BODOR TIBOR

GERŐ PÉTER

A COMMODORE 64 PROGRAMOZÁSÁNAK GYAKORLATA

ALAPISMERETEK

1

SZÁMALK

A COMMODORE–64 PROGRAMOZÁSÁNAK GYAKORLATA I.

Alapismeretek

COMMODORE PROGRAMMING IN PRACTICE I.

Introduction

TIBOR BODOR
PÉTER GERŐ

COMMODORE

PROGRAMMING IN PRACTICE

INTRODUCTION

First volume

COMPUTING APPLICATIONS AND SERVICE CO. BUDAPEST, 1985

BODOR TIBOR
GERŐ PÉTER

A COMMODORE 64 PROGRAMOZÁSÁNAK GYAKORLATA

ALAPISMERETEK

Első kötet

SZÁMÍTÁSTECHNIKA-ALKALMAZÁSI VÁLLALAT BUDAPEST, 1985

COMMODORE–64 SOROZAT

Lektorálta: *dr. Juhász Bálint*
Supervised by *dr. Bálint Juhász*

ISBN 963 553 115 X (összkiadás)
ISBN 963 553 116 8 (I. kötet)

Kiadja a Számítástechnika-alkalmazási Vállalat
Felelős kiadó: Juhász János vezérigazgató
Felelős szerkesztő: Huba Zoltánné
Műszaki vezető: Molnár Zoltán
Műszaki szerkesztő: G. Müller Zsuzsa
A fedelet tervezte: Molnár Zoltán
Az ábrákat rajzolta: Olgyay Géza
Megjelent: 11,4 (A/5) ív terjedelemben

Készült a FÜTI Nyomdaüzemében
A nyomdai megrendelés törzsszáma: 85-612
Budapest, 1985

Tartalom

Contents

Előszó

Ez a könyv azoknak a kezdő programozóknak szól, akik Commodore–64-es típusú mikroszámítógépen kívánják megtanulni a BASIC programnyelv alapjait, és ezen a gépen jó és biztonságos programokat szeretnének írni. A leendő olvasóról semmiféle számítástechnikai előismeretet és tapasztalatot nem feltételezünk. Fokozatosan haladunk úgy, hogy minden fejezet megértéséhez elegendő legyen a korábbi fejezet ismerete.

A fokozatosság a gép kiépítettségének a tárgyalásában is megnyilvánul. Először olyan lehetőségeket ismertetünk, amelyekhez csak a központi egység és a képernyő (tv) kell, majd egyre bővebb, kazettával, lemezzel, nyomtatóval felszerelt gépkonfigurációra készült alkalmazásokra térünk át.

Nem kiragadott utasításokat, programrészleteket mutatunk be, hanem teljes programokat közlünk, és mindenütt a Commodore által nyomtatott vagy a képernyőre írt program került a könyvbe. Ezek a programok (helyesen begépelve) azonnal futtathatók, kipróbálhatók. Javasoljuk, hogy az olvasó gépelje be őket, majd módosítgassa, és figyelje a változtatások hatását. Feltételezzük tehát, hogy az olvasó rendelkezik Commodore-géppel, vagy könnyen hozzá tud férni egyhez, és a könyvben leírtakat a gépén követi is.

Természetesen a legnagyobb elővigyázatosság mellett is előfordulhat, hogy a gépen valami nem pontosan ugyanúgy működik, ahogyan az a könyvben szerepel. Hibáktól, elírásoktól nyilván ez a könyv sem mentes, és a géptől sem várhatjuk el, hogy a könyvünket elolvassa és eszerint működjön. Ilyenkor érvényesítsük az ismert anekdota szabályát: „ha a terep eltér a térképtől, akkor a terep szerint kell eljárni".

Kérjük az olvasót, hogy a könyvben felfedezett esetleges hibákat és az alkalmazással kapcsolatos észrevételeit közölje a szerzőkkel (INTRONIK Számítástechnikai és Elektronikai Műszaki Fejlesztő Kisszövetkezet, 1445, Bp., pf. 348.), hogy a további kiadások már ezek figyelembevételével készülhessenek.

A programokat úgy állítottuk össze, hogy minél jellegzetesebb feladattípusokat oldjanak meg, és az olvasó minél több részletüket változtatás nélkül is átvehesse saját programjaiba. (Megjegyezzük, hogy a könyvben közölt valamennyi program kazettán és mágneslemezen is kapható.)

A könyv a Commodore–64-ben gyárilag meglevő BASIC nyelvről szól, és nem ennek külön megvásárolható kiterjesztéseiről. Ilyenekre csak néhol utalunk.

A gépi kódú programozásról, BASIC programjaink gépi tárolásáról, a végrehajtás gépközeli vonatkozásairól csak annyit szólunk, amennyi a BASIC programozáshoz szükséges lehet, s ezt is a BASIC-programozó szemszögéből írjuk le.
Ez a könyv lényegében a BASIC legalapvetőbb elemeit tartalmazza. Nem tér ki a bonyolultabb adatállomány-szervezési lehetőségekre, a hanggenerálásra, a grafikára, a bonyolultabb szerkezetű programokra sem. A szerzők és a kiadó elképzelése az, hogy a Commodore–BASIC teljes eszköztárának bemutatására sorozatot jelentet meg, amelynek ez a könyv csak az első kötete. Az előkészületek befejező szakaszában van három további kötet, amelyek közül egy a speciális szervezésű adatállományokkal, egy a Commodore–64-esen megvalósítható adatfeldolgozó programozással, egy pedig a Commodore–64 programozásának „csemegéivel", a speciális grafikával, a hanggal, a botkormánykezeléssel stb. foglalkozik.

Talán nem árt néhány mondatot arra is szánni, milyennek képzeltük könyvünk leendő olvasóját. Olyannak, amilyenek mi magunk is vagyunk, ha saját szakterületünk határain kívül próbálunk mozogni. Hadd szemléltessük ezt egy példával. Nem vagyunk tv-műszerészek, és nem is akarunk azok lenni. Mégis van tv-készülékünk, és azt akarjuk, hogy biztonságosan és kényelmesen kezelhessük, használhassuk, a lehető legkevesebbet törődve azzal, hogy mi zajlik le ezalatt a készülék belsejében.
Tudjuk, hogy ezzel számos lehetőségtől fosztjuk meg magunkat. Egy tv-műszerész például beállíthatná a készülékét úgy is, hogy a képet fordítva, tükörírással lássa. Mi ezt nem tudjuk megtenni, mert csak egyes (gyárilag beépített) kezelési sémákat tudunk végrehajtani. De általában nincs is szükségünk többre, mint ugyanarra a néhány műveletre; ezeket viszont akárhányszor biztonságosan, tévedés nélkül, gyorsan és kényelmesen akarjuk használni, mégpedig úgy, hogy ehhez ne kelljen egy új szakmát elsajátítanunk. Ha pedig esetleg mégis valami különlegeset akarunk a tv-készülékünktől, majd szakemberhez fordulunk.

Mi, akik a programozásban vagyunk szakemberek, azoknak írtuk ezt a könyvet, akik a számítógép-használatról, a programozásról gondolkodnak ugyanúgy, mint mi a tv-készülékünk kezeléséről. Módszereket, paneleket, fogásokat, sémákat adunk tehát az olvasónak: egy programozási LEGO-t, amelyből – a LEGO könnyedségével, de a LEGO kötöttségeivel is – bármit össze lehet állítani.
A didaktikai szempontokat minden mással (így pl. az utasítások egyenkénti tételes tárgyalásán alapuló módszerekkel) szemben előtérbe helyeztük. Előfordul, hogy egyes anyagrészek kiegészítésére, pontosítására más fejezetekben visszatérünk, így egy-egy utasítás tulajdonságainak ismertetése több részletben található meg. Ezért ezt a könyvet azoknak ajánljuk, akik lépésről lépésre, logikusan akarják megismerni a Commodore BASIC-programozását. Az olvasó a könyvben gyakorlati tanácsokat, tapasztalatokat talál, nem pedig olyan műszaki adatokat, amelyek gépkönyvekből vagy próbálgatás útján is könnyen megszerezhetők.

A könyvünkben előforduló szövegösszefüggésben esetenként szőrszálhasogatásnak is tűnő szigorú szakmai kifejezésmóddal szemben szívesebben alkalmaztuk az elterjedt programozói konyhanyelvet. Így például azt mondjuk, hogy egy adatot PRINT utasítással írunk fel a lemezre, pedig a BASIC megkülönbözteti a PRINT és a PRINT# utasítást, a lemezre nem is az adat, hanem a kódja kerül, a PRINT# nem is a lemezre, hanem a pufferbe ír,

nem is ő írja, hanem a lemezkezelő rendszer, és sorolhatnánk még tovább a technikai finomságokat.

Mindazonáltal a „PRINT felírja" és a hasonló kifejezések a felhasználó szemszögéből nagyon is jól tükrözik a gép működését. Vagyis igaz, hogy a PRINT nem ír, de a hatása a felhasználó számára olyan, mintha felírta volna az adatot a lemezre. Ilyen és hasonló szóhasználattal a közérthetőség kedvéért gyakran éltünk ebben a könyvben.

Végezetül jó munkát és kellemes időtöltést kívánunk a könyvhöz, a BASIC-hez és a Commodore-hoz.

A szerzők

Budapest, 1985. február

ISMERKEDÉS A GÉPPEL

A képernyő és a billentyűzet

Ahhoz, hogy a Commodore—64-est használni tudjuk, legalább
- központi egységre,
- tápegységre,
- tv-készülékre és
- antennazsinórra van szükségünk.

A tápegységet és az antennazsinórt is megtaláljuk a központi egységgel együtt csomagolva.

Tápegységből többféle van forgalomban. Ezek közül az a legbonyolultabb típus, amelyiken egy kapcsoló is található.

A központi egység a jobb hátsó sarka felől nézve a 2. ábrán látható.

ÖSSZEÁLLÍTÁS

Győződjünk meg arról, hogy a központi egység kikapcsolt állapotban van (nem úgy, mint a 2. ábrán).

A tápegység hálózati dugóját csatlakoztassuk a hálózathoz, a másik, tucheles dugaszát a központi egység tápegység-csatlakozójához. (Csak egyféleképpen megy bele, és ha még nem kopott le, akkor a tuchelen kis ráragasztott címkén piros TOP felirat és nyíl jelzi, hogy a tuchel melyik oldalának kell felül lennie.)

Az antennacsatlakozó két dugasza közül a vékonyabbat csatlakoztassuk a központi egység antennacsatlakozójához, a vastagabbat a tv-készülék antennacsatlakozójához. (Ha több antennacsatlakozó van, az UHF-et kell használnunk, ha viszont a tv-készüléken másfajta csatlakozó van, akkor szakemberhez kell fordulnunk a megfelelő átalakító elkészítéséért.)

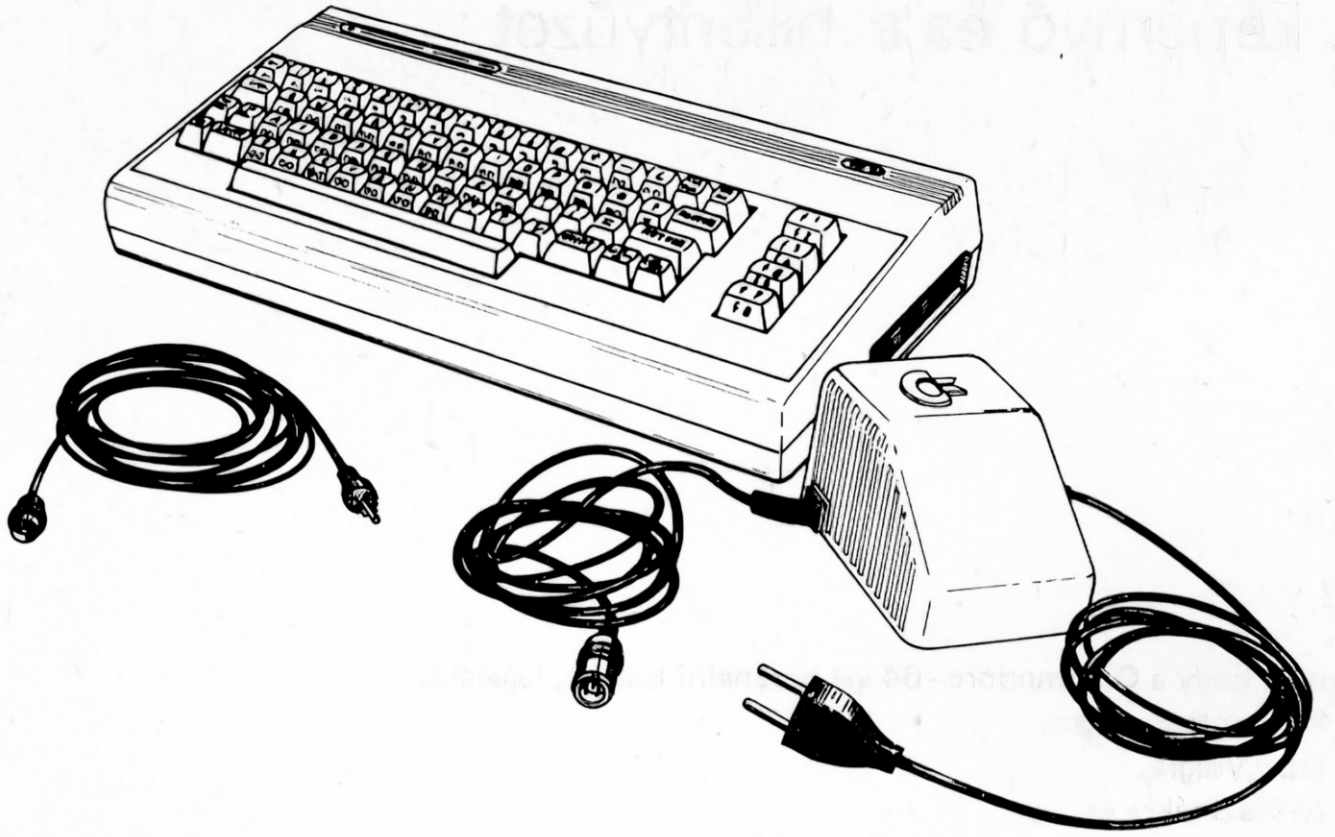
BEKAPCSOLÁS

Először a tv-készüléket kapcsoljuk be, utána a tápegységet (ha van rajta kapcsoló), utoljára a központi egységet.

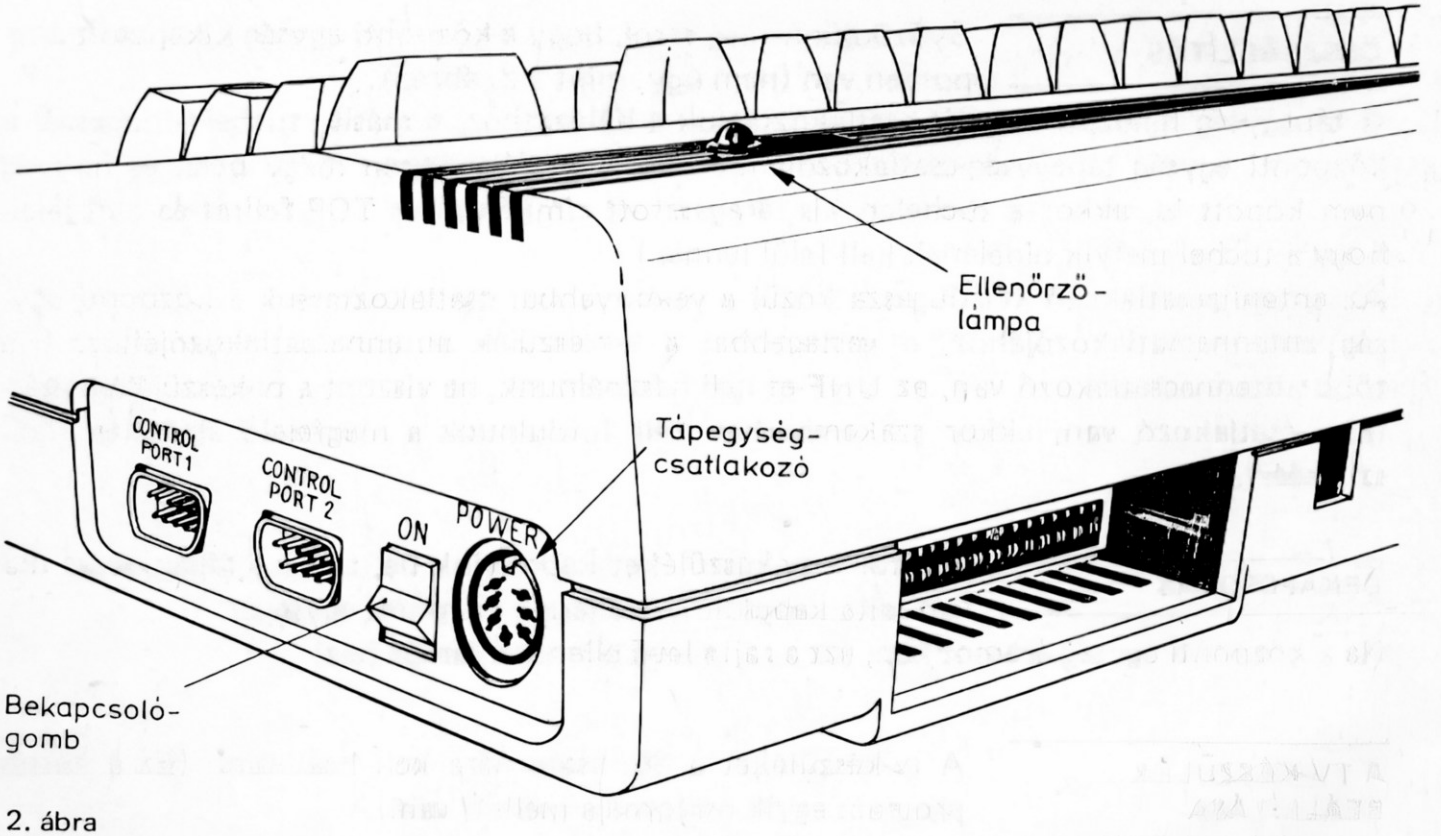
Ha a központi egység áramot kap, ezt a rajta levő ellenőrzőlámpa jelzi.

A TV-KÉSZÜLÉK BEÁLLÍTÁSA

A tv-készüléket a 36. csatornára kell beállítani. (Ez a kettes program egyik csatornája mellett van.)

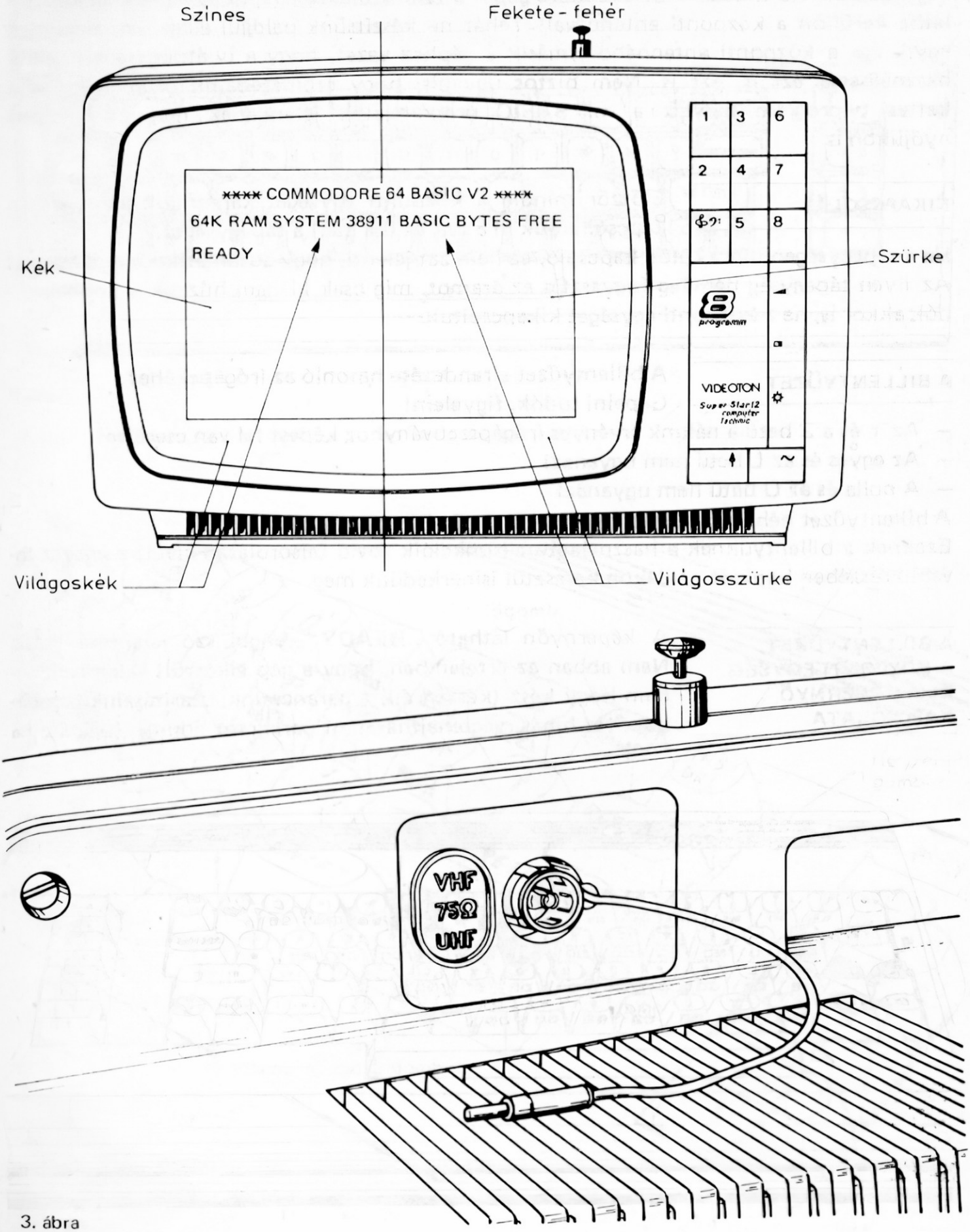


1. ábra


Ellenőrző-
lámpa
CONTROL
PORT 1
CONTROL
PORT 2
ON
POWER
Tápegység-
csatlakozó
Bekapcsoló-
gomb

2. ábra

Ha jól állítottuk be, a tv elhallgat (megszűnik a kettes program hangja, illetve adásszünetben a kellemetlen sistergő hang), és a következő képet látjuk:



3. ábra

Ha a tv-készülék már be volt állítva a megfelelő csatornára, akkor a központi egység bekapcsolása után a hang megszűnését azonnal, a kép megjelenését mintegy 3 másodperc múlva észleljük.

Vigyázzunk! Ne kössük a Commodore-gépet a tv-készülékre úgy, hogy véletlenül kapcsolatba kerüljön a központi antennával! Tehát ne készítsünk például elágazást, amelynek egyik ága a központi antennához, másik a géphez vezet, hogy a tv átdugaszolás nélkül használhassa ezt is, azt is. Nem biztos ugyanis, hogy szomszédaink örülnének, ha a kettes tv-program helyett a mi BASIC programjaink jelennének meg az ő képernyőjükön is.

KIKAPCSOLÁS

Először mindig a központi egységet kapcsoljuk ki, s ezután kapcsolhatjuk ki a tv-t és (ha kell) a tápegységet.

Ha a tápegységen nincs külön kapcsoló, ez nem azt jelenti, hogy automatikusan kikapcsol! Az ilyen tápegység némileg fogyasztja az áramot, míg csak ki nem húzzuk a konnektorból; akkor is, ha a központi egységet kikapcsoltuk.

A BILLENTYŰZET

A billentyűzet elrendezése hasonló az írógépekéhez. Gépelni tudók, figyelem!

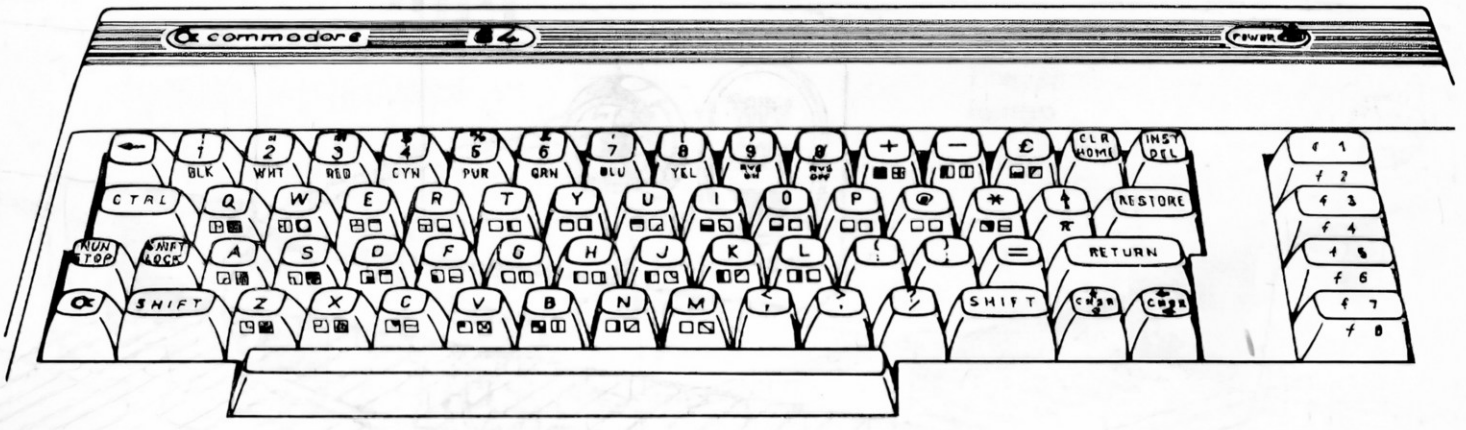
- Az Y és a Z betű a nálunk érvényes írógépszabványhoz képest fel van cserélve!
- Az egyes és az L betű nem ugyanaz!
- A nulla és az O betű nem ugyanaz!

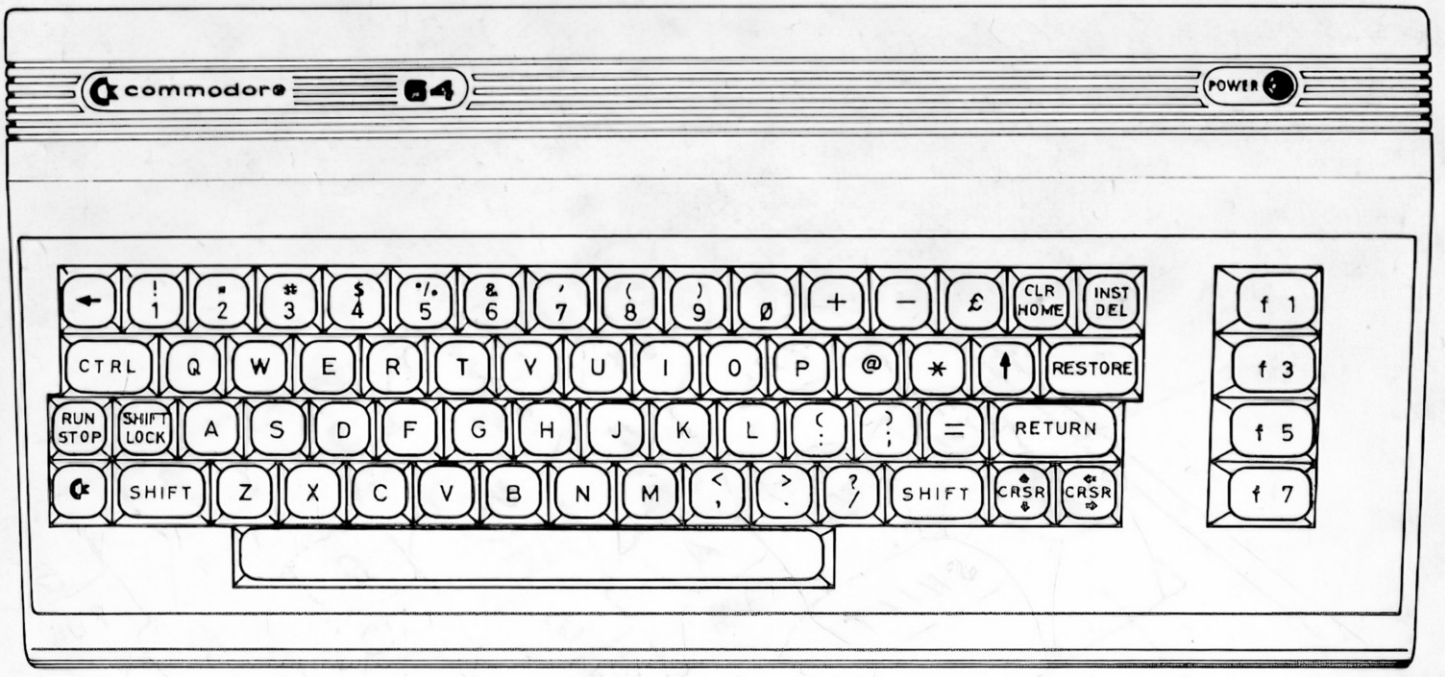
A billentyűzet néhány speciális gombját az 5–6. ábra mutatja be.

Ezeknek a billentyűknek a használatával (funkcióik rövid felsorolásán kívül) a fejezet további részében konkrét példákon keresztül ismerkedünk meg.

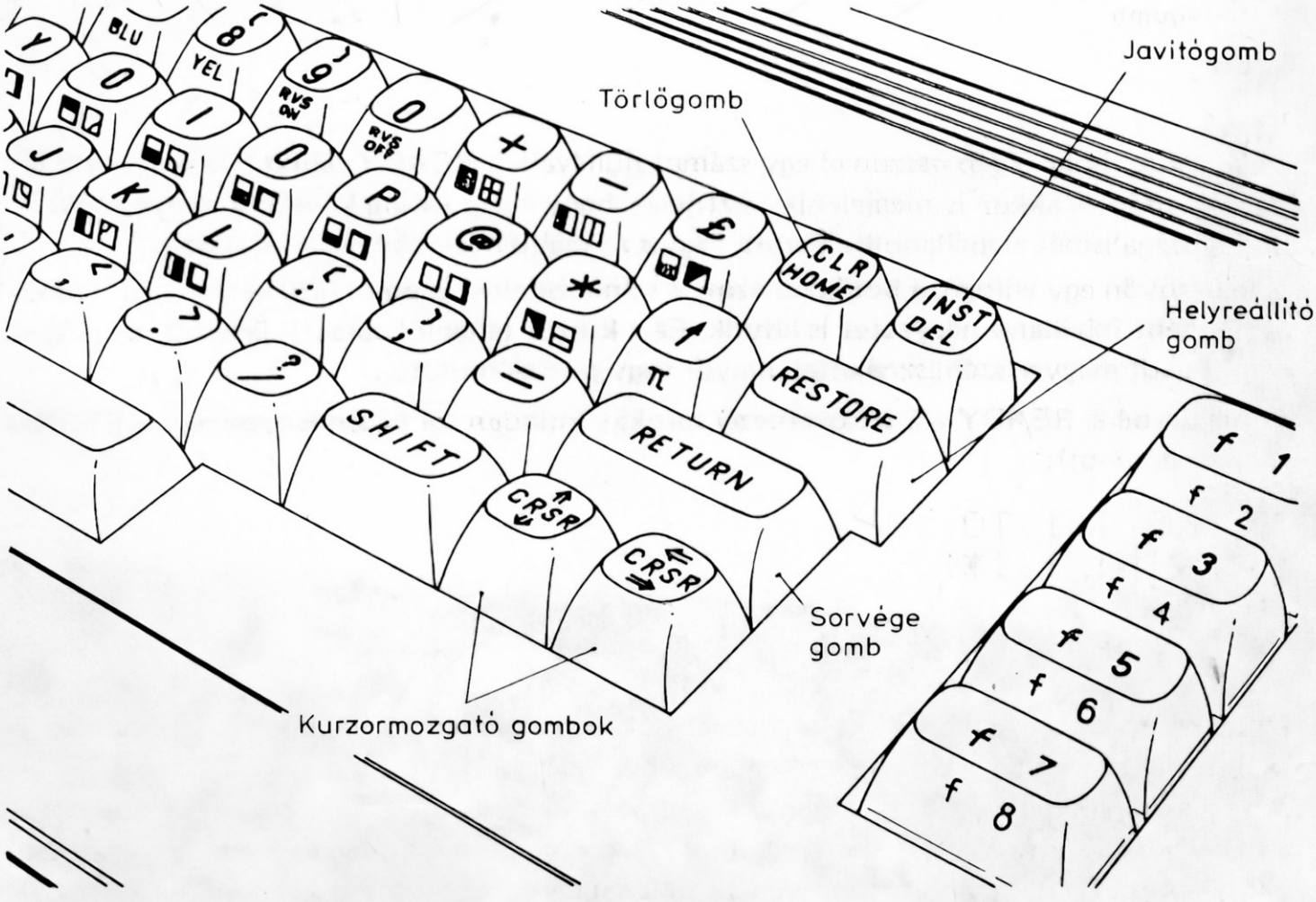
A BILLENTYŰZET, A KÖZPONTI EGYSÉG ÉS A KÉPERNYŐ KAPCSOLATA

A képernyőn látható „READY." angol szó jelentése: kész. Nem abban az értelemben, hogy a gép elkészült valamivel, hanem hogy kész (készen áll) a parancsaink, utasításaink fogadására. Ha hibás, végrehajthatatlan parancsot adunk (például ha

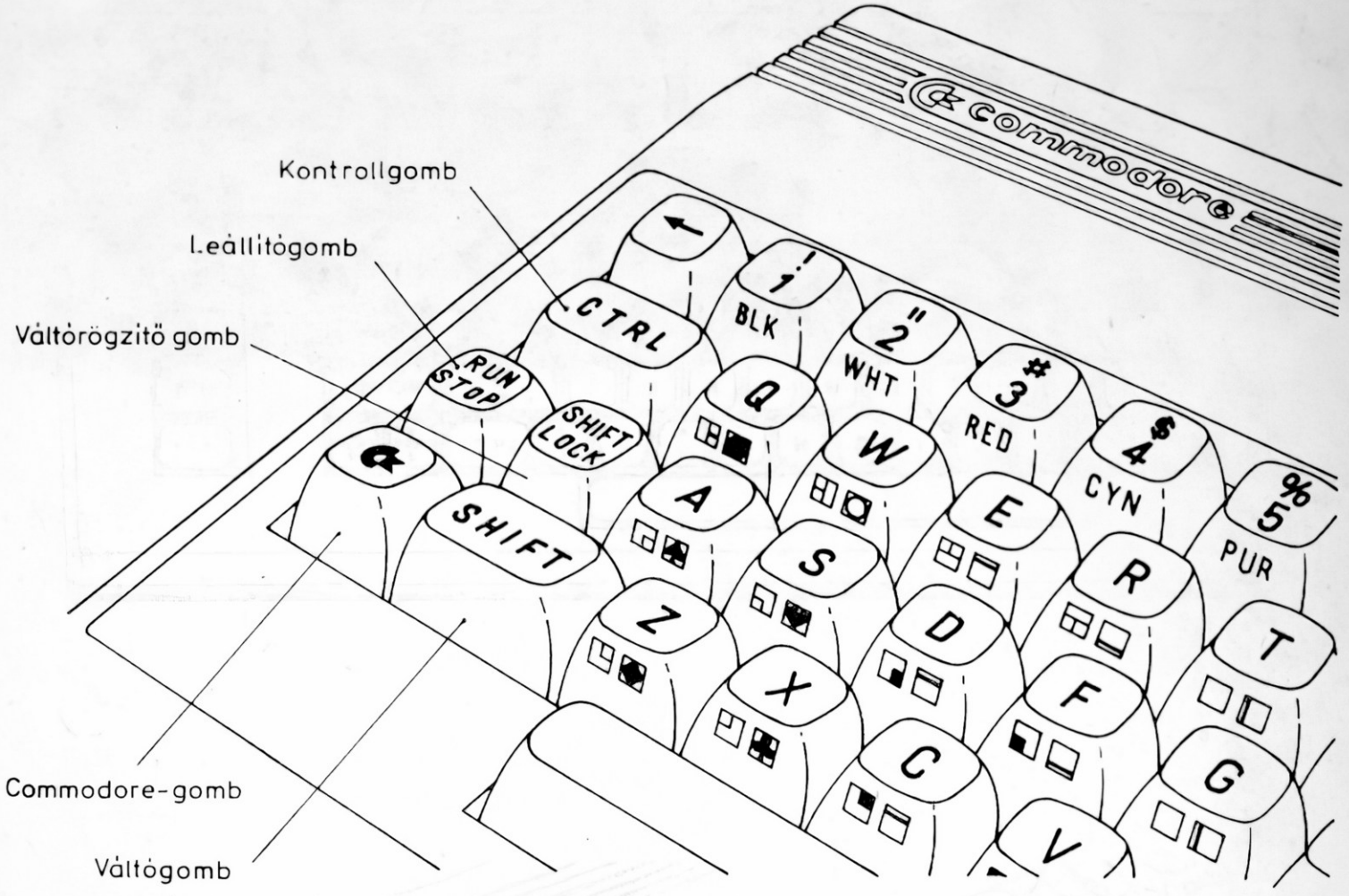



commodore 64
POWER
CTRL
RUN STOP
SHIFT LOCK
SHIFT
CLR HOME
INST DEL
RESTORE
RETURN
SHIFT
CRSR
f 1
f 3
f 5
f 7

4. ábra


Javítógomb
Törlögomb
Helyreállító-
gomb
Sorvége
gomb
Kurzormozgató gombok
CLR HOME
INST DEL
RESTORE
RETURN
SHIFT
CRSR
f 1
f 2
f 3
f 4
f 5
f 6
f 7
f 8

5. ábra



6. ábra

azt akartuk, hogy a gép osszon el egy számot nullával), a READY felirat — a megfelelő hibajelzés után — akkor is megjelenik. Azt jelzi, hogy a gép a hiba következményeit feldolgozta, utána ismét alapállapotba került, vagyis az újabb beavatkozásra várakozik.

