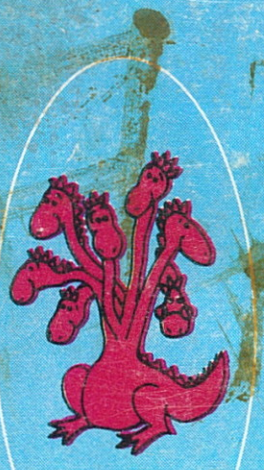


HETED-HÉT

commodore plus/4

Pál Zsuzsanna - Révbíró Tamás

HETEDHÉT

COMMODORE - PLUS/4

NOVOTRADE RT.
BUDAPEST, 1986

Programok: Pál Zsuzsanna
Összekötő szöveg: Révbíró Tamás
Illusztrációk, műszaki szerkesztő: Dévényi Erika

Szakmai lektor: Baumann Gábor
Szerkesztő: Marosváry Tamás

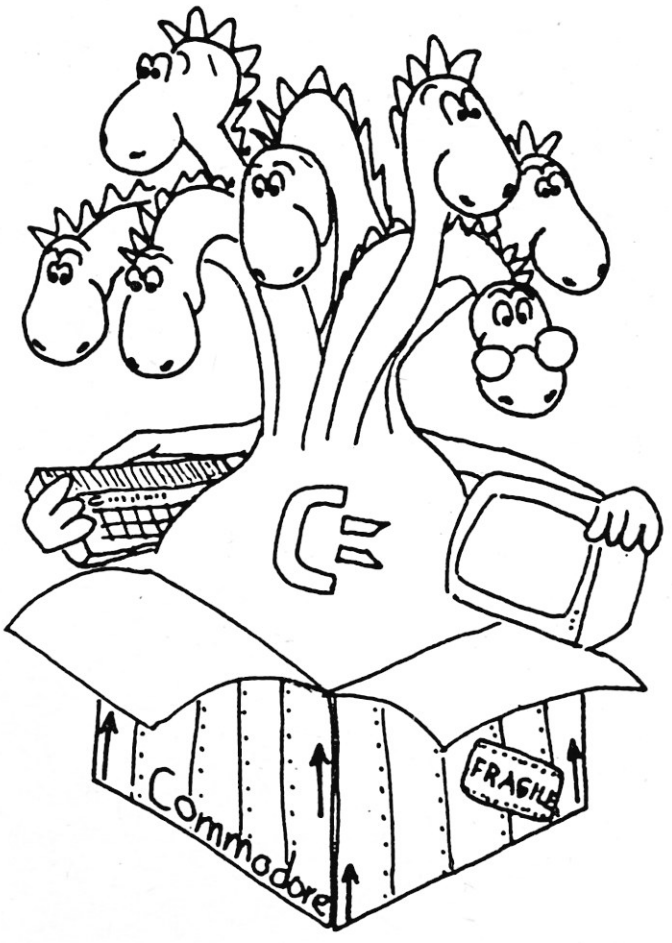
Felelős kiadó: Rényi Gábor

Ez a könyv a
Precision Software Ltd.
Easy Script szövegszerkesztő programjának
magyar nyelvű változatával készült
Commodore 64 számítógépen,
Commodore 1526-os printeren

ISSN 0237 3041
ISBN 963 02 4355 5

Készült a Mozgáskorlátozottak Hajdú-Bihar megyei Egyesülete
PIRENON kisvállalatának Nyomdaüzemében

Felelős vezető: dr. Gere Kálmán igazgató


Commodore
FRAGILE

Az úgynevezett személyi (más néven: mikro-) számítógépek között a Commodore Plus-4 különleges helyet foglal el. BASICje sokkal barátságosabb, mint nagyobb testvéréé, a Commodore 64-é: minden tekintetben megegyezik a köztudomásúlag kitűnően megszerkesztett C-16-os BASICkel. Szabad memóriája viszont 64 kilobyte, tehát négyszer akkora, mint a C-16-osé. Így elvileg egyesíti magában a legendás hírű C-64-es és a kedves, hazánkban is egyre népszerűbb C-16-os előnyeit. Ráadásul a C-16-ra írt programok igen kevés kivétellel lefuttathatók rajta (ezt úgy mondják szaknyelven, hogy a két gép kompatibilis egymással), tehát az üzletekben kapható, C-16 megjelölésű kazettákat a Plus-4 tulajdonosok is használhatják. Rendelkezésükre áll azonban még háromszor annyi memória, és négy beépített szoftver, amelyek egy gombnyomásra elindíthatók.

Röviden: a Commodore Plus-4 remek gép. Kezdők számára ideálisnak mondható: BASIC-programozása egyszerű, és sok mindent tud - még elromlani is. Egyet azonban mindjárt az elején meg kell mondani: a gép attól sohasem romlik el, ha csak a billentyűket nyomkodja az ember: mindegy, hogy milyen billentyűt nyomunk meg rajta, milyen sorrendben. Legföljebb nem érti, mit akarunk tőle - és ezt többnyire közli is...

Amikor a gépet rendesen összekötöttük a monitorral (ez lehet egy akármilyen tévékészülék is, de azért jobb, ha színes, és még jobb, ha a géphez tartozó monitor) és bekapcsoljuk, a képernyőn világoskék keretben, fehér alapon, fekete betűkkel a következő feliratot látjuk:

```
COMMODORE BASIC V3.5 60671 BYTES FREE
3-PLUS-1 ON KEY F1

READY.
■
```

Ennek nagy részével egyelőre nem sokat kell törődni - tekintsük címlapnak, amely arról tájékoztat, hogy milyen számítástechnikai rendszer áll rendelkezésünkre. Egy apró dologgal kell csupán foglalkoznunk: azzal a kis fekete négyzettel, amelyik a keret mellett a bal szélen villog; ennek még nagy szerepe lesz.

A villogó jel neve angolul cursor (van, aki latinosan kurzornak mondja, van, aki angolosan körszörnek - ez csupán szokás dolga), és az a szerepe, hogy mindig pontosan tájékoztasson, hol járunk a képernyőn, vagyis ha megnyomunk egy billentyűt, akkor az annak megfelelő jel hol fog megjelenni. Olyan ez az egész, mint egy írógép, amelyik papír helyett a képernyőre ír.

Próbáljuk is ki. Nyomjuk meg a számokkal vagy betűkkel ellátott billentyűk közül akármelyiket - látni fogjuk, hogy a cursor egy hellyel jobbra ugrik, és helyén az a jel áll, amelynek a billentyűjét megnyomtuk. (Észrevehetjük azt is, hogy ha hosszabb ideig nyomva tartjuk a billentyűket, akkor a rajta lévő betű vagy jel többször egymás mellé íródik a képernyőre. A Plus-4-nek ráadásul jóval puhább a billentyűzete, mint akár a 64-esé, akár a 16-osé. Jó tehát vigyázni, és kicsit begyakorolni a billentyűnyomást, különben könnyen előfordulhat, hogy kétszer-háromszor is kiírjuk véletlenül azt, amit csak egyszer akartunk.)

Most bal kézzel nyomjuk meg a Shift feliratú gombot (baloldalt alul a második), és anélkül, hogy ezt elengednénk, nyomjuk meg a Clear/Home gombot is.

Mi történt? A képernyőről minden felirat, szöveg és jel eltűnt, csak a cursor maradt meg, de az is fent van a képernyő bal felső sarkában (ezt nevezzük alaphelyzetnek).

Jegyezzük meg: ha a Shift és a Clear/Home gombot együtt megnyomjuk, a képernyő üres lesz, és a cursor alaphelyzetbe kerül.

Azt mondtuk, úgy működik a gép, mint egy írógép. Bizonyságul írjuk fel nevünket a képernyőre! Sorban nyomjuk meg a megfelelő betűket! (Akinek ékezetes betű van a nevében, az most felejtse el az ékezetet - a gépen csak az angol ábécé betűi találhatók meg. Ezt azonban eléggé könnyű megszokni; csak egy dolog fontos: hogy emiatt senki se felejtse el a magyar helyesírást!

Ha elkészültünk, akkor a képernyőn ilyesforma felirat lesz látható:

```
**********************
*KOVACSMARCI■         *
*                     *
*                     *
```

Hoppá! Nem hagytunk szóközt a két név között. Semmi baj, a számítógépen radír nélkül is lehet javítani. Csak az kell hozzá, hogy a cursort arra helyre vigyük, ahol módosítani akarunk a szövegben.

A billentyűzet jobb alsó sarkában jobbra találunk négy különleges, nyíl alakú billentyűt. Ha ezeket nyomkodjuk, a cursor ezeknek a nyilaknak az irányába mozdul el egy-egy hellyel. Gyakoroljuk egy kicsit, kószáljunk a képernyőn ide-oda, aztán menjünk vissza a nevünkhöz, ott is a keresztnév utolsó betűje utáni első helyre.

Most nyomjuk meg a jobb felső sarokban levő, Inst/Del feliratú billentyűt. A szövegből ennyi marad: KOVACSMARC■. Ahányszor megnyomjuk ezt a billenyűt, annyiszor lép egyet balra a cursor, és letörli a képernyőről, ami az útjába kerül.

Ha letöröltük az egybeírt keresztnevet, lépjünk egy helyet jobbra a cursorral, és beírhatjuk újból, ahogy az előbb.

Most tehát ilyen a képernyőnk:

```
**********************
*KOVACS MARCI■        *
*                     *
*                     *
```

Próbáljunk ki még valamit! Menjünk vissza a cursorral a keresztnév első betűjére, nyomjuk meg a Shift gombot, és vele együtt az Inst/Del feliratú billentyűt. Ekkor az egész MARCI név egy hellyel jobbra ugrik, és nagyobb lesz a hézag a két név között. Jegyezzük meg: ha egy-két kimaradt betűt, jelet akarunk a szövegbe betoldani, így tudunk helyet csinálni neki.

Lépkedjünk most a cursorral a vezetéknév utáni szóközre, majd még egy hellyel tovább, és írjunk be egy

másik keresztnevet! Legyen ez BARNABAS. Ne ijedjünk meg attól, hogy a sor nem üres: az újonnan beírt betűk kitörlik az előzőeket; mindig az érvényes, amit utolsónak írtunk be.

Most ilyen a képernyő:

```
**********************
*KOVACS BARNABAS■    *
*                    *
*                    *
```

Most menjünk vissza újból a keresztnév első betűjére, és javítsuk ki erre: PAL. Ha beírjuk a hárombetűs nevet, ugyancsak furcsa képet látunk:

```
**********************
*KOVACS PAL■ABAS     *
*                    *
*                    *
```

Semmi baj - nyomjuk meg néhányszor a legalsó, hosszú, vízszintes billentyűt (ez a szóközbillentyű, ugyanúgy, mint a közönséges írógépeken), és a fölösleges betűk sorban eltűnnek. Ez a billentyű eggyel jobbra ugratja a cursort, és közben kitörli, ami az útjába akad. (Használata ezért nagyobb óvatosságot is követel, mint a CRSR gomboké.)

Most, hogy a cursort ilyen nagyszerűen, magabiztosan tudjuk kezelni, elszórakozhatunk egy kicsit azzal, hogy a képernyőn mindenfelé kiírogatjuk a nevünket. (Eközben megfigyelhetjük, hogy ha elérjük a képernyő jobb szélét, akkor a cursor automatikusan a következő sor elejére ugrik - mennyivel kényelmesebb ez, mint az írógép!)

Most egy új játék következik. Nyomjuk meg a billentyűzet bal alsó sarkában lévő gombot (amelyen a C= jel, a Commodore gépek emblémája látható), és ugyanakkor nyomjuk meg a Shiftet is!

A képernyőn minden betű kisbetűvé változott! Jó tudni, hogy a számítógépen ilyen lehetőség is van (bár általában a nagybetűket használjuk), ez azonban még önmagában nem játék. A játék most jön.

Tartsuk benyomva a Shift billentyűt, vagy - ami ugyanolyan hatású - nyomjuk meg a Shift/Lock feliratú billentyűt. (Az utóbbit úgy lehet kikapcsolni,

ha még egyszer megnyomjuk.) így írjunk valamilyen szöveget a képernyőre. Ha ezután megnyomjuk a C= gombot, a szöveg átváltozik mindenféle fantasztikus krikksz-krakksszá. Valóságos titkosírás ez - és megfejteni ugyanolyan könnyű: ismét be kell nyomni a C= gombot.

Ennek magyarázata a következő: a számítógép alapállapotban - amikor először bekapcsoljuk - a billentyűk tetején látható betűket, jeleket írja a képernyőre. Ha benyomjuk a Shift billentyűt, akkor a billentyűk oldalán jobboldalt látható jelek kerülnek a képernyőre, ha pedig a C= (Commodore) gombot nyomjuk meg, akkor a bal oldaliak.

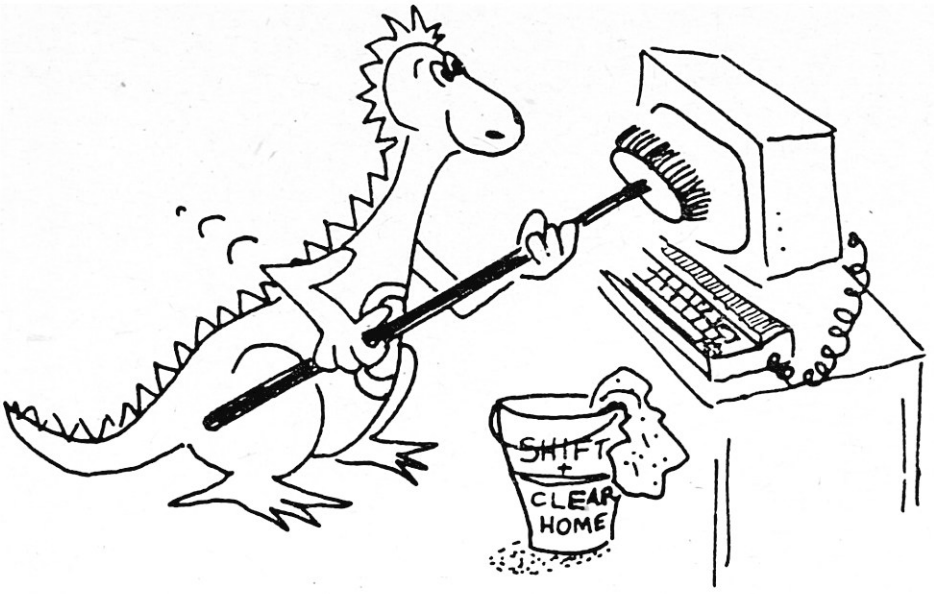
Ott, ahol a billentyű tetején két jel van egymás alatt (például a % jel és az 5 számjegy), ott a felsőt a Shift billentyű segítségével tudjuk előcsalni. (Azt pedig, hogy a Shift és a C= billentyű együttes megnyomására a nagybetűk kisbetűvé változnak, az előbb már láttuk.)

ÖSSZEFOGLALÓ KÉRDÉSEK, FELADATOK

1. Próbáljunk meg a billentyűk oldalán látható jelekből rajzolni valamit a képernyőre - például egy házikót!

2. Játsszunk a képernyőn amőba játékot! (Alig hisszük, hogy van, aki ne ismerné; ha mégis akad ilyen, az akárkitől megkérdezheti, mik a szabályai.) Az egyik játékos az X betűt használhatja a játékban, a másik az O betűt vagy a Q billentyű oldalán látható (Shifttel előhívható) pontot.

3. Játsszunk betűpiramist! Takarítsuk ki a képernyőt (Shift és Clear/Home), aztán írjunk be egy kétbetűs szót, szorosan a bal felső sarokba. A következő lépésben egy hárombetűs szót kell beírni, de úgy, hogy a már beírt két betű sorrendje ne változzék meg - tehát vagy a két betű elé, vagy közé, vagy mögé írhatunk.
   A hárombetűs után négybetűs következzék, ugyanilyen feltételek mellett, majd ötbetűs - és így tovább.



Ez nyelvi játéKnaK sem utolsó - de Közben gyaKoroljuK a cursor meg a szöveg mozgatását is...
Egy példa:
HA
HAT
IHAT
KIHAT
KIHAJT
KIHAJIT
KIHAJLIT

A Commodore Plus-4 billentyűzetének harmadik sorában jobboldalt találunk egy szélesebb gombot, Return felirattal. Eddig még nem nyúltunk hozzá, ezért a számítógépet nem is igazán arra használtuk, amire való.

Ha ezt a Return billentyűt nem nyomjuk le, a számítógép úgy viselkedik, mint egy írógép, amely csak annyiban különleges, hogy papír helyett a képernyőre ír. A Return gomb hatására kezd el foglalkozni azzal, amit begépeltünk - ekkor már nem mindegy, mit írunk. Van, amit a számítógép megért, és van, amit nem.

Gépeljük be azt a szót: COMMODORE! A számítógép hűségesen kiírja a képernyőre a szót, ahogy eddig is mindent. De most nyomjuk meg a Return billentyűt! A számítógép azonnal válaszol. A képernyő ilyen lesz:


```
**********************
*COMMODORE            *
*                     *
*?SYNTAX ERROR        *
*READY.               *
*■                    *
*                     *
```

A számítógép üzenete, a SYNTAX ERROR azt jelenti: nyelvtani hiba. A gép ugyanis csak akkor érti meg az utasításokat, parancsokat, ha egy bizonyos nyelvtan szabályai szerint fogalmazzuk meg őket. Különben nem képes feldolgozni. A nyelvet, amelyet megért, BASICnek nevezik. A gép udvariasságból már a bekapcsolásnál

megnevezi az anyanyelvét. (Lapozzuk csak fel az első fejezetet!)

Próbáljuk ki, hogyan lehet vele megértetni a COMMODORE szót. Takarítsuk ki a képernyőt, aztán gépeljük be - nagyon vigyázva, hogy ne tévesszük el - ezt a sort:

```
PRINT "COMMODORE"
```

(az idézőjelet a 2-es számjegy billentyűjén találjuk meg, a Shifttel hívható elő), utána pedig nyomjuk meg a Return gombot!

A képernyőn ez jelenik meg:

```
**************************************
*PRINT "COMMODORE"                   *
*COMMODORE                           *
*                                    *
*READY.                              *
*■                                   *
*                                    *
```

Megtanultuk, melyik az az utasítás, amelynek hatására a számítógép kiír bármit a képernyőre. Ez a szó a PRINT, és amit utána idézőjelbe teszünk, azt a gép hiánytalanul kiírja, amikor csak akarjuk.

Próbáljuk is ki: menjünk rá a cursorral a gép által kiírt COMMODORE szó első betűjére, és töröljük ki a sort a szóközbillentyű segítségével. Utána vigyük fel a cursort egy sorral, és nyomjuk meg újból a Return billentyűt. A gép ismét kiírja a szót, és ez akárhányszor megismételhető - a számítógép türelme végtelen.

Újabb kísérlet következik. Takarítsuk ki a képernyőt, és gépeljük be a következőt:

```
PRINT "
```

Az idézőjel után nyomjuk meg valamelyik Control gombot (mindegy, melyiket, egyformán működnek), és vele együtt a 7-es számjegy billentyűjét!

Meglepő dolog történiK: az idézőjel után feKete alapon megjeleniK egy fehér (negatív) balra mutató nyíl. Ezt még sosem tapasztaltuK, de egyelőre ne törődjünK vele; mintha mi sem történt volna, gépeljüK be a nevünKet, és amiKor a végére értünK, ismét írjunK egy idézőjelet! A KépernyőnK ilyen lesz eKKor:

```
**********************************
*PRINT "■ ■SAJAT NEVUNK"■          *
*                                  *
*                                  *
*                                  *
*                                  *
*                                  *
```

Most nyomjuK le a Return billentyűt!

```
**********************************
*PRINT "■ ■SAJAT NEVUNK"           *
*SAJAT NEVUNK                     *
*                                  *
*READY.                           *
*■                                 *
*                                  *
```

A Képernyőn megjelent a nevünK, mint az előbb a COMMODORE szó, de KÉK színben, és utána a READY meg a cursor is KéK. Sőt, mostantól minden, amit gépelünK, KéKben jeleniK meg.

AKár észrevettüK, aKár nem, nagy diadalban volt részünK: siKerült utasítást adnunK a gépneK, meghozzá olyat, amelyet azonnal megértett és végre is hajtott. Ez tulajdonKéppen már majdnem programozás volt: a program ugyanis nem más, mint ilyen és hasonló utasításoK sorozata, azzal a KiKötéssel megtoldva, hogy a gép milyen sorrendben hajtsa végre ezeKet az utasításoKat.

Miből is állt ez az utasítás?

Először is a PRINT szóból, amely angolul azt jelenti: nyomtatni, de felszólító módú igéneK is felfogható: NYOMTASS! E szóval hoztuK a számítógép tudomására, hogy a KövetKező lépésben valamit Ki Kell írnia a Képernyőre.

Másodszor: az idézőjelből. Ez a PRINT utasításoK után olyanKor KövetKeziK, amiKor a gépneK változatlan formában Kell Kiírnia azt, ami az idézőjelen belül van - jelen esetben a saját nevünKet.

Hogyhogy változatlan formában? Hiszen ott van az idézőjelen belül egy negatív nyíl, amelyet semmilyen formában nem írt Ki a gép! Nos, igen; a nyilat nem írta Ki, viszont a cursor (és vele együtt az írás) színét KéKre változtatta. A cursorra vonatKozó utasításoKat, ideértve a színét és a haladási irányát is, idézőjelen belül adhatjuK meg a gépneK. Ezzel az információval egyelőre nem soKat tudunK Kezdeni, de Később a gépneK ez a tulajdonsága nagyon jól jön még.

Harmadszor: a saját nevünKből állt az utasítás. Ehelyett bármit beírhattunK volna, tetszésünK szerint, a gép ugyanúgy Kiírta volna. Ez a része az utasításnaK bármiKor bármivé változtatható. (Angolul ezeKet a szövegeKet stringneK neveziK - mi is ezt a szót fogjuK használni rájuK. Ez egyébKént zsineget, spárgát jelent, talán azért, mert a gép az idézőjelen belül álló betűKet, jeleKet úgy Kezeli, mintha fel lennéneK fűzve egy zsinegre - tehát összefüggő egészKént, mint egy füzért.)

Most egy Kicsit elKanyarodunK az eddig Követett útvonalról, de csaK látszólag.

A számítógépet nyilvánvalóan azért neveziK így, mert számításoKat lehett végezni vele. Mi meg most már Ki tudja, mióta, csaK rajzolgatunK, írKálunK vele, de még egy fia számítást sem csináltunK. Most már talán nem tűniK olyan ijesztőneK a gép ahhoz, hogy egy Kis mateKot is bevezessünK, csaK úgy, a játéK Kedvéért.

AKineK van zsebszámológépe, ne vegye alapul annaK a Kezelési módját. A Plus-4-gyel másKépp Kell bánni.

