Horváth Tibor – Kiss Marietta

Számítástechnika iskolásoknak II.

Nemzeti Tankönyvkiadó

HORVÁTH TIBOR–KISS MARIETTA

SZÁMÍTÁSTECHNIKA ISKOLÁSOKNAK II.

NEMZETI TANKÖNYVKIADÓ, BUDAPEST

Bírálták

FILA GYULA

VARGA JÁNOS

Felelős szerkesztő

SZALAY SÁNDOR

Anyanyelvi lektor

FALUSSY ANNA

ISBN 963 18 7225 4

ELŐSZÓ

Az iskolásoknak készült tankönyvsorozat második kötetét tartod kezedben.

Az előző kötetből elsajátíthattad a DOS-, NORTON-alapismereteket. Megismerkedhettél a számítógépvírusokkal, s azok eltávolításának lehetséges módozataival. Tanultál a Quick BASIC alaputasításairól, s az ott tanultak alapján már képes vagy egyszerű programok elkészítésére.

Ebben a kötetben az előzőekben megszerzett ismereteidre építünk!

Megismerkedhetsz a *SZÁMÍTÓGÉPES GRAFIKA* alapjaival. Ebben a könyvben több, a grafikai ismeretekre épülő játékprogram elkészítéséhez is segítséget adunk.

Ebben a kötetben tárgyaljuk meg az adatrögzítés egy lehetséges módját a *FÁJLKEZELÉS* című fejezetben.

A programozás gyakorlati megoldásai a programnyelvtől független *PROGRAMOZÁSI ALGORITMUSOKRA* épülnek. Ezekből a legalapvetőbbeket szintén megtalálod a könyvben. Segítségükkel több száz adat kezelését, rendezését valósíthatjuk meg.

A mindennapi gyakorlat során sokszor van szükségünk az adatok nyomtatón való megjelenítésére. A *NYOMTATÓ PROGRAMOZÁSA* fejezetben ennek gyakorlati megvalósítását sajátíthatod el.

A függelékben megtalálod a Quick BASIC leggyakrabban használt utasításainak leírását, illetve a programozás során felmerülő hibaüzenetek magyar nyelvű értelmezését. Ebben a részben szerepel több mintaprogram konkrét leírása is.

Mindezen ismeretek elsajátításához mindenkinek sok kitartást kívánunk:

a szerzők

ISMÉTLŐ FELADATOK

MS–DOS

A feladatok megoldását a kipontozott részre írd!

1. Mi a DOS?

   .....

   .....

2. Az általad használt számítógép felépítése:
   A mikroprocesszor:
   típusa: .....
   frekvenciája: ..... MHz
   A RAM mérete: ..... Mbyte
   A monitor típusa: .....
   A HDD mérete: ..... Mbyte
   Az FDD mérete: ..... Mbyte

3. Mit jelent a „kompatibilis" kifejezés?

   .....

4. Mi a feladata a COMMAND.COM fájlnak, illetve az IO.SYS és az MSDOS.SYS rejtett fájloknak?

   .....

   .....

5. Milyen paranccsal lehet úgy formázni egy lemezt, hogy rákerüljenek a rendszerfájlok?

   .....

6. Húzd alá kékkel azon fájlkiterjesztéseket, amelyek elindíthatók a számítógépen! Karikázd be pirossal, amelyeket valamely programozási nyelv használ!
   .EXE .BAS .TXT .COM .BAT .PAS .DAT .SYS .LST

7. Melyik paranccsal adhatod meg az AUTOEXEC.BAT fájlban az alkönyvtárak keresési útját?

   .....

8. Mit jelentenek a DIR parancs hatására megjelent alábbi kiírások?
   ADATOK DOC 23419 10–18–93 3:42p

   ....

9. Az alábbi parancsok között van hibás. Javítsd ki, majd írd le, mit jelentenek!
   DIR A: .....
   DIR C:/W .....
   DIR C:\DOS\p .....
   DIR C:\DOS A:h .....
10. A főkönyvtár két alkönyvtára BASIC, illetve PROGRAM. Írd le azt a parancsot, amellyel átmásolhatjuk a BASIC alkönyvtár összes 'A' betűvel kezdődő .BAS kiterjesztésű állományát a PROGRAM alkönyvtárba!
    .....
11. Mit jelentenek az alábbi parancsok?
    VOL .....
    LABEL .....
12. Készíts egy BASIC.BAT fájlt DOS-ban, amely a Quick Basic programot indítja!

NORTON COMMANDER 4.0

13. Mit jelentenek az alábbi funkcióbillentyűk?
    F1 .....
    F2 .....
    F3 .....
    F4 .....
    F5 .....
    F6 .....
    F7 .....
    F8 .....
    F9 .....
    F10 .....
14. Pótold a hiányzó billentyűparancsokat!
    Meghajtóváltás a bal panelen .....
    Meghajtóváltás a jobb panelen .....
    Tömörítés .....
    Kicsomagolás .....
    Fájlkeresés .....
    Előző DOS-parancsok aktiválása .....
    EGA-VGA képernyő 24/43 sor .....
    Alkönyvtár gyors elérése .....
    Beállítások mentése .....
    A kijelölt fájl tartalmát nyomtatóra írja .....
    A másik panelra írja a lemezinformációkat .....

BASIC

Az alábbi programozási feladatokat számítógépen oldd meg!

15. Határozd meg a következő kifejezések értékét, majd írd le az eredményt!

11 + 13 : 2	(11 + 13) : 12	(45 – 27) : (28 · 3 – 72)
.....	.....	.....

16. A következő programrészletek hibáit javítsd ki!

```
X = 2A+3*3,76
A = ((A+B)*2+A+B)^2
3 = SZ+2*(Y+X)/(A+B)
A = 7*(C%5+10)+39
4A = 5*3,14
```

17. Melyik azonosítóhoz melyik érték tartozik? Kösd össze!

```
A = 3/4*5
B = 3/(4*5)
C = 3*5/4
D = 3/4/5
E = 3*4/5
F = (3/4)*5
```

2.25
3.75
0.15
2.40

18. Mi lesz az *X* és az *Y* kifejezések értéke az adott *A*; *B*; *C* értékek esetében?

```
A = 2.5          B = .4          C = 3
X=(A*B + C)*B + C + 3*C – A/5    X =
Y=(((A + B)*(A – 1)*10)^2 + ((A*C)^2 – B^2))^2    Y =
```

19. A programok begépelése nélkül döntsd el, mit írnak ki a képernyőre?

```
A = 5
B = 3
A = A + 2
B = A
C = A + B
PRINT "C=";C
```

.....

```
A = 1.2
B = 3 * A
A = A + 2
X = A : A = B : B = X
C = A*2 + B/2
PRINT "C=";C
```

.....

20. Készítsd el az alábbi programokat:
    - három beolvasott számot összeszoroz;
    - kiszámítja három beolvasott egész szám átlagát;
    - egy beolvasott tizedes tört alakú számot két tizedesjegyre kerekít;
    - két előre megadott szám közötti véletlen egész számot állít elő.

21. A programok begépelése nélkül állapítsd meg, mit írnak ki a képernyőre?

```
FOR K=0 TO 10
    PRINT K,K*K
NEXT K
```

.....

```
FOR K=0 TO 100
  IF K/7=INT(K/7) THEN PRINT K;
NEXT K
```

.....

22. Készíts programot:
   - amely kiírja az első 10 darab 12-vel osztható természetes számot;
   - amellyel teleírhatjuk a képernyő első 5 sorát előre megadott betűkkel;
   - amely 101-től 50-ig csökkenő sorrendben az 5-tel osztható természetes számokat írja ki;
   - ami a felhasználó által megadott értékig a természetes számok összegét írja ki.

23. Próbáld meg egyetlen ciklussal kiíratni ugyanazokat a számokat, amiket az alábbi két ciklus előállít!

```
I = 2                                          .....
20 PRINT I*3                                   .....
IF I < 10 THEN I = I + 1 : GOTO 20             .....
FOR J=2 TO 19                                  .....
    PRINT J/2                                  .....
NEXT J                                         .....
```

24. Mi a hiba az alábbi programokban? A javítást a keretbe, a futási eredményt a keret alá írd a kipontozott részre!

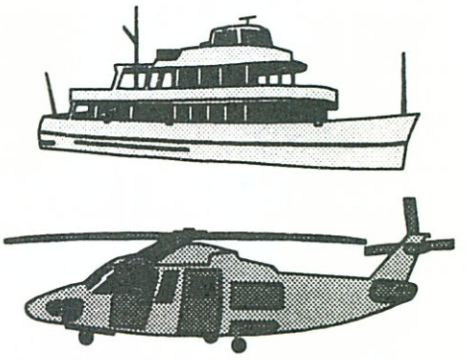
INPUT A,B **T=A*B** **K=2*A+B** **PRINT "Kerület=K"**	**DO** **INPUT A$** **UNTIL A$<=>"n"** **PRINT** ....
.....	.....

25. Készíts programot, amely 10 darab –10 és +10 közé eső véletlen számot állít elő!
26. Készíts programot, amely 100 db véletlenszerűen előállított kódú jelet a képernyő véletlenszerűen előállított helyére tesz!
27. Kockajáték! A gép 1 és 6 közötti véletlen egész számokat állít elő. Az első 3 dobás eredménye a tiéd, a másik három a partneredé. Az nyer, akinek több pontja van. A program kínáljon további játéklehetőséget! A nyertesnek játsszon le egy dallamot!

28. DATA-sorokban tárold barátaid nevét, címét és telefonszámát. Tervezz programot, amely egy név beírására kiírja a hozzá kapcsolódó adatokat!

QUICK BASIC GRAFIKA

Az előzőekben a Quick BASIC utasításai segítségével betűket, számokat, speciális kódokat írattunk a képernyőre. Ezeket összefoglaló néven karakterkészletnek neveztük. A karakterkészlet segítségével a szöveges (text) képernyőn oldottuk meg programozási feladatainkat, egyszerű ábrákat is készítettünk.



Az alábbi rajzok elkészítése a karakterkészlet alapján már nem lehetséges. Ezek túllépik az eddigi megoldási lehetőségeket, s csak új utasítások segítségével lehetséges megrajzolásuk. Ezen új utasítások a számítógépes grafika körébe tartoznak.

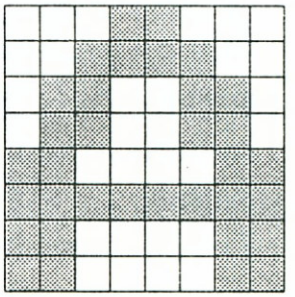
A PC-khez csatlakoztatott monitorok kétféle kijelzési módban használhatók:

- TEXT (szöveges) mód.
- GRAPHICS (grafikus) mód.

A képernyő mindkét üzemmódban sorokra és oszlopokra van osztva. A képernyő egy adott pontjának azonosítására az eddigi programokban az oszlop és a sor sorszámát használtuk. A Quick BASIC indítása után a képernyő szöveges módba kerül. A használható oszlop sorszámának kezdő értéke 1, végértéke 40 vagy 80 lehetett a beállítástól függően. A sorok számértékét 1-től 25-ig használtuk. A karakterek, szövegek megjelenítéséhez a LOCATE (oszlop, sor) parancs segítségével beállítottuk a kurzor helyét, majd a PRINT utasítással kiírattuk a szükséges adatokat. Ebben a módban a legkisebb megjeleníthető méret egy karakter.

A grafikus módban a legkisebb programozható egység a pont (pixel), amelynek mérete az adott monitor és monitorvezérlő kártya típusától függ. E kettő együtt határozza meg a monitor felbontóképességét, vagyis hány pontot tud megjeleníteni a képernyőn egymás mellett oszlopokban, illetve egymás alatt sorokban. Ha például VGA monitorod van ugyanilyen vezérlőkártyával, akkor a képernyőn 640 oszlopban és 480 sorban jelenítheted meg a pontokat.

E grafikus képpontok kifényesítése, illetve elsötétítése segítségével lehetséges a rajzolás. A monitorvezérlő kártya ezekből a pontokból állította elő az eddig használt karaktereket a képernyőn. Az ábra az **A** betű grafikus pontokból álló képének egy lehetséges rajzát mutatja. A többi karakter is ugyanilyen módon jelenik meg a TEXT-képernyőn. Az üres négyzetek a nem látható, míg a satírozottak a bekapcsolt grafikus képpontokat jelölik.



A GRAFIKA BEKAPCSOLÁSA

A következő egyszerű program segítségével a szöveges képernyőn jelenítünk meg egy karaktert:

Locate (10,45): Print Chr$(219)

A 10. sor 45. oszlopában jelenítjük meg a ■ karaktert.

Ugyanezt a grafikus képernyőn a következők szerint valósítjuk meg:

Screen 12 *Bekapcsoljuk a grafikus képernyőt.*

Pset (10,45),10 *Megjelenítünk a 10. oszlop 45. sorában egy zöld színű pontot.*

Ez a megjelenített pont egy pixel mérete.
A grafika bekapcsolásának utasítása:

SCREEN mód[,szín]

A monitortól és a vezérlőkártyától függően a grafikus képernyő más-más képernyő-felosztású lehet. Az ezekben használható maximális oszlop-, illetve sorértékeket a következő táblázat tartalmazza. Az oszlop és a sor értékei nullától kezdődnek!

KÓD	MÓD	FELBONTÁS	KÉPERNYŐ
0	TEXT képernyő	40 vagy 80 oszlop, 25 sor	CGA,EGA,VGA
1	Grafikus képernyő	320 * 200 grafikus pont	CGA,EGA,VGA
2	Grafikus képernyő	640 * 200 grafikus pont	EGA,VGA
3	Grafikus képernyő	720 * 348 grafikus pont	HERCULES
4	Grafikus képernyő	640 * 400 grafikus pont	OLIVETTI
7	Grafikus képernyő	320 * 200 grafikus pont	EGA,VGA
8	Grafikus képernyő	640 * 200 grafikus pont	EGA,VGA
9	Grafikus képernyő	640 * 350 grafikus pont	EGA,VGA
10	Grafikus képernyő	640 * 350 grafikus pont	EGA,VGA
11	Grafikus képernyő	640 * 480 grafikus pont	VGA, MCGA
12	Grafikus képernyő	640 * 480 grafikus pont	VGA
13	Grafikus képernyő	320 * 200 grafikus pont	VGA, MCGA

A parancsban a MÓD helyére a KÓD oszlopban adott számot kell írni. A felbontás első száma az oszlop, a második a sor lehetséges számértéke.

A könyvben a SCREEN 12 parancsot használjuk. Ettől eltérő képernyőméretthez a programokat neked kell átalakítani!

A SZÍN értelmezése egy logikai kifejezés, amellyel be-, illetve kikapcsolhatjuk a színeket.

Feladat

1. A táblázat adatait felhasználva gépeld be a következő programot, a kipontozott részbe írd a szükséges adatokat. A LINE segítségével keretet rajzolunk a képernyőre. A MAXX az oszlop, a MAXY a sor maximális értékét jelenti, ezeket úgy állítsd be, hogy a keret a lehető legnagyobb legyen!

```
SCREEN...
MAXX =... : MAXY =...
LINE (0, 0) – (MAXX, MAXY), 3, B
DO
    x$ = INKEY$
LOOP UNTIL x$ = CHR$(32)
SCREEN 0
```

Az 1. feladatban a SCREEN parancs hatására „leeresztjük a vásznat", így a mögötte levő „színpad", vagy TEXT-képernyő nem látszik. A LINE parancs segítségével a képernyő teljes méretében téglalapot rajzolunk. A SPACE billentyűre várakozás után a „színpadra" ismételten ránézni csak a „vászon felhúzásával" lehet. A képernyőt ismét TEXT-re állítjuk. Egy billentyű megnyomására az utolsó SCREEN parancs használata nélkül is visszaváltunk a képernyő TEXT-re.

Módosítsd a program SCREEN parancsát a monitorodnak megfelelő több beállítás szerint is a MAXX és a MAXY változókkal együtt.

SZÍNEK HASZNÁLATA

Mielőtt pontokat jelenítenénk meg a képernyőn, meg kell ismerkednünk a COLOR speciális jelentésével, mivel ez a grafikában lényegesen eltér a tanultaktól.

PARANCS	Üzemmód	Színek száma
COLOR [előtér][,háttér][,keret]	SCREEN 0	0...15
COLOR [előtér][,paletta]	SCREEN 1	0...1
COLOR [előtér][,háttér]	SCREEN 7–11	0...1
COLOR [előtér]	SCREEN 12–13	0...15

előtér *értékei a színek számában adottak lehetnek.*
paletta *A Quick BASIC két paletta 3-3 színe közötti választást engedélyez.*

A COLOR utasításban lehet palettát választani, míg a palettáról használt színt az adott grafikus utasításban kell megadni. 0 vagy páros szám esetén az első, míg 1 vagy páratlan szám esetén a második palettaszín választható. A táblázat az egyes paletták színhármasait tartalmazza.

színkód	0-ás paletta	1-es paletta
1	zöld	cián
2	vörös	bíbor
3	barna	fehér

háttér *értékei a színek számában adottak lehetnek.*

KÓD	Angolul	Magyarul	KÓD	Angolul	Magyarul
0	black	fekete	8	gray	szürke
1	blue	kék	9	light blue	világoskék
2	green	zöld	10	light green	világoszöld
3	cyan	cián	11	light cyan	világoscián
4	red	piros	12	light red	világospiros
5	magenta	bíbor	13	light magenta	világosbíbor
6	brown	barna	14	yellow	sárga
7	white	fehér	15	hight intensity white	intenzív fehér

Feladatok

2. Módosítsd a grafika bekapcsolására használt programot úgy, hogy LINE parancsban adott 3-as értéket a monitor típusának megfelelő színértékekre változtatod!
3. Módosítsd az előző programot úgy, hogy ciklusban változtatod a szín értékét a lehetséges módon, s minden beállítási lehetőség után egy billentyű megnyomására jelenjen meg az újabb szín!

PONT RAJZOLÁSA

Az előző példában már alkalmaztuk a pont megjelenítésének utasítását. Általánosan megfogalmazva a következő lehetőségeink vannak:

PSET (oszlop, sor), színkód *az adott helyen, adott színű pontot rajzol,*

illetve

PRESET (oszlop, sor), színkód

a színkód alapértelmezése ellentétes.

oszlop, sor *a képernyő beállított méretei szerinti értékek.*

színkód *az előző táblázat színértékei.*

Mintaprogram

```
SCREEN 12
PSET (10,20),1
DO
    X$=INKEY$
LOOP UNTIL X$=CHR$(32)
PRESET (10,20),1
```

A képernyő bal felső sarkától 10 grafikus ponttal jobbra és 20 ponttal lefelé egy kis pontot látsz. Ez a pont egy PIXEL KURZOR-nyi hely. A SPACE-t meg-

nyomva ez a pont eltűnik, mert ugyanerre a helyre a PSET utasítás helyett a PRESET utasítást alkalmaztuk, s ennek színkódértelmezése ellentétes az előzővel.

Ezek után próbáljunk meg egymás után több pontot is kirajzoltatni egy sorban! A pontok között 3 grafikus pontnyi helyet hagyunk.

A megoldás az, hogy egy ciklusban egymás után több pontot rajzolunk ki hármas lépésközzel. Az előző programot töröld, majd gépeld be az alábbit!

Mintaprogram

```
SCREEN 12
FOR I=10 TO 100 STEP 3
    PSET (I,20),1
NEXT I
```

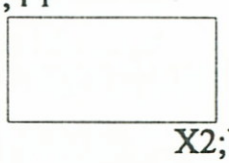
A futtatás után oldd meg az alábbi feladatokat a mintaprogram felhasználásával!

Feladatok

4. Módosítsd a mintaprogramot, hogy a pontok között 4 pixelnyi hely legyen!
5. A program függőlegesen is rajzoljon ki ugyanilyen hosszúságban pontokat, az első eset lépésközeivel!
6. Lassítsd le a pontok kirajzolását!
7. Készíts a képernyő közepére egy 100×100 grafikus pontból álló téglalapot!
8. Készíts programot, amely a grafikus képernyőre rajzol egy pontot, amelynek koordinátáit véletlenszám-generátorral állítod elő! Írasd mellé a pont koordinátáját!
9. Készíts programot, amely 100 darab véletlen számpárt (pontkoordinátát) állít elő! Ezen koordinátákra rajzolj pontokat!
10. Egy pont oszlopértéke (X) változzon 1-től 100-ig, sorértéke (Y) pedig az alábbi szabályok szerint az oszlop értékétől függjön:
    a) Y = X * 2 **b)** Y = X * X **c)** Y = INT(100/X)

SZAKASZ RAJZOLÁSA

A következő utasítás segítségével szakaszokat, illetve téglalapokat rajzolhatsz attól függően, hogy szerepel-e benne a *B* paraméter.



LINE [step](X1,Y1)-[step](X2,Y2),színkód[,B[F]][,vonaltípus]

X1,Y1	*a szakasz kezdőpontjának koordinátái. Ha nem adjuk meg, akkor a kurzor aktuális értékétől rajzol.*
X2,Y2	*a szakasz végpontjának koordinátái.*
step	*a számítás a kurzor legutolsó pozíciójától kezdődik.*
színkód	*a COLOR utasításban választott szín.*
B	*kontúros téglalapot rajzol.*

F *kifesti a téglalapot. Csak BF alakban van értelme.*
vonaltípus *értéke 0000-tól 9999-ig terjedhet.*

Mintaprogram

Az alábbi programot gépeld be, majd vizsgáld meg, hogyan valósítottuk meg a vonalak rajzolását!

```
SCREEN 12
FOR I=1 TO 199  STEP 2
    LINE (1,I)-(319/199*I,I),3,,I*2
    LINE -(319,I),1,,I*10
NEXT I
DO
    F$=INKEY$
LOOP UNTIL F$=CHR$(32)
```

Mi a feladata, és működésében miben különbözik a program 5. és 6. sora ?

.....

.....

Feladatok

11. Tervezz négyzetrácson vagy milliméterpapíron egy házat ablakokkal, kerítéssel! Készíts programot, amely az adott rajzot a grafika parancsaival valósítja meg a képernyőn!
12. Rajzolj a képernyő közepére egy 100×100-as négyzetet, majd rácsozd be 10 egységenként!
13. Tégy pontot a képernyő nulladik oszlopának közepére, majd húzz vonalakat az adott pontból a képernyő jobb szélének minden ötödik pontjához! A kapott csillagszórót készítsd el a képernyő jobb széléről balra irányuló sugarakkal is! Változtasd a vonaltípust is!

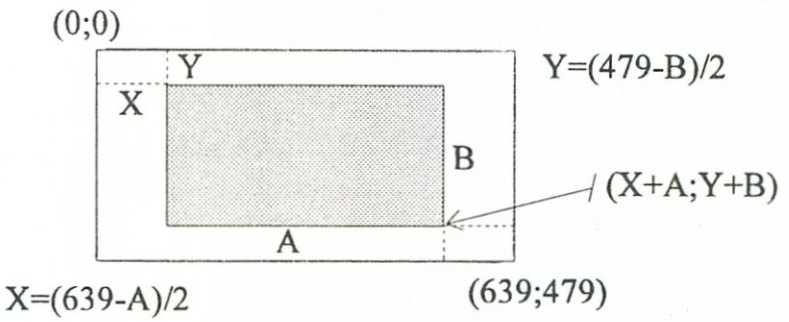
Téglalap rajzolása

Készítsünk programot, amely bekéri egy téglalap oldalainak mérőszámát, majd azt a képernyő közepére rajzolja, s kiírja a területét és a kerületét!

A program a szakasz rajzolásánál egy fokkal összetettebb, de lényegében nagyon hasonló, mert a téglalapot is szakaszok határolják. Elsőként az oldalak mérőszámát kell bekérni. A képernyő korlátozott méretű, ezért csak olyan adatokat fogadhatunk el, amit ábrázolni is tudunk. Rossz adatnak minősülnek a negatív értékek, valamint a tizedes törtek is.

```
REM *** téglalap ***
CLS
PRINT" Kérem a téglalap oldalait cm-ben."
DO
   INPUT"A=";A                    bekérjük az 'A' értékét
LOOP UNTIL A>1 AND A<639 AND A=INT(A)
                                  ha az „A" értéke 0-nál kisebb, vagy 639-nél nagyobb, vagy nem egész szám, akkor új adat kell.
DO
   INPUT"B=";B                    bekérjük a „B" értékét,
LOOP UNTIL B>1 AND B<479 AND B=INT(B)
                                  mint az „A" esetében, de értelemszerűen más határokkal.
SCREEN 12                         bekapcsoljuk grafikát.
```

Szükségünk van azon pontkoordinátákra, amelyektől megrajzolva a téglalapot, az a képernyő közepére kerül. A képernyő szélessége 640 grafikus pontnyi. Mindkét oldalon ugyanannyinak kell elmaradnia.




```
X=(639-A)\2 : Y=(479-B)\2
```

Függőlegesen ugyanígy lehet kiszámítani a pontkoordinátát. A téglalap bal felső sarkának koordinátái tehát a kiszámított **X**, **Y** értékek.

A jobb alsó sarok koordinátái ettől **A**, illetve **B** távolságra vannak.

```
LINE (X, Y)-(X + A, Y + B), 1, BF
                                  Az „F" paraméter segítségével festett téglalapot rajzolunk.
T=A*B:K=(A+B)*2                   Kiszámítjuk a területet, illetve a kerületet,
LOCATE 24,3                       majd a 24. sor 3. oszlopába kiíratjuk.
PRINT"T=";T" cm"+CHR$(253);" K=";K;"cm"
                                  A CHR$(253) kóddal tudjuk a négyzetcentiméter kis kettesét kiíratni.
```

Futtasd a programot!

Feladatok

14. Írasd a képernyőre a mintaprogramban adott téglalap oldalainak mérőszámát úgy, hogy a kiírás a téglalap oldalainak közepére essen!
15. Módosítsd a programot úgy, hogy az festett téglalapot rajzoljon!
16. Rajzoltasd meg a téglalap átlóit a festő színtől eltérő színnel és vonaltípussal!
17. Készíts programot, amely egy megrajzolt téglalapot 10 egységenként beráccsoz!

18. Készíts programot, amely beolvas egy oszlop- és egy osztásértéket, majd ezen oszlopban a képernyőn rajzol egy függőleges szakaszt, s az adott osztással számegyenest készít belőle! A számegyenes 0 pontja a képernyő közepére essen!
19. Módosítsd az előző programot úgy, hogy a leírt eljárás alapján megrajzolja a vízszintes számegyenest is! Készítsd el a számegyenes feliratozását is!
20. Készíts programot, amely az előzőleg elkészített koordináta-rendszerben ábrázolja az **X**, **Y** koordinátákkal adott pontot egy négyzet segítségével!
21. Készíts programot, amely véletlenszerűen előállítja három pont koordinátáit, s azokat egy koordináta-rendszerben háromszögként összekötve rajzolja ki!

FESTÉS

Az utasítás segítségével zárt alakzatokat tudsz a lehetséges színbeállításokkal, illetve általad tervezett parkettával kifesteni.

PAINT (X,Y)[,festék][,határvonal][,háttér]

X,Y	*A kifestendő alakzat egy belső pontja.*
festék	*A lehetséges színértékeket veheti fel, de lehet megtervezett parkettaminta is. Erről a 16-os számrendszernél lesz szó.*
határvonal	*A kifestendő alakzat határszínét adhatjuk meg.*
háttér	*Egybyte-os string, amelynek egy alakzat parkettázásánál van szerepe.*

Mintaprogram

Készítsünk három téglalapot, amelyeket befestünk a kerettől különböző színnel!

```
REM *** fest 1 ***
SCREEN 12
FOR i = 0 TO 200 STEP 100
    j = j + 1
    LINE (i, 0) - (i + 80, 150), j, B
    PAINT (i + 1, 1), j + 1, j
NEXT i
```

Futtasd a programot!

Hova helyeztük el a PAINT utasításhoz szükséges belső pont koordinátákat a kifestendő alakzatokhoz?

.....

.....

Feladatok

22. Módosítsd az előző mintaprogramot úgy, hogy minden téglalapot eggyel nagyobb színkódúra fess, mint a keret színe!
23. Írasd minden téglalap alá a keret és a festék színeinek értékeit!
24. Készíts programot, amely hegyesszögű, derékszögű és tompaszögű háromszögeket rajzol a képernyőre! Mindegyiket más-más színnel kifesti, majd mindegyik alá kiírja a területének kiszámítási képletét!
25. Írj programot, amely sokszögeket rajzol a képernyőre más-más színnel kifestve! Mindegyikhez írasd ki a területét kiszámító képletet!

HEXADECIMÁLIS (16-OS) SZÁMRENDSZER

A festés az említett eljárással nem teszi lehetővé a zárt alakzatok speciális mintákkal való feltöltését. Ahhoz, hogy ezt megtehessük, meg kell ismernünk a hexadecimális számrendszert.

Az előző kötetben már olvashattál a számrendszerekről.

A kettes számrendszerben csak két számjegyet írhattunk le: a 0-át és az 1-est. Így az 111001 szám a kettes (bináris) számrendszerbeli szám a tízes (decimális) számrendszerben leírva következőt jelenti:

$111001_{②} = 1*32 + 1*16 + 1*8 + 0*4 + 0*2 + 1*1 = 32+16+8+1 = 57_{⑩}$

A tanultak szerint tehát a 10-es számrendszerben a 0, 1, 2, ... 9-es számjegyeket használhatjuk, a 9-es számrendszerben a 0, 1, … 8-as, ... a 2-es számrendszerben pedig a 0, illetve az 1-es számjegyeket.

A 16-os számrendszerbeli számokat a: 0, 1, 2, … 10, 11, 12, 13, 14, 15 számjegyekkel írhatnánk le, ha az előző gondolatmenetet követnénk. Az viszont lehetetlen, hogy egy helyiértékre két számjegy kerüljön.

A 10, 11, … 15 számjegyeket így nem használhatjuk. Ezek helyére az ABC nagy betűit írjuk sorban egymás után a következők szerint:

10 = A; 11 = B; 12 = C; 13 = D; 14 = E; 15 = F

Mennyit ér a 10-es számrendszerben a B3F hexadecimális számrendszerben felírt szám?

Helyiértékek:	16^2		16^1		16^0	
	256		16		1	
Együtthatók:	B		3		F	ahol B = 11, F = 15
Szorzatok:	11*256	+	3*16	+	15*1	
Összeg:	2816	+	48	+	15	= 2879

Vagyis a B3F alakban felírt 16-os számrendszerbeli szám 2879-et ér a 10-es számrendszerben: $B3F_h = 2879_d$

Írjuk át 16-os számrendszerbeli számmá a 3759-et!

Helyiértékek: 16^2 16^1 16^0

256 16 1

Az együtthatókat visszaosztással kapjuk meg.

3759 : 256 = 14 és marad 175. *A 14 hexadecimálisan E-t jelent.*

175 : 16 = 10 és marad 15. *A 10 hexadecimálisan A-t jelent.*

15 : 1 = 15 és maradt 0. *A 15 hexadecimálisan F-et jelent.*

Így a $3759_d = EAF_h$ lesz.

Ellenőrizzük!

E * 256 + A * 16 + F * 1 = 14 * 256 + 10 * 16 + 15 * 1 = 3584 + 160 + 15 = 3759.

A hexadecimális számokat felhasználva készítünk egyedi parkettamintákat a festési eljáráshoz.

PARKETTÁZÁS

A parkettamintát 1 byte-on tervezzük meg. Ahova 1-est teszünk, ott lesz pont, ahova 0-át, ott üres helyet hagyunk. A tervezésre több példát adunk.

A) Az alábbi táblázat egy megtervezett mintát mutat be.

	Bitek sorszáma								Megfelelő értékek a 10-es számrendszerben								
	8.	7.	6.	5.	4.	3.	2.	1.	128	64	32	16	8	4	2	1	
0. byte	1	1	1	1	1	1	1	1	128	64	32	16	8	4	2	1	=255
1. byte	1	1	0	0	1	1	0	0	128	64	0	0	8	4	0	0	=204
2. byte	1	1	0	0	1	1	0	0	128	64	0	0	8	4	0	0	=204
3. byte	1	1	0	0	1	1	0	0	128	64	0	0	8	4	0	0	=204

A kapott tízes számrendszerbeli számokat váltsuk át 16-os számrendszerbe! A tanult osztási eljárást használjuk fel!

255 : 16 = 15 és marad 15. *A 15 hexadecimálisan F-et jelent.*

15 : 1 = 15 és marad 0. *Azaz $255_d = FF_h$*

A másik számnál ugyanígy járunk el!