A képernyőn egy villogó, a betűkkel azonos színű, betűnyi méretű, kb. kétharmad másodpercenként felvillanó négyzetet is látunk. Ez a kurzor (angolul cursor), javasolt, de még el nem terjedt magyar szóhasználattal: helyőr vagy pozíciómutató.

Gépeljük be a READY alá a következő sorokat (minden sor végén lenyomva a RETURN sorvége gombot):

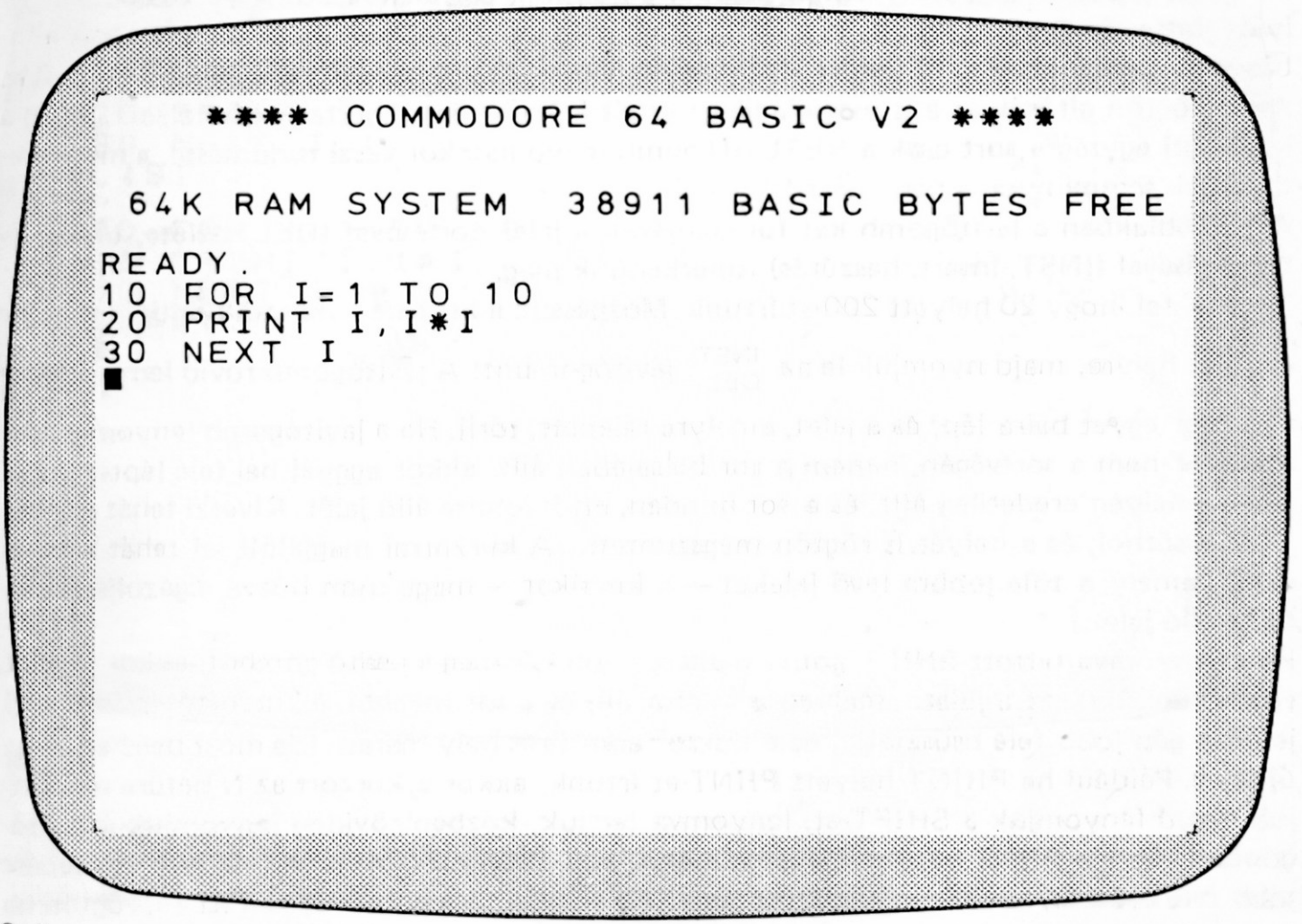
```
10 FOR I=1 TO 10
20 PRINT I,I*I
30 NEXT I
```

7. ábra

Minden egyes betű, számjegy vagy speciális jel gombjának lenyomásakor a kurzor eggyel jobbra lép, s az eddigi helyén megjelenik a lenyomott gomb jele.

A központi egység a RETURN gomb lenyomásakor „pillant rá" a képernyőre, mégpedig a képernyőnek arra az egyetlen sorára, amelyikben a kurzor éppen van. Ami abban a sorban áll a RETURN gomb lenyomásának pillanatában, azt tekinti a központi egység az általunk legújabban begépelt szövegnek. (Valójában a billentyűzet és a képernyő közti kap-

csolat is a központi egység közvetítésével valósul meg, de ennek a taglalása most túl messzire vezetne.)
A sor tudomásulvétele után a kurzor átkerül a következő sorba. Ha ez már nem férne ki a képernyőre, akkor a teljes addigi képernyőtartalom egy sorral feljebb csúszik. Ez azt is jelenti, hogy a korábbi legfelső sor most kicsúszik a képernyőről. (Speciális esetekben a Commodore a sorokat ettől eltérően kezeli — ezzel később fogunk megismerkedni.)
A képernyőn most a három fentebbi sor begépelése után ezt látjuk:



8. ábra

Voltaképpen egy kis programot írtunk. Az egyes programutasítások pontos használatát később ismerjük meg; a program célja ezúttal csak annyi, hogy a gépkezeléssel kapcsolatos első próbálkozásainkhoz felhasználhassuk.
Lehet, hogy a programot az első kísérletre rögtön helyesen gépeltük be, de lehet, hogy hibásan. Az utóbbi a valószínűbb. A számítógéppel való kommunikációban sajnos egyetlen jel elírása (pl. vessző helyett pontosvessző, nulla helyett O betű stb.) is súlyos hiba. A kezelő viszont ritkán ír először rögtön hibátlanul. Ezért nagyon érdemes jól megtanulni és begyakorolni a kényelmes utólagos javítási lehetőségeket.
A kurzor a ⟸ CRSR ⟹ gomb rövid lenyomására egyet jobbra lép, a ⇧ CRSR ⇩ gomb rövid lenyomására egyet lefelé. Ha ezeknek a gomboknak a lenyomásakor a SHIFT LOCK váltórögzítő gomb vagy valamelyik SHIFT váltógomb le van nyomva, akkor a mozgás ellenkező irányú. Ha valamelyik kurzort mozgató gombot hosszabban lenyomva tartjuk, akkor folyamatosan mozgatja a kurzort, másodpercenként kb. tíz lépéssel.

Ha a sor végén álló kurzort jobb felé mozgatjuk, átlép a következő sor elejére. (Akkor is, ha a képernyő utolsó sorában volt, de ekkor a képernyő többi része egy sorral feljebb csúszik.) Ha az első sor elején álló kurzort bal felé akarjuk mozgatni, nem mozdul; más sorok elején bal felé mozgatva átlép az előző sor végére. Ha a képernyő utolsó sorában álló kurzort lefelé mozgatjuk, a helyén marad, de a képernyő minden eddigi sora egy sorral feljebb csúszik. Ha a kurzor a képernyő felső sorában áll, akkor felfelé mozgatni nem lehet.

Ha tehát a sorban valamit elírunk, például a második sor elejére 20 helyett 2O (húsz helyett kettes és O betű) került, és a RETURN-t még nem nyomtuk le, akkor a kurzort a hibás jelre, példánkban az O betűre léptethetjük vissza, és egyszerűen rágépelhetjük a nullát. Az O rögtön eltűnik és a 0 megjelenik; a hibát kijavítottuk, folytathatjuk a sort. Mivel a központi egység a sort csak a RETURN gomb lenyomásakor veszi tudomásul, a hibás változatnak semmi nyoma nem marad.

A továbbiakban a javítógomb két funkciójával, a jelek törlésével (DEL, delete, törlés) és beszúrásával (INST, insert, beszúrás) ismerkedünk meg.

Tegyük fel, hogy 20 helyett 200-at írtunk. Mozgassuk a kurzort a második nullára vagy az ezutáni helyre, majd nyomjuk le az INST/DEL javítógombot! A javítógomb rövid lenyomására a kurzor egyet balra lép, és a jelet, amelyre rálépett, törli. Ha a javítógomb lenyomásakor a kurzor nem a sor végén, hanem a sor belsejében állt, akkor eggyel bal felé lépteti azt a jelet, amelyen eredetileg állt, és a sor minden, ettől jobbra álló jelét. Kiveszi tehát a hibás jelet a sorból, és a helyét is rögtön megszünteti. (A kurzorral megjelölt jel tehát a mozdony, amely a tőle jobbra levő jeleket — a kocsikat — maga után húzva elgázolja a tőle balra álló jelet.)

Ha a lenyomva tartott SHIFT gomb mellett nyomjuk meg a javító gombot, akkor új jelet szúrhatunk be: azt a jelet, amelyen a kurzor áll, és a sor további, a kurzortól jobbra levő jeleit a gép jobb felé csúsztatja, és a kurzor alatt üres hely marad. Ide most beírhatjuk az új jelet. Például ha PRINT helyett PRNT-et írtunk, akkor a kurzort az N betűre mozgatjuk, majd lenyomjuk a SHIFT-et, lenyomva tartjuk, közben röviden lenyomjuk a javítógombot. Erre a PRNT-ből az NT (és minden, ami a sorban ettől jobb felé van) egy lépést jobb felé csúszik, az R és az N között üres hely marad, és a kurzor éppen itt áll; rögtön be is írhatjuk a hiányzó I betűt.

Vigyázat! Nagyon gyakori hiba, hogy a programozó a sorban javít, majd a kurzormozgató gombok segítségével áttér más sorokra, és meglepődik, hogy minden sor hibás marad. A sort a központi egység csak akkor veszi tudomásul, ha a RETURN gombot lenyomjuk! Tehát: mozgassuk a kurzort a hiba helyére, javítsunk, majd (ha a sorban több javítanivaló nincs) nyomjuk le a RETURN gombot! (Akár a sor végén van a kurzor ebben a pillanatban, akár nem.) Csak ekkor történik meg a gépen belül is a javítás.

A RETURN lenyomása előtt egy-egy soron belül akárhányszor javíthatunk, és ha a RETURN gombot már lenyomtuk, a kurzorral akkor is akárhányszor visszatérhetünk a sorra, újból javíthatjuk. Mindig a legutolsó változat az érvényes.

A javítógomb és a kurzormozgató gombok még más érdekes feladatokhoz is felhasználhatók, de ezeket majd csak a későbbiekben tárgyaljuk.

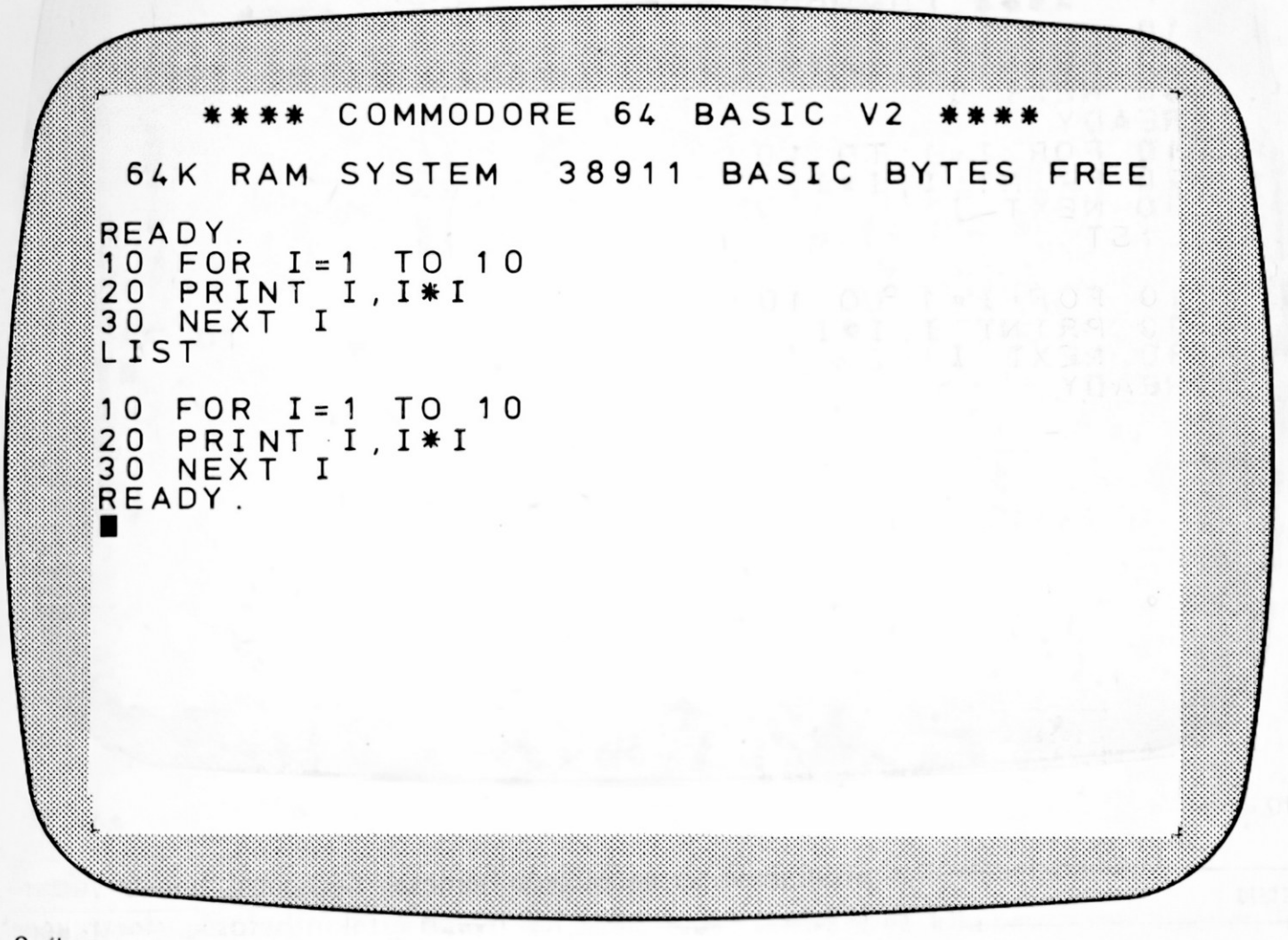
LIST

Ha látni akarjuk a programunk érvényes változatát, gépeljük be ezt a parancsot:

LIST

(A továbbiakban már nem említjük külön, hogy *minden egyes sor begépelését a RETURN sorvége gombbal kell befejezni.)*

A LIST parancs hatására megjelenik a képernyőn a teljes lista, majd a READY.



9. ábra

A későbbiekben, ha hosszabb programjaink lesznek, egyes sorok vagy programrészletek külön-külön való kilistázásának a lehetőségét is megismerjük. Akkor lesz majd szó teljes programsorok cseréjéről, beszúrásáról és törléséről is.

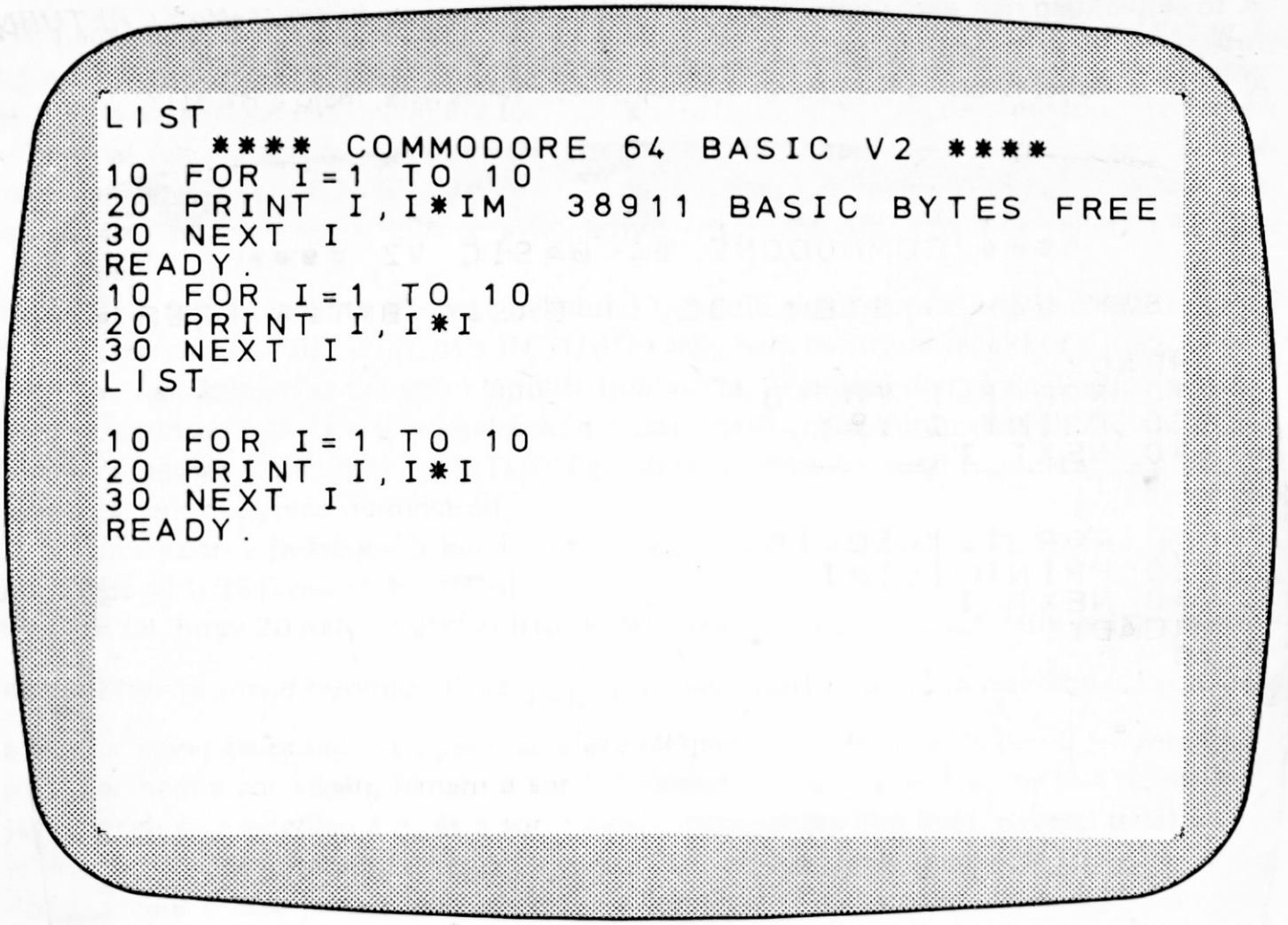
Ha egy soron belül javítani akarunk, és a sor több példányban is a képernyőn van, akkor teljesen mindegy, hogy melyik példányt javítjuk.

A $\begin{matrix}\text{CLR}\\\text{HOME}\end{matrix}$ törlőgomb lenyomása a kurzort a képernyő bal felső sarkába viszi.

Jellemző, hogy a központi egység mennyire csak azt nézi, ami a kurzor sorában van: nyomjuk le a törlőgombot, majd adjuk ki a LIST parancsot. A RETURN gomb lenyomása után a 10. ábrának megfelelő kép jelenik meg.

A legfelső sorban, ahol a LIST kiadásakor a kurzor állt, csak a LIST volt. A listázást vakon végrehajtotta a gép, nem törődve azzal, hogy mi volt a képernyő többi részén.

Ha a most kialakult kusza kép zavar bennünket, tartsuk lenyomva a SHIFT gombot, így nyomjuk le a törlőgombot! A kurzor a bal felső sarokba kerül, a képernyő törlődik. Most már zavartalanul listázhatunk. A LIST parancsot újra és újra kiadhatjuk; a program újra és újra megjelenik.



10. ábra

RUN

A programot természetesen nemcsak nézegetni, hanem futtatni is lehet. Adjuk ki (a könnyebb áttekinthetőség végett: képernyőtörlés után) a

RUN

parancsot. (A szó jelentése: fuss.)

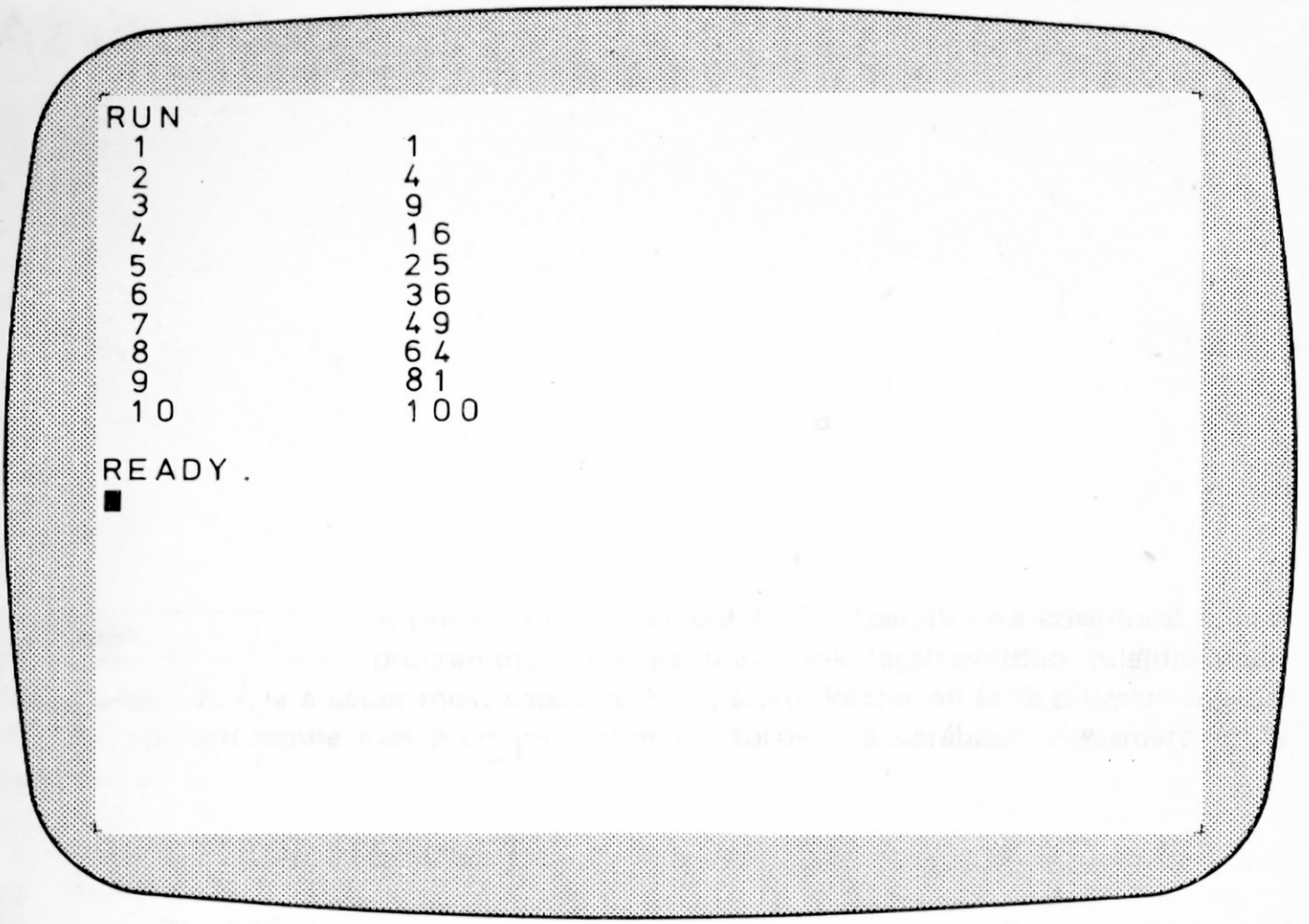
A program az egész számokat és a négyzetüket (a számok önmagukkal való szorzatát) írja ki 1-től 10-ig, majd a READY jelzi, hogy a gép újabb parancsra várakozik.
A RUN akárhányszor kiadható (képernyőtörlés nélkül és a program kilistázása nélkül is), ezzel a program újra és újra végrehajtható. Az eredmény a 11. ábrán látható.

NEW

Most adjuk ki a

NEW

parancsot. (A szó jelentése: új.) A parancs végrehajtását jelző READY megjelenése után kíséreljük meg a programot kilistázni vagy lefuttatni; egyik sem sikerül. A NEW parancs törli a programot. A gép úgy viselkedik, mintha most kapcsoltuk volna be. Bármit kezdhetünk elölről. (Kikapcsoláskor egyébként a központi egység a programot mindenképpen elfelejti, akár kiadtuk a kikapcsolás előtt a NEW parancsot, akár nem.)



11. ábra

A PROGRAM-FUTÁS LEÁLLÍTÁSA

Mielőtt az első fejezet végére érve kikapcsolnánk a gépet, ismerkedjünk meg a Commodore két vészfékjével.

A RUN STOP leállítógomb (néhány különleges esettől, pl. az ez ellen védekező programoktól eltekintve) minden programfutást leállít. Egyéb információk után megjelenik a READY, és folytathatjuk a munkát. (Mintaprogramunkkal ez is kipróbálható.)

Ha a leállítógomb esetleg nem használ, tartsuk lenyomva, és erősen üssünk rá a RESTORE helyreállító gombra. Ez a művelet (néhány különleges esettől, pl. az ez ellen külön védekező programoktól eltekintve) minden programfutást leállít. A képernyőtartalom eltűnik, csak egy READY feliratot látunk a bal felső sarokban. Ezután folytathatjuk a munkát.

Programunk a fenti két művelet egyikétől sem semmisül meg: kilistázhatjuk, újra elindíthatjuk.

Ha semmi más nem segít, ki kell kapcsolnunk a központi egységet. Az újbóli bekapcsolás előtt célszerű a kézikönyvek szerint kb. 20 másodpercet várni. Ilyenkor természetesen a gépben volt program elvész.

Egy dolog felől nyugodtak lehetünk: hibás parancsok, hibás programok, téves gombnyomogatás a Commodore-gépet nem ronthatja el. Ha fizikailag és elektromosan nem nyúltunk a géphez (nem szedtük szét, nem ejtettük le, nem kötöttük 220 V-nál nagyobb feszültségű hálózatra stb.), és a központi egység ki- és visszakapcsolása után mégsem áll vissza az alapállapot, *ez a gép hibája, nem a miénk.*

Az utasítások

PROGRAM

A program utasításokból áll.* Gépeljük be a következő mintaprogramot, hogy az utasítások legalapvetőbb tulajdonságait megismerhessük. (Ha a gépet most kapcsoltuk be, akkor készen áll az új program fogadására. Ha már van benne más program, akkor ezt töröljük a korábban megismert NEW paranccsal.)

```
10 REM -- KOR TERULETE
20 INPUT R
30 LET T=R↑2*π
40 PRINT T
50 END
```

12. ábra

A program begépeléséhez a felfelé mutató nyilat a RESTORE gombtól balra levő gombon találhatjuk meg. A π jelhez ugyanezt a gombot kell lenyomnunk, miközben lenyomva tartjuk az egyik SHIFT gombot vagy a Commodore-gombot.
Ez a program egy kör területét számolja ki. Ha a RUN paranccsal elindítjuk, a következő sor elejére kérdőjelet ír és várakozik. Most gépelhetjük be a kör sugarát. Legyen ez például 10. Ha az 1-es és a 0 számjegyet begépeltük, természetesen le kell nyomnunk a RETURN gombot is, különben a gép nem tudja, befejeztük-e már az adat beírását, vagy esetleg 100-at, 1000-et vagy mást akarunk írni. A RETURN lenyomása után a képernyő következő sorában megjelenik az eredmény: 314.159266, vagyis háromszáztizennégy egész és százötvenkilencezer-kétszázhatvanhat milliomod. (Akinek a ?REDO FROM START üzenet és újabb sorban megint kérdőjel jelent meg, az 10 helyett valószínűleg 1O-t írt: egyest

* Természetesen itt is és a továbbiakban is programon a Commodore—64-es gépen írt BASIC programokat, utasításon a Commodore—64-es gép gyári BASIC nyelvének utasításait értjük és így tovább, nem törődve azzal, hogy a könyvben szereplő fogalmak és állítások közül melyek volnának ennél szélesebb körben is érvényesek, és melyek nem.

és O betűt. Nyugodtan gépelje be a jó adatot, a 10-et, és megjelenik a jó eredmény. Akinek egyéb hibajelzés vagy hibás eredmény jelent meg, listázza ki a programot, és vizsgálja át jelről jelre, megegyezik-e azzal, ami itt a könyvben szerepel; legrosszabb esetben a NEW paranccsal törölje az egészet, és gépelje be újra.)
Ha utánaszámolunk, a 10 egység sugarú kör területe valóban nagy pontossággal 314,159266 területegység. Akinek kedve van, próbálja ki a programot más értékekre is!

REM

Most pedig nézzük végig az egészet pontról pontra! Az első sorban levő utasítás puszta megjegyzés. (A REM kulcsszó az angol „remark" szó rövidítése. Jelentése: megjegyzés.) Ebbe a sorba bármit írhattunk volna, ami a program azonosítását, megértését, működtetését segíti. A megjegyzések használata nem kötelező, csak célszerű. Látni fogjuk, hogy a programokat magyarázat nélkül sokszor nehéz áttekinteni. Hosszabb idő után a szerzőjük sem emlékszik, mit miért írt úgy, ahogyan írta. Úgy véljük, az a programozó jár el helyesen, aki meggondolja: ha egy év múlva a kezébe kerülne a program, mit nem értene belőle, mit kérdezne? Milyen választ várna a kérdéseire? Ez a válasz legyen a programban, a REM utasításokban!

SORSZÁMOK

A kulcsszó előtt sorszám (10) áll. Írjuk le most (a program után) ezt az újabb megjegyzéssort:

```
15 REM -- R A SUGAR, T A TERULET
```

Ha ismét kilistázzuk a programot, ezt látjuk:

```
10 REM -- KOR TERULETE
15 REM -- R A SUGAR, T A TERULET
20 INPUT R
30 LET T=R↑2*π
40 PRINT T
50 END
```

13. ábra

A 15-ös sorszámú sor automatikusan bekerült a 10-es és a 20-as közé; semmi nem utal arra, hogy utólag gépeltük be.
A programutasítások kilistázásának és végrehajtásának a sorrendjét ezek a sorszámok döntik el. A legkisebb sorszám a 0 lehet, a legnagyobb pedig 63999. Számozhatnánk a sorokat akár egyesével is, de ez célszerűtlen volna, mert akkor nem tudnánk új sort beírni a meglevők közé.

SOR JAVÍTÁSA

Ezzel tehát egyben megtanultuk a sorbeszúrás módját is: kiválasztunk egy eddig nem használt sorszámot, és ezzel jelölve egyszerűen leírjuk az új sort. Mi történik, ha meglevő sorszámmal új sort írunk? Próbáljuk ki például ennek a sornak a begépelésével:

```
10 REM -- R SUGARU KOR TERULETE
```

A listázás után azt látjuk, hogy az eredeti 10-es sor eltűnt, helyette a programban már az új 10-es sor van. Ha pedig csak a sorszámot írjuk le, például a 15-öst, és utána rögtön lenyomjuk a RETURN gombot, akkor az így megjelölt sor eltűnik, törlődik. Erről újabb listázással győződhetünk meg.
A REM kulcsszó után a sorba bármit írhatunk. Sem a szóköz, sem más nem kötelező úgy, ahogyan a példában szerepel; csak az áttekinthetőségre törekedtünk.
A következő utasítás a kör sugarának beolvasása. Ennek az utasításnak a hatására jelent meg a kérdőjel és kezdett a gép várakozni. Addig várt, míg a RETURN gombot meg nem nyomtuk. Ha a sor vége (a RETURN) előtt egy számot is beírtunk, akkor az lesz a sugár (az R változó) értéke. Ha pedig mást írtunk, mint számot, akkor a ?REDO FROM START (jelentése: kezdd elölről) üzenet, majd a következő sorban ismét a kérdőjel jelenik meg, és a gép vár tovább. Figyeljük meg, mi történik, ha a kérdőjelre rögtön (adat beírása nélkül) a RETURN gombbal válaszolunk! A jelenség magyarázatára még visszatérünk.
Mit jelent az, hogy „ha egy számot is beírtunk, akkor az lesz a sugár (az R változó) értéke"? Az adatoknak neveket is adhatunk, programunkban a sugárnak R a neve. Lehetne más is: RR, R1, SUGAR, QTYAFUL vagy bármi, ami

- betűvel kezdődik,
- utána csak betűket és számjegyeket tartalmaz,
- a belsejében nincs kulcsszó (így pl. az ERREMAJDSZUKSEGLESZ adatnév hibás, mert a REM betűsorozatot tartalmazza).

Az adatnévben szóközök is lehetnek, de ezek nem számítanak. Tehát SUGAR ugyanaz az adat, mint S U GA R. Az adatnévnek azonban csak az első két jele meghatározó. Ez azt jelenti, hogy ha két adatnév az első 2–2 jelében megegyezik (pl. DATUM és DARAB), akkor a gép azonosnak tekinti őket! Nagyon vigyázzunk, mert ez olyan hibát rejthet magában, amelyre semmiféle gépi hibajelzést nem kapunk!

VÁLTOZÓK

Az adatnak (más néven változónak) van tehát neve, pl. az R, és van értéke, pl. a 10. (Olyan ez, mint egy fiók, amelynek van egy ráragasztott címkéje, és van tartalma, de egyáltalán nem biztos, hogy a tartalom valóban az, ami a címkén áll. A fiókokat, amelyekben pl. ceruzák, tollak és radírok vannak, elláthatjuk olyan címkékkel, hogy CERUZA, TOLL és RADÍR, de olyanokkal is, hogy A, B és C. A programozók általában szeretik a rövid elnevezéseket, amelyeket könnyebb le-

írni. E könyv szerzői viszont jobban kedvelik — és az olvasónak szívből ajánlják — a tetszőlegesen hosszú, de értelmes elnevezéseket, amelyeket leírni talán nehezebb, de olvasni, megérteni könnyebb lesz.)
A programok elindításakor, a RUN parancs után az ilyen változók kezdőértéke automatikusan 0 lesz, és ez az értékük meg is marad, míg szándékosan meg nem változtatjuk, például míg az INPUT-ra valamilyen 0-tól különböző adatot meg nem adunk. Ez az oka annak, hogy ha a gép kérdésére rögtön a RETURN gombbal felelünk, vagyis nem írunk be semmi adatot, akkor az R-ben megmarad a kezdőérték, a nulla, vagyis a mintaprogramunk ilyenkor a nulla sugarú kör területét számolja ki.
Amint a program lefuttatásakor az eredményen láttuk, a gép tizedesvessző helyett tizedespontot használ. Ha törtszámot akarunk begépelni, nekünk is tizedespontot kell írnunk.

INPUT

Ha a programunk több adatot használ, írhatunk mindegyikre egy-egy INPUT utasítást, de egyetlen INPUT utasítással is beolvashatók. Az INPUT kulcsszó mögött (egymástól vesszővel elválasztva) annyi adatot sorolhatunk fel, amennyi csak a sorban kifér, például:

sorszám INPUT EGYIK,MASIK,HARMADIK

Ha programfuttatáskor ilyen utasításra kerül sor, akkor megjelenik a kérdőjel, és a program az első adatra vár. Ha ezt megadtuk, a következő sor elején két kérdőjel jelenik meg, és ettől kezdve mindig 2–2, míg csak az INPUT utasítás adatlistájában levő valamennyi adat értékét meg nem adjuk.
Begépelhetünk több adatot is egyszerre, egymástól vesszővel elválasztva. Ha például az előbbi INPUT végrehajtásakor a kérdőjel megjelenésére a

2,3.5,99 RETURN

választ adjuk, akkor az EGYIK értéke 2, a MASIK értéke 3,5, a HARMADIK értéke 99 lesz, és több kérdőjel nem jelenik meg. Vigyázzunk: ha

2,3,5,99 RETURN

a válaszunk (ha tehát tizedespont helyett véletlenül tizedesvesszőt írunk), akkor az EGYIK értéke 2 lesz, a MASIK értéke 3 (mert a gép azt hiszi, hogy vége az adatnak, hiszen vesszőt talált), a HARMADIK értéke 5 (mert a gép azt hiszi, hogy ez már az újabb adat), majd ?EXTRA IGNORED (jelentése: a többlet figyelmen kívül hagyva) hibaüzenet jelenik meg, jelezve, hogy a gép azt hitte, a három kérdésre négy választ kapott.
Ha a program több kérdést is feltesz, érdemes a felhasználót (aki nem feltétlenül ismeri a program belsejét) tájékoztatni arról, hogy éppen mit kell begépelnie. Az INPUT utasításban az INPUT szó és az első adatnév közé idézőjelben a kérdés szövegét is előírhatjuk, például:

20 INPUT "MENNYI A KOR SUGARA";R

Végrehajtáskor a program kiírja a képernyőre az idézőjelben megadott szöveget, mögé ír egy kérdőjelet (az INPUT miatt), és vár a válaszra. Ilyen kiírandó szöveg az INPUT utasításban csak egy lehet, mégpedig az INPUT szó és az első adatnév között.