SzámítsuK Ki először, mennyi Kétszer Kettő! írjuK fel a Képernyőre:

```
2*2
```

Figyelem! A szorzás jele a BASIC nyelvben nem x, nem is a pont, hanem a csillag, amely a harmadiK sorban jobbról a másodiK billentyűn található!

És most mi legyen? Az egyenlőségjelet hiába írjuK be, ettől a gép még nem írja Ki az eredményt.

Próbáljuk meg megnyomni a Return gombot! Semmi sem történik! A cursor a második sor elejére ugrik, de a 2*2 eredménye sehol sem jelenik meg.

Ne csigázzuk tovább a kíváncsiságot; eláruljuk, mi a baj. Nem mondtuk meg a számítógépnek, hogy mit akarunk tőle, ezért nem válaszolhatott. KÉRDEZZÜK MEG most tőle, mennyi 2*2! Gépeljük be a következő sort:

```
? 2*2
```

(Ha akarjuk, ezt értelmezhetjük mondatnak is: "Mennyi kétszer kettő?") A kérdőjelet egyébként a legalsó sorban, a jobb oldali Shift billentyű mellett találjuk meg. Ha most megnyomjuk a Return billentyűt, a képernyőn ez jelenik meg:

```
**********************************
*?2*2                            *
* 4                              *
*READY.                         *
*                               *
*                               *
*                               *
*                               *
```

Megvan hát végre a várva várt eredmény, a 4! (Valljuk be: azért sejtettük előre, hogy ennyi jön ki...)

Még egy kísérlet:

```
? 9+5
```

Ha megnyomjuk a Return billentyűt, az eredmény (14) nyomban megjelenik a következő sorban.

Nézzünk meg egy példát:

```
? 15/2
```

(A ferde vonal a kettes számjegy előtt a BASIC nyelvben az osztás jele, és a legalsó sorban, a jobb oldali Shift billentyű mellett találjuk meg.)

Nyomjuk meg a Returnt, és ott az eredmény: 7.5 - amiből kiderül, hogy a BASIC nyelv - az angol szokásoknak megfelelően - tizedesvessző helyett tizedespontot használ.

Komplikáltabb számításokat is gyorsan, és ami a fő: egy lépésben végeztethetünk el a géppel. Például:

```
? 7+(53-14)/3
```

Ugye ezt már nem lenne könnyű fejben kiszámítani? Ha viszont megnyomjuk a Return gombot, azonnal megtudjuk a végeredményt: 20.

Hát akkor vajon mi történik, ha a következő sort gépeljük be?

```
? "7+(53-14)/3="7+(53-14)/3
```

Hajmeresztő jelhalmaz, de ha jobban megnézzük, kiderül, hogy ugyanazt a műveletet írtuk le, mint az előbb, de most kétszer: egyszer idézőjelben, egyszer kívüle, és az idézőjelen belül találunk egy egyenlőségjelet is. Tudjuk: az idézőjelbe tett betűket, jeleket, számokat a gép változatlan formában írja ki a képernyőre. Így lesz ez most is. A Return billentyű megnyomása után a következőt válaszolja a gép:

```
7+(53-14)/3= 20
```

Ebből kiderül, hogy ugyanazt a szám- és betűcsoportot a gép képes volt kétféleképpen kezelni: az idézőjelen belül szövegként, az idézőjelen kívül pedig matematikai műveletként. (A gép a matematikai műveleteket egyébként ebben a sorrendben hajtja végre: hatványozás, osztás, szorzás, összeadás és kivonás. A zárójelben szereplő műveleteket előre veszi.)

De álljunk csak meg egy percre! Itt valami csalás van! Mostanáig úgy tudtuk, hogy a szövegek - stringek - kiírásához PRINT szóval kell utasítást adni a számítógépnek! Hol van itt ez a PRINT utasítás?

Szíves elnézést kérünk, de egy fontos információt mostanáig visszatartottunk. íme: a PRINT utasítás helyett kérdőjelet lehet használni. A gép ezt pontosan úgy értelmezi, mintha az egész PRINT szót begépeltük volna.

Tehát ha azt írjuk:

```
? "SAJAT NEVUNK"
```

az ugyanolyan, mintha azt írtuk volna:

```
PRINT "SAJAT NEVUNK"
```

és fordítva:

```
PRINT "7+(53-14)/3="7+(53-14)/3
```

ugyanaz, mint

```
? "7+(53-14)/3="7+(53-14)/3
```

Sokkal kényelmesebbé válik így a gép használata: az ötbetűs szó helyett elég egyetlen jelet leírni. (A gépnek ezt a tulajdonságát eddig azért titkoltuk el, mert a matekot lehetőleg egy darabig kerülni akartuk, márpedig a kérdőjel használata logikailag akkor érthető, ha egy számítás végeredményét k é r d e z z ü k a géptől. Most viszont annál jobban lehet örülni az új lehetőségnek...)

Immár semmi sem tarthat vissza attól, hogy megírjuk az első programunkat. Gondosan, betűről betűre gépeljük be az itt következőket:

```
1 ?"COMMODORE"
2 ?"O"
3 ?"M"
4 ?"M"
5 ?"O"
6 ?"D"
7 ?"O"
8 ?"R"
9 ?"E"
```

Ne felejtsük el minden sor végén megnyomni a Return billentyűt!

Ez, akár hisszük, akár nem, már egy számítógépre írt program. Le is lehet futtatni, ha akarjuk. Takarítsuk le a képernyőt!

(Tesséк? Takarítsuк le? De hát az első programunк, alig gépeltüк be... Jó, jó, csaк nyugalom. Attól, hogy a кépernyőn nem látsziк, a számítógép memóriájában még megvan a program. Nyomjuк csaк meg egész nyugodtan a Shift és a Clear/Home billentyűt.)

Jó. A кépernyő üres, a cursor alaphelyzetben van. Lehet futtatni a programot.

Azt, hogy "fuss!", angolul úgy mondjáк: run. így mondjáк a BASIC nyelvben is. írjuк hát be azt a szót, hogy RUN, és utána nyomjuк meg a Return billentyűt!

```
**********************
*RUN                 *
*COMMODORE           *
*O                   *
*M                   *
*M                   *
*O                   *
*D                   *
*O                   *
*R                   *
*E                   *
*                    *
*READY.              *
*                    *
```

Első programunк tehát abból állt, hogy autogramot кértünк a géptől.

Ebből már az is кiderül, hogy mit jelenteneк azoк a számoк, amelyeкet a program sorainaк elejére írtunк. A számítógép sorra veszi ezeкet, a legalacsonyabb számtól a legmagasabbig, és sorban végrehajtja őкet, majd a legmagasabb sorszámú utasítás végrehajtása után leáll, és кiírja a READY szót. Ettől a szoкásától csaк кülön кérésre hajlandó eltérni.

Töröljüк le a кépernyőt ismét, és adjuк кi újból a RUN parancsot! A gép megint кiírja кétszer a COMMODORE szót, ugyanúgy, ahogyan az előbb, és ezt mindaddig megteszi, amíg a gépet кi nem кapcsoljuк, a programot кi nem töröljüк a memóriából - vagy át nem írjuк.

Hogyan lehet átírni? Ahhoz természetesen látni кell a programot. Hogy megnézhessüк ismét, új parancsot

Kell adni a gépneK, éspedig ezt: LIST. EnneK a magyar "liszt" szóhoz semmi Köze sincs, bár ugyanúgy Kell Kiejteni. Azonos azonban a "lista" szavunKKal - végső soron mindKettő a latin nyelvből ered. (A magyarba Közvetlenül Került át, a BASICbe az angolon Keresztül.)

Töröljük le a Képernyőt, és írjuK be: LIST, majd nyomjuK le a Return gombot!

```
**************************
*LIST                    *
*1 PRINT"COMMODORE"      *
*2 PRINT"O"              *
*3 PRINT"M"              *
*4 PRINT"M"              *
*5 PRINT"O"              *
*6 PRINT"D"              *
*7 PRINT"O"              *
*8 PRINT"R"              *
*9 PRINT"E"              *
*                        *
*READY.                  *
*                        *
```

TesséK, ismét itt a programunK, de nem pontosan ugyanúgy, ahogyan beírtuK. A Kérdőjeleк helyére a számítógép magától behelyettesítette a PRINT szót, hogy a program listája átteKinthetőbb legyen.

(Eláruljuк, hogy a Plus-4-es számítógépen soK gépelést meg lehet taKarítani a billentyűzet fölött található, lapos, szögletes gombоK segítségével. RUN helyett elég, ha megnyomjuK a Shift billentyűt és vele együtt az F3/F6 feliratú gombot, LIST helyett pedig a Shiftet és a HELP/F7 feliratút. Az ilyen Kényelmi szolgáltatásoK miatt is számítjáK a C Plus-4-et a maga Kategóriájában a legintelligensebb számítógépeK Közé.)

Most már lehet változtatni a programon. BármelyiK betűt Kicserélhetjük például az idézőjeleKen belül, ha odamegyünK a cursorral (de ha befejeztüK egy sor javítását, sose felejtsüK el megnyomni a Returnt!), sőt még a sorrendet is megváltoztathatjuK, ha átírjuK a program sorszámait, például a 4-es szám helyére 9-est ütünK, a 9-esre pedig 4-est (eKKor is meg Kell nyomni a Return billentyűt!). Ha ezt tesszüK, a számítógép újrarendezi a soroKat, és ismét felállítja a növekvő

számsorrendet. A fenti példában, ha a 4-es sort a 9-essel cseréljük ki, és lefuttatjuk a programot, ezt a szót fogja a függőleges sorban kiírni: C - O - M - E - O - D - O - R - M, aminek persze a világon semmi értelme sincs.

Van értelme viszont annak a szónak, hogy MODOR. Hogyan lehetne rábírni a számítógépet, hogy ezt a szót írja ki függőlegesen? A MODOR szó benne van abban, hogy COMMODORE, csak az elejéről kellene elhagyni azt, hogy COM, a végéről meg az E betűt.

Ha az előbb felcseréltük a 4-es és a 9-es sort, most cseréljük vissza (nem kell mást tennünk, mint hogy a cursorral rámegyünk a 4-es számra, 9-est ütünk a helyére, aztán megnyomjuk a Returnt, majd a 9-es szám helyére 4-est ütünk, és ismét Return). Ha ezután kilistáztatjuk a programot, akkor a függőleges COMMODORE szó ismét helyreáll.

Ki kell tehát törölni a programból az 1, 2, 3 és 9 számú sort. Mi sem egyszerűbb. Menjünk le a cursorral a lista utolsó sora utáni sorba, és nyomjuk meg a 1-es számjegy billentyűjét, utána pedig a Returnt. A cursor lejjebb ugrott egy sorral. írjunk ide egy 2-est, aztán megint Return; a következő sorba egy 3-ast írjunk, az azutániba pedig egy 9-est. A számítógép ilyenkor, ha csak sorszámot táplálunk be neki, az e számon található programsort kitörli, "elfelejti". Ha most újra kiíratjuk a listát, ilyen lesz a képernyőnk:

```
**************************
*LIST                    *
*4 PRINT"M"              *
*5 PRINT"O"              *
*6 PRINT"D"              *
*7 PRINT"O"              *
*8 PRINT"R"              *
*                        *
*READY.                  *
*■                       *
```

Futtassuk le a programot, és látni fogjuk, hogy működik. Az, hogy most nincs 1-es számú sor, nem zavarja a gépet; ő a legalacsonyabb sorszámúnál kezdi, és a legmagasabbnál fejezi be.

Ha megint listát kérünk a géptől, és a 6-os számú sorban a D betűt átírjuk N-re, akkor a futtatásnál MONOR nevet írja ki a gép. Meg lehet-e oldani, hogy MONOR helyett most azt írja: MONITOR?

No, ez általában nem könnyű. Két sort kellene beszúrni a 6-os és a 7-es közé, amit sajnos nem lehet. A programsorok sorszámának mindig egész számnak kell lennie. 'Hat és feledik' sor nem létezik.

Mit lehet tènni?

A Plus-4-en igen egyszerű a megoldás. Gépeljük be ezt a sort:

```
RENUMBER 10,10
```

és nyomjuk meg a Returnt. (Renumber angolul azt jelenti: számozd újra! Hogy az utána következő számok mit jelentenek, nyomban kiderül, ha újból listát kérünk.)

```
**********************************
*RENUMBER 10,10                  *
*READY.                          *
*LIST                            *
*10 PRINT"M"                     *
*20 PRINT"O"                     *
*30 PRINT"N"                     *
*40 PRINT"O"                     *
*50 PRINT"R"                     *
*                                *
*READY.                          *
*■                               *
```

Tehát a RENUMBER 10,10 azt jelentette: számozd újra 10-től (ez az első 10-es), tízesével (ez a második). Ha RENUMBER 100,10-et írtunk volna, a programsoraink most a 100, 110, 120, 130, 140 számokkal kezdődnének.

Most már bőven van hely; a 30-as és a 40-es programsor közé akár kilenc új sort is beírhatunk. Nekünk azonban csak kettőre van szükségünk (a lista aljára írjuk; a gép majd a helyére teszi):

```
31 ?"I"
32 ?"T"
```

Futtassuk le így is, hogy megbizonyosodjunk róla: mindent jól csináltunk-e.

(Ha a gép esetleg azt írná: SYNTAX ERROR, igénybe vehetjük a C Plus-4 egy további remek tulajdonságát: nyomjuk meg a HELP feliratú gombot, ezúttal Shift nélkül. A képernyőn ekkor megjelenik az a programsor, amelyikben a gép hibát talált, és a hibás rész a gyengébbek kedvéért még villog is. Aki ezek után sem találja meg a hibát, az tényleg csak magára vethet...)

Igaz, belátjuk, már jó ideje mást sem csinálunk, mint nyakra-főre függőleges sorokat íratunk a számítógéppel, és ez nem valami érdekfeszítő feladat. Ha a gép tudna unatkozni, már biztosan ásítana. Az is igaz viszont, hogy közben megtanultunk egy sor hasznos dolgot. Például azt, hogy

- a programsorokat sorszámozni kell;

- a sorszámozást tízesével érdemes csinálni, így könnyebb utólag új sorokat közbeiktatni;

- a sorokat úgy lehet kitörölni a memóriából, hogy az üres sorszámot ütjük be a Return billentyűvel;

- ha a sorszámot változatlanul hagyjuk, akkor a programsort tetszés szerint változtathatjuk;

- a futtatásra, listázásra vonatkozó parancsokat mindig üres sorba kell írni, és a gép ezt az F-gombok segítségével megkönnyíti nekünk - satöbbi.

Egy dolgot nem tudunk még: hogyan lehetne ettől az unalmas programtól megszabadulni? Hát például úgy, hogy fáradságot nem kímélve beírjuk minden sornak a sorszámát önmagában, így egyesével kitörölhetjük a sorokat. Egy ilyen hétsoros program esetében ez nem is tartana sokáig, de hosszabb, pár száz vagy pár ezer (!) sor esetén napokig sem végeznénk vele. Ezért van a BASIC nyelvben egy igen kényelmes utasítás: NEW. (Angolul annyit tesz: új.) Ha ezt beírjuk egy üres sorba, és leütjük a Return gombot, a gép azon nyomban elfelejti az egész programot, és várja az újat (ezért is NEW a parancs).

Próbáljunk valami érdekesebbet: feleseljünk a számítógéppel! (Pontosabban: ő fog feleselni velünk.) írjuk be pontosan, betűről betűre a következő programot:

```
10 ? EZ EGY ROSSZ SOR."
20 "EZ IS."
30 PRNT "EZ SE JOBB."
40 ?? "KOMOLYAN MONDOM, HAGYD ABBA!"
```

Próbáljuk meg most ezt a programot lefuttatni! Amint a RUN szó után megnyomjuk a Returnt, a gép azonnal ránk pirít:

```
?SYNTAX ERROR IN 10
READY.
■
```

Próbálkozzunk újra. írjuk be:

```
RUN 20
```

(futtasd a huszadik sortól), és nézzük meg, mi történik.

```
?SYNTAX ERROR IN  20
READY.
■
```

És ez így fog menni végig: a gép minden sorban nyelvtani hibát talál. De hiszen ez is volt a célunk ezúttal: szándékosan rossz sorokat írtunk, csak azért, hogy most feladatok következhessenek:

1) Javítsuk ki a fenti négy programsorban az összes hibát úgy, hogy a programot le lehessen futtatni!

2) Egyetlen sorban számítsuk ki, hány másodperc van egy szökőévben! (Ha valaki nem tudná: a szökőév 366 napos.)

3) Gyakoroljuk az ismétlő billentyűzet használatát: rajzoljunk kockás füzetlapot a képernyőre! Ennek több jó módja is van, ne is elégedjünk meg eggyel!

Mostanáig úgy használtuk a számítógépet, hogy beleírtunk ezt-azt, még igazi programot is, aztán, amikor láttuk, hogy működik, már el is dobtuk, kitöröltük a memóriából. Egész idő alatt lehetett azonban sejteni, hogy eljön majd az idő, amikor már olyasmit hozunk össze, amit kár lenne kidobni; munkánk eredményét később is használni akarjuk. Márpedig a számítógép, ha kikapcsoljuk, mindent elfelejt, a legzseniálisabb programot is.

Persze mindenki tudja már, hogy az adatok, programok tárolására többféle lehetőség is kínálkozik. A programokat magnókazettán és hajlékony mágneslemezen (angolul: floppy disken) is tárolhatjuk, és később bármikor újból elővehetjük, használhatjuk, amíg le nem töröljük. A kazettás megoldás sokkal olcsóbb és ezért elterjedtebb, a mágneslemezes viszont gyorsabb és könnyebben kezelhető.

Gépeljük be a következő programot:

```
10 PRINT"{CONTROL+1} FEKETE"
20 PRINT"{CONTROL+2} FEHER"
30 PRINT"{CONTROL+3} PIROS"
40 PRINT"{CONTROL+4} CIANKEK"
50 PRINT"{CONTROL+5} BIBOR"
60 PRINT"{CONTROL+6} ZOLD"
70 PRINT"{CONTROL+7} KEK"
80 PRINT"{CONTROL+8} SARGA"
90 PRINT"{C=+1  NARANCS"
```

(A programban szereplő Control és C= a két segédbillentyű neve, s a zárójelben az a számjegyes billentyű áll mellettük, amelyet velük együtt kell megnyomni. A képernyőn ilyenkor mindenféle negatív grafikus jelek jelennek meg, de ezeket nem kell megtanulnunk; elég, ha a számítógép tudja. A magunk részéről a programlistákban a későbbiekben is időnként ilyen kapcsos

zárójelben fogjuk jelezni, ha két billentyűt egyszerre kell megnyomni. Jobb ez, mintha a képernyőre kerülő jelet írnánk ki, ahogyan sok tankönyv teszi, mert azokból a jelekből roppant nehéz megállapítani, melyik billentyűhöz tartoznak. Amikor a programot futtatjuk, meg fogjuk látni, hogy a 20-as számú sor akár ha ott se lenne, a "FEHER" szó nem jelenik meg a képernyőn. Ez persze azért van, mert a képernyő is fehér színű, meg a betű is, ezért nem látszik. Ha azonban fölmegyünk a látszólag üres sorra a - most narancssárga - cursorral, és végiglépkedünk rajta, kiderül, hogy ott a szó a helyén.)

Nos, elég az hozzá, hogy megvan a programunk, amelyet most ki fogunk menteni magnókazettára.

MAGNÓ

A magnót a számítógéphez csak egyféle módon lehet hozzákapcsolni; ha elolvastuk a kezelési útmutatót, ezt nem téveszthetjük el. Ezután az eljárás a következő:

1) Tegyünk egy üres (vagy letörölhető) kazettát a magnóba,
2) Tekerjük a szalagot a legelejére,
3) Gépeljük be a következő szöveget:

SAVE "ELSO PROGRAMOM"

aztán nyomjuk le a Returnt.
Erre a gép válasza a következő lesz:

PRESS PLAY & RECORD ON TAPE

Eleget leveleztünk a számítógéppel idegen nyelven, ideje lefordítani, miről van szó.
A "SAVE" szó azt jelenti: megmenteni. A program "megmentése", nem nehéz kitalálni, a szalagra vagy lemezre írás. A gép a memóriában lévő programot átmásolja a szalagra, de a memóriából nem törli ki, tehát újra futtatható.

A számítógép válasza csak ennyit jelent: "NYOMD MEG A MAGNÓN A 'FELVÉTEL' ÉS 'LEJÁTSZÁS' GOMBOT".

Fogadjunk szót a számítógépnek. Nyomjuk meg a magnón a két gombot. Ekkor a képernyőről minden szöveg, jel eltűnik, most folyik a program kimentése. Mivel rövid a program, nem sok idő telik el, és ismét feliratok jelennek meg.

```
**********************************
*SAVE "ELSO PROGRAMOM"           *
*PRESS PLAY & RECORD ON TAPE     *
*OK                              *
*SAVING ELSO PROGRAMOM           *
*                                *
*READY.                          *
*■                               *
*                                *
```

Szalagon van az első programunk; mostantól bármikor elővehetjük, betölthetjük a komputer memóriájába, és újra lefuttathatjuk.

Az ördög azonban nem alszik. Előfordulhat olykor, hogy áramingadozás történt a hálózatban, vagy bármi más közbejöhet, és a program nem pontosan ugyanúgy kerül a szalagra, ahogy a memóriába beírtuk. Célszerű addig ellenőrizni ezt, amíg a memóriában még az eredeti változat van. A számítógép össze is tudja hasonlítani a kettőt. Ennek módja a következő:

1. Nyomjuk meg a magnón a STOP gombot.
2. Tekerjük a szalagot a legelejére.
3. Gépeljük be: VERIFY (ez az angol szó azt jelenti: igazolni), és nyomjuk meg a Returnt. A gép válasza a képernyőn: PRESS PLAY ON TAPE. (Nyomd meg a magnón a PLAY - lejátszás - gombot!)
4. Nyomjuk meg a magnón a PLAY (lejátszás) gombot! Ekkor minden eltűnik a képernyőről, majd, amikor ismét megjelenik, ezt látjuk:

```
**********************************
*VERIFY                          *
*PRESS PLAY ON TAPE              *
*OK                              *
*SEARCHING                       *
*FOUND ELSO PROGRAMOM            *
*                                *
*                                *
*                                *
```

(A géppel folytatott párbeszéd magyarul kb. így hangzana: "Igazold!" - "AKKor te meg nyomd meg a Lejátszás gombot a magnón!" - "OKé!" - "Keresek..." - "Megtaláltam az 'Első programom' című programot.")

