merevlemezes tároló)

Ha a két tárolót összehasonlítjuk, láthatjuk, hogy a winchester gyorsabb, nagyobb tároló kapacitású, viszont nem cserélhető. Mind a floppy lemez, mind a winchester kör alakú lemezekből épül fel, csak míg a merevlemez többből, addig a hajlékony lemez csak egy lemezből. A számítógép az adatokat a lemezen koncentrikus körökön tárolja. Egy-egy kört sávnak hívunk. Minden sáv további részekre, szektorokra van osztva.



Egy lemez tárolókapacitása attól függ, hány oldalas, mennyi sáv van egy lemezen, ill. egy sávot hány szektorra osztottak fel. Egy szektor tárolóképessége minden esetben 512 byte. (A byte fogalmával később foglalkozunk részletesen. Most csak annyit róla, a jobb érthetőség kedvéért, hogy egy betűt a gép egy byte-on tárol. Azaz egy szektorban 512 betű fér el.)

Ismerkedjünk meg néhány a floppy lemezeknél alkalmazott jelöléssel, amelyeket egy lemez vásárlásánál figyelembe kell vennünk. Ezeket az adatokat a lemezen ill. annak dobozán általában feltüntetik.

SS - egy oldalas (single side); DS - két oldalas (double side); SD - alacsony sűrűségű (single density); DD - dupla sűrűségű (double density); HD - magas sűrűségű (high density).

Vegyünk egy ma használatos lemezt: DS-HD (magas sűrűségű, két oldalas): 2 oldala van, egy oldalon 80 sáv, 1 sávban 15 szektor található. Ennek megfelelően ezen a lemezen 1200 Kbyte (Kilobyte) adat tárolható. (2 oldal * 80 sáv * 15 szektor * 512 byte = 1228800 byte = 1200 Kbyte. A kilo szó 1000-szerest jelent a gyakorlatban, a számítástechnikában azonban ez 1024-szerest jelent. Azaz egy ilyen lemezen kb. 1200000 betű tárolható.)

Nézzünk egy másik lemezt: DS-DD (dupla sűrűségű, két oldalas): 2 oldal, egy oldalon 40 sáv, 1 sávban 9 szektor található. Ez a lemez 360 Kbyte (kb. 360000 betű) tárolására alkalmas. (2 oldal * 40 sáv * 9 szektor * 512 byte = 368640 byte = 360 Kbyte).

Egy floppy lemez meghajtójában (ha az képes két oldalas lemezt is olvasni, akkor) két olvasó fej található. Az egyik a lemez egyik felén, a másik a másik felén található. E fejek együtt mozognak, és hozzáérnek a lemezhez. A floppy lemez a meghajtóban csak akkor forog, ha valamit róla olvasunk, vagy írunk rá.

Nézzünk most egy winchestert. A winchesterben több lemez található. Ezek a lemezek egymás felett helyezkednek el. A winchesternél az egymás felett elhelyezkedő sávokat cilindernek nevezzük. A cilinderen belül minden sávhoz tartozik egy fej. Így ahány oldalunk van, annyi olvasó fej van egy winchesterben. (Vannak olyan meghajtók is, amelyeknél vagy a legfelső, vagy a legalsó, esetleg sem a legfelső, sem a legalsó oldalhoz nem tartozik olvasófej.) A winchesternél a fejek ugyancsak együtt mozognak, de nem érnek hozzá a lemezekhez, a lemez felett lebegnek. A winchester gyorsaságának egyik titka, hogy a winchesterben a lemezek folyamatosan mozognak a bekapcsolástól a kikapcsolásig. Vegyünk egy winchestert, amelyben 1024 cilinder (egy lemezen 1024 sáv), 9 fej, egy sávban 17 szektor található. Ennek tároló kapacitása: 9 fej (oldal) * 1024 cilinder (sáv) * 17 szektor * 512 byte = kb. 80000 Kbyte = kb. 80 Megabyte. (1 Megabyte (Mbyte) = 1024 Kbyte)

— streamer (kazetta): A magnókban található kazettákhoz hasonlítható; elsősorban adatmentésre használják.

Megnéztük, hogy min tárolja a számítógép az adatokat, de hogyan tárolja a számítógép az adatokat? Ennek könnyebb megértéséhez meg kell ismerkednünk a számrendszerekkel.

1.5 Számrendszerek

A gyakorlatban leggyakrabban a tizes számrendszert használjuk. Ebben a számrendszerben 10 db számjegy áll rendelkezésünkre egy szám felírásához. Ezek: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Vegyünk egy példát: $345_{(10)}$. (Az alsó indexbe tett és zárójelbe írt számmal jelzem azt, hogy milyen számrendszerről van szó. Jelen esetben ez 10-es. Ezt a számot nevezzük a számrendszer hatványalapjának.)

Minden számjegy esetén két értékről beszélhetünk, ez a helyi és az alaki érték. A $345_{(10)}$ számot véve: az 5-ös az egyes helyi értéken van tehát az ő helyi értéke egyes, alaki értéke pedig 5; a 4-es helyi értéke tizes, alaki értéke 4; a 3-as helyi értéke százas, alaki értéke 3.

A szám értékét az egyes helyi és a hozzájuk tartozó alaki értékek szorzatának összege adja meg. Azaz az előző számot fel tudjuk írni a következőképpen is: $345_{(10)} = 3*100 + 4*10 + 5*1 = 3*10^2 + 4*10^1 + 5*10^0$

A számítástechnikában leggyakrabban a kettes (másnéven bináris), a tizes (decimális) és a tizenhatos (hexadecimális) számrendszereket használjuk.

számrendszer	hatványalap	együtthatók
kettes	2	0,1
tizes	10	0,1,2,3,4,5,6,7,8,9
tizenhatos	16	0,1,2,3,4,5,6,7,8,9,A,B,C.D,E,F

(A tizenhatos számrendszerben a betűk értéke: A=10, B=11, C=12, D=13, E=14, F=15)

Bármely számrendszerbeli szám értékének a kiszámítása ugyancsak a helyi és az alaki értékek segítségével történik.

Vegyünk például egy kettes számrendszerbeli számot: $10110101_{(2)} = 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 1*128 + 0*64 + 1*32 + 1*16 + 0*8 + 1*4 + 0*2 + 1*1 = 128 + 32 + 16 + 4 + 1 = 181.$

Másik példaként vegyünk egy hexadecimális számot: $1AD_{(16)} = 1*16^2 + A*16^1 + D*16^0 = 1*16^2 + 10*16^1 + 13*16^0 = 1*256 + 10*16 + 13*1 = 429_{(10)}$.

1.6 Számrendszerek közötti átalakítások

I. Kettesből és tizenhatosból tizesbe:

Az átszámítás az alaki és a hozzájuk tartozó helyi értékek szorzatának ill. ezek összegének kiszámításával történik, a fent látható módon. Azaz az $10110101_{(2)} = 181_{(10)}$ és $1AD_{(16)} = 429_{(10)}$

II. Tizesből kettesbe és tizenhatosba:

Az átszámításnak két módszerét nézzük meg (mindkét módszer alkalmazható mindkét számrendszer esetén):

a. Az első esetben a példában kettes számrendszerbe váltunk:

Az átszámítandó decimális számot osztjuk kettővel (illetve 16-os számrendszerbe történő átszámítás esetén 16-tal), és külön leírjuk a maradékot. Az osztás eredményét osztjuk újra kettővel (ill. 16-tal), és így tovább egészen addig, míg az osztás eredménye nulla nem lesz. Végül az osztási maradékokat visszafelé összeolvassuk.

Pl:

13	:	2	=	6	1
6	:	2	=	3	0
3	:	2	=	1	1
1	:	2	=	0	1

$1101_{(2)} = 13_{(10)}$

b. A második esetben nézzünk egy-egy példát kettes ill. tizenhatos számrendszer esetén is:

Az átszámítandó decimális számot elosztjuk azzal a tizenhat ill. kettő hatvánnyal, ami még megvan a számban. (Pl: 25-öt véve, 25-ben 16 (16^1) még megvan egyszer, míg 256 (16^2) már nincs meg benne. Kettes számrendszer esetén: vegyük ismét a 25-öt. 25-ben 16 (2^4) megvan, míg 32 (2^5) már nincs meg.) Itt az osztás eredményét írjuk le külön. Ezután az osztás maradékát osztjuk tovább, de most már eggyel kisebb hatvánnyal. Addig, amíg a nulladik hatványig el nem érünk. A külön leírt számokat most felülről lefelé olvassuk össze.

Pl:	300	:	256	=	1	1	$(256 = 16^2; 16 = 16^1; 1 = 16^0)$
	44	:	16	=	2	2	
	12	:	1	=	12	12=C	Azaz: $300_{(10)} = 12C_{(16)}$

Pl:	201	:	128	=	1	1	$(128 = 2^7; 64 = 2^6; 32 = 2^5;$
	73	:	64	=	1	1	$16 = 2^4; 8 = 2^3; 4 = 2^2; 2 = 2^1; 1 = 2^0)$
	9	:	32	=	0	0	
	9	:	16	=	0	0	
	9	:	8	=	1	1	
	1	:	4	=	0	0	
	1	:	2	=	0	0	
	1	:	1	=	1	1	Azaz: $201_{(10)} = 11001001_{(2)}$

III. Kettesből tizenhatosba:

Az átszámítandó bináris szám számjegyeit osszuk fel négyes csoportokba (a legkisebb helyiértéktől kezdődően), majd minden egyes csoportnak számoljuk ki az értékét. Ez lesz a szám hexadecimális számrendszerbeli értéke.

Pl: Vegyünk egy bináris számot: $1001011011_{(2)}$.

10	=	2
0101	=	5
1011	=	11=B

Azaz: $1001011011_{(2)} = 25B_{(16)}$

IV. 16-osból 2-esbe:

Az előző módszert kell visszafelé alkalmazni. Azaz: a hexadecimális szám minden számjegyét átváltjuk kettesbe. Majd kiegészítjük 4 hosszúra, és összeolvassuk a bináris számokat.

Pl: $25B_{(16)}$ esetén: $2 = 10_{(2)}$, $5 = 101_{(2)} = 0101_{(2)}$, $B = 1011_{(2)}$

Összeolvasva: $25B_{(16)} = 1001011011_{(2)}$

Műveletvégzés kettes számrendszerben

A műveletvégzés a kettes számrendszerben (számítógépben) a következő módon történik. A számítógép összeadó egysége egyszerre csak két operanduson tudja elvégezni a műveleteket. Nézzük meg a legegyszerűbb eseteket.

Pl: $0 + 0 = 0$; $1 + 0 = 1$; $0 + 1 = 1$; $1 + 1 = 0$,

Az utolsó esetben átvitel keletkezett, ezt a következő helyiértéken összeadandónak kell tekinteni. Nézzünk egy példát:

	10100111	167
	+10101110	+174
átvitel:	101011100	110
	101010101	341

túlcsordulás

Előfordulhat, hogy az összeg nem fér el annyi helyen, mint az összeadandók. Ezt túlcsordulásnak nevezzük. Mivel a gép azonban csak 8 jegyet tud ábrázolni, ezért ez az érték elveszik.

A kivonást a számítógép az ún. komplemens képzés segítségével oldja meg, és nem a mindenki által megszokott módon. Először ismerkedjünk meg a komplemens fogalmával. A komplemens két, egymást kiegészítő egyikét jelenti. Nézzünk erre egy köznapi példát:

Vegyünk egy autót, amelynek kilométer órája 5 számot tud mutatni. Azaz a legnagyobb kilométer, amit ki tud jelezni, az 99999. Ezután a számláló újra elölről kezdi a számolást. Jelenleg az autó futott már 12345 km-t. Akkor 99999 - 12345 = 87654 km-t megy az autó a legközelebbi körülfordulásig. A 12345 és a 87654 egymás komplementerei.

Most nézzünk egy matematikai példát. Rendelkezésünkre áll 3 jegy a számok ábrázolására. Számoljuk ki 534 komplementerét. Három jegyen 999 a legnagyobb ábrázolható szám. Így 534 komplementere: 999 - 534 = 365.

Nézzük meg egy egyszerű példán, hogyan is végzi el a számítógép a kivonást. Legyen adva két számjegy a számok ábrázolására. Számoljuk ki a következő különbséget: 56 - 22 komplemens segítségével. (Az eredménynek 34-nek kell kijönnie.) Most számoljuk ki 22 komplemensét. Ez 99 - 22 = 77, és adjunk ehhez hozzá 1-t. 77 a 22 komplementere (másnéven egyes komplemense), 78 (77+1) a kettes komplemense. Most adjuk hozzá 56-hoz 22 kettes komplemensét:

56 + 78 = 134

Mivel csak két jegyen tudjuk ábrázolni a számokat, ezért az eredmény 34. Ezt kellett kiszámolnunk. Így a számítógép a kivonást visszavezette összeadásra.

Nézzük ezt most egy kettes számrendszerbeli számon is. Álljon rendelkezésünkre 8 hely egy szám ábrázolására. Számoljuk ki a következő különbséget: 11100101-10011011=?

Vegyük a kisebbítendőt (ez: 10011011) és számítsuk ki ennek kettes komplemensét. Kettes számrendszerben nagyon egyszerű az egyes komplemens kiszámolása: minden számjegyet az ellenkezőjére kell váltani. Azaz, ahol 1 volt, azt 0-ra; ahol 0 volt azt 1-re.

a szám:	10011011	
egyes komplemense:	01100100	
	+1	
1-et hozzá adva:	01100101	ez a kettes komplemens

Adjuk hozzá a kisebbítendőhöz (11100101-hez) a kivonandó (10011011) kettes komplemensét:

Ellenőrizzük le tizes számrendszerbe való átváltással.

```
    11100101
   +01100101
   ----------
átvitel: 111001010
   ----------
    101001010
```

túlcsordulás

$11100101_{(2)} = 229_{(10)}$; $10011011_{(2)} = 155_{(10)}$

$229_{(10)} - 155_{(10)} = 74_{(10)} = 01001010_{(2)}$

Nézzük meg mi a teendő negatív eredmény esetén.

155 - 229 = 155 + (-229) = -74

$229_{(10)} = 11100101_{(2)}$

egyes komplemense: 00011010

kettes: 00011011

```
    10011011
   +00011011
   ----------
átvitel: 10110110
   ----------
    10110110
```

Látható, hogy a nyolcadik pozíción egyes áll. Ez azt jelenti, hogy az eredmény negatív, tehát az eredményt vissza kell alakítani, rekomplementálni kell:

	10110110	
komplementere:	01001001	
	+1	
1-et hozzá adva:	01001010	= 74

Azaz: az összeadásnál a 8. pozíción álló 1-esből láthattuk azt, hogy az eredmény negatív; itt pedig megkaptuk az eredmény értékét. Így 155 -229 = -74

A szorzás visszavezethető ismételt összeadásokra, az osztás pedig kivonásokra.

1.7 Szám- és karakterábrázolás

Térjünk vissza a számítógép adattárolási rendszerére. A számítógép minden információt kétállapotú jelek sorozatával, kódolva tárol. Egy ilyen két állapotú jelet bitnek hívunk. 8 bit = 1 byte.

Ezek az információk lehetnek: betűk, számok, írásjelek, összefoglaló nevükön karakterek. A karakterábrázolás legelterjedtebb módja az EBCDI-kód. Az angol rövidítés a binárisan kódolt decimális számok kiterjesztett átalakító kódja. Minden karakter 8 biten kerül ábrázolásra, azaz összesen 256 darab karakter ábrázolható.

A 8 bit két félre van osztva (tetrádok). Az első négy bit a zónarész, a második négy a számrész. (Az egyes részek tartalma egy-egy tizenhatos számrendszerbeli számmal ábrázolható.) Ez a kódrendszer csak szöveges információk kezelésére alkalmas, a kódokkal aritmetikai műveleteket végezni nem lehet. A zónarész a karakter típusára utal (számoknál 1111), a számrész pedig a konkrét tartalomra utal. Előjel nélküli számok ábrázolásánál a zónarészt figyelmen kívül hagyva kapjuk meg a számot.

Nézzük meg hogyan ábrázolunk egy számot; Pl: 269

Binárisan:	Hexadecimálisan:
1111 0010	F2
1111 0110	F6
1111 1001	F9

Előjeles számoknál az 1100 és az 1101 előjelkódot használjuk.

Az eddigiek a zónázott decimális ábrázolást mutatták. Ha csak számokat akarunk ábrázolni, a zónarész el is hagyható. Ekkor egy bájton két számot tudunk ábrázolni. Ezt a megoldást tömörített decimális formának nevezik.

Egy másik ábrázolási forma a BCD (binárisan kódolt decimális) kódolás. Ekkor minden számjegyet 4 biten ábrázolnak, mégpedig az adott számjegy kettes számrendszerbe történő átváltásával. Pl: 29-et ábrázoljuk BCD kódban.

$0010_{(2)}$	=	$2_{(10)}$
$1001_{(2)}$	=	$9_{(10)}$
Az ábrázolandó szám:		29
A BCD-ben kódolva:		00101001

A mikroszámítógépek egy harmadik kódrendszert az ASCII kódot is használják. Ez 8 bites kód, eredetileg 7 biten kódolták az adatokat, a 8. bit ellenőrzésre szolgált. 7 biten 2^7 (128) adatot tudunk tárolni. Ez a számoknak, az angol abc betűinek és néhány speciális jelnek az ábrázolására ugyan alkalmas, de szükségessé vált például az ékezetes betűk, esetleg ciril betűk ábrázolása is. Ezért később a nyolcadik bitet is adattárolásra használták fel. Így már 2^8 (256) karaktert lehet ábrázolni ASCII kódban. Az ASCII kódrendszer 0-31. sorszámig különböző vezérlő karaktereket tartalmaz. 32-től 47-ig vannak az írásjelek, 48-tól a számok, 65-től a nagybetűk, 97-től a kisbetűk.

A számok ábrázolására két további módszer terjedt el:

— a fixpontos és

— a lebegőpontos számábrázolás.

A fixpontos számábrázolást egész számok ábrázolására használják. Ez pozitív számok esetén az adott szám kettes számrendszerbeli, negatív számok esetén pedig annak kettes komplemens alakját jelenti.

Pl: (8 biten ábrázolva a számokat)

$00010011_{(2)} = 19_{(10)}$
$11101101_{(2)} = -19_{(10)}$

(A számolás a következőképpen történik: $19_{(10)}$ (pozitív szám) esetén a számot át kell váltani kettes számrendszerbeli alakra. $-19_{(10)}$ (negatív szám) esetén pedig a szám abszolút értékének (19-nek) kettes számrendszerbeli alakját (ez 00010011) kell kiszámítani; majd ennek először egyes komplemensét (11101100), azután kettes komplemensét (11101101). Ez lesz a $-19_{(10)}$ fixpontos alakja.)

Fixpontos számábrázolás esetén nézzük meg a legnagyobb ill. legkisebb ábrázolható számot (8 biten ábrázolva):

$01111111_{(2)} = 127_{(10)}$
$01111110_{(2)} = 128_{(10)}$
...
$00000001_{(2)} = 1_{(10)}$
$00000000_{(2)} = 0_{(10)}$
$11111111_{(2)} = -1_{(10)}$
$11111110_{(2)} = -2_{(10)}$
...
$10000001_{(2)} = -127_{(10)}$
$10000000_{(2)} = -128_{(10)}$

Ez alapján a legkisebb ábrázolható szám (8 biten) -128, azaz $-2^{(8-1)}-1$; a legnagyobb pedig 127, azaz $2^{(8-1)}$.

Általánosítva: n biten történő ábrázolás esetén

a legkisebb ábrázolható szám: $-2^{(n-1)}-1$
a legnagyobb: $2^{(n-1)}$

A fixpontos számábrázolásnál csak egész számok fejezhetők ki. Valós számok ábrázolására a lebegőpontos számábrázolást használják. Ehhez azonban ismerni kell a számok normál alakját.

Minden valós szám fölírható a következő alakban: $A=M*R^k$, ahol A az eredeti szám, M az együttható, vagy mantissza, R a számrendszer alapszáma, k pedig a kitevő, vagy karakterisztika. Pl:

$123,4_{(10)} = 1,234 * 10^2$ $\qquad$ $0,00123_{(10)} = 1,23 * 10^{(-3)}$

(A mantissza értéke a következő lehet:

tizes számrendszer esetén: $0 <= M = 1$
általánosan (n a számrendszer alapja): $0 <= M = 1/n$)

A karaktertisztika, amely mindig egész szám, a tizedesvessző eredeti helyéhez képest az eltolási lépések számát jelenti, előjele pedig megadja az eltolás irányát.

32 biten a szokásos ábrázolási forma: 1 bit előjel, 7 bit előjeles karakterisztika, 24 bit mantissza. A karakterisztika kódolása: 7 biten 128 különböző állapot fejezhető ki. Ennek fele az alapérték. A pozitív kitevőt ehhez hozzáadjuk, a negatívat kivonjuk. Így a maximális kitevő 63, a legkisebb -64 lesz.

A számítógépek többsége képes a duplapontosságú számábrázolásra. Itt a mantisszára a 24 bit helyett 56 jut.

1.8 Logikai műveletek

Azokat a kifejezéseket, amelyeknek az a tulajdonsága, hogy vagy igazak, vagy nem ítéleteknek, állításoknak nevezzük. Egy állítás két értéket vehet fel: IGAZ vagy HAMIS. Ezeket nevezzük logikai értékeknek. Jelölni egy állítást az abc első néhány (nagy) betűjével szoktuk. Pl: Vegyünk egy állítást! Ez legyen a következő: Esik az eső. Jelöljük ezt A-val. Ez a kijelentés, vagy igaz lehet, vagy hamis. A logikai állítások szemléltetésére használjuk az igazságtáblát. Ebben egy ítélet értékét többféleképpen is jelölhetjük. Ha az állítás igaz, akkor jelölhetjük 1-gyel, I-vel (igaz), T-vel (true) vagy az IGAZ ill. TRUE szavakkal is; ha hamis, akkor 0-val, H-val F-fel vagy a HAMIS ill. FALSE szavakkal. Az egyszerűség kedvéért én 1-gyel és 0-val fogom jelölni.)

A digitális berendezések, így a számítógép megismeréséhez nélkülözhetetlen a logikai algebra ismerete, melyet megalkotója után BOOLE-algebrának is neveznek.

Minden állításnak létezik tagadása. Előző példában: Nem esik az eső. Jelölése: $\overline{A}$, not(A)

Bármely állítás és tagadása közül csak az egyik igaz.

A tagadás igazságtáblája a következőképpen néz ki:

A	$\overline{A}$
0	1
1	0

(Igazságtábla: Egy-egy logikai állítás könnyebb megértésére, leírására szolgál.)

Vizsgáljuk meg a következő állítást: Süt a nap, vagy meleg van. Ez az összetett állítás két részből tevődik össze. Felbontható két állításra. Egyik: süt a nap; másik: meleg van. Mikor lesz igaz ez az állítás? Akkor, ha a két rész közül legalább az egyik igaz. Azaz:

— vagy süt a nap
— vagy meleg van
— vagy süt a nap és meleg is van.

Hamis csak akkor lesz, ha mind a kettő hamis, azaz ha nem süt a nap, és nincs is meleg.

Vizsgáljuk meg a következő állítást: Süt a nap és meleg van. Ez az összetett állítás két részből tevődik össze. Felbontható két állításra. Egyik: süt a nap; másik: meleg van. Mikor lesz igaz ez az állítás. Csak akkor, ha a két rész közül mindkettő igaz. Hamis viszont akkor már, ha legalább az egyik hamis. Azaz:

— vagy nem süt a nap
— vagy nem esik az eső
— vagy nem süt a nap és nem esik az eső.

Látható, hogy ez utóbbi két esetben (amikor összetett állításról van szó), két állítás egymáshoz való viszonyát kell vizsgálnunk. Az első összetett állítást VAGY (OR,+) kapcsolatnak, az utóbbit ÉS (AND,*) kapcsolatnak nevezzük.

Összefoglalva:

A logikai VAGY kapcsolattal összekapcsolt állítások esetén az eredmény akkor igaz, ha az állítások közül legalább az egyik igaz.

A logikai ÉS művelettel összekapcsolt állítások esetén az eredmény, következtetés akkor és csak akkor igaz, ha az állítások mindegyike igaz. Az ÉS kapcsolatot a szorzás jelöli.

Most pedig írjuk fel ezek igazságtábláit!

A	B	A és B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A vagy B
0	0	0
0	1	1
1	0	1
1	1	1

Természetesen az itt megemlített összetett logikai állításokon kívül létezik még nagyon sok. Ezekkel most nem foglalkozunk.

Vannak olyan állítások, amelyek minden esetben igazak. Ezeket azonosan igaz állításoknak nevezzük és 1-gyel jelöljük. Ehhez hasonlóan léteznek azonosan hamis állítások.

A logikai műveletekre számos azonosság írható fel. Ezek közül néhány a teljesség igénye nélkül.

$1*1=1$ $\quad 1*A=A$ $\quad 1+1=1$ $\quad 1+A=1$
$0*0=0$ $\quad 0*A=0$ $\quad 0+0=0$ $\quad 0+A=A$
$A*A=A$ $\quad A*\overline{A}=0$ $\quad A+A=A$ $\quad A+\overline{A}=1$

kommutativitás (felcserélhetőség): $A*B=B*A$, $A+B=B+A$

asszociativitás (csoportosíthatóság): $ABC=A(BC)=(AB)C$, $A+B+C=(A+B)+C=A+(B+C)$

disztributivitás (felbonthatóság): $A(B+C)=AB+AC$, $A+BC=(A+B)(A+C)$

$\overline{(A+B)}=\overline{A}*\overline{B}$; $\overline{A*B}=\overline{A}+\overline{B}$ (De-Morgan-tételek)

Most hogy már ismerjük a logikai műveleteket és a bináris számokat, kapcsoljuk össze őket. Nézzünk néhány bináris számmal végezhető olyan műveletet, amelyhez szükségünk van a logikai műveletekre is. (A logikai állításoknál két értéket használtunk az elemi logikai műveletek elvégzéséhez. Ezek leírására jól alkalmazható a kettes (bináris) számrendszer.)

Számoljuk ki a következő kifejezés értékét: $133_{(10)}$ OR $87_{(10)}$ (az adatok 8 biten vannak ábrázolva).

A kiszámolás módja a következő: váltsuk át mind két számot kettes számrendszerbe, majd bitenként hozzuk őket OR kapcsolatba.

$133_{(10)}$	=	$10000101_{(2)}$
$87_{(10)}$	=	$01010111_{(2)}$
$133_{(10)}$ OR $87_{(10)}$	=	$11010111_{(2)}$

(1 OR 1 = 1; 1 OR 0 = 1; 0 OR 1 = 1; 0 OR 0 = 0; ugyanúgy, mint a logikai műveleteknél.)

$133_{(10)}$ OR $87_{(10)} = 11010111_{(2)} = 215_{(10)}$

Ehhez hasonlóan az AND művelet bitenkénti AND-et jelent kettes számrendszerbeli számmal végzett művelet esetén. Pl: $133_{(10)}$ AND $87_{(10)}$ = ?

$133_{(10)}$	=	$10000101_{(2)}$
$87_{(10)}$	=	$01010111_{(2)}$
$133_{(10)}$ AND $87_{(10)}$	=	$00000101_{(2)}$

(1 AND 1 = 1; 1 AND 0 = 0; 0 AND 1 = 0; 0 AND 0 = 0; ugyanúgy, mint a logikai műveleteknél.)

$133_{(10)}$ AND $87_{(10)} = 00000101_{(2)} = 5_{(10)}$

1.9 Operátorok, precedencia, adattípusok

Egy kifejezés (pl: 3*2-1) két részből áll:

— operátorok (az elvégzendő műveletek; előző példában: *, -)
— operandusok (azok a számok, amelyekkel a műveleteket el kell végezni; előző példában: 3, 2, 1)

Operátor

Az operátor szó helyett gyakran használják a művelet szót. Azonban az operátorok közé nem csak a szűkebb értelemben vett műveletek (+, -, *, /, ...), hanem az ún. relációk (<, >, =, <=, >=, <>) és a zárójelek is bele tartoznak.

Megjegyzés:

— A műveleteknél a / jel a számítástechnikában az osztást jelöli.
— A relációknál:

<	kisebb
>	nagyobb
<=	kisebb vagy egyenlő
>	nagyobb vagy egyenlő
<>	nem egyenlő

— A zárójelek: ( ) [ ] { }

Vannak olyan operátorok (műveletek), amelyeket szavakkal jelölnek. Pl: and, or, xor, not (lsd: logikai operátorok), div, mod (div: egész osztás; Pl: 7 : 2 = 3 maradék 1. Ebben az esetben 7 div 2 = 3. Másik példa esetén: 14 div 4 = 3. mod: maradék osztás; Előző példáknál maradva: 7 mod 2 = 1 és 14 div 4 = 2. Az osztás maradékát jelöli.)

Precedencia (elsőbbségi) szabály

Az operátorok lényeges tulajdonsága a precedencia (elsőbbség). Ez megszabja az egy kifejezésen belüli különböző operátorok végrehajtási sorrendjét, más szóval prioritását. A nagyobb precedenciájú operátorok

előbb hajtódnak végre, mint a kisebb precedenciájúak. Azonos precedencia esetén pedig a balról jobbra szabály érvényesül (balról jobbra hajtódnak végre).

Precedencia sorrend (precedencia szerint csökkenő sorrendben)

- zárójelek
- egy operandusú operátorok (+,-:ezek nem az összeadást ill. kivonást jelentik, hanem az előjeleket; pl: -5, +2)
- multiplikatív operátorok (*, /, and, div, mod)
- additív operátorok (+, -, or, xor)
- relációk
- balról jobbra szabály

Nézzünk most egy példát:

23 + 12/(16 - 6 * 2) = ?

A zárójelek miatt először a zárójelben lévő kifejezést kell kiértékelni. (A kifejezésben lévő operandusok közül ennek a legnagyobb a precedenciája.) A zárójelben a következő kifejezés található: 16 - 6 * 2. Itt két operandus található a * és a -. A kettő közül a *-nak a nagyobb a precedenciája, ezért előbb a szorzást kell elvégezni. Azaz: 16 - 6 * 2 = 16 - 12 = 4. Tehát a zárójelben lévő műveletek eredménye: 4.

23 + 12/(16 - 6 * 2) = 23 + 12 - 4

Most már csak két művelet van hátra, egy + és egy -. Mivel a precedenciájuk azonos, ezért a balról jobbra szabály érvényesül.

23 + 12 - 4 = 35 - 4 = 31.

Adattípusok

A számítógépen belül egy adat vagy nem változtatja meg az értékét, vagy időről időre új értéket vesz fel. Azt az adatot, amely állandó értékű, konstansnak nevezzük. (Gondoljunk a matematikában használt PI-re.) Azt az adatot, amely változtatja az értékét, változónak nevezzük. A változókra szimbólikus nevekkel hivatkozhatunk. (Szimbólikus nevek, változó nevek: egy vagy több betűből, számból álló jelsorozat.) A változó neveket szabadon választhatjuk meg, de ha már egy változót elneveztünk valahogyan, akkor erre a változóra csak ezzel a névvel hivatkozhatunk.

A számítógépben ábrázolt adatokat két csoportra osztjuk: elemi és összetett típusokra. Az utóbbiaknál az összetétel módja szerint csoportosíthatunk:

- azonos típusúak (tömb, szöveg, verem, sor, lista, szekvenciális, direkt, indexelt)
- különböző típusú részekből összetett (rekord)

Elemi adatok típusai:

- egész: csak egész szám lehet; pl: egy vállalat dolgozóinak létszáma csak egész lehet.
- valós: értéke a matematikában megismert valós szám lehet; pl: egy vállalatnál a dolgozók átlagfizetése.
- logikai: értéke a korábban már megismert logikai értékek lehetnek (igaz, hamis).
- karakter: értéke betű, számjegy és egyéb speciális jel lehet.

Összetett típusok: az elemi adatokból létrehozott adatstruktúrák. Ezek:

A tömb egy olyan sorozat, amelynek elemszáma rögzített és bármely elemére egy indexszel hivatkozhatunk. Legyen egy tömb neve: JEGYEK, és tároljuk benne egy iskolás félévi jegyeit. (Vegyük, hogy a tanuló félévkor 10 jegyet kap.) Az egyes jegyekre a sorszámuk alapján hivatkozhatunk: JEGYEK[1], JEGYEK[2], ..., JEGYEK[10].

A szöveg egy olyan sorozat, amely csak karakterekből áll és amelynek elemszáma változhat. Pl: Egy vállalatnál egy-egy dolgozó nevét tárolhatjuk egy szöveg típusú változóban.

A verem adatok sorozatát tartalmazza, de csak az egyik végét tudjuk kezelni. Gondoljunk egy tényleges veremre. Ebbe a verembe csak felülről tudunk beletenni valamit, és a tetejéről tudunk elvenni valamit. (Emiatt hívják LIFO-nak is az angol rövidítéséből. Amit utoljára tettünk bele, azt vehetjük ki először.)

A sor egy olyan lista, amelynek egyik végére tesszük az elemeket a másik végéről meg elvehetjük. (Hívják FIFO-nak is. Amit először tettünk bele azt vehetjük ki először.

A lista egy olyan szerkezet, amelyben minden elem tartalmazza a következő elem azonosítóját.

A rekord olyan adatszerkezet, amely különböző adatokat tartalmaz. Pl: egy vállalat nyilvántartásában szerepelnie kell egy dolgozónál a nevének, beosztásának, fizetésének, stb. Ezt rekordban szokás nyilvántartani. A rekordnak adunk egy nevet, és az egyes különböző típusú részeinek (mezőinek) szintén. A rekord mezői közül az(oka)t a mező(ke)t, amely alapján a rekordot azonosítjuk kulcsnak hívjuk. Pl: egy telefonkönyvet vizsgálva a név lehet kulcs, ez alapján tudunk keresni egy telefonszámot.

Szükség van az adatok feldolgozása, nyilvántartása során arra, hogy tároljuk is a számítógépbe vitt adatokat. Erre szolgálnak az ún. file-ok, állományok. Egy állományon belül egy-egy adat általában rekordban van tárolva. Tehát a file-t vehetjük rekordok sorozatának is.

A szekvenciális állományból vagy csak olvashatunk, vagy csak írhatunk bele. Az adatok nyilvántartása a bevitel sorrendjében történik.

A direkt állományba írhatunk is és olvashatunk is belőle, az adatok elérése tetszőleges sorrendben történhet.

Indexelt álományokhoz egy táblázat tartozik, amely az adatok egy tetszőleges szempont szerinti sorrendjét tartalmazza. A hivatkozások e táblázat alapján történnek. Pl: térjünk vissza ahhoz a példánkhoz, ahol egy vállalat dolgozóinak adatait kellett nyilvántartanunk. (azonosító szám, név, beosztás, fizetés, lakcím, születési adatok, stb.) Ehhez készítünk egy táblázatot, amely a vállalat dolgozóinak azonosítóit tartalmazza, és egy jelzést, amely jelzi, hogy az adott dolgozó hol található az adatállományunkban. Sokkal könnyebb ezt a táblázatot kezelni, mint az egész adatállományt, minden egyes mezőjével együtt. Könnyebb rendezni, és könnyebb ez alapján megkeresni egy adott kódú dolgozót. Ha a neveket tároljuk el egy táblázatba, akkor a nevek alapján tudjuk könnyen kezelni az adatainkat. Ezt a táblázatot hívjuk indextáblának.

1.10 Programozási alapfogalmak

Program utasítások sorozata, amely egy előre meghatározott feladatot hajt végre.

Gépi szintű, assembly, magasszintű programozás

Programozási nyelvnek azt a jelölésrendszert nevezzük, amely felhasználásával a feladat végrehajtásához szükséges lépéseket, azaz magát a programot leírjuk. Főbb típusok a következők:

A gépi kód

A számítógép anyanyelve. Csak ezen a nyelven írt utasításokat képes értelmezni és végrehajtani. A műveleti kódokat numerikus formában kell megadni (kettes vagy tizenhatos számrendszerben).

Az egyes műveleteket elemi lépésekre kell bontani. Ismerni kell a központi tár pontos felosztását. A gépi kódot csak ott érdemes alkalmazni, ahol ténylegesen nagyobb sebességet tudunk elérni és ahol a memóriával való takarékosság különösen fontos.

Az assembly nyelv

Az assembly nyelv utasításai 3 részre oszthatók: **cím, utasításkód, operandus** (a negyedik részben magyarázó megjegyzéseket szoktak elhelyezni a felhasználók számára). A cím arra a memória rekeszre utal, amelyben az adott utasítás található. Az utasításkód helyén az ún. mnemonikok állnak. Ezek a nyelv alapszavainak rövid, könnyen megjegyezhető formái. Az operandus tartalmazza a művelet elvégzéséhez szükséges adatot vagy címet. Az assembler: az a program, amely egy assembly-ben írt programot lefordítsa a gép számára érthető formára.

Magasszintű programnyelvek

A gépfüggetlen, univerzális nyelvek kialakítása egyszerűsítette le lényegesen a felhasználói munkát. Az így megírt program minden olyan számítógépen futtatható, amely rendelkezik az adott nyelv fordítóprogramjával (**compiler**), vagy értelmezőjével (**interpreter**).

Compiler (fordító program): az a program, amely egy magasszintű programozási nyelven írt programot lefordítja a gép számára érthető formára.

Interpreter (értelmező program): az a program, amely egy magasszintű programozási nyelven írt programot értelmez a gép számára.

Magasszintű programozási nyelvekre néhány példa: PASCAL, C, MODULA, FORTRAM, ALGOL, BASIC, stb...

A feladat megfogalmazása, elemzése

A feladat megfogalmazása mindig legyen egyértelmű és pontos, teljes, rövid és tömör, szemléletes és érthető, formalizált és tagolt formájú. Célszerű írásban rögzíteni a feladatot. Ez tartalmazza az összes bemenő adatot, az azokon végzendő műveleteket és a kívánt kimenő adatokat is.

A feladatot bontsuk fel részekre, szintekre. Minden szinten egyes jól elhatárolható részfeladatokat különíthetünk el. A részek bemenő és kimenő adataikkal kapcsolódnak egymáshoz. Ezeket a kapcsolatokat egyértelműen meghatározhatjuk. A következő szinten, szinteken a részfeladatokat újabb részekre bontjuk és az eljárást megismételjük egészen addig, míg az elemi részek szintjéig jutunk. A részek közötti műveletekre néhány általános tétel.

A párhuzamos finomítás elve

Az egyes részek bővítését, finomítását mindig az adott szint valamennyi tagján végezzük el!

A döntések elhalasztásának elve

Egyszerre csak kevés, de következetesen végrehajtott döntést hozzunk! Elegendő információ hiányában halasszuk el a döntést!

A visszalépés elve

Ha egy szinten megakadtunk és a továbblépés kétséges, egy szintet, vagy ha kell többet is menjünk vissza és keressük meg elakadásunk okát!

A nyílt rendszerű felépítés elve

Lehetőleg ne az aktuális feladatra készítsünk programot, hanem az azt magában foglaló feladatkörre!

A döntések kimondásának elve

Az algoritmus felállítása során nyilvánvaló tényeket is rögzíteni kell! Pl.: egy adat nulla értékű is lehet.

Az adatok elszigetelésének elve

Az adatok csak az előre meghatározott módon kapcsolódhatnak egymáshoz. Az adatokat el kell különíteni egymástól, az illegális kapcsolatokat meg kell akadályozni! Az adatokat a programban betöltött szerepük alapján lehet osztályozni. Beszélhetünk közös (globális) és helyi (lokális) adatokról. Az előbbiek a program minden része által használhatók, az utóbbiak csak az adott részen belül érhetők el!

A feladatok részekre bontásának három féle útja különíthető el.

- Elemi tevékenységek egymás utáni sorozatát alkalmazzuk (neve: szekvencia, felsorolás).
- Egy feltételtől függően egyik, vagy másik tevékenységet hajtjuk végre (elágazás, választás).
- Elemi tevékenységeket sokszor ismétlünk (ciklus).

1.11 Algoritmus, folyamatábra készítése

Az algoritmus olyan eljárás, amely egy nem okvetlenül matematikai feladat véges sok elemi lépésben való megoldására irányul. Ha egy algoritmus már a számítógép által értelmezhető lépésekből áll, akkor nevezzük programnak. Az algoritmusok leírására szolgáló szabályokat utasításoknak nevezzük.

A program megtervezéséhez, az algoritmus egyes lépéseinek rögzítéséhez több általános és jól használható módszer létezik.

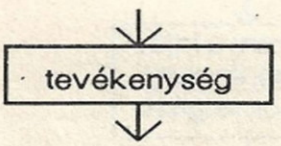
— Mondatszerű leírás. Az algoritmus lépéseit mondatokkal írjuk le.
— Folyamatábra. Ennél az egyes elemi műveleteket geometriai alakzatokkal szemléltetjük, a műveletek sorrendjét, folyamatát pedig nyilazott vonalak jelölik.
— Struktogram. Ez a mondatszerű leírás és a folyamatábra keveréke. Az algoritmust egy téglalapba írjuk, amelyet további téglalapokra osztunk fel, és ezekbe írjuk a végrehajtandó utasításokat.

Folyamatábra

Először nézzük meg azokat a geometriai alakzatokat, amelyeket egy folyamatábrában használunk, és elnevezésüket. A műveletek végrehajtásának irányát az alakzatokat összekötő vonalak, ill. az azokon szereplő nyílak mutatják.

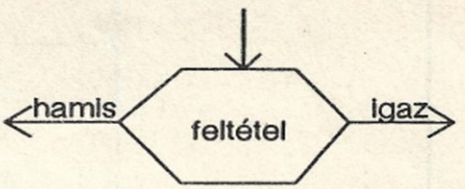
Általános művelet

Az elvégzendő tevékenységet írjuk a téglalapba.



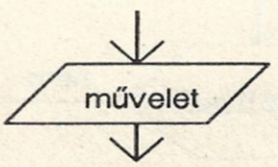
Elágazás

Ábrázolására egy hatszöget használunk. Ebbe egy feltételt írunk, amelytől függően folytatódik a program végrahajtása.



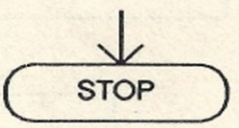
Input/output műveleti blokk

Paralelogrammával jelöljük, feltüntetve az alkalmazott változó nevét, valamint az adatáram irányát.



Határszimbólum

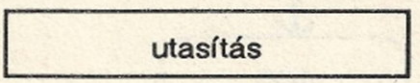
A program elejét és végét jelzi az itt lévő ábra. START felirat kerül a program elejére, ebből csak kifelé mutat a nyíl; ill. STOP felirat kerül a program végére, ide csak beérkezhet nyíl.



Struktogram

Általános művelet, értékadás

Egy téglalapba írjuk be a végrehajtandó műveleteket.

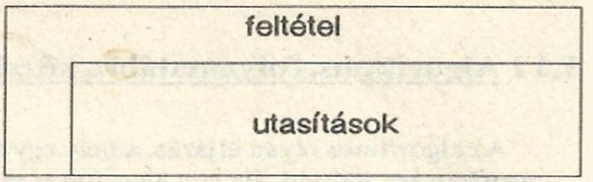


Elágazás:

A program a feltételtől függően hajtja végre a jobb, vagy a baloldali ágat.

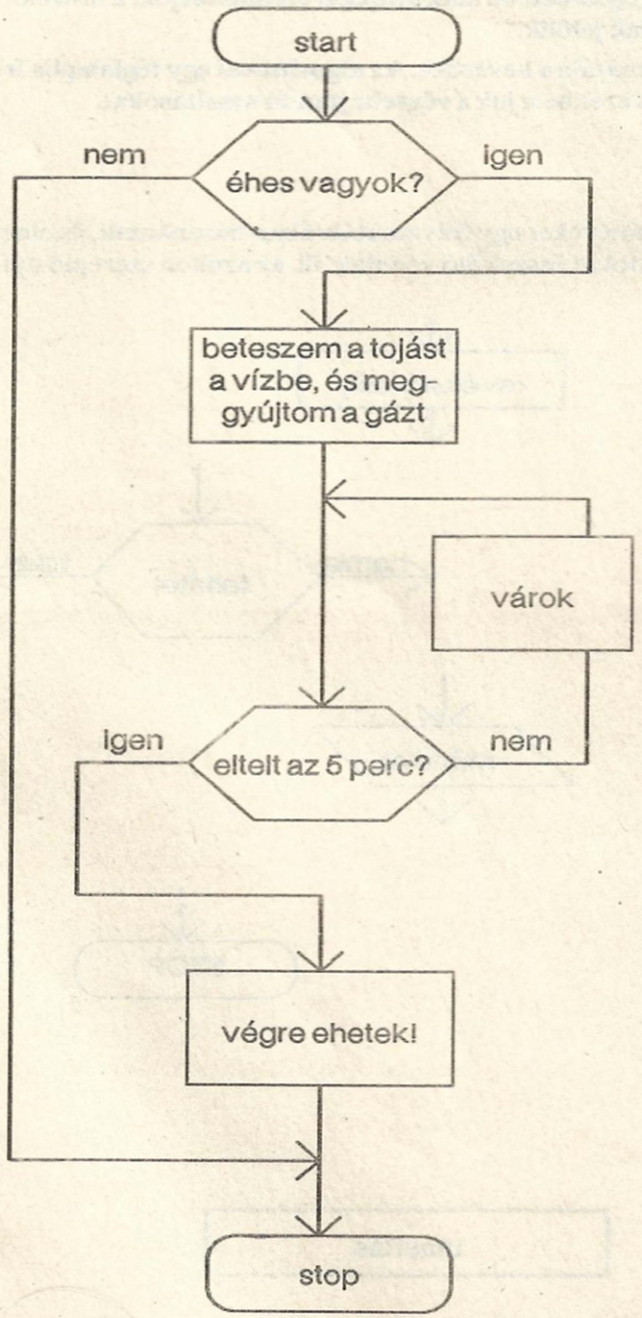
Ciklus

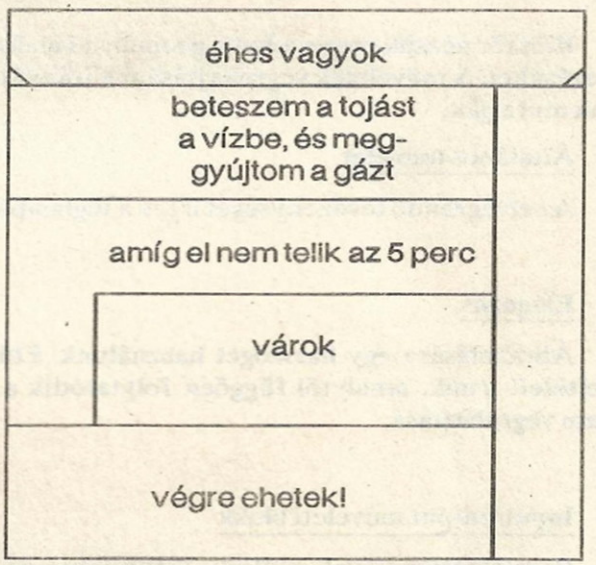
Addig hajtja végre az utasításokat, amíg a feltétel igaz. Ha a feltétel hamissá válik, akkor kilép.



Most nézzünk egy konkrét példát, amit elkészítünk folyamatábra, és struktogram segítségével is.

Írjuk le a tojáskészítés folyamatábráját és struktogramját!





Nézzünk néhány olyan algoritmust, amely a későbbiekben hasznos lehet:

Minimum kereséses rendezés

Vegyük egy vállalat dolgozóit. Adottak a dolgozók nevei egy-egy cédulán, és szeretnénk névsort készíteni, azaz szeretnénk a cédulákat névsor szerint sorrendbe rakni. Hogyan tudjuk ezt megoldani? Az első módja legyen a következő: nézzük végig az összes meglévő cédulánkat, és tegyük a névsorban legkisebbet a legelső helyre. Mivel a névsorban legkisebbet választottuk ki, ezért biztos, hogy ő már a helyére került. Most keressük meg a névsorban második helyen lévőt. Tegyük őt a második helyre, stb. Addig, amíg a cédulák végére nem érünk. Ha a végére értünk, akkor az azt jelenti, hogy minden cédulát a helyére tettünk, azaz a nevek névsorban abc szerint rendezetten szerepelnek.

Szomszédcserés rendezés

Nézzük újra az előző példát. Legyen adott néhány név, és abc szerint sorrendbe kell őket rendezni. Most vegyünk ennek megoldására egy másik módszert. Vegyük először a rendezetlen cédulákból az első két nevet. Hasonlítsuk össze őket. Ha rossz sorrendben vannak, azaz az van előrébb, amelyik nagyobb, akkor fel kell őket cserélni, és ez után lépünk tovább. Ha abc szerint jó sorrendben vannak, azaz a két cédula közül az van előrébb, amelyik abc-ben előbb van, akkor tovább lépünk. Összehasonlítjuk a másodikat és a harmadikat a nevek közül. Ha a nevek végére értünk, akkor az abc szerinti legnagyobb elem biztosan a végén van. (Minden összehasonlításkor a nagyobbat teszük hátra.) Ekkor újra kezdjük elölről a nevek hasonlítását. Legfeljebb annyiszor kell elejétől kezdve hasonlítani a neveket, ahány nevünk van. (Mivel minden kezdéskor legalább egy név a helyére kerül, ezért legkésőbb ha elemszámszor végig megyek a neveken, akkor minden a helyére kerül.)

Bináris keresés

Most már le tudunk rendezni egy névsort, de hogyan tudunk rendezett adatok között keresni. Vegyük a rendezett neveink közül azt amelyik középen található. (Ha 9 nevünk van, akkor vegyük az 5-dik nevet.) Mivel az adataink rendezettek, ezért ha a középső nevet összehasonlítjuk a keresett névvel, akkor meg tudjuk azt állapítani, hogy a keresett név a középsőtől előre, vagy hátra található-e. Ezután a neveknek azt a részét kell tovább vizsgálnunk, ahol a név található. Azt a részt felezzük tovább, stb. Így előbb-utóbb elérünk a keresett elemig, ha az a nevek között található. Ha elérünk odáig, hogy már csak egy elemünk van, és az nem egyezik a keresett elemmel, akkor az nincs a nevek között.

(Természetesen ezek az eljárások nem csak nevekre, hanem bármely más adatra is használhatóak. Feladat: Írjuk fel a most ismertetett eljárások folyamatábráját és struktogramját.)

1.12 Programok kódolása, tesztelése, dokumentálása

A kódolás a megfelelő programnyelv kiválasztásával kezdődik. A felhasznált nyelv illeszkedjen a feladathoz és az alkalmazott számítógéphez. Egyes nyelvek az alulról felfelé való kódolást támogatják (LOGO, FORTH), más nyelvek a felülről lefelé valót (BASIC). A programozás során bizonyos gyakran ismétlődő feladatok megoldására célszerű az ún. rutinkönyvtárat kialakítani. Igy a különböző feladatok viszonylag gyorsan összeállítható elemekből készíthetők el.

Egy program jóságáról csak a programhelyesség bizonyításával győződhetünk meg. Ez a feladat meglehetősen bonyolult, összetett, ezért a gyakorlatban inkább a programok teszteléses vizsgálata terjedt el. A tesztelés nem garantálja a program hibátlanságát, csupán valószínűsíti azt. Két módszer ismeretes: a statikus és a dinamikus tesztelés.

Statikus tesztelés

Ennél az eljárásnál a program szövegét vizsgáljuk. A kódellenőrzés során a programot az algoritmussal hasonlítjuk össze.

Dinamikus tesztelés

A programot különböző tesztadatokkal futtatjuk. Ilyenkor a szélsőséges eseteket is ki kell próbálni. Bár mindig vannak olyan esetek amire a programozó nem tud felkészülni, ez a módszer viszonylag megbízható eredményt ad. Két fő osztálya van: a fekete és a fehér doboz módszer. Az elsőnél a programot nem nézzük, a bemenő és a kimenő adatok alapján jutunk eredményre. A második módszer a program logikája, szerkezete alapján próbál eredményt elérni.

A tesztelés során felderített hibákat a programban meg kell találni. Ezt a tevékenységet hibakeresésnek nevezzük. Módszerei:

Indukciós módszer szerint rendszerezzük a tesztadatokat és keressük a megoldást. Dedukciós módszernél a hiba lehetséges okainak a körét szűkítjük. Visszalépéses módszernél a hiba megjelenésétől indulunk vissza.

A program dokumentálása

A kész programhoz kétféle dokumentáció készülhet: felhasználói és fejlesztői. A felhasználói dokumentáció olyan információkat tartalmaz, amelyek a program működtetéséhez nélkülözhetetlenek.

Tartalmazza a feladat megfogalmazását, a program nyelvét, nyelvjárását, a futtatáshoz szükséges géptípust, konfigurációt, a program betöltését, indítását, a program használatát, a hibajelzéseket, javítási módokat, egy tipikus futtatás leírását, a fejlesztési lehetőségeket.

A fejlesztői dokumentáció a program készítőjének szól és a fejlesztéssel párhuzamosan készül. Részei: a feladat megfogalmazása, általánosítása, az algoritmus részletes leírása, a gépi és nyelvi igények, a változótábla, a szintek, rutinok kezdő sorszámai, fejlesztési lehetőségek, a program teljes listája és egy példánya.

1.13 Programok mikroszámítógépeken

Az operációs rendszerek

A mai modern számítógépek nem létezhetnek operációs rendszerek nélkül, mely rendszerek általában jól felhasználható programok. A számítógépek bekapcsolásakor általában ezek a programok elkezdenek dolgozni és "csendesen" a háttérben maradva működnek a gép kikapcsolásáig. Az operációs rendszerrel akkor kerülünk közvetlen kapcsolatba, amikor a billentyűzetről parancsokat adunk ki, és azokat a rendszer végrehajtja. Az operációs rendszerek feladatai:

- programok betöltése a tárba és elindítása,
- input-output műveletek végzése (billentyűzet olvasása, kiírás a képernyőre ill. nyomtatóra, lemezek írása és olvasása, lemezes adatállományok létrehozása, másolása, törlése stb.),
- a lemezek tartalomjegyzékének (directoryjának) létrehozása,
- a billentyűzetről kiadott parancsok értelmezése és végrehajtása,
- a működés közben fellépett hibák kezelése.

Az IBM PC kompatibilis gépeken többféle operációs rendszerrel találkozhatunk (PC-DOS, MS-DOS, Unix, Xenix, Os-2, Novell stb.).

A DOS szó a Disk Operating System (lemezes operációs rendszer) rövidítése. A PC DOS egyfelhasználós operációs rendszer. A MS-DOS számos hasznos segédprogramot tartalmaz. A fontosabb segédprogramok funkciói:

- file-ok másolása, összehasonlítása, átnevezése, megjelenítése,
- lemezek formázása, másolása, összehasonlítása,
- fix lemezek archíválása floppyra és visszatöltése,
- szöveges fájlok létrehozása és átszerkesztése,
- gépi kódú programok nyomkövetése, hibakeresés,
- file-ok válogatása, rendezése.

Alkalmazói programok

DOS segéd programok: Megkönnyítik a DOS parancsok használatát (Pl.: Norton Commander).

Szövegszekesztők: Dokumentációk elkészítését, szerkesztését, megkönnyítő programok (Pl.:WORD).

Kiadványszerkesztő programok (Desktop Publishing): Ezek a programok nyomdai minőségű kiadványok szerkesztését teszik lehetővé (Pl.: VENTURA).

Adatbázis kezelő programok: Ezek a programok az irodai adatfeldolgozói, könyvelői munkát hivatottak könnyebbé tenni (Pl.: DBASE, CLIPPER, FOXBASE).

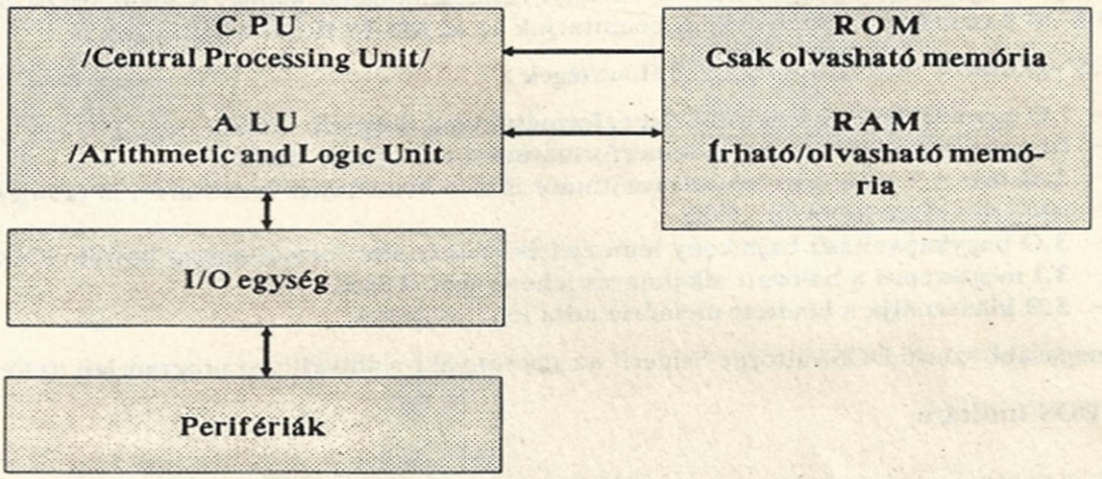
Rajzoló, tervezői programok: A mérnöki tervező munkában jól használhatóak (Pl.: AutoCAD).


Gépkezelési ismeretek
MS-DOS

2. GÉPKEZELÉS, OPERÁCIÓS RENDSZER ISMERETEK

2. 1 Számítógépek hardware elemei

A mikroszámítógépek rendszertechnikai egységeire utalva foglaljuk össze a személyi számítógépek / PC -k / felépítését, fontosabb egységeit és azok kapcsolódását az alábbi ábra alapján:



CPU /Central Processing Unit/ a számítógép központi egysége.

ALU /Arithmetic and Logic Unit/ aritmetikai és logikai műveleteket végzi.

ROM /Read Only Memory/ csak olvasható tár, melyben lévő programokat a gyártó "égette" be. A gép kikapcsolása után is megtartja tartalmát.

RAM írható és olvasható tár, mely tartalmát a gép kikapcsolása /áramkimaradás/ után elveszti.

Interface olyan elektronikai egység, mely összekapcsolja a különböző készülékeket.

Perifériák a számítógéphez kapcsolt egységek. Feladatuk adatok megjelenítése, tárolása, bevitele (monitor, nyomtató, plotter, billentyűzet, "egér" /mouse/, botkormány /joystick/, fényceruca, digitalizáló tábla, lemezmeghajtók /floppy, Winchester, szalagos tárolók, streamerek/, egyéb eszközök).

2. 2 Az operációs rendszer

Az operációs rendszer /OPERATING SYSTEM/ maga is egy program, melynek célja a számítógép erőforrásainak /hardware/ kezelése és irányítása.

2.2.1 DOS és feladatai

A DOS lemezorientált operációs rendszer /DISK OPERATING SYSTEM/, mely az alábbi alapvető feladatokat látja el :

- programok mozgatása és futtatása,
- I/O írási és olvasási műveletek végrehajtása (klaviatúrafigyelés, kiírás képernyőre, file-ok létrehozása stb.),
- a számítógépnek adott parancsok értelmezése és végrehajtása,
- hibák kezelése.

DOS a fenti feladatokat megvalósító programjainak egy részét állandóan a memóriában helyezi el, ezeket nevezzük **rezidens** /belső/ parancsoknak. **Tranziens** /külső/ parancsokat végrehajtó programokat inditásuk előtt lemezről kell a memóriába betölteni, majd végrehajtásuk után onnét törölni. A gyakrabban használt programokat /parancsokat/ a DOS állandóan a tárban tartja ezáltal gyorsítva az egész rendszer működését.

2.2.2 DOS verziók

Személyi számítógépeknél különböző DOS **verziókat** találhatunk. Ezek közül leggyakrabban az IBM PC-DOS-sal, a Microsoft MS-DOS -sal, DR-DOS-sal találkozhatunk.

A számítógépek elektronikai elemeinek fejlődése maga után vonja az adott hardware-t vezérlő operációs rendszerek fejlesztését.

A "fejlettségi fokot" az egyre nagyobb értékű verziószámokkal jelölik. A jelen anyag keretében az MS-DOS 3.3 -at használjuk, de helyenként bemutatjuk az az MS-DOS 5.0 lehetőségeit is.

DOS változatok közötti fontosabb különbségek :

- **1.O** egyoldalas és 8 szektoros lemezformátummal dolgozik -1981-
- **1.1** kétoldalas és 8 szektoros lemezformátumot támogatja -1982-
- **2.O** már a 9 szektoros lemezformátumot is és a Winchester hardware-t is támogatja, DOS készlete jelentősen kibővült -1983-
- **3.O** nagykapacitású hajlékony lemezzel és Winchester formátummal bővült /1984/
- **3.1** megteremti a hálózati alkalmazás lehetőségét /1985/.
- **5.0** kihasználja a bővített memória adta lehetőségeket

A magasabb szintű DOS változat "ismeri" az alacsonyabb szintváltozat programjait /felfelé kompatibilis/.

2.2.3 A DOS indítása

Számítógépünk bekapcsolás után a ROM BIOS FFFF:O ciméről indul. A ROM BIOS a "csak olvasható" /ROM/ tárnak az a része, melyet a gép működési ideje alatt az I/O műveletek vezérlésére "állandóan" használ. Ezután a program ellenőrzi a számítógép hardware elemeit, majd átadja a vezérlést az A: lemez betöltőrekordjának, ha nincs lemez az A: meghajtóban akkor általában a C: Wichester-t keresi meg. Ez ellenőrzi, hogy a lemezen vannak-e a DOS rendszerprogramjai az IBMBIO.COM /MS-DOS -nál IO.SYS/ és az IBMDOS.COM /MSDOS.SYS/, ha ezeket megtalálta akkor a COMMAND.COM parancsértelmező programjával együtt betölti a fenti programokat.

Az IBMBIO.COM a ROM-BIOS kiterjesztését tartalmazza. Ezek után a rendszerünk már "üzemképes". Egyedi paraméterek megadására /konfigurálásra/ ad lehetőséget a CONFIG.SYS nevű állomány, mely szintén automatikus betöltésre kerül. Itt megadható adatok közül a

```
BUFFERS=2O
FILES=32
```

megadása szükséges az adatbáziskezelők működtetéséhez. Alapértelmezés a

- BUFFERS -re XT-nél 2, AT-nál 3,
- a FILES -re 8.

Ezután kerül betöltésre és "futtatásra" az AUTOEXEC.BAT file. Ez a program tartalmazhat bármilyen érvényes DOS parancsot, melyre a számítógépünk bekapcsolása után mindig szükségünk van, de a DOS automatikusan nem biztosítja.

Összefoglalva az indítás lépései :

1. BIOS ellenőrzőrutinok végrehajtása.
2. A betöltő program /**BOOTSTRAP**/ memóriába másolja az **IO.SYS**, az **MSDOS.SYS** és a **COMMAND.COM** programokat /**IBMBIO.COM** és **IBMDOS.COM** /.
3. **CONFIG.SYS**-t megléte esetén betölti.
4. Betölti és elindítja az **AUTOEXEC.BAT** file-t.

2.2.4 Állományok és könyvtárak

Az adatokat /programokat/ a lemezen állományokba szervezve tároljuk. Ezeket az állományokat egy-egy állománynév azonosítja, melyhez max. 3 karakter hosszú kiterjesztés kapcsolható. Az állománynév max. 8 karakter hosszú lehet, de nem tartalmazhat DOS parancsok elválasztására használható jeleket /szóköz, ; , : , stb/.

Az állománynevet a kiterjesztéstől "."-tal választjuk el. Pl. : ELSO.BAT jelölésnél az állomány /file/ neve ELSO és a kiterjesztése .BAT. Az azonos állományokat célszerű egy-egy külön "könyvtárban" tárolni.

A könyvtárak elnevezésére azonos szabályok érvényesek mint az állományokra, de kiterjesztésük ".DIR". Könyvtárak /katalógusok/ a **gyökérkönyvtárból** "származnak". Minden könyvtárnak lehetnek alkönyvtárai. A könyvtári bejegyzések száma a "gyökérkönyvtár"-nál nem lehet több mint **112**, az alkönyvtáraknál a bejegyzések száma nincs korlátozva.

A kiterjesztésekre tetszőleges jelölést használhatunk, de a DOS néhány jelölést kiemelten kezel. Az alábbi felsorolás néhány gyakoribb állománynév-kiterjesztést ad meg :

- nevükkel indítható állományok **.BAT, .EXE, .COM** kiterjesztésűek /**NE**-vel a NORTON EDITOR, **FORMAT**-tal a lemezt formázó program indítható/,
- a szövegfeldolgozók **.DOC, TXT, ~**, .BAK** kiterjesztésűeket használnak,
- az adatkezelők adatállományokra **.DBF**-t, a programjaikra **.PRG**-t,
- **.BAS** a BASIC - forrásprogram, a **.PAS** a PASCAL pedig a PASCAL rendszer forrásprogramjának a kiterjesztése,
- ideiglenes munkaállományoknál általában a **.$$$** kiterjesztést alkalmazzák.

A DOS parancsokban globális állományneveket /*/, illetve helyettesítő karaktert /?/ használhatunk. A " ? " az adott pozición tetszőleges karaktert helyettesíthet.

Pl. : DIR ?.BAK parancs kiírja az összes .BAK kiterjesztésű és egy karakter hosszú állománynevű fileokat.

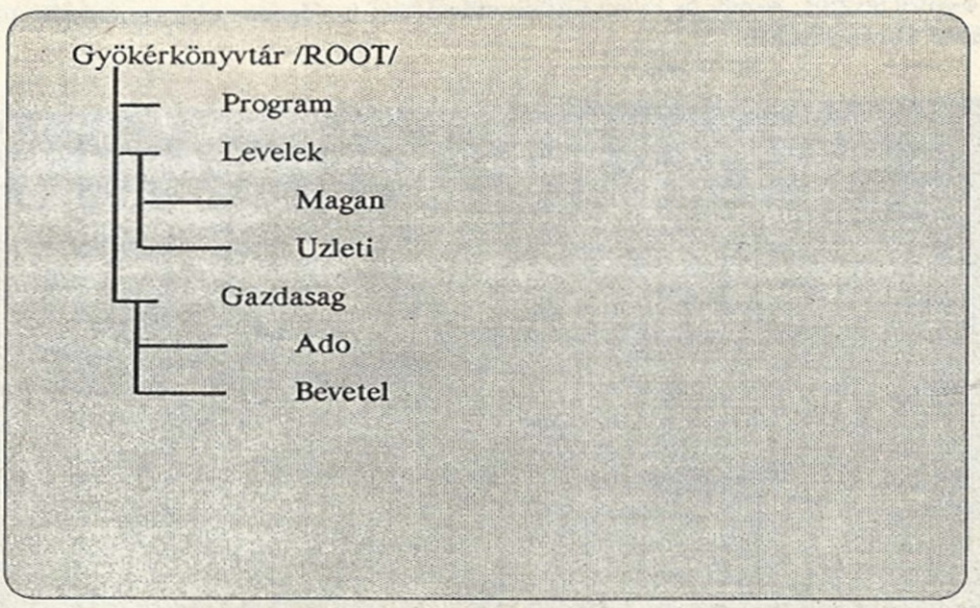
A " * " minden állománynevet /tetszőleges szöveget/ helyettesít.

Pl. : **DEL** a*.* törli az összes "a" betűvel kezdődő állományt

COPY a:*.BAT c:\TANULO*.*

parancs pedig az **A:** meghajtó aktuális könyvtárában lévő összes **.BAT** kiterjesztésű programot az eredeti neveikkel átmásolja a **C:\TANULO** könyvtárába.

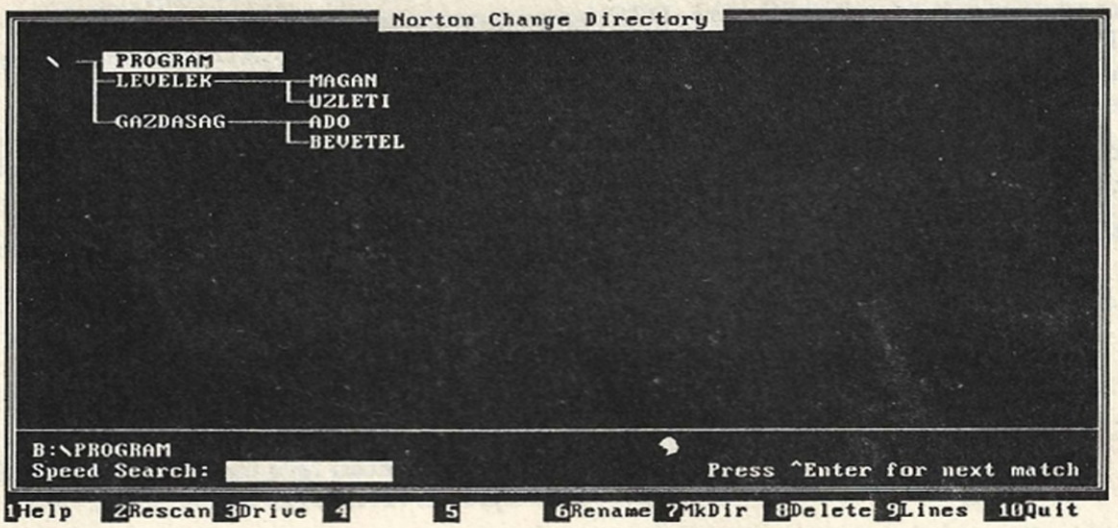
Az alábbi példa egy adott könyvtárstruktúra létrehozásának DOS parancsait mutatja be. A könyvtár felépítése :



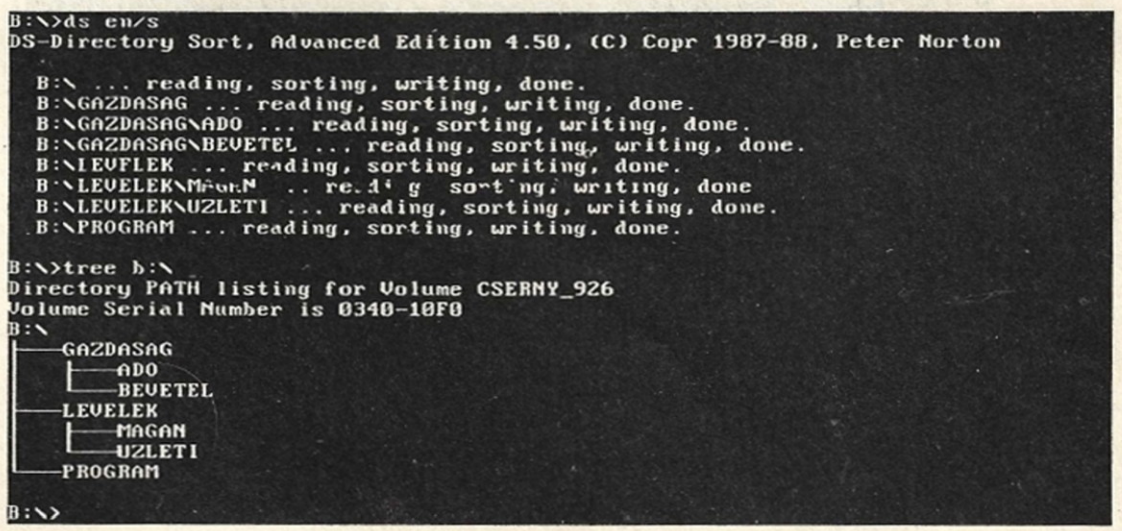
Az alábbi CSINAL.BAT program utasításai az alábbi DOS programokat "futtatva" létrehozzák a fenti struktúrát /könyvtár-szerkezetet/.

**CD **	belép a gyökérkönyvtárba
MD program	létrehozza a "program" könyvtárat
MD levelek	létrehozza a "levelek" könyvtárat
MD gazdasag	létrehozza a "gazdasag" könyvtárat
CD \levelek	belép a "levelek" könyvtárba, mert a "magan" és az "uzleti" innét "nyílik"
MD magan	létrehozza a "magan" könyvtárat
MD uzleti	létrehozza a "uzleti" könyvtárat
CD \gazdasag	belép a "gazdasag" könyvtárba, mert az "ado" és a "bevetel" innét "nyílik"
MD ado	létrehozza az "ado" könyvtárat
MD bevetel	program létrehozza a "bevetel" könyvtárat

A fenti szerkezetet szépen megjeleníti a NCD program, melynek célszerű indítása : **NCD /R.**
A **PC TOOLS** -szal is hasonló eredményt kaphatunk. Nem olyan szép, de áttekinthető képet ad a DOS **TREE** parancsa is.



A fenti szerkezet "rendezés" után. Rendezést DS programmal /Norton rendszer/ az alábbi paraméterekkel végeztem: **e**-kiterjesztés, **n**-név és /**s**-az alkönyvtárakban is végezze el a rendezést. Parancsok és az eredmény az alábbi ábrán látható.



2.3 DOS-parancsok abc sorrendben

Jelölések : [ ] a zárójelek közé írt adatok megadása nem kötelező

d: meghajtó jelölése (a: b: c: d:)

út file elérési útvonal (c:\TANULO\elso.bat)

fn file megnevezése (autoexec)

kit kiterjesztés (.BAT .EXE .COM)

| a jel két oldalán lévő értékek közül csak az egyik megadása szükséges és lehetséges

/? kiírja az adott paracs szintaxisát, paramétereit /csak 5.0 verziótól/

Pl.: DIR /? hatása (ugyanezt eredményezi a *HELP DIR* parancs is)

Displays a list of files and subdirectories in a directory.

DIR [drive:][path][filename] [/P] [/W] [/A[[:]attributes]] [/O[[:]sortorder]] [/S] [/B] [/L]

[drive:][path][filename]
Specifies drive, directory, and/or files to list.

/P Pauses after each screenful of information.

/W Uses wide list format.

/A Displays files with specified attributes.

attributes D Directories R Read-only files
H Hidden files A Files ready for archiving
S System files - Prefix meaning "not"

/O List by files in sorted order.

sortorder N By name (alphabetic) S By size (smallest first)
E By extension (alphabetic) D By date & time (earliest first)
G Group directories first - Prefix to reverse order

/S Displays files in specified directory and all subdirectories.

/B Uses bare format (no heading information or summary).

/L Uses lowercase.

Switches may be preset in the DIRCMD environment variable. Override preset switches by prefixing any switch with - (hyphen)--for example, /-W.

APPEND parancs az aktuális tartalomjegyzéken kivüli file-ok elérését is lehetővé teszi
Típusa : Külső, majd rezidens
Szintaxisa : APPEND d:út[;[d:]út...]

APPEND [/X][/E] /X kiterjeszti az APPEND működését pl. DIR parancs- ra.

/E a megadott utat programkörnyezetben is elhelyezi.

APPEND ; törli az összes útkijelölést

APPEND kiírja az érvényes útkijelöléseket

ASSIGN periféria-hozzárendelések megváltoztatását teszi lehetővé
Típusa : Külső
Szintaxisa : ASSIGN [dO[=d2|...]] a dO meghajtót d2-vé nevezhetjük át.

Pl.: ASSIGN a=c

ATTRIB az archív és irható-olvasható attribútumbit beállítása, lekérdezése
Típusa : Külső
Szintaxisa : ATTRIB [+R|-R][+A|-A][d:][út] fn [.kit]

+R az adott file csak olvasható.

-R az adott file írható is és olvasható is.

+A archív attributum bitet beálíitja

-A archív attributum bitet törli

Paraméterek nélkül az adott file attribútumait írja ki. Globális file-nevek használata megengedett.

Pl.: ATTRIB +R +A ELSO.BAT

BACKUP adott lemez tartalmának biztonsági mentése
Típusa : Külső
Szintaxis : BACKUP d:[út][fn[.kit]] d:[/S][/M][/A][/D:hh-nn-éé] [/P][/T:óó:pp:mm][/L:fn]

/S a kijelölt könyvtár alkönyvtárait is menti.

/M csak az utolsó archíválás utáni file-okat menti.

/A a céllemezen lévő file-ok nem törlődnek.

/D:hh-nn-éé az adott dátum utáni file-okat menti.

/P annyi file-t másol a céllemezre amennyit csak tud. Ez az opció csak MS-DOS alatt érvényes.

/T:óó:pp:mm az adott időpont utáni file-okat menti

/L BACKUP.LOG "napló" file íratható a céllemezre, mely tartalmazza a mentés dátumát és a file-ok neveit.

Pl.: BACKUP c:\tanulo*.BAT a:

A c: meghajtó tanulo könyvtárában lévő BAT file-okat menti az a: meghajtóban lévő lemezre.

BREAK a CTRL-BREAK billentyű figyelésének engedélyezése, tiltása illetve a BREAK állapot lekérdezése
Típusa : Belső
Szintaxisa : BREAK [OFF|ON]

ON állapotban a CTRL-BREAK megszakítást eredményez.

Paraméter nélkül a kapcsoló aktuális állapotát írja ki.
Ha a CONFIG.SYS-ben nincs BREAK parancs akkor BREAK OFF állapot lesz az érvényes.

BUFFERS lemezpufferek beállítása /a CONFIG.SYS file-ba adjuk meg/ Alapértelmezés XT-nél BUFFERS=2 AT-nál BUFFERS=3.
Típusa : Belső /konfigurációs parancs/
Szintaxisa : BUFFERS=xx ahol az xx 1-99 közötti egész szám.

Célszerű legalább 2O puffert megadni.

Pl.: BUFFERS=2O

CHCP billentyűkódlap váltás, mely csak a rendszerben létező és érvényes kód-készletekre vonatkozhat
Típusa : Külső
Szintaxisa : CHCP [xxx]

xxx a kódlap sorszáma

Paraméter nélkül megadva kiírja az aktuális kódlap számát.

Megj.: NLSFUNC program és a MODE CP PREP parancs szükséges a CHCP teljes alkalmazásához.

CHDIR vagy CD a megadott könyvtárba léptet
Típusa : Belső
Szintaxisa : CD [[d:]út] vagy CHDIR [[d:]út]

Pl.: CD a:\alfa\egyik

CHKDSK információt ad a lemezről
Típusa : Külső
Szintaxisa : CHKDSK [d:][fn[.kit]][/F][/V]

/F a fizikai tárolásnak megfelelően felújítja a FAT-ot.

/V file-neveket és elérési uatakat is kijelzi.

Pl.: CHKDSK a:/F

CLS képernyőt törli
Típusa : Belső
Szintaxisa : CLS

COMP két file-t byte-onként összehasonlít, ha nincs eltérés **"Files compare ok."** üzenetet küld.
Típusa : Külső.
Szintaxisa : COMP [d:][út][fn[.kit]] [d:][út][fn[.kit]]

Pl.: COMP c:\tanulo\elso.bat a:\a0\masik.bat

CONFYG.SYS konfigurációs paramétereket tartalmazó file

Pl.: BREAK OFF

BUFFERS=20[

COUNTRY=001

FILES=32

COPY file-okat másol, összefűz vagy billentyűzetről adatokat visz a megadott területre /lemezre, nyomtatóra, file-ba/
Típusa : Belső
Szintaxisa : COPY [/A][/B][d:][út]fn[.kit] [/A][/B][d:]
[fn][.kit][/V] forrás file(-ok) cél file

Az **A** és a **B** paraméter a másolás módját adják meg.

A **V** -nél a DOS ellenőrzi a másolat helyességét.

Pl.: COPY a:\aO*.EXE c:\tanulo*.*

Az a: meghajtó aO könyvtárából az összes EXE kiterjesztésű file-t átmásolja a c: meghajtó TANULO könyvtárába.

COPY alma*.txt osszes.txt

Az aktuális könyvtárban lévő összes alma kezdetű és txt kiterjesztésű file-t összefűzi az osszes.txt nevű file-ba.

COUNTRY nemzeti dátum, idő és egyéb formátum megadása
Típusa : Belső (CONFIG.SYS file-ban adjuk meg)
Szintaxisa : COUNTRY xxx[yyy]

xxx az adott ország telefonkódja

OOl USA ; O6l Ausztrália ; O49 NSZK stb.

yyy az adott ország kódhoz tartozó két kódlap valamelyike.

437 USA ; 86O Portuguese ; 865 Nordic stb.

CTTY alternatív konzol kijelölése
Típusa : Belső
Szintaxisa : CTTY egységazonosító

Egységazonosító lehet AUX, COMl, COM2, COM3, COM4, CON
Eredeti hozzárendelés visszaállítása : CTTY CON

DATE rendszerdátum kiíratása, módosítása
Típusa : Belső
Szintaxisa : DATE [hh-nn-éé]|[nn-hh-éé]|[éé-hh-nn]

nn = 1 .. 31 nap
hh = 1 .. 12 hónap
éé = 8O .. 79 vagy 198O .. 2O79 évszám

DEL állományok törlése / azonos az ERASE paranccsal /
Típusa : Belső
Szintaxisa : DEL [d:][út] fn [.kit]
Pl.: DEL *.BAK törli az aktuális könyvtárból az összes BAK kiterjesztésű file-t.

DIR tartalomjegyzék (könyvtár tartalma) kiíratása
Típusa : Belső
Szintaxisa : DIR [d:][út][fn[.kit]][/P][/W]
/P egy képernyőnyi tartalmat ír ki, majd egy tetszőleges gomb lenyomása után a következőt.

/W rövidített kijelzést ad

Pl.: DIR a:\tanulo\pr*.bat parancs hatására az a: meghajtó tanulo könyvtárából a pr -rel kezdődő nevű és BAT kiterjesztésű file-okat írja ki.

DISKCOMP teljes lemezek összehasonlítása. A vizsgálatot sávonként és oldalanként végzi ezért csak a DISKCOPY -val másolt floppy lemezeknél használható.
Típusa : Külső
Szintaxisa : DISKCOMP [d:][d:][/1][/8]
/1 csak egyik oldal összehasonlítását végzi.

/8 csak 8 szektort vizsgál, akkor is ha azok 9 szektorosak.

DISKCOPY egy lemez teljes másolása egy másik azonos lemezre
Típusa : Külső
Szintaxisa : DISKCOPY [d: [d:]][/1]
/1 csak egyoldalas másolatot készít.

Pl.: DISKCOPY a: a:

DISKCOPY b: a:

ECHO parancsok üzenetek megjelenítése
Típusa : Belső (BATH állományok parancsa)
Szintaxisa : ECHO [OFF | ON | üzenet]
ECHO OFF esetén a DOS parancsok nem íródnak ki a képernyőre.

Egy **PROBA.BAT** file tartalma :

```
ECHO tartalmat ír
ECHO OFF
DIR c:auto*.bat
```

A PROBA file hatására a képernyőn a következő jelenik meg:

```
C:ECHO tartalmat ír
tartalmat ir
AUTOEXEC.BAT
C:>
```

EXE2BIN állományok átalakítása EXE-ről COM formátumra. A program mérete csökken, betöltése gyorsabb lesz. Az átalakítás csak 64 Kbyte-nál kisebb programoknál végezhető el.
Típusa : Külső
Szintaxisa : EXE2BIN [d:][út]fn[.kit] [d:][út]fn[.kit]
Az első paramétercsoporttal az átalakítandó .EXE file-t, a másodikkal a .COM -ot adjuk meg.

FASTOPEN file elérés gyorsítása /csak 3.3O DOS verziótól/
Típusa : Külső
Szintaxisa : FASTOPEN d:[xxx]
xxx file bejegyzések száma amennyit a DOS a memóriában megjegyezhet.
Alapértelmezés 34, maximális 1999 és a minimális érték 11O lehet.
Ha a file-ok száma kevesebb mint az xxx érték akkor, a feldolgozás lényegesen gyorsabb lesz.

FDISK fix lemez kialakítását végzi. Az FDISK.COM program menürendszerű, így a képrnyőre kiírt lehetőségek közül választhatunk.
Típusa : Külső
Szintaxisa : FDISK

FDISK lehetőségei :

1. **Create DOS** Partition DOS partíció létrehozása
2. **Change Active Partition** az aktív partíció kijelölése
3. **Delete DOS Partition** DOS partíció törlése
4. **Display Partition Data** partícióadatok kiíratása
5. **Select Next Fixed Disk Drive** a következő egység kiválasztása

Merevlemezünket az FDISK használata előtt hardwer szinten formázni kell. Ezt a formázást DEBUG program indítása után g1C800:5 begépelésével indíthatjuk el. A további teendőket a BIOS-szal párbeszédes üzemmódban végezhetjük.