204 : 16 = 12 és marad 12. *A 12 hexadecimálisan C-t jelent.*

12 : 1 = 12 és marad 0. *Azaz $204_d = CC_h$*

Egy hexadecimálisan felírt számot ugyanúgy meg tudunk jeleníteni a CHR$(X) függvénnyel, mint a decimálisan felírt esetben, csak most jelölni kell, hogy ebben a formában használjuk. Tekintettel arra, hogy itt több byte együttes megjelenítésére van szükségünk, a stringek összefűzésére vonatkozó összeadás műveletet használjuk fel. Az itt tanultakat felhasználva módosítjuk a festésnél használt programot.

```
REM *** fest 2 ***
SCREEN 12
COLOR 0,1
MINTA$=CHR$(&HFF)+CHR$(&HCC)+CHR$(&HCC)+CHR$(&HCC)
```

```
FOR I=0 TO 200 STEP 100
    J=J+1
    LINE (I,0) - (I + 80,150),J,B
    PAINT (I + 2,2),MINTA$,J
NEXT I
```

A program most már az általunk tervezett mintával tölti ki a téglalapokat.

B) Az előzőekben ismertetett eljárás túl hosszú. Az 1 byte-on megtervezett minta kettes számrendszerbeli alakját 10-es számrendszerbe, majd az osztási eljárással hexadecimális alakba írtuk.

A bitmintát „vágjuk" két részre az ábra szerint, majd mindkét részt külön-külön írjuk át bináris számrendszerbe úgy, mintha nem egy, hanem két fél byte lenne. Mivel mindkét fél ugyanazokat a kódértékeket tartalmazza, ezért elegendő az egyik rész kódolása.

	8.	7.	6.	5.	4.	3.	2.	1.
0. byte	1	1	1	1	1	1	1	1
1. byte	1	1	0	0	1	1	0	0
2. byte	1	1	0	0	1	1	0	0
3. byte	1	1	0	0	1	1	0	0

	4.	3.	2.	1.	4.	3.	2.	1.
0. byte	1	1	1	1	1	1	1	1
1. byte	1	1	0	0	1	1	0	0
2. byte	1	1	0	0	1	1	0	0
3. byte	1	1	0	0	1	1	0	0

Helyiértékek:	2^3	2^2	2^1	2^0	
	8	**4**	**2**	**1**	
0. fél byte	1	1	1	1	$8+4+2+1=15_{10} = F_{16}$
1. fél byte	1	1	0	0	$8+4+0+0=12_{10} = C_{16}$

A másik két fél byte ugyanezeket az értékeket tartalmazza.

A teljes byte értékei hexadecimális számrendszerben leírva tehát teljesen megegyeznek az első módszer szerint leírtakkal, csak most az eljárás egyszerűbb.
Ezek után már csak ki kell íratni az előállított mintát.

Feladat

26. Tervezz meg két mintát! Rajzolj a képernyőre két téglalapot, s parkettázd be mindegyiket az egyik, illetve a másik mintával!

	8.	7.	6.	5.	4.	3.	2.	1.
0. byte	□	□	□	□	□	□	□	□
1. byte	□	□	□	□	□	□	□	□
2. byte	□	□	□	□	□	□	□	□
3. byte	□	□	□	□	□	□	□	□
4. byte	□	□	□	□	□	□	□	□
5. byte	□	□	□	□	□	□	□	□
6. byte	□	□	□	□	□	□	□	□
7. byte	□	□	□	□	□	□	□	□
8. byte	□	□	□	□	□	□	□	□

	8.	7.	6.	5.	4.	3.	2.	1.
0. byte	□	□	□	□	□	□	□	□
1. byte	□	□	□	□	□	□	□	□
2. byte	□	□	□	□	□	□	□	□
3. byte	□	□	□	□	□	□	□	□
4. byte	□	□	□	□	□	□	□	□
5. byte	□	□	□	□	□	□	□	□
6. byte	□	□	□	□	□	□	□	□
7. byte	□	□	□	□	□	□	□	□
8. byte	□	□	□	□	□	□	□	□

Írd le a mintákat meghatározó karakterláncot is!

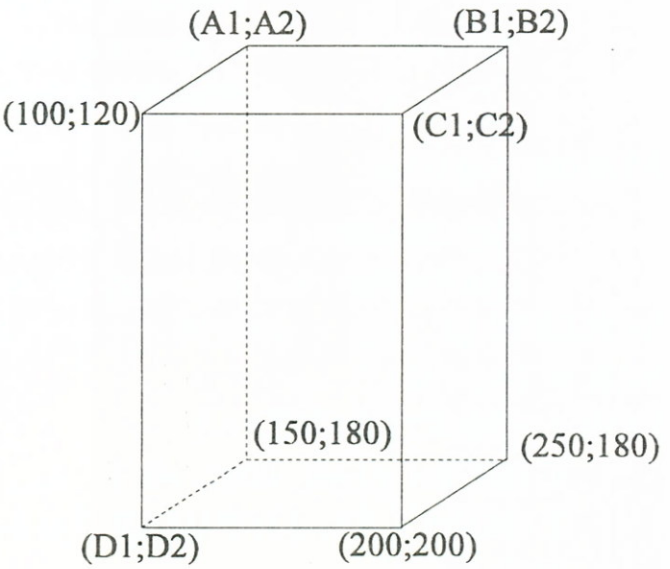
MINTA1$=.....

MINTA2$=.....

TÉGLATEST RAJZOLÁSA

Az ábrán egy téglatestet rajzoltunk meg, amelynek egyes sarkaira ráírtuk a csúcspontjainak koordinátáit is. A hiányzó adatokat a meglevőkből ki tudod számítani. (A program begépelése előtt ezt tedd is meg! Vigyázz! A képernyő koordináta-rendszerében adtuk meg a pontok koordinátáit.)

SCREEN 12
LINE (100,120) - (200,200),1,B
LINE -(250,180),1
LINE -(B1,B2),1
LINE -(A1,A2),1
LINE -(100,120),1



Először a téglatest előlapját rajzoltuk meg. A grafikus kurzor az első utasítás hatására a (200,200) pontba került. Ebből a pontból indulva rajzoltuk meg a többi töröttvonalat. Mindegyiket az előző folytatásaként, így az utasításban nem kell megadni a kezdőpont koordinátáit.

Most nézd meg, mit eredményez a program futtatása! Hiányzik még egy látható él, ezt azonban nem lehet az előző folytatásaként rajzolni, mert a grafikus kurzor jelenleg a (100,120) pontban van.

A hiányzó él koordinátáinak ismeretében egy külön vonalat rajzolunk!

LINE (C1,C2) - (B1,B2),1

Feladatok

27. A rajzórákon a nem látható éleket szaggatott vonalakkal rajzoljuk meg. A LINE parancs végén megadható a vonaltípus, így fejezd be a rajzodat!

28. Készíts három mintát, majd az előző programban elkészült téglatest mindegyik látható lapját parkettázd be! A képernyőre írja ki a téglatest felszínét és térfogatát!

29. Készíts programot, amely bekéri egy kocka élének hosszát, majd elkészíti a kocka térbeli képét! A képernyőre írja ki a kocka felszínét és térfogatát! A felhasznált képleteket is írasd ki!

KÖR RAJZOLÁSA

A parancs segítségével kört, ellipszist rajzolhatunk a grafikus képernyőre általunk meghatározott sugárral, és általunk megadott középpont körül.

CIRCLE (X,Y),R,Szín[,,,arány]

X,Y	*a kör középpontjának koordinátái;*
R	*a sugár értéke;*
Szín	*a SCREEN utasítással beállított lehetőségek szerinti szín kódja.*
Arány	*a kör torzítására szolgáló érték, ellipszisek rajzolásához. Ha nem adjuk meg, kört rajzol.*

Mintaprogram

Rajzoljunk a képernyő középpontjába köröket, amelyeknek azonos a középpontjuk és a sugaruk, de más-más értékű a torzításuk!

```
SCREEN 12
SZK = 1
FOR t = 90 TO 0 STEP -5
    CIRCLE (160, 100), 90, SZK,,, (t / 90)
    LOCATE 15, 13: PRINT "arány="; t; "/ 90"
    DO
      x$ = INKEY$
    LOOP UNTIL x$ = CHR$(32)
NEXT t
CLS
FOR t = 90 TO 5 STEP -5
   CIRCLE (160, 100), 90, SZK,,, (90 / t)
   LOCATE 15, 13: PRINT "arány="; "90/"; t
   DO
      x$ = INKEY$
   LOOP UNTIL x$ = CHR$(32)
NEXT t
```

A program begépelése után a SPACE billentyűvel váltogathatjuk az egyes arányokat! Ezek értéke is megjelenik a képernyőn!

Feladatok

30. Rajzoltass 40 egység sugarú kört, majd rajzoltasd meg két egymásra merőleges átmérőjét! Írasd ki a kör középpontjának koordinátáit!

31. Készíts programot amely a képernyő középpontjából kiindulva köröket rajzol, miközben a sugár értéke 1-től 100-ig 10-esével változik! Módosítsd a programot úgy, hogy az új kör megrajzolása előtt az előző kört letörli!
32. Készíts programot, amely egy gömböt rajzol a képernyőre!

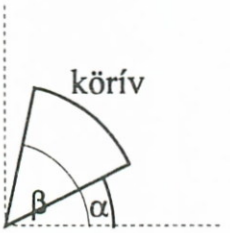
KÖRÍVEK RAJZOLÁSA

A CIRCLE parancsot bővítjük ki az alábbiak szerint:

CIRCLE (X,Y),R,Szín,α,β[,arány]

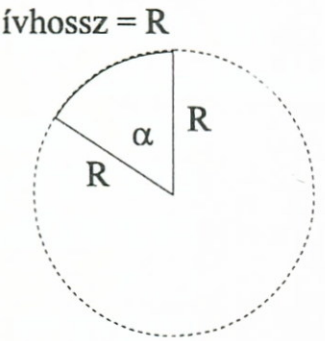
α *értéke az ív kezdő szöge radiánban.*

β *értéke az ív végszöge radiánban.*



A radián (vagy ívmérték) a szögmérés egyik módja. Egy radián az értéke annak a szögnek, amelyhez tartozó körív hossza egyenlő a sugár hosszával.

E szög: 57° 17' 44,6" = 1 rad.



Egy fokokban megadott szög ívmértékét az $\hat{\alpha} = \frac{\pi \cdot \alpha^\circ}{180^\circ}$ képlettel számíthatjuk ki, ahol $\pi = 3{,}1415926$.

Néhány fontosabb szög ívmértéke:

$180^\circ = \pi$ rad; $\quad 90^\circ = \frac{\pi}{2}$ rad; $\quad 60^\circ = \frac{\pi}{3}$ rad.

Mintaprogramok

1. *Készítsünk programot, amely egy 60 egység sugarú kör 30° és 120° közötti ívét rajzolja meg!*

```
SCREEN 12
PI=3.1415926
CIRCLE 160,100,60,2,PI *30/180,PI*120/180
```

Módosítsd a program harmadik sorát úgy, hogy más színnel rajzolja meg a 120° és a 30° közötti ívet is!

2. *Készítsünk programot, amely szaggatott vonalakkal rajzolja meg egy 60 egység sugarú kör pontjait!*

A megoldást megadó programrészletet itt találod:

```
FOR I = 0 TO 355 STEP 5
    CIRCLE (160, 100), 60, 2, PI*I/180, PI*(I+2)/180
NEXT I
```

Módosítsd a programot úgy, hogy a ciklust 360°-tól 5°-ig indítod -5°-os lépésközzel, a kezdő- és végszög értékét ellentettjére veszed! Mi a szerepe a negatív előjelű kezdő- és végszögnek?

FÜGGVÉNY DEFINIÁLÁSA

Az ívek rajzolásánál nehézkes a fokok átszámítása ívmértékbe. Az adott átszámítási eljárást több ív megrajzolása esetén a képlet külön definiálásával tehetjük egyszerűbbé.

DEF FNnév(változó)=képlet

Fnnév(változó)

Az FN után közvetlenül szereplő néven (max.: 40 karakter) hivatkozhatunk később erre a függvényre. A név nem tartalmazhat speciális karaktereket. A zárójelben levő változónév (nevek) helyére írjuk a kiszámítandó értéket (értékeket).

Képlet *A képletbe az előzőekben megadott változó matematikai összefüggéseit kell beírni.*

Az alábbi programban három függvényt definiálunk, s ezeket használjuk fel a számításokhoz:

```
DEF FnNegyzet (a) = a * a
DEF FnTegla (a, b) = (a + b) * 2
DEF FnRad (alfa) = 3.141592 * alfa / 180
CLS
PRINT "Négyzet területe="; FnNegyzet(5)
PRINT "Téglalap területe="; FnTegla(2, 3)
PRINT "40 fok ívmértéke="; FnRad(40)
```

Kör sugarainak megrajzolása

Készítsünk programot, amely megrajzolja egy kör sugarait 5 fokos beosztással, majd egy billentyű megnyomására letörli a köríveket 10 fokonként!

```
SCREEN 12
DEF FNR (x) = (3.141592 / 180) * x
```

A radián számításához szükséges függvényt definiálunk.

```
FOR Q = 330 TO 5 STEP -5
    CIRCLE (166, 100), 90, 3, -FNR(Q), -FNR(Q + 30)
```

Megrajzoltuk a körcikkeket.

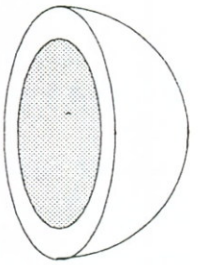
```
NEXT Q
DO
    x$ = INKEY$
LOOP UNTIL x$ = CHR$(32)
FOR Q = 355 TO 1 STEP -10
    CIRCLE (166, 100), 90, 0, -FNR(Q), -FNR(Q + 5)
```

Letöröltük a körcikkek íveit.

```
NEXT Q
```

Módosítsd a programot úgy, hogy a sugarakat 10 fokonként rajzolja, a törlést 5 fokonként végezze el!

Barack rajzolása



Készítsünk programot, amely egy félbevágott barackot rajzol a képernyőre mag nélkül!

```
REM CIRCLE2
PI = 3.14159265
SCREEN 12
CIRCLE (160, 100), 90, 3,,, (90 / 20)
PAINT (160, 100), 2, 3
CIRCLE (160, 100), 80, 0,,, (90 / 15)
PAINT (160, 100), 3, 0
SZK = 1
eleje:
    FOR T = 90 TO 25 STEP -5
        CIRCLE (160, 100), 90, SZK, PI / 2, 3 / 2 * PI, (90 / T)
    NEXT T
    CIRCLE (160, 100), 80, 0,,, (90 / 15)
    PAINT (160, 100), 0, 0
    IF SZK = 1 THEN SZK = 3 ELSE SZK = 1
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = CHR$(32) OR a$ = "q"
    PAINT (160, 100), 3, 2
    IF a$ = "q" THEN GOTO vege ELSE GOTO eleje
vege:
    END
```

Futtatás után módosítsd a programot úgy, hogy a félgömbbel szemben, tőle bizonyos távolságra, rajzolja meg a gömb másik felét is!

Feladatok

33. Készíts egy tortát, majd belőle kiemelve egy tortaszeletet!
34. Mit csinál a program? (Futtatás nélkül állapítsd meg!)

```
SCREEN 12
FOR W = 5 TO 100 STEP 5
    CIRCLE (160, W), W, 2
    CIRCLE (W, 100), W, 2
    CIRCLE (320 - W, 100), W, 2
    CIRCLE (160, 200 - W), W, 2
NEXT W
```

.....

.....

.....

.....

.....

.....

35. Készíts programot, amely egy kör területét 3:5:10 arányban osztja fel! A különböző területeket más-más mintával fesd be!
36. Készíts egy legyezőt a képernyőre, amely egy billentyű megnyomására kinyílik, majd becsukódik!

VONALAK RAJZOLÁSA, FORGATÁSA

Az utasítás egy „grafikus leíró nyelv" egybetűs parancsai alapján dolgozik. A kirajzolandó alakzatot mozgásleírással, stringben adjuk meg. A végrehajtás során a program balról jobbra értelmezi, s hajtja végre a parancsokat.

DRAW "mozgás_string"

A mozgásstring a következőket tartalmazhatja:

Előtagok

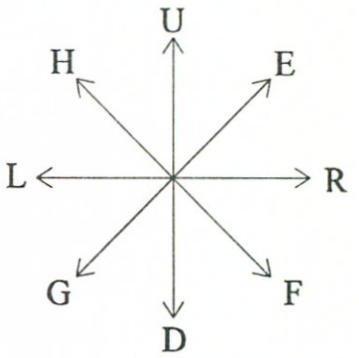
B *mozgás rajzolás nélkül,*

N *mozgás befejezése után a pixelkurzort a kiindulási pozícióba állítja vissza.*

Kurzormozgatás

A betűk után szereplő *t* betű az adott irányú elmozdulás értékét jelenti.

U t	felfelé	**D** t	lefelé
L t	balra	**R** t	jobbra
H t	balra fel	**G** t	balra le
E t	jobbra fel	**F** t	jobbra le



Az eltérő mértékű felbontások miatt eltérő mértékű vízszintes, illetve függőleges elmozdulás látszik azonosnak.

Speciális parancsok

A n — *Forgatás n° -os szöggel. A lehetséges értékek:*
0 ⇒ 0°-os forgatás; 1 ⇒ 90°-os forgatás;
2 ⇒ 180°-os forgatás; 3 ⇒ 270°-os forgatás.

TA n — *Forgatás n°-os szöggel. A lehetséges értékek:* $-360° < n < 360°$.

C s — *Szín beállítása. Az „s" lehetséges értéke az adott képernyő felbontástól függ.*

S x — *Skála beállítása. Az „x" lehetséges értékei:* $0 < x < 256$. *Hatására a mozgatóparancsoknál megadott elmozdulás értéke X-szeresére változik. (Nagyítás!)*

P b,k — *Egy zárt alakzat belsejének (b) és határának (k) színét állítja be.*

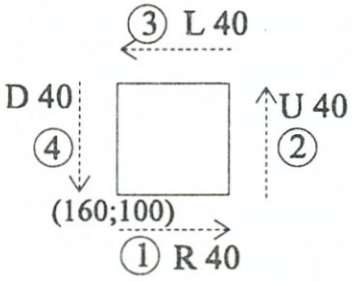
Mintaprogram

(A keretezett sorszámok a DRAW utasításban levő parancsok végrehajtási sorrendjét jelölik.)

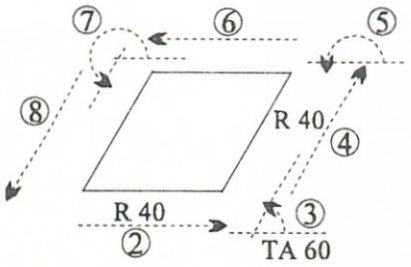
SCREEN 12

PSET (160, 100), 0 *a pixelkurzort a (160,100) pozícióba állítjuk.*

DRAW "C1 R40 U40 L40 D40" *a parancs hatására egy négyzetet rajzolunk.*



DRAW "B L100 C2 R40 TA60 R40 TA 180 R40 TA240 R40"

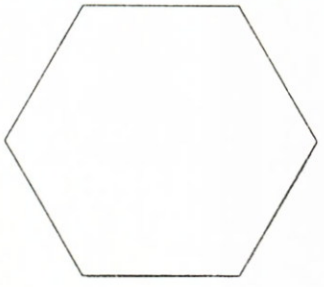


A rombusz rajzolásához az előző négyzet bal alsó, (160,100) koordinátájú csúcsától balra mozdulunk 100 egységnyit rajzolás nélkül. Ezért szerepel a utasítássor elején a B L100 parancs. A következő rajzolóparancs ettől a ponttól kezdődik kettes színkóddal. A forgatási parancsnál mindig a vízszintes értéktől kell számítanunk annak értékét.

PSET (200, 150), 3

A síkidom megrajzolása a (200,150) koordinátájú pontban kezdődik, hármas színkóddal.

DRAW "C3 R30 TA60 R30 TA120 R30 TA180 R30 TA240 R30 TA300 R30 TA360 R30"



A rajzon jelöld be a lépések sorrendjét, illetve az elmozdulások, forgatások irányát és értékét!

Feladatok

37. Rajzolj egy hajót a képernyőre a DRAW felhasználásával!

38. Készíts programot, amely egy 20 egység élű kocka térbeli képét rajzolja meg! Fesd be a három látható lapot más-más színnel, vagy mintával!

39. Módosítsd az előző programot úgy, hogy a megrajzolt kocka mellé rajzoltasd ki a nagyított képét is a DRAW utasítás „S x" parancsának felhasználásával!

40. A **38.** feladatban megrajzolt kockát helyezd a képernyő közepére, majd készíts olyan programot, amely a kockát folyamatosan nagyítja a képernyő lehetséges legnagyobb felbontásáig!

Változók használata a DRAW utasításban

Készítsünk programot, amely egy négyzetet csúcspontja körül 60 fokonként elforgat!

A megoldáshoz az ismétlési szerkezeteket használjuk. A ciklusváltozó értékét kell a forgatáshoz használnunk. Ez egy számváltozó, amelyet a DRAW parancshoz csak szövegváltozóként illeszthetünk. Számváltozót szövegváltozóvá az STR$(x) stringfüggvény alakít. A stringeket össze kell kapcsolnunk, amit az összeadás műveletével tehetünk meg.

```
SCREEN 12
PSET (100, 100), 2
MO$ = "R60 U60 L60 D60"          Előre megadjuk a négyzetet megrajzoló stringet.
FOR ALFA = 60 TO 360 STEP 60
    DRAW "TA" + STR$(ALFA)       A forgatás parancsához (TA) az összeadás művelet-
                                 tével kapcsoljuk a forgatási értéket.
    DRAW MO$                     Az elfordítás után rajzoltatjuk meg az MO$-ban
                                 rögzített alakzatot.
NEXT ALFA
```

A futtatás után módosítsd a programot úgy, hogy 10°-onként rajzolja meg a négyzeteket! Módosítsd a programot úgy, hogy mindig csak egy négyzet legyen látható! (Az előzőleg rajzolt alakzatot töröld!) Ezután „végtelenítsd" a program futását egy billentyű megnyomásáig!

Az alábbi példaprogram futtatása után írd a kipontozott részre az adott sor parancsainak értelmezését! A füzetedbe rajzold le a mozgatásra kerülő alakzatot!

```
REM DRAW2
SCREEN 12
SZK = 3
DRAW "BM160,96"                  .....
FOR SKALA = 1 TO 15 STEP 2
    DRAW "S" + STR$(SKALA)       .....
    MO$ = "U20L5U5R10D5L5"
    FOR ALFA = 0 TO 360 STEP 20
        DRAW "TA" + STR$(ALFA)   .....
        DRAW MO$
        DRAW "BM160,96"          .....
        DO
            X$ = INKEY$
        LOOP UNTIL X$ = " "
    NEXT ALFA
    SZK = INT(RND(1) * 3 + 1)
    DRAW "C" + STR$(SZK)         .....
NEXT SKALA
```

Koch-minta

Készítsünk programot, amely a képernyőre rajzol egy szakaszt! Ezután három egyenlő részre osztjuk, majd a középsőt kitöröljük, s rárajzolunk két ugyanolyan hosszú szakaszt, mint amit kitöröltünk. Minden megrajzolt szakaszt további három egyenlő részre osztunk, a középsőt kitöröljük, majd megrajzoljuk az újabb két szakaszt. Az eljárást addig folytatjuk, amíg a szakaszok rajzolásának van értelme (a szakasz hossza legalább egy).

A program elkészítése előtt célszerű az elképzeléseinket papírra rajzolni.

1. lépés: A szakasz hosszát célszerű olyan értékben meghatározni, ami hárommal osztható, mert csak így lesz a szakaszok harmada egész szám.
A 3 hatványai: 3; 9; 27; 81; 243; 729...
A képernyőre a 243 egységnyi szakasz fér fel, ezért ezt válasszuk!

2. lépés: Az első harmadolást, majd kitörlést követően csak úgy tudunk 81 egységnyi szakaszokat rajzolni a kitörölt rész fölé, ha azok szabályos háromszöget alkotnak a kitörölt résszel. A szabályos háromszög szögei 60°-osak. A szakaszok forgatása miatt nekünk ez egymás után a következő forgatásokat jelenti:

0° →; 60° ↗; –60° ↘; 0° →

3. lépés: A négy megrajzolt szakaszt is harmadolni kell (27 egység). Ez a következő irányokat jelenti egymás után:

0° →	60° ↗	–60° ↘	0° →
60° ↗	120° ↙	0° →	60° ↗
–60° ↘	0° →	120° ↙	–60° ↘
0° →	60° ↗	–60° ↘	0° →

A *4. lépésben* a szakasz hossza 9; az *5. lépésben* 3 egység; a *6. lépésben* 1 egység lesz a szakasz hossza, miközben ezek 3. lépésben számított forgatási szögek ismétlődnek. A 6. lépés után az eljárás nem folytatható, mert a szakasz hossza kisebb lenne, mint az egység. Az eljárást megfigyelve először 1, a második lépésben 4, a harmadikban 16 szakaszt kell megrajzolni és elforgatni. Így a 6. lépésben már összesen 1024 rajzolás és forgatás történik.

A forgatás szögértékeit DATA-sorokban tároljuk, majd úgy töltjük be egy erre a célra létrehozott tömbbe. Tekintettel arra, hogy összesen 16 forgatási szög értékét kell betölteni, ez 1024 : 16 = 64 ismétlést (0-tól 63-ig) jelent. Az adatmutatót minden egyes esetben vissza kell állítani. A tömbbe tehát 16-osával töltjük be az értékeket. Ezt két ciklussal oldjuk meg. A programban folyamatosan (tömbre hivatkozással) oldjuk meg az elforgatásokat, a szakaszok hosszát képlettel számítjuk ki.

A programban megoldottuk a visszafelé történő rajzolást is, így egy zárt alakzat kerül a képernyőre, amit neked kell kifestened!

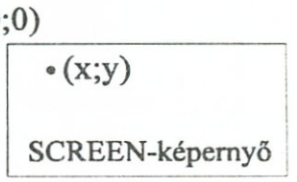
```
SCREEN 12
DIM F(1024)
30: DATA 0, 60, -60, 0, 60, 120, 0, 60 ,-60, 0, -120, -60, 0, 60, -60, 0
FOR N = 0 TO 63
    FOR M = 1 TO 16: READ K
        F(M + N * 16) = K + F(N + 1):
    NEXT M
    RESTORE 30:
NEXT N
FOR I = 0 TO 5
    KX = 243: CLS
    PSET (38, 95), 3
    KX = KX / (3 ^ I): K2 = 0
    FOR K3 = 1 TO 4 ^ I
        K2 = K2 + 1
        DRAW "TA" + STR$(F(K2))
        DRAW "R" + STR$(KX)
    NEXT K3
    K2 = K2 + 1
    PSET (243 + 38, 105), 3
    LINE (281, 95)-(281, 105)
    FOR K3 = 1 TO 4 ^ I
        K2 = K2 - 1
        IF F(K2) = 0 THEN DRAW "TA0": GOTO 230
        F = -F(K2)
        DRAW "TA" + STR$(F)
        230: DRAW "L" + STR$(KX)
    NEXT K3
    LINE (38, 95)-(38, 105)
    LOCATE 24, 16: PRINT "< SPACE >"
    IF I = 5 THEN 300
    DO
        A$ = INKEY$
    LOOP UNTIL A$ = CHR$(32)
300:NEXT I
DO
    A$ = INKEY$
LOOP UNTIL A$ = CHR$(32)
```

KX a megrajzolásra szánt szakasz hosszának értékét jelenti.

Futtatás után úgy módosítsd a programot, hogy minden lépést más-más színnel rajzoljon meg!

ABLAK DEFINIÁLÁSA

A SCREEN parancs hatására a képernyőn használható koordináták értékei eleve meghatározottak. Bármely X, Y koordinátákkal adott pont a bal felső sarokhoz képest X egységgel jobbra, majd Y egységgel lefelé kerül a képernyőre. Az adott X és Y értékek a programozás során csak nemnegatív értéket vehettek fel. Negatív érték esetén a pont nem került a képernyőre. Az említett X, Y értékeket a pont fizikai koordinátáinak is nevezzük. Értékük segítségével pontosan meghatározható, hogy a bal felső sarokhoz képest hol helyezkedik el a képernyőn az adott pont.



A matematikában megismert koordináta-rendszerben negatív értékeket is ábrázolhatunk. A pontok az origóhoz viszonyítva kerülnek megrajzolásra. Ezeket a koordinátákat az origóhoz viszonyítás miatt logikai koordinátáknak nevezzük.

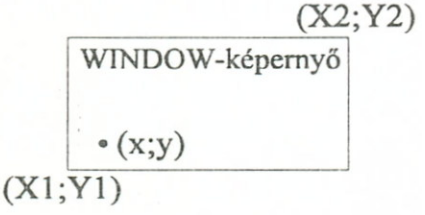
Ilyen logikai koordinátájú pontok megrajzolására a grafika keretében is lehetőségünk van a WINDOW utasítás segítségével. Az így keletkező eltérő felosztású képernyőt felhasználói képernyőnek nevezzük, felosztását tetszőlegesen határozhatjuk meg.

WINDOW[[SCREEN] (X1,Y1) - (X2,Y2)]

X1,Y1 *a képernyő bal alsó koordinátahatára*

X2,Y2 *a képernyő jobb felső koordinátahatára*

Koordináták nélkül megadva, feloldja az addig érvényes felhasználói ablakot, s visszatér a SCREEN-ben levő fizikai koordinátákhoz.



Mintaprogram

Készítsünk programot, amely ugyanazt a háromszöget háromszor rajzolja meg. Először hagyományos grafikus képernyőn, majd a WINDOW parancscsal. Legutoljára a WINDOW SCREEN segítségével.

```
SCREEN 12
LINE (10, 10)-(50, 10): LINE -(40, 30): LINE -(10, 10)
LOCATE 1, 10: PRINT"SCREEN-képernyő"
DO
    x$ = INKEY$
LOOP UNTIL x$ = CHR$(32)
WINDOW (0, 0)-(80, 80)
LINE (10, 10)-(50, 10): LINE -(40, 30): LINE -(10, 10)
LOCATE 15, 5: PRINT "WINDOW-képernyő "
DO
    x$ = INKEY$
LOOP UNTIL x$ = CHR$(32)
```

```
WINDOW SCREEN (0, 0)-(80, 80)
LINE (10, 10)-(50, 10): LINE -(40, 30): LINE -(10, 10)
LOCATE 11, 25: PRINT "WINDOW SCREEN "
```

A program működése közben figyeled meg, hogy mi a hatása a WINDOW utasítás SCREEN paraméterének!

Koordináta-rendszer rajzolása

Készítsünk programot, amely egy koordináta-rendszert rajzol a képernyőre! Ebben háromszögeket rajzolunk úgy, hogy a csúcspontok koordinátáit véletlen számokkal állítjuk elő!

```
REM window2
SCREEN 12
WINDOW (-100, -100)-(100, 100)
```
Új felhasználói ablakot hozunk létre, amelynek bal alsó sarka (-100,-100), jobb felső sarka (100,100) koordinátájú.

```
LINE (-100, 0)-(100, 0), 1
LINE (0, 100)-(0, -100), 1
FOR I = 90 TO -90 STEP -10
```
Megrajzoljuk a tengelyeket, majd mindegyik tengely osztópontjait.

```
    IF I MOD 50 = 0 THEN D = 6 ELSE D = 3
    LINE (0 - D, I)-(0, I), 1
NEXT I
FOR J = -90 TO 90 STEP 10
    IF J MOD 50 = 0 THEN D = 12 ELSE D = 6
    LINE (J, 0 - D)-(J, 0), 1
NEXT J
```
Az elkészült tengelyekhez előállítjuk a háromszög három csúcspontjának koordinátáit.

```
30: X(1) = INT(30 * RND) - 80: Y(1) = INT(80 * RND) - 90
Y(2) = Y(1): X(2) = INT(70 * RND) + 20
X(3) = INT(180 * RND) - 90: Y(3) = INT(60 * RND) + 30
LINE (X(1), Y(1))-(X(2), Y(2)), 3
```
Megrajzoljuk a háromszöget.

```
LINE -(X(3), Y(3)), 3: LINE -(X(1), Y(1)), 3
LOCATE 25, 25: PRINT "Tovább? [I/N]";
DO
    a$ = INKEY$
LOOP UNTIL a$ <> ""
IF a$ = "i" THEN
```
Az „i" betű megnyomásával

```
    LINE (X(1), Y(1))-(X(2), Y(2)), 0
    LINE -(X(3), Y(3)), 0: LINE -(X(1), Y(1)), 0
    GOTO 30
```
letöröljük a megrajzolt háromszöget, majd a programot visszairányítjuk újabb háromszög megrajzolására.

```
END IF
END
```

Feladatok

41. Módosítsd a programot úgy, hogy –10 és +10 közötti koordinátákat állítson elő! Módosítsd a WINDOW utasítást, illetve a tengelyrajzokat is az új értékeknek megfelelően!
42. Készíts programot, amely az $Y = 1/10 * (X - 2)^2 - 5$ szabály szerinti pontsorozatot ábrázolja koordináta-rendszerben!