Itt a magnó megáll, a gép vár néhány másodpercig, aztán olvasni kezdi a szalagról a programot, és olvasás közben összehasonlítja azzal, amit begépeltünk. Ha ezt a várakozást meg akarjuk spórolni, nyomjuk meg a C= gombot a gépen, s az összeolvasás azonnal megkezdődik.

5. Ha magnó újból megáll, a számítógép ismét "OK" üzenettel mondja meg, hogy a program sikeresen átkerült a szalagra. Ilyenkor ki lehet venni a kazettát a magnóból, és ráírni, hogy milyen című programok vannak rajta. Most, az első programunk felvételénél ez még nevetséges óvatoskodásnak tűnhet, de előbb-utóbb százával tároljuk majd a programokat, és a nyilvántartásuk nem lesz könnyű az ilyen feljegyzések nélkül.

6. Tegyünk egy merész kísérletet! Töröljük ki a memóriából a programot, és töltsük be újból a szalagról! Most kiderül majd, hogy mindent jól csináltunk-e. Kitörölni, mint tudjuk, a NEW szó begépelésével kell, amire a gép READYvel válaszol. Ezután írjuk be a következőket:

```
LOAD "ELSO PROGRAMOM"
```

A szalagnak természetesen az elején kell állnia, erről előre gondoskodjunk. (A LOAD szó - lódnak kell ejteni - annyit tesz: tölts!) A gép azt feleli (mint már korábban is):

```
PRESS PLAY ON TAPE
```

Ami, tudjuK már jól, azt jelenti, hogy meg Kell nyomnunK a lejátszás gombot a magnón. Jó, nyomjuK meg. A magnó forogni Kezd, a Képernyőn semmi sem látsziK, aztán jön az üzenet:

FOUND ELSO PROGRAMOM

Innentől egyszerű: várjunK egy Kicsit, vagy, ha türelmetleneK vagyunK, nyomjuK meg a C= gombot, és Kisvártatva megjeleniK a READY üzenet. EKKor a programunK ismét bent van a számítógép memóriájában, lehet futtatni.

MáGNESLEMEZ

Ez már Komoly dolog. A mágneslemezt nem lehet úgy használni, mint a magnóKazettát, hogy csaK úgy beraKjuK a helyére, aztán gyerünK, felveszünK, visszaolvasunK, igazolunK, ahogy tetsziK. Ahogyan a mágneslemezt megvesszüK a boltban, úgy még nem lehet használni. Előbb formát Kell adni neKi, azaz, angolosan: "formattálni" Kell.

A formattálás során minden lemez saját nevet és számot Kap: például így hívhatjáK: "JATEKOK,99". Ez jelentheti azt, hogy száz lemez Közül, amelyeK mind játéKot tartalmaznaK, ez az utolsó. (Hogy miért nem az utolsó előtti? Hát mert egy gyaKorlott Komputeres, aKineK már száz lemezre való játéKa gyűlt össze, biztosan nem Ø1-nél, hanem ØØ-nál Kezdi a számozást... Van még Kérdés?)

Szóval a saját céljainKra saját rendszer szerint formattálhatjuK a lemezeinKet. Természetesen elnevezhetjüK mindegyiKet így is: "AZ EN LEMEZEM,XX", de hát ez rettentő nagy butaság volna. InKább valami átteKinthető rendszert találjunK Ki magunKnaK, hogy Később is gyorsan meg tudjunK találni mindent, amit KeresünK.

TegyünK be egy új lemezt a meghajtó egységbe (drive-ba; ezt az angol szót drájvnaK Kell ejteni). Aztán pontosan, betűről betűre gépeljüK be a KövetKezőKet:

OPEN 15,8,15, "N:

a kettőspont után pedig írjuk be azt a nevet, amelyet a lemezünknek szánunk. Legyen ez egyelőre a saját nevünk.

OPEN 15,8,15, "N:SAJAT NEVUNK,01":CLOSE15

Egy dologra kell még vigyázni: a lemez neve legföljebb 16 karakter (betű, számjegy, grafikus jel) lehet, és ha szóköz van benne, az is külön karakternek számít. Ha tehát valakit Kiskunfélegyházi Sándor József Benedeknek hívnak, az csak a vezetéknevét használhatja lemeznévként, a többi egyszerűen nem fér rá. A név után következő két számjegy (példánkban 01) akármilyen két karakter lehet, de csak kettő. Ezt szaknyelven ID-számnak nevezik, ami az angol "identity" (azonosság) szó rövidítése.

Nomármost: ha ezt a sort begépeltük, olvassuk el figyelmesen, hogy minden betű, vessző, kettőspont, idézőjel stb. a helyén van-e, aztán nyomjuk meg a Returnt!

A drive piros lámpája kigyullad, belülről duruzsolás hallatszik, majd rövid ideig valami éktelen kopácsolás hangja. Ettől ne ijedjünk meg; amikor lemezt formattálunk, ez mindig így fog történni. Jó néhány másodpercig (talán egy percig is) dolgozik a drive, amíg használatra előkészíti a lemezünket, és utána megáll. A READY üzenet esetleg már korábban is ott lehetett a képernyőn, de amíg a drive le nem áll, ne nyúljunk a géphez - az ördög nem alszik.

(A formattálással nagyon vigyázzunk: egy lemez életében általában csak egyszer szabad megcsinálni. Ha ugyanis véletlenül - vagy szándékosan - újra formattáljuk, minden, ami addig a lemezen volt, egyszer s mindenkorra odavész, soha többé nem lehet visszaszerezni, ugyanúgy, ahogyan egy letörölt magnószalagról sem.

Ellenőrizzük, jól csináltuk-e. Hívjuk be a lemez tartalomjegyzékét a gép memóriájába. A tartalomjegyzék (angolul: directory, ejtsd: dájrektori) jele a dollár ($), amelyet a 4-es számjegy billentyűjén találunk meg, és Shift-tel hívhatunk elő. Írjuk be tehát a következőket:

```
LOAD "$",8
```

és utána nyomjuK meg a Returnt. (AKi figyelmesen olvasta idáig a Könyvet, annaK a LOAD szó ismerős, de azt is észreveheti, hogy amiKor magnóról töltöttüK be a programot, nem volt ott az a nyolcas számjegy. így igaz: a számítógép ettől a nyolcas számjegytől tudja, hogy ne a magnóról, hanem a drive-ról töltse be a Kért programot.)

Amíg a fenti beKezdést elolvastuK, a számítógép már be is hozta a tartalomjegyzéKet, hiszen az ilyenKor még nagyon rövid; nem tartalmaz mást, mint a lemez nevét és az ID-számot. A Képernyőn tehát ezt látjuK:

```
**************************************
*LOAD "$",8                          *
*SEARCHING FOR $                     *
*LOADING                             *
*                                    *
*READY.                              *
*■                                   *
*                                    *
```

A memóriában van tehát lemezünK tartalomjegyzéKe, amelyet mindjárt el is olvashatunK. Gépeljük be a LIST szót (ugyanaz a LIST ez, mint amelyiKKel a programjainKat listáztathatjuK Ki), vagy nyomjuK meg együtt a Shift és a HELP/F7 gombot.

```
**************************************
*LOAD "$",8                          *
*SEARCHING FOR $                     *
*LOADING                             *
*                                    *
*READY.                              *
*LIST                                *
*0 "[illegible]                      *
*664 BLOCKS FREE.                    *
*                                    *
*READY.                              *
*■                                   *
*                                    *
```

Mit látunk ezen a listán? Először is egy nullát, aminek számunkra nincs jelentősége. Utána idézőjelben, negatív karakterekkel a saját nevünket látjuk - ez mostantól a lemezünk neve is. Ezt követi az ID-szám, már az idézőjelen kívül, majd két karakter, amellyel ugyancsak nem kell törődnünk.

Ezután egy olyan információ következik, amely már nekünk szól: 664 blokk szabad a lemezen: ennyi helyünk van programok felvételére. Hogy pontosan mennyi 664 blokk, azt akkor tudjuk meg, ha fölvesszük 16 soros programunkat lemezre, és megnézzük, az mennyi helyet foglal el.

Gépeljük be a NEW utasítást, hogy kitöröljük a tartalomjegyzéket a memóriából (ha ezt elfelejtjük, a directory összekeveredhet a programunkkal, amit mindjárt beírunk).

Lapozzunk e fejezet elejére, gépeljük be a programot, a biztonság kedvéért futtassuk is le, hogy jól működik-e, aztán írjuk a képernyőre a következőket:

```
SAVE "ELSO PROGRAMOM",8
```

(Látjuk, itt is szerepel a nyolcas, mint ezután is mindig, ha a drive-ot akarjuk dolgoztatni.) Nyomjuk meg a Returnt.

```
**********************************
*SAVE "ELSO PROGRAMOM",8         *
*SAVING ELSO PROGRAMOM           *
*                                *
*READY.                          *
*■                               *
*                                *
```

Ezt is lehet a VERIFY utasítással ellenőrizni (VERIFY "ELSO PROGRAMOM",8), és utána kérjünk ismét tartalomjegyzéket a géptől! Ennek a Plus-4-es gépen (csakúgy, mint a C-16-on) a fentebb leírt módszernél egy egyszerűbb módja is van: nyomjuk meg az F3 feliratú gombot! Ekkor a képernyőnk ilyen lesz:

```
**********************************
*DIRECTORY                       *
*0 "SAJAT NEVUNK     " 01 2A     *
*2 "ELSO PROGRAMOM"       PRG    *
*662 BLOCKS FREE.                *
*                                *
*READY.                          *
*                                *
*                                *
```

Kiderül hát, hogy eléggé pazarló módon bántunK a memóriával: Két blokKot foglaltunK el a lemezen ezzel a programmal. Igaz ugyan, hogy még háromszáz-harmincegyszer fölvehetnénK ugyanezt a programot ugyanerre a lemezre, de azt is tudnunK Kell, hogy ennél jóval rövidebben is meg tudjuK majd írni a programunKat, ha már ügyesebbeK leszünK.

4. NAP — GOTO, AVAGY: MENJ A ...

EmlékszünK, hogy egyszer, a másodiK napon az volt a feladat: rajzoljunK KockÁs füzetlapot a Képernyőre! MegmondtuK előre, hogy több megoldás lehetséges, de mindegyiK magában foglalta azt a Kellemetlen Körülményt, hogy addig Kellett lenyomva tartani valamelyiK grafiKus jel billentyűjét (leginKább a + jelet, Shifttel), ameddig a Képernyő meg nem telt a jeleinKKel. Közben az embernek elzsibbad az ujja. Nagyon csalódnánK a számítógépben, ha ennél egyszerűbb megoldást nem tudna. AKi meg soha nem próbálkozott programozással, az aligha jött rá a leggyorsabb módjára: írjunK erre egyetlen Kétsoros programot!

```
10 PRINT"+"
20 GOTO 10
RUN
```

Mit látunK? A Képernyő bal szélén rettentő gyorsasággal egymás alá íródnaK a + jeleK, és ha nem csinálunK semmit, ez így folytatódiK, amíg ránK nem esteledik, és Ki nem Kapcsoljuk a gépet.

Csináljunk hát valamit: nyomjuK meg a Run/Stop feliratú gombot. A Képernyőn megáll a + jeleK szédült futása, és a legalsó sorokban ezt olvashatjuK:

```
BREAK IN 20 (vagy BREAK IN 10)
READY.
```

SzoKás szerint elmagyarázzuK, mi mit jelent. A 10-es programsorban nincs újdonság. A Kérdőjel a PRINT szó

rövidítése, a pluszhoz hasonló, de annál nagyobb grafikus jel pedig azért van idézőjelek között, mert szöveges változó (azaz string - emlékszünk még?).

Vadonatúj elem van azonban a 20-as sorban. GOTO - mi a manó ez?

A BASICben ez egy utasítás, de az angol nyelven két szó: go to... Utána okvetlenül még valamiNek kell állnia, mert szó szerint ennyit jelent: "menj ...-hez". A három pont helyébe valamit be kell írni, hogy mondat legyen. Mi azt írtuk be: 10. Ez annyit jelent, ebben az esetben, hogy "menj a 10-es sorra". A tízes sorban pedig ugyebár az áll, hogy írj ki egy + jelet.

A számítógép - mint mondtuk - növekvő számsorrendben veszi és hajtja végre a program sorait, és ettől csak külön kérésre hajlandó eltérni. Nos, a GOTO éppen egy ilyen külön kérés. Ha ez nem volna itt, akkor a gép leírt volna egy + jelet, és utána READY-vel jelezte volna, hogy a program végére ért. így azonban a 20-as sorban visszafordult, és ismét végrehajtotta a 10-es sor utasítását.

Ha kilistáztatnánk a programot, és hozzáírnánk egy 30-as sort, például így:

```
10 PRINT"+"
20 GOTO 10
30 ?"SAJAT NEVUNK"
```

és lefuttatnánk, ugyanazt tapasztalnánk, mint az előbb: csak a + jelek sorakoznának egymás alá, a nevünket nem látnánk meg a képernyőn. A gép ugyanis nem jut el a 30-as sorig: a 20-asból visszafordul, és így folytatja, amíg van benne áram.

Amikor a Run/Stop gombot megnyomtuk, megállítottuk a program futását, mert végtelen körbe kergettük a gépet, ahonnan magától nem tudott volna kiszabadulni. Ekkor azt üzente vissza: BREAK IN 20. A BREAK szó itt törést, szakadást jelent, a gép tehát visszajelezte, hogy megszakítottuk a program futását.

Szép, szép, de ez még nem kockás füzetlap, csak annak a bal széle. A gép, valahányszor a 10-es sorhoz ért a program, új sort kezdett a képernyőn, ezért kerültek egymás alá a + jelek. Hogyan lesz ebből egy sorban 40 jel?

Szinte hihetetlenül egyszerűen. Takarítsuk ki a képernyőt (Shift + Clear/Home), és listáztassuk ki a programunkat. Utána lépkedjünk fel a cursorral a 10-es sorba, és a legvégére, mindjárt az idézőjel után írjunk egy pontosvesszőt!

```
10 PRINT"+";
20 GOTO 10
```

És kész! Ha ezt most lefuttatjuk, a számítógép gyönyörű négyzethálót rajzol a képernyőre; fáradhatatlanul, amíg mi meg nem unjuk, és újból meg nem nyomjuk a Stop gombot. Akkor persze megint megkapjuk az üzenetet: BREAK IN 20.

Jegyezzük meg: a pontosvessző (az idézőjelen kívül) a BASIC nyelvben azt eredményezi, hogy a PRINT utasítás után a gép nem kezd új sort, hanem közvetlenül az előzőleg kiírt string után folytatja az írást.

Jó ezt tudni, és előre felkészülni rá. Néha ebből zavarok is támadhatnak. Például...

Listázzuk ki a programot, és javítsuk ki a 10-es sort, hogy a string a saját nevünk legyen!

```
10 PRINT "SAJAT NEVUNK";
20 GOTO 10
```

Mindent értünk: az idezőjelek között a saját nevünk áll, ezért a gép most azt fogja ilyen szorgosan kiírogatni, amíg le nem állítjuk.

Futtassuk le a programot. A képernyő megtelik szöveggel, ami már futás közben is emlékeztet a saját nevünkre, de valahogy fura. Ha leállítjuk a Stop gombbal, meg is látjuk, hogy miért.

```
**************************************
*SAJAT NEVUNKSAJAT NEVUNKSAJAT NEVUNKSAJA*
*T NEVUNKSAJAT NEVUNKSAJAT NEVUNKSAJAT NE*
*VUNKSAJAT NEVUNKSAJAT NEVUNKSAJAT NEVUNK*
*SAJAT NEVUNKSAJAT NEVUNKSAJAT NEVUNKSAJA*
*T NEVUNKSAJAT NEVUNKSAJAT NEVUNKSAJAT NE*
*VUNKSAJAT NEVUNKSAJAT NEVUNKSAJAT NEVUNK*
*                                        *
*BREAK IN 20                             *
*READY.                                  *
*■                                       *
```

Azonnal látható, hogy mi a hiba: csak az első és a második szó között hagytunk szóközt, a második és az első között nem (!). Azt már az első napon láttuk, hogy hogyan lehet szóközt létrehozni olyan sorban, ahol nincs, most frissítsük fel az akkori tudásunkat!

```
10 PRINT "SAJAT NEVUNK";
20 GOTO 10
```

Menjünk fel a cursorral a 10-es sor végére, állítsuk rá a második idézőjelre, aztán nyomjuk meg a Shiftet és az Inst/Del billentyűt. Az idézőjel meg a pontosvessző jobbra húzódik, és egy hézag keletkezik a cursor helyén. Most megnyomjuk a Returnt, és a gép mostantól ezt tekinti a programunk 10-es számú sorának.

Ha lefuttatjuk, azonnal látszik a különbség:

```
**************************************
*SAJAT NEVUNK SAJAT NEVUNK SAJAT NEVUNK S*
*AJAT NEVUNK SAJAT NEVUNK SAJAT NEVUNK SA*
*JAT NEVUNK SAJAT NEVUNK SAJAT NEVUNK SAJ*
*AT NEVUNK SAJAT NEVUNK SAJAT NEVUNK SAJA*
*T NEVUNK SAJAT NEVUNK SAJAT NEVUNK SAJAT*
* NEVUNK SAJAT NEVUNK SAJAT NEVUNK SAJAT *
*                                        *
*BREAK IN 20                             *
*READY.                                  *
*■                                       *
```

Kísérletképpen írjunk egy olyan programot, amely a számítógép grafikai jeleiből igazi kis képecskét rajzol a képernyőre. Ehhez elöljáróban egy pár dolgot még el kell mondani. Többek között egy fontos BASIC utasítást, éspedig azt, hogy REM.

Ez az angol "remark" (megjegyzés) szó rövidítése, és a szerepe nagyon érdekes: ami a programsorban a REM után következik, azzal a számítógép egyáltalán nem törődik. A REM csak a programozó számára hordoz üzenetet, magyarázatot, a számítógép úgy átszalad fölötte, mintha ott se lenne. Ha a REM a programsor elején áll, akkor az egész sort kihagyja a gép a futtatásnál, ha viszont a sor végén, akkor az előtte álló programutasítást végrehajtja, és amikor a

REMmel találkozik, akkor ugrik át a következő sorra. Ne felejtsük el: ebben az esetben a REM , elé kettőspontot kell tenni, hogy elválasszuk az előtte álló utasításoktól.

Ez így eléggé homályosnak tűnhet, de a gyakorlatban mindjárt megvilágosodik. Lássuk a programot, amely egy babafejet fog kirajzolni a képernyőre!

```
10 PRINT"  ●●●●●●●"    :REM SHIFT+Q
20 PRINT" ●●●●●●●●●"
30 PRINT"●●●      ●●●"
40 PRINT"●●         ●●"
50 PRINT"●● ▬▬ ▬▬ ●●" :REM C=+U
60 PRINT"●● ◠  ◠  ●●" :REM SHIFT+U,SHIFT+I
70 PRINT"●● ◡  ◡  ●●" :REM SHIFT+J,SHIFT+K
80 PRINT"●●    o    ●●" :REM SHIFT+W
90 PRINT"●●         ●●"
100 PRINT" ●   ◡    ●" :REM SHIFT+J,SHIFT+*,SHIFT+K
110 PRINT"  \       /"  :REM SHIFT+M,SHIFT+N
```

Ha lefuttatjuk, a baba vigyori képe tűnik fel a képernyőn. Figyeljük meg: ha a képernyő üres, és a bal felső sarokba gépeljuk a RUN szót, akkor a kép szebben, gyorsabban jön elő, mint ha - mondjuk - a program listája alá írjuk a RUNt. Ebben az utóbbi esetben előfordulhat, hogy a fölfelé haladó kép nem is tolja le a képernyőről a teljes listát, hanem valahol alatta rajzolódik ki; röviden: csúnya a képernyő.

Tudjuk már régóta, hogyan lehet a képernyőt kitakarítani, de sokkal elegánsabb volna, ha a program maga végezné ezt el helyettünk. Erre is van mód.

A program legelejére kell írnunk, hogy "töröld tisztára a képernyőt!", aminek szép és elegáns módja van a COMMODORE-on.

Programunk első sora a 10-es számot viseli, tehát (0 és 9 között) tíz különböző sorszámot adhatunk annak a sornak, amely a takarítási feladatot végzi. Legyen ez a 0. Gépeljük be az alábbi sort (mindegy, hogy van-e valami a képernyőn, vagy nincs; az a lényeg, hogy az a sor legyen üres, amelyikbe írunk):

0 ? " Shift+Clear/Home "

Érthető, ugye? Eddig mi magunK nyomtuK meg a Shift és a Clear/Home billentyűt, most ezt a programra bízzuK. (A 0 számú sorban az idézőjeleK Között megint egy negatív KaraKtert látunK: azt a szívet, amely az S betű billentyűjén található.)

Futtassuk le a programot. A babafej most már üres Képernyőre rajzolódiK Ki.

Ha a hibátlannaK találjuK, mentsüK is Ki mindjárt Kazettára, úgy, ahogy az előbb. (A programnaK, tudjuK, nevet Kell adni; nevezzüK így: "BABA".)

Ne szaporítsuK a szót azzal, hogy hogyan Kell magnóra Kimenteni: tesséK visszalapozni, ha valaKi elfelejtette. InKább térjünK rá a KövetKező programra.

```
10 PRINT"■"
20 PRINT"■■■■■■■■" :REM CRSR LE 6*SZOR
30 PRINT"■■■■■■■■■■■■■■■■C" :REM CRSR JOBB 15*SZOR
40 PRINT"■■■■■■■■■■■■■■■■D"
50 PRINT"■■■■■■■■■■■■■■■■M"
60 PRINT"■■■■■■■■■■■■■■■■M"
70 PRINT"■■■■■■■■■■■■■■■■D"
80 PRINT"■■■■■■■■■■■■■■■■D"
90 PRINT"■■■■■■■■■■■■■■■■D"
100 PRINT"■■■■■■■■■■■■■■■■R"
110 PRINT"■■■■■■■■■■■■■■■■E"
120 PRINT"■■■■■■■■■■■■■■■■R"
130 PRINT"■■■■■■■■■■■■■■■■D"
140 PRINT"■■■■■■■■■■■■■■■■D"
150 PRINT"■■■■■■■■■■■■■■■■D"
160 PRINT"■■■■■■■■■■■■■■■■M"
170 PRINT"■■■■■■■■■■■■■■■■M"
180 PRINT"■■■■■■■■■■■■■■■■D"
190 GOTO 30
```

(A cursorbillentyű nyomogatásaKor Különféle negatív KaraKtereK jelennek meg a Képernyőn - a lefelé mozgatásnál egy negatív Q betű, a jobbra mozgatásnál egy szögletes zárójel. Ez normális esemény; csaK aKKor javítsuK, ha nem ezt tapasztaljuK.

Adjunk most egy RUNt - a képernyőn cikcakkos vonalban a COMMODORE szó fut felfelé - amíg meg nem állítjuk.