FILES egyidejűleg nyitvatartható file-ok száma
Konfigurációs parancs a CONFIG.SYS -ben adjuk meg.
Szintaxisa : FILES1xxx

xxx értéke 8 - 255 között lehet /alapértelmezés 8/
dBASE -nél legalább 24 -nek kell lenni, célszerű 32-48 közötti érték beállítása.

FIND keresés, válogatás. Karaktersorozat keresése ASCII állományban, szövegben.
Típusa : Külső /un. filter parancs/
Szintaxisa : FIND [/V][/C][/N]"karaktersorozat" [d:][út]fn[.kit]

/V azokat a sorokat írja ki, melyekben a "karaktersorozat" előfordul

/C csak a "karaktersorozat" -ot tartalmazó sorok számát írja ki

/N a sort is és a sor számát is kiírja

Globális file-nevek nem használhatók.

Pl.: FIND /V"ECHO" c:\tanulo\masodik.bat parancs hatására kiíródnak azok a sorok, ahol az ECHO szöveg előfordul.

FORMAT lemez formázás
Típusa : Külső
Szintaxisa : FORMAT d:[/1][/4][/8][/t:sáv][/n:szektor][/v][/s]

/s az operációs rendszer file-jait is felírja

/v lemeznév /cimke/ adható meg

/1 egyoldalas formázás

/8 nyolc szektor/sáv -ra formáz

/4 1.2 MB-os lemezegységben 360 KB-os lemezt formázunk

/t:sáv megadható a formázandó sávok száma

/n:szektor egy sávra írható szektorok száma

Pl.: FORMAT a:/s/v/4

FOR ciklusszervezés DOS parancsok ismételt végrehajtására
Típusa : Belső (BATH állományokban használt parancs)
Szintaxisa : FOR változó IN (mondat) DO parancs

"**változó**" formája : %karakter vagy %%karakter ahol a karakter egy betű vagy számjegy

"**mondat**" legalább egy egymástól szóközökkel elválasztott elemből áll

"**parancs**" DOS parancs melynek paramétereiben szerepel a "változó"

A "**változó**" egyenként felveszi a "mondat" elemeit, és ezzel végrehajtja a "parancs"-ot.

FOR %%c IN (elso.bat masodik.bat) **DO TYPE %%**

A fenti parancs hatása azonos a TYPE elso.bat; TYPE masodik.bat parancsokéval.

GRAFTABL grafikai jelek betöltése
Típusa : Külső
Szintaxisa : GRAFTABL [437|860[|863|865|/STATUS]
437 USA /alapértelmezés/ 860 Portugál

863 Kanada /Francia/ 865 Norvég vagy Dán

/STATUS kiírja az aktuális kódlap számát

A parancs kiadása után a DOS mérete kb. 132 Kbyte-tal bővül, így ennyivel csökken a felhasználható memória.

GRAPHICS grafikus nyomtatás beállítása
Típusa : Külső
Szintaxisa : GRAPHICS [nyomtató típusa][/R][/B]
/R csak az kerül nyomtatásra ami a képernyőn fekete

/B csak színes nyomtatásnál van értelme "nyomtató"
COLOR11BM színes nyomtató fekete szalag COLOR41BM színes nyomtató RGB szalag stb.

Ha nem adunk "nyomtató"-t akkor EPSON FX lesz.
A klaviatúra Print-Screen gombjának lenyomása után a képernyő tartalma kirajzolódik a nyomtatóra.

KEYBxx nemzeti billentyűzet kiválasztása
Típusa : Külső
Szintaxisa : KEYBxx
xx lehet UK-angol ; FR-francia ; SP-spanyol

GR-német ; IT-olasz

KEYBxx programból az Alt-Ctrl-Fl gombok lenyomásával térhetünk vissza az amerikai billentyűzetre.

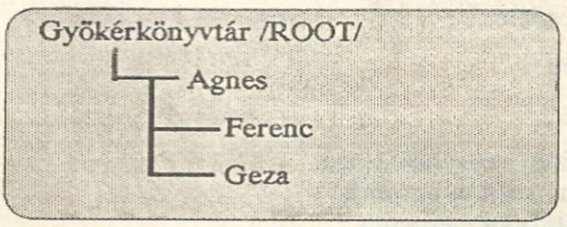
Magyar ékezetes tasztatúra-kezelő program több is létezik. Ilyenek a KEYBHU nevű programok is.

LABEL lemezazonosító cimke (név) létrehozása, módosítása
Típusa : Külső
Szintaxisa : LABEL [d:][cimke]
Ha nem adunk meg cimkét, akkor a DOS kérdést tesz fel :
Delete current volume label (Y/N)?
Az **Y** válaszra törlődik az eddigi azonosító, a **N** válaszra megmarad.

MKDIR vagy MD alkönyvtár létrehozása
Típusa : Belső
Szintaxisa : MKDIR [d:]út vagy MD [d:]út
Az "**út**" utolsó bejegyzése a létrehozandó könyvtár neve.

Egy üres, de formázott lemezen létrehozandó szerkezet :

MODE perifériák üzemmódjának beállítása



Típusa : Külső
Szintaxisa : MODE [paraméterek]
a./ Nyomtatóra : MODE LPT#[:][n][,[m][,P]]

értéke 1, 2 vagy 3 lehet /nyomtató sorszáma/.

n 80 vagy 132 az egy sorba írható karakterek száma.

m 6 vagy 8, sorok száma 1 coll25.4 mm távolságon.

P időtúllépés esetén folyamatosan ismétel.

Pl.: MODE LPT1:132,8,P

b./ Képernyőre : **MODE n** vagy **MODE [n],m[,T]**

n értéke : 40 színes/grafikus adapternél 40 kar/sor
80 színes/grafikus adapternél 80 kar/sor

BW40 fekete/fehér üzemmód 40 kar/sor

BW80 fekete/fehér üzemmód 80 kar/sor

CO40 színes üzemmódra kapcsol 40 kar/sor

CO80 színes üzemmódra kapcsol 80 kar/sor

m értéke : **R** vagy **L** lehet. A képernyőt jobbra /R/ vagy balra /L/ mozgatja.

T teszt-képernyőformát ad a beállításhoz.

Pl.: MODE 40

MORE un. filter parancs, mely képernyőnyi adat átvitele után üzenetet küld, és billentyűlenyomás után folytatódik a kiírás
Típusa : Külső
Szintaxisa : MORE [d:][út]fn[.kit]

Pl.: MORE :\autoexec.bat

De használható az alábbi formában is

DIR |MORE ez azonos hatású a DIR /P paranccsal

PATH keresési út kijelölése, mely lehetővé teszi nem az aktuális könyvtárban lévő file-ok használatát
Típusa : Belső
Szintaxisa : PATH [d:]út[;[[d:]út]...]] vagy
PATH; ez törli az összes PATH-ban megadott utat

Pl.: PATH c:\;c:\DOS;c:\NORTON;a:\TANULO

PATH paraméterek nélküli alakban használva kiírja az előírt keresési utakat.

PAUSE rendszer működésének felfüggesztése egy billentyű lenyomásáig
Típusa : Belső
Szintaxisa : PAUSE [üzenet]

"**üzenet**" max. 121 karakter lehet.

PRINT állományok nyomtatását végzi úgy, hogy közben a számítógéppel más munkát végezhetünk.
Típusa : Külső
Szintaxisa : PRINT [/D:n1][/B:n2][/U:n3][/M:n4][/S:n5][/Q:n6]
[/C][/T][/P][d:][út][fn[.kit]...]

/D:n1 nyomtatásra használható eszköz /LPT1, LPT2, LPT3, PRN, COM1, COM2, AUX/

/B:n2 PRINT bufferének méretét módosítja

/U:n3 várakozási óraciklusok száma (13255)

/M:n4 nyomtatási óraciklusok száma (óraciklus/karakter)

/S:n5 hányszor próbálkozzon a rendszer, amíg úgy dönt, hogy a nyomtató használhatatlan (15255)

/Q:n6 várakozási sor nagysága (1632)

/C megadott file-ok törlése a várakozási sorból

/T várakozási sor minden elemének törlése

/P új file-nevek írhatók a várakozási sorba

A /D, /B, /Q, /S, /U és /M csak a parancs első hívásakor használhatók.

Pl.: PRINT /D:PRN c:\TANULO\elso.bat

PROMPT a prompt karaktersorozat megváltoztatása
Típusa : Belső
Szintaxisa : PROMPT [szöveg] a "szöveg" lesz az új szöveg "szöveg" tartalmazhatja az alábbi speciális karaktereket

$t rendszeridő	**$n** aktuális meghajtó
$d rendszerdátum	**$g** > karakter
$p aktuális meghajtó	**$l** karakter és könyvtár
$v DOS verzió szám	**$e** ESC karakter
$q = karakter	**$b** | karakter
$h BACK SPACE funkció	**$_** cursor új sor elejére lép

REM megjegyzések írhatók BATH file-okba
Típusa : Belső
Szintaxisa : REM [megjegyzés]

RENAME file-ok átnevezése
Típusa : Belső
Szintaxisa : REN[AME] [d:][út]fn[.kit] fn[.kit]
Pl.: REN a:\tanulo\ezt.bat erre.txt

A parancs hatására az EZT.BAT file nevét ERRE.TXT-re változtatja.

RESTORE a BACKUP-pal mentett állományok visszaállítása
Típusa : Külső
Szintaxisa : RESTORE d: [d:][út][fn[.kit]][/S][/P][/B:dátum] [/A:dátum][/E:óó/pp/mm][/L:óó/pp/mm][/M][/N]
/S file-ok valamennyi altartalomjegyzékbe visszatöltődnek

/P visszatöltés előtt üzenetet küld a rendszer

/B a dátum vagy előtte módosított file-okat tölti

/A a dátum vagy utána módosított file-okat tölti

/E az adott időpontban vagy azt megelőzően módositott file-okat tölti vissza

/L az adott időpontban vagy azt követöen módosított file-okat tölti vissza

/M csak az utolsó mentés után módosított file-okat tölti vissza

/N csak a céllemezen nem létező file-okat tölti vissza

Pl.: RESTORE a: c:*.BAT minden *.BAT kiterjesztéssel mentett file visszatöltésre kerül

RESTORE a: c:\S valamennyi mentett file-t visszatölt

RMDIR vagy RD könyvtár törlése (csak üres könyvtár törölhető)
Típusa : Belső
Szintaxisa : RMDIR [d:]út vagy RD [d:]út
Az "út" utolsó eleme a törlendő könyvtár. Pl.: **RM a:\ao\aol\aoll** hatására az **aoll** törlődik.

SET minden végrehajtásra kerülő program előtt a memóriában megtatálhatók a DOS környezet aktuális paraméterei
Típusa : Belső
Szintaxisa : SET [kulcsszóó[paramáter]]
Pl.: SET kiírja az aktuális környezetet

SET PROMPT= törli az adott tartalmat

SET PROMPT=PG létrehozza az adott PROMPT-ot

SHIFT BATH parancssor paramétereit lépteti úgy, hogy azok eggyel kisebb értéket kapnak
Szintaxisa : SHIFT
A parancssorban max. 10 formális paraméter lehet, ha ettől több paraméter szükséges akkor alkalmazzuk a **SHIFT** parancsot.

SI rendszer-paraméterek kiíratása Szintaxisa : SI

SORT állományok rendezett kiírása
Típusa : Külső
Szintaxisa : SORT [/r][/+n] forrás vagy forrás|SORT [/R][/+n]

Pl.: DIR a:|SORT az a lemez tartalomjegyzékét abc szerinti növekvően a képernyőre írja,

SORT :\autoexec.bat az adott file tartalmát abc szerint növekvően a képernyőre írja,

SORT :\autoexec.bat rendbe.txt az autoexec.bat abc szerint növekvő tartalmát a RENDBE.TXT file- ba írja.

SYS rendszerfile-ok másolása adott meghajtóra
Típusa : Külső
Szintaxisa : SYS d:

Akkor alkalmazható ha a céllemez formázott, de üres, vagy ha a /B vagy a /S paraméterrel formázott.

A SYS parancs után át kell másolni a COMMAND.COM file-t is. A SYS az IO.SYS (IBMBIO.COM) és az MSDOS.SYS (IBMDOS.COM) rendszerprogramokat írja a céllemezre úgy, hogy azok a tartalomjegyzék első két bejegyzését képezzék.

TIME rendszeridő lekérdezése, változtatása
Típusa : Külső
Szintaxisa : TIME [óó[:pp[:mm[.xx]]]]

óó óra (0..23)

pp perc (0..59)

mm másodperc (0..59)

xx századmásodperc (0..99)

Érvénytelen paramétereket a rendszer nem fogad el. A rendszeridő a dátumhoz hasonlóan bekerül a tarta- lomjegyzékbe.

TREE a lemez elérési útjainak kiíratása
Típusa : Külső
Szintaxisa : TREE [d:][/F]

/F az összes elérhető file azonosítóját is kijelzi

Pl.: TREE a: kiírás képernyőre
TREE a:PRN kiírás nyomtatóra
TREE a:IDE.TXT kiírás az IDE.TXT file-ba

TYPE file tartalom kiíratása adott perifériára
Típusa : Belső
Szintaxisa : TYPE [d:][út]fn[.kit]

Pl. : TYPE a:\tanulo\elso.bat kiírás képernyőre
TYPE c:\autoexec.bat prn kiírás nyotatóra

VER DOS verziószám kiíratása
Típusa : Belső
Szintaxisa : VER

Megadja a forgalmazót, a főverziószámot és a vátozatot.
Pl. : VER parancs után kiírhatja az alábbi adatokat : IBM DOS Version 3.3

VERIFY a lemezre történő írás utáni ellenőrzést biztosítja
Típusa : Belső
Szintaxisa : VERIFY [ON|OFF]

Az ON kiadása után minden írásművelet után ellenőriz.
Paraméter nékül beírva a kapcsoló állapotát adja meg.

VOL lemezazonosító címkét adja meg
Típusa : Belső
Szintaxisa : VOL [d:]

Pl. : VOL C:

XCOPY file-ok adott szempontok szerinti másolását végzi

Típusa : Külső

Szintaxisa : XCOPY [d:][út][forrás-fn[.kit]] [d:][út][cél-fn[.kit]] [/A][/D][/E][/M][/P][/S][/V][/W]

/A csak az aktiv archív bit-ű file-okat másolja

/D a megadott dátum utáni file-okat másolja

/E az altartalomjegyzékeket is létrehozza

/M azonos az A-val de törli a forrás-fn archív bit-jét

/P másolás előtt megkérdezi, hogy másolja-e : Y-ra másol N-re nem

/S a forrás altartalomjegyzékeit is másolja

/V ellenőrzi a másolást, azonos VERIFY ON-al

/W másolás megkezdése előtt üzenetet küld :

Press any key to begin copying file(s) ezzel lehetővé teszi a szükséges lemezcserét

Globális file-nevek használata megengedett.

Pl. : XCOPY a:*.EXE c:\tanulo*.* parancs hatására az a meghajtó főkönyvtárából az összes .EXE file-t átmásolja a c meghajtó TANULO könyvtárába.

2.4 NORTON EDITOR

A NORTON EDITOR egy könnyen tanulható, jól használható, közepes szintű szövegszerkesztő.

Indítása : NE [+n][input file-név [output file-név]] [opciók]

+n megadja,hogy a file hányadik sorára kerüljön a kurzor,

input fn a betöltendő file teljes neve és elérési útja /ha nem adtuk meg, akkor belépés után fogja kérni/

output fn mentési file neve és tárolási helye /ha nem adjuk meg, akkor azonos az input file-al/.

Az **F1** lenyomása után a képernyőről jól áttekinthető **HELP** (segítő/magyarázó) információkat kapunk, a lapváltás is az **F1** gombbal történik.

Kurzormozgatás

a hagyományos billentyűkkel és azok CTRL kombinációival történik /CTRL lenyomása ^ -al jelezve/

- **HOME** sor elejére ugrik, ^HOME file elejére ugrik
- **END** sor végére ugrik, ^END file végére ugrik
- **PageUp** lapozás vissza, PageDn lapozás előre

Törlés a szövegből

- **Backspace** törlés a kurzortól balra
- **DEL** kurzor alatti jel törlése
- **ALT** és **L** törlés a sor végéig
- **^** és **L** törlés a sor elejéig
- **ALT** és **K** egész sor törlése

File-ok parancsai az **F3** gomb lenyomása után adhatók ki / az alsó információs sorban is láthatók /

- **E** menti a file-t és kilép az editorból
- **S** menti az addig beírtakat, de marad a szerkesztési módban
- **Q** mentés nélkül kilép az editorból, de Y megerősítést kér
- **N** új file töltése /ezután az E menti a file-t, Q nem ment /
- **X** második file betöltése úgy, hogy a képernyőt osztva használjuk, a két rész közötti váltás **F3** és **X** -el történhet
- **A** új file-t olvas és csatol a régihez
- **C** lezárja az output file-t és ezáltal új nyitható.

Blokk parancsok az **F4** gomb lenyomása után adhatók ki /blokk a szövegben két "marker" közötti rész/

- **S** blokk határ kijelölése
- **R** törli a blokk kijelöléseket
- **D** törli a kijelölt blokkot
- **C** blokk másolása a kurzor jelenlegi helyére
- **M** blokk mozgatása a kurzor jelenlegi helyére úgy, hogy a régiről törlődik

Nyomtató parancsok az **F7** gomb lenyomása után adhatók ki

- **P** kinyomtatja az egész puffert /az egész file-t/
- **B** csak a kijelölt blokkot nyomtatja
- **E** lapdobás
- **S** egy oldalra kerülő sorok /laphossz/ száma adható meg, ha nem adjuk meg akkor 62 sor kerül egy-egy lapra
- **M** baloldali margó beállítása

Kereső parancsok az ALT és CTRL gombok segítségével adhatók ki

- **ALT és F** megadott string keresése a file vége irányába
- **^ és F** keresés a file elejéig
- **ALT és C** a már megadott string további keresése a file vége irányába
- **^ és C** további keresés a file kezdete irányába

Ha a keresendő string beírása után, de a keresés megkezdése előtt **ismét** lenyomjuk az **ALT és F** gombokat, akkor egy másik string is megadható. Az így megadott szövegrészeket /stringeket/ a később megadottra cserélhetjük. Minden megtalált string-nél **Y** cseréld **N** ne cseréld válasz adható, de a * gomb lenyomása után minden megtalált string-et kicseréli a később megadottra.

Egyéb parancsok és lehetőségek

- **F9** hatására kilép a DOS-ba, onnét az **EXIT** paranccsal térhetünk vissza az editorba
- **^ és P** után speciális ASCII kódok /pl.: nyomtatást vezérlők/ írhatók be.

2.5 Lokális számítógép hálózatok

Lokális hálózatnak nevezzük az önálló számítógépek /munkaállomások/ összekapcsolásával létrehozott olyan konfigurációt, amelyben a munkaállomások elérhetik a hálózat kijelölt gépein/szerver/ elhelyezett erőforrásokat /adatok, perifériák/ és a hálózatban lévő állomások kábelen kapcsolódnak egymáshoz. Ezt a kapcsolatot egy speciális engedélyezési eljárás irányítja.

A meglévő erőforrások /nyomtatók, tárolók, információk/ jobb kihasználását, gazdaságosabb és hatékonyabb alkalmazását teszik lehetővé a helyi hálózatok. Ezek megbízhatósága magas szintű, működésük gyors, átviteli sebességük elérheti a lO Mb/s -ot is. Egyszerű a hálózatbővítés. Hátrányuk, hogy hatótávolságuk kisebb l km-nél.

A helyi hálózat kialakítását segíti a hálózatokra kidolgoztt szabvány, mely a szabványosítást software úton valósította meg.

A helyi hálózatok működésére a munkaállomások elrendezése /topológia/ és a hálózati elemekhez való hozzáférési jog biztosítása /protokoll/ jellemző és meghatározó.

Munkaállomások telepítése lehet :

- **sín** elrendezésnél egy gerincvezetékre /kábelre/ párhuzamosan kapcsolják a munkaállomásokat, kiépítése olcsó, kábeligénye kicsi, de a kábel meghibásodása az egész rendszert üzemképtelenné teszi,
- **csillag** elrendezésnél minden munkaállomás külön kábellel csatlakozik a központi géphez, kiépítési kábelköltség magas, viszont kábelhiba miatt csak az adott gép esik ki a hálózatból,
- **gyűrű** elrendezésnél a munkaállomásokat egy kábelgyűrűre sorosan telepítik, így a sok csatlakozás miatt a rendszer kevésbé megbízható mint az előzőek,
- **kombinált** elrendezéseknél alhálózatokra bontják a rendszert, így abban különböző megoldások is megvalósulhatnak.

A hálózati erőforrásokhoz való hozzáférési jog biztosítására két jelentős megoldás látható :

- **adási jog** minden munkaállomásnak azonos idejű hozzáférést biztosít a központi erőforrásokhoz, ha egy munkaállomás használni akarja a központi erőforrásokat, akkor előkészíti az "üzenetét" és amikor adási jogot kap akkor továbbítja azt,
- **ütközésfigyelési** megoldásnál a munkaállomás "figyel" és ha szabad a kábel akkor forgalmaz, ha egyszerre többnél is ez történik akkor a rendszer megismételteti a műveletet.

Számítógépek gyártói kialakították a hálózati üzemeltetést támogató operációs rendszert a **NETBIOS**-t /NETWORK BASIC INPUT OUTPUT SYSTEMS/ az alapfokú hálózati be- és kiviteli rendszert.

NETBIOS alapú rendszereknél a hálózat csomópontjai /állomásai/ tárolják és kezelik saját azonosítójukat, a kapcsolatteremtésnél név szerint keresi a partnert és azonosítja azt, keresést mindig a saját azonosítótábláján kezdi, ha ott nem találja akkor megpróbálja megtalálni és felépíteni a külső kapcsolatot.

2.6 Vírusok a számítástechnikában

Elnevezést az ember szervezetében terjedő "társai"-ról kapta. Számítástechnikában az olyan **programokat**, amelyek úgy változtatnak meg "idegen" programokat /pl.: Command.Com-ot/, lemezbejegyzéseket /pl.: FAT táblát/, hogy a rendszer működésében zavarok keletkezzenek **vírusoknak** nevezzük. Jelenleg már ezernél több azoknak az ismert vírus-programoknak a száma, mellyek az IBM PC alapú számítógépeket "fertőzik". Idegen lemezeket is használó számítógépek tulajdonosai sohasem lehetnek biztosak abban, hogy computerük nem válik-e fertőzötté, mert a vírusvédelem csak az ismert /regisztrált/ esetek ellen lehet hatékony.

Vírusok elleni védelem egyik elterjedt rendszerét John McAffe és csoportja készítette. Ez a rendszer több, különböző feladatot végző programból áll /SCAN, CLEAN, VSHIELD, NETSCAN, VALIDATE/. Rendszerről a CHIP IV. évf. 5. számának 104-106 oldalán találnak leírást /Nagy Gábor : SCAN 86/. Alkalmazásnál először a SCAN programmal vizsgáljuk gépünket, ha talált vírust akkor annak azonosítóját kiírja, ennek felhasználásával a CLEAN programmal hatástalaníthatjuk a fertőzést.

Pl.: **SCAN C:\ /M** parancs után kiírja a [Mich] kódot /ez a nálunk elterjedt Mihelangelo vírus/, akkor ennek hatástanalítása a **CLEAN C: [Mich]** paranccsal lehetséges. A [ ] zárójeleket is be kell írni!

A SCAN paraméterek nélül leírást ad annak használatáról :

```
SCAN 8.40B9 Copyright 1989-92 by McAfee Associates.  (408) 988-3832

Usage:      SCAN d1: ... d26: /A /AG /AV d:filename /CHKHI /CG /CV /D
                      /E .xxx .yyy /EXT d:filename /FR /MAINT /MANY
                      /NLZ /NOBREAK /NOEXPIRE /NOMEM /NOPAUSE /SAVE
                      /REPORT d:filename /RG /RV /SP /SUB /BELL /CERTIFY

Examples:       SCAN C:
                SCAN A: B:
                SCAN C:\TEST\*.* D:\ E:\
                SCAN A:TESTFILE.EXE /BELL /MANY

Options are:
                 \  - Scan root directory and boot area only
                /A  - Scan all files, including data, for viruses
               /AG  - Add recovery data to specified files
               /AV  - Add validation codes to specified files
             /BELL  - Ring alarm if virus found
          /CERTIFY  - List files that have not been validated
               /CG  - Check recovery data for specified files
           /CHKHI  - Check memory from 0Kb to 1088Kb
               /CV  - Check validation codes for files
                /D  - Overwrite and delete infected file
             /DATE  - Log date and time of system scan
More? ( H - Help )
```

A CLEAN paraméterek nélkül szintén alkalmazási ismertetést ad :

```
C:\>clean
CLEAN 8.40B9 Copyright 1989-92 by McAfee Associates.  (408) 988-3832

To clean entire disk(s), specify the disk(s) and the virus.
Examples:
          CLEAN C: [virus name]
          CLEAN C: D: [virus name]
          CLEAN A: [virus name]

To clean a single directory, specify the directory and virus.
Examples:
          CLEAN \newstuff [virus]
          CLEAN C:\unknown\things [virus]
          CLEAN A:\ [virus]
          CLEAN . [virus]

To clean a single file, specify the file and the virus.
Examples:
          CLEAN \unknown\prog.com [virus]
          CLEAN A:\unknown.exe [virus]


C:\>
C:\>
```

Tömörített állományoknál /PKZIP, ARJ, stb/ a virusvédelem csak azok kibontása után lesz hatékony.

Más rendszerű, de jól használható a hazai fejlesztésű **SYSDOKI** nevű program. Ezzel a magyar-vírusok jobban "kezelhetők". Ajánlatos a vírus-védelmi programokat megvásárolni, mert így folyamatosan megkapjuk az újabb vírusok elleni védelmi kódokat.

2.7 Ajánlott irodalom

Racskó Péter :
Bevezetés a számítástechnikába
SZÁMALK 1989.

Bóér-Dóra-Fenyő-Seres :
Az IBM PC belső felépítése
LSI 1990.

Dr. Kovács Magda :
Egyszerűen a mikroszámítógépről
LSI 1985.

Szenes-Dr Uri :
Az IBM DOS I-II-III
LSI 1989.

DOS User's Manual
Microsoft 1987.

Peter Norton :
PC-DOS
SZÁMALK 1988.

Peter Norton :
Az IBM PC programozása
Műszaki Könyvkiadó 1990.

Bartha Attila :
NORTON UTILITIES INTEGRATOR EDITOR COMMANDER GUIDE
LSI 1989.

Cseh Kálmán :

IBM PC alapú helyi hálózatok

SZÁMALK 1989.

Orosz Judit :
IBM PC hálózatok áttekintése
LSI 1988.

Cay Weitzman :
Elosztott mini-, és mikroszámítógépes rendszerek
Műszaki Kiadó 1984.

Bartha Attila :
ANTIVÍRUS UTILITIES 5.0
ComputerBooks 1992

Kiss János – Szegedi Imre :

Új víruslélektan

ALAPLAP KÖNYVEK

2.8 Függelék az 1. és 2. fejezethez

2.8.1 Az IBM PC klaviatúrája (billentyűzete)

2.8.2 Kérdések és feladatok az 1. fejezethez

2.8.3 Kérdések és feladatok a 2. fejezethez

2.8.4 Feladatlap a 2. fejezethez

SZÁMALK-infoNET KFT.

H-1115 BUDAPEST Bártfai utca 54
Telefon: 166-9065, fax: 185-0230

A SZÁMALK-infoNET Kft. a SZÁMALK Csoport tagjaként több évtizedes tapasztalattal rendelkezik információs rendszerek és adatbáziso' tervezése, kivitelezése, üzemeltetése valamint számítógépes hálózatok alkalmazása területén.

Jelenlegi tevékenységi körünk:

- Számítógépes hálózati továbbképző 1-1 hetes tanfolyamok szervezése (alapismeretek, hazai és nemzetközi hálózatok, alkalmazások)
- Géptípus váltás esetén teljes rendszerek kulcsrakész áttelepítése (pl. IBM kompatibilis nagygép - DEC (VAX) áttérés esetén teljes termelésirányítási rendszerek áttétele)
- Kis és nagyméretű adatbázisok szervezése, építése, telepítése, karbantartása, üzemeltetése (pl. üzleti adatbázisok)
- Saját fejlesztésű PC-s szoftverek értékesítése (pl. háziorvosi rendszer)
- Külföldi, nagygépes (main-frame) szoftverek értékesítése
- Szoftverfejlesztés megrendelésre (mikro és minigépes környezetben)
- Számítástechnikai szakirodalmi adatbázisunkból rendszeres témafigyelés és szogáltatás floppy lemezen

Pontos, gyors szolgáltatásainkkal, igen kedvező árainkkal várjuk megrendelőinket. Kívánságára szívesen adunk további információt, személyesen, telefonon vagy levélben.

SZÁMALK-infoNET KFT.

H-1115 BUDAPEST Bártfai utca 54
Telefon: 166-9065, fax: 185-0230

2.8.1 Az IBM PC klaviatúrája (billentyűzete)

Billentyű	Hatása
Alt - Ctrl - Del	Rendszer /DOS/ "melegindítása"
Ctrl - Break vagy Ctrl - C	Program megszakítása
Ctrl - NumLock	Lásba megszakítása
Backspace	Karaktertörlés balra
ENTER	Parancs, beírás vége
Ctrl - ENTER	Sortörés
ESC	Sortörés
Insert	Beszúrómód be/kikapcsolás
Delete	Karaktertörlés

Billentyű	Hatása
F1	Egy karakter másolása
F2	Másolása az adott karakterig
F3	Előző sor kiírása
F4	Törlés a megadott karakterig
F5	Előző sort újjal helyettesíti
F6, vagy Ctrl - Z	File-vége jel
Shift (lenyomva)	Betűváltó
Shift - PrintScreen	Képernyő nyomtatás
Ctrl - PrintScreen, vagy Ctrl - P	ECHO be/kikapcsolás

SZTÁV RT SZAKMAI TOVÁBBKÉPZŐ, ÁTKÉPZŐ ÉS VÁLLALKOZÁSTÁMOGATÓ RÉSZVÉNYTÁRSASÁG

Tanulmányainak elvégzése nem a képzés befejezését jelenti, ezért ismereteinek kiegészítésére szíves figyelmébe ajánljuk a SZTÁV RT – mint az ország egyik legnagyobb feinőttképzéssel foglalkozó cége – szolgáltatásait.

Tevékenységünket országos hálózatunkon keresztül az egész országban, jelentős mértékben Budapesten végezzük. Szolgáltatásaink bővítését a hazai és a külföldi piac igényeivel összhangban alakítjuk.

Számítástechnikai továbbképzéseinket a fenti irányelvnek megfelelően úgy alakítottuk ki, hogy a napi gyakorlatban munkájukban felhasználhassák a számítástechnikát.

I. Középfokú szakképesítést nyújtó számítástechnikai tanfolyamaink:

1. *Személyi számítógép szoftverüzemeltető (DOS, NOVELL, adatbáziskezelés, utilityk)*
2. *Számítógépkezelő (operátor)*
3. *Számítógép programozó*

II. Továbbképző tanfolyamaink:

1. *Szövegszerkesztés, táblázatkezelés (WORD, SYMPHONY)*
2. *Programozói tanfolyam (adatbáziskezelők, Turbo Pascal)*
3. *Számítógépes folyószámla megismerése, elsajátítása*
4. *Számítógépes könyvelési rendszer megismertetése*
5. *Számítógépes bérszámfejtés, SZTK*
6. *Értékpapír és osztaléknyilvántartó számítástechnikai rendszer*
7. *Kereskedelmi-bolti-áruházi számítástechnikai rendszer*
8. *PC rendszergazda*
9. *Számítógép a vezetői gyakorlatban*

Várjuk szíves jelentkezésüket levélben, vagy telefonon!

SZTÁV RT
Tanfolyamszervezési Üzletág

Budapest, 1096. Vendel u. 3/b.
Telefon: 133-4900/131
Telefax: 113-7651

2.8.2 Kérdések és feladatok az 1. fejezethez

1/a. Ismertesse **Neumann János** "számítógépének" jellemzőit.

1/b. Mit jelent az, hogy a számítógép órajele **16 MHz**?

1/c. Végezze el a következő műveletet: **(24 AND 8) OR 49**

2/a. Ismertesse az **IBM PC**-k felépítését, főbb rendszertecnikai egységeit.

2/b. Mit jelent az **INTERFACE** (interfész)?

2/c. Végezze el a következő műveletet: **(53 OR 11) AND 254**

3/a. Ismertesse a számítógép-generációkat és jellemzőiket.

3/b. Mit jelent a **ROM**?

3/c. Ábrázolja a **167** decimális számot **BINÁRIS** és **HEXADECIMÁLIS** alakban.

4/a. Ismertesse a számítástechnikában gyakrabban használatos számrendszerek írásjeleit.

4/b. Mit jelent a **RAM**?

4/c. Végezze el az alábbi összeadást **BINÁRIS** számokkal. Az eredményt adja meg **DECIMÁLIS** és **HEXADECIMÁLIS** alakban is.

01010111
01110001

5/a. Ismertesse a **DOS** jelentését és feladatait?

5/b. Mit jelent az **ASCII**?

5/c. Végezze el az alábbi kivonást **BINÁRIS** számokkal. Az eredményt adja meg **DECIMÁLIS** és **HEXADECIMÁLIS** alakban is.

01010111
01110001

6/a. Ismertesse a számítógépek fontosabb perifériáit /adatmegjelenítő, adattároló, adatbeviteli csoportosításban/.

6/b. Mit jelent számítástechnikában a **BIT**, **BYTE** és a **SZÓ**?

6/c. Milyen eredményt kap (**IGAZ** vagy **HAMIS**) az alábbi logikai műveletek elvégzése után

(29 <= (32 .OR. 6)) .AND. ("1971.03.19" < DATUM)

Megj.: A DATUM értéke "1970.11.27"

7/a. Ismertesse a tárgyalt **LOGIKAI** műveleteket (állítások, operátorok).

7/b. Mi a **CPU** és mi a feladata?

7/c. Írja le a **"szomszédcserés"** rendezés ciklusonkénti lépéseit.
Feladat: állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

8/a. Miért előnyös és mikor ésszerű az **INDEX-állományok** használata?

8/b. Mi az **ALU** és mi a feladata?

8/c. Írja le az alábbi feladatnál a **"minimumkeresés"** rendezés ciklusonkénti lépéseit.
Feladat: állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

9/a. Ismertesse a mágneslemezes adattárolók jellemzőit, jelöléseiket és szerkezetüket.

9/b. Mit jelent a **BCD** rövidítés ?

9/c. Készítse el a **"minimumkeresés"** rendezés **folyamatábráját** az alábbi feladathoz.
Feladat : állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

10/a. Programozási nyelvek jellemzői és csoportosításuk.

10/b. Mit jelent a **CO-processzor** jelölés és mi a feladata ?

10/c. Készítse el a **"szomszédcserés"** rendezés **folyamatábráját** az alábbi feladathoz.
Feladat : állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

11/a. Mit jelent a számítástechnikában a **"vírus"**, milyen hatásai lehetnek ?
Ismertesse a védekezési és helyreállítási lehetőségeket.

11/b. Mit jelent az **"assembler"** és az **"assembly"** ?

11/c. Készítse el a **"szomszédcserés"** rendezés **programtervét** az alábbi feladathoz.
Feladat : állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

12/a. Soroljon fel /legalább négy/ adatbeviteli perifériát és azok alkalmazási területét.

12/b. Mit jelent a **COMPILER**, hol és mire alkalmazzák ?

12/c. Készítse el a **"minimumkeresés"** rendezés **programtervét** az alábbi feladathoz.
Feladat : állítsa az alábbi lottószámsort növekvő sorrendbe.

23 . . . . .
76 . . . . .
51 . . . . .
9 . . . . .
17 . . . . .

13/a. Ismertesse a **"komplemens"** és a **"kettes komplemes"** képzésének menetét. Miért alkalmazzák a számítástechnikában ?

13/b. Milyen perifériát jelent a **MONO, CGA, EGA, VGA** jelölés ? Mik a jellemzőik ?

13/c. Mi lesz az adott számítási művelet eredménye ? **3+18/3*(4-40/5)**

14/a. Ismertesse a számítógép "EGÉSZ" típusú számábrázolási módját, a tárolható maximális értéket.

14/b. Mit jelent az **INTERPRETER**, hol és mire használják ?

14/c. Készítse el a növekvően rendezett állományban egy adott érték bináris keresésének folyamatábráját.

15/a. Ismertesse a számítógép **"lebegőpontos"** és a **"fixpontos"** számábrázolásának elvét.

15/b. Számítástechnikában mit jelent a **HARDWARE** és mit a **SOFTWARE** ?

15/c. Adja meg az alábbi állományban a bináris keresés lépéseit,
ha a keresett telefonszám : **"11-679"**.

1	2	3	4	5	6	7	8	9	10
10-003	10-127	10-354	11-092	11-389	11-631	11-679	12-444	13-876	14-922

Megj.: Felső számsor a **rekordsorszámot** jelöli.

16/a. Mit jelentenek az alábbi "árajánlat"-ban szereplő jelölések ? A konfigurációra megadott árat kedvezőnek, vagy túl magasnak találja ?

NEAT 286-16/21 MHz		1 MB RAM
1.2 MB FDD és FDD/HDD vezérlő		
1 párhuzamos	2 soros	1 game port
40 MB HDD	(3.5", 28 MS)	Baby AT ház
14"-os MONO MONITOR		101 gombos klaviatúra
89.000 Ft	1 év garancia	ÁFA nélküli ár

16/b. Milyen perifériákat jelent a **STREAMER, WINCHESTER, FLOPPY** ? Alkalmazásuk szerint jellemezze a fenti egységeket.

16/c. Készítse el az **AND**, az **OR** és az **XOR** igazságtábláit /A és B állításpár/.

17/a. Ismertesse az operációs rendszer feladatait. Milyen **operációs** rendszereket ismer a DOS-on kívül ?

17/b. Melyik lemezt használhatja az **IBM XT** típusú számítógépében ? DS DD, DS HD, SS DD, 2S HD, 1S DD

17/c. Adja meg azt a műveletet, mely az adatkód alapján kiválasztja a **vidékről bejáró férfi vezetők** kartonjait.
Adatkód bitjeinek jelentése:

0. bit	nő	- 1	férfi	- 0
1. bit	beosztott	- 1	vezető	- 0
2. bit	bejáró -	- 1	helybeli	- 0
3. bit	házas	- 1	nem	- 0
4. bit	gyereke van	- 1	nincs	- 0
5. bit	jogostívány van	- 1	nincs	- 0
6. bit	útlevele van	- 1	nincs	- 0
7. bit	nyelvvizsga van	- 1	nincs	- 0

18/a. Ismertesse a számítógép OUTPUT /csak OUTPUT/ perifériáit.

18/b. Milyen kapcsolat van a **BOOLE**-algebra, a **VENN**-diagramm és a számítógép logikai operátorai között ?

18/c. Adja meg a kódszámot az alábbi jellemzőkre : **férfi-beosztott-helybeli-nőtlen-nincs gyereke-jogosítványa van-útlevele nincs- nyelvvizsgája nincs.**
Adatkód bitjeinek jelentése:

0. bit	nő	- 1	férfi	- 0
1. bit	beosztott	- 1	vezető	- 0
2. bit	bejáró	- 1	helybeli	- 0
3. bit	házas	- 1	nem	- 0
4. bit	gyereke van	- 1	nincs	- 0
5. bit	jogostívány van	- 1	nincs	- 0
6. bit	útlevele van	- 1	nincs	- 0
7. bit	nyelvvizsga van	- 1	nincs	-0

19/a. Ismertesse a számítógép **I/O** perifériáit /mindkét műveletre alkalmas egységekkel foglalkozzon/.

19/b. Mit jelent az "elsőbbségi szabály" és a "balról jobbra szabály" (példán is mutassa be)?

19/c. Ábrázolja **VENN**-diagrammal az alábbi feltételnek megfelelőket :
nők-vezetők-házasok
A halmazokat jelentő körök tartalma : nők, vezetők és házasok

20/a Ismertesse az IBM PC számítógépek **rendszerindításának** folyamatát (folyamatábra is).

20/b. Ismertesse a **rezidens** programok jellemzőit és kapcsolatukat a DOS "**belső**" parancsaival?

20/c. Végezze el a **38+27** decimális számok **összeadását BCD** aritmetikával.

2.8.3 Kérdések és feladatok a 2. fejezethez

A feladatokban konkrét könytárakat, állományokat és meghajtókat adtunk meg, ezeket, amennyiben nincsenek, létre kell hozni!!!

1/a Hogyan tudja az A: meghajtó lemezének DOS könyvtárában lévő AUTOEXEC.BAT és AUTOEXEC.BAK szöveges file-okat összemásolni az A: lemez főkönyvtárába AUTOEXEC.BAT névre? Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_01.TXT néven.

1/b Melyik DOS paranccsal tudja lekérdezni az aktuális meghajtót és könyvtárat?

1/c Törölje az A: meghajtóban lévő lemezről két kiválasztott könyvtárat /pl.: Norton és Dos/ !

2/a Hogyan tudja törölni az A: meghajtó lemezének főkönyvtárából a TOROL.COM file-t, DOS könyvtárából az összes file-t? Milyen kérdést tesz fel a gép? Mit válaszol rá? Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_02.TXT néven.

2/b Melyik paranccsal tudja a rendszerfile-okat átmásolni egy másik meghajtó lemezére?

2/c Állítsa be a TANULO könyvtárban lévő legelső file attributumát csak olvasható, archív formára.

3/a Mi a következő hibaüzenet oka, és hogyan hárítja el a hibát?

Non-System disk or disk error.
Replace and strike any key when ready.

Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_03.TXT néven.

3/b Melyik paranccsal tudja megváltoztatni a file-ok attributumait?

3/c Írjon egy FOX.BAT file-t, amely a C: meghajtót jelöli aktuálisnak, belép a C: meghajtó FOXBASE könyvtárába és elindítja az ott lévő FOXPLUS nevű programot.

4/a Hogyan tudja átnevezni az A: meghajtó lemezének főkönyvtárában lévő FORMAT.COM file-t TOROL.COM névre, DOS könyvtárában lévő F betűvel kezdődő .COM kiterjesztésű file-okat .PRG kiterjesztésre? Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_04.TXT néven.

4/b Melyik paranccsal tudja két file tartalmát összehasonlítani?

4/c Másolja át a C: meghajtó TANULO könyvtárából az összes .TXT kiterjesztésű file-t az A: meghajtóban lévő lemez TANULO könyvtárába.

5/a Mi a hatása az alábbi DOS parancsnak és paramétereinek (/s/v/4)?

FORMAT A:/S/V/4

Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_05.TXT néven.

5/b Melyik DOS paranccsal tud winchesterről mentést (biztonsági másolatot) készíteni?

5/c Hasonlítsa össze a kiadott két DS-DD lemez tartalmát!

6/a Hogyan tudja átmásolni a C: meghajtó DOS könyvtárából a COM , EXE , SYS kiterjesztésű file-okat az A: meghajtó lemezének DOS könyvtárába? Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_06.TXT néven.

6/b Melyik DOS paranccsal tud az adott DS-DD jelű lemezről másolatot készíteni egy másikra?

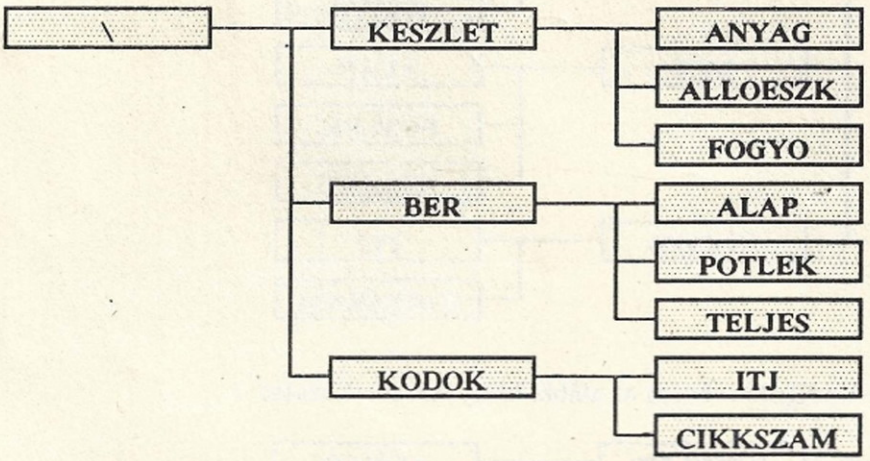
6/c Állítsa be az operációs rendszer dátumát és idejét 1988. május 15. du 4 óra 32 percre!

7/a Melyik DOS paranccsal vizsgálhatunk egy mágneslemezt (kapacitás, hibás területek stb.)? Értelmezze az üzenetet. Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_07.TXT néven.

7/b Melyik paranccsal tudja egy lemez **kötetcímkéjét** lekérdezni?

7/c Írjon egy **INDIT.BAT** file-t, amely a **C:** meghajtót jelöli aktuálisnak, belép a **C:** meghajtó **OKTATO** könyvtárába és elindítja az ott lévő **OKTATO.COM** nevű programot.

8/a Sorolja fel, melyik parancsokkal, milyen sorrendben tudja az alábbi könyvtárszerkezetet **törölni**!



Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_08.TXT néven.

8/b Melyik paranccsal tud file-okat **törölni**?

8/c Állítsa az **A:** meghajtó főkönyvtárában lévő **.COM** kiterjesztésű file-okat csak **olvasható** attributumra!

9/a Mi történik a számítógépben **bekapcsolása után**? Hol keresi először a gép az **operációs** rendszert? Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_09.TXT néven.

9/b Melyik DOS paranccsal tudja **törölni** a képernyő tartalmát ?

9/c Írjon az A: meghajtó lemezére egy **AUTOEXEC.BAT** file-t feltételezve, hogy az **A:** meghajtóban rendszerlemez van, DOS könyvtárában megtalálhatók a DOS külső parancsai, NORTON könyvtárában pedig a NORTON segédprogramok!

10/a Milyen paranccsal lépne át a 8/a-ban megadott könyvtárszerkezet

- **ALAP** könyvtárából a **KODOK**-ba ?
- **BER** könyvtárából a **POTLEK**-ba ?
- **Fő** könyvtárából a **CIKKSZAM**-ba ?
- **ALAP** könyvtárából a **POTLEK**-ba ?
- **ANYAG** könyvtárából az **ITJ**-be ?

Válaszát NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_07.TXT néven.

10/b Melyik paranccsal tudja két azoonos paraméterű lemez tartalmát összehasonlítani?

10/c Formázzon az **1.2** Mbyte-os **A:** meghajtóban egy **DS-DD** lemezt **rendszerlemeznek**!

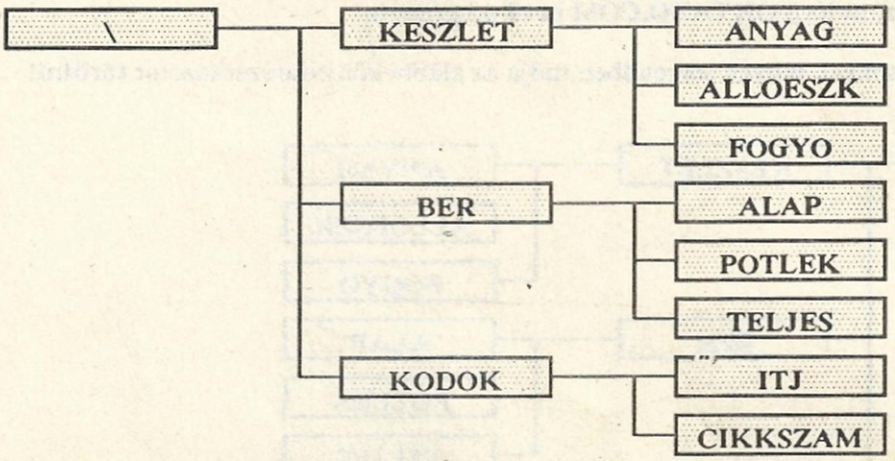
11/a Mi a hatása az alábbi DOS parancsoknak?

```
TYPE autoexec.bat
TYPE autoexec.bat > PRN
```

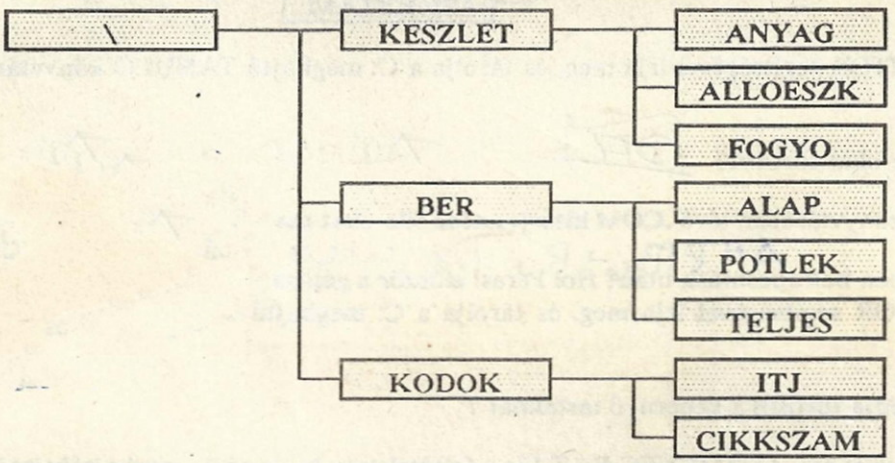
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_11.TXT néven.

11/b Melyik paranccsal tudja az összes aktív archív attribútumú file-t **másolni**?

11/c Norton Commander programmal hozzá létre az A: meghajtó lemezén az alábbi **könyvtárszerkezetet!**



12/a Írjon egy KESZIT.BAT file-t, mely létrehozza az alábbi **könyvtárszerkezetet.**



12/b Melyik paranccsal tudja a BACKUP-pel történt mentést visszatölteni?

12/c Nyomtassa ki a C: meghajtó TANULO könyvtárában található **VIZSG_12.TXT** file-t!

13/a Milyen paranccsal tudja átmásolni a C: meghajtó DOS könyvtárából az összes .COM kiterjesztésű programot az A: meghajtóban lévő lemezre? Ismertesse az esetleges hibaüzenet okát és elhárításának módját! Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_13.TXT néven.

13/b Melyik DOS paranccsal tud könyvtárakat **létrehozni?**

13/c Írassa az A: lemez főkönyvtárában lévő file-ok adatait (név, kiterjesztés, méret, dátum, idő) az A: meghajtó DOS könyvtárába TARTALOM.TXT néven!

14/a Mi a feladata az AUTOEXEC.BAT file-nak ?
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_14.TXT néven.

14/b Melyik parancs segítségével tudja a parancs file-ok keresési útját átállítani?

14/c Készítsen mentést a C: meghajtó TANULO könyvtáráról a BACKUP parancs segítségével az A: meghajtóban lévő lemezre !

15/a Mi a hatása az alábbi DOS parancsoknak?

C:DIR /P
C:DIR >PRN
C:DIR ?ab.*

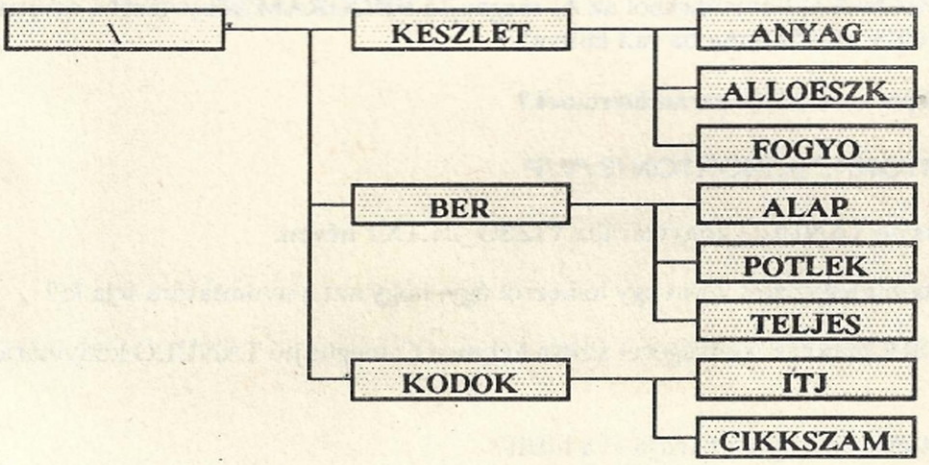
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_15.TXT néven.

15/b Melyik paranccsal és hogyan tudja az operációs rendszer **dátumát** megváltoztatni?

15/c Ellenőrizze a kiadott lemezt a SCAN programmal. Milyen üzenetet küld a SCAN program?

16/a Milyen paranccsal lépne át az alábbi könyvtárszerkezet

- **KODOK** könyvtárából a **POTLEK**-ba ?
- **POTLEK** könyvtárából a **CIKKSZAM**-ba ?
- **CIKKSZAM** könyvtárából a **Fő** könyvtárba ?
- **ANYAG** könyvtárából az **ITJ**-be ?
- **BER** könyvtárából a **KODOK**-ba ?



Válaszát tárolja a C: meghajtó TANULO könyvtárába VIZSG_16.TXT néven.

16/b Melyik parancs segítségével tud egy szöveges file-t **kinyomtatni**?

16/c Állítsa be a C: meghajtó kötetcímkéjét **VISZKI** névre!

17/a Hogyan tudja az A: meghajtó DOS könyvtárában lévő **CONFIG.SYS** és **CONFIG.BAK** szöveges file-okat összemásolni az A: meghajtó lemezének főkönyvtárába **CONFIG.SYS** névre?
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_17.TXT néven.

17/b Mi a hatása a **CLEAN A: [Mich]** parancsnak?

17/c Hasonlítsa össze a C: lemez DOS könyvtárában és az A: lemez DOS könyvtárában lévő file-ok tartalmát!

18/a Hogyan tudja törölni az A: meghajtó főkönyvtárából a **CONFIG.BAK** file-t?
Hogyan tudja törölni az A: meghajtó NORTON könyvtárából az összes file-t?
Milyen kérdést tesz fel a gép? Mit válaszol rá?
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_18.TXT néven.

18/b Melyik **DOS** paranccsal és hogyan tud file-okat átnevezni? Milyen lehetőség van még ?

18/c Nevezze át az A: meghajtó lemezének **NORTON** könyvtárát **UTILS** névre, módosítsa az **AUTOEXEC.BAT** file-t úgy, hogy a NORTON programok bármely könyvtárból indíthatók legyenek !

19/a Mi a következő hibaüzenet oka, és hogyan hárítja el a hibát?

Access denied

Válaszát tárolja a C: meghajtó TANULO könyvtárába VIZSG_19.TXT néven.

19/b Melyik paranccsal tudja a B: meghajtóban lévő 3.5"-os, DS-HD jelű lemezt formattálni?

19/c Állítsa be az elérési utat az A: meghajtó lemezének **fő, DOS** és **NORTON** könyvtárára!

20/a Nevezze át az A: meghajtó főkönyvtárában lévő AUTOEXEC.OLD file-t AUTOEXEC.BAT névre? Hogyan tudja átnevezni az A: meghajtó NORTON könyvtárában lévő D betűvel kezdődő .EXE kiterjesztésű file-okat .COM kiterjesztésűre?
Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_20.TXT néven.

20/b Melyik paranccsal tudja egy lemez paramétereit kiíratni?

20/c Másolja át a C: meghajtó TANULO könyvtárából az A: meghajtó **PROGRAM** könyvtárába azokat a file-okat, melyeknek az **archív** attributuma be van állítva!

21/a Mi a hatása az alábbi DOS parancsnak és paramétereinek?

```
XCOPY C:\NORTON\*.* A:\NORTON /S /V /P
```

Válaszát tárolja a C: meghajtó TANULO könyvtárába VIZSG_21.TXT néven.

21/b Melyik paranccsal tud tartalomjegyzéket kérni egy lemezről úgy, hogy azt a nyomtatóra írja ki?

21/c A kiadott lemezről RESTORE parancs segítségével állítsa helyre a C: meghajtó TANULO könyvtárának tartalmát.

22/a Mi a következő hibaüzenet oka, és hogyan hárítja el a hibát?

Insufficient disk space

Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_22.TXT néven.

22/b Melyik paranccsal lehet egy szöveges file tartalmát megjeleníteni a képernyőn?

22/c Nyomtassa ki a A: meghajtó DOS könyvtárának tartalomjegyzékét!

23/a Melyik DOS paranccsal vizsgálhatjuk és állíthatjuk a file-ok **attributumait**? Magyarázza meg az attributumok jelentését. Válaszát tárolja a C: meghajtó TANULO könyvtárába VIZSG_23.TXT néven.

23/b Melyik paranccsal és hogyan tudja az operációs rendszer **idejét** beállítani?

23/c Írassa az A: meghajtó DOS könyvtárába **CIMKE.TXT** néven a C: lemez kötetcímkéjét!

24/a Mi a hatása az alábbi DOS parancsoknak?

```
DIR A:\ > A:\TARTALOM.TXT
DIR A:\DOS >> A:\TARTALOM.TXT
DIR A:\NORTON >> A:\TARTALOM.TXT
```

Válaszát a NORTON EDITOR segítségével írja meg, és tárolja a C: meghajtó TANULO könyvtárába VIZSG_24.TXT néven.

24/b Melyik paranccsal léphet be a számítógépével a gépterem hálózatába, és hogyan lép ki abból?

24/c Listázza a képernyőre a C: meghajtó **TANULO** könyvtárában lévő **PROBA.PAS** file tartalmát! Használja a megfelelő szűrő parancsot a lapozáshoz!

25/a Állítsa át a képernyőt 40 karakter/sor megjelenítésűre!

25/b Rendezze a lemez tartalomjegyzékét a file-ok kiterjesztése és neve szerint! A rendezést végeztesse el az alkönyvtárakban is!

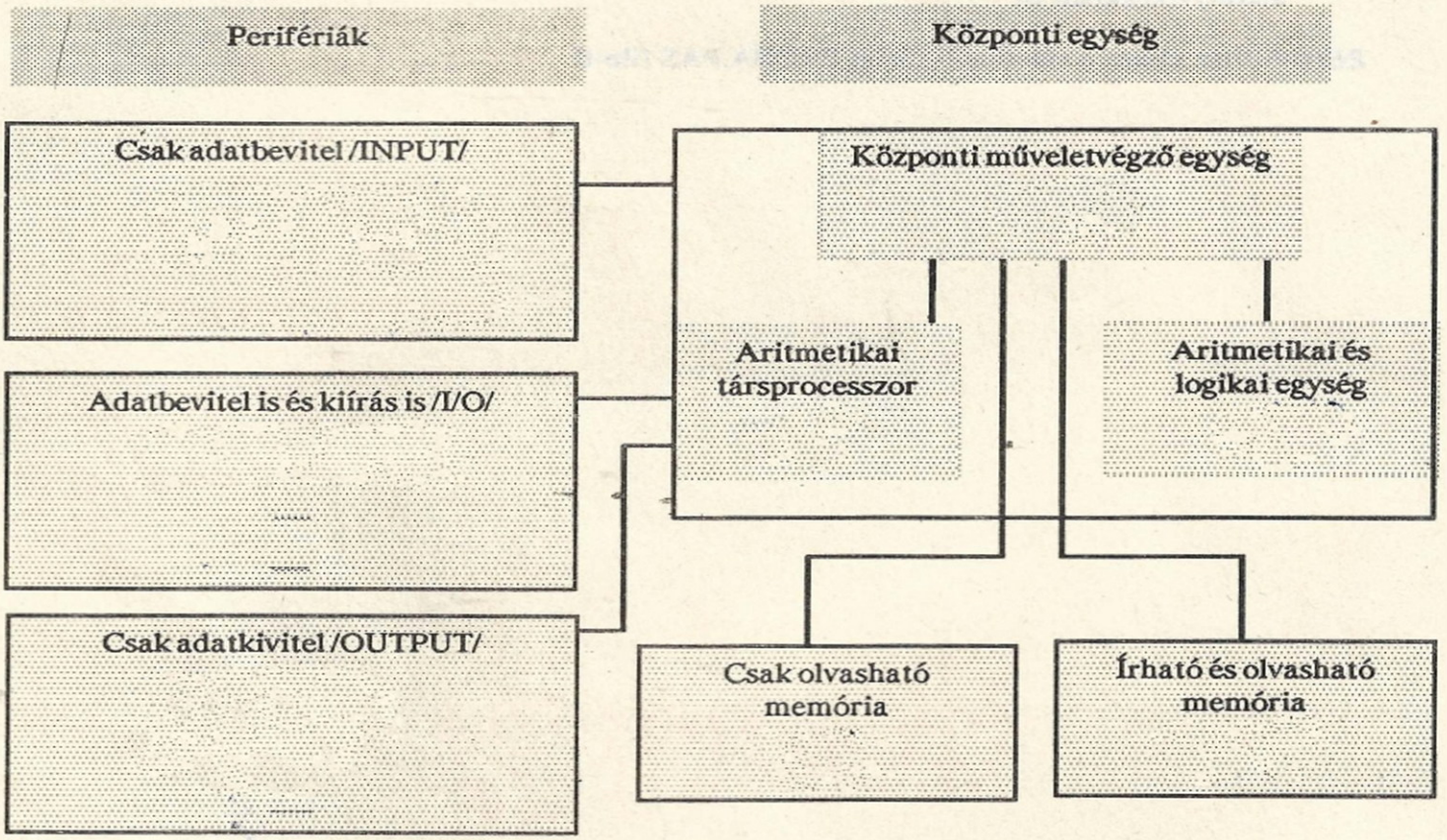
25/c Állítsa vissza a véletlenül törölt PROBA.PAS file-t!

2.8.4 Feladatlap a 2. fejezethez

1. Írja be az ábra megfelelő részébe a következő elemeket:

CPU, ALU, CO-processzor, ROM, RAM, PRINTER, WICHESTER, FLOPPY, STREAMER, SCANNER, MOUSE, KEYBOARD, DISPLAY, PLOTTER.

A megfelelő vonalakra rajzolja be az "adatáramlás" irányát.



2. A felsorolt monitorok közül melyiket alkalmazná:

- ☐ adatrögzitési feladatoknál .....
- ☐ grafikai tervezési munkánál .....
- ☐ önszántából sohasem .....

Monitorok: **MONO, CGA, EGA, VGA**

3. Milyen jelzésű 5 1/4" méretű floppy-t vásárolna az:

- ☐ IBM XT típusú gépéhez GSDD
- ☐ IBM AT típusú gépéhez DSHD

4. Jelölje "X"-szel az alábbi felsorolás IGAZ állításait :

☐	1 BYTE	=	32	BIT
☒	1 SZÓ	=	4	BYTE
☒	FÉLSZÓ	=	16	BIT
☒	1 MB	=	1,000,000	BYTE
☐	1 kB	=	1,000	BYTE
☐	A STREAMER csak OUTPUT periféria			

5. Sorolja fel Neumann János számítógépének jellemzőit :

6. Íja be a megfelelő helyre a felsorolt tárolókapacitásokat.

1.44 MB, 80 MB, 360 kB, 20 MB, 1.2 MB, 40 MB

Típus	Jelölése	Méret	Kapacitás
Floppy	DS DD	5 1/4"	
Floppy	DS HD	5 1/4"	
Floppy	DS HD	3.5"	
WINCHESTER			

7. Mi az INTERFACE és mi a feladata ?

8. Írja be a táblázat megfelelő részeibe az "IGEN" vagy "NEM" szöveget.

	Csak olvasható	Irható és olvasható	Tartalmát kikapcsolás után is megőrzi
ROM	I	N	I
RAM	N	I	N

9. Írja be a táblázatba az adott számrendszerben használt számjegyeket.

Számrendszer neve	Számrendszerek írásjelei															
Kettes /bináris/																
Tizes /decimális/																
Tizenhatos /hexadecimális/																

10. Végezze el a számrendszerek közötti átszámításokat és töltse ki az üres "kockákat".

DECIMÁLIS	BINÁRIS								HEXADECIMÁLIS
198	1	1	0	0	0	1	1	0	
	1	0	1	0	1	0	1	1	
									AF
243									
	0	1	1	1	2	1	1	0	
									C2
255									
	1	0	0	1	1	0	0	0	
									3E
1A									
	0	0	1	1	1	1	0	0	

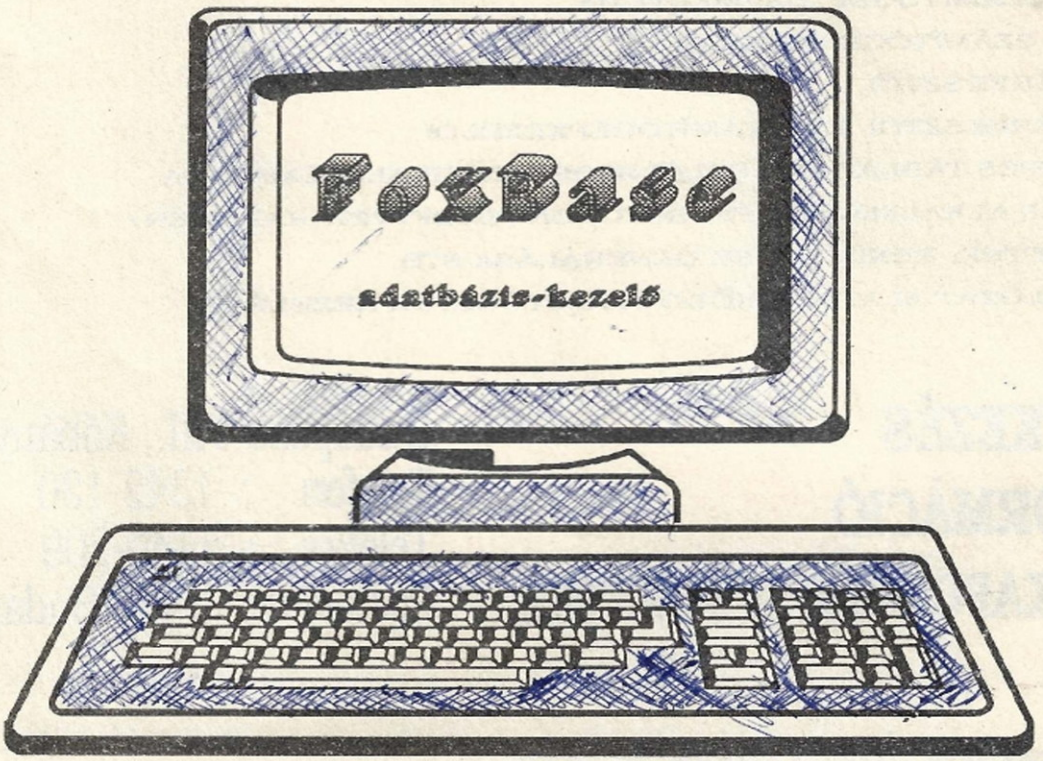
Húzza ki a fenti táblázat azon sorait, ahol az adott számrendszerben nem használható számjegyet talál.

11. Írja fel az alábbi BINÁRIS számok komplemens és a kettes komplemens alakját.

Bináris szám	1	0	1	1	0	1	1	0
Komplemense								
Kettes komplemense								

Bináris szám								
Komplemense	1	0	1	1	0	1	1	0
Kettes komplemense								

Bináris szám								
Komplemense								
Kettes komplemense	0	0	1	1	0	1	1	1


FoxBase
adatbázis-kezelő

KÉPZÉS - TOVÁBBKÉPZÉS - ÁTKÉPZÉS

TANFOLYAMI TÁJÉKOZTATÓ

SZEMÉLYI SZÁMÍTÓGÉP ÜZEMELTETŐ
ALAPFOKÚ SZÁMÍTÓGÉPKEZELŐI
SZÖVEGSZERKESZTŐI
SZÖVEGSZERKESZTŐI ÉS SZÁMÍTÓGÉPKEZELŐI
SZÁMÍTÓGÉPES TÁBLÁZATKEZELŐI SZOFTVEREK ALKALMAZÁSA
SZÁMÍTÓGÉP ALKALMAZÁSA FELSŐFOKON /SZOFTVER-HARDVER/
DOS ISMERETEK, RENDSZEREK GENERÁLÁSA STB.
NOVELL HÁLÓZAT ALATT MŰKÖDŐ SZÁMÍTÓGÉPEK KEZELÉSE

JELENTKEZÉS
INFORMÁCIÓ
KAPCSOLATFELVÉTEL

Budapest, VIII., Kőbányai u. 37. I. em.
Telefon : 1342-130
Telefax : 1140-209
Postacím : 1430 Budapest, Pf. 32

CO-NEX-TRAINING BT.

OKTATÁS

tanfolyamok
trainingek
előadások
konzultációk

SZAKMAI RENDEZVÉNYEK

konferenciák
szakmai fórumok
kiállítások
árverések

SZERVEZÉSE, LEBONYOLÍTÁSA

CO-NEX-TRAINING BT.

1300 BUDAPEST, Pf. 201

Tel/Fax: 186-9394

3. ADATBÁZISKEZELÉS

A személyi számítógépeken legjobban a dBASE kompatibilis adatbáziskezelő programok terjedtek el. Ilyenek a:

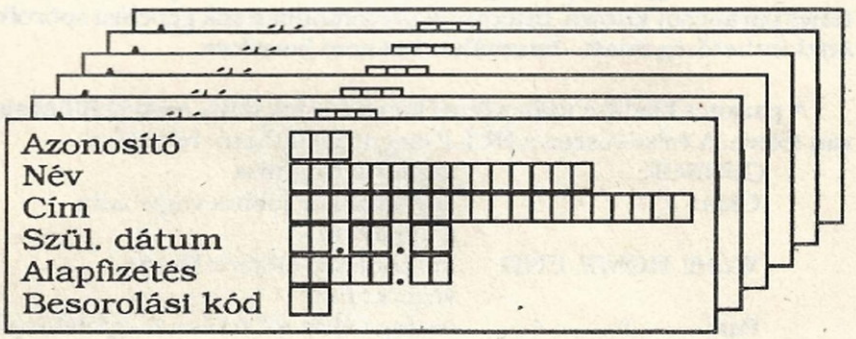
dBASE III PLUS,
dBASE IV,
FOXBASE,
FOXPRO,
CLIPPER.

Mi a FOXBASE+ programmal fogunk megismerkedni, ennek is először a párbeszédes üzemmódjával, majd a programozásával. A példákban szereplő utasítások minden paraméterét és részletes szintaktikáját a függelékben találhatják meg.

3.1 Az adatállomány

Tegyük fel, hogy egy vállalat dolgozóiról személyi adatokat akarunk nyilvántartani, valamint ebből az adatállományból különböző adatokat kigyűjteni. Ha ezt a feladatot a hagyományos módon kell megoldani, úgy a jobboldalon látható kartonokat töltenénk ki, és ezekkel dolgoznánk. A kartonok összességét adatállománynak, egy kartont rekordnak (record), a karton egy sorát mezőnek (field) nevezzük.

Ha az adatbáziskezelő programmal kívánunk dolgozni, azt el kell indítanunk.

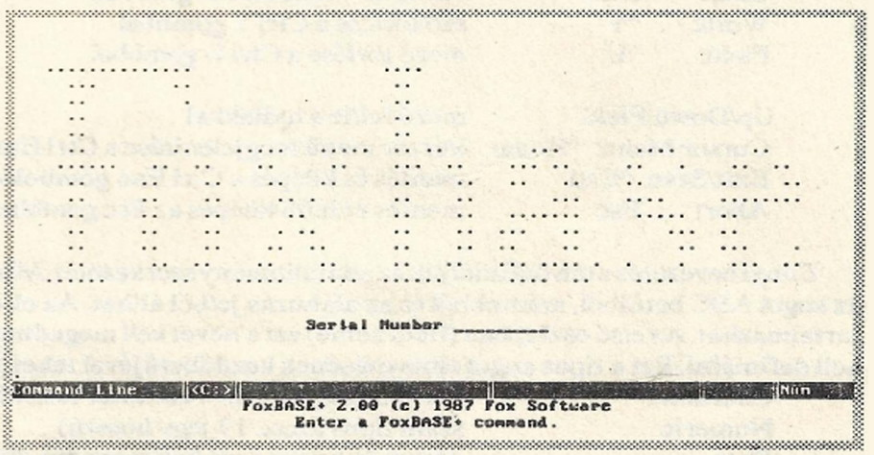


3.2 A FOXBASE indítása

Tételezzük fel, hogy a FOXBASE programcsomag megtalálható gépünk C: jelű merevlemezének FOXBASE könyvtárában. Ilyenkor a

C:
CD\FOXBASE
FOXPLUS

parancsokat kell begépelnünk az elindításhoz. Ha a program másik merevlemezen, vagy másik könyvtárban lenne, akkor természetesen azt kell a parancssorba behelyettesíteni.

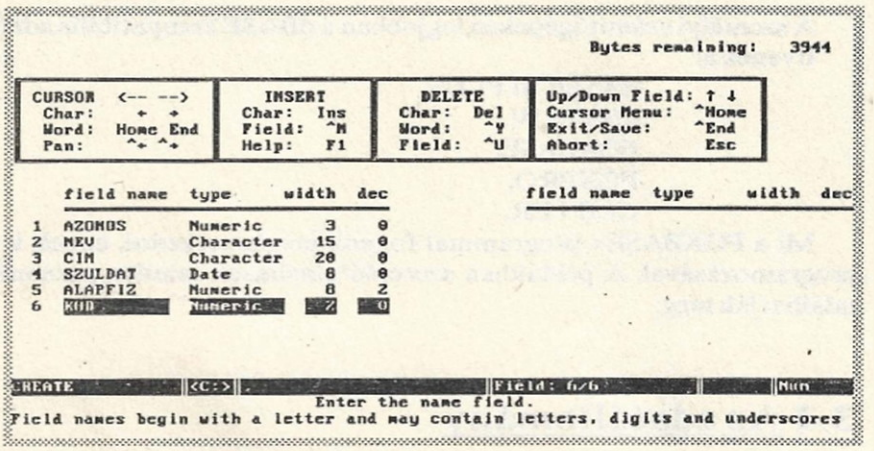


Bejelentkezés után a jobboldali képet látjuk.

Az alsó sort státusz sornak hívjuk, párbeszédes üzemmódban ez a sor sok információt közöl. Pontos jelentését majd menetközben. A státusz sor felett egy pontot látunk a kurzor e mögött villog. Ez a pont a FOXBASE promptja, mindig e mögé írhatunk. (Ebben a fejezetben azt, hogy egy parancsot kell kiadnunk, azzal jelzem, hogy a sor elejére egy pont mögé írom. A pontot természetesen nem kell begépelnünk, az a képernyőn lévő promptot jelöli.)

3.3 Az adatbázis létrehozása

Egy adabázis létrehozása a kartonok szerkezetének definiálásából áll. Először is ki kell találnunk egy nevet az adatállomány számára. Ez esetünkben legyen SZEMELY. Az adatállomány neve file-név, tehát a DOS file-név követelményeinek kell eleget tennie. Ha kiterjesztést nem adunk meg, a file kiterjesztése automatikusan .DBF lesz, egyébként amit megadunk. Az adatállomány létrehozására a CREATE parancs szolgál. Ha a parancsot önmagában adjuk ki, akkor a program megkérdezi a file nevet.



. CREATE

Enter the name of the new file: SZEMELY

De a file-nevet megadhatjuk rögtön a parancs után is:

. CREATE SZEMELY

A parancsokat írhatjuk kis- és nagybetűvel is, sőt rövidíthetjük is őket. A rövidítés azt jelenti, hogy a parancsnak csak az első négy betűjét adjuk meg. (pl.: a CREA helyettesíti a CREATE parancsot.) Én a továbbiakban mindig a teljes parancsot kiírom. Interaktív üzemmódban sok gépelést spórolhatunk meg a rövidítésekkel, de programban - az áttekinthetőség miatt - használatukat nem javaslom.

A parancs kiadása után a fenti képet fogjuk látni, annyi különbséggel, hogy itt az adatállomány szerkezete már ki van töltve. A felső részen a HELP (segítség) látható. Jelentése:

CURSOR		kurzor mozgatása
Char:		egy karakter jobbra vagy balra a nyilakkal
Word:	HOME END	kurzor a szó elejére: Home, végére : End
Pan:	^ ^	oszlop váltás a Ctrl és a megfelelő nyíl lenyomásával
INSERT		beszúrás
Char:	Ins	az Insert gombbal választhat a beszúrás és felülírás üzemmód között
Field:	^N	mező beszúrása a Ctrl N gombokkal
Help:	F1	az F1 gomb eltünteti/megjeleníti a HELP-et
DELETE		törlés
Char:	DEL	karakter törlése a Del gombbal
Word:	^Y	szó törlése a Ctrl Y gombbal
Field:	^U	mező törlése a Ctrl U gombbal
Up/Down Field:		mező fel/le a nyilakkal
Cursor Menu:	^Home	kurzor menü megjelenítése a Ctrl Home gombokkal
Exit/Save:	^End	mentés és kilépés a Ctrl End gombokkal
Abort:	Esc	mentés nélküli kilépés az Esc gombbal

Ennyi bevezetés után definiáljuk az adatállomány szerkezetét. Minden mezőnek nevet kell adnunk. Egy mező neve az angol ABC betűiből, számokból és az aláhúzás jelből állhat. Az első karakternek betűnek kell lennie, szóközt nem tartalmazhat. Az első oszlopban (field name) ezt a nevet kell megadnunk. A második oszlopban (type) a mezők típusát kell definiálni. Ezt a típus angol elnevezésének kezdőbetűjével tehetjük meg. A típusok lehetnek:

Character	karakteres (max. 254 karakter hosszú szöveg tárolására)
Numeric	szám típus (max. 19 jegy hosszú)
Date	dátum (a hosszt nem kell megadni, fixen 8)
Logical	logikai (a hosszt nem kell megadni, fixen 1.) A mező értéke csak igaz (True), vagy hamis (False) lehet)
Memo	változó hosszúságú szöveg tárolására alkalmas (az adatállományból elfoglalt hely fixen 10 byte, a valódi tárolás nem itt történik, hanem egy .DBT kiterjesztésű file-ban. Az ilyen típusú mezők csak tárolásra szolgának, velük műveletet, keresést nem lehet végezni.)

A következő oszlopban (width) a mező hosszát kell megadnunk, az utolsóban (dec) pedig a tizedes jegyek számát. Tizedesjegyet természetesen csak szám típusú mezőknél adunk meg. Itt a hosszba (width) az összes jegyet (egészek, tizedespont, tizedesek) bele kell számolni.

Ezek után töltsük ki a táblázatot a kép alapján.

A mezőnevek természetesen rövidítve lettek, pl.: az azonosító AZONOS-ra. Ha egy sort kitöltöttünk a kurzor automatikusan a következő sor elejére ugrik. Ha hibázunk, a gép ezt sípolással jelzi és kiír az alsó sorba egy hibaüzenetet. Ilyenkor először egy szóközt kell nyomnunk, csak ezután javíthatjuk a hibát. Ha egy mezőnevet, vagy típust rontottunk el és a hosszt még nem adtuk meg, a gép nem enged tovább. Be kell írni egy tetszőleges hosszt, ezután javíthatjuk a hibát. A CREATE parancsból való kilépéshez Ctrl End kombinációt kell nyomnunk. Erre az alsó sorban megjelenik egy kérdés, hogy tényleg befejeztük-e, ilyenkor az Enter lenyomásával válaszolunk. Ezután újabb kérdés következik: akarunk-e egyből adatokat is begépelni a most létrehozott adatállományba. Itt igennel (Y) vagy nemmel (N) válaszolhatunk, most válasszuk az N-t.

Ezzel az adatállomány létrehozása befejeződött. Érdemes megnéznünk a státusz sort, ebben az aktuális adatállomány neve (SZEMELY) látható.

Ha egy adatállomány szerkezetét meg akarjuk nézni, akkor adjuk ki a DISPLAY STRUCTURE, vagy LIST STRUCTURE parancsot. Ennek hatására az adatállomány szerkezete a CREATE parancshoz hasonló formában kerül kijelzésre, de ebben módosítás nem lehetséges. Ha az adatállomány szerkezetének módosítása szükséges, akkor a MODIFY STRUCTURE parancsot kell használnunk. Ennek hatása a CREATE paranccsal megegyező azzal a különbséggel, hogy itt a már kitöltött szerkezet jelenik meg és azt módosíthatjuk. Próbáljuk ki a parancsokat!

3.4 Kilépés a FOXBASE-ből

Most gyakorlásképpen lépjünk ki. Ehhez a QUIT parancsot kell begépelnünk. Erre az összes állomány lezárásra kerül és visszatérünk a DOS prompthoz.

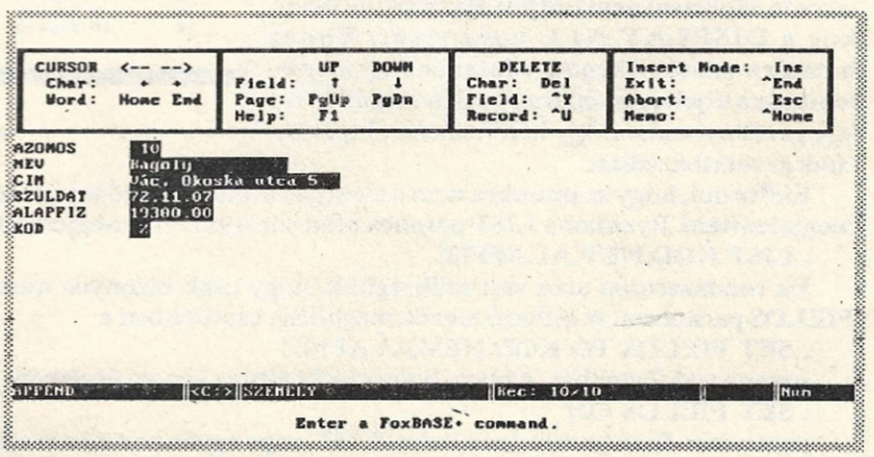
3.5 Adatállomány megnyitása

Lépjünk vissza a FOXBASE-be. Ha most megnézzük a státusz sort, nincs állomány név kiírva. Ez azt jelenti, hogy nincs nyitott adatállományunk. Ahhoz, hogy az előbb létrehozott SZEMELY nevű adatbázist használhassuk ki kell adnunk a

. USE SZEMELY

parancsot. A parancs kiadása után a státusz sorban ismét látható az állomány neve.

3.6 Adatok bevitele, módosítása



Most elkezdjük az adatállományt feltölteni. Erre az APPEND (bővítés) parancs szolgál. Ez tulajdonképpen egy üres rekordot fűz az állomány végére, majd engedi, hogy azt kitöltsük.

Begépelése után a jobboldali kép jelenik meg (egyenlőre üresen). Egyszerre egy rekordot (kartont) látunk a képernyőn. Ha ennek adatait begépeltük, automatikusan a következő rekordra lépünk. Ha javítanunk kell, akkor a nyilakkal tudunk mozogni és felülírással helyesbíteni. Ha lapozni kívánuk a rekordok között, azt is megtehetjük a PageUp, vagy PageDown gombokkal. (Ha túllapozunk az adatállomány elején vagy végén, azzal automatikusan kilépünk az APPEND parancsból, ilyenkor a parancs ismételt kiadásával lépjünk vissza). Az adatbevitelt a Ctrl End gombokkal fejezhetjük be.

Ha a CREATE parancsnál az azonnali feltöltést választottuk volna, azzal pontosan az APPEND paranccsal megegyező funkciót hajtottunk volna végre. A későbbi állományoknál érdemes ezt választani.

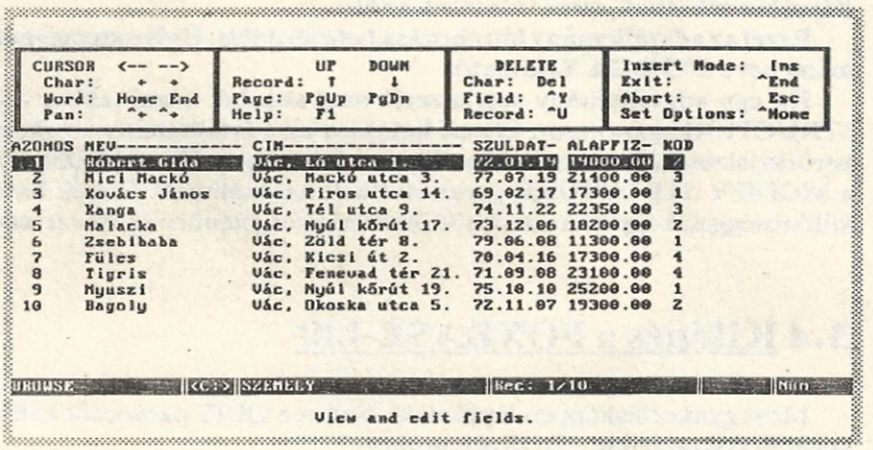
Ha módosítani kívánjuk az adatállományt, ezt az EDIT paranccsal tehetjük meg. Működése az APPEND-del teljesen megegyezik, de új rekord nem fűződik az adatállományhoz, csak a már meglévőket módosíthatjuk.

