FELFÖLDI JÓZSEF

SZ. LUKÁCS JÁNOS

TIPPEK ÉS TRÜKKÖK

ENTERPRISE

FELFÖLDI JÓZSEF – SZ. LUKÁCS JÁNOS

Tippek és trükkök ENTERPRISE

Novotrade Rt.

Lektorálta: Lipi Gábor
Szerkesztette: Stankovics János
SCAN: GAFZ 2004

A kiadásért felel RÉNYI GÁBOR, a Novotrade Rt. vezérigazgatója
Budapest, 1989

Műszaki szerkesztő: Erdősi Zoltán

Szedte: NOVOTRADE Rt. – TYPOART Kiadványszerkesztő Iroda

Készült a Somogy megyei Nyomdaipari Vállalat kaposvári üzemében
Felelős vezető: Mike Ferenc igazgató

ISBN 963 585 046 8

Copyright © Felföldi József, Sz. Lukács János, 1989

TARTALOMJEGYZÉK

Bevezetés 5

1. Memóriatérkép 7
RST rutinok 39

2. Parancsok és utasítások 63
Sorszám 63
ALLOCATE 64
BIN 65
CALL (rekurzió) 68
COLOR 76
COPY 81
CHR$ 87
DATE$ 89
DIM 89
DISPLAY 97
DO/LOOP 102
FOR-NEXT 104
GET 105
GOSUB 107
GOTO 109
GRAPHICS 109
Hálózat 115
HEX$ 118
IN (n) 120
INPUT 121
JOY 126
LET 129
LINE MODE 131
LIST 132
Logaritmus (string matematika) 136
Logikai műveletek 142

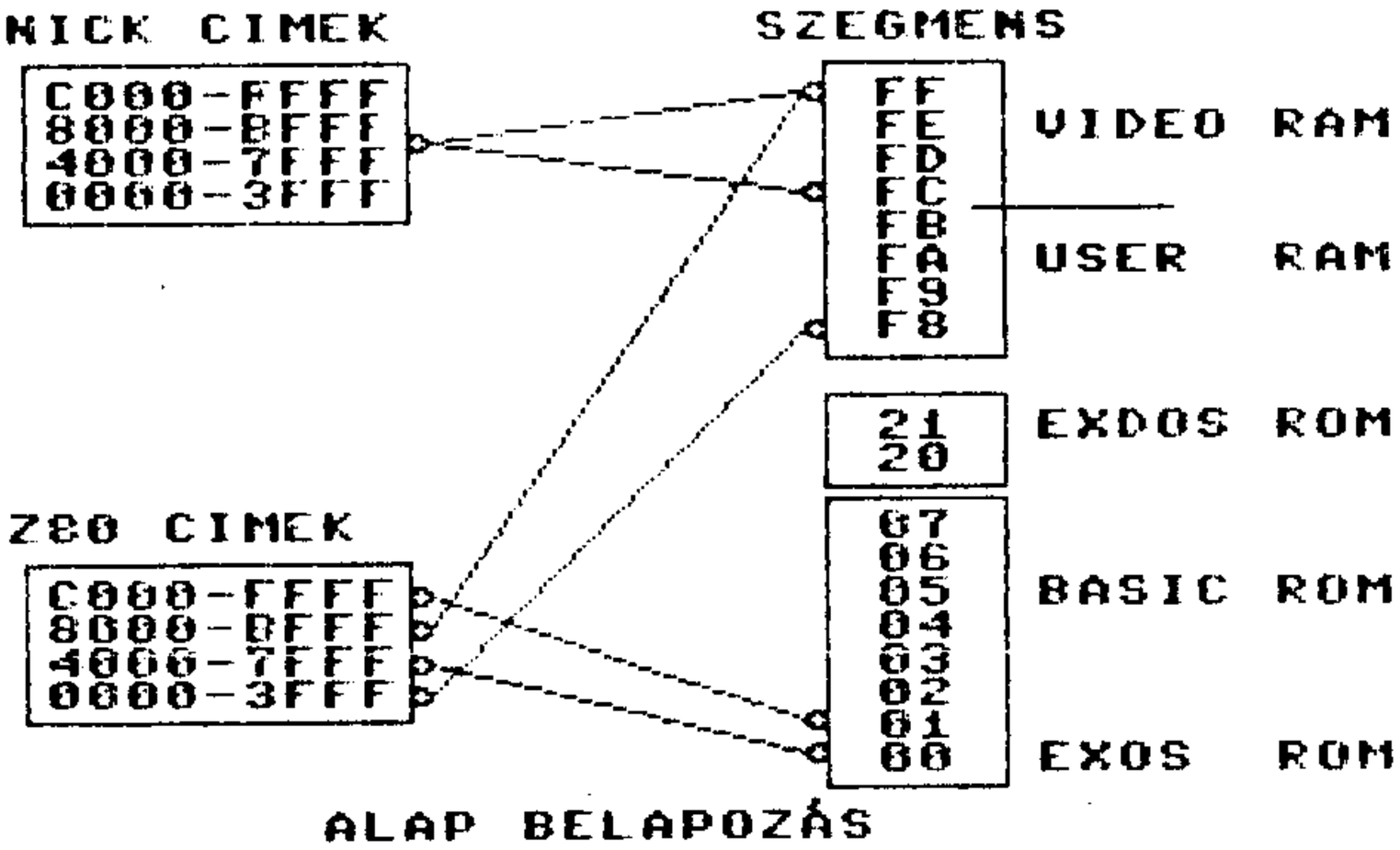
LOOK 145
MAX 148
MOD 149
NEW 153
OPEN 156
POKE 162
PRINT, LPRINT 165
SET CHARACTER 167
SET STATUS 178
Sorparaméter-tábla (LPT) 180
Szorzás, szorzótábla 182
Szögfüggvények 184
TANH 188
Teknős grafika 189
TIME$ 194
VAL 197
WHEN 202

BEVEZETÉS

Minden számítógéptulajdonos igyekszik megismerni gépe tulajdonságait, kezelését, előnyeit, esetleges fogyatékosságait. Egy ilyen sokoldalú – elfogultság nélkül mondhatjuk –, nagytudású gép kiismerése viszont szinte lehetetlennek tűnik. Minden rejtett tulajdonság már csak azért is nehezen általánosítható, mert különböző kialakítású gépek vannak forgalomban. Van amelyik csak angolul ért, van amelyik angol–német billentyűzet, ill. üzemmód között átváltható.

Példáink, megállapításaink bekapcsolás utáni alapállapotú, 128 K kiépítésű, angol–német átkapcsolási lehetőséggel rendelkező gépekre vonatkoznak.

A kezelési utasításból valami halvány képzetünk lehet a memóriakiosztás és memóriaszervezés bonyolultságáról. Nem elrettentésként, de az Enterprise memóriával való gazdálkodás első látásra követhetetlenül bonyolultnak tűnik. Egy átlag gépfelhasználónak természetesen alig kell törődnie a szegmensek különböző felhasználásával. A szegmensek belapozásával, megosztásával és a konkrét állapot számontartásával profi módon foglalkozik az operációs rendszer, az EXOS. Alapértelmezésben, bekapcsolás utáni állapotban a Z8Ø-as és a NICK által kiszolgálható szegmensek „helyzetét" az 1. ábrán követhetjük.



1. ábra Alapbelapozás

A Z8Ø-as processzor által kiszolgált, egyidejűleg „látott" szegmenseket a gép a Ø–3 lapra lapozza be, ezeket a BØH–B3H pontokon keresztül vizsgálhatjuk:

LAP	PORT		CÍM	SZEGMENST LÁT
PAGE Ø	BØH	176dec	Ø–3FFFH	F8H 248dec
PAGE 1	B1H	177dec	–7FFFH	ØØH Ødec
PAGE 2	B2H	178dec	–BFFFH	FFH 255dec
PAGE 3	B3H	179dec	–FFFFH	Ø1H 1dec

Az egyes lapokra éppen belapozott szegmensek lekérdezésére az 1. program szolgál.

```
100 ! a belapozott szegmensek
110 FOR I=176 TO 179
120   PRINT USING "#.lapra ###
      .szegmenst lapozta":I-176,
      IN(I)
130 NEXT
140 ! IN(n) beolvas a portrol
150 ! 176-179 portokon keresztul
    cimez a Z8Ø processzor
```

1. program

A programok futása közben ez az állapot változik, de mire a kurzor és az ok felirat megjelenik a képernyőn, általában ismét ez a helyzet tapasztalható, hacsak nem változtatunk rajta. Egyelőre ne tegyük, hacsak nem akarjuk, hogy a gép „elszálljon". Basic utasítással csak az egyes lapra lapozzunk be, a többi lap által látott szegmensszám csak gépi kódú programból és csak nagy körültekintéssel módosítható.

Böngésszünk tovább a memóriában! A memóriacímek és azok tartalma az alapállapot esetére vonatkoznak, de ha ezen keresztül bizonyos dolgokat sikerül meg- és felismerni, akkor talán sikerül boldogulni később bonyolultabb helyzetek esetében is.

1. MEMÓRIATÉRKÉP

Ø5– Ø7H CP/M részére foglalt
Ø8– 37 H RST rutinok ugrási címei és átmeneti tárak
38– 3E H IT (megszakítási) rutin
3F–118 H EXOS és CP/M által használt terület

A továbbiakban a RST 8 lapozó
RST 1Ø makro hívó
RST 18 EXOS hibakezelő
RST 2Ø Basic hibakezelő
RST 3Ø EXOS rendszerhívó
RST 38 megszakítási rutinokra,
azok felhasználására még visszatérünk.

2ØØ–247H Basic rendszerváltozók

Ezek tartalma a programbeírás, parancsok és utasítások végrehajtása közben változik. Vannak olyan címek, melyeket az új programsor beírásakor a tokenizálás közben más célra is felhasznál a rendszer, mint amit itt felsorolunk.

A Basic-ben a rendszerváltozó terület elejét a Z8Ø-as processzor IX regisztere tartalmazza. Ennek értéke 2ØØ H. A programok futása és szerkesztése közben e címhez viszonyítva relatív címzéssel vagy közvetlen kijelöléssel módosítja a tárolt értéket a rendszerprogram. Minthogy az egyes címek tartalmát PEEK-kel mi is ki akarjuk olvasni, vagy POKE-kal módosítani kívánjuk, célszerű ezeket a címeket decimális értékükkel is ismerni.

1FB–1FFH	Hálózat használja CHAN NET, ill. a forrásgép számának bejegyzésére
2ØØH 512dec	Az utasítások és parancsok végrehajtása során szükséges teendők bejegyzésére szolgáló byte. Az egyes bitek jelentése: Ø. ok kiírható, ha értéke 1 1. program futás közben értéke 1

2. STOP billentyű lenyomásának ellenőrzése 1, ha STOP érkezett
3. CONTINUE végrehajtható, ha értéke 1
4. nyomkövetés TRACER ON ilyenkor értéke 1
5. a beolvasási főág szubrutinként hívható, ha éréke Ø
6. csatorna nyitva jelzés, ha OPEN 1
7. program CHAIN-nal indult értéke Ø
   RUN-nal START-tal indult értéke 1

Parancsmódban értékének megváltoztatása a rendszer összeomlásához vezethet, de programfutás közben is óvatosan bánjunk a direkt beavatkozás lehetőségével.

2Ø1H 513dec
Ø. bitje a szögek fokokban (értéke 1), ill. radiánban való (értéke Ø) figyelembevételét jelöli ki
7. bitjét a billentyűzet kezelő rutin használja jelző bitként

2Ø2H 514dec
Az aktuális Basic programsor éppen feldolgozott elemének típusjelzője
Tartalma: ØH ha írásjel
2ØH változó név (numerikus)
4ØH változó név (string)
6ØH kulcsszó
8ØH string
AØH programsorszám
CØH számérték

2Ø3H 515dec
Az előbb jelzett feldolgozás alatt lévő elem hossza.
Ha 2Ø2H tartalma 2Ø, vagy 4ØH, akkor numerikus, vagy string változó. Ebből következik, hogy a változó nevének hossza 1 és 31 (dec) között lehet.
Ha 2Ø2H tartalma 5ØH, akkor 2Ø3H tartalma a kulcsszó token sorszáma, amennyiben 2Ø2H tartalma 8ØH, úgy egy macskakörmök között lévő karakter sor következik és ennek a hossza.
A 2Ø2H tartalmaként AØH sorszámot jelez, ennek tárolása mindig két byte hosszon történik. A CØ típusjelző numerikus értékre utal. Ha ez egész típusú, akkor ezen a memóriacímen 2, ha lebegőpontos a szám, akkor 6 a hosszjelző értéke (Tokenizálás közben relációjelek tárolása is ezen a címen történik.)

2Ø4H 516dec
Jelzi, hogy a feldolgozás alatt lévő elem írásjel, vagy tokenizált (kódolt)-e?
Ha írásjel, akkor annak kódját tartalmazza

2Ø5H 517dec
A billentyűzetről olvasott karakter kódját tartalmazza, pl. INKEY$ esetén

2Ø6H 518dec	Egy utasítás végrehajtása utáni teendők jelzése. Az utasítás végrehajtásának megkezdésekor értéke Ø A különböző utasítások végrehajtása közben (a Basic kulcsszó jellegével többé-kevésbé összhangban) ezt a rendszerváltozó értéket átírják. Ø ha több utasítás lehet a sorban, következik-e 1 ha csak sorvég következhet (pl. RETURN) 2 a sor hátralévő részét figyelmen kívül hagyja és a következő sor elejére ugrik (pl. NEXT, REM) 3 a 216H (534dec) két byte hosszon tárolt címre ugrik, ahol az aktuálisnak tekintendő sor eleje található (pl. GOTO) 4 blokkvég keresés esetén ideiglenes bejegyzés
2Ø7H 519dec	Hibakezelő Handler bejegyzések tartalma Ø RETRY 1 (5.F3CØ) 2 EXIT 3 END HANDLER esetén
2Ø9H 521dec	Az aktuális csatorna száma. Ø a rendszer csatorna
2ØAH 522dec	Az aktuális program sorszáma, ennek értéke a státuszsoron leolvasható
2ØBH 523dec	Megosztott szegmens sorszáma
2ØCH 524dec	A RAM szegmensek száma
2ØDH 525dec	Annak a szegmensnek a száma, amin fut az aktuális program
2ØEH 526dec	TEXT üzemmód jelző byte-ja Ø 4Ø karakteres üzem 2 8Ø karakteres üzem
2ØFH 527dec	Grafikus üzemmód jelző byte-ja. A MODE VID EXOS változó értéke 1 HIRES 5 LORES 15 ATTRIBUTE
21ØH 528dec	Nagyfelbontású üzemód színállapot számjelzés A COLOR VID EXOS változó értéke Ø ha ATTRIBUTE, vagy GRAPHICS 2 1 ha GRAPHICS 4 2 ha GRAPHICS 16 3 ha GRAPHICS 256
211H 529dec	Egy byte-os elő véletlenszám (az előző 246H tartalom)
212H 53Ødec	CHAIN paraméter az indított program száma
213H 531dec	EXOS változók írásánál átmeneti tár
214–215H 532–533dec	A programsorban legközelebb végrehajtandó elem címe a memóriában

216–217H 534–535dec	A végrehajtás alatt lévő sor elejének címe a memóriában, de INPUT esetén az input puffer elejére mutat
218–219H 536–537dec	Sorszámjelző AØH esetében a sorszámot (egész szám) ide tölti, tokenizálás közben a sorfeldolgozás jelzésére használt tár
21A–21BH 538–539dec	Az aktuális program elejére mutat. Alapértelmezésben Ø. lapon 4827dec az itt tárolt érték
21C–21DH 54Ø–541dec	CODE által módosított Basic kezdett címe
21E–21FH 542–543dec	Szerkesztőmódban 21A–21BH-val megegyező ALLOCATE és NEW módosítja, ill. állítja be
22Ø–221H 544–545dec	Bekapcsolás utáni állapotban ez is 21A–21BH-val megegyező. Programfutás közben, ha ALLOCATE-tel gépi kódú programnak helyet foglaltunk, akkor a Basic kezdetének címe
222–223H	Nem használt (legalábbis nem találtuk)
224–225H	Nem használt
226–227H 55Ø–551dec	Az aktuális szegmens utolsó szabad helye. Programfutás közben a felhasználható verem kezdetének címe

A verem kifejezés a szövegünkben két értelmezésben is előfordul. Ez az átmeneti tár mindig felülről lefelé, azaz a magasabb memóriacímtől az alacsonyabb felé építkezik. Az egyik verem a Z8Ø-as processzor-regiszterek tartalmának tárolására és a gépi kódú szubrutinok visszatérési címének tárolására szolgál (l. DBBH címen).

A másik, amelyre ebben a könyvben többször hivatkozunk, a változók, Basic utasítások végrehajtásához szükséges bejegyzések tárolására szolgál. Azt, hogy ebben a veremben hol tartunk, azt egyrészt a 228H rendszerváltozó címről tudhatjuk meg, másrészt, minthogy a processzor is használja ezeket az adatokat, ill. címeket az IY regiszter is felveszi ezt az értéket.

228–229H 552–553dec A verem tetejének címe

Az előző memóriatartalmak vizsgálatára alkalmas a 2. program.

```
1ØØ ! 538-554 dec. memoriacimek tartalma
11Ø ALLOCATE 1Ø ! 3.-4. futasnal elhagyni
12Ø CODE X=HEX$("2Ø1") !2.3.4.futasnal elhagyni
13Ø ! ha ALLOCATE nincs CODE agyonuti a programot!
14Ø FOR I=538 TO 553 STEP 2
15Ø   READ A$
16Ø   PRINT I;PEEK(I)+256*PEEK(I+1);":";A$
17Ø NEXT
18Ø DATA Basic program eleje
19Ø DATA CODE altal meg nem hasznalt
```

```
200 DATA ALLOCATE altal foglalt terulet eleje
210 DATA ALLOCATE utani elso szabad hely
220 DATA ,,verem alja,verem teteje
```

2. program

22A–22BH 22C–22DH	A következő adatot tartalmazó DATA sor elejére mutat. RESTORE-rel is módosítható
22E–22FH	Az aktuális DATA sorban a legközelebb kiolvasandó adatra mutat
230–231H 560–561dec	INPUT-tal beolvasható adat memóriabeli helyét jelzi
232–233H 562–563dec	Alapállapotban 3684dec a RAM-ban lévő kulcsszótábla kezdő címe
234–235H 564–565dec	A változó tábla következő szabad helyének címe
236–237H 566–567dec	A változó tábla elejének címét tárolja. Ez a cím a tárolt Basic program utáni első memória rekeszre mutat. Ennek ismeretében az általunk beírt Basic program hosszát mi magunk is meghatározhatjuk (3. program).

```
100 ! program hossza rendszervaltozo tartalmabol
110 LET ELEJE=PEEK(544)+256*PEEK(545)
 120 LET VEGE=PEEK(566)+256*PEEK(567)-1
 130 IF VEGE>16384 THEN LET ELEJE=16384
 140 !mert csak a 0.pr. elejet mutatja az 554
rendszervaltozo
 150 PRINT "  pr. eleje    :";ELEJE
 160 LET VEGE=PEEK(566)+256*PEEK(567)-1
 170 PRINT "  pr. vege     :";VEGE
 180 PRINT "program hossza=";VEGE-ELEJE;"byte"
 190 !ellenorzes INFO -val
```

3. program

238–239H 568–569dec	Az EXOS számára legtöbb hely biztosítása (változó tábla vége után 64dec szabad hely fenntartása)
23A–23BH 570–571dec	A Z80-as veremmutatója (Stack pointer) kezdőérték DBBH (3511dec) és az alacsonyabb memóriacímek felé terjeszkedik
23C–23DH 572–573dec	Utoljára fellépett hiba (megszakítás) kódszáma EXTYPE
23E–23FH 574–575dec	A hibát okozó programsor száma EXLINE

24Ø–241H 576–577dec	Hibakezelés során átmeneti tár
242–243H 578–579dec	Listázás előkészítésnél tokenizálásnál az áttöltött vagy beírt karakter helye utáni memóriacímre mutat
244–245H 58Ø–581dec	
246–247H 582–583dec	Elővéletlen szám

Az egy és két byte hosszon tárolt két elővéletlen számból állítja elő az RND értékét. Az RND 65 535 különféle értéket vehet fel, de periodikus ismétlődés csak 524 287 érték után következik be. Az RND kiszámítása (X+32 768)/65 536 összefüggéssel történik. A képletbe *X* értékét a 246–247H memóriarekesz tartalma szolgáltatja. A memóriarekesz tartalma az előző tartalomtól és a 211H címen tárolt értéktől függ. A fenti állítások ellenőrzésére a 4. program szolgál.

```
100 ! RND kiismerese
110 LET A=RND:LET E=1
120 PRINT "most          (211H)=";PEEK(529)
130 PRINT :PRINT
140 LET B=PEEK(582):LET C=PEEK(583)
150 IF C>128 THEN
160   LET C=256-C:LET E=-1
170 END IF
180 PRINT "RND (ahogy a gep adja):";A
190 PRINT "es ahogy kifigyelheto :";(32768+E*256*C+B)
/65536
200 ! ez az 1.szegmens CAA9H rutinjanak Basic szintu
modellje
210 PRINT :PRINT
220 PRINT "legkozelebb (211H)=";B
230 GOTO 110
```

4. program

248H 584dec	255 byte hosszú INPUT puffer kezdete a puffer első helyén a beolvasott karaktersorozat hosszát tárolja. Az 5. program az INTER PUFFER tartalmának vizsgálatával mutatja.

```
100 ! INPUT PUFFER tartalmanak vizsgalata
110 INPUT V$
120 LET K=PEEK(584)
130 FOR C=584 TO 583+K
140   LET I=PEEK(C):LET A$="."
```

```
150    IF I>32 AND I<160 THEN LET A$=CHR$(I)
160    PRINT C;I;A$
170 NEXT
180 ! puffer elso eleme a hossz
190 ! INPUT adatsor veget Ø jelzi
```

5. program

Ennek ismeretében közvetlenül az INPUT pufferből is kiolvashatjuk (adott esetben majd kezelhetjük) az ott lévő szöveget. De ez a terület általános pufferként is szolgál LIST, RENUMBER esetén is

347–348H 839–84Ødec	Az INPUT puffer adott helyére mutató pointer
349H 841dec	253 byte hosszú puffer
446H 1Ø94dec	A hibaüzenet az itt kezdődő 8Ø byte hosszú pufferből kerül a képernyőre. A hiba fellépése esetén a megfelelő RST 1Ø rutin ide mozgatja a ROM-ból a kiírandó szöveget. A hibaüzenet puffer tartalmát mi is kiolvashatjuk (természetesen csak akkor, ha már időközben valamilyen hibát sikerült elkövetni). (6. program)

```
100 ! utolso hibauzenet
110 LET I=PEEK(1094)
120 ! 1094dec a hibauzenet puffer eleje
130 FOR K=1094 TO 1094+I
140    LET A=PEEK(K):LET A$=CHR$(A)
150    IF K=1094 THEN LET A$="hossz"
160    PRINT USING "###  #####":A;A$
170 NEXT
```

6. program

A hibaüzenetek természetesen a ROM megfelelő helyein vannak. A német verziójú gépek esetében az üzenetek a 4. szegmens 26 659–3Ø 352dec címei között lévő területen találhatók meg a következő elrendezésben: sorszám, üzenethossz, és a hibaüzenet karakteres formában. Erről a 7. program segítségével könnyen meggyőződhetünk.

```
100 !nemet hibauzenetek tarolasa
110 LET X=26659
120 DO
130    LET V=SPEEK(4,X) !uzenet sorszama
```

```
140    IF V<>0 THEN
150      PRINT V;
160      LET H=SPEEK(4,X+1) !uzenet hossza
170      FOR Y=X+2 TO X+H+1
180        PRINT CHR$(SPEEK(4,Y));
190      NEXT
200      PRINT :LET X=X+H+2
210    END IF
220 LOOP UNTIL X>30000 OR V=0
230 ! rst 20 BASIC hibauzenetek
240 LET X=28381
250 DO
260    LET H=SPEEK(4,X) ! uzenet hossza
270    FOR Y=X+1 TO X+H
280      PRINT CHR$(SPEEK(4,Y));
290    NEXT
300    PRINT :LET X=X+H+1
310 LOOP UNTIL X>30352
```

7. program

Az angol hibaüzenetek ilyen könnyen nem találhatók meg, ravasz módon memóriakímélő konverzióval tárolja a ROM karaktereket az 1. szegmensen az EØ38H címtől kezdődően

496H 1174dec

Címen kezdődő, elvileg 255 byte hosszú pufferből írja ki az INPUT rutin a PROMPT szöveget, vagy csak a kérdőjelet. A kiírandó karakterek számát a 1174dec memóriacímen találhatjuk meg. A puffer ugyan megengedne ilyen hosszú szöveget, de a programsor így túl hosszú lenne, a PROMPT szöveg nem lehet 226 karakternél hosszabb (8. program).

```
100 ! INPUT PROMPT kiiras 496H pufferbol
110 INPUT A
120 ! vagy az egyik vagy a masik INPUT sor kell
130 INPUT PROMPT "Hogy is van ez ?":A
140 LET H=PEEK(1174) !kiirando uzenet hossza
150 FOR C=1174 TO 1174+H
160   LET T=PEEK(C):LET A$="."
170   IF T>31 AND T<160 THEN LET A$=CHR$(T)
180   ! A$ a puffercimen levo karakter
```

```
190    PRINT C;T,A$
200    ! vigyazz a USING is hasznalja ezt a puffert!
210 NEXT
```

8. program

A 11Ø–13Ø. programsorok közül a program kipróbálása esetén csak az egyikre van szükség. A program további része kiolvassa az INPUT üzenetként képernyőre küldendő szöveget PROMPT alkalmazásával, ill. anélkül

59EH 1438dec — Kiírásra karakteres formában előkészített számérték tárolására szolgáló puffer

DBBH — A verem kezdetének címe. A Z8Ø-as processzor az ettől a címtől kezdődő csökkenő memóriacím tartomány felé teszi meg a szükséges bejegyzéseket. Ehhez a címhez sohase nyúljunk, mert a rendszer összeomlik

DBBH–DCAH — Tartományt a LOAD, VERIFY és MERGE parancsok használják. Pl. (DBD) a program hossza, SAVE nullázza a tartalmat

DCBH — 64 byte hosszú 32 belépési címet tartalmazó táblázat eleje. A Basic-ben értelmezett függvények és a programba felvett változók értékének kiszámításához, ill. a változó táblában (szimbóliumtáblában) lévő adatok megkereséséhez

EØBH — Aritmetikai regiszter 1 a lebegőpontos számokkal való művelet végzéséhez

E12H — Aritmetikai regiszter 2.

E19H — A teknős grafika FORWARD irányának szögértéke radiánokban. $\pi/2$ ha függőlegesen felfelé indul. Inicializáláskor az alapértéket az 5. szegmensen C2CØ–C2C5 címen lévő rutinja írja be

E54H — RAM térkép. Az EXOS ide tárolja le, hogy melyik program részére melyik szegmenset utalta ki. A RAM bejegyzések sorrendjét és logikáját úgy ellenőrizhetjük, hogy a 9. program begépelése után rendre újabb programokat láncolunk be EDIT *n* segítségével, majd mindig térjünk vissza CHAIN Ø-val az itt közölt programra.

```
100 ! RAM bejegyzes  E54H cimtol
110 LET U$="  ####    ###     ###"
120 PRINT
130 PRINT "  cim  szegmens  pr."
140 FOR C=3666 TO 3680 STEP 2
```

```
  150   PRINT USING U$:C;PEEK(C);PEEK(C+1)
  160 NEXT
  170 ! ujabb inditasok elott EDIT n-nel ujabb
programokat irj be
```

9. program

Követhető, hogy mindig a legalacsonyabb sorszámú, még szabad szegmens bejegyzésekre kerül sor. A memóriatérkép méretéből következik, hogy egyidejűleg nyolc (most nincs is több) szegmens bejegyzése lehetséges

E64H

Címtáblázat a 93 db Basic utasítás végrehajtásához. Ennek a címtáblázatnak a felépítése: a kulcsszavak tokentáblában elfoglalt helyének megfelelő sorszám alapján egymásután 3 byte-on található megfelelő kulcsszó leírójának címe és szegmensszáma.
Alapkiépítésben a leíró és a Basic utasítások végrehajtó rutinjai az 5. szegmensen találhatók. A táblázat és értelmezése a 10. program alapján tanulmányozható.

```
100 ! kulcsszo leirora mutato  cimtabla
110 PRINT "token tablacim leiro cime kulcsszo"
  120 FOR T=0 TO 92
  130   LET C=3687+T*3 ! cimmutato
  140   LET A=PEEK(C):LET F=PEEK(C+1)
  150   LET K=A+256*F ! kulcsszo leiro cime az
5.szegmensen
  160   PRINT USING "   ##     ####     ### ###   ":T,C,A,
F;
  170   CALL KULCSSZO
  180 NEXT
  190 END
200 DEF KULCSSZO
210   LET H=SPEEK(5,K+5) !kulcsszo hossza
220   FOR I=K+5 TO K+5+H
230     PRINT CHR$(SPEEK(5,I)); !kulcsszo karakterei
240   NEXT
250   PRINT
260 END DEF
```

10. program

token	tablacim	leiro cime	kulcsszo
Ø	3687	167 245	ALLOCATE
1	369Ø	181 245	ASK
2	3693	19Ø 245	AUTO
3	3696	2ØØ 245	CALL
4	3699	21Ø 245	CAPTURE
5	37Ø2	223 245	CASE
6	37Ø5	233 245	CAUSE
7	37Ø8	244 245	CLEAR
8	3711	255 245	CLOSE
9	3714	1Ø 246	CODE
1Ø	3717	2Ø 246	CONTINUE
11	372Ø	34 246	COPY
12	3723	44 246	DATA
13	3726	54 246	DEF
14	3729	63 246	DEF
15	3732	72 246	DELETE
16	3735	84 246	DIM
17	3738	93 246	DISPLAY
18	3741	1Ø6 246	DO
19	3744	114 246	CHAIN
2Ø	3747	125 246	EDIT
21	375Ø	135 246	ELSE
22	3753	145 246	ELSE IF
23	3756	158 246	END
24	3759	167 246	END DEF
25	3762	18Ø 246	END HANDLER
26	3765	197 246	END IF
27	3768	2Ø9 246	END SELECT
28	3771	225 246	END WHEN
29	3774	239 246	ENVELOPE
3Ø	3777	253 246	EXIT
31	378Ø	7 247	FOR
32	3783	16 247	GOSUB

33	3786	27	247	GOTO
34	3789	37	247	GRAPHICS
35	3792	51	247	HANDLER
36	3795	64	247	IMAGE
37	3798	75	247	IF
38	38Ø1	83	247	IF
39	38Ø4	91	247	INPUT
4Ø	38Ø7	1Ø2	247	LET
41	381Ø	111	247	LINE
42	3813	121	247	LIST
43	3816	131	247	LOAD
44	3819	141	247	LOOP
45	3822	151	247	MERGE
46	3825	162	247	NEW
47	3828	171	247	NEXT
48	3831	181	247	NUMERIC
49	3834	194	247	OPEN
5Ø	3837	2Ø4	247	OPTION
51	384Ø	216	247	OK
52	3843	224	247	OUT
53	3846	233	247	PLOT
54	3849	243	247	POKE
55	3852	253	247	SPOKE
56	3855	8	248	PRINT
57	3858	19	248	PROGRAM
58	3861	32	248	RANDOMIZE
59	3864	47	248	READ
6Ø	3867	57	248	REDIRECT
61	387Ø	71	248	REM
62	3873	8Ø	248	RENUMBER
63	3876	94	248	RESTORE
64	3879	1Ø7	248	RETRY
65	3882	118	248	RETURN
66	3885	13Ø	248	RUN

```
67      3888      139 248    SAVE
68      3891      149 248    SELECT
69      3894      161 248    SET
7Ø      3897      17Ø 248    SOUND
71      39ØØ      181 248    START
72      39Ø3      192 248    STOP
73      39Ø6      2Ø2 248    INFO
74      39Ø9      212 248    STRING
75      3912      224 248    TEXT
76      3915      234 248    TOGGLE
77      3918      246 248    TRACE
78      3921        1 249    TYPE
79      3924       11 249    VERIFY
8Ø      3927       23 249    WHEN
81      393Ø       33 249    !
82      3933       4Ø 249    LLIST
83      3936       51 249    LPRINT
84      3939       63 249    EXT
85      3942       72 249    GET
86      3945       81 249    FLUSH
87      3948       92 249    LOOK
88      3951      1Ø2 249    PING
89      3954      112 249    DATE
9Ø      3957      122 249    TIME
91      396Ø      132 249    WAIT
92      3963      142 249    ON
```

1. lista A 10. program futásának eredménye

A futás eredményeként kapott lista első oszlopában a kulcsszó tokenje, a második oszlopban az adott kulcsszóra vonatkozó adatok memóriabeli helyét találjuk.
A harmadik-negyedik oszlop a leíróra mutató cím két byte-ja. Ezt a leírót még majd tüzetesebben megvizsgáljuk.
Az utolsó oszlopban az adott tokenhez tartozó kulcsszó található.

Ezek a leíróra mutató címek (hiszen RAM területen vagyunk) átírhatók és így megváltoztathatjuk az egyes kulcsszavak végrehajtásának menetét, de hogy ebből értelmes dolog adódjon, ahhoz igen alaposan végig kell gondolni a dolgokat. Erre példát a gépi kódú programozás kapcsán látunk majd.

A kulcsszavak címtáblája után a beépített függvények azonosító és ugrótáblája következik. Ez a 3968dec címen kezdődik, ha még nem „tunningoltuk" a gépet. Ennek a táblának a felépítését a 11. program futtatásával, ill. a mellékelt lista segítségével érthetjük meg.

```
 100 ! mast hajt vegre ha cimet atirom
 110 ! fuggvenyek vegrhajtas elso cimtabla
 120 FOR V=3968 TO 4800
 130   LET TIPUS=PEEK(V):LET HOSSZ=PEEK(V+1)
 140   LET CIM=V+2+HOSSZ ! a vegrehajtas atiranyitasa
itt lehet
 150   LET A=PEEK(CIM):LET F=PEEK(CIM+1) !vegrehajto
rutin cimenek also-felso byte-ja
 160   LET T$="n"
 170   IF TIPUS=13 THEN LET T$="$"
 180   LET N$=""
 190   FOR I=V+2 TO V+1+HOSSZ
 200     LET N$=N$&CHR$(PEEK(I))
 210   NEXT
 220   LET V=I+4
 230   PRINT USING "#  <########## ### ###     c:####":T
$,N$,A,F,CIM
 240 NEXT
```

11. program

```
tipus   fv.nev        rutincim    itt van
  n       ABS          33 195     c:3973
  n       ACOS         38 195     c:3984
  n       ANGLE        48 195     c:3996
  n       ASIN         98 195     c:4007
  n       ATN          38 196     c:4017
  n       BIN          44 196     c:4027
```

```
n       BLACK        250 196     c:4039
n       BLUE           0 197     c:4050
n       CEIL          93 196     c:4061
$       CHR$         162 196     c:4072
n       COS           20 197     c:4082
n       COSH          31 197     c:4093
n       COT           42 197     c:4103
n       CSC          139 202     c:4113
n       CYAN           8 197     c:4124
$       DATE$         67 197     c:4136
n       DEG          127 197     c:4146
n       EPS          145 197     c:4156
n       EXLINE       212 197     c:4169
n       EXP          237 197     c:4179
$       EXSTRING$    178 197     c:4195
n       EXTYPE       230 197     c:4208
n       FREE         141 198     c:4219
n       FP           123 198     c:4228
n       GREEN          3 197     c:4240
$       HEX$         239 198     c:4251
n       IN            70 199     c:4260
$       INKEY$        88 199     c:4273
n       INF          136 198     c:4283
n       INT          117 199     c:4293
n       IP            27 200     c:4302
n       JOY           49 200     c:4312
n       LBOUND        76 200     c:4325
$       LCASE$        40 205     c:4338
n       LEN          127 200     c:4348
n       LOG          155 200     c:4358
n       LOG2         146 200     c:4369
n       LOG10        164 200     c:4381
$       LTRIM$        18 201     c:4394
n       MAGENTA       12 197     c:4408
```

```
n     MAX          45 201     c:4418
n     MAXLEN       66 201     c:4431
n     MIN         111 201     c:4441
n     MOD         132 201     c:4451
n     ORD         181 201     c:4461
n     PEEK        206 201     c:4472
n     SPEEK       231 201     c:4484
n     PI           11 202     c:4493
n     POS          16 202     c:4503
n     RAD         121 202     c:4513
n     RED         253 196     c:4523
n     REM         150 202     c:4533
n     RGB         187 196     c:4543
n     RND         168 202     c:4553
n     ROUND       240 202     c:4565
$     RTRIM$       84 203     c:4578
n     SEC         124 203     c:4588
n     SIN         130 203     c:4598
n     SINH        235 203     c:4609
n     SIZE         10 204     c:4620
n     SGN         246 203     c:4630
$     STR$         48 204     c:4641
n     SQR          75 204     c:4651
n     TAN         155 204     c:4661
n     TANH        171 204     c:4672
$     TIME$        48 197     c:4684
n     TRUNCATE    187 204     c:4699
n     UBOUND      232 204     c:4712
$     UCASE$       37 205     c:4725
n     USR          63 205     c:4735
n     VAL          78 205     c:4745
$     VER$        104 205     c:4756
n     VERNUM      193 205     c:4769
```

```
 n        WHITE          15 197      c:4781
 n        YELLOW          6 197      c:4794
 $        WORD$         206 205      c:4806
```

2. lista A 11. programhoz tartozó lista

Egy byte jelzi, hogy a függvény numerikus, vagy string művelet végzésére alkalmas-e. Egy byte az azonosítandó függvénynév hosszát tartalmazza, ezután következik a függvénynév karakteres formában. A név után két byte a függvény végrehajtó rutinjának címét tartalmazza.

Ez a rutin (alapértelmezésű gépen) az 1. szegmensen van, de a 3. lapra belapozva kerül végrehajtásra. A futás eredményének listájában az utolsó oszlop azt a címet tartalmazza, ahova benyúlva adott esetben egy függvény végrehajtását más rutinra irányíthatjuk át. Való igaz, hogy az eddig leírt címek tartalmáról a számítógép rendszerprogramja gondoskodik. Azonban nem véletlenül helyezték el ezeket az itt tárolt adatokat egy olyan memóriaterületre, amelyhez egy átlagnál kissé jobban felkészült géptulajdonos – akár Basic utasítás – segítségével is hozzáférhet. A következőkben viszont olyan memóriaterület következik, melyet a legkezdőbb gépfelhasználó is – tudatosan, vagy sem – maga tölt fel saját programjával.

Az alapállapotú gépen a Basic program 12DBH 4827dec címtől kezdődő területen kezdődik. Arról, hogy ez a cím valóban igaz-e, a 22ØH–221H memóriarekeszek tartalmának kiolvasásával győződhetünk meg. A programsorok tárolásának mikéntjét a 12. program segítségével vizsgálhatjuk.

```
100 ! programtarolas vizsgalo
110 ! bekapcsolas utan Ø. program
120 ! megjegyzes sorok nelkul futtatni
130 LET VEGE=PEEK(566)+256*PEEK(567)-1
140 LET KEZDET=PEEK(538)+256*PEEK(539)
150 FOR I=KEZDET TO VEGE
160    LET T=PEEK(I)
170    PRINT I;T,
180    IF T>32 AND T<128 THEN PRINT CHR$(T);
190    PRINT
200 NEXT
```

12. program

```
          BASIC program tárolása


4827      42       Programsor hossza byte
4828     130       sorszám alsó byte
4829       0       sorszám felső byte
4830       0       skatulyázási mélység
```

```
4831     96      kulcsszó bevezető
4832     46      LET tokenje
4833     36      32+4 négy karakter hosszú
                 numerikus változó neve következik
4834     96  V   a változó nevének karakterei
4835     71  E
4836     89  E
4837     19      egyenlőségjel kódja
4838     36      négy karakter hosszú név következik
4839     80  P
4840     69  E
4841     79  K
4842     13      nyitó zárójel kódja
4843    194      192+2 két byte hosszon tárolt
                 egész típusú változó következik
4844     54      értékének alsó byte-ja
4845      6      értékének felső byte-ja
                 X= 6*256+54
4846      9      záró  zárójel kódja
4847     11      összeadás kódja
4848    194      egész típusú változó következik
4849      0      értékének alsó byte-ja
4850      1      értékének felső byte-ja
4851     10      szorzásjel kódja
4852     36      négy karakter hosszú név jön
4853     80  P   a név karakterei
4854     69  E
4855     77  M
4856     69  E
4857     80  P
4858     13      zárójel nyit
4859    194      egész típusú érték következik
4860     55      érték alsó byte
4861      2      érték felső byte X=2*256+55
4862     14      zárójel bezár kódja
4863     13      kivonásjel kódja
4864    194      egész típusú változó bevezetője
4865      1      érték alsó byte
4866      0      érték felső byte
4867      0      sorvég jelző

4868     21      sor hossza
4869    148      sorszám alsó byte
4870      0      sorszám felső byte
4871      0      skatulyázás mélysége
4872     95      kulcsszó bevezető
4873     81      FOR tokenje
4874     36      egybetűs változónév következik
4875     73  I   a változó neve
4876     19      egyenlőségjel kódja
4877    194      egész típusú változó jelzése
4878     21      érték alsó byte
4879     18      érték felső byte
                 X=18*256+219 ->4827 BASIC eleje
4880     34      kétbetűs név következik
4881     84  T
4882     79  O
4883     36      négybetűs változónév következik
```

```
4400      0      sorvég jelző
*
4400     19      sorhossz
4401    150      sorszám alsó byte
4402      0      sorszám felső byte
4403      1      aktulyázás mélysége
4404     96      Kulcsszó bevezető
4405     48      LET tokenje
4406     00      egybetűs változónév jön
4407     04  T   a változó betűjele
4409     19      egyenlőségjel kódja
440A      6      négy karakter hosszú név jön
440B     80  P
440C     77  E
440D     82  K
440E      0      záródjelet nyit
440F      6      egy karakter hosszú név következik
4410     80  I   a név karaktere
4411      0      záró záródjel kódja
4412      0      sorvég jelzése
*
*
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      0      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      1      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      2      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      3      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      4      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      5      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      6      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      7      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     E0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     F0      sorszám alsó byte
4498      8      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     00      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     10      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     20      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     30      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     40      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     50      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     60      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     70      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     80      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     90      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     A0      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     B0      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     C0      sorszám alsó byte
4498      9      sorszám felső byte
4499      1      aktulyázás mélysége
449A     96      Kulcsszó bevezető
449B     86      PRINT tokenje
449C      0      sorvég
*
4496     17      sorhossz
4497     D0      sorszám alsó byte
4498      9
```

A skatulyázás mélységét – egy bizonyos határig az egyre beljebb kezdett kiírást – a sorszám utáni byte jelzi. A kiírások egyre beljebb való kezdésének mértéke a TEXT üzemmódtól függ. A következő byte jelzi, hogy kulcsszó következik, majd az a sorszám, ahányadik helyen a ROM-ban lévő táblában az adott kulcsszó és a hozzátartozó információk elhelyezkednek. Ez a ROM táblázat az 5. szegmensen (az első lapon értelmezett), 323dec címtől kezdődő területén található. Mielőtt ezt a táblázatot vizsgálnánk, mi magunk is tanulmányozhatjuk a különböző utasítások tárolásának módját.

A 13. program 4Ø. sorába mindig más és más programsort beírva vizsgálhatjuk annak memóriában való elhelyezkedését. Lehet, hogy az 5Ø. sorba is kell írni valamit, mert az interpreter nemcsak a sor beíráskor végez bizonyos ellenőrzéseket, hanem a futtatáskor (pl. a FOR-NEXT szerkezeteket) is.

```
 1Ø ! sortarolas_vizsgalata
 2Ø GOTO 1ØØ
 3Ø !
 4Ø ! ide a vizsgalando sor
 5Ø !
 6Ø ! eddig ugyanigy kell beirni akkor igaz, hogy a
4Ø. sor eleje 4873 cimen van
 1ØØ LET H=PEEK(4873) ! 4Ø.sor hossza
 11Ø FOR L=4873 TO 4872+H
 12Ø   LET J=PEEK(L) !  a tarolt byte
 13Ø   PRINT L;J, !      cim,tartalom
 14Ø   IF J>31 AND J<16Ø THEN PRINT CHR$(J); !
ha kiirhato a karakter, ird ki
 15Ø   PRINT
 16Ø NEXT
```

13. program

Lesznek olyan kulcsszavak, melyeket nem hajlandó a számítógép a programsorba beépíteni. Pedig a számítógép ezeket is tokenizálja és az alapján értelmezi, valamint végrehajtja azokat parancsmódban. Azt, hogy mi alapján dönti el a gép, hogy egy adott kulcsszót utasításként, vagy parancsként, esetleg így is, úgy is elfogadjon, azt az előbb hivatkozott ROM tábla alapján végzi el. Vizsgáljuk meg ezt a táblázatot a 14. program segítségével.

```
1ØØ ! kulcsszo leiro 5. szegmensen
11Ø OUT 177,5 ! vizsgalathoz 5.szegmenst belapoz
12Ø FOR N=3Ø123 TO 31127
13Ø   LET A=PEEK(N) ! kulcsszo jelzobyte cime
```

```
140    GOSUB 200 ! byte-bit atvaltas
150    LET N=N+1:LET B=PEEK(N) !kulcsszo hossza
160    GOSUB 270 ! kulcsszo osszeallitas
170    LET N=N+4+B
180 NEXT
190 END
200 ! byte binaris kijelzese
210 LET O=128
220 FOR F=1 TO 8
230    PRINT INT(A/O);
240    LET A=MOD(A,O):LET O=O/2
250 NEXT
260 RETURN
270 ! kulcsszo osszeallitasa
280 FOR R=N+1 TO N+B
290    PRINT CHR$(PEEK(R)); ! karakterek kiirasa
300 NEXT
310 PRINT
320 RETURN
```

14. program

```
0  1  0  1  0  0  1  1 ALLOCATE
0  1  0  1  0  0  1  1 ASK
0  1  0  0  0  0  0  1 AUTO
0  1  0  1  0  0  1  1 CALL
0  1  0  1  0  0  1  1 CAPTURE
0  1  0  0  1  1  1  0 CASE
0  1  0  1  0  0  1  1 CAUSE
0  1  0  1  0  0  1  1 CLEAR
0  1  0  1  0  0  1  1 CLOSE
0  1  0  1  0  0  1  1 CODE
0  1  0  0  0  0  1  1 CONTINUE
0  1  0  1  0  0  1  1 COPY
0  0  0  0  0  0  1  0 DATA
0  1  0  0  0  0  1  0 DEF
0  1  0  0  1  0  1  0 DEF
0  1  1  0  0  0  0  1 DELETE
```

0	1	0	0	0	0	1	0	DIM
0	1	0	1	0	0	1	1	DISPLAY
0	1	0	0	1	0	1	0	DO
0	1	0	1	0	0	1	1	CHAIN
0	1	0	0	0	0	0	1	EDIT
0	1	0	0	1	1	1	0	ELSE
0	1	0	0	1	1	1	0	ELSE IF
0	1	0	0	0	0	1	0	END
0	1	0	0	0	1	1	0	END DEF
0	1	0	0	0	1	1	0	END HANDLER
0	1	0	0	0	1	1	0	END IF
0	1	0	0	0	1	1	0	END SELECT
0	1	0	0	0	1	1	0	END WHEN
0	1	0	1	0	0	1	1	ENVELOPE
0	1	0	1	0	0	1	0	EXIT
0	1	0	0	1	0	1	0	FOR
0	1	0	1	0	0	1	0	GOSUB
0	1	0	1	0	0	1	0	GOTO
0	1	0	1	0	0	1	1	GRAPHICS
0	1	0	0	1	0	1	0	HANDLER
0	0	0	0	0	0	1	0	IMAGE
0	1	0	0	0	0	1	0	IF
0	1	0	0	1	0	1	0	IF
0	1	0	1	0	0	1	0	INPUT
0	1	0	1	0	0	1	1	LET
0	1	0	1	0	0	1	0	LINE
0	1	0	0	0	0	0	1	LIST
0	1	1	0	0	0	0	1	LOAD
0	1	0	0	0	1	1	0	LOOP
0	1	1	0	0	0	0	1	MERGE
0	1	1	0	0	0	0	1	NEW
0	1	0	0	0	1	1	0	NEXT
0	1	0	0	0	0	1	0	NUMERIC
0	1	0	1	0	0	1	1	OPEN

```
0  1  0  0  0  0  1  1 OPTION
0  1  0  0  0  0  0  1 OK
0  1  0  1  0  0  1  1 OUT
0  1  0  1  0  0  1  1 PLOT
0  1  0  1  0  0  1  1 POKE
0  1  0  1  0  0  1  1 SPOKE
0  1  0  1  0  0  1  1 PRINT
0  1  0  1  0  0  1  0 PROGRAM
0  1  0  1  0  0  1  0 RANDOMIZE
0  1  0  1  0  0  1  0 READ
0  1  0  1  0  0  1  1 REDIRECT
0  0  0  0  0  0  1  1 REM
0  1  0  0  0  0  0  1 RENUMBER
0  1  0  1  0  0  1  0 RESTORE
0  1  0  0  0  0  1  0 RETRY
0  1  0  1  0  0  1  0 RETURN
0  1  0  1  0  0  1  1 RUN
0  1  1  0  0  0  0  1 SAVE
0  1  0  0  1  0  1  0 SELECT
0  1  0  1  0  0  1  1 SET
0  1  0  1  0  0  1  1 SOUND
0  1  1  0  0  0  0  1 START
0  1  0  1  0  0  1  0 STOP
0  1  0  1  0  0  0  1 INFO
0  1  0  0  0  0  1  0 STRING
0  1  0  1  0  0  1  1 TEXT
0  1  0  1  0  0  1  1 TOGGLE
0  1  0  1  0  0  1  1 TRACE
0  1  0  1  0  0  1  1 TYPE
0  1  1  0  0  0  0  1 VERIFY
0  1  0  0  1  0  1  0 WHEN
0  0  0  1  0  0  1  1 !
0  1  0  0  0  0  0  1 LLIST
0  1  0  1  0  0  1  1 LPRINT
```

```
0  1  0  1  0  0  1  1 EXT
0  1  0  1  0  0  1  1 GET
0  1  0  1  0  0  1  1 FLUSH
0  1  0  0  0  0  1  1 LOOK
0  1  0  1  0  0  1  1 PING
0  1  0  1  0  0  1  1 DATE
0  1  0  1  0  0  1  1 TIME
0  1  0  1  0  0  1  1 WAIT
0  1  0  0  0  0  1  0 ON
```

4. lista A 14. program futásának eredménye

A futás eredményét tekintsük át a következőkben. A bitsorozat értelmezése, ha az adott bit értéke 1, akkor a kulcsszó:

Ø Parancsként használható
1 Programban használható
2 Blokk vége utasítás
3 Blokk eleje utasítás
4 Több utasítás is lehet egy sorban
5 Direkt végrehajtásban a változókat törli
6 A sor kulcsszó utáni része is tokenizálandó

A ROM-beli kulcsszó táblában, tehát a kulcsszó előtt annak felhasználási, alkalmazási és tokenizálási szabályaira utaló byte található, ennek figyelése alapján dönt az interpreter. A 15. program segítségével különböző szempontok szerint kigyűjthetjük az egy kategóriába tartozó, tehát rokonságban lévő kulcsszavakat.

```
100 ! kulcsszavak csoportositasa
110 DIM T$(14)
120 CLEAR TEXT
130 FOR I=1 TO 7 ! szempont kiiras
140   READ T$(I)
150   PRINT I;T$(I)
160 NEXT
170 PRINT
180 FOR I=8 TO 14
190   READ T$(I)
200   PRINT " ";CHR$(57+I);" ";T$(I)
210 NEXT
220 OUT 177,5 ! vizsgalathoz belapoz
```

```
230 PRINT :PRINT :LET V=255
240 INPUT PROMPT "melyik szempont alapjan ? ":X$
250 LET X$=UCASE$(X$)
260 IF X$>="1" AND X$<"8" THEN
270   LET X=VAL(X$):LET V=X-1:LET W=1
280 END IF
290 IF X$>="A" AND X$<"H" THEN
300   LET X=ORD(X$)-64:LET V=X-1
310   LET X=X+7:LET W=0
320 END IF
330 IF V=255 THEN RUN
340 PRINT :PRINT T$(X)
350 FOR N=30123 TO 31127
360   LET A=PEEK(N) ! kulcsszo jelzo flag
370   LET H=PEEK(N+1) !kulcsszo hossz
380   IF (A BAND 2^V)/2^V=W THEN GOSUB 440
390   LET N=N+5+H !vegrehajtasi cimek atlepese
400 NEXT
410 PRINT "billentyut lenyomni"
420 IF INKEY$="" THEN GOTO 420
 430 RUN
 440 ! ha a keresett bitnek megfelelo a kulcsszo, akkor
kiir
 450 PRINT "     ";
 460 FOR K=N+2 TO N+H+1
 470   PRINT CHR$(PEEK(K));
 480 NEXT
 490 PRINT
 500 RETURN
 510 DATA parancsmodban hasznalhato
 520 DATA programban  hasznalhato
 530 DATA blokk vege utasitas
 540 DATA blokk kezdete utasitas
 550 DATA egy sorban tobb utasitas is lehet
 560 DATA adattorlessel kezd
 570 DATA sort kulcsszo utan is tokenizalja
 580 DATA parancsmodban nem  hasznalhato
 590 DATA programban  nem   hasznalhato
```

```
600 DATA nem blokk vege utasitas
610 DATA nem blokk kezdete utasitas
620 DATA THEN utan nem hasznalhato
630 DATA nem  adattorlessel kezd
640 DATA sort kulcsszo utan nem tokenizalja
```

15. program

Az előző programokkal és kulcsszó táblával kapcsolatban még egy kis kitérőt tehetünk.

A kulcsszó táblában két DEF és két IF kulcsszó található. A tokenizáló rutin az egy soros DEF, ill. IF struktúrákat másként értelmezi, mint a hasonló célú, de általában összetettebb blokkokat. Így a tokenizálás során és után kiderül, hogy blokk elejeként kell-e értelmezni az adott sort, és akkor blokk véget kell hozzá keresni, vagy sem. Ugyanakkor meg kell jegyezni, hogy a vizsgálatot többször végzi a számítógép, egyrészt a tokenizálás időszakában, másrészt a végrehajtás közben. Sajnos egyes esetekben nincs meg az összhang a beírás adta lehetőség, másrészt a végrehajtás során megvalósított vizsgálat között. A program beírás során pl. elfogad egy sorban több GOSUB utasítást kettősponttal elválasztva, de a végrehajtás közben olyan bejegyzést tesz a 2Ø6H rendszerváltozó címre, mely alapján a szubrutinból való visszatérés után a program futását a GOSUB-ot tartalmazó sor után fogja folytatni. Így tehát annak ellenére, hogy elfogadta a programsort a számítógép mint helyeset, a végrehajtás során elvész a kettőspont utáni programsor részlet. Próbáljuk ki és győződjünk meg ennek az állításnak igazáról a 16. program segítségével.

```
100 ! gosub hiba
110 LET A=0
120 GOSUB 200
130 !          <- ERROL NEM TUD ->
132 !          <- nem vegzi el  ->
133 !          <-nem szamozza at->
140 GOSUB 300:GOSUB 400:GOSUB 500
150 PRINT A
160 END
200 LET A=A+1
210 PRINT "200-as szubrutin"
220 RETURN
300 LET A=A+1
310 PRINT "300-as szubrutin"
320 RETURN
400 LET A=A+1
410 PRINT "400-as szubrutin"
```

```
420 RETURN
500 LET A=A+1
510 PRINT "500-as szubrutin"
520 RETURN
```

16. program

Mielőtt azonban belemerülnénk ezeknek a részleteknek a boncolgatásába, folytassuk tovább a RAM feltérképezését.