LET

Mintaprogramunk következő utasítása értékadás. Kulcsszava a LET, jelentése: (hadd) legyen, de ez elhagyható:

sorszám LET T = R ↑ 2 * π

ugyanazt jelenti, amit

sorszám T = R ↑ 2 * π

jelentett volna. Az utasítás végrehajtásakor a gép kiszámolja az egyenlőségjel jobb oldalán álló matematikai kifejezés értékét, és az egyenlőségjel bal oldalán álló változó ezt az értéket kapja meg. Ennek a változónak a korábbi értéke elvész, bármi is volt.
Az egyenlőségjel tehát itt nem azt jelenti, hogy valami valamivel egyenlő, hanem hogy valami valamivel egyenlő *legyen*.

ARITMETIKAI MŰVELETEK

A felhasználható műveletek és a jeleik:

\+ összeadás
– kivonás
* szorzás
/ osztás
↑ hatványozás

A műveletek számokra, változókra és a π-re vonatkozhatnak. A π gépbeli értéke 3,14159265. Például:

2 ↑ 10	jelentése	2^{10}
5 * 3–1	jelentése	5 és 3 szorzatából le kell vonni 1-et
(5 * 3)–1	jelentése	ugyanaz
5 * (3–1)	jelentése	3 és 1 különbségével kell 5-öt megszorozni

A felesleges többletzárójelek használata nem hiba: (((5)*(3))–1) ugyanaz, mint 5 * 3–1.

Vigyázzunk! 6/2*3 nem azt jelenti, hogy 6-ot osszuk el 2*3-mal, hanem 6-ot osszuk el 2-vel, és az eredményt szorozzuk 3-mal. Tehát nem $\frac{6}{2 \cdot 3}$, hanem $\frac{6}{2} \cdot 3$. Ha $\frac{6}{2 \cdot 3}$ értékét akarjuk géppel kiszámoltatni, írjuk így: 6/(2 * 3) vagy 6/2/3.
A műveleteket tehát a gép a következő sorrendben végzi el:
először a hatványozásokat,
utána a szorzásokat–osztásokat,
majd az összeadásokat–kivonásokat.
Ha azonos szintű műveletek állnak egymás mellett, ezeket a gép szigorúan balról jobbra hajtja végre. Zárójelezéssel a fenti sorrendet tetszőlegesen, a matematikában megszokott módon befolyásolhatjuk.
Ha egy számnak előjele van, az a szám részének minősül, tehát a gépi „tudomásulvétele" minden műveletet megelőz. Így például 2↑–10 jelentése: 2^{-10}, 5––4 jelentése 5–(–4), 5––––––––4 jelentése 5–(–(–(–(–(–(–4)))))) és így tovább.
Külön fejezet szól az értékadó utasításokban felhasználható függvényekről. Később tárgyaljuk a gép számolási pontatlanságából adódó hibákat is, amelyek esetenként azt is okozhatják, hogy A/(B * C) eredménye eltér A/B/C eredményétől.

PRINT

Mintaprogramunk 40-es sora a kiíróutasítást tartalmazza. A PRINT kulcsszó (jelentése: nyomtasd) után kell egymástól vesszővel vagy pontosvesszővel elválasztva felsorolni a kiírnivalókat. A számítástechnikában elterjedt szóhasználattal nyomtatásnak nevezünk minden kiírást, akkor is, ha nem papíron, hanem a képernyőn jelenik meg.

A képernyő egy-egy sorának negyvenkarakternyi helyét (tehát amin negyven betű, számjegy vagy speciális jel jelenhet meg) a gép négy darab, egyenként tízkarakternyi széles mezőnek tekinti, és ha a PRINT utáni listában vesszőt talál, akkor a soron következő szabad mező elejére lép. A gép akkor tekint szabadnak egy mezőt, ha a megelőző mezőnek is legalább az utolsó helye üres. Ha a megelőző mező az utolsó helyig foglalt, akkor a PRINT-be írt vessző hatására a gép a teljes következő mezőt kihagyja, és az utána levőbe ír. Természetesen ha egy kiírnivaló átnyúlik a következő mezőbe, és a PRINT-ben vessző van, akkor ebbe a következő mezőbe a gép akkor sem ír semmit, ha a következő kiírnivaló elférne a mező szabadon maradt részébe. Ha több vesszőt írunk a PRINT-be egymás után, akkor mindegyiknek a hatására egy-egy mezővel előbbre lép a kiírás. Pontosvessző hatására nincs előbbrelépés.

Ha számot kell kiírni, akkor először is megjelenik az előjel (pozitív szám vagy nulla esetén a pluszjel helyett egy szóköz), majd a szám maga, utána pedig a gép kiír még egy elválasztó szóközt is.

A PRINT utasításban is lehet idézőjelek közti szöveg, mégpedig az INPUT-tól eltérően nemcsak elöl és nemcsak egy, hanem akárhol és akármennyi. Lehet olyan PRINT is a

```
LIST

10  REM -- KOR TERULETE
15  PRINT
20  INPUT "MENNYI A SUGAR";R
30  T=R↑2*π
35  PRINT
40  PRINT "A";R;"SUGARU KOR TERULETE";T
45  PRINT
50  END
READY.
RUN

MENNYI A SUGAR? 10

A 10 SUGARU KOR TERULETE 314.159266

READY.
```

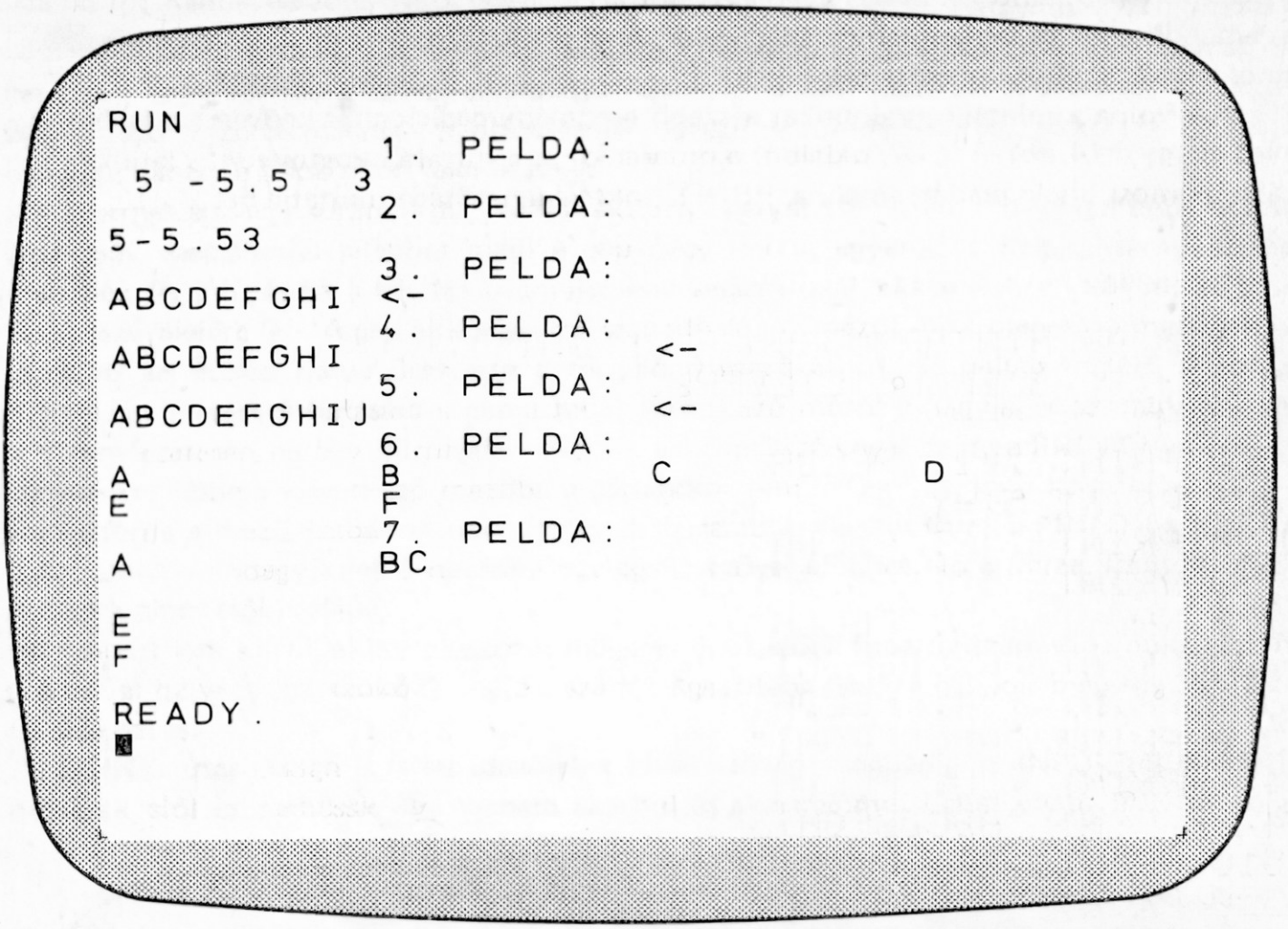
14. ábra

programunkban, amelyik csakis ilyen előre megírt szövegű kiírnivalót tartalmaz, sőt olyan is, amelyik nem tartalmaz semmi kiírnivalót; ahol a PRINT után sem változó, sem szöveg nincs, vagyis amikor üres sort kell kiírni.
Írhattuk volna a mintaprogramunkat a szebb eredménymegjelenítés kedvéért másképp is, mint ahogy a 14. ábrán (a 32. oldalon) a programot és a futtatási eredményt is látjuk.
Lássunk most olyan példát, amely a PRINT sokféle lehetőségét mutatja be:

```
10 REM -- MINTAPROGRAM A 'PRINT'
20 REM -- SZEMLELTETESERE.
30 REM
1010 PRINT ,"1. PELDA:"
1020 PRINT 5;-5;3
1110 PRINT ,"2. PELDA:"
1120 PRINT "5";"-5.5";"3"
1210 PRINT ,"3. PELDA:"
1220 PRINT "ABCDEFGHI","<-"
1310 PRINT ,"4. PELDA:"
1320 PRINT "ABCDEFGHI",,"<-"
1410 PRINT ,"5. PELDA:"
1420 PRINT "ABCDEFGHIJ","<-"
1510 PRINT ,"6. PELDA:"
1520 PRINT "A","B","C","D","E","F"
1610 PRINT ,"7. PELDA:"
1620 PRINT "A",
1630 PRINT "B";
1640 PRINT "C"
1650 PRINT
1660 PRINT "E"
1670 PRINT
1680 PRINT "F"
1690 END
```

15. ábra

Ez a mintaprogram a PRINT néhány jellegzetes hatását szemlélteti. Az 1. példa három számot, a 2. példa három szöveget ír ki. Jól megfigyelhetjük a számoknál az előjel helyét és a szám után kihagyott elválasztó helyet. A 3. példa az első mezőbe csak 9 karaktert ír, így a következő kiírnivaló a második mezőbe kerül. A következő példában ismét a második mezőbe kerülne a második kiírnivaló, de a két vessző kétmezőnyi előrelépést idéz elő. Az 5. példa azt mutatja be, hogy ha egy mező utolsó helye fel van töltve, akkor a következő mező is foglaltnak számít, vagyis a vessző hatására a kiírás az utána levő mezőben folytatódik. (Ez a helyzet akkor is, ha a mező utolsó helyét akár csak egy szám végére kerülő elválasztó szóköz foglalja is el.) A 6. példa mutatja, hogy a képernyő szélén való túllépés sorváltást eredményez: a sor ötödik mezője nem más, mint a következő sor eleje.



16. ábra

Egyébként ha egyetlen szám vagy szöveg nyúlna át a sor végén, az is elválasztva jelenne meg, és a következő sor elején folytatódna, akár logikus helyre esik a sorvégi elválasztás, akár nem. A 7. példa pedig azt mutatja, hogy a vezérlőjeleknek (a vesszőnek és a pontosvesszőnek) a szerepe az utasítás végén is ugyanaz, mint az utasítás belsejében. Közben példát látunk az üres PRINT utasításra is.

END

Eredeti mintaprogramunknak és a kiíró mintaprogramnak is az END az utolsó utasítása. A szó jelentése: vége. A végrehajtása a program leállítását jelenti. Ha a program végén van, akkor elhagyható is lenne, hiszen az utolsó utasítás után a végrehajtás úgysem folytatódhat. Az END igazi szerepét majd a bonyolultabb szerkezetű programokban fogjuk látni.

Megjegyzések, kiegészítések, tanácsok

A programjainkban tehát tudunk megjegyzéseket elhelyezni, adatokat beolvasni, velük matematikai műveleteket végezni, az eredményt áttekintő formában, magyarázó szövegekkel kiírni és a programot leállítani. Ezekkel az utasításokkal már komoly számításokat is végezhetünk.
Ideje, hogy megismerkedjünk a Commodore néhány olyan szolgáltatásával, amelyek a munkánkat könnyebbé teszik, és lehetőséget adnak arra, hogy a technikai kényelmetlenségek helyett a feladatra koncentráljunk.

SZÖVEGES VÁLTOZÓK

Létezhetnek szöveg értékű változók is. Az ilyen változókat azonban meg kell különböztetnünk a többitől, ezért a nevük végére dollárjelet kell írnunk:

```
SZOVEG$="JONAPOT!"
UJSZOVEG$=REGISZOVEG$
```

A programban lehet például X nevű, szám tartalmú változó és ugyanakkor X$ nevű, szöveg tartalmú változó is, a kettőnek semmi köze egymáshoz.
Szövegek között egyetlen művelet végezhető: az egyiknek a másik után illesztése. Jele a pluszjel. ☞ 17. ábra.

EGÉSZ VÁLTOZÓK

A numerikus (számot tartalmazó) változók között a Commodore–BASIC még egy altípust ismer: az egész változókat. Ez a mintaprogram jól mutatja a háromféle adattípus jelentését. X értéke a $\frac{10}{3}$ hányados értéke olyan pontossággal, ahogy csak a Commodore-gépen lehet. X% ennek az egész része (pontosan abban az értelemben, ahogy az egészrész függvény tárgyalásakor majd szerepel). X$ értéke pedig szöveg: négy jel, mégpedig egyes, nulla, törtvonal, hármas; anélkül, hogy a gép ezt bárhogyan is értelmezni próbálná. ☞ 18. ábra.

Figyelem!
Az egész típusú változókat a Commodore másként tárolja, mint a valósakat. Így egy egész változó tartalma –32768 és +32767 között lehet, a határokat is megengedve.

```
NEW

READY.
10  ELEJE$="TYU>"
20  VEGE$="<EM"
30  SZOVEG$=ELEJE$+VEGE$
40  PRINT  ELEJE$,VEGE$,SZOVEG$
RUN
TYU>          <EM           TYU><EM

READY.
```

17. ábra

```
NEW

READY.
10  X=10/3
20  X%=10/3
30  X$="10/3"
40  PRINT  X,X%,X$
RUN
 3.33333333          3          10/3

READY.
```

18. ábra

A SZÁMOK NORMÁLALAKJA

Próbáljuk ki, hogyan hajtja végre a Commodore a

PRINT 999999999

és a

PRINT 1000000000

parancsokat!

A gép a számára már ábrázolhatatlanul nagy számokat normálalakban jeleníti meg.

1E + 09 jelentése: $1 \cdot 10^9$

Ilyen formában mi is írhatjuk a számértékeket:

X = 3.14
X = 3.14E0
X = 0.314E1
X = 0.314E+1
X = 314E–2
X = 3140000000000E–12

mind ugyanazt jelentik. A program (INPUT-tal feltett) kérdéseire is megadhatjuk a választ ugyanilyen formában.

AZ INPUT NÉHÁNY SAJÁTOSSÁGA

Ha az INPUT-tal feltett kérdésre szöveggel kell válaszolnunk, a gép nem veszi tudomásul a szöveg elején esetleg begépelt szóközöket. Ha pedig csak szóközt gépelünk be, az INPUT üres szöveget fog előállítani. Eltünteti a szöveg legvégén levő szóközöket is.

Ha azt akarjuk, hogy a szöveget kezdő szóközök is részei legyenek a szövegnek, illetve éppen egy vagy több szóközből álló szöveget akarunk begépelni, tegyük a begépelt szövegünket idézőjelek közé. Ekkor az INPUT az idézőjelek között minden szóközt figyelembe vesz.

Tehát például a

888 INPUT X$

utasítás végrehajtásakor megjelenő kérdőjelre

ha azt gépeljük be, hogy *
akkor X$ értéke az lesz, hogy *

ha azt gépeljük be, hogy □□□A□□B□□C□□□
akkor X$ értéke az lesz, hogy A□□B□□C

ha azt gépeljük be, hogy □□□□
akkor X$ értéke üres szöveg lesz

ha azt gépeljük be, hogy "*"
akkor X$ értéke az lesz, hogy *

ha azt gépeljük be, hogy "□□□A□□B□□C□□□"
akkor X$ értéke az lesz, hogy □□□A□□B□□C□□□

és ha azt gépeljük be, hogy "□□□□"
akkor X$ értéke az lesz, hogy □□□□

(A fentiekben □ a szóközt jelenti.)

A PROGRAM BEGÉPELÉSÉT SEGÍTŐ FOGÁSOK

Most két begépelési trükk következik a PRINT használatát szemléltető program kapcsán:

Ha már az 1010-es sort leírtuk, nem érdemes az 1020-ast is leírni, ha a sorok javításában kellő gyakorlatot szereztünk. A kurzort visszamozgatjuk az 1010-es sorra, a sorszámban a nullát kijavítjuk 1-esre, a szövegben az 1-est 2-esre és kész, RETURN. Listázáskor láthatjuk, hogy létrejött a helyes 1020-as sor, miközben az eredeti 1010-es megmaradt. Nagyon hasonló sorokat tehát nyugodtan előállíthatunk egymásból is, ahelyett, hogy hosszadalmasan külön-külön begépelnénk őket.
PRINT helyett pedig nyugodtan írhatunk egy kérdőjelet. Ugyanazt jelenti, sőt ha a programot kilistázzuk, nem is a kérdőjel, hanem a PRINT szó jelenik meg.

SZÓKÖZÖK

A sorszám előtt, valamint a sorszám és az első szó között akárhány szóközt hagyhatunk: egyet se, félsornyit vagy bármenynyit. A listán a sorszám mindig a sor elején jelenik meg, utána lesz egyetlen szóköz, majd a sor tartalma. A sor további részén mindenütt annyi szóköz jelenik meg, amennyit begépeltünk, de jelentősége nincs: PRINTA ugyanúgy hajtódik végre, mint PRINT A. A szóközök csak a program olvashatóságát javítják. A kulcsszó (INPUT, PRINT stb.) belsejében azonban nem lehet szóköz, a kulcsszavakat tehát nem szabad például ritkítva írni.
Más a helyzet az idézőjelek közé írt szöveggel: itt a szóköz a szövegnek ugyanolyan fontos része, mint bármely másik jel. Ha viszont szám belsejébe írunk szóközt, az olyan, mintha ott sem lenne: 1 2 3 ugyanaz, mint 123.
Látjuk, hogy az adatok két alapvető típusa, a szöveg és a szám sok tekintetben különbözik. A különbségek még sokrétűbbek, mint eddig láttuk, de könnyen rendszerbe foglalhatók. Még visszatérünk rájuk.

PARANCS ÉS UTASÍTÁS

Sokszor használtuk már a „parancs" és az „utasítás" szavakat. Utasításnak a program részeit neveztük. Ezeket a RUN segítségével hajtjuk végre. Parancsnak például a NEW kulcsszót hívtuk, amelynek sorszáma nem volt, és a begépelésekor azonnal végrehajtódott.
Szinte minden kulcsszó használható parancsban és utasításban is. A különbségeket a 39. oldalon lévő táblázatban soroltuk fel.
A NEW és a LIST is használható utasításként, ha éppen azt akarjuk, hogy a programunk törölje vagy listázza önmagát. (A programfutás mindkét esetben megszakad.) Az INPUT nem használható parancsként, de a PRINT és az értékadás igen. Ezek segítségével a Commodore-géppel minden számítási feladatot is elvégezhetünk, használhatunk változókat és így tovább. Például az

```
R = 5
T = R ↑ 2 * π
PRINT T
```

Parancs	*Utasítás*
sorszám nélkül kezdődik (pl. PRINT X)	sorszámmal kezdődik (pl. 10 PRINT X)
begépeléskor azonnal végrehajtódik	RUN hatására hajtódik végre a tárban levő többi utasítással együtt
a végrehajtás a begépelés sorrendjében megy végbe	a végrehajtás a sorszámok sorrendjében megy végbe
végrehajtás után elvész, nem listázható	végrehajtás után megmarad, akárhányszor listázható
újbóli végrehajtásához újra be kell gépelni, vagy — ha még látszik a képernyőn — a kurzort rá kell mozgatni	újbóli végrehajtásához elég a RUN parancsot újra kiadni

parancssorozat ugyanazt a végeredményt írja ki, mint a körterületet számoló programunk, ha abban az INPUT R hatására megjelenő kérdésre 5-öt gépelünk be.

SZÍNEK KEZELÉSE

Gépünk a PAL rendszerű színes adás vételére alkalmas tv-készülékkel tizenhatféle színű keretet, tizenhatféle színű alapot és tizenhatféle színű betűt tud megjeleníteni. Fekete-fehérben ezek persze csak a szürke különböző árnyalatai, de színesben a következők lehetnek:

Sorszám	*Szín*	*Angol rövidítés*
0	fekete	BLK
1	fehér	WHT
2	vörös	RED
3	zöldeskék, türkiz	CYN
4	lila, bíbor	PUR
5	zöld	GRN
6	sötétkék	BLU
7	sárga	YEL
8	világosbarna, narancs	
9	sötétbarna	
10	téglaszínű, hússzínű	
11	sötétszürke	
12	szürke	
13	világoszöld	
14	világoskék	
15	világosszürke	

A színek persze a tv-készülék beállításától függően változhatnak. Az egymás melletti színek a színérzetet erősen befolyásolhatják!

POKE

Az alap színét úgy változtathatjuk, hogy végrehajtatjuk a géppel a POKE 53281,*i* parancsot (vagy az ezt tartalmazó utasítást), ahol *i* a kívánt szín sorszáma. A háttér (keret) színének megváltoztatása ugyanígy történik, csak 53281 helyett 53280-at kell írnunk.
A betűk (és a kurzor) színe akkor változik az első 8 szín valamelyikére, ha lenyomva tart-

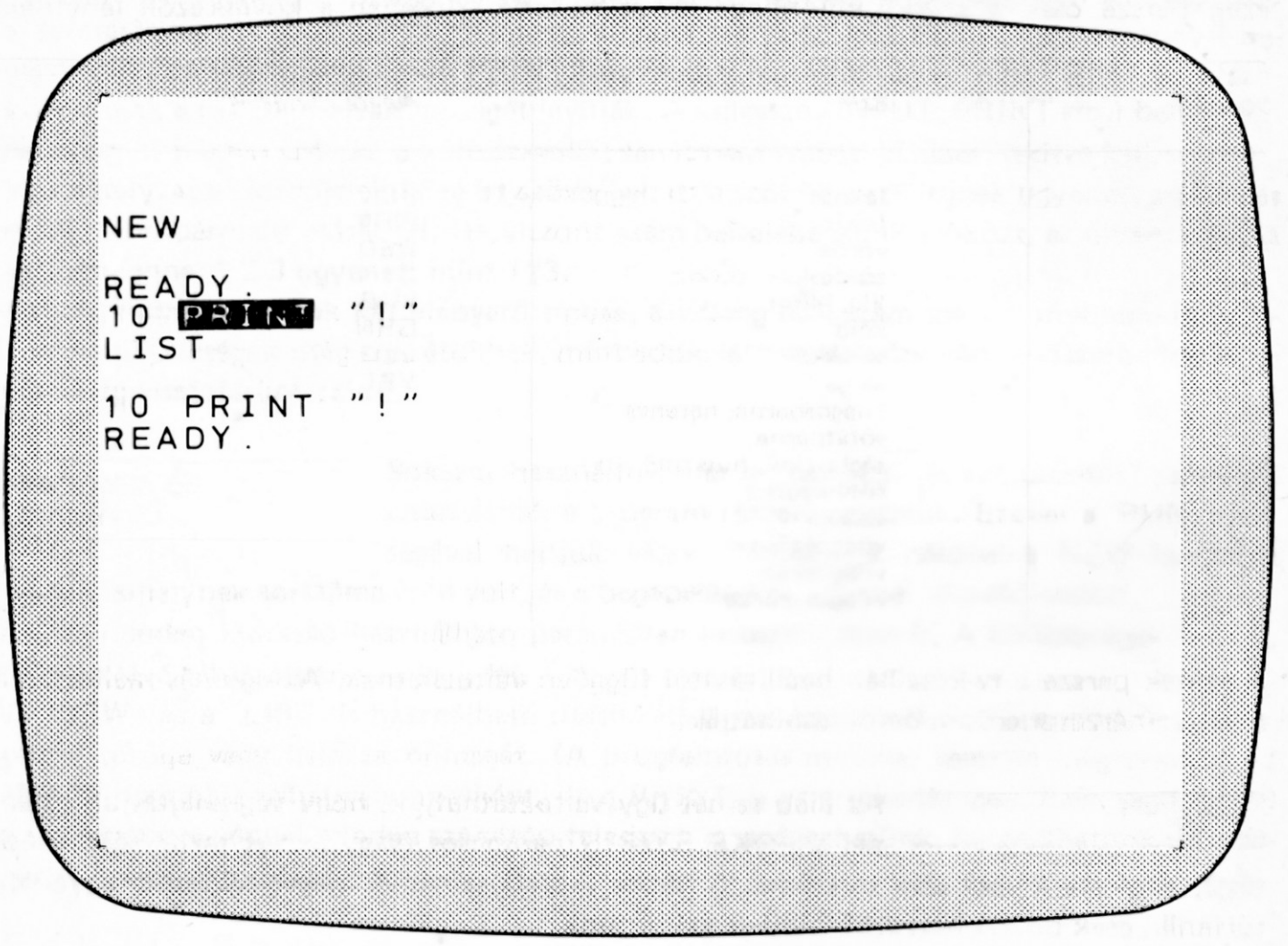
juk a CTRL vezérlőgombot, és lenyomjuk 1-től 8-ig valamelyik szám gombját. Az 1-est lenyomva a kurzor és minden ezután begépelt jel fekete lesz és így tovább. A színek angol neveinek rövidítései a megfelelő gombok előlapján ezt a választást segítik. A második 8 színt ugyanezzel a 8 számgombbal kaphatjuk meg, csak közben nem a CTRL gombot, hanem a Commodore-gombot kell lenyomva tartanunk.

Vigyázzunk az alap színének változtatásával: ha a betűk színéhez túl közeli vagy vele azonos színt választunk, lehet, hogy a teljes eddigi képernyőtartalom és a kurzor is eltűnik, mert beleolvad a megváltozott alapszínbe. Különösen ügyeljünk fekete-fehér képernyő használatakor: az egyes színpárok (pl. a vörös és a sötétkék) szinte teljesen egyforma szürkeként jelennek meg. Ilyenkor vakon kell begépelnünk a színt visszaváltoztató parancsot.

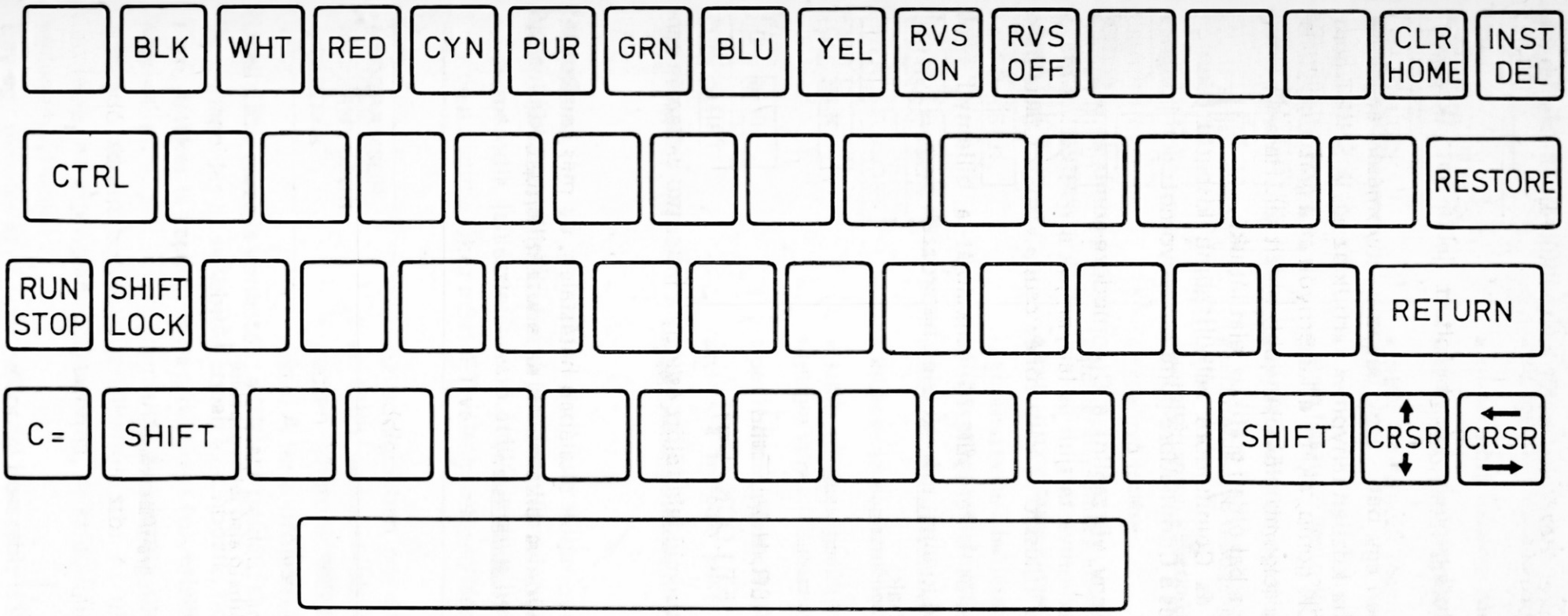
A vészleállítás (STOP és RESTORE egyszerre) a színeket visszaállítja olyanokká, amilyenek a bekapcsoláskor voltak.

Programjaink működése nem függ attól, hogy milyen színnel írtuk őket. Akár az egyes utasítások vagy utasításrészletek is lehetnek eltérő színűek. Listázáskor viszont mind egyformán jelennek meg.

A CTRL vezérlőgomb lenyomva tartásakor lenyomott 9-es gomb hatása: mostantól, amit begépelünk, inverz módon jelenik meg, még a szóköz is. (Vagyis az alap és a betű színe felcserélődik.) Ezt a hatást a RETURN vagy a CTRL 0 szünteti meg. Programunkban is használhatunk inverz jeleket, de listázáskor minden normál módon jelenik meg.



19. ábra



SPECIÁLIS JELENTÉSŰ GOMBOK

Annál érdekesebb, ha ezeket a váltójeleket idézőjelen belül használjuk. Erről később külön szólunk.

KARAKTER-KÉSZLET

A Commodore-gépben két beépített jelkészlet (karakterkészlet) van.

Alapállapotban egy betű- vagy jelgomb lenyomására az jelenik meg a képernyőn, ami a gombon van; ha közben lenyomva tartjuk az egyik SHIFT gombot, vagy le van nyomva a SHIFT LOCK gomb, akkor a képernyőn az a grafikus jel jelenik meg, amely a lenyomott betű- vagy jelgomb előlapján jobb oldalt áll. Ha a Commodore-gomb van közben lenyomva, akkor a bal oldali grafikus jelet látjuk.

A gép másik jelkészletében a SHIFT és Commodore nélküli gomb kisbetűt jelent, a SHIFT-tel lenyomott gomb nagybetűt, és a Commodore-gombbal lenyomott gomb ugyanazt, mint korábban.

Egyik jelkészletről a másikra átállni vagy visszaállni a Commodore-gomb és a SHIFT együttes használatával lehet: egyiket lenyomva tartjuk és lenyomjuk a másikat. A második jelkészletet használva tehát minden kulcsszó és változónév csupa kisbetűt tartalmaz.

A SPECIÁLIS BILLENTYŰK ÖSSZEFOGLALÁSA

A többfunkciós billentyűk fő funkcióját a billentyű felső lapján látható kiemelt méretű jelek határozzák meg.

Eszerint vannak:

- számbillentyűk (pl. 1 , 2 , 0 stb.)
- betűbillentyűk (pl. A , B , O stb.)
- jelbillentyűk (pl. : , = , . stb.)
- vezérlőbillentyűk (pl. RESTORE, CRSR, HOME stb.)
- váltógombok (pl. CTRL, SHIFT, SHIFT LOCK, C⌑)*

A másodlagos funkciók az elsődleges funkció jele fölött és/vagy a billentyű előlapján vannak feltüntetve.

A VÁLTÓGOMBOK

Önálló megnyomásuk általában hatástalan, de más gombokkal együtt megnyomva alkalmasak az adott billentyű különböző funkcióinak kiválasztására.

Gomb	*Hatása*
szám	számjegy begépelése
betű	nagybetű begépelése
jel	a billentyűn látható alsó jel begépelése
RETURN	sorvége jel bevitele
STOP	az éppen futó program leállítása

*A C ⌑ az úgynevezett Commodore-gomb jele.

HOME	a kurzor elküldése a képernyő bal felső sarkába
DEL	a kurzor előtti karakter törlése
CRSR	a kurzor mozgatása az alsó nyíl irányába (lefelé vagy jobbra)
SHIFT LOCK	a SHIFT billentyű rögzítése, amely a LOCK ismételt megnyomásával oldható
SHIFT szám	a szám feletti jel begépelése
SHIFT betű	a betű billentyűjének előlapján levő jobb oldali jel begépelése
SHIFT jel	a billentyűn levő felső jel begépelése
SHIFT RUN	a kazettán soron következő program betöltése és elindítása
SHIFT CLR	a képernyő törlése
SHIFT INST	karakter beszúrása a kurzor helyére
SHIFT CRSR	a kurzor mozgatása a felső nyíl irányába (felfelé vagy balra)
CTRL BLK	a kiírás színének beállítása feketére (ugyanígy állítható a szín fehérre, vörösre és így tovább, egészen sárgáig)
CTRL RVSON	a kiírás és az alap színének felcserélése (invertált írás)
CTRL RVSOFF	a kiírás és az alap színének visszacserélése
C= BLK	a kiírás színének beállítása narancssárgára (ugyanígy állíthatók a másodlagos színek világosszürkéig)
C= betű	a betű billentyűjének előlapján levő bal oldali jel begépelése
C= SHIFT	áttérés a második jelkészletre, illetve visszatérés az első jelkészletre
STOP RESTORE	a gép alapállapotának helyreállítása a tártartalom meghagyásával

Figyelem!

– Nyitó idézőjel után a vezérlőbillentyűk a fentiektől eltérő módon fejtik ki a hatásukat!
– A második jelkészletre való átállás után a betűbillentyűkkel kisbetűk gépelődnek be, míg a nagybetűket a SHIFT váltógomb segítségével állíthatjuk elő.

PROGRAMSOR, KÉPERNYŐSOR, UTASÍTÁS

Az eddigiekben egy képernyősorban mindig egy utasítás volt, se több, se kevesebb. Ez nem szükségszerű. Utasításokat nyugodtan írhatunk közös sorba, ha kettősponttal választjuk el őket. A fenti program a 20. ábra szerint is megírható.

Utasítások össze is vonhatók: a PRINT például tartalmazhat műveletet is. ☞ 21. ábra. Ez a megoldás az előbbitől abban különbözik, hogy előbb létrejött a SZOVEG$ nevű változó, értéket is kapott, és ha a program folytatódna, ezt az értéket később többször is felhasználhatná. Sőt, a programfutás befejezése után PRINT parancsban megjeleníthetjük, értékadó parancsban felhasználhatjuk stb. Az újabb esetben azonban ilyen változó nem jön létre; ha a program folytatódna, és szükségünk volna ugyanerre az adatra, ismét létre kellene hoznunk.