A dátum típusú mezők bekérésénél a gép a FOXBASE alapbeállítása szerint kéri a dátum megadását. Ha a konfigurációs file-ban ez nincs megadva, akkor az amerikai szabvány szerint kéri (hó/nap/év). (A konfigurációs file parancsainak leírása a függelékben található). Ha a magyar írásmód szerinti dátumbekérést szeretnénk (év.hó.nap), adjuk ki a

. SET DATE ANSI

parancsot.

Az APPEND és az EDIT parancsoknál egyszerre csak egy rekordot látunk a képernyőn. Ez néha kevés. Módosításra, illetve adatbevitelre használható a BROWSE (böngészés) parancs. Ezzel az egy rekordhoz tartozó mezőket nem egymás alatt, hanem egysorban, egymás mellett látjuk a képernyőn. Így szabadon lépkedhetünk a mezők között a nyilak segítségével. Ha a képernyő szélessége nem elegendő az összes mező kijelzésére, akkor a Ctrl gomb és a megfelelő (jobbra vagy balra) nyíl segítségével görgethetjük a képernyőt oldalra is. A meglévő rekordok felülírással módosíthatók. Ha megpróbálunk az adatállomány végén túllépni, akkor a gép megkérdezi, hogy akarunk-e új rekordot fűzni az állományhoz ha Y-nal válaszolunk, az adatállomány végén létrejön egy új, üres rekord. Ha a BROWSE parancsnál még több sort szeretnénk látni, akkor az F1 parancs segítségével kikapcsolhatjuk a help menüt.



Most töltsük fel az adatbázist pontosan a képen látható adatokkal!

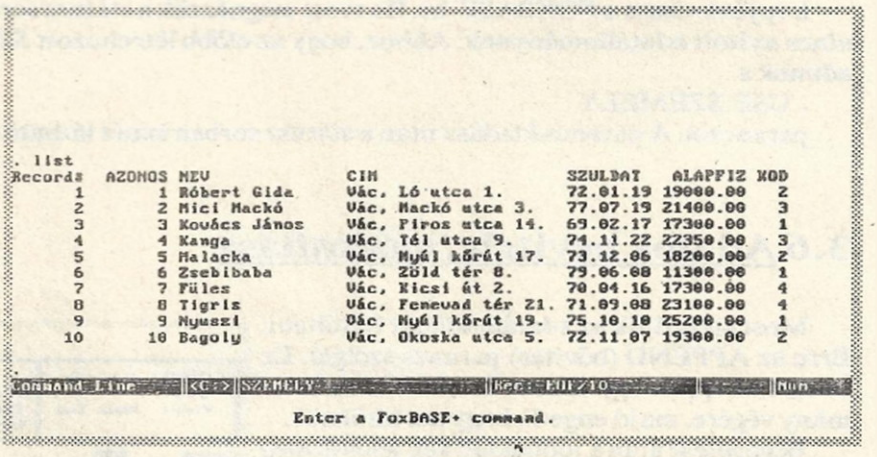
3.7 Listázás, feltételek, relációs jelek

Van egy feltöltött adatbázisunk. Hogyan készíthetünk listákat?

Próbáljuk ki először a LIST parancsot! Ennek hatása a jobboldali képen látható.

A lista fejlécében az általunk definiált mezőneveket látjuk. A lista első oszlopa a rekord fizikai sorszámát tartalmazza, ez esetünkben a felvitel sorrendje.

Ha az adatbázisunk sok rekordból áll, akkor a LIST parancs hatására gyorsan végigfut a képernyőn, elolvasni nem tudjuk. Használjuk ilyenkor a DISPLAY ALL parancsot. Ennek hatására szintén az egész adatállomány megjelenik, de a kijelzés a képernyő alján megáll és vár egy gombnyomásra. Így kényelmesen lapozhatunk az adatbázisban.



Előfordul, hogy számunkra nem szükséges minden mező kijelzése, vagy a mezőket megadott sorrendben akarjuk megjeleníteni. Ilyenkor a LIST parancs után soroljuk fel a megjeleníteni kívánt mezők neveit. pl.:

. LIST KOD,NEV,ALAPFIZ

Ha rendszeresen arra van szükségünk, hogy csak bizonyos mezők legyenek láthatók, akkor használjuk a SET FIELDS parancsot. A látható mezők megadása esetünkben a

. SET FIELDS TO KOD,NEV,ALAPFIZ

paranccsal történhet. Ahhoz, hogy ez a beállítás legyen érvényes, még ki kell adnunk a

. SET FIELDS ON

parancsot. Ezután minden kiadott LIST, vagy egyéb parancsra csak a felsorolt mezők jelennek meg, anélkül, hogy újból felsorolnánk a mezőneveket. Ez a beállítás marad érvényben, amíg egy

. SET FIELDS OFF

paranccsal nem hatástalanítjuk. Ezután megint az összes mező látható.

A rekordsorszám kijelzésére a legtöbbször nincs szükségünk, ilyenkor használjuk a LIST OFF parancsot.

Nagyon sokszor van szükségünk arra, hogy valamilyen szempont szerint válogassunk a kartonok között és csak azokat jelenítsük meg, amik valamilyen feltételnek megfelelnek. Ilyenkor a LIST parancsban egy FOR szócska után megadjuk a kívánt feltételt. Kérjünk most egy listát azokról, akiknek a besorolási kódjuk kettes:

. LIST FOR KOD=2

Akiknek az alapfizetésük kisebb, mint 15000:

. LIST FOR ALAPFIZ<15000

Eddig csak numerikus típusú mezőkre hivatkoztunk. Nézzük meg milyen relációs jeleket használhatunk:

<	kisebb
>	nagyobb
=	egyenlő
<=	kisebb vagy egyenlő (nem nagyobb)
>=	nagyobb vagy egyenlő (nem kisebb)
<>	nem egyenlő

Próbáljunk meg ezek alapján különböző listákat kérni!

A mi adatállományunkban karakteres típusú mezők is találhatók. (NEV,CIM) Nézzük meg, ezekre hogyan hivatkozhatunk. Kérjünk listát azokról, akiknek a neve Kanga! Vigyázat a gép megkülönbözteti a kis- és nagybetűket, tehát most ugyanúgy kell beírnunk a nevet, mind ahogy ezt az adatállomány felvitelénél tettük!

. LIST FOR NEV="Kanga"

Figyeljük meg, hogy ha egy konstans szöveget adunk meg azt idézőjelek közé kell tennünk. Mindegy, hogy idézőjelet ("), vagy aposztrófot (') használunk, csak arra figyeljünk, hogy a szöveg elején és végén ugyanaz a jel álljon. Próbáljunk ki egy másik parancssort!

. LIST FOR NEV="M"

Itt az eredmény elsőre meglepő lehet. Az egyenlőségjel szöveg típusú változók esetén nem követel meg pontos egyenlőséget. Esetünkben a NEV mező 15 karakter hosszú, a megadott konstans pedig csak egy karakter. Így a gép csak az első karakterig (általános esetben a rövidebbik szöveg végéig) vizsgálja az egyezőséget. Ha a teljes egyezőség a követelmény használjunk dupla egyenlőség-jelet.

. LIST FOR NEV=="Kanga"

Erre a parancssorra a gép nem listáz semmit. Miért? Az adatállományban a NEV 15 karakter hosszú, a végén szóközök vannak. Ha a konstansot is kiegészítjük szóközökkel 15 karakterre, akkor már kapunk választ a parancsra.

. LIST FOR NEV=="Kanga "

Ha folyamatosan a pontos egyezőségre van szükségünk, adjuk ki a

. SET EXACT ON

parancsot, ezután a sima egyenlőségjel is a duplához hasonlóan a pontos egyezőséget fogja jelenteni. Ezt visszaváltani a

. SET EXACT OFF

paranccsal tudjuk.

Listát kérhetünk aszerint is, hogy az adatállomány mezői az ABC sorban előrébb, vagy hátrább vannak egy konstansnál.

. LIST FOR NEV>"Kanga"

. LIST FOR NEV<="Kanga"

Itt az ABC helyett igazából az ASCII kódok sorrendjét kell érteni, amiben legelől az írásjelek, utána az angol ABC nagybetűi, majd a kisbetűk és legvégül az egyéb grafikus jelek vannak. A magyar ABC ékezetes betűi is a grafikus jelek közé vannak sorolva. Így a magyar ABC szerint nem mindig tökéletes a válogatás. Ennek a problémának a kiküszöböléséről később még lesz szó.

Szükségünk lehet arra is, hogy megkeressük azokat a neveket, amikben előfordul egy megadott karaktersorozat. (Gondoljunk egy könyvtár nyilvántartásra, ahol egy könyvnek nem tudjuk a pontos címét, hanem azokat keressük amelynek címében benne van a számítástechnika szó.)

. LIST FOR "Ma" $ NEV

Ez a parancs azokat a rekordokat fogja kilistázni, amelyek NEV mezőiben szerepel az "Ma" karaktersorozat (példánkban Mici Mackó és Malacka).

Foglaljuk össze a karakteres változók között használható relációs jeleket:

=	megegyezik (alapesetben csak a rövidebbik végéig)
==	pontosan megegyezik
>	nagyobb (ABC sorban hátrébb található)
<	kisebb (ABC-ben előbb)
>=	nagyobb vagy egyenlő (megegyeznek, vagy ABC-ben hátrább)
<=	kisebb vagy egyenlő (megegyeznek, vagy ABC-ben előbb)
<>	nem egyeznek
$	benne van (az elsőnek megadott megtalálható a második szövegben)

Próbáljunk minél több tetszőleges listát kérni!

A harmadik - példánkban előforduló - típus a dátum.

Dátum szerint is kérhetünk listákat. Nézzük meg azokat, akiknek születési dátuma 1970. január 1-nél későbbi!

. LIST FOR SZULDAT>=CTOD("70.01.01")

A CTOD() függvény használatára azért van szükség, mert dátum típusú konstans megadására a FOXBASE-ben nincs lehetőség. A CTOD() függvény az argumentumában megadott szövegkonstanst ("70.01.01") alakítja át dátum típusúvá. Ezek után az összehasonlítás már elvégezhető. Dátum típus esetében a relációs jelek lehetnek:

=	megegyező dátum
<	korábbi
>	későbbi
<=	korábbi vagy egyező
>=	későbbi vagy egyező
<>	nem egyező

3.8 Összegzés, átlagszámítás, számlálás

Lehetőségünk van numerikus mezők összegzésére a SUM paranccsal.

. SUM

A parancs hatására az összes numerikus mező összegzésére kerül, valamint megjelenik az összegzett mezők száma. A mi példánkban egyetlen mező, amelynek összegzése értelmes, az ALAPFIZ. Ha nem minden mezőt kívánunk összegezni, akkor a SUM parancs után soroljuk fel az összegezni kívántakat.

. SUM ALAPFIZ

Ha válogatni kívánunk a rekordok között, használjuk a FOR szócskát és a feltételt.

. SUM ALAPFIZ FOR SZULDAT<=CTOD("69.09.01")

. SUM FOR SZULDAT<=CTOD("69.09.01") ALAPFIZ

A feltétel és a mezők felsorolásának sorrendje felcserélhető. Kérhetünk átlagszámítást is a numerikus mezők alapján az AVERAGE paranccsal. Önmagában kiadva az összes rekord numerikus mezőiből végzi a számítást:

. AVERAGE

A SUM-hoz hasonlóan felsorolhatunk mezőket és megadhatunk feltételt is. Például kiszámíthatjuk a 4-es besorolási kódú dolgozók átlagfizetését:

. AVERAGE ALAPFIZ FOR KOD=4

Kiírathatjuk - a COUNT parancs segítségével - valamilyen tulajdonságú rekordok számát.

. COUNT FOR ALAPFIZ>17500

3.9 Logikai műveletek

Szükségünk van bonyolultabb feltételek szerint is listát kérni. Például azokat szeretnénk látni, akiknek fizetése 17000 és 21000 forint közé esik. Ilyenkor a parancs magyarul így nézne ki: Listázd azokat akiknél a fizetés>=17000 és a fizetés<=21000. A FOXBASE-ben:

. LIST FOR ALAPFIZ>=17000 .AND. ALAPFIZ<=21000

Itt a .AND. az *ÉS*t jelenti. Ne felejtkezzünk el a két pontról! *ÉS* kapcsolat esetén az állításunk csak akkor igaz, ha az *ÉS* mindkét oldalán álló feltétel igaz. Nézzünk egy másik példát. Listázzuk ki azokat akiknek fizetése 17000-nél kisebb, vagy 21000-nél nagyobb.

. LIST FOR ALAPFIZ<=17000 .OR. ALAPFIZ>=21000

A *VAGY* kapcsolat esetén az állítás csak akkor hamis, ha mindkét oldalon álló feltétel hamis.

Még egy logikai feltételünk van a *TAGADÁS* ezt akkor használjuk, amikor egy feltételt meg szeretnénk fordítani:

. LIST FOR .NOT. KOD=3

Jelentésse listázzuk azokat, akiknél nem igaz, hogy a besorolási kód megegyezik hárommal. Természetesen ennek hatása megegyezik a

. LIST FOR KOD<>3

paranccsal.

A logikai műveletek igazságtábláit megtaláljuk a függelék elején.

3.10 Függvények

Eddig a függvények közül csak a CTOD()-dal találkoztunk. Egy függvény az argumentumában megadott kifejezés(ek)ből valamilyen számítási, konvertálási művelettel készíti el az eredményt. A CTOD() például egy karakteres típusú kifejezést alakít dátummá. Nézzünk néhány példát a függvényekre:

Matematikai függvények:

ABS(x)	a numerikus argumentum (x) abszolút értékét adja
MIN(x,y)	a két argumentum (x,y) közül a kisebbet adja vissza
ROUND(x,y)	az első argumentumot (x) adja vissza, a második argumentum (y) számú tizedesjegyre kerekítve
SQRT(x)	x négyzetgyökét adja vissza (természetesen x nem lehet negatív)
stb.	

Karakteres függvények:

LEFT(x,y)	az x karakteres kifejezésből a bal oldali y darab karaktert adja vissza
RIGHT(x,y)	az x karakteres kifejezésből a jobb oldali y darab karaktert adja vissza
LOWER(x)	az x szöveg nagybetűit kisbetűkké alakítja és így csupa kisbetűvel adja vissza x-et
UPPER(x)	az x szöveg kisbetűit nagybetűkké alakítja és így csupa nagybetűvel adja vissza x-et
LTRIM(x)	az x karakteres kifejezésből eltávolítja a kezdő szóközöket és így adja vissza
stb.	

Dátummal kapcsolatos kifejezések:

CTOD(x)	az x karakteres kifejezésből dátumot készít
DATE()	az operációs rendszer dátumát adja meg (nincs argumentum)
DAY(x)	megadja az x dátum típusú kifejezésből a nap hónapon belüli sorszámát
MONTH(x)	megadja az x dátum típusú kifejezésből a hónap sorszámát
YEAR(x)	megadja az x dátum típusú kifejezésből az évet numerikusan
stb.	

Nézzünk néhány példát a függvények használatára:

Listázzuk ki azokat akiknek július 19-én van a születésnapjuk!

. LIST FOR MONTH(SZULDAT)=7 .AND. DAY(SZULDAT)=19

Összegezzük azok fizetését, akik 20 évnél idősebbek! (A mai dátum évéből vonjuk ki a születési dátum évét és vizsgáljuk meg nagyobbat kaptunk-e 20-nál)

. SUM ALAPFIZ FOR YEAR(DATE())-YEAR(SZULDAT)>20

Számoljuk meg hány dolgozó fizetése tér el 4000 forintnál többel a 18000 forinttól! (Vizsgáljuk a fizetés és a 18000 forint különbségének abszolútértékét!)

. COUNT FOR ABS(ALAPFIZ-18000)>4000

Ugyanez más módon:

. COUNT FOR ALAPFIZ<14000 .OR. ALAPFIZ>22000

Találjunk ki minél több hasonló feladatot magunknak és próbáljuk megoldani!

3.11 Rendezés, indexelés

Valamilyen szempont szerint rendezett adatállományra sokszor van szükségünk. Ezt a rendezést a FOXBASE-ben úgy tudjuk megoldani, hogy az állományról egy rendezett másolatot készítünk. Erre a SORT parancs szolgál.

. SORT TO SZEMELY2 ON KOD,SZULDAT

A TO szócska után egy új file-nevet kell megadnunk (ez lesz a rendezett állomány neve). Az ON szó után azokat a mezőket kell megadnunk ami szerint rendezünk. Most nézzük meg milyen sorrendet kaptunk az új állományban:

. USE SZEMELY2
. LIST

Ha megvizsgáljuk a listát, látjuk hogy az állomány kód szerint rendezett, az azonos kódúaknál pedig a születési dátum határozza meg a sorrendet.

Készítsünk egy másik, fordított sorrendű listát:

. SORT TO SZEMELY3 ON KOD/D,AZONOS/D

A mezők neve után írt /D a csökkenő sorrendbe való rendezést jelenti (alapértelmezés szerint a rendezés növekvő sorrendben történik).

Nézzük meg az új állományt:

. USE SZEMELY3
. LIST

Ez az állomány kód szerint, azon belül azonosító szerint rendezett.

A SORT parancsnak két hibája van. Az egyik, hogy nagy méretű állományok esetén nagyon hosszú ideig tart. A másik, hogy ha fölviszünk egy új rekordot, vagy megváltoztatunk egy mezőt a rendezettségünk máris elromlott.

Térjünk vissza az eredeti állományunkhoz:

. USE SZEMELY

Rendezés helyett sokkal inkább használjuk az indexelést. Ilyenkor fizikailag nem jön létre rendezett állomány, listáink mégis rendezettek lesznek. Az indexelés is hosszú ideig tart, de ha egyszer elvégeztük, valamint figyelünk rá, hogy az adatállomány minden változásakor nyitva legyen az indexállomány, akkor a FOXBASE gondoskodik az indexállomány aktualizálásáról. Mit jelent ez? Ha egy adatállományt megindexelünk (mondjuk rögtön a létrehozása után) akkor látszólag mindig rendezett állománnyal dolgozhatunk.

Nézzük a példákat:

. INDEX ON KOD TO SZKOD
. LIST

A parancs kiadása után kód szerint rendezett listát kapunk. Az ON szó után annak a mezőnek vagy kifejezésnek a neve van ami szerint rendezünk, a TO csócska után pedig az index állomány neve. Ha kiterjesztést nem adunk meg, a létrejövő file .IDX kiterjesztésű lesz. Indexállományból egy adatállomány mellé többet is létrehozhatunk (maximum hetet). Egyszerre azonban csak egy indexállomány lehet aktív.

. INDEX ON SZULDAT TO SZSZUL
. LIST
. INDEX ON NEV TO SZNEV
. LIST

Most van három indexállományunk, de pillanatnyilag csak az utoljára létrehozott van megnyitva, mert az INDEX parancs hatására az előző automatikusan lezárásra kerül. Amikor egy adatállományt használatba veszünk (USE), megnyithatjuk mellé az indexállományokat is.

. USE SZEMELY INDEX SZKOD,SZNEV,SZSZUL
. LIST

Ezt az INDEX szó után felsorolt indexállomány nevekkel tehetjük meg. Megnyitás után az elsőként megadott indexállomány lesz aktív, tehát a listánk kód szerint rendezett. Hogy mi szerint szeretnénk a rendezettséget azt a SET ORDER paranccsal választhatjuk meg. Ilyenkor egy számot kell megadnunk, ami azt jelzi, hogy a USE parancsba hányadikként megadott indexállomány szerint kívánjuk a rendezettséget. Ha nullát adunk meg, állományunk eredeti (rendezetlen) sorrendben látszódik.

. SET ORDER TO 3
. LIST
. SET ORDER TO 2
. LIST
. SET ORDER TO 1
. LIST
. SET ORDER TO 0
. LIST

3.12 Keresés, lépkedés az adatállományban

Amíg kicsi az adatállományunk, addig egy LIST paranccsal át tudjuk tekinteni, vagy az EDIT parancs begépelése után gyorsan odalapozhatunk a módosítani kívánt kartonra. Nagy adatállomány esetén ez sok időt venne igénybe. Az EDIT parancs azt a rekordot fogja szerkesztésre behozni, amelyiken éppen állunk. Melyiken állunk? Ha például az első karton az aktuális akkor a státuszsor második felében a következőt látjuk Rec: 1/10. Ez azt jelenti, hogy összesen 10 kartonunk van és ebből az első az aktuális. A számok minden esetben a fizikai rekordsorszámot jelzik. A számok az EDIT, APPEND, BROWSE parancsok közben is tájékoztatnak arról, hogy hol állunk. APPEND parancs kiadásakor (amikor egy új üres rekordot nyitott a gép) a következőt látjuk: Rec: EOF/10. Ennek jelentése: az állomány végén állunk és az állomány 10 rekordból áll. Ha befejezzük a karton kitöltését, akkor a kijelzés Rec: EOF/11 lesz.

Ha az EDIT parancs után egy sorszámot írunk, akkor a megadott számú rekordot fogja szerkeszteni. A fizikai rekordsorszám szerinti lépkedésnek más módja is van ez a GOTO (menj oda) parancs.

. GOTO 5
. GO 2

Mindkét parancs lépkedést jelent (mindegy, hogy GO-t vagy GOTO-t írunk, sőt parancsüzemmódban ha csak a számot írjuk, akkor is a kívánt rekordra pozícionál).

Lehetőségünk van - anélkül, hogy ismernénk az állomány méretét - az elejére, vagy a végére lépni. Ilyenkor nem mindegy, hogy indexelt-e az adatállományunk. Próbáljuk ki a következő parancsokat és figyeljük a státusz sort:

. SET ORDER TO 0

```
. LIST
. GOTO TOP                    ( Rec: 1/10 )
. GOTO BOTTOM                 ( Rec: 10/10 )
. SET ORDER TO 3
. LIST
. GOTO TOP                    ( Rec: 3/10 )
. GOTO BOTTOM                 ( Rec: 6/10 )
. SET ORDER TO 2
. LIST
. GOTO TOP                    ( Rec: 10/10 )
. GOTO BOTTOM                 ( Rec: 6/10 )
```

Figyeljük meg: a GOTO TOP parancs hatására mindig az állomány elejére kerültünk, de a sorrendet az aktuális index szerint értve (ahogy a LIST parancs is mutatta). A GOTO BOTTOM pedig mindig a végére. A fizikailag is első, vagy utolsó rekordra csak a SET ORDER TO 0 után álltunk.

Másik lehetőségünk a lépkedésre a SKIP parancs. Ez egy megadott számú lépést hajt végre előre, vagy hátra az állományban. A pozitív szám az adatállomány vége felé történő lépést jelenti, a negatív az eleje felé lép. Ha nem adunk meg számot, ez ugyan azt jelenti, mintha +1-et adtunk volna meg. A sorrend itt is index szerinti. Próbáljuk ki:

```
. SET ORDER TO 3
. LIST
. GOTO TOP
. SKIP 1
. SKIP
. SKIP 2
. SKIP -3
. SKIP 5
. SKIP -2
```

Lehetőségünk van valamilyen feltétel szerint is keresni az adatállományban a LOCATE paranccsal:

```
. LOCATE FOR KOD=2
```

A parancs hatására az első olyan rekordra állunk, amiben a besorolási kód értéke kettő. Lehetőségünk van a lépkedés folytatására a CONTINUE paranccsal:

```
. CONTINUE
```

A parancsot ismételhetjük addig, míg az END OF LOCATE SCOPE üzenetet nem kapjuk. Ez azt jelenti, hogy nincs több a megadott feltételnek megfelelő rekordunk. Ha az állományunk indexelve van, akkor a LOCATE parancs az index szerinti sorrendnek megfelelően az első adott tulajdonságú rekordot találja meg és a CONTINUE parancs is index szerinti sorrendben lépked.

Próbáljunk ki egy másik keresést:

```
. SET ORDER TO 2
. LOCATE FOR KOD>=3
. CONTINUE
. CONTINUE
```

A LOCATE parancs működését úgy képzeljük el, mintha kézzel az első kartontól kezdve elkezdenénk végig lapozni az állományt, amíg meg nem találjuk a kívánt kartont. Nagy méretű adatállomány esetén, ha a keresett karton az adatállomány elejétől távol van akkor ez a keresés még számítógéppel is percekig tarthat. Index szerinti kereséssel ez sokkal gyorsabban végrehajtódik (gyakorlatilag azonnal megtalálja a kívánt rekordot). Index szerinti keresésre a SEEK parancs szolgál. Mögötte egy - az aktuális index típusával megegyező típusú - kifejezést kell megadnunk.

```
. SET ORDER TO 1
. SEEK 3
```

Ezzel ráálltunk az első hármas besorolási kódú rekordra.

```
. SET ORDER TO 3
. SEEK CTOD("71.09.08")
```

Ezzel az első olyan rekordot választjuk ki, ahol a születési dátum 1971. július 8-a. Üzenetet a parancs csak akkor ad, ha nem talál ilyen kulcsú rekordot (No find).

Ennél a keresési fajtánál csak az aktuális index-állomány szerint kereshetünk (nem kereshetjük az első olyan dolgozót akinek besorolási kódja nagyobb mint három) és nem folytathatjuk a keresést. A keresés folytatását - mivel állományunk a keresési kulcs szerint rendezett - helyettesíthetjük SKIP parancsokkal. A megkötések ellenére - sebessége miatt - érdemes az esetek többségében ezt a keresési fajtát választani.

3.13 Törlés az adatállományból

Törölhetünk az adatállományból rekordokat a DELETE paranccsal. Fontos tudnunk, hogy ez nem jelent fizikai törlést, csak törlésre jelölést. Ezt azért szervezték így, mert a fizikai törlés időigényes művelet. Lehetőségünk van a törlésre jelölt rekordok elrejtésére, így látszólag a törlés végrehajtódott. Tehát a hosszadalmasabb fizikai törlést elég csak időszakonként végrehajtani. Ha a DELETE parancsot önmagában adjuk ki, akkor az éppen aktuális rekordot jelzi törlésre, ha egy FOR feltételt írunk utána akkor a megadott feltételnek megfelelő rekordokat. Próbáljuk ki a következő parancsokat:

. GOTO 1
. DELETE
. GOTO 7
. DELETE
. DELETE FOR KOD=3
. LIST

A listában a törlésre jelölt rekordok előtt egy csillag látható. Ha el akarjuk rejteni ezeket a rekordokat, adjuk ki a

. SET DELETED ON

parancsot. Egy újabb listát kérve, megcsillagozott sorok nem láthatók:

. LIST

Tegyük újból láthatóvá a törlésre jelölteket:

. SET DELETED OFF
. LIST

A törlésre jelölést meg lehet szüntetni a RECALL (visszahívás) paranccsal. Használható önmagában - ilyenkor az aktuális rekordot állítja vissza - vagy FOR-ral és egy feltétellel.

. GOTO 1
. RECALL
. LIST
. RECALL FOR KOD=3
. LIST

Most már csak egy törlésre jelölt rekordunk maradt. Próbáljuk ki a fizikai törlést. Ilyenkor a törlésre jelölt rekordok véglegesen eltűnnek az állományból és a FOXBASE-nek korrigálnia kell az indexállományokat is. Adjuk ki a parancsot:

. PACK
. LIST

Most már az eredetileg hetes számú rekordunk (Füles) véglegesen eltűnt az adatállományból.

Ha egy adatállományt teljesen törölni szeretnénk (itt a CREATE utáni állapotról van szó, tehát az adatállomány szerkezete megmarad, de rekordokat nem tartalmaz), akkor a ZAP parancsot kell használnunk. A parancsnak nincsenek paraméterei és rögtön fizikailag törli az összes rekordot. A parancs kipróbálását halasszuk későbbre, mert az adatokra még szükségünk van.

3.14 Mezők cseréje, feltöltése

Ha egy vagy több mező értékét ki akarjuk cserélni, akkor a REPLACE parancsot kell használnunk. Ha az adatállományban egyenként az összes nevet ki kellene cserélnünk csupa kisbetűkkel írottakra, az fárasztó művelet lenne. Megtehetjük ezt a következő sorral:

. REPLACE ALL NEV WITH LOWER(NEV)
. LIST

Az ALL szó (összes) azt jelenti, hogy az adatállomány minden rekordján el kívánjuk végezni a műveletet. A WITH szó bal oldalán a cserélni kívánt mező neve áll, jobb oldalán amire cseréljük.

Nézzünk egy másik példát: azoknál akiknél a besorolási kód 1-es, írjuk át a kódot ötösre és növeljük meg a fizetést 2000-el.

. REPLACE FOR KOD=1 KOD WITH 5,ALAPFIZ WITH ALAPFIZ+2000
. LIST

Eddig a legfontosabb alapfogalmakat néztük végig. A következő részben új állományokat fogunk létrehozni és azokkal fogunk dolgozni. Az eddigi állományainkra többé nincs szükségünk (SZEMELY.DBF, SZEMELY2.DBF, SZEMELY3.DBF, SZNEV.IDX, SZKOD.IDX, SZSZUL.IDX). Kérem ismételjük át az eddig tanultakat! Az állományokra már nem kell vigyáznunk, sőt a gyakorlás végén le is törölhetjük őket.

3.15 Munka több adatállománnyal

Vannak olyan feladatok, amikor az adatainkat célszerű több adatállományban tárolni. Tételezzük fel, hogy egy nagykereskedelmi bolt, vagy raktár eladásait és pillanatnyi készleteit kell nyilvántartanunk. Ha ezeket az adatokat egy kartonra kívánjuk felvinni, akkor minden eladásnál ki kell töltenünk egy kartont, amin szerepelnie kell a következő adatoknak:

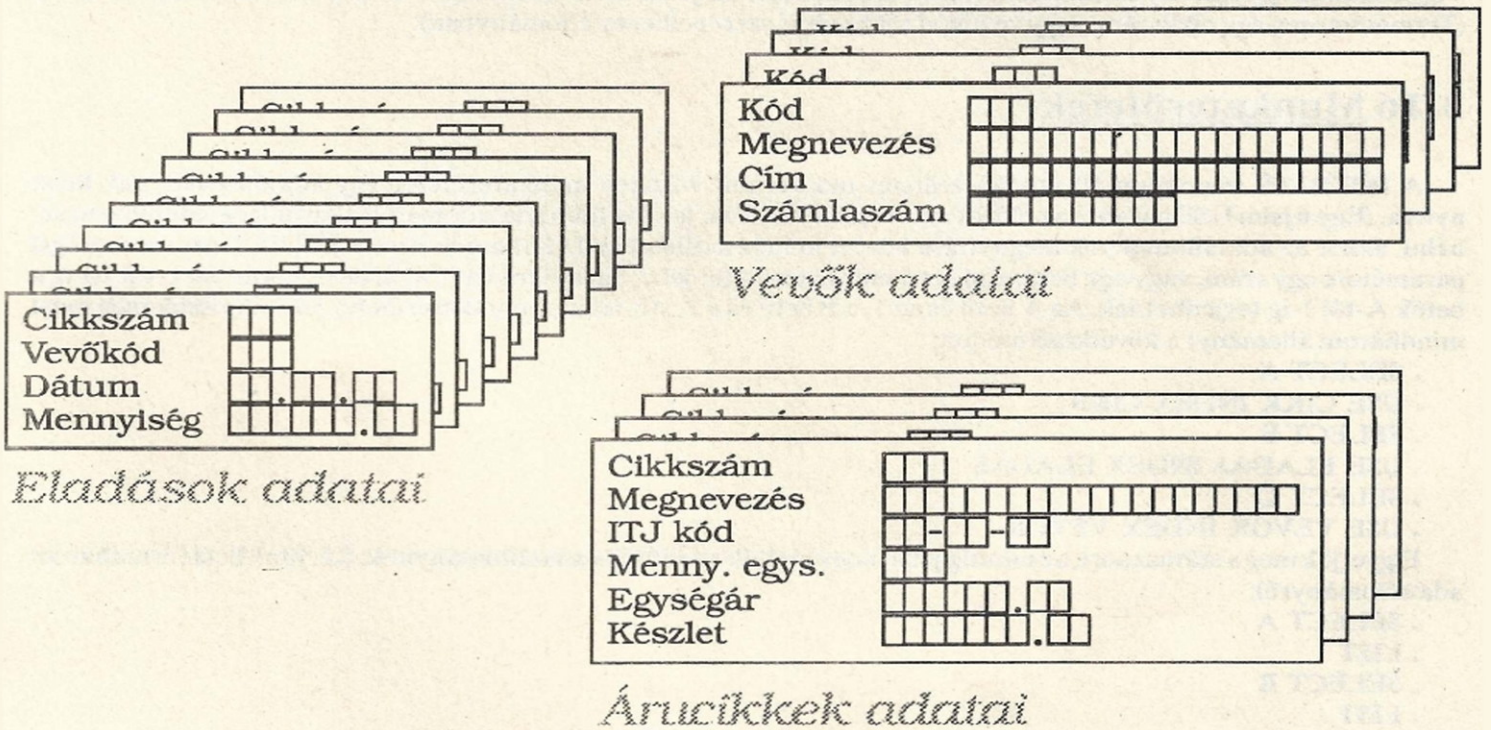
Áru adatai:
- Megnevezés
- ITJ kód
- Mennyiségi egység
- Egységár

Vevő adatai:
- Név
- Cím
- Számlaszám

Valamint:
- Eladás dátuma
- Eladott mennyiség

Mivel valószínű, hogy egy árucikkből sokat adunk el, valamint egy viszonteladó többször is megfordul a boltban, ugyanazokat az adatokat többször kellene felvinnünk. Ráadásul az áru megnevezését, vagy a vevő nevét nem biztos, hogy mindig pontosan ugyanúgy írnánk, ami nagyon megnehezítené a lekérdezéseket.

Hozzuk létre inkább a következő adatszerkezetet:



Ha egy új vevő érkezik, akkor adatait először fel kell vinnünk a vevő-állományba, az új cikket pedig a cikk állományba. Ha egy vevő már járt nálunk, akkor elég a kódjával hivatkozni rá az eladásnál. Ezzel biztosítottuk, hogy azonos adatokat nem kell többször felvinnünk, valamint a kódok alapján egyszerű a lekérdezés.

Hozzuk létre most az adatállományokat:
Az árucikkek adatainak a CIKK.DBF nevű állományt. Felépítése:

Field Name	Type	Width	Dec
CIKKSZAM	Numeric	3	
MEGNEVEZES	Character	20	
ITJ	Character	6	
MENNY_EGYS	Character	3	
EGYS_AR	Numeric	9	2
KESZLET	Numeric	10	2

Indexeljük meg cikkszám szerint, az indexállomány neve CIKK.IDX legyen! Vigyünk fel ötletszerűen adatokat (legalább 5 rekordot). Vigyázzunk, hogy minden cikkszám csak egyszer szerepeljen!
Hozzuk létre a VEVOK.DBF állományt a vevőknek:

Field Name	Type	Width	Dec
KOD	Numeric	3	
MEGNEVEZES	Character	20	
CIM	Character	20	
SZLASZ	Character	10	

Indexeljük meg kód szerint, az indexállomány neve VEVOK.IDX legyen! Vigyünk fel legalább öt vevőt, a kódjaik legyenek különbözőek!
Hozzuk létre a ELADAS.DBF állományt:

Field Name	Type	Width	Dec
CIKKSZAM	Numeric	3	
VEVOKOD	Numeric	3	
DATUM	Date	8	
MENNYISEG	Numeric	9	2

Indexeljük meg dátum szerint, az indexállomány neve ELADAS.IDX legyen! Vigyünk fel legalább húsz eladást! Vigyázzunk, hogy csak olyan cikkszámot és vevőkódot használjunk, ami a CIKK és a VEVOK állományban előfordul! (Természetesen egy cikkszám, vagy vevőkód többször is szerepelhet az állományban).

3.16 Munkaterületek

A FOXBASE maximum 10 munkaterületet tud kezelni. Minden munkaterületen egy adatállományunk lehet nyitva. (Egy újabb USE parancs az előzőt automatikusan lezárja). Ha több adatállományt akarunk egyidejűleg használni, akkor az adatállományok megnyitása között munkaterületet kell váltanunk. Erre a SELECT parancs szolgál paramétere egy szám, vagy egy betű, ami azt a munkaterületet jelzi, amelyikre lépni akarunk. A számok 1-től 10-ig a betűk A-tól J-ig terjedhetnek. Az A betű és az 1, a B betű és a 2, stb. azonos munkaterületet jelöl. Nyissuk meg most mindhárom állományt a következő módon:

```
. SELECT A
. USE CIKK INDEX CIKK
. SELECT B
. USE ELADAS INDEX ELADAS
. SELECT C
. USE VEVOK INDEX VEVOK
```

Figyeljük meg a státusz sort, az mindig jelzi, hogy melyik az aktuális adatállományunk. Kérjünk listát mindhárom adatállományról:

```
. SELECT A
. LIST
. SELECT B
. LIST
. SELECT C
. LIST
```

3.17 Adatállományok összekapcsolása

A felvitelt és az adatok karbantartását ezzel már meg tudjuk oldani, de a lekérdezéseket még nem. Listázzuk például egy megadott vevőnek, adott időszak alatt történt eladásokat úgy, hogy a listában szerepeljen az eladás dátuma, az eladott mennyiség, az áru megnevezése, mennyiségi egysége és ITJ száma. A problémát az okozza, hogy a listához szükséges adatok két adatállományban (ELADAS, CIKK) találhatók. A dátum és az eladott mennyiség mellé

az áru cikkszáma helyett annak tényleges adatait kell írnunk. Ezt elkészíthetnénk úgy, hogy kilistázzuk az ELADAS állományt, majd ennek minden soránál a cikkszám mező alapján kikeressük a CIKK állományból az áru adatait és a sor végére írjuk. Természetesen van egyszerűbb megoldás is. A SET RELATION paranccsal összekapcsolhatunk több állományt. Itt a cikkszám alapján az ELADAS állományhoz fogjuk hozzákapcsolni a CIKK állományt. Az összekapcsolás után ha az ELADAS állományban megváltoztatjuk a rekordmutató helyzetét, a CIKK állomány rekordmutatója automatikusan megváltozik, méghozzá pontosan arra a kartonra áll, amelyiknek cikkszáma megegyezik a CIKK állományban lévővel.

```
. SELECT B
. SET RELATION TO CIKKSZAM INTO CIKK
```

A parancs kiadásakor annak az állománynak kell aktuálisnak lennie, amihez hozzá akarjuk kapcsolni a másikat. A TO szócska után annak a mezőnek a nevét kell megadnunk, ami a kapcsolódást fogja jelenteni a két adatállomány között. A másik állománynak e szerint az azonosító szerint indexeltnek kell lennie (ezért indexeltünk). Az INTO szócska után a másik - az aktuálishoz csatolandó - állomány nevét kell megadnunk. A listát a következőképpen kell kérnünk:

```
. LIST DATUM,MENNYISEG,CIKK->MEGNEVEZES,CIKK->MENNY_EGYS,CIKK->ITJ
```

Listában az állománynév és -> jel megadás azt jelenti, hogy az adott mező egy másik állományban található. Természetesen ennek listázáskor csak összekapcsolt állományok esetén van értelme. A listázás közben a FOXBASE sorra veszi az ELADAS állományból az adatokat, közben a CIKK állományban is megfelelően mozgatja a rekordmutatót, így a listába mindig a megfelelő adatok kerülnek.

A feladat még nincs teljesen teljesítve, hiszen vevőt és időszakot nem adtunk meg, de ezt egy szűrő paranccsal könnyen megoldhatjuk: pl.:

```
. SET FILTER TO VEVOKOD=2 .AND. DATUM>=CTOD('88.01.01') .AND. DATUM<=CTOD('91.12.31')
. LIST DATUM,MENNYISEG,CIKK->MEGNEVEZES,CIKK->MENNY_EGYS,CIKK->ITJ
```

Ha egyidejűleg több összekapcsolást szeretnénk, akkor a következő SET RELATION parancs végére még egy ADDITIVE szót is kell tennünk, különben az előző összekapcsolások lezárásra kerülnek. Nézzünk erre is példát:

```
. SET RELATION TO VEVOKOD INTO VEVOK ADDITIVE
. SET FILTER TO CIKKSZAM=3 .AND. DATUM>=CTOD('89.07.01')
. LIST DATUM,MENNYISEG,MENNYISEG*CIKK->EGYS_AR,VEVOK->MEGNEVEZES
```

A példában a hármas cikkszámú áru 1989.07.01 utáni eladásait listázzuk. A listában szerepel az eladás dátuma, a mennyiség, az eladás értéke és a vevő megnevezése. Az eladás értékét az ELADAS állományban lévő mennyiség és a CIKK állományban lévő egységár szorzataként kapjuk.

3.18 Formátum állományok

Ha a kartonjaink papíron vannak, akkor van lehetőségünk az egyes sorok tartalmát megmagyarázni. Ha a feladatot FOXBASE-ben oldjuk meg, akkor az EDIT parancsnál azt a mezőnevet (maximum 10 karakter hosszú és csak az angol ABC betűit tartalmazhatja) látjuk, amit az adatállomány létrehozásakor megadtunk. Ez (ha nem vagyunk telhetetlenek) elég is addig, amíg mi gépeljük be az adatokat és nem mást ültetünk a gép elé. Ha szükséges a szép képernyőkép az adatok bevitelére, akkor formátum-állományt kell használnunk. Ez egy .FMT kiterjesztésű szöveges file, amiben a APPEND, EDIT parancsok képét definiálhatjuk. A file-ban háromféle parancs lehet: szöveg kiírás adott képernyő pozíción, mező beolvasása adott pozíció és maszk, keret rajzolás. Mindegyik parancs egy @ karakterrel és egy képernyő pozícióval kezdődik. A pozíciónál először a sort kell megadnunk. Ennek értéke 0 és 24 között lehet a 0 a képernyő legfelső, míg a 24 a legalsó sorát jelenti. A második az oszlop koordináta 0-79. Ahol a 0 a képernyő bal-, a 79 a jobb szélét jelenti. Nézzünk meg példaképpen egy már elkészített formátum-állományt:

```
@  9,20 TO 15,59
@ 10,25 SAY 'Cikkszám    '
@ 10,38 GET CIKKSZAM PICTURE '999'
@ 11,25 SAY 'Megnevezés  '
@ 11,38 GET MEGNEVEZES
@ 12,25 SAY 'ITJ kód     ' GET ITJ PICTURE '@R 99-99-99'
@ 13,25 SAY 'Menny. egys.' GET MENNY_EGYS
@ 14,25 SAY 'Egységár    ' GET EGYS_AR
```

Az első sor egy keretet rajzol aminek bal felső sarka a 9,20 koordinátán a jobb alsó sarka a 15,59 koordinátán található. Ha dupla vonalú keretre van szükségünk, akkor írjuk még a parancs után a DOUBLE szót. A második sor szöveg a Cikkszám szöveget írja a képernyőre, úgy hogy első betűje a 10,25 koordinátára kerüljön. A harmadik sor a CIKKSZAM mező bekérésének helyét adja meg (10,38) és egy maszkot a mező bekéréséhez. A PICTURE után idézőjelek között a három darab kilences azt jelenti, hogy itt három darab számjegyet írhatunk be. A PICTURE utasítás paraméterei a függelékben részletesen fel vannak sorolva. Tehát a SAY kiírást, a GET bekérést jelent. A hatodik sor (ITJ) arra példa, hogy a SAY utasítás mögé írhatunk egy GET-et is, így megspóroljuk a bekérési koordi-

náták megadását. A bekérés helye ilyenkor automatikusan a SAY-jel kiírt szöveg mögött lesz. Most értelmezzük a listát a függelék segítségével, majd írjuk be egy CIKK.FMT nevű file-ba. Ezt megtehetjük a FOXBASE-ből kilépve bármilyen ASCII file-okat kezelő szövegszerkesztővel (pl.: Norton Editor), vagy a FOXBASE beépített szövegszerkesztőjével. A beépített szövegszerkesztő hívható a MODIFY COMMAND CIKK.FMT paranccsal, a szöveg beírása után pedig kiléphetünk belőle a Ctrl End gombokkal. A beépített szövegszerkesztő elég gyenge, ezért hosszabb formátum állományok, vagy programok írására inkább más szövegszerkesztőt használjunk.

A file megírása és a FOXBASE-be való visszalépés után adjuk ki a

. USE CIKK INDEX CIKK
. SET FORMAT TO CIKK.FMT
. EDIT

A SET FORMAT paranccsal utasítjuk a FOXBASE-t arra, hogy a megadott formátum-állományt használja az APPEND és EDIT parancsoknál. Próbáljunk formátum-állományt írni a másik két állományhoz is.

3.19 A változókról

Eddig az adatok tárolására csak az adatállományok mezőit használtuk. Adatok ideiglenes tárolására használhatunk változókat. A változóknak a mezőkhöz hasonlóan valamilyen (numerikus, karakteres, logikai) típusuk van, de ezt nem kell előre megadnunk, az első értékadáskor automatikusan az egyenlőségjel jobb oldalán lévő kifejezés értékét és típusát veszi át.

. A=5
. B=3

A két értékadás után A értéke 5, B értéke 3 lett. Mindkét változó típusa numerikus.

. ? A+B

A változókkal végezhetünk műveleteket, a ? (kiírás) paranccsal kiírhatjuk azok értékét, vagy értékül adhatjuk egy másik változónak:

. C=A+B
. ? C*2

A példaprogramban több példát is láthatunk a változók használatára.

3.20 Programozás, program utasítások

Ha már többet használjuk a FOXBASE-t, kiderül, hogy sok parancsot rendszeresen ugyanúgy gépelünk be egymás után. Például, ha az előző feladatra gondolunk, akkor a FOXBASE-be való belépés után mindig ugyanazokat a parancsokat adjuk ki:

. SELECT A
. USE CIKK INDEX CIKK
. SET FORMAT TO CIKK.FMT
. SELECT B
. USE ELADAS INDEX ELADAS
. SET FORMAT TO ELADAS.FMT
. SELECT C
. USE VEVOK INDEX VEVOK
. SET FORMAT TO VEVOK.FMT
. SELECT B

Ha parancsainkat beírjuk egy .PRG kiterjesztésű file-ba, azt egy utasítás kiadásával végrehajthatjuk. Írjuk be az előző parancsokat egy NYIT.PRG file-ba, majd adjuk ki a

. DO NYIT

parancsot, erre a file-ban található összes parancs végrehajtódik.

Ez természetesen még nem sokkal különb a DOS-ban használt batch file-oknál, hiszen nem tartalmaz ciklust vagy feltételt. A ciklusok, feltételek megértésére jó példa van a függelékben.

Ha a FOXBASE parancsait már jól begyakoroltuk, nézzük át a következő példaprogramot. Ez az előzőekben létrehozott nyilvántartási feladatot oldja meg. Biztos vagyok benne, hogy a függelék sűrű lapozgatásával mindenkinek sikerül a programot megértenie és hasonló méretű feladatra saját programot írnia.

```
SET TALK OFF
SET SCOREBOARD OFF
SET STATUS OFF
SET DATE ANSI
SET DELETED ON
SET PROCEDURE TO BOLT.PRG
SET COLOR TO BG/N,GR+/R,N
IF .NOT. FILE('CIKK.DBF')
  ? 'Hiányzik az CIKK.DBF file!'
  QUIT
ENDIF
IF .NOT. FILE('ELADAS.DBF')
  ? 'Hiányzik az ELADAS.DBF file!'
  QUIT
ENDIF
IF .NOT. FILE('VEVOK.DBF')
  ? 'Hiányzik az VEVOK.DBF file!'
  QUIT
ENDIF
SELECT A
IF .NOT. FILE('CIKK.IDX')
  USE CIKK
  INDEX ON CIKKSZAM TO CIKK
 ELSE
  USE CIKK INDEX CIKK
ENDIF
SELECT B
IF .NOT. FILE('ELADAS.IDX')
  USE ELADAS
  INDEX ON DATUM TO ELADAS
 ELSE
  USE ELADAS INDEX ELADAS
ENDIF
SELECT C
IF .NOT. FILE('VEVOK.IDX')
  USE VEVOK
  INDEX ON KOD TO VEVOK
 ELSE
  USE VEVOK INDEX VEVOK
ENDIF
SELECT B
SET RELATION TO CIKKSZAM INTO CIKK
SET RELATION TO VEVOKOD INTO VEVOK ADDITIVE
FOMENU=0
DO WHILE FOMENU<>6
  CLEAR
  @  2,30 TO 4,49 DOUBLE
  @  3,37 SAY 'Fömenü'
  @  5,20 TO 19,59 DOUBLE
  @  7,30 PROMPT 'Eladás felvitele    '
  @  9,30 PROMPT 'Beszállítás         '
  @ 11,30 PROMPT 'Arucikkek adatai    '
  @ 13,30 PROMPT 'Vevök adatai        '
  @ 15,30 PROMPT 'Listázás, lekérdezés'
  @ 17,30 PROMPT 'Program vége, kilép '
  @ 20,20 TO 22,59 DOUBLE
  @ 21,22 SAY 'Válassz a nyilakkal és nyomj ENTER-t!'
  MENU TO FOMENU
  DO CASE
    CASE FOMENU=1
      DO ELAD
    CASE FOMENU=2
       DO BESZALL
    CASE FOMENU=3
       DO CKARB
    CASE FOMENU=4
       DO VKARB
    CASE FOMENU=5
       DO LISTA
  ENDCASE
ENDDO
SET TALK ON
SET SCOREBOARD ON
SET STATUS ON
SET DELETED OFF
CLEAR
RETURN
```

```
PROCEDURE ELAD
CLEAR
@ 10,0 TO 15,39
SELECT A
C=0
@ 11,5 SAY 'Cikkszám    ' GET C PICTURE '999'
READ
SEEK C
IF .NOT. FOUND()
  CLEAR
  @ 11,10 SAY 'Ilyen cikkszám nem létezik!'
  ?? CHR(7)
  WAIT ''
  RETURN
ENDIF
@  4,40 TO 11,79
@  5,45 SAY 'Cikkszám    '
@  5,58 SAY C PICTURE '999'
@  6,45 SAY 'Megnevezés  '
@  6,58 SAY MEGNEVEZES
@  7,45 SAY 'ITJ kód     '
@  7,58 SAY ITJ PICTURE '@R 99-99-99'
@  8,45 SAY 'Menny. egys.'
@  8,58 SAY MENNY_EGYS
@  9,45 SAY 'Egységár    '
@  9,58 SAY EGYS_AR
@ 10,45 SAY 'Készlet     '
@ 10,58 SAY KESZLET
KESZL=KESZLET
SELECT C
K=0
@ 12,5 SAY 'Vevökód     ' GET K PICTURE '999'
READ
SEEK K
IF .NOT. FOUND()
  CLEAR
  @ 12,10 SAY 'Ilyen vevö nem létezik!'
  ?? CHR(7)
  WAIT ''
  RETURN
ENDIF
@ 14,40 TO 19,79
@ 15,45 SAY 'Kód         '
@ 15,58 SAY K PICTURE '999'
@ 16,45 SAY 'Megnevezés  '
@ 16,58 SAY MEGNEVEZES
@ 17,45 SAY 'Cím         '
@ 17,58 SAY CIM
@ 18,45 SAY 'Számlaszám  '
@ 18,58 SAY SZLASZ PICTURE '@!'
SELECT B
APPEND BLANK
REPLACE CIKKSZAM WITH C,VEVOKOD WITH K,DATUM WITH DATE()
DO WHILE .T.
  @ 13,5 SAY 'Dátum       ' GET DATUM
  @ 14,5 SAY 'Mennyiség   ' GET MENNYISEG
  READ
  @ 16,5 SAY '                '
  IF READKEY()=12 .OR. READKEY()=268
    DELETE
    EXIT
  ENDIF
  IF MENNYISEG=0
    @ 16,5 SAY 'Nulla a mennyiség!'
    ?? CHR(7)
    LOOP
  ENDIF
  IF DATUM<CTOD('1980.01.01')
    @ 16,5 SAY 'Hibás a dátum!'
    ?? CHR(7)
    LOOP
  ENDIF
  IF MENNYISEG<=KESZL
    ME=MENNYISEG
    SELECT A
    SEEK C
    REPLACE KESZLET WITH KESZLET-ME
```

```
      EXIT
    ELSE
      @ 16,5 SAY 'Negatív készlet!'
      ?? CHR(7)
      LOOP
    ENDIF
  ENDDO
  RETURN


  PROCEDURE BESZALL
  CLEAR
  @  9,20 TO 18,59
  SELECT A
  C=0
  @ 10,25 SAY 'Cikkszám    ' GET C PICTURE '999'
  READ
  SEEK C
  IF .NOT. FOUND()
    CLEAR
    @ 10,30 SAY 'Ilyen cikkszám nem létezik!'
    ?? CHR(7)
    WAIT ''
    RETURN
  ENDIF
  ME=0
  @ 11,25 SAY 'Megnevezés  '
  @ 11,38 SAY MEGNEVEZES
  @ 12,25 SAY 'ITJ kód     '
  @ 12,38 SAY ITJ PICTURE '@R 99-99-99'
  @ 13,25 SAY 'Menny. egys.'
  @ 13,38 SAY MENNY_EGYS
  @ 14,25 SAY 'Egységár    '
  @ 14,38 SAY EGYS_AR
  @ 15,25 SAY 'Készlet     '
  @ 15,38 SAY KESZLET
  @ 16,25 SAY 'Szállított  '
  @ 17,25 SAY '   mennyiség' GET ME PICTURE '999999.99'
  READ
  REPLACE KESZLET WITH KESZLET+ME
  RETURN


  PROCEDURE CKARB
  ALMENU=0
  SELECT A
  DO WHILE ALMENU<>5
    CLEAR
    @  2,30 TO 4,49 DOUBLE
    @  3,35 SAY 'Árucikkek'
    @  5,20 TO 17,59 DOUBLE
    @  7,30 PROMPT 'Felvitel          '
    @  9,30 PROMPT 'Módosítás         '
    @ 11,30 PROMPT 'Törlés            '
    @ 13,30 PROMPT 'Listázás          '
    @ 15,30 PROMPT 'Vissza a fömenühöz  '
    @ 18,20 TO 20,59 DOUBLE
    @ 19,22 SAY 'Válassz a nyilakkal és nyomj ENTER-t!'
    MENU TO ALMENU
    DO CASE
      CASE ALMENU=1
          CLEAR
          C=0
          @  9,20 TO 15,59
          @ 10,25 SAY 'Cikkszám    ' GET C PICTURE '999'
          READ
          SEEK C
          IF FOUND()
            CLEAR
            @ 10,30 SAY 'Ilyen cikkszám már létezik!'
            ?? CHR(7)
            WAIT ''
            LOOP
          ENDIF
          APPEND BLANK
          REPLACE CIKKSZAM WITH C,KESZLET WITH 0
```

```
        @ 11,25 SAY 'Megnevezés  ' GET MEGNEVEZES
        @ 12,25 SAY 'ITJ kód     ' GET ITJ PICTURE '@R 99-99-99'
        @ 13,25 SAY 'Menny. egys.' GET MENNY_EGYS
        @ 14,25 SAY 'Egységár    ' GET EGYS_AR
        READ
    CASE ALMENU=2
        CLEAR
        C=0
        @  9,20 TO 15,59
        @ 10,25 SAY 'Cikkszám    ' GET C PICTURE '999'
        READ
        SEEK C
        IF .NOT. FOUND()
          CLEAR
          @ 10,30 SAY 'Ilyen cikkszám nem létezik!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        @ 11,25 SAY 'Megnevezés  ' GET MEGNEVEZES
        @ 12,25 SAY 'ITJ kód     ' GET ITJ PICTURE '@R 99-99-99'
        @ 13,25 SAY 'Menny. egys.' GET MENNY_EGYS
        @ 14,25 SAY 'Egységár    ' GET EGYS_AR
        READ
    CASE ALMENU=3
        CLEAR
        C=0
        @  9,20 TO 15,59
        @ 10,25 SAY 'Cikkszám    ' GET C PICTURE '999'
        READ
        SEEK C
        IF .NOT. FOUND()
          CLEAR
          @ 10,30 SAY 'Ilyen cikkszám nem létezik!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        IF KESZLET>0
          CLEAR
          @ 10,30 SAY 'Ezen a cikkszámon van készlet!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        DELETE
    CASE ALMENU=4
        CLEAR
        DISPLAY ALL OFF
        WAIT 'Nyomj ENTER-t!'
  ENDCASE
ENDDO
PACK
RETURN


PROCEDURE VKARB
ALMENU=0
SELECT C
DO WHILE ALMENU<>5
  CLEAR
  @  2,30 TO 4,49 DOUBLE
  @  3,37 SAY 'Vevök'
  @  5,20 TO 17,59 DOUBLE
  @  7,30 PROMPT 'Felvitel          '
  @  9,30 PROMPT 'Módosítás         '
  @ 11,30 PROMPT 'Törlés            '
  @ 13,30 PROMPT 'Listázás          '
  @ 15,30 PROMPT 'Vissza a fömenühöz  '
  @ 18,20 TO 20,59 DOUBLE
  @ 19,22 SAY 'Válassz a nyilakkal és nyomj ENTER-t!'
  MENU TO ALMENU
  DO CASE
    CASE ALMENU=1
       CLEAR
       K=0
```

```
        @  9,20 TO 14,59
        @ 10,25 SAY 'Kód         ' GET K PICTURE '999'
        READ
        SEEK K
        IF FOUND()
          CLEAR
          @ 10,30 SAY 'Ilyen vevö már létezik!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        APPEND BLANK
        REPLACE KOD WITH K
        @ 11,25 SAY 'Megnevezés  ' GET MEGNEVEZES
        @ 12,25 SAY 'Cím         ' GET CIM
        @ 13,25 SAY 'Számlaszám  ' GET SZLASZ PICTURE '@!'
        READ
    CASE ALMENU=2
        CLEAR
        K=0
        @  9,20 TO 14,59
        @ 10,25 SAY 'Kód         ' GET K PICTURE '999'
        READ
        SEEK K
        IF .NOT. FOUND()
          CLEAR
          @ 10,30 SAY 'Ilyen vevö nem létezik!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        @ 11,25 SAY 'Megnevezés  ' GET MEGNEVEZES
        @ 12,25 SAY 'Cím         ' GET CIM
        @ 13,25 SAY 'Számlaszám  ' GET SZLASZ PICTURE '@!'
        READ
    CASE ALMENU=3
        CLEAR
        K=0
        @  9,20 TO 14,59
        @ 10,25 SAY 'Kód         ' GET K PICTURE '999'
        READ
        SEEK K
        IF .NOT. FOUND()
          CLEAR
          @ 10,30 SAY 'Ilyen vevö nem létezik!'
          ?? CHR(7)
          WAIT ''
          LOOP
        ENDIF
        SELECT B
        LOCATE FOR VEVOKOD=K
        IF FOUND()
          CLEAR
          @ 10,30 SAY 'Ennek a vevönek már adtunk árut!'
          ?? CHR(7)
          WAIT ''
          SELECT C
          LOOP
        ENDIF
        SELECT C
        SEEK K
        DELETE
    CASE ALMENU=4
        CLEAR
        DISPLAY ALL OFF
        WAIT 'Nyomj ENTER-t!'
  ENDCASE
ENDDO
PACK
RETURN


PROCEDURE LISTA
ALMENU=0
DO WHILE ALMENU<>5
  CLEAR
```

```
@  2,30 TO 4,49 DOUBLE
@  3,37 SAY 'Listák'
@  5,20 TO 17,59 DOUBLE
@  7,25 PROMPT '1  Cikkszám szerint képernyöre'
@  9,25 PROMPT '2  Cikkszám szerint nyomtatóra'
@ 11,25 PROMPT '3  Vevö szerint képernyöre   '
@ 13,25 PROMPT '4  Vevö szerint nyomtatóra   '
@ 15,25 PROMPT 'V  Vissza a fömenühöz        '
@ 18,20 TO 20,59 DOUBLE
@ 19,22 SAY 'Válassz a nyilakkal és nyomj ENTER-t!'
MENU TO ALMENU
IF ALMENU=1 .OR. ALMENU=2
  CLEAR
  SELECT A
  C=0
  DK=CTOD('80.01.01')
  DV=DATE()
  @ 11,25 SAY 'Cikkszám      ' GET C PICTURE '999'
  @ 13,25 SAY 'Idöszak kezdete' GET DK
  @ 15,25 SAY 'Idöszak vége   ' GET DV
  READ
  SEEK C
  IF .NOT. FOUND()
    CLEAR
    @ 11,10 SAY 'Ilyen cikkszám nem létezik!'
    ?? CHR(7)
    WAIT ''
    RETURN
  ENDIF
ENDIF
IF ALMENU=3 .OR. ALMENU=4
  CLEAR
  SELECT C
  K=0
  DK=CTOD('80.01.01')
  DV=DATE()
  @ 11,25 SAY 'Vevökód       ' GET K PICTURE '999'
  @ 13,25 SAY 'Idöszak kezdete' GET DK
  @ 15,25 SAY 'Idöszak vége   ' GET DV
  READ
  SEEK K
  IF .NOT. FOUND()
    CLEAR
    @ 11,10 SAY 'Ilyen vevö nem létezik!'
    ?? CHR(7)
    WAIT ''
    RETURN
  ENDIF
ENDIF
DO CASE
  CASE ALMENU=1
     CLEAR
     ? 'Eladások            ',DK,' - ',DV
     ?
     AR=EGYS_AR
     MS=MENNY_EGYS
     DISP OFF
     SELECT B
     SET FILTER TO C=CIKKSZAM .AND. DATUM>=DK .AND. DATUM<=DV MENNYISEG TO ME
     ?
     SUM MENNYISEG TO ME
     ? 'Eladások mennyisége :  ',TRANSFORM(ME,'@R 999999.99'),MS
     ?
     ? 'Eladások értéke     : ',TRANSFORM(ME*AR,'@R 9999999.99 Ft')
     ?
     WAIT 'Nyomj ENTER-T!'
     CLEAR
     DISPLAY ALL DATUM,MENNYISEG,VEVOKOD,VEVOK->MEGNEVEZES,VEVOK->SZLASZ OFF
     WAIT 'Nyomj ENTER-T!'
     SET FILTER TO
  CASE ALMENU=2
     SET CONSOLE OFF
     SET PRINT ON
     ? 'Eladások            ',DK,' - ',DV
     ?
     AR=EGYS_AR
     MS=MENNY_EGYS
```

```
        DISP OFF
        SELECT B
        SET FILTER TO C=CIKKSZAM .AND. DATUM>=DK .AND. DATUM<=DV MENNYISEG TO ME
        ?
        SUM MENNYISEG TO ME
        ? 'Eladások mennyisége :  ',TRANSFORM(ME,'@R 999999.99'),MS
        ?
        ? 'Eladások értéke     : ',TRANSFORM(ME*AR,'@R 9999999.99 Ft')
        ?
        ?
        LIST ALL DATUM,MENNYISEG,VEVOKOD,VEVOK->MEGNEVEZES,VEVOK->SZLASZ OFF
        SET FILTER TO
        SET CONSOLE ON
        SET PRINT OFF
    CASE ALMENU=3
        CLEAR
        ? 'Eladások            ',DK,' - ',DV
        ?
        DISP OFF
        SELECT B
        SET FILTER TO K=VEVOKOD .AND. DATUM>=DK .AND. DATUM<=DV MENNYISEG TO ME
        ?
        SUM MENNYISEG*CIKK->EGYS_AR TO ERT
        ? 'Eladások értéke     : ',TRANSFORM(ERT,'@R 9999999.99 Ft')
        ?
        WAIT 'Nyomj ENTER-T!'
        CLEAR
        DISPLAY ALL DATUM,MENNYISEG,CIKKSZAM,CIKK->MEGNEVEZES,CIKK->ITJ OFF
        WAIT 'Nyomj ENTER-T!'
        SET FILTER TO
    CASE ALMENU=4
        SET CONSOLE OFF
        SET PRINT ON
        CLEAR
        ? 'Eladások            ',DK,' - ',DV
        ?
        DISP OFF
        SELECT B
        SET FILTER TO K=VEVOKOD .AND. DATUM>=DK .AND. DATUM<=DV MENNYISEG TO ME
        ?
        SUM MENNYISEG*CIKK->EGYS_AR TO ERT
        ? 'Eladások értéke     : ',TRANSFORM(ERT,'@R 9999999.99 Ft')
        ?
        ?
        CLEAR
        LIST ALL DATUM,MENNYISEG,CIKKSZAM,CIKK->MEGNEVEZES,CIKK->ITJ OFF
        SET FILTER TO
        SET CONSOLE ON
        SET PRINT OFF
  ENDCASE
ENDDO
RETURN
```

3.21 Függelék a 3. fejezethez

3.21.1 Jelölések és gyakoribb kifejezések

NAGYbetű	FoxBase+ kulcsszava (elegendő az első négy betű beírása)
n	egy egész szám
< >	jelek közötti szöveget a programozó adja meg
[ ]	a jelek közötti rész beírása nem kötelező (elhagyható)
... | ...	a " |" jellel elválasztott utasítások közül az egyik megadása kötelező

A fenti < >, [ ], | jelek csak értelmezésre szolgálnak, ezért nem írhatók be.

Az érvényességi kör megadja, hogy az adatbázis mely rekordjaira érvényes az adott tevékenység.

RECORD<n>	az "n" sorszámú rekord
NEXT<n>	az aktuálistól (az is beleértve) "n" darab rekord
REST	aktuálistól az állomány végéig az összes rekord
ALL	az összes rekord
FOR <feltétel>	az elsőtől kezdve minden olyan rekord, melyre "IGAZ" a feltétel
WHILE<feltétel>	az aktuálistól kezdve minden olyan rekord, melyre "IGAZ" a feltétel, de az első "HAMIS"-nál megáll

3.21.2 Operátorok és prioritási sorrendjük

Adott kifejezésben szereplő műveletek BALról JOBBra haladva prioritási sorrendben hajtódnak végre úgy, hogy először a zárójelek "()" közötti számítások kerülnek végrehajtásra.

Aritmetikai operátorok

Numerikus adatok közötti műveletek végzése.

** vagy ^	hatványozás (pl.: 2**3, de használható 2^3 alak is)
* vagy /	szorzás, vagy az osztás (pl.: 2*3, 2/3)
+ vagy -	összeadás vagy a kivonás (pl.: 2+3, 2-3)

Relációs operátorok

Azonos típusú adatok összehasonlítása, melynek eredménye logikailag IGAZ, vagy HAMIS.

Pl.: "egyik" <= "másik" reláció jelentése
Állítom, hogy az "egyik" KISEBB VAGY EGYENLŐ mint a "másik"

< , <=	kisebb mint, kisebb vagy egyenlő mint
>, >=	nagyobb mint, nagyobb vagy egyenlő mint
=	egyenlő
<> #	nem egyenlő
$	a jel baloldalán lévő jelsorozat része (megtalálható) a jobboldalinak
==	karakteres kifejezések pontos egyezése (szóközszámra is)

3.21.3 Logikai műveletek és igazságtáblázataik

.NOT.	a mögötte álló logikai értéket ellentétesre változtatja (egy operandusú művelet)
.AND.	logikai "ÉS", melynek eredménye csak akkor IGAZ, ha minden eleme IGAZ
.OR.	logikai "VAGY", melynek eredménye IGAZ, ha legalább egyik eleme IGAZ

"A" állítás	"B" állítás	A .AND. B	A .OR. B
igaz	igaz	igaz	igaz
igaz	hamis	hamis	igaz
hamis	igaz	hamis	igaz
hamis	hamis	hamis	hamis

3.21.4 Rövidítések

- **Dkif** dátum típusú kifejezés
- **Kkif** karakteres típusú kifejezés
- **Lkif** logikai típusú kifejezés
- **Nkif** numerikus típusú kifejezés
- **érv.kör** érvényességi kör
- **hiv.név** hivatkozási név
- **kif.lista** kifejezések listája
- **mem.vált** változónév

3.21.5 FoxBase+ parancsai

& <mem.vált>[.]
A mem.vált. tartalmát behelyettesíti az adott helyre, majd az így kialakított utasítást hajtja végre

?[?] <kif.lista>
Kiíratás képernyőre/nyomtatóra. A "?" nyomtatás előtti soremelést jelent

@<sor,oszlop>
Képernyő/nyomtató kiírási sor,oszlop helyének beállítása. Nyomtatónál az aktuálisnál kisebb "sor" megadása lapdobást is jelent

@<sor,oszlop> SAY<kif.lista> [FUNTION<Kkif>] [PICTURE<Kkif>]
A megadott helyen a kif.lista-ban megadottakat a Kkif- formában írja ki.
Pl.: @ 5,10 SAY ber PICTURE "@R 9999999.99 Ft"

@ <sor,oszlop> GET<változó> [VALID<Lkif>] [FUNTION<Kkif>] [PICTURE<Kkif>] [RANGE<alsó>]
A képernyő adott helyére kiírja a változó (memóriaváltozó/adatbázismező) tartalmát és az a legközelebbi READ után módosulhat.
Pl.:@ 5,10 GET ber PICTURE "@R 9999999.99 Ft"

@ <sor1,oszlop1> TO <sor2,oszlop2> [DOUBLE]
A képernyő adott területét vonallal keretezi be (kettős)
Pl.: @ 5,10 TO 14,20 DOUBLE

@ <sor1,oszlop1> <sor2,oszlop2> BOX <Kkif>
A képernyő megfelelő helyére a (max. 9 karakteres) Kkif-ben megadottnak megfelelő karakterekkel keretet rajzol és azt kitölti
Pl.: @ 5,7,9,10 BOX "123456780" hatása a képernyőn

```
12223
80004
80004
76665
```

@ <sor,oszlop>CLEAR [TO <sor2,oszlop2>]
Törli a képernyő adott területét
Pl.: @ 5,10 CLEAR TO 14,20

@ <sor,oszlop> PROMPT <Kkif> [MESSAGE <Kkif>]
Menü készítése, melyből a megadott elem könnyen kiválasztható. A MESSAGE üzeneteket kapcsol a menüpontokhoz

FUNCTION és PICTURE Kkif karakterei

@A	GET-nél csak az angol abc betűit írhatja be
@B	Balra igazít
@C	Pozitív szám után CR (Credit) betűket ír
@D	Adott dátumforma szerinti kiírás, GET-nél ellenőrzi a dátumot
@E	Európai dátumformát használ
@R	Maszkkarakterektől eltérő karakterek megjelennek, de nem tárlódnak és nem módosíthatók
@S<n>	n számu karakter széles ablak, ebben gördíthető az adatunk
@Y	Y vagy N jelenik meg a logikai T vagy F helyett
@Z	0 helyett szóközöket ír
@(	A negatív számok zárójelben lesznek
@!	Nagybetűvé alakítja az angol abc betűit
A	GET-nél csak az angol abc betűit fogadja el (A-Z és a-z)
L	GET-nél csak logikai értéket jelző betűk írhatók be (Y,N,T,F)
N	Csak az angol abc betűi és számok írhatók be
X	Tetszőleges karakter
Y	Nagybetűre alakítja az y vagy az n jelet, más nem is írható be
!	Nagybetűre alakít
$	Nyomtatáskor felülírja az adott helyen lévő karaktert
*	Szám előtti nullák helyén * jelenik meg
.	Tizedespont helye, a "," szeparátorként használható
#	Számjegyek, előjel illetve szóköz helye
9	Számjegyek, előjel helye

ACCEPT [üzenet] TO <mem.vált>

Billentyűzetről karaktersorozat írható be a mem.vál-ba

APPEND [BLANK]

Az aktív adatbázis végére egy új rekordot ír

APPEND FROM <állomány> [FIELDS<mezőlista>] [FOR <feltétel>] [[TYPE] SDF | DELIMITED [WITH <határoló> | BLANK]]

Az aktív adatbázishoz rekordokat fűz a megadott "állomány"-ból.

- ☐ ha nem adunk FOR feltételt akkor az összes rekord átvitelre kerül
- ☐ FIELDS mezőlistában megadhatja, hogy az aktív adatállomány mely mezőit töltse ki.
- ☐ DELIMITED az "állomány" mezőit vessző választja el, de a WITH határolóval ez megváltoztatható, BLANK a vesszőt szóközre cseréli és a szövegtípusú mezőket idézőjelek határolják
- ☐ Az állomány kiterjesztése DBF, de TYPE esetén TXT
- ☐ SDF határolójelek nélküli fix hosszúságú rekordok olvashatók be

Pl.:APPEND FROM ujadat FIELDS nev, szulido DELIMITED TYPE

AVERAGE [<kif.lista>] [<érv.kör>] [FOR <feltétel>] [WHILE <feltétel>] [TO <mem.vált. lista>]

Meghatározza a kif.lista-ban megadott és a feltételi körbe eső mezők átlagát, mely a TO utáni mem.vált.-ba írható. Pl.: AVERAGE ber FOR neme="2" TO atlagber

BROWSE [FIELDS<mezőlista>] [LOCK<Nkif>] [FREEZE<mező>] [NOMODIFY] [NOFOLLOW] [NOAPPEND] [NOMENU] [WITH<Nkif>]

Teljesképernyős parancs az adatbázis tartalmának megtekintésére, módosítására

- ☐ FIELDS — képernyőre kerülő mezők nevei és sorrendjük
- ☐ LOCK — a képernyőn állandóan látszó mezők száma
- ☐ FREEZE — csak az itt megadott mező tartalma módosítható
- ☐ NOAPPEND — az állomány nem bővíthető
- ☐ NOFOLLOW — indexelt állományoknál a mezőtartalom módosítása után is az eredetit követő rekordra lép (nem ugrál)
- ☐ NOMODIFY — mezők tartalma nem módosítható
- ☐ NOMENU — letiltja az F10 gombbal előhívható menü használatát
- ☐ WIDTH — mezőszélesség beállítása a képernyőn (max. 80)

Pl.: BROWSE FIELDS nev,fizetes FREEZE fizetes NOMENU NOAPPEND

CALL <modulnév> [WITH <Kkif> | <mem.vált>]

Gépi kódú külső program meghívása, a WITH paraméterátadást tesz lehetővé

CANCEL

FoxBase+ program futás megszakítása és visszatérés parancsüzemmódba (mem.változók tartalma törlődik)

CHANGE [FIELDS <mezőlista>] [<érv.kör>] [WHILE<feltétel>] [FOR<feltétel>]

Azonos az EDIT paranccsal

CLEAR

Törli a képernyőt, kurzor a balfelső sarokba kerül

CLEAR ALL

Zárja az állományokat, törli a memóriaváltozókat

CLEAR GETS

Törli az utolsó READ után aktivizálódott összes @ . . . GET utasítással előírt módosítási feladatot

CLEAR MEMORY

Memóriaváltozók törlése

CLEAR PROGRAM

Memóriából törli a FoxBase+ programokat

CLEAR TYPEAHEAD

Törli a billentyűzetpuffer tartalmát

CLOSE ALL

Típustól függetlenül zár minden állományt

CLOSE ALTERNATE

Zárja a SET ALTERNATE paranccsal nyitott állományokat

CLOSE DATABASES

Zárja az adatbázis-állományokat

CLOSE FORMAT

Zárja az aktív adatbázishoz tartozó formátumállományt

CLOSE INDEX

Zárja az aktív adatbázishoz tartozó indexállományt

CLOSE PROCEDURE

Zárja az aktív procedúra állományt

CONTINUE

A feltételnek megfelelő következő rekordra áll, ha nincs ilyen akkor a FOUND()"F" logikai érket ad vissza

```
Pl.:    LOCATE FOR ber<5000   végrehajtása után
        RECORD = 7
        CONTINUE              megkeresi a következőt
```

COPY FILE <forrásállomány> TO <célállomány>

Másolatot készít Pl.: COPY FILE torzs TO torzsmen

COPY TO <új állomány> [FIELDS <mezőlista>] [<érv.kör>] [WHILE <feltétel>] [FOR<feltétel>] [[TYPE] [SDF | [DELIMITED [WITH<határoló> I BLANK]]]]

Aktív adatbázisról készít másolatot. Az "új állomány"-ba az összes illetve a feltételnek megfelelő rekordok kerülnek, ha nem ad FIELDS paramétert akkor az összes mezőadat másolásra kerül

- ☐ TYPE — az új állomány kiterjesztése .TXT lesz
- ☐ SDF — fix hosszúságú rekordokat ír az új állományba, nincs mezőelválasztó karakter

- ☐ **DELIMITED** rekordok hossza változó lehet
- ☐ **WITH** alapértelmezésben a mezőadatokat vessző választja el egymástól, ez írható át tetszőleges karakterre
- ☐ **BLANK** a mezőelválasztó a szóköz

Pl.: COPY TO nev FIELDS nev,ber ALL TYPE SDF

COPY STRUCTURE TO <új állomány> [FIELDS<mezőlista>]

Az aktív adatbázis szerkezetét másolja az "új állomány"-ba, ha van mezőlista akkor csak az abban felsoroltak kerülnek be az új szerkezetbe

COPY STUCTURE EXTENDED TO < állomány>

Új adatbázis létrehozását készíti elő

COUNT [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>] [TO <mem.vált>]

Megadja az aktív adatbázis adott feltételeknek megfelelő rekordjainak számát, ezt az értéket a megadott mem.vált.-ba is tárolhatja

CREATE [<állomány>]

Új adatállomány létrehozása

- ☐ **Field name** mezőnévben csak az angol abc betűi és
- ☐ az "aláhúzás" szerepelhet
- ☐ **Type** adattípus kiválasztása betűkkel (c, d, l, n, m), vagy a kurzormozgató gombbal
- ☐ **Width** mezőszélesség (betűk vagy számjegyek száma), numerikus adatnál egészek+tizedespont+tizedesek
- ☐ **Decimal Places** tizedesjegyek száma (csak numerikus adattípusnál)

CREATE <új állomány> FROM <szerkesztett állomány>

A COPY STRUCTURE EXTENDED-del létrehozott állomány alapján hozza létre az "új állomány"-t

CREATE LABEL [<állomány>]

A LABEL parancshoz szükséges címkeformátum-állomány létrehozása, az állomány kiterjesztése .LBL lesz

CREATE REPORT [<állomány>]

A REPORT parancshoz listaformátum-állomány létrehozása, az állomány kiterjesztése .FRM lesz

DELETE [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>]

Az aktív adatbázis feltételeknek megfelelő rekordjait törlésre jelöli, de tényleges törlés csak a PACK parancs hatására történik

Pl.: DELE FOR ber<=0

DELETE FILE <állomány>

Tetszőleges adatállomány törlése, csak zárt állományt lehet törölni, globális nevek nem használhatók

Pl.: DELE FILE torzs.dbf

DIMENSION <tömbnév>(méret1 [, méret2]) [, ...]