NÉZŐKE DEFINIÁLÁSA

A WINDOW utasítás használatakor az egész képernyőn definiáltunk egy felhasználói koordináta-rendszert. Abban az esetben, ha ezt nem a teljes képernyőre szeretnénk megvalósítani, hanem annak csak egy részére, akkor a VIEW parancsot kell alkalmaznunk. A parancs használatát meghatározza az a tény, hogy alkalmaztunk-e előtte WINDOW utasítást. A továbbiakban azt az esetet részletezzük, amikor a WINDOW parancsot nem adtuk ki a VIEW előtt.

VIEW [SCREEN](X1,Y1)-(X2,Y2)[,[színkód][, határszín]]

X1,Y1	*a nézőke bal felső sarka,*
X2,Y2	*a nézőke jobb alsó sarka,*
színkód	*ilyen színre festi a nézőkét,*
határszín	*ilyen színű keretet rajzol.*

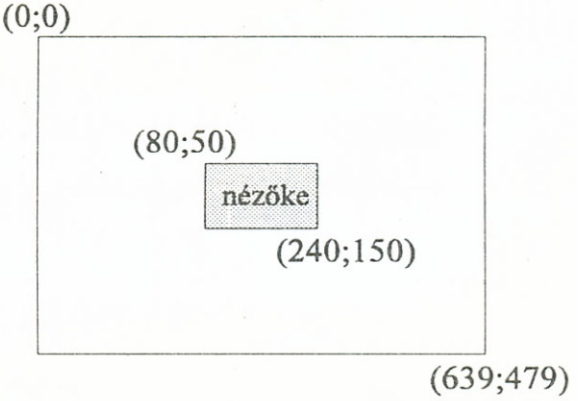
Paraméterek nélkül megadva a VIEW a teljes képernyőt jelöli ki nézőkének. Egyszerre több is definiálható, de csak egy lehet aktív. A SCREEN utasítás megszünteti a nézőkét.

Mintaprogram

```
SCREEN 12
VIEW (80, 50)-(240, 150), 1, 3
CIRCLE (10, 10), 30, 2
```

Rajzold bele a kört az itt levő nézőkébe úgy, ahogy a képernyőn látod!

Mi történik azokkal a pontokkal, amelyek nem esnek a nézőkébe?

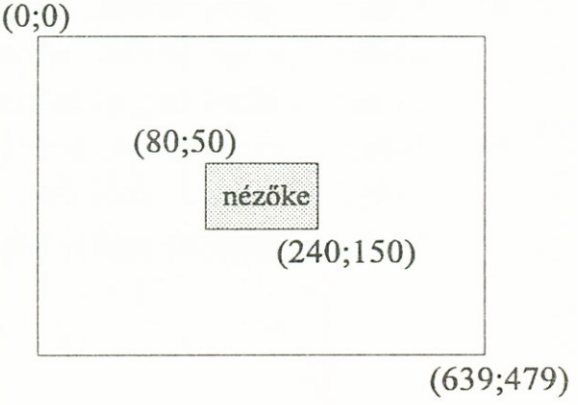


Módosítsd a programot az alábbiak szerint!

```
VIEW SCREEN (80, 50)-(240, 150), 3, 5
CIRCLE (80, 80), 30, 14
```

Rajzold bele a nézőkébe a kört úgy, ahogy a képernyőn látszik!

Van-e, illetve mi a különbség a VIEW és a VIEW SCREEN utasítások között?



Gépeld be az alábbi programot, majd írd a kipontozott részre az adott utasítások képernyőn bekövetkezett hatását!

```
REM view1
SCREEN 12
LINE (0, 0)-(319, 199),, B
FOR i = 1 TO 6
    SELECT CASE i
        CASE 1
         VIEW (50, 20)–(210, 120), 1, 1 .....
        CASE 2
         VIEW (60, 30)–(220, 130), 1, 2 .....
        CASE 3
         VIEW (70, 40)–(230, 140), 1, 0 .....
        CASE 4
         VIEW (80, 50)–(240, 150), 1, 3 .....
        CASE 5
         VIEW (90, 60)–(250, 160),, 1 .....
        CASE 6
         VIEW (100, 70)–(260, 170) .....
    END SELECT
    LINE (5, 5)–STEP(10, 10),, B
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = " "
NEXT i
VIEW .....
```

A FOR I = 1 … sor elé, egy üres sorba, szúrd be:

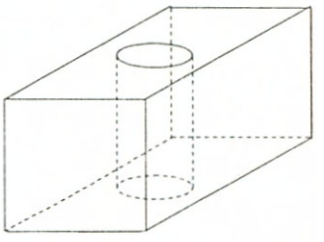
```
WINDOW (0, 0)–(200, 200)
```

Milyen módon befolyásolja a program működését ez az utasítás?

.....

Feladatok

43. Készíts programot, amely külön-külön nézőkékben ábrázolja az ábrán látható test vetületi képeit (elölnézet, felülnézet, oldalnézet, …)!



44. Készíts programot, amely négy nézőkét állít be a képernyőre, majd azokban külön-külön ábrázolja a következő szabályokkal adott pontsorozatokat:

$Y1 = 2*(X - 2);$ $Y2 = ABS(X - 2) - 3;$

$Y3 = (X - 2)^2 - 4;$ $Y4 = 16/(X + 2) - 3.$

KÉP TÁROLÁSA, MEGJELENÍTÉSE

Az előzőekben elkészített programok egyszerű ábrákat, alakzatokat rajzoltak a képernyőre. Ezen rajzok ismételt képernyőre rajzolásához több lehetőségünk van:

- Ismétlési szerkezettel, változók használatával készíthetjük el a rajzot.
- Programban tároljuk („lefényképezzük") az adott területen levő rajzot, majd a szükséges helyen megjelenítjük az ábrát.

Ez utóbbi lehetőséget részletezzük.

Az adott alakzatot tartalmazó területet tároljuk a GET utasítással, majd a szükséges helyen és időben a PUT utasítással jelenítjük meg.

GET [STEP] (X1,Y1) - [STEP] (X2,Y2),tömb_név

X1,Y1 *A tárolni kívánt téglalap bal felső sarka.*

X2,Y2 *A tárolni kívánt téglalap jobb alsó sarka.*

Step *Ez határozza meg, hogy a koordináták számítását a legutolsó pixelpozíciótól kezdve vegye-e figyelembe.*

SCREEN mód	*FB*	*Szorzó*
1	2	1
2	1	1
7	1	4
8	1	4
9	1	2
10	1	2
11	1	1
12	1	4
13	8	1

Tömb_név *Annak az adattömbnek a neve, amelyben a GET utasítás tárolja a képet. Használata előtt a DIM paranccsal létre kell hozni. A tömb méretét byte-okban a következő képlettel számíthatjuk ki:*

MÉRET = 4 + INT((X2 – X1 + 1) * (FB + 7/8) * szorzó * (Y2 – Y1 + 1))

A képletet ritkán alkalmazzuk. Helyette becsült számértéket használunk!

A képernyőtartalom visszaállítására a PUT utasítást használhatjuk:

PUT [STEP](X,Y),Tömb [Kijelzés]

X,Y *Ezen a helyen jelenítjük meg a GET-ben tárolt téglalapot. Ez a koordináta lesz a GET-ben szereplő téglalap bal felső sarkának koordinátája.*

Tömb *A GET-ben szereplő tömb neve.*

Step *Hatása megegyezik a GET-ben leírtakkal.*

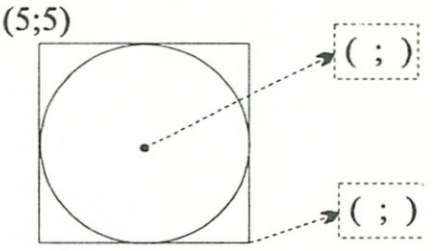
Kijelzés *Lehet:* PSET
PRESET
AND
OR
XOR

Hatásukat majd mintaprogramban vizsgáljuk meg.

Mintaprogram

Az alábbi programot gépeld be, majd válaszolj a kérdésekre!

```
SCREEN 12
CIRCLE (10, 10), 5, 2
PAINT (10, 10), CHR$(&H11) + CHR$(&H0), 2
x1 = 5: y1 = 5: x2 = 15: y2 = 15: fb = 1: szorzo = 4
meret = 4 + INT((x2 - x1 + 1) * (fb + 7 / 8) * szorzo * (y2 - y1 + 1))
DIM tomb(meret)
GET (x1, y1)-(x2, y2), tomb      A létrehozott tömbben tároljuk az adott területet.
PUT (x1, y1), tomb, XOR          A megjegyzett képet XOR-művelettel tüntetjük el.
FOR i = 1 TO 200
    PUT (i, 10), tomb
    FOR q = 1 TO 40: NEXT q
PUT (i, 10), tomb, XOR
NEXT i
```



A rajzon pótold a hiányzó koordinátákat!
Szükség szerint módosítsd a programot gyorsabb vagy lassabb mozgásra.

Feladatok

45. A programot felhasználva készíts egy ágyút, ami adott billentyűre lövéseket ad le!

46. Készíts programot, amely egy tojás leejtését szimulálja!

Mintaprogram

A □ helyére írd be felváltva a PSET, PRESET, AND, illetve OR kijelzéseket, majd futtasd a programot!

```
SCREEN 12
CIRCLE (10, 10), 5, 2: PAINT (10, 10), CHR$(&H11) + CHR$(&H0), 2
x1 = 5: y1 = 5: x2 = 15: y2 = 15: fb = 1: szorzo = 4
meret = 4 + INT((x2 – x1 + 1) * (fb + 7 / 8) * szorzo * (y2 – y1 + 1))
DIM tomb(meret)
GET (x1, y1) – (x2, y2), tomb, □
CIRCLE (100, 100), 50, 1: PAINT (100, 100), CHR$(&H0) + CHR$(&HAA), 1
FOR i = 1 TO 200
    PUT (i, 100), tomb
    FOR q = 1 TO 4000: NEXT q
    PUT (i, 100), tomb, XOR
NEXT i
```

Feladatok

47. Írd ide, mit tapasztaltál az egyes esetekben!
    PSET .....
    PRESET .....
    AND .....
    OR .....
48. Készíts egy festett kört, majd tárold egy tömbben a képét! Helyezz a képernyőre három, egymást kölcsönösen metsző kört úgy, hogy a közös részek más-más mintájúak legyenek!

Mintaprogram

Készítsünk programot, amely egy emberke „távolugrását" mutatja be!

```
REM UGRAS
DIM A(400)
DEF FNF(X)=1/50*X^2-90
SCREEN 12
DRAW "S12 BM 20,20 C2"          Megrajzoljuk az emberkét.
DRAW "U1 NL2 U2 BU1 ": CIRCLE STEP(0, 0), 2, 2
DRAW "BM20,20 G2F2 BM20,20 F3 "
GET (0, 0)-(36, 36), A: CLS     Az alakzatot tároljuk egy A nevű tömbben.
WINDOW SCREEN (–100, –100) – (100, 100)
LINE (–100, 70) – (100, 70), 2
FOR X = 85 TO –85 STEP –.5
   PUT (X, FNF(X)), A, PSET     A kiszámított helyen megjelenítjük a tárolt alakzatot.
   FOR Q = 1 TO 100: NEXT Q
NEXT X
```

Feladatok

49. Módosítsd a programot úgy, hogy más szabály szerint mozogjon az alakzat!
50. Készíts programot, amely egy korongot mozgat a következő képlettel adott pontsorozaton:

$$G(X) = SQR(20*X^2 + 9.81/4*X^4)$$

ALGORITMUSOK

Az arab *Muhammed ibu Muza al-Khvarizmi* a IX. században könyvet írt a legfontosabb számolási módszerekről, eljárásokról. A könyv latin címéből származtatjuk a feladatok megoldásait egymás utáni lépések sorozataként megadó eljárás nevét, az algoritmust.

Algoritmuson valamely egyértelműen előírt módon és sorrendben végrehajtandó tevékenységet értünk. Ez a mondat az algoritmus fogalmának egyszerű leírása. Konkrét magyarázat helyett vizsgáljunk meg egy módszert, amelyet két természetes szám legnagyobb közös osztójának meghatározására dolgozott ki Eukleidész, görög matematikus. Ezt az eljárást nevezik euklideszi algoritmusnak.

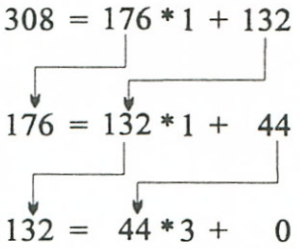
EUKLIDESZI ALGORITMUS

A matematikaórák egyik nehéz feladata a nagy számlálójú és nevezőjű törtek egyszerűsítése. Az egyszerűsítéshez a számláló és a nevező legnagyobb közös osztóját kell meghatározni, majd ezzel a számmal elvégezni az egyszerűsítést. Az alább ismertetésre kerülő módszer nagyon lerövidíti ezt a munkát.

Egyszerűsítsük a $\frac{308}{176}$ *törtet !*

A matematikaórákon ezt prímtényezős felbontással, majd a legnagyobb közös osztó megkeresésével végeztük el.

Eukleidész eljárása a következő: A nagyobb számot írjuk fel a kisebbik többszöröseként, majd vizsgáljuk meg a maradékot.



Látható, hogy az utolsó maradék nulla. Az utolsó előtti maradék pedig 44. Ezzel eloszthatók maradék nélkül a 308 és a 176 is!

308 : 44 = 7 176 : 44 = 4 és maradt a nulla.

Ez a legnagyobb közös osztójuk.

nagyobb, kisebb szám maradék

A		B			M											
308	=	176	*	1	+	132	⇨	132	=	308	–	176	*	1	⇨	1 = [308/176]
176	=	132	*	1	+	44	⇨	44	=	176	–	132	*	1	⇨	1 = [176/132]
132	=	44	*	3	+	0	⇨	0	=	132	–	44	*	3	⇨	3 = [132/44]
								M	=	A	–	B	*	□		□ = [A/B]

Általánosítsuk az eljárást! M = A – B * [A/B]
BASIC nyelven: M = A – B * INT(A/B)

Az alábbiakban elkészítettük az eljárás BASIC nyelvű programját. Gépeld be, majd kipróbálása után válaszolj a kérdésekre!

```
REM *** EUKLIDESZI ALGORITMUS ***
CLS
DO
    INPUT "A=";A
LOOP UNTIL A=INT(A) AND A>2 AND A<32767
X=A
DO
    INPUT"B=";B
LOOP UNTIL B=INT(B) AND B>2 AND B<32767
Y=B
IF A/B=INT(A/B) THEN LN=B: GOTO 80
IF B/A=INT(B/A) THEN LN=A: GOTO 80
60 N=INT(A/B):M=A-B*N
IF M>0 THEN LN=M: A=B: B=M: GOTO 60
80 PRINT "LNKO=";LN
PRINT "LKKT=";X*Y/LN
```

Mi a szerepe az X = A értékadásnak?

.....

.....

Értelmezd a 11. és a 12. sort!

.....

.....

Mi a matematikai jelentése az utolsó sor hatására a képernyőn megjelenő számnak?

.....

.....

FELADATOK

1. Készíts programot, amely egy tört számlálóját és nevezőjét beolvasva egyszerűsíti a törtet!
2. Készíts programot, amely két tört számlálóját és nevezőjét olvassa be, majd összeadja a törteket! A végeredményt egyszerűsítsd az előző eljárás segítségével!
3. Módosítsd az előző programot úgy, hogy mind a négy alapműveletet végezze el a program, s a végeredményeket is egyszerűsítse!
4. Módosítsd az előző program kiírásait úgy, hogy azokat a törteket, amelyeknek nevezője 1, egész számként írja ki!

Az euklideszi algoritmus kis átalakításával lehetőségünk nyílik adott természetes számok más számrendszerekbe való átváltásához is!

Számrendszer 1.

Váltsuk át kettes számrendszerbe a 213 tízes számrendszerbeli számot!

A korábbi eljárás szerint táblázatot használnánk.

Helyiértékek	128	64	32	16	8	4	2	1
	1	1	0	1	0	1	0	1

Eszerint : $213_d = 11010101_b$

Az osztási eljárás sokkal egyszerűbb!

53 : 2 = 26 és maradt az 1
26 : 2 = 13 és maradt a 0
13 : 2 = 6 és maradt az 1
6 : 2 = 3 és maradt a 0
3 : 2 = 1 és maradt az 1
1 : 2 = 0 és maradt az 1

↑ Ha a maradékokat fordított sorrendben olvassuk, a táblázat eredményét kapjuk!

Ezt az eljárást valósítjuk meg az *euklideszi algoritmus* felhasználásával.

A bevitt szám		A számrendszer alapszáma		Szorzó		Maradék
53	=	2	*	26	+	1
26	=	2	*	13	+	0
13	=	2	*	6	+	1
6	=	2	*	3	+	0
3	=	2	*	1	+	1
1	=	2	*	0	+	1

Az eljárás 0 szorzóig tart!

A maradékok fordított sorrendje a szám kettes számrendszerbeli alakja!

$$53_{10} = 110101_2$$

Az eljárás BASIC nyelvű megvalósítása az előző program módosítása. Az elsőnek beadott számot váltjuk át a másodikként beadott számú számrendszerbe. Problémát csak a maradékok fordított sorrendben történő kiíratása jelent.

```
REM *** SZÁMRENDSZER 1 ***'
CLS: P = 25
DO
    INPUT "A="; A
LOOP UNTIL A = INT(A) AND A > 2 AND A < 32767
X = A
DO
    INPUT "ALAPSZÁM="; B
LOOP UNTIL B = INT(B) AND B >= 2 AND B <= 16
60 N = INT(A / B): M = A – B * N
IF N > 0 THEN LOCATE 4, P: PRINT M: A = N: P = P - 2: GOTO 60
LOCATE 12, P: PRINT M
```

Feladatok

5. Készíts programot, amely a maradékokat egy tömbbe tölti, majd onnan írja ki a képernyőre!
6. Készíts programot, amely a 10-es számrendszerben megadott számot 2-től 9-es számrendszerbe váltja át!

A fenti programok a 2...9-es számrendszerbe váltották át a 10-es számrendszerben adott számokat.

Számrendszer 2.

Készítsünk programot, amely 2...16-os számrendszerbe váltja az általunk 10-es számrendszerben adott számokat.

Tudjuk, hogy a tízesnél nagyobb számrendszerekben az adott helyiértékekre az **A; B; C; D; E; F** betűket kell írni. Ezek karakterek, tehát minden számot karakterként kell kezelnünk. Ezt a 2...9 számok esetén az **STR$(X)** függvénnyel tehetjük meg. Az említett **A...F** karaktereket más módon kell előállítani, ezért ezeket karakteres tömbben tároljuk.

```
CLS: P = 25
DIM C$(6)
DATA A,B,C,D,E,F
FOR i = 1 TO 6: READ C$(i): NEXT i
```

A program további része megegyezik az előző programmal, csak az alapszám értékénél engedjük meg a 2...16 érték elfogadását is.

```
DO
    INPUT "A="; A
LOOP UNTIL A = INT(A) AND A > 2 AND A < 32767
DO
    INPUT "ALAPSZAM="; B
LOOP UNTIL B = INT(B) AND B >= 2 AND B <= 16
60 N = INT(A / B): M = A - B * N
```

*A program eddig meghatározza a maradék értékét, amelyek már lehetnek 9-nél nagyobb értékek is. Nekünk a 10-es maradék esetén **A** betűt, 11-es maradék esetén **B** betűt, ... 15-ös maradék esetében **F** betűt kell írnunk. Ezek sorban a tömb első, második, ... hatodik elemei.*

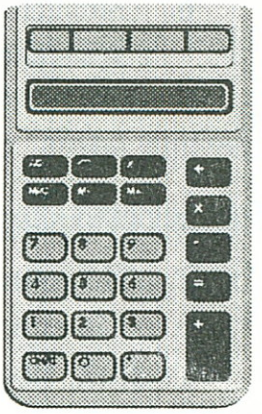
```
IF M > 9 THEN M$ = M$(M - 9) ELSE M$ = STR$(M)
```

A sorszámokat 9-nél nagyobb maradék esetén egyszerűen megkaphatjuk. A kapott maradék értékéből kivonunk 9-et. 9 vagy ennél kisebb maradék esetében a maradékot karakterré alakítjuk.

```
IF N > 0 THEN LOCATE 4, P: PRINT M$: A = N: P = P - 1: GOTO 60
LOCATE 12, P: PRINT M$
```

A programunk már képes a 16-os számrendszerbe is átalakítani egy 10-es számrendszerben megadott számot.

Feladatok



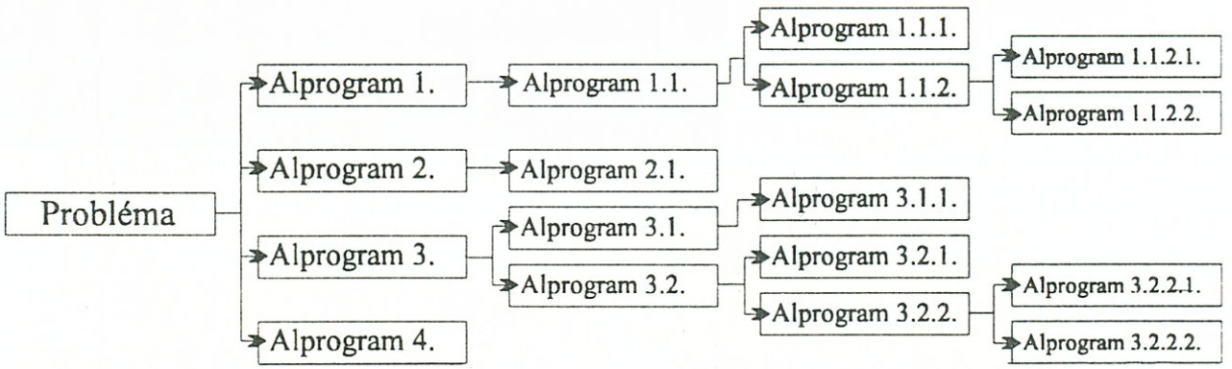
7. Készíts programot, amely bármely (2...16) számrendszerben megadott számot bármelyikbe (2...16) átvált!
8. Készíts programot, amely bekéri egy minta egy sorának kettes számrendszerben adott kódjait, majd kiírja a minta hexadecimális (16-os) számrendszerbe átszámított értékeit!
9. Módosítsd az előző programot úgy, hogy a tervezett minta sorainak értékeit kéri, majd az így kapott értékek felhasználásával befest egy téglalapot!

ALPROGRAMOK (SZUBRUTINOK)

Bonyolultabb programozási feladatok megoldásánál az eddig tanult megoldási algoritmus nem járható út. A probléma sokszor annyira összetett, hogy egyetlen programozási folyamatban nem oldható meg. Ebben az esetben a megoldási algoritmust részekre szedjük, s megpróbálunk a részproblémákra megoldást találni. Abban az esetben, ha a részfeladat is bonyolult, azt is részekre szedjük, s megpróbáljuk felírni a megoldási algoritmusát. Az eljárást addig folytatjuk, míg a részfeladatokból olyan elemi problémát kapunk, ami már megoldható.

Az alprogramokat szubrutinoknak is nevezzük.

A részekből ezután visszafelé felépítjük az egészet. Az alábbi ábra egy ilyen részproblémákra bontást mutat.



Az előző folyamatban létrehozott alprogramokat egységes rendszerré a GOSUB – RETURN utasításpárok segítségével tehetjük.

GOSUB név

Hatására az adott **névvel** ellátott szubrutinra adódik át a program vezérlése. A **név** lehet sorszám vagy rutinnév is. (A név nem tartalmazhat speciális [például: ékezetes] karaktereket.) Az adott rutinban levő utasítások végrehajtása mindaddig történik, amíg egy **RETURN** utasításhoz nem ér.

RETURN [név]

Ha a RETURN mögött nem szerepel név, a szubrutin végrehajtása befejeződik, s a program végrehajtása visszatér a **GOSUB** utáni utasításra. Ha a **RETURN** után **név** is szerepel, akkor az adott nevű címkére vagy szubrutinra tevődik át a vezérlés.

Vizsgáljuk meg, hogyan működik mindez az előző rajz egy része alapján. Az egyes részek nevéből a magyar helyesírás szabályai szerinti pontokat és szóközöket a névadás szabályai miatt hagytuk ki.

```
REM Főprogram
    GOSUB Alprogram1
    ...
END
```

```
Alprogram1:
    GOSUB Alprogram11
        ...
    RETURN
Alprogram11:
    GOSUB Alprogram111

    RETURN
```

és így tovább a nyilak szerint!

A programnak akkor lesz vége, ha minden alprogram a megadott sorrendben végrehajtódik, s a vezérlés visszatevődik a főprogram END-jére!

Az itt tapasztaltakat fogjuk felhasználni egy játékprogram elkészítéséhez!

Torpedó

A játék jól ismert! Két játékos játszik két ugyanolyan méretű táblán. Mindegyik azonos mennyiségű „hajót" helyez el a saját tábláján, s felváltva „lőnek" a „hajó" koordinátájának megadásával. Az győz, aki előbb „süllyeszti el" a másik összes hajóját. A játék egyszerűbb változatát készítjük el számítógépen. A gép elhelyez 10 golyót egy 10×10-es táblán, s te a koordináták megadásával lőhetsz. Akkor nyersz, ha mind a 10 golyót eltaláltad.

A programot szedjük szét elemeire, majd próbáljuk meg ezeket az elemeket megtervezni, s kivitelezni! Az elkészített alprogramokat ki kell próbálni, tesztelni, ezért egyes esetekben konkrét értékekkel dolgozunk!

Elsőként tehát el kell döntenünk, milyen részekből áll a program. Ezt az előző oldalon levő rajzok alapján készítjük el.

Szükségünk lesz főprogramra, amelyből az alprogramokat meghívjuk.

Alprogramok lehetnek:

- Tábla megrajzolása, feliratozása (oszlopok betűvel, sorok számokkal).
- Két különböző korong (az egyik a találatot, másik a nem talált lövést jelöli) kirajzolása.
- A korongok véletlen elhelyezéséhez szükséges egy „matematikai tábla", amelybe 10 helyre véletlen számokkal egyest írunk.
- Az „lövéshez" az oszlop (betű) és a sor (szám) beolvasását elvégző rutin.
- A találat ellenőrzését is el kell végeznünk.
- A játék feladására is lehetőséget kell adni (V betű).
- A játék megnyerése vagy feladása után illik felkínálni még egy játék lehetőségét.
- Újabb játék esetén a „matematikai tábla" értékeit nullázni kell.
- Ha nincs új játék, vége!

A táblát grafikus képernyőn fogjuk megrajzolni.

SCREEN 12

Szükségünk lesz a két korong területének megjegyzéséhez két tömbre (X1,X2), a véletlenszámok előállításához (T), illetve a lépések ellenőrzéséhez (L) oszlopokat illetve sorokat tartalmazó tömbökre.

DIM X1(64), T(10, 10), X2(64), L(10, 10)

Elkészítjük a főprogram vázát. (Az alprogramok egy részét innen hívjuk meg.)

```
REM ******* főprogram *******
FOPROGRAM:
DO
    GOSUB ...          A későbbiekben elkészülő alprogramokat itt kell meghívni.
LOOP UNTIL DB = 10
IF DB = 10 THEN LOCATE 12, 50: PRINT "B R A V O !!!"
...
END                    Ez az END csak addig szükséges, amíg a program még nem teljesen működőképes.
REM ******* főprogram VÉGE***
```

❶ Készítsük el a két különböző korongot, majd jegyeztessük meg a területüket! Mi csak az egyik korongot készítettük el. A másikat neked kell megrajzolnod!

```
KORONG:
    CIRCLE (10, 10), 9, 2: PAINT (10, 10), 1, 2
                           Megrajzoljuk, majd befestjük.
    GET (0, 0)–(20, 20), X1   Az X1 tömbben tároljuk.
    PUT (0, 0), X1, XOR       „Eltüntetjük" a korongot.
RETURN                        Visszatérés a Főprogramba.
```

Ezután szúrd be a Főprogramba a GOSUB KORONG utasítást a FOPROGRAM címke elé. A kipróbáláshoz szúrd be a korong „eltüntetését" végző sor elé: REM. (Ne felejtsd el később kitörölni!)

❷ Készítsük el a táblát! Egy négyzetrácsos táblára lesz szükségünk. Ahhoz, hogy bármilyen PC-n futtatható legyen a program, használjuk a WINDOW SCREEN utasítást. Az alprogram kipróbálásához szúrd be a FOPROGRAM címke után: GOSUB TABLA.

```
TABLA:
    WINDOW SCREEN (0, 0)-(200, 200)
    FOR S = 10 TO 140 STEP 13
        LINE (10, S)-(110, S), 3   A függőleges vonalakat rajzoljuk meg.
    NEXT S
    FOR O = 10 TO 110 STEP 10
        LINE ...                   Pótold a hiányzó sort! A vízszintes vonalakat kell itt meghúzni.
    NEXT O
    oszlop = 7
```

```
    FOR Q = 65 TO 74
        LOCATE 1, oszlop
        PRINT CHR$(Q)          Az ABC... betűket íratjuk ki a rács fölé. Ehhez az
                               adott betű kódjait használjuk.
        oszlop = oszlop + 4
    NEXT Q
    SOR = 3
    FOR p = 0 TO 9
        LOCATE ...             A sorok számának kiírása a rács elé kerüljön. A
                               sorszámokat 0-tól 9-ig íratjuk ki, mert a későbbi
                               adatbeolvasást nagyon megnehezítené egy kétjegyű
                               szám beolvasása
        SOR = SOR + 2
    NEXT p
RETURN                         Visszatérünk a GOSUB utáni helyre.
```

❸ A „matematikai táblába" elhelyezünk 10 db egyes számot. Ezek jelölik a korongok helyét. A kipróbáláskor célszerű a már megjegyzett területű korong kirajzolása a táblába (később töröld ki!). A GOSUB TABLA rutin után szúrd be GOSUB VELETLEN!

```
VELETLEN:
    RANDOMIZE TIMER
    FOR Q = 1 TO 10
        DO
            X = INT(RND(1) * 10 + 1)
            Y = INT(RND(1) * 10 + 1)
        LOOP UNTIL T(X, Y) <> 1    Addig ismételje a lépésorozatot, amíg az egy 0 érté-
                                   kű helyre nem kerül.
        PUT (X * 10 + 2, Y * 13), X2   Ha az előző feltétel teljesül, kirajzoltatjuk a koron-
                                       got,
        T(x, y) = 1                majd az adott cella értékét 1-re változtatjuk.
    NEXT Q
RETURN
```

❹ Az adatok beolvasása nekünk az adott oszlop, illetve sor értékének beolvasását jelenti. Az oszlopnál betűket, a sornál számokat kell adatként elfogadni. A képernyő adott [(5;50), illetve (7;50)] helyén fogjuk az adatbeolvasást megtenni. Az előző GOSUB után szúrd be GOSUB ADATBEOLVAS!

```
ADATBEOLVAS:
  REM *** OSZLOP ***
    LOCATE 5, 57: PRINT "  "   A többszöri szubrutinhívás miatt célszerű az előző-
                               ekben beírt számokat letörölni a képernyőről.
    LOCATE 7, 57: PRINT "  "
    LOCATE 5, 50: PRINT "OSZLOP="
```

```
    DO
        BETU$ = INKEY$          A nagybetű-beolvasást addig ismételjük, amíg az
                                „A"...„J" betűknek, illetve a feladáshoz szükséges
                                „V" betűnek megfelel.
    LOOP UNTIL BETU$ >= "A" AND BETU$ <= "J" OR BETU$ = "V"
    LOCATE 5, 57: PRINT BETU$   Az adott helyre kiíratjuk a karaktert.
    IF BETU$ = "V" THEN GOTO FELADAS
                                Ezt a sort a kipróbálás idejére REM-mel lásd el,
                                mert a játék feladása rutin még nem készült el!
    OSZL = ...                  Pótold a hiányzó részt úgy, hogy az OSZL változó a
                                BETU$-ból kapott 1...10 értéket tartalmazza!
  REM *** SOR ***
    LOCATE 7, 50: PRINT "SOR="
    DO
        SZAM$ = INKEY$          Pótold a hiányzó részt úgy, hogy a SZAM$ értéke
                                csak 0...9 lehet!
    LOOP UNTIL ...
    LOCATE 7, 57: PRINT SZAM$
    SOR = ...                   A SOR értékét a SZAM$-ból kapod. Írd be!
  REM *** HA MAR VOLT ITT LOVES ****
    IF L(OSZL, SOR) = 2 THEN    Abban az esetben volt már erre a helyre lövés, ha
                                az adott cella értéke 2. Pótold a kipontozott részt
                                úgy, hogy az előzőleg már képernyőre került beírá-
                                sokat törölje ki! A rossz lövést hang megszólaltatá-
                                sával is jelezd!
        LOCATE 10, 50: PRINT "Ide már lőttél!"
        PLAY "..."
        LOCATE ...
        LOCATE ...
        LOCATE 10, 50: ...
        GOTO ADATBEOLVAS        Ebben az esetben (Rossz lövés!) visszairányítjuk a
                                programot az adatbeolvasás elejére.
        ELSE L(OSZL, SOR) = 2   Ha jó lövést tettünk, akkor az adott lépés értékét 2-
                                re változtatjuk.
    END IF
RETURN
```

❺ Ha lőttünk egyet, meg kell vizsgálni, vajon van-e ezen a helyen korong. Módosítsd a Főprogramot, hogy ezt a szubrutint is meghívja!

```
VIZSGALAT:
    IF T(OSZL, SOR) = 1 THEN    Az ADATBEOLVAS szubrutinban az OSZL és a
                                SOR változók megkapták értéküket. Erre hivatko-
                                zunk a T(oszl,sor) segítségével. Ha a cella értéke 1,
```

```
        PUT (OSZL * 10 + 2, SOR * 13), x1
                                    akkor az adott helyen megjelenítjük a találatot jelző
                                    korongot.
        FOR Q = 1 TO 3              Egy egyszerű dallam segítségével jelezzük a találatot.
            PLAY "O2 C12 E12 G12 O3 C4"
        NEXT Q
        DB = DB + 1                 Ez a változtatás amiatt kell, hogy menet közben ellenőrizhessük, hány találatunk van. A lövéseket addig végezzük, amíg vagy eltaláltuk az összeset, vagy feladtuk.
    ELSE ...                        Ha a lövésünk nem talált, akkor a másik típusú korongot kell a képernyőre kirajzolni. Pótold!
    END IF
RETURN
```

❻ Ha az adatbeolvasás során feladjuk a partit (V betűt nyomtunk), a programot ide irányítjuk. A még meg nem talált golyókat kell kirajzoltatni a képernyőre. Ez annyit jelent, hogy a T tömbben elhelyezett 1-esek helyén akkor kell kirajzolni a korongot, ha oda még nem lőttünk, vagyis az L tömb cellaértéke nem 2. Pótold a hiányzó részt!

```
FELADAS:
    FOR I = 1 TO 10
        FOR J = 1 TO 10
            IF ... THEN             Feltétel ...
                PUT ...             Korongkirajzolás
            END IF
        NEXT J
    NEXT I
GOTO MEGEGY                         A játék feladása után természetesen kérdezze meg a program, hogy kívánunk-e még játszani. Ez a vezérlésátadás itt, illetve a Főprogramban szerepel akkor, ha eltaláltuk mind a 10 golyót. Módosítsd a Főprogramot a leírtak szerint!
```

❼ Ha még egy partit szeretnénk játszani, a képernyőre ki kell íratni, majd az adott (I/N) billentyűkre várakozás után IGEN esetben nullákkal kell feltöltenünk a két (T illetve L) tömböt, s a programot vissza kell irányítanunk a Főprogram címkéjére. A NEM esetben vége a játéknak. Pótold a hiányzó utasításokat!

```
MEGEGY:
    LOCATE 23, 50: PRINT "JÁTSZUNK MÉG EGYET? [I/N]"
    DO
    J$ = ...
    ...
```

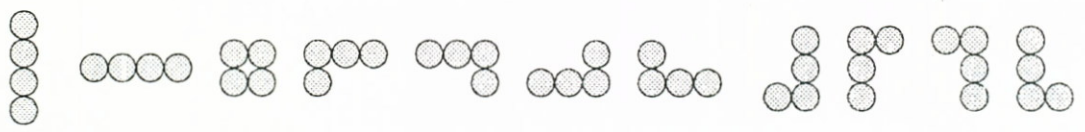
```
    IF J$ = "I" THEN
        CLS
        GOSUB NULLAZAS
        GOTO FOPROGRAM
    END IF
    IF J$ = "N" THEN GOTO VEGE
REM *** NULLÁZÁS ***
NULLAZAS:
    FOR I = 1 TO 10
        FOR J = 1 TO 10
            T(...) = ...
            L(...) = ...
        NEXT J
    NEXT I
RETURN
REM **** VEGE ****
VEGE:
...
```

Az elkészült programot többször teszteld le!