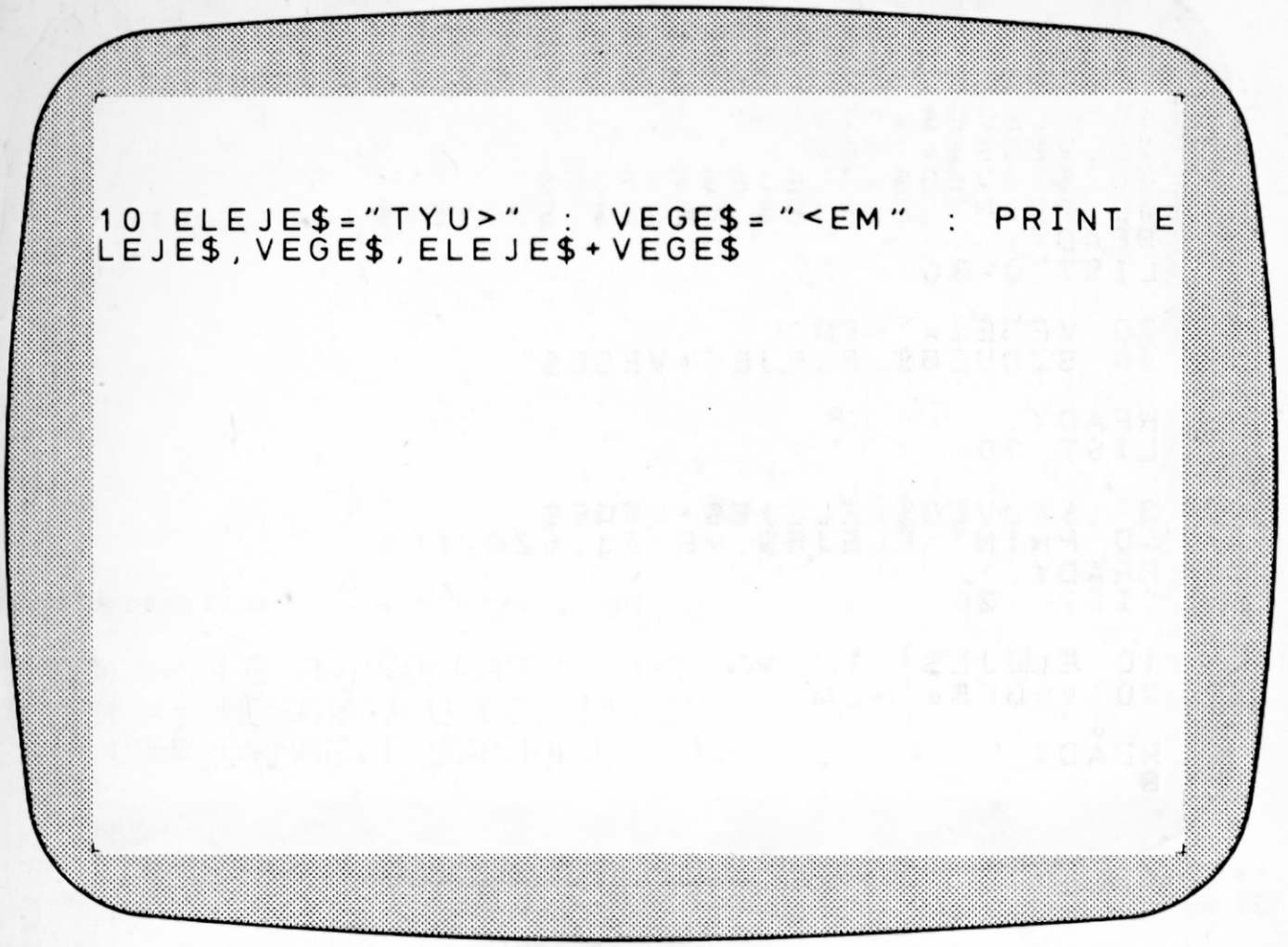
Egy-egy utasítás egy képernyősornál hosszabb is lehet ☞ 22. ábra.

```
10 ELEJE$="TYU>"  :  VEGE$="<EM"
30 SZOVEG$=ELEJE$+VEGE$
40 PRINT ELEJE$,VEGE$,SZOVEG$
```

20. ábra

```
10 ELEJE$="TYU>"  :  VEGE$="<EM"
40 PRINT ELEJE$,VEGE$,ELEJE$+VEGE$
```

21. ábra



22. ábra

Az egy sorszámhoz tartozó utasítássorozat a következő képernyősor végéig terjedhet. Az ilyen zsúfolt sorokból álló program is ugyanúgy működik, mintha minden utasításához külön sorszámot írtunk volna. Egy kicsit még gyorsabb is, és valamivel kevesebb gépi tárkapacitást vesz igénybe. De amit nyerünk időben és tárkapacitásban, annak a sokszorosát veszíthetjük áttekinthetőségben és a későbbi programmódosítások kényelmében, biztonságában.

Ha a programunk utasításai, utasításcsoportjai külön sorszámot kaptak, akkor részletenként is tudunk listázni, amint ez a 23—24. ábrán látható.
A listázóparancsban megadott határok tehát azt jelentik, hogy mi alá, illetve mi fölé nem szabad menni a listázás során. A listázás akkor is működik, ha nincs mit kilistáznia.

```
LIST

10  ELEJE$="TYU>"
20  VEGE$="<EM"
30  SZOVEG$=ELEJE$+VEGE$
40  PRINT  ELEJE$,VEGE$,SZOVEG$
READY.
LIST20-30

20  VEGE$="<EM"
30  SZOVEG$=ELEJE$+VEGE$

READY.
LIST  30-

30  SZOVEG$=ELEJE$+VEGE$
40  PRINT  ELEJE$,VEGE$,SZOVEG$
READY.
LIST  -20

10  ELEJE$="TYU>"
20  VEGE$="<EM"

READY.
```

23. ábra

```
LIST  11-39

20  VEGE$="<EM"
30  SZOVEG$=ELEJE$+VEGE$

READY.
LIST  20

20  VEGE$="<EM"

READY.
LIST  25


READY.
LIST  11-19


READY.
```

24. ábra

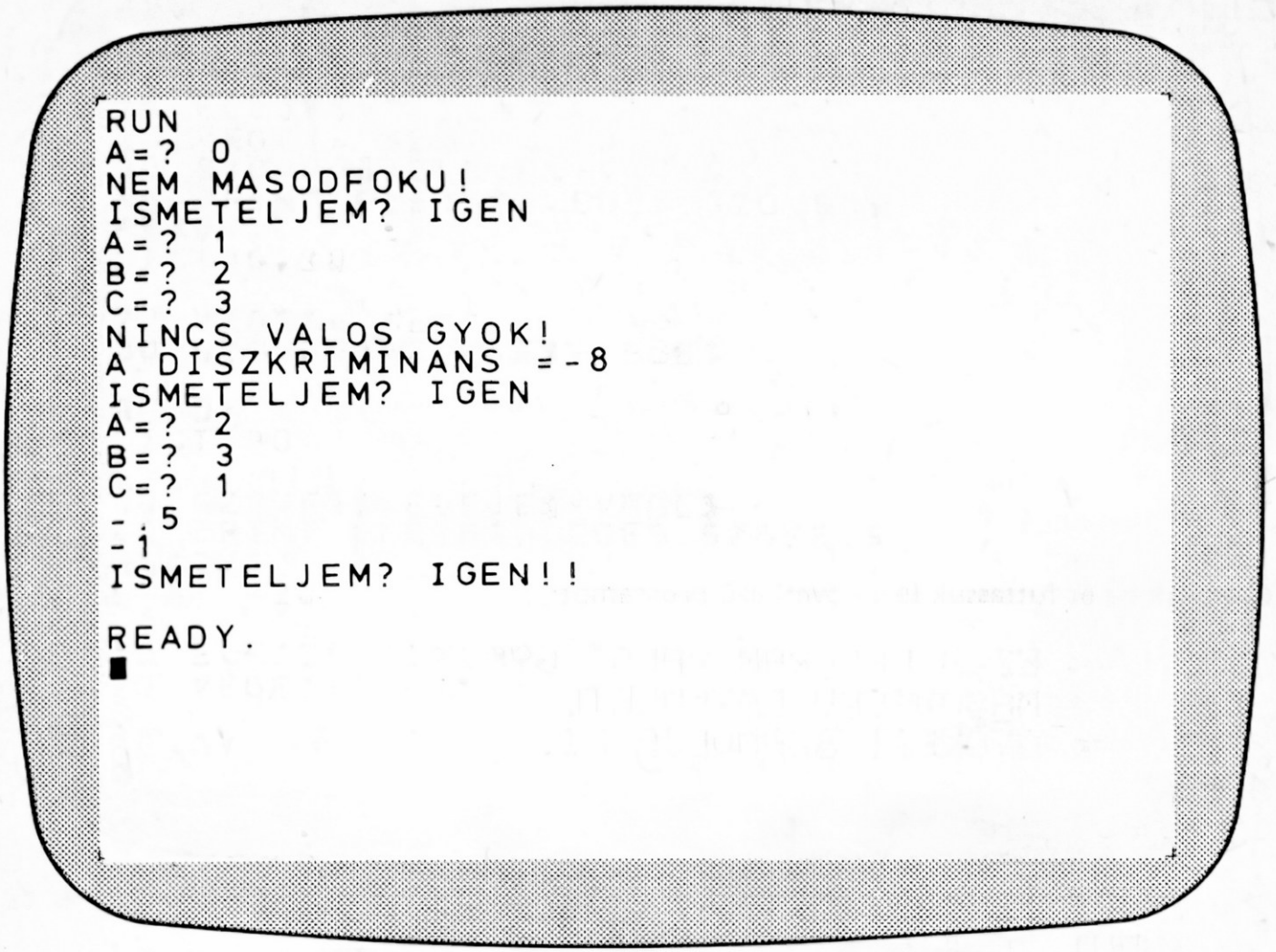
Elemi tevékenységek

Gépeljük be és futtassuk le a következő programot:

```
1 REM == EZ A PROGRAM VALOS GYOKU
2 REM == MASODFOKU EGYENLETEK
3 REM == GYOKEIT SZAMOLJA KI.
4 REM
10 INPUT "A=";A
15 IF A=0 THEN GOTO 1010
20 INPUT "B=";B
30 INPUT "C=";C
35 D=B↑2-4*A*C
36 IF D<0 THEN GOTO 2010
40 X1=(-B+D↑(1/2))/2/A
50 X2=(-B-D↑(1/2))/2/A
60 PRINT X1
70 PRINT X2
80 INPUT "ISMETELJEM";V$
90 IF V$="IGEN" THEN GOTO 10
99 END
1000 REM -- HA AZ A ERTEKE NULLA:
1010 PRINT "NEM MASODFOKU!"
1020 GOTO 80
2000 REM -- HA A DISZKRIMINANS NEGATIV:
2010 PRINT "NINCS VALOS GYOK!"
2020 PRINT "A DISZKRIMINANS =";D
2030 GOTO 80
```

25. ábra

A program megkérdezi az A értékét (10. sor). Ha ez 0, akkor a program végrehajtása az 1010-es sorral folytatódik (15. sor). Az 1010-es sor kiírja, hogy az egyenlet „NEM MASODFOKU!", majd a programvégrehajtás a 80-as sorral folytatódik.



26. ábra

Ha az A nem volt 0, akkor a programfutás a 15-ös soron áthalad, a program beolvassa a B és a C értékét, majd kiszámolja a diszkriminánst (20–35. sorok). A 36. sor megvizsgálja, vajon a diszkrimináns negatív-e, és ha igen, átadja a vezérlést a 2010-es sorra. Megjelenik a kiírás, hogy „NINCS VALOS GYOK!", kiíródik a diszkrimináns értéke (2010–2020. sorok), majd a vezérlés visszakerül a 80-as sorra.
Ha a diszkrimináns nem volt negatív, akkor a programvégrehajtás a 36-os soron is túljut. A 40–70. sorok kiszámolják és kiírják a másodfokú egyenlet két gyökének értékét.
Akármelyik úton folyt a programvégrehajtás, végül a 80-as sorhoz érkezik. „ISMETELJEM?" – kérdezi a program, és beolvassa a választ. Ha ez a válasz „IGEN", akkor a 10-es sorra, az A beolvasására tér vissza a vezérlés, vagyis elölről kezdi az egészet. Ha viszont ez a válasz bármi más (ha akár csak egyetlen jelben is különbözik az „IGEN" betűsorozattól), akkor az END hajtódik végre, és a programfutás leáll.
Végeredményben csak kétféle új utasítás volt ebben a programban.

GOTO

Egyikük a vezérlésátadás a 1020-as és 2030-as sorban. A

GOTO *sorszám*

jelentése: folytatódjon a programvégrehajtás a sorszámmal megjelölt sortól. (Ha ilyen

sor nem volna a programban, akkor a programfutás megszakadna.) A GOTO-t így is írhatjuk: GO TO. A GO jelentése: menj, a TO jelentése: -hoz, -hez, -höz.

IF THEN

A másik újdonság a feltételes utasítás a 15-ös, 36-os és 90-es sorokban. Az utasítás szerkezete:

IF *feltétel* THEN *utasítás*

ahol az IF jelentése: ha, és a THEN jelentése: akkor. **Ha** a feltétel teljesül, **akkor** végrehajtódik a THEN után álló utasítás; egyébként olyan, mintha ott sem lenne.
Az utasítás helyén bármi lehet, nemcsak GOTO, sőt nemcsak egyetlen utasítás, hanem utasítássorozat is, ahol az egyes utasításokat szokás szerint kettőspont választja el egymástól. Elvileg akár újabb IF is lehetne ebben az utasítássorozatban. A feltétel tehát nemcsak a THEN utáni egyetlen utasításra, hanem a sor további részeire is vonatkozik.
Ha a THEN után álló kulcsszó GOTO, akkor akár a GOTO, akár a THEN elhagyható. A 15-ös sor például

```
15 IF A=0 THEN GOTO 1010
15 IF A=0 THEN 1010
15 IF A=0 GOTO 1010
```

alakban egyaránt működik. Mi a könyvben az első, teljes alakot használjuk, mert olvashatóbbnak tartjuk.

FELTÉTELEK

Mi lehet a feltétel? Akár logikai kifejezés, akár matematikai.

IF *matematikai kifejezés* THEN . . .

hatására a THEN utáni rész akkor hajtódik végre, ha a kifejezés kiszámolt értéke nem nulla.

LOGIKAI KIFEJEZÉSEK

Nemcsak a numerikus kifejezéseknek van logikai értékük, hanem a logikai kifejezéseknek is van numerikus értékük. Például a

PRINT A=B

vagy bármiféle más

PRINT *feltétel*

utasítás vagy parancs hatására —1 jelenik meg, ha a feltétel igaz (pl. ha a fenti példában A egyenlő B-vel), és 0 jelenik meg, ha nem.
A logikai kifejezésnek kétféle értéke lehet: igaz vagy hamis. Ha igaz, akkor a feltétel teljesül, ha hamis, akkor nem. A jelek is eléggé egyértelműek:

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

>=
=> } nagyobb vagy egyenlő

<>
>< } nem egyenlő

Nagyon vigyázzunk a gép pontatlanságából adódó hibákra! Például ha A=7*7 és B=7↑2, akkor az A=B feltétel **nem** fog teljesülni, mert a gép szerint hétszer hét értéke 49, de hét a négyzeten értéke 49,0000001. Általában **nem** számíthatunk arra, hogy két, különböző módon kiszámolt érték egyforma lehet. Ilyenkor célszerűbb a két mennyiség különbségének abszolút értékét megvizsgálni és egy adott nagyon kis számhoz vagy maguknak az eredeti mennyiségeknek ezredrészéhez, tízezredrészéhez viszonyítani. Ha a különbség abszolút értéke ennél a kis értéknél kisebb, a két szám egyenlőnek vehető. Ehhez a témához kapcsolódik a fejezet végén található gyökvonó program is.
A feltételek össze is kapcsolhatók. Például az

IF *egyik feltétel* AND *másik feltétel* THEN . . .

utasítás THEN utáni része akkor hajtódik végre, ha az egyik és a másik feltétel **egyaránt** teljesül, az

IF *egyik feltétel* OR *másik feltétel* THEN . . .

utasítás THEN utáni része akkor hajtódik végre, ha az egyik és a másik feltétel **bármelyike** teljesül, az

IF NOT *feltétel* THEN . . .

utasítás THEN utáni része pedig éppen akkor hajtódik végre, ha a feltétel **nem** teljesül. (AND jelentése: és, OR jelentése: vagy, NOT jelentése: nem.)
Ezeknek a műveleteknek a segítségével igen bonyolult feltételek is összeállíthatók, de ilyenek használatát kezdőknek semmiképpen nem ajánljuk, inkább azt, hogy a bonyolult feltételeket szedjék szét egyszerűbbekre. A bonyolult feltételrendszerek, vizsgálatsorozatok megszervezésének szakmai fogásaira Commodore-sorozatunk egy későbbi kötetében vissza fogunk térni.
Már most bemutatunk egy speciális esetet, amikor egy szám (egy numerikus változó értéke) szerint kell több eset közül választanunk, több GOTO közül egyet végrehajtanunk. Az

```
ON I GOTO 1010,2010,3010,3010
```

utasítás ezt jelenti: folytatódjon a programvégrehajtás attól az utasítástól kezdve, amelynek sorszáma a GOTO utáni felsorolásban az I-edik. Ha tehát I értéke 1, akkor GOTO 1010, ha I értéke 2, akkor GOTO 2010, ha I értéke 3 vagy 4, akkor a gép a GOTO 3010 utasítást hajtja végre, ha pedig I értéke 0, vagy ha 4-nél nagyobb, akkor a programvégrehajtás úgy folytatódik, mintha az ON utasítás ott sem lenne. Ha az I értéke nem egész, akkor a gép annak egész részét veszi figyelembe. Ha pedig az I (illetve az I egész része) negatív vagy 255-nél nagyobb, akkor hibajelzést kapunk, és a programvégrehajtás megszakad. Elvileg tehát 256 eset közül választhatunk ezzel az utasítással; valójában ennyi nem fér el egy utasítássorban. A GOTO utáni listának üres eleme is lehet: az

```
ON I GOTO 10,,20
```

hatására ha I értéke 1, GOTO 10, ha I értéke 2, GOTO 0, ha I értéke 3, GOTO 20 hajtódik végre, ha pedig I értéke 0, 4 vagy 4-nél több (de 255-nél nem nagyobb), akkor a végrehajtás a soron következő utasítással folytatódik.

A GOTO után csak sorszámok állhatnak. Az I helyén azonban (mint mindenütt, ahol nincs kifejezetten tiltva) állhat bármilyen matematikai kifejezés is. A gép ezt kiszámolja, veszi az eredmény egész részét és aszerint hajtja végre az ON utasítást.
Két új utasítás bevezetésével nagyobb programjaink tagolása sokkal egyszerűbb lesz.

GOSUB

A GOSUB (a GO — menj — és a SUBROUTINE — alprogram — szavak összevonása; jelentése: hajtsd végre) felírása olyan, mint a GOTO-é:

GOSUB *sorszám*

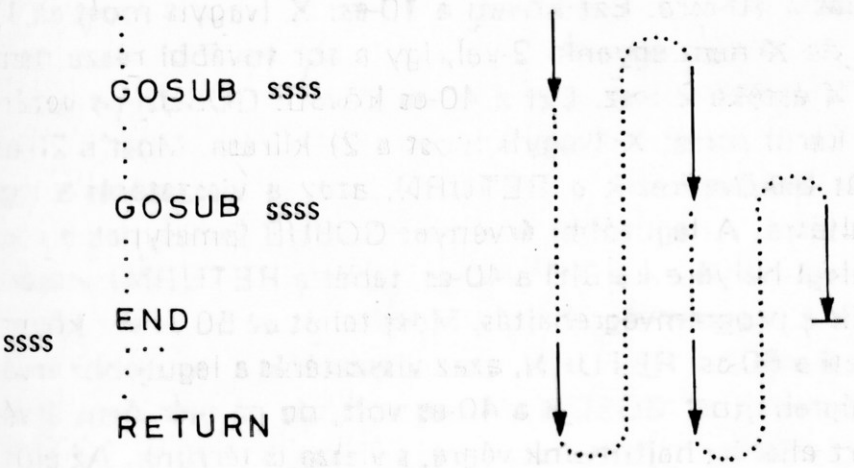
Ez az „emlékező GOTO". Megjegyzi, hogy a program melyik részén tart most, és a vezérlést ezután adja át a sorszámmal megjelölt sorba.

RETURN

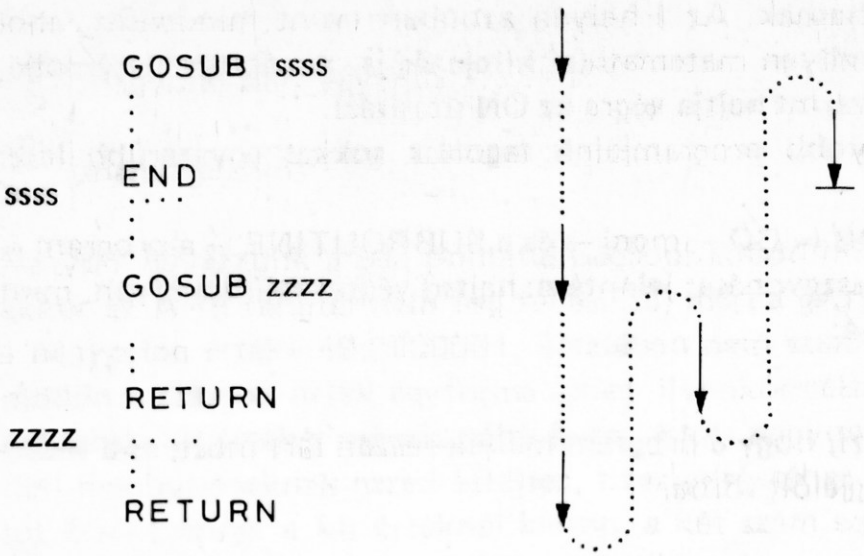
A RETURN (jelentése: térj vissza) hatása: megkeresi a nyilvántartásában, honnan adták ki a programban a legutóbbi, még érvényes GOSUB-ot, és az azt követő utasításra visszaadja a vezérlést; ezt a címet egyben törli is a nyilvántartásából.
Vagyis: a GOSUB aktivizál egy külön programrészletet, alprogramot, szubrutint, a RETURN pedig visszaadja a vezérlést az aktivizálás helyére.

27. ábra

A szubrutinhasználat lehetővé teszi, hogy programunk több pontján hajtsuk végre ugyanazt a tevékenységet anélkül, hogy az egészet többször le kellene írnunk. A végrehajtás ilyenkor a következőképpen folyik:



28. ábra



29. ábra

A GOSUB utasítással hívott, aktivizált eljárás természetesen tartalmazhat újabb GOSUB utasítást és így tovább. Az ilyen szubrutinokat egymásbaágyazottan hívott vagy röviden egymásbaágyazott szubrutinoknak is nevezzük.
Az egymásbaágyazott szubrutinhívás vezérlésátadási viszonyait a 29. ábra szemlélteti.
A GOSUB és a RETURN használatát a következő példa az elképzelhető legbonyolultabb eset kapcsán mutatja be:

```
1 X=1
2 GOSUB 10
3 PRINT 4
4 END
10 PRINT X
20 IF X=2 THEN RETURN
30 X=2
40 GOSUB 10
50 PRINT 3
60 RETURN
```

30. ábra

A program először az 1-es utasítást hajtja végre: X értéke 1 lesz. Ezután a 2-es utasítás következik: GOSUB-os vezérlésátadás a 10-esre. Ezt követi a 10-es: X (vagyis most az 1) kiírása. Ezután a 20-as kerül sorra: de X nem egyenlő 2-vel, így a sor további része nem érdekes. Most a 30-as következik: X értéke 2 lesz. Ezt a 40-es követi: GOSUB-os vezérlésátadás a 10-esre. Ezután a 10-es kerül sorra: X (vagyis most a 2) kiírása. Most a 20-as következik: a feltétel teljesül, tehát bekövetkezik a RETURN, azaz a visszatérés a legutóbbi érvényes GOSUB utáni utasításra. A legutóbbi érvényes GOSUB (amelynek a végrehajtása hatására a vezérlés a jelenlegi helyére került) a 40-es, tehát a RETURN hatására most az 50-es utasítással folytatódik a programvégrehajtás. Most tehát az 50-es sor következik: a 3-as szám kiírása. Ezt követi a 60-as: RETURN, azaz visszatérés a legutóbbi érvényes GOSUB utánra. A legutóbb végrehajtott GOSUB a 40-es volt, de ez már nem érvényes, mert hatására egy teljes, lezárt eljárást hajtottunk végre, s vissza is tértünk. Az előt-

te levő GOSUB a 2-es volt, amelyre még nem volt visszatérés, tehát ez még érvényes. A legújabb RETURN nem vonatkozhat másra, mint erre. Ezután tehát a 3-as sor következik: a 4-es szám kiírása. Végül a 4-es kerül sorra: vége.

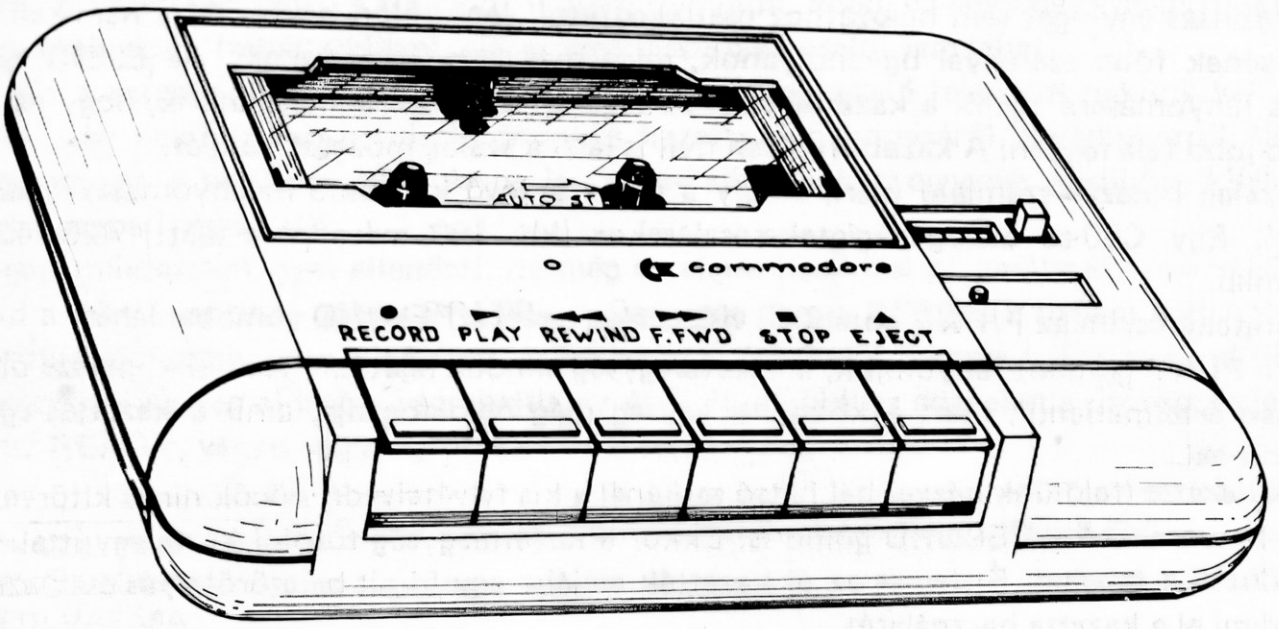
Reméljük, hogy a fenti rövid példa elegendő volt a GOSUB és a RETURN rugalmas felhasználási lehetőségeinek szemléltetésére (és arra is, hogy az olvasó elborzadjon a GOSUB és a RETURN kusza használatától). Még a gyakorlott felhasználónak is azt tanácsoljuk, hogy a GOSUB-ot lehetőleg ne használja másra, mint egy-egy önálló funkciót megvalósító programrészlet, modul különválasztására és ennek a megfelelő helyen való aktivizálására. A GOSUB-bal elérhető trükkök sok lehetőséget nyújthatnak, de sok zavart is okozhatnak, ezért óvatosan bánjunk velük! Ez természetesen nem zárja ki azt, hogy a GOSUB-bal aktivizált modulból újabb GOSUB-bal más, a hierarchia még alacsonyabb szintjén levő modult aktivizáljunk, majd ugyanezen a hívási láncon át visszatérjünk.

Súlyos bonyodalmat okozhat, ha két programegység között GOSUB- és GOTO-kapcsolat is van. A GOSUB-nak és a RETURN-nek mindig párt kell alkotnia; ezt azonban a gép nem tudja ellenőrizni. Pontosabban: csak annyit tud ellenőrizni, hogy nem akarunk-e RETURN-t végrehajtani olyankor, amikor egyetlen érvényes, még nyitott GOSUB sincs, azaz amikor a visszatérési címek nyilvántartása üres. Valamennyi többi hibalehetőségre a programozónak kell ügyelnie.

ISMERKEDÉS AZ ADATTÁROLÓ ESZKÖZÖKKEL

A kazettás egység

A Commodore–64 nem működik akármilyen magnetofonnal, csak azzal, amelyik külön ehhez a géphez készült:



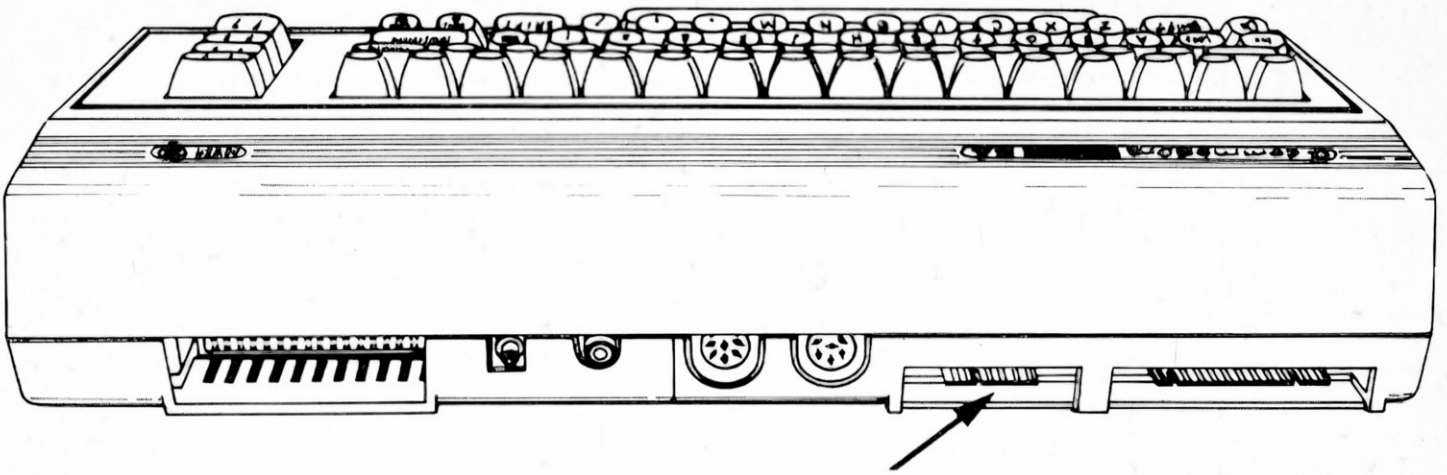
31. ábra

Ha más magnetofont akarunk a géphez használni, ahhoz külön illesztőegységre van szükségünk. A továbbiakban csak a gyári Commodore-magnetofonnal foglalkozunk.

Ennek a magnetofonnak, pontosabban kazettás egységnek jellegzetessége, hogy hangfelvételek készítésére, illetve lejátszására alkalmatlan, viszont műszaki paramétereit az adatrögzítés speciális igényeinek kielégítésére tervezték, így a készülékkel a Commodore megbízhatóbban üzemeltethető, mint az e célra átalakított hagyományos magnetofonokkal.

ÖSSZEÁLLÍTÁS

A központi egység hátulján (az 5-ös és 6-os számjegyek gombjaival körülbelül egyvonalban) megtalálhatjuk azt a csatlakozót, ahová a kazettás egység csatlakoztatható.



32. ábra

Csak egyféleképpen illeszthetőek össze; fordítva ne is erőltessük!
A kazettás egységet sem hálózathoz csatlakoztatni, sem külön bekapcsolni nem kell. Kezelésének főbb szabályai ugyanolyanok, mint más magnetofonoknál. Az EJECT gomb erős lenyomására nyílik a kazettafedél. A kazettát úgy kell behelyeznünk, hogy az üres orsó jobb felé legyen. A kazettafedélen nyíl is jelzi a szalag mozgási irányát.
A szalag hosszát számláló méri, amely a mellette levő kis gomb megnyomásával nullázható. Egy C60-as szalag végigtekercselésekor (kb. 100 másodperc alatt) 420—430-at számlál.
Előretekercselni az F.FWD gombbal, visszatekercselni a REWIND gombbal lehet.
Ha a PLAY gombot lenyomjuk, a kazettaegység elindul: lejátszás kezdődik, persze önmagában értelmetlenül, mert a központi egység még nem fogadja, amit a kazettás egység küld neki.
Ha a kazetta (felőlünk nézve) bal hátsó sarkánál a kis felvételvédő pöcök nincs kitörve, akkor lenyomható a RECORD gomb is. Ekkor a kazettaegység törölni kezd, egyúttal megkezdhető a felvétel. Érdemes az új kazetták elejébe egy kicsit beletörölni, és csak azután kezdeni el a kazetta használatát.
Az F.FWD, REWIND, PLAY és RECORD gombok hatását közülük bármelyik másik vagy a STOP gomb szünteti meg. A felvétel (törlés) és a lejátszás a szalag végén automatikusan is kikapcsol, az előre- és visszatekercselés azonban nem.
A kazettán adatokat és programokat is tárolhatunk. (Ajánlatos az adatok és a programok számára külön kazettákat fenntartani, mert így csökkenthetjük a keveredés, a véletlen törlés veszélyét.)

A PROGRAM KIMENTÉSE — SAVE

A program kimentését a

SAVE *"programnév"*

parancs végzi. Ha azt akarjuk, hogy a programunknak ne legyen neve, elég a SAVE kulcsszót begépelni. (SAVE jelentése: mentsd.)

A programjainknak 16 jelből álló nevük lehet. (Írhatunk hosszabb nevet is; ilyenkor ennek az első 16 jele kerül a szalagra, a többi elvész.) A programnévben betű, számjegy, szóköz, kötőjel és még más speciális jelek is lehetnek. Például a ! # $ % = * , . : ; jelsorozat is teljesen szabályos programnév.

A RETURN gomb lenyomása után a képernyőn a PRESS RECORD & PLAY ON TAPE (nyomja be a RECORD-ot és a PLAY-t a szalagegységen) felirat jelenik meg, és a gép várakozik. Most nyomjuk le a kazettaegység RECORD gombját gyors, határozott mozdulattal, mert amint a gombot nyomni kezdjük, a kiírási folyamat elindul. A képernyőn megjelenik az OK (rendben) és a SAVING (kimentés) felirat, a SAVING szó után a programnévvel. Ez egy pillanat múlva eltűnik, és a képernyő teljesen kiürül, minden eltűnik róla. Így is marad a kiírás végéig. Ez sokáig tart: a legrövidebb programnál is kb. 20 másodpercig. Közben a kazettaegység jelzőlámpája világít.
Ha a program kiírása befejeződött, a kazettaegység leáll, a jelzőlámpácska kialszik, és ezzel egy időben visszatér a képernyőtartalom, minden eddigi feliratával és egy READY felirattal együtt.
Ha új SAVE parancsot adunk ki anélkül, hogy a kazettaegységet közben kikapcsoltuk volna, a PRESS . . . és az OK szöveg már nem jelenik meg ismét, hanem a RETURN után azonnal megindul a kiírás.
Amíg a kazettaegységen a STOP-ot le nem nyomjuk, addig az egység normális működése nem is áll vissza (tehát addig pl. előre- vagy visszatekercselni sem lehet).
A gép a magnót csak elindítja-leállítja, a többi szalagkezelő feladatot nekünk kell ellátnunk. Így nekünk kell gondoskodnunk a kazetta behelyezéséről, továbbá arról, hogy a program ne a befűzőszalagra és ne is valamelyik másik programra kerüljön, kiférjen a szalag szabad részén stb.
A gép mindezeket nem ellenőrzi, de még az elemi beállítási paramétereket sem. Ha például a SAVE parancs után a PRESS . . . üzenetre nem a RECORD, hanem a PLAY gombot nyomjuk meg, vagyis a kazettaegységet nem felvételre, hanem lejátszásra indítjuk el, a központi egység akkor is végrehajtja az átvitelt, elküldi az adatokat a dróton, és jelenti, hogy READY, vagyis végrehajtotta a kimentést.

A KAZETTÁN LEVŐ PROGRAM BEOLVASÁSA

Ha kellően előretekercseltük a szalagot, adjuk ki a

LOAD

vagy a

LOAD *"programnév"*

parancsot (a LOAD jelentése: tölts.). Ekkor a PRESS PLAY ON TAPE (nyomja le a PLAY-t a szalagegységen) felirat jelenik meg, és a gép vár, amíg (határozott mozdulattal!) le nem nyomjuk a kazettaegység PLAY gombját. Erre OK és SEARCHING (keresés) felirat jelenik meg egy pillanatra, majd a kép eltűnik. Mintegy 15 másodperc múlva a kép visszatér, és az utolsó sorban a

FOUND *programnév*

feliratot láthatjuk (FOUND jelentése: megtalálva) vagy csak a FOUND szót, ha a szalagon soron következő program névtelen volt. Ez a felirat kb. 10 másodpercig látható. Ha ezalatt rájövünk, hogy baj van (pl. hibásan adtuk meg a beolvasandó program nevét), a

STOP gombbal még leállíthatjuk a beolvasást. (Természetesen leállíthatjuk később, menet közben is, de akkor a beolvasás már részben megtörtént, és ez felemás tárállapotot eredményez.) Ha nem állítjuk le, akkor egy villanásnyi időre megjelenik a LOADING (betöltés) felirat, majd ismét eltűnik a kép, és folytatódik az olvasás.