Tömb definiálása, max. kétdimenziós lehet a megadott tömb

Pl: DIMENSION tabla(20,3), nevsor(50)

DIR [<meghajtó>][< elérési út>][< globális állománynév>] [TO PRINT]

Könyvtár tartalomjegyzékének kiíratása, alapértelmezésben képernyőre írja ki a .DBF kiterjesztésű állományok adatait, ez átirányítható lemezre is és nyomtatóra is

DISPLAY [[FIELDS] < kif.lista> [OFF] [< érv.kör>] [WHILE <feltétel>] [FOR <feltétel>] [TO PRINT]

Adatmegjelenítés képernyőre, az OFF használatánál a rekordsorszámokat nem írja ki, a kiírás átirányítható más meghajtóra (lemez, nyomtató)

Pl.: DISPLAY nev,ber FOR ber>10000.AND.ber<20000

DISPLAY FILES [ON <meghajtó | könyvtár>] [LIKE <globális állománynév>] [TO PRINT]

Hatása azonos a DIR parancséval

Pl.: DISPLAY FILES a:\tanulo\ind*.prg

DISPLAY HISTORY [LAST <Nkif>][TO PRINT]

Kiírja az utoljára kiadott parancsokat

DISPLAY MEMORY TO PRINT

Mem.változók tartalma íratható ki

DISPLAY STATUS TO PRINT

A megnyitott adatállományokról, SET és ON beállításokról ad tájékoztatást

DISPLAY STRUCTURE TO PRINT

Az aktív adatbázis szerkezetéről (mezőadatok) ad információt

DO <állomány | procedúra> [WITH <paraméterlista>]

Program vagy eljárás indítása

Pl.: DO indul

DO CASE

CASE <feltétel1>

<utasítások>

CASE <feltétel 2>

<utasítások>

OTHERWISE

<utasítások>

ENDCASE

Többszörös elágazás a feltételeknek megfelelő esetszétválasztás szerint

```
Pl.:    DO CASE
            CASE ber<4000
                ? "kicsi pénz"
            CASE ber>=4000.AND.ber<15000
                ? "atlagos pénz"
            CASE ber>=15000
                ? "ez már több pénz"
            OTHERWISE
                ? "ez nem lehet"
        ENDCASE
```

DO WHILE <feltétel>

<utasítások>

ENDDO

Ciklusszervező parancs melynek utasításai mindaddig végrehajtódnak, amíg a "feltétel" logikai értéke IGAZ, ennek kezdeti beállításáról is és a kilépés módjáról is gondoskodni kell

```
Pl.:    mx=10                   && kezdőérték beállítása
        DO WHILE mx > 0
            ? mx
            mx=mx -1            && kilépés beállítása
        ENDDO
```

Ez a programrész 10-től 1-ig kiírja az egész számokat

EDIT <érv.kör> [FIELDS <mezőlista>] [FOR <feltétel>]

Rekordtartalom módosítása teljesképernyő-szerkesztő módon

EJECT

Lapdobás a nyomtatón, nullázza a PCOL() és a PROW() értékét

ERASE <állomány>

Hatása azonos a DELETE FILE paranccsal

EXIT

Kiugrás egy DO WHILE ciklusból az ENDDO-t követő parancsra

FIND <karaktersorozat> | <Nkif>

Adott tulajdonságú rekord megkeresése indexelt állományban, melynek indexkulcsa azonos a keresettel, sikeres keresés esetén a FOUND() IGAZ értéket ad vissza

FLUSH

Lemezre írja az I/O bufferek tartalmát úgy, hogy minden állományt lezár, majd újra megnyit

GATHER FROM <tömb> [FIELDS <mezőlista>]

A tömb elemeit írja be az aktuális rekordba, illetve annak megadott mezőibe

GO <Nkif> | GOTO TOP | GOTO BOTTOM

Rekordmutató mozgatása az Nkif-ben megadott számúra, vagy az elejére (TOP) illetve, a végére (BOTTOM)

HELP [<kulcsszó>]

Kiírja a HELP "segítő" menü információit

IF <feltétel>
<utasítások1 Feltétel logikai értékétől függő elágazás a programban,
[ELSE ha **IGAZ** akkor az **utasítások1**,
<utasítások2>] különben az **utasítások2** hajtódnak végre
ENDIF

INDEX ON <kulcs> TO <állománynév> [UNIQUE]
Az aktív adatbázis logikai rendezését végzi, kulcsként egy kifejezés adható meg, mely numerikus, karakteres vagy dátum típusú lehet

INPUT [<üzenet>] TO <mem.vált>
Adatbevitel klaviatúráról az adott mem.változóba

INSERT [BEFORE] [BLANK]
Új rekord beszúrása az aktuális mögé és EDIT-hez hasonlóan tölthető ki
- ☐ BEFORE esetén az aktuális elé kerül és kitölthető
- ☐ BLANK üres rekord kerül beírásra

KEYBOARD <Kkif>
A billentyűzet pufferében elhelyezi a Kkif-tartalmát

LABEL FORM <crea-állomány>[WHILE <feltétel>] [FOR <feltétel>] [<érv.kör>] [SAMPLE] [TO PRINT] [TO [FILE]<állomány>]
Adatok kiírása a megadott címkeformátumban, a SAMPLE paraméterrel egy tesztsorozat nyomtatható, mely segíti a beállítást, a crea-állomány helyére a CREATE LABEL-nél megadott állománynevet kell írni

LIST [[FIELDS]<kif.lista>] [OFF] [<érv.kör>]WHILE <feltétel>] [FOR <feltétel>] [TO PRINT]
Az adatbázis rekordjainak és tetszőleges kifejezések megjelenítése, az OFF opció hatására nem írja ki a rekordsorszámokat
Pl: LIST FIELDS nev,ber*12 OFF FOR beosztas="3" TO PRINT

LIST FILES [ON <meghajtó | könyvtár>] [LIKE <globális állománynév>] [TO PRINT]
Hatása azonos a DIR parancséval

LIST HISTORY [LAST <Nkif>] [TO PRINT]
Kiírja az utoljára kiadott parancsokat

LIST MEMORY TO PRINT
Memóriaváltozók tartalma íratható ki

LIST STATUS TO PRINT
A megnyitott adatállományokról, SET és ON beállításokról ad tájékoztatást

LIST STRUCTURE TO PRINT
Az aktív adatbázis szerkezetéről (mezőadatok) ad információt

LOAD <állomány>
Bináris formátumú gépikódú állomány betöltése a memóriába, ennek indítása később a CALL paranccsal történhet

LOCATE [<érv.kör>] [FOR <feltétel>] [WHILE<feltétel>]
Az adott feltételeknek megfelelő rekordok szekvenciális keresése, keresés megáll a feltételeknek megfelelő első rekordon, továbblépés a CONTINUE paranccsal
Pl.: LOCATE FOR ber<7000

LOOP
Hatására a DO WHILE ciklus végére (ENDDO-ra) lép a program és a ciklusfeltételt újból kiértékelve folytatódik az utasítások végrehajtása

MENU TO <mem.vált>
A @....PROMPT -tal írt menüt záró utasítás, a mem.vált megkapja a kiválasztott menüpont sorszámát, ha ESC gombot nyomott le akkor 0-át
Pl.:
```
@ 5,5 PROMPT " 1 Adatbeírás "
@ 6,5 PROMPT " 2 Listák "
@ 7,5 PROMPT " 3 Kilépés "
MENU TO valasztas
```

MODIFY COMMAND | FILE [<állomány>]
FoxBase szövegszerkesztőjének indítása

MODIFY LABEL <állomány>
Hatása azonos a CREATE LABEL parancsal

MODIFY REPORT <állomány>
Hatása azonos a CREATE REPORT parancséval

MODIFY STRUCTURE
Az aktív adatállomány szerkezetének módosítására ad lehetőséget

NOTE | * | && <magyarázat>
Megjegyzések, magyarázatok helyezhetők el a programszövegben, ezek nem vesznek részt a program működésében

ON ERROR [<parancs>]
A FoxBase futáshiba esetén az adott parancs kerül végrehajtásra, ezzel figyelhetjük a perifériáinkat is pl. a nyomtatónkat (papír, bekapcsolás)
Pl.: ON ERROR DO vizsgal

ON ESCAPE [<parancs>]
Az ESC billentyű lenyomása ekkor nem szakítja meg a program futását, hanem elvégzi a parancsban előírt tevékenységet pl. mentéseket
Pl.: ON ESCAPE DO eztcsinal

ON KEY [=<Nkif>][<parancs>]
Előírható egy adott Nkif értékű billentyű lenyomásának hatására végzendő tevékenység, a SET TYPEAHEAD-del 0-nál nagyobb értéket kell beállítani
Pl.: ON KEY=200 DO idemenj

PACK
Az aktív adatbázis törlésre kijelölt rekorjait törli a nyilvántartásából
Pl.: DELETE FOR kilepett="igen"
PACK

PARAMETERS <paraméterlista>
Mem.változók megadása egy eljárásban, vagy sajátfüggvénybe adatok átvitelére
Pl.: PARAMETERS alapfiz, aktivnapok,ossznapok

PRIVATE [ALL [LIKE | EXCEPT <globális mem.vált>]] | [<mem.vált.lista>]
Mem.változók lokális tulajdonságainak megadása, ezek a változók a magadás szintjétől ismertek így a hívó programban lévő azonos nevű mem.változók tartalmát a hívott programban lévő nem változtatja meg

PROCEDURE <eljárásnév>
A procedúra-állományban egy új eljárás kezdetének jelölése, egy programban max. 128 PROCEDURE utasítást lehet elhelyezni

PUBLIC <mem.vált.lista>
Az összes program által használható globális tulajdonságú memóriaváltozók létrehozása

QUIT
Kilépés a FoxBase+ adatbáziskezelőből az operációs rendszerbe (visszatérés a DOS-ba)

READ [SAVE]
A @...GET utasításokkal megadott adatbevitel indítása, ha nincs kitöltetlen @ .. GET utasítás a memóriában akkor a program egy billentyű lenyomására vár, és csak ezután folytatódik az utasítások végrehajtása
SAVE hatására a program tárolja az éppen beolvasott képernyő adatait és azokat a következő READ -del ismét szerkeszthetjük
Pl.: @ 2,20 GET datum PICTURE "@D"
@ 3,20 GET befizet PICTURE "@R 999,999,999.99 Ft"
READ

RECALL [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>]
Rekordok törlési jelének megszüntetése

REINDEX
A kiválasztott munkaterületen megnyitott indexállományok újraépítése

RELEASE [<mem.vált.lista>] [ALL [LIKE | EXCEPT <globális mem.vált>]]
Mem.változók törlése a memóriából, így memóriát szabadíthatunk fel
- ☐ ALL az összes lokális tulajdonságút törli
- ☐ LIKE a globális nevet kielégítőket
- ☐ EXCEPT a globális nevet nem kielégítőket törli

RENAME <régi állománynév> TO < az új állomány neve>
Állomány nevének megváltoztatása, globális nevek nem használhatók
Pl.: RENAME teve.dbf TO pupos.txt

REPLACE <mező> WITH <kifejezés>[,<mező2> WITH <kifejezés2>...] [<érv.kör>] [WHILE <feltétel>] [FOR<feltétel>]

Beírás az adott adatbázismezőkbe, a megfelelő kifejezés és mező típusának azonosnak kell lenni

Pl.: REPLACE FOR ber<8500 ber WITH ber+1500

REPORT FORM <listaállomány neve>[<érv.kör>][WHILE <feltétel>] [FOR <feltétel>] [[PLAIN] | [HEADING<Kkif>]] [NOEJECT] [SUMMARY] [TO PRINT] [TO [FILE]<állomány>]

A listaformátum szerint megjeleníti az aktív adatbázis rekordjait

- ☐ PLAIN — a lapsorszám és a dátum nem íródik ki minden lapra
- ☐ HEADING — a Kkif szövege a lapok tetején megjelenik
- ☐ NOEJECT — nyomtatás előtt nincs lapdobás
- ☐ SUMMARY — a rekordok helyett csak a csoportok részösszegei
- ☐ — és a végösszeg jelenik meg
- ☐ TO PRINT — nyomtatás
- ☐ TO FILE — az állománynév-ben megadott file-ba listáz,
- ☐ — a file kiterjesztése .TXT

RESTORE FROM <állomány> [ADDITIVE]

A SAVE paranccsal lemezre mentett mem.változók betöltése a memóriába. Az ADDITIVE opciót használva nem törlődnek az eddig aktív mem.változók.

Pl.: RESTORE FROM magyar
INDEX ON SYS(15,magyar,nev) TO abc
LIST nev
Ez betölti a magyar abc szerinti rendezéshez szükséges "magyar" nevű mem.változót

RESTORE SCREEN [FROM<mem.vált>]

A SAVE SCREEN-nel legutóbb mentett képernyő visszaállítása. Mem.változóból a FROM -mal tölthetjük vissza a mem.vált-ban megadott képet

Pl.: RESTORE SCREEN FROM kep2

RESUME

A SUSPEND paranccsal megszakított programot tovább futtatja

RETURN [TO MASTER]

Visszatérés az adott programból az azt hívóba, a TO MASTER opció hatására a főprogramba

RUN | !

DOS operációs rendszerbeli parancs indítása FoxBase+-ból

Pl.: ! DIR *.idx

SAVE SCREEN [TO <mem.vált>]

Képernyő tartalmának mentése, ha nem ad mem.változót akkor csak a legutolsó képernyőtartalom állítható vissza

SAVE TO <állomány> [ALL LIKE | EXEPT <glob.mem.vált.>]

Memóriaváltozók tartalmának mentése .MEM kiterjesztésű állományba.

Pl.: SAVE a:ide_ment ALL LIKE m*: az A floppyra az ide_ment.MEM file-ba menti, az összes "m"-mel kezdődő mem.változó tartalmát.

- ☐ ALL LIKE — a globális mem.vált-nak megfelelőket menti
- ☐ ALL EXEPT — a globális mem.vált.-nak nem megfelelőket menti

SCATTER [FIELDS <mezőlista>] TO <tömb>

Az adott rekord mezőinek tartalmát beírja a megadott tömbbe.

Pl.: SCATTER TO ebbe: a már definiált "ebbe" nevű tömbbe beírja az aktuális rekord mezőinek tartalmát, ha még nincs ilyen tömb akkor automatikusan definiálja.

SEEK <kifejezés>

Az adott tulajdonságú rekord megkeresése, rekordmutató rááll az első olyan rekordra, melynek aktuális indexkulcsa megegyezik a kifejezésben megadottal

Pl.: SEEK "2600": megkeresi az első olyan rekordot, melynél az IRSZ mező tartalma 2600-zal egyenlő, ha nem talál ilyent akkor a FOUND() függvény FALSE értéket ad vissza és a rekordmutató az állomány végére áll

SELECT <munkaterület> | <hivatkozási név>

Munkaterület váltása

Pl.: SELECT C vagy SELECT 3: a harmadiknak megnyitott állományt aktivizálja

SET

Környezeti paraméterek menüvezérelt, vagy paranccsal történő beállítása

- ☐ ON engedélyezi az adott tevékenységet
- ☐ OFF tiltja az adott tevékenységet

SET ALTERNATE TO <állománynév>

.TXT kiterjesztésű file-ba is beírja a képernyőre ill. a nyomtatóra kiírtakat

SET ALTERNATE OFF | ON

A SET ALTERNATE-ban megadott állományba írás engedélyezése I tiltása

SET BELL ON | OFF

Hangjelzés a mezők teljes kitöltésekor, vagy hibás beírásnál

SET CARRY ON | OFF

APPEND és INSERT parancsnál az előzőleg kitöltött rekord tartalmának bemásolását engedélyezi

SET CENTURY ON | OFF

Az évszázad kiírását engedélyezi

Pl.:
```
? DATE()
91.07.15
SET CENTURY ON
? DATE()
1991.07.15
```

SET CLEAR ON | OFF

QUIT után az utolsó képernyő törlésének engedélyezése

SET COLOR ON | OFF

Átkapcsolás színes és monokróm monitor között

SET COLOR TO [<normál kép>[,< kiemelt kép>[,<háttér>]]]

Képernyőn lévő színek beállítása, a kód utáni "+" nagyobb fényerőt, a "*" pedig villogtatást eredményez.

Pl.: SET COLOR TO BG/N,GR+/R,N

Színkódok	Betűjelei	Számkódjai
Fekete	N	0
Kék(Blue)	B	1
Zöld(Green)	G	2
Vörös (Red)	R	4
Fehér	W	7
Lila	RB	5
Barna	GR	6
Enciánkék	BG	3
Rejtett	X	-

SET CONFIRM ON | OFF

A mező teljes kitöltése után még egy ENTER leütése is szükséges

SET CONSOLE ON | OFF

Kiírás képernyőn is megjelenik, nyomtatásnál célszerű kikapcsolni

SET DATE <dátumforma>

Dátumforma beállítása

- ☐ AMERICAN hh/nn/éé
- ☐ ANSI éé.hh.nn
- ☐ BRITISH nn/hh/éé
- ☐ ITALIAN nn-hh-éé
- ☐ FRENCH nn/hh/éé
- ☐ GERMAN nn.hh.éé

Pl.: SET DATE AMERICAN

```
? DATE()
15/07/91
SET DATE ANSI
? DATE()
91.07.15
```

SET DEBUG ON | OFF

SET ECHO ON parancs kimenetének nyomtatóra küldése

SET DECIMALS TO [<Nkif.>]

Tizedesjegyek számának beállítása

Pl.:
```
? 11/8
1.37
SET DECIMALS TO 3
? 11/8
1.375
SET DECIMALS TO       ez visszaállítja az eredeti beállítást
```

SET DEFAULT TO [<meghajtó>]

Feltételezett meghajtóegység megváltoztatása, az operációs rendszer aktuális meghajtóját nem változtatja meg.

SET DELETED ON | OFF

Törlésre jelölt rekordok nem "láthatók".

SET DELIMITERS ON | OFF
SET DELIMITERS TO <Kkif.> | DEFAULT

Mezőhatárok jelölésének engedélyezése, ill. a határolójelek beállítása

SET DEVICE TO SCREEN | PRINT

Kiírás helyének beállítása SCREEN = képernyő, PRINT = nyomtató

SET DOHISTORY ON | OFF

Futtatás közbeni utasítások archíválásának engedélyezése

SET ECHO ON | OFF

Végrehajtás alatti utasítások megjelenítése

SET ESCAPE ON | OFF

Az "Esc" gomb programmegszakító hatásának engedélyezése.

SET EXACT ON | OFF

Szövegek teljes egyenlőségének vizsgálata.

SET EXCLUSIVE ON | OFF

Hálózatban az adatállomány kizárólagos használatra megnyithatók.

SET FIELDS ON | OFF
SET FIELDS TO [<mezőlista> | ALL]

A "látható" mezők megadása, ill. ennek engedélyezése

SET FILTER TO [<feltétel>]

Szűrőfeltétel beállítása

Pl.:
```
SET FILTER TO ber<=15000 .AND. ber>=10000
```
A 10.000 és 15.000 Ft közöttiek kerülnek feldolgozásra.

SET FIXED ON | OFF

Beállított számú tizedesjegy engedélyezése.

Pl.:
```
SET DECIMALS TO 3
? 4*2
8
SET FIXED ON
? 4*2
8.000
```

SET FORMAT TO [<állomány>]

Aktív adatbázishoz formátumállomány megnyitása.

SET FUNCTION <Nkif.> TO <Kkif>

Karaktersorozat hozzárendelése adott funkcióbillentyűhöz, az Nkif helyett használhatja a gombra írt jelölést. Az F1 gombhoz csak a HELP hozzárendelés lehet.

Pl.:
```
SET FUNCTION "F8" TO "Ez az F8-as gomb"
```

SET HEADING ON | OFF

Oszlopfejlécek kiírásának engedélyezése pl. LIST -nél.

SET HISTORY ON | OFF
Interaktív üzemmódban kiadott parancsok archiválásának engedélyezése.

SET HISTORY TO <Nkif>
A Nkif-ben megadott számú archivált utasítás jelenik meg a DISPLAY HISTORY parancs hatására.

SET INDEX TO [<állománylista>]
Aktív adatbázishoz indexállományok megnyitása.
Pl.: SET INDEX TO nevek,ber,eletkor

SET INTENSITY ON | OFF
Kiemelt szín engedélyezése adatbeviteli mezőknél.

SET MARGIN TO <Nkif>
Baloldali margó a nyomtatón, alapértelmezés Nkif=0

SET MEMOWIDTH TO <Nkif>
Memomezők szélességének 8 és 256 közötti beállítása, alapérték 50.

SET MENU ON | OFF
Szerkesztőbillentyűk táblázatának megjelenítése.

SET MESSAGE TO [<Kkif.> | <Nkif.>]
Kkif -el megváltoztathatjuk a státuszsor üzenetének szövegét, az Nkif-el a sorát.

SET ODOMETER TO [<Nkif>]
Beállítja, hogy a képernyőfrissítés hány rekordonként történjen, értéke 1 és 32267 között lehet, alapértelmezés 100.

SET ORDER TO [<Nkif.>]
A már megnyitott indexállományok közül kijelöli az aktuálisat, paraméter nélkül az eredeti sorrendben dolgozik.

Pl.:	USE torzs INDEX nev,ber,kodszam	most névsor szerint dolgozik
	SET ORDER TO 3	a kódszám szerint dolgozik
	SET ORDER TO	a fizikai sorrendben dolgozik

SET PATH TO <keresési utak listája>
Azonos a DOS PATH parancsával, állománykeresési utak adhatók meg. A lista elemeit "," vagy ";" választja el.

SET PRINT ON | OFF
Párhuzamos nyomtatás engedélyezése.

SET PRINTER TO \\SPOOLER [\NB] [\F=<n>] [\B=<c>] [\C=<n>] [\P=<n>]
Hálózati nyomtató kiválasztása, a Novell hálózati rendszerben alkalmazható.

☐ NB	(NO BANNER) szeparátor lap elhagyása
☐ F=n	nyomtatvány sorszáma
☐ B=c	Felhasználói név megadása, hossza max. 12 karakter
☐ C=n	nyomtatandó példányszám , értéke 1 és 255 között lehet,
☐	alapértelmezés 1
☐ P=n	Nyomtatót azonosító szám

SET PROCEDURE TO [<procedúraállomány>]
Procedúraállomány megnyitása.

SET RELATION TO [<kulcs> INTO <hiv.név> [ADDITIVE]]
Az aktív adatbázist kapcsolja össze egy másikkal úgy, hogy a rekordmutatók mindkettőben megfelelően mozognak.

```
Pl.: SET RELATION szemszam INTO torzs
     GOTO 8
     ? szemszam,torzs->nev
```

SET SAFETY ON | OFF
Figyelmeztetés egy állomány felülírása előtt.

SET SCOREBOARD ON | OFF
Állapotsor (státuszsor) információinak megjelenítése.

SET STATUS ON | OFF
Állapotsor a képernyő felső részén (OFF), alsó sorokban (ON) jelenik meg.

SET STEP ON | OFF
Lépésenkénti programvégrehajtás engedélyezése.

SET TALK ON | OFF
Parancsvégrehajtás eredményének képernyőre írását engedélyezi.

SET TYPEAHEAD TO <Nkif.>
Billentyűzetbuffer használatát engedélyezi. Nkif értéke 0 és 32000 közötti lehet. FoxBase+ 128 karaktert feltételez.

SET UNIQUE ON | OFF

Indexeléskor az azonos kulcsú rekordok szűrése.

SKIP [<Nkif>]

Rekordmutatót az Nkif-ben megadott értékkel elmozdítja.

Pl.: SKIP 5 öttel növeli az aktuális rekord sorszámát
SKIP -2 kettővel csökkenti az aktuális rekord sorszámát.

SORT TO <új állomány> ON <mező> [/A] [/C] [/D] [FIELDS <mezőlista>] [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>]

Az aktuális állományból az ON <mező> -ben szereplő adatok alapján rendezett új állományt hoz létre, ha a FIELDS paramétert nem használjuk, akkor az új állományban szerepelni fog az aktív adatbázis összes mezőadata.

- /A rendezés növekvő sorrendű
- /D rendezés csökkenő sorrendű
- /C az angol abc kis- és nagybetűi között nem tesz különbséget
- FIELDS csak az itt megadot mezők szerepelnek az új állomány szerkezetében.

Pl.: SORT kisber ON ber /D FOR ber<7000

STORE <kifejezés> TO <mem.vált. lista>

Memóriaváltozónak a <kifejezés>-ben megadott értéket adja.

SUM [<Kkif. lista> TO <mem.vált. lista>] [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>]

Az aktív adatbázis Kkif-ben megadott mezőinek összegzése a mem.vált. listában megadott változókba.

Pl.: SUM ber TO berosszes FOR beosztas="3"

SUSPEND

Programvégrehajtás megszakítása úgy, hogy az még folytatható, a változók tartalma megnézhető. Folytatása a RESUME paranccsal, kilépés a CANCEL-lel.

TEXT

<szöveg határolójelek nélkül beírva>

ENDTEXT

Képernyőn vagy nyomtatón megjelenítendő szöveg egyszerű beírási lehetősége.

TOTAL TO <új állomány> ON <mező> [FIELDS <mezőlista>] [<érv.kör>] [WHILE <feltétel>] [FOR <feltétel>]

Azonos kulcsú numerikus típusú mezők összegével új adatbázist hoz létre

TYPE <állomány> [TO PRINT]

Programlista, vagy más szöveges állomány kiírása képernyőre, a TO PRINT hatására a nyomtatás a megadott perifériára is megtörténik.

Pl.: TYPE foprog.prg TO PRINT

UNLOCK [ALL]

Hálózatban az állomány zárolásának feloldása.

UPDATE ON <kulcs> FROM <hiv.név> [RANDOM] REPLACE <mező> WITH <kif.>

Rendezett adatállomány aktualizálása egy másik már megnyitott adatbázis alkalmazásával.

A hiv.név-vel megadott adatállomány rekordjaihoz a kulcs alapján megkeresi az aktív állományban megfelelőt, ha talált ilyent akkor végrehajtja a REPLACE értékadási parancsot, majd a rekordmutatót eggyel lépteti.

- RANDOM opció lehetővé teszi, hogy rendezetlen állománnyal is helyesen dolgozzon.

USE <állománynév> [INDEX <index állományok >] [ALIAS <hiv.név>] [EXCLUSIVE]]

Adatállomány és a megfelelő indexállományok megnyitása az aktuális munkaterületen. A SET ORDER TO utasításig az INDEX listában első helyen lévő adja a rendezettséget.

- ALIAS hiv.név-vel az állománynevétől független hivatkozási név adható.
- EXCLUSIVE paraméterrel a hálózatban kizárólagos használatra nyitja meg az adatállományt.

WAIT <üzenet> TO <mem.vált.>

Programfutás felfüggesztése egy tetszőleges gomb lenyomásáig, ENTER leütése nem szükséges.

- üzenet szövege kiíródik a képernyő aktuális helyére, enélkül a
- "Press any key to continue . . ." szöveg kerül a képernyőre
- TO opció mem.vált-ba bekerül a lenyomott billentyű kódja.

ZAP

Véglegesen és visszaállíthatatlanul törli az adatállomány összes rekordját, az adatok megsemmisülnek. Hatása azonos a DELETE ALL és a PACK parancsokéval, de attól lényegesen gyorsabb.

Pl.:
```
USE torzs
ZAP
```

3.21.6 FoxBase+ beépített függvényei

ABS(<Nkif>)
Az Nkif abszolútértékével tér vissza.

ALIAS ([<munkaterület>])
Az aktív, illetve a megadott adatbázis alias nevét adja vissza.

ASC(<Kkif>)
A Kkif-ben megadott szöveg első karakterének ASCII kódjával tér vissza.
Pl.: ? ASC("Alfa")
65

AT(<Kkif1>,<Kkif2>)
Megadja, hogy a Kkif1 szöveg a Kkif2-ben hányadik karaktertől található meg, ha nem talál akkor 0-át ad vissza.
Pl.: ? AT("kutya","Amelyik kutya ugat...") "R" 9

BOF(["munkaterület"])
Értéke "TRUE", ha túlléptük az aktív, ill. a megadott adatállomány elejét, különben "FALSE". Akkor használja, ha visszafelé halad az állományban.

CDOW(<Dkif>)
A Dkif-ben megadott dátum napjának angol nevét adja.
Pl.: Ha a kelt-ben az 1991. július 15-i dátum szerepel, akkor
? CDOW(kelt)
Monday

CHR(<Nkif>)
Az Nkif-fel megadott ASCII kódú karakterrel tér vissza.
Pl.: ? CHR(65)
A

CMONTH(<Dkif>)
A Dkif-ben megadott dátum hónapnevét adja vissza.
Pl.: ? CMONTH(kelt) a "kelt" értéke a CDOW -nál megadott
July

COL()
A kurzor aktuális oszloppozícióját adja vissza.

CTOD(<Kkif>)
A Kkif karakteresen megadott dátumot alakítja át Dkif dátumtípusúra.

DATE()
Az operációsrendszer dárumát adja vissza.

DAY(<Dkif>)
A dátum napjának hónapon belüli sorszámát adja vissza.
Pl.: ? DAY(kelt) a "kelt" értéke a CDOW -nál megadott
15

DBF([<munkaterület>])
Megadja az aktuális, ill. a megadott munketerület adatbázisának nevét.
Pl.: ? DBF()
TORZS ha éppen a TORZS nevű állománnyal dolgozunk

DELETED([<munkaterület>])
Logikai "TRUE" értékkel tér vissza, ha az aktuális rekord törlésre jelölt, különben értéke "FALSE".

DISKSPACE()
Az aktuális lemezegységen lévő szabad területet adja meg BYTE-ban.

DOWN(<Dkif>)
A Dkif-ben megadott nap héten belüli sorszámát adja vissza, 1-vasárnap és 7-szombat.
Pl.: ? DOWN(kelt) a "kelt" értéke a CDOW -nál megadott
2
? CDOW(kelt)
Monday

DTOC(<Dkif>[,1])
Dkif dátumból a SET DATE és a SET CENTURY által beállított formájú karakteres kifejezést állít elő, az "1"-es paraméter ééééhhnn karaktersorozatot állít elő, ez a rendezéseknél igen hasznos lehet.
Pl.: SET DATE ANSI

EOF([<munkaterület>])
Értéke "TRUE", ha az állomány végén túljutunk, különben "FALSE".

ERROR()
Az utolsó hiba kódszáma.

EXP(<Nkif>)
Természetes logaritmus alapszámának (e=2.729...) hatványértékével tér vissza.

FCOUNT([<munkaterület>])
Az aktív, ill. a megadott munkaterület adatrekordjainak száma.

FIELD(Nkif[,<munketerület>])
A struktúra Nkif-ben megadott sorszámú mezőjének nevét adja vissza.

FILE(<Kkif>)
Értéke TRUE, ha a Kkif-ben megadott file "létezik", különben FALSE.

FKLABEL(<Nkif>)
Programozható Nkif sorszámú billentyű jelével (F2..F10) tér vissza.
Pl.: ? FKLABEL(1)
F2
? FKLABEL(FKMAX())

FKMAX()
Programozható billentyűk száma, általában F2..F10 között 9 db.

FLOCK()
Adatállomány zárolása, hálózati üzemmódban alkalmazzuk.

FOUND([<Nkif>])
Kereső utasítás sikerességét jelzi, értéke TRUE, ha a talált és FALSE, ha nem talált megfelelő rekordot.
Pl.: USE torzs INDEX ber
keres=3600
SEEK keres
? FOUND()
kiírás .T., ha talált 3600 értékű bért, .F. ha nincs ilyen.

GETENV(<Kkif>)
A Kkif-ben kért DOS környezeti paraméter értékét adja vissza.

IIF(<Lkif>,<kif1>,<kif2>)
Ha az Lkif-ben megadott logikai kifejezés IGAZ, akkor kif1, különben a kif2 értékét adja vissza.
Pl.: kora=IIF(eletkor<20,"fiatal","öreg")
? kora

INKEY(<Nkif>)
Beolvas egy karaktert a klaviatúráról és annak kódját adja meg.

Gomb jele	Váltó nélkül	Váltóval	Gomb jele	Váltó nélkül	Váltóval
Home	1	29	Del	7	-
End	6	23	Backspace	127	127
PgUp	18	31	F1	28	
PgDn	3	30	F2	-1	
Fel	5	-	F3	-2	
Le	24	-	F4	-3	
Balra	19	26	F5	-4	
Jobbra	4	2	F6	-5	
Esc	27	27	F7	-6	
Tab	9	-	F8	-7	
Tab vissza	15	-	F9	-8	
Enter	13	10	F10	-9	

INT(<Nkif>)

Az Nkif egészrészével tér vissza.

```
Pl.:    ? INT(38.42),    INT(-38.42)
        38               -38
```

ISALPHA(<Kkif>)

Ha a Kkif az angol abc betűjével kezdődik, akkor .T. igaz értéket ad vissza, különben .F. hamisat.

ISCOLOR()

Színes kártya és monitor esetén .T. igaz értéket ad, különben .F.

ISLOWER(<Kkif>)

Igaz értéket ad, ha a Kkif az angol abc kisbetűjével kezdődik, különben "hamis"-at .F.-et.

LEFT(<Kkif>,<Nkif>)

Megadja a Kkif szöveg Nkif számú baloldali karakterét.

```
Pl.:    ? LEFT("Balról négyet",4)
        Balr
```

LEN(<Kkif>)

Megadja a Kkif szövegben lévő karakterek számát /szöveghossz/.

```
Pl.:    ? LEN("Szöveghosszt adja meg.")
        21
```

LOCK()

Aktuális rekord zárolása, halózati üzemmódban alkalmazzuk.

LOG(<Nkif>)

Az Nkif-ben megadott szám természetes alapú (e=2.729) logaritmusa.

```
Pl.:    ? LOG(2.729)
        1.00
```

LOWER(<Kkif>)

A Kkif-ben megadott szöveg angol nagybetűit kisbetűre alakítja

```
Pl.:    ? LOWER("Péter")
        péter
```

LTRIM(<Kkif>)

A Kkif szövegének elején lévő szóközöket levágja.

LUPDATE()

Az adatállomány utolsó változtatásának dátumát adja meg.

MAX(<kif1>,<kif2>)

Megadja a két kif. közül a nagyobbat. A kif. dátum, vagy numerikus típusú lehet.

```
Pl:     nagyobb=MAX(348,2765)
        ? nagyobb
        2765
```

MESSAGE([1])

Hibaüzenet szövegét adja vissza, de az [1] megadásával a hibás sort írja ki.

MIN(<kif1>,<kif2>)

A két kifejezés közül a kisebbiket adja vissza. A kifejezések numerikus, vagy dátum típusúak lehetnek.

```
Pl.:    kisebb=MAX(348,2765)
        ? kisebb
        348
```

MOD(<Nkif1>,<Nkif2>)

Az Nkif1-et Nkif2-vel való osztás maradékát adja vissza.

```
Pl.     ? MOD(21,8)
        5
        MONTH(<Dkif>)
```

A Dkif dátumában szereplő hónap sorszámát adja vissza.

```
Pl.:    ? MONTH(kelt) a "kelt" értéke a CDOW -nál megadott
        7
```

NDX(<Nkif>)

Az aktuális indexállomány neve.

OS()

Az operációs rendszer nevét és számát adja vissza, hatása azonos az opeációs rendszer VER paranccsával

PCOL()

A nyomtató pillanatnyi oszloppozíciója.

PROW()

A nyomtató pillanatnyi sorpozíciója.

READKEY()

Annak a billentyűnek a kódszámát adja vissza, amivel kilépett a teljes képernyős szerkesztésből, ha a volt adatmódosítás akkor 0..36 közötti értékkel, különben 256-tal nagyobb értékkel.

Gomb jele	Módosítás nélkül	Módosítással	Gomb jele	Módosítás nélkül	Módosítással
Backspace	0	256	PgUp	6	262
Balra	0	256	PgDn	7	263
Jobbra	1	257	Esc	12	268
Home	2	258	Ctrl+End	14	270
End	3	259	Enter	15	271
Fel	4	260	Ctrl+Home	33	289
Le	5	261	Ctrl+PgUp	34	290
F1	36	292	Ctrl+PgDn	35	291

RECCOUNT([<munkaterület>])

Az aktuális vagy a megadott adatbázis fizikailag meglévő összes rekordjának számát adja meg, a SET FILTER-t nem veszi figyelembe.

RECNO([<munkaterület>])

Az aktuális vagy a megadott munkaterületen az aktuális rekord sorszámát adja vissza.

RECSIZE([<munkaterület>])

Az aktuális ill. a megadott állomány egy rekordjának hosszát BYTE-kban adja meg.

REPLICATE(<Kkif>,<Nkif>)

A Kkif kifejezést Nkif számú ismétlésével új karaktersorozatot állít elő.

Pl.: ? REPLICATE("xy",5)
xyxyxyxyxy

RIGHT(<Kkif>,<Nkif>)

A Kkif-ben megadott szöveg jobboldali Nkif számú karakterét adja vissza.

Pl.: ? RIGHT("Jobb oldalról hármat",3) kiírja a megadott három karaktert
mat

RLOCK()

Az aktuális rekord zárolása, csak hálózati üzemmódban használható.

ROUND(<Nkif1>,<Nkif2>)

Az Nkif1 értékét kerekíti az Nkif2-ben megadott számú tizedesjegyre, 0.5-től felfelé, alatta lefelé kerekít.

Pl.: ? ROUND(248.563,1)
248.600
? ROUND(248.536,-1)
250.000

ROW()

A kurzor képernyőn lévő sorát adja vissza.

RTRIM(<Kkif>)

A Kkif-ben megadott szöveg végéről levágja a szóközöket.

SELECT()

Megadja az aktuális munkaterület sorszámát.

SPACE(<Nkif>)

Nkif számú szóközből álló karaktersorozat előállítása.

SQRT(<Nkif>)

Az Nkif négyzetgyökét adja vissza.

Pl.: ? SQRT(25)
5

STR(<Nkif>[,<hossz>[,<tizedesek száma>]])

Nkif átalakítása "hossz" hosszúságú karaktersorozattá, a hosszba számítsa be a tizedespontot is.

Pl.: ? STR(9876.543,7,2)
9876.54

STUFF(<Kkif1>,<kezdő pozíció>,<hossz>,<Kkif2>)

A Kkif1 szöveg "kezdőpozíció" karakterétől, "hossz" hosszúságú részét cseréli a Kkif2 szövegére

Pl: eredeti="Buksi kutya."
ujszoveg=STUF(eredeti,7,5,"okos.")
? ujszoveg
Buksi okos.

SUBSTR(<Kkif1>,<kezdőpozíció>,<hossz>)

A Kkif1 szövegből a kezdőpozíciótól, adott hosszúságú rész kiemelése.

Pl.: m1 = "Buksi okos kutya."
kivag = SUBSTR(m1,7,5)
? kivag
okos

SYS() függvények a rendszerkörnyezet leírását adják, eredményük mindig karakteres típusú.

SYS(0)

Hálózati munkaállomás nevét adja vissza.

SYS(1)

Az aktuális dátum Juliánusz-napokban számított értékét adja vissza.

SYS(2)

0 órától eltelt másodpercek száma.

SYS(3)

A rendszerdátumból és az időből új állománynevet generál, melyet átmeneti állományok neveiként használhatunk.

SYS(5)

Megadja az aktuális meghajtóegység nevét.

Pl.: ? SYS(5)
C:

SYS(6)

Aktuális nyomtatóegység neve, a SET PRINTER TO-val megadott meghajtót adja meg.

SYS(7[,<munkaterület>])

Az aktuális, ill. a megadott munkaterületen megnyitott formátumállomány nevét adja meg.

SYS(9)

FoxBase installálási szériaszáma.

SYS(10,<Nkif>)

Az Nkif-nek, mint Juliánus-napnak a dátumát adja vissza.

SYS(11,<Kkif>)

Karaktereként megadot dátum értékét Juliánus-napokban adja meg.

SYS(12)

Az operatív tár még szabad byte-jainak számát adja meg.

SYS(13)

Nyomtató periféria READY vagy OFFLINE állapotát adja meg.

SYS(14,<Nkif1>[,<Nkif2>])

Az Nkif1-el megadott indexállomány kulcskifejezését adja meg, az Nkif2-vel munkaterületet adhatunk meg.

SYS(15,<táblázat>,<Kkif>)

A Kkif szöveg karaktereit a "táblázat"-ban megadottakra cseréli. A táblázat 254 byte hosszú mem.vált, melynek adott pozícióján lévőre cseréli ki a Kkif adott ASCII kódú jeleit. Jól használható a magyar abc szerinti indexállományok létrehozásánál.

Pl.: RESTORE FROM magyar — a "táblázat"-ot tartalmazó mem.változó betöltése.

USE torzs
INDEX ON SYS(15,magyar,nev) TO nev_abc

SYS(16[,<Nkif>])

A végrehajtás alatti program neve, Nkif-fel hívási szintet adhatunk meg.

SYS(17)
A CPU azonosítóját adja meg.

SYS(18)
A READ műveletnél éppen kitöltött cella nevét adja meg.

SYS(100)
A SET CONSOLE-lal beállított állapotot adja meg (ON vagy OFF).

SYS(101)
A SET DEVICE-szal beállított állapotot adja meg (SCREEN vagy PRINT).

SYS(102)
A SET PRINT-tel beállított állapotot adja meg (ON vagy OFF), hibakezelési eljárásoknál jól használható.

SYS(103)
A SET TALK-kal beállított állapotot adja meg (ON vagy OFF).

TIME([1])
A DOS rendszeridejét adja meg.

TRANSFORM(<kif>,<Kkif>)
Kkif-ben magadott formátumra adatátalakítás. A Kkif-re érvényes kódokat a PICTURE-nél tárgyaltuk.

Pl.:
```
szam = 54376523.98756
? TRANSFORM(szam,"@R 999 999 999.99 Ft")
54 376 523.98 Ft
```

TRIM(<Kkif>)
Hatása azonos az RTRIM függvényével, a Kkif végéről levágja a szóközöket.

TYPE("<kifejezés>")
A megadott kijezés típusát adja meg.

☐	C	karakteres
☐	L	logikai
☐	M	memo
☐	N	numerikus
☐	S	képernyő
☐	U	definiálatlan típusú ismeretlen változó

UPDATED()
Jelzi, hogy a legutóbbi READ utasításkor történt-e változtatás. Értéke .T. ha történt változás, különben .F.

UPPER(<Kkif>)
A Kkif angol betűit nagybetűkre alakítja.

VAL(<Kkif>)
A Kkif-ben szövegként megadott számokat numerikusra alakítja. A karaktersorozatot addig alakítja át, amíg az számként értelmezhető.

VERSION()
A FoxBase verzióját adja meg.

YEAR(<Dkif>)
A Dkif-ben megadott dátum évszáma az évszázaddal együtt.

Pl.:
```
? datum
91.07.15
? YEAR(datum)
1991
```

CONFIG.FX állományba írható, a SET paranccsal nem módosítható konfigurációs paraméterek

BUCKET
Meghatározza a @... GET ... utasítások számára lefoglalandó memória méretét. Értéke minimálisan 1, max. 32, alapértelmezés 4.

BUFFERS
Az I/O bufferek száma. Egy-egy buffer 2 Kbyte méretű. Értéke 4..31 közötti lehet. Alapértelmezés 31.

COMMAND
Indításkor automatikusan végrehajtandó FoxBase parancs, vagy egy elindítani kívánt program neve.

FILES
Párhuzamosan megnyitott állományok számát adja meg. Alapértelmezés 16. Maximálisan megadható 48. Minden újabb állomány 130 byte helyet foglal el.

HMEMORY

Parancsarchíválásokhoz lefoglalt memória mérete Kbyte-os egységekben. Alapértelmezése 5 Kbyte.

INDEX

Indexállományok kiterjesztése állítható be. Alapértelmezése .IDX

MAXMEM

A MODIFY COMMAND paranccsal módosítható program mérete állítható be. Értéke 8..64 közötti lehet, alapértelmezése 64.

MVARSIZ

1 kbyte-os memóriaterületeket foglal a karakterestípusú adatok részére. Értéke 1..64 közötti lehet, alapértelmezése 64.

MVCOUNT

Egyidejűleg használható memóriaváltozók számát adja meg. Értéke 128..3600 között lehet, alapértelmezése 256.

PROMPT

FoxBase várakozási jelének beállítása tetszőleges max. 20 karakter hosszúságú jelsorozatra. Alapértelmezése egy pont és egy szóköz.

TEDIT

Külső szövegszerkesztő használatára ad lehetőséget.

TIME

Megadja, hogy hányszor próbálkozzon újra egy-egy karakter nyomtatásánál. Értéke 1..1 000 000 közötti lehet. Alapértelmezés 6 000.

WP

Külső szövegszerkesztő neve adható meg.

Egy példa a CONFIG.FX file beállítására:

```
COLOR = BG/N,GR+/R,N
TEDIT = NE
COMMAND = SET DATE ANSI
```


SYMPHONY
rendszerprogram

4. SYMPHONY PROGRAMRENDSZER

4.1 A SYMPHONY rendszer interaktív kezelése

4.1.1 A rendszer bemutatása

A rendszer központja egy nagyméretű .EXE program, amely az adatkezelést végzi (**SYMPHONY.EXE**), ehhez járulnak különböző kiegészítő programok, amelyek speciális részfeladatok elvégzésére szolgálnak.

Kiegészítő programok a **PGRAPH.EXE**, amelynek segítségével kinyomtatható a központi programban elkészített és tárolt grafika, az **INSTALL.EXE**, amelynek segítségével igazíthatjuk a rendszert az adott géphez, harmadik kiegészítő program a **TRANS.COM**, ezzel a programmal lehet különböző népszerű adatbázis-kezelő programok által előállított állományokat a rendszer által ismert formátumúra hozni. Végül két kiegészítő programról kell szólni, amelyek megléte nem feltétlenül szükséges, a rendszer nélkülük is képes működni. Az első egy parancsállomány, a **TUTORIAL.BAT** nevet viseli. Ez a parancs-állomány semmi mást nem tesz, mint elindítja a központi programot oly módon, hogy felszólítja a kezelést tanító rutinok automatikus betöltésére és azonnali futtatására. (Aki tud angolul, installálás után indítsa el a TUTORIAL-t, a funkciók bemutatása és leírása után, gyakorlás közben szerezhet tapasztalatokat az adatkezelő használatáról.) A második egy keretprogram, mellyel egy menüből indíthatjuk a fent nevezett programokat. Ez az **ACCESS.COM** nevet viseli. A SYMPHONY gyártója a Lotus Development Corporation. Neve elárulja, hogy azonos a közismert LOTUS1-2-3 elnevezésű integrált programrendszer készítőjével. Azt is mondhatjuk, hogy a SYMPHONY a LOTUS1-2-3 továbbfejlesztett változata. Ennek megfelelően kezelése bizonyos mértékig megegyezik a LOTUS-éval, szolgáltatásaiban viszont felülmúlja azt.

4.1.2 A SYMPHONY program (SYMPHONY.EXE)

Üzemmódok

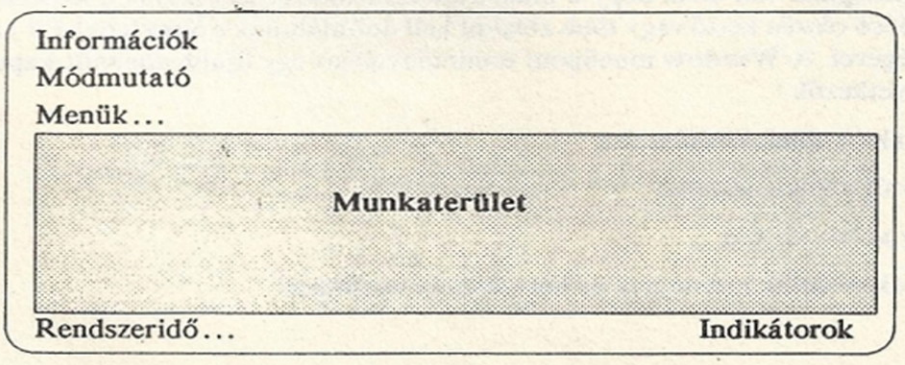
A program alapelve a többi integrált adatkezelővel szemben (FRAMEWORK, MS-WORKS...) egy nagy központi munkatáblázat (WORK SHEET) kezelése különböző módon. Ezek az üzemmódok a következők :

- Táblázatkezelő (SHEET)
- Szövegszerkesztő (DOC)
- Üzleti grafika (GRAPH)
- Adatbázis-kezelő (FORM)
- Kommunikáció (COMM)

Amikor adatainkat kezeljük, a különböző üzemmódokat figyelemmel kísérhetjük az ún. **MÓDMUTATÓ** segítségével, ami a képernyő jobb felső sarkában van.

Képernyő

A képernyő első sorát általában információk megjelenítésére, a második sort a menük megjelenítésére használja a rendszer. A legalsó sorban helyezkedik el a rendszeridő és a dátum, vagy az esetleges hibaüzenetek, valamint néhány indikátor, amelyek a billentyűzet állapotáról (például a felülírásos üzemmódról tájékoztatnak). A képernyő középső részén találhatók adataink. Ezt a részt munkaterületnek nevezzük.



Segítség a képernyőn

A programok működése során folyamatosan kérhetünk segítséget a rendszertől, az **F1** gomb leütésével. Ilyenkor a rendszer megkeresi a segítség azon részét, amelyik szerinte a leghatásosabb az adott munkafázisban, ezt kiírja a képernyőre, majd egyéb információk címszavait felkínálja egy menüben. Ebben a menüben szerepel a "Help Index" kulcs, amely lehetőséget ad arra, hogy az egész segítő alrendszerben mint egy menüben közlekedve kikeressük a szükséges információkat. A segítségből való visszatéréshez használjuk az **Esc** gombot.

Menükezelés

A SYMPHONY program három különböző menüt használ.

- Első menü a szolgáltatások menüje (**SERVICES**), amelyik az **F9** billentyűvel érhető el. Ebben a menüben találhatók a különböző file műveletek, ablak- és nyomtatókezelő parancsok, alapértelmezett beállítások stb.
- Másik menü az üzemmódok közötti váltást teszi lehetővé. Ezt a menüt az **ALT+F10** billentyűk együttes leütésével hívhatjuk elő.
- A harmadik menü nem is egy, hanem minden üzemmódban más és más. Ezek a menük tartalmazzák az adott üzemmódra vonatkozó adatkezelési lehetőségeket (pl.: szövegszerkesztő módban a szöveg rendezése, mozgatása, másolása stb...).

Az üzemmódok menüit mindig az adott üzemmódba váltva (**ALT+F10**), az **F10** gombbal vehetjük igénybe. A Menük a képernyő második sorában helyezkednek el. A menük fölötti első sorban mindig rövid tájékoztatót kapunk az éppen aktív menüpontról.

A menük kezelésére kétféle lehetőségünk van :

1. A jobbra vagy balra nyilakkal a kívánt menüpontra állunk, majd lenyomjuk az ENTER billentyűt. Ilyenkor figyelemmel kísérhető az első sorba írt segítség.
2. Ha már kicsit gyakorlottabbak vagyunk a program kezelésében és nincs szükségünk az első sorban megjelenő rövid segítségre, lehetőségünk van a kívánt menüpont közvetlen kiválasztására és elindítására, a menüpont első betűjének leütésével.

A SYMPHONY menüi hierarchikus felépítésűek. Ez azt jelenti, hogy egyes menüpontok kiválasztása egy újabb almenüt eredményez, amiből választva esetleg újabb almenüt kapunk. (Hasonlít az IBM-PC könyvtárstuktúrájához.) A menüstruktúrában egy szinttel feljebb jutni mindig az **Esc** gombbal lehet.

Hibakezelés

A képernyő legalsó sorában általában a rendszeridőt látjuk az általunk beállított dátum- és időformátumban, mindaddig míg valami hibát nem követünk el a programok kezelése során. Ilyenkor az alsó sorban jelenik meg a hibaüzenet, a jobb-felső sarokban pedig az "**ERROR**" felirat látható. A hibaüzenetek nyugtázására az **Esc** vagy az **ENTER** gombot használhatjuk. A hibaüzenetnél részletesebb leírást kapunk, ha előbb az **F1** gombot nyomjuk le. (Az Esc gombot használjuk akkor is, ha valamit érvényteleníteni akarunk.)

Ablakok kezelése

A programban lehetőségünk van arra, hogy a munkatáblázat több részét lássuk a képernyőn egyidőben. Ehhez a képernyő középső részén kettő vagy több ablakot kell definiálnunk a **Services (F9)** menü **Window** menüpontjának segítségével. A **Window** menüpont eredményeként egy újabb almenüt kapunk. Ennek legfontosabb elemei a következők :

Use	Az aktív ablak kiválasztása
Create	Egy új ablak készítése
Delete	Egy ablak törlése
Layout	Az aktív ablak méretének és helyzetének beállítása

Hide Egy ablak kirajzolásának letiltása

Isolate Az aktív ablak kivételével az összes ablak megjelenítésének letiltása

Expose Az összes ablak kirajzolása

Pane Az aktív ablak kettő ill. négy részre osztása

Ezen kívül lehetőségünk van az aktív ablak különböző paramétereinek beállítására. A paramétereket egy beállítótáblában találjuk meg, amelyet a **Settings** menüponttal "teríthetünk" a képernyőre. Az aktív ablakhoz rendelhetünk egy nevet (**Name**), megadhatjuk az ablak üzemmódját (**Type**), valamint mozgási korlátokat (**Restrict**) adhatunk az ablaknak.

A fentiek értelmében tehát láthatjuk a munkatáblázatban elhelyezett adatok egy részét mint táblázatot, más részét mint szöveget. A mozgási korlátok behatárolják azt a területet, amelyen a kurzormozgató billentyűk segítségével közlekedhetünk. (Az aktív ablak mozgási korlátai átmenetileg megszünnek egy-egy másolási (**Copy**), mozgatási (**Move**) parancs esetén. Így mozgathatunk, vagy másolhatunk adatokat egyik ablakból a másikba).

Ha egy munkában egy táblázatot kívánunk kezelni, de leírást is szeretnénk hozzá készíteni, akkor a következő lépéseket kell végrahajtanunk :

1. Nyomjuk le az **F9** gombot, majd válasszuk a **Window** menüpontot.
2. Az épp aktuális (egyetlen) ablakunk méretét állítsuk be a **Layout** opció és a kurzormozgató billentyűk segítségével, a munkaterület felső felére.
3. Ha a kezelni kívánt táblázatunk például 20 sorból és 10 oszlopból fog állni, korlátozzuk az ablakot (**Window-Settings-Restrict**) az A1..J20 tartományra. (Tartományok megadására lásd a "TARTOMÁNYOK" fejezetet.)
4. Állítsuk be az ablak típusát "táblázat"-ra (**SHEET**). Ezt a **Window-Settings-Type** menüpont kiválasztásával tehetjük meg. (Használható az **Alt+F10**-re megjelenő üzemmód menü is.)
5. Válasszuk a **Window** menüben a **Create** menüpontot.
6. A program által feltett kérdésre adjuk meg az elkészítendő ablak nevét ! (Csak még nem létező név lehet !)
7. A megjelenő üzemmódmenüben (azonos az **Alt+F10** leütése esetén megjelenő menüvel) válasszuk ki az ablak típusát. Esetünkben **DOC**.
8. Ezután meg kell adnunk az ablak méretét és elhelyezkedését. (**Window-Layout**) Most legyen a képernyő alsó fele, hogy egyszerre láthassuk a két ablakot !
9. A program automatikusan felkínálja a **Window-Settings** menüt. Lehetőségünk van megváltoztatni az épp létrehozott ablak adatait. Itt állítsuk be az új ablak mozgási korlátait a táblázat alatti tartományra (A21..IV8192).
10. A **Quit** menüponttal kiléphetünk a szolgáltatások menüjéből.

Miután így előállítottunk két ablakot, a szolgáltatások menüjében a **Window-Use** opció segítségével közlekedhetünk a két ablak között, egyszer a táblázatunkat kezelve, egyszer pedig a hozzáfűzendő szöveget írva.

4.1.3 File-műveletek

A **Services** (F9) menü másik igen fontos almenüje a **File** almenü. Ebben a menüben találhatók a különböző állományműveletek (betöltés, mentés, részek kezelése, stb...).

Save A munka elmentése. A program alapértelmezésben munkánknak a **.WR1** kiterjesztést adja, de lehetőségünk van ennek felülbírálására.

Retrieve Az elmentett munkák egyikének betöltése a memóriába. A program felkínálja a menük sorában, az aktuális alkönyvtár összes **.WR1** kiterjesztésű állományát, valamint az alkönyvtárakat. Tehát egy menüből választhatjuk ki a kívánt munkát. Az **F10** bil-

lentyű lenyomására az állományokat táblázatszerűen megjeleníti a munkaterületen és a második sorban az aktuálisnak jelölt állományról ad bővebb információkat.

Combine Az épp memóriában lévő munkánk egy általunk megjelölt tartományába olvashatjuk be lemezről egy másik munka bizonyos részét, vagy az egészet.

Xtract Az aktív munka egy kijelölt tartományát menthetjük el háttértárolóra. Ezt azután a **Combine** menüponttal beolvashatjuk egy másik munkába.

Erase Egy kijelölt állomány törlése a lemezről.

Worksheet	Munkák
Print	Nyomtatható állományok
Graph	Grafika állományok
All	Minden állomány

Bytes Az aktuális lemezen lévő üres terület hossza byte-okban.

List A DOS dir parancsához hasonlóan állománylistát ad a munkaterületen. Az alábbi lehetőségek közül választhatunk:

Table Az előző négy lehetőség valamelyikét kiválasztva állománylistát helyezhetünk el a munkaterületen. Pl.:

ADO90.WR1 91-01-30 5999

LEIRAS.PRN 91-02-04 15184

LEIRAS.WR1 91-02-07 27726

Amikor ezt a funkciót használjuk, a program ideiglenesen táblázat üzemmódra vált és az elhelyezendő lista bal felső sarkát kell kijelölni. A lista három oszlopból áll. Első oszlopa az állomány neve, a második oszlop egy numerikus érték, amely az állomány utolsó módosításának dátumát és időpontját tartalmazza, a harmadik oszlopban az állomány hossza lesz byte-okban. Ahhoz hogy a dátumot megfelelő módon jelenítse meg a képernyőn, a lista második oszlopát dátum, vagy időpont formátumúra kell hoznunk (lásd a TÁBLÁZATKEZELŐ fejezetben).

Import Más programokkal létrehozott állományok beolvasása. Lehetőségünk van például egy szövegszerkesztőben elkészített állomány beolvasására, vagy egy struktúrált adatállomány beolvasására. Struktúrált adatállomány beolvasása annyit tesz, hogy csak a számokat és az idézőjelek közötti szövegeket olvassa be a munkatáblázat egy kijelölt részére.

Directory Az aktuális könyvtár megváltoztatása.

4.1.4 Beállítótáblák

A SYMPHONY rendszer szinte mindenre kiterjedő lehetőséget biztosít a különböző beállítótáblák (**SETTINGSHEET**) alkalmazásával. Minden menüben, az üzemmódok között váltó menüt (**Alt+F10**) kivéve megtalálhatjuk a **Settings** menüpontot. Ezek a menüpontok egy-egy beállítótáblát és egy almenüt takarnak. Amikor egy táblát kezelünk, a képernyőről ideiglenesen törlődnek az adatok, és a tábla jelenik meg meg az aktuális értékekkel kitöltve. Ha még egyszer sem állítottunk be egy táblát, a paraméterek akkor is ki vannak töltve valamilyen adattal. Ezek az adatok az ún. alapértelmezett értékek (**Default**), amelyeket többnyire a rendszer telepítése közben adtunk meg. Külön beállító táblája van tehát a különböző üzemmódoknak, és a Services menü almenüinek is.

Egy beállítótábla különleges szerepéről szólni kell, ez a **Services** menü **Configuraton** almenüjének táblája. Itt is megadhatók a SYMPHONY kölönböző beállítótábláinak alapértelmezett adatai.

Nyomtatás

A nyomtatást és az adott munkához rendelt nyomtatóbeállításokat kezelhetjük a **Services** menü **Print** almenüjében. Amikor ezt a menüt aktivizáljuk, a képernyő középső részén eltűnik a munkaterület, helyette a nyomtatási paraméterek beállítótáblája jelenik meg, az aktuális értékekkel kitöltve, mintegy figyelmeztetésül, hogy nézzük át a beállításokat nyomtatás előtt. A menüpontok a következők:

Go A nyomtatás elindítása a beállításoknak megfelelően.

Line-Advance Egy soremelés (**LF**) karaktert küld a beállításnak megfelelő cél eszközre (**Destination**).

Page-Advance Egy lapdobás (**FF**) karaktert küld a céleszközre.

A beállítható paraméterek :

Page-Length	A lap hossza sorokban
Spacing	A soremelés mértéke
Print-Number	Kezdő lapszám
Start-Page	Első nyomtatandó lap
End-Page	Utolsó nyomtatandó lap
Breaks	Automatikus lap tördelés
Wait	Megáll a lap végén és vár egy gomb lenyomására
Header	Fejléc definiálása
Footer	Lábléc definiálása
Source	Nyomtatandó tartomány, vagy adatbázis megadása
Destination	Céleszköz megadása. Ez lehet a nyomtató, vagy egy állomány, amelyet a későbbiekben kinyomtathatunk mint bármely szöveges állományt, de lehet a munkatáblázat egy tartománya is.
Init-String	A nyomtatás kezdetén kinyomtatandó karaktersorozat, amivel például a nyomtatót hozhatjuk alapállapotba
Space-Compression	A szóköz karakterek átalakítása tabulátorrá
Attributes	A nyomtatási attributumok figyelembevétele
Format	A képernyőn látható, vagy táblázatos formátumú nyomtatás
Top-Labels	Minden lap tetején kiírandó címkék
Left-Labels	Minden lap baloldali oszlopába nyomtatandó címkék

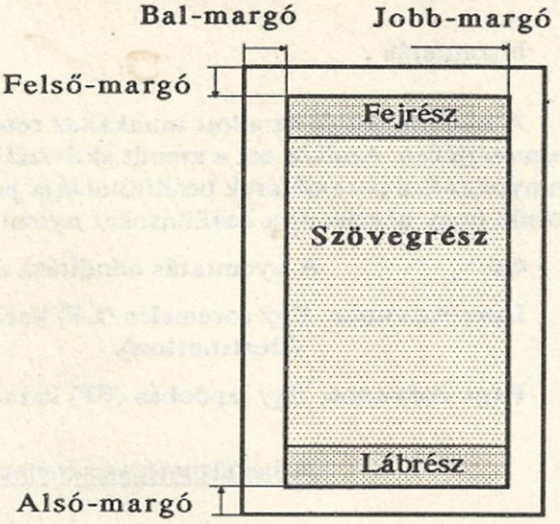
Align A lapszámozás kezdetét 1-re állítja (**Print-Number**), a sorok számolását újrakezdi. Erre a menüpontra akkor lehet szükség, ha a nyomtatás végén kézi vezérléssel állunk a lap elejére.

Settings A képernyőn látható beállítótábla adatait kezelhetjük ezzel a funkcióval. Ez a beállítás felülbírálja a szolgáltatások menüjének **Configuration** pontjában beállított értékeket. A beállítások a munka mentésekor automatikusan mentődnek, majd a munka visszatöltésekor automatikusan töltődnek.

Quit Visszatérés a munkaterületre.

A nyomtatás használatához néhány jó tanács. Mivel a program a nyomtatások végén nem dob lapot, ezért ha a céleszköz a nyomtató, minden nyomtatás után használjuk a print menü **Page-Advance** funkcióját, vagy ha kézi vezérléssel lapot dobunk a nyomtatón, használjuk az **Align** funkciót, ezzel kezdőértékre állítjuk a program belső sorszámlálóját. Minden nyomtatás előtt ellenőrizzük a kezdő lapszámot (**Print-Number**) és ha szükséges, állítsuk be. Ha a céleszköz egy állomány, vegyük figyelembe, hogy ha az állomány

létezik, akkor a karaktereket az állomány végéhez írja, nem írja felül az eddigi tartalmát. Ezért előbb használjuk a **Settings-Destination-Erase** funkciót, amivel törölhetjük a lemezről a célállományt. A nyomtatás során értelmezendő margókat az ábra szemlélteti.



A fejlécet és a láblécet a következő módon definiálhatjuk.

Írhatunk tetszőleges szöveget, vagy a @ jel segítségével az aktuális dátumot, a # jel segítségével a lapszámot tehetjük a fej- és/vagy láblécbe. Ezeket a sorokat rendeztethetjük is a | karakterrel. Az első | jel előtti részt balra, az első és a második | jel közöttit középre, a harmadik utáni részt jobbra rendezi nyomtatáskor. Például egy lábléc, amelyben a lapszámot jobbra igazítva, a dátumot balra igazítva szeretnénk látni, a **@||#.oldal** definícióval állítható elő. (A fenti példából látható, hogy nem kötelező a fej- és láblécek mindhárom részébe írni)

4.2 A táblázatkezelő

4.2.1 Cellák, cellacímek, aktuális cella

A táblázatkezelő üzemmódban a munkaterületet, mint egy négyzethálós papírt kell elképzelnünk. A munkaterület tehát sorokból és oszlopokból áll. A sorokat 1-től 8192-ig terjedő számokkal, az oszlopokat (256 db.) az angol ABC betűivel jelölték. Természetesen az angol ABC nem 256 betűből áll, ezért a Z betű után betűpárokat alkalmaztak (AA, AB, AC,... IV). Egy sor és egy oszlop kereszteződése egy kis "négyzet", amit **cellá**-nak nevezünk.

A programban egy cella valójában nem négyzet alakú, hanem egy karakter magas, alapértelmezésben kilenc karakter széles téglalap. Adatainkat táblázat üzemmódban ezekbe a cellákba vihetjük. A program betöltésekor a cellák üresek, amit a SYMPHONY egy NA-val jelöl. Ez az NA nem jelenik meg a képernyőn, de a beépített függvények segítségével bevihető ill. lekérdezhető. (Lásd a FÜGGVÉNYEK fejezetet).

Munkánk során hivatkozhatunk egy-egy cella tartalmára úgy, hogy használjuk a **CELLACÍM**-eket. Egy cellacím egy, vagy két betűből és egy számból áll. Ezt a program mint koordinátát értelmezi. Például az AB1065 cellacímként az AB oszlop és az 1065. sor kereszteződésében lévő cellára mutat.

Amikor egy tartományt másolunk, vagy mozgatunk a munkatáblázat egy másik pontjára, a SYMPHONY automatikusan korrigálja a tartományban előforduló cellacímeket.

Például, ha egy tartományban az A1 cellacím előfordul, és a tartományt egy sorral lejjebb mozgatjuk, a cellacím B1-re változik.

Lehetőségünk van az ilyen korrekció kiküszöbölésére az **abszolút cellahivatkozás** használatával. Az abszolút hivatkozást az oszlop betűjele, vagy a sor száma (esetleg mindkettő) elé tett $ karakter jelzi. A program azt a koordinátát (sorszámot, vagy oszlop betűjelet) nem változtatja, amelyik előtt $ jel van.

Az előző példánál maradva, ha azt akarjuk, hogy az A oszlophivatkozás ne változzon, akkor $A1-et kell írnunk. Ekkor, ha mozdítjuk a tartományt, csak a sorszám változik. Ezt is letilthatjuk, a A1 abszolút hivatkozással. Amikor adatot viszünk be a táblázat egy cellájába, mindig az **AKTUÁLIS CELLÁ**-val dolgozunk. Ez az aktuális cella a többitől eltérő színnel van jelölve, tartalmát a képernyő legfelső sorában láthatjuk. Ez igen hasznos abban az esetben, ha a munkaterületen valami oknál fogva nem látjuk a cella teljes tartalmát (Lásd még a **BEVIHETŐ ADATOK** fejezetet). Az aktuális cella kijelölését a kurzormozgató billentyűkkel tudjuk megválasztani (Lásd a **MOZGÁS A TÁBLÁZATBAN** fejezetet).

4.2.2 Tartományok

A program futása során bizonyos parancsok operandusaiként tartományokat (**Range**) kell kijelölnünk, például adatmásolás, mozgatás, törlés... stb. esetén.

Ezeket a tartományokat többféleképpen is megadhatjuk.

Egyik mód a rámutatás. Ilyenkor mielőtt a kívánt funkciót kiválasztanánk, az aktív mezőt a kijelölendő tartomány egyik sarkára állítjuk, ezután elindítjuk a szükséges funkciót, majd a tartomány átlós irányú másik sarkára állunk és lenyomjuk az **ENTER** billentyűt.

Másik lehetőségünk a tartomány megadására, hogy közvetlenül begépeljük a tartomány két átlós mezőjének koordinátáit. Begépeléskor mindegy, hogy kis vagy nagybetűket használunk és hogy milyen sorrendben adjuk meg az átló két végét. (A koordinátákat oszlop-sor sorrendben kell megadni) Egy szabályos tartomány megadás pl.: B12..AF450. Ez a tartomány a B oszlop 12. mezője és az AF oszlop 450. mezője által, mint átló által leírt terület. A tartomány két azonosító mezőjének koordinátáját két ponttal kell elválasztani. Amennyiben csak egyetlen mezőt kívánunk tartományként kijelölni, ezt minden további nélkül megtehetjük (A1..A1). Legtöbb esetben amikor tartományt kell megadnunk, a program automatikusan a tartomány egyik sarkának veszi az épp aktív mező koordinátáit, és ehhez képest mozoghatunk a kurzor-mozgató billentyűkkel. Ha az aktív mező nem a kívánt tartomány egyik sarokpontja volt, akkor lehetőség van a tartomány-kijelölést elölről kezdeni. Ilyen esetekben az **Esc** vagy a szürke balra mutató nyíl (**BackSpace**) lenyomásával megszüntethetjük az aktív mező sarokpontként való értelmezését. Ezután mikor a megfelelő sarokpontra álltunk, a **TAB** billentyű leütésével térhetünk vissza a tartomány kijelölésre. Amikor tartományokat jelölünk ki, az aktív ablak mozgási korlátai (lásd az Ablakok kezelése fejezetet) megszünnek.

4.2.3 Mozgás a táblázatban

A kurzormozgató billentyűk általában a kurzor (táblázatkezelő üzemmódban az aktív cella kijelölésének) mozgatására szolgálnak, amikor a Scroll Lock gomb kikapcsolt állapotban van. Ha a Scroll Lock be van kapcsolva, akkor a fent nevezett billentyűk a munkaterületen látható adatok mozgatására szolgálnak. A programban különleges szerepet kapott az END billentyű. Lenyomására csak egy kapcsoló (indikátor) áll be, amit a képernyő alsó sorában az "End" felirat jelez. A következő leütött kurzormozgató billentyűtől függ az aktív cella kijelölése, vagy a kurzor elmozdulása.

END+jobbra Az aktuális sor utolsó cellája lesz az aktív cella

END+balra Az aktuális sor első cellája lesz az aktív cella

END+föl, vagy END+PgUp Az aktuális oszlop legfelső cellája lesz az aktív cella

END+le vagy END+PgDn Az aktuális oszlop legalsó cellája lesz az aktív cella

END+Home A munka során használt cellák közül a legalsó sor legutolsó cellája lesz az aktív cella

(Az End billentyű használata során érvényben vannak az aktív ablak mozgási korlátai)

4.2.4 Bevihető adatok

Egy-egy cellában maximum 240 karakterből álló kifejezést, vagy fix értéket helyezhetünk el. Mivel a táblázat 256 oszlopból és 8192 sorból áll, természetesen minden cellát nem tudunk maximálisan kihasználni, mert gépünk memóriája korlátozott. (A SYMPHONY program képes a Lotus Intel Microsoft Extended Memory szabványnak megfelelő memóriabővítést 4 Mbyte-ig kezelni.)

A cellákba adatot bevinni úgy lehet, hogy a kívánt cellát jelöljük aktív cellának, és egyszerűen el kezdjük gépelni az adatot. Ilyenkor a képernyő második sorában végezzük a bevitelt, majd az **ENTER** vagy a kurzormozgató billentyűk valamelyikének leütésével visszük az adatot az aktív cellába. Ha a kurzormozgató billentyűk valamelyikének leütésével fejeztük be a bevitelt, akkor az aktív cella kijelölés is a megfelelő irányba mozdul a bevitel után. Amennyiben a bevitelt nem kívánjuk érvényesíteni, vagyis a begépelt adatot nem kívánjuk bevinni az aktív cellába, az Esc billentyűvel léphetünk ki a bevitelből. A cellákban elhelyezhetünk **numerikus értékeket** vagy **karakterláncokat**, számítási értékeket (**formulák**) fix operandusokkal, vagy cellaértékekre vonatkozó cellahivatkozásokkal, valamint a SYMPHONY által ismert előre elkészített **függvények** valamelyikét (Minden típusra példákat láthatunk a megfelelő alfejezetekben).

Azt, hogy az adott cellában lévő adat milyen típusú, az első karakter alapján dönti el a program. A 0 1 2 3 4 5 6 7 8 9 0 (+ - . karakterek valamelyikével kezdődő adat numerikus érték, vagy formula. A @ karakter a függvényt, a ' " ^ karakterek a szöveges értéket jelölik. Előfordulhat, hogy a bevitt adatok a táblázatban nem olyan formátumban jelennek meg, mint ahogy bevittük őket. Ennek oka, hogy a táblázatban egy-egy tartománynak, vagy akár minden cellának, különböző formátumokat írunk elő. Bevitelkor a SYMPHONY automatikus forma konverziót végez, és az általunk előírt formában teszi adatunkat a cellába.

Külön említést érdemel a dátum és az időpont típusú adat. A programban a dátumok és az időpontok csak egy formátumot jelentenek, valójában numerikus értékek. Ezek a numerikus értékek valós számok, amelyeknek az egész része az adott időpont dátuma (Julianus szám 1990. január 1-től), a törtrész az időpont (óra-perc-másodperc) sorszámmá alakítva.

A logikai adatokat is numerikusként kezeli a SYMPHONY. Az igaz értéknek megfelel az 1-es, a hamisnak a 0. A logikai érték jelölésére a # jelet használjuk első karakterként.

Numerikus értékek

Egy-egy cellába közvetlen módon vihetünk be numerikus értékeket. A bevezetőben olvashattunk a formátumokról. A SYMPHONY a numerikus értékek kiírásához a következő formátumokat ismeri :

Currency A számok elé, vagy mögé a beállított pénzjelet írja, és a meghatározott karakterekkel (vagy,) elválasztja az ezreseket.

Punctuated A Currency-hez hasonlóan elválasztja az ezreseket, de nem teszi ki a pénzjelet.

Fixed A tizedesjegyek számát rögzíti (pl: 3.2 **Fixed** típussal 3 tizedessel **Decimals** - 3.200).

% A számot megszorozza 100-zal, és egy % jelet ír a végére.

General A számok úgy jelennek meg, ahogy a legrövidebben leírhatók.

Scientific Tudományos kijelzés. A számokat 1 és 10 közé eső számok és 10 megfelelő hatványának szorzataként jeleníti meg (pl: 321 = 3.21E+02).

Bar-Graph A szám egész részének megfelelő mennyiségű + vagy - jelet ír a szám előjelétől függően.
(0 értéknél egy .-ot ír.)

A Date és Time formátumok közül egy menüből választhatunk, amelynek két-két utolsó menüpontját mi magunk állíthatjuk be a **Services** menü **Configuration-International** pontjában. Itt adhatjuk meg a **Currency** és a **Punctuated** formátumok "pontozó karakter"-eit is. (Lásd még a FORMATTÁLÁS fejezet)

Ha olyan numerikus adatot viszünk a cellába amelynek kijelzéséhez szélesebb cellára lenne szükség, a program egy sor csillagot rajzol ki. Például egy 5 karakter széles cellába kívánjuk kiíratni a -1234.56-ot. Ilyenkor segíthet a Táblázatkezelő menü (**F10**) **Width-Set** pontja, ahol beállíthatjuk az aktuális oszlop szélességét.

Példák numerikus értékekre :

-123.45	-1.26*E+02
92-jan-1	A dátum és az idő is numerikus érték,
15:30:13	csak a kiírás formátuma ilyen.
#TRUE	A logikai érték is numerikus, az igaz értéke 1
#FALSE	a hamisé 0.

Karakterláncok

Karakterlánc típusú adat a ', " vagy a ^ valamelyikével kezdődik. Ezeknek a jeleknek (amelyeket **címkeelőtag**-nak nevezünk) különleges szerepe is van, nemcsak az adat típusának a jelzése. A program a címkeelőtagtól függően rendezi az adatot a cellán belül. A ' karakter a balra igazítást, a " karakter a jobbra igazítást, a ^ karakter a középre igazítást jelöli. Mivel egy cellába akár 240 karaktert is bevihetünk, az igazítás csak akkor érvényesül, ha a cella szélessége nem kisebb a karakterlánc hosszánál. Ellenkező esetben az igazítás elmarad, valamint a karakterlánc egészen addig látszik a munkaterületen, amíg az adott cellától jobbra eső cellák rendre üresek. Az első nem üres cella adatai "lefedik" a balra lévő cella karakterláncát.

Ha egy számjeggyel kezdődő karakterláncot szeretnénk beírni egy mezőbe, akkor az adatbevitelt egy ', " , vagy egy ^ karakterrel kell kezdenünk.

Példák karakterlánc értékekre :

Kutya
ALMAFA
HELYBELI
TÉGLAGYÁR
'23.000 A címkeelőtag miatt karakterlánc

Formulák

Formuláknak nevezzük a számított értékeket. A formulák lehetnek numerikus, vagy karakteres értéket adó számítások.

A formulák előírásánál használhatunk cellahivatkozásokat, ilyenkor a program mindig a megcímzett cella aktuális értékét fogja alapul venni a számítás elvégzésénél. A számítást a táblázatkezelő beállítótáblájában megadott módon végzi. (Táblázatkezelő menü (**F10**) **Settings - Recalculation**)

Előírható az automatikus újraszámítás, amikor a cellák adatainak változása esetén minden külön felszólítás nélkül az összes formulát újraszámolja, míg a kézi (**Manual**) újraszámítási mód esetén csak az **F8 (CALC)** billentyű leütésére számol (**rekalkuláció**).

Ezen kívül azt is megadhatjuk, hogy a számolást milyen sorrendben végezze. Sorról sorra (**Row by row**), oszlopról oszlopra (**Column by column**), vagy a cellahivatkozások sorrendjében (**Natural**).

Amennyiben a formulák bevitele folyamán olyan cellára hivatkozunk, amelyiknek értéke függ az aktív cellától akár közvetlen módon is, **ciklikus cellahivatkozást** készítettünk. A program szempontjából ennek semmi akadálya, de az előbbiekben említett almenüben be kell állítani a ciklikus hivatkozások számításának mértékét (**Iterations**). A program a megadott számszor megy végig a "körön", mindig az aktuális értékkel számolva. Az iterációk száma 1-től 50-ig terjedhet.

Abban az esetben, ha téves eredményt kapnánk egy formula eredményeképp (pl: 0-val való osztás), a cella értéke amelyikben a formula van, **ERR** lesz. Ez egy különleges jelzőérték, amelyet a beépített **@ISERR** függvénnyel le is kérdezhetünk.

Ha a bevitelre szánt adat egy cellacímmel kezdődik, és számítást szeretnénk előírni, akkor a + vagy - karakterrel kezdjük a bevitelt. (pl : +A23/2).

A SYMPHONY által ismert matematikai, logikai és karakterlánc műveletek és prioritásuk

Jel	funkció	prioritás
^	hatványozás	7
+, -	előjelek	6
*, /	szorzás, osztás	5
+, -	összeadás, kivonás	4
=, <=, >=, <>, <, >	összehasonlítás	3
#NOT#	logikai nem	2
#AND#	logikai és	1
#OR#	logikai vagy	1
&	szövegek összekapcsolása	1

Példák a formulákra :

ALMA&FA===>ALMAFA +A

Különleges formula a \ jellel előírt, karakteres eredményt adó formula. Ha egy adat bevitelét a \ jellel kezdjük, akkor a \ utáni karakterlánc fog ismétlődni, a cella teljes szélességében. Még akkor is, ha utólag megnöveljük a cella szélességét.

Ha a bevitelkor a \=+ billentyűket nyomjuk le egymás után, és az aktuális cella szélessége 15 karakter, a munkaterületen a =+=+=+=+=+=+=+= karaktersorozatot fogjuk látni.

Függvények

A SYMPHONY táblázatkezelőjében különböző beépített függvényeket alkalmazhatunk számításaink elvégzésére. Ezek a függvények kivétel nélkül a @ karakterrel kezdődnek. Több csoportra oszthatók az elvégzett művelet alapján. Vannak **matematikai, karakteres, pénzügyi, dátum és idő, statisztikai, adatbázis, valamint különleges** függvények. (A függvények leírását lásd a FÜGGELÉK-ben)

Formulák alfejezetben leírt újraszámítás és értékelés a függvényekre is igaz.

Példák a függvényekre :

@NOW

@ISERR(A23)

@RATE(5000,3000,5)

4.2.5 Adatok javítása

A cellákba vitt adatokat természetesen javíthatjuk is, nem kell mind a 240 karaktert újragépelni, ha a 213. karakter hibás. Ehhez a hibás adatot tartalmazó cellára kell állítani az aktív cella kijelölést és az **F2 (EDIT)** gombot lenyomni. Ennek eredménye az lesz, hogy a cella tartalmát a program felajánlja a képenyő második sorában szerkesztésre. A kurzor a kifejezés utolsó karaktere után áll. Ha a cellatartalom nem fér el a második sorban, akkor csak a végét látjuk. Ekkor a jobbra és balra gombokkal egy karakternyit, a kontroll és jobbra ill. balra gombokkal öt karakternyit lép a kurzor a megfelelő irányban. A **Home** gombra a bevitel elejére, az **End** gombra a bevitel végére áll. A szerkesztést befejezhetjük az **ENTER** billentyűvel, ilyenkor az aktív cella kijelölés nem mozdul, a föl vagy le gombok hatására a kijelölés is elmozdul az adott irányban.

4.2.6 Másolás és többszörözés

A táblázatban elhelyezett adatok másolására a Táblázatkezelő menü **(F10) Copy** parancsát használhatjuk. A művelet mindig egy tartománnyal dolgozik, de amint azt a TARTOMÁNYOK fejezetben olvashattuk, megadhatunk egy cellát is tartományként (pl: A1..A1).

Másolás során egy igen érdekes lehetőség, hogy a másolandó tartományt akár meg is többszörözhetjük egyetlen utasítással. Ehhez a funkcióhoz is a **Copy** parancsot használjuk, de a másolandó tartománynál nagyobb tartományt adunk meg mint célt. A többszörözés lehetőségeit a 4. ábra szemlélteti.

Cella többszörözése

Forrás (from): A1 Cél (to): A3..C6

	A	B	C	D
1	15.5			
2				
3	15.2	15.2	15.2	
4	15.2	15.2	15.2	
5	15.2	15.2	15.2	
6	15.2	15.2	15.2	

Sor többszörözése

Forrás (from): A1..C1 Cél (to): B3..D6

	A	B	C	D
1	15.2	13.4	-2.8	
2				
3		15.2	13.4	-2.8
4		15.2	13.4	-2.8
5		15.2	13.4	-2.8
6		15.2	13.4	-2.8

Oszlop többszörözése

Forrás (from): A1..A3 Cél (to): C2..D2

	A	B	C	D
1	15.2			
2	3.14		15.2	15.2
3	8.6		3.14	3.14
4	25.1		8.6	8.6
5	49.32			
6				

Tartomány többszörözése

Forrás (from): A1..B4 Cél (to): A3

	A	B	C	D
1	15.2	13.4		
2	4.98	9.6		
3	15.2	13.4		
4	4.98	9.6		
5	15.2	13.4		
6	4.98	9.6		

4. ábra

FIGYELEM !!!

A tartomány többszörözése csak akkor működik, ha a forrástartomány és a céltartomány fedik egymást, az ábrán látható módon.

4.2.7 Adatmozgatás

Az adatok mozgatása azt jelenti, hogy a kijelölt tartomány celláinak tartalma megszűnik (NA lesz), és a célként kijelölt tartomány cellái veszik fel az értékeket. Amennyiben erre a funkcióra van szükségünk, a Táblázatkezelő menü (F10) **Move** parancsát használhatjuk.

Mozgatás esetén nincs lehetőségünk a többszörözésre úgy, ahogy azt a Copy parancsnál láttuk.

4.2.8 Formattálás

A BEVIHETŐ ADATOK fejezetben láthattuk, hogy a SYMPHONY hányféle formátumban képes megjeleníteni a numerikus értékeket. Az ott nevezett formátumokat a Táblázatkezelő menü (F10) **Format** parancsával tudjuk beállítani oly módon, hogy a parancs aktivizálása és a formátum kiválasztása után kijelöljük azt a tartományt (lásd a TARTOMÁNYOK fejezetet), amelyben a formátumot érvényesíteni akarjuk.

4.2.9 Oszlopok és sorok kezelése

A SYMPHONY programban lehetőségünk van egyes oszlopok, vagy sorok törlésére, esetleg a táblázatba újabb sorok vagy oszlopok beszúrására. Amikor ezekkel a funkciókkal dolgozunk megadhatjuk, hogy a be-

szúrás, vagy törlés az aktuális ablak mozgási korlátain belül, vagy az egész munkaterületen (**Global**) érvényes. Beszúrásra használjuk a Táblázatkezelő menü (**F10**) **Insert**, törlésre a **Delete** opcióját.

Mindkét esetben három lehetőség közül választhatunk. Oszloppal (**Columns**), vagy sorral (**Rows**) dolgozunk a mozgási korlátokon belül, vagy az egész munkaterületen (**Global**).

Ha Globalt választunk, akkor ezután kell megadnunk, hogy oszlopokkal, vagy sorokkal dolgozunk. Ezután jelölhetünk ki akár egyszerre több sort, vagy oszlopot is.

FIGYELEM !!!

Ha egy sort, vagy oszlopot szúrunk be a táblázatba, akkor a sor alatti, vagy az oszloptól jobbra eső sorok, vagy oszlopok adatai jobbra ill. lefelé elmozdulnak, a mozgási korlátok megnőnek.

4.2.10 Oszlopok elrejtése

A munkatáblázat egyes oszlopainak megjelenítését ideiglenesen le is tilthatjuk a Táblázatkezelő menü (**F10**) **Width - Hide** menüpontjával. Erre akkor lehet szükség, ha egyidőben szeretnénk látni a munkaterületen egymástól távol eső oszlopokat.

A rejtett oszlopok tartalma megmarad, tartomány kijelölésének idejére megjelennek a képernyőn, de a betűjelük mellett egy * jel lesz. Az elrejtett oszlopokat újra láthatóvá a Táblázatkezelő menü **Width - Display** pontjával tehetjük.

4.2.11 Táblázat nyomtatása

Mint azt a NYOMTATÁS fejezetben láthattuk, adatainkat kinyomtathatjuk közvetlenül a printerre, de egy szöveges állományba is (**Services-Print-Settings-Destination**).

Mindkét esetben választhatunk, hogy táblázatos, vagy szöveges formában szeretnénk látni az adatokat (**Services-Print-Settings-Format**). Ha szöveges módon nyomtatunk, akkor csak a munkaterületen látható adatokat fogjuk látni a célállományban, vagy a nyomtatón, ha táblázatos formát választunk, akkor a képernyőn a munkaterület keretét (az oszlopok betűjeleit, és a sorok számozását) is látni fogjuk.

A nyomtatón, vagy a célállományban a formulák, és a függvények helyén a nyomtatáskori pillanatnyi értékek lesznek. Ha a kifejezéseket és a függvényeket szeretnénk látni, nyomtatás előtt a kurrens cellák formátumát változtassuk (Táblázatkezelő menü **Format**) **Literal**-ra.

4.3 Szövegszerkesztő

A SYMPHONY program szövegszerkesztője is a munkatáblázatban (**WORK SHEET**) dolgozik. A szövegszerkesztőt az Üzemmód menüben (**Alt+F10**) kiválasztott **DOC** menüponttal vehetjük igénybe. Ilyenkor a képernyő jobb felső sarkában a módmutató **"DOC"**-ra vált.

4.3.1 Szövegbevitel

A szövegszerkesztő üzemmódban a képernyőn nem látunk cellákat, az ablak keretén nem betűket és számokat látunk, hanem az aktuális formátumnak megfelelő **formátumsor**-t (Lásd a SZÖVEG FORMÁZÁSA fejezetet). Az aktív cella kijelölés sem látható, mivel nincsenek cellák, helyette egy kurzor van az aktuális beviteli helyen. (A kurzor helyét az eltérő színnel kiírt karakter jelzi.) A szövegszerkesztőben egyes műveleteknél különleges szerepe van a bekezdéseknek (**Paragraph**).

A bekezdések végét, új bekezdés kezdetét az **Enter** billentyűvel jelölhetjük. Elég a szöveget folyamatosan gépelni, ha a gépelés során elérjük a lap szélét, a szó átkerül a következő sorba anélkül, hogy az Enter billentyűt leütnénk.

Ha egy bekezdést például rendezni szeretnénk, a program a bekezdés záró sora kivételével (amelyikben az Enter van) az összes sort rendezni fogja a bekezdésben.

A szövegszerkesztő a bevitt szövegsorokat az adott sor első oszlopában lévő cellákban helyezi el. Mivel egy sor egy cellában van, a sorok maximális hossza **240** karakter lehet.

4.3.2 Mozgás a szövegben

A szövegszerkesztő üzemmódban a "nyíl" billentyűk a kurzort mozdítják a megfelelő irányba egy karakternyit. A **Home** billentyűvel a szöveg elejére vihetjük a kurzort, a **PgUp** és a **PgDn** billentyűkre a kurzor egy képernyőnyivel feljebb, illetve lejjebb kerül. A **Ctrl** és a **jobbra** ill. **balra** nyilakkal, egy szót mozdul a kurzor. **Ctrl+PgDn** ugrás a következő lap elejére, **Ctrl+PgUp** ugrás az előző lapkezdésre.