A program által lefoglalt tárterület után a változókat tárolja a számítógép. Ez a terület változó nagyságú, és kezdete is függ a program hosszától. Azt, hogy egy adott esetben ennek a területnek mik a határai, azt a rendszerváltozó terület megfelelő címeiről olvashatjuk ki (17. program).

```
100 ! valtozo tabla memoriabeli helye,merete
110 LET A=2 ! kiserletezz ertekadasokkal
120 !vizsgald a hatasat:DIM,DEF,A$
130 LET ELEJE=PEEK(566)+256*PEEK(567)
140 LET VEGE=PEEK(564)+256*PEEK(565)
150 PRINT "valtozo tabla eleje: ";ELEJE
160 PRINT "valtozo tabla vege : ";VEGE
170 PRINT "valtozo tabla altal foglalt:";VEGE-ELEJE
```

17. program

Ezen a területen a gyanútlan szemlélőnek kusza összevisszaságban helyezkednek el a bejegyzések, pedig logikája nyomon követhető. Nézzünk bele a változóterület tárolására szolgáló memóriaterületbe a 18. program segítségével.

```
100 !szamabrazolas
110 LET SAMU=1.2345678 !ezt valtoztasd
120 LET M=PEEK(566)+256*PEEK(567)
130 ! M az elsonek bejegyzett valtozo tarolasi helye
140 LET H=PEEK(M+3) !valtozonev hossza
150 PRINT M+2;PEEK(M+2),"adattipus jelzo"
160 PRINT M+3;H,"valtozonev hossza "
170 FOR V=M+4 TO M+H+3 !nev kiiras
180   PRINT V;PEEK(V);CHR$(PEEK(V))
190 NEXT
200 FOR V=M+H+4 TO M+H+8
210   LET T=PEEK(V) !tarolt ertek
220   LET E=INT(T/16) !BCD elso tag
230   LET M=MOD(T,16) !BCD masodik tag
```

```
240    PRINT V;T,E;M
250 NEXT
260 PRINT V;PEEK(V),"hatvanykitevo"
```

18. program

A numerikus változók értékét változó táblában 5+1 byte-on tárolja a gép. Egész típusú – 1Ø ØØØØ×1Ø ØØØ nagyságú számokat két byte-on hexadecimális alakban találhatjuk a memóriában és a hatványkitevő tárolására szolgáló 6 byte tartalma 127. A 9999-nél nagyobb egészeket és a törtszámokat a gép BCD alakban ábrázolja.

Hogyan néz ki egy decimális szám BCD alakban? A tízes számrendszerű szám egy helyi értékének ábrázolásához négy bit bőven elegendő, így lehetőség van arra, hogy nyolc biten (egy byte-on) két decimális számjegyet ábrázoljunk úgy, hogy az egyik (alacsonyabb helyi értékű) számjegyet binárisan a Ø–3. biten, a másik a magasabb helyi értékű számjegyet pedig a 4–7. biten tároljuk.

Azaz 29DEC=ØØ1Ø 1ØØ1 BCD, igaz ez a szám a szokványos módon kiolvasva (hiszen PEEK-kel kiolvashatjuk) egészen más értéket reprezentál. Ezért kell tudni, hogy a kiolvasott érték BCD-ben van!

Ez a számábrázolás-tárolás kissé pazarló, hiszen öt byte-on tízjegyű decimális szám ábrázolható, míg tisztán bináris (vagy hexadecimális) formában ugyanezen az öt byte-on $2 \cdot 749 \cdot 1Ø^{14}$ értékű a maximálisan ábrázolható szám. Ez a pazarlás a műveletek elvégzése során időben megtérül. A binárisan kódolt decimális rendszerben (BCD) a 1Ø-es alapot megtartjuk, azaz a szomszédos számok egymásnak tízszeresei, de a számokat binárisan fejezzük ki.

Az elmondottak jobb megvilágítására, ill. a szerzett ismeret hasznosítására futtassuk le a 19. programot.

```
100 ! PEEK-kel kiolvasott ertek visszaalakitasa
ha az BCD
110 LET PROBA=1988.12 !csak tortet
120 LET M=PEEK(566)+256*PEEK(567)
130 LET H=PEEK(M+3)
140 IF PEEK(M+H+9)=127 THEN GOTO 330
150 LET SZORZO=.ØØØØØØØØØ1
160 LET SUMMA=Ø
170 FOR V=M+H+4 TO M+H+8
180    LET T=PEEK(V)
190    LET E=MOD(T,16) !BCD elso tag
200    LET SUMMA=SUMMA+E*SZORZO:LET SZORZO=SZORZO*1Ø
210    LET M=INT(T/16) !BCD masodik tag
220    LET SUMMA=SUMMA+M*SZORZO:LET SZORZO=SZORZO*1Ø
230    PRINT T,M;E
```

```
240 NEXT
250 LET KI=PEEK(V)
260 LET S=(KI BAND 128)/128
270 LET KI=KI-64-128*S
280 IF S=1 THEN LET ELOJEL=-1
290 IF S=Ø THEN LET ELOJEL=1
300 PRINT KI,"hatvanykitevo"
310 PRINT ELOJEL*SUMMA*1Ø^(KI+1)
320 END
330 PRINT "ez egesz szam, erre mas szabaly igaz"
```

19. program

A PEEK utasítással kiolvasott érték mellett a táblázat mutatja azt az értéket, ahogyan értelmezni kell a memóriarekeszek tartalmát. Az öt byte-on tárolt számsorozat csak a számjegyeket tárolja szépen egymás után, a helyi értéket, a nagyságrendet a karakterisztika határozza meg. Alapértelmezés $1Ø^{Ø}$, ilyenkor a kitevő byte 4ØH azaz PEEK-kel kiolvasva 64. A számítógépben maximálisan ábrázolható szám hatványkitevőjének tárolásához nincs szükség a 7. bitre ez az előjel tárolására szolgál a valós számok esetében. Ha ez a bit 1 akkor a tárolt szám negatív. Természetesen nemcsak numerikus változókat tárol a számítógép. A tárolt adat jellegét a bejegyzés 3. byte-ja jelzi. A jelző byte értéke

Ø numerikus változó A
1 string változó A$
2 numerikus tömbváltozó A(1)
3 string tömbváltozó A$(1)
4 felhasználói függvény def név
5 felhasználói függvény def név $
8 gépi kódú függvény
12 függvény numerikus érték SIN (X)
13 függvény string értékkel STR$ (A)
16 referencia adat

Jobban nyomon követhetjük a jelző byte logikáját, a bitek külön-külön való vizsgálatával.

Ø. bit numerikus=Ø, stringváltozó=1
1. bit értéke 1, ha tömbváltozó
2. bit értéke 1, ha felhasználói függvény
3. bit értéke 1, ha gépi kódú függvény
4. bit értéke 1, ha referencia adat.

A számítógép 1 szegmensén az Enterprise több konstanst tárol. Ezek tárolt értékét és visszafejtett értékét mutatja a 20. program futtatása során.

```
 100 ! 1.szegmens konstansainak visszafejtese
 110 FOR CIM=16524 TO 16590 STEP 7
 120   LET S=0:LET O=1
 130   FOR V=CIM+5 TO CIM STEP-1
 140     LET T=SPEEK(1,V)
 150     LET E=INT(T/16):LET M=MOD(T,16) !BCD vissza-
alakitasa
 160     LET S=S+(E+M/10)/O !helyi ertek es osszegzes
 170     LET O=100*O
 180   NEXT
 190   LET J=SPEEK(1,CIM+6) ! a kitevo
 200   LET L=(J BAND 128)/128
 210   LET J=J-64-128*L
 260   LET E=(L=1)-(L=0) !elojel
 270   PRINT "konstans=";E*S*10^J
 300 NEXT
```

20. program

Nézzük meg mi a helyzet, akkor ha a programunk első soraiban több értékadás is történik. Ilyenkor már derengeni kezdhet az adott változó tábla-beli bejegyzés első két byte-jának, a láncolási címnek a jelentősége (21. program).

```
 100 ! dump
 110 LET A1=1:LET A$="samu" !ertekadasok probahoz
 120 LET A=2.2:LET SAMU$="?"
 130 LET A2=1
 140 DIM B(2),C$(3)
 150 DEF Y(X)=2*X
 160 FOR I=3531 TO 3593 STEP 2 ! DCBH tablazat cimei
alapjan kereses indul
 170   LET K=PEEK(I)+256*PEEK(I+1) ! lancolasi cim
kiszamitasa
 180   IF K>4827 THEN ! ha a valtozo
BASIC RAM teruleten van
 190     LET S=I
 200     GOSUB 250
 210     LET I=S
 220   END IF
 230 NEXT
```

```
240 END
250 ! valtozo tablaban lancolasi cimen vegigfut
260 LET I=K ! vizsgalt adatcsomag eleje
270 PRINT PEEK(K+2); !valtozo tipus
280 FOR W=1 TO PEEK(K+3)
290   PRINT CHR$(PEEK(K+3+W)); ! valtozo neve
300 NEXT
310 PRINT
320 LET K=PEEK(I)+256*PEEK(I+1)
330 IF K<4827 THEN RETURN  ! vissza, ha BASIC ele
mutat a cim, lanc vege
340 GOTO 250
```

21. program

A program az eddig tárgyaltak összefoglalása is lehetne. A változó tábla bejegyzett konstansokat a DCBH címen kezdődő táblázat adatai segítségével keresi meg a rendszerprogram. Az adatok egymás után, láncolva helyezkednek el a változó táblában. A láncolási címek alapján található meg egy-egy azonos táblázatbeli címre „feltűzött" változók sora. A DUMP program ezt a vizsgálatot végzi el, és addig keres az azonosító láncban, amíg a lánc végére nem ér. A lánc végét az jelzi, hogy a láncolási cím visszamutat a Basic-ben írt felhasználói program elé. Az előbbi gondolatmenet visszafelé is végigvezethető, így a DCB tábla bejegyzésein keresztül megkereshetjük egy-egy változó tárolási helyét a memóriában.

A 22. program a más számítógépek függvénykészletében megtalálható VARPTR hatását utánozza.

```
100 ! varptr
110 LET A1=1:LET A$="samu" !ertekadas probahoz
120 LET A=2.2:LET SAMU$="?"
130 LET A2=1
140 DIM B(2),C$(3)
150 INPUT PROMPT "a keresett valtozo neve: ":K$
160 CALL VARPTR
170 PRINT :GOTO 150
180 END
190 !
200 DEF VARPTR
210   FOR I=3531 TO 3593 STEP 2 ! DCBH tabla alapjan
keres
```

```
 220      LET K$=UCASE$(K$)
 230      LET K=PEEK(I)+256*PEEK(I+1) ! lancolasi cim
meghatarozasa
 240      IF K>4827 THEN ! keres ha Basic teruletre
mutat a cim
 250         LET S=I
 260         LET I=K
 270         LET E$=""
 280         FOR W=1 TO PEEK(K+3) ! nev azonositas
 290            LET E$=E$&CHR$(PEEK(K+3+W))
 300         NEXT
 310         IF E$=K$ THEN PRINT "VARPTR(";K$;")=";K !
ha megtalalta a csomag cimet kiirja
 320         LET K=PEEK(I)+256*PEEK(I+1) !lancot tovabb
elemez
 330         IF K<4827 THEN GOTO 350 !egy vizsgalt lanc
vege
 340         GOTO 260
 350         LET I=S
 360      END IF
 370      IF E$=K$ THEN EXIT FOR !ne keress tovabb ,
mar megvan
 380   NEXT
 390   IF E$<>K$ THEN PRINT "ilyen nincs"
 400 END DEF
```

22. program

A programban az első sorokon (a használhatóság kedvéért) értékadásokat végeztünk. A keresett változó nevét INPUT-ként közölhetjük programunkkal. A DCB táblázatba bejegyzett ugrási címek vizsgálata után ugrik a változó táblába a kereső rutin. Az eredeti (a gép rendszerprogramjának) kereső rutinja RST 1Ø/8E a DCB egészét nem vizsgálja végig. Egy, a változó névből képzett áltoken – egy-egy byte hosszú sorszám – segítségével csak a DCB táblázat n. helyéről induló címláncot vizsgálja. Ha a kapcsolószám alapján elér a mi kereső programunk egy adat tárolási terület elejére, akkor a bejegyzés 4. helyén kezdődő változónevet összehasonlítja az általunk beírt K$-keresett változó névvel. Amennyiben az azonosítás sikerül, úgy kiírható a változó tárolási helyének címe, vagy akár összes adata és értéke.

RST rutinok

A RST rutinokat minden Z8Ø-as processzorra épülő számítógépben speciális, gyakran előforduló programrészek, vizsgálatok elvégzésére használják fel. Ezek felhasználása ugyan a gépi kódú programok írásánál lehet fontos, de az Enterprise RST 1Ø rutinjai olyan elegáns programozási ötleteket, fogásokat tartalmaznak, melyek tanulmányozásán keresztül a programozási logikánk – a Basic nyelvű is – sokat fejlődhet.

A következőkben fő ismérveivel felsoroljuk a RST 1Ø rutinok működését. A RST 1Ø rutinra a Ø lapon lévő 1ØH címen kell belépni.

Ez a Ø. lapon lévő címen keresztüli meghívási lehetőség azért szükséges, mert pl. az 5. szegmensen lévő Basic utasításokat végrehajtó rutinok gyakran használják az 1. szegmensen lévő szintaktikai elemzést biztosító utasítássort (makrót). A RST 1Ø rutin elemzéséből láthatjuk, hogy a program következő byte-jai egy-egy újabb rutin meghívását biztosítják, egészen addig amíg a következő byte nem nulla.

```
0010    JP 009A             tényleges végrehajtó
                            rutinra ugrik

.
.
009A    EX (SP),IY          kezdeti állapotok
009C    PUSH HL             mentése
009D    LD HL,(00DA)
00A0    EX (SP),HL
00A1    LD (00AE),A
00A4    LD A,(00FA)
00A7    PUSH AF
00A8    IN A,(B3)           P3 mentése, a
00AA    LD (00FA),A         visszaállításhoz
00AD    LD A,xx             átmeneti tárból vissza
00AF    PUSH HL
00B0    PUSH AF
00B1    LD L,(IY)           a végrehajtandó RST 10
                            rutin sorszáma L-be
0084    INC IY              a rendszerProgram
                            következő byte-jának
                            címe
0086    LD H,C0             ugrócím táblabeli
                            elem helyének meg-
                            határozását előkészíti
00BB    CALL 12CB           1. vagy 4. szegmensen
                            keresendő a rutin?
                            ha igen Cy és A=1
                                    vagy A=4
00BB    NOP                 és ugrik
00BC    JR C,00C6
00BE    PUSH DE
00BF    LD DE,00E8          5. szegmensen lévő
00C2    ADD HL,DE           táblabeli cím meghatározása
00C3    POP DE
00C4    LD A,05             akkor A=5
00C6    OUT (B3),A          aktuális szegmens P3-ra
```

```
00C8    LD A,(HL)       az uzenetek táblából
00C9    INC HL          a RST 10 aktuális
00CA    LD H,(HL)       rutinjának végrehajtási
00CB    LD L,A          címe HL-be
00CC    LD (00F6),HL    és átmeneti tárba
00CF    POP AF
00D0    POP HL
00D1    LD (00D0),IY    veremrendezés
                        további feldolgozáshoz
                        címet tárol
00D5    CALL 00F5       aktuális végrehajtó
                        rutin meghívása
00D8    LD IY,xxxx      IY visszaállítása (verem művelet
                        helyett)
00DC    JR 00AF         RST 10 további végrehajtása

00DE    INC SP          (00F8) visszatérési
00DF    INC SP                 cím törlése
00E0    INC SP          (00D8) visszatérési
00E1    INC SP                 cím törlése
                        mert ez a RST 10/0
00E2    EX (SP),HL

00E3    CALL 00F8       P3 visszaállítása
00E6    PUSH AF
00E7    LD A,H
00E8    LD (00FA),A
00EB    POP AF
00EC    POP HL          RST 10 rutinok meg-
00ED    EX (SP),HL      hívásakor fenálló
00EE    LD(00D0),HL     állapot visszaállítása
00F1    POP HL
00F2    EX (SP),IY
00F4    RET             és vissza
00F5    CALL xxxx       a címtáblából meg-
                        állapított végrehajtó
                        rutin meghívása

00F8    PUSH AF
00F9    LD A,xx         P3 visszalapozása
00FB    OUT (00),A
00FD    POP AF
00FE    RET
```

5. lista RST végrehajtó rutin

Tekintsük át a RST 1Ø után következő számok értelmezését.

ØØ Ø. lapon ØØDEH
A RST 1Ø rutinhívások sorozatából visszatér a program folytatására. Ehhez stack-pointert rendezi, visszaállítja a regiszterek alapállapotát és visszalapozza a megfelelő szegmenst.

Ø1 5. szegmens F3Ø1H
Hibakezelés a RST 18 és RST 2Ø rutinok közös vége csatorna-alapállapotok beállítása, státuszsor kiírása (ha engedélyezett). Hibaüzenet kiírása és beolvasási főágra ugrik.

Ø2 5. szegmens E8EFH
HL-ben lévő egész számot a Basic verembe teszi. A tárolási cím a 228H rendszerváltozóban tárolt érték. A rutin ennek értékét is értelemszerűen módosítja.

Ø3 5. szegmens E35CH
A 228H által megcímzett Basic verem (legutoljára tárolt) számértékét karakteres formában való kiíráshoz előkészíti úgy, hogy egyidejűleg a veremben is meghagyja. A kiírásra váró karaktersor 59EH címen kezdődő pufferbe kerül.

Ø4 5. szegmens EECDH
A HL tartalmát invertálja. Egyes komplemens képződik és ez a HL új értéke.

Ø5 5. szegmens E991H
A Basic verem tetején lévő szám előjelét vizsgálja. A vizsgált szám tárolási címét az IY regiszter, ill. a 228H rendszerváltozó tárolja.
A vizsgálat eredményeként Z flag 1, ha a szám pozitív.

Ø6 5. szegmens E9CBH
A Basic verem tetején lévő szám előjelét ellenkezőre változtatja.
(IY)=—(IY)

Ø7 5. szegmens E9C7H
A Basic verem tetején lévő szám előjelét vizsgálja, ha negatív, akkor az előjelét ellenkezőre változtatja.
Az ABS függvény végrehajtó rutinja (IY)=ABS (IY)

Ø8 5. szegmens E591H
A Basic verem tetején lévő (lebegőpontos) szám tízes komplemensét állítja elő. Az eredményt az eredeti érték helyén tárolja.

Ø9 5. szegmens E58DH
Vizsgálja a verem tetején lévő (lebegőpontos) számot, ha az negatív, előállítja annak tízes komplemensét.

ØA 5. szegmens EE26H
A tár tetején lévő számot vizsgálja.
Visszatéréskor akkumulátorban a szám kitevője.
Z flag 1, ha a szám egész típusú.

ØB 5. szegmens E7F9H
A tár tetején lévő lebegőpontos számot, ha annak értéke kisebb mint 1Ø ØØØ egész típusúra alakítja, és az eredményt a HL-be teszi. Ha az átalakítás nem végezhető el hibajelzéssel, leáll.

ØC 5. szegmens E5Ø1H
A tár tetején lévő két (lebegőpontossá alakított) számot összeadja, az eredményt a korábban tárolt szám helyére tölti. (IY+9) = (IY+9)+(IY)
A veremmutató pedig IY = IY+9

ØD 5. szegmens E4F8H
A tár legutoljára bejegyzett elemét kivonja az eggyel korábban tárolt értékből.

(IY+9) = (IY+9)—(IY)
A veremmutató IY = IY+9

ØE 5. szegmens E5A5H
A Basic tár tetején lévő két számot összeszorozza.
(IY+9) = (IY+9)×(IY)
A veremmutató IY = IY+9

ØF 5. szegmens E6CØH
A tár utoljára bejegyzett adatával osztja a korábban behelyezett értéket.
(IY+9) = (IY+9)/(IY)
A veremmutató IY = IY+9

1Ø 5. szegmens E9E7H
A tár két utolsó elemét kivonja egymásból.
(IY+9)—(IY) eredményként a jelző flaget adja
M ha (IY+9) < (IY)
P ha (IY+9) > (IY)
Z ha (IY+9) = (IY)
A veremmutató IY = IY+12H, azaz törli a tárból a két utoljára bejegyzett elemet.

11 5. szegmens EAB7H
A tár tetején lévő lebegőpontos számot 1Ø-nek az akkumulátorban hozott értékének megfelelő hatványával osztja.
(IY) = (IY) 1Ø ↑A

12 5. szegmens E9D1H
A tár tetején lévő elemet vizsgálja, ha értéke Ø, akkor egész típusú nullára alakítja és Z flaget 1-re állítja.

13 5. szegmens E865H
A tár tetején lévő számot lebegőpontosra alakítja, ha az egész típusú volt.

14 5. szegmens E8BDH
HL által kijelölt címről a Basic verembe tölti a lebegőpontos számot.
(IY)=(HL)
IY = IY—9

15 5. szegmens EA8BH
A Basic tár tetején lévő elemet még egyszer a tárra tölti.
IY = IY—9

16 5. szegmens EDE8H
Törli a tár tetején lévő elemet.

17 5. szegmens E979H
A tár legfelső elemének helyére egész típusú nullát tölt.
(IY)=Ø
IY=IY

18 5. szegmens EA63H
Az előző rutin felhasználásával a tár legfelső elemének helyére nullát tölt, majd lebegőpontos alakra alakítja a hatványkitevő módosításával.

(IY)=Ø.ØØ
IY=IY

19 5. szegmens EA6FH
A tár legfelső elemének helyére egész típusú egyet tölt.
(IY)=1
IY=IY

1A 5. szegmens EA7BH
A tár tetején lévő adat helyére lebegőpontos egyet tölt.
(IY)=1,ØØØØØØØØØ
IY=IY

1B 5. szegmens E9ØAH
A tár legfelső elemét megfelelő átalakítással a HL által címzett területre tölti.
(HL)=(IY)
IY = IY+9

1C 5. szegmens E925H
A tár tetején lévő stringet a HL által címzett területre másolja, de a tár tetején is meghagyja.

1D 5. szegmens E79CH
A tár legfelső elemének valamelyik számjegyét felülírja. A tárban a számok BCD alakban vannak. A tár IY címén kezdődő numerikus változó az alábbi „csomagban" helyezkedik el.

IY TÍPUS	SZÁMJEGY SORSZÁMA	B REGISZTER JELZÉSE
IY+1		
IY+2	9. A	C.D
IY+3 BCD	7. 8	A.B
IY+4 SZÁMJEGYEK	5. 6	8.9
IY+5	3. 4	6.7
IY+6	1. 2	4.5
IY+7	Ø	2.3
IY+8 kitevő és előjel		Ø.1

Ahol a legmagasabb helyi értékű szám az 1. sorszámú. Az átírandó tárolt számjegy ezek szerint a 4–13. „fél byte"-on helyezkedik el.
A lecserélendő számjegy sorszámát B regiszterben a számjegy új értékét az A regiszter tartalmazza.

1E 5. szegmens E7B7H
A Basic veremben tárolt legfelső adat, jelen esetben numerikus adat egy, B regiszterben megjelölt sorszámú (fél byte-on tárolt) decimális számértéket kiolvas, és az akkumulátorba tölti.

1F 5. szegmensen E93FH
A tárban lévő és nem az 1D rutinban jelzett alakban tárolt számjegyet normalizálja úgy, hogy a legmagasabb helyi értékű helyre értékes számjegy kerüljön.

2Ø 5. szegmens F259H
A programvégrehajtás közben a programsor elemzését elvégzi. Beolvassa a következő adatot.
Módosítja a 214H és 2Ø2H–2Ø4H tartalmát.
Ha sorvéget talál Z flag-et 1-re állítja.

21 5. szegmens E2E4H
Programsor végrehajtás közben a 2Ø2H és 2Ø4H tartalmát, azaz a feldolgozás alatt álló elem jellegét figyeli, elemzés alapján hívja a további rutinokat.

22 5. szegmens F253H
Színtaktikai elemzést végez, az akkumulátorban lévő és a 2Ø4H memóriacímen lévő adatok összehasonlításával. Különbözőség esetén hibajelzés. Azonosság esetén a 2ØH sorszámú RST 1Ø rutinon folytatódik.

23 5. szegmens DEE2H
Programsor elemzése közben talált numerikus kifejezés kiértékelése, értékének kiszámítása.
Az eredményt a Basic verem tetejére teszi, IY értékét növeli.

24 5. szegmens EØ96H
A programsor elemzése, végrehajtása közben talált stringműveletek elvégzése. Az eredményként kapott string a verembe kerül.

25 5. szegmens FB9EH
Az EXOS (RST 3Ø) 5. rutinjának meghívásával a billentyűzetről egy karaktert olvas be. A-ban a beolvasott karakter, amit 2Ø5H-ra és Basic verembe egy karakter hosszú stringként tárol. Ha billentyű lenyomás nem történt, Ø-val tér vissza.

26 5. szegmens EEBEH
A-ban hozott értéktől (4ØH, vagy 6ØH) függően a vizsgált karaktert nagy, vagy kisbetűsít.

27 5. szegmens FØ94H
Ellenőrizni, hogy a tárban van-e elegendő hely. Ha szükséges a tárat átszervezi.

28 5. szegmens FØ97H
DE-ben hozott hosszúságú memóriaterületet foglal a string, tárba való töltéshez. Ha szükséges és lehet, a tárat is átszervezi.

29 5. szegmens FØ1ØH
OPEN előtt a memóriát átszervezi. Az EXOS részére helyet szabadít fel.

2A 5. szegmens C6C2H
A végrehajtás és tokenizálás közben a sorfeldolgozás közben az üres helyeket [CHR$ (9) és CHR$ (32)] átlépi. A 2144 rendszerváltozó értékét az első „használható" karakterre állítja.

2B 5. szegmens E41EH
Karakteres formában adott számot a 214H rendszerváltozó által kije-

lölt helyről a szükséges BCD átalakítással az 595H címen kezdődő puf-
ferbe tölti.

2C 5. szegmens E4Ø7H
A 59DH címen kezdődő tárból a Basic verem tetejére lebegőpontos számot tölt át.
IY=IY—9

2D 5. szegmens D4D4H
Az RND függvényhez 3 byte-os álvéletlen számsorozatot állít elő. Akkumulátor értékét 211H HL értékét 246H rendszerváltozó címekre tölti.

2E 5. szegmens C3FDH
HL-ben adott sorszámú sor helyét keresi a memóriában. Ha ilyet nem talál C jelzőbit értéke 1.
A rutinból való visszatérés után HL-ben a megtalált (vagy ha ilyen nincs, akkor a következő sor) kezdetének címe a memóriában.
A BC regiszterpár a megtalált sor hosszát tartalmazza.

2F 5. szegmens C858H
A 2Ø9H rendszerváltozó által meghatározott csatornára kiírja azt a programsort, mely a memóriában a HL által meghatározott memóriacímen kezdődik.
Ismételt meghívással folyamatosan ír ki, mert visszatéréskor HL a következő programsor elejére mutat.

3Ø 5. szegmens C544H
A HL által kijelölt helyen kezdődő programsort törli a memóriából. Törli a változókat és a 2ØØH rendszer változóból a CONTINUE lehetőségét jelző bitet, azzal hogy folytatódik a 31. rutinon.

31 5. szegmens C54DH
Változók törlése, 1–1ØØ csatornák zárása a skatulyázások ellenőrzése. A rendszerváltozók beállítása pl. DATA mutató a program elejére.

32 5. szegmens EAD5H
A DEF és HANDLER tokeneket megkeresi a programban, ha talál ilyeneket, akkor a RST 1Ø–89H rutin segítségével felveszi a változó táblába azokat. A változó tábla ezekről a blokkokról a következő bejegyzéseket végzi el:

előző bejegyzés címe	két byte
felhasználói függvény jelző	egy byte értéke (4 vagy 5)
az azonosító név hossza	egy byte értéke (*n*)
az azonosító neve	*n* byte
	6 byte

a DEF HANDLER blokk
memóriabeli kezdetének helye 2 byte.

33 5. szegmens EF7AH
Blokk-kezdő utasításhoz blokkvéget keres. A kereséshez egyrészt a tokent, másrészt a skatulyázás mélységét jelző byte-ot használja fel.

Amennyiben az adott skatulyázási mélységben a megfelelő blokkvéget nem találja, akkor hibajelzést ad. Ha megtalálja a megfelelő blokkvéget, akkor a rutin végén HL a blokkvég kezdetére mutat.

34 5. szegmens EF19H
A STOP billentyű lenyomása után a 2ØØH rendszerváltozó 1. bitjének értékétől függően a STOP, vagy STOP at line üzenet kiírása. Ezután parancsmódba beolvasási főágra ugrik.

35 5. szegmens EFD7H
A 232H rendszerváltozóban adott kezdőcímen induló RAM-beli kulcsszótábla alapján a token segítségével az adott kulcsszó típusát határozza meg.
Visszatéréskor C-ben a kulcsszó sorszáma, HL az 5. szegmensben lévő kulcsszótáblában az adott kulcsszó típusjelzőjére mutat. E-ben a típusjelző, ennek jelentését már a 14. program segítségével vizsgáltuk.

36 5. szegmens FBB5H
A 2Ø9H által kijelölt csatornáról az EXOS 5 rutin segítségével a 248H címen kezdődő input pufferbe olvas. A beolvasott szöveg végére nullát ír a pufferbe és a beolvasott karakterek számát a puffer első helyén HL által mutatott 248H címre tölti. A szöveg hossza E-ben van.

37 5. szegmens C74BH
Az input pufferből az új programsort a programtárba tölti. Az aktuális programsort, tehát beilleszti a programba.

38 5. szegmens FB66H
A 2Ø9H rendszerváltozó értékétől függő csatornára kiírja az akkumulátor tartalmát.

39 5. szegmens EEDAH
A HL-ben lévő maximálisan 9999 nagyságú egész számot (mint pl. a programsor sorszáma) legalacsonyabb helyi értékre „zárva" írja ki, a nulla értékű magasabb helyi érték helyére szóköz kiírásával.

3A 5. szegmens C4DEH
Input pufferben lévő szöveg elemzése, THEN utáni kifejezés elemzése.
A programsor beépítése a programtárba.

3B 5. szegmens EC83H
A változó táblában lévő elemet vizsgálja, ha annak típusa tömbváltozó, azaz 1. bitje egy értékű, akkor a HL értékét átállítja a tömbváltozó n. elemének tárolási címére.

3C 5. szegmens CE69H
Ha a program adott utasítása csatornaszám kijelölését igényli, akkor megkeresi a programsor következő elemét.
Szintaktikai elemzést végez #-ra, ha az, akkor tovább elemzi a sort. Kiszámítja a csatornaszámot és azt az akkumulátorba tölti.

3D 5. szegmens DØ68H
Tokenizálás közben vizsgálja a sort, a hátralévő részek értelmezése érdekében.

3E 5. szegmens FB84H
A HL tartalmaként adott darabszámú karaktert ír ki a 2Ø9H által jelzett csatornára. A kiírandó szöveg a memóriában HL+1 címen kezdődik.

3F 5. szegmens E335H
A Basic tár tetején lévő számot karakteres formában kiírható alakra alakítja és 59EH címen kezdődő tárba tölti. A tárba a szám elé szóközt, vagy mínuszjelet, a szám után szóközt helyez. Az így összeállított karaktersort a RST 1Ø–3E rutin segítségével kiírja.

4Ø 5. szegmens EØ93H
A 24-es rutinhoz hasonlóan elemez egy string kifejezést, de attól eltérően nem végzi el a műveleteket.

41 5. szegmens DED9H
A 23-as rutinhoz hasonlóan elemez egy numerikus kifejezést, de attól eltérően nem tárol eredményt.

8Ø 1. szegmens D1B2H
DCBH címen kezdődő táblázat áltoken által meghatározott helyére beírja az illető függvény (vagy változó) változó tábla láncban való kereséshez szükséges címét. Az áltoken egy, a függvény (változó) nevéből képzett egy byte-os sorszám, melyet az azonosításra szolgáló karaktersorozat elemeinek összeolvasásából alakul ki. Az áltoken előállító rutin az 1 szegmensen D1A2H címen található. Az áltoken értéke Ø és 31 közötti szám.

81 1. szegmens DØ4DH
A változó tábla első üres helyére tárolja az újonnan bejegyzendő elem típusát, nevének hosszát és nevét.
Akkumulátorban a bejegyzés típusa, C-ben nevének hossza.

82 1. szegmens DØ7CH
A változó táblába előbb (81. rutin által) bejegyzett azonosító után értékadás nélküli jelzéssel 6 byte hosszú numerikus adatnak megfelelő byte sorozatot ír. Ez azt jelenti, hogy pl. PRINT A utasítás végrehajtása során, ha még nem volt. A változónak értéke, akkor a változó táblába felveszi az A-t (81. rutin) és a 82. rutin segítségével ØØ, ØØ, ØØ, ØØ, FF, 7F byte sorozatot ír. Ahol a hatványkitevőt (ill. egész típusú értéket jelző) byte előtti FF jelzi, hogy értékadás még nem történt.

83 1. szegmens DØ94H
Ugyanazt végzi, mint a 82. rutin, de stringet jegyez be. Gyakorlatilag elvégez egy memóriaellenőrzést, amint string azonosítót talál és alapértelmezésben 132dec hosszúságú memóriaterületet foglal a karaktersorozat részére. FF jelzést ír a hossz byte után. (Ha majd az értékadás is megtörténik, akkor a rendelkezésre álló területre írja a karaktereket és törli az FF jelzést.)

84–87 rutinok csak áttételesen értelmezhetők. Az 1 szegmens CØØØH címtől kezdődő területén 2 byte hosszúságú ugrócímeket tartalmazó táblázat

található az e szegmensen lévő RT 1Ø rutinok eléréséhez. A táblázat CØØ8–CØØF területén található adatok szokásos kiolvasással ugrócímnek nem értelmezhetők. Az itt található byte-ok:
6A, 67, C3, E3, F4, C3, ØF, E4
A 6A, 67 byte-on címként értelmezve e szegmens 1. lapra való belapozás esetén a PRINTER álleírójára mutatnak.

A C3E3F4 JP F4E3	belépést biztosít a szövegszerkesztőbe
A C3ØFE4 JP E4ØF	akkumulátorban hozott értéktől függően RESET-et hajt végre, vagy az idő és dátum adatok írását, olvasását teszi lehetővé.

88 1. szegmens DØE7H
A 2Ø2H rendszerváltozóban lévő típusjelzőtől függően a RST 1Ø–81 rutin végrehajtása után vagy a 82., vagy a 83. rutint hajtja végre. Így a változó táblába elvégzi az azonosító és az értékadás nélküli változó bejegyzést. A változó tábla új első szabad helyének címét betölti a 234H rendszerváltozó címre.

89 1. szegmens DØABH
A DEF-vel megadott függvényt veszi fel a változó táblába. Megvizsgálja, hogy az adott azonosítót (nevet) nem használjuk-e még. Ha igen, hibajelzést ad. A változó táblába bejegyzi a láncoláshoz szükséges adatokat, azonosítót, nevet és a függvény definiálást végző programsor memóriabeli helyét (vö.: RST 1Ø–32 rutinnal)

8A 1. szegmens D1Ø1H
Referencia paraméterrel hívott függvények hívása esetén a változó táblába felvett új adatcsomag típusjelzőjében a 4. bit egyre állításával jelzi, hogy ez az adat egy másik belépésre hivatkozás.

8B 1. szegmens D189H
Program feldolgozása közben a változó táblában keresi az aktuális nevű (azonosítójú) változó bejegyzését. (Ha nem talál ilyet, akkor értékadás nélkül elkészíti a bejegyzést.) Ha megtalálja az adattárolás helyét, akkor HL-ben a keresett terület elejét adja vissza (23. program).

```
100 ! varptr
110 ALLOCATE 100
120 CODE M=HEX$("d7,24,00,2a,28,02,23,7e,32,03,02,23,
22,47,03,d7,8e,00,c9")
  130 LET A1=1:LET A$="samu" !ertekadas probahoz
  140 LET A=2.2:LET SAMU$="?"
  150 LET A2=1
  160 DIM B(2),C$(3)
  170 INPUT PROMPT "a keresett valtozo neve: ":N$
  180 LET N$=UCASE$(N$)
  190 LET VARPTR=USR(M,0) N$
```

```
  2ØØ IF VARPTR=Ø THEN PRINT "ilyen valtozo nev nincs":
GOTO 22Ø
  21Ø PRINT "TAROLASI CIM:";VARPTR
  22Ø PRINT
  23Ø GOTO 17Ø
```

23. program

Egyes Basic verziókban a VARPTR-nak külön utasítása van (vö. 22. programmal).

8C 1. szegmens D167H
Ha program feldolgozás közben azonosítót talál, akkor azt megkeresi a változó táblában. A típusjelző 3. és 4. bitjét vizsgálva megállapítja, hogy érték újradefiniálás lehetséges-e. Ha igen, elkészíti az új bejegyzést a 88. rutin meghívásával.

8D 1. szegmens D117H
A változó tábla vége után 64dec üres helyet biztosít (a 238H rendszerváltozó értékét beállítja) és ha kell, a memóriát átszervezi.

8E 1. szegmens D153H
Aktuális azonosítójú bejegyzést keres a változó táblában. Ha megtalálja és a változónak már van értéke, akkor Carry Ø-val tér vissza.

8F 1. szegmens D1D9H
A DCB táblából törli azokat a bejegyzéseket, melyek Basic területre mutatnak, azaz csak az eleve definiált gépi kódú függvények címeit hagyja meg.

9Ø 1. szegmens C2ACH
Inicializálja a DCB táblát. Előbb nullázza a területet, majd a CDE4H területen kezdődő függvényleírók címeit betölti a változó táblában való keresés megvalósításához.

91 1. szegmens C2ACH
HL által kijelölt címről a Basic tárba (228H tárolt címre) tölti.

92 1. szegmens C28DH
A Basic tárba tölti az 1. szegmens CØ8CH területen kezdődő 43dec konstanst tartalmazó táblázatból a RST 1Ø–92H rutinját kijelölő utasítást követő számmal megnevezett elemet. Ez a RST 1Ø rutin tehát két byte hosszú!
A konstansokat mi magunk is kiolvashatjuk a 24. program segítségével.

```
1ØØ !    konstans tabla
11Ø OUT 177,1:CLEAR SCREEN:PRINT
12Ø LET Z=Ø
13Ø FOR L=16524 TO 16824 STEP 7
14Ø    LET K=PEEK(L+6):LET S$=" " !kitevo
15Ø    IF K>=128 THEN LET S$="-" ! elojel
```

```
160    PRINT USING "##   ":Z;
170    PRINT S$;:LET Z=Z+1
180    FOR J=5 TO Ø STEP-1
190      LET S=PEEK(L+J)
200      PRINT STR$(INT(S/16));
210      IF J=5 THEN PRINT "."; ! tizedespont
220      PRINT STR$(MOD(S,16));
230    NEXT
240    IF K=64 THEN GOTO 2'/Ø ! kitevo Ø ,nem irjuk ki
250    PRINT " E";
260    PRINT STR$((K BAND 127)-64);
270    PRINT
280 NEXT
```

24. program

A futás eredményét és a konstansok értelmezését a következő táblázat tartalmazza:

sor	ertek	ertelmezes
szam		
0	5.00000000000 *E-1	0.5
1	1.00000000000	1
2	9.99999999900 *E62	MAX
3	3.27680000000 *E4	32768
4	3.14159265359	PI
5	5.72957795131 *E1	180/PI
6	1.74532925199 *E-2	PI/180
7	1.57079632679	PI/2
8	1.04719755120	PI/3
9	5.23598775598 *E-1	PI/6
10	6.28318530718	2*PI
11	2.23607000000 *E-1	SQR(0.05)
12	8.94427000000 *E-1	SQR(0.8)
13	3.16227766017 *E-1	SQR(0.1)
14	-1.66666666088 *E-1	SIN
15	8.33333072056 *E-3	sorfejtes

```
16   -1.98408328231 *E-4      konstansai
17    2.75239710678 *E-6         -"-
18   -2.38683464060 *E-8         -"-
19   -1.44008344874              ATN
20   -7.20026848898 *E-1      sorfejtes
21    4.32025038919           konstansai
22    4.75222584599              -"-
23    2.67949192431 *E-1       tg(15)
24    1.73205080757            tg(60)
25    7.32050807569 *E-1
26   -2.91568143790             LOG(10)
27    3.16303491557           sorfejtes
28   -6.73581601478 *E-1      konstansai
29   -1.00704069542 *E1          -"-
30    1.69669814021 *E1          -"-
31   -8.19080045467              -"-
32    8.68588963807 *E-1       2/ln(10)
33    2.30258509299            ln(10)
34    3.32192809489            1/lg(2)
35    8.68588963807 *E-1       2/ln(10)
36    1.15100000000               EXP
37    2.92546497023 *E-4       sorfejtes
38    5.04464889506 *E2        konstansai
39    1.40082997563 *E1           -"-
40    3.32873646516 *E-2          -"-
41    1.00892977901 *E3           -"-
42    1.12094081097 *E2           -"-
```

6. lista A 24. program futásának eredménye

93 1. szegmens C29FH

Ez a rutin 3 byte hosszú. A 93H kódot követő két szám HL-be töltendő memóriacím két byte-ja. A rutin ezzel a címmel kijelölt helyről tölti a konstanst a Basic tárba, melynek aktuális címét a 228H rendszerváltozó jelöli ki.

94 1. szegmens C2A4H
Az EØBH címen kezdődő aritmetikai regiszterből a számot a tárba helyezi.

95 1. szegmens C2A9H
A E12H címen kezdődő aritmetikai regiszter tartalmát helyezi a tárba.

96 1. szegmens C2C6H
Három byte hosszú RST 1Ø rutin. A 96H rutin kijelölő kód utáni két byte azt a címet jelöli ki, ahová a tár legutoljára bejegyzett adatát át kell másolni.

97 1. szegmens C2CCH
A tár legfelső elemét a EØBH címen kezdődő aritmetikai regiszterbe tölti.

98 1. szegmens C2D1H
A tár tetején lévő elemet (összesen 6 byte-ot) áttölt az E12H címen kezdődő regiszterbe.

A 99H–AØH rutinok programfutás (ill. tokenizálás) közben szintaktikai elemzést végeznek és kiértékelik az aktuális kifejezést. A kiértékelés eredményét a tárba helyezi.

99 1. szegmens C2E4H
Nyitó és záró zárójelre vizsgál, a zárójelben lévő kifejezést kiértékeli.

9A 1. szegmens C2FØH
Nyitó zárójelbe vizsgál, és az azt követő kifejezést kiértékeli.

9B 1. szegmens C2EDH
Adat elválasztó vesszőre ellenőriz, és az azt követő kifejezést kiértékeli.

9C 1. szegmens C2F7H
Nyitó és záró zárójel között lévő string kifejezést értékel ki.

9D 1. szegmens C2FFH
Nyitó zárójelre vizsgál, és az azt követő string kifejezést értékeli ki.

9E 1. szegmens C2FCH
Adatsort elválasztó vesszőre ellenőriz, és az azt követő string kifejezést kiértékeli.

9F 1. szegmens C3Ø6H
Tömbváltozó nevet követő nyitó zárójelre ellenőriz és kiértékeli az indexet.

AØ 1. szegmens C2E7H
Vizsgálja, hogy van-e záró zárójel.

Az A1–B2 közti rutinok Basic kulcsszóval elérhető függvények végrehajtását biztosítják. Ezekre a végrehajtó RST 1Ø rutinokra a szintaktikai elemzés után tér át a rendszerprogram.

A1 1. szegmens C68DH
FREE függvény végrehajtó rutinja az aktuális program részére kijelölt memóriaterületen még rendelkezésre álló szabad byte-ok számát adja meg.

A2 1. szegmens C399H
ATN a tár tetején lévő számot, mint egy szög tangensét értelmezi. Ebből kiszámítja a szög értékét radiánban. Az eredményt a tárban adja.

A3 1. szegmens C368H
ASIN a tár tetején lévő számot, mint egy szög szinuszát értelmezi. Eredményként a tárba a szög értékét adja radiánban. A számításhoz felhasználja azt a trigonometriai összefüggést, hogy

$$\text{tg } x = \frac{\sin x}{\sqrt{1 - (\sin x)^2}}$$

és a számítást az A2 rutin segítségével végzi el.

A4 1. szegmens CB86H
A radiánokban adott szög szinuszát számítja ki a

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots\ldots - \frac{x^{11}}{11!}$$

közelítéssel.
A konstansok értékét a RST 1Ø–92 rutin segítségével olvassa be.

A5 1. szegmens C518H
A radiánban adott szög koszinuszát számolja ki a
$\cos x = \sin (x+\pi/2)$ összefüggés felhasználásával.

A6 1. szegmens CC9FH
A radiánokban adott szög tangensét határozza meg a

$$\text{tg } x = \frac{\sin (x)}{\sin (x+\pi/2)}$$ összefüggéssel

Érdemes ezt a végrehajtó rutint tüzetesebben elemezni, hogy a RST 1Ø rutinok hívás sorozatának logikáját megértsük (7. lista).

```
RST10 - A6
TAN(X) kiszamitasanak rutinja

CC9F    D7    RST 10 rutinok hivasa
CCA0    15    X megegyszer a verembe
CCA1    98    aritmetikai reg.be X
CCA2    A4    kiszamitja SIN(X)-et
CCA3    95    regiszterbol verembe X
CCA4    92    verembe a 7.konstanst
CCA5    07    7.konstans = PI/2
CCA6    0C    a ket utolso elem ossze-
              adasa (SIN(X)+PI/2)
```

```
CCA7      A4    SIN(X)
CCA8      0F    ket utolso elem hanyadosa
                SIN(X)/(SIN(X)+PI/2)
CCA9      00    RST 10 rutinhivas vege
```

7. lista TAN(X)

A7 1. szegmens C5FØH
A tár tetején lévő számot mint hatványkitevőt értelmezi és kiszámítja az e^x értékét.

A8 1. szegmens CA8ØH
A 2Ø1H rendszerváltozó értékétől függően elvégzi a tár tetején lévő radiánban értelmezett szög fokokra való átszámítását.

A9 1. szegmens C586H
A 2Ø1H rendszerváltozó értékétől függően elvégzi a tár tetején lévő fokban értelmezett szögérték radiánokra való átszámítását.

AA 1. szegmens C8A7H
A tár tetején lévő szám tízes alapú logaritmusát számítja ki.

AB 1. szegmens C98DH
Szintaktikai elemzés nélkül kiszámítja a MOD (X, Y) értékét. X értéke a tárban IY+9 címen van, az Y értéke a tár legutoljára bejegyzett értéke IY címen van.

AC 1. szegmens CC52H
A Basic verem tetején lévő szám négyzetgyökét kiszámítja a Newton-féle itteráció segítségével. A 25. program ennek Basic változatát mutatja be.

```
100 !RST 10-AC BASIC megvalositasa
110 INPUT X
120 IF X<0 THEN PRINT "negativ argumentum":STOP
130 LET A=X:LET K=0
140 IF X>1 THEN
150   LET K=K+1 !normal alakra
160   LET X=X/10
170   GOTO 140
180 END IF
190 LET Y=X-X*SQR(.8)+.223607 !kezdo ertek
200 FOR I=0 TO 4
210   LET Y=(X/Y-Y)/2+Y
220   GOSUB 270
230 NEXT
240 GOSUB 270
```

```
250 PRINT "kontroll:";SQR(A)
260 END
270 LET K1=K:LET Y1=Y
280 IF K=0 THEN GOTO 350
290 IF INT(K/2)=K/2 THEN
300   LET K=K/2
310 ELSE
320   LET Y=Y*.316227766
330   LET K=(K+1)/2
340 END IF
350 PRINT ," " Y*10^K
360 LET K=K1:LET Y=Y1
370 RETURN
```

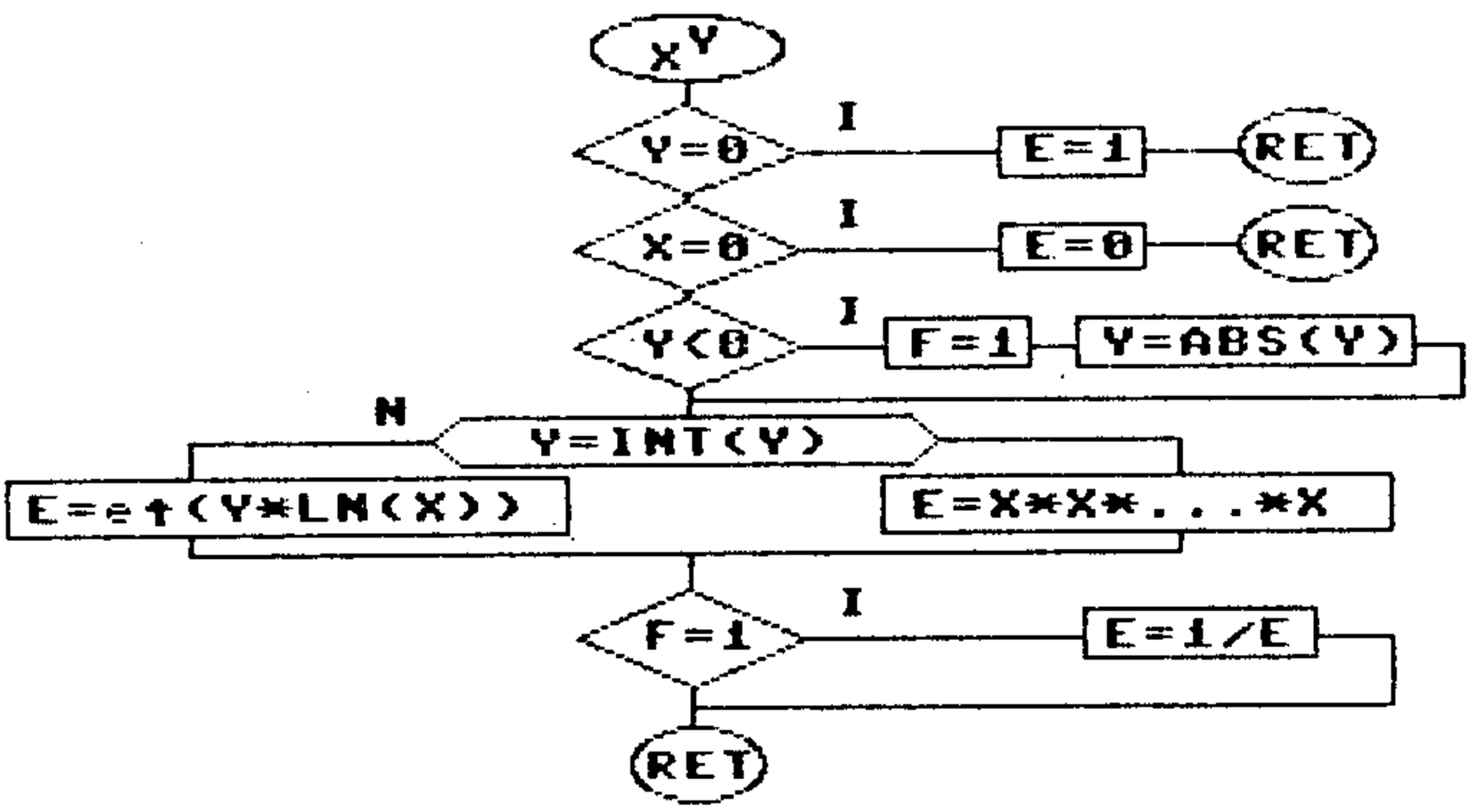
25. program

AD 1. szegmens CA8EH
A tár tetején lévő elem reciprokát számítja ki.

AE 1. szegmens C778H
A lebegőpontos számnál nem nagyobb egész számot állítja elő. A Basic INT függvény szintaktikai elemzés nélküli végrehajtó rutinja.

AF 1. szegmens CB17H
Az akkumulátorban megadott decimális jegy pontosságra kerekíti a Basic verem tetején lévő lebegőpontos számot. A Basic ROUND függvény szintaktikai elemzés nélküli végrehajtó rutinja.

BØ 1. szegmens C7BØH
A tár utoljára bejegyzett elemét hatványkitevőnek értelmezi az IY+9 címen lévő következő elemet alapnak tekinti. Ezekkel az adatokkal kiszámítja az alap n. hatványát. A végrehajtó rutin kristálytiszta logiká-



2. ábra

val dolgozik (l. blokkdiagram), a megvalósítás Basic programmal is modellezhető. Ezt szemlélteti a 26. program.

```
100 ! RST 10-B0 Basic megvalositasa
110 INPUT PROMPT "alap  :":X
120 INPUT PROMPT "kitevo:":Y
130 PRINT "eredmeny:";
140 IF Y<>0 THEN 170
150 LET W=1 !minden szam 0.hatvanya 1
160 GOTO 300
 170 IF X<>0 THEN 200
 180 LET W=0 !0 minden hatvanya 0
 190 GOTO 300
 200 LET F=SGN(Y):LET K=ABS(Y) !kitevo elojele es
ABS erteke
 210 IF Y<>INT(Y) THEN 280
 220 LET W=1
 230 FOR I=1 TO K !  egesz kitevo eseten ismetelt
szorzas
 240   LET W=W*X
 250 NEXT
 260 IF F=-1 THEN LET W=1/W !negativ kitevo miatt
reciprok ertek
 270 GOTO 300
 280 LET E=2.718281828 ! ln alapszama
 290 LET W=E^(Y*LOG(X)) ! tortkitevo eseten
alkalmazando osszefugges
 300 PRINT W
 310 PRINT "kontroll:" X^Y
```

26. program

Látható az alkalmazott összefüggés a tört kitevőjű hatvány esetében. Minthogy az alap logaritmusával számol, törtkitevőjű hatvány esetében az alap nem lehet negatív.

B1 1. szegmens C82ØH
A tár legfelső elemének tizedespont utáni részét törli. Ez a rutin a Basic IP függvény végrehajtását biztosítja.

B2 1. szegmens CBF9H
A Basic SGN függvény végrehajtó rutinja.

A tárba bejegyzett legfelső elem előjelétől, ill. értékétől függően —1, Ø vagy 1 értéket ad eredményül.

B3 1. szegmens C1D9H
A B5 rutin segítségével megkeresi, ill. bejegyzi az E54H területen kezdődő RAM térképbe a C regiszterben adott szegmensszámot.

B4 1. szegmens C1B9H
Első szabad RAM szegmenset C-ben hozott program számára lefoglalja. Elvégzi a 2ØDH 2ØBH rendszerváltozók értékének átírását.

B5 1. szegmens C1EFH
C regiszterben hozott szegmensszám (E54H címen kezdődő) RAM táblabeli bejegyzettségét vizsgálja.
Ha már be van jegyezve Z flag=1

B6 1. szegmens C1F9H
C regiszterben hozott programszámot az E54H címen kezdődő területen a 2ØCH rendszerváltozó értéke által meghatározott hosszon keres. Ha bejegyzést talált Carry=Ø.

B7 1. szegmens D1F3H
Funkcióbillentyű hatásának átdefiniálását végzi az EXOS ØB rutin meghívásával, valamint a SET végrehajtása.