Ha a LOAD parancsban nevet is megadtunk, és a szalagon soron következő program éppen az, vagy ha a LOAD-ban nem adtunk meg nevet, akkor most megtörténik a következő program beolvasása és betöltése a tárba. Ha megadtunk nevet, és a szalagon nem az következik, akkor a gép továbbolvas a következő program kezdetéig, annak is kiírja a nevét egy FOUND üzenetben és így tovább. A kazettaegység indítása-leállítása itt is automatikus.

Ha új LOAD parancsot adunk ki anélkül, hogy a kazettaegységet közben kikapcsoltuk volna, akkor a PRESS . . . és az OK szöveg már nem jelenik meg, hanem a RETURN után megindul az olvasás. Amíg a kazettaegységen a STOP-ot le nem nyomjuk, addig az egység normális működése nem is áll vissza (tehát pl. addig előre- vagy visszatekercselni sem lehet).

A szalagra különösen érvényes a programozói mondás: amelyik program csak egy példányban van kimentve, az már el is veszett. Érdemes tehát mindent több példányban is szalagra írni.

Ha a READY előtt a ?LOAD ERROR (betöltési hiba) hibaüzenetet látjuk, akkor az olvasás valami miatt nem sikerült. Egyszer-kétszer még megkísérelhetjük, de valószínűbb, hogy a programnak egy másik példányát kell igénybe vennünk.

A PROGRAM ELLENŐRZÉSE – VERIFY

Tapasztalataink szerint a Commodore szalagegysége nem túl biztonságos. Kockázatos egy nagyobb programot ellenőrzés nélkül kiírni: ha esetleg a kiírás sikertelen volt, és ezt csak akkor vesszük észre, amikor megpróbáljuk visszaolvasni, akkor (a visszaolvasás kapcsán) az eredeti, tárbeli változat is elvész, azaz a teljes program tartalék nélkül megsemmisül.

A program ellenőrzése a

VERIFY *"programnév"*

vagy a

VERIFY

paranccsal történik. (A VERIFY jelentése: vizsgáld, igazold.) Működése és a hozzá tartozó szalagegység-kezelés ugyanolyan, mint a LOAD-nál, azzal a különbséggel, hogy a névvel megjelölt (illetve név hiányában a soron következő) programot nem helyezi el a tárban, csak összehasonlítja a tárban éppen bent levő programmal, anélkül, hogy a tárban levő program bármiben is változna. Ha a szalagon és a tárban levő program között akármilyen kis eltérés van (akár azért, mert a szalagon levő program visszaolvasása nem sikerült, akár azért, mert a visszaolvasás sikeres volt ugyan, de a szalagon levő program akár csak egyetlen jelben is különbözik a tárban levőtől), akkor a ?VERIFY ERROR üzenet jelenik meg a READY előtt, ha pedig a beolvasás hibátlanul sikerül, és a szalagon jelről jelre ugyanaz van, mint a tárban, akkor az OK.

PROGRAM BETÖLTÉSE ÉS AZONNALI FUTTATÁSA

Tartsuk lenyomva a SHIFT gombot, és nyomjuk le a RUN STOP gombot! Ugyanaz történik, mintha LOAD parancsot adtunk volna ki (programnév nélkül) és már hozzányomtuk volna a RETURN-t is. Azonnal elindul tehát a szalagon a soron következő program beolvasása. A beolvasás után megjelenik a képernyőn a READY, rögtön utána a RUN is, mintha már azt is begépeltük volna, és a program mindjárt el is indul.

PROGRAM BETÖLTÉSE PROGRAMBÓL

A LOAD kulcsszó programutasításban is használható. Ha a vezérlés rákerül, úgy működik, mintha a LOAD-ot parancsként adtuk volna ki, annyi különbséggel, hogy amikor a keresett programot megtalálja, nem tart tíz másodperces szünetet, csak villanásnyi időre jelenik meg a képernyőtartalom, és folyik a beolvasás tovább. (Persze ha az utasítás végrehajtásának pillanatában nincs egy korábbi olvasás miatt lenyomva a magnó PLAY gombja, akkor erre a felhasználót a szokott PRESS PLAY ON TAPE felirat kéri, és a gép vár a gomb lenyomásáig.)
A programutasítással betöltött program is azonnal magától elindul, azaz a LOAD után behívott program első végrehajtható utasítására kerül a vezérlés. Megvan tehát a lehetőség arra, hogy egy program beolvassa és elindítsa saját folytatását. A kazettaegység kezelését ezalatt természetesen magunknak kell megoldanunk.

ADATOK SZALAGRA ÍRÁSA

Ez a program adatokat ír a szalagra. Elindítása után kiírja a PRESS RECORD & PLAY ON TAPE üzenetet, és vár, míg bekapcsoljuk a szalagegységet. Ekkor egy villanásnyira megjelenik az OK üzenet, majd a gép kb. 15 másodperc alatt kiírja az adatállomány fejlécét; ezalatt a kép eltűnik. Utána sorra megjelennek a programunk kérdései, amelyekre a szokásos módon válaszolhatunk.

```
10 OPEN 255,1,2
20 SZ=1
30 PRINT SZ;". ADAT =";
31 INPUT ADAT$
40 PRINT#255,ADAT$
50 INPUT "VAN TOBB (IGEN/NEM)";FELEL$
60 IF FELEL$="NEM" THEN GOTO 90
70 IF FELEL$<>"IGEN" THEN GOTO 50
80 SZ=SZ+1 : PRINT : GOTO 30
90 CLOSE 255
99 END
```

33. ábra

Ha az adatok teljes hossza meghaladja a 192 jelet (az egyes adatokat a következő adattól elválasztó RETURN jelet is beleszámolva), akkor ezt az adagot a gép kiírja, automatikusan indítva és leállítva a szalagegységet, és folytatja a kérdések kiírását, az adatok összegyűjtését. Ha pedig a folytatásra NEM-et válaszolunk, akkor ismét elindul a szalag, és ki-

íródik, ami eddig még hátra volt. A gép tehát blokkokba gyűjtve ír a szalagra. Az egy-egy blokknyi adatot a tárban gyűjti össze.

OPEN

Az OPEN utasítás (OPEN jelentése: nyiss, nyisd) hatása: az adatállomány fejlécének létrehozása és szalagra írása, valamint annak a megjegyzése, hogy erre az adatállományra a program további részében 255-ös számú állományként hivatkozunk. (Ez a szám 0 és 255 között bármi lehetne.) Az utána álló 1-es szám jelenti a szalagegységet. Az ezutáni 2-es helyén még 1-es is lehetne, a felvitel akkor is sikerülne; a különbség annyi, hogy így a gép az adatállomány végére egy automatikusan előállított szalagvége jelzést, végjelet is felvisz, amit programból vizsgálhatunk, amikor az adatokat majd visszaolvassuk.
Ez az adatállomány névtelen lesz. Ha nevet is akarunk neki adni, tegyünk az utolsó paraméter után vesszőt, majd idézőjelek között adjuk meg a kívánt nevet. Ez is rákerül a szalagra.
A PRINT#255, . . . jelenti a 255-ös számú adatállományba való írást.
A PRINT tehát nem azonnali szalagra írást hajt végre, hanem először csak a tárban gyűlnek a kiírnivalók. Tényleges szalagra írás akkor zajlik le, ha
— összegyűlt 192 kiírnivaló jel,
— vagy végrehajtódik a CLOSE utasítás.

CLOSE

A CLOSE (jelentése: zárd le) kiírja, ami még a tárban hátravan, megjelöli az állomány végét, és (ha az OPEN-ben 2-sel előírtuk) kiírja a szalagvége blokkot.
Vigyázzunk! Ha egyetlen PRINT utasítással több adatot írunk ki, vagy ha a változólista végén vessző vagy pontosvessző van, akkor a gép semmiféle adatvégjelet nem ír, s így az adatok egybefolynak! (Ilyenkor persze megtehetjük, hogy a RETURN jelét magunk írjuk ki, ahova kell, külön adatként. A RETURN kódja 13, a kódból a jelet a később leírt CHR$ függvény állítja elő.)

ADATOK VISSZAOLVASÁSA A SZALAGRÓL

Olvasáskor a szalagegységet ugyanúgy meg kell nyitni, mint íráskor tettük, annyi eltéréssel, hogy az OPEN harmadik paraméterének értéke nem 1 vagy 2, hanem 0.
A szalagon levő adatállomány olvasásakor fizikailag mindig egy teljes blokk (192 jel) kerül a tárba, s a gép az adatokat ebből szedegeti folyamatosan elő. Ha elfogy, akkor továbbolvas a szalagon.
Ha az OPEN utasításban nem adunk meg nevet, akkor a gép a soron következő adatállományt veszi elő; ha nevet adunk, akkor addig olvas, amíg a megadott nevű állományt meg nem találja (vagy amíg le nem állítjuk).

```
10 OPEN 10,1,0
20 INPUT#10,X$
30 PRINT : PRINT X$,ST : PRINT
40 IF ST=0 THEN GOTO 20
50 CLOSE 10
60 PRINT "*** VEGE ***"
70 END
```

34. ábra

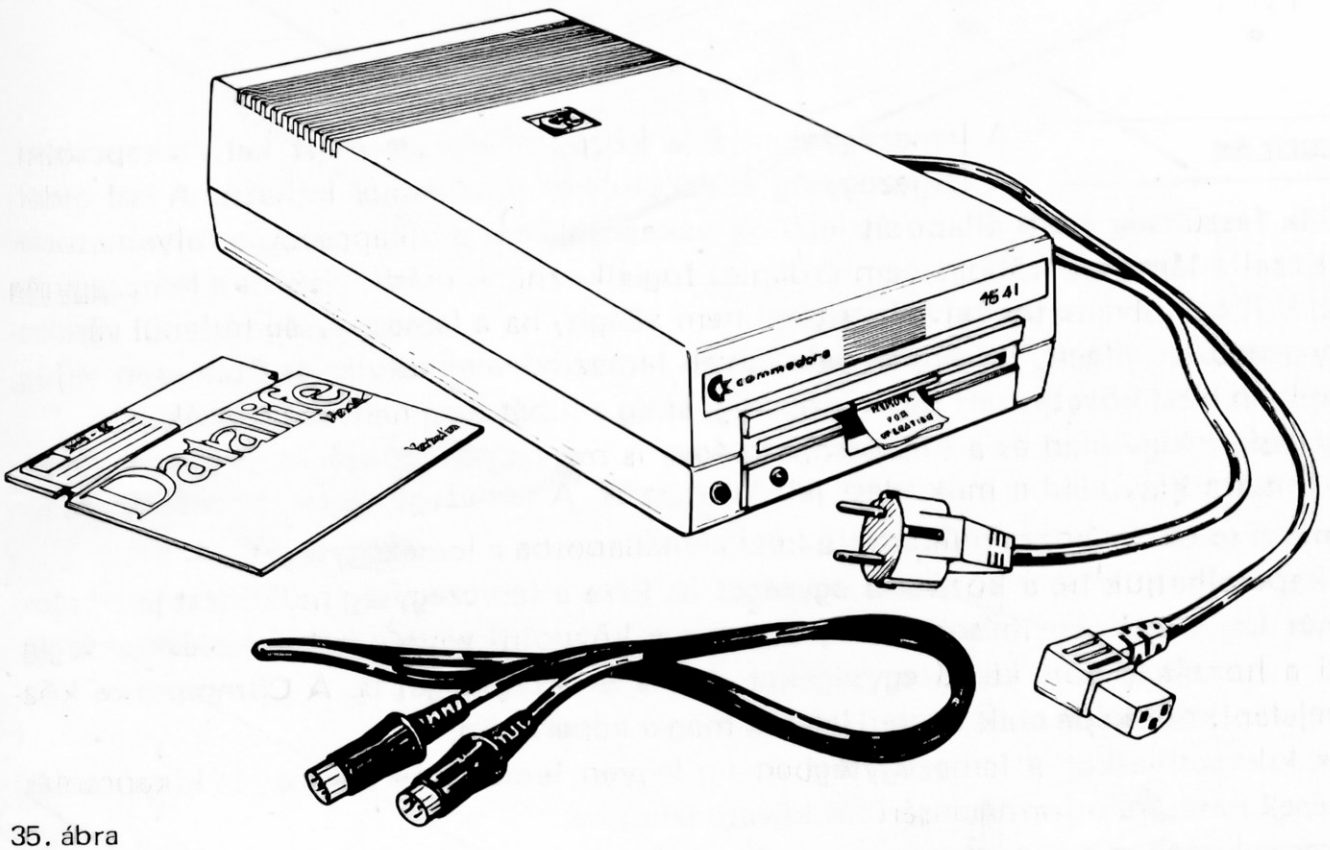
Az ST a STATUS (állapot) szó rövidítése. Az ST változót a gép maga állítja elő és tölti fel. Értéke 64 az adatállomány végén; 0, ha az adatállománynak nincs vége, és a beolvasás sikeres volt.

MEGJEGYZÉSEK A KAZETTAEGYSÉG HASZNÁLATÁHOZ

A kazettaegység használata lassú és nehézkes, számtalan hibalehetőséget is rejt a lemezhez képest, és a sokféle művelet, ami lemezzel könnyen megoldható, kazettával szinte megoldhatatlanul bonyolult. Igaz, hogy a lemezegység ma kb. négyszer annyiba kerül, mint a kazettaegység, de az egységnyi tárolókapacitású mágneslemez ugyanolyan olcsó, mint az ugyanolyan tárolókapacitású jó szalag. A lemezhasználattal járó lehetőségek és a biztonság miatt mindenképpen érdemes minden üzemszerű felhasználáshoz szalag helyett lemezt igénybe vennünk. A kazettaegység egyetlen helyénvaló felhasználási területe a játék, amikor az izgalmat külön fokozhatja, hogy vajon ma sikerül-e a programot jól betölteni, és ha igen, hányadik próbálkozásra.

A lemezhasználat alapfogalmai

Itt is, és a továbbiakban mindenütt, ha lemezegységről beszélünk, mindig a Commodore cég VC 1541 típusú lemezegységére gondolunk.



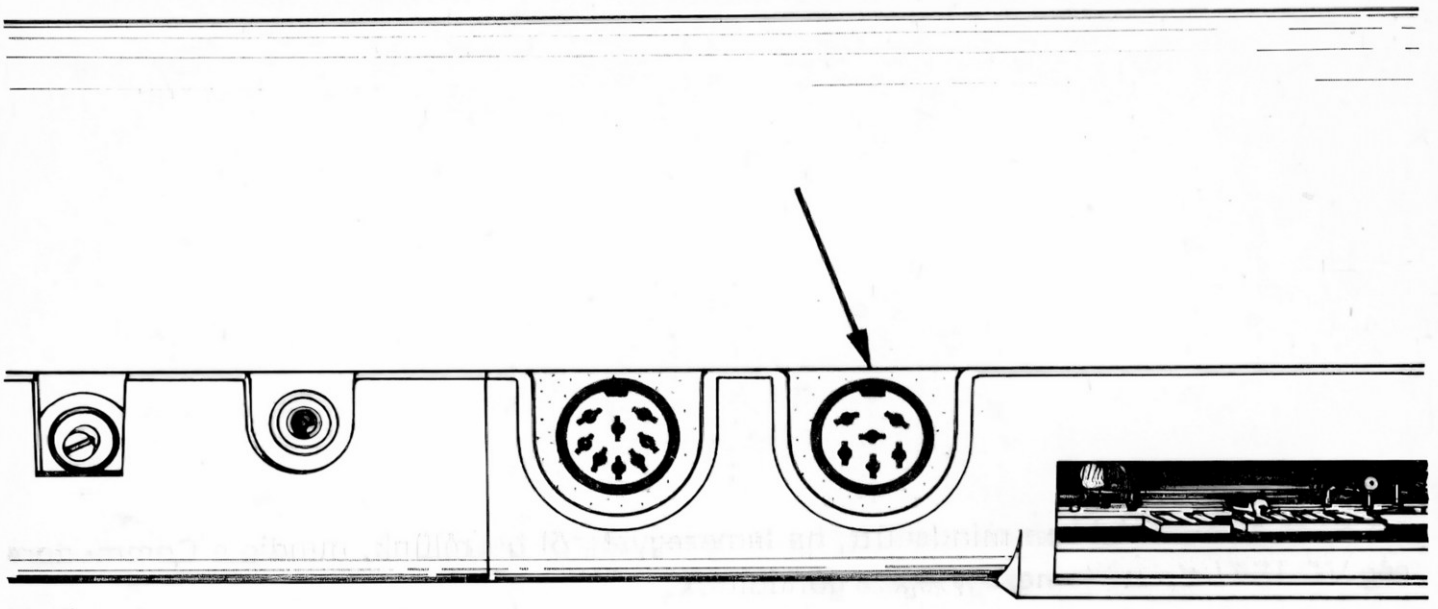
35. ábra

A lemezegységhez külön hálózati kábel és külön csatlakozókábel tartozik.

ÖSSZEÁLLÍTÁS

Győződjünk meg arról, hogy a központi egység és a lemezegység is kikapcsolt állapotban van. Ezután a lemezegység hálózati kábelét csatlakoztassuk a lemezegység hátoldalán levő aljzathoz (csak egyféleképpen lehet), és dugjuk be a konnektorba. A csatlakozókábel két végén egyforma tuchel van. Az egyiket dugjuk a lemezegység hátoldalán levő két tuchelaljzat közül bármelyikbe, a má-

sikat a központi egység hátoldalán levő két tuchelaljzat közül (hátulról nézve) a jobb oldaliba. (A másikba bele sem megy.)



36. ábra

BEKAPCSOLÁS

A lemezegységet is a központi egység előtt kell bekapcsolni. A lemezegység előlapján két jelzőlámpa látható. A bal oldali a készülék feszültség alatti állapotát jelzi. A bekapcsolástól a kikapcsolásig folyamatosan világít. Ezzel a lámpával többet nem érdemes foglalkozni. A másik viszont a lemezegység működéséről ad folyamatos felvilágosítást: nem világít, ha a lemezegység tétlenül várakozik; folyamatosan világít, ha éppen valamilyen lemezművelet folyik; és ütemesen villog, ha valamilyen hiba következett be, mindaddig, amíg a hibát meg nem szüntetjük.

Bekapcsoláskor kigyullad és a kikapcsolásig égve is marad a feszültségjelző lámpa, de néhány pillanatra kigyullad a működést jelző lámpa is. A lemezegység saját processzora ekkor ellenőrzi és ekkor hozza működésre kész alapállapotba a lemezegységet.

Ezután kapcsolhatjuk be a központi egységet is. Erre a lemezegység működést jelző lámpája ismét kigyullad egy másodpercre, ugyanis a központi egység bekapcsoláskor végigellenőrzi a hozzákapcsolt külső egységeket, így a lemezegységet is. A Commodore közismert bejelentkező képe csak ezután jelenik meg a képernyőn.

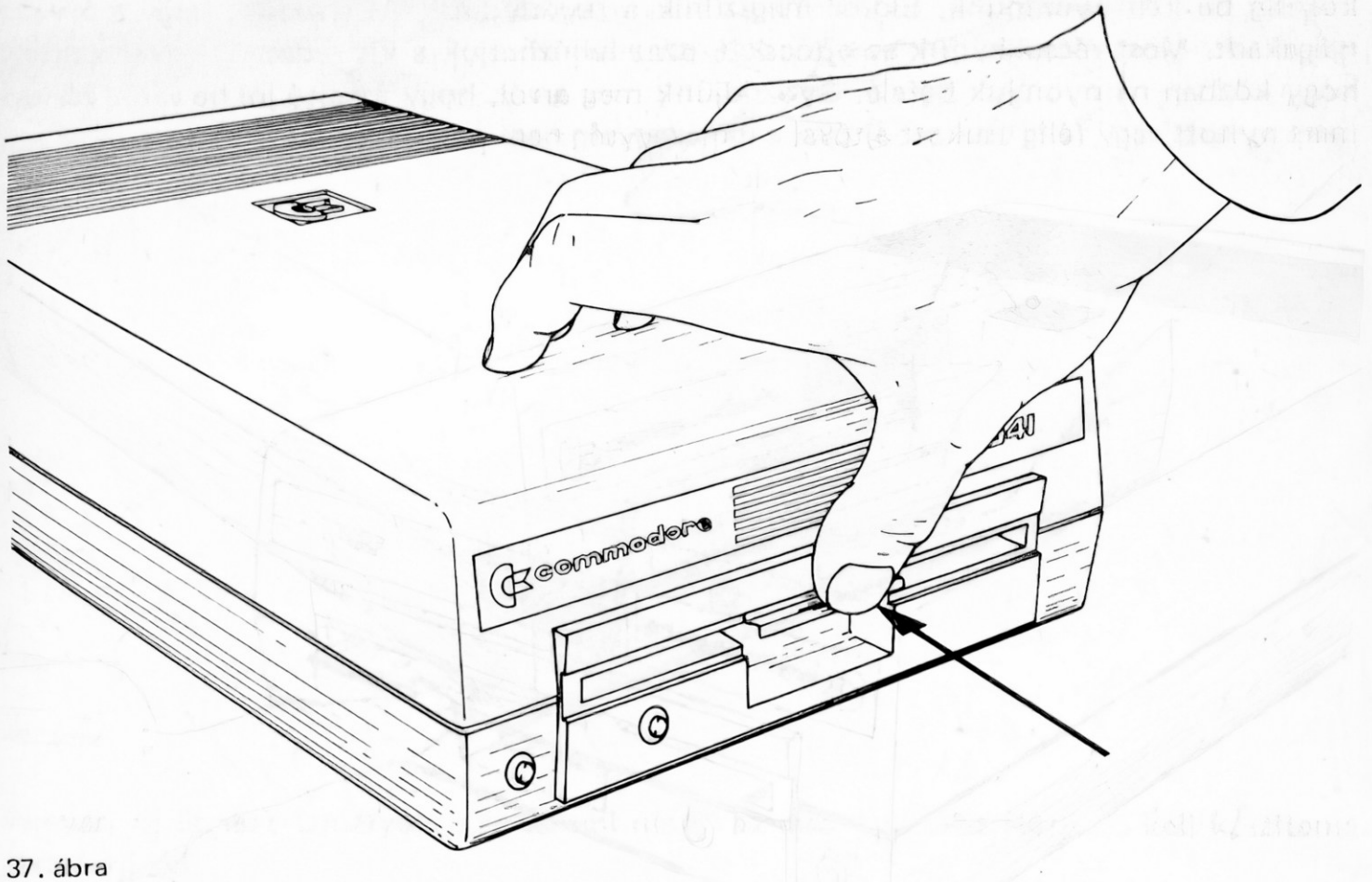
A be- és kikapcsoláskor a lemezegységben ne legyen lemez, mert a be- és kikapcsolási áramlökések hatására információsérülés következhet be.

Az új lemezegységben egy papírlemez van. Ezt érdemes megőriznünk, hogy szállításkor mi is használhassuk: a rázkódásból származó sérülésektől óvja a lemezegységet. Ezt a papírlemezt egyébként akkor is célszerű a lemezegységben tartanunk, amikor az egységet éppen nem használjuk: a porosodás ellen is nyújt némi védelmet.

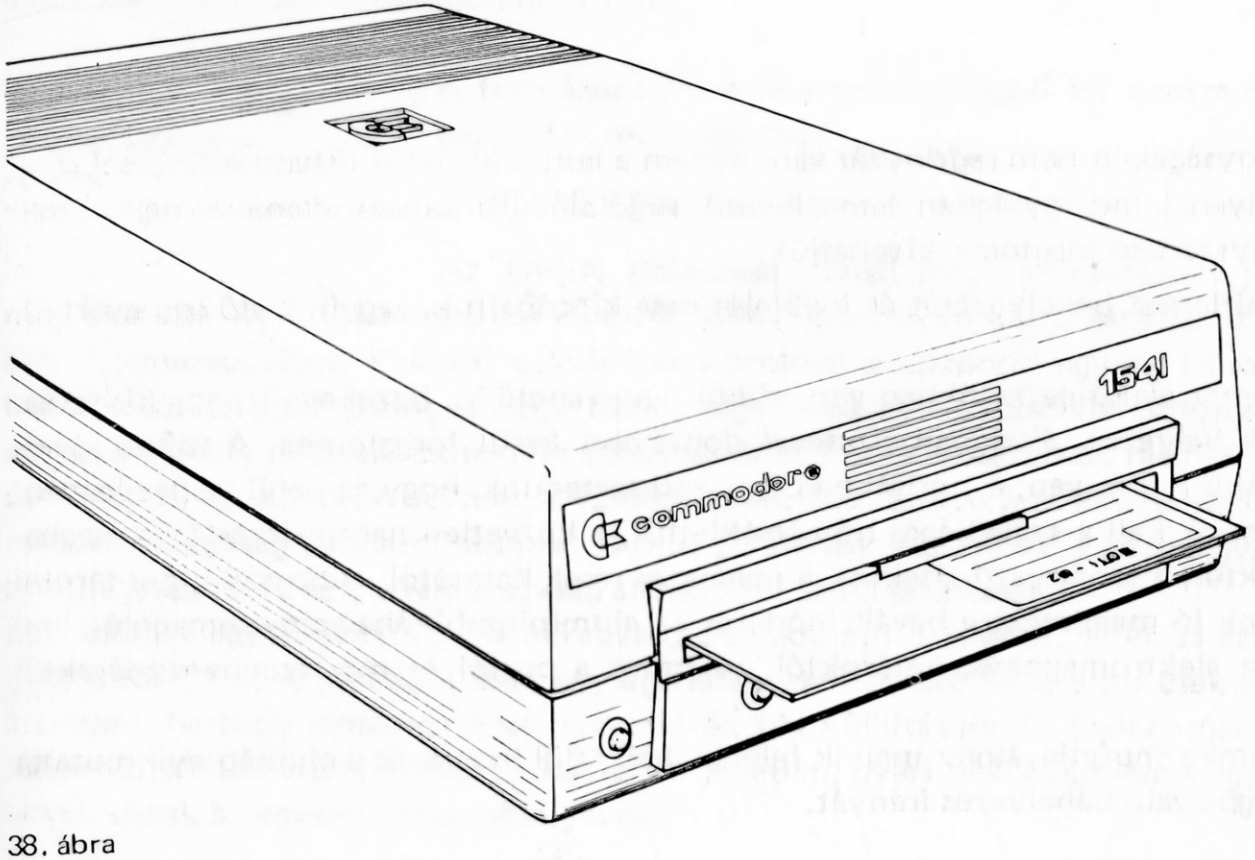
A LEMEZ BEHELYEZÉSE, KIVÉTELE

A lemezek behelyezését és kivételét is gyakorolhatjuk ezzel a papírlemezzel. A lemezegységen redőnyszerű ajtó van. Ha ezt kissé benyomva felfelé húzzuk, a lemezt a lemezegység rugója kiveti. Ekkor a lemezt megfoghatjuk és óvatosan

kihúzhatjuk. Ha akad, kissé oda-vissza megmozgathatjuk. Normális esetben teljesen akadálymentesen kell mozognia.

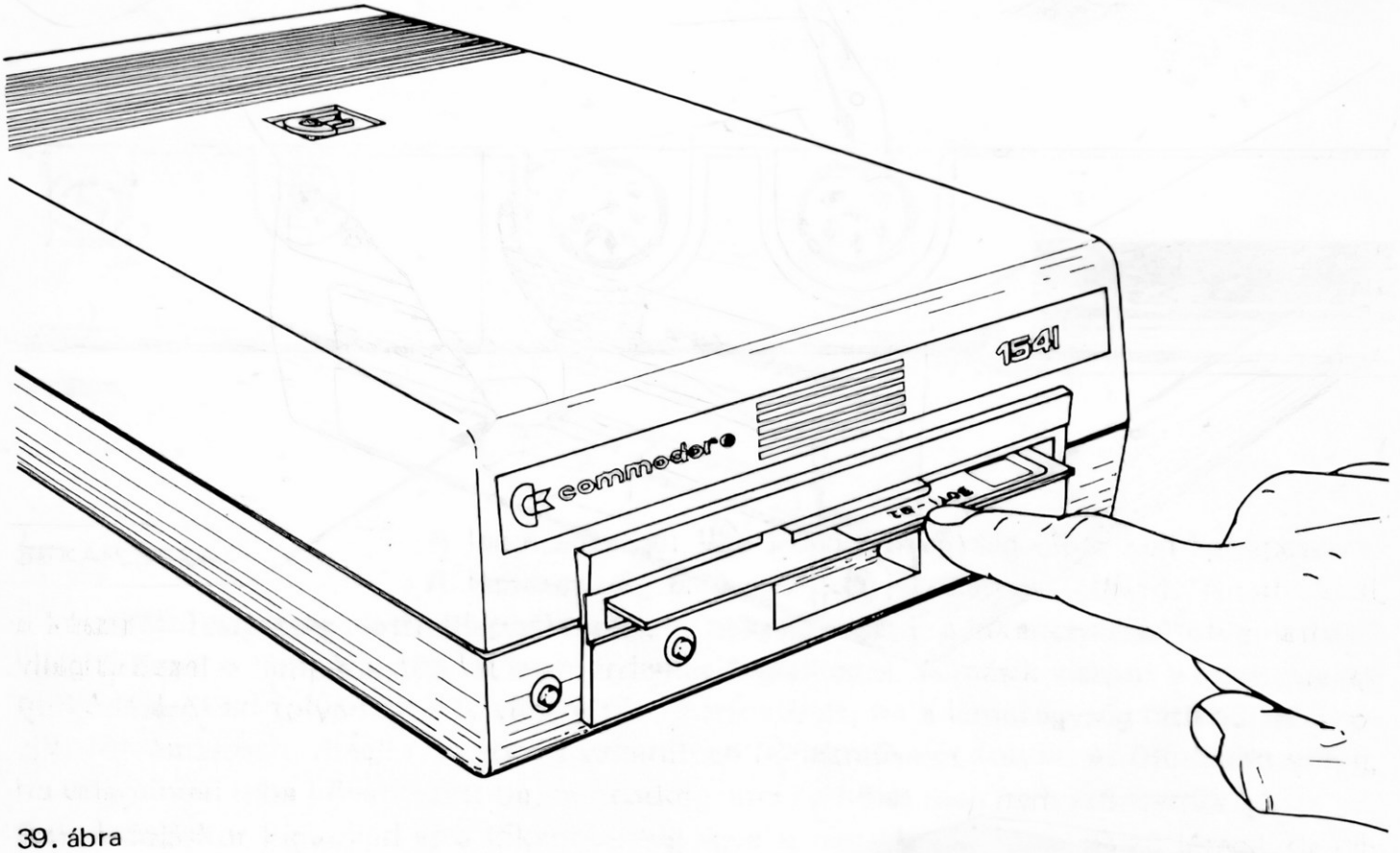


37. ábra



38. ábra

A lemez behelyezése ugyancsak egyszerű. A nyitott ajtócskán a lemezt a címkével megjelölt oldalával felfelé, a címkén látható nyíl irányába mozgatva, finoman betoljuk. Ahogy haladunk befelé, egyre jobban érezzük a lemezegység rugójának a nyomását. A lemezt ütközésig be kell nyomnunk. Ekkor megszűnik a rugónyomás, és érezzük, hogy a lemez megakadt. Most rácsukhatjuk az ajtócskát, azaz lehúzhatjuk a kis redőnyt, ügyelve arra, hogy közben ne nyomjuk befelé. Győződjünk meg arról, hogy az ajtó jól be van-e zárva, mert nyitott vagy félig csukott ajtóval a lemezegység nem működik.



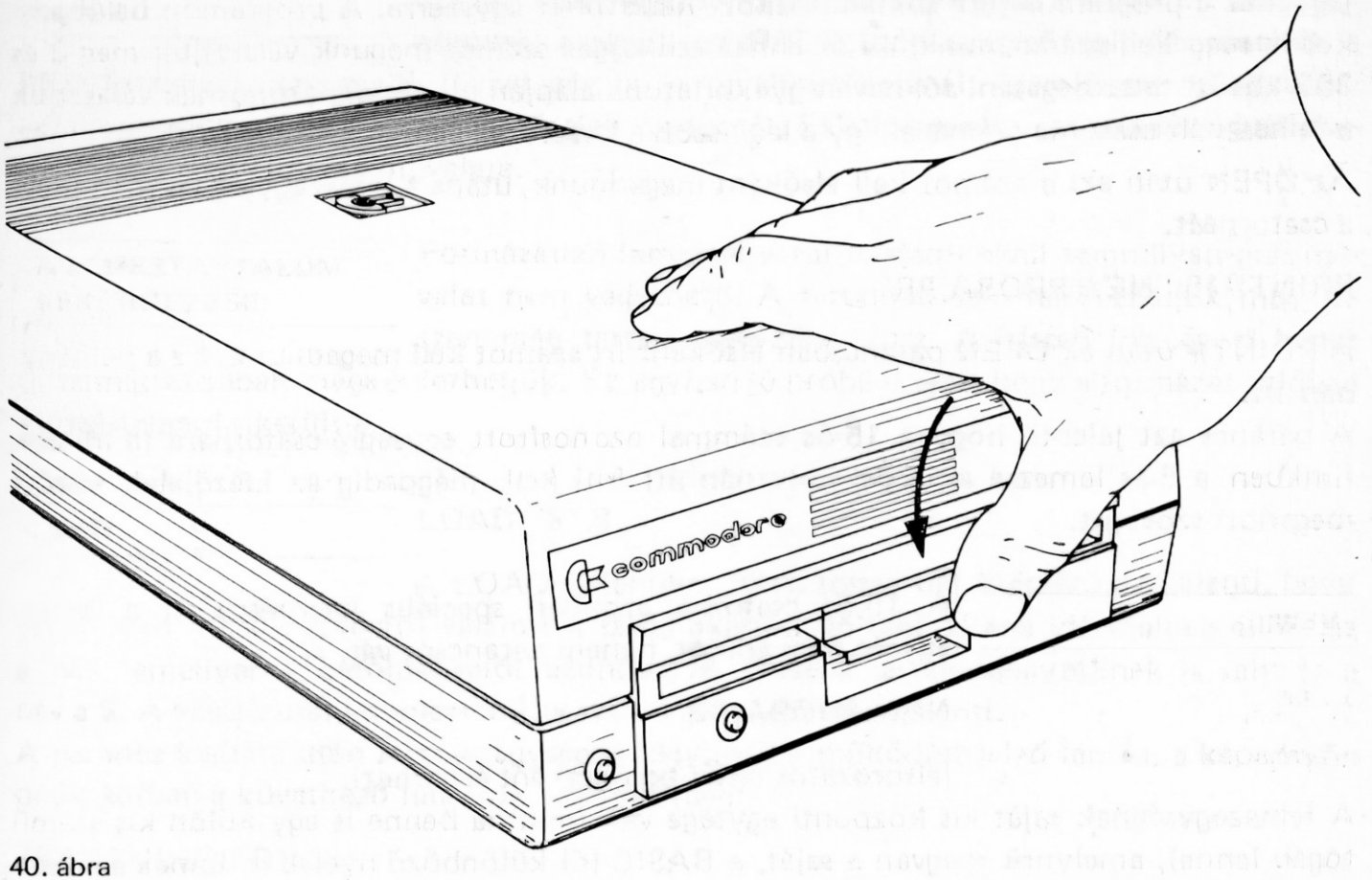
39. ábra

Egyes lemezegységeken nem redőnyzár van, hanem a lemez kis elfordítható kilinccsel rögzíthető. Az ilyen lemezegységben lemezkivető rugó sincs, a lemezt ellenállás nélkül betehetjük és egyszerűen megfogva kivehetjük.

Miután a papírlemez behelyezését és kivételét már kipróbáltuk, vegyünk elő egy gyári új, igazi lemezt.

A lemez négyzet alakú papírtokban van, ebből nem vehető ki. Ezzel együtt egy kis tartóba, borítékba helyezve, általában tízesével dobozolva kerül forgalomba. A tokon lóversenypálya alakú nyílás van; a borítékban úgy kell tartanunk, hogy ez belül, védve legyen. A lemezt óvnunk kell a szélsőséges hőmérséklettől, a közvetlen napsugárzástól, a mechanikus hatásoktól, a szennyeződésektől, a mágneses terek hatásától. A hosszú időre tárolni kívánt lemezek jó megőrzésére bevált módszer az alumíniumfóliába való csomagolás, ami a fény- és az elektromágneses határoktól, valamint a portól és más szennyeződésektől is véd.

A lemezen címke mutatja, hogy melyik felének kell felül lennie, és a címkén nyíl mutatja a lemezegységbe való behelyezés irányát.



40. ábra

A gyári új lemezt (amelyen még semmi nincs) az első használat előtt elő kell készíteni: formázni kell.

Vigyázzunk! Ha a most következő formázási eljárást véletlenül olyan lemezen hajtjuk végre, amelyen már van valami (pl. a géppel együtt vásárolt Commodore-bemutatólemezen), akkor ezzel a lemezről mindent letörlünk.

GYÁRI ÚJ LEMEZ FORMÁZÁSA – OPEN

A formázás három résztevékenységből áll, ezekre most egy-egy parancsot mutatunk be.