Az End billentyűnek a szövegszerkesztőben is kapcsoló szerepe van. Hatását a következő felsorolásban olvashatjuk :

End+le	A bekezdés végére
End+fel	A bekezdés elejére
End+balra	sor elejére
End+jobbra	sor végére
F2	a következő formátumsorra
Alt+F2	a megelőző formátumsorra

4.3.3 A szövegszerkesztő editáló billentyűi

A szövegszerkesztő üzemmódban egyes billentyű-kombinációkhoz rendeltek egy-egy funkciót, amelyet így közvetlen módon, a Szövegszerkesztő menü használata nélkül is aktivizálhatunk. A következő felsorolás ezeket a funkciókat és billentyű-kombinációkat mutatja be :

Ins	Beszúró mód ki/be kapcsolása
Ctrl+N	új lap kezdete
Alt+F3	új sorba viszi a kurzortól jobbra eső karaktereket
Ctrl+C	Szövegrész másolása
Ctrl+M	Szövegrész mozgatása
F4	Szövegrész törlése
Ctrl+D	Sor törlése
Ctrl+T	A sor törlése a kurzor előtt
Ctrl+Y	A sor törlése a kurzor után
Ctrl+BackSpace	Egy szó törlése a kurzor előtt
Ctrl+P	A legutóbb másolt, törölt, vagy mozgatott blokkot a kurzor pozíciójában kiírja
Ctrl+S	Adott szövegrész keresése
Ctrl+R	Adott szövegrész keresése és cseréje
Ctrl+F	új formátum sor készítése
Alt+F4	A sor középre igazítása
Ctrl+J	Automatikus hasábos rendezés be/ki kapcsolása
F2	Bekezdés hasábos rendezése
Ctrl+O	Karaterek összefűzése (Amikor a szöveget nyomtatjuk, akkor ezen a helyen a nyomtatófej egy karakternyit visszalép, így egymásra nyomtatja az előző és az ezt követő karaktereket.)
Ctrl+X	A kurzor pozíciójában lévő karaktert kis-, vagy nagybetűs párjára cseréli
F3	Bekezdés bal margójának átállítása

Ctrl+B Nyomtatási attributum kijelölés kezdete

Ctrl+E Nyomtatási attributum kijelölés vége

4.3.4 Szöveg másolása, mozgatása, törlése

A fejezetcímben említett műveletekhez tartozó parancsokat a Szövegszerkesztő menüjében (**F10**) találjuk. A másolást és a mozgatást csak a szövegben kijelölt tartományokkal (**blokk**-okkal) végezhetjük.

Másoláshoz a **Copy** parancsot, mozgatáshoz a **Move** parancsot használjuk. Mindkét esetben miután kiválasztottuk a kívánt funkciót, a szövegrész kijelölése következik.

A program a kurzor helyét automatikusan a műveletben résztvevő blokk egyik végének tekinti. A tartomány kijelöléshez hasonlóan ezt az **Esc** billentyűvel érvényteleníthetjük, majd a megfelelő helyre állva, a **Tab** billentyűvel rögzíthetjük. Ezután a kurzormozgató billentyűkkel a blokk másik végére állunk. Bármely billentyűkombináció használható amelyik elmozdítja a kurzort, így például az **End+Home** is. A kijelölt blokk sorfolytonos, vagyis az első karaktertől indul a sor végéig, és a következő sor elejétől folytatódik.

Miután a blokkot kijelöltük, szintén a kurzormozgató billentyűk segítségével meg kell jelölnünk, hogy hova kérjük a másolatot. Másolás esetén a kijelölt blokk a szövegben marad, mozgatás esetén átkerül az új helyre.

FIGYELEM !!!

A másolás és a mozgatás mindig beszúrásos üzemmódban készül. Tehát a másolt, vagy mozgatott szöveg "helyet csinál magának" a célként kijelölt helyen. Egyetlen karakter törlésére használjuk a **Del** vagy a **"szürke balra nyíl"** (**BackSpace**) gombot. A Del gomb hatására a kurzor helyén látható karakter, a BackSpace hatására a kurzor előtti karakter törlödik, majd a törölt karaktert követő karakterek egyet előbbre lépnek. A BackSpace használatakor, a kurzor egy karakternyit előbbre lép.

Szövegrész törlésére a Szövegszerkesztő menü (**F10**) **Delete** pontját, vagy a **Ctrl+F4** billentyűkombinációt használjuk. A másoláshoz hasonlóan kijelöljük a szövegrészt, majd az **Enter** billentyűvel érvényesítjük a törlést.

4.3.5 Keresés és csere

Szövegünkben a SZÖVEGSZERKESZTŐ BILLENTYŰI fejezetben leírt billentyűkombináció (**Ctrl+S**), vagy a Szövegszerkesztő menü **Search** (keresés) parancsának segítségével megkereshetjük egy adott szövegrész (szó) előfordulásait. Miután megadtuk a keresendő szöveget, egy menüből választhatunk, hogy a kurzor utáni (**Forward**), előtti (**Backward**) részen keresse. A keresés funkciót a **Quit** paranccsal, vagy az **Esc** billentyűvel fejezhetjük be. Ha a **Quit** parancsot választjuk, a kurzor az utolsóként megtalált szövegrész elején marad, **Esc** esetén visszamegy a keresés előtti pozícióba.

A **Ctrl+R** billentyűkkel, vagy a Szövegszerkesztő menü **Replace** (csere) parancsával a megtalált szövegrészt ki is cseréltethetjük. A funkció indítása és a cserélendő majd az új szövegrész megadása után a kurzor utáni szövegrészben megkeresi az adott szövegrész első előfordulását, és a következő menüpontokat ajánlja fel :

Once A megtalált szó cseréje után a funkciónak vége

Continue A megtalált szó cseréje után a következő előfordulásra áll

Skip A szót nem cseréli, és a következő előfordulásra áll

All-Remaining Minden előfordulást cserél, és a funkciónak vége.

Ebből a funkcióból is kétféleképp léphetünk ki, a Quit menüponttal, vagy az Esc billentyűvel. Ha az Esc billentyűt ütjük le, a kurzor visszakerül a csere előtti pozícióba, de a cserék megmaradnak.

4.3.6 Szöveg formázása

Szövegünk végső kinézetét különböző módon befolyásolhatjuk. Egyrészt rendeztethetjük a sorokat a beállított margók között, másrészt kiemelt módon írhatunk szavakat. A sorok rendezettségét előírhatjuk a **formátumsorok** használatával, vagy a SZÖVEGSZERKESZTŐ BILLENTYŰI fejezetben említett billentyűkkel.

Formátumsor kezelésére használjuk a Szövegszerkesztő menü (**F10**) **Format** almenüjét. Egy-egy formátumsort a szövegben bárhol elhelyezhetünk a Szövegszerkesztő menü (**F10**) **Format-Create** menüpontjával, vagy módosíthatunk egy elhelyezett formátumsort a **Format-Edit** menüponttal. A **formátumsor**-ban látható tömör jobbra mutató nyílfejek a tabulátorok helyét jelzik. A **Tab** billentyű lenyomása esetén a kurzor mindig a következő ilyen jelölt helyre megy. A sor elején látható **L** betű a bal, a tabulátorok után látható **R** betű a jobb margó helyét mutatja. A szövegszerkesztő e között a két margó között engedi a sorok bevitelét. A formátumsor utolsó előtti karaktere egy **c**, **l**, **r**, vagy egy **e** karakter. Ez a karakter a két margó közötti elrendezést jelöli. Az **l** betű esetén (**left**) a sorokat, ahogy bevittük annyi szóközzel a bal, **r** esetén (**right**) a jobb margóhoz "húzza". C esetén (**center**) a sorokat a két margó között középre igazítja. **J** betű a hasábos rendezést (**justified**) jelzi.

Ilyenkor a sorokat a szövegszerkesztő automatikusan annyi szóköz karakterrel tölti fel, hogy a sor első betűje a bal margón az utolsó betű pedig a jobb margón legyen.

Kiemelt szavak írására a **nyomtatási attributumok**-at használhatjuk. Ha egy szó gépelése előtt lenyomjuk a **Ctrl+B** billentyűket, egy egykarakteres menüpontokat tartalmazó menüt kapunk a képernyő második sorában. A menü karaktereinek jelentése a következő :

B	Bold (**Vastag**)	**2**	Bold Italic Underlined
U	Underlined (Aláhúzott)	**3**	Italic Underlined
I	Italic (*Dőlt*)	**4**	Bold Superscript
+	Superscript ($^{\text{Felső index}}$)	**5**	Italic Superscript
-	Subscrip ($_{\text{Alsó index}}$)	**6**	Bold Subscript
X	Strike-through (~~Áthúzott~~)	**7**	Italic Subscript
0	Bold Italic	**8**	Bold italic Subscript
1	Bold Underlined	**9**	Bold Italic Superscript

A nyomtatási attributumok érvényességének végét, a normál karakterhez való visszatérést a Ctrl+E billentyűk leütésével jelezhetjük.

4.3.7 Szöveg nyomtatása

A begépelt szöveget a NYOMTATÁS fejezetben említett **Print** almenü segítségével nyomtathatjuk. Ha céleszközként egy állományt határozunk meg, ugyanolyan szöveges állomány jön létre, mintha azt egy bármilyen szövegszerkesztővel készítettük volna.

Szöveg nyomtatásánál nincs értelme a táblázatos formátumú nyomtatásnak, bár a beállítótáblában (**Settings**) kiválaszthatjuk.

4.4 Grafika

Az ÜZEMMÓDOK fejezetben olvashattuk, hogy a SYMPOHONY bizonyos grafikák elkészítésére és kinyomtatására is képes. A grafikák nem tetszőleges ábrák és rajzok, hanem a menedzserszerű munkák megkönnyítését szolgáló görbék, vagy diagramok.

A grafika kezelését támogató menühöz kétféleképp juthatunk el. Egyik módszer, hogy az Üzemmód váltó menü (**Alt+F10**) segítségével **GRAPH** módra váltunk, majd lenyomjuk az **F10** gombot (**Grafikai** menü).

Elérhetjük ugyanezt a Grafikai menüt, ha a **Táblázatkezelő** menü **Graph** pontját választjuk. Ez azért hasznos a számunkra, mert mint a későbbiekben is látni fogjuk, grafikonjainkat a munkatáblázatban elhelyezett adataink alapján készíthetjük, és így nem kell az üzemmódok között váltogatni.

A Grafikai menü pontjai :

Preview	Az elkészített grafikon megjelenítése
1-st Settings	Első beállítótábla
2-nd Settings	Második beállítótábla
Image-Save	Kép elmentése háttértárolóra (lásd később)
Quit	Kilépés a Grafikai menüből

4.4.1 Grafikon készítése

Grafikonjainkat a munkatáblázatban elhelyezett adataink alapján tudjuk elkészíteni. Ebben áll rendelkezésünkre a **GRAPH** üzemmód két beállítótáblája.

Először tekintsük át az első tábla adatait :

Type A grafikon típusát választhatjuk meg. A grafikonunk lehet vonal (**Line**), oszlop (**Bar**), pakolt oszlop (**Stacked Bar**), **XY** formátumú, mint egy matematikai függvény, kör (**Pie**) vagy tőzsdei (**High-Low-Close-Open**) grafikon.

Range A grafikon elemeinek értékkészleteit és értelmezési tartományát adhatjuk meg. Vonal, oszlop és pakolt-oszlop grafikonnál hat értékkészletet, XY és körgrafikon esetén egy, tőzsdei grafikon esetén négy értékkészletet adhatunk meg. Tehát a grafikus üzemmódban egy X értékhez maximum hat Y értéket rendelhetünk.

Hue Színezési információk beállítása. Nem színes grafikon esetén csíkozási információ megadása.

Format Vonal és XY grafikon esetén választhatunk, hogy csak a pontokat tűzze ki (**Symbols**), csak a pontokat összekötő vonalakat rajzolja meg (**Lines**), mindkettőt (**Both**) vagy egyiket sem (**Neither**).

Data-Labels A kitűzött pontokhoz rendelt tájékoztató adatok megadása. Kérhetjük a pontok fölött (Above), alatt (Below), mellett jobbra (Right) vagy balra (Left) történő megjelenítést. Az elhelyezkedés megadásának csak vonal és XY grafikon megadása esetén van értelme.

Legend A grafikon alatt kigyűjti a jelöléseket és rövid megjegyzéseket fűzhetünk hozzájuk.

A második beállítótábla adatai :

Titles A grafikon fölé írandó első (**First**) és második (**Second**) sort, valamint az X (**X-Axis**) és Y (**Y-Axis**) tengelyre írandó szöveget adhatjuk meg.

Y-Scale A függőleges skála adatainak beállítása.

Type A skála típusa lehet beállított kezdő és végpontú (**Manual-Linear**), számított (**Automatic-Linear**), vagy logaritmikus (**Logarithm**).

Format A táblázatkezelőben leírt adatformátumoknak megfelelően beállítható, hogy milyen formában írja ki az osztásokat.

Exponent A kiírt skálaosztásokat automatikusan elosztja 10-zel, 100-zal, 1000-rel és az osztás mértékét angolul kiírja (**Automatic**), vagy 10-nek az általunk megadott hatványával osztja el a skála értékeit (**Manual**).

Width A skála adatainak kiírására, mennyi karaktert engedélyezünk a grafikon bal oldalán?

X-Scale Ugyanazokat az adatokat adhatjuk meg az X tengely kirajzolásához, mint az Y tengely esetén, a szélesség (**Width**) adat kivételével.

Other menüpont alatt találhatóak a következő adatok :

- **Grid** Vízszintes (**Horizontal**), függőleges (**Vertical**), vagy mindkét irányú vonalak rajzolása a skálaosztásokhoz.
- **Hide** A cimkék és skálaosztások kijelzésének letiltása.
- **Color** Színes grafika engedélyezése. Ha nyomtatni kívánjuk a grafikont, ne kapcsoljuk be ezt az opciót, mert a nyomtatásban minden oszlop, vagy csomópont egyforma lesz.
- **Skip** Megadhatjuk, hogy minden "hanyadik" osztást írjon ki az X tengelyre.
- **Origin** A megadott értékhez egy vízszintes vonalat rajzol és ehhez a vonalhoz képest fölfelé, vagy lefelé rajzolja az oszlopokat.
- **Aspect** Kör-grafikon esetén a megadott számnak megfelelően forgatja el a grafikont a "térben". Az 1-nél kisebb számok a függőleges tengely, az 1-nél nagyobbak a vízszintes tengely mentén történő elforgatást jelentik.

Most nézzük végig egy grafikon elkészítésének lépéseit.

Tegyük fel, hogy van egy olyan munkatáblázatunk, amely négy oszlopból áll, és a következő információkat tartalmazza :

	A	B	C	D
1		Éves bevételek 1990-ben		
2				
3		Fizetés		Egyéb bevétel
4		bruttó	nettó	
5				
6	jan	6.543,00-	4.907,25-	2.345,00-
7	feb	6.781,00-	5.085,75-	
8	már	5.988,00-	4.491,00-	9.763,00-
9	ápr	7.345,00-	5.508,75-	2.000,00-
10	máj	7.320,00-	5.490,00-	
11	jún	6.784,00-	5.088,00-	1.850,00-
12	júl	6.853,00-	5.139,75-	8.534,00-
13	aug	7.201,00-	5.400,75-	
14	szept	6.977,00-	5.232,75-	
15	okt	6.457,00-	4.842,75-	5.670,00-
16	nov	7.003,00-	5.252,25-	
17	dec	6.935,00-	5.201,25-	5.000,00-

Szeretnénk ezekből az adatokból egy olyan grafikont készíteni, amelyen egy-egy vonal jelöli a bruttó és a nettó bevételeket, valamint egy-egy pont az egyéb jövedelmeket.

A következő lépéseket kell végigcsinálnunk:

- a **Táblázatkezelő** menüben kiválasztjuk a **Graph** almenüt,
- majd azon belül az első beállítótáblát (**1–st Settings**)
- beállítjuk a grafikonunk típusát (**Type - Line**)
- ezután definiáljuk az X tengely osztását (**Range - X - A6..A17**).

Ennek eredményeképp a grafikonon az X tengely az A oszlopban lévő 12 hónapra lesz elosztva. Definiáljuk a különböző értékkészleteket. Az A–val jelölt görbe legyen a bruttó, a B–vel jelölt a nettó, és a C–vel jelölt az egyéb bevételek görbéje.

A kijelölés lépései :

Range **- A -** **B6..B17** (Tartomány kijelölés)

- B - **C6..C17**

- C - **D6..D17**

Quit

Ezután az egyéb jövedelmek formátumát beállítjuk az eredeti szándékainknak megfelelően úgy, hogy csak a pontokat tűzze ki, de ne kösse össze őket. (**Format - C - Symbols**).

Hogy ne csak pontok legyenek szétszórva a grafikonunkon, az egyéb jövedelmeket jelző szimbólumok mellé ki is íratjuk az összegeket. (**Data-Labels - C - D6..D17 - Right**).

Az első táblában már csak a grafikon aljára kigyűjtött megjegyzéseket kell megadnunk. (**Legend - A - BRUTTO FIZ.** /begépelni/- **B - NETTO FIZ. - C - EGYEB BEV. - QUIT**).

Az első tábla után a **Switch** menüponttal kapcsoljunk át a második beállítótáblához. Itt adhatjuk meg a grafikon fölé és a tengelyekre írandó címkéket, a skálaosztásokat, stb. Először adjunk nevet a grafikonunknak és a tengelyeknek :

Titles

First	- JOVEDELMEK
Second	- 1990-BEN
X–Axis	- Honapok
Y–Axis	- Osszegek

Quit.

Az első (**First**) és a második (**Second**) címke egymás alatt lesznek a grafikon fölött, középen. Az Y skála legnagyobb értéke ha a programra bízzuk a beosztások elkészítését, a kijelölt értékkészletek legnagyobb értéke lesz, vagyis egy olyan adat biztosan lesz majd, amelyik a grafikonunk felső szélét érinti.

Ebben a grafikonban most ezt küszöböljük ki, a következő utasítássorozattal :

Y–Scale

Type	**Manual**
Lower limit :	0
Upper limit :	10000.

Mivel az Y skála összegeket jelöl, a skálaosztások kiírásához alkalmazzuk a Currency formátumot (lásd a TÁBLÁZATKEZELŐ fejezetet) :

Y–Scale

Format

Currency

decimals : 0.

A skálaosztások értékét a program elosztaná 1000-rel, és az Y tengely mellé kiírná zárójelben a **THOUSANDS** szót ami arra utal, hogy a skálaosztásokat ezresekben kell értelmezni. Ezt is elkerülhetjük a következő módon :

Y–Scale

Exponent

Manual - 0 /begépelni/.

Hogy a programnak elég helye legyen kírni a skálaosztásokat teljes mértékben pénzjellel együtt, az **Y–Scale - Width - 12** utasítássorral 12 karakternyi területet adunk a kiírásra.

A második tábla további adatait hagyjuk alapértéken. Az első tábla kezelésére szolgáló menüben van még két menüpont. Az egyik a **Name**, ez az elkészített beállítás elnevezésére (**Create**), vagy egy elnevezés törlésére (**Delete**), kiválasztására (**Use**), a sorban következő (**Next**) vagy az előző (**Previous**) névvel ellátott beállítás kiválasztása.

Az elnevezés lehetőségét kihasználva egy paranccsal megváltoztathatjuk grafikonunkat a kívánt formátumúra. Ebben az almenüben lehet visszaállítani az összes beállítást az alapértékre (**Initial-Settings**).

A második menüpont a **Cancel**. Ebben az almenüben a beállítás egyes részleteit törölhetjük. A második tábla kezelésére szolgáló menüben a Name paranccsal ugyanazt az almenüt hívhatjuk elő, amit az első beállító menüben.

4.4.2 Grafikon megtekintése

Elkészült grafikonjainkat megnézhetjük a képernyőn a Grafikai menü **Preview** pontjának kiválasztásával. Erre azonban csak akkor van lehetőség, ha a program telepítésénél a megfelelő képernyőtípust adtuk meg, és a képernyő képes a grafikus üzemmódra.

4.4.3 Grafikon nyomtatása

Az elkészült grafikon nyomtatására a központi programban "sajnos" nincs lehetőség. Csak a RENDSZER BEMUTATÁSA fejezetben említett **PrintGraph** programmal tudunk nyomtatni. Ehhez szükséges, hogy a rendszer telepítésekor megfelelő grafikai lehetőségeket biztosító nyomtatót adjunk meg az erre vonatkozó kérdésre, valamint meg kell lennie a **PGRAPH.EXE** programnak. A nyomtatás menete a következő. Miután a **Preview**-val átnéztük a grafikont és megfelelőnek találtuk, az **Image–Save** menüponttal mentsük el a grafikont háttértárolóra.

FIGYELEM !!!

Fontos, hogy az elmentett grafikon kiterjesztése **.PIC** legyen (Ha nem adjuk meg a program ezt automatikusan megteszi) !

Indítsuk el a **PrintGraph** programot (pl: **C:\PGRAPH és Enter**). A képernyő középső részén megjelenő menüből a következő opciókat választhatjuk :

Image-Select A beállított könyvtárban lévő **.PIC** kiterjesztésű állományokat felsorolja a képernyőn, és ezek közül választhatunk, mint egy menüből.

Settings Ebben a programban is helyet kapott egy beállítótábla, ahol a képre vonatkozó előírásainkat (**Image**), a képeket tartalmazó alkönyvtárat (**Hardware - Graphs-Directory**), betűtípusokat tartalmazó alkönyvtárat (**Hardware - Fonts-Directory**), a papír méretét (**Size-Paper**), stb. adhatjuk meg.

Go A nyomtatás kezdete a beállításoknak megfelelően.

Align Hasonlóan a SYMPHONY programban látottakhoz, ha kézi vezérléssel álltunk a lap elejére, ezzel a menüponttal kell újra beállítani a PrintGraph belső számlálóit.

Page Egy lapdobás (**Form Feed**) a nyomtatón.

Exit Kilépés a PrintGraph programból.

Tegyük fel, hogy szükségünk van az előbb elkészített grafikon beillesztésére az elkészült jelentés egyik oldalának alsó felére. Sőt a tisztelt megrendelő a grafikont 45 fokos szögben elforgatva szeretné látni. Az eljárás a következő :

- Elindítjuk a **PrintGraph** programot, és a Settings menüpontot választva megadjuk az elmentett grafikonunk helyét (pl: F:\PROGS\SYMPHONY\MUNKAK).
- Megadjuk a címke nyomtatásához használatos betűtípusokat tartalmazó állományok helyét (Ezek az állományok **.FNT** kiterjesztéssel rendelkeznek, és általában abban a könyvtárban vannak, ahol a rendszer többi állományai. pl: F:\PROGS\SYMPHONY).
- Kiválasztjuk a megfelelő nyomtatót (**Settings- Hardware - Printer**).
- Beállítjuk az alkalmazott papír méretét **inch**-ben (**Settings - Hardware - Size–Paper** /1 inch = 2.54 cm/).
- A beállítótábla **Actions** menüpontjával beállíthatjuk, hogy nyomtatás előtt várjon–e egy billentyű leütésére (**Pause**), és a végén a következő lap elejére álljon–e (**Eject**).
- Ezután, mivel a képet a papír alsó felén szeretnénk látni, az **Image - Size** (Képméret) almenüben először a félkép (**Half**) beállítást választjuk.

Ekkor a program a papírméretnek megfelelően kitölti a beállítótábla következő adatait :

Top A grafikon felső szélének távolsága a lap felső szélétől (inch).

Left A grafikon bal szélének távolsága a lap bal szélétől (inch).

Width A grafikon szélessége (inch).

Height A grafikon magassága (inch).

Rotation A grafikon elforgatásának mértéke (fok).

Miután a programmal kitöltettük az adatokat, már csak a Top és a Rotation értékét kell megváltoztatnunk. Ehhez a Settings - Image - Size - Manual almenüt vesszük igénybe. A Top értékhez, ha szükséges hozzáadjuk a papír hosszának felét, a Rotation értékhez beírjuk a 45-öt. Amikor a beállításokkal készen vagyunk, visszatérünk a főmenühöz, és kiválasztjuk a letárolt grafikont, az Image Select opcióval, majd a Go paranccsal kinyomtatjuk.

4.5 Állománykezelő

4.5.1 Alapok (mezők, rekordok)

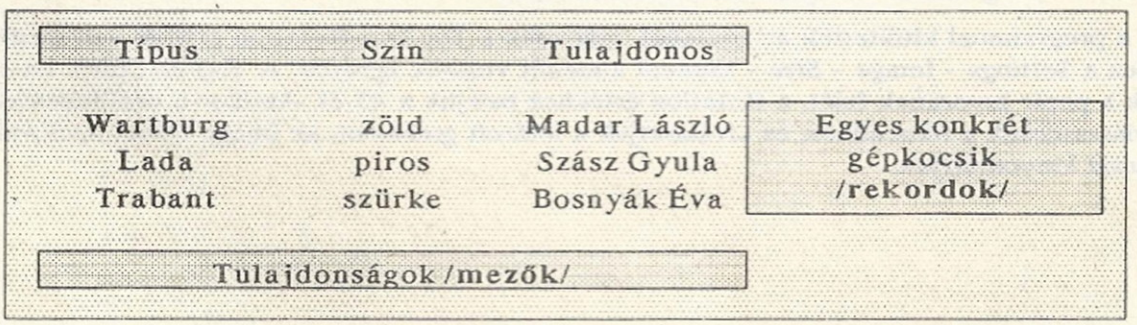
Amikor adatainkat a számítógépben tároljuk, rendszerezzük és különböző műveleteket végzünk velük, többnyire adatbázisokkal dolgozunk. Az adatbázis nem más, mint egy struktúra, amelyet az ember visz a szerkezettelen valóságba, a valóságos adatok feldolgozhatósága érdekében.

A struktúra legelemibb építőköve a kiválasztott adathalmaz egy konkrét elemének egy konkrét tulajdonsága. Egy ilyen tulajdonságot mezőnek nevezünk. (pl.: ha a kiválasztott adathalmaz a gépkocsikat jellemző adatok halmaza, akkor egy konkrét eleme lehet az utcán parkoló zöld Wartburg, ennek egy konkrét tulajdonsága a színe (zöld), egy másik konkrét tulajdonsága a típusa (Wartburg), stb...

Ezek a tulajdonságok lehetnek mezők egy adatállományban. Az előző példából kiderül, hogy az adathalmaz egy konkrét elemét alkotó mezők logikailag összetartoznak. Az összetartozó mezők együttesét rekordnak nevezzük. (Az előző példánál maradva, egy rekord az összes olyan tulajdonság, amelyik az utcán parkoló zöld wartburgot jellemzi. Színe, típusa, tulajdonosa, rendszáma, fogyasztása, forgalmi engedélye száma, stb...) Tehát a rekordok mezőkből állnak. Az adatállományok pedig rekordokból állnak, ugyanis a kiválasztott adathalmazunkat nevezzük adatállománynak, amennyiben az említett struktúrának megfelelően építjük fel. (Adatállomány tehát a gépkocsikat jellemző adatok halmaza) Az adatállományt legkönnyebben egy táblázatban tudjuk elképzelni, ahol a sorok az egyes elemeket (gépkocsikat) jelölik, az oszlopok pedig az azonos jellegű tulajdonságokat.

Ha egy adatállományt így felépítünk, már könnyebb a megfelelő elemet (gépkocsit) megtalálni benne,

Gépkocsik halmaza /adatállomány/



hiszen tudjuk, hogy hol kell keresnünk az ismert tulajdonságokat.

Például, ha Madar László kocsijának színét szeretnénk megtudni, nem egy rendezetlen halmazban nézünk át esetleg több ezer adatot, hanem a tulajdonos mezőket sorra megvizsgálva, amikor megtaláljuk a Madar László nevet, ugyanabban a rekordban megnézzük a szín mezőt és máris tudjuk, hogy a keresett szín a zöld.

4.5.2 Adatállomány készítése

A SYMPHONY program lehetőséget ad az ALAPOK fejezetben bemutatott struktúrájú adatmodell kezelésére. Mielőtt azonban elkezdenénk adatainkat felvinni az adatállományba, el kell készítenünk a megfelelő szerkezetet. Először is át kell gondolnunk, hogy milyen adatokra és milyen tulajdonságokra lesz szükségünk. Készítsünk példaként egy adatállományt az éves jövedelmekről, hogy év végén amikor az adóbevallást írjuk, ne felejtsünk ki semmilyen bevételt.

Milyen mezőkre lesz szükség ? Szükségünk lehet arra, hogy tudjuk melyik cégtől, vagy kitől kaptuk a pénzt. Azután, hogy mennyit kaptunk, ebből mennyit fizettünk be adóelőleget, feljegyezhetjük azt is, hogy mikor kaptuk.

Miután kigondoltuk a rekordjaink szerkezetét, készítsük fel a SYMPHONY-t az adatok kezelésére. Először táblázatkezelő módban egy sorban, az egymást követő cellákban nevezzük meg mezőinket. Amikor a neveket megadjuk, még ugyanabban a cellában egy kettőspont után megadhatjuk a mező típusát (milyen

típusú értékek lesznek benne), egy újabb kettőspont után a kívánt hosszát karakterekben számolva. A mezőtípusok megadására egy karaktert használhatunk, amelynek értékei a következők lehetnek

L	szöveges mező (**Literal**)
N	számjegymező (**Numeric**)
D	dátum mező (**Date**)
T	időpont mező (**Time**)
C	számított érték mező (**Computed**) A számított mezőbe adatot bevinni nem lehet. Az ilyen mezőt a SYMPHONY tölti majd ki, az általunk megadott számítási formula alapján.

Legyen az A1 cella tartalma KITOL:L:40, az A2 cella BRUTTO, az A3 cella pedig ELOLEG. A többi mezőt majd az adatbázis készítésekor fogjuk meghatározni.

Az adatállomány-kezelő funkció az **Üzemmód** menü (**Alt+F10**) **FORM** parancsával érhető el. Amíg az adatállomány készítését nem végeztük el, képernyő első sorában a "No Definition range defined" üzenet figyelmeztet, hogy az adatállomány-kezelőhöz szükséges tartományok nincsenek definiálva. A tartományok definiálását az **ADATBÁZIS** menü (**FORM** üzemmód és **F10**) Generate menüpontjával végeztethetjük el.

Miután kiválasztjuk a **Generate** pontot, a program megkérdezi az alapértelmezett mezőtípust és hosszt. Az itt megadott jellemzőket azoknál a mezőknél használja, ahol a felsorolásban nem adtuk meg.

Esetünkben két ilyen mező van, a BRUTTO és az ADOALAP. Ezeknek a típusa **Numeric**, a hossza **15** karakter legyen, tehát a kérdésekre ennek megfelelő válaszokat adjunk.

Amikor megadtuk a típust és hosszt, meg kell mutatnunk azt az egysoros tartományt, ahol a program a mezőneveket találja. Ilyenkor a munkaterület ideiglenesen táblázat üzemmódra vált.

A TÁBLÁZATKEZELŐ fejezetben ismertetett tartomány kijelölést használva adjuk meg az A1..A3 tartományt.

Ezután a SYMPHONY automatikusan elkészíti azokat a szükséges tartományokat, amelyeket az adatállomány-kezelő használ (részletesebben lásd az ADATÁLLOMÁNY-KEZELŐ ÉS TÁBLÁZATKEZELŐ fejezetben). A munkaterületen az előre megadott mezőket látjuk egymás alatti sorokban, és minden mező neve után annyi "_"karaktert, amilyen hosszúra definiáltuk. Mezőinkbe értékeket ezeknek a karaktereknek a felülírásával vihetünk be.

Először rendezzük el a képernyőn a már meglévő mezőinket. Az Adatállomány-kezelő menü **Field - Move** pontjaival mozgassuk mezőinket a kívánt helyre. Először a mozgatandó mezőt kell kijelölni a kurzormozgató billentyűk segítségével, majd az ENTER lenyomása után a mező új helyére visszük a kurzort, és ismét lenyomjuk az ENTER-t.

A többi mező megadásához használjuk a **Field - Insert** menüpontot. Miután megadtuk az új mező nevét, már ismerős kérdéseket kapunk. A kérdésekre csakúgy mint az adatbázis készítésekor, egy típust és egy hosszt kell megadnunk, majd el kell helyeznünk a munkaterületen az új mezőt. A program megkérdezi, hogy biztos szükségünk van-e az új mezőre, mert az adatbázis számára lefoglalt tartományok kiterjedése meg fog növekedni a munkatáblázatban. Válaszoljunk **Yes**-szel !

Az utolsó helyre még beillesztettünk egy megjegyzés mezőt, sose lehet tudni ki mit szeretne még feljegyezni egy bevételről.

Ezzel adatállományunk lényegében készen van. További lehetőségeket az ADATÁLLOMÁNY ÉS MÁS ÜZEMMÓDOK fejezetben olvashatunk.

4.5.3 Rekord felvitele és módosítása

Amikor elkészítettük a beviteli képernyőt, a kurzor az első mező első karakterére áll, és várja a mező értékének begépelését. Minden Mezőadat bevitelét az **ENTER** billentyűvel zárjuk le !!! Ekkor a kurzor a sorrendben (balról-jobbra és lefelé) a következő mezőre ugrik. Ha csak egy mezőt is kitöltöttünk, és a bevitelt ENTER-rel lezártuk, az adatállomány-kezelő létrehoz egy rekordot, és az adatállományunk végéhez fűzi. Vagyis, az adatállomány-kezelő nem ellenőrzi, hogy minden mező ki van-e töltve, felvihetünk hiányos rekordokat is.

A már felvitt rekordok között a **PgUp** billentyűvel az első rekord felé, a **PgDn** billentyűvel az utolsó rekord felé lapozhatunk. Amikor a **PgDn** billentyűt használjuk és elérjük az adatállomány végét, egy üres "kitöltetlen" rekordképet kapunk a munkaterületen, és a képernyő első sorában a "**NewRecord**" felirat jelenik meg. Ez az újabb rekord felvitelét jelzi. Amikor folyamatosan kívánunk több rekordot egymás után felvinni, a mezők kitöltése után használjuk az **Ins** billentyűt, hogy egy új rekordot hozzáfűzzünk az adatokhoz, és egy újabb üres rekordképet kapjunk kitöltésre.

Ha meggondoljuk magunkat, és egy felvitt rekordot törölni szeretnénk, használjuk a **Del** billentyűt. Miután az adatállomány-kezelő megkérdezi, hogy biztosak vagyunk-e a dolgunkban (**Are you sure you want to delete this record ?**), a rekordot kiveszi az adatállományból.

Egy létező rekordot úgy módosíthatunk, hogy miután a **PgUp** vagy **PgDn** billentyűkkel megjelenítettük a képernyőn, a változtatni kívánt mezőre állunk, és elkezdjük gépelni a jó adatot. Ilyenkor a módmutató a képernyő jobb-felső sarkában "**EDIT**" felíratra vált. Ha elrontottuk az új adat begépelését, de még nem hagytuk el a mezőt az **ENTER** billentyűvel, nyomjuk le egyszer az **Esc** billentyűt. A mező újból üres lesz, és a kurzor a mező elejére áll. Ha kiderül, hogy mégis az eredeti érték volt jó a mezőben, nyomjuk meg mégegyszer az **Esc** billentyűt, és a mezőbe visszakerül a javítás előtti adat.

Másik módja a rekord javításának, hogy mielőtt elkezdjük gépelni a jó adatot, lenyomjuk az **F2** billentyűt a megfelelő mezőn állva. A módmutató most is **EDIT**-re áll, de a mezőből nem tűnik el az adat, elég csak azt az egy betűt felülírni, amelyik hibás volt.

4.5.4 Keresés az adatállományban

Az adatállomány-kezelőben külön kereső parancs nincs. Rekordjainkat egy vagy több különleges célra használt, félig vagy teljes egészében kitöltött rekord segítségével találhatjuk meg. Ezeket a segéd rekordokat kritériumrekord-oknak nevezzük, mezői megegyeznek az adatrekordok mezőivel. Kezelésüket az Adatállomány-kezelő menü **Criteria** menüpontjával végezhetjük el.

Kritérium-rekordokat készíthetünk az **Edit** paranccsal, ilyenkor az adatrekordok felvitelének megfelelő módon járhatunk el. A kritériumrekordok készítését az **Esc** billentyűvel, vagy a **PgUp** billentyű többszöri lenyomásával hagyhatjuk el. Az elkészült kritériumok az adatállománynak nem lesznek részei, viszont ha előírjuk a programnak használatukat, akkor az adatállomány-kezelő minden jelentésben (a képernyőn való kijelzést is beleértve) csak azokat az adatrekordokat fogja megjeleníteni, amelyek legalább az egyik alkalmazott kritériumnak megfelelnek. A megfelelést úgy dönti el, hogy a kritérium rekordban kitöltött mező értékének, és az adatrekord azonos mezőjének értéke meg kell hogy egyezzen. A kritériumok használatát a **Criteria - Use** paranccsal írhatjuk elő. Ha újra szeretnénk megjeleníteni az összes létező rekordot a jelentésekben, használjuk a **Criteria - Ignore** parancsot.

Ha az elkészítendő jelentésben csak az első negyedévi bevételeinket szeretnénk látni, a **Criteria - Create** paranccsal készítsünk három kritérium-rekordot. A kritériumok készítésének módja megegyezik az adatrekordok felvitelével. Az első kritériumrekordban a hónap helyére írjuk a januárt, a másodikba a februárt, a harmadikba a márciust. Ezután adjuk ki a **Criteria - Use** parancsot, és készítsük el a jelentést.

4.5.5 Rendezés

A SYMPHONY adatállomány-kezelőjében az adatrekordok rendezése fizikai rendezést jelent. Vagyis, ha kiadjuk a rendező parancsot (**Adatállomány-kezelő** menü **RecordSort**), a program az adatrekordokat a munkaterületen átrendezi úgy, hogy a rekordokat tartalmazó tartomány celláinak tartalma a sorrendnek megfelelően megváltozik (lásd még az ADATÁLLOMÁNY–KEZELŐ ÉS TÁBLÁZATKEZELŐ részt).

Ahhoz, hogy a rendezési szempontokat megadhassuk, az adatállomány-kezelő beállítótábláját kell igénybe vennünk. A beállítótáblában három rendezési kulcsot adhatunk meg (**1-st key, 2-nd key, 3-rd key**). Ezek a kulcsok egy-egy mezőt jelentenek, és számozási sorrendben vesznek részt a rendezéskor annak eldöntésében, hogy egy adott rekord egy másik rekord elé kerüljön, vagy mögé. Mindhárom kulcsnál különkülön meghatározhatjuk, hogy növekvő (**Ascending**) vagy csökkenő (**Descending**) sorrendben vegye figyelembe a program.

Ha adatainkat rendezni szeretnénk év, és azon belül hónap szerint, vegyük elő az adatállomány-kezelő beállítótábláját (**F10 - Settings**), közöljük a programmal, hogy rendezési kulcsot kívánunk megadni (**Sort-Keys**), adjuk meg elsődleges kulcsként az évet tartalmazó mezőt (**1-st key**), másodlagos kulcsként a hónap

mezőt (**2-nd key**). Mindkét kulcsmező legyen növekvő sorrendű (**Ascending**). Ezután adjuk ki a **Record Sort** parancsot.

FIGYELEM !!!

Mivel a rekordok rendezése fizikai rendezés, ha sok rekordunk van, néhány percig is eltarthat ! Ha ezek után a **PgUp** vagy a **PgDn** gombokkal lapozgatunk az adatrekordok között, a rekordokat már rendezve láthatjuk a képernyőn.

4.5.6 Jelentések készítése

Adatainkról jelentéseket is készíthetünk, a **Szolgáltatások** menüjének **Print** almenüje segítségével. A jelentésben azok az adatrekordok lesznek, amelyek megfelelnek a kritériumoknak, függetlenül attól, hogy a kritériumfigyelés be van-e kapcsolva (**Criteria - Use vagy Ignore**). Ha a jelentésben a rekordokat valamilyen sorrend szerint szeretnénk látni, előbb használjuk a rendező parancsokat (lásd a RENDEZÉS fejezetet).

Először is meg kell adnunk, hogy adatállományról kívánunk nyomtatni. Ehhez használjuk a **Print** almenü **Settings - Source** menüpontját, ahol az adatbázist (**Database**) fogjuk kijelölni forrásként. Ezután a képernyő második sorában egy menüben választhatjuk ki azt az adatállományt, amelynek rekordjait szeretnénk nyomtatni. (A SYMPHONY-ban annyi adatállományt kezelhetünk, amennyit a memória és a munkatáblázat megenged.)

Mielőtt nyomtatni kezdenénk, ellenőrizzük a következő beállításokat. A **Print** almenü beállítótáblájában ki kell kapcsolni a laptördelést (**Settings - Page - Break - No**).

A nyomtatási formátumot képernyőn látható módra kell állítani, mert különben csak az adatállomány tartományába eső cellák címeit és azok értékét kapjuk jelentésként (**Settings - Other - Format - As-Displayed**).

A beállítások elvégzése után, a jelentés első sorában a mezők nevei lesznek, a második sortól kezdve a rekordok kerülnek, rendre egymás alatt, egy rekord egy sorba.

KITOL	BRUTTO	ELOLEG	EV	HONAP	MEGJEGYZÉS
1. CÉG	12.000	2.350	1991	jan	xxxxxxxxxxxx
2. CÉG	12.000	2.380	1991	mar	yyyyyyyyyyyyy
3. CÉG	8.000	1.870	1991	feb	zzzzzzzzz
4. CÉG	4.000	670	1991	jan	vvvvvvvvv

4.5.7 Adatállomány és más üzemmódok

Az adatállomány-kezelő további lehetőségeit csak akkor tudjuk kihasználni, ha alapfokon tisztában vagyunk a táblázatkezelő és a szövegszerkesztő kezelésével.

A szövegszerkesztőt akkor hívhatjuk segítségül, ha a beviteli képernyőt szeretnénk megváltoztatni. Váltsunk át **DOC** módba ! Könnyen megtalálhatjuk a munkatáblázatban azt a területet, amit a rekordok beolvasásához használ a program (Itt vannak az _ karakterek). Ezen a részen a szövegszerkesztővel beírhatunk bármilyen információt, új sorokat is beszúrhatunk, sőt még a beolvasandó mezők elé írt mezőneveket is megváltoztathatjuk. A beolvasási területet jelképező karaktereket is elmozdíthatjuk.

FIGYELEM !!!

Két dologra kell vigyáznunk. Nem szabad a beolvasást jelölő karakterek számát megváltoztatni (törölni vagy újat beírni), és az összetartozó _ karakterek közé beszúrni más karaktert. A másik nagyon fontos szabály, hogy az utolsó beolvasandó mezőnek is el kell férni az adatállomány-kezelő számára definiált ablakban (Teljes méretű ablaknál 20 sorban).

A táblázatkezelővel kihasználható lehetőségek előtt tisztában kell lennünk azzal, hogy az adatállomány-kezelő milyen tartományokat definiál az adatállomány számára. Az első tartománnyal az adatállomány készítésekor megismerkedtünk. Itt vannak a rekorok mezőinek nevei, típusai és hosszai. Alatta van a beviteli

képernyőképet leíró tartomány. Ezt módosítottuk a szövegszerkesztővel (Módosíthatjuk táblázatkezelő üzemmódban is, de szövegként egyszerűbb).

A beviteli képernyő alatt egy leíró tartomány van, amelyben mindig az utoljára bevitt, vagy épp bevitel alatt álló rekord mezőit vizsgálhatjuk, alakíthatjuk. E tartomány cellái folyamatosan változnak, ahogy a bevitel halad.

A tartomány egy kis táblázat. Sorai egy-egy mező adatait tartalmazzák, oszlopai a következők :

Name A mezők nevei.

Value Az ellenőrzött, átalakított mezőértékek kerülnek ebbe az oszlopba. A rekord bevitele után ebből az oszlopból kerülnek át a mezők az adatállományba.

Type A mezők típusa és hossza egy kettősponttal elválasztva.

Default A mező alapértelmezett értéke. Ide írhatjuk azokat a mezőértékeket, amelyek a bevitel során többnyire változatlanok.

A jövedelem adatbázisnál az EV mező Default oszlopába írjuk be szöveges értékként '1991. Minden újabb rekord felvitelénél az EV mező értéke 1991 lesz, ami azonban megváltoztatható.

Formula Ebben az oszlopban előírhatjuk, hogy a beolvasott és ellenőrzött mezőt milyen módon alakítsa át a program, mielőtt a Value oszlopban elhelyezné. Ha a HONAP mező adatai a 15. sorban vannak és az Input oszlop a G, akkor a HONAP mező Formula oszlopába írjuk be @UPPER(G15). Ennek hatására a HONAP mezőbe írt adatot a mezőbevitel végén az adatállomány-kezelő átalakítja nagybetűssé.

Validity Az ellenőrzés módját írhatjuk elő ebben az oszlopban. Használhatunk kifejezéseket, vagy függvényeket. Ha korlátozni szeretnénk a bruttó összeget 0-tól 200.000Ft-ig tartó határok közé, az **Input** oszlop a **G** és a **BRUTTO** mező a 12. sorban van, írjuk be a **BRUTTO** mező **Validity** oszlopába **0<=G12<=200000** A program csak a 0 és 200.000 közé eső értékeket engedi majd bevinni.

Input Amikor egy mezőt beviszünk, ebben az oszlopban helyezi el a program. Tehát az ellenőrzésnél előírt kifejezésekben erre az oszlopra kell hivatkoznunk. Ha a mezőérték az ellenőrzés után megfelelőnek bizonyul, innen kerül át a **Value** oszlopba a **Validity** átalakításoknak megfelelően.

Prompt Minden mezőhöz előírhatunk egy max. 80 karakteres üzenetet, amit a program akkor ír ki a képernyő legfelső sorába, ha az adott mezőn áll a kurzor.

Segítő információként írjuk be a HONAP mező Prompt oszlopába "A hónapot kérem betűvel kiírni " szöveget. Az adatállomány-kezelő minden esetben ezt fogja a képernyő első sorába kiírni amikor a HONAP mezőn áll a kurzor.

Az előbbiekben leírt oszlopokból tehát a **Default**, a **Formula**, a **Validity** és a **Prompt** oszlopok celláit írhatjuk át "büntetlenül".

A mezőleíró tartomány alatt a riport tartomány található. Ebben a tartományban is érdemes néhány dolgot megváltoztatni. Amikor az adatállományt elkészítjük ez a tartomány két résztartományra oszlik. Az első a rekordok nyomtatása előtt kinyomtatandó szövegeket tartalmazza, ami nem más mint a mezők nevei. A második résztartomány a rekordok mezőire hivatkozik, oly módon hogy a kinyomtatandó helyen egy + jellel kezdve egy-egy mezőnév van. Amikor a jelentést nyomtatja, ezekre a helyekre rendre behelyettesíti a kritériumoknak megfelelő rekordok mezőit, és azokat nyomtatja ki.

Amikor létrehoztuk az adatállományunkat a következő jelentés-formátumot definiálta nekünk az adatállomány-kezelő :

```
KITOL,  BRUTTO    ELOLEG EV  HONAP   MEGJEGY
+KITOL  +BRUTTO   +ELOLEG    +EV +HONAP  +MEGJEGY
```

Van még egy résztartomány amit megadhatunk, ez azonban az adatállomány készítésekor nem jön létre automatikusan. Ez a tartomány a jelentés után kinyomtatandó szövegrészeket tartalmazza. A három tartomány adatait táblázatos üzemmódban módosíthatjuk, de ha a tartományok határain kívül is helyeztünk el adatot, állítsuk be a méreteket az Adatállomány-kezelő menü (**FORM** üzemmód és **F10**) **Settings - Report** parancsával. Három tartományt adhatunk itt meg, a jelentés elején nyomtatandó (**Abowe**), a jelentés törzsét képező (**Main**) és a jelentés után nyomtatandó (**Below**) tartományokat.

4.6 A SYMPHONY programozása

A rendszer nem rendelkezik olyan fejlett programnyelvvel mint a FRAMEWORK, de néhány alapvető feladat ellátására a SYMPHONY-ban is készíthetünk programokat. Ezek a programok nem is igazi programok, mert csak arra valók, hogy bizonyos sűrűn ismétlendő feladatot végrehajtsanak úgy, mintha a felhasználó kezelné a gépet. Ezért nem is programoknak, hanem **makróknak** nevezzük ezeket. A későbbiekben látni fogjuk, hogyan lehet a makrók segítségével egy-egy feladatot egyetlen gomb lenyomásával elvégeztetni.

4.6.1 Makrók készítése

A makrókat is, mint minden adatot a rendszerben, a munkatáblázatban (**WORKSHEET**) kell elhelyeznünk. Ezért mielőtt a programozásba kezdenénk, meg kell terveznünk az elkészítendő program menetét, az általa használt adatok elhelyezkedését. Ez azért különösen fontos, mert a makró futása során létrejövő adatok esetleges rossz elhelyezése felülírhatja a makrót, ami attól kezdve működésképtelen lesz.

Egyszerre több makrót is tárolhatunk a munkatáblázatban, akár adatokkal együtt is.

Egy-egy makró hasonlóan más programnyelvekhez, utasításokból áll. A SYMPHONY belső értelmezője ezeket az utasításokat fogja végrehajtani az olvasás szabályainak megfelelően, vagyis jobbról balra és föntről lefelé. Az utasításokat ugyanolyan módon vihetjük be, ahogy azt a TÁBLÁZATKEZELŐ fejezetben olvashattuk.

Minden makróutasítás szöveges információként tárolódik a cellákban, és mivel a szöveges információt a jobbra eső cella tartalma "lefedi", érdemes az utasításokat egymás alatti cellákban elhelyezni. Egy cellába egymás után több utasítást is írhatunk.

Az olvashatóságra gondolva a SYMPHONY készítői úgy határozták meg, hogy egy makró addig tart, míg egy üres cellákat tartalmazó sort nem talál az értelmező. Másik lehetőség a makró végének jelzésére, a **QUIT** utasítás.

Egy makró attól válik makróvá, hogy a számítógépünk vezérlését át tudjuk adni az értelmezőnek, és rendre végrehajtatjuk utasításainkat. Ehhez a makró első utasítását tartalmazó cellát el kell nevezni mint tartományt (**Táblázatkezelő menü - Range - Name - Create**). Ha a makrót egyetlen gombnyomással szeretnénk indítani, a névnek egy \ jelből és egy betűből kell állnia. Ekkor az **Alt** és a megadott betű együttes lenyomásával indul a makró végrehajtása. Ha a \ jel után egy számjegyet írunk **0-tól 9-ig**, a makró az **F7** billentyű lenyomása után a megfelelő funkcióbillentyű lenyomásával (**F1 - F10**) indul. Ha nem \ jellel kezdjük a makró nevét, akkor az indítást az F7 billentyű lenyomása után begépelt makrónévvel kezdeményezhetjük. (F7-re megjelenik a munkaterület alatt a "User" felirat, ilyenkor vár a program egy **makrónév** begépelésére, vagy egy funkcióbillentyű lenyomására.)

4.6.2 Makróutasítások szintaktikája

A SYMPHONY-ban minden makróutasítást a { } jelek közé kell tenni. Tehát a már ismert makró lezáró utasítás helyesen : **{QUIT}**

Azokban az utasításokban, ahol adatokat kell átadnunk az értelmezőnek, az adatok elválasztására a Szolgáltatás menü **Configuration - Other - International - Punctuation** almenüjében beállított **Argument separator** (paraméter elválasztó) karaktert kell használnunk.

Makróinkban használhatunk előre meghatározott billentyűneveket, amelyeket a { } jelek közé írva a végrehajtás során a SYMPHONY úgy értelmez, mintha a billentyűzeten lenyomták volna az adott gombot.

A gombok nevei mellé, egy számot is írhatunk egy szóköz karakter után, ilyenkor úgy veszi, mintha azt a gombot annyiszor egymás után lenyomtuk volna. Ezt a számot ismétlési faktornak nevezzük.

Pl : **{Right 2}**

Ez az utasítás a jobbra mutató kurzormozgató billentyű kétszeri lenyomását szimulálja.

Ha nem vezérlőbillentyű leütését kívánjuk szimulálni, akkor a megfelelő karaktert kell csak leírnunk a { } jelek nélkül. Az **ENTER** billentyűt a ~ karakterrel jelölhetjük.

Pl : **proba**
Ez az utasítás a makróba írva a 'proba' szó begépelését szimulálja.

A makróutasítások és a billentyűnevek listája a FÜGGELÉK-ben olvasható. Lehetőség van a makrókból újabb makrókat indítani, a makrók első utasítását tartalmazó tartomány nevének { } jelek közötti leírásával. A {MYMACRO} utasítás a MYMACRO nevű tartomány első utasítására adja a vezérlést. Visszatérni a hívó makró következő utasításához a {RETURN} utasítással lehet.

Ez esetben a MYMACRO mint egy szubrutin használható.

4.6.3 Makrók automatikus indítása

A SYMPHONY-ban lehetőségünk van arra, hogy az elkészített makrót az azt tartalmazó munka betöltése után azonnal elindítsuk, minden külön gombnyomás nélkül. Ehhez természetesen el kell menteni az adott munkát a **Szolgáltatás** menü **File - Save** almenüjével, előbb azonban az **Auto-Execute - Set** menüpontjánál meg kell adnunk az elindítandó makró nevét. Ezután a munka betöltésekor a makró automatikusan indul.

Ha azt szeretnénk hogy a makró mégse fusson le a munka betöltésekor, használjuk a Szolgáltatás menü **Settings-Auto-Execute-Cancel** menüpontját.

Ha azt szereténk, hogy a programunk kezelőjének még a munkatáblázatot se kelljen lemezről betölteni, a Szolgáltatás menü **Configuration - Auto** menüpontjánál megadhajuk az elmentett munka nevét. Ezután a **Configuration - Update** menüpontot lefuttatva az előírt beállítást el kell mentenünk, különben a program végén elvesznek a megadott értékek.

Ha a fentiekben leírt változtatásokat elvégeztük és elmentettük, a SYMPHONY következő betöltésekor a megnevezett munka automatikusan betöltődik, és a makró elindul.

4.6.4 "ÖNTANÍTÓ " üzemmód

Ha az elkészítendő makró csak gombnyomások sorozata, használjuk ki a SYMPHONY **tanuló** üzemmódját, amikor a program saját magának lejegyzi mint egy makrót, az általunk lenyomott billentyűket.

Ehhez először is meg kell jelölnünk azt a tartományt, ahová az elkészítendő makrót írja (Szolgáltatás menü **Settings - Learn - Range**). Amikor a program a makrót "tanulja" csak üres területre ír, a cellatartalmakat nem törli. Ha nem üres cellát talál, a tanulást az utána következő sorban kezdi. Ahhoz hogy a tanuló terület biztosan üres legyen, használjuk a **Learn** almenü **Erase** pontját. Ha elérjük a kijelölt tartomány utolsó sorát is a tanulás a **"Learn range is full"** üzenettel leáll. A tanulási terület kijelölését a **Learn** almenü **Cancel** pontjával lehet megszüntetni. A **Learn** almenü **Yes** vagy **No** pontjával is be ill. ki lehet kapcsolni az öntanító módot, de az **Alt+F5** billentyűvel ez kényelmesebb.

4.6.5 Makró–könyvtár kezelő

Amikor egyszerre több makrót is írunk, és azokat különböző névvel látjuk el, tulajdonképpen **makró-könyvtárat** készítünk. Egy-egy ilyen makró-könyvtárban célszerű elhelyezni a logikailag összetartozó makrókat. Ezeket a makrókat később együtt tudjuk majd memóriába tölteni, vagy törölni onnan.

A SYMPHONY egyes lehetőségeit nem építették be közvetlenül a programba. Ezeket betöltve a memóriába, újabb funkciók ellátására szolgáló almenüket kaphatunk. Ezeket az alprogramokat **alkalmazás**oknak (**Application**) nevezzük. Egyik ilyen alkalmazás a **Makró-könyvtár kezelő**. Arra szolgál, hogy az elkészített makró-könyvtárunkat ne kelljen a munkatáblázatban tartani, mégis futtathatók legyenek a könyvtárat alkotó makrók.

Kezeléséhez a **Szolgáltatás** menü **Application** almenüjét használjuk. Az almenü felépítése :

Attach A memóriába tölti a kiválasztott alkalmazást (A makró-könyvtár kezelő neve MACROMRG.APP).

Invoke A memóriába töltött alkalmazások egyikének aktivizálása (futtatása).

Detach Egy betöltött alkalmazás törlése a memóriából.

Clear Az összes betöltött alkalmazás törlése a memóriából.

Amikor egy betöltött alkalmazást futtatni szeretnénk, nem kell minden esetben az **Invoke** parancsot kiadni. Vannak olyan alkalmazások, amelyek a SYMPHONY egyik menüjébe épülnek be, futtatásukhoz a megfelelő menüpontot kell kiválasztani. A makró-könyvtár kezelő is ilyen. Amikor az **Attach** paranccsal betöltjük, a **Szolgáltatás** menüben utolsó menüpontként megjelenik a "**Macros**" szó. Ennek kiválasztásával aktivizálhatjuk.

A makró-könyvtár kezelő menüje

Amikor a **Macros** szót kiválasztjuk, a következő menüt kapjuk :

Load Makró-könyvtár betöltése a memóriába.

Save Egy elkészített makró-könyvtár elmentése lemezre és a memóriába. Ilyenkor a megírt makrók a munkatáblázatból törlődnek.

Edit A memóriában lévő makró-könyvtár betöltése a munkatáblázatba javítás, vagy továbbírás céljára.

Remove A makró-könyvtár törlése a memóriából.

Name-List A makrókönyvtárban használt tartománynevek listájának betöltése a munkatáblázat kijelölt helyére.

4.6.6 Vonalazás makróval

Először nézzünk egy egyszerű billenyűleütési sorozatot szimuláló makrót. A makró feladata, hogy az aktuális cella sorában a munkatáblázat teljes szélességében húzzon egy dupla vonalat. Először fel kell tölteni az aktuális cellát = karakterekkel. Ezután a cellát lemásoljuk az aktuális ablak jobb széléig, majd a bal széléig.

{WINDOWSOFF} Kikapcsoljuk a képernyőfrissítést, hogy ne villogjon. '\=~'Az aposztrof után álló két karakter lenyomását imitáljuk. Az utasítás végén álló ~ karakterrel jelöljük az ENTER billentyűt. Ezzel feltöltöttük = jellel az aktuális cellát.

{MENU} Lehívjuk a Táblázatkezelő menüt.

C A C billentyűvel kiválasztjuk a Copy parancsot.

~{RIGHT}{TAB}{END}{RIGHT}~ Ezzel a billentyűsorozattal lemásoljuk az aktuális cella tartalmát jobbra az ablak széléig. Az aktuális cellakijelölés nem mozdul el.

{MENU}C újra lehívjuk a Táblázatkezelő menüt.

~{LEFT}{TAB}{END}{LEFT}~ Lemásoljuk balra is.

{WINDOWSON} Engedélyezzük az ablakkirajzolást. (Az adatok frissítésre kerülnek a képernyőn.)

Ez a makró feltételezi, hogy az aktuális üzemmód a táblázat üzemmód (**SHEET**). Nem vizsgálja azt sem, hogy az aktuális cellától balra, vagy jobbra vannak-e cellák, szükség van-e a vonal balra ill. jobbra történő másolására. Ezért, ha az aktuális ablak egyik szélén állunk és ott indítjuk el a makrót, hibajelző sípolást fogunk hallani, de ez a makró futását nem befolyásolja. Az előző példából tehát a következőket láthattuk:

Minden billentyű lenyomását imitálhatjuk a karakter utasításként való leírásával, { ~ } jelek nélkül. Egyetlen kivétel van ez alól, a ~ kartakter, ami az ENTER billentyű lenyomását jelzi. A "vezérlő" billentyűk lenyomását, a { } jelek között leírt, a program által használt billentyűnévvel jelezhetjük.

4.6.7 Változók használata

A SYMPHONY-ban változóként egy-egy cella használható. A cellába írhatunk mi is adatot, de akár a cellában elhelyezett függvény értékét is felhasználhatjuk.

Ha egy makróban arra van szükség, hogy az aktuális cellakijelölést megváltoztassuk, de a makró végén vissztérjünk az eredetihez, a következőket kell megtennünk. Nevezzünk el egy mezőt mint tartományt a

Táblázatkezelő menü **Range - Name - Create** pontjával **AKTPOS**-ra. Erre akkor lesz szükség, amikor a kész makrót kimentjük, majd egy későbbi alkalommal visszatöltjük. Előfordulhat, hogy a makrót másik területre töltjük vissza. Ilyen esetben a tartomány nevére való hivatkozás is automatikusan áthelyeződik, míg ha egy fix cellára hivatkozunk, felülírhatunk egy fontos adatot.

Következő lépésként ebbe a cellába írjuk be a **@CELLPOINTER("ADDRESS")** függvényt, ami az aktuális cella címét adja. Azért, hogy az **AKTPOS** változó aktualizált legyen, kezdjük a makrót a **{CALC}** paranccsal. A makró végén az utolsó két utasítás a **{GOTO}** és a **{BRANCH AKTPOS}** legyen. A **{GOTO}** utasításra, mintha az **F5** billenyűt lenyomtuk volna, a legfelső sorban megjelenik a kérdés

Goto where : _

Ezután a **{BRANCH AKTPOS}** utasítás hatására, a vezérlés átadódik az aktpos cellára, amelynek tartalma a makró előtti aktuális cella címe. Amikor a program ezt a címet mint utasítást végrahajtja, úgy tesz, mintha a billentyűzetről begépeltük volna a címet. Végül már csak egy **ENTER** billentyűre van szükségünk, és a makró befejezését jelző {Quit} utasításra. Vagyis, az **AKTPOS** változóként viselkedő cella alatti cellában a **~{QUIT}** utasításokat helyezzük el. A fenti példa azt is megmutatta, hogy egy célszerűen elhelyezett változót akár utasításként végre is hajtathatunk. (Természetesen ha a változót a makróban helyezzük el, nincs szükség a **{BRANCH AKTPOS}** utasításra.)

{CALC} Az AKTPOS aktualizálása . .

.

. A makró utasításai

.

{GOTO} F5 billentyű imitálása

@CELLPOINTER Ez az AKTPOS nevű cella

~{QUIT} Makró vége

4.6.8 Egy interaktív makró

Készítsünk makrót, amelyik bekér egy számot, majd a szám négyzetét kiírja a munkaterületre (interaktív makró).

Először is szükségünk lesz egy **X** nevű változóra, amit képviseljen most, a **B10** cella (**Menü - Range - Name - Create :** X - Enter - B10 - Enter)

{PANELOFF}

{GOTO}A1~ A munkaterületen az A1 cella legyen a bal felső arokban, hogy majd láthassuk a végeredményt.

{BLANK B10..D10} Töröljük a B10..D10 tartományt, ide írjuk majd az eredményt.

{WINDOWSOFF} Kikapcsoljuk az ablak frissítését, hogy ne villogjon munka közben.

{INDICATE NÉGYZET} A módmutató helyére kiírjuk a "NÉGYZET" szót.

{GETNUMBER "Kérem a számot :";X} A szám beolvasása az X-el jelölt tartományba (B10 cella).

{IF @ISERR(X)}{BRANCH HIBAPROC} Ha a beolvasott érték nem szám, meghívjuk a hibakezelő procedurát, majd újra ellenőrizük a bevitt adatot.

{GOTO}C10~ B10 cellában látható a beolvasott érték, C10 cellába írjuk a NÉGYZETE~ szót.

{GOTO}D10~+X*X~ D10 cellába írjuk az X négyzetét előállító képletet.

{GOTO}A1~ A1 cellára állunk, hogy majd ha bekapcsoljuk az ablak frissítését, ne villogjon.

{INDICATE} Módmutató visszaállítása.

{WINDOWSON} Ablakfrissítés visszakapcsolása. Az eredmény és a felírat megjelenik a munkaterületen.

{PANELON}

{QUIT} Makró vége.

{BEEP 0} Hangjelzés.

{GETNUMBER "Számot kértem ! Próbálja újra! =>";X} Szám újraolvasása.

{BRANCH JOSZAM} Viszatérés a főágba.

A makróban két címkét használtunk. A címke is mint a makró kezdetét jelölő makrónév, tulajdonképp a **Menü - Range - Name** almenüjében létrehozott tartománynév. A tartomány pedig egyetlen cellából áll, abból amelyre a címke mutat. Egyik címke a **HIBAPROC**. Ez a címke a **{QUIT}** utáni **{BEEP 0}** sort jelöli. Itt kezdődik a hibakezelő procedúra. Másik címke a **JOSZAM** azt a sort jelöli, ahol ellenőrizzük X helyességét. Vagyis, a hiba procedúra miután beolvasott egy új számot, az **{IF @ISERR...}** sorra tér vissza.

4.7 Függelék a 4. fejezethez

4.7.1 Funkcióbillentyűk

Funkcióbillentyűk szerepe az üzemmódokban

Gomb		SH	DOC	GR	FRM	COMM	
F1	[HELP]	X	X	X	X	X	Segítség (magyarázat) kérése
F2	[EDIT]	X			X		Mező módosítása
F3	[ABS]	X					Abszolút cellahivatkozás
F4	[CAPTURE]					X	Céleszköz kijelölése
F5	[GOTO]	X	X		X		Ugrás
F6	[WINDOW]	X	X	X	X	X	Következő ablak
F7	[USER]	X	X	X	X	X	Makró indítása
F8	[CALC]	X					Újraszámítás
F9	[SERVICES]	X	X	X	X	X	Szolgáltatások menü
F10	[MENU]	X	X	X	X	X	Üzemmódok menük

ALT+ . . . billentyűk szerepe

ALT+funkció-gomb		SH	DOC	GR	FRM	COMM	
F1	[COMPOSE]		X			X	Két karaktert egymásra ír
F2	[WHERE]		X				Szövegrész keresése
F3	[SPLIT]		X				Új sor
F4	[CENTER]		X				Sor közepére igazítás
F5	[LEARN]	X	X	X	X		Öntanító üzemmód be/ki
F6	[ZOOM]	X	X	X	X	X	Ablak növelése
F7	[STEP]	X	X	X	X	X	Makró lépésenkéti futtatása
F8	[DRAW]	X	X	X	X	X	Ablak újrarajzolása
F9	[SWITCH]	X	X	X	X	X	Előző üzemmódra kapcsol
F10	[TYPE]	X	X	X	X	X	Üzemmódváltó menü

4.7.2 A szövegszerkesztő különleges billentyűi

Ctrl+PgDn	Ugrás a következő lap elejére
Ctrl+PgUp	Ugrás az előző lapkezdésre
Ctrl+N	Új lap kezdete
Ctrl+B	Nyomtatási attribútum kijelölés kezdete
Ctrl+E	Nyomtatási attribútum kijelölés vége
Ctrl+C	Szövegrész másolása
Ctrl+M	Szövegrész mozgatása
Ctrl+D	Sor törlése
Ctrl+T	A sor törlése a kurzor előtt
Ctrl+Y	A sor törlése a kurzor után

Ctrl+BackSpace	Egy szó törlése a kurzor előtt
Ctrl+J	Automatikus sorrendezés be- és kikapcsolása
Ctrl+F	Új formátum sor készítése
Ctrl+O	Karaterek összefűzése (Amikor a szöveget nyomtatjuk, akkor ezen a helyen a nyomtatófej egy karakternyit visszalép, így egymásra nyomtatja az előző és az ezt követő karaktereket.)
Ctrl+P	A legutóbb másolt, törölt, vagy mozgatott blokkot a kurzor pozíciójában kiírja
Ctrl+X	A kurzor pozíciójában lévő karaktert kis-, vagy nagybetűs párjára cseréli
Ctrl+S	Adott szövegrész keresése
Ctrl+R	Adott szövegrész keresése és cseréje

4.7.3 A módmutató értékei

COMM	Kommunikációs üzemmód
DOC	Szövegszerkesztő
FORM	Adatbáziskezelés
GRAPH	Grafikon rajzolása
SHEET	Táblázatkezelés
CRIT	Kritériumrekord szerkesztése
EDIT	Javító üzemmód
ERROR	Hibaüzenet van
FILES	A munkaterületen állománylista van
FIND	Szövegrész keresése folyamatban
HELP	A munkaterület helyén a segítség látható
LABEL	Táblázatkezelőben szöveges adat bevitele
MENU	Valamelyik menü aktív
NAMES	A képernyőn tartomány, vagy ablaknevek aktívak
POINT	Tartománykijelölés van folyamatban
VALUE	Táblázatkezelőben numerikus adatot viszünk be
WAIT	A program dolgozik

4.7.4 A SYMPHONY indikátorai

CALC	A táblázat adatai nincsenek átszámítva
CAPS	Caps Lock billentyű aktív
CAPTURE	COMM üzemmódban a fogadó állomány aktív
CIRC	A ciklikus cellahivatkozás jelölése
DRAW	Az ablak nincs aktualizálva
END	Az END billentyűt egyszer lenyomtuk
LEARN	A program tanuló üzemmódban van
MACRO	Makró végrehajtása folyik
NEW RECORD	Új rekordot viszünk fel

NUM A NUM LOCK billentyű aktív

ONLINE A gép kommunikációs fogadásra kész

OVR Szövegszerkesztőben felülíró mód

PAUSE Makróvégrahajtás lépésenként történik

SCROLL A SCROLL LOCK billentyű aktív

STEP A makrót lépésenként fogja futtatni

USER A SYMPHONY a makró nevére vár

4.7.5 Beépített függvények

Matematikai függvények

@ABS(X) X abszolut értéke

@ASIN(X), @ACOS(X), @ATAN(X) X arkusz szinusza, koszinusza, tangense radiánban

@ATAN2(X,Y) X és Y arkusz tangense radiánban

@EXP(X) X természetes hatványa

@INT(X) X egészrésze

@LN(X) X e alapú logaritmusa

@LOG(X) X 10-es alapú logaritmusa

@MOD(X,Y) X / Y művelet maradéka

@PI értéke radiánban

@RAND 0 és 1 közé eső "véletlenszám"

@ROUND(X,n) X-et n-edik jegyre kerekíti. (-15 n +15)

@SIN(X), @COS(X), @TAN(X) X szinusza, koszinusza, tangense. (X radián)

@SQRT(X) X négyzetgyöke. X és Y lehet fix érték, vagy cellahivatkozás !

Logikai függvények

@TRUE Igaz állítás. Értéke 1.

@FALSE Hamis állítás. Értéke 0.

@IF(X,I,H) Ha X logikai kifejezés igaz, értéke I egyébként H (NA) értéke "Nincs érték".

@ISNA(X) Ha X cella értéke "Nincs érték", akkor ez a függvény 1, egyébként 0.

@ERR A cellába hibát jelző ERR értéket tesz.

@ISERR(X) Ha X cella értéke ERR, akkor a függvény értéke 1, különben 0.

@ISNUMBER(X) Ha X cella, vagy kifejezés értéke numerikus, akkor a függvény értéke 1, egyébként 0. A függvény értéke akkor is 1, ha a cella üres !

@ISSTRING(X) Azonos az előző függvénnyel, de értéke akkor 1, ha X értéke szöveges.

Dátum és időkezelő függvények

@DATEVALUE(dátum szövegesen) Visszaadja a dátum Julianus sorszámát. A szövegnek a beállított dátumformátum alapján értelmezhetőnek kell lennie.

@DATE(év,hó,nap) Visszaadja a dátum Julianus sorszámát. Az évet 1900 levonásával kell megadnunk.

@YEAR(sorszám) Megadja a sorszámhoz tartozó évszámot. Ehhez 1900-at hozzá kell adni.

@MONTH(sorszám) A sorszámhoz tartozó hónapot adja meg.

@DAY(sorszám) A sorszámhoz tartozó napot adja vissza.

@NOW A rendszerdátumot valamint a rendszeridőt adja meg numerikus formában.

@TIMEVALUE(idő szövegesen),

@TIME(óra,perc,másodperc) Az idő sorszámát adják meg.

@HOUR(sorszám), @MINUTE(sorszám), @SECOND(sorszám) A sorszámhoz tartozó órát, percet, vagy másodpercet adja meg.

Statisztikai függvények

@MIN(adatlista) Az adatlista legkisebb elemét adja.

@MAX(adatlista) Az adatlista legnagyobb elemét adja.

@SUM(adatlista) Az adatlista elemeinek összegét adja.

@AVG(adatlista) Az adatlista elemeinek átlagát adja.

@COUNT(adatlista) Az adatlista nem üres elemeinek száma.

@VAR(adatlista) Az adatlista elemeinek szórásnégyzete.

@STD(adatlista) A @VAR függvény négyzetgyöke.

Az adatlista elemei lehetnek akár tartományok is. Az elemeket a beállított határolójellel kell elválasztani!

Különleges függvények

@CHOOSE(X,adatlista) X értékének egészrészétől függően az adatlista egy elemét adja. Itt az adatlista elemei nem lehetnek tartományok, csak cellák.

@HLOOKUP(X,tartomány,sorszám) Az adott tartomány legfelső sorában megkeresi az X értéknél még kisebb értékű cellát, majd az adott cella alatti "sorszám"-adik cella értékét adja vissza.

@VLOOKUP(X,tartomány,oszlopszám) Az adott tartomány első oszlopában keres, és a tőle jobbra eső "oszlopszám"-adik cella értékét adja.

@@(cella koordináták) A cellakoordináták által megadott cellában leírt cellakoordinátákkal meghatározott cella értékét adja vissza.

@CELL("tulajdonság",tartomány) Az adott tartomány bal-felső cellájának kért tulajdonságát adja. A tulajdonságot idézőjelek közé kell tenni.

Tulajdonságok:

"ROW"	sor
"COL"	oszlop
"WIDTH"	szélesség
"PREFIX"	szöveges érték előtagja
"ADDRESS"	cím
"TYPE"	típus
"FORMAT"	formátum
"CONTENTS"	tartalom

@CELLPOINTER("tulajdonság") Az aktuális cella tulajdonsága.

4.7.6 Makróutasítások

{BEEP szám} Hang előállítása a szám értékétől függő frekvenciával. A szám 1-től 4-ig terjedhet.

{BLANK hely} Törli az adott cella, vagy tartomány tartalmát.

{BRACH hely} Feltétel nélküli ugrás. A megadott hely első cellájára adja a vezérlést, a makró végrehajtását innen folytatja.

{BREAKOFF} A BREAK billentyű és a makróból való nyomtatás letiltása. A makró nem állítható le.

{BREAKON} A makró megszakításának és a nyomtatásnak az engedélyezése.

{CLOSE} Az {OPEN} utasítással megnyitott állomány lezárása.

{CONTENTS célhely, forráshely,[,szélesség] [,formátum]} A forrásként megadott cella, vagy tartomány tartalmát másolja a célként megadott cellába, vagy tartományba. Megadható az átvitt adatok szélessége és formátuma is. A formátumot egy 0 és 127 közé eső számmal kell jeleznünk.

Az értékek jelentése a következő :

0-15 **Fixed** típus, a számnak megfelelő tizedesekkel.

16-31 **Scientific.** A számjegy a tizedesek számát jelzi 0 és 15 között. A 20-as érték (20-16=4) 4 tizedes jegyet ír elő.

32-47 **Currency,** 0 és 15 közé eső tizedesekkel, mint Scientific formátumnál.

48-63 **%** kijelzés. A szám itt is tizedeseket jelzi.

64-79 **Punctuated.**

112 Előjeles szám.

113 **General.**

114 **Date** D1. (NN-HHH-ÉÉ)

115 **Date** D2. (NN-HHH)

116 **Date** D3. (HHH-ÉÉ)

117 **Text.** Szöveges formátum.

118 **Hidden.** Rejtett adatformátum.

119 **Time** T1. (óó-pp-mm AM/PM)

120 **Time** T2. (óó-pp AM/PM)

121 **Date.** A Configuration menüben, az International beállításoknál megadott teljes dátum formátum.

122 **Date.** A Configuration menüben, az International beállításoknál megadott rész dátum formátum.

123 **Time.** Az International beállításnál megadott teljes időformátum.

124 **Time.** Az International beállításnál megadott rövid időformátum.

127 **Default.** A céltartományban alapértelmezett formátumra hozza az átvitt adatokat.

{DEFINE hely1[:típus],hely2[:típus],...} Szubrutinban használatos a paraméterek átvételekor. A szubrutin első utasítása kell legyen, ha paraméterrel hívjuk meg. Az átvett paramétereket egy megadott cellába teszi. A paraméterek típusa **value**, vagy **string** lehet.

{DISPACH hely} A helyként megadott cella tartalma egy cellacím, ahonnan a makró futását folytatni kell.

{FILESIZE hely} A megadott cellába teszi az {OPEN} utasításban megnyitott állomány hosszát byte-okban.

{FOR ciklusváltozó, kezdőérték, végérték, lépésköz, makró név} A megadott makró ciklikus végrehajtása, az értékektől függő számszor. Ha a kezdőérték nagyobb a végértéknél, akkor egyszer sem hajtódik végre a megadott makró.

{FORBREAK} A FOR ciklus megszakítása. A végrehajtás a FOR utasítás utáni utasításon folytatódik.

{GET hely} Várakozás bilentyűleütésre. A billentyű leütése után annak nevét, vagy karakteres értékét az adott cellába, mint szöveget elhelyezi. A CALC billentyű leütésére nem reagál.

{GETLABEL üzenet, hely} Kiírja az üzenetet a képernyő legfelső sorába, majd egy karaktersorozat begépelésére vár. A karaktersorozatot az ENTER billentyűvel kell lezárni. A beolvasott szöveget a megadott cellában elhelyezi.

{GETNUMBER üzenet, hely} Azonos az előző funkcióval, de egy számot olvas be. A szám ellenőrzése a beállított "pontozási" karaktereknek megfelelő. Ha a beadott karaktersorozat nem szám, a megadott cella értéke ERR lesz, egyébként numerikus értékként kerül bele az adat.

{GETPOS hely} Az {OPEN} utasításban megnyitott állomány mutatójának értékét teszi az adott cellába.

{HANDSHAKE szöveg, válasz, idő [,vételi terület]} A megadott szöveget elküldi a kapcsolt számítógépnek, adott ideig vár az adott válaszszövegre, majd ha megadtunk vételi területet, a választ elhelyezi azon.

{IF feltétel} Ha a feltétel értéke nem 0, vagyis igaz, a végrehajtás az ugyanabban a cellában lévő makróutasításokkal folytatódik.

{INDICATE [szöveg]} A módmutató helyére a megadott szöveget, vagy ha nem adtunk meg szöveget, a módmutató eredeti értékét írja.

{LET hely, szám vagy szöveg[:típus]} Az adott helyre teszi a megadott szöveget, vagy számot. Ha szöveg eredményű kifejezést adunk meg, ki kell tenni a **:value** típusjelzőt. Ha egy számot, mint karaktersorozatot szeretnénk elhelyezni, akkor használjuk a **:string** típust.

{LOOK hely} Megnézi, hogy van-e olyan billentyűleütés a pufferben, amit még nem olvastunk be, és ha van elhelyezi azt a megadott helyre. Ha nincs leütött billentyű, akkor egy szóközt helyez el. A billentyűnyomást nem veszi ki a pufferből !

{MENUBRANCH hely} A megadott cellától kezdődően elhelyezett menü meghívása, majd a kiválasztott menüponthoz tartozó makró lefutása után befejezi a működést. A menü három sorból áll. Első sor a Menüpontokat tartalmazza, egymás utáni cellákban. Minden szó különböző nagybetűvel kell kezdődjön ! A második sorban a menüpontokhoz tartozó egysoros magyarázatok, a harmadik sortól minden menüpont alatt a menüponthoz tartozó utasítások legyenek !

{MENUCALL hely} Azonos a {MENUBRANCH} utasítással, de a makró futása a menü végrahajtása után a következő utasításon folytatódik.

{ONERROR cella[, üzenet helye]} A program hibaüzeneteit és azok kezelését irányíthatjuk át ezzel az utasítással. Az utasítás használata után, ha hibát észlel a program, a makró utasítását a megadott cellától folytatja. Ha a második paramátert is megadjuk, a képernyő alsó sorában megjelenő hibaüzenetet az adott cellában is elhelyezi.

{OPEN név, mód} A "név"-vel nevezett szöveges állomány megnyitása. A "mód" lehet R(read), W(write), vagy M(modify). R módban megnyitott állományból csak olvasni tudunk, W mód esetén csak írni, míg M mód esetén írni is olvasni is. **FIGYELEM !** Ha W módban nyitunk meg egy állományt, akkor a már létező azonos nevű törlölni fogja a lemezről.

{PANELOFF} Kikapcsolja a képernyő két legfelső sorában lévő üzenetek és menük megjelenítését. A makró működése közben nem villog a képernyő, pl. amikor egy billentyűleütési soro-

zattal használunk egy menüt. Ha a makró interaktív, tehát kérdéseket tesz fel a felhasználónak, a bekapcsolásról nekünk kell gondoskodni.

{PANELON} Az előző utasítás ellentettje. Bekapcsolja a kijelzést.

{PHONE telefonszám} A számítógépre kapcsolt "modem"-en keresztül felhívja a karaktersorozatként megadott telefonszámot.

{PUT tartomány, oszlop, sor, érték} A megadott tartomány megadott oszlop- és sorpozíciónak megfelelő cellában helyezi el a szöveget, vagy számértéket. Az oszlopok és a sorok a tartományon belül értendők, és a számozásuk 0-tól kezdődik. Ha a cella az adott tartományon kívülre esik, a makró futása leáll, és az {ONERROR} utasítás sem segít.

{QUIT} A makró futását megszakítja. Nem feltétlenül szükséges, a makró futása az első üres cellánál is leáll.

{READ szám, hely} A megnyitott állományból adott számú karaktert olvas be, és az adott cellában helyezi el. Az állománymutató az utoljára olvasott karakter mögé mutat.

{READLN hely} Működése azonos a {READ} utasításával, de egy egész sort olvas be.

{RECALC tartomány, [feltétel,] hányszor} Az adott tartomány kifejezéseinek újraszámítása. Ha ciklikus hivatkozás van a kifejezésekben, a "hányszor" paraméter adja meg az újraszámítási ciklusok számát, valamint az újraszámítás mindaddig tart, amíg a megadott feltétel nem teljesül.

{RECALCCOL tartomány,[feltétel,] hányszor} Azonos az előző paranccsal, de az újraszámítást nem hivatkozási, hanem oszlop sorrendben végzi.

{RESTART} Ezt az utasítást akkor kell kiadni, ha egy meghívott szubrutint megszakítunk, és nem akarjuk, hogy a makró a szubrutint indító utasítás utáni utasításon folytatódjon.

{RETURN} Szubrutinból való visszatérés. A makró végrehajtása a szubrutint hívó utasítás utáni utasításon folytatódik.

{SETPOS pozíció} A megnyitott szöveges állomány mutatóját állíthatjuk ezzel az utasítással. A karakterek számozása 0-val kezdődik.

{WAIT idő} A megadott ideig történő várakozás. Az időt sorszámmal is és kifejezéssel is megadhatjuk.

{WINDOWSOFF} Kikapcsolja a képernyő közepén elhelyezkedő munkaterület újrarajzolását műveletek esetén. A cellákba vitt adatok nem jelennek meg a képernyőn.

{WINDOWSON} A munkaterület újrarajzolás engedélyezése.

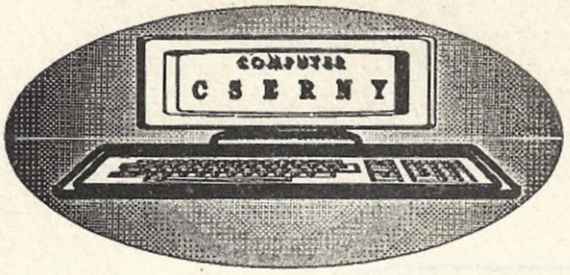
{WRITE szöveg} A megnyitott szöveges állományba, az állománymutató által mutatott helyre kiírja az adott szöveget.

{WRITELN szöveg} A kiírt szöveg végére kiír egy CR-LF (kocsivissza-soremelés) karakterpárt is.

{név} Adott nevű szubrutin hívása. A név egy érvényes cellanév kell legyen, ahol a meghívott szubrutin első utasítása van. Ha paramétert is át kívánunk adni, akkor azokat a név után felsorolva adhatjuk meg, ne felejtsük el a {DEFINE} utasítást.

{?} Megszakítja a makró futását, és az ENTER billentyű lenyomásáig bármilyen beavatkozást megenged. ENTER után, a következő utasításon folytatódik a végrehajtás.


FrameWork
adatkezelő-,
táblázatkezelő-,
és szövegszerkesztő
rendszer



Számítástechnikai szolgáltatások

minden elemét

megkaphatja egy helyen.