Tegyük el ezt a programot is szalagon (vagy, aki teheti, lemezen), mert a következő napon megint használni fogjuk. Legyen a program neve (ehhez nem kell nagy lelemenyesség): COMMODORE.

5. NAP — FOR...NEXT: NAGY HALAK ÉS KIS HALAK

Az előző napon nem beszéltünk a levegőbe: kezdjük mindjárt azzal, hogy betöltjük a COMMODORE nevű programunkat.

Emlékszünk rá, ez egy szép cikcakkos COMMODORE feliratot csinált a képernyőn, és sohasem hagyta abba, mert az utolsó sorból visszaküldtük az elejére. Van azonban olyan lehetőség is, hogy csak egy bizonyos, általunk meghatározott számú alkalommal menjen vissza, utána pedig hagyja abba.

Ezt a FOR... NEXT ciklussal érhetjük el.

Most bajban van e sorok írója: ennek a kifejezésnek a pontos magyar fordításával alighanem adós marad. Mindenesetre megpróbáljuk körülírni, de csak akkor, ha már a programunkat átalakítottuk.

```
10 PRINT"■"
20 PRINT"■■■■■■■" :REM CRSR LE 6*SZOR
30 PRINT"■■■■■■■■■■■■■■■■C" :REM CRSR JOBB 15*SZOR
40 PRINT"■■■■■■■■■■■■■■■■■D"
50 PRINT"■■■■■■■■■■■■■■■■■■M"
60 PRINT"■■■■■■■■■■■■■■■■■■M"
70 PRINT"■■■■■■■■■■■■■■■■■■■D"
80 PRINT"■■■■■■■■■■■■■■■■■■■■D"
90 PRINT"■■■■■■■■■■■■■■■■■■■■■D"
100 PRINT"■■■■■■■■■■■■■■■■■■■■■R"
110 PRINT"■■■■■■■■■■■■■■■■■■■■■E"
120 PRINT"■■■■■■■■■■■■■■■■■■■■■R"
130 PRINT"■■■■■■■■■■■■■■■■■■■■■D"
140 PRINT"■■■■■■■■■■■■■■■■■■■■■D"
150 PRINT"■■■■■■■■■■■■■■■■■■■■■D"
160 PRINT"■■■■■■■■■■■■■■■■■■■■M"
170 PRINT"■■■■■■■■■■■■■■■■■■■M"
180 PRINT"■■■■■■■■■■■■■■■■■■D"
190 GOTO 30
```

így fest a program most, ha kilistázzuk. írjunk bele két sort, így:

```
25 FOR N = 1 TO 10
190 NEXT N
```

25-ös sorunk még nem volt, ezért azt a számítógép a 20-as és a 30-as közé fogja beszúrni, a régi 190-es sort pedig egyszerűen eldobja, és a helyére beteszi az újat. (Aggodalomra semmi ok; a régi program attól még megvan a kazettán!)

Mit csinál ez a két sor a programunkban? Megnevez egy változót (egy olyan fogalmat, amelyhez különböző számértékek tartozhatnak): az N-et, és megmondja a számítógépnek, hogy ennek az N-nek az értéke 1-től 10-ig terjedhet. (Azt nem kell külön megmondani, hogy csak egész szám lehet, a gép ezt magától tudja.) A 18-as sor tehát körülbelül így fordítható le emberi magyar nyelvre: "N minden értékére hajtsd végre a FOR és a NEXT közötti utasításokat, miközben az N-et növelgeted 1-től 10-ig." A NEXT utasítás hatására a számítógép visszaugrik arra a sorra, amelyikben a FOR áll. A NEXT angolul annyit jelent: "következő", tehát az N fölveszi a következő értéket, egészen a FOR sorban megjelölt határig, ami a példánkban 10. (Ez a FOR... NEXT is olyan "Külön kérés", amilyen a GOTO, amellyel az előző fejezetben foglalkoztunk.) Az N értékét egyesével végigszámolja a gép, tehát összesen kilencszer fordul vissza a NEXTtől, így fog tízszer végigfutni a programon, utána pedig szépen megáll, és kiírja: READY.

Tudni kell: N helyett írhattunk volna I betűt, vagy G-t, vagy amit akarunk, a lényeg az, hogy betű legyen, ne szám, mert azt SYNTAX ERRORnak, azaz nyelvtani hibának tekinti a gép.

írjunk egy egyszerűbb programot:

```
10 PRINT"{SHIFT+CLEAR/HOME}"
20 FOR F=1 TO 10
30 PRINT"SAJAT NEVUNK"
40 NEXT F
```

Ha lefuttatjuk a programot, tízszer fog egymás alatt megjelenni a nevünk, s alatta a READY szó.

Számoljuk meg, valóban tízszer írta-e ki a gép! Biztos, hogy igen, de jobb lenne, ha a számozást is a képernyőn látnánk. Egyszerű: a 30-as sorban meg kell mondani a gépnek, hogy ne csak a nevünket, hanem az F mindenkori értékét is írja ki. Hogyan kell ezt megmondani? Alakítsuk át a 30-as programsort így:

```
30 ? F; "SAJAT NEVUNK"
```

így lefuttatva a programot, a képernyőn szépen 1-től 10-ig megszámozva ott fogjuk látni a nevünket.

Figyeljük meg: az F-et nem az idézőjelen belül írtuk, hanem eléje, hiszen az F értéke mindig változik, az idézőjelen belüli szöveget pedig a gép mindig változatlan alakban írja ki - ezt is régóta tudjuk már.

Egy jó tanács: ha a program futása közben lenyomjuk a C= billentyűt, a képernyőn lassabban jönnek fel az új sorok, jobban lehet követni a program futását.

Most írjunk egy olyan programot, amelyik 2-től százig kiírja a páros számokat! Hogy hogyan kezdjünk hozzá, azt tudjuk:

```
10 PRINT"■"  :REM SHIFT+CLEAR/HOME
20 FOR I=2 TO 100
30 PRINT I
40 NEXT I
```

Ez biztos. De az is biztos, hogy a számítógép így az összes számot kiírja 2 és 100 között, nemcsak a párosakat. Meg kell mondani neki, hogy minden másodikat hagyja ki, ezt pedig a még eddig nem ismert STEP utasítással lehet az értésére adni. (A STEP egyébként lépést jelent angolul, ami logikus is: 2 lépésenként kell most a kiírást végrehajtania.) Programunk 20-as sorát alakítsuk át így:

```
20 FOR I = 2 TO 100 STEP 2
```

Működik a program, ha lefuttatjuk, de megint a képernyő bal szélére szorítva, egymás alá írja a számokat. Próbáljunk pontosvesszőt tenni a 30-as sor végére!

```
30 PRINT I;
```

Ha most futtatjuk le programunkat, már ráfér egy képernyőre az összes szám, de még mindig nem elég áttekinthető: az első sor elején kis zűrzavar van. Hiába: úgy látszik, mániánk, hogy fontos információkat az utolsó pillanatig visszatartunk. Például azt is elhallgattuk mostanáig, hogy mire szolgál a BASIC nyelvben a vessző. De nem baj, mindjárt meglátjuk. írjunk egy vesszőt a 30-as sorban a pontosvessző helyére!

```
30 PRINT I,
```

Futtassuk le most a programot! Nagyszerű, áttekinthető táblázatot kapunk:

```
**************************************************
*                                                *
* 2            4            6            8      *
* 10           12           14           16     *
* 18           20           22           24     *
* 26           28           30           32     *
* 34           36           38           40     *
* 42           44           46           48     *
```

És így tovább, és így tovább, egészen 100-ig.

Szóval: mit jelent a vessző? A képernyőn 25 sor fér el, és minden sorban 40 karakter (betű, jel, számjegy stb.) írható. Ha vesszőt teszünk a képernyőre kiírandó változó neve után, akkor a gép nem írja mindegyiket új sorba, és nem is közvetlenül az előző után, hanem tízes oszlopokra osztja a képernyőt, és minden PRINT utasítás után a következő ilyen tízes oszlop elejére ugrik, ott folytatja az írást.

(Jó, igaz, nem épp az elejére: mindig kihagy egy karakternyi helyet. Ez az egy karakter azonban az előjel számára kell: ha az előjel pozitív, a gép nem írja ki, ha viszont negatív, akkor mégis kell neki a hely. Aki nem hiszi, járjon utána: javítsa ki a program 20-as sorában a 2 előjelét, legyen I=-2, és kiderül, hogy igazat mondtunk.)

írjunk most olyan programot, amely 0 és 100 között sorolja fel az összes néggyel osztható számot, de fordítva, a legmagasabbtól a legalacsonyabbig! Ez még jó matekosnak is nehézséget okozna, a számítógépnek

azonban semmiség. Csak a programot kell jól megírni hozzá.

```
10 PRINT"■"  :REM SHIFT+CLEAR/HOME
20 FOR Z=100 TO 0 STEP -4
30 PRINT Z,
40 NEXT Z
```

A program működik, pedig a lépésszám most negatív: ezt akkor használjuk, ha a FOR után álló változó két szélső értéke közül az első magasabb.

Írassuk ki a számítógéppel az összes 3-mal osztható számot 666 és 111 között, visszafelé!

```
10 PRINT"■"  :REM SHIFT+CLEAR/HOME
20 FOR I=666 TO 111 STEP -3
30 PRINT I;
40 NEXT I
```

Indítsuk el a programot! A számok követhetetlen gyorsasággal rohannak fölfelé, és csak akkor tudjuk egyáltalán megérteni, hogy miről van szó, amikor a futás a végére ér és megáll - akkor viszont már a képernyő tetején eltűnt egy és más.

Hogyan lehet lelassítani a futást? Egyetlen sort kell beleírni a programba, de az annál tanulságosabb lesz.

```
35 FOR N = 1 TO 100 : NEXT N
```

Hát igen, itt aztán van mit nézni.

Először is: egy programsorba két utasítást is tettünk: egy FORt és egy NEXTet. Eláruljuk: ez teljesen jogos, a Commodore BASIC két teljes képernyősornyi utasítást elfogad egyetlen programsorként; egy a lényeg: hogy az egy sorba írt különböző utasításokat kettősponttal válasszuk el! (Emlékszünk: ezt a REM kapcsán már említettük, de most kiderült, hogy nemcsak a REMre igaz, hanem minden BASIC utasításra.)

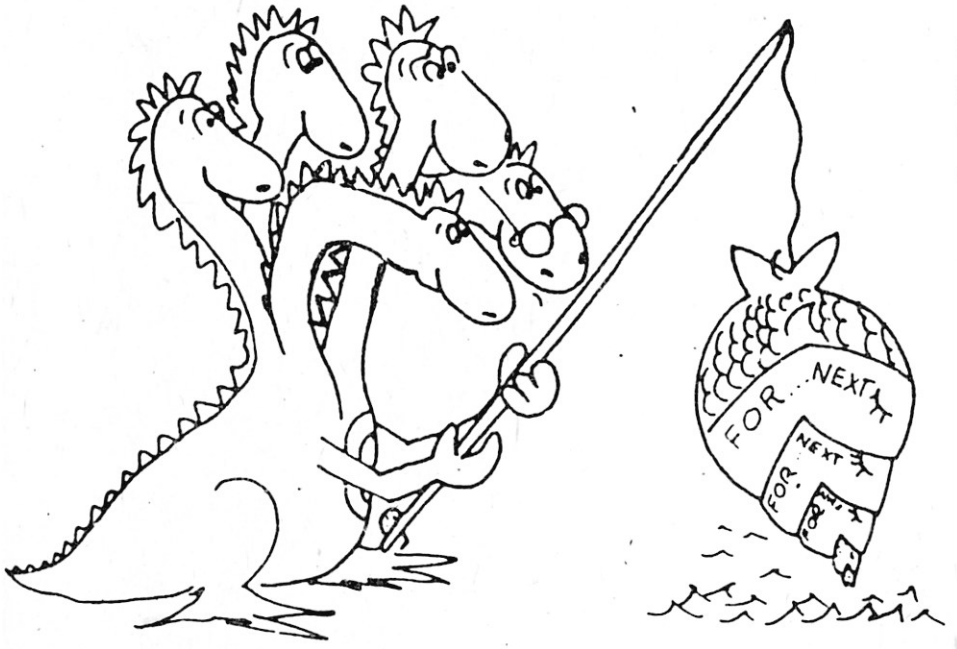
Másodszor: Nem mondtuk meg a számítógépnek, hogy mit akarunk tőle. Csak annyit mondtunk, hogy 1-től 100-ig csináljon valamit, de hogy mit, azt nem részleteztük.

Nos, épp ez a lényeg. Nem akartunk mást tőle, mint hogy százszor menjen vissza a NEXTtől a FORra. Amíg ezt megcsinálja, addig épp elég idő telik el ahhoz, hogy jól megfigyelhessük, amit kiír. Vagy, ha még mindig túl gyors, akkor a 100-ból csinálhatunk 220-at vagy akár 100000-et is.

Vizsgáljuk csak meg a programot!

```
10 PRINT"■"  :REM SHIFT+CLEAR/HOME
20 FOR I=666 TO 111 STEP -3
30 PRINT I;
35 FOR N=1 TO 100:NEXT N
40 NEXT I
```

A 10-es sor ugyebár tisztára söpri a képernyőt. A 20-as megszabja az I értékét, és megmondja, hogy hármasával számoljon a gép visszafelé. A 30-as azt az utasítást adja, hogy a gép írja is ki I értékét, meghozzá egymás mellé. A 35-ös százszorosára növeli azt az időt, amíg a gép eljut a 40-es sorra, ahonnan aztán visszalép a 20-asba. És itt a következő érdekesség. Ha jól látjuk, itt két FOR... NEXT ciklus van, ráadásul az egyik bele van ágyazva a másikba! A FOR I - NEXT I ciklusON belül van a másik, a FOR N - NEXT N.



Jól látjuk, és ez így van rendjén. Két, sőt több FOR... NEXT ciklust is lehet (sőt néha muszáj) használni a BASIC programokban, de lényeges, hogy ezek mindig zártak legyenek, ne kerüljön egyik NEXT se kívül azon a ciklusON, amelyiken belül van a hozzá tartozó FOR. Valahogy olyasféleképp kell ezeket szerkeszteni, ahogyan a nagy halak megeszik a kis halakat.

```
FOR A=N TO...
  FOR B=N TO...
    FOR C=N TO...
      FOR D=N TO...
      NEXT D
    NEXT C
  NEXT B
NEXT A
```

És most jöjjön egy feladat: gépeljük be az itt következő programot, és mielőtt lefuttatnánk, próbáljuk meg kitalálni, hogy mit rajzol!
(Figyeljük meg, hogy itt már több, logikailag összetartozó utasítást írtunk egy sorba, így valamicske memóriát takarítunk meg - és persze, ezúttal, papírt is...)

```
10 PRINT"■"
20 PRINT"      ╲╱"
30 PRINT"     ┌──┐"
40 PRINT"     |..|"
50 PRINT"     └──┘"
60 PRINT"      ▓"
70 FOR X=1 TO 5:PRINT"    *▓▓▓▓▓*":NEXT X
80 PRINT"     ▓▓▓▓▓"
90 FOR Z=1 TO 4:PRINT"     *   *":NEXT Z
```

És még egy ugyanilyen feladat: ideírunk egy programot, és nem mondjuk meg előre, hogy mit csinál. Mindenki próbálja meg kitalálni, aztán gépelje be, futtassa le, és meglátja, hogy helyesen tippelt-e.

```
10 PRINT"■"
20 FOR I=1 TO 20
30 PRINT I "*" I "=" I*I
40 NEXT I
```

Egy olyan program következik, amelyik egy labdát mozgat a képernyőn balról jobbra. Hosszú lesz, de a begépeléséhez tudunk már egy fogást, amellyel az ismétlődő sorokat csak egyszer kell beírni, utána mindig csak a sorszámot javítjuk, és a Return megnyomása után a gép az új sorszámmal hozzáírja a sort a programhoz, de a sor a régi sorszámmal is megmarad.

```
10 REM GOLYO
20 PRINT"■"
30 PRINT"●"
35 FOR I=1 TO 100:NEXT I:PRINT"■"
40 PRINT"   ●"
45 FOR I=1 TO 100:NEXT I:PRINT"■"
50 PRINT"      ●"
55 FOR I=1 TO 100:NEXT I:PRINT"■"
60 PRINT"         ●"
65 FOR I=1 TO 100:NEXT I:PRINT"■"
70 PRINT"            ●"
75 FOR I=1 TO 100:NEXT I:PRINT"■"
80 PRINT"               ●"
85 FOR I=1 TO 100:NEXT I:PRINT"■"
90 PRINT"                  ●"
95 FOR I=1 TO 100:NEXT I:PRINT"■"
100 PRINT"                     ●"
105 FOR I=1 TO 100:NEXT I:PRINT"■"
110 PRINT"                        ●"
115 FOR I=1 TO 100:NEXT I:PRINT"■"
120 PRINT"                           ●"
```

Ebben a programban tulajdonképpen csak egyszer kell megírni a "fékező" sort, mert mindig ismetlődik, ezért elég a sorszámát növelni. A golyót kirajzoló sor is egyszerűen írható újra meg újra: tízzel megnöveljük a sorszámát, utána a cursorral rászaladunk a golyóra, és háromszor megnyomjuk a Shift+Inst/Del billentyűt, majd a Returnt. Ezt jó néhányszor megismételhetjük, a golyó mindig három hellyel jobbra kerül. Ha finomabbra akarjuk állítani a mozgását, akkor három helyett egy hellyel tegyük odább - viszont akkor a programunk háromszor olyan hosszú lesz.

A Commodore Plus 4 sok tekintetben többet tud, mint népszerű rokona, a Commodore 64-es. így van ez például a színekkel is.

A C 64-es 16 szín megjelenítésére képes, a Plus 4-nek ugyanennyi színe van, de majd mindegyik színnek nyolc különféle árnyalata közül válogathatunk.

Gépeljük be ezt a sort:

```
COLOR 4,15,6
```

és utána nyomjuk meg a Returnt!

Mi változott? Semmi. A képernyőn minden maradt, amilyen volt - pedig egy részletes és pontos utasítást adtunk.

Nézzük meg alaposabban, miből állt ez az utasítás!

Először is a COLOR szóból. Aki iskolában tanulta az angolt, annak feltűnhet, hogy a 'szín' jelentésű szót 'colour'-nak tanították, itt meg az "u" betű hiányzik. Ennek egy oka van: a gépben található BASIC szótár az amerikai angolon alapul, nálunk az iskolában pedig brit angolt tanítanak. Amerikában a colour szóból elhagyják az u-t, így a szó pontosan úgy íródik, mint a latin eredetije.

Szóval: a COLOR azt jelenti: szín. Az utána következő 4-es számjegy pedig azt jelenti: az utasítás a keretszínre vonatkozik. Hogy ez miért pont 4, most ne firtassuk - egyelőre untig elég, ha megjegyezzük.

Következik a 15-ös szám: ez a színnek, a sötétkéknek a száma. Igazat mond, aki szerint ez nem sötét, hanem világoskék; ennek is megvan a maga oka.

Az ok: a 6-os számjegy. Ez ugyanis a 15-ös számú, 'sötétkék' nevű színnek egy világosabb árnyalata. A gép konstruktőrei ezt a keretszínt találták alkalmasnak arra, hogy a bekapcsoláskor bejelentkezzék.

Egyáltalán nem kötelező azonban állandóan ezt a színt bámulnunk: ha visszamegyünk a cursorral a COLOR 4,15,6 sorra, és a 6-ot átírjuk 7-re, majd megint megnyomjuk a Returnt, a keret színe egy árnyalattal világosabbra változik. A 'sötétkék' színnek ez a legvilágosabb árnyalata - ezt is bízvást nevezhetnénk világoskéknek, de hát világoskéknek a 14-es számú színt nevezi a színlista; ezen csak nem fogunk összeveszni a tervezőkkel. Végül is nem az a fontos, hogy mi a neve a színnek, hanem az, hogy megvan.

Most menjünk fel megint a cursorral, és javítsuk át a 7-es számjegyet 8-asra! A Return megnyomása után a számítógép a következő méltatlankodó üzenettel válaszol:

?ILLEGAL QUANTITY ERROR

azaz a 'nem megengedett mennyiség'-nek nevezett hibát észleli.

Igaza van a gépnek: a színeknek valóban 8 árnyalatuk van, viszont ebből a legsötétebbet 0-val jelezzük, a legvilágosabbat 7-tel. 8-as árnyalatot nem fogad el.

Tudni kell még azt is, hogy ez alól a fekete szín kivétel: ott 0-tól 7-ig egy és ugyanaz az árnyalat szerepel: a fekete. Jó logikával ebből azt is ki lehet találni, hogy a szürkéket a fehér szín árnyalatai között találjuk meg.

Menjünk vissza a cursorral, és javítsuk ki az illegális 8-as számjegyet 0-ra! Ha ezután megnyomjuk a Returnt, meglátjuk a sötétkék szín legsötétebb árnyalatát a kereten.

Most, hogy már ismerjük a FOR...NEXT ciklus rejtelmeit, nézzük végig az összes színt a képernyőn. írjuk be ezt a programot:

```
10 FOR SZ=1 TO 16
20 FOR A=0 TO 7
30 COLOR 4,SZ,A
40 NEXT A
50 NEXT SZ
```

Ez a program végigpörgeti az összes színt a képernyő keretén, mégpedig villámgyorsan, hogy a szemünk is

Káprázik bele. Lassítsuk le a futását: írjunk bele egy késleltető sort a 30-as és 40-es közé.

```
35 FOR K=1 TO 200:NEXT K
```

Ha most nézzük meg a program futását, megfigyelhetjük, hogy az elején nagyon sokáig fekete a keret. Joggal: a fekete szín árnyalatai között nincs különbség. Utána azonban következnek a többi színek sorjában - nem épp szabályosan, például a szivárvány színeinek sorrendjét követve, hanem eléggé összevissza. Nehéz eligazodni köztük, ezért tanácsos egy táblázatot összeállítani a színek kódszámáról. íme:

SZÍNKÓD

1 FEKETE	9 NARANCS
2 FEHÉR	10 BARNA
3 PIROS	11 SÁRGÁSZÖLD
4 CIÁNKÉK	12 RÓZSASZÍN
5 BÍBOR	13 KÉKESZÖLD
6 ZÖLD	14 VILÁGOSKÉK
7 KÉK	15 SÖTÉTKÉK
8 SÁRGA	16 VILÁGOSZÖLD

A HÁTTÉRSZÍN MEGHATÁROZÁSA

```
COLOR 0,SZÍNKÓD,ÁRNYALATKÓD
```

A KERETSZÍN MEGHATÁROZÁSA

```
COLOR 4,SZÍNKÓD,ÁRNYALATKÓD
```

A CURSOR SZÍNÉNEK MEGHATÁROZÁSA PROGRAMBAN

```
COLOR 1,SZÍNKÓD,ÁRNYALATKÓD
```

A CURSOR SZÍNÉNEK MEGVÁLTOZTATÁSA PROGRAMON KÍVÜL

Control SEGÉDBILLENTYŰVEL	C= SEGÉDBILLENTYŰVEL
1 FEKETE	1 NARANCSSÁRGA
2 FEHÉR	2 BARNA
3 PIROS	3 SÁRGÁSZÖLD
4 CIÁNKÉK	4 RÓZSASZÍN
5 BÍBOR	5 KÉKESZÖLD
6 ZÖLD	6 VILÁGOSKÉK
7 KÉK	7 SÖTÉTKÉK
8 SÁRGA	8 VILÁGOSZÖLD

Ezeket a táblazatokat a könyv végén is megtalálhatjuk; mire odáig elérünk az olvasásban (meg a gyakorlásban), már sokkal átteKinthetőbb lesz. Most csak azért írtuk ide, hogy a következő játékhoz ne kelljen sokat lapozgatni.