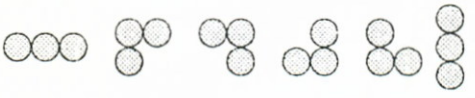
Javítási ötletek

- Ha mind a 10 golyót kilőtted, a program játsszon le egy dallamot!
- A program nem vizsgálja azokat az eseteket, amikor két vagy több golyó a négyzetrácson egymás mellé kerül (egyes golyók esetében). Módosítsd a programot!
- Az igazi TORPEDÓ játékban több golyó is egymás mellé kerülhet úgy, hogy csak a négyzetek oldalaival érintkezhetnek egymással.

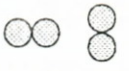
A rajzok a lehetséges formációkat mutatják!
Helyezz el a táblán összesen:
egy „négyes hajót”,



két „hármas hajót”,



három „kettes hajót”,



illetve négy „egyes hajót”
úgy, hogy a más típusú hajók még sarkaikkal sem érintkezhetnek!

SZUBRUTINSZERVEZÉS

Az előzőekben használt GOSUB – RETURN páros itt nagyon lassúvá tenné a program működését. A Quick BASIC kitűnő lehetőséget ad a szubrutinok szervezésére.

Szubrutin meghatározása

DECLARE SUB Szubrutin_név (változólista)

Szubrutin_név *A már ismert módon a szubrutin általunk választott nevét tartalmazza. Ezzel a névvel hivatkozunk rá.*

változólista *Azokat a változóneveket kell felsorolni vesszővel elválasztva, amelyeket át szeretnénk adni a hívott rutinnak.*

Szubrutin megadása

SUB szubrutin_név (paraméterek) [STATIC]

szubrutin_név *Megegyezik a DECLARE utasításban használttal.*

paraméterek *A DECLARE utasításban adott változólistára vonatkoznak.*

STATIC *A kulcsszó megadásával elérhetjük, hogy az alprogramok megőrizzék a változóik értékét a hívások között.*

Szubrutin hívása programból

1. **Rutin_név (változó)** *Beírjuk a szubrutin nevét, majd utána azt a változónevet (neveket), amit a rutin használni fog. (Zárójelek közé is lehet írni!)*
2. **CALL Rutin_név (változó)** *CALL rutin_név majd a változónevet (neveket), amiket a szubrutin használhat.*

Mintaprogram

```
DECLARE SUB Rutin1 (x)
DECLARE SUB Rutin2 (y)
CLS
FOR i = 1 TO 5
    Rutin1 (i)
    CALL rutin2(i)

NEXT i
SUB rutin1 (i)
```

Két szubrutint deklarálunk.

Az első szubrutint hívjuk meg.

A második szubrutin hívását a CALL utasítással tesszük.

*A szubrutin nevét begépelve eltűnik a főprogram. Válaszd a Quick BASIC menüiből a VIEW pontot, s ezen belül a SPLIET almenüt. A képernyő két részre osztódik. Az egyik részben az előzőleg begépelt Főprogram, a másikban az alprogram látszik Bármikor megnyomva az **F2 billentyűt**, egy megjelenő*

ablakban kiválaszthatod, hogy a program melyik részét (főprogram, szubrutin) szeretnéd szerkeszteni az adott képernyőterületen.

```
    Z = Z + i * i
```
Z-be tesszük a főprogram ciklusváltozójának négyzetét az előző érték hozzáadásával.

```
    PRINT "Első rutin. Szorzat ="; Z,
```
A képernyőre szeretnénk íratni a részszorzatok összegét.

```
END SUB
SUB rutin2 (i) STATIC
```
Szeretnénk ha a rutinban szereplő változó megtartaná értékét!

```
    S = S + i
```
Össze szeretnénk adni a ciklusváltozók értékét.

```
    PRINT "Ez a második rutin.";
    PRINT "Összeg ="; S
END SUB
```

Futtasd a programot!

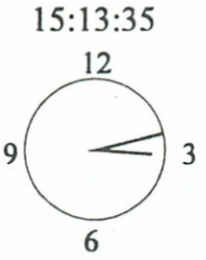
Az első szubrutinban a következő értékadás szerepelt: Z=Z+i*i. Ellenőrizd a gép számításait! Mit tapasztalsz?

.....

Módosítsd a szubrutint, hogy a helyes eredményt kapd!

Óra

Készítsünk programot, ami egy olyan órát rajzol a képernyőre, amely a gépben beállított időt (órát, percet) folyamatosan mutatja!



Minden magyarázat a program mellett megtalálható.

```
DECLARE SUB ORA (MIN$)
```
Egy ORA nevű szubrutint definiálunk, amelyben olyan változónév (MIN$) szerepel, amely csak később kap értéket.

```
SCREEN 12
```
Gyakorlatilag innen kezdődik a főprogram.

```
DO
    CLS
    MIN$ = MID$(TIME$, 4, 2)
    ORA MIN$
```
Most hívjuk meg a szubrutint.

```
    DO
        LOCATE 2, 37
        PRINT TIME$
```
Ellenőrzés céljából kiíratjuk a pontos időt.

```
        TEST$ = INKEY$
    LOOP WHILE MIN$ = MID$(TIME$, 4, 2) AND TEST$ = ""
LOOP WHILE TEST$ = ""
END
```
A programnak itt a vége.

```
SUB ORA (MIN$) STATIC          Innen kezdődik a szubrutin. Ahogy Entert ütsz a sor
                               végén, az előző programrész eltűnik, egy üres sor
                               alatt a szubrutin végét (END SUB) látod.
    LOCATE 23, 30
    PRINT "Bármely billentyűre megáll"
    CIRCLE (320, 100), 175
    HR = VAL(TIME$)            Az óra értékét kapjuk a HR változóban.
    MIN = VAL(MIN$)            A perc értékét a főprogramban kapott értékből vesz-
                               szük.
    KICSI = 360 - (30 * HR + MIN / 2)
                               A kismutató jelenlegi helyzetét így számítjuk ki,
    NAGY = 360 - (6 * MIN)     a nagymutatóét pedig ezzel a képlettel.
    DRAW "TA=" + VARPTR$(KICSI) + "NU40"
    DRAW "TA=" + VARPTR$(NAGY) + "NU70"
                               A két mutatót e két utasítással rajzoltatjuk ki a kép-
                               ernyőre.
END SUB                        A szubrutinnak vége, visszatérés a főprogramba.
```

Feladatok

1. Készíts az előző programhoz olyan szubrutint, amely az óra beosztását (vonalkázás) is elkészíti! Írasd ki a 3; 6; 9 és a 12-es számokat is!
2. Készíts másodperceket is kirajzoltató szubrutint!

GRAFIKUS PONT KOORDINÁTÁINAK LEKÉRDEZÉSE

A grafikus kurzor pillanatnyi pozíciójáról, színéről tájékozódhatunk. Használatának csak a WINDOW utasítást követően van értelme. A függvényt kétféle formában használhatjuk:

POINT (X,Y) *Előállítja a képernyő (X;Y) koordinátájú pontjának színkódját.*

POINT(N) *Az aktuális pont fizikai/felhasználói (WINDOW) koordinátáit állítja elő.*

0	*fizikai x koordináta,*
1	*fizikai y koordináta,*
2	*felhasználói x koordináta,*
3	*felhasználói y koordináta.*

Az alábbi programban egy WINDOW utasítás után megrajzoltuk a koordinátatengelyeket is. A (10, 40) koordinátájú pont megrajzolása után lekérdezzük annak minden értékét.

```
SCREEN 12
WINDOW (-100, -100)-(200, 200)
LINE (-100, 0)-(200, 0), 2
```

```
LINE (0, -100)-(0, 200), 2
CIRCLE (10, 40), 2, 1
PAINT (10, 40), 2, 1
szk = POINT(10, 40): PRINT "színkód = "; szk
X = POINT(0): PRINT "Fizikai x="; X,
Y = POINT(1): PRINT "Fizikai y="; Y
X1 = POINT(2): PRINT "Logikai x="; X1,
Y1 = POINT(3): PRINT "Logikai y="; Y1
```

Futtasd a programot! Változtasd meg a pont koordinátáit, s vizsgáld a megkapott értékeket!

Bolygók mozgása

Készítsünk programot, amely a képernyőn mutatja egy bolygó mozgását!

A megoldás a gyakorlatban egy körvonalon kör mozgatását jelenti. A DRAW parancs segítségével 0 színkódú, ciklusban adott szöggel elforgatott vonalat rajzolunk. Leolvastatjuk a végpont koordinátáját, majd erre a helyre kirajzoltatjuk az előre megjegyzett területű korongot. Kis várakozás után letöröljük, majd további elforgatást készítünk. A kipontozott részekre írd le, mi a feladata az adott soroknak!

```
DECLARE SUB csillag (n)                  Egy szubrutinban N darabszámú csillagot rajzoltatunk ki.

SCREEN 12
DIM a(400)                               Ebben a tömbben fogjuk tárolni a kört.
CIRCLE (5, 5), 3, 2                      .....
GET (0, 0)-(10, 10), a                   .....
PUT (0, 0), a, XOR                       .....
WINDOW SCREEN (0, 0)-(319, 199)          .....
N = INT(RND(1) * 500 + 100)              .....
csillag (n)                              .....
DO
    q$ = "r80"
    FOR y = 0 TO 360 STEP 10
        DRAW "b m 160,100 c0"            .....
        DRAW "ta" + STR$(y)              .....
        DRAW q$                          .....
        X1 = POINT(0): Y1 = POINT(1)     .....
        PUT (X1, Y1), a, PSET            .....
        FOR q = 1 TO 4000: NEXT q        A ciklus végértékét a géped befolyásolja.
        PUT (X1, Y1), a, XOR             .....
    NEXT y
    x$ = INKEY$
```

```
LOOP UNTIL x$ = " "
SUB csillag (n)
    RANDOMIZE TIMER
    FOR L = 0 TO n
        O = INT(RND(1) * 319 + 1)
        S = INT(RND(1) * 199 + 1)
        PSET (O, S), 3
    NEXT L
END SUB
```

Feladatok

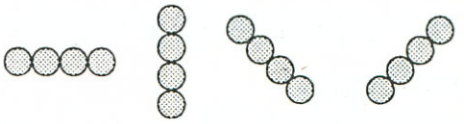
3. Módosítsd a programot úgy, hogy a mozgó korong a HOLD legyen, középen pedig a FÖLD látsszon!
4. A programot felhasználva készíts programot, amely a NAP körül keringő két bolygó, a VÉNUSZ és a FÖLD mozgását modellezi. A FÖLD körül a HOLD keringjen!
5. Az előző programot módosítsd úgy, hogy a keringési idők közel azonosak legyenek a földrajzi atlaszodban leírtakkal!
6. Készíts programot, amely egy csőbe golyókat ejt! A golyók egymásra esnek, de a csőben maximum 9 golyót lehet egymásra helyezni.

Csőamőba

A csőamőbát a következők szerint kell játszani:

Felváltva tesz két játékos egy-egy golyót a 9 cső valamelyikébe, amelyek természetesen leesnek a cső aljára. Az nyer, akinek először sikerül négy golyót nyerő helyzetbe hoznia.

A nyerő helyzetek a következők:

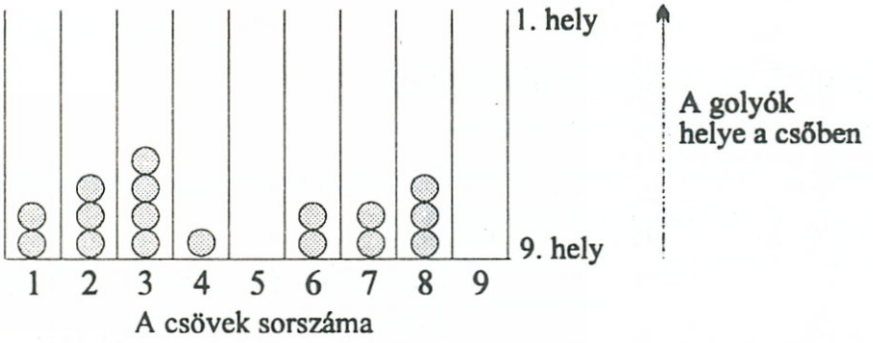


- egymás mellett vagy alatt négy golyó,
- átlós irányban négy golyó.

A **6.** feladat megoldásával gyakorlatilag ennek a játéknak az alapjait készítetted el. Az itt levő megoldás csak egy lehetséges módját mutatja a probléma feldolgozásának. A számítógéppel ennek a játéknak a lépéseit, illetve az ellenőrzését fogjuk megvalósítani!

Egy 9×9-es táblán fogjuk a golyókat elhelyezni. A 9 oszlopban egymás fölött helyezkednek el a golyók. A cső sorszámának benyomására az adott játékosnak megfelelő golyó az adott csőbe hullik.

A golyó táblában elfoglalt helyét a T(9, 9) tömbben fogjuk vizsgálni. Az adott cella értéke a két játékos szerint 1 vagy 2 lesz. Ha nincs az adott helyen golyó, a cella értéke 0.



A cső sorszámát X-szel, a benne levő golyó helyét egy mutatóval jegyezzük meg. Erre a MUT(x) tömböt használjuk. A mutató mindig az adott cső következő üres helyére mutat. Így a rajz szerinti 3. csőben (X = 3) felül levő golyó a 6. helyen van, ugyanekkor MUT(x) = 5. A negyedik cső (X = 4) mutatója MUT(x) = 8. Írd be a rajz szerint a következő értékeket:

Ha X = 1, akkor MUT(x) = ... Ha X = 9, akkor MUT(x) = ...

Mi lesz a mutató értéke, ha a 3. cső megtelik? X = ...; MUT(x) = ...

A megadott lépésekben tehát egy golyó betételével 3 adat változik meg. Az X, illetve a MUT(x) értéke, valamint az adott cella értéke. Ez a T tömbre vonatkoztatva a következőt jelenti:

T(x,MUT(x)) = 1 vagy 2 aszerint, hogy melyik játékosé az adott golyó.

Ezek után készítsük el a programot.

```
CLS
LOCATE 1, 1: INPUT "Kérem az első játékos nevét"; A$
LOCATE 2, 1: INPUT "Kérem a második játékos nevét"; B$
DIM T(9, 9), K1(264), K2(264), MUT(9)
Kezd:
CLS
FOR i = 1 TO 9
    FOR j = 1 TO 9
        T(i, j) = 0
    NEXT j
    MUT(i) = 9
NEXT i
SCREEN 1
WINDOW SCREEN (0, 0)-(510, 200)
GOSUB GOLYOK
END
```

*Ebben a részben lenullázzuk a **T** tömb értékeit.*
Erre a többszöri meghívások miatt van szükség.
Ugyanitt állítjuk be a mutatók értékeit a cső aljára.

*Ez az END csak addig szerepel itt, amíg a **Golyók** és a **Tábla** szubrutint kipróbáljuk. A főprogram megírásával egy időben töröld ki!*

Elsőként a golyókat megrajzoló rutint készítjük el!

```
GOLYOK:
    CIRCLE (8, 8), 7, 2: PAINT (8, 8), 1, 2
    GET (0, 0)-(16, 16), K1
    ...                                   Töröltesd le a megjegyzett területet!
    CIRCLE (8, 8), 7, 1: PAINT (8, 8), 2, 1
    ...                                   Jegyeztesd meg a kört a K2 tömbbe, majd töröld le
                                          a megjegyzett területet!
    FOR i = 1 TO 8
        PUT (10, i * 8), K1               A képernyő két szélére 8-8 golyót rajzoltatunk ki.
        PUT (490, i * 8), K2
    NEXT i
RETURN
```

A játéktáblarutin megírása után a GOSUB GOLYOK rész alá szúrd be!

```
GOSUB TABLA
```

A csövek falának megrajzolása pontos tervezőmunka eredménye, mivel a csövek vastagságát, a golyók méretéhez kellett viszonyítani.

```
TABLA:
    LINE (0, 0)-(510, 200), 3, B      A vízszintes vonalat (csövek alját), majd
    LINE (80, 175)-(422, 175), 3      az első függőleges vonalat (cső falát) rajzoljuk meg.
    FOR W = 80 TO 422 STEP 38
        LINE (W, 100)-(W, 174), 3     Ezután ciklusba szervezve rajzoljuk meg a többi
                                      függőleges vonalat is.
    NEXT W
    ...                               A kipontozott rész helyére írd azt a programrészt,
                                      amely a csövek alá írja azok sorszámát!
RETURN
```

Ezek után készítsük el a főprogramot, amely az összes további alprogram működését irányítja. Gyakorlatilag egy adatbeolvasó programrészre van szükségünk, amely az alábbiakat biztosítja:

- a két játékos egymás után felváltva tehessen golyókat a csövek valamelyikébe;
- idő előtt abbahagyhassuk a játékot (teszteléshez);
- beolvassa a kiválasztott cső sorszámát;
- az adott játékosnak megfelelő golyó mozgatását végrehajtó szubrutint hívja meg.

Ezt a programrészt a GOSUB TABLA rész alá, a GOLYO szubrutin elé írd meg!

```
REM **** FŐPROGRAM *****
    a = 1                          A játékos sorszámát állítjuk be. Az alprogramokban
                                   állítjuk a következőre (A=2).
    kezdes:                        Ezzel a címkével biztosítjuk, hogy az alprogramok-
                                   ból vissza tudjunk térni.
    DO                             A belső rész addig ismétlődik, amíg a Q betűt meg
                                   nem nyomjuk.
        DO
            x$ = INKEY$
        LOOP UNTIL x$ = "q" OR x$ = "Q" OR x$ >= "1" AND x$ <= "9"
         IF X$>="1" And X$<="9" THEN x = ...
                                   Pótold a hiányzó részt úgy, hogy X értéke a cső
                                   sorszáma legyen.
         IF a = 1 THEN GOSUB Elso ELSE GOSUB Masodik
                                   A feltétel eredményeként azokat a szubrutinokat
                                   hívjuk meg, amelyek az első, illetve a második játé-
                                   kos golyóját a megadott csőbe juttatják.
    LOOP UNTIL x$ = "q" or X$="Q"
END
```

A főprogramból az ELSO szubrutin meghívására, a képernyő bal szélén már meglevő golyók közül kell egyet a kapott sorszámú csőbe juttatni. Mindezt csak abban az esetben tehetjük meg, ha az adott csőben még van hely. Ha van hely, egy egyszerű értékadással kiválasztjuk a következő játékost. A rutint a program végére írd!

```
ELSO:
    IF mut(x) = 0 THEN BEEP: BEEP: BEEP: a = 2: GOTO kezdes
                                   Ha nincs hely a csőben, vissza az adatbeolvasásra.
    a = a + 1
    LOCATE 3, 10: PRINT b$; " következik!      "
                                   Elsőként a golyót jobbra, vízszintesen kell mozgatni
                                   a megadott cső „szájáig".
    SOR = 64                       A cső sorszámát az X változó tartalmazza. A sor
                                   értékét (a későbbi módosítási lehetőség miatt) kez-
                                   detben rögzített értéknek vesszük.
    FOR O = 20 TO x * 38 + 48 STEP 5
        PUT (O, SOR), K1           A golyó kirajzolása a megadott helyen.
        FOR W = 1 TO 100: NEXT W
        PUT (O, SOR), K1, XOR      A golyó eltüntetése a megadott helyen.
    NEXT O
    O = O - 2
```

A golyónk már az adott cső fölött van. Innen lefelé kell mozgatni a cső aljáig, vagy a már csőben levő golyó fölé. A játék megoldási elvénél már rögzítettük, hogy a csőben a következő üres helyet a MUT(x) mutatóval jelöljük.

```
FOR L = 64 TO mut(x) * 8 + 90 STEP 2
    PUT (O, L), K1
    FOR W = 1 TO 100: NEXT W
    PUT (O, L), K1, XOR
NEXT L
L = L – 2
PUT (O, L), K1
```

*Az előző ciklusban az utolsó helyre beérkezett golyót letöröltük, ezért újra kirajzoljuk. A helyére mozgatott golyó a játék ellenőrzéséhez kevés. A **T** tömb adott cellájának értékét meg kell változtatni (jelenleg nulla), hogy a játék ellenőrzését figyelhessük.*

```
T(x, Mut(x)) = a
```

A cella értéke vegye fel a játékos sorszámát (vagyis 1-et).

```
...
```

Erre a helyre fogjuk később beírni az ellenőrzést végző rutin hívását.

```
mut(x) = mut(x) - 1
```

Az adott cső mutatóját eggyel csökkentjük, mert erre a helyre most tettük be a golyót.

```
RETURN
```

Az eddigi program többszöri tesztelése miatt célszerű egy játékossal megvizsgálni a golyók csőbe kerülését. Emiatt e rutin elejére az **a = a + 1** sor elé írd be: **REM**!

A következő rutin elkészülte után töröld ki a REM-et!

A második játékos golyóját a képernyő jobb széléről kell a kiválasztott csőbe juttatni. A megoldás sokban hasonlít az első játékos rutinjára, így nagy részét te is meg tudod írni. A kipontozott részekre írd be a hiányzó utasításokat!

```
MASODIK:
    IF mut(x) = ...
    a = a - 1
    LOCATE 3, 10: PRINT...
    SOR = 64
    FOR O = 460 TO x * 38 + 48 STEP -5
        PUT...
        ...
        PUT...
    NEXT O
    O = O + 3
    FOR L = 64 TO mut(x) * 8 + 90 STEP 2
    PUT...
    ...
    NEXT L
    ...
```

```
    T(..., ...) = A
    L = L - 2
    ...                          Erre a helyre fogjuk később beírni az ellenőrzést
                                 végző rutin hívását.
    mut(x) =...
  RETURN
```

Az eddig elkészült program már csak a nyerő helyzeteket nem vizsgálja. Most ezt fogjuk megtenni. Az első rész elkészülte után az előző két rutinba, az üresen hagyott helyekre írd be: **GOSUB TESZT**

A sor vizsgálata

Elsőként a csőben egymás mellett elhelyezkedő golyókat vizsgáljuk. A **T** tömbben egymás mellett négy ugyanolyan számnak kell lennie. A számokat a **T(x, Mut(x)) = a** értékadással helyeztük a tömbbe. Az **X** változó a cső sorszámát, a **Mut(x)** pedig az adott csőben a következő üres hely sorszámát tartalmazza. Nem szükséges az egész tábla vizsgálata, mivel az utolsóként betett golyó esetében lehet csak nyerő helyzet.

Vizsgálni kell tehát az összes csőben (1-től 9-ig) a Mut(x) sorában levő értékeket. (Ennek a rutinnak a hívása azelőtt történik, mielőtt a Mut(x) értékét csökkentettük volna eggyel.) A Mut(x) sorában (1-től 9-ig) összeszámoljuk, hogy egymás mellett szerepel-e 4 (**a-val**) egyenlő érték . Ezt az **Ossz** változó segítségével teszszük meg.

```
TESZT:
    REM *** Sor ***
    Ossz = 0: s1 = 1
    DO
    IF T(s1, mut(x)) = a THEN ossz = ossz + 1 ELSE ossz = 0
                                 Ha a közvetlenül mellette levő érték nem ugyanaz,
                                 mint az előző, az Ossz értékét nullázzuk.
    IF ossz = 4 THEN GOTO Nyert  Ha elértük a 4-et, elirányítjuk a programot a Nyert
                                 címkére.
    s1 = s1 + 1
    LOOP UNTIL s1 > 9
    ...                          Innen folytatjuk az oszlop és a két átló vizsgálatával.
RETURN
```

Tekintettel arra, hogy tesztelnünk kell a vizsgálatot, a **Nyert** címkéjű részt is el kell készítenünk.

```
Nyert:
    REM *** Nyert ***
    IF a = 1 THEN LOCATE 10, 15: PRINT a$; " nyert!"
```

```
IF a = 2 THEN LOCATE 10, 15: PRINT B$; " nyert!"
```

Akármelyik játékos is nyert, neve kikerül a képernyőre. A nyertes jutalma egy kis zene. Az alábbi két sor csak ötleteket ad a program színesítésére.

```
PLAY "o3l4gl8cdefl4gccal8fgabo4l4co3l4"
PLAY "ccfl8gfedl4el8fedco2l4bo3l8cdecl16el2d "
LOCATE 6, 7: PRINT "Játszunk még egyet? (I/N) "
```

A játék folytatási lehetőségét is ebben az alprogramban végezzük el.

```
DO
    c$ = INKEY$
LOOP UNTIL c$ = "i" Or c$ = "n" Or c$="I" Or c$="N"
IF c$ = CHR$(105) THEN GOTO kezd
```

Ha a folytatást választjuk, az irányítást a ***Kezd*** *címkére helyezzük,*

```
IF c$ = CHR$(110) THEN END
```

az N betűre vége.

TESZTELD AZ EDDIG LEÍRTAKAT!

Az oszlop vizsgálata

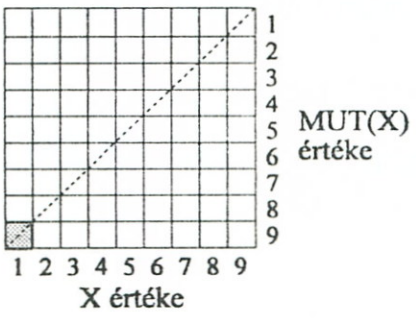
Az oszlop vizsgálatánál ugyanaz a teendőnk, mint a sor esetében. A TESZT rutinba a jelzett helyre írd be! A hiányzó részeket pótold!

```
REM *** Oszlop ***
Ossz = 0: ...
DO
    IF T(x,...) ...
    IF ossz = 4 THEN GOTO nyert
    S2 = S2 + 1
LOOP UNTIL...
```

Az átló vizsgálata 1.

Az átló vizsgálata két fő részből áll. Az első részben azokat az eseteket vizsgáljuk, amelyek balról jobbra haladva helyezkednek el átlós irányban (↗). Vizsgáljuk meg, hogy az egyes cellákba bekerülő golyók esetében milyen oszlop és sor kezdő- és végértékeket kell ellenőrizni a cső (**X**) és hely (**Mut(x)**) sorszámával.

Az alábbi táblázatban adott pontok helyét jelöld a rácson, s húzd meg az adott ponthoz tartozó átlót! (Az első értéket megrajzoltuk!) Pótold a táblázat hiányzó értékeit!



		Oszlopérték		Sorérték	
X	Mut(x)	Kezdő	Vég	Kezdő	Vég
1	9	1	9	9	1
1	8	1	8	8	1
1	7	1	7	7	1
3	1	1	3	3	1
6	7	4	9	9	4
8	6	5	9	9	5
6	3				
9	5				
		O1	O2	S1	S2

Nem kell vizsgálni a nyerő helyzetet, ha a golyó a bal felső, illetve a jobb alsó sarok bizonyos helyeire esik azaz, ha az **x + mut(x) <= 4** vagy az **x + mut(x) >= 16** reláció teljesül. (Próbáld meg állításunkat indokolni!)

Különbséget kell tenni azon golyók vizsgálata között is, amelyek az általunk berajzolt szaggatott vonal (főátló) fölött és rajta, valamint alatta helyezkednek el.

Főátlón vagy fölötte levő golyók helyzetének vizsgálata

A táblázat alsó sorában levő változókra teljesül:

Ha **x + mut(x) <= 10**, akkor **S1 = X + Mut(x) – 1** **S2 = 1**
O1 = S2 **O2 = S1**

Főátló alatt levő golyók helyzetének vizsgálata

A táblázat alsó sorában levő változókra teljesül:

Ha **x + mut(x) > 10**, akkor **S2 = X + Mut(x) – 9** **S1 = 9**
O1 = S2 **O2 = S1**

Ezek alapján készítjük el a programrészt, amelyet az oszlop vizsgálata alá írj be!

```
REM *** Átló1 ***
IF x + mut(x) <= 4 OR x + mut(x) >= 16 THEN RETURN
```

Ha a következő átlóvizsgálatot is elkészítetted, akkor a RETURN helyére írd be: ***GOTO LEATLO***

```
IF x + mut(x) <= 10 THEN
    S1 = x + mut(x) - 1
    S2 = 1
ELSE
    S1 = 9
    S2 = x + mut(x) - 9
END IF
O1 = S2: O2 = S1: Ossz = 0
DO
    IF T(O1, s1) = a THEN ossz = ossz + 1 ELSE ossz = 0
    IF ossz = 4 THEN GOTO nyert
    O1 = O1 + 1: s1 = s1 - 1
LOOP UNTIL O1 >= O2
```

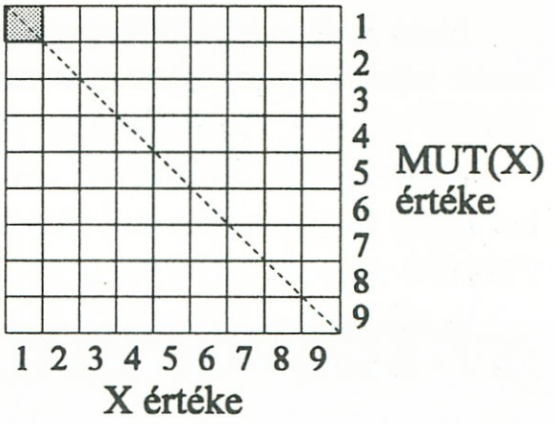
Futtasd a programot, s állíts elő olyan nyerő helyzetet, amely az előzőekben megírtaknak megfelel!