OPEN 15,8,15

Az OPEN (jelentése: nyisd meg) kulcsszó azt jelenti, hogy meg akarunk nyitni, használatba akarunk venni valamilyen külső eszközt: a mi esetünkben a lemezegységet. Ezekkel a külső eszközökkel a központi egység 16 csatornán tud kapcsolatot tartani. (Ahogyan egy rádió- vagy tv-készülék többféle csatornát tud fogni, egy adó-vevő is többféle csatornán adhat és vehet.) Ezek a csatornák meg vannak számozva, 0-tól 15-ig. Közülük egynek, a 15-ösnek speciális szerepe van: ezen a központi egységtől a lemezegység felé nem adatok, hanem parancsok küldhetők, a lemezegységtől a központi egység felé pedig a lemezegység állapotáról szóló jelentések.

Egy időben egyébként négy lemezegység kapcsolható a géphez. Ezek is számozva vannak, 8-tól 11-ig. Gyárilag a 8-as egységszámra vannak beállítva, amit csak akkor szokás átállítani, ha több lemezegységet használunk. (Az átállítás módja csak sorozatunk egy későbbi kötetében szerepel majd.) Ha tehát egyetlen, gyári beállítású lemezegységünk van, akkor annak az egységszáma mindig 8-as.

Ugyanaz a program persze sokféle eszközt használhat egyszerre. A programon belül ezeket is meg kell számoznunk, de az ehhez szükséges számot magunk választjuk meg 0 és 255 között tetszőlegesen. Jól bevált gyakorlatunk alapján mi mindig azonosnak választjuk a felhasznált csatorna számával, így a legkisebb a keveredés veszélye.

Az OPEN után ezt a számot kell elsőként megadnunk, utána a lemezegység számát, majd a csatornáét.

PRINT#15,"NEW:PROBA,PR"

A PRINT# után az OPEN parancsban elsőként írt számot kell megadnunk. Ez a példánkban 15.
A parancs azt jelenti, hogy a 15-ös számmal azonosított egységre-csatornára (a mi esetünkben a 8-as lemezre a 15-ös csatornán át) írni kell, mégpedig az idézőjelek között megadott szöveget.

NEW

A 15-ös csatorna azonban speciális jelentésű: itt a lemezegység nem adatot, hanem parancsot vár. A

NEW:PROBA,PR

jelsorozatot tehát parancsként értelmezi.
A lemezegységnek saját kis központi egysége van (mintha benne is egy külön kis számítógép lenne), amelynek megvan a saját, a BASIC-től különböző nyelve is. Ennek a nyelvnek egyik kulcsszava a NEW (jelentése: új), amely a lemez formázására ad parancsot.
A parancs kiadása után képernyőnkön megjelenik a READY, hiszen a központi egység feladata csak annyi, hogy a parancsot a megfelelő csatornán elküldje, és ezt egy pillanat alatt meg is teszi. A lemezegység viszont még kb. 2 percig folyamatosan működik: mikroprocesszora most hajtatja végre lépésről lépésre a teljes formázási folyamatot.

Ennek során a berendezés:
- végigellenőrzi a lemezt, és meggyőződik a hibátlanságáról,
- az egészet végigtörli,
- létrehoz rajta egy tartalomjegyzéket, amely egyelőre üres, de ebben később a lemezen levő programok, adatállományok legfontosabb adatait tárolja,
- létrehoz egy lemeztérképet, amelyen számon tartja, hogy hol vannak a lemezen szabad és hol vannak foglalt területek; egyelőre persze a tartalomjegyzék és a térkép területének kivételével minden terület szabad.

A lemez 256 bájtos egységenként, blokkonként formázható. Mire a formázás befejeződik, 664 ilyen blokkunk lesz; ez megadja a lemez tárolókapacitását is.
A NEW: után megadott nevet (a mi példánkban azt, hogy PROBA) formázáskor a gép felviszi a tartalomjegyzék elejére. Ez lesz a lemez neve. A hossza legfeljebb 16 jel lehetne, de ezen belül bármi, akár 16 szóköz is.
A vessző után megadott nevet (a mi példánkban azt, hogy PR) a gép minden egyes blokkba felviszi. (A 256 bájtba ez nem számít bele.) Ez lesz a lemezazonosító. Pontosan két jelből kell állnia, se több, se kevesebb nem lehet. Ha munka közben lemezt cserélünk, a gép ezt a lemezazonosító megváltozásából veszi észre, a lemez bármelyik részén is tart. Ezért érdemes ügyelnünk arra, hogy ne legyen két egyforma azonosítójú lemezünk.

CLOSE

CLOSE 15 (jelentése: zárj, zárd le) után azt a számot kell megadnunk, amelyik az OPEN után az első volt (és amelyik a PRINT#-ben is szerepelt). Ezzel zárjuk le (gyakran használt szemléletes kifejezéssel: ezzel engedjük el) az OPEN-nel lefoglalt csatornát. Ez a formázás – és minden egyéb csatornahasználat – utolsó művelete.

A LEMEZTARTALOM LEKÉRDEZÉSE

Formázatlan lemezzel a formázáson kívül semmilyen más művelet nem végezhető. A tartalmát sem kérdezhetjük meg, hiszen még tartalomjegyzéke sincs. A frissen formázott lemez tartalmát azonban megkérdezhetjük. Ez egyben jó próba is arra, hogy a formázás valóban kifogástalanul sikerült-e.

LOAD

LOAD "$",8

A LOAD (jelentése: tölts, töltsd be) kulcsszó azt jelenti, hogy valamilyen külső eszközről valamit a tárba akarunk tölteni. Utána idézőjelben állhat az a név, amellyel a betöltenivalót azonosítjuk. Neve a tartalomjegyzéknek is van: ez a név a $. A vessző utáni nyolcas a 8-as számú lemezegységet jelenti.
A parancs kiadása után a lemezegységen kigyullad a működést jelző lámpa, a képernyőn pedig sorban a következő feliratok jelennek meg:

SEARCHING FOR $

(a $ keresése)

LOADING

(betöltés; ez akkor jelenik meg, ha a tartalomjegyzéket a gép megtalálta)

READY

(kész; a tartalomjegyzék betöltése megtörtént).
A korábban már megismert

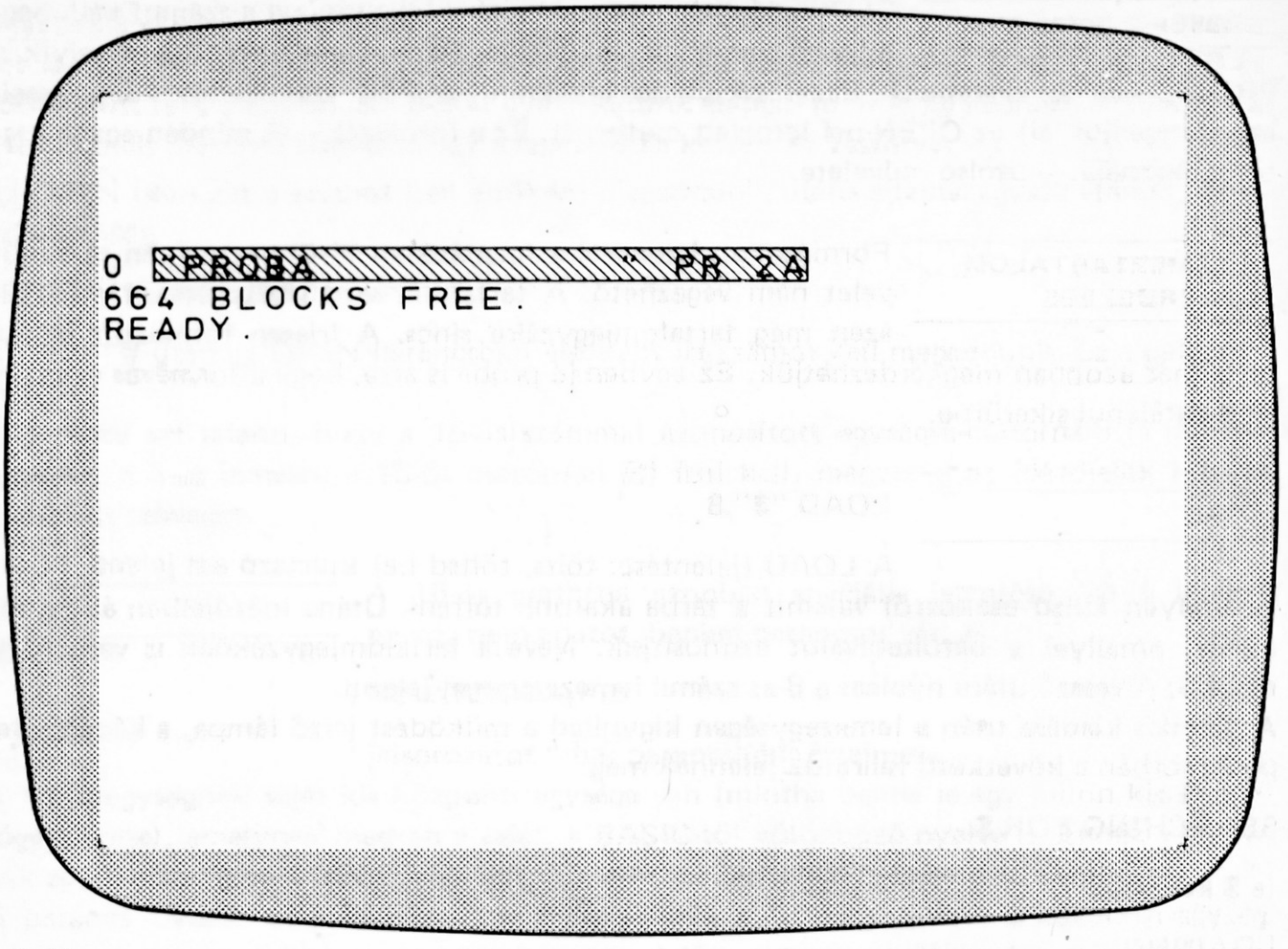
LIST

parancs hatására a képernyőn megjelenik a tartalomjegyzék, amely a mi frissen formázott lemezünk esetében a következőképpen fest:

0 "PROBA " PR 2A
664 BLOCKS FREE

Látható, hogy
– a lemez neve PROBA,
– a lemezazonosító PR,
– a lemez tartalomjegyzéke üres,
– a lemezen 664 szabad blokk van.

Megjegyezzük, hogy a „2A" a lemezkezelő rendszer jele. A képernyőn a lemeznév, az azonosító, és ez a jel inverz írással, azaz a betűk színének megfelelő alapon az alap színének megfelelő betűkkel jelenik meg. ☞ 41. ábra. A következőkben néhány gyakoribb hibalehetőséget sorolunk fel.



41. ábra

HIBALEHETŐSÉGEK

Ha a lemez a helyén van, de az ajtót elfelejtettük lecsukni, vagy ha lecsuktuk az ajtót, de nincs bent lemez, akkor a SEARCHING FOR $ utáni sorban

?FILE NOT FOUND ERROR

(az állomány nem található) hibaüzenet jelenik meg.
Ha a 8-as számot kifelejtjük a parancsból, a gép azt hiszi, hogy kazettás egységről kell olvasnia, és a

PRESS PLAY ON TAPE

üzenetet adja (nyomja le a PLAY gombot a szalagegységen). Ilyenkor meg kell nyomnunk a STOP gombot, amire a gép

?BREAK ERROR

(programmegszakítás) hibaüzenettel reagál.
Ha nem 8-as egységszámot adunk meg, vagy elfelejtettük bekapcsolni a lemezegységet,

?DEVICE NOT PRESENT ERROR

(a lemezegység nem üzemkész) hibaüzenetet kapunk.

Ha bármelyik után helyes tartalomjegyzék-beolvasást hajtunk végre, sikerülni fog. A felsorolt hibák nem rontják el sem a lemezegységet, sem a lemeztartalmat, sem a központi egységet.

A PROGRAM KIMENTÉSE LEMEZRE

Gépeljünk be egy programot. (Ehhez persze, ha a tárban bármi is volt, azt a NEW paranccsal törölni kell. A lemezről beolvasott tartalomjegyzéket is! Új program írásába akkor kezdhetünk, ha a LIST parancs semmit nem listáz ki.)
Legyen a begépelt programunk akár a könyv bármelyik korábbi mintaprogramja, akár bármi más.

SAVE

SAVE *"programnév"*,8

Ebben a parancsban a SAVE (jelentése: mentsd ki) kulcsszó utal arra, hogy egy program kimentését akarjuk végrehajtani.
Idézőjelek között meg kell adnunk a program nevét. Ez tetszőleges, de legfeljebb 16 jel hosszúságú szöveg lehet, mint például MINTA, ELSOPROGRAM, 1.PROGRAM stb. Ez a név kerül be a tartalomjegyzékbe, és ellenőrzéskor, visszaolvasáskor is ezen a néven találhatjuk meg a programunkat. Ugyanazon a lemezen tehát két azonos nevű program nem lehet. Végül – vessző után – meg kell adnunk a lemezegység számát. Esetünkben ez 8.
Figyelem! A programozónak kell gondoskodnia arról, hogy a lemezegységben legyen lemez, és éppen az, amelyikre a programot ki akarjuk menteni!
A parancs kiadása után a gép

SAVING *programnév*

üzenetet ad (jelentése: a programnév nevű program kimentése folyik), majd READY jelzi, ha a kimentés befejeződött.

A KIMENTETT PROGRAM ELLENŐRZÉSE

A Commodore-lemezegység általában megbízhatóan működik. Ha a kimentéskor nem kaptunk hibaüzenetet, megjelent a READY, a lemezegység működését jelző lámpa nem villogott, akkor szinte biztosak lehetünk abban, hogy a kimentés valóban sikerült, a program a lemezen van.
A most bemutatandó parancs a lemezre kiírt program ellenőrzését végzi, anélkül, hogy a program tárban levő példánya ezalatt megváltozna, megsérülne.

VERIFY

VERIFY *"programnév"*,8

A parancs szerkezete azonos a SAVE parancséval. A VERIFY kulcsszó jelentése: ellenőrizd.

A parancs kiadása után a gép

SEARCHING FOR *programnév*

üzenettel jelzi, hogy keresni kezdte a programot,

VERIFYING

(az ellenőrzés folyik) üzenettel jelzi, hogy megtalálta, és megkezdte az összehasonlítását a tárban levő programmal, majd

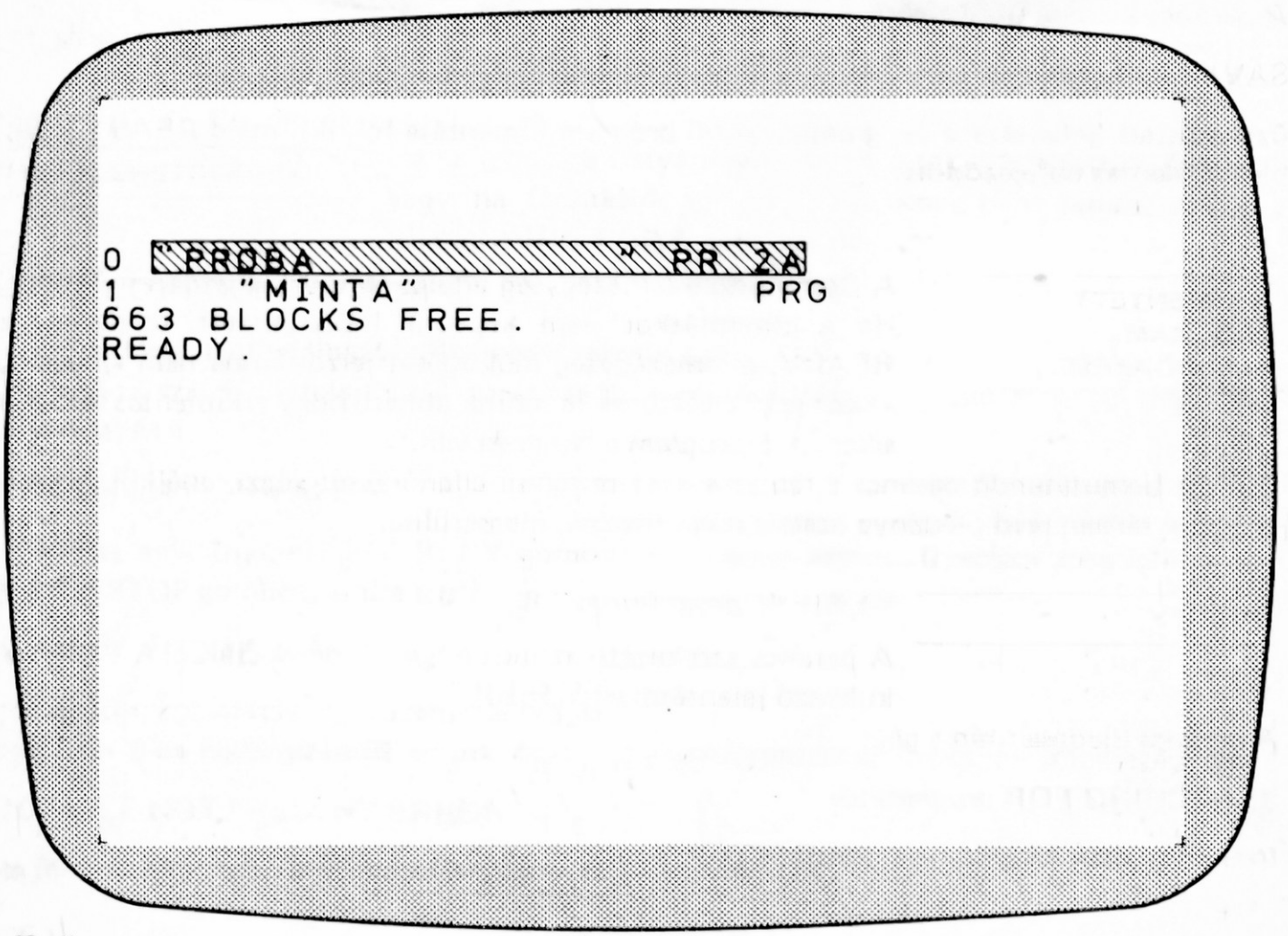
OK

(rendben) üzenettel jelzi, hogy a lemezen ugyanaz van, mint a tárban.
Most listázzuk ki a képernyőre a programot, és módosítsuk, akár csak egyetlen betűben, jelben! Ezután adjuk ki újból a VERIFY parancsot; a VERIFYING üzenet után

?VERIFY ERROR

(sikertelen ellenőrzés) hibaüzenetet kapunk, jelezve, hogy a lemezen nem ugyanaz van, mint a tárban.
A programozónak kell eldöntenie, hogy a hiba miért következett be: mert valóban eltér a két program egymástól, mert a tárban levő programot (a név hibás begépelése miatt) egy másik program lemezre írt példányával hasonlítottuk össze, vagy másért.
Bár a Commodore lemezegysége (ismételjük) eléggé megbízható, értékesebb programjainkat mégis célszerű
— kimentés után, mielőtt bármi mást tennénk, azonnal ellenőrizni,
— legalább két példányban, két különböző lemezre kimenteni.
Ha a program kimentése után nézzük meg a tartalomjegyzéket a korábban megismert

LOAD "$",8
LIST parancsokkal, ezt látjuk:



42. ábra

Látjuk a programunk nevét, a PRG rövidítésből azt, hogy program, a neve előtti számból, hogy hány blokkot foglal el, és a tartalomjegyzék zárósorából látjuk a még szabadon levő blokkok számát is.

A PROGRAM BETÖLTÉSE LEMEZRŐL – LOAD

A programot ugyanúgy tölthetjük be a lemezről a tárba, mint a tartalomjegyzéket:

LOAD *"programnév"*,8

A betöltés folyamata, a betöltés közbeni üzenetek és a lehetséges hibák ugyanazok, mint a tartalomjegyzék betöltése esetében.
Vigyázzunk! Ha a program nevét akár csak egyetlen jellel, egyetlen szóközzel is másképp adjuk meg, mint kimentéskor a SAVE parancsban, akkor már

?FILE NOT FOUND ERROR

(az állomány nem található) hibaüzenetet kapunk, a betöltés nem sikerül.
A sikeresen beolvasott programot ugyanúgy listázhatjuk, futtathatjuk, mintha most gépeltük volna be.
A programbetöltés (és a tartalomjegyzék betöltése is) mindent töröl a tárból, ami korábban bent volt: a programot is, a tartalomjegyzéket is. LOAD előtt tehát a NEW parancs nem hibás, de felesleges.

További lemezkezelési lehetőségek

A PROGRAM FELÜLÍRÁSA – SAVE

Ha egy programot — amelyet lemezre már kimentettünk — a tárban módosítunk, akkor a programunknak végül is két különböző változata létezik. Ha azt szeretnénk, hogy a lemezen is az új változat legyen, akkor ezt is ki kell mentenünk.

Próbáljuk ki! Adjuk ki a

SAVE *"programnév"*,8

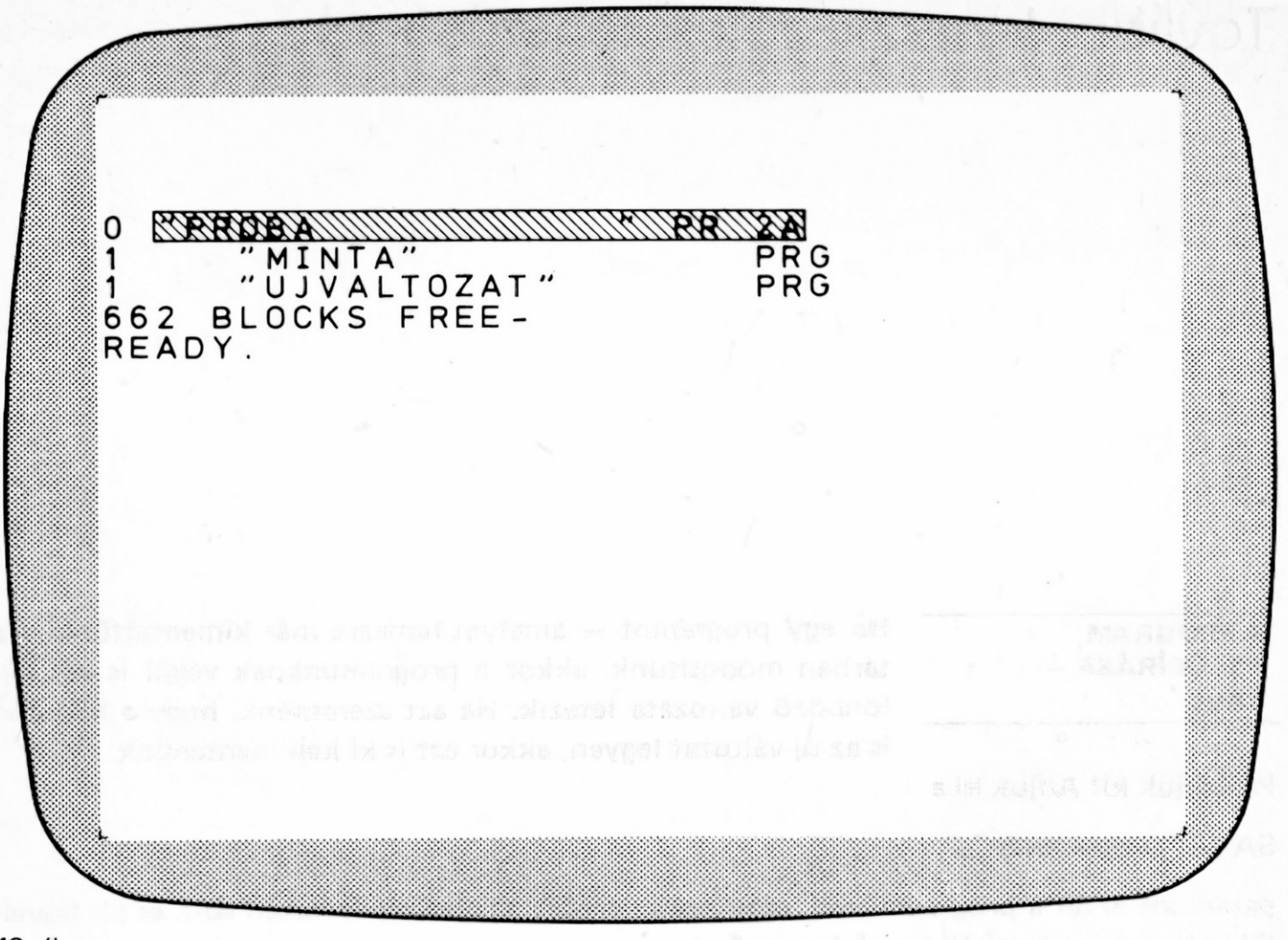
parancsot azzal a programnévvel, amellyel az eredeti program a lemezen volt, és jól figyeljük a lemezegységet! Noha a képernyőn a

SAVING *programnév*
READY.

üzenetek jelennek meg, a lemezegység működését jelző lámpa nem alszik ki, hanem villog. Ez pedig lemezkezelési hibára utal, vagyis a program kimentése nem sikerült.
Ennek az az oka, hogy a lemezkezelő rendszer észrevette, hogy a lemezen már van olyan nevű program, mint amit most ki akartunk írni. Ő nem tudhatja, hogy ez az egyezés szándékos volt-e, mert valóban ugyanannak a programnak az újabb változatáról van szó, vagy pedig véletlen, és most tévedésből megsemmisítenénk egy régebbi programunkat. Mindenesetre nem engedi meg a kimentést.
Nyilvánvaló, hogy ha nincs a lemezen olyan program, mint amilyen nevet a SAVE parancsba írtunk, akkor a kimentés lezajlik. Ezt is kipróbálhatjuk a

SAVE *"új változat neve"*,8

paranccsal, ahol az új változat neve bármi lehet, csak ne legyen azonos valamelyik régivel.
Ha ezután megnézzük a lemez tartalomjegyzékét, azt látjuk, hogy mind a két programváltozat benne van. ☛ 43. ábra.
Most tehát a lemezen két változatban is megvan a programunk. Adott esetben nagyon hasznos lehet egy-egy program több változatát is megtartani, most azonban nem ezt szerettük volna, hanem azt, hogy az új program a régi helyett szerepeljen, mivel a régire már nincs szükségünk.



43. ábra

Ez is megvalósítható, de előbb töltsük vissza a lemezről az új programot, mert a tartalomjegyzék betöltésével a programunk a tárban megsemmisült. (Ha meg akarunk győződni arról, hogy valóban az a program került-e vissza, amelyet kimentettünk, a LOAD után adjunk LIST parancsot is.)
Most tehát kimenthetjük a programot úgy, hogy írja felül, semmisítse meg a régit. Erre a SAVE egy újabb változatát használjuk.

SAVE "@:*programnév*",8

Ez csak annyiban tér el a megszokott SAVE parancstól, hogy a programnév előtt az @: jelcsoport szerepel benne.
Ezzel informáljuk a lemezkezelőt: ezúttal nem kell törődnie azzal, hogy a kettőspont után megadott programnév szerepel-e már a lemezen.
A fenti parancs hatására a tartalomjegyzékben semmi sem változik, a programnév benne marad, de — amint erről LOAD paranccsal meggyőződhetünk — ezen a néven már a program új változata lesz a lemezen.

A PROGRAM TÖRLÉSE

Most tehát két példányban van a lemezen ugyanaz a program: a programnév és az új változat neve különböző ugyan, de a két program azonos. (Ha nem hisszük, betölthetjük és kilistázhatjuk előbb az egyiket, azután a másikat.)
Ha úgy döntünk, hogy az egyik felesleges, mondjuk az egyébként is szándékunk ellenére felvitt új változat, akkor törölhetjük a lemezről. Ehhez azonban már használnunk kell a formázáskor megismert parancssorozatot és parancscsatornát.

SCRATCH

```
OPEN 15,8,15
PRINT#15,"SCRATCH:új változat neve"
CLOSE 15
```

A SCRATCH szó jelentése: levakarni, vagyis letörölni valamit valahonnan. A parancssorozat hatására a megadott nevű program eltűnik a lemezről, és a lemez még szabad blokkjainak számából is látjuk, hogy a helye felszabadult.
Megjegyezzük, hogy egy paranccsal több program is törölhető. Például a
PRINT#15,"SCRATCH:ALFA,BETA,GAMMA"
hatására az ALFA, a BETA és a GAMMA nevű program is törlődik.

A PROGRAM NÉVVÁLTOZTATÁSA – RENAME

Ha megbántuk, hogy az új változatot töröltük le és nem az eredetit, ezen is segíthetünk: a programok tartalomjegyzékbeli neve is megváltoztatható az előzőhöz hasonló parancsokkal:

```
OPEN 15,8,15
PRINT#15,"RENAME:új név=programnév"
CLOSE 15
```

A program ennek hatására nem változik, csak a tartalomjegyzékbeli neve, amely – ha eddig *programnév* volt – ezután *új név* lesz. Ha be akarjuk tölteni, ezentúl már így kell megneveznünk.

A PROGRAM LEMÁSOLÁSA – COPY

Ha mégis úgy döntünk, hogy a programunk két példányban legyen a lemezen, akkor másolatot készíthetünk róla.

```
OPEN 15,8,15
PRINT#15,"COPY:másolat=új név"
CLOSE 15
```

Az előbb az új néven egyetlen példányban létező program ezután másolat néven is megvan. A két program betűről betűre való megegyezéséről könnyen meggyőződhetünk, ha a

LOAD *"új név"*,8

végrehajtása után

VERIFY *"másolat"*,8

parancsot adunk.
Egyébként programot másolni úgy is lehet, hogy betöltjük a programot egy LOAD paranccsal, majd (más néven) kimentjük egy SAVE paranccsal. Minthogy ilyenkor a LOAD

és a SAVE között van időnk lemezt is cserélni, ezzel a módszerrel a program másik lemezre is átmásolható; a COPY segítségével nem. A COPY előnye viszont az, hogy gyorsabb, és közben nem vész el a tár tartalma sem.

A LEMEZ TÖMÖRÍTÉSE

Törlések, felülírások során a lemezen kihasználatlan területek keletkezhetnek, noha a lemezkezelő törekszik ezek kihasználására. Ha valahol két program között például csak egy-két blokknyi üres terület van, ez feltöltetlen marad mindaddig, amíg ilyen kis programot nem akarunk kimenteni.
Ezért célszerű a lemezt időnként tömöríteni, ami azt jelenti, hogy a lemezen levő programokat, állományokat összébb húzzuk, a kimaradt területeket feltöltve.

VALIDATE

```
OPEN 15,8,15
PRINT#15,"VALIDATE"
CLOSE 15
```

A lemez újraszervezése viszonylag időigényes, bár nem tart annyi ideig, mint a teljes formázás.
Figyelem! Tilos a lemezt tömöríteni, ha úgynevezett random állományokat használunk. (Ezekkel majd később ismerkedünk meg.) Ezek az állományok ugyanis pontosan a „hulladék" területeket használják fel adattárolásra, ezért a tömörítés megsemmisítené őket.

A LEMEZ TÖRLÉSE – NEW

Ha a lemezen levő összes programot törölni akarjuk, ennek egyik módja, hogy mindegyikre SCRATCH parancsot adunk ki. Van azonban egyszerűbb lehetőség is.

```
OPEN 15,8,15
PRINT#15,"NEW:a lemez új neve"
CLOSE 15
```

A parancs majdnem olyan, mint formázáskor, itt azonban nem adunk meg lemezazonosítót. Hatására a lemezen levő minden program, állomány törlődik, a lemez neve is kicserélődik, de a lemezazonosító megmarad.
Megjegyezzük, hogy a lemez újraformázással is törölhető, s ekkor a lemezazonosítót is megváltoztathatjuk. Az újraformázás azonban lényegesen lassúbb, mint az imént bemutatott törlés.

A LEMEZEGYSÉG HELYREÁLLÍTÁSA

A lemezegység a lemez használata közben a lemeztérképet saját processzora tárjában tartja (tehát nem a Commodore központi egységében). Lemezhasználat közben eszerint tájékozódik. Csak akkor frissíti fel, ha új lemezazonosítót talál: ebből veszi észre, hogy a legutóbbi lemezművelet óta lemezt cseréltünk.
Ettől az esettől eltekintve a lemeztérképet nem nézi meg: egyszer beolvasta, s attól kezdve karbantartja, hogy új olvasás nélkül is tudja, mi hol van a lemezen.
Előfordulhat azonban olyan hiba, amikor a lemezegység „eltéved" karbantartási hiba miatt, vagy például azért, mert van két egyező azonosítójú lemezünk: az azonosító nem

változik, mégsem ott vannak az adatok, ahol a lemezegység keresi. Ilyenkor például a lemezegység lámpája villog, és ez az állapot nem szüntethető meg.
Ezen esetenként még segíthet az újrainicializálás: a lemeztérkép újbóli beolvastatása.

INITIALIZE

```
OPEN 15,8,15
PRINT#15,"INITIALIZE"
CLOSE 15
```

Hatására a lemezegység olyan állapotba kerül, mintha most kapcsoltuk volna be.
Végső esetben a lemezegységet ténylegesen kapcsoljuk ki, majd kisvártatva újból be kell kapcsolnunk.
Figyelem! Ha erre a drasztikus megoldásra kényszerülünk, előtte ne felejtsük el kivenni a lemezt a lemezegységből, és kapcsoljuk ki a központi egységet.

A PARANCSCSATORNA LEKÉRDEZÉSE

Vannak lemezkezelési hibák, amelyekről a központi egység nem kap információt. Ilyenkor hibaüzenet nélkül megjelenik a READY, de a lemezegység működését jelző lámpa villog. Erre láttunk példát, amikor a SAVE parancsban már létező program nevét adtuk meg. Ilyenkor is lekérdezhető a lemezegység állapota, mégpedig a parancscsatornán keresztül. Így hozzáférhetünk a lemezkezelő által beállított hibakódhoz és a hibaüzenet szövegéhez is.

```
60000 REM: HIBALEKERDEZES
60001 REM: ----------------
60002 REM:
60010 OPEN 15,8,15
60020 INPUT#15,HKOD,HUZENET$
60030 PRINT
60040 PRINT HKOD;HUZENET$
60050 PRINT
60060 CLOSE15
60099 END
```

44. ábra

Gépeljük be ezt a programot, és mentsük ki a lemezre, mondjuk HIBALEKERDEZES néven!
A programot dokumentációs okokból fejléccel kezdjük (60000–60002. sorok), majd megnyitjuk a parancscsatornát (60010. sor), ezután leolvashatjuk róla a hibakódot és a hibaüzenetet (60020. sor). Ha látni is akarjuk, ki kell írnunk őket a képernyőre (60030–60050. sorok). Végül lezárjuk a parancscsatornát (60060. sor), és ezzel vége is a programnak (60099. sor).
Most adjuk ki a RUN parancsot! Ha a legutóbbi lemezművelet (pl. a program kimentése) helyesen lefutott, a programunk

```
0 OK
```

üzenetet ír ki a képernyőre. Ha tehát minden rendben van, akkor a hibakód 0, az üzenet pedig OK.

Próbáljuk meg még egyszer kimenteni a programot ugyanazon a néven! A képernyőn a helyes kimentésre jellemző szöveg jelenik meg, de a lemezegység lámpája villog. Valami tehát nincs rendben, de hibaüzenetet nem kaptunk.
Futtassuk le ismét a hibalekérdező programot! Két jelenséget látunk. Először is a képernyőn megjelenik a

63 FILE EXISTS

üzenet, amelyet nem a Commodore-rendszer, hanem a programunk írt ki (jelentése: az állomány létezik). Ebből már tudjuk, hogy a kimentés azért nem sikerült, mert a program ezen a néven már a lemezen van. (Ennek a hibának a hibakódja: 63). Másodszor pedig: megszűnik a villogás. Ez összhangban van a korábbi tapasztalatainkkal: a hiba után sikeresen végrehajtott új lemezművelet „helyre teszi", kihozza a hibaállapotból a lemezegységet.
A hibalekérdező program további előnyét is tapasztalhatjuk a kétes esetekben. Hagyjuk például nyitva a lemezegység ajtaját, miközben a lemez bent van, és próbáljuk meg betölteni a tartalomjegyzéket vagy bármelyik programot. Ez természetesen nem sikerül. A képernyőn megjelenő hibaüzenet azonban ez lesz:

?FILE NOT FOUND

azaz nem találja az állományt. Miért viselkedik úgy a gép, mintha nem találta volna meg a keresett programot vagy éppen a tartalomjegyzéket, noha csak az ajtó nem volt rendesen becsukva?
Pontosabb információhoz jutunk, ha nem hagyatkozunk a központi egység hibaüzenetére, hanem megnézzük a lemezkezelő hibaüzenetét is. Minthogy a betöltés nem sikerült, hibalekérdező programunk a tárban maradt; futtassuk le. Ezt írja ki:

74 DRIVE NOT READY

vagyis 74-es hiba: a meghajtó (ti. a lemezegység) nem üzemkész. Így már egyszerűbb a hiba megtalálása.
Vajon eredetileg miért nem ez jelent meg a képernyőn? Mert a központi egység a hibára csak annak hatásából következtet. Kért egy programot, nem kapta meg, nyilván ilyen program nincs. Hogy pontosan mi történt, ezt csak a lemezkezelő érzékeli. Persze a felhasználó szempontjából az lenne kényelmes, ha ezt a lemezegység automatikusan közölné a központi egységgel. Ezt a Commodore-nál sajnos nem valósították meg.
De térjünk vissza a hibalekérdezéshez. Esetünkben ez könnyen ment, hiszen a program bent volt a tárban. Mit tegyünk akkor, ha hiba állt elő, a hibalekérdező program pedig nincs bent? Be nem olvashatjuk, mert ezzel új lemezműveletet végzünk, vagyis megszüntetjük azt a lemezegység-állapotot, amelyet pedig éppen le akarunk kérdezni!
Nem tehetünk mást, mint hogy gyorsan begépeljük a hibalekérdezés leglényegesebb utasításait. Ehhez adjuk ki a NEW parancsot, majd gépeljük be a 44. ábra programját.
Ezután adjunk RUN parancsot, és megjelenik, amire kíváncsiak voltunk.
Ha a tárban levő programot nem akarjuk megsemmisíteni, begépelhetjük NEW nélkül is a 45. ábrán látható programrészletet, feltéve, hogy a tárban levő programban nincsenek ilyen sorszámú sorok. Ekkor RUN 60010 paranccsal elindíthatjuk a programot a 60010-es sortól, a mi kezdősorunktól. Ha az eredeti programot akarjuk ismét futtatni, ne feledjük törölni az újonnan begépelt programrészt!

```
60010 OPEN 15,8,15
60020 INPUT#15,H,H$
60040 PRINT H;H$
60060 CLOSE 15
```

45. ábra

Sajnos a parancscsatornát csak programból kérdezhetjük le. Ha a fenti utasításokat parancsként próbáljuk kiadni, az INPUT-ra

?ILLEGAL DIRECT ERROR

(jelentése: szabálytalan közvetlen parancs) hibaüzenet jelenik meg. INPUT kulcsszó parancsban nem szerepelhet.