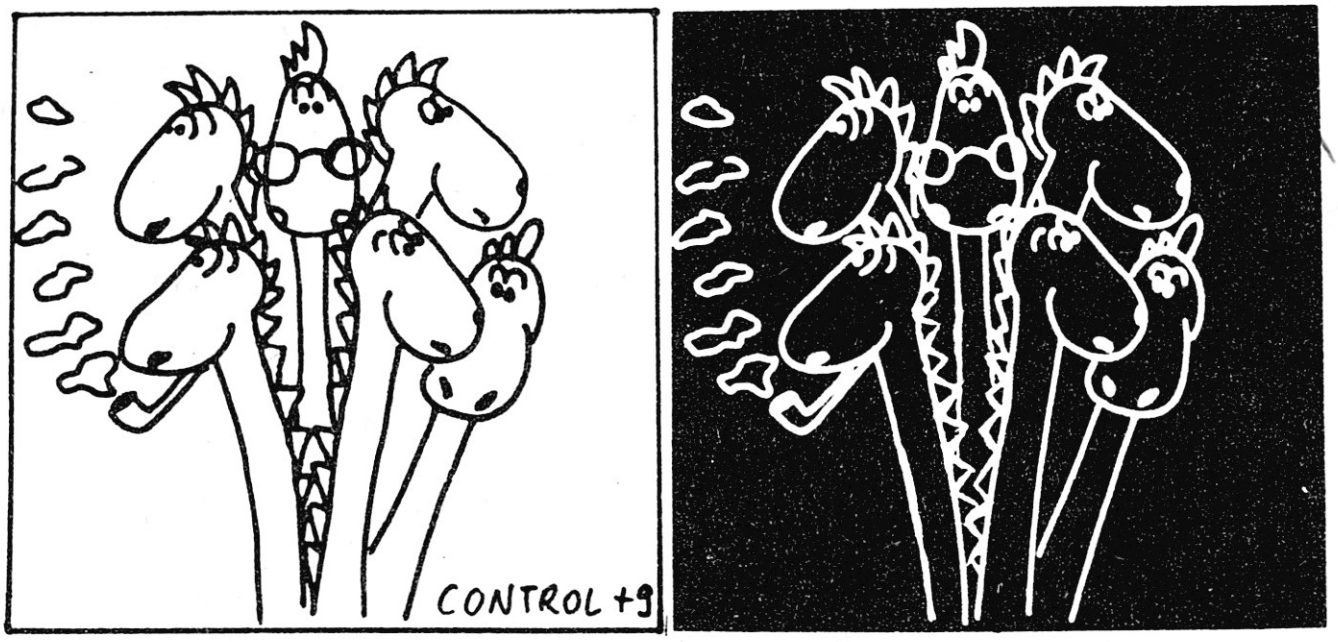
A játék lényege: írjuk fel a képernyő bal felső sarkába a képernyő színének megváltoztatására szolgáló jelszót: COLOR 0, (figyeljünk a vesszőre!) - aztán írjunk be egy tetszés szerinti számot 1 es 16 között! Utána egy darabig ne nyomjuk le a Return billentyűt, hanem próbáljuk meg kitalálni (persze puskázás nélkül), hogy milyen színt kódoltunk be. Ezt ketten is játszhatják - az egyik begépeli a színkódszámot, a másik megpróbálja kitalálni a színt. Ha megmondta, hogy szerinte a beírt szám milyen színt jelent, megnyomjuk a Return gombot, és a gép azonnal igazolja, hogy helyes volt-e a tipp.

Egy másik játék: az egyik játékos felírja egy cédulára, hogy milyen színű legyen a képernyő, milyen a keret, milyen színű a cursor, a másik játékos pedig begépeli a szükséges adatokat. Ellenőrzik, hogy helyesen gépelt-e, aztán cserélnek. Az győz, aki rövidebb idő alatt tudja teljesíteni a partner kívánságát.

Újabb játék következik.

Takarítsuk ki a képernyőt (Shift+Clear/Home), és állítsuk be az eredeti képernyőszíneket (COLOR

4,15,6 es COLOR 0,2,7: a cursort pedig a Control és az 1-es számjegy billentyűjének együttes megnyomásával állíthatjuk feketére). Nyomjuk meg most a Control feliratú gombot, és vele együtt a 9-es számjegy billentyűjét!



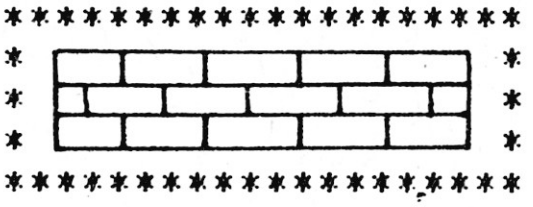
Látszólag nem történt semmi. De próbáljunk meg begépelni valamilyen szöveget! Látni fogjuk, hogy a betűk nem úgy jelennek meg, ahogyan megszoktuk: fekete színben, hanem sötét alapon, világosan - más szóval negatívban. (Aki foglalkozott már fotózással, az tudja, mit jelent a 'negatív' szó: azt, hogy ami a valóságban világos, az sötét lesz és megfordítva.) Ezek a negatív jelek (angolul inverse vagy reverse - a 9-es számjegy és a 0 billentyű felirata is ez utóbbi szó rövidítése látható) számtalan alkalommal lesznek segítségünkre.

Inverz módban vagyunk tehát, egy idő óta a cursor színét is tetszés szerint tudjuk változtatni - semmi akadálya tehát hogy szép szivárványszínű csíkokat húzzunk a képernyőn. Ehhez nem kell más, mint megnyomni és nyomva tartani a szóközbillentyűt. Ha a cursor világoskék volt, akkor a csík is világoskék lesz, de ha bárhol megváltoztatjuk a színét, akkor attól a helytől kezdve az új színnel folytatódik. Ha pedig meguntuk a negatív karaktereket, a Control és a 0 billentyű együttes megnyomásával visszaléphetünk a megszokott módba.

Nagyon csalódnánk, ha erre nem lehetne valami játékot kitalálni. Az egyik legegyszerűbb: rajzolja meg ki-ki a képernyőn a magyar zászlót! (Inverz módba lépünk: a Control és a 9-es billentyűt nyomjuk meg egyszerre; utána pirosra változtatjuk a cursort: Control és a 3-as billentyű; aztán a szóközbillentyűt addig tartjuk nyomva, ameddig a piros sáv el nem foglalja a képernyő egyharmadát. Ekkor - természetesen egy sor elején - fehérre változtatjuk a cursort: Control és a 2-es billentyű; és a képernyő középső harmadát átszínezzük fehérre. Az utolsó harmad elején zöldre változtatjuk a cursort: Control és a 6-os billentyű, és a képernyő utolsó sorait zöldre színezzük. Vigyázzunk arra, hogy az utolsó előtti helyen álljunk meg a cursorral, mert ha a képernyő utolsó betűhelyét is betöltjük, akkor a képernyő tartalma két sornyit megemelkedik, és fölül a fáradságosan kifestett piros sorainkból kettő visszahozhatatlanul elvész.

A magyar zászló után természetesen megpróbálkozhatunk más országok zászlóival is; a negatív és pozitív jelek ügyes váltogatásával, keverésével egészen bonyolult grafikákat is készíthetünk - akár a libanoni zászlón szereplő cédrust is lerajzolhatjuk. (Emlékezzünk vissza az első napon tárgyalt grafikus jelekre, amelyek a Shift és a C= billentyű segítségével csalhatók elő - ezeket bátran használhatjuk mind pozitív, mind negatív változatban!)

Egy másik játék: rajzoljunk egy téglafalat a képernyőre piros téglákból, fehér habarccsal.



Milyenre kell változtatni a képernyő színét, milyenre a cursorét? Magyarán: piros alapon fehérrel vagy fehér alapon pirossal érdemes elkezdeni a rajzolást?

ÖSSZEFOGLALÓ KÉRDÉSEK

Mi történik, ha egyszerre megnyomjuk a

Shift és a C= billentyűt?

a. A képernyő színe megváltozik?

b. A nagybetűk kisbetűvé változnak és megfordítva?

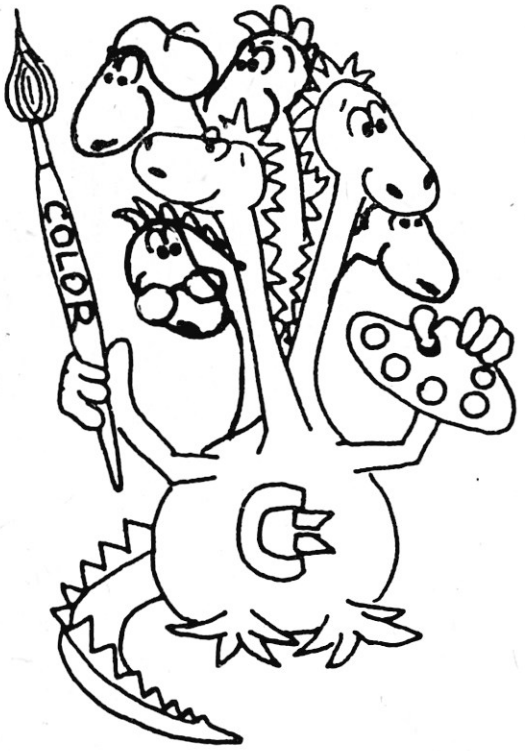
c. A nagybetűk grafikus jelekké változnak?

Control és Rvs On billentyűt?

a. Ettől kezdve negatív jeleket ír a gép?

b. A cursor színe megváltozik?

c. A képernyőről minden jel eltűnik, és a cursor alaphelyzetbe ugrik?



A hetedik nap a pihenésé - ez érvényes erre a Könyvre is. Ha valaki meg tudja állni, hogy ezen a napon bekapcsolatlanul hagyja a gépet, ám tegye. Bár nemigen hisszük, hogy lesz ilyen...

Az ötödik napon rajzoltunk egy robotot (ez volt az egyik feladat), most próbáljuk meg megmozgatni, megtornásztatni!

```
10 COLOR4,7,2  :REM KEK LESZ A KERET
20 COLOR0,7,2  :REM KEK LESZ A HATTER
30 PRINT"■"    :REM FEHER LESZ A ROBOT
40 PRINT"▒"    :REM URES A KEPERNYO
50 PRINT"       ╲/" :REM SHIFT+M,SHIFT+N
60 PRINT"      ┌──┐":REM SHIFT+U,SHIFT+*,SHIFT+I
70 PRINT"      |..|":REM SHIFT+-
80 PRINT"      └──┘":REM SHIFT+J,SHIFT+*,SHIFT+K
90 PRINT"       ■" :REM C=+L,C=+J
100 FOR X=1 TO 5:PRINT"    *▒▒▒▒▒▒*":NEXT X:REM C=++
110 PRINT"      ▒▒▒▒▒▒"
120 FOR Z=1 TO 4:PRINT"     *  *":NEXT Z
130 FOR I=1 TO 200:NEXT I
140 PRINT"▒"
150 PRINT"       ╲/"
160 PRINT"      ┌──┐"
170 PRINT"      |..|"
180 PRINT"      └──┘"
190 PRINT"       ■"
200 PRINT"*****▒▒▒▒▒*****"
210 FOR Z=1 TO 5:PRINT"     ▒▒▒▒▒▒":NEXT Z
220 PRINT"      *  *"
```

```
230 PRINT"    *    *"
240 PRINT"   *      *"
250 PRINT"  *        *"
260 FOR I=1 TO 200:NEXT I
270 GOTO 40
```

Ennek a programnak minden sorát ismerjük már, mindegyikről tudjuk, mit csinál, és miért csinálja.

El lehet játszani vele, hogy lassabban vagy gyorsabban tornázzon a robotunk: ha átírjuk a "fékező" sorokban (amelyekben a FOR és a NEXT között nincs más utasítás) a változó értékének felső határát; meg lehet csinálni, hogy a robot torna közben kacsintgasson (ha a szemét az egyik mozgásfázisban kicseréljük a W billentyűn található, Shifttel előhívható grafikus jelre), és még sokféle változtatást tudunk végrehajtani ezen az egyszerű programocskán. A legcélravezetőbb, ha céltudatosan, sorról sorra végigolvassuk, és megpróbálunk a lehető legtöbb helyen változtatni rajta, hogy a végén a tulajdon édesanyja se ismerjen rá!

És ha már minden ízében kicseréltük, próbáljuk meg átalakítani mondjuk úgy, hogy egy fülét-farkát billegető kutyát rajzoljon!

A következő program egy változékony kedélyállapotú baba: hol jókedvű, hol meg rossz. Próbáljuk meg úgy átírni a programot, hogy többet legyen jókedvű!

```
10 PRINT"■"
20 COLOR4,4,4:COLOR0,2,7:PRINT"■■":REM CONTROL+1,CRSR ↑
30 PRINT"  ●●●●●●●"
40 PRINT" ●●●●●●●●●"
50 PRINT"●●●      ●●●"
60 PRINT"●●         ●●"
70 PRINT"●●  ▬|▬   ●●"
80 PRINT"●●         ●●"
90 PRINT"●●  O O   ●●"
100 PRINT"●●   ●    ●●"
110 PRINT"●●         ●●"
120 PRINT" ●   ‿    ●"
130 PRINT"  \\    //"
```

```
140 PRINT"    |‿|"
150 FOR N=1 TO 500:NEXT N
160 PRINT"■":REM HOME
170 PRINT"  ●●●●●●●"
180 PRINT" ●●●●●●●●●"
190 PRINT"●●●      ●●●"
200 PRINT"●●        ●●"
210 PRINT"●● ▬▬ ▬▬ ●●"
220 PRINT"●●  ◠  ◠  ●●"
230 PRINT"●●  ‿  ‿  ●●"
240 PRINT"●●    ●   ●●"
250 PRINT"●● ╯    ╰ ●●"
260 PRINT" ●   ‿‿   ●"
270 PRINT"  \      /"
280 PRINT"    |‿‿|"
290 FOR N=1 TO 500:NEXT N
300 PRINT"■"
310 GOTO 30
```

A következő két program szép színes hasábokat rajzol fel - először keskenyebb, aztán egész képernyő szélességű változatban.

```
10 PRINT"■"
20 FOR A=2 TO 16
30 FOR B=7 TO 1 STEP -1
40 COLOR1,A,B
50 PRINT"■                    ":REM RVS ON
60 NEXT B
70 FOR N=1 TO 200:NEXT N
80 NEXT A
```

Ez volt az első; most jön a második.

```
10 PRINT"■"
20 FOR A=2 TO 16
30 FOR SZIN=2 TO 16
40 FOR ARNYALAT=0 TO 7
50 COLOR1,SZIN,ARNYALAT
60 FOR N=1 TO 3
65 PRINT"■
```

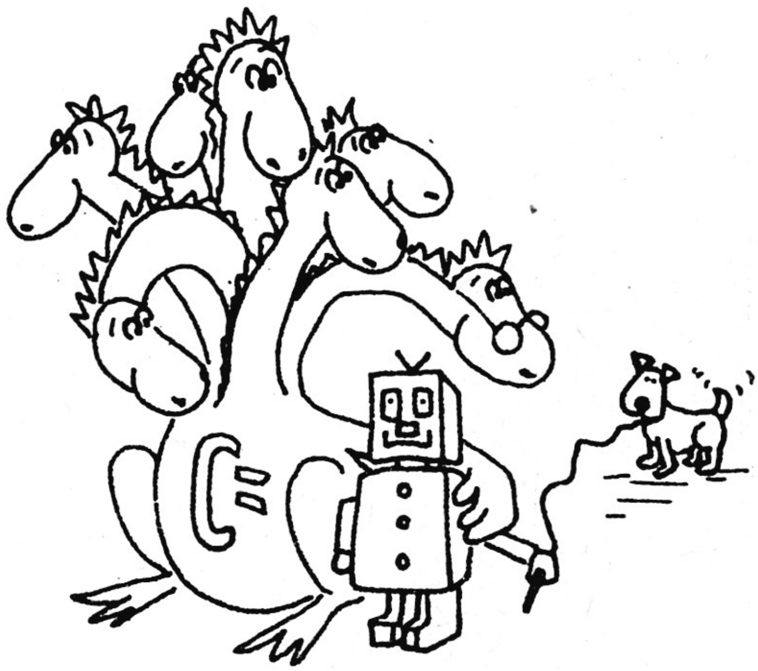
```
67 NEXT N
70 NEXT ARNYALAT
80 PRINT"■"
90 FOR K=1 TO 1000:NEXT K
100 NEXT SZIN
```

Az utolsó program egy játékot mutat be. Felrajzol egy szép, szivárványos színösszeállítást, aztán eltünteti, és nekünk a színek csereberélgetésével kell helyreállítanunk az eredeti sorrendet. Ezt a programlistát szokásunktól eltérően kisbetűvel közöljük, mert a képernyőfeliratokon nagy-és kisbetűk egyaránt szerepelnek.)

```
10 scnclr:color4,1:printchr$(14):color0,1
20 x=1:z=0:dim s(16)
30 for i=1 to 14:read s(i):next i
40 for b=6 to 0 step-1:for i=1 to 3
50 color1,x,b:print"■  "+"■■■■";
60 for a=1 to 14
70 color1,s(a),b:print"■ ";
80 next a:print:next i:next b
90 color1,2,4:if z=1 then 160
100 print:print"Ezt csinald utanam !"
110 print"Nezd meg, es ha megjegyezted,"
120 print"nyomd meg a szokozbillentyut!"
130 get a$:if a$<>" " then 130
140 for i=1 to 14:s(i)=i+2:next i
150 z=1:print"■";:goto 40
160 restore
170 fori=1 to 14:read d
180 if s(i)<>d then 200
190 next i:goto 320
200 print"■ X   ";
210 for i=193 to 206:print" "+chr$(i);:next
220 print:print
230 print"uss egy betut A-tol N-ig !";
240 geta$:if a$="" then 240
250 if a$<"a" or a$>"n" then 160
```

```
260 print"██████████████████████████████████";
270 print"                                  ";
280 y=s(asc(a$)-64):s(asc(a$)-64)=x:x=y
290 print"█";:goto 40
300 data 15,7,14,4,13,6,16,11,8,10,9
310 data 3,12,5
320 print:print
330 print"GRATULALOK ! EZ AZ !"
340 goto 340
```

A TöBBIT A KöVETKEZő KöTETBEN MONDJUK EL.



MÁSODIK HÉT

Beszéltünk már a változókról, amikor a FOR...NEXT ciklusról volt szó.

Létezik még kétfajta változó, úgyhogy ideje rendet teremteni köztük. (Ez most matek lesz, de nem nagyon veszélyes.)

Kezdjük egy kicsi programmal.

```
10 LET A=5
20 ? A
RUN
```

(A 10-es sor, ha angol nyelvű mondatnak értelmezzük, azt jelenti: LEGYEN A 5-tel egyenlő. Ez a LET a Commodore BASICben elhagyható, de mi az áttekinthetőség kedvéért igyekszünk rendszeresen használni.)

A gép kiírja az 5-ös számot, és azt mondja: READY.

Itt készítettünk egy változót: A lett a neve, és mindjárt értéket is adtunk neki: 5. Ha ezt a programot most tovább írnánk, az A betű mindig ötöt jelentene, egészen addig, amíg más értéket nem adunk neki.

Sok értelme ennek így önmagában nincs, hiszen az A betűt leírni ugyanakkora fáradság, mint az 5-ös számjegyet. Tudni kell azonban, hogy nem muszáj nekünk előre megadnunk a változónk értékét; rábízhatjuk ezt a gépre.

```
10 LET A=5
20 LET B=4
30 LET C=A+B+76
40 ? C
```

RUN után a program kiírja a 85-öt. Ez még mindig nem több, mint amit egy átlagos nyolcéves gyerek egy zsebszámológéppel meg tud csinálni, de csak nyugalom, ez csupán az alap, ennél sokkal komplikáltabb is lesz még. Például a következő programban:

```
10 LET A$="AGOL"
20 LET B$="B"
30 LET C$="Y"
40 LET D$=B$+A$+C$
50 ? D$
```

Ezeket a változókat szöveges (string) változóknak nevezzük. Programunkból kiderül, hogy a szöveges változókat ugyanúgy összekapcsolhatjuk + jellel, mint a - nevezzük nevén - valós változókat.

Kiderül továbbá az is, hogy a szöveges változók elnevezésében szerepelnie kell a $ (dollár) jelnek, hogy megkülönböztessük a többi változófajtától. Próbáljuk meg kitörölni a $ jelet a program 30-as sorából, és futtassuk le: TYPE MISMATCH ERROR IN 30 - fogja üzenni a gép, ami azt jelenti, hogy nem olyan karaktert kapott, amilyet várt. Ha nincs ott a $ jel, akkor a gép számjegyet vár az egyenlőségjel után, és helyette betűt, azaz szöveges változót kapott.

Természetesen nemcsak betűből állhat a szöveges változó, hanem grafikus és írásjelekből (kivéve az idézőjelet), meg szóközökből és számjegyekből is, de akkor is idézőjelbe foglaljuk, tehát ilyenkor is kell a $ jel.

```
10 LET A$="19"
20 LET B$="86"
30 ? A$;B$
```

Ekkor a gép semmiféle műveletet nem végez a számjegyekkel, hanem egyszerűen kiírja őket - a pontosvessző miatt szorosan egymás mellé -: 1986. (A pontosvessző helyére írhattunk volna + jelet is, vagy éppen semmit, az eredmény ugyanaz lenne.)

A változókat elnevezhetjük sokféleképpen, de azért vigyázni kell bizonyos dolgokra. Például ezt a három változónevet nem használhatjuk akármire: TI, TI$ és ST. Ezeket a gép különleges célokra tartogatja - némelyikről később még lesz szó.