Az átló vizsgálata 2.

Most azokat az eseteket vizsgáljuk, amelyek balról jobbra haladva helyezkednek el a másik átlós irányban (↘). Vizsgáljuk meg most is a lehetséges nyerő szituációkat!

Nem kell vizsgálni a nyerő helyzetet, ha az **ABS(x – mut(x)) >= 6** feltétel teljesül! A többi esetben meg kell vizsgálnunk a **SZAM=ABS(x – mut(x))** értékeit. A következő részben a pontok helyére írd be a hiányzó adatokat!

X	Mut(x)	Oszlopérték Kezdő	Oszlopérték Vég	Sorérték Kezdő	Sorérték Vég
1	1	1	9	1	9
1	2	1	8	2	9
1	3	1	7	3	9
3	1	3	9	3	7
6	7	1	8	2	9
8	6	3	9	1	7
6	6				
9	5				
		O1	O2	S1	S2



Főátlón vagy fölötte levő golyók vizsgálata

A táblázat alsó sorában levő változókra teljesül:

Ha **X<= Mut(x)**, akkor **S1 =... S2 =...**
O1 =... O2 =...

Főátló alatt levő golyók helyzetének vizsgálata

A táblázat alsó sorában levő változókra teljesül:

Ha **X > Mut(x)**, akkor **S1 = ... S2 = ...**
O1 = ... O2 = ...

Az előző átlóvizsgálat alá írd be a következő programrészt!

```
REM *** Átló2 ***
Leatlo:
IF ABS(x - mut(x)) >= 6 THEN RETURN
SZAM = ABS(x - mut(x))
IF x <= mut(x) THEN
    S1 =.....: S2 =.....: O1 =.....: O2 =.....
ELSE
    S1 =.....: S2 =.....: O1 =.....: O2 =.....
END IF
Ossz = 0
DO
    IF T(O1, s1) = a THEN ossz = ossz + 1 ELSE ossz = 0
    IF ossz = 4 THEN GOTO nyert
    O1 = O1 + 1: s1 = s1 + 1
LOOP UNTIL O1 >= O2
```

A játékprogram elkészítésével végeztünk. Használatához jó szórakozást!

Vadászat

A vadászat alkalmával egy tisztáson elénk kerülő, s a tisztáson keresztül futó vadakat kell lelőni. A vadász a tisztás egyik szélén mozog, ugyanekkor a vadak bárhol előbukkanhatnak a tisztás szélén, s keresztülfutnak rajta. Készítsünk programot, amely szimulálja ezt a játékot. A lelőtt vadak természetesen nem futnak tovább!

A program elkészítéséhez az alábbi részfeladatokat kell átgondolni.

- Elő kell állítani egy puskás vadász rajzát, amelyet területmegjegyzéssel, a képernyő szélén mozgathatunk, s tetszőleges billentyűre lövéseket adhat le.
- Készíteni kell egy vadat ábrázoló rajzot, amelynek területét ugyanúgy meg kell jegyeztetni. Ezt az ábrát folyamatosan mozgatni kell a képernyőn!

Az alábbi program elkészítésekor csak az elvet közöljük, de az alakzatokat neked kell elkészíteni a program adott részeire! Az elv elkészítése után a program színesebbé tételére találsz módosító javaslatokat.

```
REM *** Vadászat ***
    SCREEN 12
REM ** VADÁSZ ***
    ...
```

*Ide készítsd el a vadász ábráját! A képernyő bal felső sarkában maximum 40×40-es területre. A megrajzolt alakzatot az **A** tömbben tároljuk. A **B** tömbbe az állat rajzát mutató alakzatot helyezzük.*
A megrajzolt alakzatokat eltüntetjük, majd a képernyő megadott helyén jelenítjük meg. Figyelj a koordinátákra!

```
    DIM a(400), b(400)
    GET (0, 0) – (40, 40), A
    PUT (0, 0), A, XOR
REM *** ÁLLAT ***
    ...
```

A vad ábráját itt kell elkészítened! Figyelj a méretre!

```
    GET (0, 0) - (11, 11), b
    PUT (0, 0), b, XOR
REM *** mozgat ***
    x = 320
```

Beállítjuk a vadász helyzetét. Mivel a képernyő alsó szélén fog mozogni, ezért nem szükséges a másik koordináta beállítása.

```
    Y = 10: P = 10
```

*Az **Y** és a **P** értékek a vad mozgásának kiinduló koordinátái.*

```
    PUT (x, 400), a
```

*Megjelenítjük a vadászt a képernyő 400-adik sorában. Ha neked olyan monitorod van, amely csak ennél kisebb értékeket tud kezelni, akkor a **SCREEN** után használd a **WINDOW** utasítást!*

```
agyu:
```

Egy címkét hozunk létre, hogy a programot később vissza lehessen állítani erre a programrészre.

```
    RANDOMIZE TIMER
```

A következőkben egy billentyűbeolvasó eljárással együtt fogjuk a kívánt mozgásokat megvalósítani.

```
    DO
```

A képernyőn a vad folyamatos mozgatását biztosítjuk azzal, hogy miközben a program egy billentyű megnyomására várakozik, a vad ábráját tartalmazó területet folyamatosan a képernyőre helyezzük úgy, hogy a terület oszlopértékét változtatjuk, miközben annak sorértéke változatlan marad. A folyamatos oszlopváltoztatás közben a vad kicsúszik a képernyőn. Ezért új sorértékének helyét véletlenszámgenerátorral állítjuk elő, miközben az oszlop értékét nullára állítjuk.

```
        x$ = INKEY$
        Y = Y +.3
        PUT (Y, P), b
        PUT (Y, P), b, XOR
        IF Y > 600 THEN Y = 0: P = INT(RND(1) * 150 + 1)
    LOOP UNTIL x$="4" OR x$="6" OR x$="v" OR x$="V" OR x$=CHR$(32)
    IF x$ = CHR$(32) THEN GOTO LOVES
    PUT (x, 400), a, XOR
```

A vadász mozgását a numerikus billentyűzet 4-es és hatos gombjával állítjuk, míg a lövéseket a SPACE billentyűvel végezzük. A programot a V betűvel állíthatjuk meg.

```
    IF x$ = "v" OR x$ = "V" THEN END
    IF x$ = "4" THEN x = x - 5: IF x < 10 THEN x = 10
    IF x$ = "6" THEN x = x + 5: IF x > 630 THEN x = 630
    SOUND (100),.2: SOUND (120),.01
```

A hanghatással a vadász mozgását jelezzük.

```
    PUT (x, 400), a
GOTO agyu
```

A lövéseket egy téglalap mozgatásával állítjuk elő. A lövést mindig a vadász adott helyzetéből indítjuk, ezért szerepel a koordináta megadásánál az X.

```
LOVES:
    FOR I = 396 TO 10 STEP -2
        LINE (x + 25, I) – (x + 26, I – 1), 14, BF
        FOR q = 1 TO 100: NEXT q
        LINE (x + 25, I) – (x + 26, I – 1), 0, BF
        Y = Y +.3
```

A lövés leadása közben a vad természetesen tovább mozog. Ezért szerepel itt is a vad területét megjelenítő, majd eltüntető rész.

```
        PUT (Y, P), b
```

Akkor történik találat, ha a golyó, illetve a vad találkozik. Ezt a golyó és a vad koordinátái különbségének abszolútértéke adja. A programban elég nagy

tűréshatár van beállítva. A feltételben szereplő értékeket csökkentve nehezebb, növelve könnyebb eltalálni a vadat.

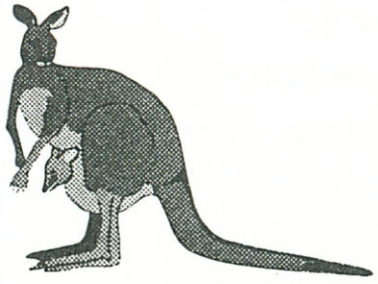
```
            IF ABS(x+25-Y)<=10 AND ABS(I-P)<=10 THEN GOTO talalt
            PUT (Y, P), b, XOR
            IF Y > 600 THEN Y = 0: P = INT(RND(1) * 150 + 1)
        NEXT I
        SOUND (40),.1: SOUND (800),.3: SOUND (40),.1
    GOTO agyu
REM *** TALALT ***
    talalt:
        SOUND (60), 1: SOUND (600), 1
        Y = 0: P = INT(RND(1) * 150 + 1)
```

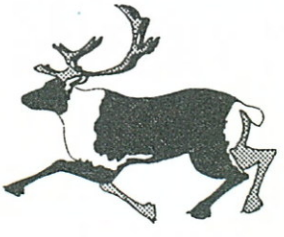
Találat esetén a vad „helyben marad", s az új vad megjelenik a képernyő szélén.

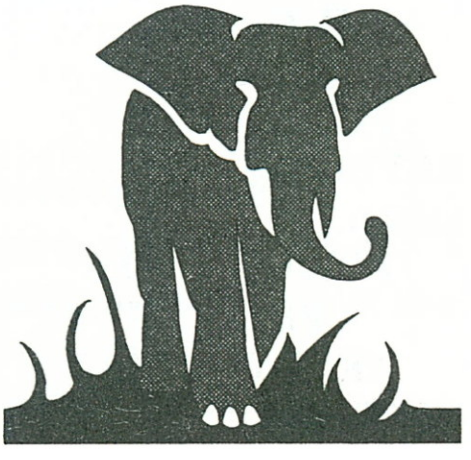
```
        GOTO agyu
```

Módosítsd a programot a következők szerint!

- Csak bizonyos számú tölténye legyen a vadásznak!
- Több vadász (több játékos) is szerepelhessen a játékban (esetleg más-más színnel jelölve). Hirdess versenyt köztük, s add meg a verseny végeredményét! *(A vadászoknak is hozz létre például egy **V(10)** tömböt, s ebbe tárold a pontszámukat. A végeredmény kihirdetése csak annyit jelentsen, hogy kiíratod a vadászok sorszámát, illetve az ahhoz a sorszámhoz tartozó pontszámot!)*
- Többféle vad jelenjen meg a tisztáson: *(A **DIM A(400)** utasítást módosítsd **DIM A(10,400)**-ra! Ekkor lehetőséged nyílik 10-féle állat megrajzolására, illetve a megrajzolt terület megjegyzésére a következők szerint: Az első vad képét az **A(1,400)**, a másodikét az **A(2,400)**, ..., a tizedikét az **A(10,400)** jelölje. Ekkor egy véletlenszám-generátorral váltogatni tudod a megjelenő állatokat. A programot is módosítsd!)*
- Módosítsd a programot úgy, hogy a vadak mozgása véletlenszerűen változzon aszerint, hogy a tisztás melyik oldaláról futnak át a másik oldalra!
- A megjelenő vadaknak bizonyos pontszám feleljen meg. Ha a vadász eltalálta a vadat, akkor pontszáma jelenjen meg a képernyőn, s adódjon az előző pontszámához!







PROGRAMOZÁSI TÉTELEK

A vadászatprogram utolsó kiegészítésénél felmerült az eredményhirdetés készítése is. A verseny győztesének kiválasztásához a maximális pontszámot elért vadász sorszámát kellett volna kiíratni. A verseny végeredményét pedig az elért pontszámok sorba rendezésével lehet megadni. A következő eljárások ezen probléma megoldására is használhatók, de a későbbi programozást is segítik. Mielőtt konkrétan rátérnénk a tételek megfogalmazására, készítsünk egy keretprogramot, amely véletlenszámokat helyez egy tömbbe, majd kiíratja azokat! Ezt a programrészt fogjuk használni mindegyik programozási tétel megvalósítása során.

```
n = 6: DIM A(n)          Az elemek számát 6-ra állítjuk, s egy ilyen elemszámú A tömböt hozunk létre.

CLS
GOSUB elemek             Meghívjuk az A tömböt véletlenszámokkal feltöltő rutint.

PRINT "   Az eredeti számsor:"
GOSUB sorszam            Meghívjuk azt a rutint, amely az előállított elemek fölé kiírja a sorszámukat!

GOSUB kiiras             Meghívjuk az elemeket kiíró rutint.
END
elemek:
    FOR K = 1 TO n       A cikluson belül adunk értékeket az A tömbnek. Pótold a hiányzó részt úgy, hogy a gép –100 és +100 közötti véletlen egész számokat állítson elő!

        RANDOMIZE TIMER
        A(K) = ...
    NEXT K
RETURN
sorszam:
    COLOR 0, 7
    FOR i = 1 TO n
        PRINT USING "###"; i;: PRINT ".";
    NEXT i
    COLOR 7, 0
    PRINT
RETURN
kiiras:
    FOR Q = 1 TO n
        PRINT USING "####"; A(Q);
    NEXT Q               Vedd fel a programot ALAP néven!
RETURN
```

MAXIMUMKIVÁLASZTÁS

Az előzőekben elkészített **ALAP** program N = 6 véletlenszámot állított elő.

Készítsünk programrészletet, amely kiválasztja a legnagyobb elemet, s megadja a talált elem tömbön belüli sorszámát is!

Kiindulásként legyen a maximális elem a számok közül az első: **MAX = A(1).** Vizsgáljuk meg a többi számot is! Indítsunk ciklust 2-től N-ig!

Ha az **A(i)**-edik elem nagyobb, mint a **MAX**, akkor a **MAX** értéke vegye fel az **A(i)** értékét!

A programrészt írd be az eddig megírt program után!

```
MAXIMUM:
    MAX = A(1): HELY=1
    FOR i=2 TO N
        IF A(i) > MAX THEN MAX = A(i): HELY=i
    NEXT i
    PRINT "A maximum helye: "; HELY ;" Értéke: ";MAX
RETURN
```

A főprogramba szúrd be a **GOSUB Kiiras** után **GOSUB Maximum**!
Ellenőrizd , helyesen működik-e a program!

Az előző kötetben tanultál a programok tervezéséről, leírásáról. Ott a *Blokkdiagramok* segítségével készíthetted el a program tervezetét. Ennek megvalósítása hosszadalmas, bonyolult. A játékprogramok készítésénél, illetve itt a programrész leírásánál már használtunk egyszerű szöveges leírásokat, amelyek a leendő program elvi megvalósítását, a program leendő részeit írták le.

Az előzőekben megvalósított maximumkiválasztás szövegszerű leírása a következő:

Eljárás : Maximum kiválasztás
Adott: N,A(n)
MAX := A(1); Hely := 1
Ciklus I := 2-től N-ig
Ha A(i) > MAX, akkor MAX := A(i); HELY := i
Ciklus vége
Ki: MAX, HELY
Eljárás vége.

A szövegszerű leírásnál a program által adatként kezelt értékeket az *Adott:*, a megjelenítendő eredményt a *Ki:* kezdetű sorokban, az értékadásokat a := jelet közrefogva adjuk meg. Több értékadást pontosvesszővel választunk el.

Az adott eljárások bármely programozási nyelvben megvalósíthatóak.

Feladatok

1. Módosítsd a programot 50 elemszámúra, az előállított véletlenszám-tartományt –1000 és +1000 közöttire! Ellenőrizd a program működését!
2. Készíts programrészt, amely az előzőleg előállított véletlenszámok közül a **MINIMUM**-értéket és annak helyét határozza meg!
3. Készítsd el a minimumkiválasztás szövegszerű leírását!

Eljárás : Minimum kiválasztás
.....
.....
.....
.....
.....
.....
Eljárás vége.

Módosítsuk az előző programot úgy, hogy a program a felhasználók által megadott értékek szerint dolgozzon!

A megírt – véletlenszámokat előállító – rutint kell csak módosítani!

```
elemek:
    FOR K = 1 TO N
        PRINT K;". elem =";:INPUT A(K)
    NEXT K
RETURN
```

(*Az N értékét állítsd át 5-re!)* Futtasd a programot!

Feladatok

4. Töltsd be a *Vadászat* programját, majd módosítsd a programot úgy, hogy az eredményhirdetésnél csak a győztes és az utolsó helyezett sorszáma és pontszáma jelenjen meg!
5. Módosítsd a programot úgy, hogy:
   – 6 nevet kérjen be adatként,
   – határozza meg a program a névsorban legelső és legutolsó tagot.

   *(Az adatok szövegváltozók legyenek a teljes programban! A **DIM A(n)** utasítást is javítsd át **DIM A$(n)**-re!)*
7. A program begépelése nélkül állapítsd meg, mit csinál!

```
DIM e(6)
RANDOMIZE TIMER
FOR i = 1 TO 100
    x = INT(RND(1) * 6 + 1)
    e(x) = e(x) + 1
```

```
NEXT i
FOR j = 1 TO 6
    PRINT j; ". eredmény ="; e(j)
NEXT j
```

MEGSZÁMLÁLÁS

Töltsd be az **ALAP**-programot, mert azt egészítjük ki új eljárással! Módosítsd az elemszámot 20-ra, a véletlenszámok előállítását a **–10 ... +10** tartományra!

Készítsünk programot, amely egy sorozat elemei között megszámolja a pozitív számokat!

Egy szám akkor pozitív, ha nagyobb, mint nulla. Az ilyen tulajdonságú elemeket kell megszámolni.

```
Szamol:
    Db=0
    For i=1 To n
        If A(i) > 0 Then Db=Db+1
    Next i
    PRINT "A pozitív elemek száma =";DB
RETURN
```

Szúrd be a főprogramba a **Gosub kiiras** után **Gosub Szamol,** majd ellenőrizd a program helyességét!

Az alábbiakban közöljük a megszámlálás szövegszerű leírását.

Eljárás: Megszámlálás
Adott: N, A(n), Tulajdonság
DB := 0
Ciklus I := 1-től N-ig
    Ha A(i) adott tulajdonságú , akkor DB := DB+1
Ciklus vége
Ki: DB
Eljárás vége.

Feladatok

8. Módosítsd a programot úgy, hogy a negatív elemeket is számlálja meg a program, de ne írj új szubrutint! (A programrészben egy új számlálót vezetünk be. A **DB=0** után írd be: **DN=0**. Ugyanebbe a ciklusba kell egy új feltételt írnunk az előző után, majd a negatív elemek számát is írasd ki!)
9. Készíts programrészletet, amely az előzőekben előállított véletlenszámok közül megszámlálja, hány esik a **–5 ... +5** tartományba!

10. Készíts programot, amely 100 db **1...1000** közé eső számot állít elő, majd megszámlálja, hány olyan szám van a sorozatban, amely osztható 3-mal!
11. Készíts programot, amely egy mondatban megszámolja az **a** betűket! Ha nincs a mondatban ilyen betű, akkor ezt is jelezze a gép!

KIVÁLOGATÁS

Az előzőekben meghatároztuk egy sorozat pozitív elemeinek számát, de nem tudtuk megmondani, melyek ezek az elemek. A feladat általános megoldása a következő:

__Eljárás:__ Kiválogatás
Adott: N, A(n), Tulajdonság
M := 0
Ciklus I := 1-től N-ig
Ha A(i) adott tulajdonságú, akkor M := M + 1;B(M)=A(i)
Ciklus vége
Ki: B(1..M)
Eljárás vége.

Az eljárásból látható, hogy egy **B** tömbre is szükségünk van, amelyet ugyanúgy dimenzionálni kell. Elemszáma legfeljebb ugyanannyi, mint az **A** tömbbé.

DIM A(N), B(N)

Készítsd el az algoritmus alapján a program részletet!

A program eredményének megjelenítése a **B** tömb elemeinek kiíratását jelenti ciklusban. A kiíratni kívánt elemek száma az **M** értéke.

```
kiiras2:
    PRINT "Negatív elemek:";
    FOR Q = 1 TO M
        PRINT USING "####"; B(Q);
    NEXT Q
RETURN
```

Feladatok

12. Készítsd el azt a programot, amely 50 db –100... +100 tartományba eső véletlenszámok közül kiválogatja, majd kiírja az összes negatív számot!
13. Egészítsd ki az előző programot úgy, hogy az előállított negatív számok közül kiválogassa az összes páros számot!
14. Készíts programot, amely egy adott szóból kiválogatja a magánhangzókat!
15. Készíts programot, amely bekéri egy osztály tanulóinak nevét és személyi számát, majd kiválogatja az osztályból a fiúkat és a lányokat!

16. Módosítsd az előző programot úgy, hogy a program az összes, májusban született tanuló nevét válogassa ki!
17. Készíts eljárást, amely a tantárgyi jegyeid alapján megállapítja, melyiknek a legrosszabb az átlaga!

Összegzés

Az előző könyvben már találkoztatok olyan programmal, amely bizonyos értékek közé eső számsorozatot összeadott. Ehhez például az **S = S + 1** értékadást használtuk, ami azt jelentette, hogy az **S** értéke legyen az **S** előző értékénél 1-gyel nagyobb.

Töltsd be az **ALAP**-programot, majd készíts olyan eljárást, amely összeadja a tömbben levő számokat!

```
osszegez:
    S = 0
    FOR I = 1 TO N
        S = S + A(i)
    NEXT I
RETURN
```

Eljárás: Összegzés
Adott: N, A(n)
S := 0
Ciklus I := 1-től N-ig
S := S + A(i)
Ciklus vége
Ki: S
Eljárás vége.

Az eljárással összeadtuk a tömb elemeit.

Készíts programrészt, amely kiírja az összeget, majd meghatározza a tömb elemeinek átlagát!

Feladatok

18. Készíts programot, amely 100 darab +1 és +6 közötti véletlen egész számot állít elő, de nem írja ki a képernyőre! Határozd meg az előállított elemek átlagát!
19. Készíts programot, amely egy 10 főből álló tekeverseny pillanatnyi állását írja a képernyőre! (A program folyamatosan bekéri a 10 versenyző által elvégzett dobás eredményét, majd egy teljes sorozat – mind a 10 fő dobott – után kiírja a képernyőre a verseny állását!)

20. Készíts programot, amely az iskolai kézilabda-bajnokság eddig lejátszott eredményei alapján (ismerjük, hogy melyik csapat melyikkel milyen eredményt játszott) kiírja a bajnokság állását. (Győzelemért 2, döntetlenért 1 pont jár.)
21. Egy horgászversenyen minden fogott halért 20 és még annyi pont jár, ahány dekagrammos a hal. Készíts programot, amely egy ilyen verseny pillanatnyi állását, s a verseny végeredményét készíti el!

22. Készíts programot, amely 100 db véletlen betűt állít elő az **A...Z** tartományban, majd kiválogatja, s megszámlálja a magánhangzókat!

ADATOK GRAFIKUS ÁBRÁZOLÁSA

Készíts programot, amely 20 db véletlen egész számot állít elő a –150... +150 tartományban, majd ezeket oszlopgrafikonon ábrázolja! Kiválogatja az összes 3-mal osztható számot, s ezen oszlopokat kifesti!

```
REM **** ábrázol ****
n = 20: DIM a(n): CLS
GOSUB elemek
GOSUB kiiras
GOSUB oszto
END
REM *** Főprogram vége ***
elemek:
    FOR K = 1 TO n
        RANDOMIZE TIMER
        A(K) = INT(RND(1) * 301 - 150)
    NEXT K
RETURN
kiiras:
```

A kiírást grafikus képernyőn végezzük el . Folyamatosan a képernyőre rajzoljuk az 5 egység széles és A(q)-ban tárolt hosszúságú téglalapokat.

```
    SCREEN 12
    WINDOW SCREEN (0, 0)-(300, 300)
    B = 0
    FOR q = 1 TO n
        B = B + 10
```

Tekintettel arra, hogy negatív hoszzúságok is előfordulhatnak, ezért ezeket a képernyő középvonalától lefelé, míg a pozitívokat felfelé rajzoljuk meg.

```
        LINE (B, 150 - a(q))-(B + 5, 150), 2, B
    NEXT q
RETURN
oszto:
    FOR i = 1 TO n
        IF a(i) MOD 3 = 0 THEN
```

Ha 3-mal osztható számot találunk, akkor az előző rutinban megrajzolt téglalapot színezzük be. A koordináták megállapítását az előző rutin alapján végezzük el.

```
            LINE (i * 10, 150 - a(i))-(i * 10 + 5, 150), 2, BF
```

```
            LOCATE i, 70: PRINT USING "###"; a(i)
```

A képernyőre kiíratjuk az adott tulajdonsággal rendelkező számokat olyan helyre , ahol nem szerepel téglalap.

```
        END IF
    NEXT i
RETURN
```

Futtasd a programot!

Feladatok

23. Módosítsd a mintaprogramot úgy, hogy az ábrázolt téglalapok közül a legnagyobbat, illetve a legkisebbet színezze ki!
24. Az előző programban cseréld fel a legnagyobb és a legkisebb elemet az A tömbben, illetve a grafikonon!

ADATOK RENDEZÉSE

Készíts programot, amely a grafikusan ábrázolt adatokat sorba rendezi, s bemutatja a rendezési elvet is!

```
n = 20: DIM a(n)
SCREEN 12
WINDOW SCREEN (0, 0)-(300, 300)
GOSUB elemek
GOSUB kiiras
DO
    x$ = INKEY$
LOOP UNTIL x$ = " "
GOSUB rendez
END
elemek:
    FOR K = 1 TO n
        RANDOMIZE TIMER
        A(K) = INT(RND(1) * 201 - 100)
    NEXT K
RETURN
kiiras:
    B = 0
    FOR q = 1 TO n
        B = B + 10
        LINE (B, 150 - a(q))-(B + 5, 150), 2, B
    NEXT q
RETURN
```

Az alábbi részprogram elkészítése után adunk magyarázatot működési elvére.

```
rendez:
    FOR i = n TO 2 STEP -1
        max = i
        FOR j = 1 TO i - 1
            IF a(max) < a(j) THEN max = j
        NEXT j
        GOSUB csere
    NEXT i
RETURN
```

A csereeljárás már ismert, így itt csak annak grafikus megvalósítása nehéz!

```
csere:
    LINE (i * 10, 150 - a(i))-(i * 10 + 5, 150), 0, B
    LINE (max * 10, 150 - a(max))-(max * 10 + 5, 150), 0, B
    SOUND (200), 1: SOUND (0), 2
    x = a(i): a(i) = a(max): a(max) = x
    LINE (i * 10, 150 - a(i))-(i * 10 + 5, 150), 2, B
    LINE (max * 10, 150 - a(max))-(max * 10 + 5, 150), 2, B
    SOUND (200), 1: SOUND (0), 2
RETURN
```

Ha túl gyorsan történik a csere végrehajtása, akkor ez utóbbi hang időtartamát növeld meg! Ez a teendőd 4 sorral lejjebb is.

A program alapján válaszolj a kérdésekre!

Mely elemeket válogatja ki a **Rendezés** programrészlet?

.....

Hogyan valósítja meg az elemek sorba rendezését?

.....

Maximumkiválasztásos rendezés

A mintaprogramban leírt eljárás tehát sorba rendezi a tömb elemeit. Ennek szövegszerű leírása a következő:

Eljárás: Maximum kiválasztásos rendezés
Adott: N,A(n)
Ciklus I := N-től 2-ig
MAX := I
Ciklus J := 1-től i – 1-ig
Ha A(Max) < A(J) akkor MAX := J
Ciklus vége
Csere (A(i) ; A(Max))
Ciklus vége
Eljárás vége.

A rendezett sorozatot, ha szükséges, ki kell íratni!

Az eljárás nyomon követésére gépeld be az alábbi mintaprogramot!

```
REM *** MAX KIV ***
    N = 5: DIM A(N)
    FOR K = 1 TO N
        RANDOMIZE TIMER
        A(K) = INT(RND(1) * 89 + 10)
    NEXT K
    CLS
    PRINT "Az eredeti számsor"
    FOR P = 1 TO N
        PRINT USING "###"; P;: PRINT ".";
    NEXT P
    PRINT, "I  J  MAX"
    PRINT: GOSUB kiiras
    PRINT: PRINT: PRINT "  A rendezett részsorozatok"
    REM **** Rendezés ****
    FOR I = N TO 2 STEP -1
        MAX = I
        FOR J = 1 TO I - 1
            IF A(MAX) < A(J) THEN MAX = J
            GOSUB kiiras
        NEXT J
        X = A(I): A(I) = A(MAX): A(MAX) = X
        PRINT "_____"
    NEXT I
    END                                    A rendezés vége.
    kiiras:
        FOR Q = 1 TO N
            PRINT USING "####"; A(Q);
        NEXT Q
        PRINT, I; J; MAX
    RETURN
```

A futtatás után megjelennek a változók adott pillanatban felvett értékei.

Feladatok

25. Hány összehasonlítást kell elvégezni

- 3 elem sorba rendezése esetén? .....
- 4 elem sorba rendezése esetén? .....
- 5 elem sorba rendezése esetén? .....
- 6 elem sorba rendezése esetén? .....

– 10 elem sorba rendezése esetén? .....
– N elem sorba rendezése esetén? .....

26. Az eljárás alapján, program nélkül rendezd a következő számsorozatot!

A(1)	A(2)	A(3)	A(4)	A(5)	I	J	MAX
18	17	16	15	14			

Buborékos rendezés

Adunk egy új rendezési eljárást. Az előző mintaprogramot írd át erre a rendezési eljárásra!

```
Eljárás: Buborékos rendezés
Adott: N, A(n)
i:=N
Ciklus amíg i > 1
  CS := 1
  Ciklus J := 1-től i-1 -ig
    Ha A(J)>A(J + 1) akkor Csere ( A(J); A(J+1)): Cs := J
  Ciklus vége
  i:=CS
Ciklus vége
Eljárás vége.
```

Az eljárás megírásakor a **DO WHILE** ciklust alkalmazd!

FELADATOK

27. Hány összehasonlítást kell a buborékos rendezésnél elvégezni:

– 3 elem sorba rendezése esetén? .....
– 4 elem sorba rendezése esetén? .....
– 5 elem sorba rendezése esetén? .....
– 6 elem sorba rendezése esetén? .....
– 10 elem sorba rendezése esetén? .....
– N elem sorba rendezése esetén? .....

28. A buborékos módszer segítségével rendezd az alábbi számsort! (Használd a táblázatot!)

A(1)	A(2)	A(3)	A(4)	A(5)	I	J	MAX
18	17	16	15	14			

Beillesztéses rendezés

Az alábbi programrészlet egy újabb rendezési elvet valósít meg. Módosítsd az előző programot erre az eljárásra, majd vizsgáld meg az összehasonlítások számát!

```
REM **** Beillesztéses rendezés ****
FOR i = 2 TO n
    j = i – 1: x = A(i)
    DO WHILE (j > 0 AND A(j) > x)
        A(j + 1) = A(j): j = j - 1
    LOOP
    A(j + 1) = x
NEXT i
GOSUB kiiras
```

Készítsd el a beillesztéses rendezés szövegszerű leírását!

Eljárás: Beillesztéses rendezés
Adot : N,A(n)
.....
.....
.....
.....
.....
.....
.....
Eljárás vége.

Feladatok

29. Az eljárást felhasználva rendezd a táblázatban adott számsort!

A(1)	A(2)	A(3)	A(4)	A(5)	I	J	X
8	–7	6	–5	4			

30. A tanult háromféle rendezési eljárást – programjaik felhasználásával – hasonlítsd össze a következők szerint:
Állítson elő a gép 50 db véletlen számot, majd mindegyik eljárásban számláltasd meg, hogy hány lépésben tudta az adott eljárás rendezni a sorozatot!
Melyik eljárás a leggyorsabb? .....

Névsorkészítés

Készítsünk programot, amely adott neveket sorba rendez!

A megoldás három részből áll:

- Az N értékének beolvasásából.
- Az N db név beolvasásából.
- A névsor készítéséből.
- Az adatok kiírásából.

```
REM *** NÉVSOR ***
CLS
INPUT "Hány nevet rendezünk?"; n
DIM Nev$(n)
```

Az N értékétől függően hozunk létre egy neveket tartalmazó tömböt. Mivel szöveget tartalmaz, ezért használjuk a $ jelet.

```
REM *** Adatbeolvasás ***
PRINT "Kérem a neveket!"
FOR i = 1 TO n
```

Ciklusban olvassuk be a tömb értékeit.

```
    PRINT "A(z) ";i; ". nevet kérem";: INPUT Nev$(i)
NEXT i
```

```
REM *** Rendezés ***                    A beillesztéses rendezéssel készítjük el a névsort.
FOR i = 2 TO n
    J = i – 1: X$ = Nev$(i)             Azonos típusú változókat tudunk csak összehasonlítani.
    DO WHILE J > 0 AND Nev$(J) > X$
        Nev$(J + 1) = Nev$(J): J = J – 1
    LOOP
    Nev$(J + 1) = X$
NEXT i
REM *** Kiírás ***
CLS
PRINT "A rendezett névsor:"             Az eljárást te készítsd el!
...
```

A program legnagyobb hibája, hogy mindig csak a beolvasott értékekkel dolgozik, s nem tárolja a bemenő adatokat.

Feladatok

31. Módosítsd a programot úgy, hogy az adatbeolvasás helyett **DATA**-sorokban tárolod a neveket, majd azokat töltsd be a **NEV$** tömbbe!