B8 1. szegmens D2ACH
A gépi feltételek azonosítása a D2DD címen kezdődő táblázat segítségével ASK, SET és TOGGLE végrehajtó rutinjához. Ha azonosítható az alábbi táblázat valamelyik eleme, akkor A, B, C értékét táblázatból kiveszi és Carry=1-gyel tér vissza DE az azonosított feltétel táblabeli címét mutatja a 27. program.

```
 1ØØ ! ASK TOGGLE feltetelek RST1Ø-B8
 11Ø PRINT "feltetel       B    C    A"
 12Ø FOR I=53981 TO 54313 STEP 5 ! azonosito tabla
 13Ø   LET A=SPEEK(1,I)*256+SPEEK(1,I+1) !
azonositando szoveg
 14Ø   LET H=SPEEK(1,A):LET A$="" ! feltetel szoveg
hossza
 15Ø   FOR K=1 TO H !feltetel karakterei
 16Ø     LET A$=A$&CHR$(SPEEK(1,A+K))
 17Ø   NEXT
 18Ø   LET C=PEEK(I+3):GOSUB 27Ø
 19Ø   LET C$=H$
 2ØØ   LET C=PEEK(I+3):GOSUB 27Ø
 21Ø   LET D$=H$
 22Ø   LET C=PEEK(I+4):GOSUB 27Ø
 23Ø   LET E$=H$
```

```
 240   PRINT USING "<##########    ###   ###   ###":A$,C$
,D$,E$
 250 NEXT
 260 END
 270 !    dec-->hex$ atalakito
 280 LET E=INT(C/16):LET M=MOD(C,16)
 290 LET E$=CHR$(E+48-7*(E>9))
 300 LET M$=CHR$(M+48)
 310 LET H$=E$&M$
 320 RETURN
```


A táblázatot és a kiolvasott értékeket a következőkben tekinthetjük át:

	feltetel	B	C	A
1	DEFAULT	80	80	00
2	CHANNEL	84	84	00
3	TIMER	85	85	00
4	KEY	80	80	00
5	CLICK	87	87	00
6	RATE	8A	8A	00
7	DELAY	8B	8B	00
8	INTERRUPT	80	80	00
9	STOP	88	88	00
10	KEY	89	89	00
11	NET	93	93	00
12	CODE	82	82	00
13	TAPE	80	80	00
14	SOUND	8C	8C	00
15	PROTECT	A2	A2	00
16	LEVEL	A3	A3	00
17	FAST	80	80	00
18	SAVE	A1	A1	00
19	SOUND	80	80	00
20	STOP	8D	8D	00

21	BUFFER	8F	8F	00
22	SPEAKER	8E	8E	00
23	SERIAL	80	80	00
24	BAUD	90	90	00
25	FORMAT	91	91	00
26	VIDEO	80	80	00
27	MODE	96	96	00
28	COLOR	97	97	00
29	COLOUR	97	97	00
30	Y	99	99	00
31	X	98	98	00
32	BORDER	9B	9B	00
33	BIAS	9C	9C	00
34	STATUS	9A	9A	00
35	EDITOR	80	80	00
36	VIDEO	9D	9D	00
37	KEY	9E	9E	00
38	BUFFER	9F	9F	00
39	REM1	A4	A4	00
40	REM2	A5	A5	00
41	NET	80	80	00
42	CHANNEL	94	94	00
43	NUMBER	92	92	00
44	MACHINE	95	95	00
45	PRIORITY	A6	A6	00
46	ATTRIBUTES	61	61	65
47	PALETTE	43	43	65
48	COLOR	63	63	65
49	COLOUR	63	63	65
50	PAPER	50	50	65
51	INK	49	49	65
52	CHARACTER	4B	4B	66
53	CURSOR	00	00	00
54	CHARACTER	2E	2E	66

55	COLOR	4D	4D	66
56	COLOUR	4D	4D	66
57	SCROLL	00	00	00
58	ON	53	53	66
59	OFF	73	73	66
60	UP	55	55	66
61	DOWN	44	44	66
62	BEAM	00	00	00
63	ON	53	53	65
64	OFF	73	73	65
65	LINE	00	00	00
66	STYLE	2E	2E	65
67	MODE	4D	4D	65

8. lista

B9 1. szegmens D43ØH
DELETE és LIST kulcsszavak után alkalmazható szöveges FIRST, LAST sorszám és sorszámtartomány meghatározás. HL-be tölti az első, DE-be az utolsó programsor számát.

BA 1. szegmens D4A8H
Sorszámmal adott programsorokat Carry' értékétől függően listázza, vagy törli. Listázás végrehajtását a Carry'= 1 esetében.

BB 1. szegmens D546H
A Basic AUTO parancs kiértékelése, ill. az alapértelmezésű sorszám és lépésköz meghatározása.

BC 1. szegmens D5CFH
A Basic RENUMBER parancs kiértékelése és végrehajtása.

BD 1. szegmens DA48H
PRINT utasítás után, formatált kiírásra utaló utasításkódok kiértékelése. A kiírási formátumot A értéke határozza meg
USING=3, AT=2, #=1, ha egyik sem, akkor akkumulátor tartalma Ø

BE 1. szegmens DA9DH
PRINT végrehajtó rutin.
A PRINT formátus soron belüli alakítására használható segéd információk kiértékelése TAB, vessző, pontosvessző.

BF 1. szegmens DF4BH, ha angol, vagy
4. szegmens EDEØH, ha német üzemmód
HL-ben hozott hibasorszám alapján a hibaüzenetet a ROM megfelelő helyéről a 446H címen kezdődő hibaüzenet pufferbe tölti. A hibaüzenetek sokfélesége ellenére nem minden esetben tud részletes hibaüze-

netet kiírni, ilyenkor sorszámtól függően EXOS, vagy általános típusra utaló üzenetet ad.

CØ 1. szegmens D746H
A WAIT Basic utasítás végrehajtó rutinja.

C1 1. szegmens D773H
A blokkok egymásba ágyazását, annak szabályszerűségét vizsgálja. Az egymásba ágyazás mélységét, a skatulyázást, mely legfeljebb 127 szintű lehet, a programsor tárolás során a sorszám utáni byte-on írja be. Amennyiben a 7. bit is használatba kerülne e vizsgálat során, akkor hibajelzést ad. Egyébként a skatulyázás jelző 7. bitjét futás közben jelzőbitként használja a rendszer.

C2 1. szegmens D7B5H
Az EXOS 1F rutin felhasználásával az időpont beállítását végzi el. A TIME Basic utasítás végrehajtó rutinja.

C3 1. szegmens D7C5H
Az EXOS 21 rutin segítségével beállítja a dátumot. A DATE Basic utasítás végrehajtó rutinja.

C4 1. szegmens CD2FH
A 228H rendszerváltozó által meghatározott kezdőcímű puffer tartalmát B regiszter tartalmától függően kisbetűsíti (ha B=4ØH), vagy nagybetűsíti (ha B=6ØH).
Gyakorlatilag az UCASE$ és LCASE$ Basic függvények végrehajtó rutinja.

C5 1. szegmens D812H
Visszaállítja a funkcióbillentyűk alapértelmezését. Az alapértelmezés szerinti funkciók megnevezésének táblázata az 1. szegmens D842H területen kezdődik.

A RST rutinok közül még egyet említünk itt meg. A RST 38H a megszakításkezelő rutinja. Programból kiiktatható ez a rutin, így valamelyest gyorsabb programfutás valósul meg. Ennek az az ára, hogy a géphez billentyűzetről a továbbiakban nem lehet hozzáférni. Aki tehát a RST 38H rutin belépési címét átírja POKE 56,2Ø1 segítségével, az számítson arra, hogy ha még a billentyűzetről bármit a program, a gép tudomására akar hozni, akkor előbb még a programból az eredeti tartalmat vissza kell állítania, vagy RESET-hez kell folyamodnia. Ennek a rutinnak a kiiktatása azzal is együtt jár, hogy a belső óra leáll.

A számítógép rendszerprogramjai, ill. az eddig ismertetett makrói tulajdonképpen egytől egyig a Basic nyelven írt programok értelmezését és végrehajtását teszik lehetővé. Amennyiben nem akarunk megelégedni a Basic utasítások és függvények Felhasználói kézikönyvben leírt lehetőségeivel, vagy ha egy-egy utasítást, vagy függvényt más célra akarunk felhasználni, akkor az eddig tárgyaltakat hasznosítva egy sor új lehetőséget kaphatunk. Természetesen a Basic mindig korlátokat jelent mind sebesség, mind pedig extra kívánságok teljesítése területén. Vannak feladatok, melyek csak gépi kódú program segítségével oldhatók meg.

A továbbiakban a számítógép utasításkészletét – többé kevésbé – ABC sorrendbe szedve tekintjük át, a számítógéphez adott Felhasználói kézikönyv felépítéséhez hasonlóan. A könyv ezen részében nem – vagy csak nagyon kivonatosan ismertetjük a kulcsszavak hatását, ill. az alkalmazásuk során követendő szintaktikai szabályokat. Célunk sokkal inkább az, hogy a nem kézenfekvő felhasználásra, vagy egy feladat, látvány, vagy sebesség szempontjából célszerűbb, vagy csak más megoldására példát adjunk.

2. PARANCSOK ÉS UTASÍTÁSOK

Sorszám

A program végrehajtása során az interpreter szépen sorba, egymás után hajtja végre az utasításokat. Amennyiben a programban nincs sorszám, hivatkozás, azaz nincs feltételhez kötött vezérlésátadás (GOTO), vagy nem hívunk meg szubrutint (GOSUB), akkor az interpreter végrehajtás közben nem is a sor elején lévő sorszámot, hanem az adott sor, ill. soron belül végrehajtott karakter, utasítás, adat memóriabeli címét használja fel a végrehajtáshoz. Így azután elképzelhető a 28. program, amit hibátlanul végre is hajt a számítógép.

```
88 ! azonos sorszamok,  szoveg kigyozik
88 FOR I=Ø TO 2*PI STEP PI/2Ø
88   PRINT TAB(14+1Ø*SIN(I)) "ENTERPRISE"
88 NEXT
88 RUN
```

28. program

Igaz, hogy nem lehet ilyen formában beírni, hiszen az azonos sorszámmal beírt programsorok közül csak az utoljára beírt érvényes, csak azt tárolja a számítógép. Amennyiben beírjuk a 29. programot és futtatjuk azt, akkor eredményként az előző programhoz hasonló listájú, működő programot kapunk.

```
  Ø FOR I=4827 TO 488Ø
  Ø   PRINT I;PEEK(I)
  Ø NEXT
1ØØ FOR I=4827 TO 488Ø
11Ø   PRINT I;PEEK(I)
12Ø NEXT
14Ø ! elso sor eleje Ø programkent 4827 cimen van
15Ø POKE 4828,Ø
16Ø ! a masodik sor eleje 4827+sorhossz
17Ø POKE 4828+PEEK(4827),Ø
```

```
  180 ! a harmadik sor eleje 4827+peek(4827)+peek(4827+
peek(4827)) ez most 4865
  190 POKE 4866,0
  200 ! a cimek a program elejenek futtatasaval
ellenorizhetok
```

29. program

Minthogy a memóriában lévő programba utólag POKE utasítással belenyúlunk, fontos, hogy a programba utólag beszúrt érték valóban oda kerüljön, ahová szántuk. Ezért első nekifutásra pontosan így írjuk be a számítógépbe a felülírni kívánt programot. Később majd keresünk olyan megoldást, amivel nem nekünk kell kiszámolni a felülírni kívánt memóriacímeket.

ALLOCATE

Már láttuk a rendszerváltozók esetében, hogy az IS–BASIC a gépi kódú program részére a memóriában az ALLOCATE utáni kifejezés értékének megfelelő byte-ot foglal le. Minthogy ez a helyfoglalás a Basic terület mozgatásával jár együtt az ALLOCATE mellékhatása a változók törlése is. Ezért az ALLOCATE Ø, a program tetszőleges helyén az Enterprise Basic-ből hiányzó értékadás törlés végrehajtására is alkalmas. Adódhat programfutás közben olyan helyzet, hogy tömböket akarunk újra definiálni, ilyenkor végrehajthatunk egy ALLOCATE Ø utasítást. Győződjünk meg ennek hatásáról a 30. program segítségével.

```
100 ! ALLOCATE mint valtozok torlese
110 DIM C(10)
120 LET A=1:LET B=2 ! valtozok inicializalasa
130 PRINT A,B
140 ALLOCATE 0 !       CLEAR valtozo helyett
150 DIM C(10)
160 PRINT "c nevu tomb masodszor deklaralva"
170 PRINT A !ez viszont nem megy, mar elfelejtette!
```

30. program

Az ALLOCATE hatása azonban többnyire csak a program futásának idejére érvényes. Legyünk óvatosak gépi kódú programjaink közvetlen meghívásával, mert ugyan a 31. program futása közben a Basic program a 10 byte-tal feljebb helyezkedik el a memóriában, mint az alapértelmezés esetében, de a program

megszakítása, vagy befejezése után a program „visszacsúszik" az alapértelmezésű helyre. Így azután hiába próbálnánk a vélt helyről indítani a gépi kódú részt, az bizony a gép lemerevedéséhez vezethet.

```
 1Ø ! ALLOCATE hatasa BASIC elejere
 12 PRINT "Basic eleje:";PEEK(544)+256*PEEK(545)
 14 ALLOCATE 1Ø
 15 PRINT "ALLOCATE 1Ø utan"
 16 PRINT "Basic eleje:";PEEK(544)+256*PEEK(545)
 2Ø CODE A=HEX$("1,2,3,4,5,6,7,8,9") !proba kedveert
1ØØ FOR I=4827 TO 4844 !memoria elejenek vizsgalata
11Ø   PRINT I;PEEK(I) !cim es tartalom
12Ø NEXT
```

31. program

Azt, hogy ez most így van, arról úgy győződhetünk meg a legegyszerűbben, hogy a 32. programot betöltjük 1. programként, akkor amikor a 31. program Ø.-ként él a gépben.

```
1ØØ ! RUN-nal indits
11Ø ! 1-en futtatva ha a 31.pr a Ø.-on van
12Ø FOR I=4827 TO 4844 !Ø.program tarolasara szolgalo
terulet vizsgalata
 13Ø   PRINT I;PEEK(I) !cim es tartalom
 14Ø NEXT
```

32. program

Futtassuk le mégegyszer a 31. programot, ezután váltsunk programot, és RUN-nal indítsuk a 32. programot. A képernyőn megjelenő értékek mutatják, hogy a Ø. programként beírt sorok „nyugalomban" visszacsúsznak a Basic terület elejére.

BIN

A függvény argumentumában megadott egyesekből és nullákból álló karaktersorozat decimális értékét adja. A függvény végrehajtó rutinja az 1. szegmens C42CH címen kezdődik. Van némi szépséghibája ennek a rutinnak. Elfogad maximum 12 karaktert, de kiértékelni ebből csak tízet képes. Így azután a BIN

(111111111111) értékére a helyes 4Ø95dec helyett, 4Ø92 decimális értéket jelez ki a számítógép. Amennyiben 10 helyi értékűnél hosszabb bináris számot kell átalakítani decimálissá, akkor nekünk kell saját programot írni, például olyant, mint a 33. program.

```
100 !bin ----> dec
110 INPUT PROMPT "binaris ertek:":B$
120 LET D=Ø ! egyelore hibajelzesre segedvaltozo
130 LET H=LEN(B$) ! a binaris szam hossza
140 FOR I=1 TO H !valoban csak Ø es 1 karaktereket
tartalmaz a B$ ?
150   LET V$=B$(I:I) ! szeleteljuk le karakterenkent
160   IF V$<"Ø" OR V$>"1" THEN LET D=1 !ha eltero
karakter akkor hibajelzo beallitasa egyre
170 NEXT
180 IF D=Ø THEN GOTO 210 ! ha visszaalakithato, akkor
D a decimalis ertek kezdoerteke
190 PRINT "hibas karakter"
200 GOTO 110
210 FOR I=H TO 1 STEP-1
220   LET D=D+VAL(B$(I:I))*2^(H-I) ! a decimalis
ertek osszeallitasa a helyi erteknek megfeleloen ketto
      adott hatvanyaval valo szorzassal
230 NEXT
240 PRINT B$;"bin=";D;"dec"
```

33. program

Hasonlóan szép feladat a tízes számrendszerből binárisra átalakítani egy számot. Ezt elegáns rövid programmal elvégezhetjük, ha felhasználjuk a „bináris és" függvényt (34. program).

```
100 ! dec ---> bin
110 INPUT PROMPT "decimalis ertek:":D$
120 LET H=Ø ! hibajelzesre szolgalo valtozo
130 FOR V=1 TO LEN(D$)
140   LET V$=D$(V:V) ! karakter vizsgalatanak elo-
keszitese
150   IF V$<"Ø" OR V$>"9" THEN LET H=1 ! szamjegyre
vizsgal
160 NEXT
```

```
 17Ø IF H=1 THEN PRINT "hibas adat":RUN  ! nem szam-
jegyet tartalmazo adat eseten ujraindul
 18Ø LET D=VAL(D$)
 19Ø IF D>2Ø97151 THEN PRINT "a dec.szam tul nagy"
 2ØØ PRINT D;"dec=";
 21Ø FOR I=2Ø TO Ø STEP-1
 22Ø   PRINT STR$((NOT(D BAND 2^I))+1); ! BAND fel-
hasznalasaval binaris jegyeket hataroz meg
 23Ø NEXT
 24Ø PRINT " bin"
```

34. program



3. ábra

Érdemes egyszer a kettes számrendszert rajzolásra felhasználni (3. ábra).

Szinte filozófiai mélységek felismeréséhez vezet, ha a képernyőn egy számegyenes mentén haladva, arra merőlegesen, a szám kettes számrendszerbeli alakjának megfelelően pontokat helyezünk el a 35. program segítségével.

```
100 ! kettes szamrendszer minta rajza
 110 GRAPHICS 16
 120 SET INK WHITE
 130 FOR I=Ø TO 159
 140   LET V=I
 150   FOR B=7 TO Ø STEP-1 ! dec ertek binarisra
alakitasa
 160     LET E=INT(V/2^B) ! E Ø vagy 1 lehet
 170     LET V=MOD(V,2^B) ! tovabbi vizsgalathoz
erteket alakit
 180     IF E=Ø THEN 240 !Ø ertekre nem rajzol pontot
 190     ! 1 erteknek megfelelo helyre pontot rajzol
 200     PLOT I*8,344+B*8
 210     PLOT I*8,34Ø+B*8
 220     PLOT I*8,34Ø-B*8
 230     PLOT I*8,336-B*8
 240   NEXT
 250 NEXT
```

35. program

CALL (rekurzió)

A strukturált programozást az Enterprise a személyi számítógépek közül a legmagasabb fokon támogatja. Egy blokkra nevével hivatkozhatunk meghívásnál, listázásnál, törlésnél, átsorszámozásnál. A blokkok nagyfokú önállósága mellett azok szervesen épülnek a programba. Szabályos szerkezetüket az interpreter ellenőrzi.

A programozási gyakorlatban – ebben az összeállításban is – lépten-nyomon felhasználjuk a CALL utasítást egy-egy függvény (blokk) hívására, de azt, hogy egy függvény önmagát hívja, arra nem szoktunk gondolni. Az önmagát hívó szubrutinokkal, függvényekkel óvatosan kell bánni!

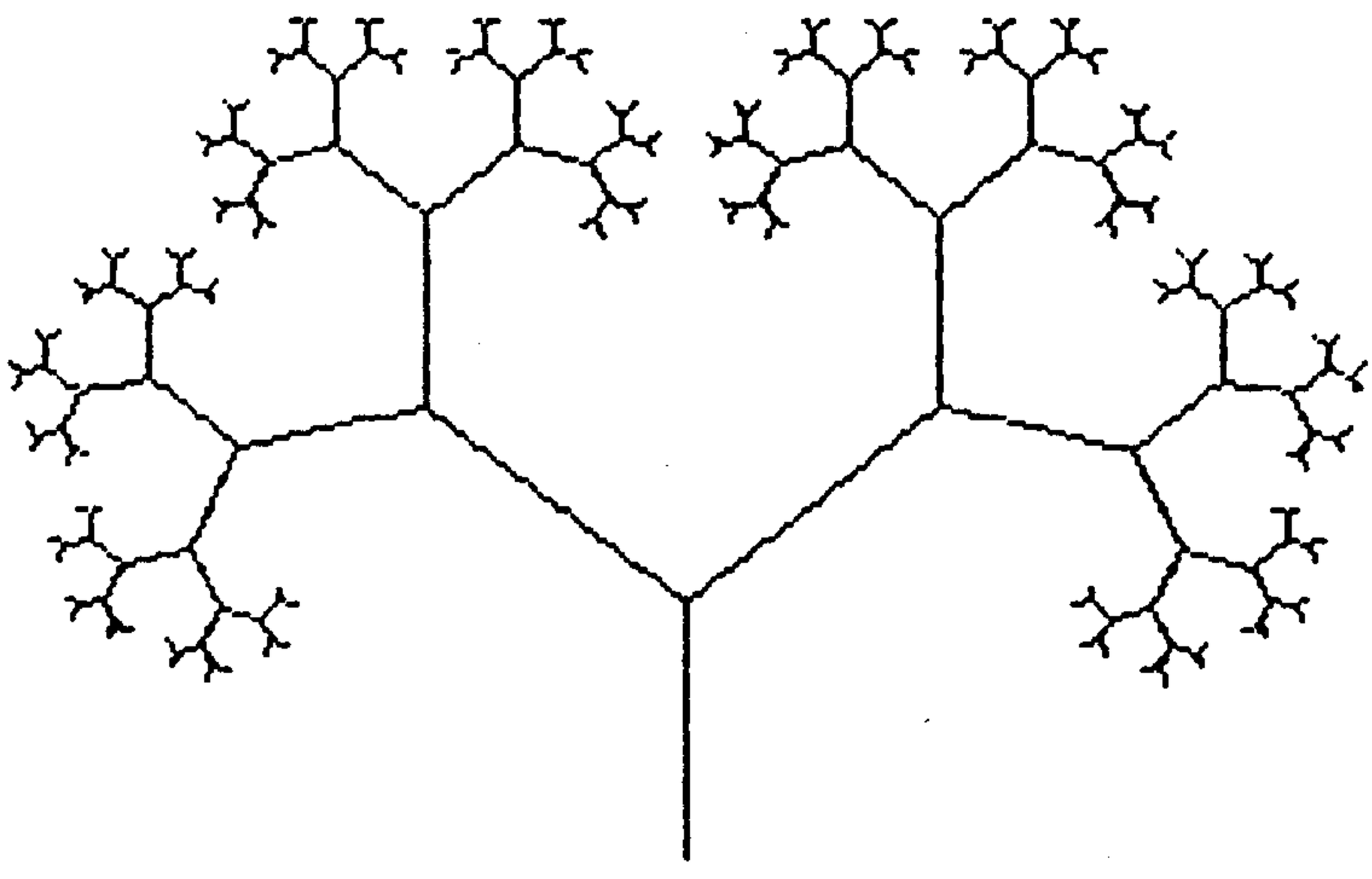
Elvetemült programozó (?) pillanatok alatt kiakaszthatja a számítgépet pl. a

```
1Ø DEF Y (x)=2×Y (x)
2Ø PRINT Y (1)
```

programmal. Csak a RESET segít, de az összes létező és szerencsére nem ismert hibaüzenet a képernyőre kerülhet (pl. INTERNAL CHEKSUM ERROR). De miért ne próbálkozhatnánk az önmagát hívó programkészletekkel. Azokat a programozási fogásokat, melyek során valamely eljárás (függvény, blokk) önmagát hívja, rekurziónak nevezik.

Az ismételt meghívás során egyre „mélyebbre" kerülünk a programban, de az eljárás végén, a visszatéréskor, az adott mélységből a programfutás még mindig az eljáráson belül, de eggyel kisebb mélységben folytatódik.

A rekurzív feladatmegoldás során a bonyolult feladatot a legegyszerűbb alapállapot megkeresésével és annak megoldásával kezdjük. Innen a probléma kezdeti megfogalmazásából és megoldásából azután megoldhatók az összetett, bonyolult, de az „őseredetre" visszavezethető problémák. A rekurzióval megoldható feladatok iskolapéldái közül vegyünk sorra néhányat: A 4. ábra a bináris fát



4. ábra

mutatja. A fa ágai a program szerint egy csomópontból jobbra-balra 45°-ra indulnak és minden ág végén újabb elágazás következik az előző szabály szerint. Ez az elágaztatás a végtelenségig folytatható lenne, valahol gátat kell szabni a rajzolásnak (4. ábra).

A program valahol az élet rendjének kissé ellentmond, hiszen a fa valahonnan egy törzsből (szimmetrikusan) fejlődik, ágazik szét. A mi rekurzív programunk elindul a törzsből és mindig a balra ágazó útvonalon gyorsan megkeresi a fa egyik ágának hegyét és onnan visszafelé kiépíti a törzshöz visszavezető ágakat. Ezt a rekurzív megoldást mi magunk is kipróbálhatjuk a 36. program futtatásával.

```
100 !rekurziv binaris fa
110 GRAPHICS 4
120 OPTION ANGLE DEGREES
130 SET INK WHITE
140 PLOT ANGLE 90
150 LET X=640:LET Y=0
160 LET L=300 ! a faag kezdeti hossza
170 LET A=.6 !rovidulesi tenyezo az elagazasok utani
vonalhoz
180 PLOT X,Y; !kezdopont kijelolese
190 CALL FA(L)
200 END
210 DEF FA(L)
220   IF L<10 THEN EXIT DEF !kilep az adott meghivasi
melysegbol ha a fa aga rovidebb mint a hatarertek
230   PLOT FORWARD L;
240   PLOT LEFT 45;
250   CALL FA(L*A) !a rovidulesi tenyezonek megfelelo
hosszal meghivja onmagat
260   PLOT RIGHT 90;
270   CALL FA(L*A)
280   PLOT LEFT 45;
290   PLOT BACK L !visszaall az elagazasra
300 END DEF
```

36. program

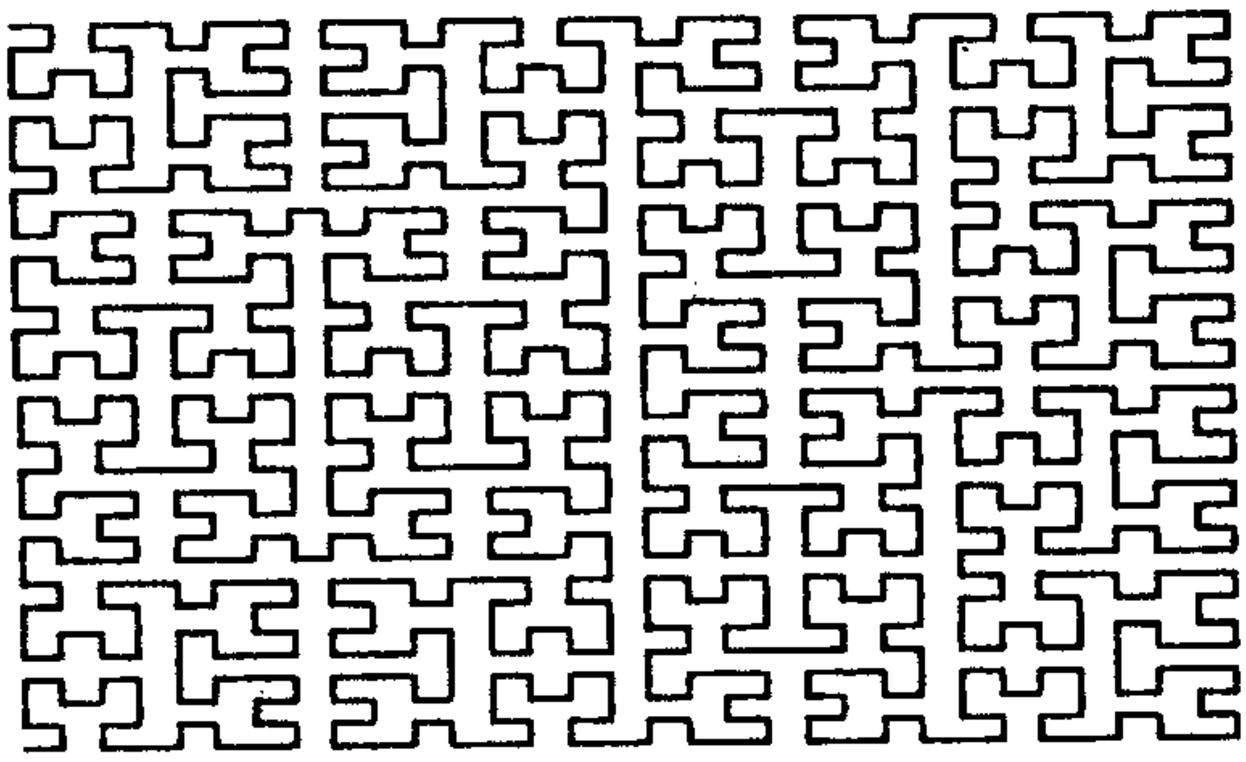
Érdemes összehasonlítani a rekurzió nélküli, bináris fát felrajzoló programot (37. program) az előzővel.

```
100 REM binaris fa
110 GRAPHICS
120 SET INK WHITE
130 OPTION ANGLE DEGREES
140 LET N=7:LET H=400
150 LET S=60 !elagazas szoge
160 LET F=.6 !rovidulesi aranyszam
170 LET N1=2^(N-1)
180 !csomopontok koordinatainak, az indulo agak
hajlasszogenek tarolasahoz tomb kijelolese
190 DIM X(N,N1),Y(N,N1),W(N,N1)
200 LET X(0,0)=600:LET Y(0,0)=0
210 LET X(1,1)=600:LET Y(1,1)=300
220 LET W(1,1)=90:LET K=1
230 PLOT X(0,0),Y(0,0);X(1,1),Y(1,1) ! az elso ag
kirajzolasa
240 FOR L=2 TO N ! az egyes szintek kivalasztasa
250   LET H=H*F:LET K=K*2 ! ezen a szinten ilyen
hosszu es ennyi agat kell rajzolni
260   FOR J=1 TO K STEP 2 !mert az agak fele jobbra,
fele balra indul
270     LET KK=INT(J/2)+1
280     LET W(L,J)=W(L-1,KK)+S:LET Z=W(L,J) ! balra
elagazas
290     LET X(L,J)=X(L-1,KK)+H*COS(Z) !az ag vegpont-
janak x koordinataja
300     LET Y(L,J)=Y(L-1,KK)+H*SIN(Z) !az ag vegpont-
janak y koordinataja
310     PLOT X(L-1,KK),Y(L-1,KK);X(L,J),Y(L,J) !
a balra elagazo ag rajza
320     LET W(L,J+1)=W(L-1,KK)-S:LET Z=W(L,J+1) !
jobbra elagazas
330     LET X(L,J+1)=X(L-1,KK)+H*COS(Z)
340     LET Y(L,J+1)=Y(L-1,KK)+H*SIN(Z)
350     PLOT X(L-1,KK),Y(L-1,KK);X(L,J+1),Y(L,J+1) !
a jobbra elagazo vonal rajza
360   NEXT
370 NEXT
```

37. program

Ennél a programnál rendre kiszámítjuk a faágak kezdő- és végpontjainak koordinátáit, a számítás eredményeit tömbváltozóként tároljuk, majd a kiszámított koordináták közé megrajzoljuk az egyenes szakaszokat. A program szemmel láthatóan is hosszabb, de ha a tömbök miatt lefoglalt memóriaigényt is figyelembe vesszük, akkor a rekurzív program előnye, memóriatakarékossága vitathatatlan.

David Hilbert leírt egy módszert, mellyel egy véges négyzet alakú síkfelület végtelen hosszúságú törtvonallal kitölthető. Egy ilyen görbét mutat az 5. ábra.



5. ábra

A Hilbert-görbéket vizsgálva a következőket deríthetjük ki: A 2. rendű görbe 4 db 1. rendű görbéből áll, amelyeket 3 „híd" köt össze. Hasonlóan a 3. rendű görbe 4 db 2. rendű görbéből áll, 3 összekötő híddal stb. Sajnos az N. rendű görbét felépítő 4 db N–1. rendű görbe körüljárási iránya nem egyezik meg, ezt különböző paritásnak nevezik. A teknős grafika számára a következő algoritmust írhatjuk elő:

1. Fordulj 90°-ot balra
2. Rajzold meg az N–1. rendű görbét *S* oldalhosszúsággal és 1 paritással
3. Menj *S* lépést előre (1. híd)
4. Fordulj 90°-ot jobbra
5. Rajzold meg az N–1. rendű görbét *S* oldalhosszúsággal és —1 paritással
6. Menj előre S lépést (2. híd)
7. Rajzold meg az N–1. rendű görbét *S* oldalhosszúsággal és —1 paritással
8. Fordulj 90°-ot jobbra
9. Menj előre S lépést (3. híd)
10. Rajzold meg az N–1. rendű görbét *S* oldalhosszúsággal és 1 paritással
11. Fordulj 90°-ot balra

Az algoritmus alapján megírhatjuk a 38. programot, mely a rekurzió elemeit mutatja, de legalább áttekinthető.

```
100 REM Hilbert gorbe
110 LET N=6:LET H=640:LET X0=960:LET Y0=320
120 FOR I=1 TO N
130    LET H=H/2
140    LET X0=X0+INT(H/2):LET Y0=Y0+INT(H/2)
```

```
150    LET X=X0-320:LET Y=Y0 ! indulasi feltetelek
160    GRAPHICS
170    SET INK WHITE
180    PLOT X,Y;
190    CALL A(I)
200    IF INKEY$="" THEN GOTO 200 !a kovetkezo szintu
Hilbert gorbe megrajzolasa billentyunyomasra indul
210 NEXT
220 END
230 DEF A(I)
240    IF I>0 THEN !tovabbi hivasok a feltetel szerint
250       CALL D(I-1):LET X=X-H:PLOT X,Y; ! jobbra
260       CALL A(I-1):LET Y=Y-H:PLOT X,Y; ! lefele
270       CALL A(I-1):LET X=X+H:PLOT X,Y; ! balra
280       CALL B(I-1)
290    END IF
300 END DEF
310 DEF B(I)
320    IF I>0 THEN
330       CALL C(I-1):LET Y=Y+H:PLOT X,Y; ! felfele
340       CALL B(I-1):LET X=X+H:PLOT X,Y;
350       CALL B(I-1):LET Y=Y-H:PLOT X,Y;
360       CALL A(I-1)
370    END IF
380 END DEF
390 DEF C(I)
400    IF I>0 THEN
410       CALL B(I-1):LET X=X+H:PLOT X,Y;
420       CALL C(I-1):LET Y=Y+H:PLOT X,Y;
430       CALL C(I-1):LET X=X-H:PLOT X,Y;
440       CALL D(I-1)
450    END IF
460 END DEF
470 DEF D(I)
480    IF I>0 THEN
490       CALL A(I-1):LET Y=Y-H:PLOT X,Y; !balra
500       CALL D(I-1):LET X=X-H:PLOT X,Y; !lefele
510       CALL D(I-1):LET Y=Y+H:PLOT X,Y; !jobbra
```

```
52Ø      CALL C(I-1)
53Ø    END IF
54Ø END DEF
```

38. program

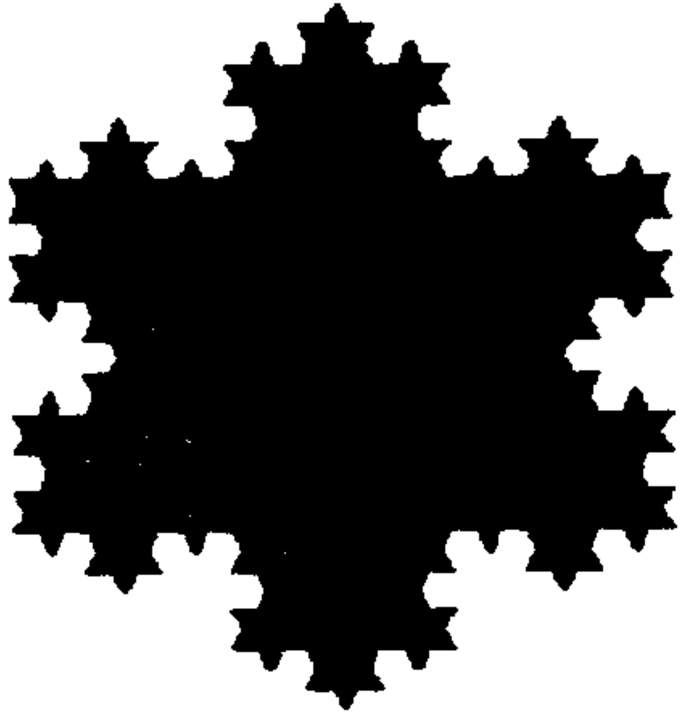
A teljes rekurzióval megírt programot a 39. program mutatja.

```
1ØØ !HILBERT gorbe teljes rekurzioval
11Ø GRAPHICS
12Ø OPTION ANGLE DEGREES
13Ø SET INK WHITE
 14Ø LET X=94Ø:LET Y=7ØØ:LET P=1 !paritas
 15Ø PLOT X,Y,ANGLE 18Ø;
 16Ø LET N=3 ! 6>= N >=1 a gorbe "rendje"
 17Ø LET S=1Ø*2^(6-N) !oldalhossz
 18Ø CALL HILBERT(S,N,P) ! jellemzok :
oldalhossz,renduseg,paritas
 19Ø END
 2ØØ DEF HILBERT(S,N,P)
 21Ø    IF N=Ø THEN EXIT DEF ! visszateresi feltetel
eggyel magasabb szintre
 22Ø    PLOT LEFT 9Ø*P; ! paritastol fuggoen balra
vagy jobbra
 23Ø    CALL HILBERT(S,N-1,-P) ! sajat magat meghivja
eggyel alacsonyabb szinten es negativ paritassal
 24Ø    PLOT FORWARD S;RIGHT 9Ø*P;
 25Ø    CALL HILBERT(S,N-1,P)
 26Ø    PLOT FORWARD S; !osszekotes
 27Ø    CALL HILBERT(S,N-1,P)
 28Ø    PLOT RIGHT 9Ø*P;FORWARD S;
 29Ø    CALL HILBERT(S,N-1,-P)
 3ØØ    PLOT LEFT 9Ø*P;
31Ø END DEF
```

39. program

Ez már kevésbé tekinthető át. A 24Ø. soron lévő megszakítási feltétel biztosítja, hogy a program az N. szintről az N+1. szintre térjen vissza a meghívások sorozatában.

Egy másik, a számítógépes grafika által kedvelt görbe, a Koch-görbe (6. ábra).



6. ábra

Rajzoljunk egy szabályos háromszöget! Osszuk három egyenlő részre minden oldalát. Rajzoljunk a középső harmadra kifelé egy újabb szabályos háromszöget, majd radírozzuk ki a középső harmadot. Az így keletkezett alakzat minden egyes szakaszára ismételjük meg az eljárást, tetszőlegesen sokszor. Ezután az alakzat rendjének nevezzük azt a számot, ahányszor az adott görbét előállító algoritmust megismételjük.

Vegyük szemügyre az ábrát! A teknős grafika a következő utasításokra rajzolja meg a hópehely görbe valamely oldalának soronkövetkező szakaszát:

1. Menj S/3 lépést előre
2. Fordulj 60°-ot balra
3. Menj S/3 lépést előre
4. Fordulj 120°-ot jobbra
5. Menj S/3 lépést előre
6. Fordulj 60°-ot balra
7. Menj S/3 lépést előre

Ebből már elkészíthetjük a rekurzív – önmagát különböző paraméterekkel hívó programrészletet.

A Koch-görbét a 40. programmal rajzolhatjuk meg.

```
100 ! KOCH gorbe
110 GRAPHICS
120 OPTION ANGLE DEGREES
130 SET INK WHITE
140 LET X=980:LET Y=360 !indulasi pont
150 PLOT X,Y;:PLOT ANGLE 210;
160 FOR I=1 TO 3 !egyenlooldalu haromszogre epit
170    LET S=700 !a kiindulasi oldalhossz
180    LET N=3 ! 1<= N <=6 a rajzolt hatarvonal rendje
190    CALL KOCH(S,N)
200    PLOT RIGHT 120;
210 NEXT
220 END
```

```
 230 DEF KOCH(S,N) ! az eljaras valtozoi:
oldalhossz,a gorbe rendje
 240    IF N=0 THEN PLOT FORWARD S;:EXIT DEF ! eljaras
adott melysegebol valo visszateres feltetele
 250    CALL KOCH(S/3,N-1) !oldal mindig harmada a
korabbinak
 260    PLOT LEFT 60; !balra
 270    CALL KOCH(S/3,N-1)
 280    PLOT RIGHT 120; !jobbra
 290    CALL KOCH(S/3,N-1)
 300    PLOT LEFT 60; !balra
 310    CALL KOCH(S/3,N-1)
 320 END DEF
```

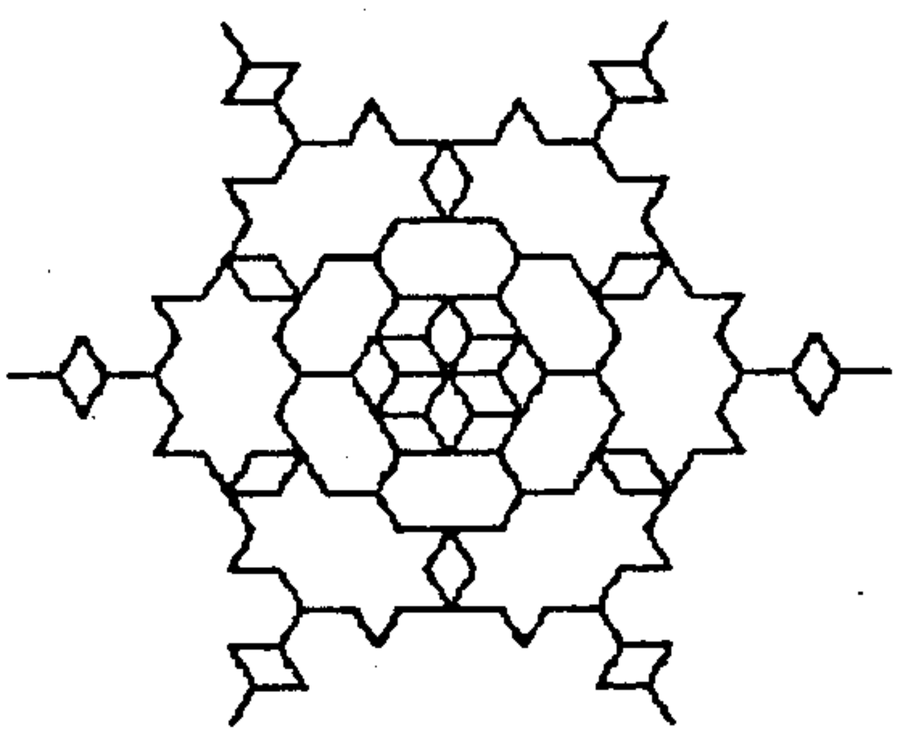
40. program

Amennyiben a KOCH (S, N) eljárást a 41. program szerint megváltoztatjuk, hasonló logikájú, de egészen más megjelenésű képet kapunk (7. ábra).

```
100 ! kiforditott KOCH-gorbe
110 GRAPHICS
120 OPTION ANGLE DEGREES
130 SET INK WHITE
140 LET X=1000:LET Y=20 !indulasi pont
150 PLOT X,Y;:PLOT ANGLE 180;
160 FOR I=1 TO 3 !egyenlooldalu haromszogre epit
170    LET S=810 !a kiindulasi oldalhossz
180    LET N=3 ! 1<= N <=4 a rajzolt hatarvonal rendje
190    CALL KOCH(S,N)
200    PLOT RIGHT 120;
210 NEXT
220 PLOT 1000,20:PLOT 0,0,PAINT
 230 END
 240 DEF KOCH(S,N) !eljaras valtozoi:oldalhossz,a gorbe
rendje
 250    IF N=0 THEN PLOT FORWARD S;:EXIT DEF !az eljaras
adott melysegebol valo visszateres feltetele
 260    CALL KOCH(S/3,N-1)
 270    PLOT RIGHT 60; !jobbra
 280    CALL KOCH(S/3,N-1)
```

```
290    PLOT LEFT 120; !balra
300    CALL KOCH(S/3,N-1)
310    PLOT RIGHT 60; !jobbra
320    CALL KOCH(S/3,N-1)
330 END DEF
```

41. program



7. ábra

COLOR

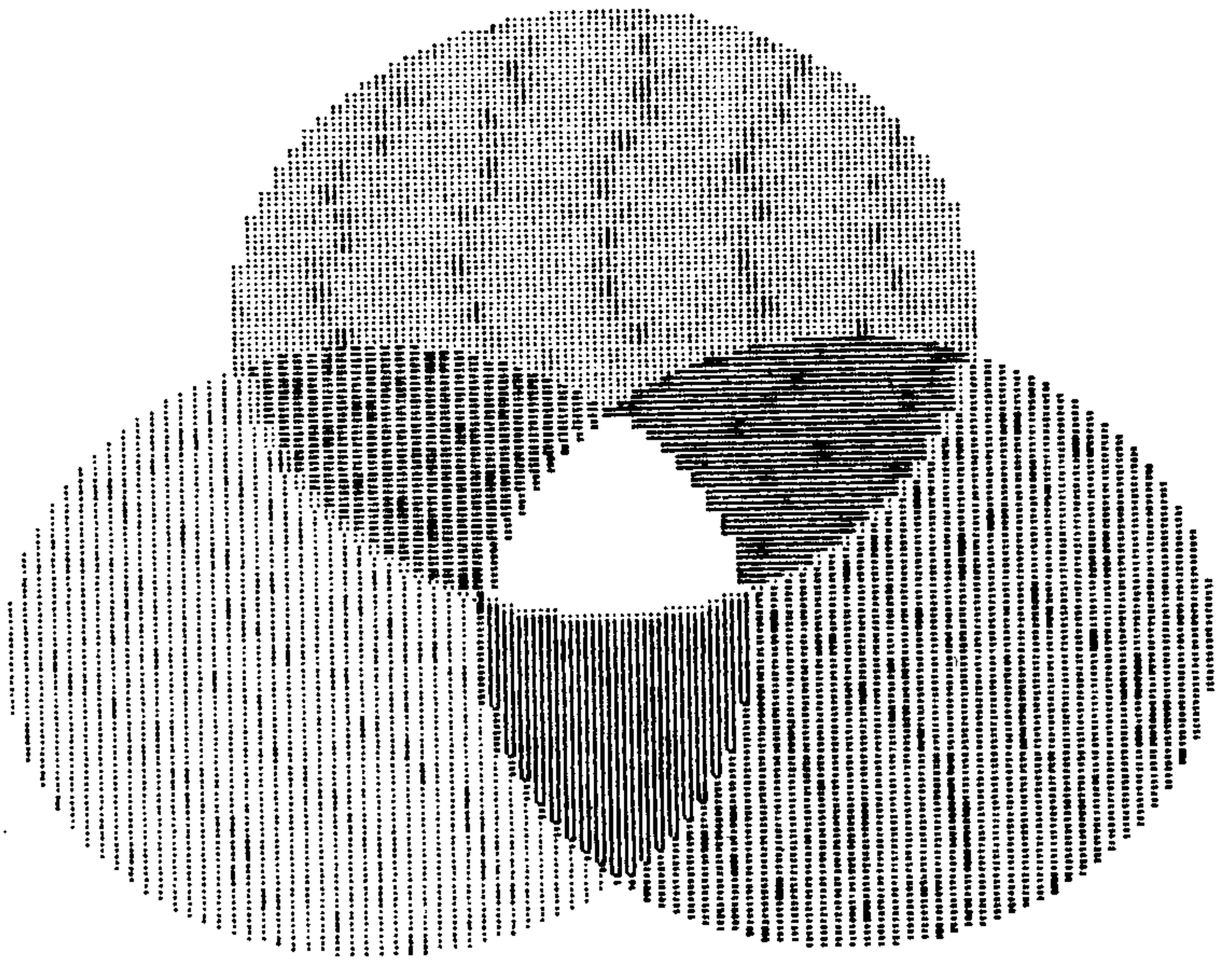
Az Enterprise által megjeleníthető színgazdagságnak a televízió szab korlátokat. Számítógépünk egyidejűleg 256 színt tud a képernyőre küldeni. Az egyidejűleg megjeleníthető színszám a GRAPHICS-módtól függ. A színek kijelölésére több lehetőség is kínálkozik. Az RGB viszont csak a 256 színű üzemmódban adja a helyes színárnyalatokat.

Nézzünk egy-két színes lehetőséget! A színkeverés szabályait mutatjuk be a 42. program segítségével.

```
100 !szinkor
110 GRAPHICS 256
120 SET INK RGB(1,0,0) !piros
130 PLOT 510,250,ELLIPSE 250,250,PAINT
140 SET INK RGB(0,1,0) !zold
150 PLOT 770,250,ELLIPSE 250,250,PAINT
160 SET INK RGB(1,1,0) !sarga
170 PLOT 640,200,PAINT
180 SET INK RGB(0,0,1) !kek
190 PLOT 640,460,ELLIPSE 250,250,PAINT
```

```
200 SET INK RGB(0,1,1) !cyan
210 PLOT 700,350,PAINT
220 SET INK RGB(1,0,1) !lila
230 PLOT 500,350,PAINT
240 SET INK RGB(1,1,1) !feher
250 PLOT 640,250,PAINT
```

42. program



8. ábra

Nyomtatóra is kiküldhetjük az ábrát, igaz itt inkább csak árnyalatokat tudunk megkülönböztetni (8. ábra).

Idővel majd csak felnőnek a nyomtatók lehetőségei a számítógépekéhez, áruk pedig leszáll azokéhoz. A 43. program egyelőre csak a szivárvány félköreit rajzolja meg, de nyitóképnek, hangulatteremtőnek és a színek bemutatására jó lehetőséget kínál.

```
  1 !voros,narancs,sarga,zold,kek,skek,ibolya
 100 !szivarvany 1
 110 GRAPHICS 256
 120 LET N=640:GOSUB 180 !felso burkolovonal rajzolasa
 130 FOR N=600 TO 340 STEP-40 !tovabbi szivarvanyivek
rajzolasa
 140   GOSUB 180
 150   PLOT 640,20+N,PAINT !a szivarvanyiv beszinezese
```

```
  160 NEXT
  170 END
  180 READ R,G,B
  190 SET INK RGB(R,G,B)
  200 PLOT 640,0;ELLIPSE N,N !kepernyo szelen hiba
 nelkul felkorivet rajzol
  210 RETURN
  220 DATA 1,0,0, 1,0,0, 1,.5,0, 1,1,0, 0,1,0, 0,1,.5,
0,1,1, 1,0,1
  230 !
  240 !
  500 !szivarvany 2     RUN 500
  510 GRAPHICS 16 ! finomabb feloldassal
  520 SET PALETTE RGB(1,0,0),RGB(1,.5,0),RGB(1,1,0),RGB
(0,1,0),RGB(0,0,1),RGB(0,0,.5),RGB(1,0,1),0
  530 LET SZIN=1
  540 FOR N=640 TO 340 STEP-40
  550    SET INK SZIN:LET SZIN=SZIN+1
  560    PLOT 640,0;ELLIPSE N,N
  570    IF N<640 THEN PLOT 640,N+10,PAINT
  580 NEXT
```

43. program

Mind a 256 színt a képernyőre csalhatjuk a 44. programmal.

```
100 !256 szin a kepernyon
110 GRAPHICS 256
120 LET SZIN=0
130 FOR V=0 TO 1200 STEP 80
140    FOR F=0 TO 700 STEP 44
150       SET INK SZIN !szin beallitas
160       FOR X=0 TO 72 STEP 8
170          PLOT V+X,F;V+X,F+43 !szines teglalap rajza
180       NEXT
190       LET SZIN=SZIN+1
200    NEXT
210 NEXT
```

44. program

Itt szép szabályosan sorakoznak a színes felületek, de a 45. program a kaleidoszkóp véletlenszerű színkavalkádját biztosítja.

```
100 !kaleidoszkop 1
110 GRAPHICS 256
120 RANDOMIZE
130 SET LINE MODE 3
140 FOR V=0 TO 1279 STEP 80
150   SET INK RND(255) !veletlen szin kivalasztas
160   PLOT 0,0;V,719 !vonal bal also sarokbol
170   SET INK RND(255)
180   PLOT 1279,0;V,719 !vonal jobb also sarokbol
190   SET INK RND(255)
200   PLOT 0,719;V,0 !vonal bal felso sarokbol
210   SET INK RND(255)
220   PLOT 1279,719;V,0 !vonal jobb felso sarokbol
230 NEXT
240 !
250 !kaleidoszkop 2  RUN 250
260 GRAPHICS 256
270 SET VIDEO COLOUR 3
280 RANDOMIZE
290 FOR N=1 TO 300
300   LET X=RND(600):LET Y=RND(300)
310   SET INK RND(255) !veletlen szin kivalasztas
320   FOR F=0 TO 8 STEP 4 !vizszintes , fuggoleges
pontmeret aranyainak korrigalasa
330     PLOT 640+X,310+Y+F !kozeppontos szimmetria
biztositasa
340     PLOT 640-X,310+Y+F
350     PLOT 640-X,310-Y-F
360     PLOT 640+X,310-Y-F
370   NEXT
380 NEXT
390 IF INKEY$="" THEN GOTO 390
400 RUN 250
```

45. program

A képernyő vagy annak egy-egy adott színű felülete a másodperc tört része alatt átszínezhető oly módon, hogy a szín PALETTE adott sorszámú színelemének új értéket adunk. Megtehetjük ezt a szín átdefiniálást a paletta több színével is. Ha az egymás mellett lévő felületeket úgy színezzük át, hogy azok mindig a tőlük jobbra (balra) levő színét veszik fel, akkor már három szín alkalmazása esetén is fényfüzért, fénypontok folyamatos mozgásának élményét idézhetjük fel a kép előtt ülőben. A 46. program a képre írt szöveg körül körbe forgó fénykeretet mozgat, forgat.

```
100 !KERET-PÖRGETES
110 NUMERIC D(3),A
120 LET D(0)=10:LET D(1)=20:LET D(2)=30
130 CALL KERET
140 PLOT 360,356,
150 PRINT #101:" FELIRAT "
160 SET INK 1
170 PLOT 360,356,
180 PRINT #101:"*       *"
190 CALL FOROG
200 GOTO 190
210 DEF KERET
220   GRAPHICS 16
 230   SET PALETTE 0,255,0,0,0,10,20,30
 240   LET C$=CHR$(159) !a keretet alkoto idom (lehetne
csillag,kor ...is)
 250   FOR N=5 TO 7
 260     SET INK N
 270     FOR I=-5 TO 9 STEP 3
 280       PLOT 100+72*(I+N),200,
'290       PRINT #101:C$
 300       PLOT 1108-72*(I+N),520,
 310       PRINT #101:C$
 320     NEXT
 330     PLOT 100,280+80*(7-N)
 340     PRINT #101:C$
 350     PLOT 1108,280+80*(N-5)
 360     PRINT #101:C$
 370   NEXT
380 END DEF
390 DEF FOROG
```

```
 400   LET A=D(0) !a keretet alkoto negyzetek szineinek
tovabbleptetese a palettan
 410   LET D(0)=D(1):SET COLOR 5,D(0)
 420   LET D(1)=D(2):SET COLOR 6,D(1)
 430   LET D(2)=A:SET COLOR 7,D(2)
 440 END DEF
```

46. program

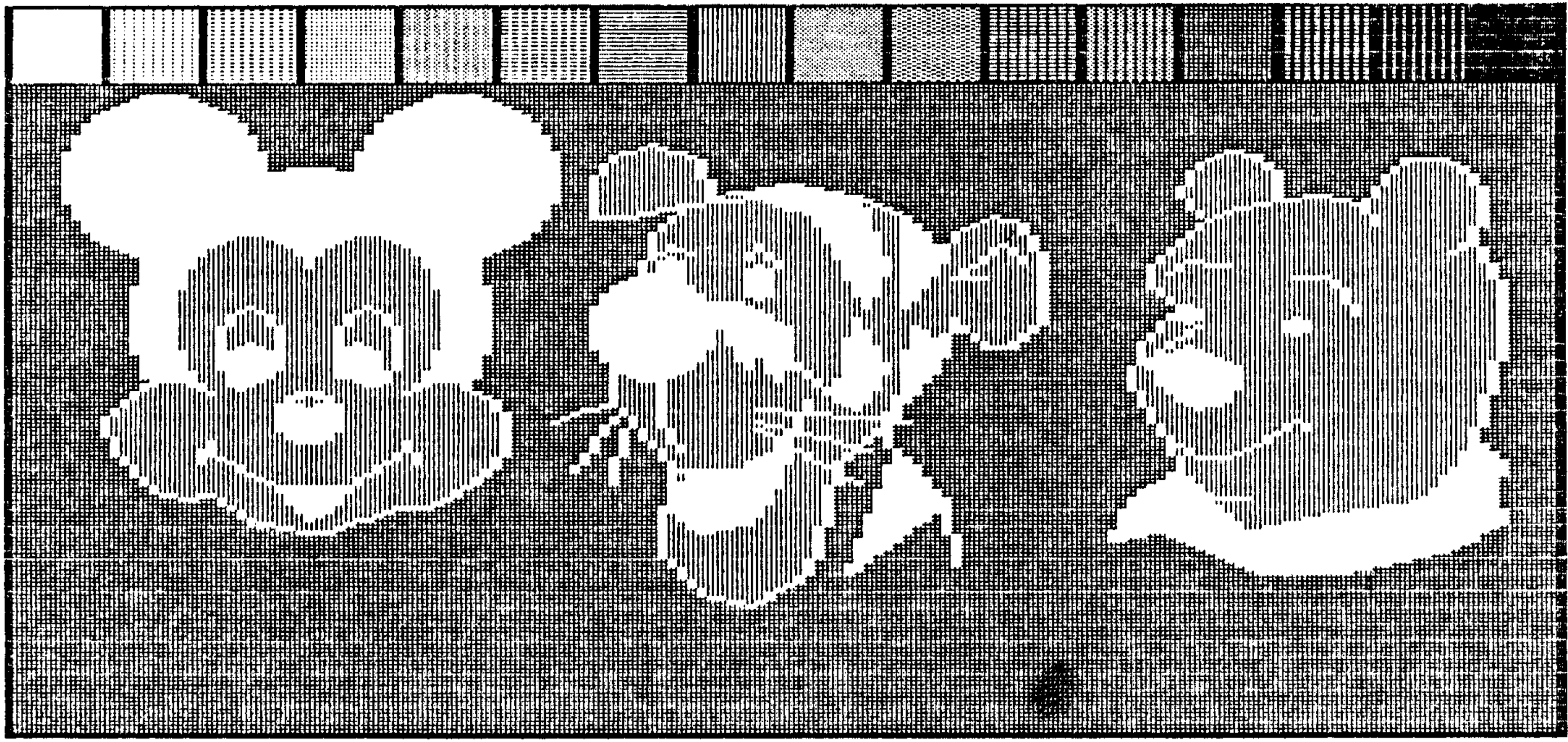
COPY

Az utasítás lehetőségei a forrás és célcsatorna kijelölésével sokrétűen variálhatók. Legáltalánosabban – alapértelmezésként – használható az editor csatorna nyomtatóra való kiküldése, kimásolása. A grafikus képernyő tartalmának kinyomtatására a német verziójú Enterprise gépeknél a VDUMP biztosít lehetőséget. A VDUMP végrehajtó rutinja a 4. szegmens F6974 kezdődik. Ez a hardcopy rutin még a színhatásokat is visszaadja, igaz ezt a vonaltípusok megválasztásával biztosítja. A demo kazetta ábráját a 9. ábrának megfelelően adja vissza a nyomtatón.