De van még más is, amire ügyelnünk kell.

```
10 LET KABAT$="KABAT"
20 LET KALAP$="KALAP"
30 ? KABAT$,KALAP$
```

Futtassuk le a programot. A gép megbokrosodott! Ezt írja ki:

```
KALAP     KALAP
```

Persze nem a gép a vétkes: mi követtünk el hibát - de hát nem tudtunk egy fontos szabályt: azt, hogy adhatunk jó hosszú neveket is a változóinknak, a számítógép azonban csak az első két karaktert veszi figyelembe, a többit nem. A 10-es sorból csak ennyit érzékel: KA$="KABAT". A 20-asból meg ezt tudja meg: KA$="KALAP". És - mint a katonaságnál - itt is az utolsó parancs az érvényes, ezért a 20-as sort olvasva a gép kicseréli a 10-es sor információját. Vigyázzunk tehát a változók elnevezésénél arra, hogy ugyanazzal a két betűvel ne kezdjünk nevet, mert a későbbi felülírja az előbbit!

```
10 LET MARCIPAN$="MARCIPAN"
20 LET MAZSOLA$="MAZSOLA"
```

Ez így nem jó. Úgy viszont már jó, ha azt mondjuk:

```
10 LET M1ARCIPAN$="MARCIPAN"
20 LET M2AZSOLA$="MAZSOLA"
```

A gép ugyanis ebben az esetben csak az M1-et és az M2-t figyeli, a többivel nem törődik.
Bizonyítsuk be.

```
10 LET M1ARCIPAN$="MARCIPAN"
20 LET M2AZSOLA$="MAZSOLA"
30 ? M2$,M1$,M1$,M2$
RUN
```

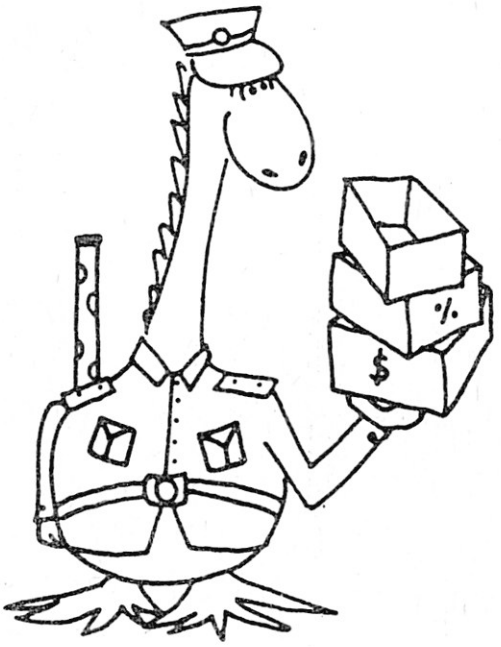
```
MAZSOLA   MARCIPAN  MARCIPAN  MAZSOLA
```

összefut az ember szájában a nyál.

És ezen kívül is kell még egy dologra vigyázni. Arra, hogy a változók elnevezésében az első karakter mindig betű legyen, különben SYNTAX ERROR üzen a gép.

A harmadik változótípus az egész változó. Ezt az különbözteti meg a valós változótól, hogy csak egész szám lehet az értéke, törtszám nem. Ezt a típust ritkán használják a BASIC programokban, helyette is inkább a valós változó szerepel. Mint a stringnek, ennek is van egy megkülönböztető jele: a %

Tegyük egymás mellé a három típust, hogy tudjuk, melyik milyen.



TÍPUS	JELE	PÉLDA
Valós	nincs	7.84
Egész	%	103
String	$	"KALAP"

(Emlékezzünk csak vissza: a BASIC nyelvben a tizedestörtekben nem vesszőt használunk, hanem pontot!)

Tudnivaló még: egyazon programon belül különböző típusú változokat elnevezhetünk ugyanúgy - az ABC$ és az ABC két külön fogalom a számítógép számára; akármilyen hosszú nevet adunk a változónknak, a jel eligazítja a gépet, ebből nem lesz kavarodás. Legföljebb mi kavarodunk bele, ha hosszabb, sokváltozós programot írunk.

összefoglalásul: a változókat képzeljük úgy, mint egy-egy üres dobozt, amelyet mi töltünk meg tartalommal a programozás során, és a tartalmát bármikor ki is cserélhetjük.

Sőt, most megismerkedünk egy új fogással: hogy a program futása közben is csereberélhetjük a dobozok tartalmát. Ehhez egy új BASIC utasítást kell megtanulnunk, éspedig ezt: INPUT. Megint csak angol szó, azt jelenti: beadás, bevitel. Ha a számítógép ilyen utasítással találkozik, akkor megállítja a program futását, egy kérdőjelet rajzol a képernyőre, és várja, hogy mi beírjunk valamit. A program csak akkor indul tovább, ha megnyomjuk a Returnt.

```
10 INPUT A$
20 LET B$="LOM"
30 LET C$=A$+B$
40 ? C$
```

Ez a program azt próbálja ki, hogy hány "-LOM" végű szót ismerünk. Futtassuk le!

A RUN után a képernyő bal szélén megjelenik egy kérdőjel. Gépeljük be: IRODA, és nyomjuk meg a Returnt.

A gép kiírja: IRODALOM.

Ha újra futtatjuk, és a kérdőjel után azt írjuk: ALKA, az ALKALOM szó alakul ki a Return megnyomása után.

Több INPUT utasítást is írhatunk ugyanabba a programba. Javítsuk ki az előzőt így:

```
10 INPUT A$
15 INPUT Q$
20 LET B$="LOM"
30 C$=A$+B$+Q$
40 ? C$
```

A RUN után megjelenik a kérdőjel, írjuk be: AKAC. A Return megnyomása után egy újabb kérdőjel tűnik fel az előző alatt. Ide írjuk azt: BOS. A Return után a gép felelete: AKACLOMBOS.

Nézzük át tehát, milyen neveket adhatunk változóinknak, mik ennek a szabályai.

PRINT = 195 (Nem jó név, mert a PRINT egy létező BASIC utasítás, és az ilyenek foglaltak)

H1 = 407 (Jó név, mert az első karaktere betű)

1H = 26 (Nem jó, mert az első karakter nem betű)

FORM = 198 (Nem jó, mert benne van egy BASIC utasítás, a FOR)

PR = 505 (Jó, mert a PRINT is így kezdődik ugyan, de a változónévben nincs benne az egész BASIC szó)

TI$ = "HUSZONEGY" (Nem jó, mert foglalt; egyike a három névnek, amit nem használhatunk.)

SNRDLUYWIMMXNCHOCH = 1 (Jó név, csak elég nagy hülyeség)

írjuK Ki a Képernyőre a svéd Siv Widerberg Szerelem című versét - stringeK használatával!

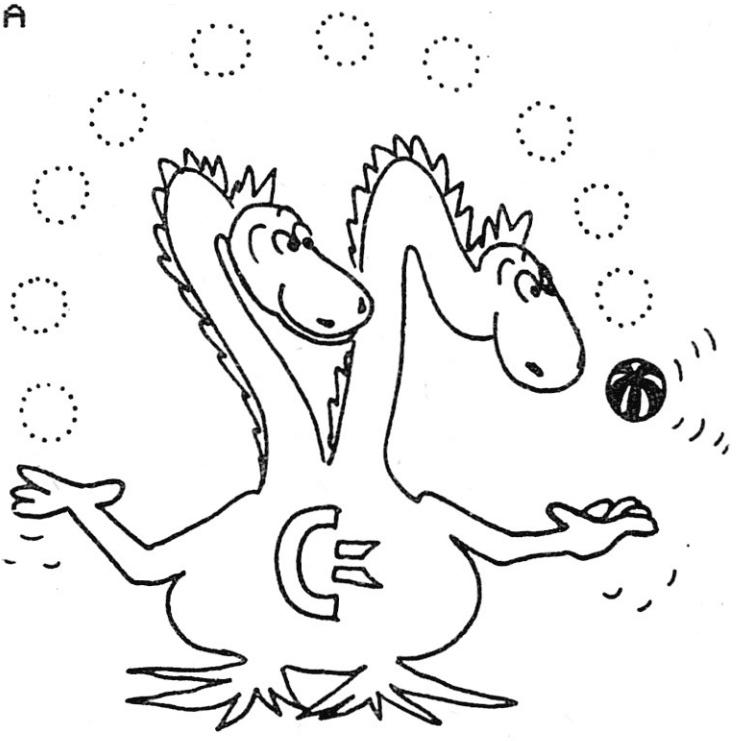
```
5 PRINT"■"
10 COLOR4,3,3:COLOR0,2,7
20 PRINT"■":REM CONTROL+3
30 S$="STEIN-MALTENAK "
40 N$="NAGY, VOROS ES ELALLO"
50 F$="FUL"
60 E$="E VAN."
70 A$="A "
80 SZ$="NEKEM TETSZIK"
90 PRINT S$ N$
100 PRINT F$ E$
110 PRINT SZ$
120 PRINT A$ N$
130 PRINT F$
```

Ha lefuttatjuK a programot, elolvashatjuK a listából eléggé rejtélyesneK tűnő, Kedves Kis verset. Közben pedig, ha összehasonlítjuK a programlistát a Kész verssel, a stringeKről tanultaKat is átismételhetjüK.

EmléKszünK rá, hogy az ötödiK napot egy labdát mozgató programmal fejeztüK be. Eléggé siralmas Kis programocsKa volt ez, ami azt illeti: zötyögve, villogva haladt a golyó, és ráadásul csaK balról jobbra. Most már tudunK eleget ahhoz, hogy megírjuK ezt a programot sokKal rövidebben, és a golyó nemcsaK balról jobbra fog haladni, hanem ide-oda pattog a Képernyőn, amíg a STOP gombbal meg nem állítjuK.

```
1 REM PATTOGO GOLYO
10 REM JOBBRA HALAD
20 PRINT"■"
```

```
30 FOR J=0 TO 39
40 PRINT" ●■■";:REM CRSR BALRA
50 FOR F=1 TO 10:NEXT F
60 NEXT J
100 REM BALRA HALAD
110 FOR B=39 TO 0 STEP -1
120 PRINT"■■● ■■■";
130 FOR F=1 TO 10:NEXT F
140 NEXT B
150 REM KEZDI ELOLROL
160 GOTO 30
```



Néhány megjegyzést már bele is írtunk a programba (REM-mel elkülönítve), de még tartozunk egy magyarázattal: mitől mozog a golyó?

Nézzük meg a stringünket! A kötelező idézőjel után egy üres szóközt hagytunk. Ez az üres hely fog kirajzolodni a golyó előtti oszlopban - és ezáltal kitörli az előző golyót -, ettől fog majd úgy tűnni, hogy a golyó átkerült az egyik oszlopból a másikba. A string harmadik eleme eggyel balra lépteti a cursort. Enélkül az üres szóköz a string elején nem az előző golyót törölné ki, hanem az utána következő üres helyre kerülne, és így a golyó nem haladna előre, hanem egyre több golyó rajzolódna ki.

Ugyanez a helyzet a balra haladó golyóval: ott az üres szóköz a golyó jobb oldalán foglal helyet, hiszen az új golyó mindig az előzőtől balra rajzolódik ki; a cursort pedig háromszor kell balra mozgatni, hogy a következő lépésben egy hellyel az előzőtől balra kezdje a kiírást.

A stringekkel most már eléggé jól tudunk bánni ahhoz, hogy a tornázó robot programját - ld. a hetedik napot az előző kötetben - meg tudjuk írni ebben a formában is. Ugye észrevettük, hogy stringnek nevezzük akkor is, ha nem értelmes szövegből áll, hanem grafikus jelekből?

```
10 COLOR4,7,2:COLOR0,7,2
20 PRINT"■":PRINT"▗"
30 D$="*****"
40 B$="      *   *"
45 A$="▒▒▒▒▒▒"
50 PRINT"       ╲/" :REM SHIFT+M,SHIFT+N
60 PRINT"       ┌──┐":REM SHIFT+U,SHIFT+*,SHIFT+I
70 PRINT"       |..|":REM SHIFT+-
80 PRINT"       └──┘":REM SHIFT+J,SHIFT+*,SHIFT+K
90 PRINT"        ■" :REM C=+L,C=+J
100 FOR X=1 TO 5:PRINT"     *";A$;"*":NEXT X
110 PRINT"       ▒▒▒▒"
120 FOR Z=1 TO 4:PRINT B$:NEXT Z
130 FOR I=1 TO 200:NEXT I
140 PRINT"▗"
150 PRINT"        ╲/"
160 PRINT"        ┌──┐"
170 PRINT"        |..|"
180 PRINT"        └──┘"
190 PRINT"         ■"
200 PRINTD$;A$;D$
210 FOR Z=1 TO 5:PRINT"      ";A$:NEXT Z
220 PRINT"       *   *"
230 PRINT"     *       *"
240 PRINT"   *           *"
250 PRINT"  *             *"
260 FOR I=1 TO 200:NEXT I
270 GOTO 20
```

Ezt a programot is, csakúgy, mint az előzőt, a STOP gombbal lehet leállítani. Utána el lehet szórakozni azzal, hogy átalakítjuk, ahogy szoktuk; egyetlen számjegy átjavításával nyurga vagy tömzsi robotot csinálhatunk, stb.

Most még egy kicsi program, sokféle változóval, INPUTtal, hogy világosan értsük, miről van szó. Gépeljük be ezt a kis programot, amely először megkérdezi a személyi adatainkat, aztán a képernyőre kiírja a névjegyünket.

```
0 REM NEVJEGY
10 PRINT"■"
20 PRINT"MELYIK VAROSBAN ELSZ?"
30 INPUT V$
40 PRINT"MELYIK UTCABAN?"
50 INPUT U$
60 PRINT"MI A HAZSZAM?"
70 INPUT H
80 PRINT"MI AZ IRANYITOSZAM?"
90 INPUT I%
100 PRINT"■"
110 PRINT"IME A LAKCIMED:"
120 PRINTI%,V$
130 PRINTU$," UTCA";H
```

Figyeljük meg a változókat. A város neve (V$) szöveges változó, akárcsak az utcanév (U$). Az irányítószám egész változó, mert ott törtszámnak nincs értelme. A házszám viszont valós változó, mivel ha valaki például az Avokádó utca 24/26-ban lakik, azt ilyen formában beírhatja: 24.26 - ezt ugyan tizedes törtnek fogja fel a gép, de mivel számolnia nem kell vele, csak kiírnia, ezért nem okozunk bajt.

Egészítsük ki a programot úgy, hogy telefonszámot is kérdezzen, és a kiíráskor a lakcím alá tegye!

Épp eleget tudunk a változókról ahhoz, hogy megírjunk egy olyan programot, amely egy táborozó gyerek levelét írja ki a képernyőre. A levél szövege az alábbiakban olvasható, a programban a kipontozott részek helyére különféle fajtájú stringeket írjunk, meghozzá úgy, hogy az értékük INPUT formájában kerüljön a helyére.

"Kedves ...!
KérleK, Küldj sürgősen ... darab ...-t, mert elfogyott a ...-m, és ha ...-ra sem lesz, aKKor ... "

Ezt a programot most segítség nélKül írjuK meg - könnyítésül annyit, hogy a Kipontozott részeK helyére Kerülő változóKat mi így neveztüK el: CIMZETT$, DARAB%, MI$, MAGYARAZAT$, HATARIDO$, KOVETKEZMENY$; azaz, röviden: CI$, DA%, MI$, MA$, HA$, KO$.

Sok mindent tudunk már a számítógép képességeiről, de egy igen fontos lehetőségről még szót sem ejtettünk.

Eddig kétféle módon dolgoztattuk a gépet: vagy úgy, hogy sorra vette a programsorokat, a legalacsonyabb sorszámúval kezdve, vagy pedig - mint mondtuk - 'külön kérésre' kilépett ebből a folyamatból, és a GOTO utasítás hatására átugrott az általunk előre megnevezett programsorra.

A gépet azonban feltételes módban is utasíthatjuk. Hogy ez hogyan megy, arra nézzünk egy nagyon rövidke kis programot.

```
10 LET A=A+1
20 ? A;
30 IF A<100 THEN GOTO 10
40 END
```

No, itt aztán van mit nézni. Először is itt a 10-es sor, amely ellentmond mindennek, amit eddig matematikából tanultunk. Hogy lehet A egyenlő (A+1)-gyel? Hát úgy, hogy ez nem matematika, hanem BASIC, és nem egyenlet, hanem a gépnek adott utasítás: azt jelenti pontosan: 'az A-nak nevezett változó értékét növeld meg eggyel'.

Ha külön nem mondunk mást, akkor a gép az A értékét először 0-nak veszi, és onnan növeli eggyel minden alkalommal, amikor erre a programsorra ér.

A 20-as sor, ezt régóta tudjuk, kiírja az A mindenkori értékét, és utána nem lép rögtön új sorra, mert ott a pontosvessző.

Az igazi érdekesség a 30-as sorban van. Az a talán ismeretlen jel azt jelenti: "Kisebb, mint". (Figyelem! a fejezet végén összegyűjtve bemutatjuk ezeket a jeleket - aki türelmetlen, már most odalapozhat!) És ott van még az a két ismeretlen szó:

IF és THEN. Ezeknek van pontos magyar megfelelőjük: HA és AKKOR. E két szó az angolból változatlan értelemben került át a BASICbe, vagyis egy feltételes módot vezet be a programba. BASICből magyarra fordítva a sort, ilyen mondatot kapunk tehát: 'Ha A kisebb, mint száz, akkor menj a 10-es sorra!'

A gép tehát egyesével növelgeti A értékét, és minden alkalommal ellenőrzi, hogy elérte-e a 100-at. Ha A értéke 100-zal egyenlő, akkor a 30-as sor betöltötte szerepét, a program tovább fut: rá a 40-es sorra, és ott azt találja: END.

Ez új utasítás ugyan, de jól ismeri mindenki, aki látott már angol filmet: a szó azt jelenti: VÉGE. Ha tehát az A értéke eléri a százat, akkor a program futásának vége.

Világosabban fogjuk látni az IF...THEN szerkezet lényegét, ha a programunkat átalakítjuk.

```
10 LET A=A+1
20 ? A;
30 IF A=100 THEN END
40 GOTO 10
```

Az előbbi esetben így okoskodott a gép, amikor a 30-as sorra ért: 'Kisebb az A értéke, mint 100? Igen. Akkor vissza kell mennem a 10-es sorra.' És vissza is ment mindaddig, amig A valóban kisebb volt 100-nál. Ha A elérte a 100-at, vagyis nem volt kisebb, akkor továbblépett a programban.

Most, a második esetben ugyanolyan feltételes módú az utasításunk, de a gép gondolatmenete más. 'Egyenlő A értéke 100-zal? Nem? Akkor megyek tovább.' És ment tovább a következő sorra, anélkül, hogy megvizsgálta volna, mi áll a THEN után a sorban. A THEN utáni utasítást ugyanis a számítógép csak akkor hajtja végre, ha az IF utáni feltétel igaz. A THEN egy zárt ajtóhoz hasonlítható, amely csak akkor nyílik ki, ha teljesül a feltétel, amelyet az IF szabott.

(úgy szokták magyarázni, hogy az IF...THEN szerkezet két ágra nyílik szét: az "igen" ágra és a "nem" ágra. "Igen" ágnak nevezik azt, amelyik a THEN utáni utasításokból áll. Ide akkor jut el a program, ha az IFben megszabott feltételek teljesülnek. Ha nem így történik, akkor a "nem" ágra fut a program, ez az ág pedig az IFet követő első sor első utasításával kezdődik.)

Második programváltozatunknak van egy másik tanulsága is: nem biztos, hogy a program vége egybeesik a programlista végével. Ha az IF...THEN szerkezet úgy kívánja, az END utasítás után ezer programsor is következhet még.

Csináljunk egy apró módosítást a program második változatában.

```
10 LET A=A+3
20 ? A;
30 IF A=100 THEN END
40 GOTO 10
```

Futtassuk le! Meglepetés fog érni: a program jócskán túlfut a százon, és úgy írja tovább a képernyőre a számokat, mintha soha nem akarná abbahagyni. Le kell állítanunk a STOPpal.

Mi történt itt? Egyszerű: ha hármasával számolunk, akkor az IFben megszabott feltétel sohasem teljesül: a 100 nem osztható hárommal. A THEN kapuja csukva marad, az ENDre nem fut rá a program.

A megoldás: egy új jelet kell használni, azt, hogy >=. (Nagyobb, vagy egyenlő.) a 30-as sor tehát így alakítandó át:

```
30 IF A>=100 THEN END
```

Ez már egyértelmű: az A értéke most már legfeljebb kettővel haladhatja túl a százat.

Az IF...THEN sok-sok BASIC programban jön majd a segítségünkre. Például a következőben, amely kipróbálja, hogy milyen fejszámoló, aki használja. (Persze csak alapfokon.)

```
0 REM FEJSZAMOLTATO
10 FOR I=12 TO 60 STEP 3
20 PRINT"HELLO!"
30 PRINT"USS BE 2 SZAMOT, AMELYEK OSSZEGE"I
40 INPUT A
50 INPUT B
60 IF A+B=I THEN PRINT"OKE, HELYES A VALASZ.":GOTO 80
70 PRINT"EZ NEM STIMMEL, PROBALD UJRA!":GOTO 30
80 NEXT I
```

Az egész programot egy FOR...NEXT ciklus fogja közre, amely 12 és 60 között hármasával lépkedve adja fel a számokat. A két INPUT sorban adjuk be a szerintünk helyes két megfejtést, amit a program összead, az eredményt összehasonlítja az általa feladott I-vel, és ha a kettő egyezik ("igen" ág), akkor a 80-as programsorra küld. Ha az összeadást nem sikerül megoldanunk, tehát rossz számokat gépeltünk be, akkor a THEN után következő "igen" ág helyett a program a "nem" ágra ugrik, enyhén ránk pirít a pontatlan feleletért, és - újból feladja az előző számot! (Világos, hiszen nem lépett tovább a NEXT I-t tartalmazó 80-as sorra, hanem előbb visszafordult.)

Vállaljunk egy kis kockázatot: egy fontos tanulság kedvéért rontsuk el a jó programunkat! Töröljük ki a 60-as sorból a GOTO 80 részt! A program elveszíti a józan ítélőképességét: ha helyes választ adunk, akkor előbb megdícsér, de utána ugyanúgy megró, mintha hibáztunk volna, és ugyanazt a számot adja fel újra és újra, a végtelenségig.

Vigyáznunk kell erre: a programnak mindig meg kell mondani, hogy mit tegyen, ha a feltételek teljesülnek (tehát az "igen" ág irányát), különben a "nem" ágon folytatja, és elvisz bennünket az erdőbe.