32. A névsort nyomtatóra is írathatod, ha a kiíratást végző eljárásban **LPRINT** utasítást használsz!

FÁJLKEZELÉS

A névsor készítése során felmerült az igény az adatok fájlba mentésére. A következőkben ezt valósítjuk meg a legegyszerűbb módon.
A fájl megnyitásához használjuk az **OPEN** parancsot.

OPEN "mód",fájl_szám,fájl_név

mód	értéke lehet:	
I	Input	*az adatok fájlból kiolvasására;*
O	Output	*az adatok fájlba írására;*
A	Append	*az eddig beírt adatokhoz hozzáfűzi a következőket.*
fájl_szám		*Értéke maximum 5 lehet.*
fájl_név		*Ilyen néven menti az adatokat.*

A megnyitott fájl bezárására a **CLOSE** parancsot használjuk.

CLOSE *Az összes megnyitott fájlt bezárja.*
CLOSE fájl_szám *Csak az adott számú fájlt zárja be.*

Adatok fájlba írására a:
PRINT #fájl_szám,adat utasítást használjuk. Az adatokat a fájlszámra hivatkozva írjuk a fájlba.

A fájlba írt adatokat kiolvasni az:
INPUT #fájlszám,adat utasítással lehet.

ADATOK MENTÉSE

A leírtak kipróbálásához töltsd be a *Névsorkészítés* programot, majd töröld a *Kiírás* részt a programból! Helyére írd a következőket:

```
CLS
INPUT "Milyen néven mentsem az adatokat?"; F$
```

*Az **F$**-ba megadott néven fogjuk elmenteni az adatokat. Add meg a fájl nevét és kiterjesztését is. (Például: **6a.dat**) Figyelj a névadás szabályaira!*

```
OPEN "a", #1, F$
```

A parancs hatására megnyitunk egy fájlt az előzőekben megadott néven adatok kiírására. (Az eddig kiírt adatokhoz hozzáfűzzük a soron következőket.)

```
FOR i = 1 TO n
    PRINT #1, Nev$(i)
```

*A **PRINT #1** hatására a **Nev$(i)** adatot az **F$** nevű fájlba írja.*

```
NEXT i
CLOSE 1
```

Végeztünk az adatok fájlba írásával, így bezárjuk a fájlt.

A fájlba írás során nem kerültek a képernyőre az adatok.

ADATOK OLVASÁSA FÁJLBÓL

Több névsor elkészítése után válogathatunk, hogy melyik névsort akarjuk a képernyőre vagy a nyomtatóra írni. Készítenünk kell egy új programot, amely az elmentett adatokat beolvassa az adott nevű fájlból.

```
REM *** Beolvasás ***
CLS: S := 0
INPUT "Milyen nevű fájlt nyissunk meg? "; F$
OPEN "I", #1, F$          A fájl nevének (6a.dat) megadása után a fájlt adatbeolvasásra nyitjuk meg.

kezd:
IF EOF(1) THEN GOTO zar   Az EOF(fájl_szám) függvény megvizsgálja, hogy van-e még adat a fájlban. Értéke lehet:
                          –1, ha nincs több adat, vagy üres a fájl.
                           0, ha van még olvasható adat.
INPUT #1, n$              N$-ba töltjük a soron következő adatot, majd
PRINT n$: S := S + 1      a képernyőre íratjuk a kapott értéket! A nyomtatóra írás esetében módosítsd LPRINT-re.
IF S = 20 THEN S = 0: GOSUB enter  Ha az adatok nem férnek el a képernyőn egymás alatt, akkor egy ENTER billentyű megnyomása után töröljük a képernyőt, majd folytatjuk az adatok kiírását.
GOTO kezd
zar:
    CLOSE 1
    GOSUB enter
END
enter:
    PRINT: PRINT "<ENTER>"
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = CHR$(13)
    CLS
RETURN
```

Feladatok

1. Az előzőekben elkészített program egy névsort ír fájlba. Módosítsd úgy, hogy az adatbeolvasó eljárásban újabb neveket fűzz a már meglévőkhöz, majd olvastasd be a neveket egy tömbbe! Készítsd el a bővített nevek listájának rendezését!
2. Készíts programot, amely egy menüből vezérelve adatokat ír ki, illetve hozzáfűz egy fájlhoz!

3. Módosítsd a 2. feladat menüjét úgy, hogy lehetőség legyen az adatok rendezésére is!
4. Töltsd be a *Vadászat* programját! Készíts a programhoz olyan programrészt, amely a játék mindenkori állását lemezen tárolja!
5. A grafikus programok elkészítésekor tömbökben tároltuk a megrajzolt képeket. Készíts programot, amely a képet tartalmazó tömböt egy képfájlba tölti, illetve onnan kiolvassa!
6. Készíts programot, amely egy család havi kiadásait *Bevétel, Élelmiszervásárlás, Ruházat, Lakásfenntartás, Törlesztés* pontokban tárolja! Az adatok folyamatosan bővíthetők legyenek! Hónap végén készítsen a program egy összegzést a havi kiadásokról!
7. Készíts programot, amely barátaid nevét és telefonszámát tárolja!

CÍMJEGYZÉK

Készítsünk programot, amely az ismerősök adatait tárolja!

```
REM *** Címjegyzék ******
20:
    CLS
    COLOR 0, 7
    PRINT "                    CÍMJEGYZÉK                    "
    COLOR 7, 0
    PRINT: PRINT: PRINT
    PRINT "     MENU"
    PRINT: PRINT
    PRINT "1. Adatbeírás (Név – Irányítószám – Város – Lakcím)"
    PRINT: PRINT
    PRINT "2. Adatok listázása képernyőre"
    PRINT: PRINT
    PRINT "V. Vége"
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = "1" OR a$ = "2" OR a$ = "v"
    IF a$ = "1" THEN CLS: GOTO file
    IF a$ = "2" THEN CLS: GOTO Olvas
    IF a$ = "v" THEN END
file:
    OPEN "a", #1, "nevsor"
    INPUT "Név: "; N$
    INPUT "Irányítószám: "; I$
    INPUT "Város: "; V$
```

```
    INPUT "Lakcím: "; C$
    PRINT #1, N$
    PRINT #1, I$
    PRINT #1, V$
    PRINT #1, C$
    CLOSE 1
    PRINT: PRINT:
    PRINT "Van még adat? I/N"
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = "i" OR a$ = "I" OR a$ = "n" OR a$ = "N"
    IF a$ = "i" OR a$ = "I" THEN PRINT: PRINT: GOTO file
    IF a$ = "n" OR a$ = "N" THEN GOTO 20
Olvas:
    S = 0: X = 1: OPEN "I", #1, "nevsor"
280:
    IF EOF(1) THEN 390
    INPUT #1, N$
    INPUT #1, I$
    INPUT #1, V$
    INPUT #1, C$
    PRINT X; ".Rekord": PRINT
    PRINT "       Név:"; N$
    PRINT "Irányítószám:"; I$
    PRINT "      Város:"; V$
    PRINT "    Lakcím:"; C$
    PRINT: S = S + 1: X = X + 1
    IF S = 3 THEN S = 0: GOSUB 1000
    GOTO 280
390:
    CLOSE 1
    GOSUB 1000
    GOTO 20
1000:
    PRINT: PRINT "<ENTER>"
    DO
        a$ = INKEY$
    LOOP UNTIL a$ = CHR$(13)
    CLS
RETURN
```

Feladatok

8. A mintaprogram nem tartalmazza a telefonszámok rögzítését. Módosítsd a programot!
9. Készíts programot, amely egy sakkverseny résztvevőinek rajtszámát, nevét, nemzetiségét tárolja, majd nyomtatóra írja a nevezett versenyzők névsorát a rajtszámokkal együtt!
10. Bővítsd ki a 2. feladatban elkészített programot úgy, hogy egy új állományban (a fájl_szám értéke 2 legyen) rögzíti a program a rajtszámot, illetve az elért pontszámot! Készíts eredményjelző táblát!

A NYOMTATÓ PROGRAMOZÁSA

Az előbbiekben már találtatok utalást a nyomtató használatára. A megfelelően megtervezett nyomtatási kép előállítása a nyomtatóhoz tartozó kézikönyv alapján végezhető el. Az alábbiakban az EPSON FX 850-es típusú nyomtató beállítási értékeinek felhasználását mutatjuk meg. Bármely más típusú nyomtatóhoz annak kézikönyve alapján módosíthatod a példaprogramokat.

Csatlakoztasd a számítógépedhez a nyomtatót, majd gépeld be az alábbi programot!

```
E$=CHR$(27)
LPRINT E$;"t";CHR$(1);
LPRINT E$;"M";"elit +1 "
LPRINT E$;"E";"elit félkövér";E$;"F"
LPRINT E$;"G";"elit két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"elit félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"-";CHR$(1);"elit aláhúzott";E$;"-";CHR$(0)
LPRINT E$;"-";CHR$(1);E$;"E";"elit félkövér – aláhúzott";E$;"-"; CHR$(0); E$; "F"
LPRINT E$;"w";CHR$(8);"elit nyújtott";E$;"w";CHR$(0)
LPRINT CHR$(15);"elit keskeny"
LPRINT E$;"E";"elit keskeny – félkövér";E$;"F"
LPRINT E$;"G";"elit keskeny – két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"elit keskeny – félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"-";CHR$(1);"elit keskeny – aláhúzott";E$;"-";CHR$(0);CHR$(18)
LPRINT CHR$(14);"elit széles egy sorra"
LPRINT E$;"W1";"ELIT széles"
LPRINT E$;"E";"elit széles – félköver";E$;"F"
LPRINT E$;"G";"elit széles – két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"elit széles – félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"-";CHR$(1);"elit széles – aláhúzott";E$;"-";CHR$(0)
LPRINT E$;"@"
```

A nyomtatód kézikönyve alapján nézz utána a következő utasításoknak!

s z é l e s í r á s	bekapcsolása	.....
széles írás	kikapcsolása	.....
keskeny írás	bekapcsolása	.....
keskeny írás	kikapcsolása	.....
félkövér írás	bekapcsolása	.....
félkövér írás	kikapcsolása	.....
dőlt betűs írás	bekapcsolása	.....
dőlt betűs írás	kikapcsolása	.....

Nyomtatás közben használhatunk további utasításokat is.

Stílusváltások

LPRINT CHR$(27);"x1"	*levélminőség bekapcsolása;*
LPRINT CHR$(27);"x0"	*levélminőség kikapcsolása;*
LPRINT CHR$(27);"4"	*dőlt betű bekapcsolása;*
LPRINT CHR$(27);"5"	*dőlt betű kikapcsolása.*

Sortávolság beállítása

LPRINT CHR$(27);"0"	
LPRINT CHR$(27);"1"	
LPRINT CHR$(27);"2"	
LPRINT CHR$(27);"3";CHR$(n)	*pontegységenként, ahol n = 3...255;*
LPRINT CHR$(27);"A";CHR$(n)	*n/72 inch-enként, ahol n = 3...255.*

A következő mintaprogram egy másik betűtípus beállítási lehetőségeit mutatja.

Mintaprogram

```
REM e_pica
E$=CHR$(27)
LPRINT E$;"t";CHR$(1);
LPRINT E$;"P";"pica"
LPRINT E$;"E";"pica félkövér";E$;"F"
LPRINT E$;"G";"pica keskeny – két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"pica félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"-";CHR$(1);"pica aláhúzott";E$;"-";CHR$(0)
LPRINT E$;"–";CHR$(1);E$;"E";"pica félkövér – aláhúzott";E$;"–";CHR$(0);E$;"F"
LPRINT E$;"w";CHR$(8);"pica nyújtott";E$;"w";CHR$(0)
LPRINT CHR$(15);"pica keskeny"
LPRINT E$;"E";"pica keskeny – félkövér";E$;"F"
LPRINT E$;"G";"pica keskeny – két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"pica keskeny – félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"pica keskeny – aláhúzott";E$;"–";CHR$(0);CHR$(18)
LPRINT CHR$(14);"pica széles egy sorra"
LPRINT E$;"W1";"PICA széles"
LPRINT E$;"E";"pica széles – félkövér";E$;"F"
LPRINT E$;"G";"pica széles – két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"pica széles – félkövér – két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"pica széles – aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"@"
```

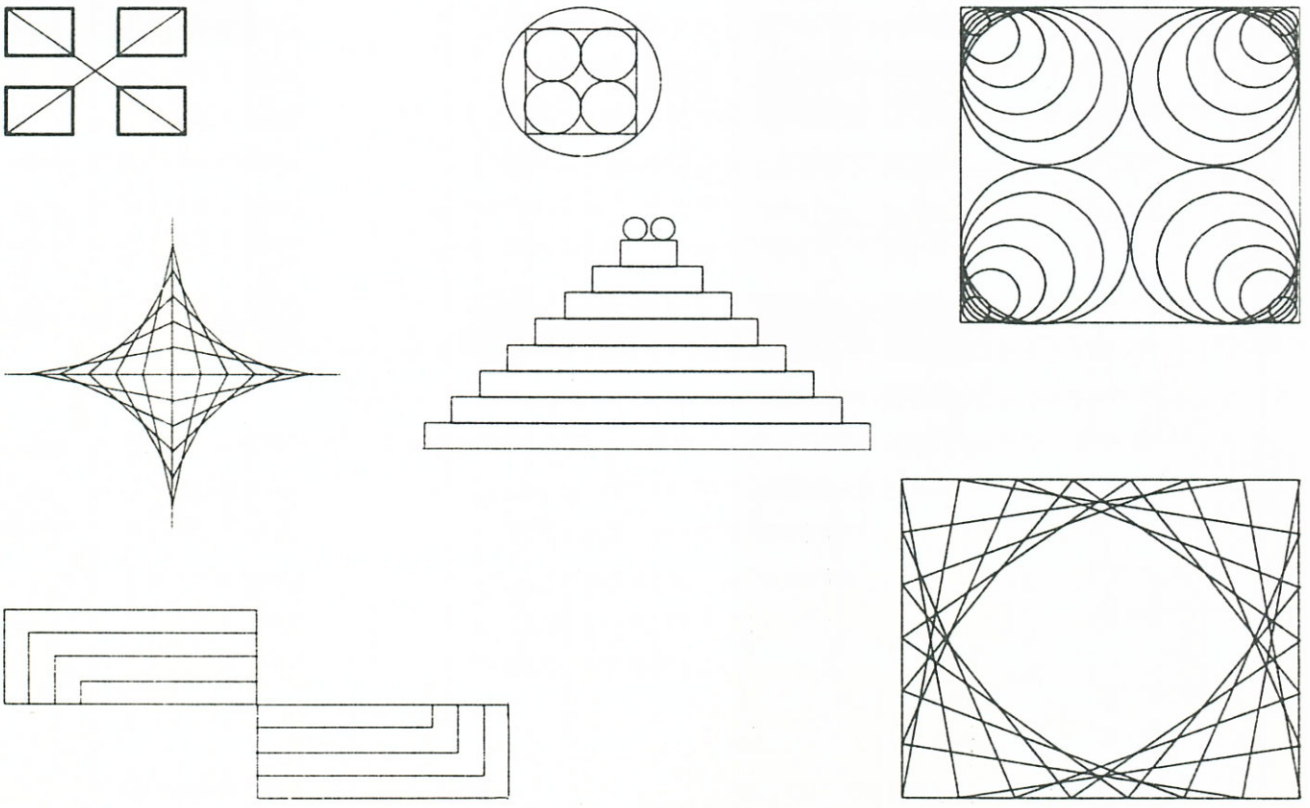
Mi a szerepe a programban a legutolsó sornak?

.....

A nyomtató programozásával kapcsolatban további mintaprogramokat találsz a Függelékben!

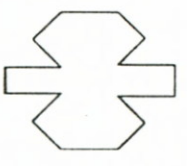
ÖSSZEFOGLALÓ FELADATOK

1. Minél kevesebb utasítással készíts programot, amely az alábbi rajzokat készíti el:



2. Készíts olyan programot, amely a következőket tudja:
   - a képernyő közepére rajzol egy 400×150-es téglalapot;
   - a SPACE billentyű megnyomására törli a képernyőt, majd középre rajzol egy olyan téglalapot, amely az előző téglalap oldalainak a fele;
   - ezt az eljárást addig folytatja, míg a téglalap rajzolásának van értelme.
3. Készíts olyan programot, amely a képernyő közepére tesz egy pontot és
   - a jobb felső sarokba mindig kiírja egy téglalapba a pont koordinátáit;
   - a képernyő legalsó sorába kiírja azon billentyűket, amelyek segítségével a pontot nyolc irányba mozgathatod;
   - választható legyen, hogy a mozgatás közben a rajzolás nyomot hagyjon-e, vagy sem.
4. Készíts programot, amely egy egyszerű dal kottáját a képernyőre rajzolja!
5. Készítsd el a következő programot:
   - a képernyőn rajzol egy 10×5-ös méretű üres téglalapot;
   - a téglalapot két billentyű segítségével nagyítja, illetve kicsinyíti;
   - egy billentyű leütésére mintával tölti ki a téglalapot;
   - a felhasználható billentyűk a képernyő alsó szélén legyenek láthatók!

6. Készítsd el az alábbi programot!
   - a képernyőn rajzol egy 10 egység sugarú körvonalat;
   - a kör két billentyű segítségével nagyítható, illetve kicsinyíthető legyen; (A képernyő méreteire ügyelj!)
   - egy billentyű leütésére mintával festhető legyen a kör; (Ugyanazt a billentyűt ismételten megnyomva a mintát törölje!)
   - a használható billentyűk menüben, a képernyő alsó szélén legyenek láthatók!
7. Rajzolj egy háromszöget a képernyőre, majd a koordináta-rendszer tengelyeit! Készíts programot, amely tükrözi a háromszöget az Y, majd az X tengelyre!
8. Készíts programot, amely szabályos sokszögekkel kiparkettázza a képernyőt!
9. Készíts programot, amely a képernyő közepére rajzol egy 50 egység széles, 100 egység magas téglatestet; (A vastagságát tetszőlegesen állítsd be úgy, hogy ez az él a meglévő élekkel 45 fokos szöget zárjon be!)
   a) Rajzold meg a nem látható éleit is!
   b) Készíts három különböző mintát, s azokkal tapétázd be a látható lapokat.
10. Készíts programot, amely:
    - megrajzolja az első feladatban rajzolt egyik ábrát;
    - megismétli a rajzolást, de háromnegyed akkora adatokkal;
    - még háromszor ismétli az eljárást!
11. Készíts programot, amely a DRAW parancs felhasználásával megrajzolja egy 40 egység élű kocka testhálóját, majd kiszámítja a felszínét, térfogatát, és elkészíti a kocka térbeli képét!
12. Készíts programot, amely a DRAW parancs segítségével szabályos 3...15 szög képét rajzolja a képernyőre. Az oldalak számát véletlenszám-generálással állítsd elő!
13. Készíts programot, amely véletlenszám-generátor segítségével előállítja egy sokszög 3...15 közötti oldalszámát, majd előállít ugyanennyi pontkoordinátát, s megrajzolja a sokszöget a képernyőre!
14. Készíts programot, amely a DRAW parancs segítségével a képernyőre rajzolja az alábbi ábrát:
    A képet három billentyű segítségével lehessen kicsinyíteni, nagyítani és festeni!



MINTAPROGRAMOK

A SZIGET

```
SCREEN 12
CLS
LOCATE 3, 30: INPUT "Kérem a neved! "; nev$
CLS
LOCATE 5, 30: PRINT nev$; " a szigeten"
LOCATE 7, 30: PRINT "Klasszikus játék új feldolgozásban."
LOCATE 9, 30: PRINT "Információt kérsz, vagy rögtön kezdjük a játékot (I/J)? ";
DO
    x$ = INKEY$
LOOP UNTIL x$ = "i" OR x$ = "I" OR x$ = "j" OR x$ = "J"
PRINT x$
IF x$ = "j" OR x$ = "J" THEN GOTO jatekele
CLS
LOCATE 3, 20: PRINT "Te, "; nev$; " egy szerencsétlen"
LOCATE 4, 20: PRINT "hajótörött vagy, az egyetlen túlélő,"
LOCATE 5, 20: PRINT "aki egy lakatlan szigeten raboskodik."
LOCATE 6, 20: PRINT "Rajtad múlik hogy túléled-e."
LOCATE 7, 20: PRINT "Fogy az élelmed, s a vized,"
LOCATE 8, 20: PRINT "de sebaj mert foghatsz halat,"
LOCATE 9, 20: PRINT "esős időben pedig gyűjthetsz vizet."
LOCATE 10, 20: PRINT "Palackpostát dobhatsz a vízbe,"
LOCATE 11, 20: PRINT ",s ha valaki megtalálja, m e g m e n e k ü l s z !"
LOCATE 12, 20: PRINT "Egy nap egy dolgot csinálhatsz,"
LOCATE 13, 20: PRINT "de sietned kell, mert egy hurrikán közeledik!"
LOCATE 14, 20: PRINT "SOK SZERENCSÉT!"
DO
    x$ = INKEY$
LOOP UNTIL x$ = " "
jatekele:
    hal = 15
    viz = 15
    nap = 1
    pihentseg = 5
    deszka = 0
jatek:
    CLS
    RANDOMIZE TIMER
    hajo = INT(RND * 7) + 1
    IF viz <= 0 THEN GOTO halal
```

```
    IF hal <= 0 THEN GOTO halal
    IF nap = 30 THEN GOTO hurrikan
    IF palack$ = " " AND hajo = 7 THEN GOTO hajo
    DEF fnf (x) = x * 3.14 / 180
    CIRCLE (450, 400), 80, 14, fnf(94), fnf(86), 2 / 5
    DRAW "bm 446,400 r8 u100 d100 l8 u100"
    LINE (446, 300)-(450, 250), 15: LINE -(455, 300), 15
    LINE (446, 300)-(450, 350), 15: LINE -(455, 300), 15
    LINE (446, 300)-(396, 310), 15
    LINE -(455, 300), 15: LINE (446, 300)-(496, 290), 15
    LINE -(455, 300), 15: LINE (446, 300)-(421, 275), 15
    LINE -(455, 300), 15: LINE (446, 300)-(471, 275), 15
    LINE -(455, 300), 15: LINE (446, 300)-(421, 325), 15
    LINE -(455, 300), 15: LINE (446, 300)-(471, 325), 15
    LINE -(455, 300), 15
    LOCATE 5, 20: PRINT "Mit csinálsz ma "; nev$; "?"
    LOCATE 8, 10: PRINT "1. HALAT (élelmet) FOGSZ"
    LOCATE 9, 10: PRINT "2. VIZET GYŰJTESZ"
    LOCATE 10, 10: PRINT "3. ÜZENETET DOBSZ"
    LOCATE 11, 10: PRINT "4. DESZKÁT GYŰJTESZ"
    LOCATE 12, 10: PRINT "5. PIHENSZ"
    IF pihentseg = 5 THEN LOCATE 15, 10: PRINT "Nagyon kipihent vagy!";
    IF pihentseg = 4 THEN LOCATE 15, 10: PRINT "Kipihent vagy!";
    IF pihentseg = 3 THEN LOCATE 15, 10: PRINT "Pihent vagy!";
    IF pihentseg = 2 THEN LOCATE 15, 10: PRINT "Fáradt vagy!";
    IF pihentseg = 1 THEN LOCATE 15, 10: PRINT "Halálosan kimerült vagy!";
    IF pihentseg = 0 THEN GOTO halal
eso:
    RANDOMIZE TIMER
    eso = INT(RND * 3) + 1: RANDOMIZE TIMER
    neso = INT(RND * 2) + 1
    IF eso = 3 AND neso = 2 THEN PRINT "Várhatóan nagy vihar lesz.";
    GOSUB villamfelho
    IF eso = 3 AND neso = 1 THEN GOTO eso
    IF eso = 2 THEN PRINT "Várhatóan eső lesz.";: GOSUB felho
    IF eso = 1 AND neso = 1 THEN PRINT "Felhőtlen az ég.";
    IF eso = 1 AND neso = 2 THEN GOTO eso
    LOCATE 16, 10: PRINT viz; " liter vized van, ami "; viz \ 3; " napig elég."
    LOCATE 17, 10: PRINT hal; " kg halad van, ami "; hal \ 3; " napig elég."
    LOCATE 18, 10: PRINT nap; " napja vagy a szigeten."
    DO
        x$ = INKEY$
    LOOP UNTIL x$ <= "5" AND x$ >= "1" OR x$ = "q" OR x$ = "Q"
    IF x$ = "2" THEN GOTO vizgyujtes
    IF x$ = "3" THEN GOTO uzenetdobas
```

```
    IF x$ = "4" THEN GOTO deszka
    IF x$ = "5" THEN GOTO pihenes
    IF x$ = "q" OR x$ = "Q" THEN END
halfog:
    CLS
    LINE (0, 200)-(650, 200), 15
    RANDOMIZE TIMER
    halhely = INT(RND * 5) + 1
    LOCATE 22, 20: PRINT "Nyomj meg egy számot 1 és 5 között!";
    halhely = halhely * 100
    DRAW "bm" + STR$(halhely) + ",200 c15"
    hali$ = "f10 g10 d10 f20 r30 e30 r20 f10 r5 u30 l5 g10 l20 h30 l30 g20 br10 c15 u5 r5 d5 l5 b l10 c15"
    horog$ = "l10 d5 r10"
    DRAW hali$
    DO
        h$ = INKEY$
    LOOP UNTIL h$ >= "1" AND h$ <= "5"
    h = VAL(h$)
    h = h * 100
    LINE (h, 0) – (h, 200)
    DRAW "bm" + STR$(h) + ",200 c15" + horog$
    IF halhely = h THEN GOTO megfogtad:  ELSE GOTO elrontottad
megfogtad:
    DRAW "bm" + STR$(halhely) + ",200 c0" + hali$
    DRAW "ta -90 c15" + hali$
    FOR q = 1 TO 1000: NEXT q
    CLS
    LINE (h, 200)-(h, 100), 0
    DRAW "ta -90 c15" + horog$
    DRAW "c15" + hali$
    FOR q = 1 TO 1000: NEXT q
    CLS
    LINE (h, 100)-(h, 0), 0
    DRAW "ta -90 c15" + horog$
    DRAW "c15" + hali$
vissza:
    RANDOMIZE TIMER
    nagysag = INT(RND * 5) + 1
    RANDOMIZE TIMER
    uj = INT(RND * 2) + 1
    IF nagysag = 1 AND uj = 1 OR nagysag = 5 AND uj = 1 THEN GOTO vissza
    nagysag = nagysag * 3
    LINE (0, 200)-(650, 200), 15
    LOCATE 20, 10
    IF nagysag = 3 THEN
```

```
      PRINT "Ez a 3 kg-os hal nem túl nagy, csak 1 napig elég. Próbáld meg holnap! "
      hal = hal + 3: GOTO vege
    ENDIF
    IF nagysag = 6 THEN PRINT "Ez a hal 6 kg-os, 2 napig elég. Szép fogás!"
      hal = hal + 6: GOTO vege
    ENDIF
    IF nagysag = 9 THEN PRINT "Ez a hal 9 kg-os; 3 napig elég. Szép fogás!"
      hal = hal + 9: GOTO vege
    ENDIF
    IF nagysag = 12 THEN PRINT "Ez a hal 12 kg-os; 4 napig elég. Szép nagy fogás! "
      hal = hal + 12: GOTO vege
    ENDIF
    IF nagysag = 15 THEN
      PRINT "Ez igen, a hal 15 kg-os, 5 napig elég. GRATULÁLOK!"
      hal = hal + 15: GOTO vege
    ENDIF
elrontottad:
    LOCATE 2, 20: PRINT "Hát ez bizony mellé ment! Próbáld meg legközelebb!"
vege:
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = " "
vege1:
    nap = nap + 1
    viz = viz - 3
    hal = hal - 3
    pihentseg = pihentseg - 1
    GOTO jatek
vizgyujtes:
    CLS
    DRAW "bm 265,135"
    DRAW "c15 r5 d181 r100 u181 r5"
    IF eso = 1 THEN GOTO oda
    RANDOMIZE TIMER
    IF eso = 2 THEN esoviz = INT(RND * 3) + 2
    RANDOMIZE TIMER
    IF eso = 3 THEN esoviz = INT(RND * 5) + 4
    IF esoviz > 5 THEN esoviz = 5
    esoviz = esoviz * 3
    FOR i = 10 TO esoviz * 10 STEP 10
        LINE (270, 315 - i)-(370, 315 - i), 15
        PAINT (271, 315 - i + 1), 15, 15
        FOR q = 1 TO 200: NEXT q
    NEXT i
    LOCATE 5, 20: PRINT "Gratulálok! "; esoviz; " liter vizet sikerült gyűjtened,";
```

```
    PRINT "ami "; esoviz / 3; " napig elég": viz = viz + esoviz: GOTO vege
oda:
    LOCATE 5, 20: PRINT "Sajnos felhőtlen volt az ég, nem sikerült vizet gyűjtened"
    GOTO vege
uzenetdobas:
    CLS
    uveg$ = "d50 r170 e20 d5 r5 u20 l5 d5 h20 l170"
    szoveg$ = "d30 r130 u30 l130"
    LOCATE 5, 25: PRINT "S O S"
    LOCATE 5, 10: PRINT "1."
    LOCATE 13, 25: PRINT "Segítség!"
    LOCATE 13, 10: PRINT "2."
    LOCATE 21, 25: PRINT "Mentsetek meg!"
    LOCATE 21, 10: PRINT "3."
    DRAW "bm 140,45" + uveg$
    DRAW "bm 170,55" + szoveg$
    DRAW "bm 140,175" + uveg$
    DRAW "bm 170,185" + szoveg$
    DRAW "bm 140,300" + uveg$
    DRAW "bm 170,310" + szoveg$
    LOCATE 2, 30: PRINT "Válassz egy számot!";
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = "1" OR x$ = "2" OR x$ = "3"
    PRINT x$
    LOCATE 16, 30: PRINT "Üzenetedet talán megtalálja valaki!" palack$ = " "
    GOTO vege
deszka:
    CLS
    RANDOMIZE TIMER
    capak = INT(RND * 20) + 5
    FOR i = 25 TO capak + 25
        LOCATE 25, i: PRINT "A "
    NEXT i
    LINE (0, 200)-(650, 200), 15
    LOCATE 15, 40: PRINT "-"
    LOCATE 5, 20: PRINT "Biztosan bemész, "; capak; " cápa van a vízben? (I/N) ";
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = "i" OR x$ = "n" OR x$ = "I" OR x$ = "N"
    PRINT x$
    IF x$ = "n" OR x$ = "N" THEN LOCATE 7, 20
      PRINT "Bölcs döntés volt."
      GOTO vege
    ENDIF
```

```
    IF capak < 15 THEN GOTO OOO
    RANDOMIZE TIMER
    megegy = INT(RND * 3) + 1
    IF megegy = 3 THEN GOTO OOO
    capahalal$ = " ": GOTO halal
OOO:
    LOCATE 15, 40: PRINT " "
    LOCATE 8, 20: PRINT "SIKERÜLT...";
    tutaj = tutaj + 1
    PRINT tutaj; " deszkád van. Még "; 8 - tutaj; " darab kell!"
    IF tutaj = 8 THEN GOTO tutaj
    GOTO vege
pihenes:
    CLS
    LOCATE 3, 30: PRINT "Bizony rád fér egy kis pihenés!"
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = " "
    CIRCLE (320, 240), 300, 14,,, 2 / 5
    LOCATE 10, 35: PRINT "Z...Z... Z... "
    RANDOMIZE TIMER
    alom = INT(RND * 3) + 1
    IF alom = 2 THEN GOTO alom2
    IF alom = 3 THEN GOTO alom3
alom1:
    CIRCLE (320, 240), 300, 14,,, 2 / 5
    LOCATE 11, 30: PRINT "Gyönyörűt álmodtál...!"
    LOCATE 12, 30: PRINT "TE nyerted meg a lottó "
    LOCATE 13, 30: PRINT "teljes nyereményét e g y e d ü l !"
    LOCATE 14, 30: PRINT "Bizony ez szép álom volt!"
    LOCATE 15, 30: PRINT "Ha van fantáziád, még "
    LOCATE 16, 30: PRINT "fokozni is tudod az örömöt."
    LOCATE 17, 30: PRINT "Kipihenten fogsz ébredni!"
    pihentseg = 6
    GOTO vege
alom2:
    CIRCLE (320, 240), 300, 14,,, 2 / 5
    LOCATE 11, 30: PRINT "Gyönyörűt álmodtál...!"
    LOCATE 12, 30: PRINT "A világ legerősebb sportolójává"
    LOCATE 13, 30: PRINT "kiáltottak ki az eredményeid"
    LOCATE 14, 30: PRINT "alapján. Gondolj bele! Dől"
    LOCATE 15, 30: PRINT "a pénz... az elismerés... "
    LOCATE 16, 30: PRINT "Kipihenten fogsz ébredni!"
    pihentseg = 6
    GOTO vege
```

```
alom3:
    LOCATE 11, 30: PRINT "BORZASZTÓT ÁLMODTÁL!!!!"
    LOCATE 12, 30: PRINT "Egy nap, mikor a múzeumban"
    LOCATE 13, 30: PRINT "a dinoszaurusz csontvázát"
    LOCATE 14, 30: PRINT "nézegetted, az életre kelt és"
    LOCATE 15, 30: PRINT "feléd fordult, s ... biztos tudod,"
    LOCATE 16, 30: PRINT "mi történt. Kimerülten fogsz ébredni!"
    LOCATE 17, 30: PRINT "Alszol még egyet? (a nap nem megy)"
    LOCATE 18, 30: PRINT "(I/N)"
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = "i" OR x$ = "n" OR x$ = "I" OR x$ = "N" PRINT x$
    IF x$ = "n" OR x$ = "N" THEN pihentseg = 2: GOTO vege
    CLS
    RANDOMIZE TIMER
    alomok = INT(RND * 2) + 1
    IF alomok = 1 THEN GOTO alom1
    IF alomok = 2 THEN GOTO alom2
hurrikan:
    n = 10
    FOR i = 440 TO 100 STEP -10
    CIRCLE (320, i), n, 15,,, 2 / 5
    n = n + 2
    NEXT i
    LOCATE 3, 30: PRINT "H U R R I K Á N"
    LOCATE 4, 35: PRINT "A hurrikán sajnos"
    LOCATE 5, 35: PRINT "megérkezett, de "
    LOCATE 6, 35: PRINT "választhatsz következő"
    LOCATE 7, 35: PRINT "játékot is (J/V)? ";
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = "v" OR x$ = "V" OR x$ = "J" OR x$ = "j"
    PRINT x$
    IF x$ = "v" OR x$ = "V" THEN END:  ELSE GOTO jatekele
halal:
    CLS
    DRAW "bm 290,270 r20 e20 u20 h20 u10 e30"
    DRAW "u30 h30 l20 g30 d30 f30 d10 g20 d20 f20"
    CIRCLE (280, 150), 5, 15
    CIRCLE (320, 150), 5, 15
    CIRCLE (300, 240), 10, 15
    IF viz <= 0 THEN LOCATE 25, 20: PRINT "Szomjhalál.";: GOTO ujatek
    IF hal <= 0 THEN LOCATE 25, 20: PRINT "Éhhalál.";: GOTO ujatek
    IF pihentseg = 0 THEN LOCATE 25, 20: PRINT "Halálos kimerültség.";: GOTO ujatek
    IF capahalal$ = " " THEN LOCATE 25, 20: PRINT "Cápák öltek meg.";: GOTO ujatek
```

```
ujatek:
    PRINT "Újra kezded? (I/N) ";
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = "i" OR x$ = "I" OR x$ = "n" OR x$ = "N"
    PRINT x$
    IF x$ = "n" OR x$ = "N" THEN END:  ELSE GOTO jatekele
tutaj:
    CLS
    LOCATE 3, 20: PRINT "SIKERÜLT! ÉLJEN! Kész a tutaj! Megmenekültél!"
    DRAW "bm 100,300 R100 u10 l100 d10"
    DRAW "r20 u10 r20 d10 r20 u10 r20 d10"
    DRAW "l20 u10 u60 r60 l140 u80 r140 d80"
    DRAW "l80 d60 "
    PLAY ON
    PLAY "O3 L8 C L12 C C L8 C L12 E C"
    PLAY "L8 G L12 G G L8 G O4 L12 C O3 L12 G"
    PLAY "L8 E L12 G E L8 C L12 E C"
    PLAY "O2 L2 G "
    PLAY "O3 L8 C L12 C C L8 C L12 E C"
    PLAY "L8 G L12 G G L8 G P8"
    PLAY "L8 G L12 E C L8 E G"
    PLAY "L8 C L12 C C L8 C P8"
    LOCATE 15, 35
    GOTO ujatek
felho:
    CIRCLE (510, 100), 50, 15,,, 2 / 5
    PAINT (510, 100), 15, 15
RETURN
villamfelho:
    CIRCLE (510, 100), 50, 15,,, 2 / 5
    PAINT (510, 100), 15, 15
    LINE (508, 130)-(500, 150), 14
    LINE -(530, 145), 14
    LINE -(520, 160), 14
    LINE -(515, 155),14
    LINE -(530, 185), 14
    LINE -(545, 155), 14
    LINE -(535, 155), 14
    LINE -(550, 135), 14
    LINE -(510, 140), 14: LINE -(515, 130)
RETURN
hajo:
    CLS
    LOCATE 15, 30: PRINT "Üzenetedet megtalálta valaki!"
```

```
    LOCATE 16, 35: PRINT "De vajon ki?"
    DO
        x$ = INKEY$
    LOOP UNTIL x$ = " "
    CLS
    DRAW "bm 200,400"
    DRAW "r200 e150 l500 f150"
    DRAW "u30 r230 l30 d30 l100 u30 l130"
    DRAW "h30 r320 l30 d30 l100 u30 l100 d30 l60"
    DRAW "h60 r30 d30 u30 r100 d30 u30 r100 d30 u30 r100 d30 u30 r50"
    DRAW "e30 l30 d30 u30 l100 d30 u30 l100 d30 u30 l100 d30 u30 l110"
    DRAW "h30 r30 d30 u30 r100 d30 u30 r100 d30 u30 r100 d30 u30 r100 d30 u30"
    DRAW "r20 u40 r20 l50 u80 r50 d80 l30 d40"
    DRAW "l175 u40 r50 u160 l120 d160 r70 l20 d40"
    DRAW "l200 r20 u40 r20 l50 u80 r50 d80 l30 d40"
    RANDOMIZE TIMER
    hajok = INT(RND * 3) + 1
    IF hajok = 1 THEN LOCATE 5, 35: PRINT "KOSSUTH": GOTO oooda
    IF hajok = 2 THEN LOCATE 5, 36: PRINT "PETŐFI": GOTO oooda
    LOCATE 27, 10: PRINT "Hát ezek kalózok! Úgy látszik, nem fogsz sokáig élni!"
    DRAW "bm 295,190 r20 e20 u10 h20 u10 e25"
    DRAW "u10 h25 l20 g25 d10 f25 d10 g20 d10 f20"
    CIRCLE (305, 165), 10, 15
    CIRCLE (290, 100), 7, 15
    CIRCLE (320, 100), 7, 15
    LOCATE 5, 10
    GOTO ujatek
oooda:
    LOCATE 27, 35: PRINT "Megtaláltak!"
    LOCATE 28, 30: PRINT "M E G M E N E K Ü L T É L!!"
    LOCATE 5, 10
    PLAY ON
    PLAY "O3 L8 C L12 C C L8 C L12 E C"
    PLAY "L8 G L12 G G L8 G O4 L12 C O3 L12 G"
    PLAY "L8 E L12 G E L8 C L12 E C"
    PLAY "O2 L2 G "
    PLAY "O3 L8 C L12 C C L8 C L12 E C"
    PLAY "L8 G L12 G G L8 G P8"
    PLAY "L8 G L12 E C L8 E G"
    PLAY "L8 C L12 C C L8 C P8"
Goto Ujjatek
```