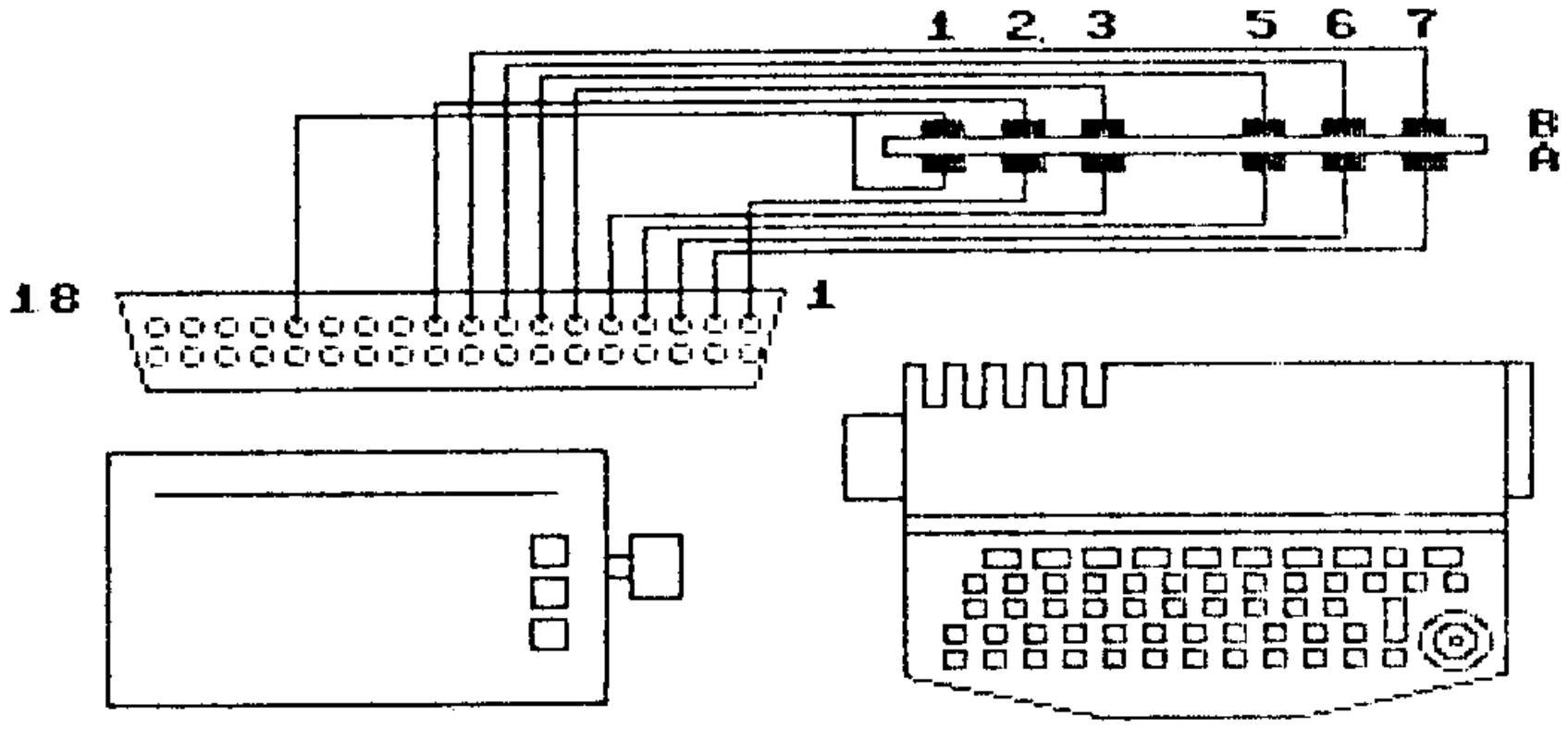
A hardcopy rutint mi magunk is megírhatjuk, erre több lehetőség is kínálkozik. A program letapogatja a képernyőt a LOOK utasítás felhasználásával. A képernyőn lévő információt át kell alakítani a nyomtató által grafikus karakter nyomtatását lehetővé tevő adatsorrá.

Nézzük meg, milyen adatokból képes grafikus jelet nyomtatni az Enterprise számítógéphez minden külön interface nélkül kapcsolható EPSON nyomtató. Először is a 10. ábra alapján – ahol a csatlakozókat a készülék háta felől nézve ábrázoltuk – el kell készíteni a csatlakozó zsinórt. Az alkatrészigény igen szerény.

A nyomtatóhoz, vagy a Centronics interface-hez, amphenol, a számítógéphez 2,54 mm osztástávolságú élcsatlakozó kell. Az összekötő kábel 12 eres (az sem lenne rossz, ha árnyékolt lenne, de rövid összekötő zsinórnál ez nem létfontosságú). A nyomtató vezérléséhez, működéséhez nélkülözhetetlen a nyomtató néhány fontos vezérlő parancsának ismerete:



9. ábra



10. ábra Nyomtató csatlakoztatása

PARANCS-KÓD	PARANCS-SZIMBÓLUM	FUNKCIÓ
CHR$(7)	BEL	Jelzőcsengő megszólaltatása
CHR$(8)	BS	Nyomtatási fej visszaállítása egy karakter pozícióval
CHR$(9)	HT	Vízszintes tabuláció végrehajtása
CHR$(1Ø)	LF	Soremelés
CHR$(11)	VT	Függőleges tabuláció végrehajtása
CHR$(12)	FF	Következő lap elejére állás
CHR$(13)	CR	Sor elejére állás +...
CHR$(14)	SO	Nyújtott karakternagyság kiválasztása
CHR$(15)	SI	Zsugorított CICERO karakternagyság kiválasztása
CHR$(18)	DC2	Zsugorítás megszüntetése
CHR$(2Ø)	DC4	SO, ESC SO parancsok nyújtó hatásának megszüntetése
CHR$(127)	DEL	Az utolsó nyomtatandó karakter törlése
CHR$(27)	ESC	után:
„*”	*	Képpontnyomtatás
„_”	_	Szövegaláhúzás beállítása és megszüntetése
„Ø”	Ø	1/8 inches soremelkedés beállítása
„1”	1	7/72 inches soremelés beállítása
„2”	2	1/6 inches soremelés beállítása
„3”	3	n/126 inches soremelés beállítása
„4”	4	Dőlt állású karaktertíp. kiválasztása
„5”	5	Egyenes állású karaktertíp. kiválasztása
„8”	8	Papírvég észlelés tiltás
„9”	9	Papírvég észlelés eng.
„<”	<	Egyirányban történő nyomtatás beállítása
„@”	@	Inicializálás
„A”	A	n/72 inches soremelés beállítása

PARANCS-KÓD	PARANCS-SZIMBÓLUM	FUNKCIÓ
„C”	C	Nyomtatási formátum hosszának meghatározása
„E”	E	Kiemelt nyomtatás kiválasztása
„F”	F	Kiemelt nyomtatás megszüntetése
„G”	G	Duplázott nyomtatás kiválasztása
„H”	H	Duplázott nyomtatás megszüntetése
„J”	J	n/216 inches soremelés beállítása
„K”	K	Normál képpontnyomtatás
„L”	L	Dupla sűrűségű képpontnyomtatás
„M”	M	Normál ELIT karakterméret kiválasztása
„N”	N	Nyomtatási formátumok közötti rés meghatározása
„O”	O	Nyomtatási formátumok közötti rés megszüntetése
„P”	P	Normál CICERO karakterméret kiválasztása
„Q”	Q	Jobb oldali margó beállítása
„R”	R	A 11 belső karakterkészlet közül egy kiválasztása
„S”	S	Kitevő, ill. index pozícióba nyomtatás
„T”	T	Normál pozícióba nyomtatás visszaállítása
„V”	V	Nyomtatási irány beállítása
„W”	W	Nyújtás kiváltása és megszüntetése
„Y”	Y	Dupla sűrűségű és sebességű képpontnyomtatás
„Z”	Z	Négyszeres sűrűségű képpontnyomtatás
„e”	e	Tabuláló paraméterek beállítása
„f”	f	Tabuláció végrehajtás megadott paraméterrel
„l”	l	Bal oldali margó beállítása
„m”	m	A 128–159 közé eső ASCII kódok értelmezése
„s”	s	Nyomtatási sebesség beállítása

A hardcopyhoz ezek közül a soremelés nagyságának beállítását kell először tisztázni, hiszen ahogy a képernyőn, itt is összefüggő képpontsort kell kapnunk. Az összefüggő képet, a 8/72 inches soremelés beállításával biztosíthatjuk.

.Egy sorban a grafikus üzemmódtól függően n független képpontoszlop különíthető el. Hogy hány képpontoszlopot akarunk kinyomtatni, azt a képpontnyomtatás módját kijelölő „K” parancs segítségével adhatjuk meg.

A képernyőről letapogatott információkat úgy kell átadni a nyomtatónak, hogy az a vett adatbyte-ot egy képpontoszlop (8 függőlegesen elhelyezkedő képpont) nyomtatási képeként értelmezhesse. A képpontoszlop egyes képpontjait az adat-

byte egyes értékeinek megfelelően: nyomtatja, ha a hozzárendelt bit 1 értékű, üresen hagyja, ha a hozzárendelt bit Ø értékű.

A hozzárendelési szabály: legmagasabb helyi értékű bit – legfelső képpont stb. Egy adott képpontoszlopot meghatározó, adatbyte-ot a következőképpen kell képezni:

$$x = \sum_{i=1}^{8} k_i \cdot 2^{(8-i)}$$

ahol k_i=1, ha az i. pont nyomtatandó; k_i=Ø, ha az i. üresen hagyandó. Ennyi ismeret birtokában megírhatjuk a saját hardcopy programunkat.

```
100 ! HARDCOPY    EPSON RX-80 ra
 110 ! elobb demo 4 be  pr szam valtas!
 120 LPRINT CHR$(27);"A";CHR$(8) !soremelest 8/72"-ra
beallit
 130 FOR Y=610 TO 48 STEP-32 !a kekernyon a figurak
ezen a teruleten vannak
 140   LPRINT CHR$(27);"K";CHR$(64);CHR$(1); ! egy
osszefuggo sor 320=1*256+64 pont szeles a nyomtaton
 150   FOR X=Ø TO 319 ! letapogatjuk a kepernyot
vizszintesen
 160     LET K,P=Ø
 170     FOR F=Y TO Y+31 STEP 4 ! a nyomtathato
karakter magassaga 8 pixel
 180       LOOK AT X*4,F:R !R a kepelem papir- vagy
attol eltero szinu
 190       LET V=-(NOT(R)) !V=Ø ha a pont papirszinu
 200       LET P=P+V*2^K:LET K=K+1 !egy grafikus
karakter oszlop osszeallitasa
 210     NEXT
 220     LPRINT CHR$(P); ! pufferbe gyujti a grafikus
karaktereket
 230   NEXT
 240   LPRINT CHR$(13) !ha osszejott a 320 karakter
akkor kinyomtatja, es sort emel
 250 NEXT
 260 END
```

47. program

A 47. programot írjuk be n. programként a számítógép memóriájába. Ha az itt közölt – a DEMO kazetta negyedik részéről ismert – 11. ábrát ki akarjuk nyomtatni, akkor a Ø. programként töltsük be a DEMO programot. Futtassuk azt le, majd amikor az ábra már a képernyőn van STOP-pal állítsuk le ezt a programot. CHAIN n-nel indítsuk el a korábban beírt hardcopy programunkat. Bizony az ilyen BASIC nyelven írt program igen hosszú idő alatt készíti el a másolatot.



11. ábra

Az *x*, *y* ciklusváltozók kezdő- és végértékének megváltoztatásával természetesen a képernyő bármelyik területéről készíthetünk másolatot. Vigyáznunk kell, hogy a „K" parancs utáni egy sorba nyomtatni kívánt karakterszám beállítást összehangba hozzuk az *x* ciklusváltozó kezdő- és végértékével!

A képen látott ábra a számítógép memóriájában is megtalálható. A rajz pontjait és szín információit a számítógép a grafikus üzemmód jellegétől függően tárolja. A tárolás módja az Enterprise memóriagazdálkodásától megszokott módon összevissza történik. Ez az összevisszaság persze igen korrekt logika alapján rendszerbe van foglalva.

Minden esetre, a 48. program használata a bekapcsolás utáni alapállapotú gép esetére igaz.

```
 100 ! hardcopy memoriabol elso 125 sor
 110 GRAPHICS ATTRIBUTE
 120 CALL DEMO
 130 LPRINT CHR$(27);"A";CHR$(1) ! pixelsoronkent
nyomtatunk, a soremeles 1/72"
 140 TIME "00:00:00" ! nyomtatasi ido meresehez ido
nullazas
 150 CALL HARDCOPY
 160 PRINT TIME$
 170 END
 180 DEF HARDCOPY
 190    FOR F=0 TO 125 ! a DEMO csak a kepernyo felso
reszet foglalja el
 200      LPRINT CHR$(27);"K";CHR$(64);CHR$(1); ! egy
```

```
sorba 32Ø kepernyopontot nyomtatunk
 21Ø      FOR V=Ø TO 39
 22Ø        LET K=SPEEK(254,113Ø8+F*4Ø+V) !memoriabol
kiolvassuk a grafikus kepernyo tartaom egy byte-jat
 23Ø        FOR I=7 TO Ø STEP-1 ! leszeleteli a
kepernyomemoria egy byte-jat
 24Ø          LPRINT CHR$(-1*((K BAND 2^I)=2^I)); !
a bitterkepbol Ø vagy 1 megallapitasa
 25Ø        NEXT
 26Ø      NEXT
 27Ø      LPRINT CHR$(13) ! a pufferbe gyujtott
32Ø grafikus nyomtatasi karaktert kikuldi, es sort eme

 28Ø    NEXT
 29Ø END DEF
 3ØØ DEF DEMO
 31Ø    FOR R=1Ø TO 3ØØ STEP 2Ø
 32Ø      PLOT 64Ø,34Ø,ELLIPSE 2*R,R
 33Ø    NEXT
 34Ø    PLOT 1ØØ,6ØØ,ELLIPSE 8Ø,8Ø,PAINT
 35Ø    PLOT PAINT 2ØØ,715;1279,715
 36Ø END DEF
```

48. program

Aki több videocsatornát nyit meg és azok részeit helyezi a képernyőre, annak azért erősen módosítania kell az itt közölt programot. Az alapértelmezésű gépben a grafika memóriaterülete a 254. és 255. szegmensen helyezkedik el. A kép egy része 57Ø5 byte a 254. szegmensen, míg a további 9328 byte információ a 255. szegmensen található.

CHR$

A karakterek közül nagyon sok vihető billentyűzetről a képernyőre. Vannak azonban képernyőn megjeleníthető karakterek, és nem látható, ún. vezérlőkarakterek, melyek csak CHR$(n) segítségével érhetők el. A TEXT üzemmódban értelmezhető karakterek táblázatát a következőkben tekinthetjük át:

```
CHR$(n)

  0  NUL      32  SPACE    64  @        96  `
  1  SOH      33  !        65  A        97  a
  2  STX      34  "        66  B        98  b
  3  ETX      35  #        67  C        99  c
  4  EOT      36  $        68  D       100  d
  5  ENQ      37  %        69  E       101  e
  6  ACK      38  &        70  F       102  f
  7  BEL      39  '        71  G       103  g
  8  BS       40  (        72  H       104  h
  9  TAB      41  )        73  I       105  i
 10  LF       42  *        74  J       106  j
 11  VTAB     43  +        75  K       107  k
 12  FF       44  ,        76  L       108  l
 13  RETURN   45  -        77  M       109  m
 14  SO       46  .        78  N       110  n
 15  SI       47  /        79  O       111  o
 16  DLE      48  0        80  P       112  p
 17  DC1      49  1        81  Q       113  q
 18  DC2      50  2        82  R       114  r
 19  DC3      51  3        83  S       115  s
 20  DC4      52  4        84  T       116  t
 21  NAK      53  5        85  U       117  u
 22  SYN      54  6        86  V       118  v
 23  ETB      55  7        87  W       119  w
 24  CAN      56  8        88  X       120  x
 25  EM       57  9        89  Y       121  y
 26  SUB      58  :        90  Z       122  z
 27  ESC      59  ;        91  [       123  {
 28  FS       60  <        92  \       124  |
 29  GS       61  =        93  ]       125  }
 30  RS       62  >        94  ^       126  ~
 31  US       63  ?        95  _       127  DEL

Kombinációk:
  CONTROL + SHIFT + P
  CONTROL + SHIFT + K
  CONTROL + SHIFT + L
  CONTROL + SHIFT + M
  CONTROL + SHIFT + N
  CONTROL + SHIFT + O
  CONTROL + SHIFT + I
  CONTROL + SHIFT + J
  CONTROL + SHIFT + H
  CONTROL + SHIFT + U
  CONTROL + SHIFT + Y
  CONTROL + SHIFT + T
  CONTROL + SHIFT + R
  CONTROL + SHIFT + E
  CONTROL + SHIFT + W
  CONTROL + SHIFT + Q
  CONTROL + SHIFT + A
  CONTROL + SHIFT + S
  CONTROL + SHIFT + D
  CONTROL + SHIFT + F
  CONTROL + SHIFT + G
  CONTROL + SHIFT + Z
  CONTROL + SHIFT + X
  CONTROL + SHIFT + C
  CONTROL + SHIFT + V
  CONTROL + SHIFT + B
  CONTROL + SHIFT + N
  CONTROL + SHIFT + M
  CONTROL + SHIFT + ,
  CONTROL + SHIFT + .
  CONTROL + SHIFT + /
  CONTROL + SHIFT + 1
  CONTROL + SHIFT + 2
  CONTROL + SHIFT + 3
  CONTROL + SHIFT + 4
  CONTROL + SHIFT + 5
  CONTROL + SHIFT + 6
  CONTROL + SHIFT + 7
  CONTROL + SHIFT + 8
  CONTROL + SHIFT + 9
  CONTROL + SHIFT + 0
  CONTROL + SHIFT + -
  CONTROL + SHIFT + =
  CONTROL + SHIFT + ;
  CONTROL + SHIFT + '
  CONTROL + SHIFT + [
  CONTROL + SHIFT + ]
  CONTROL + SHIFT + \
  CONTROL + SHIFT + `
```

```
  0  NUL
  1  SOH
  2  STX
  3  ETX
  4  EOT
  5  ENQ
  6  ACK
  7  BEL
  8  BS
  9  TAB
 10  LF
 11  VTAB
 12  FF
 13  RETURN
 14  SO
 15  SI
 16  DLE
 17  DC1
 18  DC2
 19  DC3
 20  DC4
 21  NAK
 22  SYN
 23  ETB
 24  CAN
 25  EM
 26  SUB
 27  ESC
 28  FS
 29  GS
 30  RS
 31  US
```

```
 32  SPACE
 33  !
 34  "
 35  #
 36  $
 37  %
 38  &
 39  '
 40  (
 41  )
 42  *
 43  +
 44  ,
 45  -
 46  .
 47  /
 48  0
 49  1
 50  2
 51  3
 52  4
 53  5
 54  6
 55  7
 56  8
 57  9
 58  :
 59  ;
 60  <
 61  =
 62  >
 63  ?
```

```
 64  @
 65  A
 66  B
 67  C
 68  D
 69  E
 70  F
 71  G
 72  H
 73  I
 74  J
 75  K
 76  L
 77  M
 78  N
 79  O
 80  P
 81  Q
 82  R
 83  S
 84  T
 85  U
 86  V
 87  W
 88  X
 89  Y
 90  Z
 91  [
 92  \
 93  ]
 94  ^
 95  _
```

```
 96  `
 97  a
 98  b
 99  c
100  d
101  e
102  f
103  g
104  h
105  i
106  j
107  k
108  l
109  m
110  n
111  o
112  p
113  q
114  r
115  s
116  t
117  u
118  v
119  w
120  x
121  y
122  z
123  {
124  |
125  }
126  ~
127  DEL
```

149	a	181	lap aljára	213	
150	a	182	enyved vég	214	
151	a	183		215	
152	a	184	balra	216	
153		185	enő elé	217	
154		186	enyved elé	218	
155		187		219	
156		188	jobbra	220	
157		189	enő utánra	221	
158		190	enyved vég	222	
159		191		223	

245	sortörlés
246	szinváltás
247	
248	új sor
249	vonalzáson
250	vált
251	
252	vonalzáson
253	sort töröl
254	sort visszatöröl
255	

9. lista Karakterek táblázata

DATE$

A számítógép igen optimista naptárral rendelkezik. A naptári dátumot egészen 2079 december 31-ig képes követni. (Jó lesz, ha időben lecseréljük a számítógépünket!)

A beírt dátumot a hónap–nap egyeztetést az 1. szegmens E5Ø2H–E513H táblázat segítségével végzi el. Elegánsan kezeli a szökőévek februárjait is. Bekapcsoláskor ugyan egy kicsit félszeg módon 1980 nulladik hónapjának nulladikát írja ki, de ilyen zagyvaságot tőlünk nem fogad el.

A DATE$ mellett kiolvashatjuk a naptári adatokat a BF75H–BF77H memóriacímekről is.

A BF77H 49Ø15dec az 1980 óta eltelt évek számát tartalmazza.

BF76H 49Ø14 címről a hónap míg a

BF75H 49Ø13 címről a nap olvasható ki.

A memóriacímek tartalma BCD-ben van. Innen adódik, hogy az egy byte-on ábrázolható legnagyobb szám 99dec. Most már megérthetjük; hogy miért „csak" 2079-ig használható a naptárunk.

1980+99 = 2079

DIM

A nagy tömegű adattal – igen sok változóval – dolgozó programok indexes változókat, tömbváltozókat (ki melyik elnevezést szereti jobban) igényelnek. Ezt a sok adatot előre kijelölt tömbökben (vektor, vagy mátrix) tárolhatjuk.

Kezdjük a kijelöléssel! Más számítógép típusoktól eltérően erre a célra a DIM mellett a NUMERIC és a STRING utasítás is rendelkezésünkre áll. A tömbök deklarálása kapcsán számítógépünknek több sajátossága (furcsasága?) van. A

```
1Ø DIM A (1Ø)
2Ø DIM A (1Ø)
```

utasítássor a kétszeres inicializálás hibával áll le. A

```
1Ø FOR I=1 TO 2
2Ø DIM A (1Ø)
...
...
5Ø NEXT
```

programrészlet szintaktikai hibához vezet, ez egyébként a skatulyázás jelzőbyte 7. bitjének állapotával hozható összefüggésbe, ahová a rendszer inicializálás jelzést helyez el a ciklus előszöri lefutása során a DIM-et tartalmazó programsor feldolgozásakor.

Vigyázni kell, mert azonos nevű tömbváltozó és változó alkalmazása is hasonló hibákhoz vezet, mint előbb és a hibajelzés is attól függ, hogy milyen sorrendben fordul elő a két azonos betűjel (név) használata:

```
1Ø DIM A (1Ø)
2Ø LET A=1Ø
```

vagy

```
1Ø LET A=1Ø
2Ø DIM A (1Ø)
```

A karakteres változó esetében is hasonló helyzet áll fent, mint a numerikus változók névhasználata között, de megengedett az A$=„SAMU” értékadás mellett az azonos betű (név) használata egy numerikus tömbhöz, pl. DIM A (5) alakban egy programon belül. A karakter, ill. számváltozók tárolásánál láttuk azok helyigényét. A stringeket tartalmazó tömbök alapértelmezés szerint 132 karakter hosszúak lehetnek. Erre az esetre foglal le helyet a megfelelő RST 1Ø rutin. Ez azonban azt is jelenti, hogy igen hamar kimeríthetjük az Enterprise 128-as bejelentkezéskor kiírt 114858 byte méretű memóriáját. Vigyázat ez így félreértésre ad okot!

Az Enterprise memória gazdálkodás zseniális, de a Z8Ø processzor által csak 65535-ig címezhetünk. Basic-ben a Ø. program számára maximálisan 44259 byte áll rendelkezésre, a Ø-tól eltérő sorszámú program részére pedig 327Ø2 byte használható fel. Ez a Ø. program részére biztosított több mint 43 kbyte nagyon hamar kimerülhet, ha a karakteres tömbváltozók részére csak úgy DIM S$(n) utasítással foglalunk helyet. Ilyenkor már a 1Ø DIM S$(329) egysoros program elindítása után is „kevés a memória” hibajelzést kapunk, márpedig egy tűrhető méretű adatkezelő programban is pár száz adatpárt kívánunk tárolni. Ilyen esetben – de általában mindig – célszerű csak a várható maximális hosszúságú stringnek megfelelő méretű tömböt lefoglalni.

Az IS–BASIC sajátossága, hogy a tömbök legfeljebb két dimenziósok lehetnek. Más gépeknél mód van a három dimenziós (vagy akár, a hétköznapi gondolkozással nehezen értelmezhető hat dimenziós) tér pont koordinátáinak egy

tömbben való tárolására a P (X,Y,Z) alakban. Nekünk ott az X (N), Y (N), Z (N) alakot célszerű alkalmazni, de a sík pontjainak leírásánál lehetőségünk van a P (X,Y) alakú adattárolásra.

A sík pontjainak tárolására egy speciális lehetőség a képernyőtartalom tömbben való tárolása. Így mintegy a képernyőn látható kép „tükörképét" is tárolhatjuk egy tömbben. Esetenként könnyebb ebben a tömbben elvégezni a vizsgálatot, mint a képernyőn, ill. a képernyő memóriában. Erre a lehetőségre ad példát a 49. program a DEMO kazettán is megtalálható bombázó játék egy másik megvalósításával.

```
100 ! bombazo jatek
110 SET STATUS OFF
120 DIM W(25,38)
130 LET R=0:LET T$=""
140 LET R$=" >-+":LET H$=CHR$(159)
150 LET B$=" "&CHR$(180)&CHR$(184)&"*"
160 LET P=0:LET T$="":PRINT AT 10,15:"kis turelmet"
170 CALL VAROS
180 FOR F=2 TO 23
190   FOR V=0 TO 34
200     PRINT AT 1,1:"pont=";P
210     IF W(F,V+3)=1 THEN GOTO 570 ! utkozott, vege
a jateknak
220     PRINT AT F,V:R$
230     IF T$=" " THEN GOTO 280
240     FOR T=1 TO 50
250     NEXT
260     LET T$=INKEY$ !bomba kioldas szokoz billentyu
vel
270     IF T$=" " THEN LET X=V+2:LET Y=F
280     IF T$=" " THEN CALL BONT
290   NEXT
300   PRINT AT F,V:"     "
310 NEXT
320 GOTO 170
330 DEF BONT
340   PRINT AT Y,X:B$
350   IF Y=22 THEN PRINT AT Y+1,X:" ":LET W(Y+1,X)=0:
LET T$=""
360   IF W(Y,X)=1 THEN LET P=P+1:LET W(Y,X)=0
```

```
370    LET D$=INKEY$:LET Y=Y+1
380 END DEF
390 DEF VAROS
400    FOR V=1 TO 38
410       FOR F=1 TO 25
420          LET W(F,V)=Ø
430       NEXT
440    NEXT
450    CLEAR SCREEN
460    PRINT AT 1,2Ø:"rekord=";R
470    RANDOMIZE
480    PRINT AT 24,2:"=========================
=====";
490    FOR V=7 TO 33
500       LET M=RND(17)
510       FOR F=23 TO 23-M STEP-1
520          PRINT AT F,V:H$
530          LET W(F,V)=1
540       NEXT
550    NEXT
560 END DEF
570 LET R=MAX(P,R) !rekord meghatarozasa
580 FOR Y=F TO 22 !rajzilag leesik
590    PRINT AT Y,V:"     "
600    PRINT AT Y+1,V:"*--*"
610 NEXT
620 PRINT AT 2,2:"tovabb, vege t/v"
630 LET Q$=INKEY$ !ujabb jatek inditasa
640 IF Q$="t" THEN GOTO 160
650 IF Q$<>"v" THEN GOTO 630
660 PRINT AT 1,2Ø:"rekord=";R
```


A játékot a szóköz billentyűvel vezérelhetjük. A képernyő tartalma a W (25,38) nevű, ill. méretű tömbben is megtalálható. Ahol célpont van, azt a helyet a tömbben 1, míg az üres képernyőterületet Ø jelzi. Erre azért van szükség, mert a játékot pontszámra kívánjuk játszani. A folyamatosan előre és egyre lejjebb szálló repülőgépről (R$) a szóköz billentyű segítségével indíthatjuk el a bombát (B$). A bomba útját a képernyőre rajzolja a program, de a W tömb-

ben vizsgálja, hogy ér-e pontot a lövés. Ha a W tömb sor, oszlop elemének értéke 1, akkor a pontszámot növeli, egyben a mátrix aktuális elemét Ø-ra állítja. A játék mindaddig folytatható, amíg a repülőgépünk neki nem ütközik a képernyőn lévő rajzjelnek, azaz a R$ V,F koordinátájának megfelelő mátrix elem érték nulla.

Új játékot a „t" billentyű lenyomásával indíthatunk. A Basic-ben írt játék sebessége természetesen vitatható. A továbbra is Basic programhoz ragaszkodók, pl. a GET-nél és a PRINT-nél találnak más, futást gyorsító megoldást, amit ebben a programban is felhasználhatnak.

Az 50. program szintén a képernyő tartalmát tárolja, itt most az L (19,36) tömbben.

```
100 ! Ariadne
110 DIM L(19,36),H(500)
120 NUMERIC X,Y
130 LET U,S,H(Ø)=Ø:RANDOMIZE
140 PRINT AT 1,1:"turelem"
150 CALL LETREHOZ
160 CALL RAJZOL
170 IF Y=3 THEN 350 ! megtalaltuk a kijaratot
180 IF L(Y,X+1)<>-1 THEN 240
190 IF L(Y,X+1)=-1 AND L(Y-1,X)=-1 THEN 320
200 LET YE=Y:LET XE=X:LET Y=Y-1:CALL TANUL
210 GOTO 170 ! ha meg lehet menj felfele
220 IF L(Y+1,X)<>-1 THEN 290
230 IF L(Y,X+1)=-1 THEN 170
240 LET YE=Y:LET XE=X:LET X=X+1:CALL TANUL
250 GOTO 220 !ha meg lehet menj jobbra
260 IF Y=19 THEN 480 !visszajutott a bejarathoz
270 IF L(Y,X-1)<>-1 THEN 330
280 IF L(Y+1,X)=-1 THEN 220
290 LET YE=Y:LET XE=X:LET Y=Y+1:CALL TANUL
300 GOTO 260 ! probald megegyszer lefele
310 IF L(Y-1,X)<>-1 THEN 200
320 IF L(Y,X-1)=-1 THEN 260
330 LET YE=Y:LET XE=X:LET X=X-1:CALL TANUL
340 GOTO 310 ! probaljuk balra
350 LET X=35:LET Y=19:LET R$="*":CALL UTVONAL
360 PRINT AT 22,5:"u= barmi m=megegyszer ez"
370 LET A$=INKEY$
```

```
380 IF A$="m" THEN 410
390 IF A$="" THEN 370
400 GOTO 130
410 LET R$=" ":CALL UTVONAL
420 LET R$="*":CALL UTVONAL
430 GOTO 360
440 DEF HELY
450    IF INKEY$="u" THEN RUN  !ujat kezdunk
460    LET U=U+1:LET H(U)=100*X+Y
470 END DEF
480 PRINT AT 22,4:"sajnos nem lehet atjutni"
490 IF INKEY$="" THEN 490
500 RUN
510 DEF TANUL
520    PRINT AT YE,XE:" "
530    PRINT AT Y,X:CHR$(154):CALL HELY
540    IF L(Y,X)<>0 THEN LET U=L(Y,X)
550    IF L(Y,X)=0 THEN LET L(Y,X)=U
560 END DEF
570 DEF UTVONAL
580    LET W=0
590    FOR I=1 TO U
600       LET X=INT(H(I)/100):LET Y=MOD(H(I),100)
610       PRINT AT Y,X:R$
620    NEXT
630 END DEF
640 DEF LETREHOZ
650    FOR F=3 TO 19
660       LET T=(F/2<>INT(F/2))
670       FOR V=4 TO 36
680          LET L(F,V)=T
690       NEXT
700       LET L(F,4),L(F,36)=-1
710       IF F=3 OR F=19 THEN 760
720       LET A=2-T*4+RND(3)
730       FOR I=1 TO A
740          LET L(F,7+RND(28))=NOT(T)
750       NEXT
```

```
760    NEXT
770    LET X=35:LET Y=19:LET L(3,5),L(Y,X)=Ø
780 END DEF
790 DEF RAJZOL
800    CLEAR TEXT
810    FOR F=3 TO 19
820       FOR V=4 TO 36
830          IF L(F,V)=-1 THEN PRINT AT F,V:CHR$(159)
840       NEXT
850    NEXT
860 END DEF
```

50. program

Ebben a programban a kép előbb készül el a tömbben és a mátrix elemeinek értéke alapján rajzolja a képernyőre a labirintus képét a program a 76Ø–83Ø sorok segítségével. Ha a labirintus így felülről nézve nem járható át, akkor célszerű az „u" betű lenyomásával új labirintust rajzoltatni. A R$ egy emberke, vagy egy csillag karakter a képernyő jobb alsó sarkán lévő bejáraton „bemegy" a labirintusba és a jobb fal súrolásának elvét következetesen betartva, megkeresi a kijáratot (ha van ilyen). A fal vagy átjáró felismerése itt is a tömb elemeinek figyelésével történik.

A H egydimenziós tömbben (vektor) az áthaladás során azoknak a pontoknak a koordinátáiból képzett értéket tárolja a program, amelyek az áthaladáshoz szükségesek. A kerülőket, a kétszer, vagy többször bejárt pontokat nem veszi fel ebbe a vektorba. Így, mire a bábu keresztülvergődik a labirintuson, addigra „meg is tanulja" a program a helyes útvonalat. Az „m" betű lenyomására újra meg újra keresgélés nélkül felrajzolja a kitapasztalt célszerű útvonalat. Ehhez a koordinátákat a H tömb értékeiből számítja ki. Új labirintus, mint mindig, a megtalált útvonal után is „u" betűvel kérhető a programtól.

A királynők a sakktáblán című 51. program meghatározza azt (azokat) a bábu elhelyezési lehetőségeket egy $n \times n$ méretű sakktáblán, mely mellett az n db királynő egymást nem képes leütni.

```
100 ! kiralynok a sakktablan
110 CLEAR TEXT
120 INPUT PROMPT "vezerek szama: ":V
130 IF V<3 OR V>7 THEN GOTO 120
140 CLEAR TEXT
150 PRINT "vezerek szama: ";V
160 CALL TABLA
170 PRINT AT 15,8:"abcdefgh"
```

```
 180 SET CHARACTER 154,0,60,60,24,126,24,60,102,0 !
kiralyno fekete alapra
 190 SET CHARACTER 155,255,195,195,231,129,231,195,153
,255 ! kiralyno feher alapra
 200 DIM T(V),S(8,8)
 210 FOR X=1 TO 8
 220   FOR Y=1 TO 8
 230     LET S(X,Y)=((X+Y)/2=INT((X+Y)/2)) !saktabbla
feher fekete jelzas bejegyzese
 240   NEXT
 250 NEXT
 260 LET I=0:LET C=3
 270 LET I=I+1:LET T(I)=1
 280 IF I=1 THEN 320
 290 FOR K=1 TO I-1 ! utes vizsgalat
 300   IF T(I)=T(K) OR ABS(T(I)-T(K))=I-K THEN 460
 310 NEXT
 320 IF I<V THEN 270
 330 CALL TABLA
 340 FOR L=8 TO 7+V
 350   PRINT AT 14-T(L-7),L:CHR$(154-S(8-T(L-7),L-7))
!a sakktabla szinetol fuggo vezer kirajzolasa
 360 NEXT
 370 PRINT AT 1,20:"a b c d e f g h"
 380 PRINT AT C,19:"";
 390 FOR A=1 TO V !eredmenytabla kiirasa
 400   PRINT T(A);CHR$(184);
 410 NEXT
 420 IF C>=22 THEN LET C=3
 430 LET C=C+1
 440 IF C>=22 THEN LET C=3
 450 LET I=I-1
 460 LET T(I)=T(I)+1
 470 IF T(I)<=V THEN 280
 480 LET I=I-1
 490 IF I>0 THEN 460
 500 IF C=3 THEN PRINT AT 20,3:"nincs megoldas"
 510 IF C<>3 THEN PRINT AT 20,3:"kesz"
```

```
520 END
530 DEF TABLA
540   FOR J=1 TO 8
550     PRINT AT 5+J,5:9-J;
560     LET A$=" "&CHR$(159)
570     IF J/2=INT(J/2) THEN LET A$=CHR$(159)&" "
580     FOR A=1 TO 4
590       PRINT A$;
600     NEXT
610   NEXT
620 END DEF
```

51. program

A sakktáblán biztosítani kell a sötét és világos mezőn lévő eltérő színű bábu rajzolási lehetőségét. Ez az üthetőség szempontjából ugyan közömbös, de a láthatóság megköveteli. E célra az átdefiniált CHR$ (154), CHR$ (155)-öt használja a program.

Az egy-egy vezér által pásztázott mezőket az S tömbben jelzi a program. A vezéreket sorba elhelyezi a nem veszélyeztetett mezők valamelyikére és az újonnan táblára helyezett (képzelt) vezér által veszélyeztetett mezőket jelöli. A vizsgálatot elvégzi, majd a táblán rajzban is megjeleníti és táblázatos formában ki is írja a megoldásokat.

DISPLAY

A képernyőn ablakot nyit meg, amelyen egyszerre több is lehet. Az ablak tartalma a háttérben is előkészíthető és a kívánt esetben a kész képernyőrészlet (szöveg vagy ábra) előhívható, megjeleníthető. Ezzel a lehetőséggel élve szinte mozgófilmszerű hatás érhető el Basic utasítások segítségével. Fényújságot mutat be az 52. program.

```
100 ! fenyujsag 53
110 SET VIDEO Y 2
120 SET VIDEO COLOUR 3
130 SET VIDEO MODE 1
140 OPEN #2:"video:"
150 OPEN #3:"video:"
160 CLEAR SCREEN
```

```
 170 !input k$
 171 LET K$="Kellemes karacsonyi ünnepeket kivanok mind
en kedves ENTERPRISE tulajdonosnak!     "
 180 LET K$="          "&K$
 190 LET H=LEN(K$):LET S=0
 200 LET S=S+1:LET A$=K$(S:S+8)
 210 FOR I=88 TO 16 STEP-32
 220    SET #2:INK 255
 230    PRINT #2:CHR$(26)
 240    PLOT #2:I,70,:PRINT #2:A$
 250    DISPLAY #2:FROM 1 TO 2 AT 10
 260    SET #3:INK 255
 270    PRINT #3:CHR$(26)
 280    PLOT #3:I-16,70,:PRINT #3:A$
 290    DISPLAY #3:FROM 1 TO 2 AT 10
 300 NEXT
 310 GOTO 200
```

52. program

A szöveg max. 247 karakter hosszú lehet, ebből a képernyőn egyidejűleg nyolc – a VIDEO MODE és COLOUR által meghatározott méretű – nagybetű látható. A szöveg mindig egy karakterrel lép balra, hogy ez a léptetés ne okozza a kép szétszakadását, a szöveg elmosódását, ezért felváltva íratjuk ki az egyik és a másik video lapra az egy-egy karakterrel odébb léptetett szöveget. Amikor a léptetés elkészült, akkor jelenítjük meg a kész képernyőtartalmat. Ehhez hasonló megoldásokat találhatunk még a TIME$ függvény kapcsán is, az idő kijelzésére.

A DISPLAY utasítás segítségével animáció valósítható meg az Enterprise 128-ason. Az Enterprise 128-as a hazánkban elterjedt személyi számítógépek között egyedülálló mértékben segíti az animációt. Sőt a hardver által lehetővé tett szolgáltatások nagy része Basic-ből is elérhető. Ezen a géptípuson tisztán Basic nyelven, viszonylag kis munkaráfordítással írható olyan (vagy jobb) program, mint amilyen más típuson csak speciális alkalmazói szoftver (pl. C64, GIGA MOVIE), vagy gépi kódú rutinok felhasználásával készíthető.

A folyamatos, vibrálásmentes mozgásérzet elérésének kulcsa:

- A látható képernyőn ne történjen nagyobb változás. Az egyes fázisok közötti átalakulás a háttérben, „láthatatlanul" menjen végbe!
- Az egyes fázisrajzok kellő gyakorisággal váltsák egymást! Elvileg másodpercenként legalább 16 képváltás szükséges a folytonosság-érzethez.

Az Enterprise IS–BASIC-ben mindkét feltétel teljesíthető. A munka során nagyszámú grafikus csatorna (egy-egy rajzlap) nyitható. Ezek bármelyikére rajzolhatunk akár láthatóan, akár „vakon" – a háttérben – és bármikor megjelenít-

hetőek a képernyőn. Ezzel teljesül az első feltétel: egy-egy elkészített fázisrajz készen áll a memóriában a megjelenítésre várva. A második feltétel is a hallatlanul sokoldalú video egység (NICK-chip) és a kiszolgáló szoftver révén teljesíthető. A két fázisrajz közötti átkapcsolás gyakorlatilag pillanatszerű: az egyik tv-képet még az első rajz alapján generálja a chip, az 1/25-öd másodperc múlva következő képen már a másik rajzot látjuk.

Természetesen nem kis probléma a fázisrajzok időre való elkészítése. Az elvileg ideális megoldás – amelyben az új rajzot a megjelenések közötti időben a program készíti el – a Basic sebessége mellett nemigen érhető el. Csak egészen egyszerű változások esetén kapunk viszonylag folytonos mozgásképet. Ebben a technikában lényegében két lapon dolgozunk.

A mozgássor felépítésének algoritmusa a következő lehet:

- az n. fázisrajz elkészítése az 1. lapon
- az 1. lap megjelenítése
- a 2. lap előkészítése [az (*n*—1). fázisrajz törlése]
- az (*n*+1). rajz elkészítése a 2. lapon
- a 2. lap megjelenítése
- az 1. lap előkészítése (az *n*. rajz törlése)
- az (*n*+2). rajz elkészítése az 1. lapon; vissza a ciklus elejére

Lényegesen egyszerűbb a helyzet, ha egy ciklikusan ismétlődő rajzsorozatot szeretnénk megjeleníteni (ilyen az 53. mintaprogram), vagy ha tetszőleges sorrendben, de adott rajzkészletből dolgozunk.

```
100 ! mozgoabra
110 SET VIDEO MODE 1
120 SET VIDEO COLOR 1
130 SET VIDEO X 15
140 SET VIDEO Y 15
150 FOR N=1 TO 12
160   OPEN #N:"video:"
170   CALL FAZIS(N)
180 NEXT
190 FOR N=1 TO 12
200   DISPLAY #N:AT 1 FROM 1 TO 15
210   FOR T=1 TO 20
220   NEXT
230 NEXT
240 FOR N=12 TO 1 STEP-1
250   DISPLAY #N:AT 1 FROM 1 TO 15
260   FOR T=1 TO 20
270   NEXT
```

```
280 NEXT
290 GOTO 190
300 DEF FAZIS(N)
310    DISPLAY #N:AT 1 FROM 1 TO 15
320    SET #N:INK 255
330    PLOT #N:260+N*5,270+6*(18-N) ELLIPSE 20,20
340    PLOT #N:220-N*5,270+6*(18-N) ELLIPSE 20,20
350    PLOT #N:240,250-6*(18-N);ELLIPSE N*10,20
360    PLOT #N:240,260;ELLIPSE 20,5*(15-N)
370    PLOT #N:240,270,ELLIPSE 12*N+80,270-16*N
380    PLOT #N:240,260+6*(15-N),PAINT
390 END DEF
```

53. program

Ekkor ui. a fázisrajzok (bemutatás előtt) tetszőleges idő alatt készíthetőek el az egyes video lapokon. A megjelenítéskor ismét élvezhetjük az Enterprise előnyeit. A többi géptípuson általában egy (vagy igen kevés) memóriaterület jeleníthető meg. A kijelzésre szánt képet – ideiglenes helyéről – egy blokkmozgató rutinnal kell erre a területre vinni, aminek végrehajtási ideje általában nem elhanyagolható. Az Enterprise gépen a felső négy szegmensben bármilyen memóriaterület megjeleníthető. Ezért ezen a gépen nem kell adatokat mozgatni. Elég a NICK-chippel „közölni", hogy honnan vegye a képinformációt, ami – még a Basic interpreter közvetítésével is – elég gyors az animációhoz.

Az egyetlen szűk keresztmetszetet a lapok száma jelenti, ami persze függ a lapok választott méretétől is. Pl. a mintaprogramban választott 1Ø × 1Ø-es lapok esetében az operációs rendszer 29 lapnak tud helyet foglalni a memóriában. Hatékonyabb memóriagazdálkodást az operációs rendszer ismeretében, gépi kódú programmal érhetünk el.

Az 54. mintaprogram – az animációs lehetőségek egyszerű illusztrációjaként – két egymást metsző színes kört mozgat egy 1Ø × 1Ø karakternyi méretű grafikus ablakban.

```
100 ! golyok
110 SET VIDEO MODE 1
120 SET VIDEO COLOR 1
130 SET VIDEO X 10
140 SET VIDEO Y 10
150 FOR N=1 TO 20 !lapok nyitasa
160    OPEN #N:"video:"
170    CALL RAJZOK(N)
180 NEXT
```

```
 190 LET A$=INKEY$
 200 IF A$="" THEN GOTO 190 !animacio billentyunyomasra
indul
 210 FOR N=1 TO 20 !megjelenites
 220    DISPLAY #N:AT 1 FROM 1 TO 10
 230    CALL KESLELTET
 240 NEXT
 250 FOR N=20 TO 1 STEP-1
 260    DISPLAY #N:AT 1 FROM 1 TO 10
 270    CALL KESLELTET
 280 NEXT
 290 GOTO 210
 300 DEF KESLELTET
 310    FOR T=1 TO 10
 320    NEXT
 330 END DEF
 340 !fazisrajzok
 350 DEF RAJZOK(N)
 360    DISPLAY #N:AT 1 FROM 1 TO 10
 370    SET #N:PALETTE BLACK,GREEN,YELLOW,RED
 380    PLOT #N:0,0;319,0;319,359;0,359;0,0,
 390    SET #N:INK 1
 400    PLOT #N:-10+16*N,160,ELLIPSE 70,70,
 410    SET #N:INK 2
 420    PLOT #N:PAINT
 430    SET #N:INK 3
 440    PLOT #N:328-16*N,200,ELLIPSE 70,70,
 450    PLOT #N:328-16*N,245,PAINT
 460    SET #N:INK 1
 470    PLOT #N:-10+16*N,130,
 480    PLOT #N:PAINT
 490 END DEF
```

54. program

11Ø–14Ø.: beállítja a fázisrajzok közös video feltételeit
- nagy felbontású grafikus üzem
- 4 szín
- 1Ø × 1Ø-es méret

15Ø–18Ø.: megnyit 20 video lapot, és ezeken elkészíti a fázisrajzokat

19Ø–2ØØ.:	billentyű lenyomásra vár
21Ø–28Ø.:	megjeleníti a kész fázisrajzokat
21Ø–24Ø.:	„előre" sorrendben
25Ø–28Ø.:	„hátra" sorrendben
29Ø.:	végtelen ciklusban ismétli a megjelenítést
3ØØ–33Ø.:	késleltetés az optimális mozgásérzet eléréséhez
36Ø.:	megjeleníti a megnyitott lapot (ez a sor futtatáskor elhagyható)
37Ø.:	színválaszték
38Ø.:	megrajzolja a lap keretét
39Ø–42Ø.:	megrajzolja és sárgára festi az első kört
43Ø–45Ø.:	megrajzolja és pirosra festi a második kört
46Ø–48Ø.:	zöldre festi az 1. kör 2.-kal nem közös részét

Más rajzsorozatnál a 15Ø–18Ø. sorok helyett építhető be a saját rutin, vagy az azt hívó utasítás. Természetesen az előre-hátra pörgetés is fölösleges lehet. A DISPLAY utasításokat AT utáni paraméterével adhatjuk meg, hogy a 10 sornyi grafikus ablak a képernyő hányadik sorától jelenjen meg. Ha az animált ablakot egy előre megrajzolt grafikus képernyő közepére „vetítjük", ügyes rajztechnikával elérhető, hogy a mozgó képrészletet alul-felül egy álló háttér egészíti ki.

DO/LOOP

A program egy részének ismétlése legegyszerűbben (és a legkevesebb veremterület felhasználásával), ezzel az utasításpárral valósítható meg.

A DO/LOOP utasításpár önmagában végtelen ciklust eredményez. A ciklus befejezését feltétel teljesüléséhez (vagy nem teljesüléséhez) kell kötni. A ciklus szervezhető elöl és hátul teszteléssel, így bizonyos feltételek teljesülésétől, és a feltétel vizsgálatának helyétől függően más-más futási eredményt kaphatunk. A DO/LOOP struktúrának előnye a FOR/NEXT szerkezettel szemben az is, hogy a ciklus magjaként lefutó programrész más-más lépésközt alkalmazhat. Erre mutat példát az 55. program, melynek 3ØØ–37Ø. sorai egy függvény gyökének meghatározására szolgálnak.

```
100 ! gyokkereses
110 INPUT PROMPT "y=":F$
120 TEXT
130 PRINT "220 def y(x)=";F$
140 PRINT "230 f$=";"""";F$;"""
150 PRINT "run 200"
160 SET FKEY 2 CHR$(177)&CHR$(13)&CHR$(13)&CHR$(13)
```

```
17Ø ! angol POKE 48816,66:POKE 48818,4
18Ø POKE 43915,29:POKE 43917,4 !nemet
19Ø END
2ØØ TEXT
21Ø CLEAR FKEYS
22Ø !!!!!!!!!!!!!!!!!
23Ø !!!!!!!!!!!!!!!!!
24Ø PRINT "y=";F$:PRINT
25Ø INPUT PROMPT "xmin=":XA
26Ø INPUT PROMPT "xmax=":XF
 27Ø LET S1=SGN(Y(XA))
 28Ø IF S1=SGN(Y(XF)) THEN PRINT "nem garantalt a gyok
!"
 29Ø INPUT PROMPT "pontossag=":EP
 3ØØ DO
 31Ø   LET V=(XA+XF)/2
 32Ø   IF S1=SGN(Y(V)) THEN
 33Ø     LET XA=V
 34Ø   ELSE
 35Ø     LET XF=V
 36Ø   END IF
37Ø LOOP UNTIL ABS(XA-XF)<2*EP
38Ø PRINT "Egy gyok:";(XA+XF)/2
```

55. program

A program további ötletet-trükköt is tartalmaz. A függvény inputként közölhető a számítógéppel, és azt kvázi direkt módban a program – önmagának felülírásával – felveszi a 2ØØ–23Ø. sorokba. A program újraindítását a billentyűpuffer és a funkcióbillentyű átdefiniálásának lehetősége biztosítja. A billentyűpuffer a különböző gépeknél más-más helyen helyezkedik el, ezért vagy a 17Ø., vagy a 18Ø. sor építendő be a programba. A függvény vizsgált tartományának birtokában a DO/LOOP struktúra számközfelező eljárással megkeresi a függvény EP pontosságú gyökét. Minthogy a gyökkereső algoritmus a függvényérték előjelváltozását figyeli, csak olyan esetben várható az adott intervallumban lévő gyök megtalálása, ha a függvény átmetszi a y=Ø egyenest.

A gyök behatárolásának gyorsaságát és pontosságát a függvényérték kiszámítása során felhasznált változó V érték biztosítja (31Ø. sor).