Az utazás PÉNZ és IDŐ!

SZÁMÍTÁSTECHNIKAI

szolgáltatások:

rendszerszervezés és tervezés

programozás

szaktanácsadás

oktatás

kiadványok szerkesztése

(szedés, tördelés, nyomtatás)

reklámok, névjegyek,

rajzi dokumentációk készítése

számítógépes rendszerek és

elemeik forgalmazása

Cserny Péter
okl.gm.
számítástechnikai rendszerszervező
AutoCAD tervező
2600 Vác, Árpád út 90.
Tel.: 06-(27)-11859

5. FRAMEWORK

5.1 A FRAMEWORK II interaktív használata

5.1.1 Alapok

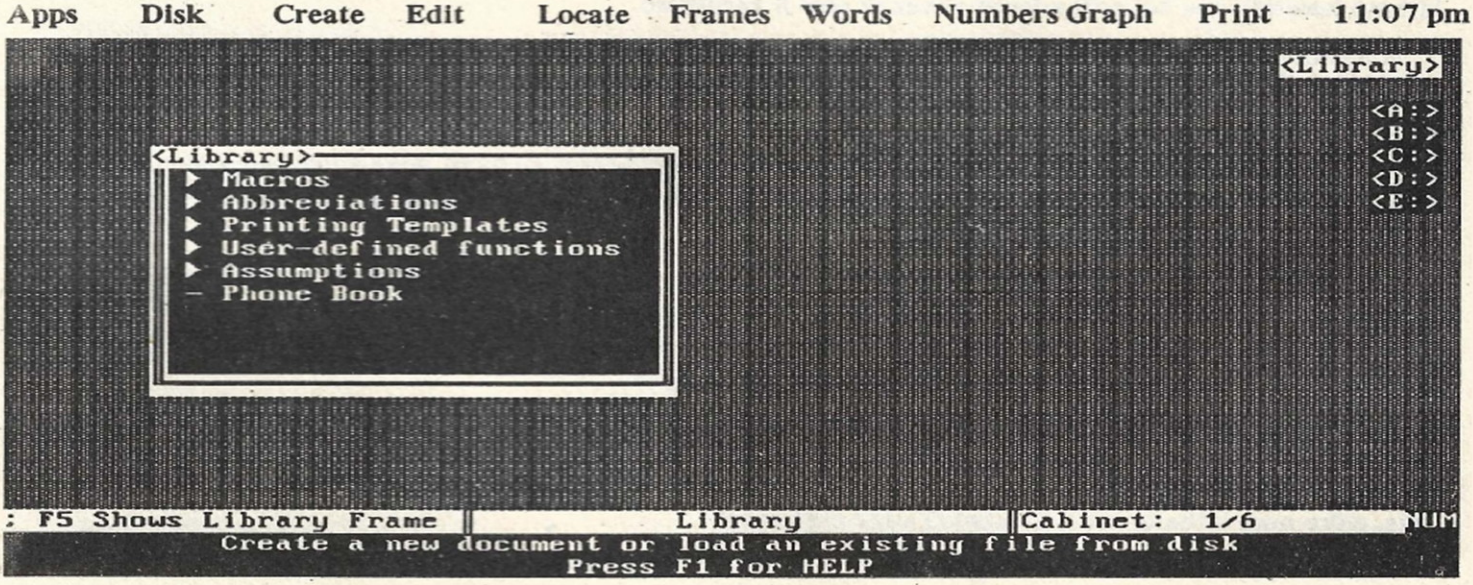
Az alkalmazói programok többféleképpen csoportosíthatók, egyféle szétválasztás lehet pl.: célprogramok és integrált programcsomagok. Az első csoportba tartozók lehetnek szövegszerkesztők, táblázatkezelők, adatbáziskezelők, stb. Ezek egy jól körülhatárolt feladat magasszintű megoldását teszik lehetővé. Ilyen például az MS-Word 5.0 nevű szövegszerkesztő, amely a professzionális igényeket is kielégíti.

A FRAMEWORK II, mely az amerikai Ashton-Tate szoftverház terméke, az integrált programcsomagok közé tartozik, egyesíti magában a különálló célrendszerek funkcióit.

A program által nyújtott szolgáltatások:

- **szövegszerkesztő** (szövegfeldolgozó funkciókkal),
- **táblázatkezelő,**
- **adatállománykezelő,**
- **grafika,**
- **távadatfeldolgozás**
- **vázlatkészítés,** mely lehetővé teszi az összetartozó feladatok összekapcsolását.

A program alapfilozófiája: az irodai munka szimulálása. Amikor belépünk a programba, a képernyőn megjelenik az ún. **DESKTOP** (asztallap), ez a program felhasználói felülete.

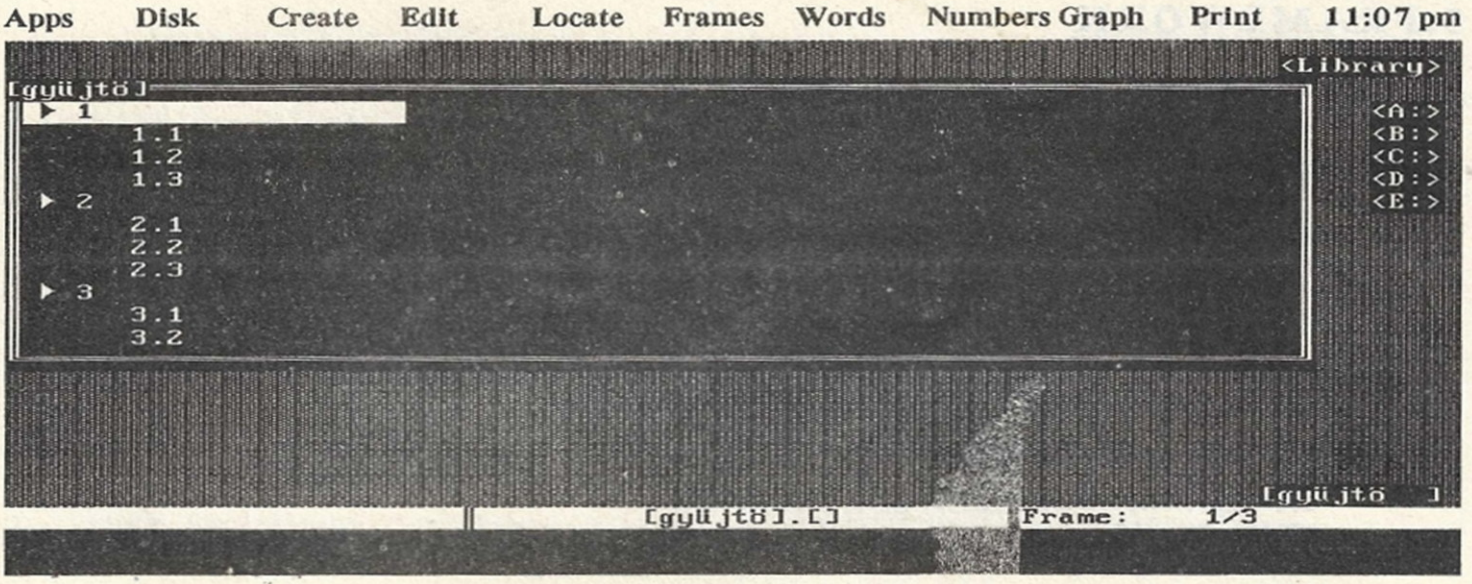


A képernyő felső sorában látható a főmenü és az óra, az asztallap jobb oldalán találhatók a **CABINET**-ek (iratszekrények). Ezek az iratszekrények egyrészt a különböző lemezegységek, másrészt a **LIBRARY**-k (könyvtárak). A képernyő alján a harmadik sorban látható a státuszpanel, mely az aktuális feldolgozásról informál bennünket, az alatta lévő sorban történik majd az adatok, formulák szerkesztése és az utolsó sorba írja a rendszer az üzeneteit.

A FrameWork II legfontosabb részei az un. **FRAME**-ek (keretek). Bármely feladatra használjuk is a programot, egy keretben dolgozunk, melyet nekünk kell létrehozni a főmenü **CREATE** almenüjében.

Négyféle kerettípus közül választhatunk:

- **Empty/Word Frame** szövegszerkesztéshez, vagy grafikához
- **Spreadsheet** táblázatkezeléshez
- **Database** adatállományok kezeléséhez
- **Outline** ún. gyűjtőkeret, melyben az összetartozó, más típusú kereteket tároljuk.



Az íróasztalunkon a keretek két állapotban lehetnek: **nyitva** vagy **zárva**.

A **Create** menüvel természetesen nyitott kereteket lehet létrehozni, ekkor a kurzor a keret bal felső sarkában villog, ide gépelhetjük a keret nevét. A keret létrehozásával egyidejűleg az íróasztal jobb sarkán látható kerettálcába bekerül a keret nevének első 8 karaktere.

A nyitott keret lezárása az **ENTER** billentyűvel történik, ha a kurzor a kereten villog. A keret belsejébe a **DOWNLEVEL** (szürke+) billentyű segítségével léphetünk, vissza a keret szélére az **UPLEVEL** billentyűvel juthatunk.

Ha egyszerre több keret van az íróasztalon, azaz egyszerre több munkán dolgozunk (vagy egy munkát több részre bontunk), a keretek közötti mozgás az **UPARROW** (fel nyíl) és a **DOWNARROW** (le nyíl) történik, ha előbb a keret szélére álltunk .

Az iratszekrénybe betekinteni a **SCROLL LOCK** billentyű lenyomása után lehet. A kabinetek közötti mozgásra itt is a kurzor-billentyűk szolgálnak. A lemezegységek tartalmának megtekintésére az **ENTER** ad lehetőséget, ezután a **DOWNLEVEL** billentyűvel lépünk a keret belsejébe, ott a kurzor-billentyűkkel kiválasztjuk a kívánt állományt, majd az **ENTER**-rel betölthetjük.

A lemezre mentés a **CTRL-ENTER** billentyűkombináció segítségével történik, az állomány abba a kabinetbe kerül, ahonnan betöltöttük, ill. első mentéskor az installálásnál meghatározott alapértelmezett keretbe. A mentést célszerű kb. negyedóránként végrehajtani, így megelőzhetjük az áramkimaradásból, vagy más, előre nem látható okból keletkező adatvesztéseket.

A **SCROLL LOCK** ismételt lenyomásával abba a keretbe jutunk vissza, ahonnan beléptünk a kabinetbe.

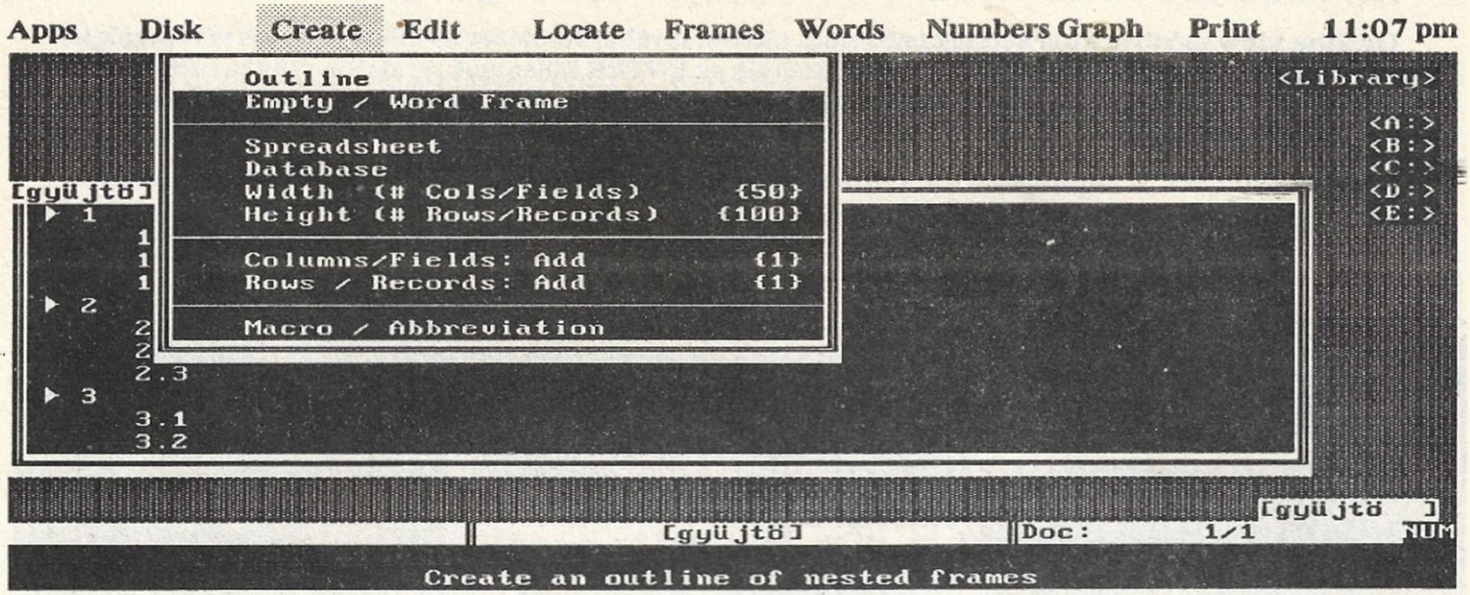
5.1.2 Gyűjtőkeretek

A gyűjtőkeretek (**OUTLINE** - talán vázlat lenne a helyes fordítása) a munka megszervezésében:

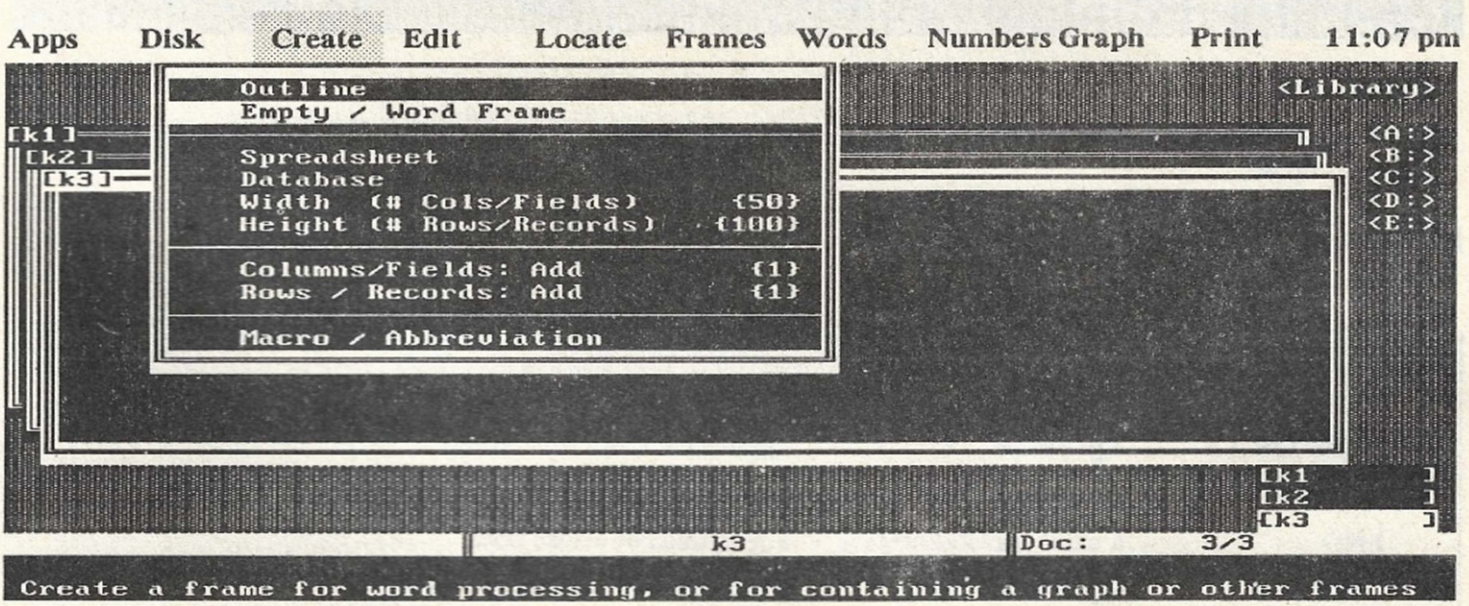
- lehetővé teszi egy dokumentum összeállítását, módosítását, részeinek átcsoportosítását és áttekintését
- támogatja a felülről lefelé, vagy az alulról felfelé történő feladatmegoldást.

Gyűjtőkeretet kétféleképpen tudunk létrehozni:

1. A **Create** menü **Outlines** opcióját választva a FRAMEWORK II egy decimálisan számozott kétszintű keretrendszert hoz létre, melyet később az igényeinknek megfelelően módosíthatunk.



2. Szintén a **Create** menüben, de az **Empty/Word Frame** opciót válasszuk, lépjünk be a keretbe és ott hozzunk létre újabb kereteket. Ez a másik lehetőség a keretek egymásba ágyazására. Ha egy gyűjtő keretben azonos szinten több keretet kívánunk létrehozni, akkor az első, már létrehozott keret címére állva válasszuk ki a kívánt típusú keretnek megfelelő menütételt, többször egymás után.



Bármelyik módon hoztuk is létre a keretrendszert, adhatunk nevet az egésznek és külön-külön a részeknek is. Ez úgy történik, hogy a keret szélére lépünk és begépeljük a nevet. A második módszerrel létrehozott gyűjtőkeret esetén a fejezetek számozását is nekünk kell elkészíteni, ezt a **Frames** menü **Number Labels** menüpontja végzi el.

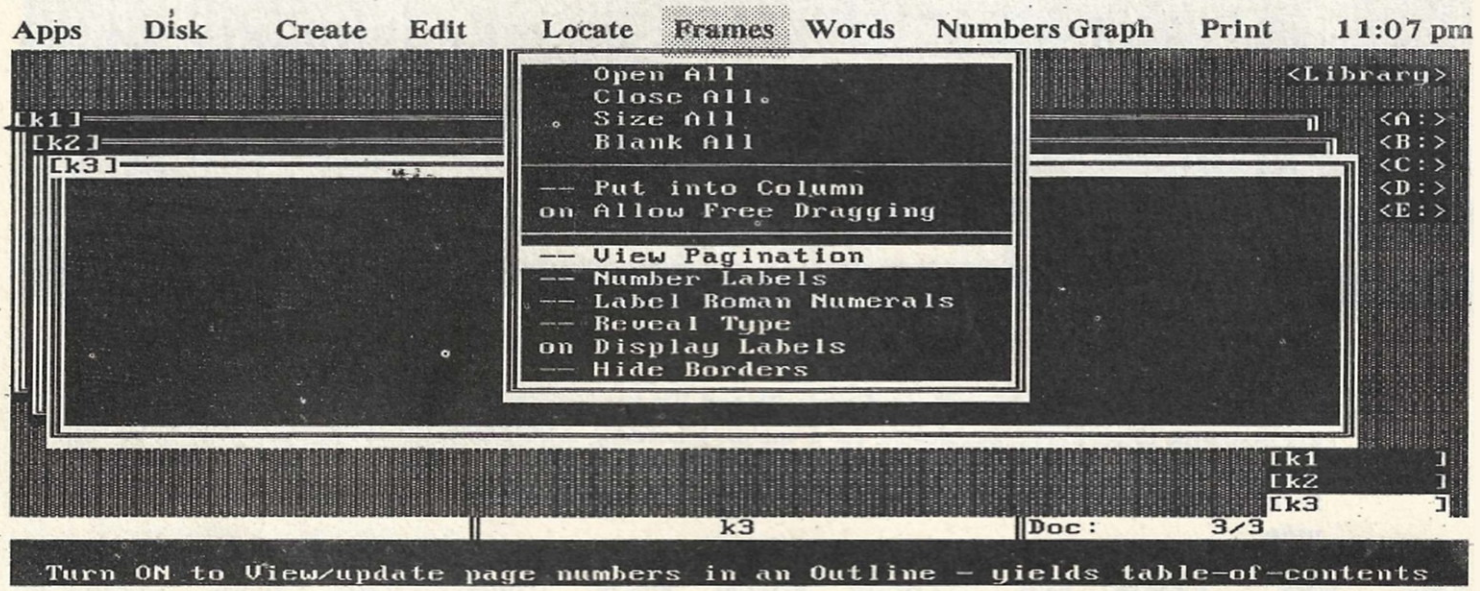
A gyűjtőkeret kizárólag kereteket tartalmazhat, semmi mást.

A keretrendszer struktúráját az **F10** billentyű segítségével ún. **outline view** (vázlatszerű nézet) módban tekinthetjük meg, ekkor a rendszer úgy mutatja be a szerkezetet, mintha tartalomjegyzék lenne. Az **F10** újabb lenyomására bemutatja a keretek tényleges elhelyezkedését, ez a **contents view** (tartalom megtekintés).

Az **F9** billentyűvel a kiválasztott keretet teljes képernyő méretűre nagyíthatjuk.

Outline view módban azon keretek neve előtt, melyek további kereteket tartalmaznak, kis háromszögjel jelenik meg. Ha egy ilyen keret címére állunk és leütjük az **ENTER** billentyűt, az alsóbb szinten levő keretek címe eltűnik, újabb **ENTER** leütésére ismét megjelenik. Ez lehetővé teszi, hogy munkánkat mindig a kívánt mélységig tekintsük át.

A FRAMEWORK II segítségével elkészíthetjük munkánk tartalomjegyzékét is, ha a **Frames** menüből a **View Pagination** opciót választjuk ki.



A mentést a **CTRL-ENTER** billentyűkombinációval végezve a FRAMEWORK II a teljes keretrendszert lemezre menti.

Mozgás a gyűjtőkeretben

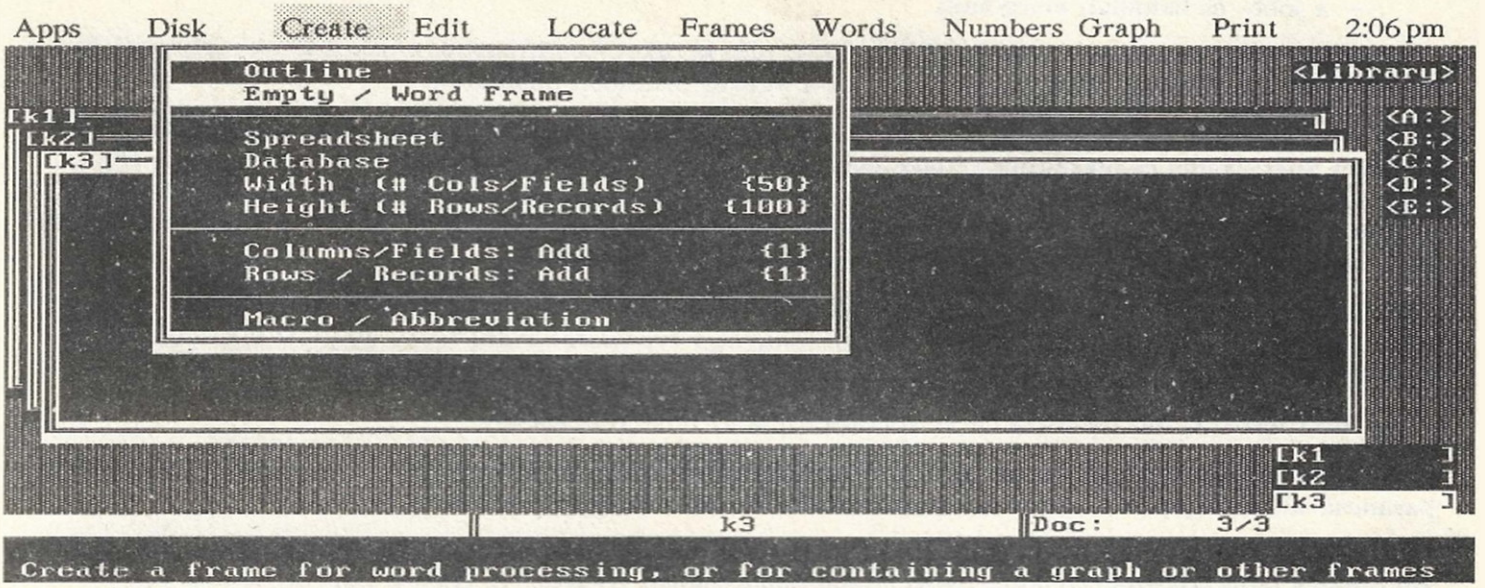
Billentyű	Hatása	CTRL + Billentyű
KURZOR FEL	egy sort fel	ugyanaz a szint, előző keret
KURZOR LE	egy sort le	ugyanaz a szint, köv. keret
HOME	a szint első keretére	a gyűjtőkeret első sora
END	a szint utolsó keretére	a gyűjtőkeret utolsó sora
UPLEVEL	egy szinttel fel	a gyűjtőkeret határa
DOWNLEVEL	egy szinttel le	a legalsó szint

5.1.3 A szövegszerkesztő modul

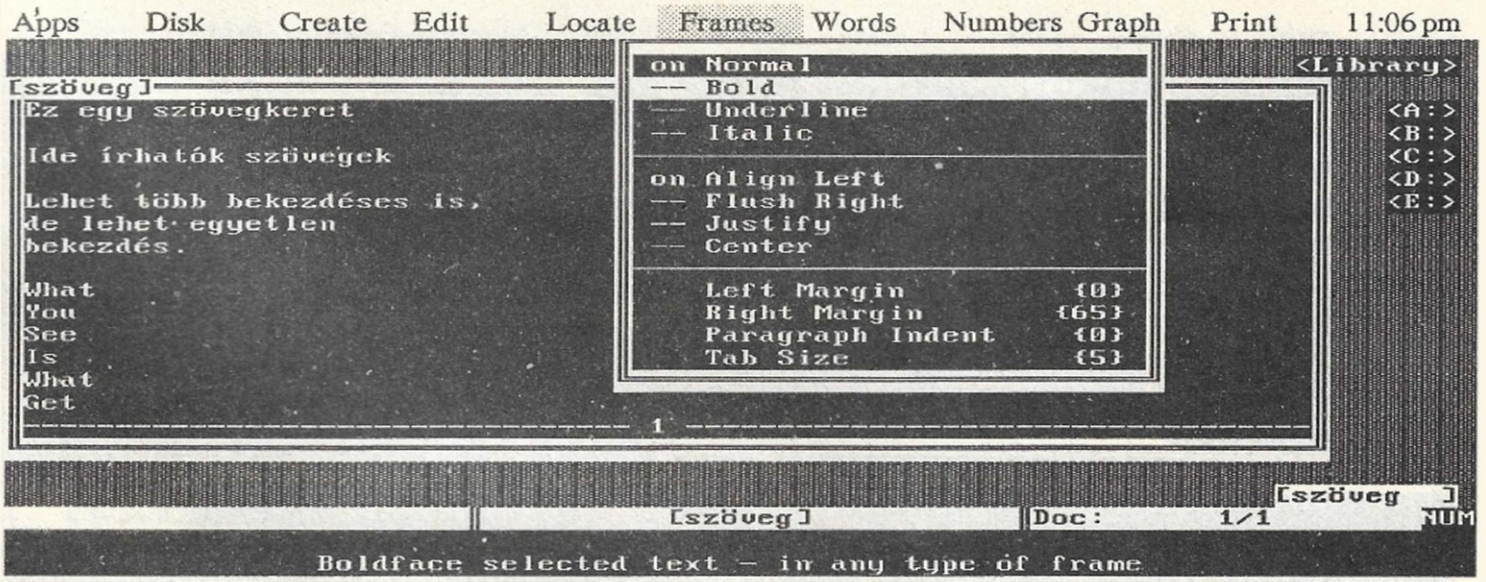
Bár nem veheti fel a versenyt az élvonalbeli szövegszerkesztőkkel, de meglepően széleskörű szolgáltatásokkal áll a rendelkezésünkre.

Funkcióit még a táblázatkezelő és adatállománykezelő modulokban is használhatjuk, valamint rendelkezésünkre állnak gyűjtőkeretek is, ezzel a FRAMEWORK II integráltságban felülmúlja versenytársait. A szövegszerkesztő segítségével a kisebb javításoktól a hosszabb szövegrészek mozgatásán át a dokumentum teljes átszerkesztése elvégezhető. Használhatunk magyar ékezetes betűket is, persze csak azokat, amelyeket a számítógép karakterkészlete tartalmaz.

Dokumentumokon **Word** típusú keretekben dolgozhatunk, ezt a Create menü **Empty/Word Frame** menüpontjával hozhatjuk létre.



A szövegszerkesztő a What You See Is What You Get (WYSIWYG) elv alapján működik, vagyis a nyomtatón azt kapjuk, ami a képernyőn megjelenik.



Lépjünk be a létrehozott keretbe, és kezdjünk el begépelni egy tetszőleges szöveget. A gépelés közben észrevehetjük, hogy a jobb margót elérve a szavak automatikusan kerülnek át az új sorba (**word wrap**), valamint, ha a szövegbe betoldunk, vagy törlünk belőle, akkor a rendszer automatikusan kiigazítja a szerkesztett sorokat.

Az **ENTER** használatát a **word wrap** funkció feleslegessé teszi, csak akkor szükséges, ha új bekezdést akarunk nyitni, vagy üres sort szeretnénk beszúrni.

Több oldalas szöveg esetén a laptördelést láthatóvá tehetjük a **Frames** menü **View Pagination** opciójával.

A sorokat többféleképpen formázhatjuk:

- alapértelmezésben balra igazítottak,
- igazíttathatunk jobbra,
- középre,
- a jobb- és baloldali margóhoz.

Használhatunk "vastagon" szedett, aláhúzott, v. dőlt betűket, beállíthatjuk a bekezdések mélységét és a tabulátorjelek távolságát. Mindezek a funkciók a **Words** menüben találhatók.

A **word wrap** funkció miatt előfordul, hogy egyik-másik sor foghíjas lesz, ezen segíthetünk az ún. **Hyphen Soft** ("puha elválasztójel") funkcióval, azaz olyan elválasztójeleket szúrunk a szövegbe, melyek a sor közepén nem láthatók, csak a sor végén aktiválódnak (**Edit** menü). Ezzel ellentétes az "összekötő szóköz", melyet a **CTRL-SPACE** billentyű kombinációval iktathatunk a szövegbe, ilyenkor azt a két szót, melyek közé összekötő szóközt tettünk, a rendszer nem teszi két külön sorba.

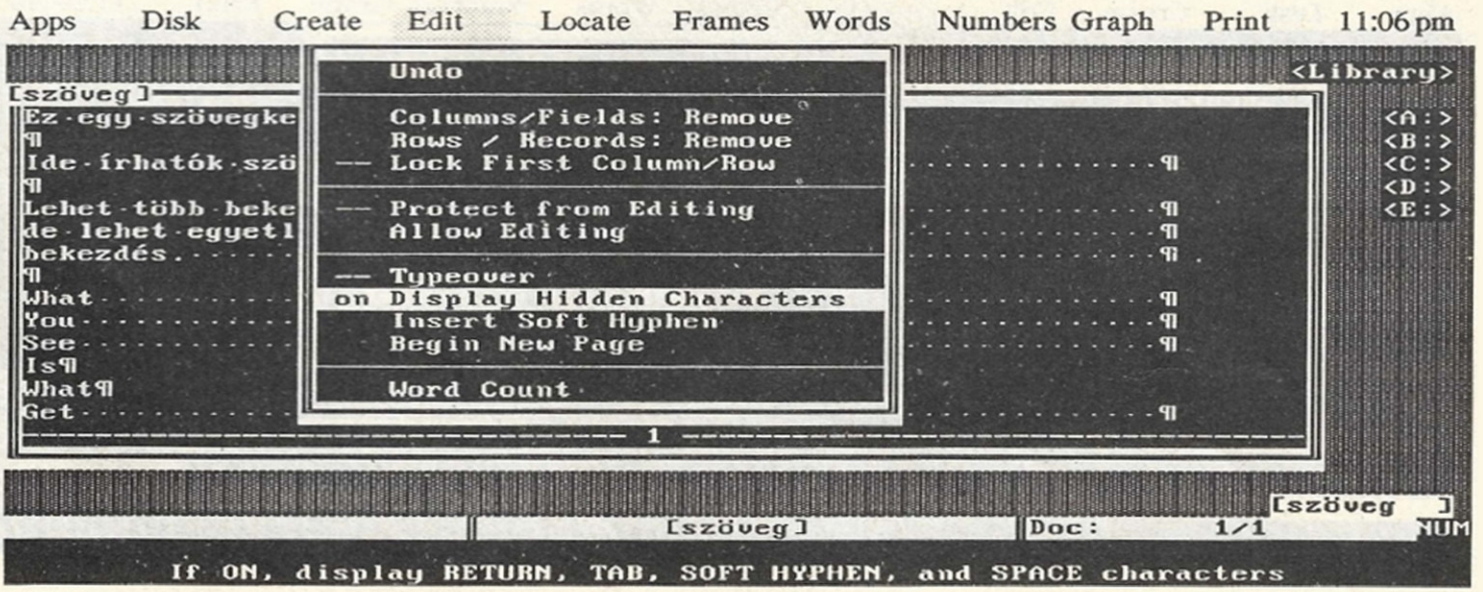
Szövegrészeket mozgathatunk, másolhatunk dokumentumon belül, illetve más keretekből, vagy keretekbe. Az **F6** billentyűvel tudjuk megjelölni a másolni, vagy mozgatni kívánt részt, majd az **F7**, vagy **F8** billentyű lenyomása után pozícionáljunk a szöveg új helyére és nyomjuk le az **ENTER**-t.

Az Edit menü **Undo** parancsával visszaállíthatjuk az eredeti állapotot.

Betűket, szavakat, hosszabb szövegrészeket megkerestethetünk az aktív keretben a **Locate** menü **Search** parancsával, a megtalált részeket ki is cseréltethetjük, ha a **Replace** parancsot választjuk.

A **Labels Included** a keretek nevében is keres, a **Contents Included** a keretek belsejében, a **Formulas Included** a táblázatkezelő celláiban és az adatállományokban is keres. Az **Ignore Capitalization** figyelmen kívül hagyja a kis– és nagybetűk közötti különbséget.

A FRAMEWORK II segítségével láthatóvá tehetjük a szövegben levő bekezdészáró karaktereket, a "puha" és "merev" elválasztójeleket, a rendszer által betoldott igazító szóközöket (**Edit** menü **Display Hidden Characters**).



Az elkészített dokumentumot a **Print** menü segítségével tudjuk kinyomtatni. Nyomtatáskor a keretszegély nem jelenik meg a kinyomtatott dokumentumon, de a keret neve, ha akarjuk, igen. Készíthetünk fej- és lábléceket, további szövegformálásokat is végezhetünk nyomtatás előtt.

Billentyűhasználat a szövegszerkesztőben

Billentyű	Funkció	CTRL + billentyű
KURZOR FEL	egy sor fel	előző mondatra
KURZOR LE	egy sor le	következő mondatra
KURZOR BALRA	egy karakter balra	előző szóra
KURZOR JOBBRA	egy karakter jobbra	következő szóra
HOME	sor eleje	a keret első karakterére
END	sor vége	a keret utolsó karakterére
PgUp	egy lap vissza	előző bekezdésre
PgDn	egy lap előre	következő bekezdésre
DEL	a megjelölt karakter(ek) törlése	a kurzortól jobbra levő szó törlése
BACKSPACE	a kurzortól balra lévő karakter törlése	a kurzortól balra levő szó törlése
TAB	kurzor a következő tabulátor pozícióra	
ENTER	új bekezdés	
CTRL + Szám	leütése után az első begépelt karakter annyiszor íródik le, amekkora számot választottunk.	

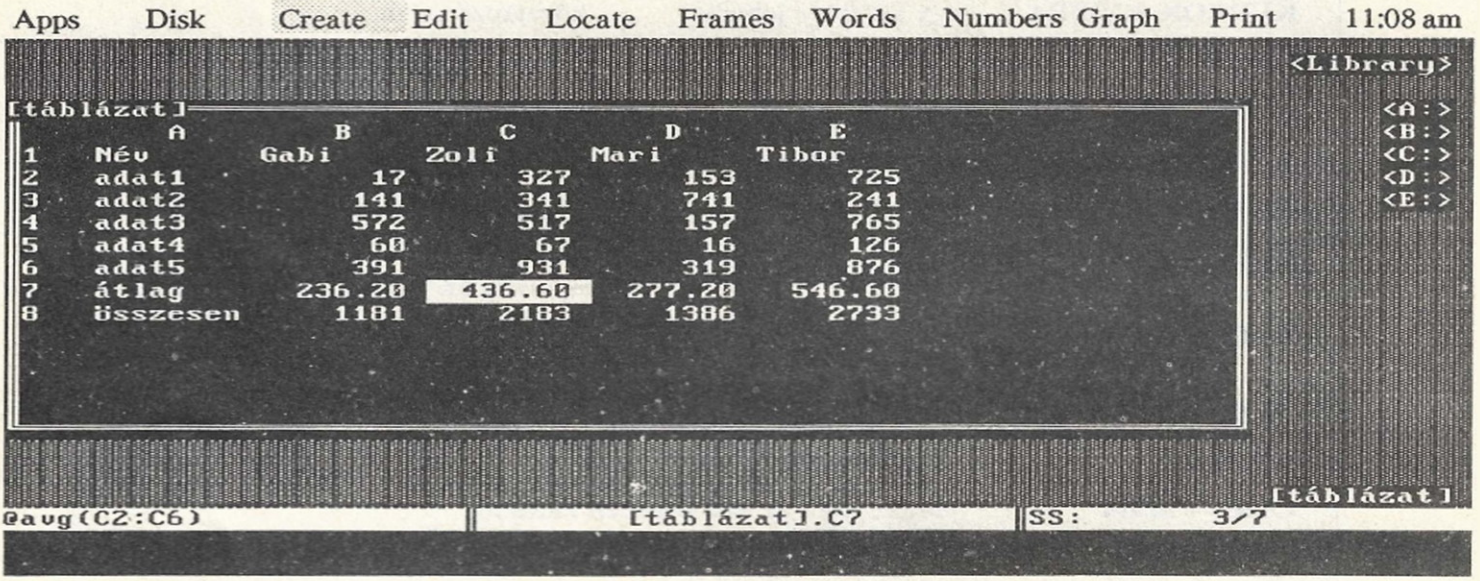
5.1.4 A táblázatkezelő modul

A munkalap egy négyzethálós papírlaphoz hasonlít, az adatokat a négyzetháló négyzeteibe, az ún. **cellákba** írjuk. Ami mégis megkülönbözteti a papírtól munkalapunkat az, hogy a cellák között kapcsolatot teremthetünk. Így, ha egy cella tartalma megváltozik, a vele kapcsolatban álló cellák tartalma is változik.

A munkalapot különféle gazdasági, statisztikai, műszaki számítások gyors elvégzésére, tervek, előrejelzések elkészítésére használhatjuk.

A munkalapot a **Create** menüből hozhatjuk létre a **Spreadsheet** opció választásával. A rendszer alapértelmezésben **50** oszlopos, **100** soros, üres munkalapot tartalmazó keretet hoz létre. A méreteket még a létrehozás előtt megváltoztathatjuk a **Create** menü **Width** és **Height** menüpontjaival, vagy munka közben az **Edit** menü **Column/Fields: Add, Rows/Records: Add** menüopciókkal növelhetjük, ill. az **Edit** menü **Column/Fields: Remove, Rows/Records: Remove** menüopciókkal csökkenthetjük.

A szükséges méretet meg nem haladó nagyságú táblázatot érdemes használni, ezzel takarékoskodhatunk a rendelkezésre álló memóriával.



A munkalap **celláira** a sakkból ismert módon hivatkozhatunk, az **oszlopokat** az ABC betűivel, a **sorokat** számokkal jelölik meg. Adott oszlop és adott sor kereszteződésében levő cellára az oszlop és a sor egymás mellé írt jelével hivatkozhatunk. Pl. az E oszlop negyedik sorában levő cellára az E4 jelöléssel.

Az ilyen hivatkozás egyértelműen azonosítja a cellát, a szakirodalomban referenciaként emlegetik. A használt munkalap általában nagyobb, mint a keretünk, így ezt a keretet, mint egy ablakot mozgathatjuk a táblázatunkon.

Adatokat abba az ún. **aktív cellába** vihetünk be, amely **inverz** kijelzésben jelenik meg a képernyőn. Az aktív cella **koordinátái** a státuszpanel középső részében is kiíródnak. Az említett mozgási lehetőségeken túl a **Locate** menü **Goto** utasításával egy meghatározott keretre, vagy cellára ugorhatunk. A cellákba numerikus vagy alfanumerikus adatokat vihetünk, a szöveges adatokkal közvetlenül számítási műveletek nem végezhetők. Ha betűvel kezdődik az adat, gond nélkül bevihető, ha számmal (pl. telefonszám), melyet szöveges adatként kívánunk tárolni, a bevitelt a szóköz billentyű leütésével kell kezdenünk. Már a cellában levő szöveges adat szerkesztése szintén a szóköz billentyű leütésével kezdhető meg. A bevitelt a **TAB, ENTER, ESC** billentyűk valamelyikével zárhatjuk le.

Numerikus adatok bevitele többféleképpen történhet:

- számok közvetlenül begépelhetők
- szintén közvetlenül gépelhetők be a "@", ",", "+", "-", "(", "#", "$" vagy '"' szimbólumokkal kezdődő adatok

– cellahivatkozásokat az F2 billentyű leütése után írhatunk be, vagy a kurzormozgató billentyűk segítségével, rámutatásként.

A bevitelt a **TAB** vagy az **ENTER** billentyű zárja le. Már a cellában levő érték, vagy kifejezés szerkesztése az **F2** billentyű lenyomása után lehetséges. Az aktív cella tartalma a **DEL** billentyűvel törölhető.

Egyszerre több cella tartalmát az **F6** és a kurzormozgató billentyűk használata után tudjuk törölni. Ha a munkalap minden adatát törölni kívánjuk, válasszuk a **Frames** menü **Blank All** opcióját.

Az oszlopszélesség alapesetben 9 karakternyi, ez a munka során megváltoztatható az **F4** billentyűvel. Ha a cellában tárolt információ hossza meghaladja az oszlop szélességét és a jobb oldali cellák üresek, akkor a teljes szöveg megjelenik, ha nem üresek akkor a rendszer levágja a kilógó részt.

Numerikus adat esetén csillagok (*) jelennek meg a cellában, jelezve, hogy meg kell növelni az oszlopméretet. Megtehetjük, hogy egy vagy több oszlop szélességét 0-ra változtatjuk, ekkor az oszlop jele is eltűnik. Használhatjuk ezt akkor, ha egymástól távol levő oszlopok adatait szeretnénk megjeleníteni.

Természetesen az elrejtett cellák adatai nem vesznek el, a szélesség helyreállítása után újra megtekinthetők.

A munkalapon megjelenített adatok formátumának kialakítását a **Numbers** menüvel végezhetjük el, ahol beállíthatjuk a megjelenítendő tizedesek számát, a megjelenítés pontosságát, a kiírási módot és a cellán belüli igazítást. Használhatjuk még a **Words** menü kiírást befolyásoló opcióit is.

A munkalap bármely cellája tartalmazhat formulát is, amely tetszőleges algebrai kifejezés, függvény, vagy más cella adatára való hivatkozás lehet (Pl.: **@sum(A5:A8)+D7**). Az aktív cella formulája a **státuszpanel** bal szélén jelenik meg, a cellában a formula kiszámolt eredménye olvasható.

A formulák beírásakor be kell tartani a szintaktikai szabályokat, különben hibaüzenetet kapunk. A formulák írását általában az **F2** billentyűvel kezdjük, kivéve, ha a formula számmal, vagy a "@", ",", "+", "-", "(", "#", "$" vagy '"' szimbólumok valamelyikével kezdődik. A formula beírását **ENTER**-rel zárjuk le.

A rendszer lehetővé teszi, hogy formuláink akár több oldalasak legyenek, egy sorba maximálisan **255** karaktert írhatunk, sőt formuláinkba a ";" után megjegyzéseket is tehetünk. A munkalap használatának legfőbb előnye, ha valahol megváltoztatunk valamit, akkor a változás hatása pillanatokon belül megjelenik a táblázat többi részén is.

A FRAMEWORK II alapbeállításban minden változásnál automatikusan **újra számolja** az egész munkalapot természetes sorrendben, azaz először azokat a cellákat, melyekre más cellák hivatkoznak.

Ezt az átszámolási sorrendet a **Numbers** menü **Options for Recalc** almenüjének **Row-wise** kapcsolójával sorfolytonosra változtathatjuk. Ugyanebben az almenüben állíthatjuk át az automatikus átszámolást kézi vezérlésűre.

Az **Edit** menü **Lock First Column/Row** parancs hatására az első oszlop és az első sor mindig a képernyőn marad. A cellahivatkozásoknál az oszlop és a sor jelek helyett használhatjuk az első sorba, illetve az első oszlopba írt, akár magyar nyelvű neveket is, ha begépeljük a felkiáltójelet. Természetesen figyelni kell az egyedi névadásra. Az általunk használt munkalapokat össze is kapcsolhatjuk úgy, hogy az aktív munkalapon közvetlenül hivatkozunk egy másik munkalap cellájára, ekkor a cella értékét kapjuk meg, de nem kényszeríti ki a másik munkalap újbóli átszámítását.

A másik módszer, hogy a hivatkozást megelőzi a @ jel, ez kikényszeríti az átszámítást.

Táblázatunkat nemcsak egy másik táblázattal kapcsolhatjuk össze, hanem egy adatállománnyal, vagy egy grafikus kerettel is. A cellákat áthelyezhetjük, vagy másolhatjuk a munkalap egy más pontjára, mégpedig külön-külön a benne levő értéket, vagy a formulát.

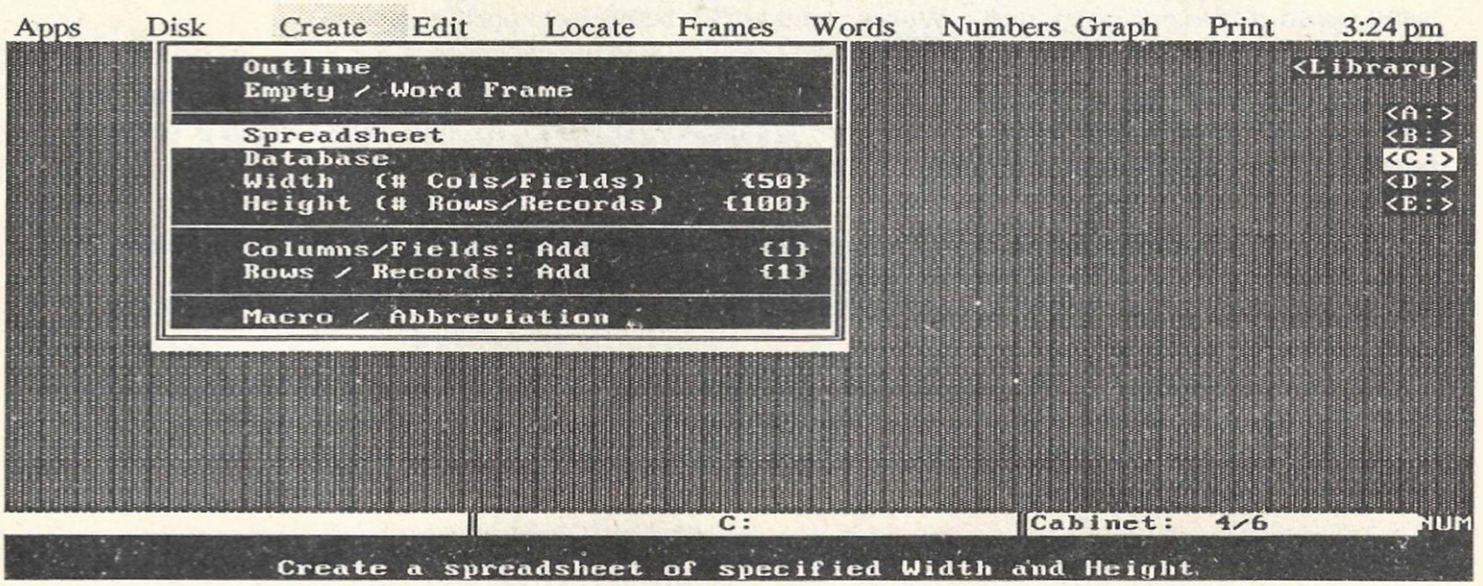
Ha formulát mozgatunk, vagy másolunk, lehetnek benne relatív hivatkozások, ekkor FRAMEWORK II az új helyre illeszti a hivatkozásokat, de lehetnek benne abszolút hivatkozások, ekkor nem illeszt (**C5** relatív, **C5** abszolút, **$C5** vegyes hivatkozás).

Az elkészített táblázat kinyomtatásában megint csak a **Print** menü van a segítségünkre ugyanazokkal a lehetőségekkel, mint a szöveges keretek nyomtatásakor. A kinyomtatott munkalap teljes egészében megegyezik a képernyőn láthatóval, a zérus szélességű oszlopok a nyomtatásban sem jelennek meg.

Példa: Hozzunk létre egy 5 oszlopból és 5 sorból álló táblázatot, töltsük fel értékekkel, majd állítsuk be az oszlopok kijelzési formátumát az alábbiak szerint:

- az A oszlop egész számként jelenítse meg az értékeket,
- a B oszlop 4 tizedesjeggyel,
- a C oszlop az elérhető legnagyobb pontossággal,
- a D oszlopban írassuk ki a valuta szimbólumot is,
- az E oszlopban exponenciális kiíratást alkalmazzunk.

Lépjünk be a **Create** menübe, állítsuk a **Width (# Cols/Fields)** menüpont segítségével az oszlopok számát 5-re, majd a **Height (# Rows/Records)** menüpont segítségével a sorok számát tudjuk 5-re állítani, ezután válasszuk ki a **Spreadsheet** opciót.



Ezzel létrehoztuk a táblázatunkat, adjunk neki nevet: Kinézet.

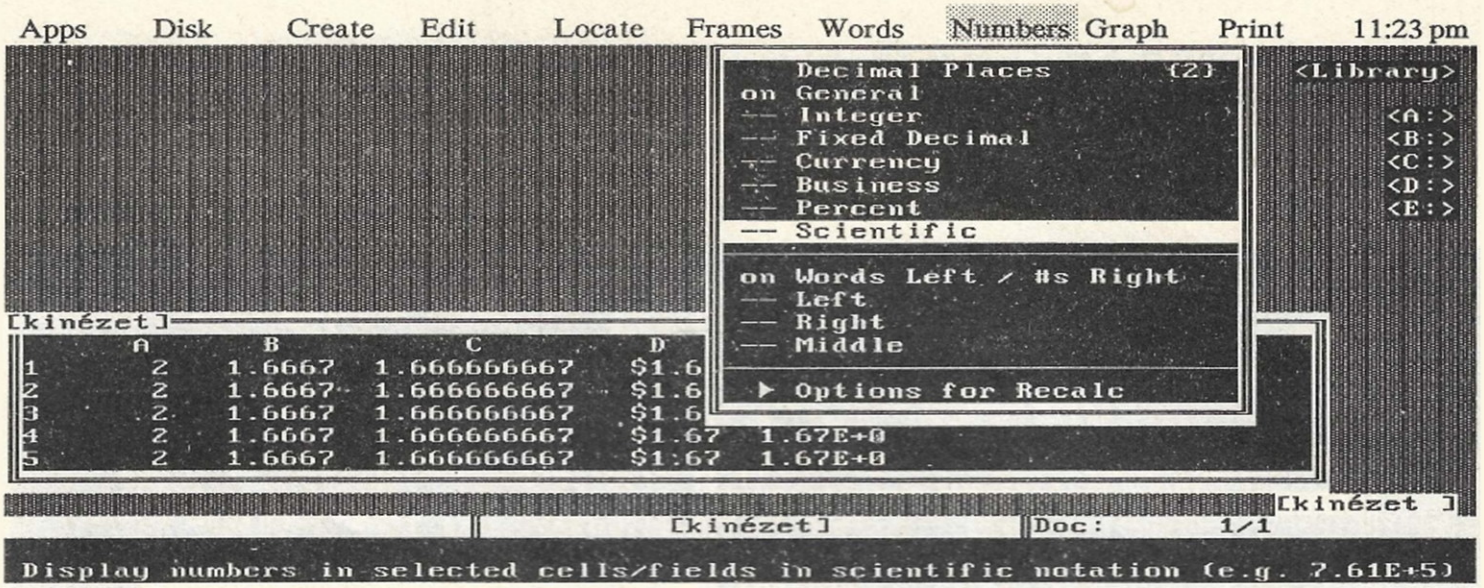
Jelöljük ki az **A1:A5** tartományt, lépjünk a **Numbers** menübe (**Ctrl+N**), válasszuk ki az **Integer** menüpontot. Vigyük a kijelölést a B oszlopra, majd a **Numbers** menüben állítsuk be a tizedesjegyek számát 4-re.

A C oszlopban az elérhető legnagyobb pontossággal íratjuk ki az értékeket, ezt ismét a **Decimal Places** menüpontnál érhetjük el. Mivel az alapértelmezett oszlopszélesség kicsinek bizonyult, az **F4** billentyű lenyomása után állítható a szélesség.

A D oszlop a **Currency** menüpont segítségével a pénznem szimbólumot is tartalmazni fogja.

Végül az E oszlop a **Scientific** menüpont hatására exponenciális formát ölt.

A végleges formátum:



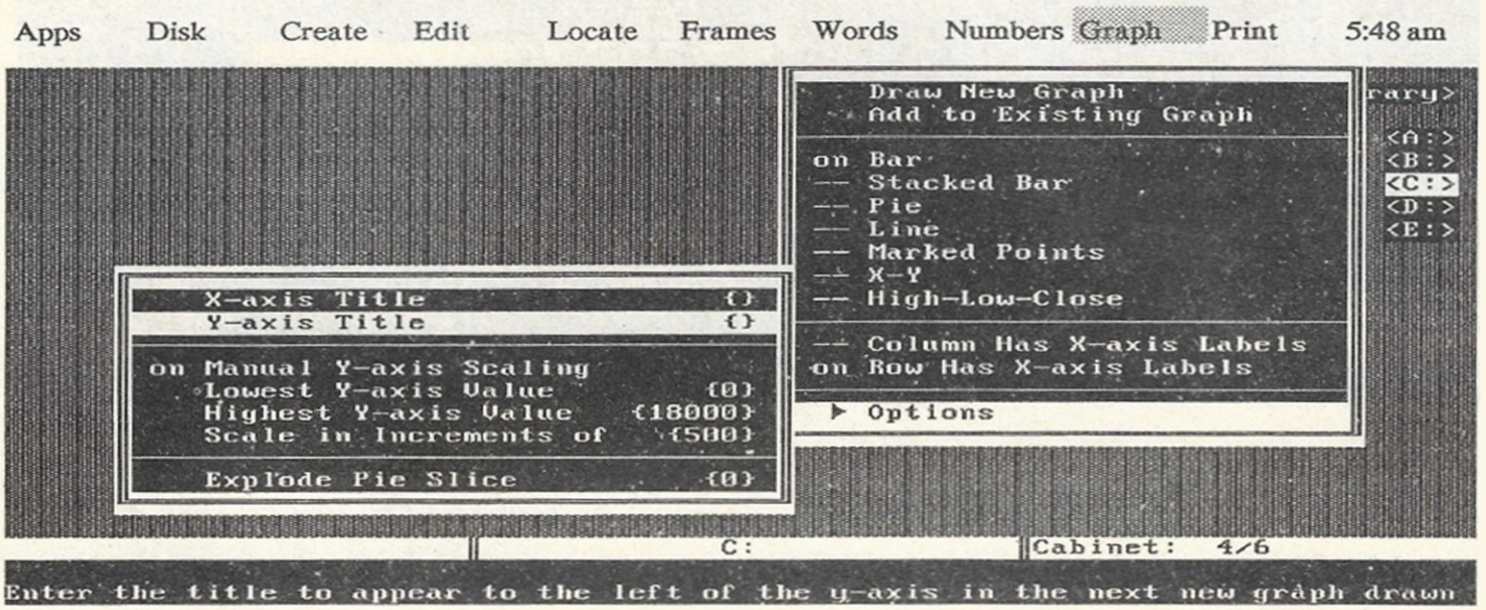
Mozgás a munkalapon

Billentyű	Hatása	CTRL + billentyű
KURZOR FEL	egy cellával feljebb	első adatsorra
KURZOR LE	egy cellával lejjebb	utolsó adatsorra
KURZOR JOBBRA	egy cella balra	első adatoszlopra
KURZOR BALRA	egy cella jobbra	utolso adatoszlopra
HOME	a sor első cellája	bal felső sarokba
END	a sor utolso cellája	jobb alsó sarokba
PgUp	lapozás fel	oszlop első cellája
PgDn	lapozás le	oszlop utolsó cellája
TAB	adatbevitel befejezése, egy cellával jobbra	-----
ENTER	adatbevitel befejezése, egy cellával balra	-----

5.1.5 Grafikai modul

A FRAMEWORK II grafikai modulja lehetővé teszi a munkalapon, vagy adatállományban tárolt adatok közti kapcsolatok ábrázolását. A grafikon készítése menüvezérelt, egyszerű, gyors és kényelmes.

A Graph menüben **hétféle** grafikontípus közül választhatunk:



Az elkészült grafikonok szerkeszthetők, újra rajzolhatók, léptékük és méretük változtatható, sőt már meglevő grafikonra másikat is rárajzolhatunk.

Grafikon és munkalap összekapcsolható úgy, hogy minden, a grafikonon ábrázolt adat változása a grafikon újrarajzolását eredményezi.

A grafikon előállításának lépései:

- az ábrázolni kívánt **adatokat** kiválasztjuk
- a **Graph** menü **Options** almenüjének a segítségével megtervezzük a léptéket, nevet adunk a grafikonnak és a tengelyeknek
- a **Draw New Graph** opció választásával megrajzoltatjuk a grafikont, ha már létezőre akarunk egy másikat rárajzolni, akkor választjuk az **Add to Existing Graph** opciót.

Megválaszthatjuk, hogy az **X** tengely nevét és beosztását az első oszlop vagy az első sor adatai határozzák meg. Alapértelmezésben az **X** tengelyen az első sor adatai jelennek meg, míg az **Y** tengelyt a FRAMEWORK II az ábrázolandó adatok nagyságrendjét figyelembe véve automatikusan skálázza.

Ha az automatikus beosztást nem tartjuk megfelelőnek, áttérhetünk a kézi skálázásra, ekkor meg kell adni a legkisebb és legnagyobb skálaértéket, valamint a léptéket.

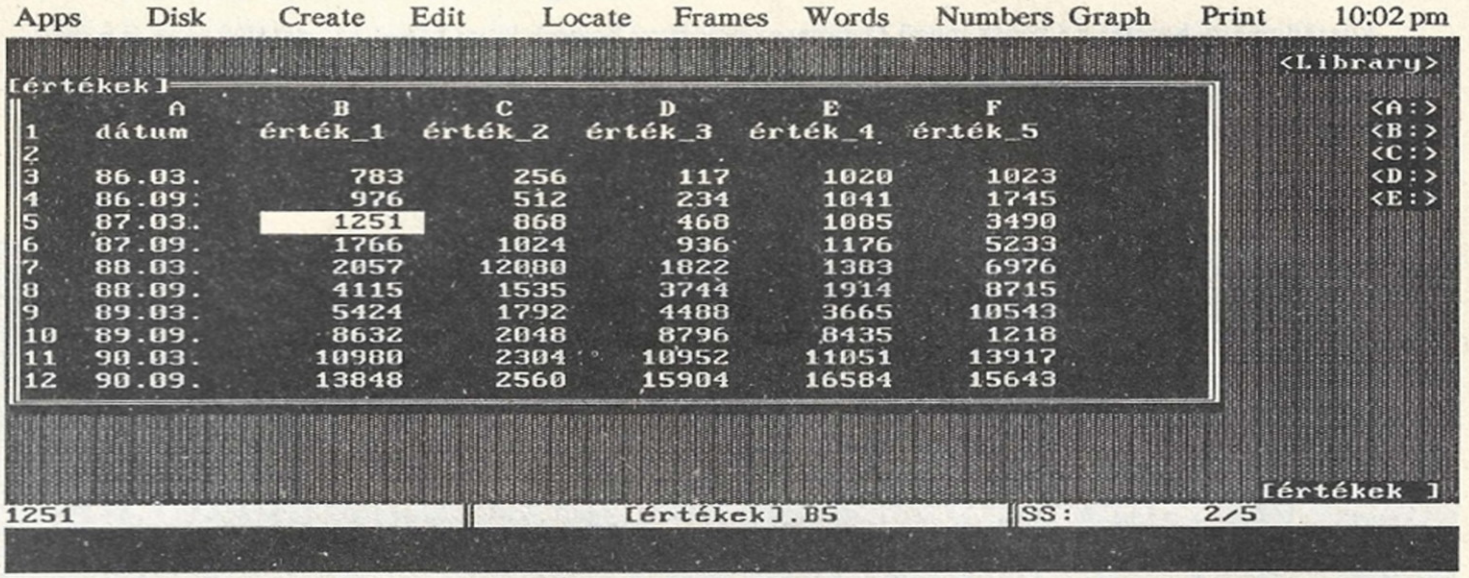
A grafikonok mindig külön keretbe kerülnek, melynek már létező, de üres keretnek kell lennie, vagy a program hozza azt létre.

A keretnek adott név lesz a **grafikon neve** is. Ha nem grafikus kártyát használunk, a grafikon karakterekből lesz kirajzolva. **CGA**, vagy **EGA** kártya használata esetén színesben látható a grafikon.

Papíron mindig a nyomtató, vagy a plotter maximális felbontásában jelenik meg az ábra. Az adatok és grafikus megjelenítésük dinamikus összeköttetésbe hozható a munkalap típusú keret és a grafikus keret összekapcsolásával, azaz a munkalap adataiban bekövetkező bármilyen változás hatására újrarajzolódik az ábra is.

Ahhoz, hogy ezt a hasznos lehetőséget ki tudjuk használni, mindkét keretnek **nyitva** az asztalon kell lennie. Célszerű ezért az összekapcsolt kereteket egy **gyűjtőkeretben** tárolni és együtt elmenteni.

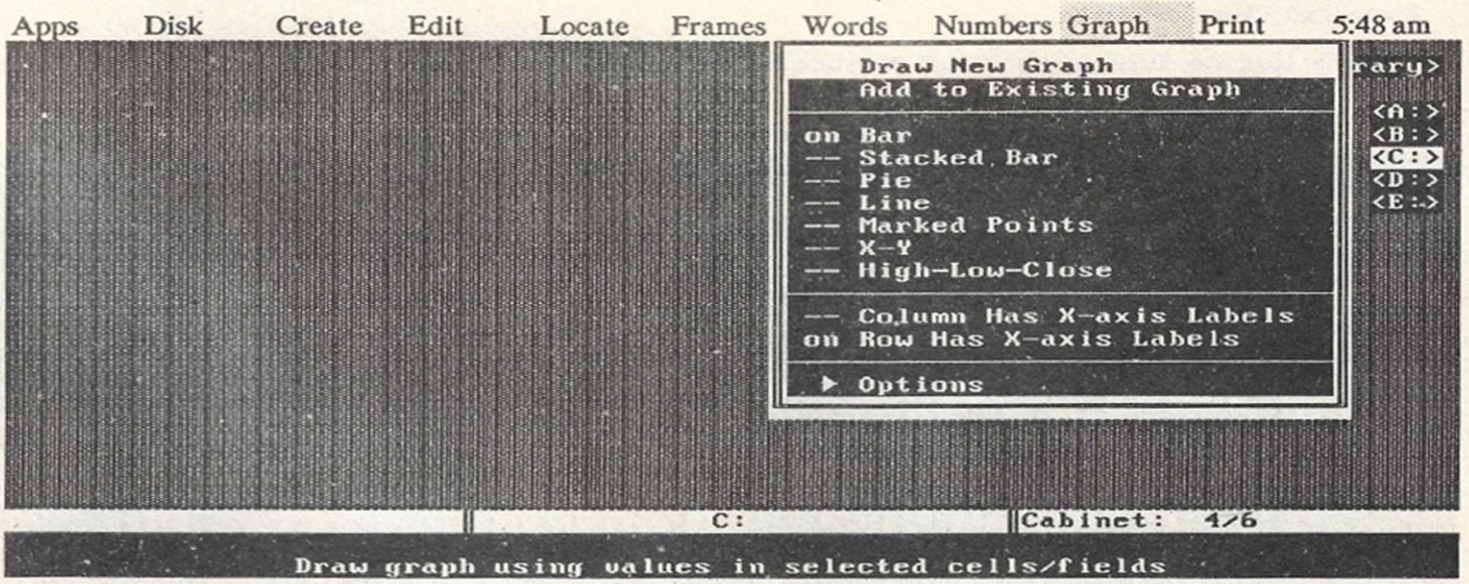
Példa: Hozzunk létre egy táblázatos keretet, mely 6 oszlopból és 12 sorból áll, legyen a neve ÉRTÉKEK, töltsük fel adatokkal az alábbiak szerint:



	A	B	C	D	E	F
1	dátum	érték_1	érték_2	érték_3	érték_4	érték_5
2						
3	86.03.	783	256	117	1020	1023
4	86.09.	976	512	234	1041	1745
5	87.03.	1251	868	468	1085	3490
6	87.09.	1766	1024	936	1176	5233
7	88.03.	2057	12080	1822	1383	6976
8	88.09.	4115	1535	3744	1914	8715
9	89.03.	5424	1792	4488	3665	10543
10	89.09.	8632	2048	8796	8435	1218
11	90.03.	10980	2304	10952	11051	13917
12	90.09.	13848	2560	15904	16584	15643

Lépjünk a **B1** cellára, nyomjuk le az **F6** billentyűt, majd vigyük a kurzort a **B12** cellára és nyomjuk meg az **ENTER** billentyűt. Ezzel kiválasztottuk a B oszlop értékes elemeit, ezt fogjuk **oszlop-diagramon** ábrázolni.

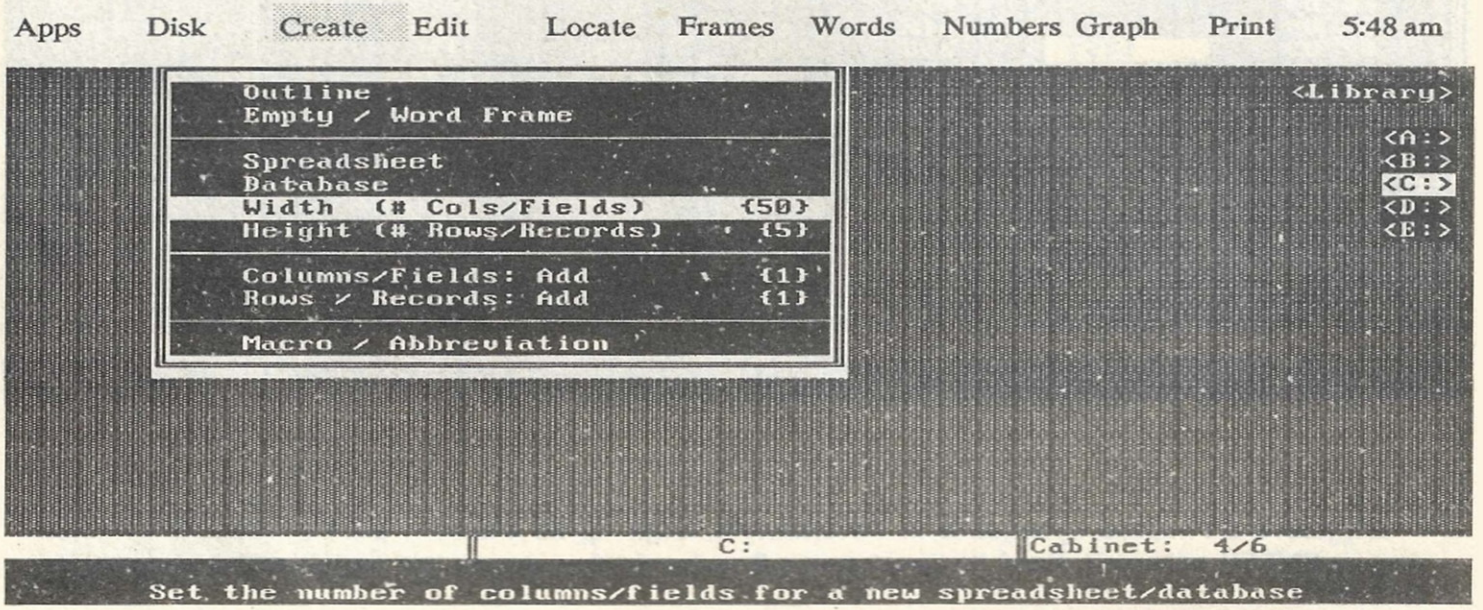
Lépjünk a **Graph** menübe (**Ctrl-G**), az alapértelezéseket változatlanul hagyva válasszuk ki a **Draw New Graph** menüpontot. Ekkor a rendszer rákérdez, hogy új keretet hozzon létre, vagy egy már meglevő grafikus keretbe helyezze a rajzot. Az **ENTER** billentyű lenyomására az új keretbe helyezést választhatjuk ki.



5.1.6 Adatállománykezelő modul

Az adatállománykezelő információk rendszerezett tárolását és gyors visszakeresését teszi lehetővé.

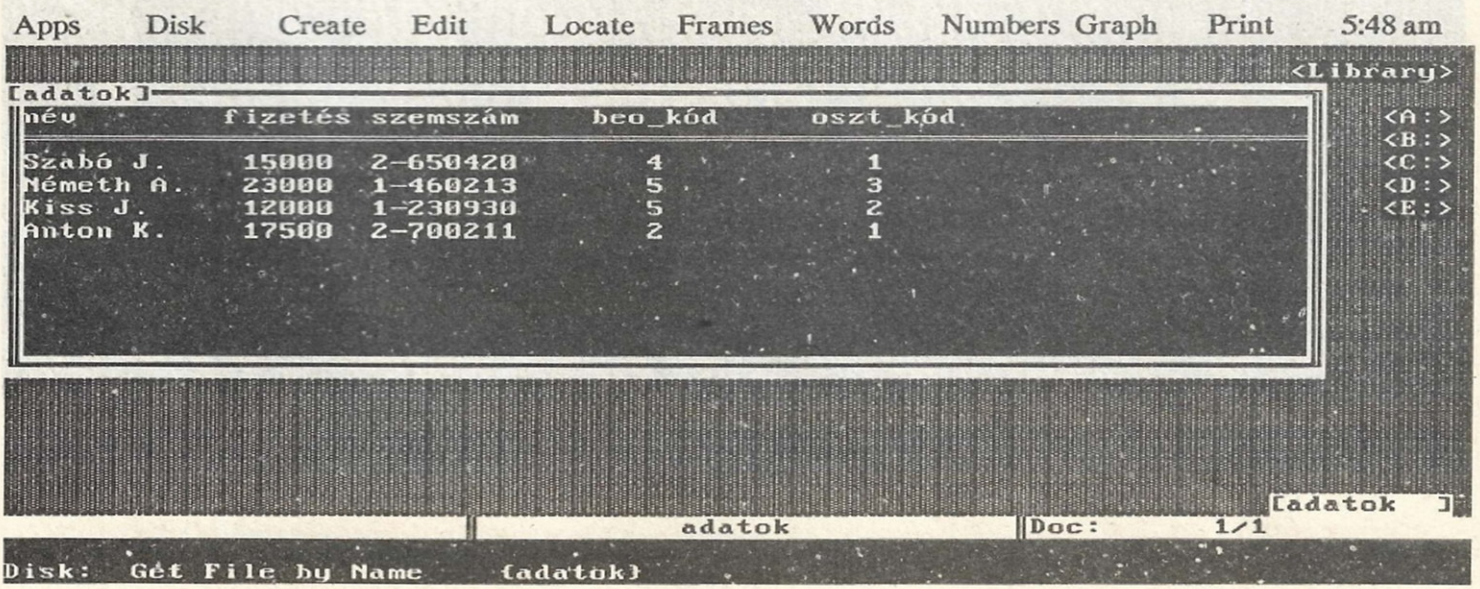
Adatállomány keretet a **Create** menü Database opciójával hozunk létre. Mivel az adatállomány is a memóriában helyezkedik el, ezért méretét a számítógép memóriája korlátozza. Létrehozáskor alapesetben **100** rekordból álló, rekordonként **50** mezőt tartalmazó keretet kapunk, ezt a **Create** menü **Width** illetve **Height** opcióival befolyásolhatjuk.



Munka közben is változtathatunk a méreteken, növelhetünk a **Create** menü **Column/Fields: Add**, illetve a **Rows/Records: Add** opciókkal, csökkenthetünk az **Edit** menü **Column/Fields: Remove**, illetve a **Rows/Records: Remove** opcióival.

Mindig csak akkora állományt érdemes használni, amekkorára szükségünk van, így gazdaságosabban használhatjuk a memóriát.

Példa: Hozzunk létre egy táblázatot, melynek **5** mezője és **50** rekordja van, a neve legyen **Adatok**.

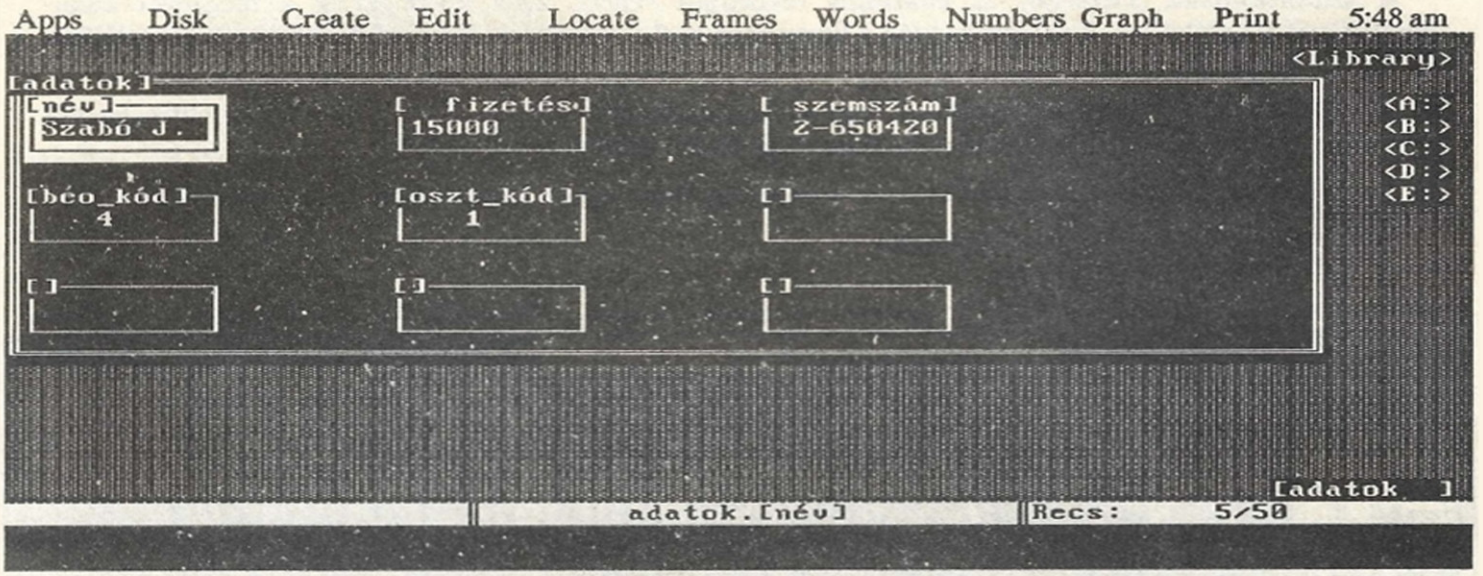


Lépjünk a **Create** menübe, állítsuk be a megfelelő értékeket, majd hozzuk létre az állományunkat a **Database** menüponttal. Ha sikerült, vigyünk fel néhány adatot, hogy kipróbálhassuk az adatállománykezelés lehetőségeit.

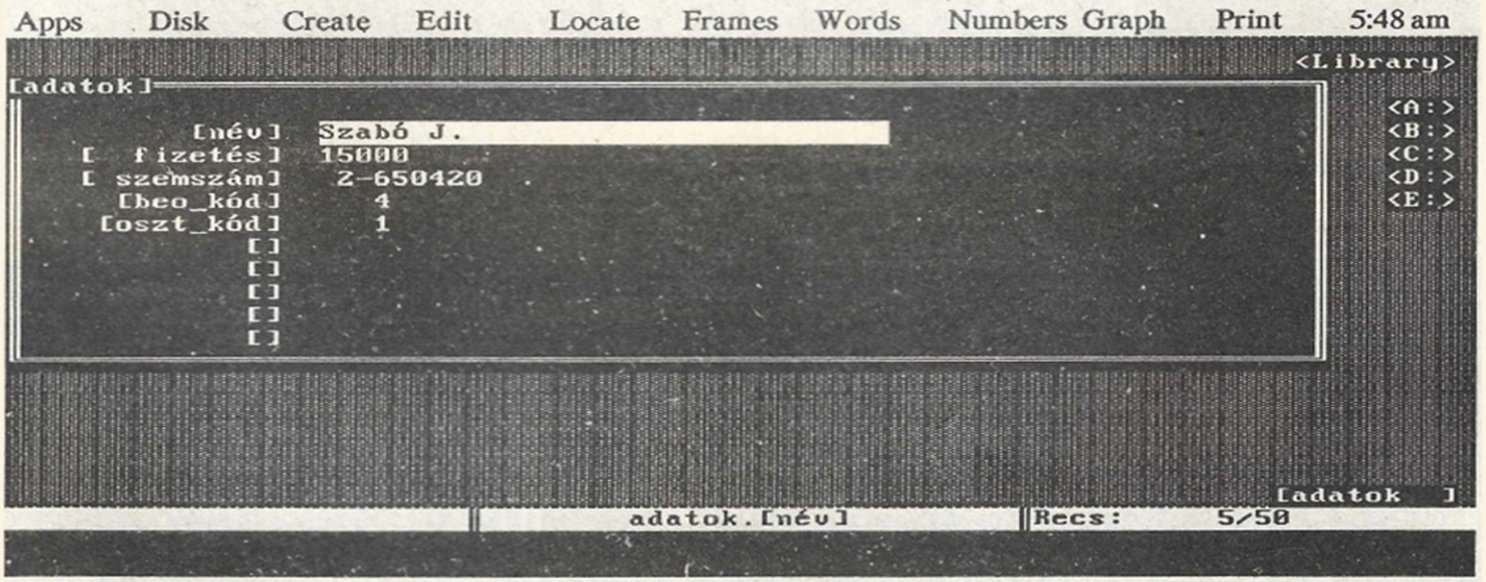
Adatállományainkat **három** megjelenítési formában láthatjuk viszont a képernyőn, az első a táblázatos megjelenítés:

A munkalaphoz hasonlóan sorokba és oszlopokba vannak rendezve az adatok. A sorokat **rekordoknak**, a sorokban minden bejegyzést **oszlopnak** nevezünk, minden oszlop azonos típusú információt tartalmaz (előző lapon).

A második megjelenítési forma az **űrlapformátum**, ahol egyszerre egy rekord adatai vannak a képernyőn, az egyes mezők tulajdonképpen külön keretek, melyek helyzete szabadon változtatható. Főleg akkor alkalmazzuk, ha adott formanyomtatványokról viszünk fel adatokat.



A harmadik az ún. **dBASE** formátum, ami megegyezik a dBASE Edit, vagy Append parancsánál látható beviteli képernyőképpel: egyszerre csak egy rekord látható, a mezők egymás alatt helyezkednek el.



A három megjelenési forma között az **F10** billentyűvel kapcsolgathatunk. Mint már tapasztalhattuk, az adatállomány-keretek és a munkalap-keretek kezelése nagyon hasonló.

A mezők szélességét az **F4** és a kurzormozgató billentyűkkel állíthatjuk be, adatok mozgatása/másolása az **F7/F8** billentyű és a kurzormozgató billentyűk segítségével történik. Az adatok megjelenítésén a **Words** és a **Numbers** menükkel változtathatunk.

Az adatbevitel is hasonló a táblázatkezelő moduléhoz, ugyanolyan típusú adatokat tudunk bevinni az ott megismert szabályok szerint. Két fő formulatípust használhatunk:

- **definiáló** és
- **szürő** formulát.

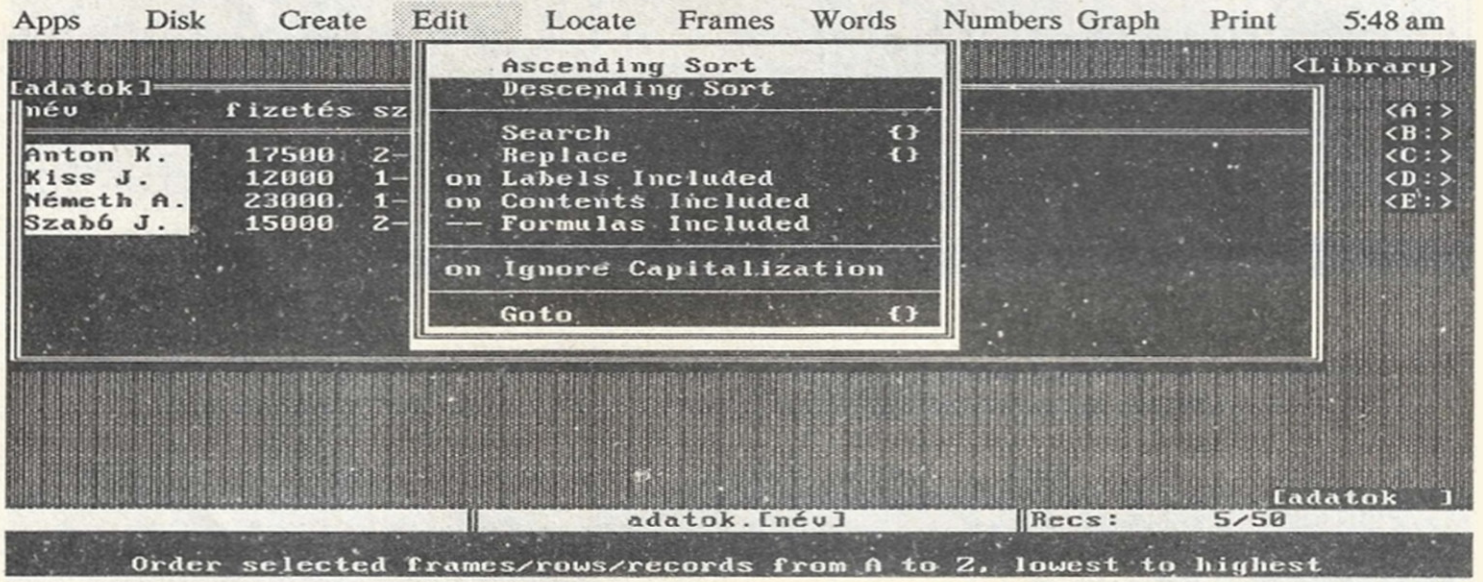
A **definiáló** formulák közé tartoznak az alapműveletektől a bonyolult függvénykapcsolatokig a táblázatkezelőnél megismert formulafajták. Az adatállomány fejrészébe, a **mezőnév** mögé kell beírnunk a formulánkat, ez olyan, mintha az oszlop **minden** mezőjébe ugyanazt a formulát írtuk volna. Ha a formulát csak egy adott mező mögé írjuk, akkor az csak a mezőt tartalmazó rekordon belül fejti ki a hatását.

A szűrőformulák (**filterek**) az adatbázis rekordjait szűrik, azaz kiválogatják a megadott szempont(ok)nak megfelelően. A válogatás után a képernyőn csak a feltételeknek megfelelő rekordokat találjuk, ez lesz az állomány **aktív** része.