A PARANCSOK RÖVIDÍTÉSEI

A lemezkezelő funkciók kulcsszavai rövidíthetők:

NEW	N
SCRATCH	S
RENAME	R
COPY	C
VALIDATE	V
INITIALIZE	I

Így például a

PRINT#15,"RENAME:UJ=REGI"

helyett írhatjuk azt is, hogy

PRINT#15,"R:UJ=REGI"

Ez kényelmes lehetőség, de megvannak a veszélyei. Ha például I helyett véletlenül V betűt gépelünk be, komoly kárt okozhatunk: megsemmisülhetnek a lemezen levő random adatállományaink. Ha viszont egy hosszabb parancsban írunk el egy betűt, legfeljebb hibaüzenetet kapunk, és meg kell ismételnünk a parancs begépelését.
Másik rövidítési lehetőség, hogy a lemez kezelésére vonatkozó információk nemcsak a PRINT, hanem az OPEN parancsban is megadhatók. Ilyenkor a PRINT-re nincs is szükség. Vagyis például

```
OPEN 15,8,15
PRINT#15,"INITIALIZE"
CLOSE 15
```

helyett ezt is írhatjuk:

```
OPEN 15,8,15,"I"
CLOSE 15
```

A PROGRAMNEVEK RÖVIDÍTÉSEI

A programok, állományok neveinek rövidítésére két lehetőségünk van.
Csillaggal helyettesíthetők a név végén levő jelek. Például az AL* névaz összes AL-lal kezdődő, akármilyen hosszú nevet jelenti. Ha a lemezen ALFONZ,

AMBRUS, ARISZTID, ARNOLD és ALBIN nevű programjaink vannak, akkor például a

PRINT#15,"SCRATCH:AL*"

parancs az ALFONZ és az ALBIN nevű programokat törli. Ha pedig csak a csillagot adjuk meg, azaz ha

PRINT#15,"SCRATCH:*"

parancsot adunk, az összes program (és állomány) törlődik.

Kérdőjellel helyettesíthetők egyes jelek, de nemcsak a név végén. Ezzel a név nem rövidül, viszont a mi munkánk egyszerűsödik. Például ha a lemezen ALFA1, ALFA2, BETA1, BETA2, GAMMA1, GAMMA2 nevű programok vannak, akkor a

PRINT#15,"SCRATCH:????1"

parancs az ALFA1 és BETA1 nevűeket törli, mert ezeknek a nevében áll négy akármilyen jel után 1-es.

Törölni lehet több programot is, de például a tárba betölteni egyszerre csak egyet. Ha ilyen funkcióra adunk parancsot, és abban a név nem egyértelmű, akkor a leírásunknak megfelelő nevek közül az elsőre következik be a művelet végrehajtása. Például a

LOAD "ALF*",8

parancs a lemezen levő ALFONZ, ALFRED és ALFA programok közül azt fogja betölteni, amelyik a tartalomjegyzékben a másik kettőt megelőzi.

RENAME, SAVE és COPY funkciókat használó parancsokban sem a csillag, sem a kérdőjel nem használható; ezekben egyértelműen kell a neveket megadnunk.

A lemezparancsokra vonatkozó minden rövidítési lehetőséggel kapcsolatban az a megalapozott álláspontunk, hogy csak akkor használjuk őket, ha biztonságos kezelésükhöz már megfelelő gyakorlatot szereztünk.

Nyomtató

Üzemszerű felhasználás aligha képzelhető el nyomtató nélkül. Az adatok és a programok tárolásának, megőrzésének megbízható, jól hozzáférhető, olcsó eszköze a papír. A képernyőnél lassabban jelenik meg rajta az információ, de nem vész el, hordozható, egyszerűen sokszorosítható. A lemeznél terjedelmesebb, de kevésbé sérülékeny, és külön berendezés nélkül is olvasható. Hátránya, hogy a Commodore számára elérhetetlenek a rajta rögzített információk: ha a géppel újból fel akarjuk dolgozni, ismét be kell gépelnünk őket, hacsak nincsenek meg más adathordozón (pl. a lemezen) is. A programok javítása, módosítása viszont igen nehézkes anélkül az áttekinthetőség nélkül, amit a kinyomtatott lista nyújt.

A továbbiakban a Commodore MPS—801-es típusú nyomtatóval foglalkozunk. (A géphez másfajta nyomtatók is csatolhatók, de többségükre az itt leírtak ugyanúgy érvényesek.)

ÖSSZEÁLLÍTÁS

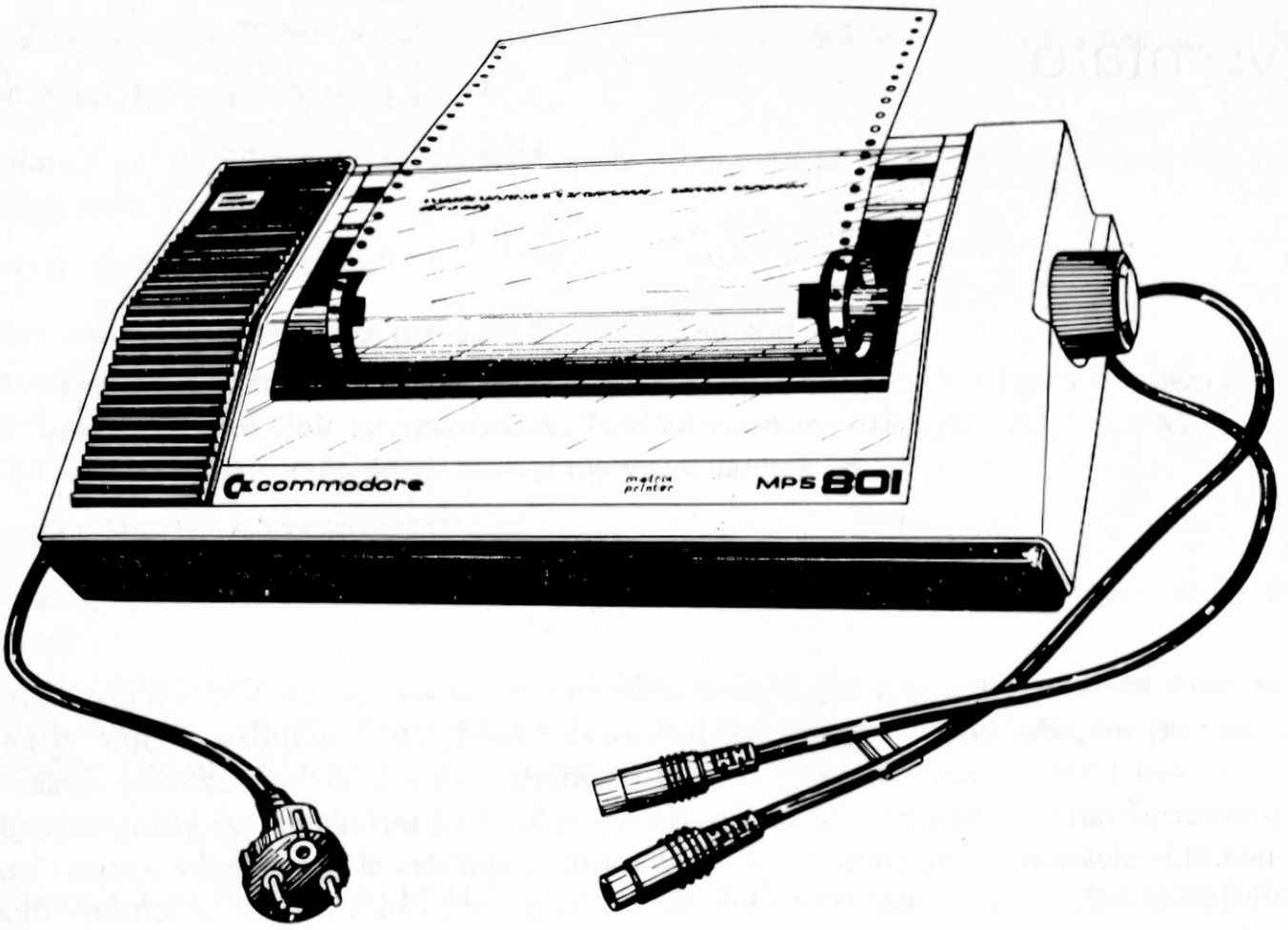
A nyomtatóhoz külön csatlakozókábel tartozik, két tuchellel a végén. ☞ 46. ábra.

A nyomtató csatlakozóaljzatai a készülék hátoldalán találhatók. ☞ 47. ábra.

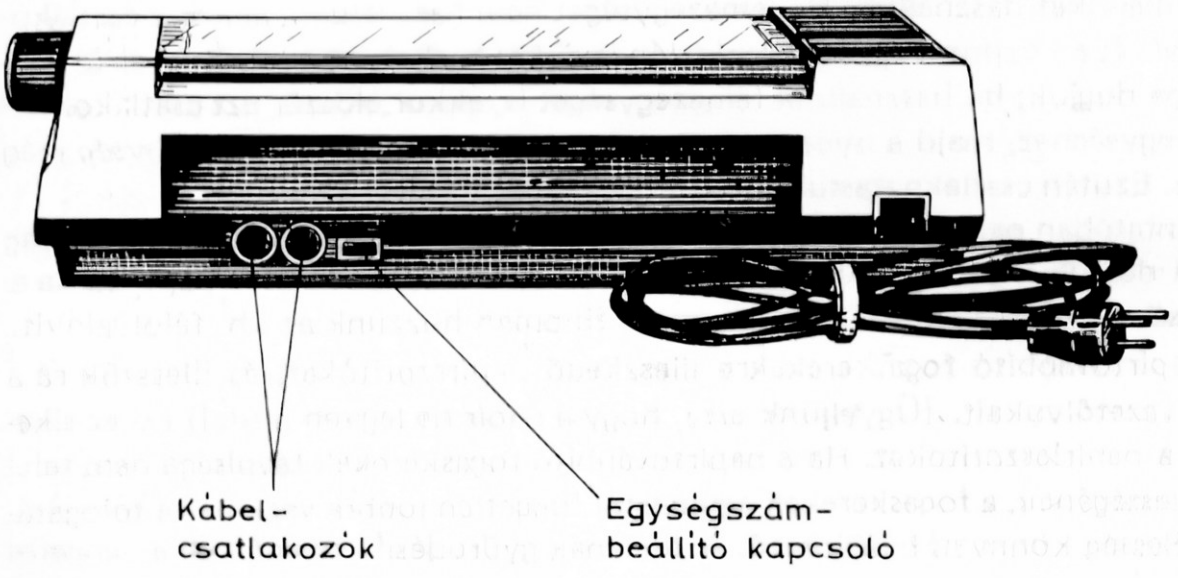
Mindegy, hogy melyiket használjuk. Ha lemezegységet nem használunk, akkor a csatlakozókábel másik végét a központi egység hátoldalán levő két tuchelaljzat közül (elölről nézve) a bal oldaliba dugjuk; ha használunk lemezegységet is, akkor először ezt csatlakoztassuk a központi egységhez, majd a nyomtató csatlakozókábelét dugjuk a *lemezegység* még szabad aljzatába. Ezután csatlakoztassuk a nyomtatót a hálózathoz is.

Ha nincs a nyomtatóban papír, azt is be kell fűznünk. Ehhez először vegyük le a műanyag fedelet. Hátulról dugjuk a papírt a henger alá, úgy, mint az írógépbe. Ha a papír sarka a henger felénk eső oldalán előbújt, fogjuk meg, és finoman húzzunk át kb. féloldalnyit. Nyissuk ki a papírtovábbító fogaskerekekre illeszkedő papírszorítókat, és illesszük rá a fogakra a papír vezetőlyukait. (Ügyeljünk arra, hogy a papír ne legyen ferde!) Ha ez sikerült, csukjuk le a papírleszorítókat. Ha a papírtovábbító fogaskerekek távolsága nem felel meg a papír szélességének, a fogaskerekek egymástól független jobbra vagy balra tologatásával a kellő szélesség könnyen beállítható. A papírnak gyűrődésmentesen kell a hengerre rásimulnia.

A papírt az írófejhez képest is be kell állítanunk. Meg kell akadályoznunk ugyanis, hogy

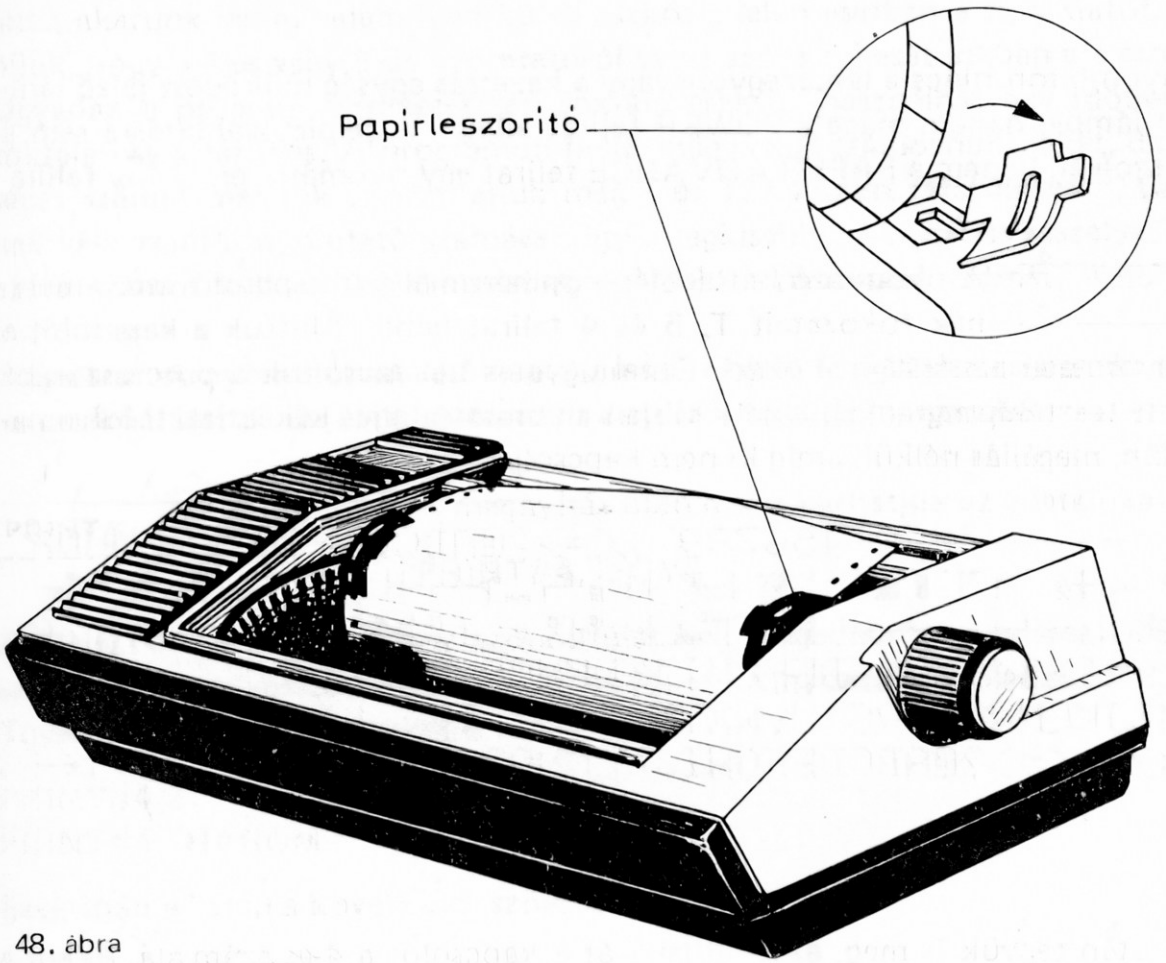


46. ábra



47. ábra

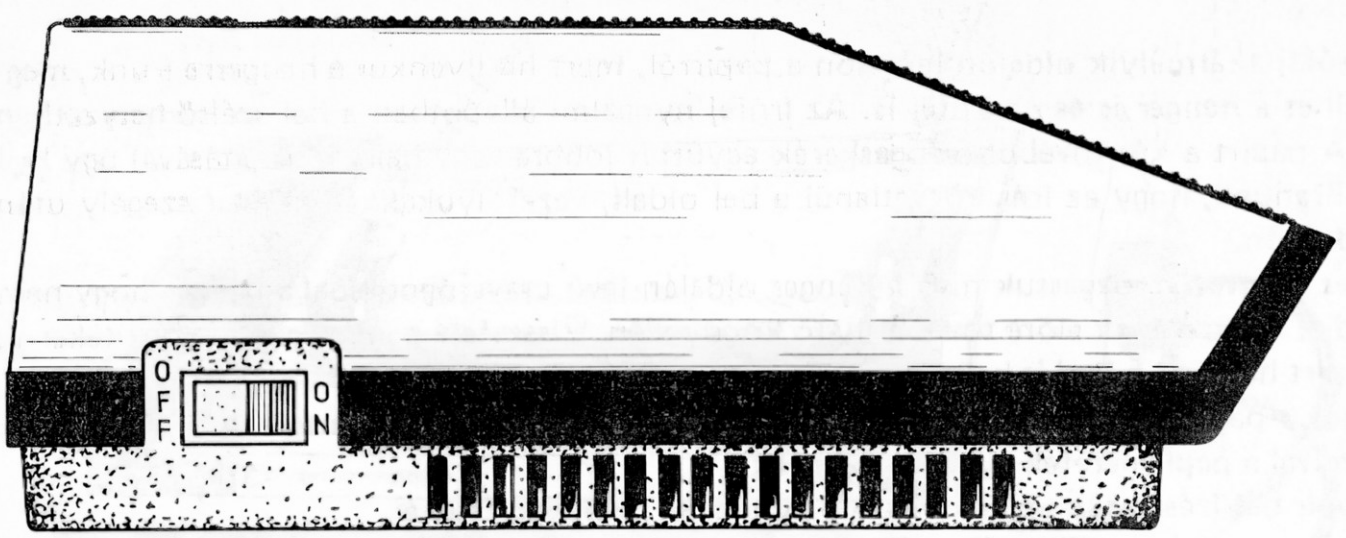
az írófej akármelyik oldalon lefusson a papírról, mert ha ilyenkor a hengerre írunk, megsérülhet a henger is és az írófej is. Az írófej nyugalmi állapotban a bal szélső helyzetben áll. A papírt a két továbbító fogaskerék együttes jobbra vagy balra mozgatásával úgy kell beállítanunk, hogy az írás közvetlenül a bal oldali, vezetőlyukakkal ellátott szegély után kezdődjön.
Ha ez megvan, mozgassuk meg a henger oldalán levő csavarógombbal a papírt, hogy nem akad-e. A papír csak előre továbbítható könnyedén. Visszafelé nem elég a gombot tekerni, a papírt hátulról húzni is kell. Ez azonban a készüléknek nem tesz jót.
Sajnos a papírbeállítást szinte mindig újra el kell végeznünk, ha új csomag papírt fűzünk be, mivel a papírméretek gyakran eltérőek.
A papír befűzése után tegyük vissza a nyomtató műanyag fedelét.



48. ábra

BEKAPCSOLÁS

A nyomtató kapcsolója (elölről nézve) a készülék bal oldalán található. Bekapcsoláskor kigyullad a POWER feliratú jelzőlámpa, majd az írófej hangos berregéssel elmegy a papír közepéig, ezután visszaáll a sor elejére. (A nyomtató saját mikroprocesszora ilyenkor ellenőrzi a készülék működőképességét.) A POWER lámpa folyamatosan világít, amíg a nyomtatót ki nem kapcsoljuk.
A nyomtatón található egy PAPER ADVANCE (papírtovábbítás) felirat is, amelynek megnyomásával a papír továbbítható. A papír minden nyomásra egy-egy sort lép előre. Ha folyamatosan nyomjuk, akkor addig lépked egy-egy soros lépésekkel, amíg el nem engedjük.



49. ábra

Az MPS–801 nyomtatón nincs a lemezegység vagy a kazettás egység működést jelző lámpájához hasonló lámpa, hanem maga a POWER felirat világít. A papírtovábbításra sem a szokásos gomb szolgál, hanem a PAPER ADVANCE felirat egy nyomásra érzékeny felületet jelöl.

TESZTELÉS

A nyomtató hátoldalán egy háromállású kapcsoló van, amelynek fokozatait T, 5 és 4 felirat jelöli. Állítsuk a kapcsolót a T betű alá: a nyomtató azonnal írni kezd. Ezzel ugyanis bekapcsoltuk a processzorába gyárilag beégetett tesztelőprogramot, amely kiírja a nyomtató teljes jelkészletét, folyamatosan egymás után, megállás nélkül, amíg ki nem kapcsoljuk.

50. ábra

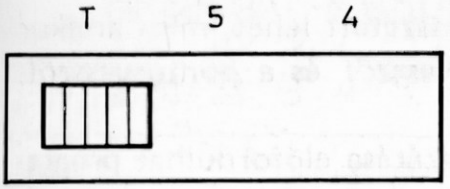
Ezt néhány sor után tegyük is meg, azaz állítsuk át a kapcsolót a 4-es szám alá. Ekkor a tesztelőprogram leáll. Ha szükséges, a tesztelés eredménye alapján most még állíthatunk a papír helyzetén. ☞ 51. ábra.

Azzal, hogy a kapcsolót a 4-es számhoz toltuk, egyben be is állítottuk a nyomtató egységszámát. Ezentúl az utasításainkban és parancsainkban mindig ezzel a számmal kell hivatkoznunk rá. (Ebből látszik, hogy egyszerre két nyomtatót használhatnánk, 4-es, illetve 5-ös számmal.)

Mindezek után bekapcsolhatjuk a rendszer többi egységét is; utolsónak a központi egységet. (A teszteléshez a nyomtatónak nincs szüksége a számítógépre!)

A központi egység bekapcsolásakor a nyomtató ismét végrehajtja az ellenőrzést, az írófej megint elmegy a lap közepéig és vissza.

Ezzel a nyomtatóval kibővített rendszerünk működésre kész.



51. ábra

ADATOK KINYOMTATÁSA – OPEN

Az adatok kinyomtatásához a nyomtató és a központi egység közti kapcsolatot „meg kell nyitnunk":

```
OPEN 4,4
```

Az OPEN (jelentése: nyiss, nyisd) kulcsszó azt jelenti, hogy meg akarunk nyitni, használatba akarunk venni valamilyen külső eszközt: jelen esetben a nyomtatót. Meg kell jelölnünk, hogy a 4-es vagy 5-ös nyomtatóról van-e szó: a mi esetünkben a 4-esről.
Ugyanaz a program természetesen sokféle eszközt használhat egy időben, ezek között sokféle nyomtatót is. A programon belül meg is kell számoznunk őket, de az ehhez szükséges számot magunk választhatjuk meg 1 és 127 között tetszőlegesen. Célszerű azonosnak választani a nyomtató számával; így a legkisebb a keveredés veszélye. Az OPEN után ezt a számot kell elsőként megadnunk, majd a nyomtató számát. Nálunk tehát mindkettő 4-es.
Figyelem! A nyomtatóval való gépi kommunikáció is egyfajta csatornán át zajlik, mint a lemezeknél láttuk, de ezt a csatornát a rendszer foglalja le, nem mi.

PRINT#

A megnyitás után máris kiírhatjuk az adatainkat:

```
PRINT#4,"EGY"
```

A PRINT# után az állomány azonosítóját kell megadnunk, vagyis az OPEN után írt első számot. Ezután vessző következik, majd jöhet a kinyomtatandó adat.
További hasonló parancsokat is kiadhatunk:

```
PRINT#4,2
PRINT#4,"HAROM"
```

Ezek után a listán a következő szöveget látjuk:

```
EGY
 2
HAROM
```

CLOSE

Ha a nyomtatást befejeztük, a nyomtatót le kell zárnunk:

```
CLOSE 4
```

ahol a CLOSE után ugyanazt az állományazonosítót kell megadnunk, amit a PRINT-ben.
Ha a # jelet véletlenül elhagyjuk, akkor például a

```
PRINT4,"HAROM"
```

parancs hatására a képernyőn jelenik meg a 4 és a „HAROM" szöveg is.

Természetesen a PRINT# utáni változólista is ugyanolyan összetett lehet, mint amikor a képernyőre írtunk, valamint ugyanúgy tartalmazhatja a vesszőt és a pontosvesszőt, amelyeknek a szerepe itt is hasonló, mint a képernyőn.
A nyomtató megnyitása, a nyomtatóra írás és a nyomtató lezárása előfordulhat programokban is, nemcsak parancsokban. Így a programjaink is írhatják papírra az eredményeiket.

A TARTALOMJEGYZÉK ÉS A PROGRAM KILISTÁZÁSA

Akár program van a tárban, akár a lemez tartalomjegyzéke, egyformán írhatjuk ki a papírra.
A nyomtatót a már látott módon kell megnyitnunk:

OPEN 4,4

CMD

Ezután egy speciális parancsot kell kiadnunk:

CMD 4

Ebben a parancsban a CMD kulcsszó után azt az állományazonosítót kell megadnunk, amely az OPEN-ben a vessző előtt állt. Mostantól aminek egyébként képernyőre kellene mennie, listára megy: már a CMD parancs utáni READY is.

LIST

Ha most kiadjuk a

LIST

parancsot, a tárban levő program vagy tartalomjegyzék a nyomtatóra kerül, a kiírás befejeztével pedig a READY is.

Több LIST paranccsal több példányt is készíthetünk. Kiadhatjuk mindazokat a sorszámmal kombinált LIST-változatokat is, amelyekkel a képernyőn elértük, hogy csak a program egyes sorai vagy részletei jelenjenek meg; a papíron is ugyanaz lesz a hatásuk.
Ha már nem kívánunk többet listázni, vissza kell állítanunk a normális állapotot. (Addig ugyanis a gép mindig a nyomtatót tekinti képernyőnek, ami zavarokat okozhat.) A visszaállítás két paranccsal történik:

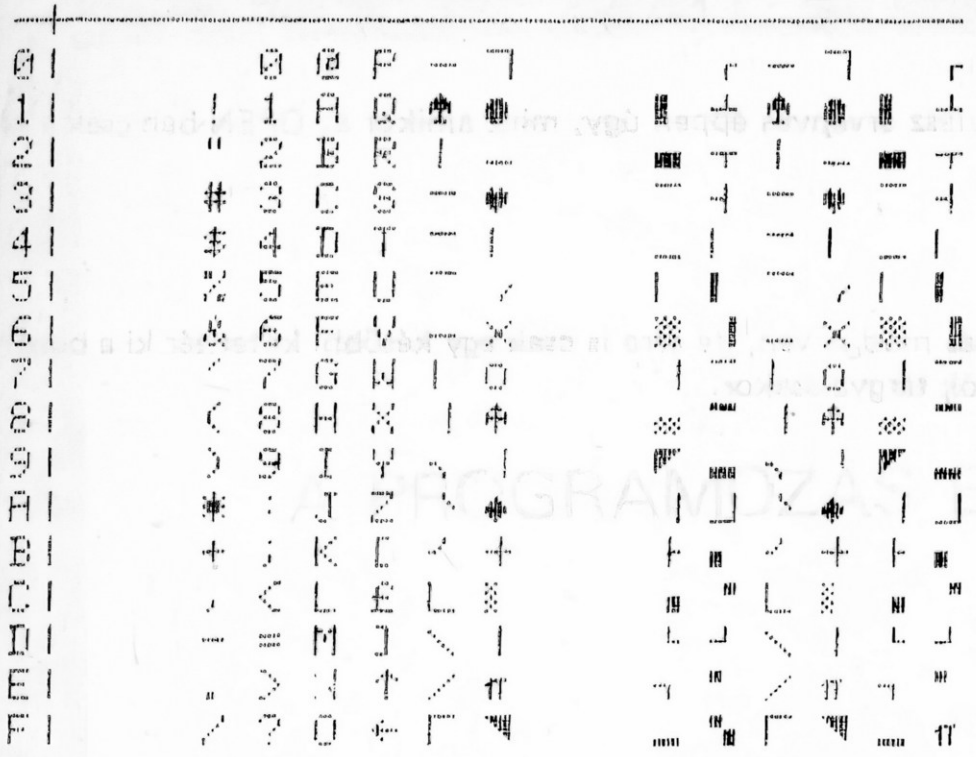
PRINT#4
CLOSE 4

A nyomtató erre kiír egy üres sort, majd visszaáll a normális állapot, a CLOSE utáni READY-t már a képernyőn látjuk.

SPECIÁLIS LEHETŐSÉGEK

A nyomtatáshoz a nyomtató által ismert bármelyik jel felhasználható. Választhatunk továbbá, hogy a kiírás fehér alapon fekete jelekkel, fekete alapon fehér jelekkel, normál vagy dupla szélességű jelekkel, illetve ezek kombinációival jelenjen meg.
Ezeknek a lehetőségeknek a használatához azonban a Commodore BASIC nyelvének bővebb ismerete szükséges, ezért a sorozatunk egy későbbi kötetében térünk csak ki rájuk.

MPS-801 ELSO JELKESZLET



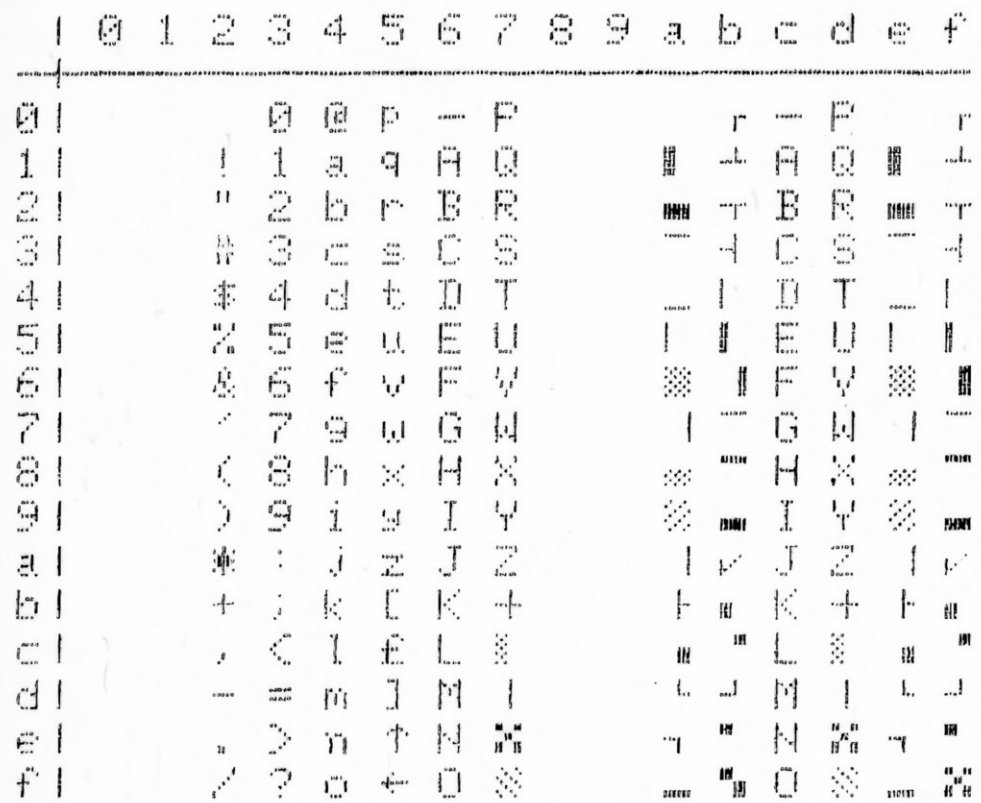
52. ábra

A NYOMTATÓ JELKÉSZLETEI

A nyomtatónak kétféle jelkészlete van. Az elsőt nagybetűs vagy grafikus jelkészletnek, a másodikat kisbetűs vagy szöveges jelkészletnek is szokás nevezni.

(Az 52–53. ábráról mellesleg a jelek ASCII-kódjának hexadecimális értéke is leolvas-

mPs-801 masodik jelkeszlet



53. ábra

ható. A táblázatok üres helyei olyan kódokhoz tartoznak, amelyeknek nem felel meg nyomtatható jel. Noha a jelek kódjai most nem igazán érdekelnek bennünket, ismeretük később még hasznos lesz.)

OPEN 4,4,0

hatására az első jelkészlet lesz érvényes éppen úgy, mint amikor az OPEN-ben csak a két 4-est adjuk meg;

OPEN 4,4,7

hatására pedig a második.
A jelkészlet átállítására más mód is van, de erre is csak egy későbbi kötet tér ki a bonyolultabb nyomtatási funkciók tárgyalásakor.

A PROGRAMOZÁS ELEMEI

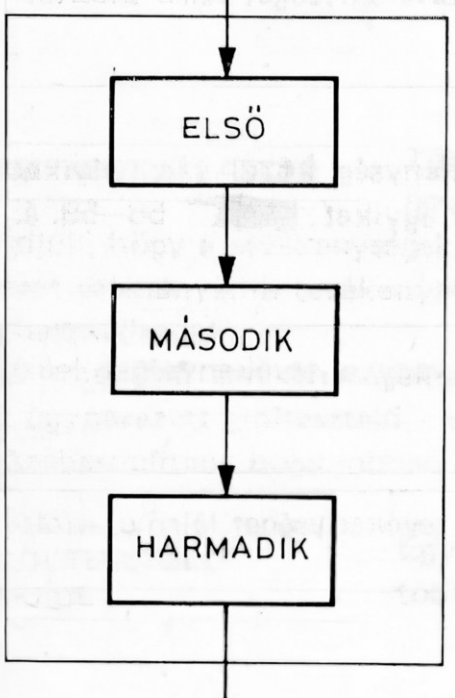
Szabványos tevékenységszerkezetek

Az eddig megismert módszerekkel a következő jellegzetes tevékenységszerkezeteket tudjuk megvalósítani. (Hatféle alapvető tevékenységszerkezet struktúra- és folyamatábráját közöljük.)

SZEKVENCIÁLIS

Egy összetett tevékenység több résztevékenység egymásutánjából áll:

54. ábra



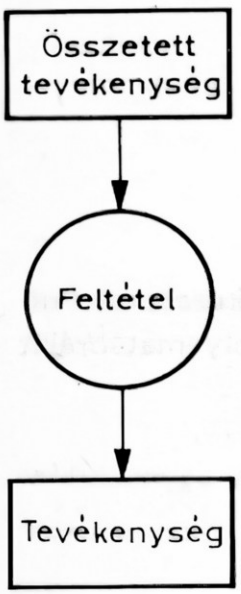
Ez a teljes összetett tevékenység

55. ábra

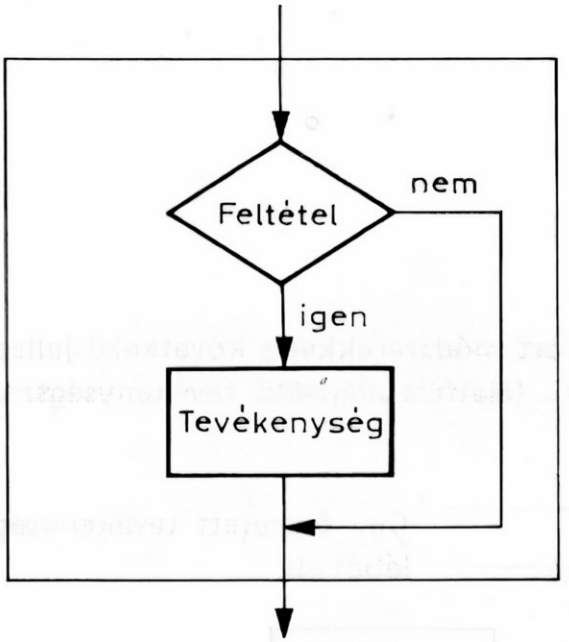
Megvalósítása: Egyszerűen le kell írnunk sorban, növekvő utasítássorszámokkal az ELSŐ, a MÁSODIK, majd a HARMADIK résztevékenység utasításait vagy az őket helyettesítő GOSUB utasításokat. Természetesen ugyanez a helyzet, ha nem három, hanem akárhány résztevékenység alkot egy összetett tevékenységet.