A ciklus szervezésének több lehetséges módját biztosítja az IS–BASIC. Hogy melyik megoldást választja a felhasználó, azt a programozó előtanulmányai, programozói stílusa jelöli ki. A strukturált, áttekinthető programírást a DO/LOOP struktúra talán jobban szolgálja, mint a FOR/NEXT szerkezet.

FOR–NEXT

A számítógép Neumann János által megfogalmazott ismérvei alapján kell, hogy ciklus szervezésére alkalmas legyen. A program egyes részeinek ismétlését különböző utasításokkal, szerkezettel lehet megvalósítani.

Az Enterprise FOR–NEXT struktúrája a kezdeti és végértékeket a ciklus elején vizsgálja, így ennél a számítógépnél – más típusú Basic nyelvjárástól eltérően – van nullaszor lefutó ciklus is, ha a végérték kisebb mint a kezdőérték, akkor a ciklust mintegy átlépi, de ilyenkor is mint minden más esetében a ciklus után a ciklusváltozó a végérték plusz egy értékét veszi fel. Az IS–BASIC érdekessége (de különösebb jelentősége nincs), hogy a NEXT utáni változónevet nem veszi figyelembe. Így a ciklusok egymásba ágyazásának rendjét az interpreter „kézben" tudja tartani. Nem fordulhat elő (mint más gépek esetében), hogy egy korábban megkezdett ciklust idő előtt zárjunk le (56. program).

```
10 ! NEXT furcsasagok
20 FOR A=1 TO 4
30    FOR B=1 TO 4
40       PRINT "a=";A,"b=";B
50    NEXT ABRAKADABRA
60 NEXT AKARMI
```

56. program

A ciklusok végrehajtásához a Basic veremben az interpreter 24 byte hosszú „csomagot" tárol. Ez a csomag Basic-ből hozzáférhető, mert a ciklus feldolgozása közben a veremnek ez az utolsó bejegyzett eleme. Erre az utolsó elemre a 228H–229H rendszerváltozó mutat. Márpedig, ha írásra és olvasásra is elérhető, akkor mi magunk is benyúlhatunk ebbe a területbe. De minek? Például arra, hogy menetközben, a ciklus futása közben a ciklus bizonyos jellemzőit megváltoztassuk. Ezek közül a jellemző értékek közül a végérték és a lépésköz jöhet számításba.

A végérték módosítása a kezdőérték felé nem lehet indokolt, mert ha az eredeti végérték elérése előtt kívánjuk befejezni a ciklust, akkor használhatjuk az egy bizonyos feltételhez kötött EXIT FOR utasítást. A lépésköz megváltoztatása (súrolja a programozói jólneveltség határait, de) igen érdekes futásokat eredményezhet. Nézzük pl. a Fibonacci-sor előállítását.

A Fibonacci-sor előállításának szabálya:

A sor első két eleme 1 és 1. A továbbiakat mindig az előző két elem összegeként értelmezzük

$$n_i = n_{i-1} + n_{i-2}$$

Más értelmezésben számegyenesen egy adott értékhez képest olyan nagyot kell előre lépni, mint az előző elem értéke volt. A kiírandó számok egymásután, tehát mindig változó lépésközzel (STEP?!) következnek.

Nézzük meg az 57. programot.

```
100 ! Fibonacci maskent
110 LET E=1
120 FOR I=0 TO 76000 STEP 0 !step valtozni fog !!!
130   PRINT I+E !n(x)=n(x-2)+n(x-1) ahol n(x-1)=I
140   ! ha a lepeskoz egesz akkor a FOR csomag 9. 10.
elemet eleg modositani
150   POKE PEEK(552)+256*PEEK(553)+9,MOD(E,256)
160   POKE PEEK(552)+256*PEEK(553)+10,INT(E/256)
170   LET E=I !n(x-2)=n(x-1)
180 NEXT
```

57. program

A program megértéséhez tudni kell, hogy a FOR–NEXT csomag a Basic veremben az IY regiszter tartalma által meghatározott címen kezdődik. Ehhez mi nem tudunk hozzáférni, de ez az érték a 228H–229H címről PEEK-kel kiolvasható.

A csomagban a lépésköz (STEP) értéke a számábrázolásnál megismert módon található meg. A tárolt érték a bejegyzés (IY által meghatározott címtől) 9–15. byte-ja. A Fibonacci-sor lépésközei egész számok lévén nekünk itt csak az első két (BCD számábrázolás!) helyen tárolt érték változtatásával kell foglalkoznunk. A program 14Ø–15Ø. sorai végzik el a lépésköz a STEP értékének megváltoztatását, mindig az előző eredmény értékére. Minthogy a lépésköz értéke 255-nél is nagyobb lehet, ezért kell mind a két byte-ot helyi érték szerint helyesen megváltoztatni.

GET

Az utasítás egy karakter beolvasását teszi lehetővé a kijelölt csatornáról. Általában a beolvasást a billentyűzetről, a magnóról, lemezmeghajtóról vagy a hálózatról végezzük.

Alig jut az eszünkbe, hogy a beolvasást a képernyőről végezzük el. Pedig GET esetében kijelölhető a képernyő is, mint ahonnan karaktert akarunk leolvasni. Jó hasznát vehetjük ennek a lehetőségnek, ha TEXT üzemmódban kívánjuk meghatározni azt, hogy a képernyő adott pozícióján van-e és ha igen, milyen karakter. Az 58. program egyszerű játék lehetőséget biztosít.

```
100 !   kepernyotartalmat figyel
110 REM ! gyorsitashoz
120 CALL INIC
130 LET Y=5+RND(12):LET X=3+RND(6):LET P=0
140 PRINT AT Y,X:"*";
150 PRINT AT 5+RND(12),30+RND(6):"O";
160 LET A=JOY(0)
170 IF A=0 THEN GOTO 160
180 LET YE=Y:LET XE=X
190 LET Y=Y+(A=8)-(A=4):LET X=X+(A=2)-(A=1)
200 !
210 !**********************
220 PRINT #102,AT Y,X:; !*
230 GET #102:P$ !        *
240 !**********************
250 !
260 IF P$=CHR$(31) OR P$="O" THEN
270   PRINT AT 1,15:"""T""""nyomasara var       ";
280   PING :PING
290   IF INKEY$<>"T" THEN GOTO 290
300   RUN
310 END IF
320 PRINT AT YE,XE:CHR$(159)
330 PRINT AT Y,X:"*"
340 LET P=P+1:PRINT AT 1,1:"pontszam=";P;
350 GOTO 160
360 DEF INIC
370   RANDOMIZE
380   SET STATUS OFF
390   SET KEY CLICK OFF
400   !  SET KEY RATE 1
410   !  SET KEY DELAY 10
420   CLEAR SCREEN
430   FOR I=3 TO 22
440     PRINT AT I,2:CHR$(159);
450     PRINT AT I,38:CHR$(159);
460   NEXT
470   FOR I=2 TO 38
```

```
480     !     PRINT #102,AT 2,I:CHR$(159);
490     PRINT AT 2,I:CHR$(159);
500     !     PRINT #102,AT 23,I:CHR$(159);
510     PRINT AT 23,I:CHR$(159);
520   NEXT
530   FOR I=1 TO 200
540     LET X=3+RND(35)
550     LET Y=3+RND(20)
560     PRINT AT Y,X:CHR$(159);
570   NEXT
580   PRINT AT 1,15:"""*""botkormannyal mozog   ";
590 END DEF
```

58. program

A képernyőn véletlenszerűen akadályok helyezkednek el. A karaktert kell ebben a labirintusban a botkormány segítségével a Ø karakterhez elvezetni. A program 22Ø–23Ø. sora figyeli a képernyő adott pozícióját. Ha a bábunkat akadálynak vezetjük vagy ha elérjük a célt, vége a „játéknak". Valahány lépést megteszünk mindig egy pontot kapunk. A listában ! jellel jelölt 4ØØ–41Ø., valamint 48Ø. és 5ØØ. sorok alkalmazása gyorsabbá teszik a program futását. Ilyenkor viszont a 49Ø. és 51Ø. sorokra nincs szükség.

GOSUB

Az alprogram hívás utasítása. A GOSUB utasítást az interpreter, a tokenizáló rutin más utasítással közös programsorban is elfogadja, de csak a sor utolsó utasításaként használjuk. Ez azért célszerű, mert a GOSUB utasítás végrehajtása után az alprogramból nem a meghívást követő memóriacímre, hanem a következő programsorra tér vissza a program a RETURN hatására. Az alprogram önmagát is többször akár 1472Ø alkalommal is meghívatja. Ez a meghívási mélység csak a szabad memóriaterület nagyságától függ. Nyilván ilyen meghívás sorozatot nem végzünk, de egy kvázi rekurzió kedvéért többször is felhasználhatja a szubrutin saját magát. Az 59. program szerinti feltételhez kötött visszatérés esetén az interpreter a Basic vermet rendben hagyja maga után.

```
1ØØ ! szubrutin onmagat hivja
11Ø DEF Y=PEEK(552)+256*PEEK(553)
12Ø PRINT Y
13Ø LET U=1:LET A=-1:GOSUB 16Ø
```

```
150 END
160 LET A=A+1
170 PRINT A;".hivas  ","verem mutato:";Y
180 IF A=6 THEN RETURN
190 GOSUB 160
200 PRINT U;".rendez ","verem mutato:";Y
210 LET U=U+1
220 RETURN
```

59. program

Olyan esetben, ha valamely feltétel teljesülése, vagy nem teljesülése miatt az alprogramból (subrutinból) úgy kellene visszatérni, hogy ne az interpreter által kijelölt hívás utáni helyre kerüljön a vezérlés, akkor a vermet előbb rendezni kell. Ehhez tudni kell azt, hogy a GOSUB utasítás három byte hosszú csomagot tárol, amit a RETURN leemel, és ebből állapítja meg a visszatérési címet.

A RETURN n jellegű utasítás úgy valósítható meg a 60. program szerint, hogy bizonyos feltétel teljesülése esetén GOTO-val túllépünk a RETURN utasításon. Ott leemeljük a GOSUB csomagot, ez után GOTO-val, vagy ON A GOTO utasítással már tetszőleges helyen folytatható a program futása.

```
100 ! verem rendezes RETURN n
110 ! felkialtojeles sorok elhagyhatok
120 PRINT PEEK(552)+256*PEEK(553) !
130 GOSUB 200
140 PRINT PEEK(552)+256*PEEK(553) !
150 PRINT "szabalyosan vissza"
160 END
170 PRINT "mashol vagyok"
180 PRINT PEEK(552)+256*PEEK(553) !
190 END
200 PRINT PEEK(552)+256*PEEK(553) !
210 PRINT "menu":PRINT
220 PRINT " Ø  akkor RETURN-nel vissza"
230 FOR I=1 TO 5
240   PRINT I;I*1000;".sorra GOTO-val"
250 NEXT
260 INPUT PROMPT "ha nem Ø akkor mashova ":A
270 IF A=Ø THEN RETURN
280 LET P=PEEK(552)
```

```
290 IF P<253 THEN
300   POKE 552,P+3
310 ELSE
320   POKE 552,P-253
330   POKE 553,PEEK(553)+1
340 END IF
350 PRINT "verem rendezve"
360 IF A>5 THEN GOTO 210
370 ON A GOTO 1000,2000,3000,4000,5000
380 END
1000 PRINT "most: 1000.sor":STOP
2000 PRINT "most: 2000.sor":STOP
3000 PRINT "most: 3000.sor":STOP
4000 PRINT "most: 4000.sor":STOP
5000 PRINT "most: 5000.sor":STOP
```

60. program

GOTO

A feltételhez kötött, vagy feltétel nélküli ugró utasítást biztosítja a megadott sorszámra. Csak élő sorszám lehet, amit az utasításban megadunk.

A GOTO végrehajtó utasításnak van egy igen széles körű vizsgálati szakasza (5. szegmens DIECH–D211H), mely az ugrás elvégzése előtt a Basic vermet rendbe teszi. Így FOR–NEXT, DO/LOOP blokkokból is ki lehet lépni GOTO-val. Ez a vizsgálat ugyanazokat a verem tisztításokat elvégzi, mint az EXIT FOR, vagy EXIT DO utasítás, ha a GOTO-val megcímzett sor a struktúrán kívül helyezkedik el. Azoknál talán még jobb is, mert nemcsak a blokkzáró utasítás utánra, hanem a blokk kezdete elé is el lehet ugorni. Az örök szabályra viszont vigyázni kell, a blokkokba „beugrani" nem lehet!

GRAPHICS

A rajzolási, a nagyfelbontású grafikus lehetőségek az Enterprise számítógépen Basic-ből könnyen elérhetők, ehhez azonban a GRAPHICS utasítással a megfelelő lapot ki kell választani. Egy grafikus képernyő 144ØØ byte memóriaterüle-

tet vesz igénybe. Alapértelmezésű gépben a grafikus képernyő két részben, két külön szegmensen található meg. Az első rész a 254. szegmensen 113Ø8–16383 és a második rész a 255. szegmensen Ø–9323 címen található. Természetesen elég rugalmas az Enterprise memória szervezése ahhoz, hogy ez is megváltoztatható legyen. Ha eleve belapozzuk a 177dec portra a 254. szegmenst, akkor az egész grafikus képernyőt egy helyen is megtaláljuk a 27692–42Ø91 címen. A kétféle lehetőség kihasználására, adott esetben a program futásának idejére való hatását a 61. és a 62. program segítségével tanulmányozhatjuk.

```
100 !kifeherito POKE grafikus kepernyo memoriaterulete
re
110 GRAPHICS
120 TIME "00:00:00"
130 FOR I=11308 TO 16383 !a kepernyo eleje az osztott
memoriaban
140   SPOKE 254,I,255
150 NEXT
160 FOR I=0 TO 9323 ! a kepernyoterulet masodik resze
170   SPOKE 255,I,255
180 NEXT
190 PRINT TIME$
```

61. program

```
100 !kifeherito POKE athelyezett  grafikus kepernyo me
moriateruletere
110 GRAPHICS
120 OUT 177,254
130 TIME "00:00:00"
140 FOR I=27692 TO 41836
150   POKE I,255
160 NEXT
170 PRINT TIME$
```

62. program

A grafikus üzemmód használatának több lehetősége van. Az egyidejűleg használni kívánt színkombinációktól függően kell megválasztani a grafikus módot.

A HIRES GRAPHICS üzemmódban a képernyő egy sora 80 byte-ból áll, az ATTRIBUTE módban pedig 40-ből.

GRAPHICS 2. üzemmód. A sor egy byte-jának bitjei határozzák meg a rajzolást. Ahol 1-es áll, ott az INK 1 rajzol, ahol Ø, ott pedig az INK Ø.

GRAPHICS 4. üzemmód. Ebben a módban egy byte négy kétbites részre oszlik. Az ilyen esetben alkalmazható négy szín kódja a PALETTA szín sorszáma Ø–3 között. A byte tartalmát a kódok és pozíciók kölcsönhatása határozzák meg, az alábbiak szerint:

POZÍCIÓ	PALETTA SORSZÁM	Ø	1	2	3
1		Ø	16	1	17
2		Ø	32	2	34
3		Ø	64	4	68
4		Ø	128	8	136

GRAPHICS 16. mód esetében egy byte két részre oszlik az $1 \leq N \leq 2$ pozíció színkódjának tárolására.
Az adott pozícióról tárolt információ a 16 színárnyalatot reprezentálja az alábbiak szerint:

SZÍN	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15
X=	Ø	64	4	68	16	8Ø	2Ø	84	1	65	5	69	17	81	21	85

X=N.X

GRAPHICS 256. üzemmód. Az adott pozícióról tárolt információ a pont színkódja, így az egész képpont a byte által meghatározott színű lesz.
GRAPHICS ATTRIBUTE. Attribútum üzemmódban a képernyő száznyolcvan 4 byte szélességű sorból áll. Ez azt jelenti, hogy ahol eddig két pont, két byte volt, ott most egy, kétszer olyan széles pont található. Ezeket a byte-okat úgy kell elképzelni, mint a GRAPHICS 2 módban.
A különböző grafikus módok hatását ki kell ismernie a felhasználónak, ehhez minden lehető és lehetetlen alkalmat fel kell használni.
A 63. program a képernyőre egyidejűleg négy ablakot nyit, melyekre különböző VIDEO COLOR MODE-okat állít be. Az ablakokban azonos szöveg és a beállítás jellemzője látható. Ez a program biztosítja, hogy egymás mellett láthassuk, ill. összehasonlíthassuk az utasítás hatását.

```
100 ! grafikus modok
110 SET VIDEO X 40 !megnyitando lapok szelessege
120 SET VIDEO Y 6 ! megnyitando lapok magassaga
130 LET X=0
140 FOR I=3 TO 0 STEP-1
150    SET VIDEO MODE 1
160    SET VIDEO COLOUR I !vizszintes keppontszam
kijeloles x=640/2^i
170    OPEN #I+10:"video:"
180    DISPLAY #I+10:AT I*6+1 FROM 1 TO 6 !kepernyon
ablakot jelol ki a megjeleniteshez
190    SET #I+10:PALETTE 50,100,150,200
```

```
200    SET #I+10:PAPER I
210    PLOT #I+10:0,215;1279,215
220    PLOT #I+10:0,0;1279,0
230    PLOT #I+10:(640-40*2^(I+1)),200,
240    PRINT #I+10:"ENTERPRISE"
250    PRINT #I+10:"colour :";I
260 NEXT
```

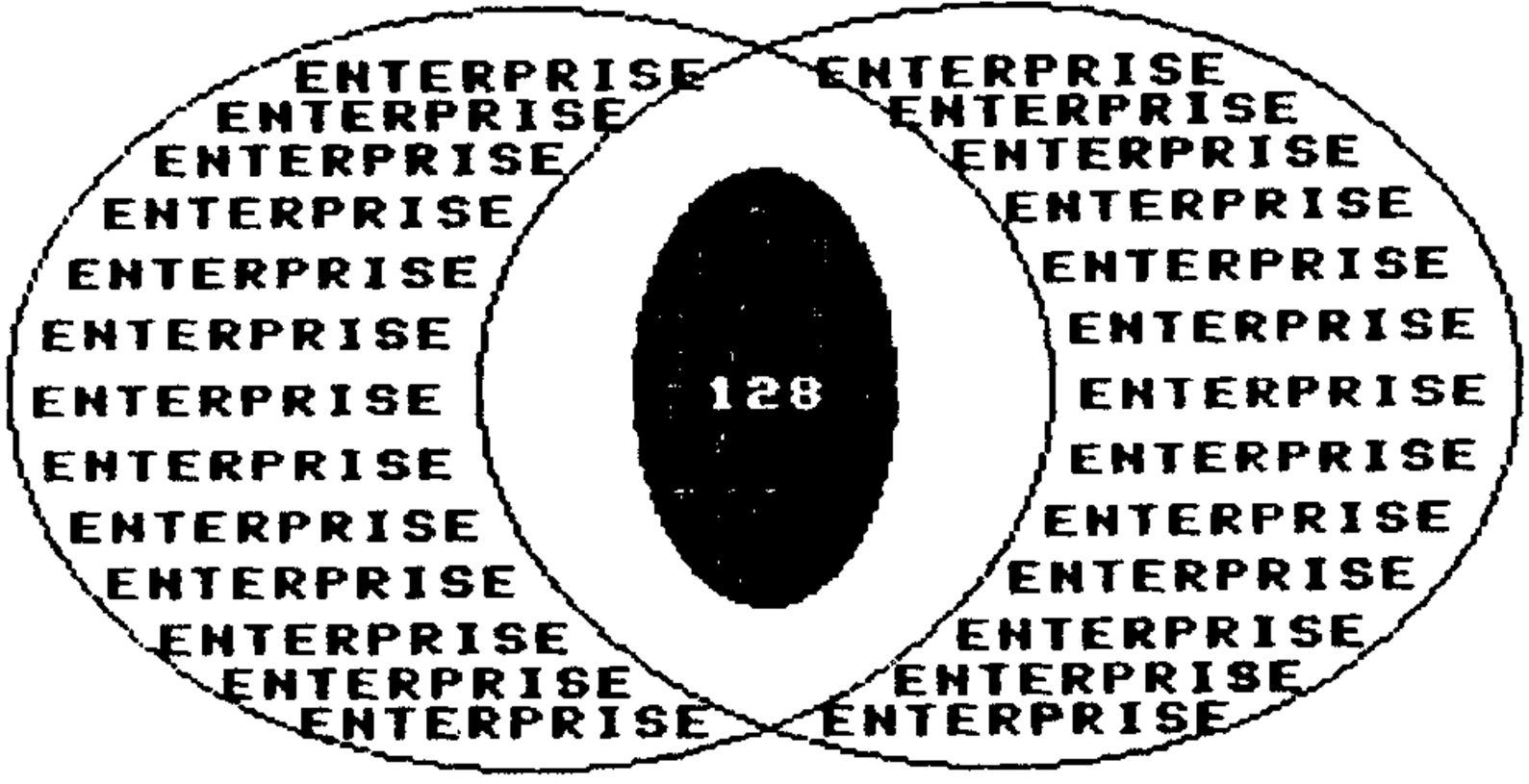
63. program

A 64. program ENTERPRISE reklámszöveget ír a képernyőre a fekete-fehér televízió előtt ülő géptulajdonosok kedvéért csak két színű üzemmódban.

```
100 !grafikus kepernyore ir
110 SET STATUS OFF
120 GRAPHICS
130 SET PALETTE 255,0
140 SET BORDER WHITE
150 SET PAPER 1
160 SET INK 2
170 OPTION ANGLE DEGREES
180 PLOT 480,365,ELLIPSE 400,350
190 PLOT 840,365,ELLIPSE 400,350
200 FOR X=0 TO 360 STEP 10
210    IF X=70 THEN LET X=120
220    IF X=250 THEN LET X=300
230    LET V=500+400*COS(X):LET F=370+340*SIN(X)
240    PLOT V,F,
250    PRINT #101:"ENTERPRISE"
260 NEXT
270 PLOT 660,365,ELLIPSE 100,200,PAINT
280 SET LINE MODE 3
290 PLOT 615,375,:PRINT #101:"128"
```

64. program

A 15Ø. soron kezdődő ciklus kört rajzol. Igaz, hogy van erre tökéletesebb, gyorsabb lehetőség, de itt mi nem is a kör kerületét akarjuk megrajzolni, hanem a kör kerületének egyes pontjaira van szükségünk, hogy ezektől a pontoktól indulva szöveget tudjunk a képernyőre írni. A futás eredménye a 12. ábrán látható.



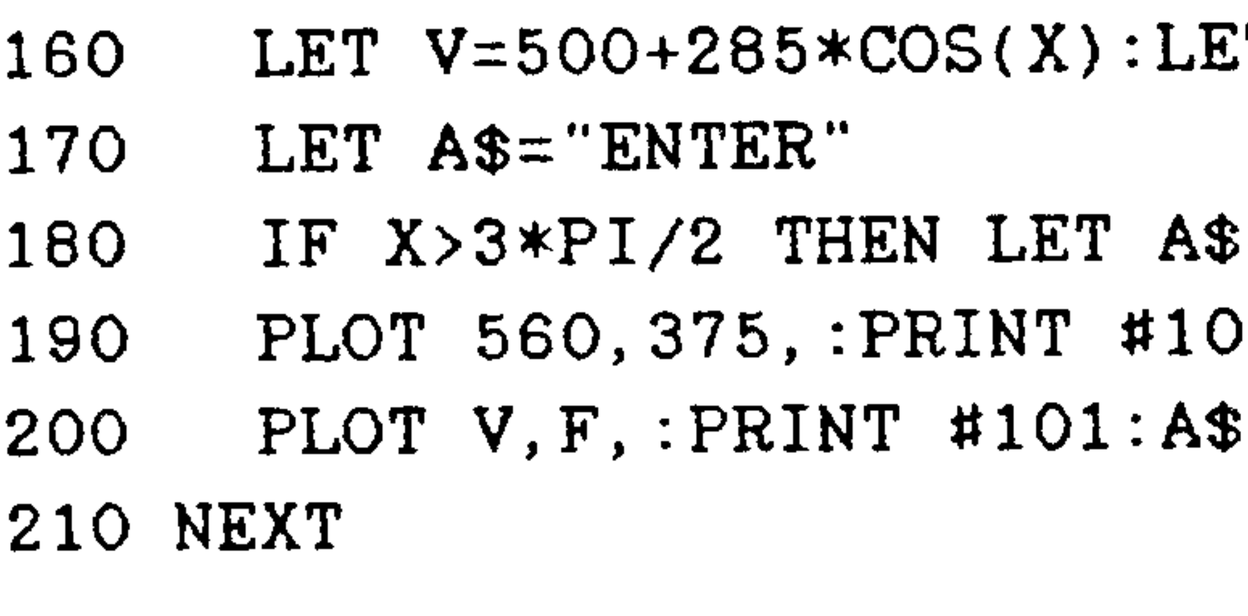
12. ábra

Még mindig reklám és még mindig kör polárkoordinátákkal felírt egyenlete, de most már GRAPHICS 16 üzemmód és a kör nem is teljes kör, hanem annak csak egy része.

Ezzel a programmal állítható elő a 13. ábra.

```
100 !grafikus kepernyore ir 2
110 GRAPHICS 16
130 PLOT 660,345,ELLIPSE 100,180,PAINT
140 SET LINE MODE 3
150 FOR X=.7*PI TO 2.3*PI STEP PI/10
160    LET V=500+285*COS(X):LET F=370+340*SIN(X)
170    LET A$="ENTER"
180    IF X>3*PI/2 THEN LET A$="PRISE"
190    PLOT 560,375,:PRINT #101:"128"
200    PLOT V,F,:PRINT #101:A$
210 NEXT
```

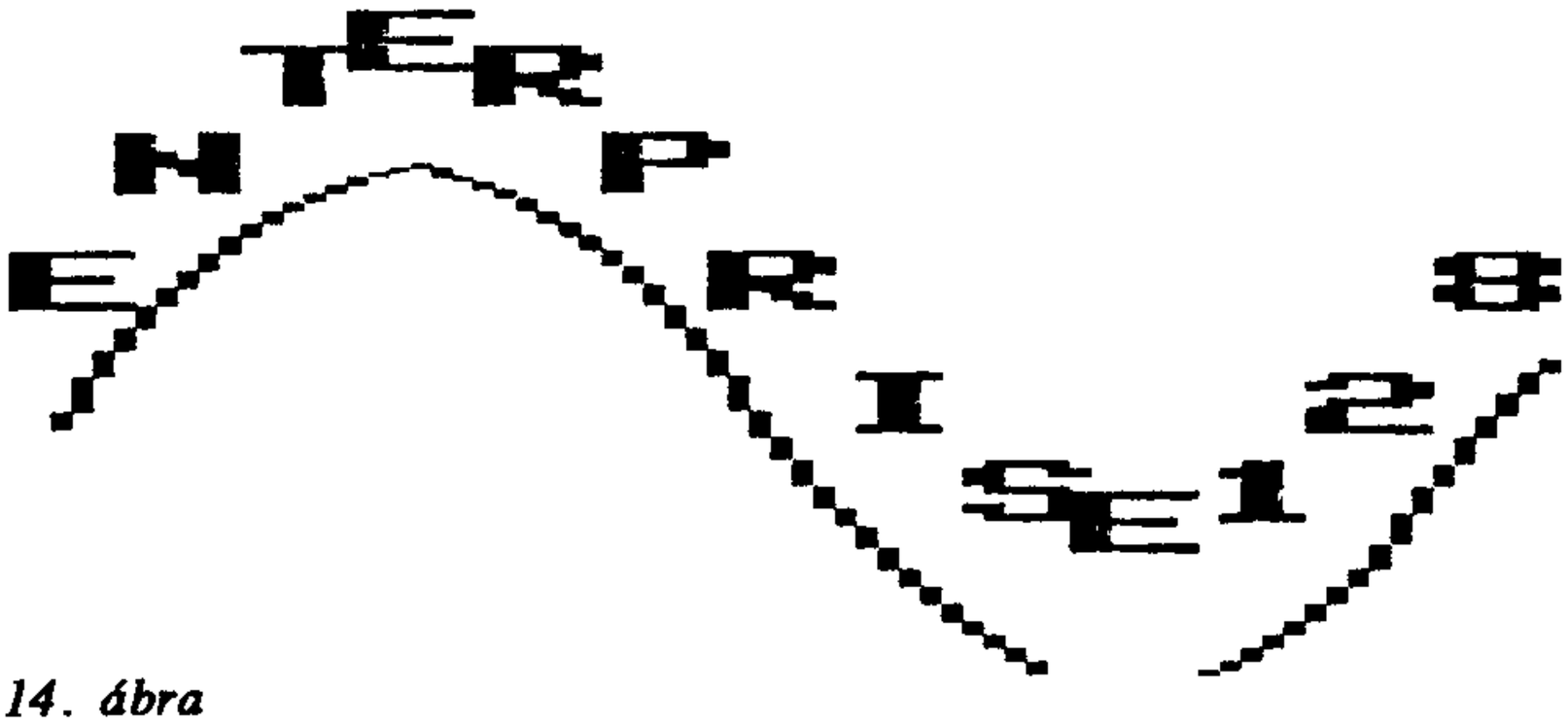
65. program



13. ábra

A színes lehetőségek kihasználását bemutató program minden betűt és a szinuszhullám pontjait más-más színnel rajzolja a képernyőre. A pontok színei véletlenszerűen változnak a 14. ábrán.

Ezt a képet a 66. programmal állíthatjuk elő.



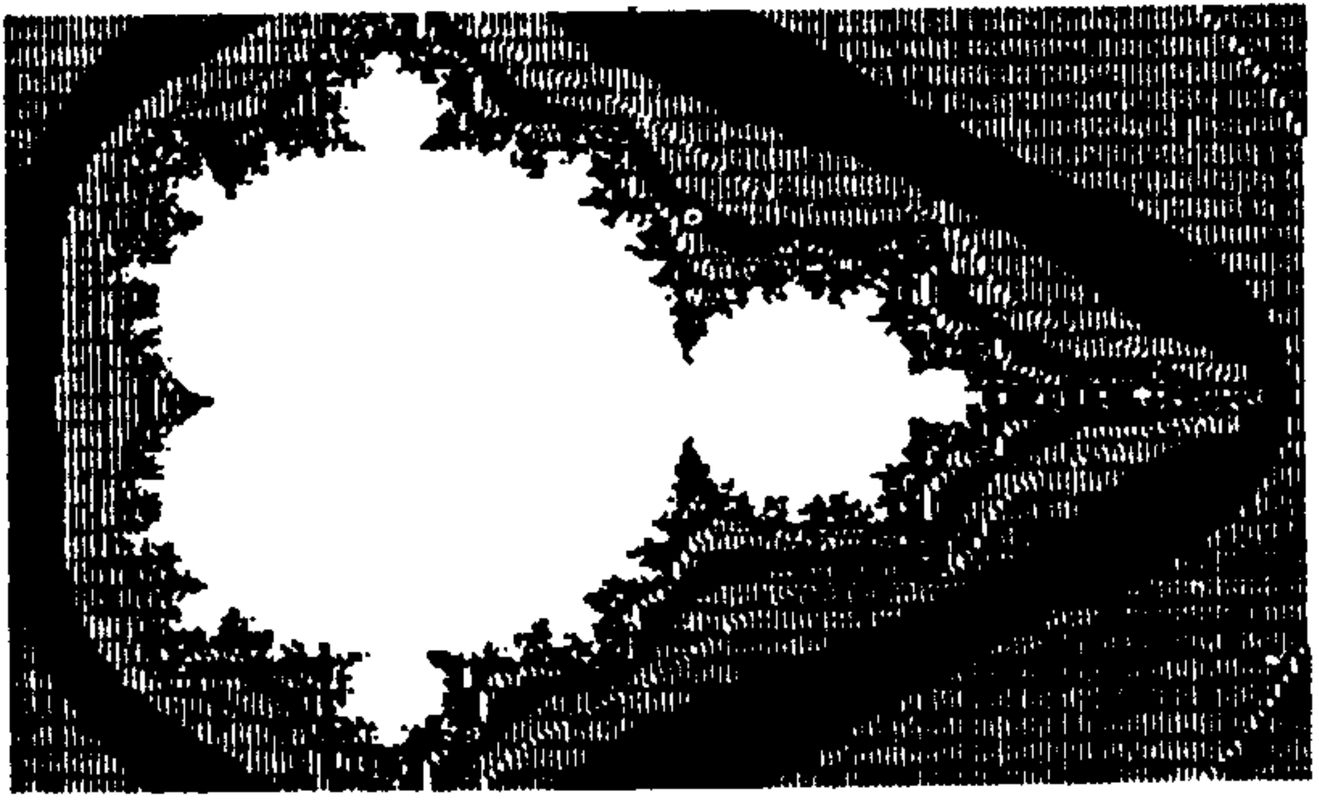
14. ábra

```
100 !grafikus kepernyore villogo szines szoveg
110 GRAPHICS 256
120 LET J=0
130 FOR X=0 TO 26 !sinusgorbe
140   LET V=45*X+50:LET F=240+240*SIN(X*PI/12)
150   SET INK RND(255):PLOT V,F;
160 NEXT
170 PLOT V,F,
180 LET A$="ENTERPRISE128"
190 FOR X=0 TO 12 !a$ hullamvonalu kiirasa
200   LET F=400+230*SIN(X*PI/6)
210   LET S=X+1
220   IF J=1 THEN LET S=RND(250)+1
230   SET INK S:PLOT 90*X,F,
240   PRINT #101:A$(X+1:X+1)
250 NEXT
260 LET J=1
270 GOTO 130
```

66. program

A számítógépes grafika kedvenc alakja a 15. ábrán látható.

Mandelbrot-féle almaemberke. Az előállítására szolgáló 67. program futása bizony több órát vesz igénybe. A futás idejére az itterációk száma (K) döntő hatással van, de vagy gyorsan futó, vagy szépen tagolt ábrát biztosító megoldást eredményez az értékének megváltoztatása.



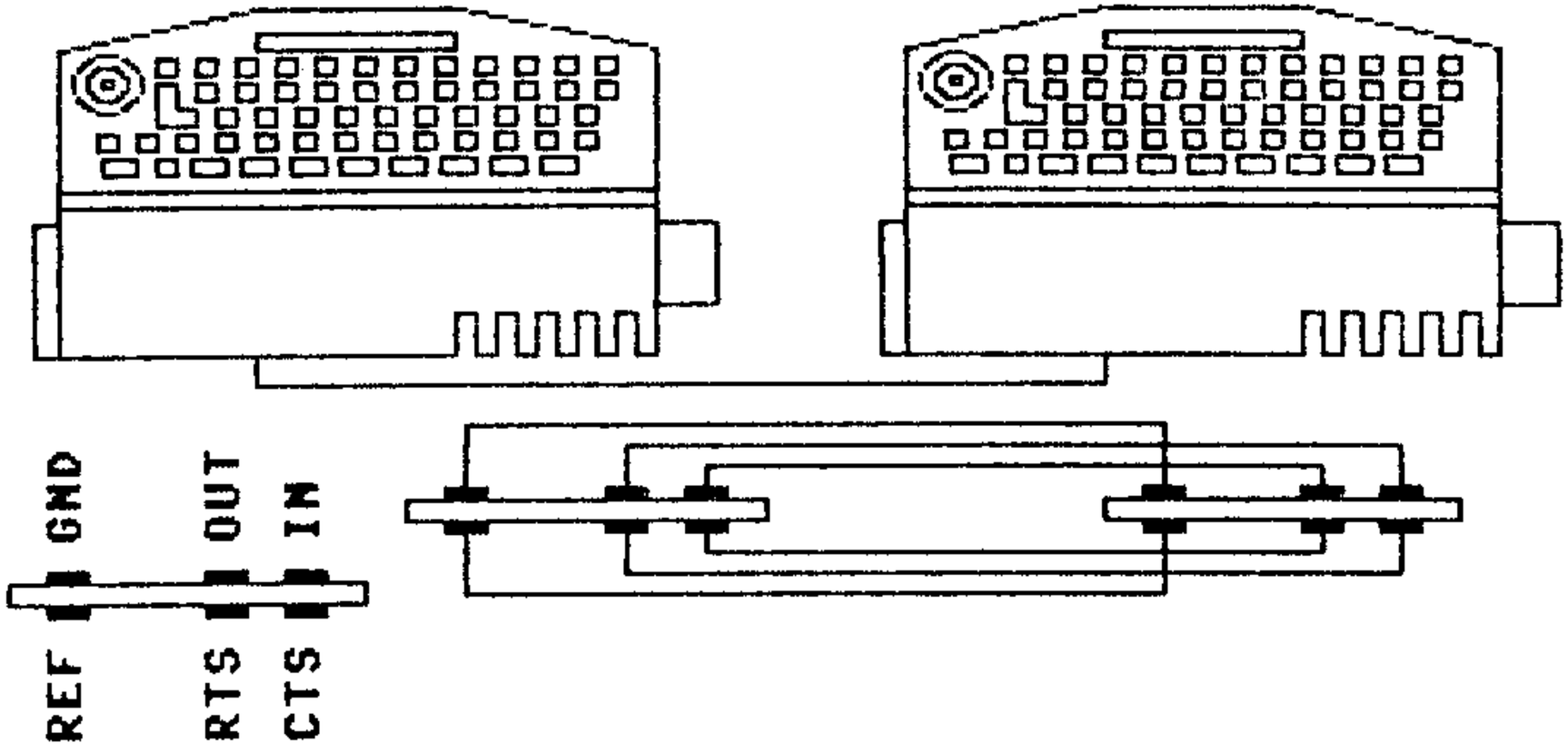
15. ábra

```
100 ! MANDELBROT szines
110 GRAPHICS HIRES 16
120 SET PALETTE 0,4,8,16,32,64,128,255
130 LET XA=-.8:LET XF=2.2
140 LET YA=-1.2:LET YF=1.2
150 LET DX=(XF-XA)/1200:LET DY=(YF-YA)/700
160 FOR M=350 TO 0 STEP-4
170   FOR N=0 TO 1200 STEP 8
180     LET K=0:LET XZ=0:LET YZ=0
190     LET XC=XA+N*DX:LET YC=YA+M*DY
200     LET XX=XZ*XZ:LET YY=YZ*YZ
210     LET YZ=2*XZ*YZ-YC:LET XZ=XX-YY-XC
220     LET R=XX+YY:LET K=K+1
230     IF R<4 AND K<70 THEN GOTO 200
240     SET INK INT(K/10)
250     PLOT N,M:PLOT N,704-M
260   NEXT
270 NEXT
```

67. program

Hálózat

Igen egyszerű huzalozással össze lehet kapcsolni két vagy több (akár 32 számítógépet). Maradjunk mi a házilag valószínű kiépítésnél, amikor két számítógépet kapcsolunk össze. Az összeköttetést a 16. ábra szerint készíthetjük el.



16. ábra Két gép összekapcsolása

A csatlakozó kábel házi elkészítéséhez tudni kell, hogy a csatlakozó osztástávolsága 2,54 mm, az összekötő kábel pedig árnyékolt 5 eres legyen. Hogy mire jó az ilyen összekapcsolás, azt kipróbálhatjuk játék szinten, programfejlesztés és programírás közben is. Programfejlesztés, programírás közben az egyik gépben írt program a másikba tölthető SAVE paranccsal. A hálózati gépkijelöléseket természetesen a Felhasználói kézikönyv alapján el kell végezni. Nem említi a kézikönyv, hogy minden olyan parancs, utasítás, ami a magnetofonra kiadható, használható a két számítógép között is. Lehetőség van programok összefűzésére (MERGE), de listázhatjuk (LIST) az egyik számítógép programját a másik számítógép képernyőjére.

Az adatforgalomnál, adat átküldésénél óvatosan kell eljárni. A két gépet összhangba kell hozni – még Basic szinten is – mert, ha oda-vissza adatforgalom van, akkor elképzelhető, hogy utolérik egymást a programok. Ilyenkor azután egymásra várnak. A két gép közti adatforgalomra, az egyik gépből kapott információ másik gép általi felhasználására mutat példát a 68. és 69. program.

```
100 ! ez talalja ki
110 !ezt a programot kell masodikkent inditani!!!!!!!!
120 SET INTERRUPT NET OFF
130 SET NET NUMBER 1
140 OPEN #1:"net:"
150 LET ALSO=0:LET FELSO=100 !a szamtartomany 1 es 100
kozott
 160 LET TIPP=INT((ALSO+FELSO)/2)
 170 LET A$=CHR$(TIPP)
 180 PRINT "a tippem:";TIPP
 190 PRINT #1:A$; !halozatra kuld
 200 FLUSH #1 !    csatornat urit
 210 GET #1:VALASZ$
```

```
  220 IF VALASZ$="" THEN 210 !addig var amig a 2.geptol
valasz nem erkezik
  230 LET VALASZ=ORD(VALASZ$)
  240 ON VALASZ GOTO 260,270,280
  250 END
  260 LET ALSO=TIPP:GOTO 160
  270 LET FELSO=TIPP:GOTO 160
  280 PRINT "hurra eltalaltam a szam:";TIPP
```

68. program

```
100 !ez gondol
110 !ezt a programot kell elsokent inditani
120 SET INTERRUPT NET OFF
130 SET NET NUMBER 2
140 OPEN #1:"net:"
150 RANDOMIZE
160 LET SZAM=RND(100):LET X=1
170 PRINT "gondoltam, talald ki";SZAM
180 GET #1:TIPP$
190 IF A$="" THEN 180 !az 1.gep tippjere var
200 LET TIPP=ORD(TIPP$)
210 PRINT X;". tipped:";TIPP,
220 LET X=X+1
230 IF TIPP=SZAM THEN LET VALASZ$=CHR$(3)
240 IF TIPP<SZAM THEN
250   LET VALASZ$=CHR$(1)
260   PRINT "ez keves"
270 END IF
280 IF TIPP>SZAM THEN
290   LET VALASZ$=CHR$(2)
300   PRINT "ez sok   "
310 END IF
320 PRINT #1:VALASZ$;
330 FLUSH #1 !valaszt kikuld az elso gepnek
340 IF TIPP<>SZAM THEN GOTO 180
350 PRINT "ezaz!"
```

69. program

Amikor két gép dolgozik egymástól kapott információval, akkor általában két különböző programot kell a gépekbe tölteni. Az ötletként bemutatott programok külön-külön valósítják meg a Basic tanfolyamok alapprogramjaként ismert számkitalálós játékot. Ennek a programnak két verziója van. Az egyik, amikor a számítógép „gondol" egy számot és azt kell a gép előtt ülőnek kitalálnia. A másik lehetőség, hogy felkészítjük a gépet arra, hogy az általunk gondolt számot kitalálja. A két stratégiát a két összekapcsolt számítógépbe is betölthetjük. A tippet pedig nem mi adjuk billentyűzetről, hanem majd a másik számítógép fogja a számkitaláló program adta tippeket küldeni a számot „gondoló" gépnek. Természetesen mint ahogy két ember játéka közben, itt is illik kisegítő információt adni a számkitalálással küszködő partnernek. Tehát a tippet küldő gép „joggal vár" visszajelzést a számot „gondolótól".

Kössük össze a számítógépeket, majd helyezzük feszültség alá a rendszert! Elég az egyik számítógéphez háttértárat, magnetofont kapcsolni.

Jelöljük ki a számítógépeket a hálózat első és második gépének! (SET NET NUMBER 1) (SET NET NUMBER 2) Gépeljük be, vagy töltsük be magnetofonról az első számítógépbe a 69. programot! Ezután töltsük a programot a második számítógépbe. (SAVE "NET–2:", LOAD "NET–1:") (Természetesen dolgozhatunk párhuzamosan is, ilyenkor mindenki a saját gépébe a saját programját írja be.) Ha az egy gépen történő programírást végezzük, akkor az először betöltött program áttöltése után beolvashatjuk a 68. programot az 1. számú gépbe.

A játék úgy kezdődik, hogy elindítjuk azt a gépet, amelyikben az a program van, mely a kitalálandó számot állítja elő. Ha ennek a 2. számú gépnek a képernyőjén megjelenik a felszólítás, hogy találja ki a számot, akkor elindíthatjuk at 1. számú, számkitaláló stratégiával rendelkező számítógép programját. A tippet GET utasítás olvassa be. A két program szinkronban futását az biztosítja, hogy a beolvasást mindaddig ismétli, amíg a tipp be nem érkezik. A beérkezett tippet a számot „gondoló" program kiértékeli és attól függően, hogy ennek értéke hogyan viszonylik a kitalálandó számhoz segítő, vagy nyugtázó információt küld vissza a tippet küldőnek. Az üzenetek és tippek a számítógépek képernyőin láthatók. Amennyiben az indítási sorrendet elrontjuk, akkor mindkét program beolvasási fázisban várakozik. Ebből az állapotból csak a programok leállítása és helyes sorrendű újraindításával lehet kikerülni.

HEX$

Eredendően egy gépi kódú program byte sorozatát lehet vesszővel elválasztva tárolni e függvény segítségével. Az ALLOCATE utasítással lefoglalt memóriaterületre kerülnek a függvényben tárolt byte-ok.

A megfelelő címre letárolt, eredendően hexadecimális formában megadott értékeket PEEK segítségével visszaolvashatjuk az adott memóriacímről. Ezt felis-

merve készíthetünk a HEX$ függvény segítségével olyan programot, mely a hexadecimális számot decimálisra váltja át. A megoldás trükkje:

DEC=ORD [(HEX$ (H$)]

A felhasználható programot pedig a következő szempontok alapján készíthetjük el:

- INPUT karaktersorozattal beírhatjuk a hexadecimális számot
- biztos ami biztos, gondoskodjunk arról, hogy a karakterek nagybetűk legyenek, ámbár a HEX$ elég intelligens végrehajtó rutinnal rendelkezik ahhoz, hogy helyesen értelmezze a betűket. (A rutin az 1. szegmens C6EF címen kezdődik.)
- ellenőrizzük, hogy a karakterek valóban hexadecimális számjegyként értelmezhetők-e. (Ez a vizsgálat is megtalálható a gépi kódú végrehajtó rutinban.)
- két karakterenként leszeletelve a H$-et a helyi értéknek megfelelően figyelembe vehetjük annak decimális értékét.

A 70. program a konkrét megoldást mutatja be.

```
100 ! hex -> dec
110 INPUT PROMPT "HEX szam: ":H$
120 LET H$=UCASE$(H$) !nagybetusiti a H$ karaktereit
 130 LET J,D=0
 140 FOR V=1 TO LEN(H$) !karakterek vizsgalata
 150   LET V$=H$(V:V)
 160   IF V$>"F" OR V$<"0" OR(V$>"9" AND V$<"A") THEN
LET J=1
 170 NEXT
 180 IF J=1 THEN GOTO 110 !ha hex szamjegykent nem
 ertelmezheto karaktert talalt
 190 IF LEN(H$)/2<>INT(LEN(H$)/2) THEN LET H$="0"&H$
 200 FOR I=1 TO LEN(H$) STEP 2
 210   LET D=256*D+ORD(HEX$(H$(I:I+I))) ! HEX$ szele-
telese, es decimaliskent valo ertelmezese
 220 NEXT
 230 PRINT H$;"H  = ";D;"D"
 240 PRINT :PRINT
 250 GOTO 110
```

70. program

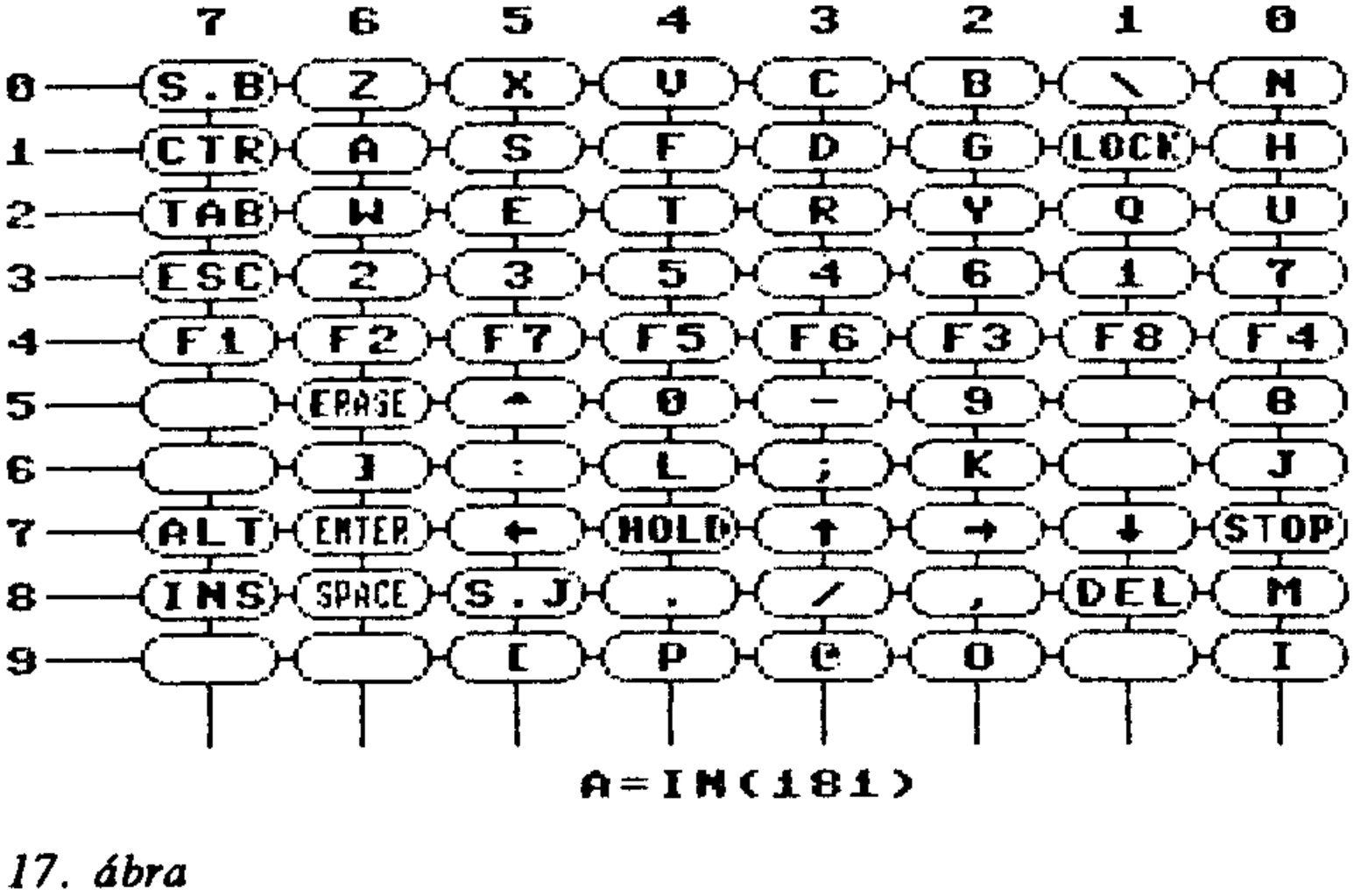
IN (n)

Ez a függvény az argumentum által meghatározott portról beolvas egy értéket. A 181dec port a billentyűzetet olvassa be. Korrekt beolvasást gépi kódú rutinnal végezhetünk, de Basic-ből is lekérdezhetjük a billentyűzetet. A billentyűzet beolvasható az INKEY$ függvénnyel, ezt a függvényt felhasználhatjuk, pl. billentyűgyakorlást elősegítő program írására (71. program).

```
100 ! betu gyakorlo
110 RANDOMIZE
120 SET VIDEO Y 2
130 SET VIDEO COLOUR 3
140 SET VIDEO MODE 1
150 OPEN #10:"video:"
160 CLEAR SCREEN
170 LET R=10000:LET X=1:LET A=0:RANDOMIZE
180 SET #10:INK 255
190 PRINT #10:CHR$(26) !kepernyotorles
200 LET A$=CHR$(33+RND(90)) !veletlen karakter eloalli
tasa
210 PRINT AT 8,20:A$
220 PLOT #10:570,71,:PRINT #10:A$
230 DISPLAY #10:FROM 1 TO 2 AT 10
240 LET S=0
250 LET V$=INKEY$:LET S=S+1:LET B=S*.14
260 PRINT AT 2,21:"Ez     ";B;"   "
270 IF V$<>A$ THEN 250
280 IF B<R THEN LET R=B
290 PRINT AT 2,1,USING "Rekord###.#":R
300 LET A=A+B
310 PRINT AT 3,1,USING "atlag ###.#":A/X
320 LET X=X+1
330 GOTO 180
```

71. program

Játékprogramokban is jól felhasználható egy billentyű lenyomásának figyelésére az INKEY$. Ezzel a függvénnyel viszont csak egy billentyű lenyomását lehet érzékelni. Adódhat olyan eset, amikor több billentyű együttes lenyomását kell kiértékelni. Ezt az együtt, több billentyűt figyelő funkciót az IN függvény



17. ábra

segítségével valósíthatjuk meg. A billentyűállapot lekérdezéséhez ismerni kell a billentyűmátrixot. Ezt a 17. ábra mutatja be.

A 181dec (B5H) port Ø–3. bitje a billentyűmátrix sorkiválasztását teszi lehetővé. A sorkiválasztás után IN-nel beolvasott érték az adott sor Ø–7. bitjének alakulásától függő értéket ad eredményül. A billentyű lenyomásával érintett bit értéke nullába billen. Így az egy sorban lévő karakterek együttes lenyomásával a megfelelő adatbitek értéke az egész sorról beolvasható értéket együttesen módosítják. A billentyűmátrix kiismerésére szolgáló 72. programot célszerű többször lefuttatni, mert az angol, ill. német verziójú gépeken jelentős eltérés van.

```
100 !billentyumatrix kiismerese
110 INPUT PROMPT "sor kijeloles: ":Y
120 IF Y<0 OR Y>9 OR Y<>INT(Y) THEN GOTO 110
130 OUT 181,Y !sorkijeloles 181.porton
140 LET A=IN(181) !beolvasas
150 IF A=255 THEN GOTO 130 ! ha nincs billentyu
lenyomva
160 PRINT A
170 GOTO 130
```

72. program

INPUT

Esendő adatbevitelt biztosító utasítás. Általában billentyűzetről, de magnetofonról, hálózatról is használhatjuk. Ellentétben a GET-tel meg ne kíséreljük a képernyőt kijelölni Input csatornaként, mert ilyenkor csak a RESET segít. Az

utasításban jelölt változótípus vár. Ha numerikus változónevet adtunk meg és betűt akarunk beírni, akkor a program hibajelzéssel leáll. Ellentétben pl. a Microsoft Basic-kel, nem ad lehetőséget új adatbevitelre. Amennyiben karaktersorozatot kell kapnia és az vesszőt tartalmaz, akkor alapértelmezésben a vessző és az utána következő karakterek elvesznek. Ha mégis szükség van a vesszőre vagy a karaktersor elején lévő szóközökre, akkor az adatokat macskakörmök között, vagy a LINE INPUT utasítás alkalmazásával kell beírni. Ez utóbbi esetben minden karaktert, még szöveg eleji macskakörmöt is tartalmazhat a beírt szöveg.

Amennyiben ez a lehetőség is kevés és a program futását minden szempontból a felhasználótól függetlenül, biztosnak akarjuk tudni, akkor ellenőrzött INPUT rutint kell írni, hiszen nem biztos, hogy egy hiba elfogó rutinnal a később fellépő hibát korrigálni tudjuk. Ilyen adatbeírás közbeni ellenőrzési szempontok lehetnek, hogy csak számokat vagy csak bizonyos karaktereket tartalmazó, vagy csak adott hosszúságú karaktersorozatot fogadjon el a program. Az ellenőrzéseket bizonyos esetekben az INPUT utasítást tartalmazó sor után is elvégezhetjük.