A TORUS

Az alábbi program a BASIC-telepítő példaprogramjai között található meg. Begépelését azoknak ajánlom, akik látványos grafikát szeretnének megnézni, de nem rendelkeznek ezzel a programmal. Javaslom azonban mindenkinek az áttanulmányozását, mert érdekes megoldásokat rejt magában a grafikus lehetőségek kihasználására.

```
'============================================================
'TORUS
'============================================================
DEFINT A-Z
DECLARE SUB GetConfig ()
DECLARE SUB SetPalette ()
DECLARE SUB TorusDefine ()
DECLARE SUB TorusCalc (T() AS ANY)
DECLARE SUB TorusColor (T() AS ANY)
DECLARE SUB TorusSort (Low, High)
DECLARE SUB TorusDraw (T() AS ANY, Index())
DECLARE SUB TileDraw (T AS ANY)
DECLARE SUB TorusRotate (First)
DECLARE SUB Delay (Seconds!)
DECLARE SUB CountTiles (T1, T2)
DECLARE SUB Message (Text$)
DECLARE SUB SetConfig (mode)
DECLARE FUNCTION Inside (T AS ANY)
DECLARE FUNCTION DegToRad! (Degrees)
DECLARE FUNCTION Rotated (Lower, Upper, Current, Inc)

CONST PI = 3.14159
CONST TRUE = -1, FALSE = 0
CONST BACK = 0
CONST TROW = 24, TCOL = 60
CONST RNDM = -1
CONST START = 0
CONST CONTINUE = 1
CONST VGA = 12
CONST MCGA = 13
CONST EGA256 = 9
CONST EGA64 = 8
CONST MONO = 10
CONST HERC = 3
CONST CGA = 1
```

```
TYPE Tile
    x1      AS SINGLE
    x2      AS SINGLE
    x3      AS SINGLE
    x4      AS SINGLE
    y1      AS SINGLE
    y2      AS SINGLE
    y3      AS SINGLE
    y4      AS SINGLE
    z1      AS SINGLE
    xc      AS SINGLE
    yc      AS SINGLE
    Tcolor  AS INTEGER
END TYPE

TYPE Config
    Scrn    AS INTEGER
    Colors  AS INTEGER
    Atribs  AS INTEGER
    Xpix    AS INTEGER
    Ypix    AS INTEGER
    TCOL    AS INTEGER
    TROW    AS INTEGER
END TYPE

DIM VC AS Config

TYPE TORUS
    panel       AS INTEGER
    Sect        AS INTEGER
    Thick       AS SINGLE
    Xdegree     AS INTEGER
    Ydegree     AS INTEGER
    Bord        AS STRING * 3
    Delay       AS SINGLE
END TYPE

DIM TOR AS TORUS, Max AS INTEGER

DIM Pal(0 TO 300) AS LONG

DIM InitRows AS INTEGER, BestMode AS INTEGER, Available AS STRING

TOR.Thick = 3: TOR.Bord = "YES"
TOR.panel = 8: TOR.Sect = 14
TOR.XDegree = 60: TOR.YDegree = 165
GetConfig
VC.Scrn = BestMode
```

```
DO TorusDefine
    DO
        Tmp = TOR.panel
        Max = TOR.panel * TOR.Sect
        REDIM Index(0 TO Max - 1) AS INTEGER
        ON ERROR GOTO MemErr
        REDIM T(0 TO Max - 1) AS Tile
        ON ERROR GOTO 0
    LOOP UNTIL Tmp = TOR.panel
    FOR Til = 0 TO Max - 1
        Index(Til) = Til
    NEXT
    Message "Számolás"
    TorusCalc T()
    TorusColor T()
    Message "Rendezés"
    TorusSort 0, Max - 1
    SCREEN VC.Scrn
    SetPalette
    WINDOW (-(TOR.Thick + 1), -(TOR.Thick+1))-(TOR.Thick+1, TOR.Thick +1)
    TorusDraw T(), Index()
    DO WHILE INKEY$ = ""
        Delay (TOR.Delay)
        TorusRotate CONTINUE
    LOOP
    SCREEN 0
    WIDTH 80
LOOP
WIDTH 80, InitRows
END
VideoErr:
    SELECT CASE BestMode
      CASE VGA
        BestMode = MCGA
        Available = "12BD"
      CASE MCGA
        BestMode = EGA256
        Available = "12789"
      CASE EGA256
        BestMode = CGA
        Available = "12"
      CASE CGA
        BestMode = MONO
        Available = "A"
```

```
        CASE MONO
            BestMode = HERC
            Available = "3"
        CASE ELSE
            PRINT "Sajnálom. A grafika nem használható. A TORUS nem futtatható!"
            END
    END SELECT
    RESUME
EGAErr:
    BestMode = EGA64
    Available = "12789"
    RESUME NEXT
MemErr:
    LOCATE 22, 1
    PRINT "Nincs elég memória"
    PRINT "Cserélje a videokártyát "; TOR.panel;"-ról";  TOR.panel - 1;"-ra!"
    PRINT "Csökkentse a szekciók számát"; TOR.Sect; "-ról"; TOR.Sect - 1;"-re!'
    DO WHILE INKEY$ = "": LOOP
    TOR.panel = TOR.panel - 1
    TOR.Sect = TOR.Sect - 1
    RESUME NEXT
RowErr:
    IF InitRows = 50 THEN
        InitRows = 43
        RESUME
    ELSE
        InitRows = 25
        RESUME NEXT
    END IF
SUB CountTiles (T1, T2) STATIC
    LOCATE TROW - 1, TCOL: PRINT SPACE$(19);
    IF T1 > 0 AND T2 > 0 THEN
        LOCATE TROW - 1, TCOL
        PRINT "Mozaik";
        PRINT USING " ###"; T1;
        PRINT USING " ###"; T2;
    END IF
END SUB
FUNCTION DegToRad! (Degrees) STATIC
    DegToRad! = (Degrees * 2 * PI) / 360
END FUNCTION
SUB Delay (Seconds!) STATIC
    Begin! = TIMER
    DO UNTIL (TIMER - Begin! > Seconds!) OR (TIMER - Begin! < 0)
    LOOP
```

```
END SUB
SUB GetConfig STATIC
    SHARED InitRows AS INTEGER, BestMode AS INTEGER, Available AS STRING
    InitRows = 50
    ON ERROR GOTO RowErr
    LOCATE InitRows, 1
    BestMode = VGA
    Available = "12789BCD"
    ON ERROR GOTO VideoErr
    SCREEN BestMode
    ON ERROR GOTO EGAErr
    IF BestMode = EGA256 THEN SCREEN 8,, 1
    ON ERROR GOTO 0
    SCREEN 0,, 0
    WIDTH 80, 25
END SUB
FUNCTION Inside (T AS Tile) STATIC
    SHARED VC AS Config
    DIM Highest AS SINGLE, Lowest AS SINGLE
    Border = VC.Atribs - 1
    T.xc = T.x2 + ((T.x3 + (T.x4 - T.x3) / 2 - T.x2) / 2)
    T.yc = T.y2 + ((T.y3 + (T.y4 - T.y3) / 2 - T.y2) / 2)
    IF POINT(T.xc, T.yc) = Border THEN
        Inside = FALSE
        EXIT FUNCTION
    END IF
    Highest = T.y1
    Lowest = T.y1
    IF T.y2 > Highest THEN Highest = T.y2
    IF T.y2 < Lowest THEN Lowest = T.y2
    IF T.y3 > Highest THEN Highest = T.y3
    IF T.y3 < Lowest THEN Lowest = T.y3
     IF T.y4 > Highest THEN Highest = T.y4
    IF T.y4 < Lowest THEN Lowest = T.y4
    X = PMAP(T.xc, 0)
    YU = PMAP(T.yc, 1)
    YD = YU
    H = PMAP(Highest, 1)
    L = PMAP(Lowest, 1)
    IsUp = FALSE
    IsDown = FALSE
    DO
        YU = YU - 1
        YD = YD + 1
        IF NOT IsUp THEN
```

```
            IF POINT(T.xc, PMAP(YU, 3)) = Border THEN IsUp = TRUE
        END IF
        IF NOT IsDown THEN
            IF POINT(T.xc, PMAP(YD, 3)) = Border THEN IsDown = TRUE
        END IF
            IF IsUp AND IsDown THEN
                Inside = TRUE
                EXIT FUNCTION
            END IF
    LOOP UNTIL (YD > L) AND (YU < H)
    Inside = FALSE
END FUNCTION
SUB Message (Text$) STATIC
    SHARED VC AS Config
    LOCATE TROW, TCOL: PRINT SPACE$(19);
    LOCATE TROW, TCOL
    COLOR 7      ' Fehér
    PRINT Text$;
    COLOR 23     ' Villogó
    PRINT "...";
    COLOR 7      ' Fehér
END SUB
FUNCTION Rotated (Lower, Upper, Current, Inc)
    Current = Current + Inc
    IF Current > Upper THEN Current = Lower
    IF Current < Lower THEN Current = Upper
    Rotated = Current
END FUNCTION
SUB SetConfig (mode AS INTEGER) STATIC
    SHARED VC AS Config, BestMode AS INTEGER
    SELECT CASE mode
        CASE 1
            IF BestMode = CGA OR BestMode = MCGA THEN
                VC.Colors = 0
            ELSE
                VC.Colors = 16
            END IF
            VC.Atribs = 4
            VC.XPix = 319
            VC.YPix = 199
            VC.TCOL = 40
            VC.TROW = 25
```

```
CASE 2
    IF BestMode = CGA OR BestMode = MCGA THEN
        VC.Colors = 0
    ELSE
        VC.Colors = 16
    END IF
    VC.Atribs = 2
    VC.XPix = 639
    VC.YPix = 199
    VC.TCOL = 80
    VC.TROW = 25
CASE 3
    VC.Colors = 0
    VC.Atribs = 2
    VC.XPix = 720
    VC.YPix = 348
    VC.TCOL = 80
    VC.TROW = 25
CASE 7  '16 színű, közepes felbontású EGA vagy VGA grafika
    VC.Colors = 16
    VC.Atribs = 16
    VC.XPix = 319
    VC.YPix = 199
    VC.TCOL = 40
    VC.TROW = 25
CASE 8
    VC.Colors = 16
    VC.Atribs = 16
    VC.XPix = 639
    VC.YPix = 199
    VC.TCOL = 80
    VC.TROW = 25
CASE 9
    VC.Colors = 64
    IF BestMode = EGA64 THEN VC.Atribs = 4 ELSE VC.Atribs = 16
    VC.XPix = 639
    VC.YPix = 349
    VC.TCOL = 80
    VC.TROW = 25
CASE 10
    VC.Colors = 0
    VC.Atribs = 2
    VC.XPix = 319
    VC.YPix = 199
    VC.TCOL = 80
```

```
            VC.TROW = 25
            VC.Colors = 216
            VC.Atribs = 2
            VC.XPix = 639
            VC.YPix = 479
            VC.TCOL = 80
            VC.TROW = 30
        CASE 12  ' 16 színű, nagyfelbontású VGA grafika
            VC.Colors = 216
            VC.Atribs = 16
            VC.XPix = 639
            VC.YPix = 479
            VC.TCOL = 80
            VC.TROW = 30
        CASE 13
            VC.Colors = 216
            VC.Atribs = 256
            VC.XPix = 639
            VC.YPix = 479
            VC.TCOL = 40
            VC.TROW = 25
        CASE ELSE
            VC.Colors = 16
            VC.Atribs = 16
            VC.XPix = 0
            VC.YPix = 0
            VC.TCOL = 80
            VC.TROW = 25
            VC.Scrn = 0
        EXIT SUB
    END SELECT
    VC.Scrn = mode
END SUB
SUB SetPalette STATIC
    SHARED VC AS Config, Pal() AS LONG
    IF VC.Colors THEN
      SELECT CASE VC.Scrn
        CASE 1, 2, 7, 8
          Index = 0
          FOR Bs = 0 TO 1
            FOR Gs = 0 TO 1
              FOR Rs = 0 TO 1
```

```
                FOR Hs = 0 TO 1
                  Pal(Index) = Hs * 8 + Rs * 4 + Gs * 2 + Bs
                  Index = Index + 1
                NEXT
              NEXT
            NEXT
          NEXT
        CASE 9
          Index = 0
          FOR Bs = 0 TO 1
            FOR Gs = 0 TO 1
              FOR Rs = 0 TO 1
                FOR HRs = 0 TO 1
                  FOR HGs = 0 TO 1
                    FOR HBs = 0 TO 1
                      Pal(Index) = Rs*32+Gs*16+Bs*8+HRs*4+HGs*2+HBs
                      Index = Index + 1
                    NEXT
                  NEXT
                NEXT
              NEXT
            NEXT
          NEXT
        CASE 11, 12, 13
          Index = 0
          FOR Rs = 0 TO 63 STEP 11
            FOR Bs = 0 TO 63 STEP 11
              FOR Gs = 0 TO 63 STEP 11
                Pal(Index) = (65536 * Bs) + (256 * Gs) + Rs
                Index = Index + 1
              NEXT
            NEXT
          NEXT
        CASE ELSE
      END SELECT
      IF VC.Atribs > 2 THEN TorusRotate RNDM
      END IF
END SUB
SUB TileDraw (T AS Tile) STATIC
    SHARED VC AS Config, TOR AS TORUS
    Border = VC.Atribs - 1
    IF VC.Atribs = 2 THEN
        LINE (T.x1, T.y1)-(T.x2, T.y2), T.TColor
        LINE -(T.x3, T.y3), T.TColor
        LINE -(T.x4, T.y4), T.TColor
```

```
            LINE -(T.x1, T.y1), T.TColor
            EXIT SUB
        ELSE
            LINE (T.x1, T.y1)-(T.x2, T.y2), Border
            LINE -(T.x3, T.y3), Border
            LINE -(T.x4, T.y4), Border
            LINE -(T.x1, T.y1), Border
        END IF
        IF Inside(T) THEN
            PRESET (T.xc, T.yc)
            PAINT STEP(0, 0), BACK, Border
            PAINT STEP(0, 0), T.TColor, Border
        END IF
        IF TOR.Bord = "YES" THEN
            Border = BACK
        ELSE
            Border = T.TColor
        END IF
        LINE (T.x1, T.y1)-(T.x2, T.y2), Border
        LINE -(T.x3, T.y3), Border
        LINE -(T.x4, T.y4), Border
        LINE -(T.x1, T.y1), Border
END SUB
DEFSNG A-Z
SUB TorusCalc (T() AS Tile) STATIC
        SHARED TOR AS TORUS, Max AS INTEGER
        DIM XSect AS INTEGER, YPanel AS INTEGER
        XRot = DegToRad(TOR.XDegree)
        YRot = DegToRad(TOR.YDegree)
        CXRot = COS(XRot)
        SXRot = SIN(XRot)
        CYRot = COS(YRot)
        SYRot = SIN(YRot)
        XInc = 2 * PI / TOR.Sect
        YInc = 2 * PI / TOR.panel
        FirstY = (TOR.Thick + 1) * CYRot
        T(0).x1 = FirstY
        T(TOR.Sect - 1).x2 = FirstY
        T(Max - 1).x3 = FirstY
        T(Max - TOR.Sect).x4 = FirstY
        T(0).y1 = 0
        T(TOR.Sect - 1).y2 = 0
        T(Max - 1).y3 = 0
        T(Max - TOR.Sect).y4 = 0
        T(0).z1 = -(TOR.Thick + 1) * SYRot
```

```
FOR XSect = 1 TO TOR.Sect - 1
    sx = (TOR.Thick + 1) * COS(XSect * XInc)
    sy = (TOR.Thick + 1) * SIN(XSect * XInc) * CXRot
    sz = (TOR.Thick + 1) * SIN(XSect * XInc) * SXRot
    ssx = (sz * SYRot) + (sx * CYRot)
    T(XSect).x1 = ssx
    T(XSect - 1).x2 = ssx
    T(Max - TOR.Sect + XSect - 1).x3 = ssx
    T(Max - TOR.Sect + XSect).x4 = ssx
    T(XSect).y1 = sy
    T(XSect - 1).y2 = sy
    T(Max - TOR.Sect + XSect - 1).y3 = sy
    T(Max - TOR.Sect + XSect).y4 = sy
    T(XSect).z1 = (sz * CYRot) - (sx * SYRot)
NEXT
FOR YPanel = 1 TO TOR.panel - 1
    sx = TOR.Thick + COS(YPanel * YInc)
    sy = -SIN(YPanel * YInc) * SXRot
    sz = SIN(YPanel * YInc) * CXRot
    ssx = (sz * SYRot) + (sx * CYRot)
    T(TOR.Sect * YPanel).x1 = ssx
    T(TOR.Sect * (YPanel + 1) - 1).x2 = ssx
    T(TOR.Sect * YPanel - 1).x3 = ssx
    T(TOR.Sect * (YPanel - 1)).x4 = ssx
    T(TOR.Sect * YPanel).y1 = sy
    T(TOR.Sect * (YPanel + 1) - 1).y2 = sy
    T(TOR.Sect * YPanel - 1).y3 = sy
    T(TOR.Sect * (YPanel - 1)).y4 = sy
    T(TOR.Sect * YPanel).z1 = (sz * CYRot) - (sx * SYRot)
    FOR XSect = 1 TO TOR.Sect - 1
        CountTiles XSect, YPanel
        ty = (TOR.Thick + COS(YPanel * YInc)) * SIN(XSect * XInc)
        tz = SIN(YPanel * YInc)
        sx = (TOR.Thick + COS(YPanel * YInc)) * COS(XSect * XInc)
        sy = ty * CXRot - tz * SXRot
        sz = ty * SXRot + tz * CXRot
        ssx = (sz * SYRot) + (sx * CYRot)
        T(TOR.Sect * YPanel + XSect).x1 = ssx
        T(TOR.Sect * YPanel + XSect - 1).x2 = ssx
        T(TOR.Sect * (YPanel - 1) + XSect - 1).x3 = ssx
        T(TOR.Sect * (YPanel - 1) + XSect).x4 = ssx
        T(TOR.Sect * YPanel + XSect).y1 = sy
        T(TOR.Sect * YPanel + XSect - 1).y2 = sy
        T(TOR.Sect * (YPanel - 1) + XSect - 1).y3 = sy
        T(TOR.Sect * (YPanel - 1) + XSect).y4 = sy
```

```
            T(TOR.Sect * YPanel + XSect).z1 = (sz * CYRot) - (sx * SYRot)
        NEXT
    NEXT
    CountTiles -1, -1
END SUB
DEFINT A-Z
SUB TorusColor (T() AS Tile) STATIC
    SHARED VC AS Config, Max AS INTEGER
    LastAtr = VC.Atribs - 2
    Atr = 1
    FOR Til = 0 TO Max - 1
        IF (Atr >= LastAtr) THEN
            Atr = 1
        ELSE
            Atr = Atr + 1
        END IF
    T(Til).TColor = Atr
    NEXT
END SUB
SUB TorusDefine STATIC
    SHARED VC AS Config, TOR AS TORUS, Available AS STRING
    CONST ENTER = 13, ESCAPE = 27
    CONST DOWNARROW=80, UPARROW=72, LEFTARROW=75, RIGHTARROW=77
    CONST COL1 = 20, COL2 = 50, ROW = 9
    LOCATE 1, COL1
    PRINT "UP..... Következő"
    LOCATE 2, COL1
    PRINT "DOWN.....Előző"
    LOCATE 3, COL1
    PRINT "LEFT..... Forgatás felfelé"
    LOCATE 4, COL1
    PRINT "RIGHT..... Forgatás lefelé"
    LOCATE 5, COL1
    PRINT "ENTER.... Indítás"
    LOCATE 6, COL1
    PRINT "ESCAPE..... Kilépés"
    LOCATE ROW, COL1, 1, 1, 12
    IF LEN(Available$) = 1 THEN Fields = 10 ELSE Fields = 12
    DO
        LOCATE ROW + Fld, COL2 + 2
        DO
            K$ = INKEY$
        LOOP WHILE K$ = ""
        Ky = ASC(RIGHT$(K$, 1))
        SELECT CASE Ky
```

```
        CASE ESCAPE
            CLS: END
        CASE UPARROW, DOWNARROW
            IF Ky = DOWNARROW THEN Inc = 2 ELSE Inc = -2
            Fld = Rotated(0, Fields, Fld, Inc)
        CASE RIGHTARROW, LEFTARROW
            IF Ky = RIGHTARROW THEN Inc = 1 ELSE Inc = -1
        SELECT CASE Fld
            CASE 0
                TOR.Thick = Rotated(1, 9, INT(TOR.Thick), Inc)
                PRINT USING "#"; TOR.Thick
            CASE 2
                TOR.panel = Rotated(6, 20, TOR.panel, Inc)
                PRINT USING "##"; TOR.panel
            CASE 4
                TOR.Sect = Rotated(6, 20, TOR.Sect, Inc)
                PRINT USING "##"; TOR.Sect
            CASE 6
                TOR.XDegree = Rotated(0, 345, TOR.XDegree, (15 * Inc))
                PRINT USING "###"; TOR.XDegree
            CASE 8
                TOR.YDegree = Rotated(0, 345, TOR.YDegree, (15 * Inc))
                PRINT USING "###"; TOR.YDegree
            CASE 10
                IF VC.Atribs > 2 THEN
                    IF TOR.Bord = "YES" THEN
                        TOR.Bord = "NO"
                    ELSE
                        TOR.Bord = "YES"
                    END IF
                END IF
                PRINT TOR.Bord
            CASE 12
                I = INSTR(Available$, HEX$(VC.Scrn))
                I = Rotated(1, LEN(Available$), I, Inc)
                VC.Scrn = VAL("&h" + MID$(Available$, I, 1))
                PRINT USING "##"; VC.Scrn
            CASE ELSE
        END SELECT
        CASE ELSE
    END SELECT
    SetConfig VC.Scrn
LOOP UNTIL Ky = ENTER
LOCATE 1, 1, 0
SELECT CASE VC.Scrn
```

```
        CASE 1
            TOR.Delay =.3
        CASE 2, 3, 10, 11, 13
            TOR.Delay = 0
        CASE ELSE
            TOR.Delay =.05
    END SELECT
    RANDOMIZE TIMER
END SUB
SUB TorusDraw (T() AS Tile, Index() AS INTEGER)
    SHARED Max AS INTEGER
    FOR Til = 0 TO Max - 1
        TileDraw T(Index(Til))
    NEXT
END SUB
SUB TorusRotate (First) STATIC
    SHARED VC AS Config, TOR AS TORUS, Pal() AS LONG, Max AS INTEGER
    SHARED T() AS Tile, Index() AS INTEGER, BestMode AS INTEGER
    DIM Temp AS LONG
    IF VC.Colors THEN
        SELECT CASE First
            CASE RNDM
                FirstClr = INT(RND * VC.Colors)
            CASE START
                FirstClr = 0
            CASE ELSE
                FirstClr = FirstClr - 1
        END SELECT
        IF VC.Colors > Max - 1 THEN
            LastClr = Max - 1
        ELSE
            LastClr = VC.Colors - 1
        END IF
        IF FirstClr < 0 OR FirstClr >= LastClr THEN FirstClr = LastClr
        IF VC.Atribs = 2 THEN
            LastAtr = VC.Atribs - 1
        ELSE
            IF LastClr < VC.Atribs - 2 THEN
                LastAtr = LastClr
            ELSE
                LastAtr = VC.Atribs - 2
            END IF
        END IF
        Work = FirstClr
        FOR Atr = LastAtr TO 1 STEP -1
```

```
            PALETTE Atr, Pal(Work)
            Work = Work - 1
            IF Work < 0 THEN Work = LastClr
        NEXT
    END IF
    IF VC.Atribs = 2 THEN
        FOR I = 0 TO Max - 1
            T(I).TColor = Toggle
        NEXT
        TorusDraw T(), Index()
        Toggle = (Toggle + 1) MOD 2
    END IF
    IF VC.Scrn = 1 AND (BestMode = CGA OR BestMode = MCGA) THEN
        COLOR, Toggle
        Toggle = (Toggle + 1) MOD 2
        EXIT SUB
    END IF
END SUB
SUB TorusSort (Low, High)
    SHARED T() AS Tile, Index() AS INTEGER
    DIM Partition AS SINGLE
    IF Low < High THEN
        IF High - Low = 1 THEN
            IF T(Index(Low)).z1 > T(Index(High)).z1 THEN
                CountTiles High, Low
                SWAP Index(Low), Index(High)
            END IF
        ELSE
            RandIndex = INT(RND * (High - Low + 1)) + Low
            CountTiles High, Low
            SWAP Index(High), Index(RandIndex%)
            Partition = T(Index(High)).z1
            DO
                I = Low: J = High
                DO WHILE (I < J) AND (T(Index(I)).z1 <= Partition)
                    I = I + 1
                LOOP
                DO WHILE (J > I) AND (T(Index(J)).z1 >= Partition)
                    J = J - 1
                LOOP
                IF I < J THEN
                    CountTiles High, Low
                    SWAP Index(I), Index(J)
                END IF
            LOOP WHILE I < J
```

```
            CountTiles High, Low
            SWAP Index(I), Index(High)
            IF (I - Low) < (High - I) THEN
                TorusSort Low, I - 1
                TorusSort I + 1, High
            ELSE
                TorusSort I + 1, High
                TorusSort Low, I - 1
            END IF
        END IF
    END IF
END SUB
```

A QUICK BASIC GRAFIKAI UTASÍTÁSAI

BLOAD fájl_név[,eltolás]

Egy meóriaképfájlnak a memóriába töltésére használjuk. Az eltolás értéke a 0...65535 tartományba eshet. Az eltolás értéke hozzáadódik a legutoljára kiadott DEF SEG-ben adott szegmenscímhez, így a fájl az adott memóriacímtől kezdődően töltődik a memóriába. Ha nem adjuk meg, akkor a BSAVE utasításban adott eltolási értéket veszi figyelembe.
Például: Grafikus kép betöltése:

```
DEF SEG = &HB800 (CGA képernyő)
BLOAD "grafika",0
```

BSAVE fájl_név,eltolás,hossz

Az adott néven, adott eltolással (lásd: BLOAD), adott méretű (0...65535) adat tárolására szolgál.
Például: Grafikus kép tárolása:

```
DEF SEG = &HB800
BSAVE "grafika",0,&H400
```

CIRCLE [STEP] (X,Y),sugár [,[szín] ,[kezdő_szög][,[vég_szög][,arány]]]

Hatására kört, ellipszist, ívet rajzolhatunk adott középponttal. (Lásd: 20. oldal)

CLS [{0 | 1 | 2 }]

A képernyő törlése.