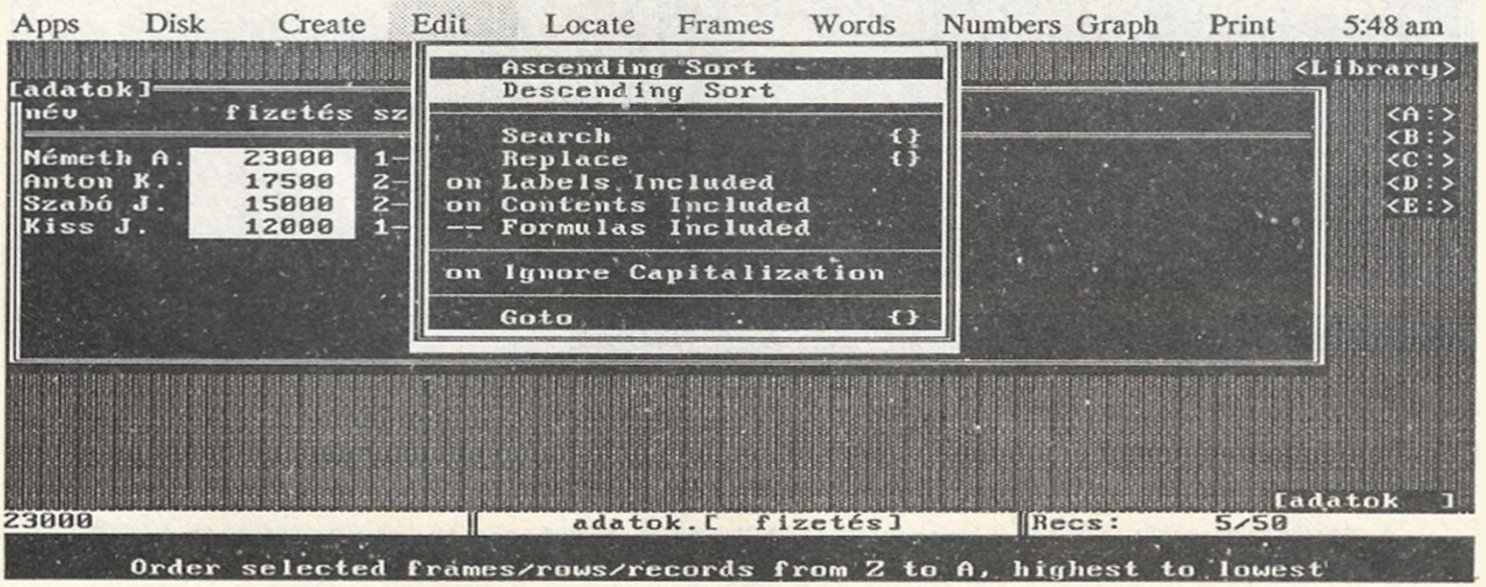
A **státuszpanel** jobb oldalán olvasható, hogy az összes rekord közül mennyi az aktív. A szűrt állományból bármikor visszatérhetünk a teljes állományhoz, vagy újabb szűrést is végezhetünk.

Adatainkat rendezhetjük a **Locate** menü **Ascending Sort**, vagy a **Descending Sort** utasításaival.

Növekvő rendezés a **Név** oszlop alapján



Csökkenő rendezés a **Fizetés** oszlop alapján



Az adatok keresésére a már megismert **Search** utasítást használjuk, lehetőség van a megtalált adatok cseréjére is (**Replace**).

A dBASE II, vagy dBASE III állományok a FRAMEWORK II-be közvetlenül **behívhatók**, a FRAME-'WORK II-ben létrehozott adatállományok pedig **konvertálhatók** a dBASE formátumra.

Kinyomtatásánál a táblázatkezelőnél megismert mód szerint járhatunk el.

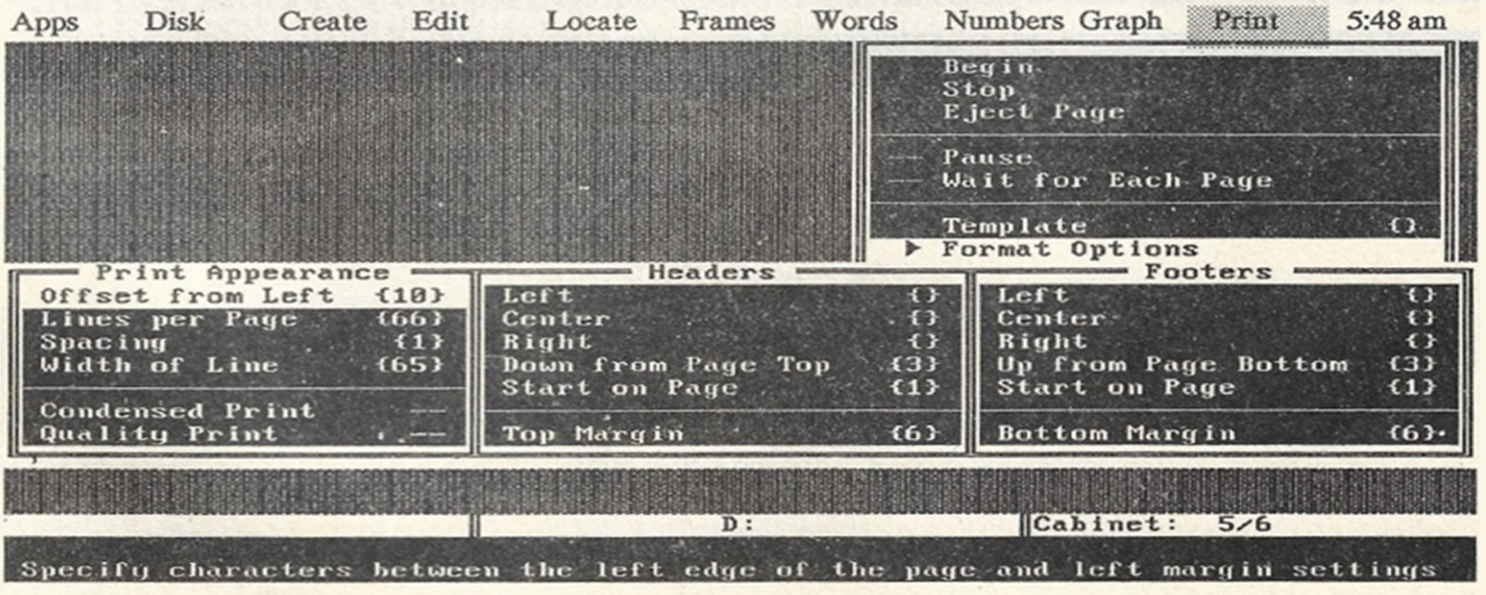
Mozgás az adatkezelőben

BILLENTYŰ	Hatása
KURZOR FEL	felette lévő mezőre lép
KURZOR LE	alatta lévő mezőre lép
KURZOR BALRA	balra lévő mezőre lép
KURZOR JOBBRA	jobbra lévő mezőre lép
HOME	rekord első mezője
END	rekord utolsó mezője
PgUp	előző rekord
PgDn	következő rekord

5.1.7 Nyomtatás

A FRAMEWORK II teljesen menüvezérelten nyomtat, a **WYSIWYG** elv alapján.

Nyomtatási paramétereket a PRINT menübe lépve állíthatunk be.



Bármilyen típusú keretet kinyomtathatunk, a nyomtatás minősége a használt nyomtatótól függ.

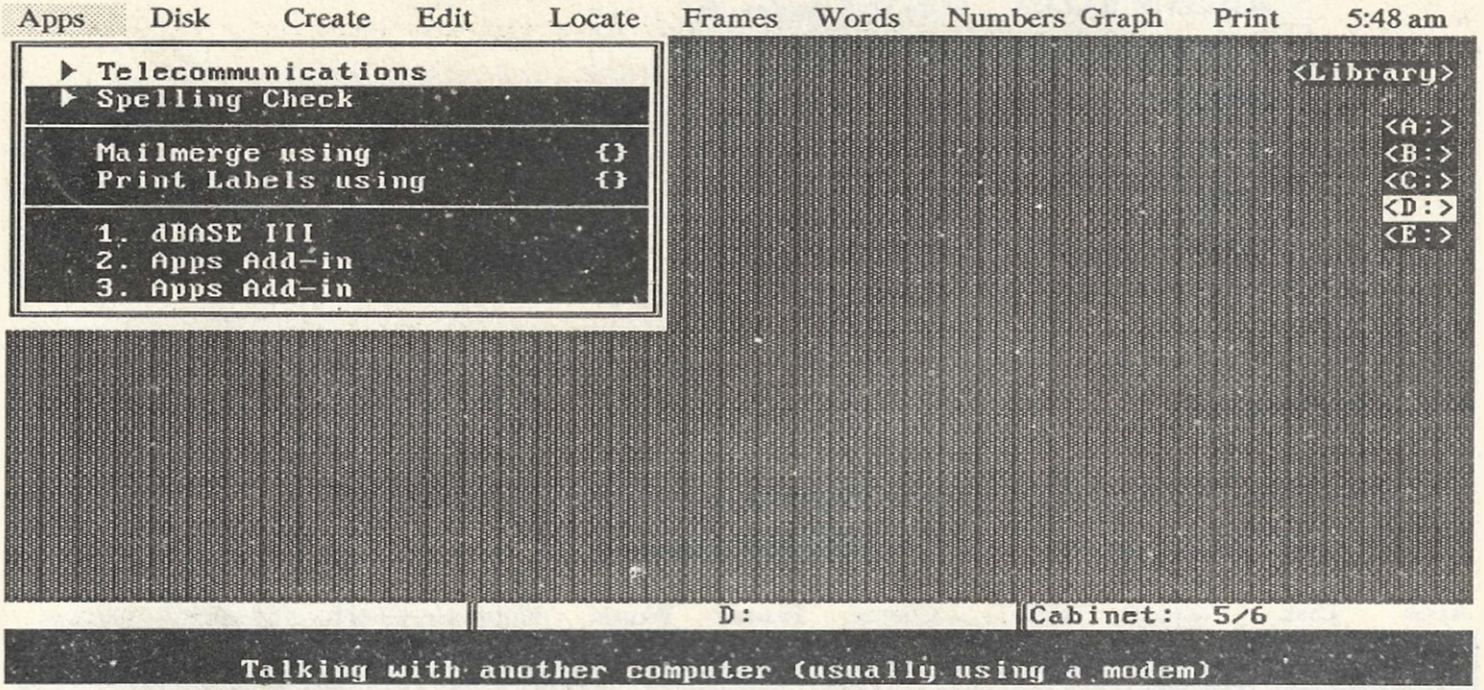
Kinyomtathatjuk a teljes dokumentumunkat, ekkor álljunk a gyűjtőkeret szegélyére és **Contents View** módban nyomtassunk. Ekkor az összes befoglalt, nyitott állapotban levő keret tartalma kinyomtatódik. Ha csak egy keret tartalmát akarjuk nyomtatni, a keret szegélyéről indítsuk a nyomtatást.

Ha valamely keret tartalmának csak egy részét kívánjuk kinyomtatni, jelöljük be a kívánt részt, majd indítsuk a nyomtatást. A nyomtatás ún. spooling üzemmódban folyik, vagyis a lemezre kerülnek a kinyomtatandó adatok egy átmeneti állományba és onnan nyomtat a rendszer, miközben mi zavartalanul dolgozhatunk tovább.

A FRAMEWORK II más modulokban megismert nyomtatási lehetőségein túl használhatjuk a **Mailmerge** és **Print Labels** funkciókat is.

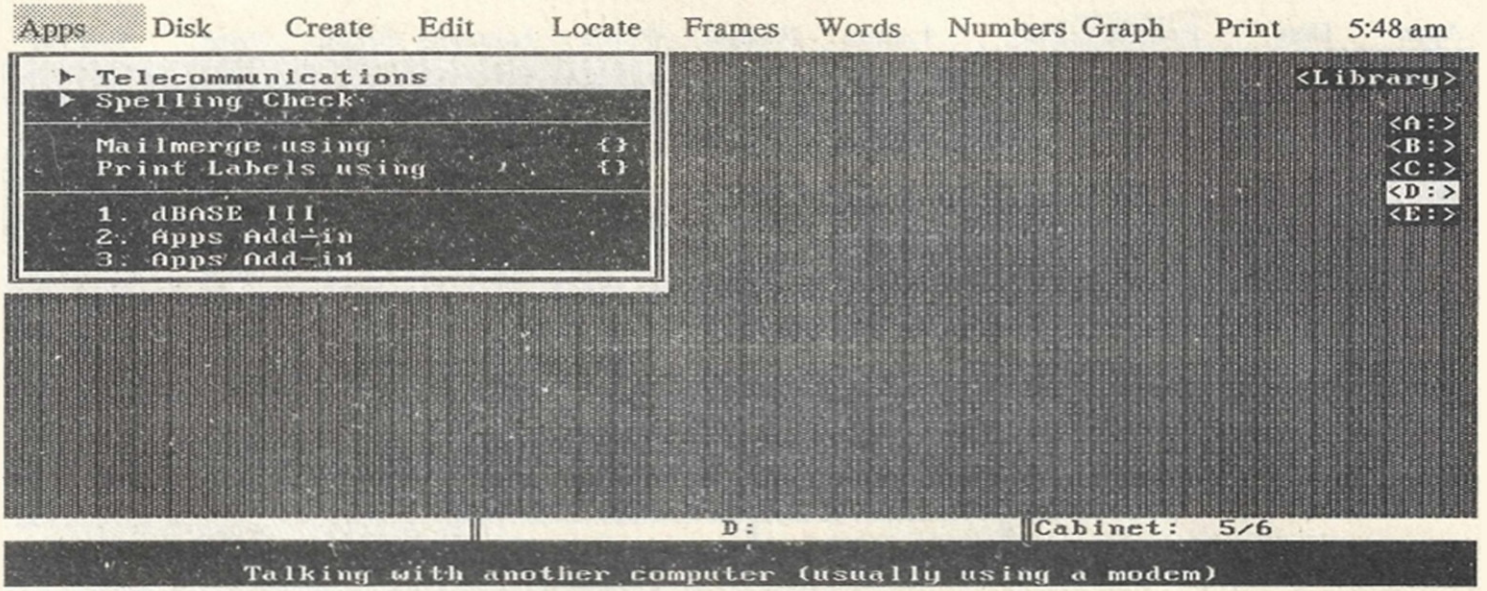
A **Mailmerge** (körlevél) szolgáltatás lehetővé teszi, hogy ugyanaz a dokumentum különböző címzettekhez kerüljön, anélkül, hogy mindig át kellene írnunk a címzett megváltozása miatt. A változó adatokat tartalmazó részeket ún. **markerekkel** kell megjelölnünk, az adatokat egy adatállományban tartjuk. Az **Apps** menü **Mailmerge** using opcióját választva megadjuk ennek az adatállománynak a nevét, majd a FRAMEWORK II onnan veszi az adatokat a megfelelő számú és a megfelelő helye(ke)n különböző tartalmú dokumentumok előállításakor.

A **Print Labels** (borítékcímzés) funkció a körlevélhez kapcsolódik, az eljárás is hasonló az ottanihoz.



5.1.8 A FRAMEWORK II menüi

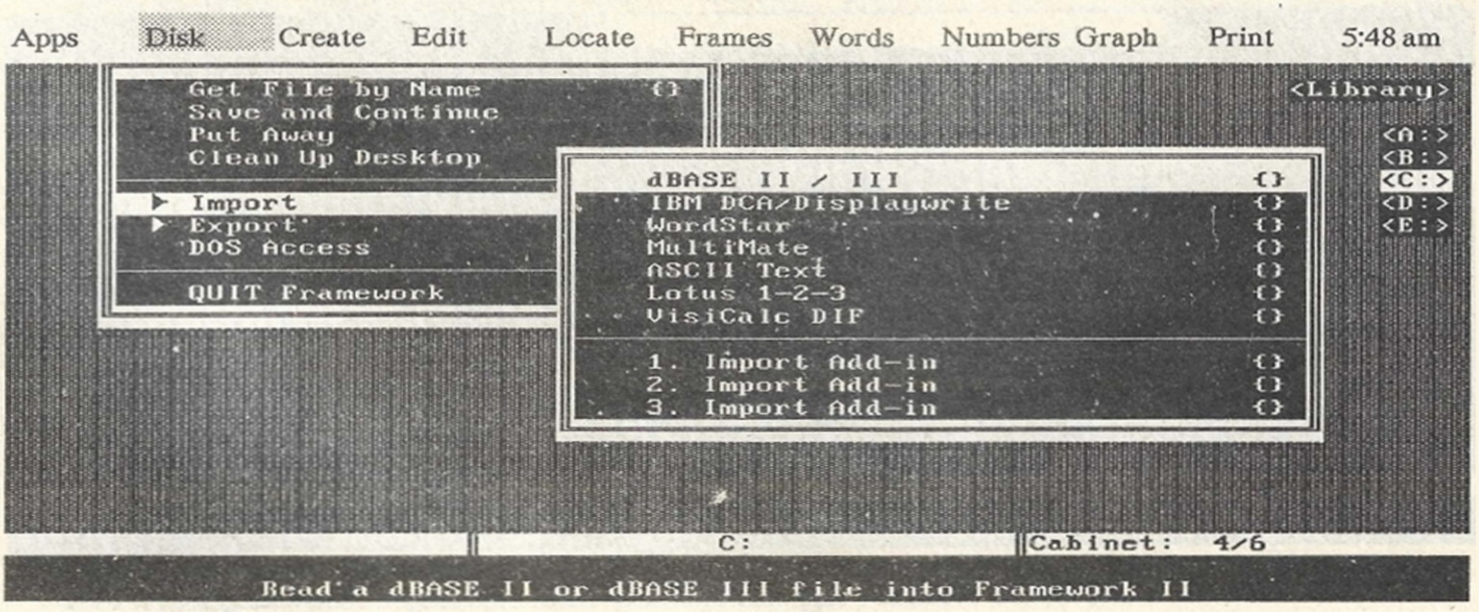
Applications menü



Az **Applications** menüben találjuk a távadatátviteli lehetőségeket, összeköttetést teremthetünk két PC között, vagy belépheţünk **on-line** információs rendszerekbe.

A **Spelling Check** almenü a helyesírási és betűzési parancsokat tartalmazza. Ebben a menüben találjuk még a körlevél és borítékcímzési funkciókat.

Disk menü



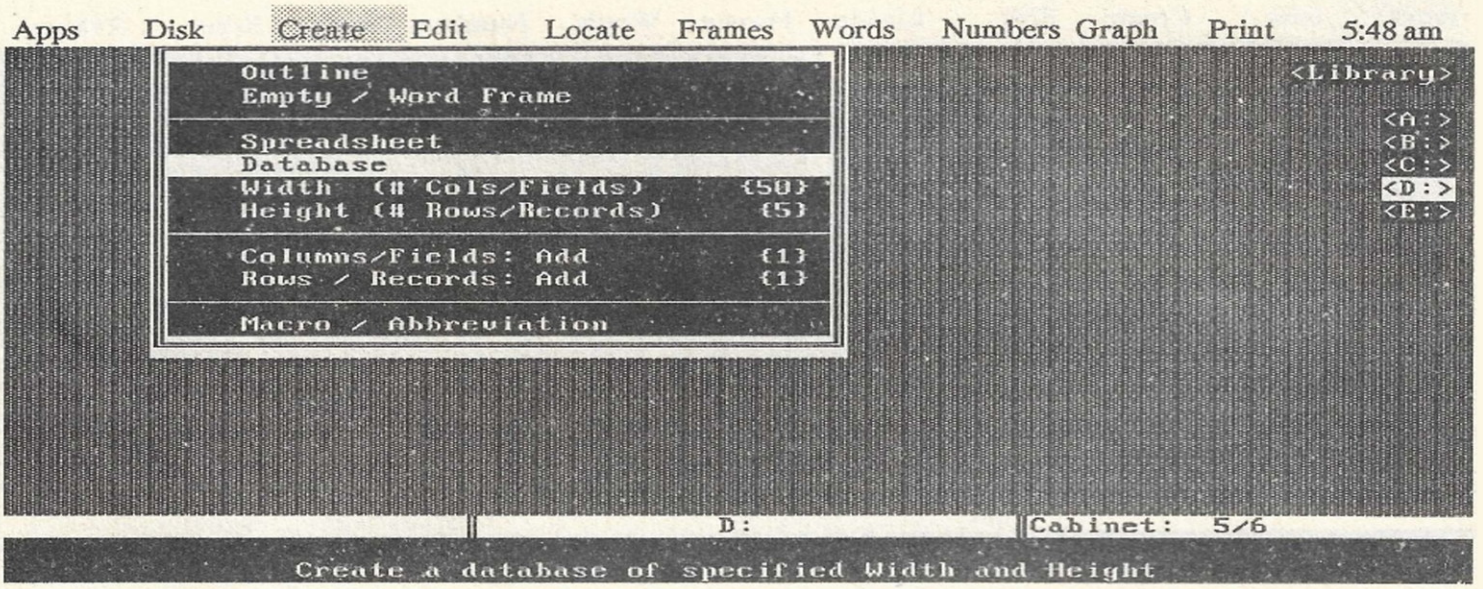
A **Disk** menüben lehetőségünk van névvel megadott keret betöltésére, a használt keret elmentésére, majd a munka folytatására, DOS parancsok használatára és itt tudunk kilépni a FRAMEWORK II-ből.

Disk-Import almenü Az Import almenü segítségével tudunk más típusú file-okból adatot behozni a FRAMEWORK II-be, pl. **dBASE** adatbáziskezelő állományaiból, **WordStar, Multi-**

Mate szövegszerkesztők állományaiból, **Lotus 1 2 3 és VisiCalc** táblázatokból. Lehetőség van normál szöveges állományok átvételére is (**ASCII text**).

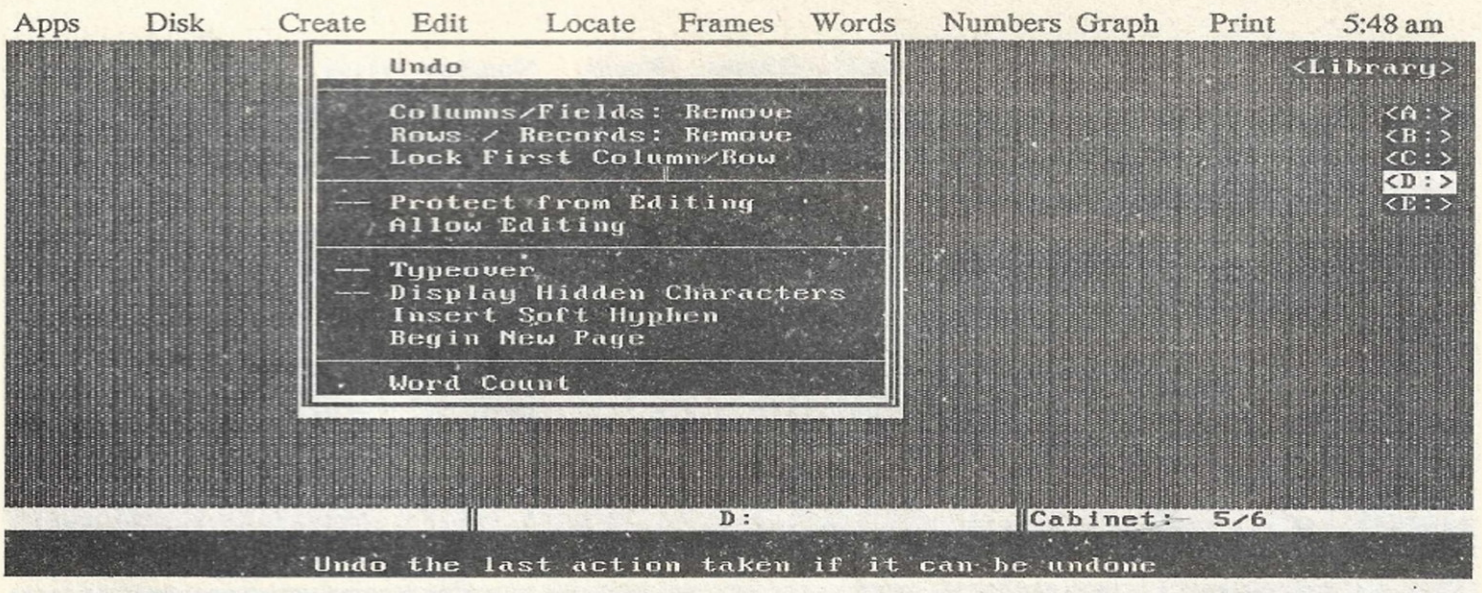
Disk-Export almenü Az **Export** almenü segítségével a FRAMEWORK II-ből tudunk adatokat átadni az előbb felsorolt típusú állományokkal dolgozó programoknak.

Create menü



A Create menüben hozhatunk létre **új** (vázlat, szöveges, táblázat és adatbázis típusú) **kereteket**, beállíthatjuk a táblázat-, vagy adatállomány keret szélességét (oszlopok, illetve mezők számát), magasságát (sorok, illetve rekordok számát), megadott számú új sort/rekordot, vagy oszlopot/mezőt szúrhatunk be.

Edit menü

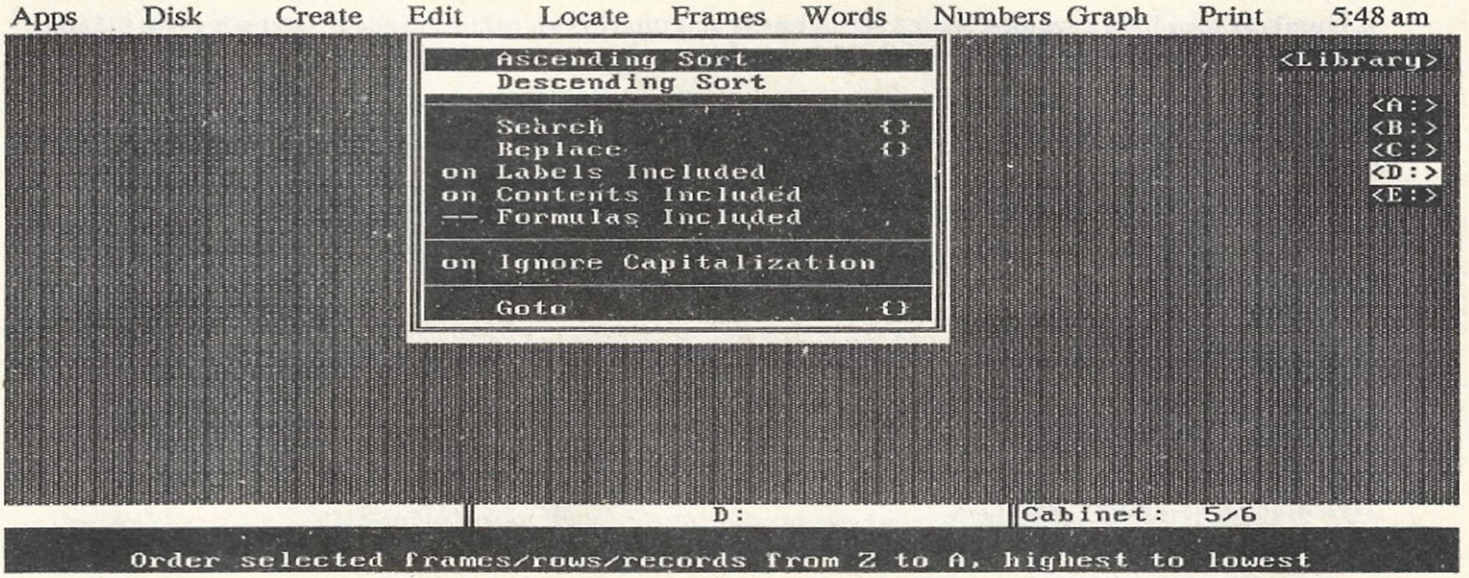


Az Edit menüben a **már létező** táblázat-, vagy adatállomány keret szélességét, illetve magasságát csökkenthetjük, beállíthatjuk a különböző védelmi módokat. Rejtett karaktereket láthatóvá tehetünk, "lágy" elválasztó jeleket tehetünk a szövegbe.

Kapcsolhatunk a beszúró, vagy felülíró üzemmód között, elhelyezhetünk lapdobás karaktert is a szövegünkben.

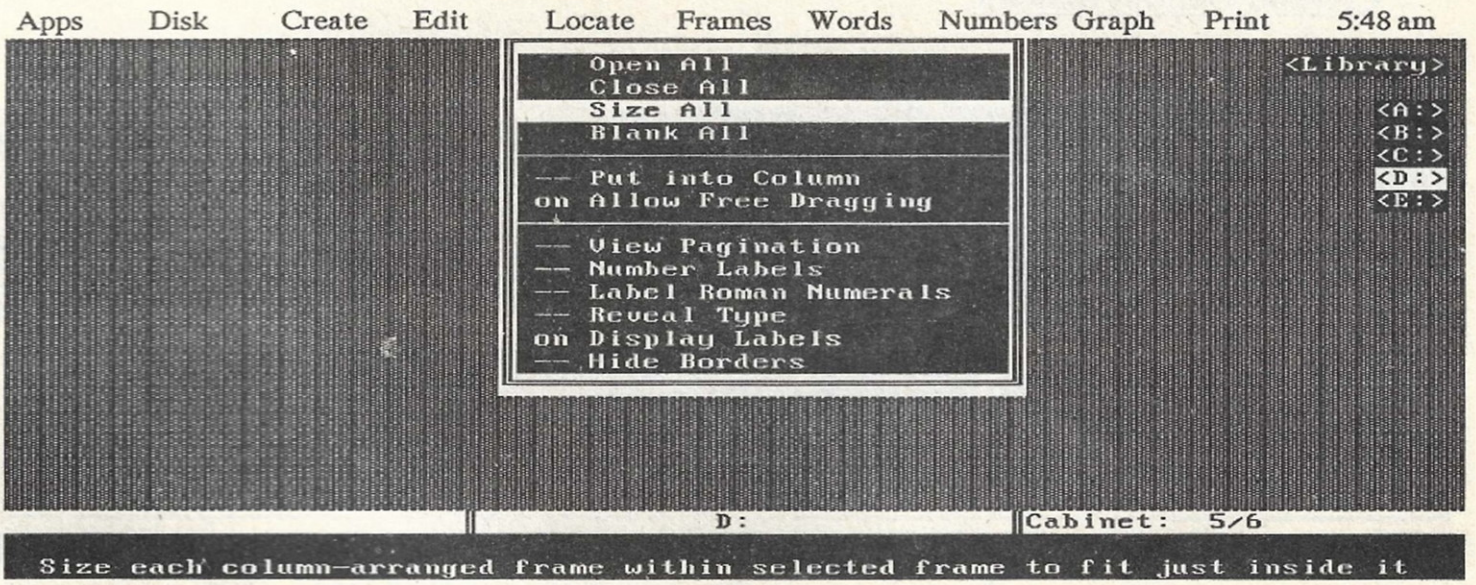
Lehetőségünk van egyes műveletek nem kívánt eredményének a megszüntetésére is (**Undo** parancs).

Locate menü



A Locate menüben végezhetjük el a **rendezéseket, kereséseket** és **cseréket**, beállíhatjuk a keresések és cserék kapcsolóit.

Frames menü



A Frames menüben a keretek **jellemzőit** állíthatjuk be, keretek megnyitását, zárását, ürítését végezhetjük innen.

Words menü

A Words menüben lehet beállítani a kiírási jellemzők egy részét, mint pl.:

- aláhúzott, dőlt, vagy vastag betűket,
- a szövegigazítási jellemzőket: balra, jobbra, középre, sor két szélére.
- állíthatjuk a bal és a jobb margókat,
- a bekezdés mélységét és a tabulátor méretét.

Numbers menü

A Numbers menüben a számok kiírási formátumát alakíthatjuk ki, beállíthatjuk a **tizedesjegyek** számát, a cellán belüli **igazítás**okat és az **újraszámítás** módját.

Graph menü

A kiválasztott adatok ábrázolásának indítása, a **grafikon** jellemzőinek beállítása.

Print menü

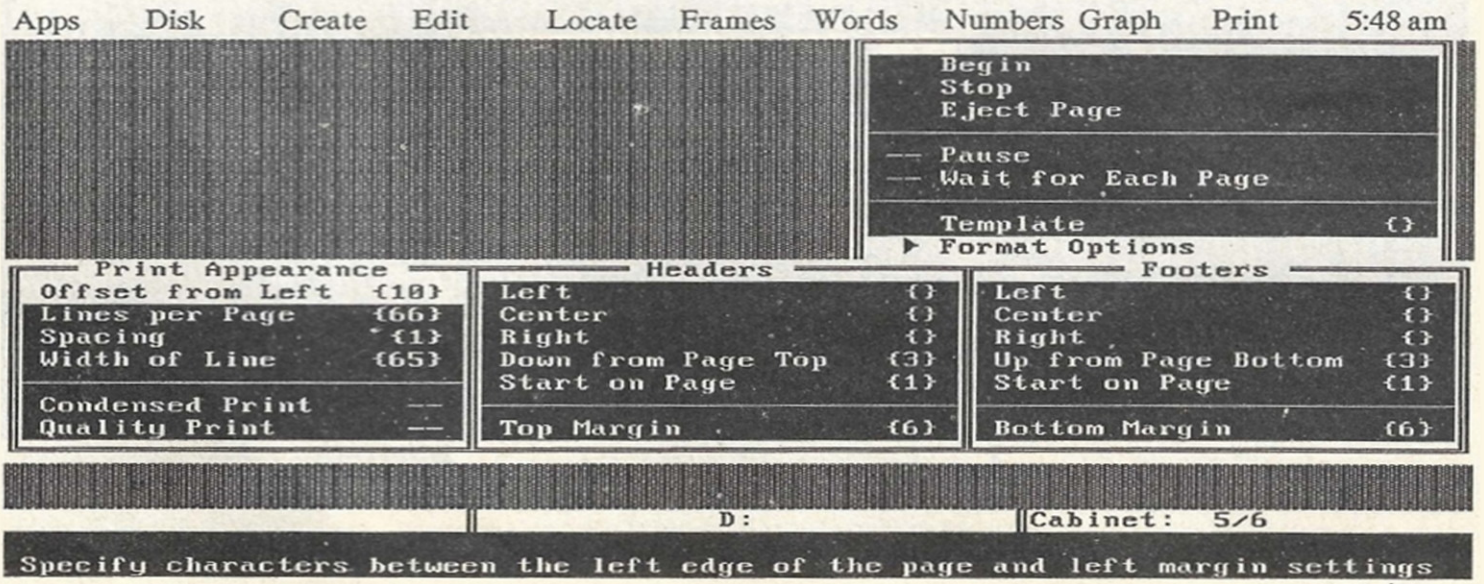
Nyomtatás indítása, befejezése, szüneteltetése, papírcsere elvégzésének lehetősége.

Format Options almenü

Nyomtatási formátum beállítása:

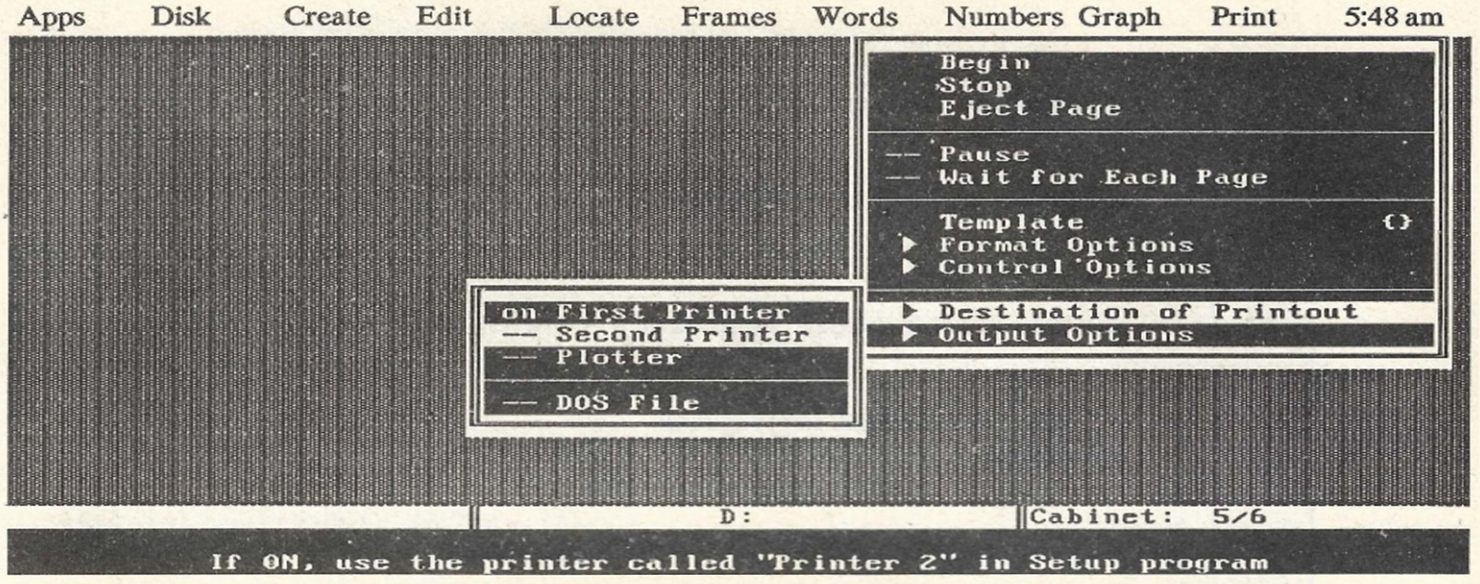
- bal **margó**,
- laponkénti **sorszám**,
- **sortávolság**,
- **sorszélesség**,
- nyomtatási **minőség** beállítása,
- **fejlécek**, láblécek készítése.

Controll Options almenü



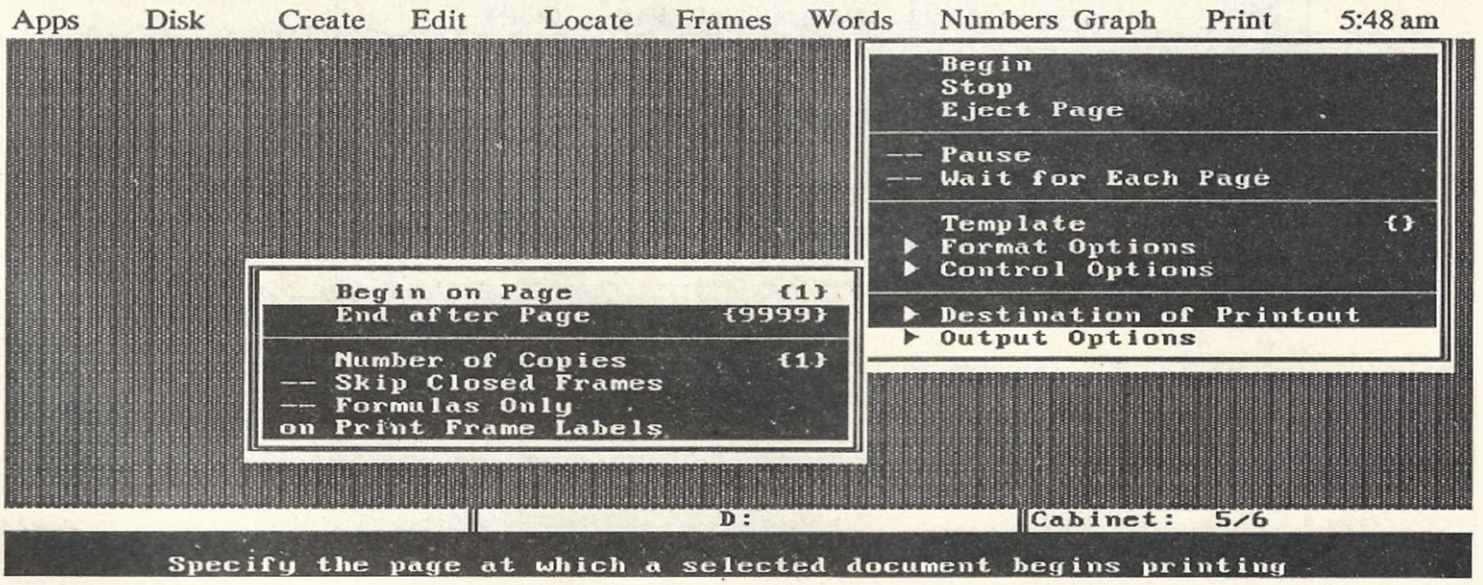
Keretek nyomtatását új oldalon kezdje-e, vagy sem, lapszámozás **befolyásolása**, nyomtatás előtt és után a nyomtatóra küldött vezérlőkódok beírása.

Destination of Printout almenü



Kiválaszthatjuk, hogy **printerre**, **plotterre** vagy **lemezfile**-ba írja az adatokat.

Output Options almenü



Oldalszámozás, példányszám beállítása.

5.1.9 A FRAMEWORK II funkcióbillentyűinek jelentése

F1	Helyzetfüggő segítség adása	**HELP**
F2	A képletek beírását kezdő billentyű	**EDIT FORMULA**
F3	Keretek, mezők helyének beállítására	**DRAG**
F4	Keretek, cellák, mezők nagyságának beállítására	**SIZE**
F5	Képletek újraszámítására	**RECALC**
F6	Szövegek, cellák, mezők megjelölésére	**EXTEND SELECT**
F7	Megjelölt elemek mozgatása	**MOVE**
F8	Megjelölt elemek másolása (duplikálás)	**COPY**
F9	Adott keret teljes képernyős megjelenítésére	**ZOOM**
F10	Megjelenítési módok közötti kapcsoló	**VIEW**

5.2 A FRAMEWORK II programozása

A FRAMEWORK II olyan programozási lehetőségeket nyújt, melyekkel felülmúlja vetélytársait. Lehetőségünk van az ismétlődő billentyűzési feladatokat makróutasításokra bízni, mely egyfajta automatizmust tesz lehetővé. Írhatunk képleteket keretek hátoldalára (szegélyére), számolótáblák celláiba, vagy adatállomány keretek mezőibe, mint azt az interaktív használatnál már láttuk.

Innen már csak egy lépés a FRAMEWORK II programozási nyelvén, a **FRED** nyelven programokat írni, melyek segítségével akár bonyolult alkalmazói feladatok is megoldhatók.

A FRED nyelvből a FRAMEWORK II valamennyi menüje, művelete elérhető, ill. az interaktív üzemmódban rendelkezésünkre állnak a **FRED** függvényei, parancsai.

5.2.1 Makrók a FRAMEWORK II-ben

Ismerkedjünk meg először a makrók létrehozásával és használatával.

A **makró** olyan billentyűsorozat, melyhez egy **nevet** rendelünk. Minden olyan műveletet tartalmazhat, melyet kézzel is el tudnánk végezni. Rendszer a **név beütése** után azonnal végrehajtja a makrót. Létrehozásukra, mint annyi minden más feladatnál, több módszer is a rendelkezésünkre áll.

Hozzunk létre egy makrót, mely a kurzor pozíciójától leírja a "FRAMEWORK II" szöveget, ezt a makrót rendeljük az **Alt-F** billentyűkhöz. Az első lehetőség a **CREATE** menü **Macro/Abbreviation** menüpontjának kiválasztása. A rendszer bekéri a makró nevét (nyomjuk le együtt az **ALT** és az **F** billentyűket), létrehoz egy üres, szöveges keretet, megnyitja és beleírja az általunk leütött billentyűk jelét (FRAMEWORK II).

Teszi ezt egészen a **Ctrl-Break** billentyűkombináció lenyomásáig, majd a keretet a **Library**-be helyezi. A későbbiekben az **Alt-F** billentyűket egyszerre lenyomva, lejátssza a tárolt leütéseket. A makrót a könyvtárban a saját nevén szereplő **alkeret** formájában találjuk meg, ezt a későbbiekben korlátozások nélkül módosíthatjuk.

Egy makrót tartalmazó keretben a normál billentyűk változatlan formában, a speciális billentyűk kapcsos zárójelek között szerepelnek. Az egyidejű leütést igénylő billentyűkombinációkat kötőjel kapcsolja össze.

5.2.2 A speciális billentyűk

Keretkezeléssel kapcsolatos billentyűk:

{uplevel}	**{ctrl-uplevel}**	**{dnlevel}**
{ctrl-dnlevel}	**{scroll-lock}**	**{esc}{ins}**
{ctrl-ins}		

Navigációs billentyűk:

{home}	**{ctrl-home}**	**{end}**	**{ctrl-end}**
{pgup}	**{ctrl-pgup}**	**{pgdn}**	**{ctrl-pgdn}**
{uparrow}	**{ctrl-uparrow}**	**{dnarrow}**	**{ctrl-dnarrow}**
{leftarrow}	**{ctrl-leftarrow}**	**{rightarrow}**	**{ctrl-rightarrow}**

Szerkesztő billentyűk:

{del} **{ctrl-del}** **{backspace}**

{ctrl-backspace} **{tab}** **{backtab}**

Funkcióbillentyűk:

{f1}..{f10}

{ctrl-f1}..{ctrl-f10}

{alt-f1}..{alt-f10}

Egyebek:

{return} **{ctrl-return}** **{ctrl-prtsc}** **{ctrl-break}**

{null} **{all}** **{ctrl-bill.}** **{alt-bill.}**

(bill.: bármely egyéb billentyű, betű, szám)

A makrók létrehozásának másik formája, hogy egy keret szegélyére írjuk őket - ezek az ún. programmakrók. A "valódi" **FRED** programoktól csak abban különböznek, hogy indításuk egy **Alt**-billentyű kombinációval történik.

Ilyen programmakróból néhányat be is építettek a FRAMEWORK II-be, a **LIBRARY**-ben találjuk meg őket. Az egyik közülük, mely az **Alt-F3** billentyűk együttes lenyomására az operációs rendszer **dátumát** a képernyőre írja: March 3, 1991. Alakítsuk át úgy ezt a makrót, hogy a magyar sorrend szerint írja ki a dátumot.

Lépjünk a **SCROLL LOCK** billentyű segítségével a könyvtárba és keressük meg az **[{Alt-F3}]** nevű keretet. Ha a keret szegélyére írt képletet az **F2** billentyűvel lehozzuk a szerkesztő sorba és kinagyítjuk az **F9** billentyű segítségével, a következő sorokat látjuk:

```
types at cursor position
@performkeys(@date4(@today))
```

Az első sorba általában a program (programmakró) tevékenységének rövid ismertetését írjuk, megjegyzésként. A megjegyzések a ";" karakter után írhatók, végrehajtáskor nem veszi figyelembe azokat a rendszer.

```
;Aktuális dátum kiírása a kurzor pozíciójától

;Lokális változók létrehozása - kezdőértékük 0 !

@local(DatStr,Ev,Ho,Nap,Hossz,Kezdet),

;Értékadás: ":="

;Minden függvény a "@" karakterrel kell kezdődjön, függetlenül attól, hogy
rendszerfüggvény, vagy mi hoztuk létre

;@TODAY az op.rendszerbeli dátumot és időt adja vissza, dátum típusú adatként.

;@DATE1(Dátum) a paraméterként kapott dátum típusú adatból"hh nn, éééé" alakú sztringet
készít.

;Ez még nem jó nekünk, át fogjuk alakítani.

DatStr:=@date1(@today),

;Ha a nap értéke 9-nél nagyobb, az év száma a 9. karakteren kezdődik, egyébként a 8-on.

Kezdet:=9,

;Ha a nap értéke 9-nél nagyobb, a nap karaktereinek száma 2, egyébként 1.

Hossz:=2,

;@MID(miből,honnan,mennyit) A "miből" sztring típusú paraméterből a "mennyit"
paraméterbe
```

```
;megadott számú karaktert emel ki, a "honnan" paraméterben megadott pozíciótól kezdve.
@if(@mid(DatStr,6,1)=",",
@list(Kezdet:=8,Hossz:=1)),
;A "&" operátor a sztringek összefűzését végzi el.
Nap:=@mid(DatStr,5,Hossz)&".",
Ev:=@mid(DatStr,Kezdet,4)&".",
;Magyar nevű hónapok használata
@if(@mid(DatStr,1,3)="Jan",Ho:=" Január"),
@if(@mid(DatStr,1,3)="Feb",Ho:=" Február"),
@if(@mid(DatStr,1,3)="Mar",Ho:=" Március"),
@if(@mid(DatStr,1,3)="Apr",Ho:=" Április"),
@if(@mid(DatStr,1,3)="May",Ho:=" Május"),
@if(@mid(DatStr,1,3)="Jun",Ho:=" Június"),
@if(@mid(DatStr,1,3)="Jul",Ho:=" Július"),
@if(@mid(DatStr,1,3)="Aug",Ho:=" Augusztus"),
@if(@mid(DatStr,1,3)="Sep",Ho:=" Szeptember"),
@if(@mid(DatStr,1,3)="Oct",Ho:=" Október"),
@if(@mid(DatStr,1,3)="Nov",Ho:=" November"),
@if(@mid(DatStr,1,3)="Dec",Ho:=" December"), Ho:=Ho&" ", ;
;@PERFORMKEYS(sztring) a függvény a sztringben tárolt karaktereket úgy játssza vissza,
mintha a billentyűzetről adtuk volna be
@performkeys(Ev&Ho&Nap)
```

Fejezzük be a szerkesztést az **UPLEVEL** billentyűvel, mentsük el, majd nézzük meg a munkánk eredményét, üssük le az **Alt-F3** billentyűket : **1991. Március 3.**

5.2.3 A FRED nyelv lehetőségei

Mielőtt továbblépnénk a programozásban, tekintsük át a **FRED** nyelv lehetőségeit. A **FRED** nyelv valahol a **BASIC** és a **PASCAL** nyelv között foglal helyet, könnyen megtanulható, egyszerűen alkalmazható. Több, mint **170** függvénye van, segítségükkel kihasználhatjuk a FRAMEWORK II összes lehetőségét, speciális szolgáltatásait.

Integráns része a FRED nyelv a FRAMEWORK II-nek, elérhető bármelyik menü bármely funkciója.

Felhasználói **függvényeket** definiálhatunk, programunk futását lépésről lépésre nyomon követhetjük.

Lehetőségünk van vezérlésátadásra az **IF-THEN-ELSE** szerkezet segítségével és elöltesztelő ciklus alkalmazására.

Egy program egy sora legfeljebb **255** karakter hosszú lehet, a program maximális mérete **64000** karakter, de ezt az utóbbi korlátot a program modulokra bontásával könnyen átléphetjük.

Ha a program futása során hiba lép fel, visszakapjuk a vezérlést, a hibás utasítás inverz módban jelenik meg. A hiba kijavítása után újra futtathatjuk a programunkat.

A függvényeknek átadott paramétereket zárójelen belül kell elhelyezni, egymástól vesszővel kell elválasztani. A program neve annak a keretnek a neve, amelynek **formula** területére írtuk a programunkat, indítása úgy történik, hogy a keret szegélyére lépünk és lenyomjuk az **F5** billentyűt.

A több modulból álló programok minden modulja egy-egy keret, ezeket célszerű egy **gyűjtőkeretbe** foglalni.

Minden függvény egy számot, sztringet, dátumot, **FRED**-konstanst, vagy hibaüzenetet ad eredményül.

A függvények nevét a @ operátor előzi meg, ha egy keret formula területére írt programot szeretnénk végrehajtatni, ehhez egy @ jel után le kell írni a végrehajtandó **keret** nevét.

A függvények paramétereinek a helyére konstansokat (**14, "alma"**), kifejezéseket (**8*4, "alma"&"fa"**), változókat, keret-, vagy cellahivatkozásokat (**HelyiValtNeve, [SzámolóTábla], d6, e4:g10**), más függvényeket (**@value("153"**) írhatunk.

A helyi változókat a **@local(változólista)** utasítással hozhatjuk létre, ezek csak abban a keretben használhatók, ahol definiálva lettek.

Négyféle konstanst ismer a **FRED** nyelv: numerikust, sztringet, dátumot és az ún. **FRED-konstansokat.**

A **FRED-konstansok** egyik csoportja a már megtárgyalt **billentyűnevek**, a másik csoportja a programok **végrehajtásával** kapcsolatos, harmadik csoportját pedig a különböző **indikátorok** alkotják.

Hibakonstansok:

#N/A! (Not Available)	Nincs visszatérési értéke a kerethivatkozásnak, v. hibás képlet
#VALUE!	Különböző típusú adatok keveredése
#REF!	Nem találja a hivatkozást
#NAME!	Hibás név a hivatkozásban
#DIV/0!	Osztás nullával
#NUM!	Túl nagy v. túl kicsi numerikus érték (alul, ill.túlcsordulás)
#NULL!	Tartomány vége
#TBD!	Hibás formula, ill. egyes függvények visszatérési értéke

Logikai konstansok :

#TRUE	logikai IGAZ	**#FALSE**	logikai HAMIS
#YES	logikai IGEN	**#NO**	logikai NEM

Függvényértékek

#FUNCTION	@item v. @return függvény hívása
#GRAPH	újraszámított v. újrarajzolt ábra
#PICTURE	újraszámított v. újrarajzolt ábra
#MACRO	@setmacro függvény hívása
#KEY	@nextkey függvény hívása
#FILTER	@keyfilter függvény hívása

Rajzolás konstansai

#COLUMN	oszlop	**#ROW**	sor
#BAR	oszlop grafikon	**#STACKEDBAR**	pakolt oszlop grafikon

#PIE	körgrafikon	**#LINE**	vonal-grafikon
#MARKEDPOINTS	szimbólumokkal jelölt grafikon		
#X-Y	x-y grafikon	**#HIGHLOW**	tőzsdei grafikon

@ECHO függvény állandói

#ON	BE
#OFF	KI

Elhelyezési paraméter

#BEFORE	**@unit** függvény paramétere
#AFTER	**@unit** függvény paramétere

Operátorok

-	előjelváltás
%	százalék
^	hatványozás
*,/	szorzás, osztás
+,-	összeadás, kivonás
&	összekapcsolás (konkatenáció)
<, >, <=, >=, =, <>	logikai operátorok

A **precedencia** sorrend felülről lefelé, ill. balról jobbra csökken, zárójelekkel a precedencia sorrend megváltoztatható.

Egyebek

:=	értékadás (ua., mint a @set() függvény)
"	sztring határoló jel
,	paraméterátadásnál elválasztójel
.	azonosító elválasztó ([Keret].[Alkeret]), vagy tizedespont
:	tartomány elválasztó (a1:d9)
[]	azonosító kezdet/végjelző
{}	billentyűnév kezdet/vég jelző
@	függvény jelölő, hivatkozás végrehajtatás
;	megjegyzés a programban

Változók

A változó nevének az **angol** ábécé valamelyik betűjével, vagy aláhúzás karakterrel ("_") kell kezdődnie, tetszőleges hosszúságú lehet, tartalmazhat **szóközt** vagy **tabulátorjelet**. Ha a változó neve ékezetes betűket, szóközt, tabulátort, "#", "!", ".", "/", "+", "-" karaktereket tartalmaz, **szögletes zárójelek** közé kell foglalnunk.

5.2.4 A FRED nyelv függvényei

Rövidítések

n numerikus érték.

tj tizedes jegyek száma.

lista cellahivatkozások, keretnevek, adatállomány mezők, egymástól vesszővel elválasztva.

Kerekítő függvények

@int(n)	az **n** egész része.
@round(n, tj)	n kerekített értéke.
@floor(n)	n lefelé kerekített értéke.
@ceiling(n)	n felfelé kerekített értéke.

Numerikus függvények

@abs(n)	n abszolút értéke.
@sign(n)	n előjele: (n<0 **-1** n=0 **0** n>0 **+1**).
@mod(n1, n2)	n1/n2 maradéka (moduló függvény).
@sqrt(n)	n négyzetgyöke.
@rand	0 és 1 közötti "véletlen" számot eredményez.
@log(n)	n 10-es alapú logaritmusa.
@ln(n)	n természetes (e) alapú logaritmusa.
@exp(n)	természetes (e) alap n-edik hatványa.
@sin(n)	n szinusza radiánban.
@cos(n)	n koszinusza radiánban.
@tan(n)	n tangense radiánban.
@asin(n)	n arkusz szinusza radiánban.
@acos(n)	n arkusz koszinusza radiánban.
@atan(n)	n arkusz tangense radiánban.
@atan2(n1, n2)	n1/n2 arkusz tangense radiánban.
@pi	A **Ludolf**-féle szám értéke radiánban.

Statisztikai függvények

@max(lista)	Az adatlista numerikus elemei közül a legnagyobb érték.
@min(lista)	Az adatlista numerikus elemei közül a legkisebb érték.
@count(lista)	Az adatlista numerikus elemeinek száma.
@sum(lista)	Az adatlista numerikus elemeinek összege.
@avg(lista)	Az adatlista numerikus elemeinek átlaga.
@std(lista)	Az adatlista numerikus elemeinek standard szórása.

@var(lista) Az adatlista numerikus elemeinek szórásnégyzete.

Pénzügyi függvények

@pv(részfizetés,kamatláb,időszak) Részletfizetések adott kamatlábbal emelt végösszegének jelenlegi értéke.

@fv(kifizetés,kamatláb,időszak) Kifizetések kamatos kamattal emelt jövőbeni értéke.

@npv(kamatláb,részletlista) Kifizetendő részletek rögzített kamattal csökkentett összegének jelenlegi értéke.

@pmt(tőke,kamatláb,időszak) Tőke hozadékának értéke v. kölcsön törlesztési részlete.

@irr(becslés,részletlista) Belső megtérülési ráta (tényleges kamatláb) kiszámítása.

@mirr(jövőbeni kamatláb, jelenlegi kamatláb, részletlista) Módosított belső megtérülési ráta.

Logikai függvények

@if(f, I, H) Ha az f feltétel igaz, az I kifejezést hajtja végre, ha hamis, a H kifejezést. A H kifejezés megadása opcionális.

@and(lista) Visszatérési értéke **#TRUE**, ha a lista valamennyi kifejezésének logikai értéke #TRUE, különben **#FALSE**.

@or(lista) Visszatérési értéke **#TRUE**, ha a lista valamelyik kifejezésének logikai értéke #TRUE, különben **#FALSE**.

@not(kifejezés) Visszatérési értéke **#TRUE**, ha a kifejezés logikai értéke #FALSE, különben **#TRUE**.

@iserr(kifejezés) Visszatérési értéke **#TRUE**, ha a kifejezés értéke hibakonstans, különben **#FALSE**.

@isna(kifejezés) Visszatérési értéke **#TRUE**, ha a kifejezés értéke **#N/A!** hibakonstans, különben #FALSE.

@isabend(kifejezés) Visszatérési értéke **#TRUE**, ha a kifejezés hibás végrehajtásra utaló jelet tartalmaz, különben **#FALSE**.

@isalpha(kifejezés) Visszatérési értéke **#TRUE**, ha a sztring kifejezés csak betűket tartalmaz, különben #FALSE.

@isnumeric(kifejezés) Visszatérési értéke **#TRUE**, ha a kifejezés numerikus érték, különben **#FALSE**.

@isbop(sztring,index) Visszatérési értéke **#TRUE**, ha a sztring kifejezés egy bekezdés kezdete, különben **#FALSE**. Az index adja meg a sztringen belül a vizsgálandó karaktert.

@iscapslock Visszatérési értéke **#TRUE**, ha a **Caps Lock** bekapcsolt állapotban van, különben #FALSE.

@isnumlock Visszatérési értéke **#TRUE**, ha a **Num Lock** bekapcsolt állapotban van, különben #FALSE.

@isdocument(kerethivatkozás) Visszatérési értéke **#TRUE**, ha a keret szövegkeret, különben #FALSE.

@sense(menükapcsoló,"keret") Visszatérési értéke **#TRUE**, ha a megadott menükapcsoló bekapcsolt (ON) állásban van, különben **#FALSE**.

@exact(sztring1, sztring2) Visszatérési értéke **#TRUE**, ha a sztring1 pontosan megegyezik sztring2-vel, különben **#FALSE**.

Szövegkezelő függvények

@len(kifejezés) Visszatérési értéke a sztring kifejezés hossza.

@wordcount(keret) Visszatérési értéke a keretben található szavak száma.

@mid(kifejezés, k, db) A sztring típusú kifejezésből a k-adik karaktertől kezdve db számú karaktert ad eredményül.

@rept(kifejezés, db) Adott sztring típusú kifejezést db-szor megismétlel.

@value(kifejezés) A számokat tartalmazó sztring típusú kifejezésből numerikus értéket készít.

@integer(n, tj) Az **n** számot **tj** tizedesjegyre kerekíti, majd sztringet készít belőle.

@decimal(n, tj) Adott **tj** számú tizedesjegyre kerekíti az **n** számot, majd sztringgé alakítja **Fixed Decimals** formátumban.

@currency(n, tj) Adott **tj** számú tizedesjegyet tartalmazó **CURRENCY** formátumú sztringgé alakítja az **n** számot.

@business(n, tj) Adott **tj** számú tizedesjegyet tartalmazó **BUSINESS** formátumú sztringgé alakítja az **n** számot.

@scientific(n, tj) Adott **tj** számú tizedesjegyet tartalmazó **SCIENTIFIC** formátumú sztringgé alakítja az **n** számot.

@nationalize(valutajel, formátum) Szabványos valutajel és kijelzési formátum definiálása.

@dollar A dollár ($) jelet adja a **CURRENCY** formátum szimbólumának.

@pound A font (£) jelet adja a **CURRENCY** formátum szimbólumának.

@yen A jen (¥) jelet adja a **CURRENCY** formátum szimbólumának.

@unit("sztring", elhelyezés) új pénznem v. mértékegység jelet definiálhatunk, megadva, hogy a szám elé (#BEFORE), v. mögé (#AFTER) helyezze.

@thousands A formátum paraméter lehet a **@NATIONALIZE** függvényben, a tizedeseket pont (".") az egészek hármas csoportját vessző (",") választja el.

@milli A formátum paraméter lehet a **@NATIONALIZE** függvényben, a tizedeseket vessző (",") , az egészek hármas csoportját pont (".") választja el.

@chr(n, attribútumok) Az **n** számot, ha az 1 és 253 között van, **ASCII** karakterré alakítja, az attribútum a karakter megjelenési formáját befolyásolja.

@textselection Az aktuális kiválasztott szöveget sztringként kezeli.

Dátum és idő függvények

@date Visszatérési értéke az operációs rendszer dátuma.

@date(év, hó, nap) Beállítja a rendszerdátumot.

@today Ua. mint a **@date** paraméter nélkül.

@diffdate(dátum1, dátum2) Megadja a dátum1 és dátum2 közti napok számát.

@sumdate(dátum, növekmény) A növekmény értékét hozzáadja a dátumhoz. A megadott dátumot HHH NN, ÉÉÉÉ sztringként írja ki.

@date2(dátum) A megadott dátumot HHH ÉÉÉÉ sztringként írja ki.

@date3(dátum) A megadott dátumot HHHNN sztringként írja ki.

@date4(dátum) A megadott dátumot Hónap NN, ÉÉÉÉ sztringként írja ki.

@datetime(dátum, idő) A FRAMEWORK II formátumban megadott dátumot és időt adja vissza.

@time Visszatérési értéke az operációs rendszerben tárolt idő.

@time(óra, perc, másodperc, századmásodperc) Beállítja a rendszerórát.

@time1(idő) A megadott időpontot 12 órás formátumban írja ki, óó:pp AM/PM.

@time2(idő) A megadott időpontot 24 órás formátumban írja ki, óó:pp.

@time3(idő) A megadott időpontot 24 órás formátumban írja ki, századmásodperc pontossággal, óó:pp:mp,szmp.

Vezérlési függvények

@list(lista) A függvény segítségével a listában levő kifejezések egyetlen utasításként kezelhetők.

@set(hivatkozás, kifejezés) A hivatkozott hely (keret, cella, mező, változó) a kifejezés értékét veszi fel. Hatása megegyezik a ":=" operátoréval.

@while(feltétel, lista) A FRAMEWORK II ciklusutasítása. Mindaddig végrehajtja a listában lévő utasításokat, míg a feltétel igaz.

@select(index,lista) A listából kiválasztja az index-edik kifejezést.

@result(kifejezés) A függvény befejezi egy program végrehajtását, a program visszatérési értéke a kifejezés lesz.

@execute(hivatkozás, képlet) A hivatkozott hely képletterületére írja a kifejezést, majd végrehajtatja azt.

@printreturn Nyomtatási művelet befejezése.

@getformula(hivatkozás) A hivatkozott hely képletterületén levő képletet adja meg, szöveges formában.

@setformula(hivatkozás,képlet) A hivatkozott hely képletterületére írja a kifejezést.

@run(hivatkozás,"parancssor") A parancssorban megadott külső program végrehajtatása.

@writetextfile("útvonal", hivatkozás) A hivatkozott keretet az útvonallal megadott könyvtárba írja, .TXT kiterjesztéssel.

@writeframefile("útvonal", hivatkozás) A hivatkozott keretet az útvonallal megadott könyvtárba írja, .FW2 kiterjesztéssel.

@setdrive Aktuális lemezegység lekérdezése.

@setdrive(drive) Aktuális lemezegység beállítása.

@setdirectory Aktuális könyvtár lekérdezése.

@setdirectory("útvonal") Aktuális könyvtár beállítása.

@getenv(környezeti változó) A környezeti változó értékét adja vissza.

@memavail A szabad RAM mérete, byte-okban.

@trace(hivatkozás, állapot) Nyomkövetés bekapcsolása: állapot=#ON, kikapcsolása: állapot=#OFF.

Tartományokra vonatkozó függvények

@choose(n, lista) Az n-dik elemet választja ki a listából.

@hlookup(n, tartomány, eltolás) A megadott tartomány első sorában keresi azt az értéket, amely még nem nagyobb, mint n. Ennek az oszlopnak az eltolás paraméterben megadott sorában lévő értéket választja ki és adja vissza.

@vlookup(n, tartomány, eltolás) A megadott tartomány első oszlopában keresi azt az értéket, amely még nem nagyobb, mint **n**. Ennek a sornak az eltolás paraméterben megadott oszlopában lévő értéket választja ki és adja vissza.

@fill(tartomány, kezdőérték, növekmény) A tartomány kezdő elemébe a kezdőértéket teszi, majd balról jobbra, ill. felülről lefelé haladva feltölti a tartományt a növekménnyel folyamatosan növelt értékkel.

@get(tartomány) A függvény a tartomány aktív elemének értékét veszi fel.

@next(tartomány) A függvény a tartomány következő elemét teszi aktuálissá.

@put(tartomány, kifejezés) A tartomány aktuális elemének a kifejezés értékét adja.

@reset(tartomány) A tartomány első elemét aktuálissá teszi.

Függvénykészítő függvények

@[hivatkozás] Végrehajtja a [hivatkozás]-ban megadott helyen lévő programot. (Felhasználói függvény definiálása!)

@[hivatkozás](kifejezés1,...kifejezés16) Végrehajtja a [hivatkozás]-ban megadott helyen lévő programot és átadható neki max. 16 paraméter.

@item1 ... @item16 Paraméter átvevő függvény.

@item(index) Az index-ben megadott sorszámú paraméter átvétele. (Pl.: @item(5) megegyezik @item5-tel).

@itemcount Az átadott paraméterek száma.

@local(változó, változó,...) Lokális változó definiálása.

@return(kifejezés) Programvégrehajtás befejezése, visszatérési érték a kifejezés értéke lesz.

Makróépítő függvények

@setmacro({alt-?}, hivatkozás) A hivatkozott helyen levő képletet (programot) makróként az {alt-?} billentyűhöz kapcsolja.

@performkeys(kifejezés) A függvény végrehajtja a sztring típusú kifejezést.

@echo(állapot) Ha állapot #ON, akkor a billentyű leütések hatása észrevehető a képernyőn, ha #OFF, akkor nem.

@nextkey(idő) A függvény megvárja a következő billentyűleütést, ill. az idő paraméterben megadott időtartam leteltét. Az idő paraméter opcionális.

@key Utolsó billentyűleütés lekérdezése.

@keyname(billentyű) A paraméterként megadott változóból billentyűnév típusú stringet készít.

@keyfilter(billentyű, hivatkozás) Szűri a billentyűleütéseket: a szűrendő billentyű leütésekor a hivatkozott helyen található programot hajtja végre.

@setselection("keret") Kiválasztja az adott keretet.

@panel1(keret) A státuszsor harmadik harmadában lévő számpár első tagját adja meg.

@panel2(keret) A státuszsor harmadik harmadában lévő számpár második tagját adja meg.

@frametype(keret) A keret típusára jellemző számot ad meg.

@frameview(keret) A keret megjelenítésére jellemző számot ad meg.

Felhasználói felület funkciók

@prompt(üzenet, eltolás) A képernyő alsó sorában megjeleníti az üzenetet, eltolásnyi szóköz kiírása után.

@eraseprompt Törli az üzenetsort.

@inputline(üzenet, alapértelmezett szöveg, kurz. rámutatás, automatikus törlés, első karakter)

A FRAMEWORK II adatbeolvasó függvénye. Az üzenetsor közepére kiírja az üzenetet, az alapértelmezett szöveg a beolvasandó adat helyén jelenik meg. Ha a kurzoros rámutatás értéke #YES,megengedi a kurzormozgatással történő rámutatást. Adatbeíráskor az alapértelmezett szöveg automatikusan törlődik, ha az automatikus törlés értéke #YES. Ha az utolsó paraméter #YES, csak a beviendő adat első karaktere jelenik meg inverzen.

@display(hivatkozás) Megjeleníti a hivatkozott keretet a képernyőn.

@menu(hivatkozás) Felhasználói menü létrehozása.

@quitmenu Kilépés a felhasználói menüből.

@hide(hivatkozás) Eltünteti a hivatkozott keretet.

@unhide(hivatkozás) Megszünteti a hivatkozott keret rejtettségét.

Grafikus függvények

@drawgraph(terület, orientáció, típus, cím, X-tengely neve, Y-tengely neve, legkisebb Y-érték, legnagyobb Y-érték, növekmény) Grafikon rajzolására szolgáló függvény, ez az alak kördiagram rajzolására nem alkalmas.

Paraméterek:

Terület : táblázat v. adatállomány összefüggő tartománya.

Orientáció : a vízszintes tengely cimkéit #COLUMN: az első oszlopból, **#ROW**: az első sorból veszi.

Típus : **#BAR** oszlop, **#STACKEDBAR** pakolt oszlop, **#LINE** vonal, **#MARKEDPOINTS** pont, **#XY** x-y, **#HIGHLOW** tőzsdei grafikon.

Cím : ábra címe, sztring típusú.

X-tengely neve, Y-tengely neve szintén sztring típusú. Beállíthatunk az automatikus y-tengely beosztástól eltérő, saját beosztást is, ilyenkor meg kell adni a legkisebb és a legnagyobb y értéket, valamint a növekményt is.

@drawgraph(terület, orientáció, típus, cím, X-tengely neve, Y-tengely neve, körcikk kiemelés) Kördiagram rajzoló függvény. A paraméterek megegyeznek az előzőekben ismertetettekkel. A körcikkek közül kiemeltethetünk egyet, vagy az összeset.

Egyéb függvények

@beep(angmagasság, időtartam) Hangkeltés a beépített hangszórón keresztül. A hangmagasság értékek az alapoktávnál: **C = 523 D = 587 E = 659 F = 698 G = 784 A = 880 B = 988 C = 1047** Magasabb oktáv esetén az értékeket duplázni, alacsonyabb oktáv esetén felezni kell. Az időtartamot századmásodpercben kell megadni.

@dbasefilter(állománynév,szűrőfeltétel,kezdő rekordszám, végső rekordszám) dBASE II, dBASE III v. dBASE III+ állományt betöltő függvény.

Állománynév: betöltendő file azonosítója, elérési útvonallal együtt.
Szűrőfeltétel: logikai kifejezés, csak àzokat a rekordokat tölti be, melyek megfelelnek a feltételnek. Ha a rekordszámokat is megadjuk, csak az intervallumba tartozó rekordokon végzi el a szűrést.

5.3 Ajánlott irodalom

dr. Barakonyi Károly

FRAMEWORK II

dr. Barakonyi Károly

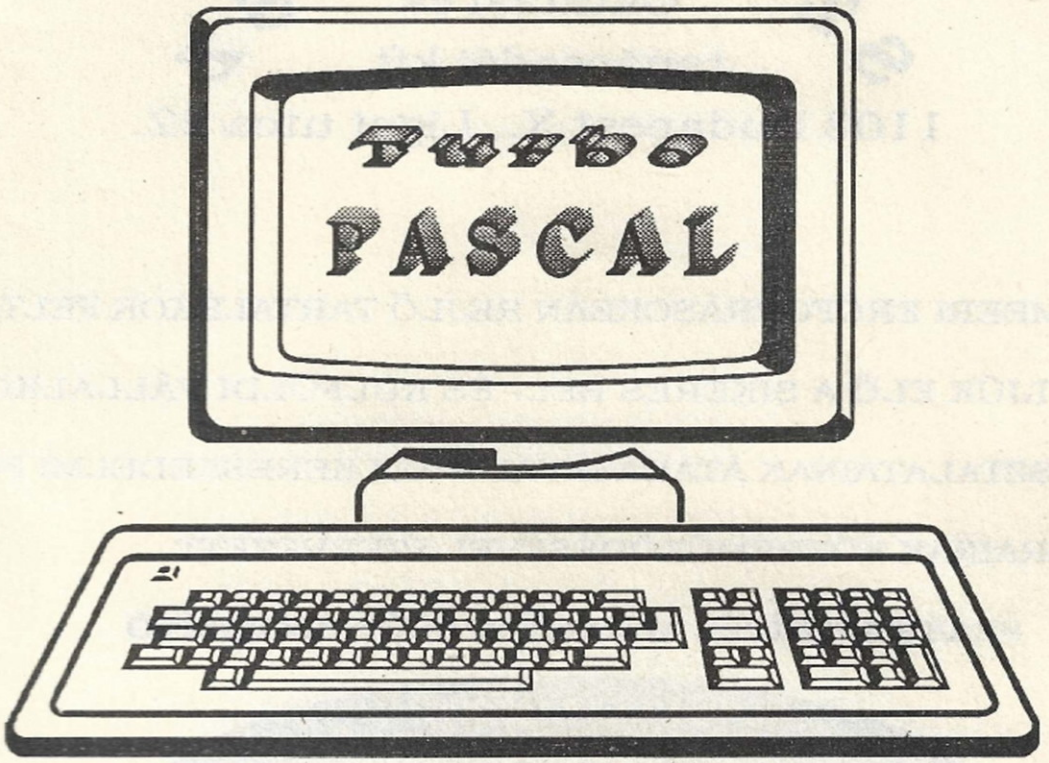
A FRAMEWORK II használata kezdőknek

Pápay Kálmán

Bevezetés a FRAMEWORK II használatába

Bill Harrison

Bevezetés a FrameWork III használatába


Turbo
PASCAL

SZÓKRATÉSZ

Oktatási és
tanácsadói kft.
1102 Budapest X., Liget utca 22.

AZ EMBERI ERŐFORRÁSOKBAN REJLŐ TARTALÉKOK FELTÁRÁSÁT SEGÍTJÜK ELŐ A SIKERES BEL- ÉS KÜLFÖLDI VÁLLALKOZÁSOK TAPASZTALATAINAK ÁTADÁSÁVAL A KÜLKERESKEDELMI FŐISKOLA TANÁRAINAK KÖZREMŰKÖDÉSÉVEL SZERVEZETT

SZAKKÉPESÍTÉS MEGSZERZÉSÉT BIZTOSÍTÓ

TANFOLYAMAINKON.

TANÁCSADÓ SZOLGÁLATUNK TÁMOGATJA AZ INDULÓ VÁLLALKOZÁSOKAT.
SZAKÉRTŐINK - FELKÉRÉSRE - SZAKTANÁCSADÓKÉNT MŰKÖDNEK KÖZRE A PRIVATIZÁCIÓS PROGRAMOKBAN.

TANFOLYAMAINKRÓL ÉS SZOLGÁLTATÁSAINKRÓL
FELVILÁGOSÍTÁST ADUNK

1102 Budapest, Liget utca 22.

Telefon/fax: 127-5008

6. A TURBO PASCAL PROGRAMOZÁSI NYELV

Nicklaus Wirth 1968-ban készítette el új programozási nyelvét, melyet Blaise Pascalról, a XVII. századi francia matematikusról nevezett el. Az eltelt két évtized alatt a nyelv igen nagy népszerűségre tett szert, valamennyi számítógéptípuson rendelkezésünkre áll, s talán a legáltalánosabban használt programozási nyelv. Népszerűségére mi sem jellemzőbb, minthogy a programozás tanításának eszköze lett, egyetemeken, főiskolákon, tanfolyamokon használják, hazánkban és szerte a nagyvilágban egyaránt.

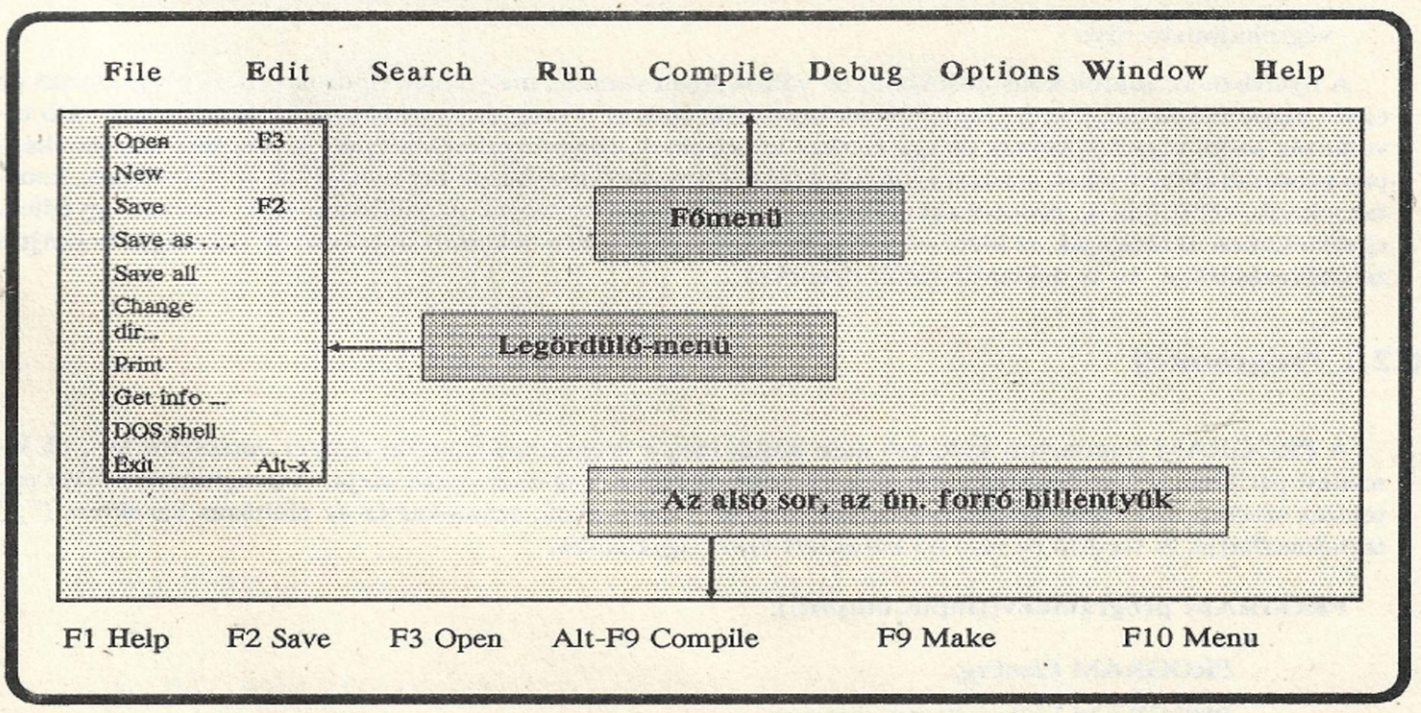
A siker okai között az is megtalálható, hogy idejekorán sikerült szabványosítani a nyelvet, így a nyelvjárások között, az alap utasításkészletben nincsenek túl nagy eltérések. A Turbo Pascal is egy ilyen nyelvjárás, a legjobban sikerültek közül való. Az Egyesült Államok-beli Borland cég fejlesztette ki, az első változat 1984-ben jelent meg, azóta folyamatosan tökéletesíti a nyelvet, jelenleg a 6.0-ás verziónál tart. Jelen jegyzet az 5.0-ás verzió tulajdonságait ismerteti, az alapok elsajátításához ez is elegendő, a fejlettebb programozási lehetőségek (objektum orientált programozás, assembly rutinok beillesztése) ismertetése nem volt célunk. A Turbo Pascal a többi programozási nyelv közül kimagaslik előnyös tulajdonságaival, többször is az év nyelvének választották. A standard Pascal-hoz hasonló, de attól több ponton is eltér, alkalmas számítások elvégzésére, adatfeldolgozó, grafikai megjelenítő programok írására és sok egyéb, igen szerteágazó feladat megoldására. Beépített szövegszerkesztővel, gyors fordító és futtató rendszerrel, kényelmes hibakeresési lehetőséggel rendelkezik. Mindez együtt egy hatékony és kényelmes munkát tesz lehetővé.

6.1 A rendszer elemei

Turbo.exe Integrált fejlesztői környezet (szövegszerkesztő, fordító, futtató, hibakereső) Turbo.tpl Programkönyvtár (Turbo Pascal Library) Turbo.hlp Segítő információk gyűjteménye Tpc.exe Az integrált környezeten kívüli fordító program Tinst.exe Az integrált környezet telepítése, jellemzőinek megváltoztatása

Indítása: C:\tp\turbo

Az integrált fejlesztői környezet képernyőjének felépítése:



A felső sor a főmenű, az alábbi menüpontok közül választhatunk:

File	Állománykezelés, lemezműveletek, könyvtárkezelés, kilépés
Edit	Program forrásszövegének létrehozása
Run	Program futtatása
Compile	Program fordítása, futási hibák keresése
Options	Fordítási, szerkesztési paraméterek, fordítási direktívák beállítása, elmentése, betöltése
Debug	Hibakeresés
Break/Watch	Figyelő ablak, töréspontok beállítása.

Az alsó sor, az ún. forró billentyűk:

F1-Help	Segítő ablak nyitása
F5-Zoom	Aktív ablak nagyítása/visszaállítása
F6-Switch	Kapcsolás az ablakok között
F7-Trace	Program soronkénti futtatása
F8-Step	Program soronkénti futtatása, de nem megy bele az alprogramokba
F9-Make	Lefordítja és szerkeszti a forrásprogramot
F10-Menu	A főmenű és az aktív ablak között kapcsol.
Alt-X	Kilépés a programból

6.2 A Pascal program szerkezete

Egy Pascal program blokkokból áll, a nyelv kifejezetten alkalmas a struktúrált programozásra.

A program 3 fő részből áll:

- programfej
- deklarációs rész
- végrehajtandó rész.

A nyelvben az adatok konstansokban, ill. változókban vannak, melyeknek típusuk van. A programban az egyes alaptevékenységeket procedúráknak nevezzük, azokat a programrészeket, melyek egy rájuk való hivatkozás során végrehajtódnak és egy értéket képviselnek, függvényeknek hívjuk. A program tulajdonképpen procedúrák hívásának sorozata, ezt a sorrendet a vezérlő utasítások befolyásolják. A típusoknak, konstansoknak, változóknak, procedúráknak, függvényeknek azonosítójuk (nevük) van, ezek a nevek egyediek, egyértelműen azonosítják az illető objektumot. Bár a nyelv előre deklarált objektumok sokaságát bocsájtja rendelkezésünkre, mi is deklarálhatunk ilyeneket.

6.2.1 Programfej

A PROGRAM fenntartott kulcsszó után adjuk meg a programazonosítót, melyet opcionális be-, ill. kimeneti periféria felsorolás követ, zárójelek között. A fejlécet pontosvessző zárja. A programazonosító egy tetszés szerinti szó, mely betűvel kezdődik, betűket (angol abc!), számokat és az aláhúzás karaktert ("_") tartalmazhatja. A fordító az első 63 karaktert veszi figyelembe.

PROGRAM programnév[(input,output)];

```
PROGRAM ElsoPrg;
PROGRAM Listazo(input, output);
```

PROGRAM Nyomtato(input, LST);

A Turbo Pascal rendszerben a 4.0 verziótól kezdve minden előre deklarált típus, konstans, változó, procedura és függvény a programkönyvtárban foglal helyet (UNIT-okban, egységekben).

A TURBO.TPL tartalma:

SYSTEM	standard Pascal procedúrák, függvények
DOS	DOS procedúrák és függvények
CRT	képernyő és billentyűzetkezelés
PRINTER	nyomtató kezelés
OVERLAY	overlay technika alkalmazásához
GRAPH	grafikai rutinok

Alapesetben a GRAPH unit-ot nem tartalmazza a TURBO.TPL, de bármikor beleszerkeszthető. Minden esetben, amikor a standard Pascal-tól eltérünk, a megfelelő unit-ot kell befoglalni, a SYSTEM unit-ot automatikusan a programhoz fordítja a rendszer. A programfejhez tartozik az a fordítónak szóló utasítás, mely a könyvtárból behívja a kívánt unit-okat. A USES fenntartott szó után soroljuk fel a befoglalandó unit-ok neveit.

USES unit-név [, unit-név,...];

Pl.: USES CRT;

USES CRT,PRINTER;

USES GRAPH;

Használhatók a *.TPU kiterjesztésű unit könyvtárak is, ez a felhasználói programkönyvtár.