Itt most nézzünk egy csokorra való ellenőrzött adatbeolvasást biztosító programot! A programok közös jellemzője, hogy a további feldolgozásra átadott adatot karakterenként olvassuk be (GET, INKEY$). Természetesen az ilyen módosított beolvasó rutinnak is (majdnem) úgy kell viselkednie mint az eredetinek. Tehát az adatok karaktereit látni kell, azokat törölni is tudni kell. Az adatbevitel végét az ENTER jelzi, az sem lenne baj, ha valami kurzor, kurzorféle is látható lenne az adatbevitel sorában.

Csak az ábécé kisbetűit fogadja el a 73. program ellenőrzött INPUT rutinja.

```
 100 ! ellenorzott INPUT
 110 LET HOSSZ=15:LET E$="a":LET U$="z" ! beolvasasi
korlatok
 120 LET X=10:LET Y=10 !kiiras helye
 130 STRING Q$
 140 CLEAR SCREEN
 150 CALL ELL_INPUT(15,"a","z",X,Y)
 160 PRINT Q$
 170 END
 180 DEF ELL_INPUT(HOSSZ,E$,U$,X,Y)
 181    PRINT "a rutin elfogad"
 182    PRINT "max:";HOSSZ;"karaktert"
 183    PRINT E$;"-";U$;" tartomanyban"
 190    LET Q$=""
 200    DO
 210       PRINT AT Y,X:Q$;"? "
 220       LET A$=INKEY$:LET J=0 !karakterenkent beolvas
```

```
 230      IF A$=CHR$(13) THEN EXIT DO ! ENTER eseten
beolvasas vege
 240      IF LEN(Q$)>0 AND A$=CHR$(164) THEN !vissza-
torolhetoseg ellenorzese
 250        LET Q$=Q$(1:LEN(Q$)-1):LET J=1
 260      END IF
 270      IF J=1 THEN 210
 280      IF LEN(Q$)>=HOSSZ OR A$<E$ OR A$>U$ THEN 210
!karakter elfogadhatosag vizsgalata
 290      LET Q$=Q$&A$ !karakter hozzafuzese
 300    LOOP
 310    PRINT AT Y,X:Q$;"   "
 320 END DEF
```

73. program

A 74. program csak számjegyeket, ill. a számsor elején (és csak ott) fogad el mínusz előjelet.

```
100 ! ellenorzott INPUT
110 LET X=10:LET Y=10 !kiiras helye
120 STRING S$
130 CLEAR SCREEN
140 CALL ELL_INPUT
150 PRINT S$
160 END
170 DEF ELL_INPUT
180   LET S$="":LET TP=0:LET EJ=0
190   PRINT "a rutin csak szamot fogad el"
200   DO
210     PRINT AT Y,X:S$;"? "
220     LET A$=IŃKEY$:LET J=0 !karakterenkent beolvas
230     LET H=LEN(S$)
240     IF A$=CHR$(13) THEN EXIT DO ! ENTER eseten
beolvasas vege
250     IF H>0 AND A$=CHR$(164) THEN ! vissza-
torolhetoseg ellenorzese
260       IF S$(H:H)="." THEN LET TP=0
270       IF S$(H:H)="-" THEN LET EJ=0
280       LET S$=S$(1:H-1):LET J=1
```

```
290      END IF
300      IF J=1 THEN 210
310      IF A$="." AND TP=1 THEN 210
320      IF A$="-" AND TP=1 THEN 210
330      IF A$="-" AND S$<>"" THEN 210
340      IF A$="." THEN LET TP=1 !tizedespont jelzo
350      IF A$="-" THEN LET EJ=1 !elojel jelzo
360      IF A$="." OR A$="-" THEN 380
370      IF A$<"Ø" OR A$>"9" THEN 210
380      LET S$=S$&A$ !karakter hozzafuzese
390    LOOP
400    PRINT AT Y,X:S$;" "
410 END DEF
```

74. program

A beírás során csak egy tizedes pontot írhatunk be az input adatba. Itt tehát a programnak figyelnie kell, hogy hol érkezett és érkezett-e már egy adott karakter. Természetesen azt is észre kell vennie a programnak, ha javítási szándékkal időközben kitöröltük pl. a tizedes pontot.

A csak számjegyeket és pontosan 11 karakterből álló adat elfogadását a MOD-nál is bemutatott személyi szám ellenőrző program, egy más megvalósításán keresztül figyelhetjük meg a 75. program BEVITEL blokkjában.

```
100 ! szemelyi szam  ellenorzott inputtal
110 DIM H$(12),N(12)
120 CLEAR TEXT
130 PRINT AT 8,15:"12345678901"
140 LET H=Ø:LET SUMMA=Ø:LET S$=""
150 CALL BEVITEL
160 FOR I=1 TO 12
170   READ H$(I),N(I)
180   IF I<11 THEN LET SUMMA=SUMMA+VAL(S$(I:I))*I
190 NEXT
200 LET V=VAL(S$(1:1))
210 IF V<1 OR V>8 THEN PRINT "elso karakter hibas  ":
GOTO 470
220 LET A$="nem magyar ":LET B$="ferfi ":LET E$="19"
230 IF V<5 THEN LET A$="magyar "
240 IF V/2=INT(V/2) THEN LET B$="no "
250 PRINT "az illeto: " A$ "allampolgarsagu " B$
```

```
260 IF (V-1) BAND 2 THEN LET E$="18"
270 PRINT "szuletesi ido: " E$ S$(2:3);" ";
280 LET V=VAL(S$(2:3))
290 IF V/4=INT(V/4) THEN LET N(2)=29
300 LET V=VAL(S$(4:5))
310 IF V<1 OR V>12 THEN
320   PRINT "hibas honap adat":GOTO 470
330 ELSE
340   PRINT H$(V);" ";
350 END IF
360 IF N(V)>=VAL(S$(6:7)) THEN GOTO 380
370 PRINT "hibas honap, nap egyeztetes":GOTO 470
380 PRINT S$(6:7)
390 PRINT "napi sorszam: " S$(8:10)
400 PRINT "ellenorzo osszeg: ";
410 LET ELL=MOD(SUMMA,11)
420 IF ELL=VAL(S$(11:11)) THEN
430   PRINT S$(11:11)
440 ELSE
450   PRINT "hibas"
460 END IF
470 PRINT :PRINT "bill. nyomasra indul"
480 IF INKEY$="" THEN GOTO 480
490 RUN
500 DEF BEVITEL
510   PRINT "szemelyi szam:"
520   PRINT AT 9,15:S$;"? "
530   LET Q$=INKEY$:LET H=LEN(S$)
540   IF Q$=CHR$(13) AND H=11 THEN GOTO 600
550   IF H=11 AND(Q$<>CHR$(164) AND Q$<>CHR$(184))
THEN GOTO 530
 560   IF H>0 AND(Q$=CHR$(164) OR Q$=CHR$(184)) THEN
LET S$=S$(1:H-1)
 570   IF Q$>="0" AND Q$<="9" THEN LET S$=S$&Q$ !
 csak szamot fogad el
 580   PRINT AT 9,15:S$;"  "
 590   GOTO 520
 600   PRINT AT 9,15:S$;"  ":PRINT
```

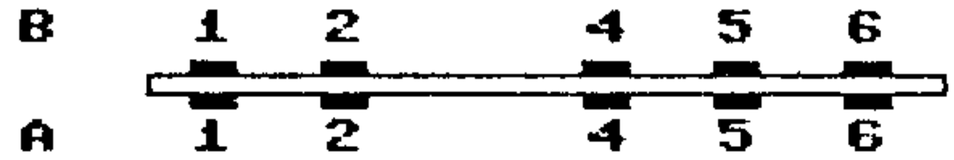
```
610 END DEF
620 DATA januar,31,februar,28,marcius,31
630 DATA aprilis,30,majus,31,junius,30
640 DATA julius,31,augusztus,31,szeptember,30
650 DATA oktober,31,november,31,december,31
```

75. program

Ez a megoldás a személyi szám további elemzése során egyszerűsíti a vizsgálatokat.

JOY

A számítógép beépített botkormánya a kurzor mozgatására szolgál. A nagyobb igénybevételnek ne tegyük ki. A játékokhoz jobb a külső – erőteljesebb kialakítású – botkormányt használni. Ha nem a saját botkormányt csatlakoztatjuk, hanem valamilyen más típusút, akkor ismernünk kell az Enterprise CONTROL port csatlakozójának kiosztását. A bekötés elvégzéséhez segíthet a 18. ábra.



B1	GND	A1	GND
B4	+5	A4	JOBBRA
B5	BALRA	A4	LE
B6	FEL	A4	TÜZ

18. ábra Botkormány csatlakoztatása

A saját botkormányt programfutás közben különböző célra használhatjuk fel, állapotát többféle módon kérdezhetjük le. A beépített botkormány állapota lekérdezhető. A=JOY (Ø) értékadó utasítással. Minthogy a botkormány a billentyűzethez tartozik, a billentyű lekérdezhető INKEY$-vel is, ahol most a botkormány állását A$-be olvashatjuk be. De a billentyűzet elérhető a portról is 181dec (B5H) címen.

A botkormány állapotot a billentyűmátrix 7. sorából beolvasott 1., 2., 3. és 5. bit jelzi:

A különböző módon beolvasott botkormány állások:

	JOY (Ø)	A=ORD (A$)	IN(181)
É/FEL	8	176	247
ÉK		–	243
K/JOBBRA	1	188	251
DK	5	–	249
D/LE	4	18Ø	253
DNY	6	–	221
NY/BALRA	2	184	223
ENY	1Ø	–	215
TŰZ	16	32	

Ezek felhasználásával egyszerű rajzoló programot írhatunk. A program tegye lehetővé a pont mozgatását a botkormány által kiválasztható nyolc irány bármelyike felé egészen a képernyő határáig, de a pont ne tudjon „lemászni" a képernyőről. A botkormány és a szóköz billentyű egyidejű lenyomása mellett az aktuális pozíción lévő pont radírozható legyen. A kitűzött feladat megoldására mutat egy lehetséges megoldást a 76. program.

```
1ØØ ! botkormannyal rajzol
11Ø GRAPHICS :LET X=64Ø:LET Y=34Ø
12Ø SET KEY CLICK OFF
 13Ø SET STATUS OFF
 14Ø SET PALETTE BLACK,WHITE
 15Ø PLOT X,Y;
 16Ø LET A=JOY(Ø)
 17Ø IF (A=2 OR A=6 OR A=1Ø OR A=18 OR A=22 OR A=26)
AND X>Ø THEN LET X=X-4
 175 IF (A=1 OR A=5 OR A=9 OR A=17 OR A=21 OR A=25)
AND X<12ØØ THEN LET X=X+4
 18Ø IF (A=4 OR A=5 OR A=6 OR A=2Ø OR A=21 OR A=22)
AND Y>Ø THEN LET Y=Y-4
 185 IF (A=8 OR A=9 OR A=1Ø OR A=24 OR A=25 OR A=26)
AND Y<71Ø THEN LET Y=Y+4
 19Ø SET INK(-1*(A<15)) !szokoz billenytuvel egyutt
radiroz
 2ØØ IF INKEY$="w" THEN RUN  !ujrainditas
 21Ø GOTO 15Ø
```

76. program

A 17Ø–185. sorok hatása a LET-nél tárgyalt értékadási lehetőség felhasználásával egyszerűbben is megoldható (77. program).

```
1ØØ ! botkormannyal rajzol
11Ø INPUT PROMPT "kezdopont x,y :":X,Y
12Ø SET STATUS OFF
13Ø GRAPHICS
14Ø SET PALETTE BLACK,WHITE
15Ø PLOT X,Y;
16Ø LET A=JOY(Ø)
17Ø !
 175 LET X=X+4*(A BAND 2)*(X>Ø)-8*(A BAND 1)*(X<12ØØ)
 18Ø LET Y=Y+4*(A BAND 4)*(Y>Ø)-(A BAND 8)*(Y<71Ø)
 185 !
 19Ø SET INK(-1*(A<15)) ! szokoz billenytuvel egyutt
radiroz
 2ØØ IF INKEY$="w" THEN RUN  !ujrainditas
 21Ø GOTO 15Ø
```

77. program

A botkormány használatának egy igen hasznos és elegáns alkalmazási lehetősége a menütechnikára épülő programok egy-egy alprogramjának kijelölését biztosító alkalmazás. A 78. program lehetővé teszi, hogy a választási lehetőségeket kiíró listából a botkormány és a szóköz billentyű segítségével válasszuk ki a kívánt menüpontot.

```
100 ! menubol valaszt botkormannyal
110 NUMERIC AA
120 CALL MENU
130 CALL VALASZTAS
140 ON AA GOTO 160,170,180,190,200,210,220,230
150 END
160 PRINT "  1. programresz":STOP
170 PRINT "  2. programresz":STOP
180 PRINT "  3. programresz":STOP
190 PRINT "  4. programresz":STOP
200 PRINT "  5. programresz":STOP
210 PRINT "  6. programresz":STOP
220 PRINT "  7. programresz":STOP
230 PRINT "  8. programresz":STOP
```

```
240 DEF MENU
250   SET STATUS OFF
260   CLEAR SCREEN
270   FOR Y=1 TO 8
280     PRINT AT Y*2+3,8:Y ".   lehetoseg"
290   NEXT
300 END DEF
310 DEF VALASZTAS
320   LET K=5:LET K$=">"
330   LET P=JOY(0) !menuben kijeloles botkormannyal
340   IF P>15 THEN 410 !szokoz billenytuvel kivalaszt
350   IF P=8 AND K>4 THEN PRINT AT K,7:" ":LET K=K-2
360   IF K<5 THEN LET K=19 !kijeloles "korbe forog"
370   IF P=4 AND K<20 THEN PRINT AT K,7:" ":LET K=K+2
380   IF K>20 THEN LET K=5
390   PRINT AT K,7:K$
400   GOTO 330
410   LET AA=(K-3)/2
420   CLEAR SCREEN
430   PING
440 END DEF
```

78. program

A botkormánnyal felfelé-lefelé mozgathatjuk a kijelölést láthatóvá tevő jelet. Ha megnyomjuk a szóköz billentyűt, akkor a program a választásnak megfelelő programágra ugrik az ON választás GOTO S1, S2, S3 utasítás hatására.

LET

A változók értékadása az IS–BASIC-ben mindig kötelező. Nincs alapértelmezésű értéke semmilyen változótípusnak sem. Szerencsés lehetőséget biztosít az egy értékadó utasításban több változónak azonos értéket adó LET A, B, C=Ø alakú utasítás. Ha azonban olyan eset adódik, amikor nagyon sok különböző nevű változónak kell értéket adni, akkor ez az eljárás nehézkes. Amennyiben az összes nagybetűs változónevet fel akarjuk használni, akkor a 79. program szerint olyan megoldást követhetünk, melynél a program elején az értékadó utasításban a változó nevét felülírva minden felhasznált változónak azonos értéket adunk.

```
100 FOR WW=65 TO 91
110   LET A=WW-1
112   !   ^ ez a valtozonev most a 4854. memoria-
cimen van
120   POKE 4854,WW
122   PRINT WW;
130 NEXT
140 PRINT A;B;C;D;E;F;G;Z
150 ! vigyazz ertekadas programfelulirassal!
```

79. program

Ehhez a trükkhöz így két dolgot nagyon alaposan végig kell gondolni.

A ciklusban történő értékadásnál, a listában most látható „A" változónév memóriabeli elhelyezkedését meg kell határozni, hiszen ha POKE-kal hozzányúlunk egy programsorhoz, akkor végzetes hibát is elkövethetünk. Az értékadást természetesen akár kétbetűs változónevek minden variációjára is elvégezhetjük a 80. program felhasználásával, de talán ilyen esetben már inkább (sőt lehet, hogy az előző példánál is) tömbváltozókat illik használni.

```
100 FOR I=65 TO 90
110   FOR J=65 TO 90
120     LET AA=0
 130    !   ^ ez a karakter most 4872 memoriacimen
van
 140    !    ^ ez pedig most 4873 memoriacimen van
 150    POKE 4872,I
 160    POKE 4873,J
 170   NEXT
 180 NEXT
 200 ! ketbetus valtozok alapertelmezese program
felulirassal
 210 ! osszesen  676 valtozonak ad ertcket
```

80. program

Ennek ellenére nézzük meg a 81. programban, hogy az ABC minden betűjével jelzett stringnek is lehet ilyen módon alap, vagy konkrét értéket adni.

```
100 FOR I=65 TO 91
110   LET A$="samu"
120   POKE 4853,I
130 NEXT
```

```
140 FOR I=65 TO 90
150   POKE 4938,I
160   PRINT CHR$(I);"$ =";A$
170 NEXT
180 !idaig ugyanigy beirni ! ket helyen is felul-
irjuk a programot
190 ! futtatas elott akar ellenorizhetjuk a memoria-
cimeket es tartalmukat  RUN 200-zai
200 FOR I=4827 TO 5000
210   PRINT I;PEEK(I)
220 NEXT
```

81. program

Ezt ugyan már láttuk, de ebbe a programban a változókat hasonló módon ki is listázzuk. Ebben az esetben már két helyen is benyúltunk POKE utasítással az eredeti programba. A programtárolás szabályait ismerők a program végén lévő ciklussal győződhetnek meg az átírandó memóriacímek helyességéről. Ezért ha beírtuk a programlistát, először RUN 180-nal indítsuk azt el és ha a címeink helyesek, akkor próbáljuk ki élesben azt. A feltételhez kötött értékadásnál használhatunk IF utasítást, pl.: IF V=1 THEN LET A=3.

Van egy elegánsabb, de esetenként 10%-kal lassabb végrehajtási idejű értékadási lehetőség is, az előző példánál maradva:

LET A=—3 * (V=1)

Ezzel az értékadási móddal többször találkozhatunk. A zárójel tartalmát egyenlőség (igazság) szempontjáról ellenőrzi a rendszerprogram. Ha igaz, akkor logikailag igaz és 1 értéket kap a zárójelben lévő kifejezés. Ha nem igaz, akkor értéke 0 lesz.

LINE MODE

A rajzok, színek képernyőn való találkozásának egymás felülírásának módját határozza meg ez a video feltétel. A fekete-fehér képernyőn is látványos képet kaphatunk a vonalak kizáró, vagy (XOR) módban való találkoztatásával. A 82. program sakktáblaszerű beosztást rajzol a képernyőre.

```
100 !racs
110 GRAPHICS
120 SET PALETTE BLACK,WHITE
130 FOR Y=0 TO 719 STEP 120 !vizszintes vonalak
```

```
140   FOR X=0 TO 1279 STEP 4
150     PLOT X,Y;X,Y+60
160   NEXT
170 NEXT
180 SET LINE MODE 3 ! ahol van ott torolj
190 FOR X=0 TO 1279 STEP 128 !fuggoleges vonalak
200   FOR Y=0 TO 719 STEP 4
210     PLOT X,Y;X+64,Y
220   NEXT
230 NEXT
240 FOR R=10 TO 330 STEP 30
250   PLOT 640,360,ELLIPSE R,R
260 NEXT
```

82. program

LIST

Ezt a parancsot gyakran használjuk programunk írása közben. Az elkészült programot nyomtatóra is kiírathatjuk az LLIST paranccsal. Sokszor jó lenne, akár a program futása közben egy-egy programsort kiírni a képernyőre, vagy az elkészült programról több listát készíteni a nyomtatón. A LIST (LLIST) kulcsszót viszont azokhoz a kulcsszavakhoz hasonlóan, melyek típusjelzőjének első bitje Ø, nem veszi fel a tokenizáló rutin a programba, hanem hibajelzést ad. Amennyiben a kulcsszót, mint utasítást be tudjuk csempészni a programba, akkor azt végre fogja hajtani. A 83. programot úgy állíthatjuk elő, hogy először beírjuk a 13Ø PRINT programsort, majd azonnal kézből, parancsmódban végrehajtatjuk a POKE 4832,42 parancsot.

```
100 ! LIST programban
110 FOR I=1 TO 5
120   PRINT I;".lista"
130   LIST
140   PRINT
150   IF INKEY$="" THEN GOTO 150
160 NEXT
```

83. program

Megtehetjük ezt (és csak akkor tehetjük meg, ha NEW után Ø. programban vagyunk), mert az első beírt programsor a 4827dec memóriacímen kezdődik, a kulcsszó sorszáma pedig a tárolt programsor 6. byte-ja. Ott most a PRINT tokenje (56dec) helyezkedik el.

A POKE felülírja az ezen a címen lévő értéket, ha 42dec értéket írunk be, akkor a LIST kulcsszóra.

Ezek után már „felöltöztethetjük" a programot a közölt lista szerint, majd futtathatjuk és végre fogja hajtani most már az így programban beépült LIST utasítást.

Nyilván hasznosabb lenne a nyomtatóra kiírni a programot. Ilyenkor a javasolt megoldás:

Készítsük el azt a programot, amit ki akarunk listázni a nyomtatón. Ha készen van, RENUMBER utasítással hozzuk rendbe a sorszámokat. Írjuk most már elé a 84. programot.

```
  1 POKE 4865,82
  2 FOR I=1 TO 2
  3    PRINT 1ØØ-
  4    LPRINT :LPRINT
  5 NEXT
  6 ! a korabban beirt 1ØØ sorszamon kezdodo program
kilistazasa
1ØØ REM igazi program kezdete
```

84. program

Amennyiben a program egy tetszőleges helyén kívánjuk ezt a trükköt alkalmazni, akkor is hasonló eljárást alkalmazhatunk. Írjuk meg a programot Ø-ként és az ominózus sorba írjuk be a PRINT kulcsszó után a megkívánt LIST, vagy LLIST utáni szintaktikával a kívánt teendőt. Valahogy így:

n PRINT FIRST TO LAST
n PRINT 1ØØ–12Ø
n PRINT eljárás

A teendőnk most az adott sorszámú sor memóriabeli helyének megkeresése és ebben a sorban a 6. byte értékének felülírása

82-re, ha a nyomtatóra
42-re, ha a képernyőre

akarjuk a listát kiíratni. Írjuk be valamelyik szabad program helyére a 85. programot.

```
 1ØØ ! Ø. programban atiras, EZT 2. PROGRAMKENT KELL
BEIRNI !!!
 11Ø INPUT PROMPT "melyik sorban:   ":N
 12Ø INPUT PROMPT "1=LIST  2=LLIST ":V
```

```
 130 IF V<1 OR V>2 THEN 120
 140 LET X=-42*(V=1)-82*(V=2)
 150 LET K=4827 ! a 0. program valtozatlanul ezen a
cimen kezdodik
 160 DO
 170   LET H=PEEK(K) ! a sor elso byte-ja a sor
hosszat adja
 180   IF PEEK(K+1)+256*PEEK(K+2)=N THEN EXIT DO !
megtalalta a keresett sorszamot
 190   IF PEEK(K+H)=0 THEN EXIT DO ! megtalalta a
program veget, de a keresett  sorszamot nem
 200   LET K=K+H ! kovetkezo sor eleje
 210 LOOP
 220 IF PEEK(K+1)+256*PEEK(K+2)<>N THEN GOTO 270
 230 LET CIM=K+5 ! a modositando byte:K->sor hossza,
K+1->skatulyazas melysege,K+2->sorszam also-,
 231 ! K+3->sorszam felsobyte, K+4->kulcsszo bevezeto
(96),K+5-> kulcsszo token
 240 POKE CIM,X
 250 PRINT "elso program modositva"
 260 END
 270 PRINT "ilyen sorszam nincs"
 280 END
```

85. program

Ezt a segédprogramot más célra is felhasználhatjuk. Az inputként közölt programsorszám ismeretében az adott sor memóriabeli elhelyezkedésére máskor is szükség lehet. Igaz a haladók RST 10 rutin segítségével hamarabb is megtalálhatják a keresett sor elejét, mint ahogy azt a GOTO, GOSUB rutinok végrehajtása közben végzi a számítógép rendszerprogramja is. Ez a program most lehetővé teszi, hogy az INPUT utasításként közölhető sorszámú sorban a sorszámot követő első kulcsszó tokenjét lecserélje a LIST, vagy LLIST kulcsszó tokenjére. A módosított 0. programhoz az EDIT 0-val térhetünk vissza.

A programba utólag becsempészett karakterek ravasz dolgok megvalósítását teszik lehetővé. Így pl. megtehetjük azt, hogy beírjuk a 86. programot, mellyel elérhetjük minden kezdő, középhaladó programozó vágyát, hogy saját Basic programját listázás ellen védje.

```
100 ! program titkositas, LIST letiltas
110 POKE 4833,0
120 PRINT "samu"
130 PRINT
140 PRINT
150 PRINT
160 PRINT
170 FOR I=4827 TO 4850
180   PRINT I;PEEK(I)
190 NEXT
```

86. program

Amennyiben a program listázása közben olyan helyen talál nullát az interpreter a programban, ahol azt nem várta, akkor azt programvégnek tekinti. Így csąk az a dolgunk, hogy az első sorban a REM utáni kötelezően szóköz karakter helyére becsempésszük a Ø-t.

Variálhatjuk ezt a védelmet úgy, hogy a sorszám és az első utasítás se látszódjék, ilyenkor a kulcsszó után kurzor visszaléptető karakterek kódját kell beírni annyiszor, hogy e szöveg elé kerüljünk. Erre ad ötletet a 87. program.

```
100 !12345(c) ENTERPRISE!
110 !^ ez 4833 cimen     ^ ez pedig   4852 cimen van
120 !
130 !
140 ! program LIST letiltas, elso sort sem irja ki
150 !
160 !
170 FOR I=4833 TO 4837
180   POKE I,184
190 NEXT
200 FOR I=0 TO 30 STEP .3
210   PRINT TAB(15+14*SIN(I));"ENTERPRISE"
220 NEXT
```

87. program

Az első megjegyzés sorba az öt pont helyére tölti a rövid kis ciklus a CHR$(184) kurzor visszamozgató kódokat, majd a (c) szöveg után egy álprogramvég jelzés kerül a sorvégén lévő felkiáltójel helyére. Így ez a program sem árul többet el magából a listázás hatására. A program természetesen továbbra is él, futtatható és SAVE, LOAD parancsokkal menthető és tölthető. Viszont az egyszerű LIST parancsra nem sokat árul el magáról. A haladók viszont tudják,

hogy becsaphatjuk ezt a lista védelmet úgy, hogy LIST sorszám utasítással átlépjük a védelmet biztosító sort, vagy egyszerűen töröljük ezt a programsort. Hogy megkeserítsük annak az életét, aki ki a akarja törölni az első helyen lévő n sorszámú programsorunkat, készítsünk olyan sorszámot, amit nem lehet átírni. Ez a programsorszám a Ø. Igaz ilyet beírni sem lehet, de ha már egyszer van ilyen sor, akkor az él a programban. Szinte sejteni is lehet már a megoldást. Írjuk be a 88. programot.

```
1ØØ ! marpedig ilyen sorszam nincs
11Ø ! az elso sor sorszama a 4828/29 dec cimen van
12Ø POKE 4828,Ø ! ezt atirjuk Ø. sorszamra
13Ø PRINT "nocsak":PRINT "lassuk a listat:"
14Ø LET VEGE=PEEK(566)+256*PEEK(567) ! a beirt
program vege utanra mutat
15Ø POKE VEGE-3,42 ! az utolso, egy utasitast
tartalmazo programsorban az utasitas tokenjenek atirasa
16Ø PRINT
```

88. program

A program futtatása után megdöbbenve láthatjuk, hogy Ø. sorszámot is tartalmaz a programunk. Ezt a sort sem Ø, sem DELETE Ø, sem a RENUMBER nem bántja. Igaz ezt a sort a DELETE FIRST parancs el tudja intézni. Akkor keressünk erre a listázás tiltásra valami tökéletesebbnek tűnő megoldást.

Mint láttuk a LIST végrehajtását a végrehajtás során a Ø. szegmens elején lévő ugrócímtáblán keresztül biztosítja a rendszerprogram. Írjuk át tehát ezt a címet, legalább annyira, hogy a LIST, LIST sorszám parancsot ne hajtsa végre a számítógép. Ennek egy járható útja, hogy programunkba beírjuk, vagy gépünknek parancsban kiadjuk a POKE 3813, 16 utasítást. Természetesen jó, ha legalább mi magunk tudjuk, hogy visszaállítható az eredeti ugrócímtábla-beli érték és a LIST parancs hatása, ha az alapértelmezést visszaállítjuk POKE 3813, 121 segítségével.

Logaritmus (string matematika)

Ez a függvény az argumentum természetes alapú logaritmusát adja. Ennek a jelölése a matematikában az *ln*. A logaritmus számítás alapszáma az *e*. Ez az érték nem található meg konstansként letárolva a számítógép memóriájában, igaz az EXP (1) függvény kiszámítása után az eredmény az *e*, a természetes logaritmus alapszáma. Ezt az értéket a matematikusok az alábbi formulával definiálták:

$$x = \sum_{i=\emptyset}^{\infty} \frac{1}{i!}$$

Számítógépünk 10 jegy pontossággal tudja kijelezni az értékeket, így nekünk sem kell a végtelenségig folytatni az összegzést. A 89. program 12 lépésben mutatja a közelítés egyre pontosabb értékeit és végül összehasonlítja a számítógép által nyújtott értékkel.

```
100 ! term log alapszama
110 LET E=1:LET F=1
120 FOR I=1 TO 12 !kozelites
130    LET F=F*I
140    LET E=E+1/F
150    PRINT E
160 NEXT
170 PRINT
180 PRINT EXP(1) !kontroll kiiratas
```

89. program

De mi van akkor, ha valamai oknál fogva több tizedesjegy pontossággal kívánjuk kiíratni az *e* értékét? Ilyenkor a string matematikához kell folyamodni. A string matematika alkalmazásával tetszőlegesen sok jegyen tudunk elvégezni összeadást, kivonást. Ez a tetszőlegesen sok jegy, ha az adatokat egy stringben kívánjuk tárolni, természetesen csak a maximálisan ábrázolható, tárolható string hosszával megegyező lehet.

Ha belegondolunk, hogy mit jelent tíz jegy pontosság, akkor láthatjuk, hogy a Föld átmérőjét centiméter pontossággal le tudjuk írni vele. A Föld–Nap távolság milliméter pontossággal való leírásához 26 számjegyet kellene leírni. Ennek ellenére tűzzük ki magunk elé, a harminc jegyig való összeadást.

A 90. program A$ és B$ stringben kell beírnunk az összeadandó számjegy sorozatokat.

```
100 ! string osszeadas max 30 jegyre
110 LET N$="                              "
120 TEXT :LET F=0:PRINT :PRINT :PRINT
130 INPUT PROMPT "1. tag: ":A$
140 LET B$=A$
150 CALL ELEMZES
160 IF F=1 THEN GOTO 120
170 LET F=0
180 INPUT PROMPT "2. tag: ":B$
190 CALL ELEMZES
200 IF F=1 THEN GOTO 180
```

```
 210 LET A$=N$&A$:LET B$=N$&B$
 220 LET A$=A$(LEN(A$)-30:LEN(A$)) !helyi erteknek
megfeleloen egyforma hosszura alakitja a stringeket
 230 LET B$=B$(LEN(B$)-30:LEN(B$))
 240 PRINT AT 10,5:A$
 250 PRINT AT 11,4:"+"&B$
 260 PRINT AT 12,4:"----------------------------"
 270 FOR I=31 TO 1 STEP-1
 280   LET E=VAL(A$(I:I))+VAL(B$(I:I))+F !helyiertek
szerint osszead atvitel figyelembevetelevel
 290   IF E>9 THEN
 300     LET F=1:LET E=MOD(E,10) !ha atvitel van
 310   ELSE
 320     LET F=0 !osszeadasnal atvitel nem keletkezett
 330   END IF
 340   PRINT AT 13,4+I:STR$(E)(1:1)
 350 NEXT
 360 FOR I=4 TO 35
 370   PRINT #102,AT 13,I:; !kepernyon vizsgalat
 380   GET #102:X$
 390   IF X$>"0" THEN EXIT FOR
 400   PRINT AT 13,I:" " ! amig helyi ertekpotlo
nullakat talal azokat torli
 410 NEXT
 420 END
 430 DEF ELEMZES ! csak szamjegyeket tartalmaz a
karaktersorozat ?
 440   LET H=LEN(B$)
 450   IF H>30 THEN LET F=1
 460   FOR V=1 TO H
 470     IF B$(V:V)<"0" OR B$(V:V)>"9" THEN LET F=1
 480   NEXT
 490 END DEF
```

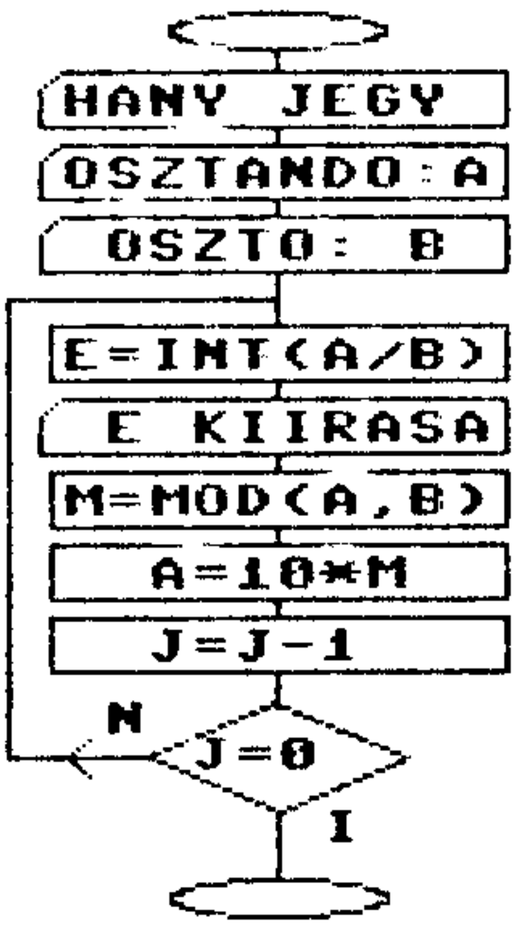
90. program

Az elemzés megvizsgálja, hogy valóban számjegyeket és csak számjegyeket tartalmaznak-e a stringek.

A két összeadandó számjegy sorozatot helyi érték szerint helyesen írja egymás alá a program. Meg kell ezt tenni nemcsak a kiírás kedvéért, hanem azért

is, hogy valóban azonos helyi értékű számjegyeket adjunk össze. A két string egyforma hosszát azzal biztosíthatjuk, hogy az A$, ill. B$ stringek elé ragasztunk harminc helyi érték pótló nullát. Majd a számsor vége felől számolva a harmincadik jegyen túliakat elhagyjuk. Az összeadást sorban, jegyenként elvégezzük az átvitel figyelembevételével. Így a számítógépnek mindig csak egy-két számjegy hosszúságú adatokkal kell dolgoznia, igaz az eredményként leírt számjegyek továbbra is csak egy string részét képezik. Az eredménystring számmá a szokásos VAL függvénnyel csak úgy alakítható át, hogy abban csak tíz számjegy és a hatványkitevő marad meg.

A program tartalmaz még egy érdekességet. Az eredmény értékes számjegyei előtt helyi érték pótló nullák sora éktelenkedik, márpedig ezekre nincs szükségünk. Ezeket a nullákat a képernyőről, programból letörölhetjük. Az eredmény sorát balról jobbra haladva – a képernyőn lévő karakterek vizsgálatával – elemezzük és mindaddig, amíg értékes számjegyet nem találunk, addig töröljük a képernyő adott pozícióján található karaktert. Minthogy az *e* kiszámításánál szorzásra, ill. osztásra is szükség van, nézzük meg a tetszőleges pontosságú osztás blokkdiagramját (19. ábra).



19. ábra

A gondolatmenet alapján megvalósított osztó programot a 91. program szemlélteti.

```
100 ! osztas akarhany jegyig
 110 INPUT PROMPT "hany jegyig szamoljon:? ":N
 120 INPUT PROMPT "osztando:":A
 130 INPUT PROMPT "oszto:":B
 140 IF B=0 THEN GOTO 130 !nullaval nem osztunk !!
 150 PRINT STR$(INT(A/B))&"."; !egeszresz es tizedes-
pont kiirasa
 160 GOTO 180
 170 PRINT STR$(INT(A/B));
```

```
 180 LET A=10*MOD(A,B) ! az osztando a maradek tiz-
szerese
 190 LET N=N-1 ! hol tartunk az osztasban
 200 IF N>0 THEN GOTO 170
 210 PRINT
 220 END
```

91. program

Ennyi előismeret után már nekiláthatunk az *e* értékének kiszámításához. Azonban vegyük észre, hogy a tagok egymás után igen egyszerű szabállyal előállíthatók az előzőből. A nevezők rendre faktoriálisokat tartalmaznak:

$$(n+1)! = n!*n+1).$$

A fentiek alapján követhető a feladat megoldásának menete a 92. program kapcsán.

```
100 ! e kiszamitasa  string matek
110 LET S$="000000000000000000000000000000000000000"
120 LET B$=S$:LET E$="":LET A$="1"
130 FOR N=2 TO 33
140    CALL OSZTAS
150    CALL OSSZEADAS
160    PRINT "2.";B$(2:37)
170 NEXT
180 END
190 !
200 DEF OSZTAS
210    LET M=0:LET E$=""
220    FOR I=1 TO 37
230       LET A=10*M+VAL(A$(I:I))
240       LET E=INT(A/N):LET E$=E$&STR$(E) !eredmeny
i.szamjegye
250       LET M=MOD(A,N) ! a maradek
260    NEXT
270    LET A$=E$
280 END DEF
290 !
300 DEF OSSZEADAS
310    LET F=0 !atviteljelzo
320    FOR I=37 TO 1 STEP-1
330       LET E=VAL(E$(I:I))+VAL(B$(I))+F
```

```
 340      IF E>9 THEN
 350        LET F=1:LET E=MOD(E,10) ! atvitel keletke-
zett az osszeadasnal
 360      ELSE
 370        LET F=0
 380      END IF
 390      LET B$=B$(1:I-1)&STR$(E)&B$(I+1:37)
 400    NEXT
 410 END DEF
 420 ! e= 2.71828182845904523536028747135266248 1
```

92. program

Természetesen a normál számítások elvégzéséhez bőven elegendő a számítógép nyújtotta tíz jegy pontosság. Az ilyen pontosságú számolást az Enterprise elvégzi. Ez a számítógép igen sok területen a hagyományos gondolkozást, matematikai ismeretek alkalmazását teszi lehetővé. Így a korábbi géptípusoktól eltérően lehetőséget biztosít a tízes alapú logaritmus használatára is. Igen érdekes az a Basic-ban közölt, de az 1. szegmens C8A4 címen megtalálható rutin, ami előállítja az X szám tízes alapú logaritmusát.

```
 100 !tizes alapu logaritmus x  az 1.szegmens C8A4h
rutin basic megfeleloje
 110 INPUT PROMPT "x= ":X
 120 LET W=X:LET S=SQR(10)
 130 IF X>0 THEN 160
 140 PRINT "ervenytelen argumentum"
 150 END
 160 LET K=0:LET O=.1
 170 IF X<=S AND X>S/10 THEN 220
 180 IF X>S THEN LET O=10
 190 DO UNTIL X<=S AND X>S/10
 200   LET X=X/O:LET K=K-(O=10)+(O=.1) !karakterisz-
tika meghatarozasa, mantissza konvertalasa
 210 LOOP
 220 LET A=(X-1)/(X+1)
 230 LET C1=-.6735816014 !  konstansok a RST10/92
alapjan
 240 LET C2=3.163034915
 250 LET C3=-2.915681437
 260 LET C4=-8.190800454
```

```
270 LET C5=16.9669814
280 LET C6=-10.07040695
290 LET C7=.8685889638
300 ! log sorfejtes kozelitese
310 LET A1=((C1*A^2+C2)*A^2+C3)*A^2
320 LET A2=((A^2+C4)*A^2+C5)*A^2+C6
330 LET E=(A1/A2+C7)*A
340 PRINT "lg(";X;")=          ",E+K
350 PRINT "az eredeti rutinnal",LOG10(W)
360 RUN
```

93. program

A 93. program lépéseit követve egy igazi matematikus bizony rosszallóan csóválná a fejét, de ezt majd minden Enterprise által megvalósított végtelen sor összegzésnél megteheti. Az argumentumokat egyrészt mindig úgy alakítja, módosítja, konvertálja, hogy azok egy jól behatárolható intervallumba kerüljenek. Ezekre az aránylag szűk, mindenesetre a lehetséges argumentum tartományához képest igen szűk tartományra azután olyan közelítő konstansokkal végzi el a sor összegzést, hogy a 10 értékes jegy pontosságot garantálni tudja. A RST 1Ø/92 konstanstáblából nyeri a konstansokat, amelyek közül nemelyik 3–4 tizedesjegyig, valamelyik korrekt matematikai értéket képvisel, a többi számjegy már nem az osztás várt eredménye, hanem a tartomány és a számítás kedvéért módosított érték.

Logikai műveletek

A számítógép neumanni ismérve között szerepel az is, hogy a gépnek logikai műveletek végzésére is alkalmasnak kell lennie. Az Enterprise ezen a területen is elkényezteti a felhasználot. Lehetőséget kínál logikai, valamint bináris ÉS, VAGY, TAGADÁS műveletének elvégzésére.

A logikai műveleteket többféle módon használhatjuk fel programjainkban. Vegyük pl. az AND logikai összefüggést! Használhatjuk feltétel vizsgálatok összekapcsolására:

ha az egyik feltétel igaz és a másik is igaz, akkor...

azaz

```
IF A=B AND C=5 THEN...
```

Itt a két feltétel igazságértéke (—1 ha igaz, Ø ha hamis) alapján az összekapcsolt ítéletek igazságértékét határozza meg és ha igaz, akkor végrehajtja a sor további részét. Felhasználható értékadásra is, pl.:

A=(B AND C)

alakban, azonban ez csak akkor ad értelmes használható értéket, ha B és C Ø vagy —1 igazság értéket reprezentál. Ha a zárójelben Ø-tól eltérő értékek vannak, akkor A=C értékadást végez. A bináris ÉS kapcsolat bitenkénti megegyezőséget vizsgál és eredményként az azonos biten 1 érték esetén ad 1-et.

```
      14   111Ø
BAND   7   Ø111
      ----------
       6   Ø11Ø
```

A vagy, ill. tagadás műveletét is vizsgálhatjuk hasonló gondolatmenet szerint. A logikai műveletek összekapcsolhatók, de vannak olyan gyakran előforduló logikai műveletek, melyek önálló névvel rendelkeznek. A nagyobb számítógépek ezeket a logikai műveleteket is képesek egy kulcsszó segítségével elvégezni. Ne irigykedjünk, hiszen a három alap logikai művelet segítségével minden logikai kapcsolat és igazságtáblázatuk felírhatók. Ennek megvalósítását mutatja a 94. program.

```
100 ! logikai muveletek
110 CLEAR SCREEN:PRINT :PRINT
120 PRINT "1. tagadas (NOT)    A NOT A"
130 PRINT "2. es      (AND)    A AND B"
140 PRINT "3. vagy    (OR )    A OR  B"
150 PRINT "4. kizaro vagy      A XOR B"
160 PRINT "5. ekvivalencia     A EQV B"
170 PRINT "6. implikacio       A IMP B"
180 PRINT :PRINT "melyik igazsagtablajat ?"
190 LET V$=INKEY$&" "
200 LET V=VAL(V$)
210 IF V<1 OR V>6 THEN GOTO 190
220 LET L$="  ## 1   ##"
230 LET J$="  # 1 # 1     #"
240 PRINT :PRINT
250 IF V=1 THEN
260   PRINT "   A 1  NOT A":PRINT "-------------"
270   FOR I=0 TO 1
280     PRINT USING L$:I,-1*(NOT(I))
290   NEXT
300 END IF
310 SELECT CASE V
320 CASE 1
```

```
330 CASE 2
340   PRINT "  A 1 B 1  A AND B"
350 CASE 3
360   PRINT "  A 1 B 1  A OR B"
370 CASE 4
380   PRINT "  A 1 B 1  A XOR B"
390 CASE 5
400   PRINT "  A 1 B 1  A EQV B"
410 CASE 6
420   PRINT "  A 1 B 1  A IMP B"
430 END SELECT
440 PRINT "------------------"
450 FOR I=0 TO 1
460   FOR J=0 TO 1
470     SELECT CASE V
480     CASE 1
490     CASE 2
500       PRINT USING J$:I,J,(I BAND J)
510     CASE 3
520       PRINT USING J$:I,J,(I BOR J)
530     CASE 4
540       PRINT USING J$:I,J,(I BOR J)-(I AND J)
550     CASE 5
560       PRINT USING J$:I,J,-1*(NOT((I BOR J)-(I AND
J)))
570     CASE 6
580       PRINT USING J$:I,J,(-1)*(NOT(I)) BOR J
590     END SELECT
600   NEXT
610 NEXT
620 PRINT
630 PRINT "billentyu nyomasra tovabb"
640 IF INKEY$="" THEN GOTO 640
650 RUN
```

94. program

LOOK

Ennek az utasításnak a segítségével – a gépkönyv szerint – tetszőleges változóba töltetjük az *x*, *y* koordinátájú pont színének palettaszámát. A praktikus felhasználás szempontjából megtudhatjuk, hogy az adott képernyőponton van-e valami? Jó hasznát vehetjük ennek játékprogramok esetén ütközések, találkozások érzékelésére. Nézzünk egy példát arra, hogy egy zárt idomot hogyan tudunk ennek az utasításnak a segítségével bevonalazni. Olyan programot kell írni, mely egy zárt síkidom, pl. kör határát megkeresi, ha megtalálta, akkor húzzon vonalat egészen a túloldali határvonalig. A megvalósítást a 95. program mutatja.

```
100 !   look - sraffozas
110 GRAPHICS 16
120 SET INK 255:PLOT 640,360
130 PLOT ELLIPSE 300,300
140 SET INK 0:PLOT 640,360:SET INK 255
150 FOR Y=82 TO 655 STEP 40
160   FOR X=330 TO 950 STEP 8
170     LOOK AT X,Y:V
180     IF V<>0 THEN CALL SRAFF ! keresem a hatar-
vonalat
190   NEXT
200 NEXT
210 END
220 DEF SRAFF
230   DO
240     LET X=X+8
250     LOOK AT X,Y:V !tapogatom a kepernyot
260     PLOT X,Y;
270     IF V<>0 THEN EXIT DO ! a tuloldali hatar-
vonalba utkoztem
280   LOOP
290   LET X=1000 !hogy uj ciklust kezdjen a vissza-
teres utan
300 END DEF
```

95. program

A 15Ø–2ØØ. sorok keresik az idom határát. A SRAFF eljárás a vonalhúzást végzi és megkeresi a túloldali határoló vonalat. A LOOK utasítást felhasználva a szöveg tükörképét a 96. program, vagy ferdén való kiírását a 97. program valósíthatja meg.

```
 1ØØ ! szoveg tukorkepe
 11Ø GRAPHICS 16
 12Ø SET PALETTE BLACK;WHITE
 13Ø PLOT 28Ø,4ØØ,
 14Ø PRINT #1Ø1:"ENTERPRISE" !kiirjuk a fejreallitando
szoveget
 15Ø PLOT Ø,32Ø;1279,32Ø.!tukrozesi tengely
 16Ø PLOT 6ØØ,5ØØ ! tukrozesi kozeppont
 17Ø FOR Y=Ø TO 6Ø STEP 4
 18Ø   FOR X=28Ø TO 92Ø STEP 8
 19Ø     LOOK AT X,35Ø+Y:V ! letapogatjuk a szoveget
 2ØØ     IF V=Ø THEN GOTO 23Ø ! ha nincs keppont
keressunk ujat
 21Ø     PLOT X,292-Y !  tengelyre vett tukorkep meg-
rajzolasa
 22Ø     PLOT 12ØØ-X,64Ø-Y !pontra vett tukorkep meg-
rajzolasa
 23Ø   NEXT
 24Ø NEXT
```

96. program

```
 100 ! szoveg ferden fel-le
 110 SET STATUS OFF
 120 GRAPHICS 16
 122 SET PALETTE BLACK,WHITE
 130 SET LINE MODE 1
 140 PLOT 330,719,
 150 PRINT #101:"ENTERPRISE" !az athelyezendo szoveg ki
irasa
 160 FOR X=330 TO 980 STEP 4
 170   FOR Y=0 TO 52 STEP 4
 180     LOOK AT X,667+Y:V
 190     IF V=0 THEN GOTO 230
 200     PLOT 310-2*Y,X-290 !fuggolegesen felfele ir
```

```
210      PLOT 980+1.5*Y,1000-X !fuggolegesen lefele ir
220      PLOT X+16-Y*.7,(X-330)+Y*.7 !ferden felfele ir
230    NEXT
240 NEXT
```

97. program



20. ábra

Ez utóbbi program futásának eredményét a 20. ábra mutatja.

A COPY programnál alkalmazott LOOK utasítás felhasználás mellett, a képernyőn való ábra áthelyezésre is felhasználhatjuk azt. A 98. program az ábra áthelyezése mellett lehetőséget biztosít arra, hogy az ábrát ne csak egyszerűen áthelyezzük, hanem az ábrát minden, vagy csak egy irányba nagyítsuk, torzítsuk.

```
100 !grafikus abra nagyitas
110 GRAPHICS 16
120 OPTION ANGLE DEGREES
130 SET INK 0:PLOT 340,600
140 SET INK 255:PLOT ELLIPSE 100,100 !feje
150 PLOT 380,650 ELLIPSE 16,12 !szeme
160 PLOT 300,650 ELLIPSE 16,12 !szeme
170 PLOT 340,600 ELLIPSE 14,30 !orra
180 SET INK 0:PLOT 340,600:SET INK 255
190 FOR A=30 TO 150 STEP 6
200   PLOT 340-50*COS(A),590-50*SIN(A) !szaja
210 NEXT
220 FOR Y=500 TO 700 STEP 4
230   FOR X=240 TO 440 STEP 8
240     LOOK AT X,Y:V !letapogatja a kepernyot
250     IF V=0 THEN 280
```

```
260      CALL LOFEJ !fuggoleges iranyu nagyitas
270      CALL NAGYFEJ !mindket iranyu nagyitas
280    NEXT
290 NEXT
300 END
310 DEF LOFEJ !fuggoleges torzitas
320    PLOT X-200,10+2*(Y-500);X-200,10+2*(Y-500)+5
330 END DEF
340 DEF NAGYFEJ ! mindket iranyba nagyit
350    FOR Q=0 TO 6
360      LET V1=530+3*(X-240):LET F1=10+3*(Y-500)+Q
370      PLOT V1,F1;V1+18,F1
380    NEXT
390 END DEF
```

98. program

MAX

Ez a függvény az argumentumként adott két változó közül a nagyobbat adja eredményül. Előnyösen használható számsorokban szélső érték keresésére, ill. rendező programban.

A program áttekinthetőbb lesz, uganakkor – amiről mi magunk is meggyőződhetünk a 99. program futása során – a végrehajtási idő kb. 5%-kal hosszabb lesz.

```
100 INPUT PROMPT "elemek szama: ":N
110 CLEAR SCREEN:RANDOMIZE
120 LET MX=1E-60:LET MN=1E60
130 TIME "00:00:00"
140 FOR I=1 TO N
150    LET A=RND(2000) !elloallitja a szamokat
160    PRINT I;A,
170    IF A>MX THEN LET MX=A !maximum kereses IF-fel
180    IF A<MN THEN LET MN=A
190    PRINT MX;MN
200 NEXT
210 PRINT TIME$
220 IF INKEY$="" THEN GOTO 220
```

```
 230 !INPUT PROMPT "elemek szama: ":N
 240 !
 250 CLEAR SCREEN:RANDOMIZE
 260 LET MX=1E-60:LET MN=1E60
 270 TIME "00:00:00"
 280 FOR I=1 TO N
 290    LET A=RND(2000)
 300    PRINT I;A,
 310    LET MX=MAX(MX,A):LET MN=MIN(MN,A) ! maximum
kereses MAX fuggvennyel
 320    PRINT MX;MN
 330 NEXT
 340 PRINT TIME$
```

99. program

MOD

Mindennapi, általános- és középiskolai matematika tanulmányunkban ilyen formában nem találkoztunk ezzel a függvénnyel, pedig igen sokszor használtuk a maradék meghatározását különböző osztási feladatok elvégzésénél.

A hétköznapi életben is számos felhasználási területe adódik. Egy ilyen mindenki számára ismert alkalmazási terület a személyi számunk. A személyi szám nemre, állampolgárságra és születési dátumra ad egyértelmű felvilágosítást. Ez viszont a 11 jegyből csak 7-et igényel. A napi sorszám (ha figyelembe vesszük, hogy a statisztikai valószínűség szerint Magyarországon naponta nem születik 500 ember sem) három számjeggyel leírható. Az utolsó számjegy a személyi számról első ránézésre megmondja, nincs-e benne elírás. Az utolsó számjegy ui.

$$x = \mathrm{MOD}\left[\sum_{i=1}^{10} i \cdot s(i),\ 11\right]$$

Kissé érthetőbben, a személyi szám 11. jegyét úgy kapjuk meg, hogy rendre összeadjuk az első tíz számjegy tagjainak a személyi számban elfoglalt helyükkel való szorzatát és az eredményt elosztjuk tizeneggyel. Az osztás maradéka a személyi szám tizenegyedik jegye.

A 100. program ezt a bonyolultnak látszó számítást elvégzi.

```
100 ! szemelyi szam  MOD
110 DIM H$(12),N(12)
120 FOR I=1 TO 12 ! honap nap egyeztetes elokeszitese
130   READ H$(I),N(I)
140 NEXT
150 CLEAR TEXT
160 PRINT AT 8,15:"12345678901" !adatbevitel segitese
170 LET H=0:LET SUMMA=0
180 INPUT PROMPT "szemelyi szam :":S$
190 PRINT
 200 IF S$(LEN(S$):LEN(S$))=" " THEN
 210   LET S$=S$(1:LEN(S$)-1) !szokozok elhagyasa
 220   GOTO 200
 230 END IF
 240 IF LEN(S$)<>11 THEN PRINT "hibas hossz":GOTO 560
!szoveg hossz ellenorzese
 250 FOR I=1 TO 11
 260   IF S$(I:I)<"0" OR S$(I:I)>"9" THEN LET H=1 !
 szemelyi szam karaktereinek ellenorzese
 270   LET SUMMA=SUMMA+VAL(S$(I:I))*I ! szemelyi szam
 szamjegyeibol sulyozott osszeget kepez
 280 NEXT
 290 IF H=1 THEN PRINT "hibas karakter":GOTO 560
 300 LET V$=S$(1:1):LET V=VAL(V$)
 310 IF V<1 OR V>8 THEN PRINT "elso karakter hibas    "
:GOTO 560
 320 LET A$="nem magyar ":LET B$="ferfi ":LET E$="19"
 330 IF V<5 THEN LET A$="magyar "
 340 IF V/2=INT(V/2) THEN LET B$="no "
 350 PRINT "az illeto: " A$ "allampolgarsagu " B$
 360 IF (V-1) BAND 2 THEN LET E$="18"
 370 PRINT "szuletesi ido: " E$ S$(2:3);" ";
 380 LET V$=S$(4:5):LET V=VAL(V$)
 390 IF V<1 OR V>12 THEN
 400   PRINT "hibas honap adat":GOTO 560
 410 ELSE
 420   PRINT H$(V);" ";
 430 END IF
```

```
 44Ø IF V=2 AND(VAL(S$(2:3)) AND 3=Ø) AND S$(6:7)="29"
AND S$(2:3)<>"ØØ" THEN GOTO 47Ø
 45Ø IF N(V)>=VAL(S$(6:7)) THEN GOTO 47Ø
 46Ø PRINT "hibas honap, nap egyeztetes":GOTO 56Ø
 47Ø PRINT S$(6:7)
 48Ø PRINT "napi sorszam: " S$(8:1Ø)
 49Ø PRINT "ellenorzo osszeg: ";
 5ØØ LET ELL=MOD(SUMMA,11) !ellenorzo osszeg MOD !!!!
 51Ø IF ELL=VAL(S$(11:11)) THEN
 52Ø   PRINT S$(11:11)
 53Ø ELSE
 54Ø   PRINT "hibas"
 55Ø END IF
 56Ø PRINT :PRINT "bill. nyomasra indul"
 57Ø IF INKEY$="" THEN GOTO 57Ø
 58Ø RUN
 59Ø DATA januar,31,februar,28,marcius,31
 6ØØ DATA aprilis,3Ø,majus,31,junius,3Ø
 61Ø DATA julius,31,augusztus,31,szeptember,3Ø
 62Ø DATA oktober,31,november,31,december,31
```


Természetesen, ha már programot írunk attól több szolgáltatás is elvárható, mint egy maradék kiszámítása. A személyi számban egy sor egyeztetést kell ellenőrizni, így a hónap és nap összhangját is vizsgálni kell. Itt ezt a 44Ø–45Ø. sorok végzik el. Az ellenőrzéshez szükséges adatokat a DATA sorok tartalmazzák, de igénybe lehetne venni a gép rendszerprogramjában lévő naptár ellenőrző rutint. Erről a DATE-nél bővebben.