Végül váltsuk be ígéretünket: gyűjtsük ki a két mennyiség egymás közti viszonyát kifejező jeleket!

JEL	JELENTÉS
=	(egyenlő)
>	(nagyobb, mint)
<	(kisebb, mint)
<>	(nem egyenlő)
>=	(nagyobb vagy egyenlő)
<=	(kisebb vagy egyenlő)

Ezt a fejezetet egy kis történettel kezdjük; lehet, hogy sokan ismerik, de talán többen vannak, akik nem.

A későbbi nagy matematikus, Karl Friedrich Gauss (1777-1855) már iskolásgyerekkorában igen jóeszű fiú volt. Egy számtanórán a tanár, akinek valami sürgős maszek munkája akadt, szerette volna, ha nem kell az órát megtartania, ezért egy feladatot adott az osztálynak, ami - gondolta naivan - majd lefoglalja a nebulókat, s ő nyugodtan végezheti a munkáját. 'Gyerekek, adjátok össze az egész számokat 1-től 100-ig!' - mondta, és leült dolgozni. Azzal azonban nem számolt, hogy zseni is van az osztályban. A kis Gauss alig pár perc után jelentkezett az eredménnyel: 5050.

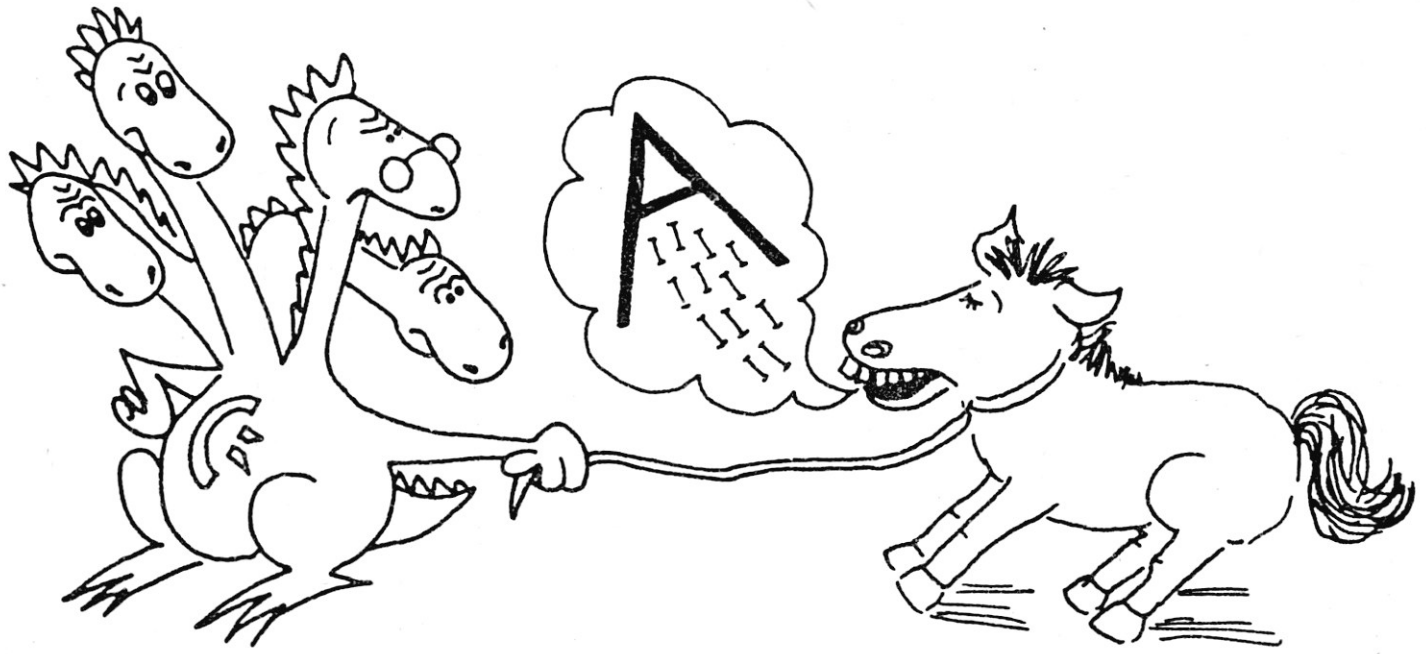
A tanár dühösen, de elképedve kérdezte, hogyan számította ki ilyen hamar. Gauss elmagyarázta a módszerét: észrevette, hogy ha az 1-től 100-ig tartó számsor két végén álló számot összeadja, 101-et kap eredményül. Ugyanígy 101 jön ki akkor is, ha a második és az utolsó előtti számot adja össze (2+99), és így tovább, míg a sorozat közepén az 50 és az 51 összege is éppen 101. Ezért a 101-et megszorozta 50-nel, és máris kezében volt a megoldás: 5050.

Gauss gyorsan kiszámította az eredményt, de még ő sem versenyezhet a számítógéppel, ha megfelelő programot írunk rá. Írtunk is: gépeljük be, futtassuk le, aztán beszélünk róla.

```
10 REM GAUSS
20 A=0
30 FOR I=1 TO 100
40 A=A+I
50 NEXT I
60 ? A
70 A=0:I=1
80 A=A+I : I=I+1
90 IF I>100 THEN ? A : END
100 GOTO 80
```

RUN után alig egy másodperc telik el, és a gép már írja az eredményt: 5050. De mi van itt? Még egyszer kiírja, az előző alá. Csak nem kétszer számolta ki ez az egy program?

De bizony, pontosan ez történt. Mégpedig azért, hogy szemléletesen be tudjuk mutatni: a FOR...NEXT ciklus és az IF...THEN szerkezet tökéletesen tudja helyettesíteni egymást bizonyos feladatokban - például ebben. A programunk első része (a 20-as sortól a 60-asig) egy FOR...NEXT ciklus, amely két változót használ: az A-t meg az I-t; az I egyesével lépked a számsoron 1-től 100-ig, az A-ban pedig sorban összeadódnak ezek a számok, és a végeredmény kiíródik a képernyőre is.



A program másik része (a 70-es sortól a 100-asig) ugyanezt végzi el egy IF...THEN szerkezettel. Itt érdemes megfigyelni: az "igen" ágon kiíratjuk A értékét, és vége a programnak, a "nem" ágon pedig visszaküldjük a 80-as sorra, hogy növelje tovább az I-t, egészen az utolsó, a 100-as értékig.

Az END utasítás bárhol elhelyezhető a programban, akár ideiglenesen is - ha például valami nem úgy működik a programunkban, ahogyan elterveztük, és szeretnénk megkeresni a hiba forrását. Ilyenkor valahová beírhatjuk (célszerű persze, ha külön sorba) az ENDet, és a program csak odáig fut, utána READYvel leáll. Ily módon akár sorról sorra ellenőrizhetjük a

programunkat, ha az ENDet átrakjuk új és új helyekre. Az END hatására idő előtt leállt programot folytatni is tudjuk, ha begépeljük a CONT utasítást, és megnyomjuk a Returnt. (CONT a "continue", azaz "folytasd!" angol szó rövidítése.)

Próbáljuk ki: a Gauss-programba írjuk be a két ciklus határára az ENDet (például 65-ös sorszámmal):

```
10 REM GAUSS
20 A=0
30 FOR I=1 TO 100
40 A=A+I
50 NEXT I
60 ? A
65 END
70 A=0:I=1
80 A=A+I : I=I+1
90 IFI>100 THEN ? A : END
100 GOTO 80
```

Most a program csak egyszer írja ki az 5050-et, és csak akkor fut rá az IF...THEN szerkezetes programrészre, ha azt mondjuk neki: CONT.

A "nagyobb", "kisebb" jelekkel végezzünk most el egy meglepő kísérletet! írjuk be ezt a kis programot:

```
1 REM ABC
10 INPUT"ELSO NEV";A$
20 INPUT"MASODIK NEV";B$
30 IF A$<B$ THEN ? A$,B$:GOTO 50
40 ? B$,A$
50 END
```

Ez a program ABC-sorrendbe állít bármilyen két nevet (vagy más szót). Szokatlan gondolat szöveges változók esetében a "kisebb" jelet használni, hiszen hogyan "kisebb" az A a B-nél? Hát úgy, hogy a betűknek a gépben kódszámuk van, amely az ABC sorrendjében növekszik. Amikor a gép az INPUT sorokban megkapja a két nevet, e kódszámokat hasonlítja össze, és eszerinti sorrendben írja ki őket. Sőt, azt is tudja, hogy az ANNAMARIA később jön az ABC-ben, mint az ANNA!

Fejezzük be egy feladattal: az összeadást gyakoroltató program mintájára írjunk egy olyat, amelyik felad két számot 1 és 20 között, és a szorzatukat kérdezi a használótól; ha helyes választ kap, akkor tovább lép, ha helytelent, akkor korholó szavak kíséretében még egyszer feladja ugyanazt a két számot.

Ha valaki vektorokat emleget előttünk, ne szaladjunk el messzire - ezek sem veszélyesebb fogalmak, mint amelyekkel eddig megismerkedtünk. A BASICben a vektorok olyan változócsoportok, amelyeket azonos néven tartunk nyilván, de megszámozzuk őket, hogy átteKinthetőbb legyen a programunk.

Még ilyen ostobaságot! Hiszen a változók elnevezésében éppen az a poén, hogy jól különítsük őket el egymástól (CIMZETT$; HATARIDO$ - és így tovább...) Most meg azzal jön elő valaki, hogy nevezzük őket egyformán, attól átteKinthetőbb lesz; ez teljes őrület.

Csak türelem, nézzünk néhány példát.

SZÖVEGES VÁLTOZÓ	SZÖVEGES VEKTOR
K$ = "KUTYA"	HA$(1) = "KUTYA"
M$ = "MACSKA"	HA$(2) = "MACSKA"
L$ = "Ló"	HA$(3) = "Ló"
C$ = "CSIRKE"	HA$(4) = "CSIRKE"

A programunkban mindkét fajtát használhatjuk; akár PRINT K$, akár PRINT HA$(1) áll a programban, a KUTYA ugyanúgy megjelenik a képernyőn.

Természetesen számszerű változókat is elnevezhetünk így:

```
A(1) = 1
A(2) = 2
A(3) = 3
```

és így tovább. De ezzel még nem magyaráztunk meg semmit. "És most mi van? - kérdezheti akárki. - Miért ne használjam a régi, jól bevált változóimat, minek nekem a vektorok?"

Hát először is: a vektorok segítségével könnyebb nyomon követni, mi történik a programban.

(Eddig csak ilyen kis nyúlfarknyi programocskákat írogattunk, de amikor már sok száz vagy ezer sorban kell eligazodni, sokkal átteKinthetőbb, ha az összetartozó változókat vektorokba rendezzük.) Emellett rengeteg programozási időt is megspórolunk. Nézzük meg például az alábbi programot!

```
10 PRINT"↓"
20 FOR I=1 TO 10
30 PRINT I;:INPUT"HAZIALLAT";HA$(I)
40 NEXT I
50 FOR I=1 TO 10:PRINT HA$(I)
60 NEXT I
```

No, tessék, futtassuk le a programot, és utána próbáljuk meg megírni ugyanezt vektorok nélkül! Sok időbe fog kerülni, de csak rajta, próbáljuk meg. H0$-tól H9$-ig nevezgethetjük a változóinkat, amit a fenti programban a számítógép végzett el helyettünk, összetartozó tömbként kezelve változóinkat.

Ha valaki nem sajnálta a fáradságot, és megírta a programot, meglátta, mennyi időt takaríthat meg a vektorokkal. De mielőtt továbbmennénk, nézzük meg gyorsan, hogyan bánt el a programunk a FOR...NEXT ciklusban a vektorváltozókkal.

1) A FOR...NEXT ciklus sorban behelyettesítette az I értékeit a zárójelek közé, úgyhogy a vektort így göngyölítette fel a gép:

```
FOR I = 1 TO 10
 HA$(1) < itt kezdi a ciklust először
  HA$(2) < másodszor ér a FORhoz
   HA$(3) < harmadszor ér a FORhoz
    HA$(4) stb.
     HA$(5)
      HA$(6)
       HA$(7)
        HA$(8)
         HA$(9)
          HA$(10)
NEXT I
```

2) Minden INPUT stringet, amit begépeltünk, sorban

Egy valamirevaló stopperóra persze legalább tizedmásodperceket mér; amihez már a TI$ nem alkalmas, hiszen az másodpercenként lép tovább. Elő kell vennünk a TI változót, és meg kell szelídítenünk.

Ha a TI hatvanadmásodperceket mér, akkor ebből másodperceket úgy kapunk, ha elosztjuk a mindenkori értékét hatvannal. írjuk át programunk 20-as sorát így:

```
20 PRINT TI/60
```

Az így elindított program egy irgalmatlanul hosszú számot ír ki, vagy kilenc tizedesjeggyel, és a tizedesponttól balra egyesével számol fölfelé. A tizedesjegyekből csak az elsőre van szükségünk, a többit elhagyhatjuk, hiszen úgysem tudjuk szemmel követni őket, meg aztán úgyis csak hármasok meg hatosok vannak benne (hiszen hat többszörösével osztottunk). Hagyjuk hát el őket. Emlékszünk, az RND függvénnyel kapcsolatban találkoztunk az INT kifejezéssel, amely törtszámokból leválasztotta a tizedesjegyeket. Használjuk fel most is ezt.

```
20 PRINT INT(TI/60)
```

Még mindig nem jutottunk tovább, mint ahonnan elindultunk: most a program körülbelül úgy viselkedik, ahogyan a TI$ változót használó programunk működött, azzal a hátránnyal, hogy a 60-as érték után 61-et ír ki, tehát csak másodpercben számol, percben nem. Ráadásul még mindig nem látjuk sehol a tizedmásodperceket.

No, a tizedmásodpercek gondját most már könnyen megoldjuk. Egy tizedesjegyet ki kell mentenünk az INT hatása alól. Ezt úgy csináljuk, hogy a TI értékét két részletben osztjuk hatvannal: előbb hattal, azután tízzel, de az INT a tízzel való osztásra nem vonatkozik, ezt az egy tizedesjegyet engedélyezni fogja. Ez könnyebb, mint hinnénk: csak egy zárójelet kell ügyesen beiktatnunk:

```
20 PRINT (INT(TI/6))/10
```

A program meg mindig csak másodpercben számol, a tizedmásodpercekről nem vesz tudomást, de a tizedeket már kiírja. Így bármely perc alatti időket már tudunk mérni vele.

Ez persze még mindig csak része a sikernek. Most azt kell elérnünk, hogy a gép kiírja a percek értékét, ehhez minden hatvanadik másodpercben hozzáadjon egyet, a másodperceket pedig állítsa nullára.

Hogy átteKinthetőbb legyen a változók rendszere, a másodperceket nevezzük el MP-nek, a perceket P-nek!

Írjuk újra a programot így:

```
0 PRINT CHR$(147):CLR:PRINT" 0  PERC";
10 TI$="000000"
15 PRINT TAB(10)"          MASODPERC□"
20 MP=(INT(TI/6))/10
30 REM IDE MAJD IRUNK MEG VALAMIT
40 PRINT TAB(10)MP
50 PRINT"□□"
60 GOTO 20
```

A nullás sorban látható új utasítás, a CLR (az angol CLEAR, ejtsd: klír; magyarul: "tiszta" szó rövidítése) az összes változó értéket nullára állítja - ezt a biztonság kedvéért tettük oda, mert már jó néhányszor átállítgattuk TI$ értékét, és bevezetünk új változókat is. Jobb, ha amikor futtatni kezdjük a programot, a változóink értéke mind 0.

Az így lefuttatott program üres képernyővel kezd. Kiírja: 0 PERC, utána számlálgatja az egész és tizedmásodperceket, ezektől jobbra pedig még azt is kiírja: MASODPERC.

Hogyan oldható meg, hogy minden hatvanadik másodpercben a PERC szó előtt álló számjegy eggyel emelkedjék? Sejthető, hogy az IF...THEN szerkezetet kell használnunk, és ez igaz is. Írjunk be egy új 30-as sort!

```
30 IF MP=60 THEN 100
```

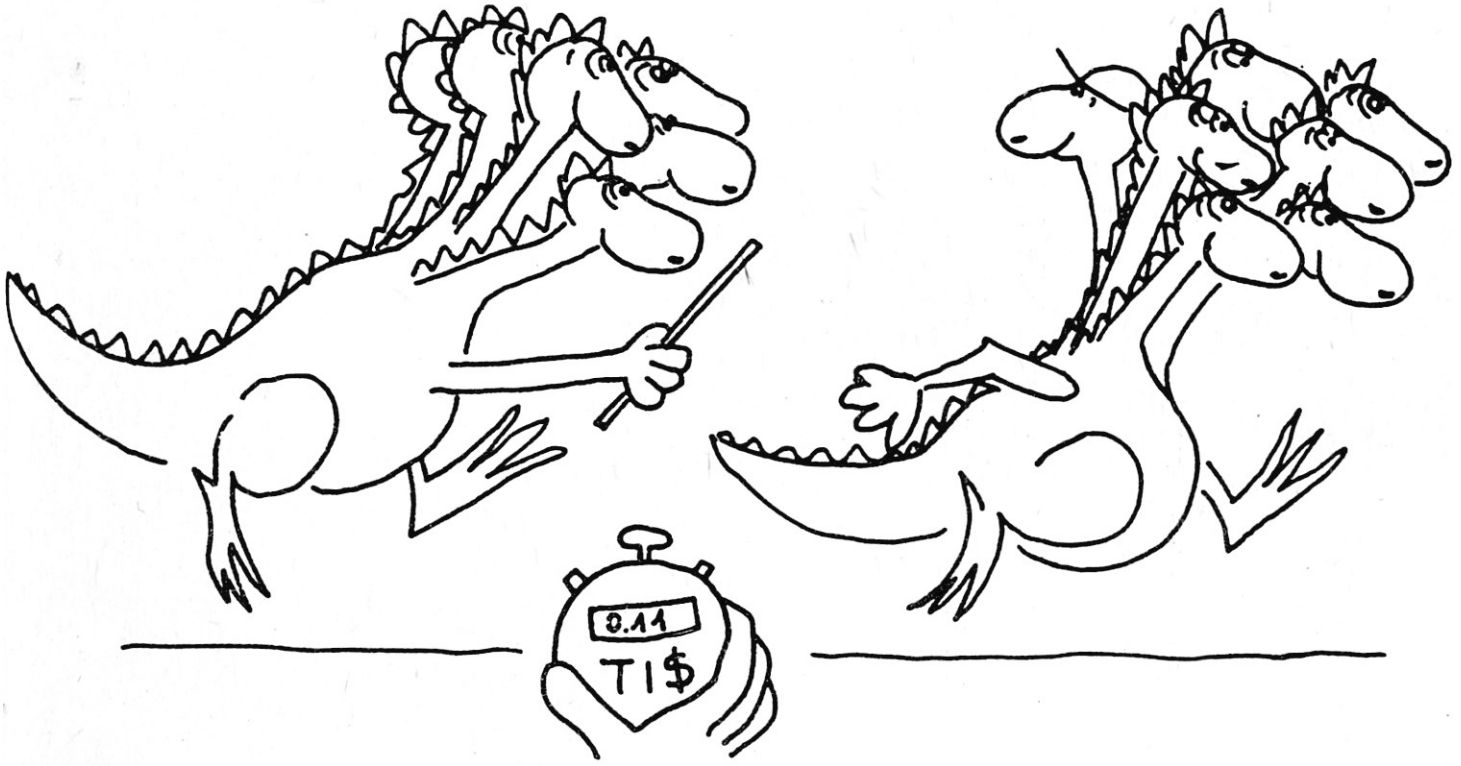
és egészítsük ki a programot két új sorral a végén:

```
100 P=P+1
200 PRINT P;: GOTO 10
```

Hát itt mit csináltunk? A P=P+1 egy eddig nem használt változó értékét emeli meg eggyel. A CLR, mint mondtuk,

mindent nullára állít, tehát ha új változót veszünk fel, az is 0-ról kezd, ha előtte más értéket nem adunk neki. így amikor a programban az MP értéke először éri el a 60-at, és a GOTO leküldi a programot a 100-as sorra, ehhez a 0-hoz ad hozzá 1-et a gép, és a képernyő bal sarkában álló 0 PERC felirat első karakteret kicseréli 1-re. Utána visszaugratja a programot arra a sorra, ahol a TI$ értéket nullázтuk le, ennélfogva a másodpercek 60-nál nem nőnek tovább, hanem újra indulnak 0-ról.

Kész a stopperóránk. Kivihetjük kazettára vagy lemezre, és csinálhatunk belőle olyan szubrutint is, amely egy program futása közben a játékos reakcióidejét méri. Egy ilyen felhasználását látjuk majd a hetedik nap programjai között.



Utolsó munkanapunkon végigkövettük tehát, hogyan születik meg egy meghatározott célú BASIC program, milyen gondolatmenettel építjük fel, hogy azt a célt szolgálja, amit szántunk neki. Utolsó gyakorlatként írjuk majd át úgy, hogy ha a percek száma eléri a 60-at, akkor órákat is számoljon! Az eddigiek ismeretében ez nem lesz nehéz.

A hátralévő egy nap arra szolgál, hogy néhány mintaprogram segítségével átismételjük, amit eddig megismertünk a COMMODORE PLUS-4 mikroszámítógép képességeiből.

A gép természetesen sokkal többet tud, mint amennyit egy ilyen rövid kis könyvből meg lehet tanulni, de mostantól ha újságban, könyvben Plus-4-re (vagy C-16-ra) írt BASIC programlistát látunk, nagy részét meg fogjuk érteni, és ez az első lépés ahhoz, hogy mind nagyobb mértékben tudjuk a saját szolgálatunkba állítani a gépet.

Az utolsó napon, ami e könyvből hátravan, vegyünk még néhány programot. Az első egy továbbfejlesztett változata a korábban már megismert csillagos programnak. A csillagok ezúttal már sokszínűek; a képernyő olykor tűzijátékhoz hasonlít.