- CLS : A TEXT képernyő törlése
- CLS 0 : A SCREEN és a TEXT képernyő törlése.
- CLS 1 : Csak a SCREEN képernyőt törli.
- CLS 2 : Csak a TEXT képernyő törlése.

COLOR

A képernyőn használható színek beállítása. (Lásd: 10. oldal)

DRAW karakteres_kifejezés

Egy ábra rajzolása a karaktersorozatban megadott értékekkel. (Lásd: 24. oldal)

GET [STEP] (X1,Y1) - [STEP](X2,Y2),tömb[(index)]

A képernyő egy adott részének tárolása a tömbbe. (Lásd: 33. oldal)

LINE [[STEP](X1,Y1)]-[STEP](X2,Y2)[,[szín][,[B[F]][,vonaltípus]]]

Vonal, téglalap rajzolása a képernyőre. (Lásd: 12. oldal)

PAINT [STEP](X,Y)[,[szín][,[keret][,háttér]]]

A képernyő egy adott részének feltöltése a kiválasztott színnel. (Lásd: 15. oldal)

PALETTE régi_szín, új_szín

A paletta színeinek megváltoztatása.

PMAP (koordináta, függvény)

Kizárólag grafikus üzemmódban a koordináták átszámítására használható.

Koordináta	Az átszámítandó érték
Függvény	Lehetséges értékei:
0	felhasználói X leképezése fizikai X-re.
1	felhasználói Y leképezése fizikai Y-ra.
2	fizikai X leképezése felhasználói X-re.
3	fizikai Y leképezése felhasználói Y-ra.

POINT (X,Y)

A képernyő adott grafikus pontjának színkódját adja vissza. (Lásd: 50. oldal)

POINT (N)

A képernyő aktuális koordinátáját adja vissza. (Lásd: 50. oldal)

N	lehetséges értékei:
0	aktuális fizikai X koordináta
1	aktuális fizikai Y koordináta
2	aktuális felhasználói X koordináta
3	aktuális felhasználói Y koordináta

PRESET [STEP] (X,Y)[,szín]

Egy képpont rajzolása a megadott koordinátákra. (Lásd: 11. oldal)

PSET [STEP] (X,Y)[,szín]

Egy képpont rajzolása a megadott koordinátákra. (Lásd: 11. oldal)

PUT [STEP](X,Y),tömb[(index)][kijelzés]

A GET-tel elmentett képernyőtartalom visszaállítása. (Lásd: 33. oldal)

SCREEN (sor,oszlop)

Az aktív képernyő meghatározott karakterének ASCII kódját adja.

Sor, oszlop

a TEXT képernyő koordinátája.

SCREEN [kép_mód][,[szín]

A képernyőmód beállítására használjuk. (Lásd: 9. oldal)

VIEW [[SCREEN](X1,Y1(-(X2,Y2)[,[szín][,keret]]]

A képernyőn egy vagy több „nézőkét" definiál. (Lásd: 31. old.)

VIEW PRINT [felső TO alsó]

Szöveg kiírási területének meghatározása a felső és az alsó sor határának beállításával. A paraméterek elhagyásával az egész képernyő lesz a kiírási terület.
Például.: CLS: VIEW PRINT 3 TO 12

WINDOW [[SCREEN] (X1,Y1)-(X2,Y2)]

Felhasználói ablakméret beállítása. (Lásd: 29. oldal)

A FÁJLKEZELÉS UTASÍTÁSAI

CALL szubrutin_név [(változó_lista)]

Átadja a vezérlést az adott nevű szubrutinnak. A változólista tartalmazza azoknak a változóknak a felsorolását vesszővel elválasztva, amelyeknek a szubrutin értéket adhat. A szubrutint meghívhatjuk a CALL utasítással vagy anélkül is. A CALL utasítás elhagyásával a változólistát nem lehet zárójelek közé tenni, és a DECLARE utasítást sem kell alkalmazni. (Lásd: 48. oldal)

Például:

```
A=1: B=2
CALL szamcsere (a,b)
Print a;b
END                          (Főprogram vége!)
SUB szamcsere (x,y)
    Csere = X: X=Y: Y=Csere
END SUB
```

CHAIN fájl_név

Egy már létező fájl nevére és elérési útvonalára hivatkozva, az adott programból egy másik programra térünk át a változók értékeinek meghagyásával. A programnak a Quick BASIC környezetében kell lennie. Ahhoz, hogy kicseréljük az információt a két program között, a COMMON utasítást kell alkalmazni. A vezérlés, csak a program végrehajtása után kerül vissza.

CLOSE [[#] fájl_szám] [,[#] fájl_szám]...

Egy vagy több nyitott fájl lezárása. A fájl_szám értéke mindig egy olyan létező szám, amelyet előzőleg az OPEN utasításban használtunk. (Lásd: 78. oldal)

COMMON [shared][láncolt_blokk_név]változó_lista [([dimenzió])][AS típus]

Változók értékének átadása egy láncolt programnak. A CHAIN végrehajtásakor fejti ki hatását. A behívott új programnak azon változók értékei adódnak át, amelyeket a COMMON-nal közös változóként definiáltunk.

SHARED Alkalmazásával minden szubrutinban érvényben lesznek a változók.

Láncolt_blokk_név Tartalmazza az összefüggő változók közös blokknevét.

Változó_lista Azon változóneveket kell itt megadni, amelyek a láncolt programban átadásra kerülnek. Ha a változó egy tömb, akkor a dimenziót is meg kell határozni, a típussal együtt.

Például:

```
COMMON a,b,c
a=1: b=2: c=3
CHAIN "Common.bas"
COMMON X,Y,Z
Print x,y,z
```

EOF (fájl_szám)

A fájlvége feltétel ellenőrzésére szolgál.

fájl_szám Meghatározza az adatfájlhoz hozzárendelt, az OPEN utasítással megnyitott fájl számát. A visszaadott érték akkor igaz (1) ha olvasáskor elértük a fájl végét, egyébként hamis (0).

FILEATTR(fájl_szám, fájl_infó)

A fájl elérési módjának lekérdezése.

fájl_szám Meghatározza az adatfájlhoz hozzárendelt, az OPEN utasítással megnyitott fájl számát.

fájl_infó Egy numerikus kifejezés, amely meghatározza a hozzáférési módot.

Értékei lehetnek:

1	INPUT (bevitel)
2	OUTPUT (kivitel)
4	RANDOM (véletlen)
8	APPEND (hozzáfűzés)
32	BINARY (bináris)

FILES [karakteres_kifejezés]

Az aktuális meghajtón levő fájlnevek megjelenítése.
Például:

```
FILES
FILES "*.bas"
FILES "A:\QBASIC\PELDA\*.dat"
```

FREEFILE

Szabad fájlszám lekérdezése. Hatására lekérdezi az eddig megnyitott fájlok számát, és felveszi a következő, megnyitásra alkalmazható fájl_szám értékét.

GET [#]fájl_szám [,[rekord_szám][,változó]]

Egy fájl egy rekordjának beolvasása a fájlpufferba. (Lásd: 33. oldal)

Fájl_szám — Amivel az OPEN utasításban a fájlt megnyitottuk.

rekord_szám — A beolvasandó rekord sorszáma a random fájlban. Értéke 1 és 2 147 483 647 között lehet.

Változó — Annak a változónak a neve, amelybe az adatot helyezzük.

INPUT #fájl_szám, változó

Adatok beolvasása szekvenciális adatfájlból. (Lásd: 78. oldal)

LOC(fájl_szám)

Az aktuális rekordszámot adja vissza.

Fájl_szám — Az OPEN utasítással megnyitott fájl száma. Közvetlen hozzáférésű (random) fájlok esetében az utoljára olvasott, vagy felírt rekord sorszámát adja vissza. Szekvenciális fájlok esetében az OPEN utasítás kiadása óta felírt vagy olvasott blokkok számát adja meg (egy blokk 128 byte).

LOF(fájl_szám)

A fájl hosszát adja meg bájtokban kifejezve.

OPEN Mód,[#] fájl_szám,fájl_név[,rekord_hossz]

Egy fájl vagy eszköz előkészítése használatra. (Lásd: 78. oldal)

Mód A fájl használatának módja. Lehetséges értékei:

O	OUTPUT	Szekvenciális kiviteli mód
I	INPUT	Szekvenciális beviteli forma
A	APPEND	Szekvenciális összefűzési mód
R	RANDOM	Véletlen elérési mód
B	BINARY	Bináris elérési mód

Fájl_szám Egész kifejezés, kezdőértéke 1. Maximális értékét az adott környezet határozza meg.

Fájl_név Egy karakterlánc, amely a fájl nevét és elérési útját tartalmazza.

Rekord_hossz Értéke egész szám. Megadása esetén a direkt elérésű fájlok számára határozza meg a rekord hosszát. Maximális értéke:

szekvenciális fájlnál:	512
random fájlnál:	128
a többi esetben:	327 687

RESET Az összes nyitott fájl lezárása.

WRITE [[#]fájl_szám,] kifejezés_lista

Adatok kiírása képernyőre vagy szekvenciális fájlba.

EGYÉB UTASÍTÁSOK

CONST szimbolikus_név = kifejezés [szimbolikus_név = kifejezés]...

Egy szimbolikus állandó értékadása.
Például:

```
CONST het = 7
CONST nap$= "Hétfő"
DIM ev(het)
```

DECLARE {FUNCTION|SUB} rutin_név [CDECL] [ALIAS"név"] [([lista])]

Egy létező rutin paramétereinek meghatározása típusellenőrzéssel. Minden esetben alkalmazni kell, ha a programunkban hivatkozunk egy létező rutinra, függetlenül attól, hogy a hívást a CALL utasítással, vagy anélkül végezzük.

FUNCTION|SUB — Meghatározza a rutin típusát (függvény vagy alprogram).

CDECL — Jelzi, hogy az adott rutint a CALL utasítással kezeljük.

ALIAS — Az utána megadott név mutatja, hogy a rutin hivatkozási névvel rendelkezik a programon belül.

lista — Ebben adjuk meg, vesszőkkel elválasztva azokat a változóneveket, amelyeket az adott rutin használni fog.

DEF FNnév[(változólista)]=kifejezés

Felhasználói függvény definiálása. (Lásd: 22. oldal)

EXIT

Kilépés a DO vagy FOR ciklusból, illetve funkció vagy szubrutin elhagyása. Hatása az őt követő parancstól függ!

EXIT DO | FOR — Ha a DO vagy FOR ciklusból lépünk ki, akkor a program futása a ciklus utáni programrészen folytatódik.

EXIT FUNCTION | SUB — Ha a FUNCTION vagy SUB esetén alkalmazzuk, akkor hatására a program kilép az éppen aktuális rutinból.

FREE(kifejezés)

Kifejezés

A szabad memória lekérdezése.
Lehet numerikus:

–1 az adott pillanatban szabad memóriaterület értéke bájtokban;

–2 az adott pillanatban szabad verem mérete bájtokban;

vagy karakteres (az előző két számtól eltérő):

Ebben az esetben az adott időpillanatban éppen szabad változóterület méretét kapjuk bájtokban oly módon, hogy először tömöríti a szóközöket egyetlen blokkba, majd meghatározza az így kialakult szabad terület méretét.

FUNCTION név [(argumentum] [STATIC]
név = kifejezés
END FUNCTION

Felhasználói függvény definiálása. Használata megegyezik a DECLARE SUB esetében leírtakkal.

GOSUB név

Vezérlés átadása egy szubrutinnak. (Lásd: 41. oldal)

LBOUND (változónév [,dimenzió])

A dimenzionált tömbök kezdő dimenzióinak lekérdezése.

Változónév

A tömb neve, amelyet a DIM utasítással létrehoztunk.

Dimenzió

Több dimenziós tömb esetében a dimenzió sorszáma.

LPRINT [lista]

Adatok kiírása nyomtatóra.

LPRINTUSING forma;lista

Az adatok formázott kiírása nyomtatóra. Használata megegyezik az előző kötetben leírt PRINTUSING utasítással.

ON numerikus_kifejezés {GOTO | GOSUB } címke [,címke....]

Egy kifejezés értékétől függő programelágazás.

Numerikus kifejezés tetszőleges szám a 0...255 tartományban. Szükség esetén értékét a Quick BASIC kerekíti. Amennyiben értéke 0, akkor a következő utasításon folytatódik a program.

Címke A numerikus értéktől függő címkéket vesszővel válasszuk el. A program futása az adott címkén folytatódik.

ON KEY(n) {GOTO | GOSUB } címke

Egy funkcióbillentyű megnyomásától függő elágazás.

SHARED változó[,változó,....] A főprogramban definiált változó értékének átadása szubrutin számára.

STATIC változó [,változó,...] Segítségével a változók értékeit megőrizhetjük, s átadhatjuk a szubrutinnak.

SUB szubrutin_név [(paraméterek)][STATIC]

Szubrutin készítése a főprogramtól függetlenül. (Lásd: 48. oldal)

TYPE felhasználói_típus
rekord_név
....
END TYPE

Létrehozhatunk egy mintasablont a változók deklarálásához. Ahhoz, hogy kialakítsa ezen típus változóját, a DIM vagy a SHARED utasításokat kell használnunk. Például:

```
TYPE proba
    nev AS STRING*20
    adat AS SINGLE
END TYPE
DIM pro AS proba
    pro.nev = "István"
    pro.adat = 30000
PRINT pro.nev, pro.adat
```

A NYOMTATÓ PROGRAMOZÁSA

Az EPSON FX 850-es típusú nyomtató beállítási értékeinek felhasználásával, bármely más típusú nyomtató kézikönyve alapján megadhatók az alábbi lehetőségek.

KARAKTERSŰRŰSÉG SZABÁLYOZÁSA

LPRINT CHR$(27);"P"	*6 pontos betűt nyomtat*
LPRINT CHR$(27);"M"	*8 pontos betűt nyomtat*
LPRINT CHR$(27);"W1"	*széles írás bekapcsolása*
LPRINT CHR$(27);"W0"	*széles írás kikapcsolása*
LPRINT CHR$(27);CHR$(15)	*keskeny írás bekapcsolása*
LPRINT CHR$(27);CHR$(18)	*keskeny írás kikapcsolása*
LPRINT CHR$(27);"E"	*félkövér nyomtatás bekapcsolása*
LPRINT CHR$(27);"F"	*félkövér nyomtatás kikapcsolása*
LPRINT CHR$(27);"G"	*felfelé kettős nyomtatás bekapcsolása*
LPRINT CHR$(27);"H"	*felfelé kettős nyomtatás kikapcsolása*
LPRINT CHR$(27);"-";CHR$(1)	*aláhúzott nyomtatás bekapcsolása*
LPRINT CHR$(27);"-";CHR$(0)	*aláhúzott nyomtatás kikapcsolása*

STÍLUSVÁLTÁSOK

LPRINT CHR$(27);"x1"	*levélminőség bekapcsolása*
LPRINT CHR$(27);"x0"	*levélminőség kikapcsolása*
LPRINT CHR$(27);"4"	*dőlt betű bekapcsolása*
LPRINT CHR$(27);"5"	*dőlt betű kikapcsolása*

SORTÁVOLSÁG BEÁLLÍTÁSA

LPRINT CHR$(27);"0"	
LPRINT CHR$(27);"1"	
LPRINT CHR$(27);"2"	
LPRINT CHR$(27);"3";CHR$(n)	*pontegységenként, ahol n = 3...255*
LPRINT CHR$(27);"A";CHR$(n)	*n/72 colonként, ahol n = 3...255*

MINTAPROGRAMOK

1.
```
A$="01234567890123456789012345678901234567890123456789"
B=-1.5678
WIDTH "lpt1:",30
LPRINT A$;CHR$(10)
OPEN "lpt1:" AS 1
PRINT#1,1,14,28
LPRINT TAB(10);"10"
LPRINT SPC(10);"10"
PRINT#1,A$
WIDTH "lpt1:",50
LPRINT A$
PRINT#1,USING "####.##";B
WIDTH#1,80
CLOSE 1
```

2.
```
REM esc_A
E$=CHR$(27)
LPRINT E$;"M"
FOR I=7 TO 12:S$=MID$(STR$(I),2)
    LPRINT E$;"A";CHR$(I)
    FOR J=1 TO 3
        LPRINT "sortávolság: "S$"/72""
    NEXT J
NEXT I
LPRINT E$;"@"
```

3.
```
REM esc_B
E$=CHR$(27)
VT$=CHR$(11)
LPRINT E$;"2";
LPRINT E$;"B";CHR$(2);CHR$(8);CHR$(20);CHR$(0);
LPRINT "1. sor";
LPRINT VT$;"2.sor";
LPRINT VT$;"8.sor";
LPRINT VT$;"20.sor "
LPRINT E$;"@"
```

4.
```
REM esc_c
CLS:REM táblázathossz
INPUT "táblázathossz [1,127]: ";FL
INPUT "perforációátugrás: ";P
IF FL<1 OR FL>127 THEN 20
LPRINT CHR$(27);"C";CHR$(FL);
LPRINT CHR$(27);"N";CHR$(P);
FOR Z=1 TO FL
    S=S+1:LPRINT Z;CHR$(132);S+P+INT((S-1)/(FL-P))
NEXT Z
LPRINT CHR$(12);:REM lapdobás
LPRINT CHR$(27);"@"
```

5.
```
REM esc_ht
E$=CHR$(27):HT$=CHR$(9)
LPRINT E$;"P";E$;CHR$(15)
LPRINT "012345678901111111112222222222333333333344444444445";E$;"1"
LPRINT"123456789012345678901234567890123456789012345678901234567890";E$;"2";CHR$(13)
LPRINT E$;"D";CHR$(5);CHR$(15);CHR$(25);CHR$(35);CHR$(0);
LPRINT HT$;"5";
LPRINT HT$;"15";
LPRINT HT$;"25";
LPRINT HT$;"35"
LPRINT E$;"@"
```

6.
```
REM esc_n
E$=CHR$(27)
LPRINT E$;"C";CHR$(0);CHR$(12)
LPRINT E$;"N";CHR$(4)
FOR I=1 TO 2
    FOR J=1 TO 10
        LPRINT "teteje"
    NEXT J
    LPRINT CHR$(12)
NEXT I
```

7.
```
REM esc_p
E$=CHR$(27)
LPRINT E$;"t";CHR$(1);
LPRINT E$;"p1";"prop  +–ă"
LPRINT E$;"E";"prop félkövér";E$;"F"
LPRINT E$;"G";"prop keskeny–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"félkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"–";CHR$(1);E$;"E";"félkövér–aláhúzott";E$;"–";CHR$(0);E$;"F"
LPRINT E$;"w1";"nyújtott";E$;"w0"
LPRINT CHR$(15);"keskeny"
LPRINT E$;"E";"keskeny–félkövér";E$;"F"
LPRINT E$;"G";"keskeny–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"keskeny–félkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"keskeny–aláhúzott";E$;"–";CHR$(0);CHR$(18)
LPRINT CHR$(14);"széles egy sorra"
LPRINT E$;"W1";"széles"
LPRINT E$;"E";"széles–félkövér";E$;"F"
LPRINT E$;"G";"széles–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"széles–félkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"széles–aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"4"
LPRINT E$;"–";CHR$(1);"dőlt széles–aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"@"
```

8.
```
REM esc_s
E$=CHR$(27)
LPRINT E$;"M";E$;"E";
LPRINT E$;"S0";
LPRINT "Felsőindex ";E$;"T";"szöveg"
LPRINT E$;"S1";
LPRINT "Alsóindex  ";E$;"T";"szöveg"
LPRINT E$;"@"
```

9.
```
REM esc_sp
E$=CHR$(27)
LPRINT E$;"M";E$;"E";
FOR SZ=0 TO 10
LPRINT E$;" ";CHR$(SZ);
LPRINT "karakterszéthúzás"
NEXT SZ
LPRINT E$;"@"
```

10.
```
REM esc_t
E$=CHR$(27)
A$=CHR$(201)+STRING$(40,205)+CHR$(187)
B$=CHR$(186)+"  Kazinczy Ferenc Általános Iskola, Tapolca  "+CHR$(186)
C$=CHR$(200)+STRING$(40,205)+CHR$(188)
LPRINT E$;"t";CHR$(1);
LPRINT A$
LPRINT B$
LPRINT C$
LPRINT E$;"@"
```

11.
```
REM esc_w
E$=CHR$(27)
LPRINT E$;"t";CHR$(1);E$;"w1"
LPRINT E$;"E";"félkövér";E$;"F"
LPRINT E$;"G";"keskeny–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"felkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"–";CHR$(1);E$;"E";"félkövér–aláhúzott";E$;"–";CHR$(0);E$;"F"
LPRINT E$;"w1";"nyújtott"
LPRINT CHR$(15);"keskeny"
LPRINT E$;"E";"keskeny–félkövér";E$;"F"
LPRINT E$;"G";"keskeny–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"keskeny–félkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"keskeny–aláhúzott";E$;"–
";CHR$(0);CHR$(18)
LPRINT CHR$(14);"széles egy sorra"
LPRINT E$;"W1";"széles"
LPRINT E$;"E";"széles–félkövér";E$;"F"
LPRINT E$;"G";"széles–két beütés";E$;"H"
LPRINT E$;"G";E$;"E";"széles–félkövér–két beütés";E$;"H";E$;"F"
LPRINT E$;"–";CHR$(1);"széles–aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"4"
LPRINT E$;"–";CHR$(1);"dőlt széles–aláhúzott";E$;"–";CHR$(0)
LPRINT E$;"@"
```

QUICK BASIC HIBAÜZENETEK

Hibaüzenet	A hiba értelmezése
Advenced Feature	Továbbfejlesztett lehetőség
Array not definied	Nem definiált tömb
Bad file mode	Hibás fájlmegnyitási mód
Bad file name	Hibás fájlnév
Bad file number	Rossz fájlszám
Bad record length	Hibás rekordhossz
Cannot continue	Nem folytatható
CASE SELECT expected	ELSE nélküli CASE
Device I/O error	Beviteli/Kiviteli eszköz hibája
Device timeout	Időtúllépés
Device unavailable	Nem használható készülék
Disk media error	Lemezhiba
Disk not ready	A lemezegység nem áll készen
Disk write protected	Írásvédett lemez
Division by zero	Nullával való osztás
Duplicate Definition	Többszörös definíció
Duplicate label	Megismételt címke
Field overflow	Mezőtúlcsordulás
File already exists	Már létező fájl
File already open	Már megnyitott fájl
File not found	A fájl nem található
FOR without NEXT	NEXT nélküli FOR
Illegal direct	Nem megengedett parancs
Illegal Function call	Hibás függvényhívás
Internal error	Belső hiba
Line buffer overflow	Sor-buffer túlcsordulás
Missing operand	Hiányzó operandus
NEXT without FOR	NEXT nélküli FOR
No Resume	Nincs Resume
Out of DATA	Nincs több adat
Out of memory	Betelt a memória
Out of paper	Kifogyott a papír
Out of string space	Stringterület betelt

Overflow	Túlcsordulás
Path not found	Az elérési út nem létezik
Rename Across disks	Átnevezés más lemezre
RESUME without error	RESUME hiba nélkül
RETURN without GOSUB	RETURN nélküli GOSUB
String formula too complex	Túl bonyolult stringkifejezés
String to long	Túl hosszú karakterlánc
Subprogram not definied	Nem definiált szubrutin
Subscript out of range	Tartományon kívüli index
Syntax error	Formai hiba
Too many file	Túl sok fájl
Type mismatch	Típus-összeférhetetlenség
Undefined line number	Nem definiált sorszám
Undefinied user function	Nem definiált függvény
Unprintable error	Nem nyomtatható hiba
Variable required	Hiányzik a változó
WEND without WHILE	WHILE nélküli WEND
WHILE without WEND	WEND nélküli WHILE

QUICK BASIC UTASÍTÁSOK ÖSSZEFOGLALÓ TÁBLÁZATA

Utasítás	Rövid leírás
ABS(X)	Az X abszolút értéke
ASC(X$)	Az X$ első karakterének kódja
BEEP	Sípoló hangjelzés
BLOAD	Memóriaképfájl memóriába töltése
BSAVE	Memóriakép tárolása periférián
CALL	BASIC szubrutin hívása
CBDL(X)	Az X duplapontosságú értékét adja
CHAIN	Vezérlés átadása egy másik programnak
CHDIR	Az aktuális alkönyvtár kijelölése
CHR$(kód)	Adott kódú karakter előállítása
CINT(X)	Az X egészre kerekített értéke
CIRCLE	Kör, ellipszis, ív rajzolása
CLNG(X)	Az X kerekítése 4 bájtos egész számmá
CLOSE	Egy vagy több nyitott fájl lezárása
CLS	Képernyő törlése
COLOR	Képernyő színeinek beállítása
COMMON	Változók értékének átadása egy programnak
CONST	Egy szimbolikus állandó értékadása
CSNG(X)	Az X értékének egyszeres pontosságúra alakítása
CSRLIN	A kurzor függőleges koordinátája
DATA	Numerikus és karakteres adatok tárolása
DATE$	Dátum lekérdezése
DATE$=dátum	Dátum megadása
DECLARE	Egy létező rutin paramétereinek meghatározása
DEF FNnév	Felhasználói függvény definiálása
DEF SEG	Szegmenscím kijelölése a memóriában
DEFDBL	Változócsoportok típusának duplapontos beállítása
DEFINT	Változócsoportok típusának egész számra való beállítása
DEFLNG	Változócsoportok típusának hosszú egész számra való beállítása
DEFSNG betűk	Változócsoportok típusának egyszeresen pontos számra való beállítása

DEFSTR betűk	Változócsoportok típusának karaktertípusú változóra való beállítása
DIM	Tömb definiálása
DO - UNTIL	Igaz feltételig történő ismétlési szerkezet
DO - WHILE	Hamis feltételig történő ismétlési szerkezet
DRAW RS	Grafikus ábra készítése
END	Program befejezése, az összes fájl lezárása
EOF(szám)	A fájlvége feltétel ellenőrzése
ERASE	Tömbök törlése a programban
EXIT	Kilépés a DO, FOR ciklusból, vagy szubrutinból
FILES	Fájlok neveinek megjelenítése
FIX(X)	Az X szám egésszé alakítása csonkítással
FOR	Ismétlési szerkezet (FOR-TO-STEP-NEXT)
FREE	Szabad memóriaterület lekérdezése
FREEFILE	Szabad fájlszám lekérdezése
FUNCTION	Felhasználói függvény definiálása
GET	A grafikus képernyő adott részének tárolása
GET#	Egy direktfájl egy rekordjának beolvasása
GOSUB	Vezérlés átadása szubrutinnak
GOTO	Vezérlésátadás címkére
HEX$(x)	Az X decimális szám hexadecimális alakja
IF..THEN	Feltételtől függő programelágazás
INKEY$	Karakter beolvasása billentyűzetről
INPUT	Adatok bekérése billentyűzetről
INPUT#	Adatok beolvasása szekvenciális fájlból
INPUT$(Y)	Y darabszámú karakter beolvasása billentyűzetről vagy fájlból
INSTR	Egy karaktersorozat keresése egy másik karaktersorozatban
INT(X)	Az X egész számmá konvertálása
KEY	Funkcióbillentyűk által helyettesített karaktersorozat beállítása, megjelenítése
KEY(n)	Billentyűzetesemény figyelése
KILL	Fájl törlése lemezről
LBOUND	Dimenzionált tömbök kezdő dimenziójának lekérdezése
LCASE$(X$)	Karakteres kifejezés kisbetűssé alakítása
LEFT$()	Baloldali N számú karakter lekérdezése
LINE	Vonal, téglalap rajzolása

LINE INPUT	Teljes sor beolvasása billentyűzetről
LOC()	Aktuális rekordszám
LOCATE	Kurzor pozíciójának beállítása
LOF(szám)	A lemezfájl hosszát adja
LPRINT	Adatok kiírása nyomtatóra
LPRINT USING	Adatok formázott kiírása nyomtatóra
LTRIM$(X$)	Az X$ bal oldali szóközeinek törlése
MID$()	Karakterlánc adott részének kiválasztása
MKDIR	Új alkönyvtár létrehozása
OCT$(x)	Decimális szám oktálissá alakítása
ON	Egy kifejezés értékétől függő programelágazás
OPEN	Egy fájl vagy eszköz megnyitása
PAINT	Képernyő adott részének festése
PALETTE	A paletta színeinek megváltoztatása
PLAY A$	A$-ban tárolt zene lejátszása
PMAP()	Koordináták lekérdezése
POINT()	Az adott pont színének vagy koordinátájának lekérdezése
POS()	A kurzor oszloppozíciójának lekérdezése
PRESET	Képpont rajzolása
PRINT	Adatok kiírása
PSET	Képpont rajzolása
PUT	Képernyőtartalom visszaállítása
RANDOMIZE	Véletlenszám-generátor újraindítása
READ	A DATA utasításban adottak elolvasása
REDIM	Adattömb méretének megváltoztatása
REM	Magyarázó szöveg
RESET	Az összes nyitott fájl lezárása
RESTORE	A DATA sorok ismételt használata
RETURN	Visszatérés szubrutinból
RIGHT$()	A karakteres kifejezés jobb oldali N számú karakterének vizsgálata
RMDIR	Könyvtár törlése
RND()	Véletlenszám előállítása
RTRIM$(X$)	Szóközök levágása a karakteres kifejezés végéről
RUN	Program futtatása memóriából vagy lemezről
SCREEN()	Egy karakter elvolvasása a képernyőn
SCREEN	Képernyőmód beállítása

SEEK()	Az aktuális fájlmutató helyzete
SELECT	Kifejezés kiértékelése utáni elágazás
SGN(X)	Az X előjelét adja
SHARED	Szubrutin számára a változó értékének átadása
SHELL	Egy DOS-parancsot hajt végre
SLEEP	Adott ideig felfüggeszti a program futását
SOUND	Adott frekvenciájú hang előállítása
SPACE$(n)	N darab szóköz előállítása
SPC(N)	N darab szóköz kiíratása
STATIC	Értékmegőrzés a szubrutinok számára
STOP	Megállítja a program végrehajtását
STR$(X)	Az X konvertálása karakteressé
STRING$()	Egy adott karakter ismétlése
SUB	Szubrutinkészítés főprogramtól függetlenül
SWAP X,Y	Az X és Y értékének felcserélése
SYSTEM	Visszatérés a vezérlőrendszerbe
TAB(n)	Az adott sor N-edik pozíciójára ugrás
TIME$	Az aktuális rendszeridőt adja
TIME$=idő	Az adott időpont beállítása
TIMER	Az éjfél óta eltel másodperceket adja
TROFF	Nyomkövetés kikapcsolása
TRON	Nyomkövetés bekapcsolása
TYPE	Felhasználói típus definiálása
UBOUND()	Adattömb felső határának lekérdezése
UCASE$(X$)	Az X$ nagybetűssé alakítása
VAL(X$)	Az X$ numerikussá alakítása
VIEW	Képernyőn nézőke beállítása
VIEW PRINT	Szövegkiírási terület beállítása
WHILE	Utasítások ciklusban történő végrehajtása a feltétel igaz állapotáig
WINDOW	Felhasználói ablak definiálása
WRITE	Adatok kiírása képernyőre vagy fájlba

TARTALOMJEGYZÉK

Nemzeti Tankönyvkiadó Rt.
A kiadásért felel: dr. Ábrahám István vezérigazgató
Raktári szám: 639
Műszaki vezető: Héjjas Mária igazgatóhelyettes
Műszaki szerkesztő: Járdi Emília
Terjedelme: 12,15 (A5) ív
1. kiadás, 1996

96/2317 Franklin Nyomda és Kiadó Kft., Budapest
Felelős vezető: Győri Géza ügyvezető igazgató

Ez a kiadvány a Dunaújvárosi Finompapírgyár Kft.
BIANCOPRINT Super White 80 g/m² papírjára készült.

A sorozatot mindazoknak ajánljuk, akik:
– az alapoktól szeretnék elsajátítani a PC használatát,
– tanítani szeretnék a számítástechnikát,
– kezdők a számítástechnika tanulásában.

A sorozat további köteteiben
– a Quick BASIC grafika,
– a Windows és alkalmazói programjai,
– az adatbáziskezelés,
– a Turbo Pascal programnyelv
tanításához, tanulásához adunk segédeszközt
a tanulók, tanárok kezébe.

639

9 789631 872255