Visszatérve a MOD-hoz. A különböző számrendszerek közötti átváltó programokban mindig szükség van a maradék további vizsgálatára. A MOD függvény ilyen célú alkalmazását találjuk a 101. programban.

```
 1ØØ !dec --> akarmi
 11Ø INPUT PROMPT "milyen alapra:":A
 12Ø IF A<2 OR A>9 THEN GOTO 11Ø
 13Ø INPUT PROMPT "atvaltando dec szam:":D$
 14Ø LET H=Ø
 15Ø FOR V=1 TO LEN(D$)
 16Ø   IF D$(V:V)<"Ø" OR D$(V:V)>"9" THEN LET H=1
 17Ø NEXT
```

```
 180 IF H=1 THEN PRINT "hibas karakter":GOTO 110
 190 LET D=VAL(D$):LET J=0
 200 IF D>65536 THEN GOTO 110
 210 IF D<0 OR D<>INT(D) THEN GOTO 110
 220 FOR O=16 TO 0 STEP-1
 230   LET A$=STR$(INT(D/A^O))
 240   IF A$>"0" THEN LET J=1 ! szam ele nullakat
nem irunk ki
 250   IF J=1 THEN PRINT A$;
 260   LET D=MOD(D,A^O)
 270 NEXT
 280 PRINT
```

101. program

Az előző probléma megoldásához hasonló gondolatmenetet igényel a címletező program megírása. Igaz, itt az egymásután következő helyi értékek nem mértani sorban következnek egymásután. A 102. program végzi a címletek számának meghatározását.

```
100 !cimletezes
110 INPUT PROMPT "osszeg:":SUMMA
 120 FOR I=1 TO 9
 130   READ CIMLET
 140   LET DARAB=INT(SUMMA/CIMLET)
 150   IF DARAB=0 THEN GOTO 180
 160   LET SUMMA=MOD(SUMMA,DARAB*CIMLET)
 170   PRINT USING "#### Ft  #### db ######":CIMLET,DAR
AB,DARAB*CIMLET
 180 NEXT
 190 DATA 1000,500,100,50,20,10,5,2,1
```

102. program

Adódhat probléma futás közben, főleg ha a 15Ø. sort elhagyjuk. Ugyanis előfordulhatnak olyan esetek, amikor nullával való osztás maradékát kellene megállapítani. A MOD (X, Ø) nem nullával való osztás hibát, hanem 3ØØ6 típusú matematikai hibát jelez.

NEW

Ez az utasítás törli a programot a tárból. De tényleg törli? A programsorok tárolásának elemzése során láttuk, hogy az átlag Basic nyelvű programok a RAM-ban úgy helyezkednek el, hogy az első programsor a 4827dec memóriacímen kezdődik, ahol a sor hossza található. A program végén pedig egy nulla van, minthogy az utolsó programsor után Ø hosszúságú sor következik. Ha végrehajtás közben Ø hosszúságú sorhoz érkezik az értelmező, akkor azt programvégnek tekinti. Futtasuk le a következő 103. számú programot.

```
100 FOR I=1 TO 10
  110   PRINT I
  120 NEXT
  130 !
  140 !elozo  sornak az eleje most 4862 cimen van es tar
talma 7
  150 FOR I=65 TO 75
  160   PRINT CHR$(I)
  170 NEXT
  200 ! NEW es vissza probaprogram
```

103. program

A futás után hajtassuk végre a géppel a POKE 4862,Ø parancsot. Listázás után meggyőződhetünk arról, hogy már csak a 1Ø–3Ø. sorokról vesz tudomást a gép. Amíg nem indítjuk el ezt az új programot, addig POKE 4862,7-tel visszanyerhetjük az eredeti programlistát. A megrövidített program lefuttatása után viszont már hiába próbálnánk meg az eredeti visszaállítását, hiszen közvetlenül a program utáni területet a gép használja és felülírja. Ezek ismeretében már több lehetőség is kínálkozik a program visszanyerésére.

Az egyik lehetőség az, hogy a programot mindig egy ismert hosszúságú programsorral kezdjük.

```
Pl.:        1 REM              1 PRINT
            1 !                1 TEXT
```

Ezek alkalmazása esetén a 4827 memóriacímen 7 található. Ez akkor is így van, ha ALLOCATE-val gépi kódú programrésznek is helyet foglaltunk. Tehát egy olyan programot, amely úgy kezdődik, hogy: 1 REM NEW parancs után egyszerűen és biztonsággal visszanyerhetünk – mindaddig, amíg a NEW után nem zavartuk a gépet – pl. parancsmódban elvégzett POKE 4827,7 segítségével.

Más megoldást kell találni arra az esetre, ha nem ismerjük a meggondolatlan NEW utasítással megsemmisített programunk első sorának hosszát. Ilyenkor,

hagyjuk békében ezt a memóriaterületet és a gép adta lehetőséget választva váltsunk át egy másik programra, pl. CHAIN 111. Igaz, hogy ha itt még nincs program, akkor csak egy tiszta lapot kaptunk. Írjuk be és futtassuk a 104. programot!

```
100 ! OLD program Ø
110 ! ha ez az N. program
120 LET H=Ø ! az elso programsor hosszanak megallapi-
tasahoz kezdo ertek
130 LET V=4832 ! itt van a sertetlen token
140 DO
150    LET A=PEEK(V)
160    LET H=H+1 ! hossz novelese
170    LET V=V+1 ! kovetkezo elem tarolasi cime
180 LOOP UNTIL A=96 ! megtalaltuk a kovetkezo sor
kulcsszo bevezetojet
190 PRINT H ! tehat az elozo sor hossza =H
200 POKE 4827,H ! ezt visszaallitjuk
```

104. program

Így a korábbi Ø. program visszaállítható. Erről CHAIN Ø vagy EDIT Ø utáni listázással meggyőződhetünk. A NEW hatása mindig az aktuális programra érvényes. Ha nem adjuk ki a NEW ALL parancsot, akkor a többi programunk sértetlenül megmarad a memóriában. Így megtehetjük, hogy „biztonsági másolatot" készítünk a programunkról. Az itt közölt megoldás a Basic adta sebességek és a számítógép tárral való gazdálkodása miatt csak max. 1 kB programhosszúság jöhet számításba. Bekapcsolás utáni alapállapotú gépbe írjuk be Ø., 1. és 2. programként az 1 REM-ből álló programot. Ezzel tulajdonképpen csak a memóriatérképet építtettük fel a számítógéppel.

Első sorszámú programként írjuk be az átmozgató 105. programot.

```
100 ! atmozgato elore bejegyzett 2.program helyere,
ez az 1-es!!
110 IF PEEK(3671)=Ø THEN PRINT "nincs bejegyezve a
 celprogram terulete  ":STOP
120 LET CEL=PEEK(3670)
130 PRINT "Ø. program attoltes a ";CEL "szegmensre  "
;PEEK(3671);".proramkent"
140 ! a nulladik program hosszanak megallapitasa
150 LET KEZD=4827:LET HOSSZ=Ø
160 DO
```

```
170    LET SORHOSSZ=PEEK(KEZD)
180    LET HOSSZ=HOSSZ+SORHOSSZ
190    IF SORHOSSZ=Ø THEN EXIT DO
200    LET KEZD=4827+HOSSZ
210 LOOP
220 PRINT "az atmozgatando byte szam:";HOSSZ
230 FOR I=Ø TO HOSSZ
240    SPOKE CEL,16384+I,PEEK(4827+I)
250 NEXT
260 PRINT "attoltes kesz"
```

105. program

Majd térjünk át a Ø. programra és írjuk be. Írjuk meg a kívánt programunkat. Erről a programról biztonsági másolat készíthető az 1. program segítségével. Az átmásolandó byte számot az INFO által Ø. programra szolgáltatott adatból ciklus végértékeként is megadhatjuk. Ha a program lefutása után visszatérünk a Ø. programra és azt NEW-val töröljük, akkor sincs baj, hiszen a 2. programot meghívva ott még sértetlenül rendelkezésünkre áll a korábban Ø-ként beírt programunk.

A NEW csak parancsmódban használható. Logikus ez, de programvédelmi szempontból van aki szereti az „öngyilkos" programot. Készíthetünk ilyen programot a NEW felhasználásával, természetesen van más lehetőség is.

Nézzük, hogyan kerül a programba a NEW mint utasítás! Írjuk be a 106. programot!

```
100 ! ongyilkos program
110 INPUT PROMPT "jelige":J$
120 IF J$<>"SAMU" THEN GOTO 160
130 PRINT "Oke"
140 END
150 ! a legutolso, a hivatkozott sor utani csak egy
PRINT legyen
160 POKE PEEK(566)+256*PEEK(567)-3,46
170 PRINT
```

106. program

Elindítás után kéri a jelszót, aminek birtokában a program használható. Ebben a példában a jelszó SAMU, csupa nagybetűvel! Amennyiben a program használója nem ezt írja be, akkor a program 12Ø. sora szerint a program az utolsó sorra adja a vezérlést. Itt most a listában PRINT utasítás található, de lehetne bármilyen semleges utasítás REM, END, STOP! Minthogy ezen a soron NEW-ra

lenne szükségünk át kell írni az utolsó sorban lévő kulcsszó tokent a NEW tokenjére. Ehhez ismerni kell a NEW tokenjét, valamint azt, hogy a memória melyik címén található a lecserélendő byte.

A program végét a 23AH rendszerváltozó segítségével kereshetjük meg. Így az elkészült, vagy készítés alatt lévő programunk utolsó sorában lévő kulcsszavát mindig lecserélhetjük a parancsmódban elvégzett

POKE PEEK (566)+256 * PEEK (567) —3,46

segítségével.

Ezután már a listában is látható a NEW kulcsszó és természetesen az adott sorszámú sor végrehajtása esetében programból elvégzi a program törlését. Ha a programírás közben, amikor beírtuk a 107. programba az addigi legmagasabb sorszámmal a feltétel vizsgálatot – itt most a 12Ø. sort – akkor végezzük el a PRINT token lecserélését.

```
100 ! ongyilkos program /2
110 INPUT PROMPT "jelige":J$
120 IF J$<>"SAMU" THEN NEW
130 PRINT "Oke"
140 FOR I=1 TO 5
150   PRINT I;". ";J$
160 NEXT
```

107. program

Majd, mintha mi sem történt volna folytathatjuk a programírást. Amennyiben nem ragaszkodunk a program kiírásának elvégzésekor a NEW kulcsszóhoz, akkor beépíthetjük programunkba az

IF A$ <>"SAMU" THEN EXT "WP"

utasítás sort, ami tökéletesebben elvégzi a program törlését, mint maga a NEW.

OPEN

A különböző célra igénybevett csatornákat – hacsak az alapértelmezésből nem következik – a felhasználás előtt meg kell nyitni. A háttértár – egyszerűbb esetben a magnetofon – adattárolásra is alkalmas. A programmentés SAVE ugyanúgy magába foglalja az adatáramlás irányát, mint ahogy a LOAD is kijelöli a csatorna használatának irányát. Ha nem ezzel a két kulcsszóval nyitjuk meg a magnetofon irányába a 106. csatornát, akkor a megnyitás mellett az igénybevétel irányát is meg kell határozni. Ennek megvalósítására nézzünk egy konkrét példát (108. program).

```
100 ! adatbazis  telefonkonyv
110 LET W=100 ! ennyi adatra van felkeszitve
120 LET N=0 !betoltott adatok szama
130 STRING N$(W)*20,S$(W)*8
140 CLEAR TEXT
150 PRINT :PRINT :PRINT
160 PRINT "   menu":PRINT :PRINT
170 PRINT "     1.  kereses":PRINT
180 PRINT "     2.  adatbevitel":PRINT
190 PRINT "     3.  adat magnora":PRINT
200 PRINT "     4.  adat magnorol":PRINT
210 PRINT "     5.  vege          ":PRINT
220 LET A$=INKEY$
230 ON VAL(A$) GOTO 250,400,480,570,800
240 GOTO 220
250 ! kereses
260 GOSUB 640
270 INPUT PROMPT "mire kivancsi: ":K$
280 IF K$="x" THEN 140
290 LET FLAG=0:LET I=1
300 DO
310   IF K$=N$(I) OR K$=S$(I) THEN
320     PRINT N$(I);" : ";S$(I)
330     LET FLAG=1
340   END IF
350   LET I=I+1
360 LOOP UNTIL FLAG=1
370 IF FLAG=0 THEN PRINT "sajnos nem talalom"
380 PRINT
390 GOTO 250
400 ! adatbevitel
410 GOSUB 640
420 LET N=N+1
430 INPUT PROMPT "nev   :  ":N$(N)
440 IF N$(N)="x" THEN LET N=N-1:GOTO 140
450 INPUT PROMPT "szama:  ":S$(N)
460 PRINT
470 GOTO 420
```

```
480 ! magnora
490 OPEN #106:"telefon" ACCESS OUTPUT
500 PRINT #106:N !kiviendo adatparok szama
510 FOR I=1 TO N
520   PRINT #106:N$(I)
530   PRINT #106:S$(I)
540 NEXT
550 CLOSE #106
560 GOTO 140
570 ! magnorol
580 OPEN #106:"telefon" ACCESS INPUT
590 INPUT #106:N !beolvasando adatparok szama
600 FOR I=1 TO N
610   INPUT #106:N$(I),S$(I)
620 NEXT
630 GOTO 550
640 PRINT :PRINT
650 PRINT "menuhoz vissza """ X """ betuvel"
660 PRINT :PRINT
670 RETURN
```

108. program

Ez az adatbáziskezelő lehetővé teszi adatpárok beírását, összetartozó értékek egyikének ismeretében a hozzárendelhető keresését. Az ilyen nyilvántartó programok akkor használhatók, ha adatait háttértárra kimenthetjük, ill. onnan visszaolvashatjuk. Az ilyen feladatok megvalósítását tanulmányozhatjuk ezen a példán. Természetszerűleg a menü választékát tovább bővíthetjük listázási, adatjavítási ágakkal.

Ha nagy mennyiségű adatot (több száz) akarunk tárolni, akkor a keresés során az adat elérése az adathalmazban való elhelyezkedéstől függően több-kevesebb időt vesz igénybe. Az elérési idő csökkenthető, ill. egységesíthető, ha az adatsor rendezett. A rendezésre igen sokféle program áll rendelkezésre. A szakirodalmi összehasonlítások alapján a Quick rendező az egyik leggyorsabb. A 109. program ezt mutatja be.

```
100 ! quick rendezo
110 INPUT PROMPT "elemek szama: ":N
120 STRING A$(N)*20
122 NUMERIC F(N),A(N)
130 CLEAR SCREEN
140 FOR I=1 TO N !eloallitja az N db adatot
```

```
150    LET A$(I)=""
160    FOR K=1 TO 6 !egy adat hossza 6 karakter lesz
170      LET A$(I)=A$(I)&CHR$(65+RND(26))
180    NEXT
190    PRINT I;A$(I)
200 NEXT
210 GOSUB 230
220 GOTO 420
230 LET Z=0:LET A(1)=1:LET F(1)=N
240 GOSUB 250
250 LET Z=Z+1
260 IF A(Z)>=F(Z) THEN GOTO 410
270 LET X=A(Z):LET Y=F(Z):LET V$=A$(INT((X+Y)/2))
280 DO UNTIL X>Y
290    DO WHILE A$(X)<V$
300      LET X=X+1
310    LOOP
320    DO WHILE A$(Y)>V$
330      LET Y=Y-1
340    LOOP
350    IF X>Y THEN EXIT DO
360    LET S$=A$(X):LET A$(X)=A$(Y):LET A$(Y)=S$
370    LET X=X+1:LET Y=Y-1
380 LOOP
390 LET F(Z+1)=Y:LET A(Z+1)=A(Z):GOSUB 250
400 LET A(Z+1)=X:LET F(Z+1)=F(Z):GOSUB 250
410 LET Z=Z-1:RETURN
420 FOR I=1 TO N !rendezett allomany kiirasa
430    IF N>22 THEN GOTO 450
440    PRINT AT I,20:"";
450    PRINT I;A$(I)
460 NEXT
```

109. program

Ha rendezett az adatállomány, akkor bináris keresést lehet alkalmazni. Ez a névsorban mindig a rendelkezésünkre álló halmaz felezésével határolja be a konkrét adatot. Ez azt jelenti, hogy bármelyik (legkedvezőtlenebb helyen lévő) adat $\log_2 N$ számú kereséssel megtalálható.

A háttértáron (magnetofon) való adattárolás speciális esete, mely során a képernyő tartalmát kívánjuk rögzíteni, tárolni. A képernyőről, ill. képernyőme-

móriából kell az adatokat a magnetofonra juttatni. A 110. program a szöveges (TEXT) képernyő 102. csatorna tartalmát írja ki adatállományként a magnetofonra.

```
100 ! szoveges kepernyo magnora es vissza
108 OPEN #106:"kep" ACCESS OUTPUT
110 FOR Y=1 TO 24
120   LET A$=""
130   FOR X=1 TO 40
140     PRINT #102,AT Y,X:;
150     GET #102:B$ !  kivalasztom a kepernyon levo
karaktert
 160     LET A$=A$&B$ ! egy sor tartalma egy stringbe
 170   NEXT
 190   PRINT #106:A$ ! egy sor tartalma magnora
 210 NEXT
 211 CLOSE #106
 212 PRINT "kivitel kesz, gombnyomas es magno bealli-
tas utan visszatolt"
 230 IF INKEY$="" THEN 230
 231 TEXT
 240 OPEN #106:"kep"&STR$(Y) ACCESS INPUT
 250 FOR Y=1 TO 24
 270   INPUT #106:A$
 280   PRINT A$
 300 NEXT
 390 CLOSE #106
 400 GOTO 400
```

110. program

Az adott file nevű képinformáció a visszatöltés programággal visszanyerhető és a magnetofonról képernyőre tölthető.

Izgalmasabb és indokoltabb lehet a grafikus képernyőtartalom magnetofonra való mentése a COPY utasítás segítségével. Ennek Basic programmal való megvalósítására mutat példát a 111. program. A magnetofonra küldő és a magnetofonról visszatöltő programrészek egymással összhangban biztosítják a megfelelő adatmozgatásokat.

```
100 ! grafikus kepernyo magnora
110 STRING A$*161
120 GRAPHICS 16
130 SET INK 255
140 !
150 ! DEMO IDE IRHATO BE
160 !
170 CALL KEP_MAGNORA
180 PRINT "vissza"
190 IF INKEY$="" THEN 190
200 CALL VISSZATOLT
210 END
220 !
230 DEF KEP_MAGNORA
240   FOR Y=0 TO 719 STEP 4
250     LET A$=""
260     FOR X=0 TO 1279 STEP 8
270       LOOK AT X,Y:V
280       LET A$=A$&CHR$(V+128)
290     NEXT
300     SET REM2 ON
310     OPEN #106:"kep" ACCESS OUTPUT
320     PRINT #106:A$
330     CLOSE #106
340     SET REM2 OFF
350   NEXT
360 END DEF
370 !
380 DEF VISSZATOLT
390   GRAPHICS 16
400   FOR Y=0 TO 719 STEP 4
410     SET REM1 ON
420     OPEN #106:"kep" ACCESS INPUT
430     INPUT #106:A$
440     CLOSE #106
450     SET REM1 OFF
460     FOR X=1 TO 161
470       SET INK(ORD(A$(X:X))-128)
```

```
480         PLOT (X-1)*8,Y
490       NEXT
500     NEXT
510 END DEF
```

111. program

POKE

Az IS–BASIC komfortsága szinte feleslegessé teszi ezt az utasítást. Ebben a könyvben mi nagyon sokszor használtuk, hiszen a Basic határait így tudtuk feszegetni. Az EXOS változók, ill. átállítása a SET és TOGGLE utasítással lehetséges. Az EXOS változók a számítógép BFC5H memóriaterületen találhatók. A gépi feltételek sokrétűségét, a lehetséges variációkat a 27. program segítségével vizsgálhatjuk. Hogy az azonosított feltétellel mi a teendő, azt a Ø. szegmens C9Ø2H címen kezdődő rutin dönti el és egyben meghatározza a módosítandó memóriacímet is.

Az EXOS változók sorszáma, megnevezése a tárolt alapértelmezés és a tárolási cím decimális értéke a következő táblázatból követhető:

Ø – IRQ – ENABLE–STATE
bØ – hang IRQ engedély beállítás
b2 – 1 Hz IRQ engedély beállítás
b4 – video IRQ engedély beállítás
b6 – külső IRQ engedély beállítás
Alapérték Ø, cím 49Ø93

1 – FLAG–SOFT–IRQ
Ha egy eszköz ezt a byte értékét nullától különbözőre állítja az szoftvermegszakítást okoz.
Alapérték Ø, cím 49Ø94

2 – CODE–SOFT–IRQ
A megszakításkezelő programok ezt a byte-ot vizsgálják, és innen tudható meg a megszakítás oka.
Alapérték Ø, cím 49Ø95

3 – DEF–TYPE
A csatlakoztatott perifériáktól függően
Ø=(ha magnetofon)
1=(ha lemezmeghajtó).
Cím 49Ø96

4 – DEF–CHAN
Az alapértelmezés szerinti csatornaszám.
Alapérték Ø, cím 49Ø97

5 – TIMER
1 Hz számláló. Szoftvermegszakítást generál, ha eléri a nullát.
Alapérték Ø, cím 49Ø98

6 – LOCK–KEY
A LOCK billentyű állapota
alapérték Ø
caps 1
Shift 2
Alt 8
Cím 49Ø99

7 – CLICK–KEY
Billentyűhang engedélyezés.
Alapérték Ø, cím 491ØØ

8 – STOP–IRQ
Ø=a STOP billentyű szoftver IRQ-t okoz
Ø=a STOP billentyű hatásának kiiktatása.
Cím 491Ø1

9 – KEY–IRQ
Ø bármely billentyű szoftver IRQ-t eredményez visszatérési kóddal.
Alapérték 255, a cím 491Ø2

1Ø – RATE–KEY
Billentyűzet automatikus ismétlési sebesség 1/50-ed másodpercben.
Alapérték 3, cím 491Ø3

11 – DELAY–KEY
Késleltetés az automatikus ismétlés beindulása előtt.
Alapérték 3Ø, cím 491Ø4

12 – TAPE–SND
Ø=szalag hang engedélyezés.
Cím 491Ø5

13 – WATT–SND
Ø=a hangmeghajtó vár, ha a sor tele van.
Cím 491Ø6

14 – MUTE–SND
Ø=belső hangszóró aktív
ha az érték nem nulla, a belső hangszóró kikapcsolva.
Cím 491Ø7

15 – BUF–SND
Burkológörbe tárméret „fázis"-okban megadva.
Alapérték 2Ø, a cím 491Ø8

16 – BAUD–SER
A soros vonal sebessége (baud), alapérték 15, ami 9600 baud átviteli sebességet jelent.
Cím 491Ø9

17 – FORM–SER
A soros vonal szóformátuma.
Alapérték Ø, a cím 4911Ø

18 – ADDR–NET
A gép hálózati címe.
Cím 49111

19 – NEXT–IRQ
Ø = a hálózaton érkező adat megszakítást okoz.
Cím 49112

2Ø – CHAN–NET
A hálózatról jövő blokk csatornaszáma.
Alapérték 255, a cím 49113

21 – MACH–NET
A hálózatról jövő blokk, forrásgép száma.
Alapérték 255, a cím 49114

A következő négy változó a video lap jellemzőit határozza meg.

22 – MODE–VID video mód.
Alapérték Ø, cím 49115

23 – COLR–VID szín mód.
Alapérték Ø, cím 49116

24 – X–SIZ–VID X lapméret.
Alapérték 4Ø oszlop, cím 49117

25 – Y–SIZ–VID Y lapméret.
Alapérték 24 sor, cím 49118

26 – ST–FLAG
Ø=állapotsor kijelezhető.
Cím 49119

27 – BORD–VID
A képernyő keretszíne.
Alapérték Ø, a cím 4912Ø

28 – BIAS–VID
A színeltolás a paletta felső (8...16) színeihez.
Alapérték Ø, a cím 49121

29 – VID–EDIT
Az editor video lapjának csatornaszáma.
Alapérték 1Ø2, cím 49122

3Ø – KEY–EDIT
Az editor billentyűzetének a csatornaszáma.
Alapérték 1Ø5, cím 49123

31 – BUF–EDIT
Az editorpuffer mérete (256 byte-os lapokban).
Alapérték 8, cím 49124

32 – FLG–EDIT
Az editor olvasásánál használt jelző.
Alapérték 2Ø, cím 49125

33 – SP–TAPE
Kazettamentés sebessége.
Alapérték Ø (gyors), cím 49126

34 – PROTECT
Nem nulla védett file magnetofonra írásához.
Alapérték 255, cím 49127

35 – LV–TAPE
Kazettás kimeneti szint vezérlés.
Alapérték 2, cím 49128

36 – REM1
Kazettásmagnetofon távvezérlő.
Cím 49129

37 – REM2
Tartalma
nulla – kikapcsol,
nem nulla – bekapcsolt állapot. .
Cím 4913Ø

38 – SPRITE
A külső „szellem" szín prioritást vezérli.
Alapérték Ø, cím 49131

39 – RANDOM–IRQ
Minden megszakítás esetén inkrementálódik. Véletlenszám generátorként használható.
Cím 49132

Az EXOS változók alapértékét a Ø. szegmens elején lévő tábla alapján állítja be a rendszer. Ezek alapján mi magunk a SET helyett POKE, az ASK helyett PEEK segítségével módosíthatunk, ill. kiolvashatunk gépi feltétel állapotokat.

PRINT, LPRINT

Mint minden programnál, az utasítás megfogalmazásnál fordított arány áll fent a program egyszerűsége és a végrehajtás sebessége között.

A PRINT utasítás végrehajtó rutinja (5. szegmens D4Ø8H) RST 1Ø makró felhasználásával végzi el a szükséges teendőket. A kiírás-kiíratás során – minthogy nem mindegy melyik csatornára küldjük a szöveget – a csatorna számot meg kell adnunk, vagy rá kell bízni a számítógépre, hogy az alapértelmezésű (de-

fault) csatornát kijelölje. Ez az alapértelmezésű csatorna kijelölés az interpreternek több munkát ad, mintha a teljes szintaktika szerint adjuk ki a PRINT utasítást.

A PRINT # 1Ø2: alakban csatornakijelölés megoldásával kényelmesebb helyzetbe hozzuk az interpretert, így ha képernyőre írunk az utasítást sokkal gyorsabban hajtja végre.

Az LPRINT utasítás alapértelmezés szerint a nyomtatóra küldi a karaktereket. Ehhez természetesen el kell végezni a 10. ábra szerinti csatlakoztatást. A nyomtató szabványos Centronics csatlakozóval kell, hogy rendelkezzen. Csatorna kijelöléssel természetesen az LLIST utasítást akár magnetofonra, akár a hálózatba kötött másik számítógépre is ki lehet adni.

A magnetofonra (lemezegységre) kilistázott program – mint adatfile – az Enterprise szövegszerkesztőjébe visszaolvasható. Ez az utóbbi lehetővé teszi a logikus tördelésű programlisták készítését. Ha a nyomtatóra küldjük a listát, akkor a nyomtató vezérlőkarakterei segítségével szoríthatjuk a listát két margó közé. Ilyenkor a sorok vége merev geometriai szabályokhoz igazodik.

A képernyőn tördelt lista viszont negyven karakter hosszú – mint leghosszabb –, de a sor tördelését az EXOS logikai szempontok szerint végzi. Így pl. nem vág ketté egy 2–3 karakter hosszú DATA sorban tárolt adatot vagy kulcsszót. Ha a listázást a magnetofonra végezzük el, és a magnetofonról az adatállományt (listát) a szövegszerkesztőbe visszaolvassuk, akkor a képernyőn kialakult lista képét tudjuk a nyomtatóra kiküldeni. A megvalósítás módját és hatását a 10., 11. programlista mutatja be.

```
   10 OPEN #106:"probaprogram" ACCESS OUTPUT
   20 POKE 4827+PEEK(4827)+PEEK(4827+PEEK(4827))+5,42
   30 LIST #106:100-
   40 !^^  itt is csalok, hogy programkent is bemutathas
sam  a direktmodban elvegzendo teendoket
   50 CLOSE #106
   60 ! erre a bemutato kepernyon megjeleno lista alapja
n listazo programra MERGE segitsegevel toltheto a megfel
elo program
   70 END  ! de lehetne EXT "WP"
   80 !
   90 !
  100 FOR I=1 TO 9
  110    READ A$
  120    PRINT A$
  130 NEXT
  500 DATA az,Enterprise,egy,sor,meglepetest,tartogat,a,
gyanutlan,szemlelonek
```

10. lista

```
   10 OPEN #106:"probaprogram" ACCESS
OUTPUT
   20 POKE 4827+PEEK(4827)+PEEK(4827+
PEEK(4827))+5,42
   30 LIST #106:100-
   40 !^^  itt is csalok, hogy
programkent is bemutathassam  a
direktmodban elvegzendo teendoket
   50 CLOSE #106
   60 ! erre a bemutato kepernyon
megjeleno lista alapjan listazo
programra MERGE segitsegevel toltheto
a megfelelo program
   70 END  ! de lehetne EXT "WP"
   80 !
   90 !
  100 FOR I=1 TO 9
  110   READ A$
  120   PRINT A$
  130 NEXT
  500 DATA az,Enterprise,egy,sor,
meglepetest,tartogat,a,LIST
```

11. lista

SET CHARACTER

A szokásos CHR$(n) utasítás végrehajtása során – attól függően, hogy melyik képernyő módban vagyunk – vezérlőkaraktereket, színkódot és karaktereket jeleníthetünk meg. A képernyőn megjelenő karakterek képei alapértelmezésben a ROM-ban tárolt és a karakter RAM-ba áttöltött helyről kerülnek a képernyőre. Egy-egy karakter tárolására 9 byte szükséges.

A 255. szegmens 348ØH (1344Ødec) címtől kezdődő területen található a karakterkészlet. A tárolás módja olyan, hogy az egy karakterhez tartozó 9 byte egymástól 128 byte távolságra található. Ennek ismeretében már kiolvashatjuk a karakterek tárolásának módját. L. 112. program.

```
100 ! karakterek tarolasanak vizsgalata
110 LET K=13440
120 FOR KARAKTER=0 TO 127
130   FOR TARTALOM=0 TO 8
140     LET Y=SPEEK(255,K+KARAKTER+128*TARTALOM)
150     LET H=INT(Y/16):GOSUB 250
160     LET H=MOD(Y,16):GOSUB 250
170     PRINT " ";
180   NEXT
190   LET KOD=KARAKTER
200   IF KOD<32 THEN LET KOD=KARAKTER+128
210   PRINT CHR$(KOD);
220   PRINT USING "#####":KOD
230 NEXT
240 END
250 !memoriatartalom tartalmanak kiirasa hexadecimalis
alakban
260 LET H$=STR$(H)
270 IF H>9 THEN LET H$=CHR$(H+55)
280 PRINT H$;
290 RETURN
```

112. program

A tárolás módját a 12. listán követhetjük figyelemmel.

```
1C 63 6F 73 6F 63 1C ØØ ØØ    128
36 36 7F 36 7F 36 36 ØØ ØØ    129
66 3C 66 66 7E 66 66 ØØ ØØ    130
66 3C 66 66 66 66 3C ØØ ØØ    131
66 42 66 66 66 66 3C ØØ ØØ    132
18 ØØ 3C 66 7E 66 66 ØØ ØØ    133
Ø2 3C 6E 66 66 76 3C 4Ø ØØ    134
7E 6C 6C 7E 6C 6C 6E ØØ ØØ    135
Ø6 18 3C 66 7E 6Ø 3C ØØ ØØ    136
3E 6Ø 6Ø 6Ø 6Ø 6Ø 3E ØC 7C    137
6Ø 18 3C Ø6 3E 66 3E ØØ ØØ    138
18 3C 5A 18 18 18 18 ØØ ØØ    139
ØØ 18 ØC 7E ØC 18 ØØ ØØ ØØ    140
```

```
ØØ ØØ 18 24 24 24 66 ØØ ØØ   ⎍    141
7E 7E 7E 7E 7E 7E 7E 7E ØØ   ▒    142
ØE 18 18 3C 18 18 18 7Ø ØØ   f    143
7F 36 36 36 36 36 36 ØØ ØØ   ╥    144
7E 66 3Ø 18 3Ø 66 7E ØØ ØØ   Σ    145
66 ØØ 3C Ø6 3E 66 3E ØØ ØØ   ä    146
66 ØØ 3C 66 66 66 3C ØØ ØØ   ö    147
66 ØØ 66 66 66 66 3C ØØ ØØ   ü    148
18 ØØ 3C Ø6 3E 66 3E ØØ ØØ   á    149
ØØ Ø2 3C 6E 66 76 3C 4Ø ØØ   ¢    15Ø
ØØ ØØ 3E ØA 3E 68 3E ØØ ØØ   ª    151
6Ø 18 3C 66 7E 6Ø 3C ØØ ØØ   ê    152
3E 6Ø 7C 66 3E Ø6 7C ØØ ØØ   ⌠    153
3C 3C 18 7E 18 3C 66 ØØ ØØ   ⌡    154
18 18 18 18 5A 3C 18 ØØ ØØ   ↓    155
ØØ 18 3Ø 7E 3Ø 18 ØØ ØØ ØØ   ←    156
ØØ ØØ 66 66 66 66 3C 3Ø 6Ø   ⌡    157
ØØ 18 3Ø 7E 3Ø 18 ØØ ØØ ØØ   ←    158
FF FF FF FF FF FF FF FF FF   ▒    159
ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ          32
18 18 18 18 18 ØØ 18 ØØ ØØ   !    33
6C 6C 6C ØØ ØØ ØØ ØØ ØØ ØØ   "    34
ØØ 36 7F 36 36 7F 36 ØØ ØØ   #    35
ØC 3F 68 3E ØB 7E 18 ØØ ØØ   $    36
6Ø 66 ØC 18 3Ø 66 Ø6 ØØ ØØ   %    37
3Ø 4C 4C 3Ø 4E 4C 32 ØØ ØØ   &    38
18 18 18 ØØ ØØ ØØ ØØ ØØ ØØ   '    39
18 3Ø 6Ø 6Ø 6Ø 3Ø 18 ØØ ØØ   (    4Ø
18 ØC Ø6 Ø6 Ø6 ØC 18 ØØ ØØ   )    41
ØØ 18 7E 3C 7E 18 ØØ ØØ ØØ   *    42
ØØ 18 18 7E 18 18 ØØ ØØ ØØ   +    43
ØØ ØØ ØØ ØØ ØØ 18 18 3Ø ØØ   ,    44
ØØ ØØ ØØ 3E ØØ ØØ ØØ ØØ ØØ   -    45
ØØ ØØ ØØ ØØ ØØ 18 18 ØØ ØØ   .    46
```

```
ØØ Ø6 ØC 18 3Ø 6Ø ØØ ØØ ØØ   /     47
3C 66 6E 7E 76 66 3C ØØ ØØ   0     48
18 38 18 18 18 18 3C ØØ ØØ   1     49
3C 66 Ø6 3C 6Ø 6Ø 7E ØØ ØØ   2     5Ø
7E Ø6 ØC 1C Ø6 66 3C ØØ ØØ   3     51
ØC 1C 3C 6C 7E ØC ØC ØØ ØØ   4     52
7E 6Ø 7C Ø6 Ø6 66 3C ØØ ØØ   5     53
1E 3Ø 6Ø 7C 66 66 3C ØØ ØØ   6     54
7E Ø6 Ø6 ØC 18 3Ø 6Ø ØØ ØØ   7     55
3C 66 66 3C 66 66 3C ØØ ØØ   8     56
3C 66 66 3E Ø6 ØC 78 ØØ ØØ   9     57
ØØ ØØ 18 18 ØØ 18 18 ØØ ØØ   :     58
ØØ ØØ 18 18 ØØ 18 18 3Ø ØØ   ;     59
ØC 18 3Ø 6Ø 3Ø 18 ØC ØØ ØØ   <     6Ø
ØØ ØØ 3E ØØ 3E ØØ ØØ ØØ ØØ   =     61
3Ø 18 ØC Ø6 ØC 18 3Ø ØØ ØØ   >     62
3C 66 Ø6 1C 18 ØØ 18 ØØ ØØ   ?     63
3C 6Ø 3C 66 3C Ø6 3C ØØ ØØ   @     64
18 3C 66 66 7E 66 66 ØØ ØØ   A     65
7C 66 66 7C 66 66 7C ØØ ØØ   B     66
3C 66 6Ø 6Ø 6Ø 66 3C ØØ ØØ   C     67
7C 66 66 66 66 66 7C ØØ ØØ   D     68
7E 6Ø 6Ø 7C 6Ø 6Ø 7E ØØ ØØ   E     69
7E 6Ø 6Ø 7C 6Ø 6Ø 6Ø ØØ ØØ   F     7Ø
3E 66 6Ø 6Ø 6E 66 3E ØØ ØØ   G     71
66 66 66 7E 66 66 66 ØØ ØØ   H     72
3C 18 18 18 18 18 3C ØØ ØØ   I     73
Ø6 Ø6 Ø6 Ø6 Ø6 66 3C ØØ ØØ   J     74
66 6C 78 7Ø 78 6C 66 ØØ ØØ   K     75
6Ø 6Ø 6Ø 6Ø 6Ø 6Ø 7E ØØ ØØ   L     76
63 77 7F 6B 6B 63 63 ØØ ØØ   M     77
66 66 76 7E 6E 66 66 ØØ ØØ   N     78
3C 66 66 66 66 66 3C ØØ ØØ   O     79
7C 66 66 7C 6Ø 6Ø 6Ø ØØ ØØ   P     8Ø
```

```
3C 66 66 66 6E 6E 3C Ø6 ØØ   Q    81
7C 66 66 7C 78 6C 66 ØØ ØØ   R    82
3C 66 6Ø 3C Ø6 66 3C ØØ ØØ   S    83
7E 18 18 18 18 18 18 ØØ ØØ   T    84
66 66 66 66 66 66 3C ØØ ØØ   U    85
66 66 66 66 3C 18 18 ØØ ØØ   V    86
63 63 6B 6B 7F 7F 63 ØØ ØØ   W    87
66 66 3C 18 3C 66 66 ØØ ØØ   X    88
66 66 3C 18 18 18 18 ØØ ØØ   Y    89
7E Ø6 ØC 18 3Ø 6Ø 7E ØØ ØØ   Z    9Ø
66 3C 66 66 7E 66 66 ØØ ØØ   [    91
66 3C 66 66 66 66 3C ØØ ØØ   \    92
66 42 66 66 66 66 3C ØØ ØØ   ]    93
ØØ 18 3C 66 ØØ ØØ ØØ ØØ ØØ   ^    94
ØØ ØØ ØØ ØØ ØØ ØØ 7E ØØ ØØ   _    95
1C 18 ØC ØØ ØØ ØØ ØØ ØØ ØØ   `    96
ØØ ØØ 3C Ø6 3E 66 3E ØØ ØØ   a    97
6Ø 6Ø 7C 66 66 66 7C ØØ ØØ   b    98
ØØ ØØ 3E 6Ø 6Ø 6Ø 3E ØØ ØØ   c    99
Ø6 Ø6 3E 66 66 66 3E ØØ ØØ   d    1ØØ
ØØ ØØ 3C 66 7E 6Ø 3C ØØ ØØ   e    1Ø1
ØC 18 18 3C 18 18 18 ØØ ØØ   f    1Ø2
ØØ ØØ 3E 66 66 66 3E Ø6 1C   g    1Ø3
6Ø 6Ø 7C 66 66 66 66 ØØ ØØ   h    1Ø4
18 ØØ 38 18 18 18 3C ØØ ØØ   i    1Ø5
ØC ØØ ØC ØC ØC ØC ØC 6C 38   j    1Ø6
3Ø 3Ø 36 3C 38 3C 36 ØØ ØØ   k    1Ø7
18 18 18 18 18 18 18 ØØ ØØ   l    1Ø8
ØØ ØØ 36 7F 6B 6B 63 ØØ ØØ   m    1Ø9
ØØ ØØ 7C 66 66 66 66 ØØ ØØ   n    11Ø
ØØ ØØ 3C 66 66 66 3C ØØ ØØ   o    111
ØØ ØØ 7C 66 66 66 7C 6Ø 6Ø   p    112
ØØ ØØ 3E 66 66 66 3E Ø6 Ø6   q    113
ØØ ØØ 1E 38 3Ø 3Ø 3Ø ØØ ØØ   r    114
```

```
ØØ ØØ 3E 6Ø 3C Ø6 7C ØØ ØØ   s    115
ØØ 18 3C 18 18 18 ØC ØØ ØØ   t    116
ØØ ØØ 66 66 66 66 3C ØØ ØØ   u    117
ØØ ØØ 66 66 66 3C 18 ØØ ØØ   v    118
ØØ ØØ 63 6B 6B 7F 36 ØØ ØØ   w    119
ØØ ØØ 66 3C 18 3C 66 ØØ ØØ   x    12Ø
ØØ ØØ 66 66 66 66 3E Ø6 1C   y    121
ØØ ØØ 7E ØC 18 3Ø 7E ØØ ØØ   z    122
66 ØØ 3C Ø6 3E 66 3E ØØ ØØ   ä    123
66 ØØ 3C 66 66 66 3C ØØ ØØ   ö    124
66 ØØ 66 66 66 66 3C ØØ ØØ   ü    125
1C 26 66 6C 66 66 4C ØØ ØØ   ß    126
7Ø 4Ø 58 18 1E 16 1E ØØ ØØ   ┌    127
```

12. lista

Ha módosítani akarjuk valamelyik karakter képernyőn megjelenő képét, akkor a megfelelő címek tartalmát kell átírni. A címek kiszámítását és az átírást végzi el a SET CHARACTER n utasítás.

A Basic utasítás természetesen elvégzi a megfelelő adatok, megfelelő címre való bejegyzését. A megfelelő adatok, a karakterek egyes rasztersorainak bit térképét adják. A rasztersor adatait akár bináris számként, akár decimálisan is megadhatjuk. A 113. programban DATA sorba helyeztük el az új karakter alakját meghatározó byte sorozatot.

```
1ØØ ! karakteratalakito SET CHARACTER n,.... helyett
11Ø LET K=13440
12Ø PRINT "nyomd le azt a billentyut amit "
13Ø PRINT "at akarsz alakittatni"
14Ø LET A$=INKEY$
15Ø IF A$<" " THEN GOTO 14Ø
16Ø LET KOD=ORD(A$):LET KEZD=K+KOD
17Ø PRINT "a CHR$(" KOD ") --> ";
18Ø FOR I=Ø TO 8
19Ø    READ MIT
2ØØ    SPOKE 255,KEZD+128*I,MIT
21Ø NEXT
22Ø PRINT CHR$(KOD)
23Ø DATA 62,65,85,65,73,65,93,65,62
```

113. program

Ezt az új karakteralakot bármelyik – jelen esetben INKEY$-vel – kijelölt karakter helyére tölthetjük. A problémát a karakter átírásánál nem az új raszterpontok kijelölése, hanem az egyes rasztersorok nyolc bites kettes számrendszerű számjegyének meghatározása jelenti. A karakterátalakító 114. program segít az adatok meghatározásában, de abban is, hogy ezeket az adatokat közvetlenül felvegyük a programba.

```
  1 LET Q=1 ! karakteratalakito
100 CLEAR SCREEN:LET A$=" "
110 DIM B(10,10)
120 FOR F=1 TO 9
130    FOR V=1 TO 8
140       PRINT AT 4+F,16+V:"+"
150       LET B(F,V)=0
160    NEXT
170 NEXT
180 FOR L=0 TO 3 !karakterkeszlet kiirasa
190    FOR J=32 TO 63
200       PRINT AT 15+L,J-28:CHR$(32*L+J)
210    NEXT
220 NEXT
230 PRINT AT 14,0:"botkormannyal es SPACE-szel
valassz"
240 LET X=0:LET Y=0:GOSUB 640
250 PRINT AT 14,0:CHR$(161) !sorvegig torol
260 PRINT AT 2,1:"0 torol "
270 PRINT AT 3,1:"1 helyez"
280 PRINT AT 4,1:"botkormany"
290 PRINT AT 5,1:"SPACE kesz"
300 LET X=17:LET Y=5
310 DO
320    PRINT AT Y,X:CHR$(142)
330    LET Z$="+"
340    IF B(Y-4,X-16)=1 THEN LET Z$=CHR$(159)
350    PRINT AT Y,X:Z$
360    LET A$=INKEY$
370 LOOP WHILE A$=""
380 IF A$=" " THEN GOTO 470
390 LET X=X-(X<24 AND A$=CHR$(188))+(X>17 AND A$=CHR$
(184))
```

```
  400 LET Y=Y-(Y<13 AND A$=CHR$(180))+(Y>5 AND A$=CHR$(
176))
  410 IF A$="1" OR A$="0" THEN
  420    LET W=43+99*VAL(A$)
  430    PRINT AT Y,X:CHR$(W)
  440    LET B(Y-4,X-16)=VAL(A$)
  450 END IF
  460 GOTO 310
  470 INPUT AT 20,0,PROMPT "Tarol , Hiba ujra T/H    : "
:V$
  480 LET V$=UCASE$(V$)
  490 IF V$="H" THEN RUN
  500 IF V$<>"T" THEN GOTO 470
  510 TEXT
  520 PRINT Q+1;"data ";
  530 FOR F=1 TO 9
  540    LET X=0
  550    FOR V=1 TO 8
  560       LET S=B(F,V):LET X=X+S*2^(8-V)
  570    NEXT
  580    SPOKE 255,13440+Z+(F-1)*128,X
  590    PRINT X;CHR$(184);",";
  600 NEXT
  610 PRINT CHR$(184);" ":PRINT "run"
  620 POKE 4837,PEEK(4837)+1
  630 END
  640 LET Z=32+X+32*Y
  650 DO  ! kvazikurzor villogtatasa
  660    PRINT AT 15+Y,X+4:CHR$(142)
  670    PRINT AT 15+Y,X+4:CHR$(Z)
  680    LET A$=INKEY$
  690 LOOP WHILE A$=""
  700 IF A$=" " THEN RETURN
  710 LET X=X-(X<31 AND A$=CHR$(188))+(X>0 AND A$=CHR$(
184))
  720 LET Y=Y-(Y<3 AND A$=CHR$(180))+(Y>0 AND A$=CHR$(1
76))
  730 GOTO 640
```

114. program

A program sorszámait a 114. program szerint írjuk be. Az 1. programsortól a 1ØØ. sorig hagyjunk lehetőséget újabb programsorok beírására. A program a képernyőre kirajzolja a karakterek 8 × 9 képpont méretű területét. Ezen a mezőn a beépített botkormány segítségével mozgathatjuk kurzorunkat. Az adott raszterpontra érve Ø vagy 1 billentyű lenyomásával helyezhetjük, vagy törölhetjük az új képpontot.

A szóközbillentyű lenyomásával kiszállhatunk a szerkesztési üzemmódból. Ha úgy érezzük, hogy elkészültünk a „nagy művel", akkor a T billentyű segítségével befejezhetjük az új karakter szerkesztését és adatait beépíthetjük egy DATA sorba.

Az új karakter adatainak tárolását úgy végzi a program, hogy a képernyőre kiír egy sorszámot, „data" szöveget, és ezután kiírja az adatokat a programsorban szokásos módon vesszővel elválasztva. A szerkesztendő sor után még a képernyőre ír egy „RUN" szöveget. Ezután a számítógép parancsmódba kerül. A felhasználó feladata ilyenkor az, hogy a kurzort a megfelelő sorokra visszavigye és a sort RETURN utasítással lezárja. Ha a Return-billentyűt kétszer nyomjuk le, akkor a RUN szöveget is végrehajtandó parancsként értelmezi a gép, és folytathatjuk a további karakterek szerkesztését. Ezt a sok billentyű lenyomást megtakaríthatjuk, ha a program 1Ø2. sorában átdefiniáljuk, pl. a 8. funkcióbillentyűt az alábbiak szerint:

SET FKEY8 CHR$(177) & CHR$(13) & CHR$(13) & CHR$(13)

Ebben az esetben az OK megjelenése után elég a nyolcas funkcióbillentyűt lenyomni.

A program felülírja önmagát. Az első sorba mindig újabb értéket ír be. A számlálót tovább növeli, mert így újabb és újabb programindítás után tudni fogja az utoljára beszerkesztett programsor számát. A karakterek átdefiniálása után a képernyőn lévő összes adott kódszámú karakter azonnal átvált az új alakra. A karakter megjelenítés ezen tulajdonságát érdekes, látványos képi hatások elérésére használhatjuk fel.

A képernyőn jobbra-balra mozgó vonaldarabokkal áramlás hatását érhetjük el a 115. program segítségével.

```
1ØØ ! aramlas vizszintesen
11Ø CLEAR SCREEN
12Ø FOR X=1 TO 38 !kepernyo feltoltes  A,B betukkel
13Ø   FOR Y=1 TO 7
14Ø     PRINT AT Y,X:"A"
15Ø     PRINT AT Y+14,X:"A"
16Ø     PRINT AT Y+7,X:"B"
17Ø   NEXT
18Ø NEXT
19Ø FOR I=Ø TO 7 !karakterek atdefinialasa ciklusban
mindig ugyanazt masra
```

```
200    LET A=2^I:LET B=2^(7-I)
210    SET CHARACTER 65,A,A,A,A,A,A,A,A,A
220    SET CHARACTER 66,B,B,B,B,B,B,B,B,B
230 NEXT
240 GOTO 190
```

115. program

A programban az A és a B betűt ciklusban átdefiniáljuk karakter magasságú függőleges vonaldarabbá, melyben a nyolc bit szélességű karakter mindig másik helyén található. A függőleges áramlás illúzióját keltő kép állítható elő a 116. programmal.

```
100 ! aramlas fuggolegesen
110 CLEAR SCREEN
120 FOR Y=1 TO 11
130    PRINT AT Y,18:">AAABBB<"
140    PRINT AT Y+11,18:">BBBAAA<"
150 NEXT
160 DIM A(10)
170 FOR I=2 TO 9
180    LET A(I)=0
190 NEXT
  200 LET A(1)=255
  210 FOR I=1 TO 9 !ciklusban karakter atdefinialas
  220    SET CHARACTER 65,A(1),A(2),A(3),A(4),A(5),A(6),
A(7),A(8),A(9)
  230    SET CHARACTER 66,A(9),A(8),A(7),A(6),A(5),A(4),
A(3),A(2),A(1)
  240    LET A(I+1)=A(I):LET A(I)=0 ! a rasztersorokat
 eggyel tovabb leptetem
  250 NEXT
  260 GOTO 200
```

116. program

A két program birtokában már előállítható egy, a kereten körbefutó áramlás-kép is. A karakterek tárolásának ismeretében a karakter RAM-ból visszaolvashatjuk azok bitmintáját. Ennek segítségével nagyméretű betűt – betűket – állíthatunk elő. A 117. program az INPUT-ként beírt, max. négy karakter hosszú szöveget a képernyő közepére nagyítva kiírja.

```
 100 !   nagybetu
 110 LET K=13440
 120 INPUT PROMPT "kiirando szoveg: ":V$
 130 LET S=LEN(V$):LET X1=11-S*4
 140 IF S>4 THEN 120 !sajnos ez csak ennyit tud
 150 CLEAR SCREEN
 160 FOR G=1 TO LEN(V$)
 170   LET A$=V$(G:G) ! a szoveg leszeletelese
 180   LET L=K+ORD(A$) !az aktualis karakter reszter-
sorainak tarolasi helye hol kezdodik a memoriaban
 190   FOR C=0 TO 8
 200     LET A=SPEEK(255,L+128*C) !vegyuk kezelesbe az
erteket
 210     LET B$="":CALL BIN !binaris ertek alapjan a
raszterpontok betoltesenek megallapitasa
 220     PRINT AT C+8,X1+G*8:B$ ! egy rasztersor
kiirasa
 230   NEXT
 240 NEXT
 250 RUN 110
 260 DEF BIN
 270   LET O=128
 280   FOR I=1 TO 8
 290     !LET T$=CHR$(159) ! amilyen karakterbol
akarjuk felepiteni a betut
 300     !LET T$=A$
 310     LET T$=CHR$(154) !most ezt valasztottam
 320     IF (A/O)<1 THEN LET T$=" "
 330     LET B$=B$&T$
 340     LET A=MOD(A,O):LET O=O/2 !dec-bin atalakitas
 350   NEXT
 360 END DEF
```

117. program

A nagyítás olyan mértékű, hogy minden pixel helyett egy karaktert ír ki. Azt, hogy a pixelek helyett milyen karaktert írjon ki a program, a T$ értékadásával jelölhetjük ki.

SET STATUS

A státuszsor egyrészt a szerkesztés közben kijelzi az egyébként 2Ø48 byte hosszú editor puffer még szabad helyeinek számát, ha ez már 1ØØ byte-nál kevesebb, másrészt jelzi, hogy hányadik program tekintendő aktuálisnak. Ezzel a sorral óvatosan kell bánni! Ha beállítása nem szabályos, az első RESET hatására teljes hideg indítást hajt végre a rendszer. A státuszsorra kiírandó üzenet szövege attól függ, hogy mi a BFDFH memóriacím tartalma. Ha ennek a címnek a tartalma egyenlő Ødec, akkor láthatjuk a szokásos információkat. Amennyiben erre a címre nullától és 42dec-től eltérő értéket töltünk, akkor a státuszsor láthatatlan. A státuszsor kezelését a Ø. szegmens D33DH címen kezdődő rutinja végzi.