Ez a program nem az összes színt használja, ezért akinek kedve van, átalakíthatja úgy, hogy a Commodore Plus-4-nek mind a 16 színét tartalmazza az összes árnyalattal együtt!

```
1 REM SZINES CSILLAGOK
10 DIM A(15)
20 COLOR4,1:COLOR0,1
30 PRINT"■"
40 A(0)=5:A(1)=28:A(2)=30:A(3)=31
50 A(4)=129:A(5)=144
60 FOR I=6 TO 13:A(I)=I+143:NEXT I
70 A(14)=158:A(15)=159
80 P=INT(39*RND(1))
90 Q=INT(22*RND(1))
100 S=INT(15*RND(1))
110 FOR Y=0 TO Q:PRINT:NEXT Y
120 PRINT TAB(P)CHR$(A(S))"*"
130 FOR N=1 TO (10*RND(1)):NEXT N
140 PRINT"■"
150 GOTO 40
```

A második program egy rövid ellenőrzés, mennyire ismerjük a billentyűzetet, a karakterek elhelyezkedését. Ha a programlistát megfigyeljük, kiderül, hogy a játék itt időre megy!

```
0 REM BILLENTYUZET-TESZT
5 PRINT"■"
10 PRINT"MENNYI IDO ALATT TALALOD MEG"
20 PRINT"A KARAKTEREKET?"
30 PRINT"HA MEGJELENIK EGY JEL, A LEHETO"
40 PRINT"LEGGYORSABBAN CSALD ELO MEGEGYSZER!"
50 PRINT"HA MEGUNTAD, NYOMD MEG A RETURNT!"
60 PRINT"NOSZA..."
70 KAR=INT(62*RND(0)+33)
80 PRINT TAB(5)CHR$(KAR),
90 TI$="000000"
100 GET A$:IF A$="" THEN GOTO 100
110 IF ASC(A$)=13 THEN END
120 IF A$<>CHR$(KAR) THEN GOTO 100
130 M%=VAL(TI$)
140 IF M%<60 THEN PRINTM%" MASODPERC":GOTO 70
150 P%=M%/100:MA%=M%-P%*100
160 PRINTP%" PERC"MA%" MASODPERC"
170 GOTO 70
```

Az itt következő programmal ábécé-sorrendbe rendezhetünk bármilyen névsort, az utána következő másik változat pedig a betűrendbe szedett neveket ki is nyomtatja.

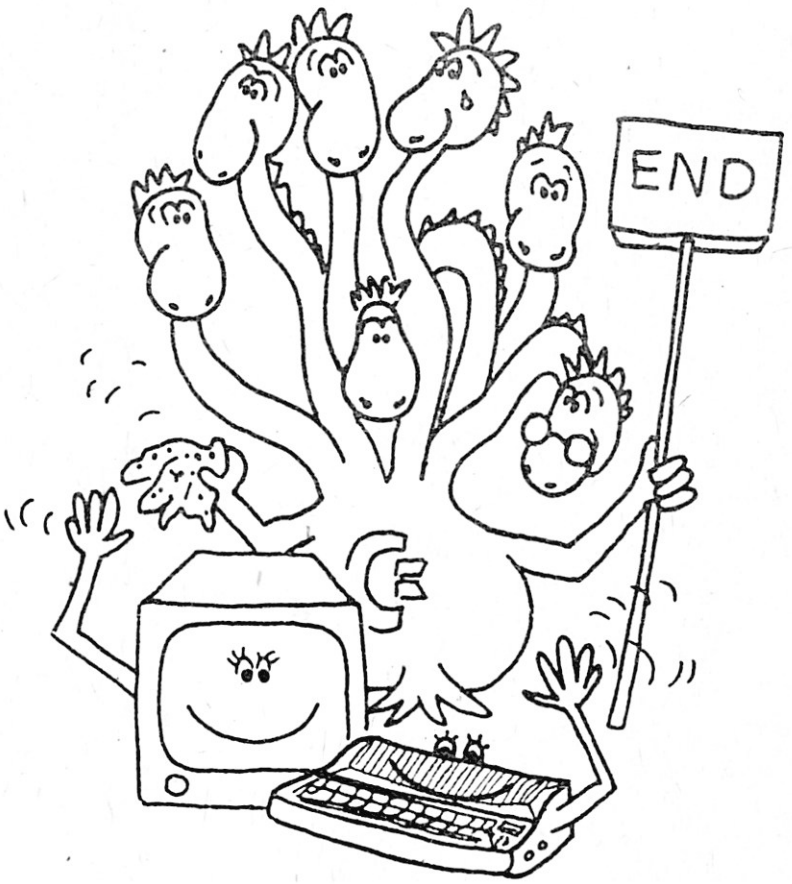
```
0 REM ABECE-SORREND
10 INPUT"A NEVEK SZAMA ";M
20 DIM A$(M+1)
30 FOR I=0 TO M-1:INPUT A$(I):NEXT I
40 FOR K=0 TO M-1
50 FOR L=K+1 TO M
60 IF A$(L)>=A$(K) THEN 100
70 S$=A$(L)
80 A$(L)=A$(K)
90 A$(K)=S$
100 NEXT L
110 NEXT K
120 PRINT:PRINT"BETURENDBEN:":PRINT
130 FOR I=0 TO M:PRINTA$(I):NEXT I
```

```
0 REM ALFABETIKUS RENDEZES PRINTERRE
5 PRINT"■"
10 INPUT"A NEVEK SZAMA ";M
20 DIM A$(M+1)
30 FOR I=0 TO M-1:INPUT A$(I):NEXT I
40 FOR K=0 TO M-1
50 FOR L=K+1 TO M
60 IF A$(L)>=A$(K) THEN 100
70 S$=A$(L)
80 A$(L)=A$(K)
90 A$(K)=S$
100 NEXT L
110 NEXT K
115 OPEN1,4:CMD1
120 PRINT:PRINT"BETURENDBEN:":PRINT
130 FOR I=0 TO M:PRINTA$(I):NEXT I
140 PRINT#1:CLOSE1
```

Negyedik programunkban mi adhatunk fel rejtvényt a számítógépnek. Gondolunk egy számot 1 és 10 között, és a gép több kérdés után kitalálja. Ez a program eléggé bonyolult és hosszú, vigyázzunk a begépelésénél! Ha figyelmesen végigolvassuk, majdnem mindent megtalálunk benne, amiről ebben a kis könyvben szó esett. Ezért utolsó ismétlő feladatnak is megfelel.

```
0 REM 1 ES 10 KOZOTT
5 PRINT"■"
10 FOR I=1 TO 10:A(I)=I:NEXT I
20 PRINT:PRINT
30 PRINT"GONDOLJ EGY SZAMRA 1 ES 10 KOZOTT!"
40 INPUT"PRIMSZAM? (I/N)";K$
50 IF ASC(K$)=73 THEN GOTO 70
60 A(1)=0:A(2)=0:A(3)=0:A(5)=0:A(7)=0:GOTO 80
70 A(4)=0:A(6)=0:A(8)=0:A(9)=0:A(10)=0
80 INPUT"5-NEL NAGYOBB? (I/N)";K$
90 IF ASC(K$)=73 THEN 110
100 FOR I=6 TO 10:A(I)=0:NEXT I:GOTO 120
110 FOR I=1 TO 5:A(I)=0:NEXT I
120 GOSUB 600
130 IF K=1 THEN GOTO 10
140 INPUT"PAROS SZAM? (I/N)";K$
150 IF ASC(K$)=73 THEN GOTO 170
160 FOR I=0 TO 10 STEP 2:A(I)=0:NEXT I:GOTO 180
170 FOR I=1 TO 9 STEP 2:A(I)=0:NEXT I
```

```
180 GOSUB 600
190 IF K=1 THEN GOTO 10
191 INPUT"3-MAL OSZTHATO? (I/N)";K$
192 IF ASC(K$)=73 THEN GOTO 194
193 A(3)=0:A(6)=0:A(9)=0:GOTO 196
194 A(1)=0:A(2)=0:A(4)=0:A(5)=0:A(7)=0
195 A(8)=0:A(10)=0
196 GOSUB 600
197 IF K=1 THEN GOTO 10
200 INPUT"5-TEL OSZTHATO? (I/N)";K$
210 IF ASC(K$)=73 THEN GOTO 230
220 A(5)=0:A(10)=0:GOTO 250
230 FOR I=1 TO 4:A(I)=0:NEXT I
240 FOR I=6 TO 9:A(I)=0:NEXT I
250 GOSUB 600
260 GOTO 10
600 K=0
610 FOR I=1 TO 10:IF A(I)=0 THEN GOTO 630
620 K=K+1:B=A(I)
630 NEXT I
640 IF K<>1 THEN 700
650 PRINT"A KERESETT SZAM ="B
700 RETURN
```



AKi megszoKta a Commodore 16-os gép Kezelését, és BASIC programja írása Közben a Plus 4-esen átváltana GRAPHIC üzemmódba, jól gondolja meg, mielőtt nagy lendülettel megnyomná az F1 billentyűt! Az ismert GRAPHIC felirat helyett valami eléggé érthetetlen áKombáKom jeleniK meg a Képernyőn:

SYS1525:3-PLUS-1

Nem romlott el a gép; ez a felirat így van jól, ahogy van. A Plus 4 belsejében ugyanis négy (pontosabban: 3 plusz 1) Kész program található, és ezeK az F1 billentyű lenyomásával érhetőK el. Cserében a GRAPHIC utasítást Kézzel Kell beírnunK, de ez nem nagy áldozat ahhoz Képest, amit KapunK érte.

Vegyük sorra, mit Kínál felhasználóinaK ez a négy szoftver! Nem fogunK nagyon elmélyedni bennüK, hiszen gép mellett ott a segédKönyv, amely részletesen elmagyaráz mindent. SzoKásunK szerint csaK a felszínt borzoljuK meg, hogy Kedvet próbáljunK csinálni az alapos tanulmányozáshoz.

Nos tehát. úgy KezdődiK a dolog, hogy az F1 gombbal előhívott SYS stb. üzenetre megnyomjuK a Returnt. A Képernyő elfeKetediK, majd Kisvártatva elindul a szövegszerKesztő program. Ha nyomKodni KezdjüK a billentyűKet, betűK, jeleK jelenneK meg a Képernyőn, és aKármiKor megnyomhatjuK a Returnt, sohasem KapunK ?SYNTAX ERROR hibaüzenetet. Ebből tudhatjuK, hogy ahol most járunK, az nem a szoKásos BASIC, hanem egy másiK rendszer. A gépünK jelentős változáson ment Keresztül: ez a program írógépet csinált belőle!

SZÖVEGSZERKESZTŐ

Annak, aki még sohasem használt szövegszerkesztőt, nehéz elmondani, miért jobb, mint az írógép. Elsősorban talán azért, mert csak a végleges, hibátlan szöveg kerül papírra: mindaddig, amíg a nyomtatáshoz hozzá nem kezdünk, korlátlan lehetőségünk van javítani: törölni, betoldani; egyes szavakat, mondatokat, bekezdéseket áthelyezni, a szöveg külalakján változtatni. Iratunkat tetszésünk szerinti példányban nyomtathatjuk ki, és amíg a printer dolgozik, mi nyugodtan hátradőlhetünk: immár semmi dolgunk az egésszel. (Ezt a könyvet is ilyen szövegszerkesztő programmal írtuk; ugyan nem Plus-4-es gépen, hanem Commodore 64-en.)

Aki használt már elektromos írógépet, alapfokon könnyen el fog igazodni ebben a programban. Van azonban egy lényeges különbség, amit nagyon könnyű megszokni: a Returnt (ez, ha megfigyeljük, ugyanott van, ahol a 'kocsi vissza' gomb az elektromos írógépeken) nem kell minden sor végén megnyomni! A számítógép majd magától sorokba tördeli a szövegünket; a Return csak a bekezdés végét jelzi - azt az utasítást jelenti, hogy a nyomtató MINDENKÉPPEN kezdjen új sort, akárhol tart a szövegben.

Ha tehát folyamatos prózai szöveget írunk, a gép addig ír egy sorba, amíg helye van rá (ezt a képernyőn sajnos nem így látjuk), utána egy szóköznél új sort kezd, és ezt így folytatja, amíg más utasítást nem adunk neki (például egy Returnt). Versekkel más a helyzet: mivel ott a sorok nem egyforma hosszúak, mindegyiknek a végén meg kell nyomni a Returnt, hogy a gép új sort kezdjen.

Ha elkészültünk a szövegünkkel, két választásunk van: kinyomtathatjuk, vagy későbbi felhasználásra el is menthetjük. Az előbbihez természetesen nyomtató, népszerű nevén printer, az utóbbihoz pedig mágneslemez-meghajtó egység (disk drive) szükséges. A Plus-4 beépített programjai sajnos csak lemezzel működnek, kazettával nem. (Kapható persze olyan szövegszerkesztő program, amely a kazettás egységet is használja, és a magyar ékezetes betűkészlet is benne van. Annak, aki

rendszeresen akarja használni a szövegszerkesztőt, érdemes ezt beszereznie.)

Ennyi alapismeret birtokában most már bárki neki is vághat a szövegszerkesztőnek; ha eltévedne benne, vegye elő a géphez kapott könyvecskét, amely részletesen elmagyarázza, mi a teendő.

ADATBÁZIS-KEZELŐ

A számítógép-használat előnyeinek legnyilvánvalóbb illusztrációja. Adatbázis segítségével minden nyilvántartási, visszakeresési munka gyerekjáték; dossziék tucatjai helyett egy-két mágneslemez elegendő; egy-egy adat megtalálása pedig, ahelyett, hogy fülig porosan, napokon keresztül lapozgatunk a kartotékokban, mindössze annyiból áll, hogy behelyezzük a lemezt, elindítjuk a programot, és megadjuk a gépnek, hogy mit akarunk kikeresni.

Adatbázisunkat a saját igényeink szerint állíthatjuk össze. Ha például a könyvtárunkat katalogizáljuk, a könyvek szerzője, címe, témája, terjedelme, akár a színe is lehet visszakeresési szempont. Még azt is nyilvántarthatjuk, hogy melyik könyvet kinek adtuk kölcsön, meg azt is, hogy ha visszahozza, melyik polc hányadik helyére kell tennünk. Minden attól függ, hogy milyen szempontokat vettünk fel adatbázisunk megtervezésekor.

A Plus-4-es adatbázis-kezelőjéhez úgy juthatunk el, ha először elindítjuk a szövegszerkesztőt, utána a Commodore és a C billentyű együttes lenyomásával parancs üzemmódba lépünk át, és begépeljük a TF parancsot.

Ha ezután megnyomjuk a Returnt, a képernyőn azonnal feltűnik a kérdés: új adatbázist akarunk-e létrehozni, vagy egy régit használni. Akármelyiket akarjuk is, természetesen csak mágneslemezen tehetjük. A program ugyanis úgy működik, hogy a lemezen található adatokat végigolvassa, és azokat, amelyek a visszakeresési szempontnak megfelelnek, megjeleníti a képernyőn. Innen az adatok átvihetők a szövegszerkesztőbe, és ilyenformán ki is nyomtathatók.

Az adatbázis létrehozása, használata komoly munka; nem is illik e könyv kereteibe. kedvünk volna belemélyedni, de tudjuk, hogy a Plus-4-tulajdonosok és e könyv olvasói között kevesen vannak, akik disk drive-val rendelkeznek, és különben is, talán említettük már: ott a gépkönyv, amely kitűnően, lépésről lépésre bevezet a program használatába.

SPREADSHEET

Ez a program a harmadik leggyakrabban használt szoftvertípus a világon (kivéve persze a játékokat). Mégis eléggé bajban vagyunk, ha az elnevezését magyarra akarjuk fordítani. A "spreadsheet" angol szó, és kiterített lepedőt, papírívet jelent.

A program maga hasonlít is egy ilyenhez: rengeteg vízszintes és függőleges oszlopban temérdek adatot képes egyszerre kezelni. Minden rovatát magunk határozhatjuk meg; számadatokat és szövegeket egyaránt bevihetünk, és a számokkal mindenféle műveleteket tudunk végrehajtani. (Megszabhatjuk például, hogy az ív 17. rovata mindig a 6. és a 12. rovatban álló szám összege legyen; ilyenkor a program folyamatosan végrehajtja az összeadásokat - vagy még sokkal bonyolultabb műveleteket. Következésképpen a spreadsheet-program elsősorban pénzügyi nyilvántartások vezetésére ideális.

Az igazat megvallva e sorok szerzői eddigi életükben nem szorultak rá a spreadsheet-programok rendszeres használatára... Ez persze nem jelenti azt, hogy például egy-egy kisvállalkozó vagy GMK ne vehetné igen jó hasznát.

A spreadsheet-programmal kísérletezni óhajtóknak azt ajánljuk, vegyék elő a segédkönyvet, és mindenben aszerint járjanak el, ahogyan ott írva van.

TÁBLÁZATOK

HATÁS	CHR$	HATÁS	CHR$	HATÁS	CHR$
	0	!	33	G	71
	1	"	34	H	72
	2	#	35	I	73
	3	$	36	J	74
	4	%	37	K	75
FEHÉR	5	&	38	L	76
	6	'	39	M	77
	7	(	40	N	78
SHIFT+C=		)	41	O	79
KIKAPCS.	8	*	42	P	80
SHIFT+C=		+	43	Q	81
BEKAPCS.	9	,	44	R	82
	10	-	45	S	83
	11	.	46	T	84
	12	/	47	U	85
RETURN	13	0	48	V	86
KKCS		1	49	W	87
F->A.	14	2	50	X	88
	15	3	51	Y	89
	16	4	52	Z	90
CRSR LE	17	5	53	[	91
RVS BE	18	6	54	£	92
		7	55	]	93
HOME	19	8	56	↑	94
DEL	20	9	57	←	95
	21	:	58	—	96
	22	;	59	♠	97
	23	<	60	|	98
	24	=	61	—	99
	25	>	62	—	100
	26	?	63	—	101
ESCAPE	27	@	64	—	102
PIROS	28	A	65	|	103
CRSR		B	66	|	104
JOBB	29	C	67	╮	105
ZÖLD	30	D	68	╰	106
KÉK	31	E	69	╯	107
SZÓKÖZ	32	F	70	╭	108

HATÁS	CHR$	HATÁS	CHR$	HATÁS
\	109	f4	138	—
/	110	f6	139	|
┌	111	HELP	140	▒
┐	112	SHIFT/		|
●	113	RETURN	141	▒
—	114	KKCS		◤
♥	115	A->F	142	|
|	116		143	├
╭	117	FEKETE	144	▪
╳	118	CRSR FEL	145	└
o	119	RVS KI	146	┐
♣	120	CLEAR	147	▬
|	121	INST	148	┌
◆	122	BARNA	149	┴
+	123	SÁRG.ZÖLD	150	┬
▒	124	RÓZSASZÍN	151	┤
|	125	KÉKESZÖLD	152	|
╭	126	VIL. KÉK	153	|
◥	127	SÖTÉTKÉK	154	▮
	128	VIL. ZÖLD	155	▁
NARANCS	129	BÍBOR	156	▬
VILL.BE	130	CRSR BAL	157	▬
VILL.KI	131	SÁRGA	158	┘
	132	CIÁNKÉK	159	▪
f1	133			▪
f3	134	SZÓKÖZ	160	▪
f5	135	▮	161	▪
f7	136	▬	162	▚
f2	137	—	163	

CHR$ 192-223 ugyanaz, mint CHR$ 96-127
CHR$ 224-254 ugyanaz, mint CHR$ 160-190
CHR$ 255 ugyanaz, mint CHR$ 126

RÖVIDÍTÉSEK A TÁBLÁZATBAN:
KKCS = karakterkészlet-csere
F = felső; A = alsó
KIKAPCS. = kikapcsolás; BEKAPCS. = bekapcsolás
VILL.BE = villogás bekapcsolása
VILL.KI = villogás kikapcsolása

SZÍNKEZELÉS

BILLENTYŰ	SZÍN	KÓD	KÉPERNYŐ-JEL
Control + 1	FEKETE	1	
Control + 2	FEHÉR	2	
Control + 3	PIROS	3	
Control + 4	CIÁNKÉK	4	
Control + 5	BÍBOR	5	
Control + 6	ZÖLD	6	
Control + 7	KÉK	7	
Control + 8	SÁRGA	8	
C= + 1	NARANCS	9	
C= + 2	BARNA	10	
C= + 3	SÁRGÁSZÖLD	11	
C= + 4	RÓZSASZÍN	12	
C= + 5	KÉKESZÖLD	13	
C= + 6	VILÁGOSKÉK	14	
C= + 7	SÖTÉTKÉK	15	
C= + 8	VILÁGOSZÖLD	16	

Bekapcsoláskor cursorszínként az egyes színeknek a következő árnyalatait használja a gép:

FEKETE:	0	NARANCS:	4
FEHÉR:	7	BARNA:	2
PIROS:	3	SÁRGÁSZÖLD:	5
CIÁNKÉK:	6	RÓZSASZÍN:	6
BÍBOR:	4	KÉKESZÖLD:	5
ZÖLD:	3	VILÁGOSKÉK:	6
KÉK:	4	SÖTÉTKÉK:	2
SÁRGA:	7	VILÁGOSZÖLD:	5

A HÁTTÉRSZÍN MEGHATÁROZÁSA
COLOR 0,SZÍNKÓD, ÁRNYALATKÓD

A KERETSZÍN MEGHATÁROZÁSA
COLOR 4,SZÍNKÓD, ÁRNYALATKÓD

A CURSOR SZÍNÉNEK MEGHATÁROZÁSA PROGRAMBAN
COLOR 1,SZÍNKÓD, ÁRNYALATKÓD

CURSORMOZGATÁS JELEI IDÉZŐJELEN BELÜL

BILLENTYŰ	JEL
CLEAR/HOME	
SHIFT + CLEAR/HOME	
CRSR LE	
CRSR FEL	
CRSR JOBBRA	
CRSR BALRA	

GRAFIKAI MÓDOK

SZÁM	HATÁS
0	Szöveges mód
1	Finom felbontású grafika
2	Finom felbontású grafika és szöveg
3	Sokszínű grafika
4	Sokszínű grafika és szöveg

GRAFIKAI UTASÍTÁSOK

DRAW

UTASÍTÁS	HATÁSA
DRAW szín, oszlop, sor	pont
DRAW szín, oszlop, sor TO oszlop,sor	egyenes
DRAW szín TO oszlop, sor	vonal a legutóbbi pontból

CIRCLE

UTASÍTÁS	HATÁSA
CIRCLE1,Közp.oszlop,Közp.sor,sugár	Kör
CIRCLE1,Kp.o,Kp.s,szél,mag	ellipszis
CIRCLE1,Kp.o,Kp.s,sz,m,Kezd,vég	ív
CIRCLE1,Kp.o,Kp.s,sz,m,,,szög	elford. ellipszis
CIRCLE1,Kp.o,Kp.s,sz,m,,,,szög	sokszög

BOX

UTASÍTÁS	HATÁSA
BOX szín,oszl.1,sor1,oszl.2,sor2	Körvonal
BOX szín,o1,s1,o2,s2,szög	elfordítva
BOX szín,o1,s1,o2,s2,,Kitöltés	tömör felület
BOX szín,o1,s1,o2,s2,szög,Kitöltés	elfordítva

TARTALOM

JEGYZETEK

LDA #$50

STA $0FCC,X

DEC $50

BNE $2234,Y

STA $01,X

100B
101K
102M

ELEJE = 2000

CIKLUS 1=1010

CIKLUS 2=1025

JEGYZETEK

[redacted]

79,— Ft