6.2.2 Deklarációs rész

Itt deklarálunk és definiálunk minden olyan azonosítót, mely egy-egy típust, konstanst, változót, eljárást v. függvényt azonosít és amelyre a program végrehajtandó részében hivatkozunk. A deklarációs rész feladata előkészíteni a végrehajtandó részt, valamint lefoglalni a változóknak a szükséges memóriarészt. A deklarációs rész további öt részre osztható:

- címke deklarálás
- típus definiálás
- konstans definiálás
- változó deklarálás
- procedúra és függvény deklarálás.

Címke deklarálás

A program végrehajtandó részében bármely utasítást elláthatunk címkével, így a GOTO utasítással átadhatjuk a vezérlést az adott utasításra. A később felhasználandó címkéket előre le kell fektetni ebben a részben: a LABEL fenntartott szó után, egymástól vesszővel elválasztva soroljuk fel a címkéket. Max. 4 betű és/vagy szám lehet egy címke.

LABEL címke [, címke,...];

Pl.: LABEL Vege;

LABEL Kezd, P1, 9999;

Ez az utasítás nem való struktúrált programba, használata kerülendő!

Típus definiálás

A konstansoknak, változóknak, függvényeknek típusuk van, ezeket a típusokat a létrehozás előtt kell definiálnunk.

Egyszerű típus: további összetevőkre nem bontható.

Összetett (struktúrált) típus: egyszerű típusok valamilyen kombinációja.

Mutató típus: dinamikus változók létrehozására szolgál.

Skalár típus: értéke a számegyenesen ábrázolható, két ilyen érték egymással összemérhető.

Sorszámozott típus: elemeihez egy-egy egész szám rendelhető, mint sorszám (rendszám).

Előre definiált egyszerű típusok:

SHORTINT	-128 .. +127
INTEGER	-32768 .. +32767
BYTE	0 .. 255
WORD	0 .. 65535
LONGINT	-2147483648 .. +2147483647
REAL	2.9 E-39 .. 2.7 E 38
BOOLEAN	TRUE, FALSE
CHAR	bármely ASCII karakter
STRING	0 .. 255 db. ASCII karakter

STRING típus esetén a fenntartott szó után szögletes zárójelben adjuk meg azt a karakterszámot, amelyet az ilyen típusú változó maximálisan felvehet. Max. 40 karaktert tartalmazó string típusú változó deklarálása: **STRING[40]**.

A típus definiáló részben a TYPE fenntartott szó után soroljuk fel az egyes típusleírásokat: típusazonosító, egyenlőségjel, típusleírás, pontosvessző.

TYPE azonosító = leírás; [azonosító = leírás; . .. .]

```
Pl.:    TYPE EGESZ = INTEGER;
        VALOS = REAL;
        SZOVEG = STRING[80];
        BETU = CHAR;
        LOGIKAI = BOOLEAN;
```

A későbbiekben az itt definiált típusazonosítókat és az előre definiáltakat egyaránt felhasználhatjuk. Ennek a résznek a strukturált típusok leírásánál van nagy szerepe, egyszerű típusok esetén nem mindig használják.

Konstans definició

Itt adjuk meg a programban később használt konstansokat, azaz összerendelünk egy nevet és egy értéket, de ezen az összerendelésen a későbbiekben nem változtathatunk.

CONST azonosító = érték; [azonosító = érték; . . . .]

A típus megadására nincs szükség, mert a fordító az érték alapján automatikusan felismeri.

```
Pl.;    CONST   SorPerLap = 66;
                Fejlec = 'Lista';
```

TermAlap= 2.71828;

Változó deklarálás

Itt adjuk meg a programban később használt változókat. A végrehajtandó részben csak ezeket használhatjuk. *A VÁLTOZÓKNAK A PROGRAM INDíTÁSAKOR NINCS KEZDŐÉRTÉKÜK!*

VAR azonosító:típus; [azonosító:típus; . . .]

BEGIN Végrehajtandó rész kezdete

Utasítások, procedura v. függvényhívások

END. Vérgehajtandó rész vége .

A programban minden utasítást, procedurát pontosvesszővel zárunk le. Az utasítások, procedurák több sorban is elhelyezkedhetnek, nem feltétlenül kell egy sorba írni őket.

Az azonosítók kisbetűvel, nagybetűvel v. vegyesen is írhatók.

Szövegkonstansnak mindig aposztrófok között kell lennie.

6.2.3 Paraméterátadás szabályai

A proceduráknak, függvényeknek átadhatunk adatokat, melyekkel azok valamilyen műveletet végrehajtanak. Híváskor ezeket, mint konstansokat és/vagy változókat a procedura v. függvény neve után, zárójelek között írhatjuk be. A zárójelen belül, ha több szerepel belőlük, vesszővel kell elválasztani őket.

Procedure függvény-név[(paraméter[,paraméter....])];

Ha konstanst vagy konstans-azonosítót adunk át, azt a procedura vagy függvény mint értéket használja fel. Ha változó-azonosítót adunk meg paraméterként, azt a procedura vagy függvény fel is használhatja, de változtathat is rajta.

6.3 Adatok be- és kivitele

Valamely változóba a READ v. a READLN procedurával lehet adatot bevinni a standard inputról.

A READLN az adatbevitel után sort emel a standard inputon.

READ(változó[,változó,...]); READLN(A,Q);

Adatbevitelkor a beviendő adat típusának meg kell egyeznie a változó típusával.

A változó típusa és a bevihető érték

EgészSzámjegyek, '+','-' jelek (Shortint, Byte, Integer, Word, Longint)

ValósSzámjegyek, /+ , - , jelek, (Real) tizedespont, a tíz hatványát jelző E betű

Karakteres bármely ASCII karakter (Char, String)

Logikai közvetlenül nem bevihető, (Boolean) értékadással programon belül kapnak értéket (TRUE és FALSE szavak).

Az adatbevitelt az ENTER (RETURN) zárja le, ekkor az érték a változóba kerül. Bevitel során a sor szerkeszthető is:

BackSpace utolsó karakter visszatörlése

Esc egész sor visszatörlése

Ctrl-D egy karakter az előző bevitt sorból

Ctrl-F előző bevitt sor lehívása

Változó tartalmának kiírása

Valamely változó tartalmának kiírására a WRITE és a WRITELN procedurák szolgálnak. Paraméterként egy v. több kifejezés szerepelhet a zárójelen belül.

Kifejezés: változók, konstansok, függvények műveleti jellel összekapcsolt kombinációja v. műveleti jel nélküli leírása. A WRITELN a READLN-hoz hasonlóan a kiírás elvégzése után sort emel.

WRITE(kifejezés[,kifejezés...]);

WRITELN(kifejezés[,kifejezés...]);

Pl.: WRITE(Sorszam);

WRITELN('Hányados: ', Osztandó/Osztó:8:2); :8:2 a hányados formázott kiírását teszi lehetővé (99999.99)

WRITELN; {Soremelés}

Lássuk hát az első programunkat, amely bemutatja a deklarációs rész megírását, valamint a be-, és kiviteli utasításokat:

```
Program Elso;
TYPE  Szoveg=STRING[80];
      Betu=CHAR;
      Szam=INTEGER;
VAR   Mondat:Szoveg;
      Karakter:Betu;
      Egesz:Szam;
BEGIN
   WRITE('Kérem a számot: ');READLN(Egesz);
   WRITE('Kérem a betüt: ');READLN(Karakter);
   WRITE(Kérem a mondatot: ');READLN(Mondat);
   WRITELN('A szám: ',Egesz);
   WRITELN('A mondat: ',Mondat);
   WRITELN('A betü: ',Karakter);
END.
```

A programban megjegyzéseket helyezhetünk el a '{' és a '}' jelek, vagy a '(*' és a '*)' jelek között. A megjegyzések több sorosak is lehetnek, a fordító nem fordítja bele a programba.

Pl.: { Ez megjegyzés } (* Ez is megjegyzés *)

A '[' helyett '(.', a ']' helyett '.)' használható egyes billentyűzeteken.

Pl.: STRING [80] - STRING (.80.)

A programnyelv a számokat alapértelmezésben decimális számként értelmezi. Ettől eltérhetünk és megadhatunk hexadecimális számokat is, de ezeknek a '$' jellel kell kezdődniük.

Pl.: $FFh=255d

$FFFFh=65535d

CHAR típusú változó v. konstans úgy is felvehet értéket, hogy ASCII kódot adunk meg.

Pl.: CONST D_Betu = 'D';

A_Betu =#65;

B_Betu = #$42;

Vezérlő karaktereket is megadhatunk így, de máshogy is:

Pl.: CONST Escape = #27;

Típusazonosság

Két típus azonos, ha azonosítójuk ugyanaz, vagy az egyik típust a másik azonosítójával deklaráljuk.

Típus kompatibilitás

Két típus kompatibilis, ha az alábbiak közül valamelyik igaz:

- a két típus azonos, vagy leírásuk megegyezik
- vagy 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 v. karaktertömb típusú
- az egyik típus nélküli, a másik bármilyen típusú mutató
- mindkettő eljárás vagy függvény típusú, azonos paraméterszámmal, paramétertípusokkal és függvény esetén azonos eredmény típussal.

A típus kompatibilitás kifejezések kiértékelésénél szükséges.

Értékadás kompatibilitás:

Megengedett a T1 típusú változó T1:= T2 típusú kifejezés, azaz T2 értékadás szempontjából kompatibilis T1-el, ha az alábbiak egyike igaz:

- T1 és T2 azonos típusok, de egyik sem állomány, ill. azt tartalmazó struktúrált típus
- T1 és T2 kompatibilis sorszámozott v. valós típusok és T2 értékkészlete beleesik T1 értékkészletébe
- T1 valós, T2 egész típus
- T1 karakterlánc, T2 karakter, karakterlánc v. karaktertömb típus
- 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, eljárás v. függvény típusok
- T1 eljárás v. függvénytípus, T2 eljárás v. függvény azonos paraméterszámmal, paramétertípusokkal és függvények esetén azonos eredménytípussal.

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

6.4 Műveletek

6.4.1 Aritmetikai, halmaz, karakterlánc és logikai műveletek

Aritmetikai :

+,-	előjel
NOT	bitenkénti "NEM"
*	szorzás
AND	" ÉS"
/	valós osztás
SHL n	eltolás balra n bittel
DIV	egész osztás
SHR n	eltolás jobbra n bittel
MOD	maradékképzés

OR	bitenkénti VAGY
+ ; -	összeadás ; kivonás
XOR	bitenkénti kivonás KIZÁRÓ VAGY

Halmaz:

*	metszet
+	unió
-	különbség

Karakterlánc:

+	összefűzés
<M>	vizsgálat

Logikai:

x IN y	elemvizsgálat (x eleme y halmaz, ha , x típusa AND " ÉS megegyezik y OR " VAGY adattípusával).
NOT	logikai NEM
XOR	"KIZÁRÓ VAGY" reláció (összehasonlító):

Precedencia sorrend

1. NOT, +, -
2. *, /, DIV, MOD, AND, SHL, SHR A prioritási sorrendek
3. +, -, OR, XOR zárójelzéssel megváltoztathatók
4. =, IN

6.4.2 Mintapéldák a fenti műveletekre

```
Program Hasonlit;
Var x,y, a,b,c,d,e,f:INTEGER;
    r, x,y, oszto:REAL;
    s1,s2:STRING[20];
BEGIN
    CLRSCR;
    WRITE('Kérem az első számot: ');READLN(x);
    WRITE('Kérem a második számot: ');READLN(y);
    IF x>y THEN WRITELN('Az első szám nagyobb a másodiknál!')
    ELSE
        IF x=y THEN WRITELN('A két szám egyenlő!')
        ELSE WRITELN('A második szám a nagyobb!');
    WRITELN; WRITE('Kérem az első szöveget: ');READLN(s1);
    WRITE('Kérem a második szöveget: '); READLN(s2);
    IF s1>s2 THEN WRITELN('Az első szöveg "nagyobb" a másodiknál!')
    ELSE
        IF s1=s2 THEN WRITELN('A két szöveg egyenlő!')
        ELSE WRITELN('A második szöveg a "nagyobbik"');
    WRITELN; WRITE('Kérek egy valós számot: '); READLN(r);
    IF (r=10) and (r20) THEN WRITELN('A szám 10 és 20 közé esik!')
    ELSE WRITELN('A szám nem esik 10 és 20 közé!');
    WRITELN;
          { Kétismeretlenes egyenletrendszer megoldása: }
    WRITELN('ax+by=c'); WRITELN('dx+ey=f');
```

```
        WRITELN('--------'); WRITELN;
        WRITE('a értéke: ');READLN(a);
        WRITE('b értéke: ');READLN(b);
        WRITE('c értéke: ');READLN(c);
        WRITE('d értéke: ');READLN(d);
        WRITE('e értéke: ');READLN(e);
        WRITE('f értéke: ');READLN(f);
        oszto:=a*e-b*d;
        IF oszto=0 THEN WRITELN('Nincs megoldás!)
        ELSE
        BEGIN
            x:=(c*e-b*f)/oszto;
            y:=(a*f-c*d)/oszto;
            WRITELN('x=',x,' y=',y);
        END;
        WRITELN;
    END.
```

6.4.3 Típusos konstans, intervallum típus, felsorolt típus

Típusos konstans

Nevével ellentétben nem konstans, hanem olyan változó, amelynek kezdőértéke is van. A típusos konstans előnye a konstansokkal szemben, hogy csak egyszer jelenik meg a lefordított programban, míg a konstansok annyiszor, ahányszor használjuk. Definiálása a konstans definiciós részben történik:

Azonosító: típusazonosító = érték;

Pl.: Const Szam:INTEGER = 20000;

SzorzoTabla:ARRAY[1..5, 1..5] OF BYTE= ((1,2,3,4,5),(2,4,6,8,10), (3,6,9,12,15), (4,8,12,16,20),(5,10,15,20,25));

Betuk:SET OF CHAR = ['a'..'z', 'A'..'Z'];

Intervallum típus

Alaptípusa sorszámozott kell hogy legyen. Definiáláskor megadjuk, hogy az ilyen típusú változó milyen határok közötti értékeket vehet fel. A két (alsó és felső) határértéket két pont (..) választja el egymástól, ez az intervallumot jelentő jel.

Azonosító: alsó határérték .. felső határérték;

Pl.: VAR Napok: 1..31;

Hónapok:1..12;

NagyBetuk:'A'..'Z';

Felsorolt típus

Definiáláskor zárójelek között, egymástól vesszővel elválasztva soroljuk fel azokat az értékeket, melyeket az ilyen típusú változó felvehet. Ezekhez a fordító egy-egy rendszámot (sorszámot) rendel, 0-tól indulva.

Azonosító:(érték1,érték2,..., értékN);

Pl.: VAR HetVege:(Szombat, Vasarnap);

MunkaNapok:(Hetfo, Kedd, Szerda, Csutortok,Pentek);

Ez a felsorolás egyben nagyság szerinti sorrend is.

Adott elem sorszámát az ORD függvény segítségével kapjuk meg.

Pl.: ORD(Hetfo) értéke 0.

Az eredmény egész típusú. A Pred függvény az előzőnek, a Succ függvény a következőnek felsorolt értéket adja vissza:

Pred(Szerda) értéke Kedd, Succ(Szerda) értéke Csutortok.

Más standard függvény nem alkalmazható ennél a típusnál.

ORD(x) x sorszámozott típus, az ORD függvény x sorszámát adja vissza, mint LONGINT típusú értéket.(ORD:Ordinal number):

Pl.:

```
VAR     F:(Ha, He, Ho, Hu); B:Boolean;
        C:Char;

        F:=Ho;B:=TRUE;C:='B';
        WRITE(ORD(F));  {2, mert a Ho rendszáma 2}
        WRITE (ORD(B)); {1,TRUE rendszáma} {Boolean = False (0), TRUE (1)}
        WRITE (ORD(C);  {66 a 'B' ASCII kódja-rendszáma}
```

PRED(x) { PREDECESSOR } x sorszámozott típus, a függvény x sorszám szerinti előző értékét adja vissza, a fv.g típusa megegyezik x típusával.

Pl.: előző példából

```
        WRITE (PRED (F)); {He}
        WRITE (PRED(B)); {FALSE}
        WRITE (PRED(C)); {'A'}
        SUCC(x) { SUCCESSOR }
```

x sorszámozott típusú kifejezés sorszám szerinti következő értékét adja vissza, a függvény típusa megegyezik x típusával.

Pl.: WRITE (SUCC(F); { Hu } WRITE (SUCC(C); { 'C'}

6.4.4 Egyszerű utasítások

Értékadás :=

```
Pl.: Var  L:Boolean;
          I:Integer;
          C:CHAR
          .
          .
          .
          I:= 5
          C:= 'D'; L:= True; L:= I= 8;
          { az I=8 kifejezés értéke lehet igaz vagy hamis, I értékétől függően (most hamis)}
```

Goto;

A címke deklaráló részben leírt címke szerepelhet az utasítás után, majd;.

```
Pl.:    GOTO Vege;
```

```
.
.
.
Vege: WRITE ('Itt a vég.!');
```

Mint a címke deklarálásnál szó volt róla, nem struktúrált utasítás, nem való Pascal programba!

BEGIN-END

BEGIN ... END utasítás-zárójelek

A két szó közé írt utasításokat, procedurahívásokat egy utasításként kezeli a fordító. Ennek a feltételes és ismétlő utasításokban van nagy szerepe, mert ezek csak egy(!) további utasítást tartalmazhatnak.

Programon belül az END után pontosvessző következik.

```
BEGIN
    utasítás;
    utasítás;
    {a fordító egy utasításként kezeli ezeket}
    utasítás;
End;
```

IF feltétel THEN utasítás;

IF-THEN-ELSE szerkezet vezérlés átadásra szolgál:

{ vagy utasítászárójelek között utasítás sorozat }

IF feltétel THEN utasítás

{ ELSE elé nem szabad pontosvesszőt írni! } ELSE utasítás;

CASE szerkezet

Esetszétválasztás;

```
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, v. a Word intervallumába kell esnie.

Szelektor és állandó kompatibilis típusok kell legyenek!

Pl.:

```
Var C: CHAR;
    . . .
    READLN(C);
    CASE c OF
        'A'..'Z':WRITE ('Nagybetű');
        'a'..'z':WRITE ('kisbetű');
```

```
            ELSE WRITE ('egyéb karakter');
        END;
        ...
```

6.5 Ciklusok

Elöltesztelő ciklus

WHILE belépési feltétel DO utasítás;

Pl.:
```
Var I:Integer;
    ...
    I:=0;
    WHILE I<10 DO
    BEGIN
        I:=I+1;
        WRITE(I);
    END;
```

Előfordulhat, hogy a ciklustörzs egyszer sem (!) kerül végrehajtásra.

Hátultesztelő ciklus

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

Pl.:
```
Var I:Integer;
    .
    .
    .
    I:=0;
    REPEAT
        I:= I+1;
        WRITE (I);
    UNTIL I10;
```

Egyszer mindenképpen végrehajtódik a ciklustörzs.

Elöltesztelő, léptető ciklus

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

ahol a ciklusváltozó lokális, sorszámozott típusú változó. Kezdő és végérték azzal értékadás-kompatibilis kifejezések. TO esetén ciklusváltozó eggyel nő, DOWNTO esetén eggyel csökken.

Pl.:
```
VAR I:Integer:
    .
    .
    .
FOR I:= 1 TO 10 DO WRITE (I);
```

A ciklusváltozót a ciklusmag utasítása nem módosíthatja. Sem magában a FOR utasításban, sem pedig az azt tartalmazó blokkban deklarált eljárásokban v. függvényekben nem szerepelhet tehát értékadás bal

oldalán, Read vagy Readln eljárásban, vagy másik FOR utasítás ciklusváltozójaként. A kezdő és a végértéket a gép csak egyszer határozza meg, a FOR utasításból kilépve a ciklusváltozó értéke határozatlan lesz!

6.6 Karakterláncokat kezelő eljárások, függvények

STRING típusú adatokkal végeznek különböző műveleteket. A STRING aposztrófok között max. 255 db ASCII karakter.

Pl.: Ures = ''; Fejlec = 'Készlet állomány lista';

String típusú változó: 0..255 db ASCII karakter, a hossza függ a deklarálástól.

Pl.: VAR S:STRING[80]; { 0..80 db karakter tárolására}

A zárójelekben megadott maximális hossz elhagyható, ekkor az alapértelmezés 255.

Pl.:

```
VAR S2:STRING[40];
  .
  .
  .
S2:='EZ EGY SZÖVEG';
```

Concat(S1,S2[,S3...]:String):String;

A paraméterként átadott karakterláncokat fűzi össze, ha a visszaadott karakterlánc hossza 255-nél nagyobb lenne, akkor levágja a 255. utániakat. Helyettesíthető a '+' művelettel.

Pl: WRITE (CONCAT('PA','SCA','L')); {'PASCAL'}

SOR:=CONCAT('Ez',' egy',' szöveg sor'); de egyszerübb:

SOR:='Ez'+' egy'+' szöveg sor';

Copy(Miből: String; Honnan, Mennyit: Integer): String;

A Miből karakterlánc Honnan pozíciójától számított Mennyi db karakterét, mint részkarakterláncot adja vissza. Ha Honnan nagyobb, mint Miből karakterlánc aktuális hossza, a függvény értéke üres karakterlánc lesz.

Ha a Mennyit nagyobb, mint ahány karakter a Honnan-adik pozíció után van, akkor csak a karakterlánc hátralévő karaktereit adja vissza.

Delete(VAR Miből: String; Honnan; Mennyit: Integer);

A Miből karakterlánc típusú változóból törli a Honnan-adik karaktertől számított Mennyi karaktert. Ha Honnan nagyobb, mint a Miből karakterlánc aktuális hossza, akkor nem töröl. Ha Mennyit nagyobb, mint ahány karakter a Honnan-adik karakter után van, akkor csak a hátralévő karaktereket vágja le.

Insert(Mit:String; Var Mibe: String; Index: Integer);

Mit karakterláncot beszúrja Mibe kakakterlánc változóba az Index-edik pozíciótól. Ha a beszúrás után Mibe hossza 255-nél hosszabb lenne, a 255. utániakat levágja. Ha Index nagyobb, mint Mibe aktuális hossza, akkor Mibe végéhez fűzi hozzá Mit karaktereit.

Pl.: VAR S1, S2:String; I1,I2,I3:Integer;

```
. . .
    S1:='Ez egy hosszú szöveg sor';
    S2:=Copy(S1,8,6); { S2 ='hosszú'}
    Delete (S1,8,6); { S1 = 'Ez egy szöveg sor'}
    Insert (S2,S1,8); { S1 visszakapta eredeti értékét}
```

Length(S:String):Integer;

S karakterlánc aktuális hosszát adja vissza.

Pl.: S:='karakterlánc';
WRITE(LENGTH(S); {12} .

Pos(Rész, Egesz: String):Byte;

Visszaadja, hogy a Rész karakterlánc az Egesz karakterlánc hányadik pozíciójától kezdődően található meg először. Ha nem található meg, a visszatérési érték 0.

Pl.: Rész:='ber';
Egesz:='December';
WRITE(Pos(Resz,Egesz)); {6}

Str(x[:Hossz[:Tizedes]];VAR S:String);

X egész v. valós típusú kifejezés értékét az őt reprezentáló karakterlánccá alakítja át és azt visszaadja S-ben. Hossz a mezőhosszt, Tizedes a valós típusú értékeknél a tizedesek számát adja meg. A karaktereket a megadott mezőben jobbra igazítja, bevezető szóközök után. Ha a Hossz hiányzik v. túl kicsi, akkor x-et annyi karakteren ábrázolja, amennyi szükséges.

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

S karakterlánc típusú kifejezést V típusának megfelelően numerikussá alakítva elhelyezi V egész v. valós típusú változóban. A Hiba az átalakítás során talált első hibás karakter pozícióját tartalmazza, egyébként 0. Hiba esetén V értéke nem változik. A bevezető szóközöket és tabulátor karaktereket átugorja.

Pl.: Szám:= 85.24; Egész:= 164;
Str(Szam:7:2,5);
Str(EGESZ,S);
Val(S, Egesz, Hiba);

Inc(x[,n]);

x sorszámozott változót növeli eggyel, vagy ha van, n-nel. Optimális gépi kódú rutin a fordítás eredménye, gyorsul a futás. INC(x) x=x+1 x=SUCC(x) INC(x,n) x=x+n { n egész-típusú }

Pl.: Var Szam:BYTE; . . . Szam:=5; INC(SZAM); {szám=6}
INC(SZAM,10); {szám=16}

Dec(x[,n])

x sorszámozott változót csökkenti eggyel, vagy ha van, n-nel. Szintén optimális kódot ad.

Azonos eredményt adnak a
DEC(x), x=x-1 és az x=PRED(x) művelet, illetve a
DEC(x,n) és az x=x-n.

6.7 Formázott kiírás

WRITE(változó1:n[:m][,....]);

n: hány karakterhelyre írja ki - ha kevesebb, mint a szám hossza, figyelmen kívül hagyja

m: valós számoknál a tizedesjegyek száma

Pl.:
```
VAR I:Integer;
    R:Real;
    .
    .
    .
I:= 165;
R:= 23.46;
WRITE(I:5); { 165}
WRITE(R:5:1); { 23.5}
```

6.8 Képernyő- és billentyűzetkezelő eljárások és függvények

ClrEol

A kurzor és a sor vége közötti karaktereket törli, a kurzor pozíciója nem változik.

ClrScr

Törli a képernyőt és a kurzort a bal felső sarokba viszi (1. sor, 1. oszlop). Az oszlopok száma 1 és 80 közötti, a sorok száma 1 és 25 közötti értéket vehet fel.

GoToXY(X,Y:Byte)

A kurzort az X oszlop és Y sor pozícióba teszi.

KeyPressed:Boolean

Igaz (True), ha van a billentyűzetpufferben beolvasásra váró karakter, egyébként hamis (False). A puffer tartalma nem változik.

ReadKey:Char

Ha a billentyűzetpuffer nem üres, kivesz onnan egy karaktert, ha üres vár egy billentyű leütésre, majd visszatér a kivett ill. beolvasott karakterrel. Bizonyos billentyű leütések esetében (vezérlőbillentyűk, funkcióbillentyűk) két karakter kerül a pufferbe: #0, majd a billentyű scan kódja.

WhereX:Byte

A kurzor oszloppozícióját (X)adja vissza.

WhereY:Byte

A kurzor sorpozícióját (Y) adja vissza.

Window(X1,Y1,X2,Y2)

Ablakot definiál a képernyőre, ahol

X1 a bal felső sarok oszlop koordinátája, Y1 a bal felső sarok sor koordinátája

X2 a jobb alsó sarok oszlop koordinátája, Y2 a jobb alsó sarok sor koordinátája

Max. érték: window (1,1,80,25);

Min. érték pl.: window (1,1,1,1);

TextColor(n)

Beállítja a tintaszínt a képernyőre íráshoz, **n** BYTE típusú érték, 0...15 között.

TextBackground(n)

Beállítja a háttérszínt a képernyőre íráshoz, **n** BYTE típusú érték, 0...7.

Delay(MS:WORD)

A programfutás felfüggesztését teszi lehetővé, a századmásodpercben megadott időig.

Sound(Hz:WORD)

A Hertz-ben megadott magasságú hang keltése.

NoSound

Hangkeltés vége.

6.9 Procedurák

Procedura deklarálás

A deklarációs részben történik, ezek a később deklarált procedúrákban v. a végrehajtandó részben felhasználhatók.

Procedurák lehetnek részprogramok v. szubrutinok. A procedura felépítése:

- procedura fej
- deklarációs rész
- végrehajtandó rész

Procedura fej: a PROCEDURE fenntartott szóval indul, ezután a procedúra neve, majd vagy ';' vagy zárójelek között paraméterlista, majd a ';'. Ha van paraméterlista, az itt szereplő azonosítók a procedurán belül felhasználhatók, a program és a procedura közötti illesztés a szerepük.

PROCEDURE azonosíto[([var] parameter [;[var] parameter...])];

A procedurafej akárhány paramétert tartalmazhat, ezeket formális paramétereknek hívjuk. Ha a paraméter azonosítója előtt nincs ott a VAR kulcsszó, akkor a procedura nem módosíthat a programtól kapott értéken, ezt konstans paraméternek, vagy értékparaméternek nevezzik. Ha szerepel a VAR kulcsszó, akkor változtathat, ez visszatükröződik a főprogramban is! Ezek a változó paraméterek.

A VAR szó hatása a következő pontosvesszőig tart. A procedurafejben felsorolt azonosítókat nem kell mégegyszer a deklarációs részben újra leírni.

Ha STRING v. összetett típus átadására van szükség, akkor azt a program típus-definiáló részében előre definiálni kell, az itt megadott típusazonosítót kell használni a paraméterátadásra.

Pl.:

```
PROCEDURE XY; {paraméter nélküli procedura}
PROCEDURE ABC(VAR A:BYTE); { procedura egy változó paraméterrel}
PROCEDURE IJK( SZAM:WORD; {érték paraméter}
   VAR   Q: CHAR; {változó paraméter}
   VAR   S: BYTE; {változó paraméter}
         MAX: INTEGER); {érték paraméter}
   Type Str80 = STRING [80];
   Tomb = ARRAY [3...9] OF Str80;
PROCEDURE Akarmi (S:Str80; VAR T:Tomb); { érték változó paraméter}
```

Procedúra hívása

A procedúrát annyi és olyan típusú paraméterrel kell meghívni, amennyi és amilyen típusú paramétert a procedúra fejben leírtunk.

Értékparaméterként a hívásban bármilyen kifejezés állhat, változó-paraméterként csak változó!

Hívás: névvel, ha van, akkor paraméterrel (aktuális paraméter(ek)); a paramétereket egymástól vessző (',') választja el.

Pl.:

```
Var Sorszam: WORD;
.
.
.
PROCEDURE Barmi(B: BOOLEAN; Var w: WORD);
{ Deklaráció }
.
.
.
Barmi(True,Sorszam); { Hívás }
```

A procedúra deklarációs része saját azonosítókkal is rendelkezhet (a paraméterek mellett), ezek lehetnek: típus, konstans, változó, procedúra és függvény. Minden azonosítóra, amit a procedúrán belül deklaráltunk, csak a procedúrán belül hivatkozhatunk ! (Lokális azonosítók) A procedúrán kívül leírt azonosítókra hivatkozhatunk a procedúrán belül (globális azonosítók).

Pl.:

```
Program P;
   Var Z: CHAR;
.
.
.
PROCEDURE X(K:Byte; W:WORD);
   Var A: Longint;
.
.
.
```

Procedúrán belül az A,K,W,Z azonosítók ismertek, procedúrán kívül csak a Z.

A procedúra végrehajtandó része a BEGIN-END utasítások között van, itt egyaránt szerepelhetnek egyszerű és összetett utasítások, procedúra és függvény hívások.

Pl.:

```
PROCEDURE P;
BEGIN
   utasítások;
   .
   .
   .
END;
```

6.10 Menük készítése

A menük általában több funkcióval rendelkeznek, ezek közül az egyik funkció a kilépés.

A kiválasztás módja sokféle lehet: kezdőbetűvel, sorszámmal, nyilakkal, egérrel, piktogrammal stb.

A menübe való belépés után legalább egyszer választani kell (REPEAT-UNTIL ciklus).

A menüből való kilépés feltétele: a menű kilépés funkciójának választása.

Mindig jelezni kell:

- hogyan lehet az egyes funkciókat kiválasztani,
- valamint azt is, hogy milyen szinten vagyunk: menüben v. valamely kiválasztott funkcióban.

A választás általában a CASE szerkezet segítségével történik.

Pl.:

```
.
.
.
REPEAT
   - a választható funkciók megjelenítése
   - a választás módjának megjelenítése
   - funkciókód bekérdése
   CASE funkciókód of
      1. funkciókód: 1. rutin;
      2. funkciókód: 2. rutin;
      .
      .
      .
      n. funkciókód: n. rutin;
   END;
UNTIL kilépési-feltétel;
.
.
.
```

Feladat: Olvasson be két számot, majd jelenítse meg az alábbi menüt:

ÖSSZEG = 1
KÜLÖNBSÉG = 2
SZORZAT = 3
ÁTLAG = 4
KILÉPÉS = 5
ÜSSE BE A VÁLASZTÁSÁT:

Legyen lehetőség a folyamatos választásra, a kiválasztott tevékenységet elvégezve adjon tájékoztatást az eredményről, majd térjen vissza a menühöz. A program befejezése csak a KILÉPÉS menüpont kiválasztásával legyen lehetséges.

```
Program Menü;
Uses CRT;
VAR SZAM1,SZAM2:REAL; gomb:CHAR;
PROCEDURE Osszeg;
   BEGIN
      WRITE('A két szám összege:',(Szam1+Szam2):20:4);
   END;
PROCEDURE Szorzat;
   BEGIN
      WRITE('A két szám szorzata:',(Szam1*Szam2):20:4);
   END;
PROCEDURE Kulonbseg;
   BEGIN
      WRITE('A két szám különbsége:',(Szam1-Szam2):20:4);
   END;
PROCEDURE Atlag;
   BEGIN
      WRITE('A két szám átlaga:',((Szam1+Szam2)/2):20:4);
   END;
PROCEDURE Menu;
   VAR valaszt:CHAR;
   BEGIN
      REPEAT
         CLRSCR;
         GOTOXY(25, 4); WRITE('ÖSSZEG = 1');
         GOTOXY(25, 6); WRITE('KÜLÖNBSÉG = 2');
         GOTOXY(25, 8); WRITE('SZORZAT = 3');
         GOTOXY(25,10); WRITE('ÁTLAG = 4');
         GOTOXY(25,12); WRITE('KILÉPÉS = 5');
         GOTOXY(25,14); WRITE('ÜSSE BE A VÁLASZTÁSÁT:');
         REPEAT
            valaszt:=READkey;
         UNTIL valaszt in ['1'..'5'];
         GOTOXY(5,18);
         CASE valaszt of
            '1':Osszeg;
```

```
                '2':Kulonbseg;
                '3':Szorzat;
                '4':Atlag;
            END;
            GOTOXY(5,20); WRITE('Üssön le egy billentyűt a folytatáshoz');
            gomb:=READkey;
        UNTIL valaszt ='5';
    END;
BEGIN
    Menu;
END.
```

6.11 Tömbtípus

A típus leírása az **ARRAY** kulcsszóval indul, amely után szögletes zárójelek között adjuk meg a dimenziókat, egymástól elválasztva; minden dimenzió-leírás egy intervallum, a megadott két határérték között címezhetjük majd a tömb valamely elemét. Ezután az OF kulcsszó következik, majd a tömb alaptípusa, mely bármilyen lehet.

ARRAY [alsó..felső [,alsó..felső...]] OF alaptípus;

A tömb bármennyi dimenzióval rendelkezhet. A dimenzió határaként csak sorszámozott típusú állandók adhatók meg, kivéve a LONGINT típust.

Pl.: VAR TOMB:ARRAY [1..10] OF BYTE;

T:ARRAY [5..8, 99..122] OF CHAR;

T2:ARRAY ['A'..'Q'] OF WORD;

Irásmód: ARRAY [1..9] OF ARRAY [3..5] OF REAL
ugyanaz, mint az
ARRAY [1..9, 3..5] OF REAL

6.11.1 Tömb adott elemének címzése

- az azonosító után szögletes zárójelek között írjuk be az indexet (vagy indexeket, egymástól vesszővel elválasztva).
- az azonosító után minden indexet szögletes zárójelek között írunk egymás után.

Pl.: Var T:ARRAY [4..8, 2..10] OF BYTE;

T [4,2]= T[4][2] {a tömb első eleme}

A tömbök kezelésénél komoly gondot jelent, hogy az elemek száma konstans, ez már a program futása előtt eldől, amikor megadjuk az alsó és felső index értékeket. A futás során ezen már nem tudunk változtatni, de ennek ellenére a tömbkezelő eljárásainkat és függvényeinket elég általánosan meg tudjuk írni, így akár különböző programokban is tudjuk használni őket. Törekedjünk arra, hogy lehetőleg minél több információt adjunk át paraméterekkel.(Pl.: ne csak a tömböt, hanem az index alsó és felső határát is !).

Globális objektumok is segítenek az általánosításban - elsősorban a típusdeklarálással. Alprogramjaink ugyanis hivatkozhatnak egy közös "tömbtípus"-ra és az alaptípust sem kell az alprogramban rögzíteni.

Pl.:

```
CONST   MinIndex = 1;
        MaxIndex = 100;
TYPE    IndexTipus = MinIndex ..MaxIndex;
        ElemTipus = INTEGER;
        TombTipus = ARRAY[IndexTipus] OF ElemTipus;
VAR     A,B,C:TombTipus;
        i:WORD;
PROCEDURE tombini(VAR tomb:TOMBTIPUS;also,felso: WORD;ertek:ELEMTIPUS);
    VAR I:WORD;
    BEGIN
        IF also<MinIndex THEN also:=MinIndex;
        IF felso>MaxIndex  THEN felso:= MaxIndex;
```

```
        FOR i:= also TO felso DO
            tomb[i]:=ertek;
    END;
BEGIN
    TombIni(A,MinIndex,MaxIndex,0);
    TombIni(B,10,45,6);
    TombIni(C,50,MaxIndex,ORD('A'));
    FOR i:= MinIndex TO MaxIndex DO
        WRITELN(A[i]:10, B[i]:10, C[i]:10);
END.
```

6.11.2 Adatfeldolgozás tömbökben

Általános adatfeldolgozási funkciók:

- adat felvitele
- adat módosítása
- adat törlése
- adat keresése
- adatok rendezése

Tulajdonságok

- a tömb mérete korlátozott (így a felvihető adatok száma is!)
- a tömb rendezetlen
- a felvitt adatok száma minden pillanatban ismert
- az adatok a program futása során érhetők csak el, a futás után elvesznek!

Programozási megfontolások

- az adatok egyediek v. duplikáltak-e?
- az egyes, összetartozó adatelemeket egy sokdimenziós tömbben, vagy több, egydimenziós tömbben (vektorban) tároljuk?
- az adat melyik eleme lesz az azonosító? (Az azonosítónak mindig egyedinek kell lennie, hogy egyértelműen azonosítsa az adatot, mint egészt)

6.11.3 Az egyes funkciók áttekintése

Felvitel

- ellenőrzés: Van-e még hely a tömbben? ha nincs, hiba: nem vihető be az adat
- azonosító bekérése
- azonosító ellenőrzése: Van már ilyen a tömbben? ha igen, hiba: nem vihető fel mégegyszer az adat (a duplikáció tiltott!)
- az adat többi részének bekérése
- adat felvitele a tömbbe
- tömbmutató növelése

Módosítás

- ellenőrzés: Van-e adat a tömbben? ha nincs, hiba: nincs mit módosítani
- azonosító bekérése
- ellenőrzés: Van ilyen azonosító? ha nincs, hiba: nincs mit módosítani
- az adat többi elemének módosítása
- az adat többi elemének bevitele a tömbbe

Törlés

- ellenőrzés: Van-e adat a tömbben? ha nincs, hiba: nincs mit törölni
- azonosító bekérése
- ellenőrzés: Van ilyen azonosító? ha nincs, hiba: nincs mit törölni
- a tömb hátralévő adatainak előre másolása
  (vagy az utolsót átmásolni a törlendő helyére - gyorsabb)
- a tömb mutató csökkentése

Keresés

- ellenőrzés: Van-e adat a tömbben? ha nincs, hiba: nincs mit keresni
- azonosító bekérése
- ellenőrzése: Van ilyen azonosító? ha nincs, hiba: nincs a keresett elem
- az adat többi elemének megjelenítése

6.11.4 Adatkezelési mintafeladat

```
PROGRAM TombAdat; {Adatkezelés tömbökben}
Uses CRT;
CONST   MaxAdat = 5;
TYPE    NevTipus = STRING[30];
        NevTombTipus = array[1..MaxAdat] OF NevTipus;
        CimTipus = STRING[40];
        CimTombTipus = array[1..MaxAdat] OF CimTipus;
        SzSzTipus = STRING[11];
        SzSzTombTipus = array[1..MaxAdat] OF SzSzTipus;
        FizTipus = LONGINT;
        FizTombtipus = array[1..MaxAdat] OF FizTipus;
VAR     NevTomb:NevTombTipus; {Nevek tárolására}
        CimTomb:CimTombTipus; {Címek tárolása}
        SzSzTomb:SzSzTombTipus; {Személyi számok}
        FizTomb:FizTombTipus; {Fizetések tárolására}
        SzemSzam:SzSzTipus; {azonosító bekéréséhez}
        i,Hiba:LONGINT {i ciklusváltozó, Hibaszám STRING - LONGINT}
        AdatSzam:INTEGER; {a tömbben lévő adatok száma}
        valaszt:CHAR; {választás a menüben}
        FizStr:STRING; {fizetés beolvasásához}
PROCEDURE Varakozo;
  VAR varakozik: CHAR;
  BEGIN
    GOTOXY(20,25);
    Write('Folytatás billentyű lenyomásra!');
    Varakozik:=READkey;
  END;
PROCEDURE TombIni;
  VAR i: BYTE;
  BEGIN
    FOR i:= 1 TO MaxAdat DO BEGIN
      SzSzTomb[i]:='';
      NevTomb[i]:='';
      CimTomb[i]:='';
      FizTomb[i]:=0;
    END;
  END;
FUNCTION IgenNem: BOOLEAN;
  VAR valaszt2: CHAR;
  BEGIN
    REPEAT
      valaszt2:= upCASE (READkey);
    UNTIL valaszt2 in ['I','N'];
    IgenNem:=valaszt2='I';
  END;
PROCEDURE Felvitel;
  BEGIN
    IF AdatSzam=MaxAdat THEN BEGIN
      GOTOXY(5,24); WRITE('Nem vihető fel több adat!');
      varakozo;
      end
    ELSE BEGIN
      CLRSCR;
      GOTOXY(5,5); WRITE('Felvitel');
      GOTOXY(5,8);WRITE('Kérem a személyi számot: ');READLN (SzemSzam);
      i:=1;
      WHILE (iAdatSzam) and (SzSzTomb[i]SzemSzam) do
      INC(i);
      IF SzSzTomb[i]=SzemSzam THEN BEGIN
        GOTOXY(5,24); WRITE ('Van már ilyen azonosító');
        varakozo;
        END
      ELSE BEGIN
```

```
                INC(AdatSzam);
                GOTOXY(50,10); WRITE ('Kérem a nevet :');
                READLN (NevTomb[AdatSzam]);
                GOTOXY(5,12); WRITE ('Kérem a címet :');
                READl(CimTomb[AdatSzam]);
                GOTOXY(5,14); WRITE ('Kérem a fizetést :);
                REPEAT
                    GOTOXY(30,14); clreol;
                    READLN(FizStr);
                    VAL(FizStr, FizTomb[AdatSzam],Hiba);
                UNTIL Hiba = 0;
                SzSzTomb[AdatSzam]:= SzemSzam;
                END;
            END;
        END;
PROCEDURE Modositas;
    BEGIN
        IF AdatSzam0 THEN BEGIN
            CLRSCR;
            GOTOXY(5,5); WRITE('Módosítás');
            GOTOXY(5,8); WRITE('Kérem a személyi számot:');
            READLN (SzemSzam);
            i:=1;
            WHILE (iAdatSzam) AND (SzemSzamSzSzTomb[i]) DO
                INC(i);
            IF SzemSzamSzSzTomb[i] THEN BEGIN
                GOTOXY(5,24); WRITE ('Nincs még ilyen azonosító!');
                Varakozo;
                END
            ELSE BEGIN
                GOTOXY(5,10);
                WRITELN ('A név:',NevTomb[i],'Módosítja? (I/N]');
                IF IgenNem THEN BEGIN
                    GOTOXY(5,11); WRITE ('Az új név:');
                    READLN (NevTomb[i]);
                    END;
                GOTOXY(5,13);
                WRITELN('A cim: ',CimTomb[i],' Módosítja? [I/N]')
                IF IgenNem THEN BEGIN
                    GOTOXY(5,13); WRITE('Az új cím:');
                    READLN (CimTomb[i];
                    END;
                GOTOXY(5,14);
                WRITELN('A fizetés:',FizTomb[i],' Módosítja? [I/N]');
                IF IgenNem THEN BEGIN
                    GOTOXY(5,15);
                    WRITE('Az új Fizetés:');
                    REPEAT
                        GOTOXY(20,15); clreol;
                        READLN(FizStr);
                        VAL(FizStr,FizTomb[i], Hiba);
                    UNTIL Hiba=0;
                    END;
            END;
        END
        ELSE BEGIN
            GOTOXY(5,24); WRITE('Nincs adat a tömbben!');
            Varakozo;
            END;
    END;
PROCEDURE Torles;
    BEGIN
        IF AdatSzam0 THEN BEGIN
        clrsr;
        GOTOXY(5,5); WRITE('Törlés');
        GOTOXY(5,8); WRITE('Kérem a személyi számot:');
        READLN(SzemSzam);
        i:=1;
        WHILE (iAdatSzam) and (szemSzamSzSzTomb[i]) do
        INC(i);
        IF SzemSzamSzSzTomb[i] THEN BEGIN
            GOTOXY(5,24);
            WRITE('Nincs még ilyen azonosító!');
            Varakozo;
                END
```

```
            ELSE BEGIN
                GOTOXY(5,10); WRITE('A név: ',NevTomb[i]);
                GOTOXY(5,12); WRITE('A cím: ',CimTomb[i]);
                GOTOXY(5,14); WRITE('A fizetés:',FizTomb[i]:10);
                GOTOXY(5,16); WRITE('Ezt akarja törölni? [I/N]');
                IF IgenNem THEN BEGIN
                    SzSzTomb[i]:=SzSzTomb[AdatSzam];
                    NevTomb[i]:=NevTomb[AdatSzam];
                    CimTomb[i]:=CimTomb[AdatSzam];
                    FizTomb[i]:=FizTomb[AdatSzam];
                    DEC(AdatSzam);
                    END;
                END;
            END
            ELSE BEGIN
                GOTOXY(5,24); WRITE(Nincs adat a tömbben!');
                Varakozo;
                END;
            END;
    PROCEDURE Kereses;
        BEGIN
            IF AdatSzam0 THEN BEGIN
                CLRSCR;
                GOTOXY(5,5); WRITE('Kérem a személyi számot:');READLN(SzemSzam);
                i:=1;
                WHILE (iAdatSzam) AND (SzemSzamSzSzTomb[i]) DO
                    INC(i);
                IF SzemSzamSzSzTomb[i] THEN BEGIN
                    GOTOXY(5,24); WRITE('Nincs még ilyen azonosító!');
                    Varakozo;
                    END
                ELSE BEGIN
                    GOTOXY(5,10); WRITE('A név:',NevTomb[i]);
                    GOTOXY(5,12); WRITE('A cím:',CimTomb[i]);
                    GOTOXY(5,14); WRITE('A fizetés:',FizTomb[i]10);
                    Varakozo;
                    END;
                END
            ELSE BEGIN
                GOTOXY(5,24); WRITE('Nincs adat a tömbben!');
                Varakozo;
                END;
            END;
    BEGIN {főprogram}
        TombIni;
        AdatSzam:=0;
        REPEAT
            CLRSCR;
            GOTOXY(25, 8); WRITE('Felvitel = F');
            GOTOXY(25,10); WRITE('Módosítás = M');
            GOTOXY(25,12); WRITE('Törlés = T');
            GOTOXY(25,14); WRITE('Keresés = K');
            GOTOXY(25,16); WRITE('Vége = V');
            GOTOXY(25,18); WRITE('Üsse be a választását:');
            REPEAT
                valaszt:= UPCASE(READKEY);
            UNTIL valaszt in ['F', 'M', 'T', 'K', 'V',];
            CLRSCR;
            CASE valaszt of
                'F': Felvitel;
                'M': Modositas;
                'T': Torles;
                'K': Kereses;
            END;
        UNTIL valaszt= 'V';
    End.
```

6.11.5 Forward deklarálás

Bizonyos esetekben előfordulhat, hogy két procedura egymást hívja. Alapesetben ez nem lehetséges, hisz csak olyan azonosítókra hivatkozhatunk, ami már ismert a fordító számára. Ezt az ellentétet oldja fel a **FORWARD** deklaráció. Egymás után deklarálva a P1 és P2 procedurákat, akkor a később deklarált P2

hívhatja P1-t, de fordítva nem. Ahhoz, hogy ez lehetséges legyen, az elsőt FORWARD-ként kell deklarálnunk:

- a procedura-fej után a FORWARD kulcsszó áll,
- a deklarációs részt később írjuk le (de még a deklarációs részben).

```
PROCEDURE P2(paraméterek); FORWARD; P2;
PROCEDURE P1;
    BEGIN
        .
        .
        . P2(paraméterek);
        .
        .
        .
    End;
PROCEDURE P2;
    BEGIN
        .
        .
        .
    END;
```

* A második deklarációnál már csak a PROCEDURE szó és a név áll, ha van paraméter is, akkor az(ok) az első leírásban szerepelnek.

6.11.6 Rekurzív eljárások

A Pascal-ban lehetséges az, hogy egy procedúra v. függvény önmagát hívja meg, vagyis a rekurzivitás.

Pl.:

```
Program P;
Var I: INTEGER;
PROCEDURE kiiro(N:INTEGER);
    BEGIN
        IF N0 THEN BEGIN
            WRITE('-');
            kiiro(N-1);
        END;
    END;
BEGIN
    Write ('Mennyi kötőjelet (-) írjunk?');
    READLN (I);
    kiiro(I);
END.
```

FIGYELEM: A rekurziónak véget kell vetni, különben "elszáll" a program!

6.12 Függvények

Tulajdonképpen programrészlet (mint a PROCEDURE), a rá való hivatkozáskor (nevének leírásakor) végrehajtódik, ekkor a függvény értéket kap, ez az érték a hivatkozó kifejezésben a hivatkozás helyére behelyettesítődik. Függvény nem állhat értékadó kifejezés bal oldalán (kivéve, amikor értéket kap a saját végrehajtandó részében!).

A procerúrához hasonlóan három fő részből áll:

- függvény fej
- deklarációs rész
- végrehajtandó rész

A függvényfejet a FUNCTION kulcsszó vezeti be, melyet a függvény általunk választott neve követ. Ezután v. a fejet záró típusmegadás, majd a ";" következik, vagy a paraméterek zárójelben, majd típusmegadás, majd a ";".

FORMÁTUM: FUNCTION fv.név:fv. típus;

vagy

FUNCTION fv. név (P1[,P2...]):fv. típus;

A függvény típusa csak egyszerű típus és string lehet. Ha van paraméterlista, az itt leírt azonosítók a függvényen belül felhasználhatók, feladatuk a program és a függvény közötti illesztés.

A végrehajtandó részben legalább egyszer értéket kell adni a függvénynek, különben határozatlan lesz a visszatérítési értéke!

Pl.:

```
FUNCTION Egyenlok (A,B: WORD): BOOLEAN;
BEGIN
Egyenlok:= A=B;
END;
```

Bonyolultabb függvények esetén célszerű a függvényen belül deklarálni egy ugyanolyan típusú változót, mint a függvény típusa, majd ezt használni a végrehajtandó részben. A függvény utolsó utasításaként a függvény nevét egyenlővé tesszük a változóval - ezzel elkerülhetjük a nem kívánt rekurziót!

Pl.:

```
PROGRAM p;
Type str11=STRING[11];
VAR RSzam:REAL;
FUNCTION RealToStr(R:REAL):Str11;
    VAR S: Str11;
    BEGIN
        STR(R:11:2,S);
        RealToStr:=S;
    END;
BEGIN
    RSzam:= Pi;
    WRITE('A szám: ', RealToStr(RSzam));
END.

Program Hatvany;
Uses CRT;
VAR Alap:REAL;
        Kitevó:WORD;
Function Hatvanyozo(A: REAL; K:WORD): REAL;
    VAR Z: REAL;
    BEGIN
        Z:=A;
        WHILE K1 DO BEGIN
            Z:=Z*A;
            DEC(K);
        END;
        Hatvanyozo:=z;
    END;
BEGIN
    CLRSCR;
    WRITE ('Írja be az alapot:');READLN (Alap);
    WRITE ('Írja be a kitevöt:');
    READLN (kitevö);
    WRITELN ('A(z) ',Alap:0:2,kitevo:5,'.hatvanya: ',
    Hatvanyozo(Alap,Kitevo):15:3);
END.
```

Rekord típus

A rekord olyan struktúrált típus, mely adott számú és típusú komponensből épül fel. Ezeket a komponenseket mezőknek nevezzük.

Definiálása: az azonosító után egyenlőségjel, majd a 'RECORD' kulcsszó, azután a mezők felsorolása, végül az 'END' szó és egy ';'.

Pl.:

```
Program P;
Type  AruTipus=RECORD
            CikkSzam:LONGINT;
```

```
            AruNev:STRING[40];
            Mennyiseg:LONGINT;
            Egyseg:STRING[3];
            Ar: REAL;
        END;
Var Aru1, Aru2: AruTipus;
BEGIN
    WITH Aru1 DO BEGIN
        CikkSzam:= 11415;
        Arunev:='Reszelő';
        Mennyiseg:= 415;
        Egyseg:= 'DB';
        Ar:= 26.50;
    END;
    Aru2:= Aru1;
    WRITE ('Egységár:');
    READLN (Aru2.Ar);
    WRITELN (Aru2. Mennyiseg * Aru2.Ar);
END.
```

CikkSzam, Arunev, Mennyiseg, Egyseg, Ar: mező azonosítók, Aru1 és Aru2 AruTipus típusú rekordok.

A rekordok mezőire hivatkozás: Rekordnév.Mezőazonosító.

Ha egy sok mezőből álló rekord mezőire akarunk hivatkozni, sorban egymás után, a rekordnév többszöri leírása helyett használjuk a WITH utasítást:

WITH Rekordnév DO {BEGIN} utasítás {END}; utasítás:

BEGIN-END utasítás-zárójelpár között utasítás sorozat is lehet.

Mint a példából látható, a rekordok mezői értéket a billentyűzetről, közvetlen értékadó utasításokkal kaphatnak; egy rekord összes mezője kaphat értéket egy utasítással: Aru2:=Aru1; ekkor Aru2 összes mezője sorban felveszi Aru1 mezőinek értékét.

A rekordok egymásba is ágyazhatók:

```
Type SzulIdő = RECORD
        Ev, Ho, Nap: WORD;
    END;
    Szemely = RECORD
        Nev: STRING[30];
        SzulDat: SzulIdo;
        Cim: STRING [30];
    END;
```

6.13 Adatkezelés tömb és rekordtípusok segítségével

Alakítsuk át az adatkezelő programunkat, hogy ne 4 külön tömbbel dolgozzunk, hanem egy, rekordokból felépülővel.

```
PROGRAM RekTomb;
USES CRT;
Const MaxAdat= 5;
Type  SzSzTipus = STRING[11];
      SzemelyTipus = RECORD
          SzSz:SzSzTipus;
          Nev:STRING[30];
          Cim:STRING[40];
          Fiz:LONGINT;
      END;
      SzemelyTombTipus=array [1..MaxAdat] OF SzemelyTipus;
VAR Szemelyek: SzemelyTombTipus;
      SzemSzam: SzSzTipus;
.
. {többi megegyezik a Felvitel proceduráig}
.
PROCEDURE TombIni;
   VAR i: WORD;
   BEGIN
```

```
        FOR i:= 1 TO MaxAdat do
        WITH Személyek[i] DO BEGIN
            SzSz:='';
            Nev:='';
            Cim:='';
            Fiz:=0;
        END;
    END;
PROCEDURE Felvitel;
    BEGIN
        IF AdatSzam= MaxAdat THEN BEGIN
            GOTOXY(5,24);WRITE('Nem vihető fel több adat!');
            Varakozo;
        END
        ELSE BEGIN
            CLRSCR;
            GOTOXY(5,5);WRITE('Felvitel');
            GOTOXY(5,8);WRITE('Kérem a személyi számot:');READLN (SzemSzam);
            i:=1;
            WHILE (i Adatszam) and (SzemSzamSzemelyek[i].SzSz) DO  INC(i);
            IF SzemSzam=Szemelyek[i].SzSz THEN BEGIN
                GOTOXY(5,24);WRITE('Van már ilyen azonosító!');
                Varakozo;
            END
            ELSE BEGIN
                INC(AdatSzam);
                WITH Szemelyek [AdatSzam] DO BEGIN
                    SzSz:= SzemSzam;
                    GOTOXY(5,10); WRITE ('Kérem a nevet:');READLN (Nev);
                    GOTOXY(5,12); WRITE ('Kérem a címet:');READLN (Cím);
                    GOTOXY(5,14); WRITE ('Kérem a fizetést:');
                    REPEAT
                        GOTOXY(23,14); clreol;
                        READLN(FizStr);
                        VAL(FizStr, Fiz, Hiba);
                    UNTIL Hiba = 0;
                END;
            END;
        END;
    END;
```

6.14 Típusos állományok

A lemezen tárolt adatokat file-okban helyezzük el. A típusos-file azonos típusú komponensekből felépülő adatszerkezet, ahol a komponens bármilyen típusú lehet - kivéve a file típust (vagyis file-okat tartalmazó file-t nem hozhatunk létre). A komponensek számának csak a tárolókapacitás szab határt.

A file-műveletek közül a legfontosabb az írás (output) és az olvasás (input).

Az input művelet a file-ból a memóriába olvas egy komponenst, az output fordítva: a memóriából a file-ba ír ki egy komponenst. Az I/O (input/output) műveleteknél a rendszer egy közbülső memória-területet (puffert) használ, ezen keresztül bonyolódnak le a háttértár műveletek.

A puffer kezelése általában automatikus, a standard eljárások elvégzik helyettünk.

A file fontos jellemzője az aktuális file-pozíció, mely megadja, hogy a következő I/O művelet mely komponensen hajtódik végre. Az aktuális file-pozíciót egy ún. file-mutató jelzi.

Deklarálása: **azonosító:FILE OF típus;**

Pl.:

```
Var F:FILE OF BYTE;
```

vagy

```
Type TombType = array[1...10] OF STRING[30];

Var   TombFile:file OF TombType;
vagy
Type  SzemAdatTipus = record
      SzemSzam:str[11];
      Nev:str[25];
```

```
        Cim:str[30];
        Fizetes:WORD;
END;
Var AdatFile:file OF SzemAdatTipus;
```

6.14.1 Típusos állományok feldolgozása

1. Soros (szekvenciális) feldolgozás:

A file megnyitása írásra vagy olvasásra történhet, ha még nem létezik, akkor létre kell hozni a megnyitás előtt. Ha írásra nyitottunk egy file-t, akkor abból olvasni csak úgy tudunk, hogy előbb lezárjuk, majd megnyitjuk olvasásra. Olvasásra nyitott file- ba írni csak lezárás utáni írásra megnyitással lehet. A file-ba íráskor a rekordok folyamatosan követik egymást, nincs köztük semmilyen elválasztójel, az írás mindig a file végére történik. Hasonlóan, olvasáskor folyamatosan olvassuk be a rekordokat, az elsőtől az utolsóig. A módosítás nehézkes művelet: megnyitjuk olvasásra az adatfile-t, létrehozunk egy átmeneti állományt, megnyitjuk írásra, majd az első rekordtól egyenként olvassuk az adatállományt, ha nem módosítandó rekord, kiírjuk az átmeneti állományba. Ha megtaláltuk a módosítandó rekordot, elvégezzük a módosítást, kiírjuk az átmeneti állományba. Ezután átírjuk a hátralevő rekordokat is az átmeneti állományba, lezárjuk, majd kitöröljük az eredeti adatállományt, lezárjuk az átmeneti állományt és átnevezzük az eredeti nevére. A nagyobb adatbiztonság miatt, az eredetit sem töröljük ki, csak átnevezzük!

A törlés is hasonlóan történik, különbség csak annyi, hogy ha megtaláltuk a törlendő rekordot, azt átugorva folytatjuk az átírást. Nem igazán "kellemes" felhasználási mód, nehézkes, kényelmetlen és lassú!

2. Közvetlen hozzáférésű (direkt, random) állományok feldolgozása

A szavak szótárban való kereséséhez hasonlóan, a közvetlen hozzáférésű állományok bármelyik rekordjához, bármilyen sorrendben hozzáférhetünk. A soros állományoktól eltérően, ezeket az állományokat egyidejűleg tudjuk írni, olvasni, mivel a file-t alkotó komponensek mérete ismert, ezért a file-mutatót a file bármely elemére pozícionálhatjuk, majd ezt beolvashatjuk, vagy a kiválasztott helyre írhatunk. A módosításnál jelentkezik igazán az előnye ezeknek a file-oknak, hisz a megtalált módosítandó rekordnak megjegyezzük a pozícióját, elvégezzük a módosítást, majd a megfelelő pozícióra visszaírjuk a rekordot. Nincs szükség átmeneti állományra!

6.14.2 Típusos file-kezelő eljárások és függvények

ASSIGN (VAR f; nev:STRING);

A file deklarációban egy logikai nevet, a file típusú változót rendeltük a file-hoz, ezzel a változóval fogunk a file-ra hivatkozni a program futása során. Ezt a változót össze kell kapcsolni egy ténylegesen létező fizikai file-lal, amely valamely tárolón található (floppy, winchester,...). Az ASSIGN ezt az összekapcsolást végzi el, így ha a későbbiekben a logikai file-ra hivatkozunk, az a hozzárendelt fizikai file-ra vonatkozik. Az összekapcsolás addig érvényes, amíg új fizikai file-t nem rendelünk a logikai névhez.

f: a file változó, név: a fizikai név, lehet string konstans v. string típusú változó. A fizikai névre a DOS szabályai érvényesek.

RESET(VAR f);

Használat előtt a file-okat meg kell nyitni, a már létező file-ok megnyitását végzi el az eljárás. Használata előtt az ASSIGN-nal el kell végezni a logikai és fizikai file-ok egymáshoz rendelését. Nem létező file-ra alkalmazáskor hibajelzést kapunk. Már megnyitott file-ra alkalmazva a file lezárul, majd újra megnyílik. Az aktuális file-pocíciót mutató file-mutató a file első elemére - a 0.-ra - fog mutatni.

REWRITE(VAR f);

Használat előtt a file-okat meg kell nyitni, az eljárás új file-ok létrehozására szolgál, már létező file-okra alkalmazva, akkor annak tartalma elvész és új, üres file-ként jön létre újra. Használata elött az ASSIGN-t használni kell! A létrehozott file mutatója az üres file kezdetére áll.

READ(VAR f; VAR k1 [,k2,...]);

Beolvassa a file-ból azt a komponenst, melyre a file-mutató mutat. Az eljárást végrehajtva a file-mutató a következő komponensre fog mutatni.

k1 [,k2...]: olyan változó(k), mely(ek) azonos típusú(ak) a file- változó típusával.

Pl.:

```
VAR     f:file OF LONGINT;
        k1,k2:LONGINT;
```

WRITE(VAR f; k1[, k2...]);

Kiírja a file-ba a paraméterként kapott változó(k) tartalmát, attól a komponenstől kezdve, amelyre a file-mutató mutat. Ha a file-mutató az utolsó rekord mögé mutat, hozzáfűzés történik. Az eljárást végrehajtva, a file-mutató a következő komponensre fog mutatni.

k1 [, k2...]: olyan változó(k), mely(ek) azonos típusú(ak) a file-változó típusával.

CLOSE (VAR f);

Lezárja a paraméterben megadott file-változóhoz rendelt fizikai file-t, a lemez katalógusa aktualizálódik. Csak a RESET v. REWRITE eljárásokkal megnyitott file-okat lehet lezárni!

SEEK (VAR f; v: LONGINT);

A rendszer minden file-hoz egy mutatót rendel, mely a file aktuális elemére mutat. Mivel a file-t alkotó elemek hossza azonos, könnyen kiszámítható, hogy az egyes komponensek hol kezdődnek. A SEEK eljárás kezeli a file-mutatót, lehetővé teszi a komponensek közvetlen elérését, azaz a direkt file-kezelést. A file-mutató minden I/O művelet (READ/WRITE) után a file következő elemére mutat, míg a SEEK eljárás segítségével bármely, általunk megadott sorszámú elemet aktuálissá tehetünk (vagyis arra fog mutatni a file-mutató).

v: az aktuálissá teendő komponens sorszáma

```
Pl.:    SEEK( f, 0 );  { a file első eleme lesz az aktuális }
```

Megjegyzendő, hogy a logikai számozás 0-val kezdődik, míg a fizikai 1-el!

FILEPOS(VAR f):LONGINT;

A file-mutató pillanatnyi értékét adja vissza a függvény, azaz jelzi, hogy pillanatnyilag hányadik elem feldolgozásánál tartunk. A sorszámozás 0-val kezdődik.

FILESIZE(VAR f):LONGINT;

A függvény értéke a file elemeinek száma.

```
Pl.: SEEK(f,FILESIZE(f)); { file végére pozícionálás }
```

EOF(VAR f): BOOLEAN;

A függvény visszatérési értéke igaz (TRUE), ha a file-mutató az utolsó komponens után áll, egyébként FALSE. Üres file-okra mindig TRUE.

```
Pl.:    IF EOF(f) THEN
            vagy
        WHILE  NOT EOF(f) DO
```

IORESULT : INTEGERé

Input/Output műveletek eredményességét vizsgáló függvény. Egy, a fordítónak szóló utasítás (compiler direktíva) segítségével beállíthatjuk, hogy I/O hiba esetében a program futása megszakadjon-e vagy sem.

Az I/O hibák figyelését kikapcsolhatjuk: a {$I -} kombinációval, ekkor nem történik hibajelzés, a program futása folytatódhat, de az I/O műveletek végrehajtása felfüggesztődik, míg a hiba okát az IORESULT függvény segítségével fel nem derítjük. (Hibátlan végrehajtás esetén is meg kell hívni a függvényt!). A függvény lekérdezi a SYSTEM egység INOUTRES nevű változójának az értékét, (ez lesz a visszatérési értéke), majd nullázza azt. Az I/O műveletek addig függesztődnek fel, amíg ennek a változónak az értéke nem 0.

Hiba kódok (az IORESULT visszatérési értéke)

- 0 nincs hiba
- 2 file nem létezik
- 3 útvonal nem létezik
- 4 túl sok állomány van egyszerre nyitva
- 5 tiltott állomány-hozzáférés
- 100 olvasás file-vég után
- 101 írási hiba - pl.:betelt a lemez
- 102 elmaradt ASSIGN 103 nincs nyitva az állomány

Az I/O hibák figyelésének bekapcsolása a {$I+} direktívával történik, ha ezután I/O hiba lép fel, a program futási hibával leáll.

Pl.:

```
..
ASSIGN(f,'PROBA. XXX');
{$I-} RESET(F); {$I+}
IF IORESULT0 { Megnyitási hiba } THEN REWRITE(f);
```

ERASE(VAR f);

Törli a lemezről a file-t. A file nem lehet nyitva, de az összerendelést az ASSIGN-nal el kell végezni!

RENAME(VAR f; újnév:STRING);

File átnevezése

f: olyan file változó, melyre az ASSIGN-t elvégeztük

újnév: az új nevet (útvonallal is lehet) tartalmazó stringkonstans v. string típusú változó.

TRUNCATE(VAR f);

A file aktuális pozíciójától elvágja a file-t, azaz a file-mutatótól kezdődően minden elemet töröl a file-ból. A file-mutató ezután a file végét jelöli. A file-nak nyitva kell lennie az eljárás alkalmazásakor.

FILEMODE:BYTE;

A SYSTEM unitban deklarált típusos konstans. A FileMode BYTE tükrözi a megnyitott file I/O státuszát:

FILEMODE =0: csak olvasható
=1: csak írható
=2:írható/olvasható

A RESET eljárás figyelembe veszi az értékét, a REWRITE nem: értéke mindenképpen 2 lesz.

6.14.3 Adatkezelés típusos file-okban

Funkciók:

- új adat felvitele
- meglévő adat módosítása
- adat törlése - logikai - fizikai
- adat kikeresése
- adatok listázása
- adatok rendezése - később

Tulajdonságok

- az adatok mennyiségét a háttértár mérete korlátozza!
- az állományban lévő adatok mennyisége lekérdezhető
- az adatok futása után is megmaradnak!

Programozási megfontolások

- duplikáció megengedett?
- rekordleírás elkészítése
- az adat mely eleme(i) legyen(ek) az azonosító(k) ?
- először csak logikai törlés (pl. státusz mező alkalmazása), majd egyszerre fizikai törlés.

6.14.4 Az egyes funkciók áttekintése

Felvitel

- azonosító bekérése
- azonosító keresése a file-ban Van ilyen azonosító: hiba, nem vihető fel! (Duplikáció tiltott!)
- a rekord többi mezőinek felvitele
- státusz mező igazra állítása
- törölt rekord keresése: -- van: beírni a helyére az újat -- nincs: file végére felírni az új rekordot -- nem sikerült: hiba, betelt a lemez!

Módosítás

- módosítandó adat azonosítójának bekérése
- azonosító keresése a file-ban Nincs: hiba, nincs módosítandó adat
- módosítások elvégzése
- adat visszaírása

Törlés

- logikai
- törlendő adat azonosítójának bekérése
- azonosító keresése a file-ban Nincs: hiba, nincs törlendő adat
- rekord státusz mezőjének átállítása
- rekord visszaírása
- fizikai
- átmeneti állomány létrehozása
- amíg nincs vége az adat állománynak, a nem törölt rekordok átírása az átmenetibe
- adat állomány lezárása, átmeneti átnevezése az adat állomány nevére

Keresés

- azonosító bekérése
- azonosító keresése a file-ban Nincs: hiba, nincs meg a keresett adat
- adat kiírása

Teljes lista

- amig nincs vége az adat állománynak
- adat beolvasás
- adat kiírása

6.14.5 Mintafeladat állományok kezelésére

A végén nézzünk egy komplett feladat megoldását a típusos állományok kezelésére:

```
PROGRAM rekfile; { Adatkezelés file-okban rekordtípus segítségével}
USES CRT;
CONST KeyF1 = #59; KeyF2 = #60; KeyF3 = #61; KeyF4 = #62; KeyF5 = #63;
    KeyF6 = #64; KeyEscape = #27; NullChar = #0;
    Kilephet = [KeyF1, KeyF2, KeyF3, KeyF4, KeyF5, KeyF6, KeyEscape];
    FileName = 'adat.dat';
TYPE SzSzTipus = STRING[11];
    SzemelyTipus = RECORD
        Status:BOOLEAN; { True: aktiv, False: törölt }
        SzSz:SzSzTipus;
        Nev:STRING[30];
        Cim:STRING[40];
        Fiz:LONGINT;
    END;
VAR SzFile:file OF SzemelyTipus;
    Szemely:SzemelyTipus;
    SzemSzam:SzSzTipus;
    i,Hiba:INTEGER;
    valaszt:CHAR;
    FizStr:STRING;
    Ciklus:BOOLEAN;
PROCEDURE Varakozo;
    VAR varakozik:CHAR;
    BEGIN GOTOXY(20,25); WRITE('Folytatás billentyű lenyomására !');
        varakozik:=READkey; IF varakozik=#0 then varakozik:=READkey;
    END;
FUNCTION Space(mennyi:BYTE):STRING;
    VAR i:BYTE; Atm:STRING;
    BEGIN
        Atm[0]:=chr(Mennyi);
        FOR i:=1 TO Mennyi DO
        Atm[i]:=' '; Space:=Atm;
    END;
PROCEDURE Uzenet(Mit:STRING);
    BEGIN GOTOXY(20,24);WRITE(Mit); Varakozo;
        GOTOXY(20,24);WRITE( Space(LENGTH(Mit)) );
    END;
PROCEDURE RekIni;
    BEGIN
        WITH Szemely DO BEGIN Status:=true; SzSz:='';
            Nev:=''; Cim:=''; Fiz:=0;
        END;
    END;
FUNCTION IgenNem:BOOLEAN;
    VAR valaszt2:CHAR;
    BEGIN
        REPEAT
            valaszt2:=upCASE(READkey);
        UNTIL valaszt2 in ['I','N'];
        IgenNem:= Valaszt2='I';
    END;
PROCEDURE Felvitel;
    VAR AtmSzem:SzemelyTipus;
    BEGIN
        CLRSCR;
        GOTOXY(5,5);WRITE('Felvitel');
        GOTOXY(5,8);WRITE('Kérem a személyi számot: ');READLN(SzemSzam);
        {$I-} reset(SzFile); {$I+}
        IF IOResult0{ Nincs még ilyen file!} THEN REWRITE(SzFile);
        IF IOResult0{ Tele a lemez! } THEN Uzenet('Tele van a ...!')
        ELSE BEGIN
            IF FileSize(SzFile)0 THEN BEGIN
                Ciklus:=true;
                WHILE (NOT(EOF(SzFile)) AND Ciklus) DO BEGIN
                    READ(SzFile,Szemely);
                    IF ((SzemSzam=Szemely.SzSz) AND Szemely.Status) THEN Ciklus:=false;
                END;
            END;
            IF ((SzemSzam=Szemely.SzSz) AND Szemely.Status) THEN  Uzenet('Van már ilyen azonosító !')
```

```
            ELSE BEGIN
                WITH Szemely DO BEGIN
                    SzSz:=SzemSzam; GOTOXY(5,10); WRITE('Kérem a nevet:');READLN(Nev);
                    GOTOXY(5,12); WRITE('Kérem a címet:');READLN(Cim);
                    GOTOXY(5,14); WRITE('Kérem a fizetést: ');
                    REPEAT
                        GOTOXY(23,14);clreol; READLN(FizStr);
                        VAL(FizStr,Fiz,Hiba);
                    UNTIL Hiba=0;
                END;
                SEEK(SzFile,0);
                AtmSzem.Status:=true;
                WHILE (NOT(EOF(SzFile)) AND AtmSzem.Status) DO READ(SzFile,AtmSzem);
                IF NOT(EOF(SzFile)) THEN SEEK(SzFile,FilePos(SzFile)-1);
                {$I-} WRITE(SzFile,Szemely); {$I+}
                IF IOResult0 THEN Uzenet('Tele van a lemez!');
            END;
            CLOSE(SzFile);
        END;
    END;
PROCEDURE Modositas;
    BEGIN
        {$I-} RESET; (SzFile); {$I+}
        IF IOResult=0 THEN BEGIN
            CLRSCR;
            IF FileSize(SzFile)0 THEN BEGIN
                GOTOXY(5,5);WRITE('Módosítás');
                GOTOXY(5,8); WRITE('Kérem a személyi számot:');READLN(SzemSzam);
                Ciklus:=true;
                WHILE (NOT(EOF(SzFile)) AND Ciklus) DO BEGIN
                    READ(SzFile,Szemely);
                    IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN Ciklus:=false;
                END;
                IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN BEGIN
                    WITH Szemely DO BEGIN
                    GOTOXY(5,10); WRITELN('A név: ',Nev, 'Módosítja ? [I/N]');
                    IF IgenNem THEN BEGIN
                        GOTOXY(5,11);WRITE('Az új név: '); READLN(Nev);
                    END;
                    GOTOXY(5,12); WRITELN('A cím: ',Cim, ' Módosítja ? [I/N]');
                    IF IgenNem THEN BEGIN
                        GOTOXY(5,13); WRITE('Az új cím: '); READLN(Cim);
                    END;
                    GOTOXY(5,14); WRITELN('A fizetés: ',Fiz:10, ' Módosítja ?[I/N]');
                    IF IgenNem THEN BEGIN
                        GOTOXY(5,15); WRITE('Az újfizetés: ');
                        REPEAT
                            GOTOXY(20,15);clreol; READLN(FizStr);VAL(FizStr,Fiz,Hiba);
                        UNTIL Hiba=0;
                    END;
                END;
                SEEK(SzFile,FilePos(SzFile)-1); WRITE(SzFile,Szemely);
            ELSE Uzenet('Nincs ilyen azonosító !');
            END
            ELSE Uzenet('Üres azállomány!');
        END ELSE Uzenet('Nincs még ilyen állomány!');
    END;
PROCEDURE LogTorles;
    BEGIN
        {$I-} reset(SzFile); {$I+}
        IF IOResult=0 THEN BEGIN
            CLRSCR;
            IF FileSize(SzFile)0 THEN BEGIN
                GOTOXY(5,5);WRITE('Logikai törlés');
                GOTOXY(5,8); WRITE('Kérem a személyiszámot: '); READLN(SzemSzam);
                Ciklus:=true;
                WHILE (NOT(EOF(SzFile)) AND Ciklus DO BEGIN
                    READ(SzFile,Szemely);
                    IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN Ciklus:=false;
                END;
                IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN BEGIN
                    WITH Szemely DO BEGIN
                        GOTOXY(5,10); WRITE('A név: ',Nev);
                        GOTOXY(5,12); WRITE('A cím: ',Cim);
                        GOTOXY(5,14); WRITE('A fizetés: ',Fiz:10);
```

```
                END;
            GOTOXY(5,16);WRITE('Ezt akarja törölni? [I/N]');
            IF IgenNem THEN BEGIN
                Szemely.Status:=false;
                SEEK(SzFile,FilePos(SzFile)-1);
                WRITE(SzFile,Szemely);
            END
            END ELSE Uzenet('Nincs ilyen azonosító !');
        END ELSE Uzenet('Üres az állomány!');
    END ELSE Uzenet('Nincs még ilyen állomány!');
END;
PROCEDURE FizTorles;
    CONST AtmFNev = 'adat.ppp';
    VAR OK:BOOLEAN; AtmFile:file of
        SzemelyTipus;
    BEGIN
        {$I-} RESET(SzFile); {$I+}
        IF IOResult=0 THEN BEGIN
            IF FileSize(SzFile)0 THEN BEGIN
                ASSIGN(AtmFile,AtmFNev);
                {$I-} REWRITE(AtmFile); {$I+}
                IF IOResult0 THEN Uzenet('Tele van a lemez!')
            ELSE BEGIN
                CLRSCR; GOTOXY(5,5); WRITE('Fizikaitörlés');
                GOTOXY(5,8); WRITE('Kis türelmet kérek, másolom azállományt! ');OK:=true;
                WHILE (NOT(EOF(SzFile))) and OK DO BEGIN
                    READ(SzFile,Szemely);
                    IF Szemely.Status THEN BEGIN
                        {$I-} WRITE(AtmFile,Szemely); {$I+}
                        IF IOResult0 THEN BEGIN
                            OK:=false; Uzenet('Nincs elég hely a másoláshoz!');
                            CLOSE(AtmFile);
                            Erase(AtmFile);
                        END;
                    END;
                END;
                CLOSE(SzFile);
                IF OK THEN BEGIN
                    CLOSE(AtmFile);
                    ERASE(SzFile);RENAME(AtmFile,FileName);
                END;
            END
            END ELSE Uzenet('Üres az állomány!');
        END ELSE Uzenet('Nincs még ilyen állomány!');
    END;
PROCEDURE Kereses;
    BEGIN
        {$I-} reset(SzFile); {$I+}
        IF IOResult=0 THEN BEGIN
            CLRSCR;
            IF FileSize(SzFile)0 THEN BEGIN
                GOTOXY(5,5);WRITE('Keresés');
                GOTOXY(5,8); WRITE('Kérem a személyi számot:'); READLN(SzemSzam);
                Ciklus:=true;
                WHILE (NOT(EOF(SzFile)))) AND Ciklus DO BEGIN
                    READ(SzFile,Szemely);
                    IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN Ciklus:=false;
                END;
                IF (SzemSzam=Szemely.SzSz) AND Szemely.Status THEN BEGIN
                    WITH Szemely DO BEGIN
                        GOTOXY(5,10); WRITE('A név: ',Nev);
                        GOTOXY(5,12); WRITE('A cím: ',Cim);
                        GOTOXY(5,14); WRITE('A fizetés: ',Fiz:10);
                    END;
                    Varakozo;
                END
                ELSE Uzenet('Nincs ilyen azonosító !');
            END
            ELSE Uzenet('Üres az állomány!');
        END ELSE Uzenet('Nincs még ilyen állomány!');
    END;
PROCEDURE Listazas;
    VAR Sor:BYTE;
    PROCEDURE Fejlec;
        BEGIN
```

```
            WRITELN(' Személyi szám Név ', 'Cím Fizetés');
            WRITELN('-----------------------------------','----------------------------------------
            WRITELN; Sor:=4;
        END;
    BEGIN
        {$I-} reset(SzFile); {$I+}
        IF IOResult=0 THEN BEGIN
            CLRSCR;
            IF FileSize(SzFile)0 THEN BEGIN
                Fejlec;
                WHILE (NOT(EOF(SzFile))) DO BEGIN
                    READ(SzFile,Szemely);
                        IF Szemely.Status THEN BEGIN
                            WITH Szemely DO BEGIN
                                GOTOXY( 3,Sor);WRITE(SzSz);
                                GOTOXY(18,Sor);WRITE(Nev);
                                GOTOXY(40,Sor);WRITE(Cim);
                                GOTOXY(62,Sor);WRITE(Fiz:7);
                            END;
                            INC(Sor);
                            IF Sor=24 THEN BEGIN
                            Varakozo; CLRSCR; Fejlec;
                        END;
                    END;
                END;
                IF Sor4 THEN Varakozo;
            END
            ELSE Uzenet('Üres az állomány!');
        END
        ELSE Uzenet('Nincs még ilyen állomány!');
    END;

BEGIN
    ASSIGN(SzFile,FileName);
    REPEAT RekIni;
        CLRSCR;
        GOTOXY(25, 6); WRITE('Felvitel == F1');
        GOTOXY(25, 8); WRITE('Módosítás == F2');
        GOTOXY(25,10); WRITE('Fizikai törlés == F3');
        GOTOXY(25,12); WRITE('Logikai törlés == F4');
        GOTOXY(25,14); WRITE('Keresés == F5');
        GOTOXY(25,16); WRITE('Listázás == F6');
        GOTOXY(25,18); WRITE('Vége == Esc');
        GOTOXY(22,20); WRITE('Írja be a választását: ');
        REPEAT
            Valaszt:=READkey;
            IF Valaszt=NullChar THEN Valaszt:=READkey;
        UNTIL Valaszt in Kilephet;
        CASE Valaszt OF KeyF1;
            Felvitel;KeyF2:Modositas;
            KeyF3:FizTorles;
            KeyF4:LogTorles;
            KeyF5:Kereses;
            KeyF6:Listazas;
        END;
    UNTIL valaszt=KeyEscape;
END.
```

6.15 Ajánlott irodalom

Angszter - Kertész:
Turbo Pascal 5.0 - 5.5 'A'..'Z'

Angszter - Kertész:
Turbo Pascal 5.5 feladatgyjűtemény

Áts László :
turbo PASCAL kezdőknek

Benkő Tiborné - Hegedűs András :
IBM PC PROGRAMOZÁSA TURBO PASCAL NYELVEN

Benkő Tiborné - Hegedűs András - Benkő László - Kiss Zoltán - Tóth Bertalan :
IBM PC PROGRAMOZÁSA TURBO PASCAL 6.0-BAN PÉLDATÁR I. ÉS II.

Jensen – Wirth:
A PASCAL programozási nyelv

OBJEKTUM-ORIENTÁLT PROGRAMOZÁS TURBO PASCAL 6.0-BAN

Pirkó József:
Turbo Pascal 5.5

Reméljük,

IBM PC XT/AT

kezelése

ALAPISMERETEK ÉS ALKALMAZÓI PROGRAMOK

hogy ez
a könyv
segítette Önt a
tanulásban,
megszerettette a
számítógépet, amely

leendő munkáit könnyíteni fogja.

BINOM Szervezői, Számítástecnikai és Kereskedelmi BT.

Ehhez további segítséget nyújthatunk:

- az Ön igényeinek és feladatainak legjobban megfelelő számítástechnikai eszközök szállítása és szervizellátása
- az Ön igényeinek és feladatainak legjobban megfelelő program elkészítése, kiválasztása, alkalmazásának betanítása
- meglévő programjaink közül szíves figyelmébe ajánljuk az

ÜGYVITEL 2001,
számos referenciahellyel rendelkező
programcsomagunkat, melynek moduljai:
Raktárnyilvántartás Bizományosi eladás /ügynökök/
Konszignációs raktár Saját bolt elszámoltatás
Rendelés - Import - Kalkuláció - Bérszámfejtés
Deviza - Piaci árak - Partner- és személyzeti nyilvántartás
Számlázás Folyószámla vezetése Főkönyv

Jó munkát kíván a

BINOM Szervezői, Számítástechnikai és Kereskedelmi Betéti Társaság
2600 Vác, Cserje utca 30.