Amennyiben a BFDH tartalmát, a számítógépes szakirodalom által ST–FLAG-nek nevezett értéket 42dec-re átírjuk, akkor a státuszsoron az Enterprise IS–BASIC szerzőinek neve jelenik meg. A szerzők nevének kódjai a ROM Ø. szegmens C3C3H címen kezdődő területén is megtalálható. Ennek a területnek a kiírását biztosítja a 118. program.

```
100 ! szerzok neve
110 ! kiir status sorra
120 POKE 49119,42
130 !kiolvassuk a ROM-bol
140 PRINT "0.szegmens C3C3h-tol :"
150 LET KEZD=17347
160 FOR I=KEZD TO KEZD+33
170   LET K=SPEEK(0,I) BAND 127
180   PRINT CHR$(K);
190 NEXT
200 PRINT
```

118. program

A számítógépgyártó cégek nemcsak a szoftver készítőire, hanem saját nevükre is büszkék. Az ENTERPRISE név minden ROM szegmensen megtalálható, akár mint kiírásra kerülő üzenet, akár csak mint jelzés. A 119. program segítségével kiírhatjuk ezeket a szövegeket.

```
100 ! ENTERPRISE (c) szovegek a ROM-ban
110 CLEAR TEXT
120 FOR X=1 TO 5
130   READ S,K,V
140   PRINT S;".szegmensen:":PRINT
```

```
150    FOR I=K TO V
160      PRINT CHR$(SPEEK(S,I));
170    NEXT
180    PRINT :PRINT
190 NEXT
200 IF INKEY$="" THEN 200
210 DATA 1,19839,19902,1,26377,26412
220 DATA 1,26419,26428,4,17712,17721
230 DATA 5,31275,31356
```

119. program

A képernyőre csalhatjuk a (©) szöveget a:

HELP ISL1985

hívással is.

A státuszsort azért bizonyos korlátozásokkal mi is használhatjuk. A program futása után a parancsmódra való áttéréskor az 5. szegmens C2FCH–C333H közti rutinja felújítja a státuszsort. A program futás közben viszont a státuszsorra mi is írhatunk, akár a saját programunk „Copyright" jelzését. A 120. program

```
100 ! amig fut a status sor az enyem
110 LET K$="(c) ENTERPRISE" ! tetszoleges, max 28
karakter hosszu szoveg lehet
 120 IF LEN(K$)>28 THEN STOP
 130 LET S$="----------------"
 140 LET P$=S$(1:(28-LEN(K$))/2)
 150 LET K$=P$&K$&P$:LET K=LEN(K$)
 160 FOR P=1 TO K
 170   POKE 48828+P,ORD(K$(P:P))
 180 NEXT
 190 TEXT
 200 FOR F=1 TO 20 STEP .2
 210   PRINT AT 24,(15+12*SIN(F)):"enterprise"
 220 NEXT
 230 GRAPHICS
 240 FOR I=10 TO 300 STEP 5
 250   PLOT 640,370,ELLIPSE 2*I,I
 260 NEXT
 270 GOTO 190
```

120. program

11Ø. sorában a kiírandó – max. 28dec karakter hosszú K$ – szövegnek értéket kell adni.

Ez a szöveg a 16Ø–18Ø. programsorok segítségével kerül a státuszsorra és mindaddig ott marad, amíg a program fut.

Sorparaméter-tábla (LPT)

Az Enterprise memóriakiosztása, memóriával való gazdálkodása igen összetett szabályok alapján történik. A sorparaméter-tábla B9ØØH címhez képest épül fel (ez a Ø. szegmens D26ØH címen tanulmányozható), azaz a képernyőtartalom a memóriában e címhez képest megtalálható. A szöveges képernyő adott sorának tartalmát a kezdőcímhez képest LPS változó által megadott értéken keresztül kell keresnünk. A sor tartalmát a sor tárolási címének SC ismeretében sorban kiolvashatjuk. A sorok, a képernyőtartalom rendre kiolvasható a 121. program segítségével.

```
1ØØ ! LP alapjan kepernyotartalom
11Ø TEXT
12Ø PRINT "Ø123456789Ø123456789Ø123456789Ø1234567"
13Ø FOR I=2 TO 16
  14Ø   PRINT USING "##":I
  15Ø NEXT
  16Ø FOR I=1 TO 2ØØ
  17Ø   PRINT AT 2+RND(15),4+RND(34):CHR$(32+RND(9Ø)) !
legyen mit keresni, feltoltjuk a kepernyo egy reszet
  18Ø NEXT
  19Ø PRINT AT 18,1:"                                  "
  2ØØ PRINT AT 19,1:"                                  "
  21Ø PRINT AT 18,1:"";
  22Ø INPUT PROMPT "sor,oszlop: ":S,O
  23Ø CALL TARTALOM
  24Ø GOTO 19Ø
  25Ø DEF TARTALOM
  26Ø   LET LPS=47369+16*S
  27Ø   LET SC=PEEK(LPS+4)+256*PEEK(LPS+5) !
LP tablaban sor cimenek meghatarozasa
  28Ø   LET KC=SC+O ! karakter tarolasi cime a
LP tablaban
```

```
290    LET A=SPEEK(255,KC):LET A$=CHR$(A)
300    PRINT AT 21,20:"tartalom: ";A$
310 END DEF
```

121. program

Igen jól felhasználható ennek az ismerete játékprogramok, képernyőtartalom azonosítás, találkozás, ütközés számontartására.

A képernyőre a PRINT utasítással, a PRINT AT utasítással tetszőleges helyre írhatunk. Adott esetben célszerű lehet a kiírást nem PRINT utasítással, hanem a képernyőmemóriára közvetlen beírással POKE segítségével elvégezni. Ehhez a fent említett sorparaméter-tábla címből kell az adott célterület címét meghatározni.

A képernyőterületek a 252–255. szegmensen „szétszórva" helyezkedhetnek el. Az aktuális sor, oszlop koordináta címét a 25Ø. sor segítségével határozza meg a program. E cím ismeretében helyezi el az adott karaktert a memóriában a 122. program.

```
100 ! kepernyore POKE
110 TEXT
120 PRINT AT 19,1:CHR$(25)
130 PRINT AT 18,1:CHR$(25):PRINT AT 18,1;"";
140 INPUT PROMPT "hova:sor,oszlop: ":S$,O$
150 LET S=VAL(S$):LET O=VAL(O$)
160 IF S>23 OR S<1 OR S<>INT(S) THEN GOTO 120
170 IF O>39 OR O<Ø OR O<>INT(O) THEN GOTO 120
180 INPUT PROMPT "mit: ":A$
190 CALL TARTALOM
200 GOTO 120
210 DEF TARTALOM
220    LET LPS=47360+16*S !lp tabla eleje
230    LET SC=PEEK(LPS+4)+256*PEEK(LPS+5) !sor cime a
LP tablaban
240    LET KC=SC+O ! a sorban levo karakter tarolasa-
nak cime a line parameter tablaban
250    SPOKE (252+INT(KC/16384)),MOD(KC,16384),ORD(A$)
! jo lenne ha a Z8Ø ott latna ahol a NICK
260 END DEF
```

122. program

A sorparaméter-tábla elérhető a 82H, 83H portokon keresztül is. A portra küldendő értékek kiszámítás kacifátosnak tűnik, de az előző programok, ill. annak ismeretében, hogy a képernyő szerkesztést a NICK chip által saját memóriacímein keresztül végzi már nem is olyan rémisztő.

A sorparaméter-tábla felépítéséhez a kezdőcímet el kell osztani 4Ø96-tal. Az eredmény egész része és a 192 között bináris vagy műveletet kell végezni. Az így kapott érték kiküldendő a 83H portra. A másik byte-ot úgy kell kiszámítani, hogy a kezdőcím 4Ø96-tal osztott értékének törtrészét be kell szorozni 256-tal. Ezt már ki lehet küldeni 82H-ra.

Szorzás, szorzótábla

Minden alapfokú Basic tanfolyamon elkészítik a résztvevők a kis 1×1-et. Ha a számokat szépen, helyi érték szerint helyesen akarjuk kiíratni, akkor már a USING feltételeit is figyelembe kell venni. Ilyet láthatunk a 123. programban.

```
100 ! szorzotabla
110 CLEAR SCREEN
120 SET STATUS OFF
130 FOR X=1 TO 10
140    FOR Y=1 TO 10
150       PRINT USING "###":X*Y;
160    NEXT
170    PRINT :PRINT
180 NEXT
```

123. program

A számítógép számtani műveletek, így szorzás elvégzésére is alkalmas. A gyors számoláshoz egy sor trükköt, programozási fogást tartalmaznak a beépített gépi kódú rutinok.

Az Enterprise IS–BASIC-je a szorzást BCD számok között decimális aritmetikával végzi el, egy segédtáblázat segítségével. Ez a szorzótábla az 5. szegmens elején helyezkezik el. Olvassuk ki mi is ezt a szorzótáblát a 124. program segítségével!

```
 1ØØ REM belso szorzotabla  ahogy az interpreter is
hasznalja
 11Ø CLEAR SCREEN:PRINT
 12Ø PRINT
```

```
130 INPUT PROMPT " x= ":X
140 IF X<Ø OR X>9 THEN GOTO 130
150 INPUT PROMPT " y= ":Y
160 IF Y<Ø OR Y>9 THEN GOTO 150
170 LET HELY=1Ø*X+Y+16396
180 LET SZORZAT=SPEEK(5,HELY)
190 LET DEC_ALAK=1Ø*INT(SZORZAT/16)+MOD(SZORZAT,16)
200 PRINT X;"*";Y;"=";DEC_ALAK
210 GOTO 120Ø
```

124. program

A szorzótáblát úgy használja a számítógép, hogy a két egyjegyű tényezőt egymás mellett kétjegyű számként fogja fel, és ez a kétjegyű szám adja az eredmény táblázatbeli sorszámát. A sorozatot mi is kiemelhetjük a fenti algoritmus segítségével a szorzótáblából (125. program).

```
100 !   belso szorzotabla kiiratasa
110 OUT 177,5:CLEAR SCREEN:PRINT
120 FOR L=16396 TO 16494 STEP 10
130   FOR J=0 TO 9
140     LET S=PEEK(L+J) !ez eredendoen BCD ertek
150     LET S=10*INT(S/16)+MOD(S,16) !BCD -> DEC atsza
mitas
160     PRINT USING "##  ":S;
170   NEXT
180   PRINT
190 NEXT
```

125. program

(Kedves olvasó gondolj bele, egy ilyen összetett, sokat tudó számítógépnek is tudnia kell az 1 × 1-et. Így tán nekünk is illik egészen egyszerű általános iskolás tananyagokat ismerni, még a mai kor – számítógéppel kiszolgált – világában is!)

Szögfüggvények

Az Enterprise számítógép a szögfüggvények csaknem teljes választékának kiszámítására megfelelő kulcsszóval rendelkezik. A szögfüggvény argumentumát a 2Ø14 rendszerváltozó első bitjének értékétől függően radiánban, vagy fokban értelmezi. A szögfüggvényeket a sin szögfüggvény segítségével származtatja.

A *szinusz* szögfüggvényt (a szög megfelelő tartományba való konvertálása után) a

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}$$

hatványsorral közelíti. (L. 1. szegmens CB82H)

A *koszinusz* szögfüggvényt a

$$\cos(x) = \sin(x + \pi/2)$$

összefüggés alapján határozza meg az 1. szegmens C5144 címen kezdődő rutin.

A *koszekáns* függvény értélmezését $1/\sin(x)$. A számítást a 1. szegmens CA8BH rutinja, míg a *szekáns* függvényt az $1/\cos(x)$-et az 1. szegmens CB7CH rutinja határozza meg.

A *tangens* függvény a $\sin(x)/\cos(x)$-re vezethető vissza. A $\cos(x)$ pedig mint láttuk kifejezhető $\sin(x)$ ismeretében. A számítást az 1. szegmens CC9BH rutinja végzi el.

Mint láthatjuk minden szögfüggvény a szinusz szögfüggvényen keresztül határozható meg. Ennek a rutinnak viszont van egy korlátja.

A teljes körülfordulásnál nagyobb szögeket a szögfüggvény számítás szempontjából X=MOD (szög, 2*π)-vel kell figyelembe venni. A szinusz rutin a szöget csak

$$-57{,}2958 < x < 57{,}2957 \text{ radián között}$$

azaz

$$-3282{,}8 < x < 3282{,}8 \text{ fok közötti}$$

tartományban képes feldolgozni.

Ha várhatóan e határon kívül lévő értékek szögfüggvényeit kell meghatároznunk, akkor a maradék részt mi magunknak kell előállítani a függvény argumentumában.

Még szerencse, hogy ez földi méretekben ritkán okoz problémát. A 126. program segítségével – a gömbháromszögtani összefüggések felhasználásával – két, szélességi és hosszúsági koordinátával adott, földrajzi hely távolságát határozhatjuk meg.

```
100 ! foldrajzi_helyek_tav
110 PRINT "szogek fokban!"
120 OPTION ANGLE DEGREES
130 PRINT "elso hely:"
```

```
140 INPUT PROMPT "szelesseg (+eszaki)     ":F1
150 IF ABS(F1)>180 THEN GOTO 140
160 INPUT PROMPT "hosszusag (+Gr keletre) ":H1
170 IF ABS(H1)>180 THEN GOTO 160
180 PRINT "masodik hely:"
190 INPUT PROMPT "szelesseg (+eszaki)     ":F2
200 IF ABS(F2)>180 THEN GOTO 190
210 INPUT PROMPT "hosszusag (+Gr keletre) ":H2
220 IF ABS(H2)>180 THEN GOTO 210
230 PRINT "a tavolsag a gombfeluleten merve:"
240 LET T=SIN(F1)*SIN(F2)+COS(F1)*COS(F2)*COS(H2-H1)
250 LET L=ACOS(T)
260 PRINT L*111.32;"km"
```

126. program

Ha már a földgömbbel foglalkozunk, érdemes beírni a 127. programot, melynek segítségével a 12Ø. soron megadott S1, S2, S3 szögekkel elforgatott gömb axonometrikus képét rajzolhatjuk meg.

```
1ØØ ! foldgomb
11Ø NUMERIC X2,Y2
 12Ø LET S1=-1:LET S2=.4:LET S3=Ø ! a megrajzolando
gomb terbeli elforgatasanak szogertekei
 13Ø LET R=16Ø
 14Ø GRAPHICS
 15Ø SET INK 255
 16Ø ! a forgatasi matrix, helyvektorok kiszamitasahoz
 17Ø LET F11=COS(S2)*COS(S3)
 18Ø LET F12=COS(S1)*SIN(S3)+SIN(S1)*SIN(S2)*COS(S3)
 19Ø LET F13=SIN(S1)*SIN(S3)-COS(S1)*SIN(S2)*COS(S3)
 2ØØ LET F21=-COS(S2)*SIN(S3)
21Ø LET F22=COS(S1)*COS(S3)-SIN(S1)*SIN(S2)*SIN(S3)
22Ø LET F23=SIN(S1)*COS(S3)+COS(S1)*SIN(S2)*SIN(S3)
23Ø LET F31=SIN(S2)
24Ø LET F32=-SIN(S1)*COS(S2)
25Ø LET F33=COS(S1)*COS(S2)
26Ø PLOT 64Ø,34Ø;ELLIPSE 2*R,2*R
27Ø LET JEL=Ø
28Ø FOR H=Ø TO 3 STEP .45
```

```
290    FOR I=Ø TO 6.4 STEP .2
300      CALL RAJZ
310    NEXT
320    PLOT X2,Y2,
330 NEXT
340 FOR I=-1.22 TO 1.22 STEP .34
350    FOR H=Ø TO 6.4 STEP .2
360      CALL RAJZ
370    NEXT
380    PLOT X2,Y2,
390 NEXT
400 DEF RAJZ
410    LET Z=SIN(I):LET X=COS(I)*COS(H):LET Y=COS(I)*S
IN(H)
420    LET X1=F11*X+F21*Y+F31*Z
430    LET Y1=F12*X+F22*Y+F32*Z
440    LET Z1=F13*X+F23*Y+F33*Z
450    LET X2=64Ø+2*R*X1:LET Y2=34Ø+2*R*Z1
460    IF Y1<Ø THEN LET JEL=1:PLOT X2,Y2, !lathatosag
470    IF Y1<Ø THEN EXIT DEF
480    IF I=Ø OR JEL=1 THEN
490      LET JEL=Ø:PLOT X2,Y2,
500    END IF
510    PLOT X2,Y2;
520 END DEF
```

127. program

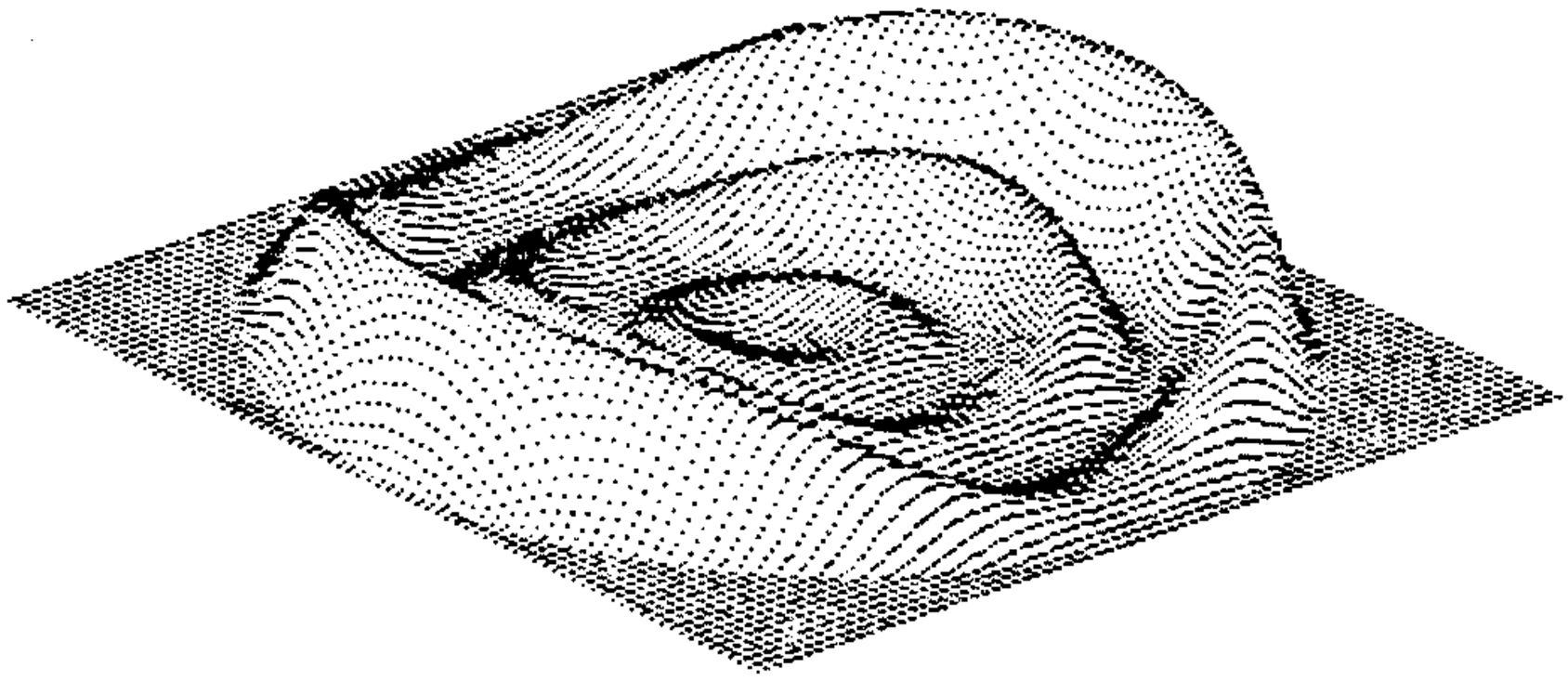
A gömb, ill. felületére rajzolt hosszúsági és szélességi körök elforgatási (és ebből adódó láthatósági) értékeit a 15Ø–23Ø. sorok forgatási mátrixa határozza meg. Bátran változtassuk a 12Ø. sor kiindulási adatait, hogy különböző irányból szemlélhessük a gömbünket. Figyeljünk a PLOT utasítások fénysugár be-, ill. kikapcsoló hatására, mert könnyen összekuszálhatjuk a rajzolt ellipszis íveket.

A térhatású ábrák egyik szokásos változatát a 21. ábra mutatja. A 128. program segítségével készíthetjük el azt, vagy hozzá hasonlót.

```
100 ! kalap lathatosaggal
110 DIM M(320)
120 FOR I=0 TO 320
130   LET M(I)=0
140 NEXT
```

```
150 GRAPHICS
160 SET INK 255
170 LET S=2 ! pontok surusege 1<=S<8
180 LET Q=.77:LET A=160:LET D,R=0
190 LET W=100:LET F=4
200 FOR X=A TO A+154 STEP S
210   LET L=D+(X-A)/2
220   LET P=SQR((((((X-A)/W)-Q)^2)+(((R/W)--Q)^2))
230   LET Y=10*(1-COS(MOD(6*F*P,6.28)))*(P<.75)
240   LET Y=L-Y*(1.2+COS(F*((X-A)/W)))
  250   IF M(X)>Y OR Y<0 THEN GOTO 270 !mert akkor ez ta
karasban van
  260   LET M(X)=Y:PLOT X*4,Y*4 !jelold a takarasi hatar
erteket, de ezt meg rajzoljad meg
  270 NEXT
  280 LET R=R+S:LET A=160-R:LET D=D+S/2
  290 IF A<6 THEN STOP
  300 GOTO 200
```

128. program



21. ábra

A programban lévő képletek kísérletezéssel, vagy a szakirodalomban található, más gépekre írt összefüggések felhasználásával írhatók fel.

A láthatóságot, a takart pontok elhagyását azzal a trükkel valósítja meg a program, hogy a rajzot „elölről" a képernyő alján kezdi el. Amennyiben a később rajzolt függvénygörbe egy *x* koordinátájához tartozó *y* értéknél korábban már előfordult nagyobb *y* érték, akkor ezt a pontot nem rajzoltatjuk ki. A már felrajzolt *y* koordináta értékeket az M jelű tömbben tároljuk. Ha ezt a megoldást nem alkalmazzuk, akkor sokkal kuszább, áttekintetlenebb ábrát kaptunk volna.

TANH

A hiperbolikus függvények a SINH, COSH, TANH a természetes logaritmus, az *e* hatványaiból számíthatók

$$\text{SINH} = \frac{e^x - e^{-x}}{2}$$

a

$$\text{TANH} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$ összefüggéssel.

A TANH végrehajtó rutinja a Ø. szegmens CACBH címén található. Ez a programrész iskolapéldája a jól szervezett RST 1Ø rutinhívások sorozatának. Érdemes vele megismerkedni, mert szép programozási stílus és a RST 1Ø rutinok használata sajátítható el segítségével.

CCAB	D7	RST 1Ø hívás
CCAC	99	szintaktikai elemzés és argumentum kiértékelés *x* a Basic verembe
CCAD	A7	e^x eredmény *x* helyére a veremben
CCAE	15	e^x még egyszer a verembe helyezi
CCAF	AD	az utolsó bejegyzés reciproka
CCBØ	97	egyes aritmetikai regiszterbe tölti az utolsó értéket (e^{-x})
CCB1	98	a kettes aritmetikai regiszterbe tölti az utolsó értéket (e^x)
CCB2	95	visszatölti a verembe az e^x-t
CCB3	94	visszatölti a verembe az e^{-x}-t
CCB4	ØD	$e^x - e^{-x}$
CCB5	95	visszatölti a verembe az e^x-t
CCB6	94	visszatölti a verembe az e^{-x}
CCB7	ØC	$e^x + e^{-x}$
CCB8	ØF	az utóbbi verem bejegyzéssel $(e^x + e^{-x})$ elosztja a még élő előző bejegyzést $(e^x - e^{-x})$
CCB9	ØØ	RST rutinok vége a veremben a TANH(*x*)
CCBA	C9	rutinból vissza

Ennek a rutinnak Basic-ben való megírásához egy főprogramot kell elképzelnünk, mely sorozatban hív meg CALL, vagy GOSUB utasítással alprogramokat. A strukturáltság ilyen foka már a programozás művészetét jelenti! Erre kell törekednünk és akkor nem lesz nehéz a Basic-tól eltérő programnyelvek elsajátítása.

Teknős grafika

A LOGO programozási nyelv grafikus utasítás készletének, eljárásainak egyszerűsített megvalósítását teszi lehetővé az IS–BASIC néhány utasítása.

A legegyszerűbben úgy is fogalmazhatnánk, hogy a szokványos derékszögű koordináta-rendszerbeli ábrázolásmód mellett előtérbe kerül a polár koordinátákkal történő helymeghatározás, ill. rajzolás.

A teknős grafika FORWARD (előre) kezdő iránya alapértelmezés szerint függőlegesen felfelé mutat (l. memóriacímek tárgyalásánál az E18H-nál). Ha azt az ismeretet felhasználjuk, akkor a PLOT ANGLE kezdő irány kijelölés mellett lehetőségünk adódik az adott tárolási hely értékének beállítására. A kezdő irányt ennél a 129. programnál a gépi kódú program 3. byte-ja által kijelölt érték határozza meg (l.: RST 1Ø/92).

```
 1ØØ !   teknos kezdo irany
 11Ø ALLOCATE 1Ø
 12Ø CODE MC=HEX$("d7,92,Ø4,96,19,Øe,ØØ,c9")
 13Ø !--------------------^^ a RST1Ø/92 altal felhasz-
nalando konstans sorszama
 14Ø CALL USR(MC,Ø)
 15Ø GRAPHICS
 16Ø SET INK WHITE
 17Ø OPTION ANGLE DEGREES
 18Ø PLOT 56Ø,3ØØ;
 19Ø FOR I=1 TO 4
 2ØØ    LET R=2Ø
 21Ø    FOR J=1 TO 13
 22Ø       PLOT FORWARD R;:PLOT RIGHT 9Ø;
 23Ø       LET R=R+16
 24Ø    NEXT
 25Ø NEXT
```

129. program

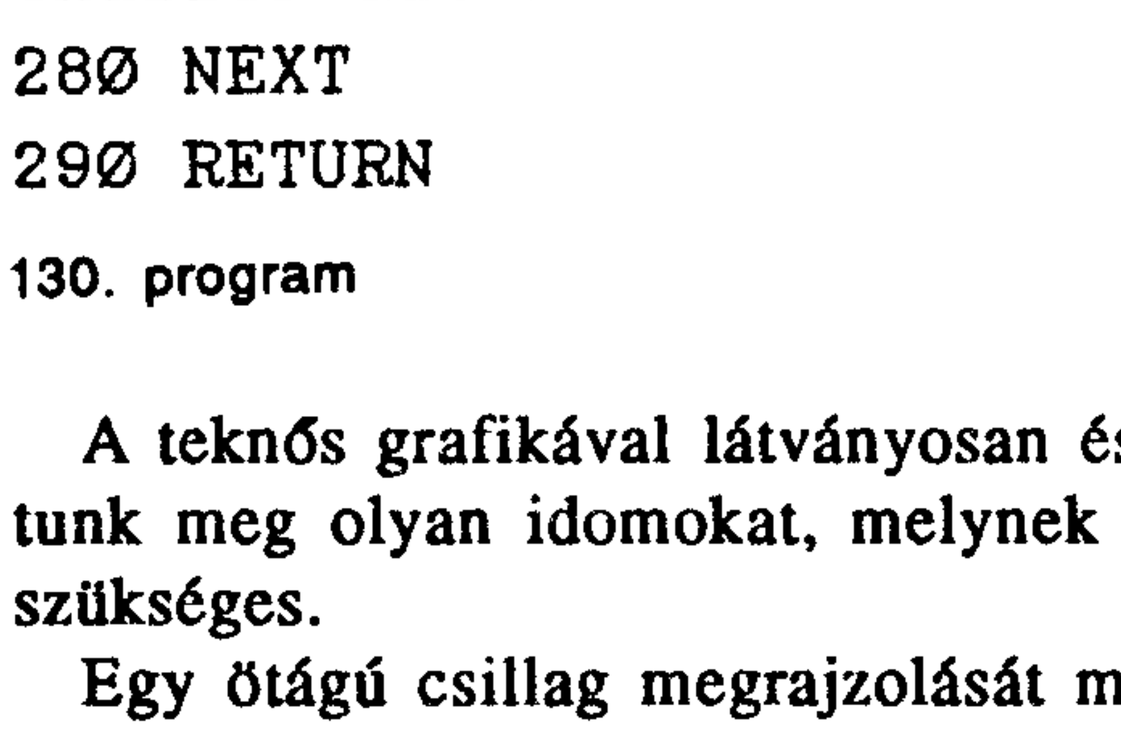
A kezdő irány beállítását úgy is elvégezhetjük, hogy INPUT utasításként beírjuk annak értékét és azt a tárolás helyéről áttöltjük az E18H címre. A konstans értéke ui. a változótáblában és a teknős grafika kezdő irányát tároló memóriaterületen azonos alakban található. A konkrét megvalósítást a 130. program mutatja.

```
 100 !   teknos kezdo irany
 110 LET W=3.14 ! kell ez az ertekadas, mert igy
tudni fogom, hogy a tablaban hol tarolja
 120 INPUT PROMPT "indul fok: ":W
 130 LET W=W*PI/180
 140 GOSUB 240
 150 GRAPHICS
 160 SET PALETTE BLUE,YELLOW
 170 OPTION ANGLE DEGREES
 180 PLOT 600,300;
 190 FOR I=20 TO 300 STEP 20
 200    PLOT FORWARD I;
 210    PLOT RIGHT 90;
 220 NEXT
 230 GOTO 120
 240 !      IRANY
 250 LET K=PEEK(566)+256*PEEK(567)+5 !itt tarolja a
teknos kezdo iranyt
 260 FOR I=0 TO 5
 270    POKE 3610+I,PEEK(K+I) ! valtozo tablabol,
rendszervaltozo teruletre attolt
280 NEXT
290 RETURN
```

130. program

A teknős grafikával látványosan és igen egyszerű rövid programmal rajzolhatunk meg olyan idomokat, melynek felrajzolásához valamilyen szabályszerűség szükséges.

Egy ötágú csillag megrajzolását minden kisiskolás megteszi (22. ábra).



22. ábra

A szabály kézenfekvő. Ha például a bal alsó csúcsnál indulunk, akkor a vízszintessel 72°-os szögbe kell előremenni 1 egységet és ott jobbra 144°-ot fordulva ismét 1 egységet mindaddig, amíg a kezdőpontba nem érünk vissza. Ennyi az egész. A 131. program ennek kidolgozását mutatja be.

```
100 !   otagu egy vonallal
110 OPTION ANGLE DEGREES
120 GRAPHICS
130 LET H=360
140 SET PALETTE BLACK,RED
150 FOR X=24 TO 1200 STEP 400
160   PLOT X,440;
170   PLOT ANGLE 0;
180   FOR Z=0 TO 5
190     PLOT FORWARD H;
200     PLOT RIGHT 144;
210   NEXT
220   PLOT FORWARD 0,  ! ne huzzad tovabb a vonalat
230 NEXT
```

131. program

Talán bonyolultabb, hiszen több szakaszból áll az az ötágú csillag, melynek kerületét keresztező vonalak nélkül rajzoljuk meg. Itt a szabály az, hogy az előző irányhoz képest egyszer jobbra 144°-kal, egyszer balra 72°-kal kell haladni. Ennek az idomnak a megrajzolását teszi lehetővé a 132. program.

```
100 !   otagu
110 OPTION ANGLE DEGREES
120 GRAPHICS
130 SET PALETTE GREEN,RED
140 PLOT 600,440;
150 PLOT ANGLE 72;
160 LET H=280
170 FOR Z=0 TO 5
180   PLOT FORWARD H;
190   PLOT RIGHT 144;
200   PLOT FORWARD H;
210   PLOT LEFT 72;
220 NEXT
230 PLOT BACK 0
240 PLOT 650,440,PAINT
```

132. program

Ebben a csillag rajzolási gondolatmenetben eljuthatunk a Koch-görbéig (133. program).

```
100 !   hopehely/ KOCH-gorbe
110 GRAPHICS
120 OPTION ANGLE DEGREES
130 SET PALETTE BLUE,YELLOW
140 FOR S=20 TO 64 STEP 44
150   READ X,Y
160   LET A=0:PLOT X,Y;
170   PLOT ANGLE A;
180   FOR I=1 TO 6
190     CALL VONAL
200     PLOT LEFT 60;
210     CALL VONAL
220     PLOT RIGHT 120;
230   NEXT
240   PLOT FORWARD 0,
250 NEXT
260 PLOT 500,430,PAINT
270 END
280 DEF VONAL
290   PLOT FORWARD S;
300   PLOT LEFT 60;
310   PLOT FORWARD S;
320   PLOT RIGHT 120;
330   PLOT FORWARD S;
340   PLOT LEFT 60;
350   PLOT FORWARD S;
360 END DEF
370 DATA  530,430,340,540
```

133. program

Ennek az idomnak a megrajzolásánál már szembetűnhet a teknős grafika utasításkészlet alkalmazásának előnye.

Írjuk be ugyanennek a síkidomnak a megrajzolását biztosító másik programot, a 134. programot.

```
100 !   hopehely  elore kiszamitott koordinatakkal
110 LET A=.5:LET B=SQR(3)/2
120 LET H=.5:LET N=3:LET F=0
```

```
130 DIM X(1000),Y(1000)
140 LET X(0),X(3)=360:LET X(1)=950
150 LET Y(0),Y(1),Y(3)=500
160 LET X(2)=650:LET Y(2)=5
170 GRAPHICS :LET F=F+1
180 SET PALETTE BLACK,WHITE
190 LET X1=X(0):LET Y1=Y(0)
200 FOR J=1 TO N
210   LET X2=X(J):LET Y2=Y(J)
220   PLOT X1,Y1;X2,Y2
230   LET X1=X2:LET Y1=Y2
240 NEXT
250 PLOT 620,380,PAINT
260 IF F=5 THEN STOP
270 PRINT "gombnyomasra tovabb"
280 IF INKEY$="" THEN 280
290 PRINT :PRINT "turelem"
300 LET N=N*4
310 FOR J=N TO 4 STEP-4
320   LET X(J)=X(J/4):LET Y(J)=Y(J/4)
330 NEXT
340 FOR J=0 TO N-4 STEP 4
350   LET C=(X(J+4)-X(J))/3
360   LET D=(Y(J+4)-Y(J))/3
370   LET X(J+1)=INT(X(J)+C+H)
380   LET Y(J+1)=INT(Y(J)+D+H)
390   LET X(J+2)=INT(A*C-B*D+X(J+1)+H)
400   LET Y(J+2)=INT(B*C+A*D+Y(J+1)+H)
410   LET X(J+3)=INT(X(J)+2*C+H)
420   LET Y(J+3)=INT(Y(J)+2*D+H)
430 NEXT
440 GOTO 170
```

134. program

Ennek ennyire bonyolultnak nem kellene lennie, ha csak egy Koch-görbét rajzolna. Viszont ez a változat az n. rendű görbe megrajzolása után – egy billentyű lenyomása után – (ugyanazzal a programrészlettel) megrajzolja az *n*+1. rendűt. A teknős grafika alkalmazására a CALL címszó alatt a rekurzióval kapcsolatban is találhatunk példákat.

TIME$

Az idő mérését, kijelzését, beállítását az EXOS 31–34 funkciók teszik lehetővé. Élményszámba menő, ahogy az időt lépteti, ahogy az időpont beállításnál az adatok korrektségét vizsgálja a rendszerprogram. Ezt a részt az 1. szegmens E4ØFH–E52AH területen találhatjuk meg.

Vegyünk egy példát és alkalmazási lehetőséget! A bekapcsolás (RESET) óta eltelt időt a 135. program segítségével másodpercben kaphatjuk meg.

```
100 ! bekapcsolas ota eltelt ido
110 LET ORA=VAL(TIME$(1:2))
120 LET PERC=VAL(TIME$(4:5))
130 LET MASODPERC=VAL(TIME$(7:8))
140 LET IDO=3600*ORA+60*PERC+MASODPERC
150 PRINT TIME$;" => ";IDO
160 !
170 PRINT "kozvetlenul memoriabol kiolvasva:"
180 PRINT 256*PEEK(49137)+PEEK(49136)
```

135. program

Ugyanezt az értéket kiolvashatjuk a memóriából, ha tudjuk azokat a címeket, ahol a másodpercszámláló elhelyezkedik.

Az eltelt idő másodpercben a

PRINT 256 * PEEK(49137) + PEEK(49136)

paranccsal írható ki.

A TIME$ óra, perc, másodperc értékeit BCD alakban pedig a BF72H–BF74H (49Ø1Ø–19Ø12dec) területről olvashatnánk ki. A képernyőre ki is írhatjuk az időt. Ilyenkor mint óra működik a számítógépünk. A szép nagybetűs kiíráshoz megfelelő VIDEO módot kell választanunk. A digitális órát a 136. program valósítja meg.

```
100 !   digitalis ora
110 REM ido beallitas
120 SET VIDEO Y 2
130 SET VIDEO COLOUR 3
140 SET VIDEO MODE 1
150 OPEN #100:"video:"
160 OPEN #101:"video:"
170 CLEAR SCREEN
```

```
180 LET C=100:CALL VALTAS
190 LET C=101:CALL VALTAS
200 GOTO 180
210 DEF VALTAS !villanasmentes idopont kijelzes
220    SET #C:INK 0
230    PLOT #C:140,71;
240    SET #C:INK 255
250    PRINT #C:CHR$(26)
260    SET #C:INK 0
270    PLOT #C:140,71;
280    SET #C:INK 255
290    PRINT #C:TIME$
300    DISPLAY #C:FROM 1 TO 2 AT 10
310 END DEF
```

136. program

Vannak konzervatív emberek, akik a mutatós (analóg kijelzésű) órát jobban szeretik. Az ő igényeiket elégíti ki a 137. program.

```
100 !    ora
110 INPUT PROMPT "ora:perc:mp=":T$
120 TIME T$
130 GRAPHICS :LET R=300:LET P1=-1:LET O1=-1
140 OPTION ANGLE DEGREES
150 SET INK 255:PLOT 640,350;
160 PLOT ELLIPSE 350,350
170 FOR I=0 TO 360 STEP 30
180    PLOT 640+R*COS(I),350+R*SIN(I);
190    PLOT ANGLE I;:PLOT FORWARD 40
200 NEXT
210 LET O=VAL(TIME$(1:2))
220 LET P=VAL(TIME$(4:5))
230 PLOT 640,350;:PLOT ELLIPSE 8,8
240 IF P1=P THEN GOTO 350
250 SET INK 0:PLOT 640,350;
260 PLOT ANGLE 90-6*P1;:PLOT FORWARD 250
270 SET INK 255:PLOT 640,350;
280 PLOT ANGLE 90-6*P;:PLOT FORWARD 250
290 IF P/5<>INT(P/5) THEN GOTO 350
```

```
300 PLOT 640,350; !5 percenkent oramutato is lep
310 SET INK 0:PLOT 640,350,
320 PLOT ANGLE 90-30*O1-.5*P1;:PLOT FORWARD 190
330 SET INK 255:PLOT 640,350;
340 PLOT ANGLE 90-30*O-.5*P;:PLOT FORWARD 190
350 LET P1=P:LET O1=O
360 PRINT AT 2,17:TIME$
370 GOTO 210
```

137. program

Természetesen az a legpraktikusabb, ha az órát folyamatosan láthatjuk a képernyő adott területén (pl. a felső sarokban), attól függetlenül, hogy éppen programsorokat írunk-e be, vagy kész programunkat futtatjuk.

Ez az Enterprise gépeken is elérhető a megszakítási rendszerbe való minimális belenyúlással. Ennek módjával, lehetőségeivel a gépi kódú kötet foglalkozik részletesen, de a 138. program az operációs rendszer mélyebb ismerete nélkül is felhasználható.

```
100 PROGRAM "STOPPER"
110 POKE 49119,42
120 FOR N=0 TO 33
130    POKE 45616+N,32
 140 NEXT
 150 POKE 540,128
 160 POKE 541,1
 170 CODE M=HEX$("CB,6A,C8,21,00,00,7E,3C,27,3C,27,77,2
0,06,3F,23,7E,3C,27,77,DB,B3,F5,3E,01,D3,B3,3A,00,00,21,
3A,B2,CD,6F,C5,2B,36,2C,3A,01,00,CD,6F,C5,F1,D3,B3,C9")
 180 CODE MB=HEX$("F3,36,80,23,36,01,FB,C9")
 190 CODE MK=HEX$("F3,36,00,23,36,00,FB,C9")
 200 DEF STOPPER
 210    CALL USR(MK,49133)
220    POKE 0,0
230    POKE 1,0
240    CALL USR(MB,49133)
250    FOR T=1 TO 548
260    NEXT
270    CALL USR(MK,49133)
280 END DEF
```

138. program

A 11Ø–14Ø. sorok egy adott RAM területet jelenítenek meg az állapotsorban, majd ezt szóközökkel töltik fel. A továbbiakban a program letárolja a megszakítási rendszert kiegészítő gépi kódú rutint, ami – aktivizálása után – az 1 Hz-es megszakítások során beolvassa az óra állapotot és a szokásos kijelzési formátumban beírja azt az állapotsorba.

A funkció aktivizálásához csak a felhasználói megszakítási rutin címét (esetünkben Ø18ØH) kell beírni a megfelelő rendszerváltozóba (USR–ISR). Ezt nem tanácsos BASIC-ban megtenni, mert ha a cím alsó és felső byte-jának beírása között megszakítás történik, az a rendszer összeomlásához vezethet. Ezért a címbeírást is egy rövid gépi kódú rutin (MB) végzi el, letiltott interrupt mellett.

Futtatás után az óra megjelenik (a bal felső sarokban) és folyamatosan jár. Akkor, ha a programot NEW-val kitöröljük, a gépen parancsmódban vagy programmódban tovább dolgozunk. Szükség esetén a kijelzett állapotsor lapozható a szokásos BASIC üzenet (POKE 49119,Ø), vagy az óra között (POKE 49119,42). Vigyázni kell, mert a rendszerterület átírása miatt a RESET gomb egyszeri megnyomása is hidegindítást eredményez.

VAL

A karaktersorozat numerikus értékét állítja elő. A karaktersorozatot az első nem numerikus karakterig értelmezi. Ilyen alkalmazásra azokban a Basic tájszólásokban van különös jelentősége, ahol a szám és karakter tárolása más memóriaigénnyel jelentkezik. Számunkra is érdekes lehet, ha a számítógép eredendően stringnek értelmez olyan értékeket, amit mi számnak gondolunk. Ilyen pl. a bekapcsolás óta eltelt időt jelző TIME$. Ezt a függvényt viszont alkalmazhatjuk olyan esetben is, amikor időtartamot akarunk mérni anélkül, hogy nulláznánk az aktuális időt.

Ebben az esetben a TIME$ függvény vég és kezdő értékét egymásból ki kell vonni. Minthogy stringekkel matematikai művelet nem végezhető, ez egyszerűen nem megy. Előbb számszerűsíteni kell az egyébként számjegyekből álló karaktersorozatot. A 139. program segítségével felszeleteljük a TIME$ függvényt és kiszámítjuk az egyes időadatok értékét. A kezdő és vég időpontok különbsége így már előállítható.

```
100 !val TIME$ kombinacio stopperhez
110 PRINT :PRINT
120 PRINT "idomeres indul = SPACE"
130 IF INKEY$<>" " THEN GOTO 130
140 LET X$=TIME$
150 GOSUB 250
160 LET KEZD=T
```

```
 170 PRINT "idomeres vege  = SPACE"
 180 IF INKEY$<>" " THEN GOTO 180
 190 LET X$=TIME$
 200 GOSUB 250
 210 LET IDO=T-KEZD
 220 PRINT IDO "sec"
 230 GOTO 110
 240 END
 250 LET T=3600*VAL(X$(1:2))+60*VAL(X$(4:5))+VAL(X$(7:8
))
260 RETURN
```

139. program

A VAL függvénynek van egy SINCLAIR gépeknél alkalmazott változata, amely a stringet mint függvénykapcsolatot képes értelmezni és azt kiértékelni. A kiértékelés során elvégzi a műveleteket a kifejezésben lévő változók aktuális értékével és függvénykapcsolatával. Ez olyan esetben biztosít nagy előnyt, ha egy függvényt INPUT adatként akarunk a programnak további elemzésre, feldolgozásra, ábrázolásra átadni. A VAL függvény ilyen értelmezését az IS–BASIC sem ismeri, hiszen helyette adja a DEF-el való közvetlen függvény megadás lehetőségét.

Ez ugyan így igaz, de itt is újra meg újra át kellene írni azt a programsort, melyben a függvényt definiáltuk. Miért ne tehetnénk ezt meg programból. A 140. program ennek egy lehetséges megoldását adja.

```
 100 GOTO 150
 110 LET Y= ! vigyazz   felulirodik az elso sorokon ne
valtoztass
 120 RETURN
 130 !
 140 !
 150 LET LJ=2.2:PRINT "y=";:INPUT F$
 160 LET F$=UCASE$(F$)
 170 GOSUB 400
 180 GOSUB 300
 190 PRINT "Xmin=";:INPUT XA
 200 PRINT "Xmax=";:INPUT XF
 210 PRINT "lepes";:INPUT L
 220 FOR X=XA TO XF STEP L
 230    GOSUB 110
 240    PRINT X,Y
```

```
250 NEXT
260 PRINT
270 GOTO 150
280 !
290 !
300 ! sor beiras a megfelelo memoriateruletre
310 FOR I=1 TO LEN(P$)
320   LET S$=P$(I:I):LET C=4845+I
330   POKE C,ORD(S$)
340   IF S$="^" THEN POKE C,27
350 NEXT
360 POKE I+4845,1
370 RETURN
380 !
390 !
400 ! kvazi tokenizalas
410 LET Q=LEN(F$):LET P$="":LET V=0
420 DO
430   LET V=V+1
440   LET S$=F$(V:V)
450   IF S$>="A" THEN
460     ! fuggveny kereses
470     RESTORE 760
480     READ D
490     FOR Y=1 TO D
500       READ B$
510       IF F$(V:V+2)=B$ THEN LET P$=P$&"#"&B$:LET V
=V+2
520     NEXT
530     IF F$(V:V)="X" THEN LET P$=P$&"!X"
540   END IF
550   IF S$>="(" AND S$<"0" THEN LET P$=P$&CHR$(ORD(S
$)-32)
560   IF S$="^" THEN LET P$=P$&S$
570   IF VAL(S$)>0 THEN
580     !   szam atalakitas
590     LET LJ=VAL(F$(V:Q))
600     LET H=LEN(STR$(LJ))-1
```

```
610      IF INT(LJ)<>LJ OR ABS(LJ)>9999 THEN GOTO 660
620      LET E$=CHR$(MOD(LJ,256))
630      LET M$=CHR$(INT(LJ/256))
640      LET P$=P$&CHR$(194)&E$&M$
650      GOTO 710
660      LET P$=P$&CHR$(198)
670      LET K=PEEK(566)+PEEK(567)*256+6
680      FOR R=K TO K+5
690        LET P$=P$&CHR$(PEEK(R))
700      NEXT
710        LET V=V+H
720      END IF
730 LOOP UNTIL V=Q
740 RETURN
750 ! a program altal felismert fuggvenyek tablaja.
Bovitheto!
760 DATA 10,ABS,ATN,COS,COT,EXP
770 DATA LOG,SGN,SIN,SQR,TAN
```

140. program

A VAL függvény helyett itt a 2. programsor átírásával, programból történő felülírásával adjuk meg az egyváltozós függvény egyenletét. Minthogy programból a POKE utasítással visszük be ide a megfelelő értékeket, az első 4 sort nem tanácsos megváltoztatni. Mindenesetre az Y= karakterek programfutás közbeni memóriacímét nagyon pontosan kell ismernünk! Az INPUT-ként közölt függvénystringet most már csak ide át kell mozgatni, de úgy, hogy az a programsor tárolási, tokenizálási szabályainak megfelelően helyezkedjen el az előre lefoglalt memóriaterületen. Ehhez ismerni kell a tokenizálási szabályokat, a numerikus értékek programsorban való tárolásának módját és talán mindenek előtt azt, hogy a számítógép minden nem macskakörmök között elhelyezkedő és DATA sorban lévő karaktert nagybetűsít. Így a program először nagybetűsíti a függvénystringet, majd lejátssza a tokenizálást. (Ez a programrész utánozza a 5. szegmens C5AØ címen kezdődő rutint.) Mivel a rendszerprogram is alkalmaz egy konverziót, nekünk is módosítanunk kell a matematikai műveletek, zárójelek, relációjelek CHR$ kódját. Az érdeklődők ezt a konverziót az 5. szegmens C636–C6C1 területen követhetik. Mi ebben a programban csak a középiskolai matematika gyakorlatban előforduló relációjelek átalakítására készítettük fel a programot. Így a DATA sorban is csak azt a 10 függvényt vettük fel, amit leggyakrabban szoktunk alkalmazni.

A számokat külön kell átalakítani és be kell jegyezni a programsorba az egész és valós típusú számok jelzésére szolgáló értéket. A számátalakításhoz felhasználhatjuk a szám változóterületen tárolt alakját, ehhez előbb azt le kell tárolni

és ismerni kell a tárolás helyét. Ehhez azt a trükköt alkalmazhatjuk, hogy ha a sor feldolgozása közben számot találunk, akkor azt a változótábla első helyére letároljuk, majd onnan kiolvasva tároljuk az átmeneti P$-ben.

Ha az egész INPUT-ként adott függvénystringet feldolgoztuk és áttöltöttük a P$-be, akkor már áttölthetjük ennek a stringnek karaktereit a program második sorába. Ezután a program elején lévő szubrutint már különböző *x* értékekkel meghívhatjuk és visszatéréskor a függvény értékét kapjuk az Y változóban. Erről meg is győződhetünk a próbafutás során.

Természetesen, ha már ennyire végiggondoltuk a tokenizálás folyamatát, akkor akár közvetlenül az INPUT pufferen keresztül, ill. az INPUT pufferben is elvégezhetjük a tokenizálást. Az egész tokenizáló rutinba ilyenkor az 5. szegmens C3B9H címén kell belépni. Az előbbi program körülményesnek látszó tokenizáló rutinját még összetettebben, még szerteágazóbb vizsgálatok elvégzésével valósítja meg a meghívott rendszerprogram. Ennek megvalósítását a 141. program mutatja be.

```
  1ØØ ! inputkent adott fuggveny kiertekelese VAL
  11Ø ALLOCATE 5Ø
  12Ø CODE MC1=HEX$(" db,b3,f5,3e,Ø5,d3,b3,dd,cb,ØØ,ae,
21,Ø1,Ø3,22,14,Ø2,cd,b9,c3,f1,d3,b3,c9")
  13Ø CODE MC2=HEX$(" db,b3,f5,3e,Ø5,d3,b3,dd,cb,ØØ,ae,
21,Ø1,Ø3,22,14,Ø2,cd,bc,c3,f1,d3,b3,c9")
  14Ø INPUT PROMPT "y=":F$
  15Ø LET F$="val="&F$
  16Ø LET N=LEN(F$)
  17Ø LET KC=768 ! Ø3ØØh
  18Ø FOR I=1 TO N
  19Ø   POKE KC+I,ORD(F$(I:I))
  2ØØ NEXT
  21Ø POKE KC+I,Ø !sorveg jelzo
  22Ø LET X=1
  23Ø CALL USR(MC1,Ø) !tokenizalas
  24Ø PING
  25Ø ! kiertekeles
  26Ø GRAPHICS
  27Ø OPTION ANGLE DEGREES
  28Ø SET PALETTE Ø,GREEN
  29Ø FOR X=8 TO 1ØØØ STEP 8
  3ØØ   CALL USR(MC2,Ø)
  31Ø   ! koordinata erteket, transzformaciot mindig
az aktualis fuggveny alapjan valtoztatni kell
```

```
320    ! most y=sin(x) a feltetelezett fuggveny
330    PLOT X,320+250*VAL
340 NEXT
```

141. program

Ebben a programban nemcsak a tokenizáláshoz, de a kiértékeléshez is közvetlenül meghívjuk a ROM-ban található rutint. (Az ilyen meghívásokat, ROM rutin felhasználásokat kellő körültekintéssel kell elvégezni. Ennek ismertetésétől most eltekintünk, de az Enterprise gépi kódú programozása című összeállításban megismerkedhetünk ennek szabályaival.

WHEN

Nagyon kényelmes és egyben rettentő veszélyes a számítógépek, így az Enterprise hibakezelő lehetősége. Veszélyes, mert láthatatlanná teszi a hibát. Ha általános hibakezelést végzünk, akkor egy közönséges szintaktikai hibát sem tudunk megtalálni. Ezért csak kipróbált „belőtt" program esetében építsük be a hibaelfogó rutint, rutinokat és mindig csak az adott várható hibára korlátozzuk annak hatását. Kellemetlen hiba, ha rajzolás, függvényábrázolás közben a pont koordinátája a képernyőn kívülre kerül. Az ábrázolt függvény pl. egy ferde hajtás függvényképe olyan is lehet, mely ugyan egy bizonyos helyen kilép a képernyőről, de azután visszakerülne oda. Ilyen esetben két lehetőség is kínálkozik. A 142. program kiszámítja az ábrázolandó program koordinátáit. Ha az képernyőterületre esik, akkor elvégzi az ábrázolást, ha pedig nem akkor átlépi a rajzolási utasítást tartalmazó programsort.

```
100 !hibaelfogas rajzolas kozben
110 GRAPHICS
120 SET PALETTE BLUE,YELLOW
130 OPTION ANGLE DEGREES
140 PLOT 0,500;1239,500 !tengely
150 FOR X=0 TO 720 STEP 4
160    LET Y=500+280*SIN(X)
170    IF Y<0 OR Y>719 THEN GOTO 200
180    IF X<0 OR X>600 THEN GOTO 200
190    PLOT 2*X,Y
200 NEXT
```

142. program

A 143. program azt a lehetőséget mutatja be, amikor a WHEN EXCEPTION USE struktúrát vesszük igénybe, de ebben a struktúrába csak a (szintaktikailag, matematikai műveletek szempontjából kifogásolhatatlan) PLOT utasítást foglaljuk be.

```
100 REM hiba elfogas
110 GRAPHICS
120 SET PALETTE BLACK,WHITE
130 OPTION ANGLE DEGREES
140 PLOT 0,500;1239,500
150 FOR X=0 TO 720 STEP 8
160   LET Y=SIN(X)
170   WHEN EXCEPTION USE HIBA_ELFOGO
180     PLOT X*2,500+280*Y
190   END WHEN
200 NEXT
210 !
220 HANDLER HIBA_ELFOGO
230   PRINT EXSTRING$(EXTYPE)
240   CONTINUE  !ne torodj vele,folytasd
250 END HANDLER
```

143. program

A STOP billentyű lenyomása is egyfajta hibát jelent a számítógép életében. A STOP billentyű hatásának kiiktatása, a program, vagy egyes részeinek megszakíthatóságának megakadályozása fontos lehet. A STOP billentyűt, annak hatását iktatja ki a 144. program. A STOP letiltására a POKE címszónál az EXOS változók kapcsán is találhatunk ötletet.

```
100 ! STOP letiltas
110 WHEN EXCEPTION USE NO_STOP
120   FOR I=1 TO 100
130     PRINT I;"na most allitsd le ha tudod !"
140   NEXT
150 END WHEN
160 END
170 HANDLER NO_STOP
180   IF EXTYPE<>9229 THEN 200
190   RETRY
200 END HANDLER
```

144. program

178,— Ft

A könyv azokhoz szól, akik az alapszintű BASIC programozás után tovább akarnak lépni. Bemutatja az Enterprise azon sajátosságait, amelyek a feladatok megoldásához a leginkább szükségesek, és alternatív programokat közöl.
A könyv főbb témakörei: Képernyőkezelés, Grafika, Függvényábrázolás, Programtárolás, Utasítások végrehajtása, Számábrázolás—aritmetika.

*

Enterprise 128-as tulajdonosoknak ajánljuk:
Brányi László: Enterprise 128-as ROM visszafejtése
Felföldi József — Sz. Lukács János: Gépi kódú programozás

NOVOTRADE