FELTÉTELES

Egy tevékenységet egy feltételtől függően vagy végre kell hajtani, vagy nem:



56. ábra



57. ábra

Ez a teljes összetett tevékenység

Megvalósítása:

IF *feltétel* THEN GOSUB . . .

vagy

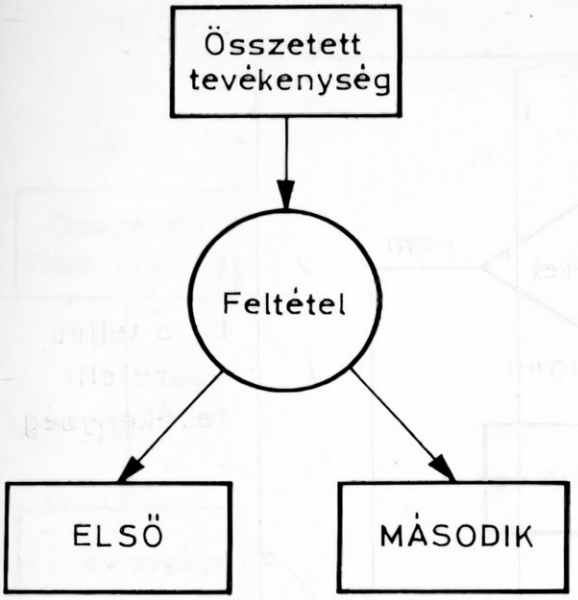
IF NOT *feltétel* THEN GOTO
itt akár a tevékenységre utaló GOSUB, akár a tevékenységet leíró *utasítássorozat* állhat
a program folytatása

ALTERNATÍV

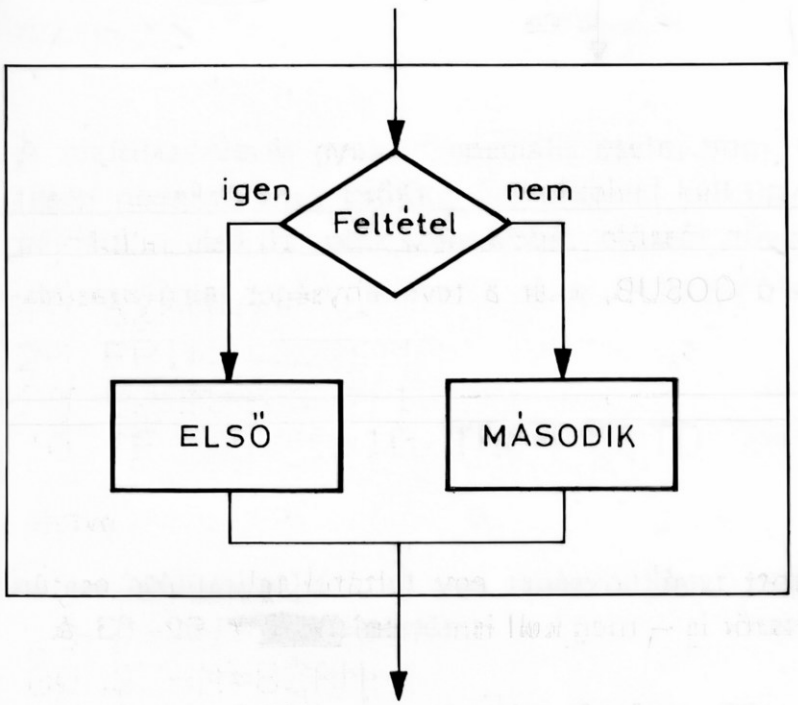
Egy feltételtől függően két tevékenység közül valamelyiket kell végrehajtani, de mindig csak az egyiket. 58–59. á.

Megvalósítása:

IF *feltétel* THEN GOTO
itt a MÁSODIK tevékenységre utaló GOSUB vagy a tevékenységet leíró *utasításssorozat* áll
GOTO
itt az ELSŐ tevékenységre utaló GOSUB vagy a tevékenységet leíró *utasítássorozat* áll
a program folytatása



58. ábra



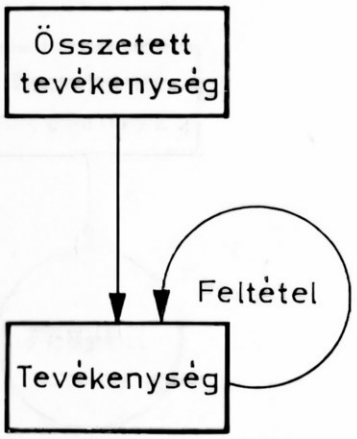
Ez a teljes összetett tevékenység

59. ábra

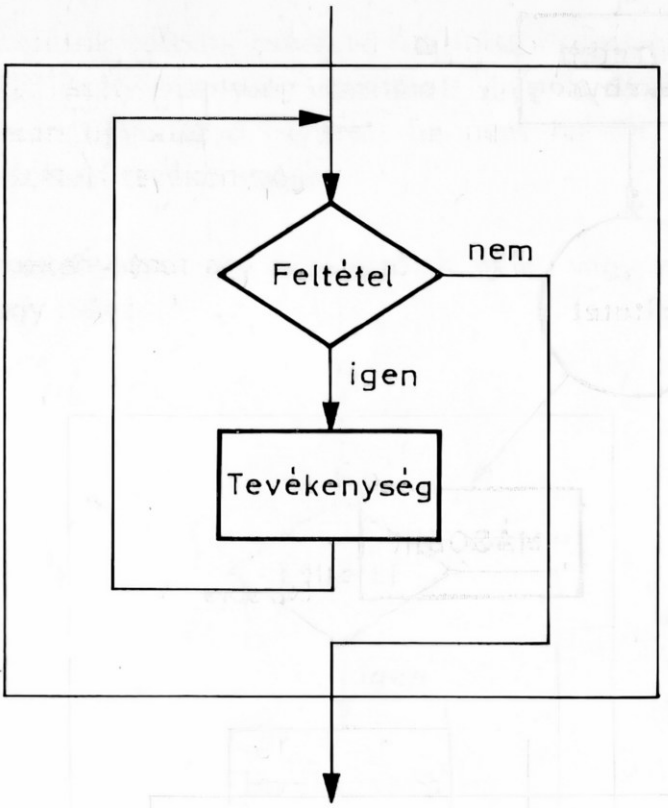
ESETSZÉTVÁLASZTÓ Több lehetséges tevékenység közül kell egy feltételtől függően egyet kiválasztani (ugyanígy építhető fel). Speciális esetben, ha tudjuk, hogy a tevékenységek közül valamiféle sorszám szerint is lehet választani (egy sorozat valahányadik tevékenységét kell végrehajtani), akkor a már megismert ON utasítást is használhatjuk.

Az ismétlődő tevékenységek megszervezéséhez kétféle szerkezet is használható; egyrészt az úgynevezett elöltesztelő, másrészt az úgynevezett hátultesztelő ciklus. (Érdemes őket összehasonlítani, hogy jobban megérthessük a jellegzetességeiket.)

ELÖLTESZTELŐ CIKLUS Egy tevékenységet egy feltétel teljesülése esetén kell — esetleg többször is — végrehajtani.



60. ábra



61. ábra

Megvalósítása:

IF NOT *feltétel* THEN GOTO
itt akár a tevékenységre utaló GOSUB, akár a tevékenységet leíró *utasítássorozat* állhat
GOTO
a program folytatása

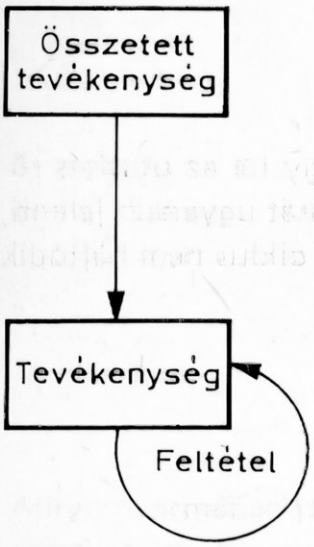
HÁTULTESZTELŐ CIKLUS

Egy végrehajtott tevékenységet egy feltétel teljesülése esetén — esetleg többször is — meg kell ismételni. ☛ 62–63. á.

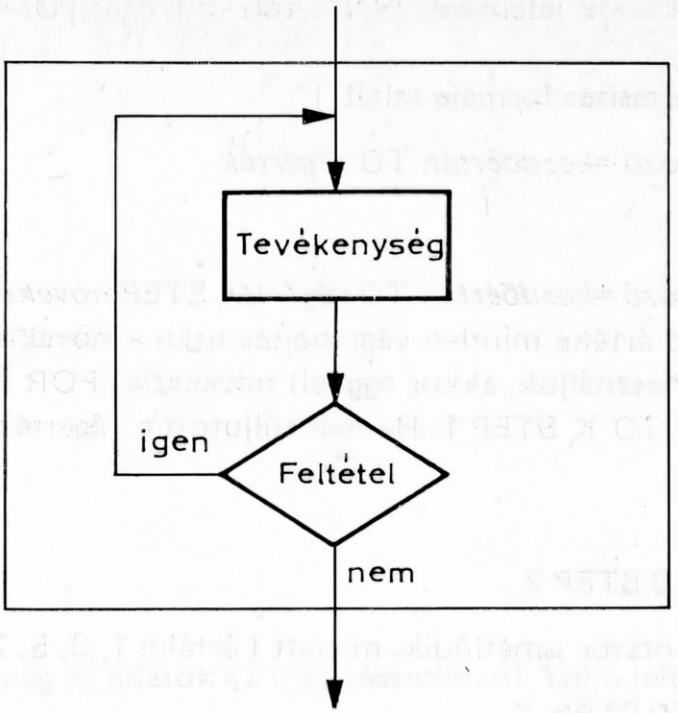
Megvalósítása:

itt akár a tevékenységre utaló GOSUB, akár a tevékenységet leíró *utasítássorozat* állhat
IF *feltétel* THEN GOTO
a program folytatása

Minden program, amely a gépünkön megvalósítható, összeállítható ezekből az építőkövekből, panelekből is, sőt még fel sem kell használnunk mindegyiküket. Érdemes mégis ismerni valamennyit és ezekből építkezni, mert amennyi megkötést, annyi előnyt is jelentenek. Az így felépülő program egyes részletei áttekinthetőbbek, egymástól függetlenebbek lesznek, egy-egy módosítás helye könnyebben megtalálható, és a módosításokat anélkül tudjuk végrehajtani, hogy a módosításnak a programra nem kívánt mellékhatása lenne.



62. ábra



63. ábra

Ez a teljes összetett tevékenység

A ciklusszervezés gyakori speciális esete, hogy valamilyen változó különböző, egyenletesen növekvő vagy csökkenő értékeivel kell ugyanazt a műveletet elvégeznünk. Írjuk ki például az első tíz egész szám köbét, először növekvő, majd csökkenő sorrendben!

```
10 SZAM=1
20 PRINT SZAM↑3
30 SZAM=SZAM+1
40 IF SZAM<=10 THEN GOTO 20
```

64. ábra

illetve

```
10 SZAM=10
20 PRINT SZAM↑3
30 SZAM=SZAM-1
40 IF SZAM>=1 THEN GOTO 20
```

65. ábra

Ezekre az esetekre a BASIC-nek külön utasításpárja van. E két kis programot így is írhatjuk:

```
10 FOR SZAM=1 TO 10
20 PRINT SZAM↑3
30 NEXT SZAM
```

66. ábra

illetve

```
10 FOR SZAM=10 TO 1 STEP -1
20 PRINT SZAM↑3
30 NEXT SZAM
```

67. ábra

Az egyes kulcsszavak jelentése: FOR: -tól, -től, óta; TO: -ig; STEP: lépj; NEXT: soron következő.
A cikluskezdő utasítás formája tehát:

FOR *ciklusváltozó=kezdőérték* TO *végérték*

vagy

FOR *ciklusváltozó=kezdőérték* TO *végérték* STEP *növekmény*

A ciklusváltozó értéke minden végrehajtás után a növekménnyel (vagy ha az utasítás rövidebb alakját használjuk, akkor eggyel) növekszik. FOR I=J TO K tehát ugyanazt jelenti, mint FOR I=J TO K STEP 1. Ha már túljutott a végértéken, akkor a ciklus nem hajtódik végre többször.
Így például a

FOR I=1 TO 10 STEP 2

kezdetű ciklus ötször ismétlődik, mialatt I értéke 1, 3, 5, 7 és 9. A

FOR I=0 TO 10 STEP .5

kezdetű ciklus 21-szer hajtódik végre, s ezelatt I értéke 0, 0,5, 1, 1,5, . . . 9,5 és 10. A

FOR I=3 TO −3 STEP −1

kezdetű ciklus 7-szer hajtódik végre, I értéke közben 3, 2, 1, 0, −1, −2 és −3. A

FOR I=5 TO 0

kezdetű ciklus egyszer fut végig (mert ez a ciklus hátultesztelő, tehát legalább egyszer akkor is végrehajtódik, ha a ciklusváltozó eleve túl van a végértéken), mialatt I értéke 5.
A kezdőérték, a végérték és a növekmény természetesen változó vagy kifejezés is lehet. Az ilyen változóknak vagy magának a ciklusváltozónak a cikluson belüli megváltoztatása — bár formailag szabályos és megengedett — a program áttekinthetőségét és biztonságát rontja, ezért nem célszerű.
Ismét felhívjuk a figyelmet a számok (főleg a nem egész számok) pontatlanságaira, kerekítési hibáira. A

FOR I=10000000000 TO 10000000001

kezdetű ciklus végtelen, mert a gép csekély számábrázolási képességének túllépése miatt 10000000000+1 kerekítve ismét 10000000000 lesz, vagyis a 10000000001-et soha nem éri el. A

FOR I=1000 TO 2000 STEP 1

kezdetű ciklus természetesen 1001-szer hajtódik végre, de a

FOR I=1 TO 2 STEP 0.001

kezdetű ciklus (kerekítési hibák miatt) csak 1000-szer.
A FOR–NEXT utasításpárral határolt programrészletben (a ciklustörzsben, ciklusmagban) bármi lehet, amit ismételgetni akarunk; akár újabb ciklus is.
Konkrét példát a ciklusszervezésre az ismétlődő adatok feldolgozásával foglalkozó fejezetben láthatunk.

Egyedi adatok

Milyen formában jelenhetnek meg az adatok az utasításainkban? Több lehetőség közül választhatunk. Egy adat megadható:

– konstans, pl. 25, 3.4, –42.5E3 vagy "ALFA"
– változó, pl. X%, Y vagy Z$
– függvényérték, pl. INT(A), SQR(B) vagy LEFT$(C$,2)
– kifejezésérték, pl. A%+B*SIN(F) vagy STR$(G)+EQ

formájában. A bemutatott példák egész, illetve valós numerikus, valamint karakteres (szöveges) adatokat határoznak meg.

A BASIC nyelvtani szabályai általában nem korlátozzák, hogy mikor melyik adatmegadási módot használjuk. Egy és ugyanaz az adat, például a kettő négyzetgyöke többféleképpen is megadható.

A programozóra van bízva, hogy az adott feladatnak leginkább megfelelő kifejezési formát válassza.

Tekintsük át a lehetőségeket a 68. ábra szerint.

Az első esetben a gyök 2 értékét egyszerűen egy konstanssal adtuk meg. A második esetben azt a számot (a kettőt), amelyből gyököt akarunk vonni, konstanssal fejeztük ki, a gyökvonás eredményét pedig egy egyszerű kifejezéssel határoztuk meg. A harmadik esetben a gyökvonás paramétere változóként szerepel, a gyököt egy összetett kifejezés adja. A negyedik esetben a paraméter továbbra is változó, a gyököt függvény állítja elő. Végül az ötödik esetben a változóként megadott paraméterből egy külön eljárás állítja elő – ugyancsak változóként kifejezve – a kívánt adatot, a kettő négyzetgyökét.

Az adatoknak tehát nemcsak a korábban említett közvetlen, hanem közvetett megadási módja is van, amikor nem magát az adatot adjuk meg, hanem az előállítására, kiszámolására vonatkozó eljárást.

Ha a fenti programhoz begépeljük a gyökvonó eljárást, és a programot elindítjuk, a képernyőn sorra megjelenő eredményeket összehasonlítva láthatjuk, hogy mind az öt adatmegadási módunk ugyanazt az adatot határozta meg. ☞ 69. ábra.

Az itt szereplő gyökvonó eljárás egyébként nemcsak a gép pontatlanságára és annak programbeli kezelésére mutat be példát, hanem a matematikai, természettudományos, műszaki számítások egyik igen gyakori megoldástípusának, a közelítő eljárásoknak a progra-

```
10 X=2
20 PRINT : PRINT , 1.41421356
30 PRINT : PRINT , 2↑0.5
40 PRINT : PRINT , X↑(1/2)
50 PRINT : PRINT , SQR(X)
60 GOSUB 2010 : PRINT : PRINT , Y
99 END
```

68. ábra

```
10 X=2
20 PRINT : PRINT , 1.41421356
30 PRINT : PRINT , 2↑0.5
40 PRINT : PRINT , X↑(1/2)
50 PRINT : PRINT , SQR(X)
60 GOSUB 2010 : PRINT : PRINT , Y
99 END
2010 LET G=0
2020 IF X<=0 THEN GOTO 2410
2030 IF X>=200 THEN GOTO 2410
2040 LET A=0
2050 LET F=X
2060 LET G=X/2
2110 LET N=G*G
2120 IF ABS(N-X)<0.000000001 THEN GOTO 2410
2130 IF N>X THEN GOTO 2310
2210 LET A=G
2220 LET G=G+(F-G)/2
2230 GOTO 2110
2310 LET F=G
2320 LET G=G-(G-A)/2
2330 GOTO 2110
2410 LET Y=G
2999 RETURN

1.41421356

1.41421356

1.41421356

1.41421356

1.41421356
```

69. ábra

mozására is. Ezért mindenkinek, akit ez a terület érdekel, ajánljuk az áttanulmányozását és kipróbálását különféle változtatásokkal. Úgy véljük azonban, hogy az eddig tanultak alapján semmi nehézséget nem okozhat az önálló megfejtése, ezért magyarázatot nem fűzünk hozzá.
A függvények kipróbálására bemutatott programok az eredményeket a képernyőn jelenítik meg, könyvünkben pusztán nyomdatechnikai okokból szerepelnek nyomtatott listák, amelyekre ugyanazt írattuk ki, mint amit egyébként a képernyőn látnánk.

FÜGGVÉNYEK

Figyelmünket most arra az adatmegadási módra fordítjuk, amellyel eddig még nem találkoztunk: a függvényekre.

A Commodore-nak gyárilag beépített függvénykészlete van, de emellett a nyelv lehetővé teszi azt is, hogy a programozó saját függvényeket is definiálhasson.
Az úgynevezett beépített függvényeket a gép ismeri, ezeket külön definiálni nem kell, sőt nem is szabad. Bármely programban bárhol felhasználhatók adatmegadásra. Például:

```
50 PRINT SQR(X)
```

A függvénnyel való adatmegadás módja az, hogy először leírjuk a függvény nevét, majd zárójelek között a függvény paraméterét, amelyre a függvényértéket meg akarjuk kapni.
A beépített függvények neve kulcsszó, semmilyen más célra nem használható. Az egyes függvények paramétereinek száma, típusa, sorrendje kötött. A paraméterek azonban a korábban megismert bármelyik közvetlen adatmegadási móddal meghatározhatók, így lehetnek akár olyan kifejezések is, amelyek további függvényeket tartalmaznak. Például:

```
777 Y=SIN(SQR(A)*28/VAL(B$))
```

A programozó saját függvényeit viszont nem ismeri a gép, azokat a programozónak minden programban definiálnia kell, ahol használni akarja. A saját függvényekre ebben a fejezetben még részletesen visszatérünk.
A függvényeket hagyományosan három kategóriába soroljuk:
— numerikus,
— szövegkezelő,
— egyéb függvények.

NUMERIKUS FÜGGVÉNYEK

ABS(X)

Abszolútérték-függvény. Paramétere bármilyen numerikus érték lehet. A függvényérték a paraméter matematikai értelemben vett abszolút értéke. ☞ 70. ábra.

ATN(X)

Arkusztangens-függvény. Paramétere bármilyen numerikus érték lehet. A függvényérték az a szög — ívmértékben kifejezve —, amelynek a tangense a megadott paraméter. ☞ 71. ábra.

```
10 PRINT
11 PRINT " X=-3","ABS(X)=";ABS(-3)
20 PRINT
21 PRINT " X=0","ABS(X)=";ABS(0)
30 PRINT
31 PRINT " X=5","ABS(X)=";ABS(5)
```

```
X=-3            ABS(X)= 3

X=0             ABS(X)= 0

X=5             ABS(X)= 5
```

70. ábra

```
10 PRINT
11 PRINT " X= 1","ATN(X)=";ATN(1),
12 PRINT "[=";ATN(1)/π*180;"° ]"
20 PRINT
21 PRINT " X= 0","ATN(X)=";ATN(0),
22 PRINT "[=";ATN(0)/π*180;"° ]"
30 PRINT
31 PRINT " X=-1","ATN(X)=";ATN(-1),
32 PRINT "[=";ATN(-1)/π*180;"° ]"
40 PRINT
41 PRINT " X=10000000000","ATN(X)=";ATN(10000000000),
42 PRINT "[=";ATN(10000000000)/π*180;"° ]"
```

```
X= 1            ATN(X)= .785398163          [= 45 ° ]

X= 0            ATN(X)= 0                   [= 0 ° ]

X=-1            ATN(X)=-.785398163          [=-45 ° ]

X=10000000000   ATN(X)= 1.570796327         [= 90 ° ]
```

71. ábra

Az arkusztangens-függvény csak a szögek főértékeit adja, vagyis a függvényérték fokokra átszámítva -90° és $+90^\circ$ között lesz.

COS(X)

Koszinuszfüggvény. Paramétere bármilyen numerikus érték lehet. A függvényérték annak a szögnek a koszinusza, amelyet a paraméter ívmértékben kifejezve meghatároz.

```
10 FOR FOK=-30 TO 120 STEP 30
20 PRINT
30 X=FOK*π/180
40 PRINT "FOK=";FOK,
41 PRINT "RAD   =";X
42 PRINT , "COS(X)=";COS(X)
50 NEXT FOK
```

```
FOK=-30          RAD    =-.523598776
                 COS(X)= .866025404

FOK= 0           RAD    = 0
                 COS(X)= 1

FOK= 30          RAD    = .523598776
                 COS(X)= .866025404

FOK= 60          RAD    = 1.04719755
                 COS(X)= .5

FOK= 90          RAD    = 1.57079633
                 COS(X)= 0

FOK= 12          RAD    = 2.0943951
                 COS(X)=-.5
```

72. ábra

EXP(X)

Exponenciális függvény. Paramétere bármilyen numerikus érték lehet –88,0296919 és +88,0296919 között. A felső határt meghaladó paraméter esetén OVERFLOW ERROR (túlcsordulás) hibaüzenetet kapunk, az alsó határ alatti értéket írva pedig a függvényérték mindig 0 lesz. Egyébként a függvényérték a természetes alapú logaritmus alapszámának (a Commodore pontosságával e=2,71828183) a paraméter által meghatározott értékre emelt hatványa.

```
10 FOR X=-1 TO 3 STEP 0.5
20 PRINT
30 PRINT " X=";X,"EXP(X)=";EXP(X)
40 NEXT X
```

73. ábra

A határok túllépését a

PRINT EXP(99)
PRINT EXP(–99)

parancsok kiadásával próbálhatjuk ki.

```
X=-1          EXP(X)= .367879441
X=-.5         EXP(X)= .60653066
X= 0          EXP(X)= 1
X= .5         EXP(X)= 1.64872127
X= 1          EXP(X)= 2.71828183
X= 1.5        EXP(X)= 4.48168907
X= 2          EXP(X)= 7.3890561
X= 2.5        EXP(X)= 12.182494
X= 3          EXP(X)= 20.0855369
```

73. ábra folytatása.

FNf(X)

A programozó által definiálható saját függvény. Minthogy ezek nevére, paraméterére, függvényértékére, használatára sajátos szabályok vonatkoznak, tárgyalásuknak külön fejezetet szentelünk a 122. oldalon.

INT(X)

Törtet egésszé alakító függvény (egészrészfüggvény). Paraméterre bármilyen numerikus érték lehet. A függvényérték az a legnagyobb egész szám, amely még nem nagyobb a paraméternél.
A függvény az egész részt nem a tizedesjegyek levágásával képezi. A negatív törtszámok egész része abszolút értékben nagyobb a törtnél. Ha a paraméter egész, akkor a függvényérték egyenlő lesz a paraméterrel — ezt használhatjuk ki, ha éppen azt akarjuk vizsgálni, hogy egy szám egész-e.

```
10 PRINT
20 PRINT " INT(2.9)     = ";INT(2.9)
30 PRINT " INT(-2.1)    = ";INT(-2.1)
40 PRINT " INT(0)       = ";INT(0)
50 PRINT " INT(5)       = ";INT(5)
60 PRINT " INT(-4)      = ";INT(-4)
```

```
INT(2.9)     =  2
INT(-2.1)    = -3
INT(0)       =  0
INT(5)       =  5
INT(-4)      = -4
```

74. ábra

LOG(X)

Logaritmusfüggvény. Paramétere bármilyen pozitív numerikus érték lehet. Negatív vagy nulla paraméter esetén ILLEGAL QUANTITY ERROR (itt: érvénytelen paraméter) hibajelzést kapunk. A függvényérték a paraméterként megadott érték természetes alapú logaritmusa.

```
10 PRINT
20 FOR X=1 TO 56 STEP 5.5
30 PRINT " X=";X,"LOG(X)=";LOG(X)
40 NEXT X
50 PRINT
60 PRINT " X=-4","LOG(X)=";LOG(-4)

 X= 1          LOG(X)= 0
 X= 6.5        LOG(X)= 1.87180218
 X= 12         LOG(X)= 2.48490665
 X= 17.5       LOG(X)= 2.86220088
 X= 23         LOG(X)= 3.13549422
 X= 28.5       LOG(X)= 3.34990409
 X= 34         LOG(X)= 3.52636053
 X= 39.5       LOG(X)= 3.67630067
 X= 45         LOG(X)= 3.80666249
 X= 50.5       LOG(X)= 3.92197334
 X= 56         LOG(X)= 4.02535169

 X=-4          LOG(X)=
?ILLEGAL QUANTITY  ERROR IN 60
```

75. ábra

PEEK(X)

A tártartalmat megadó függvény. Paramétere bármilyen numerikus érték lehet, amelynek egész része 0 és 65535 közé esik (a határokat is megengedve). Ha a paraméter nem egész, a függvény csak az egész részét veszi figyelembe. 65535-nél nagyobb vagy negatív egész részű paraméter esetén ILLEGAL QUANTITY ERROR hibaüzenetet kapunk. A függvény értéke a tár azon bájtjának tartalma, amelynek (decimális) címe a megadott paraméter. ☛ 76. ábra.
Az eredmény egy-egy 0 és 255 közötti szám vagy az említett hibaüzenet.

RND(X)

Véletlenszámfüggvény. Paramétere bármilyen numerikus érték lehet. A függvényérték mindig 0 és 1 közé eső úgynevezett álvéletlenszám. Ha a paraméter negatív, akkor azonos paraméterérték esetén mindig ugyanazt a függvényértéket kapjuk. Ha egy negatív paraméterrel előállított véletlenszám után pozitív paraméterrel hozunk létre véletlenszám-sorozatot, akkor az azonos paraméterértékekhez mindig ugyanaz a függvényérték-sorozat tartozik. Ezt a kötöttséget nulla értékű paraméter használatával oldhatjuk fel. ☛ 77. ábra.
Látható, hogy a negatív, valamint az utána következő pozitív paraméterek mindig ugyan-

```
10 PRINT
20 FOR X=12345 TO 54321 STEP 5000
30 PRINT " X=";X,"PEEK(X)=";PEEK(X)
40 NEXT X
50 PRINT
60 X=66666
70 PRINT " X=";X,"PEEK(X)=";PEEK(X)

 X= 12345                PEEK(X)= 80
 X= 17345                PEEK(X)= 231
 X= 22345                PEEK(X)= 4
 X= 27345                PEEK(X)= 255
 X= 32345                PEEK(X)= 32
 X= 37345                PEEK(X)= 49
 X= 42345                PEEK(X)= 157
 X= 47345                PEEK(X)= 105
 X= 52345                PEEK(X)= 0

 X= 66666                PEEK(X)=
?ILLEGAL QUANTITY  ERROR IN 70
```

76. ábra

```
10 FOR I=1 TO 2
20 PRINT
30 PRINT " RND(-1)=";RND(-1)
40 PRINT " RND(1)=";RND(1)
50 PRINT " RND(1)=";RND(1)
60 PRINT
70 PRINT " RND(0)=";RND(0)
80 PRINT " RND(1)=";RND(1)
90 PRINT " RND(1)=";RND(1)
99 NEXT I

RND(-1)= 2.9919647E-08
RND(1)= .328780872
RND(1)= .978964086

RND(0)= .886719644
RND(1)= .803752639
RND(1)= .444224451

RND(-1)= 2.9919647E-08
RND(1)= .328780872
RND(1)= .978964086
```

```
RND(0)= .386720002
RND(1)= .0979644984
RND(1)= .116913769
```

77. ábra

olyan, a nulla és az utána következő pozitív paraméterek mindig különböző függvényértékeket adnak.

```
10 FOR I=1 TO 2
20 PRINT
30 PRINT " RND(-9)=";RND(-9)
40 PRINT " RND(1)=";RND(1)
50 PRINT " RND(1)=";RND(1)
60 PRINT
70 PRINT " RND(0)=";RND(0)
80 PRINT " RND(1)=";RND(1)
90 PRINT " RND(1)=";RND(1)
99 NEXT I

RND(-9)= 3.3647666E-08
RND(1)= .18740319
RND(1)= .26714906

RND(0)= .507814467
RND(1)= .639558418
RND(1)= .291990096

RND(-9)= 3.3647666E-08
RND(1)= .18740319
RND(1)= .26714906

RND(0)= .125000715
RND(1)= .975544257
RND(1)= .873517376
```

78. ábra

Ugyanezt tapasztaljuk más paraméterekkel is, ilyenkor azonban a fentiektől eltérő függvényértékek állnak elő.
Megjegyezzük, hogy más számok, például N és M között is előállíthatunk véletlenszámsorozatot az

INT(RND(X)*M)+N

kifejezéssel. Például öt véletlenszerű lottószámot a 79. ábra programjával generálhatunk. Ez a programocska nem gondoskodik arról, hogy a sorozatban ne fordulhassanak elő azonos számok, és a lottószámokat nem növekvő sorrendben adja meg. Minderről — ha kívánjuk — külön gondoskodhatunk.

```
10 PRINT
20 PRINT " LOTTOSZAMOK:"
30 PRINT
40 FOR I=1 TO 5
50 PRINT INT(RND(1)*90)+1;
60 NEXT I
70 PRINT

LOTTOSZAMOK:

71  47  13  32  32

LOTTOSZAMOK:

9  58  86  48  69
```

79. ábra

SGN(X)

Előjelfüggvény. A paraméter tetszőleges numerikus érték lehet. A függvényérték 1, 0 vagy −1 aszerint, hogy a paraméter értéke pozitív, nulla vagy negatív.

```
10 PRINT
20 PRINT " SGN(5)      =",SGN(5)
30 PRINT " SGN(0)      =",SGN(0)
40 PRINT " SGN(-3)     =",SGN(-3)
50 PRINT " SGN(6.28)   =",SGN(6.28)
60 PRINT " SGN(-2E3)   =",SGN(-2E3)

SGN(5)      =  1
SGN(0)      =  0
SGN(-3)     = -1
SGN(6.28)   =  1
SGN(-2E3)   = -1
```

80. ábra

SIN(X)

Szinuszfüggvény. Paramétere bármilyen numerikus érték lehet. A függvényérték annak a szögnek a szinusza, amelyet a paraméter ívmértékben kifejezve meghatároz. ☞ 81. ábra.

SQR(X)

Négyzetgyökfüggvény. Paramétere bármilyen nem negatív numerikus érték lehet. Negatív paraméter esetén ILLEGAL QUANTITY ERROR hibajelzést kapunk. A függvényérték a paraméterként megadott szám négyzetgyöke. ☞ 82. ábra.

```
10 PRINT
20 FOR FOK=-30 TO 450 STEP 30
30 SNX=SIN(FOK*π/180)
35 IF ABS(SNX)<0.00000001 THEN SNX=0
40 PRINT " FOK=";FOK;"   ";"SIN(X)=";SNX
50 NEXT FOK
```

```
FOK=-30        SIN(X)=-.5
FOK= 0         SIN(X)= 0
FOK= 30        SIN(X)= .5
FOK= 60        SIN(X)= .866025404
FOK= 90        SIN(X)= 1
FOK= 120       SIN(X)= .866025404
FOK= 150       SIN(X)= .5
FOK= 180       SIN(X)= 0
FOK= 210       SIN(X)=-.499999998
FOK= 240       SIN(X)=-.866025403
FOK= 270       SIN(X)=-1
FOK= 300       SIN(X)=-.866025405
FOK= 330       SIN(X)=-.5
FOK= 360       SIN(X)= 0
FOK= 390       SIN(X)= .5
FOK= 420       SIN(X)= .866025402
FOK= 450       SIN(X)= 1
```

81. ábra

```
10 PRINT
20 FOR X=0 TO 72 STEP 8
30 PRINT " X=";X,"SQR(X)=";SQR(X)
40 NEXT X
50 PRINT
60 PRINT " X=-4","SQR(X)=";SQR(-4)
```

```
 X= 0          SQR(X)= 0
 X= 8          SQR(X)= 2.82842713
 X= 16         SQR(X)= 4
 X= 24         SQR(X)= 4.89897949
 X= 32         SQR(X)= 5.65685425
 X= 40         SQR(X)= 6.32455532
 X= 48         SQR(X)= 6.92820323
 X= 56         SQR(X)= 7.48331478
 X= 64         SQR(X)= 8
 X= 72         SQR(X)= 8.48528138

 X=-4          SQR(X)=
?ILLEGAL QUANTITY  ERROR IN 60
```

82. ábra

TAN(X) Tangensfüggvény. Paramétere bármilyen numerikus érték lehet, ha arra a függvény matematikailag értelmezve van. A függvényérték annak a szögnek a tangense, amelyet a paraméter ívmértékben kifejezve meghatároz. A paraméter nem veheti fel a 90 fok páratlan számú pozitív vagy negatív többszöröseinek megfelelő radiánértékeket. Ha mégis megteszi, vagy rendkívül pontatlan eredményt, vagy jobb esetben DIVISION BY ZERO ERROR (nullával osztás) hibaüzenetet kapunk.

Próbáljuk ki a függvényt –30, 0, 45, 270 és 90 fokkal!

```
10 PRINT
11 X=-30*π/180
12 PRINT " X=";X,"TAN(X)=";TAN(X)
20 PRINT
21 X=0*π/180
22 PRINT " X=";X,"TAN(X)=";TAN(X)
30 PRINT
31 X=45*π/180
32 PRINT " X=";X,"TAN(X)=";TAN(X)
40 PRINT
41 X=270*π/180
42 PRINT " X=";X,"TAN(X)=";TAN(X)
50 PRINT
51 X=90*π/180
52 PRINT " X=";X,"TAN(X)=";TAN(X)

 X=-.523598776              TAN(X)=-.577350269

 X= 0                       TAN(X)= 0

 X= .785398163              TAN(X)= .999999999

 X= 4.71238898              TAN(X)= 68356527S

 X= 1.57079633              TAN(X)=
?DIVISION BY ZERO  ERROR IN 52
```

83. ábra

Látható, hogy míg az első három esetben többé-kevésbé a várt értéket kapjuk, csak 90 fok esetén jelenik meg a logikus hibaüzenet, 270 foknál a függvény teljesen értékelhetetlen eredményt ad. Minthogy a gép a többi kritikus szögnél ugyancsak következetlenül működik, a tangensfüggvény használata előtt a paramétert mindig meg kell vizsgálnunk, és 90 fok páratlan többszöröseire nem is szabad aktivizálnunk.

Megjegyezzük, hogy ez a következetlenség részben az ívmértékre történő átszámítás pontatlanságából származik. Hibaüzenetet csak a megfelelően pontos radián érték esetén várhatunk.

ASC(X$)

Kódkonvertáló függvény. Paramétere bármilyen nem üres szöveges érték lehet. Üres szöveg megadása esetén ILLEGAL QUANTITY ERROR hibaüzenetet kapunk. A függvényérték a paraméterként megadott szöveg első jelének ASCII (amerikai szabvány) kódja. (Ez 0 és 255 közötti egész.)

```
10 PRINT
20 INPUT "X$=";X$
30 PRINT
40 PRINT "X$=!";X$;"!",
50 PRINT "ASC(X$)=";ASC(X$)

X$=? A

X$=!A!          ASC(X$)= 65

X$=? ABC

X$=!ABC!        ASC(X$)= 65

X$=? 1

X$=!1!          ASC(X$)= 49

X$=? ♥

X$=!♥!          ASC(X$)= 211
```

84. ábra

Próbáljuk ki a fenti programot „A", „ABC", „1" értékkel, valamint egy speciális jellel.

CHR$(X)

Kódkonvertáló függvény. Paramétere bármilyen numerikus érték lehet, amelynek egész része 0 és 255 közé esik (a határokat is megengedve). A függvényérték az a jel, amelynek ASCII kódja a paraméter értékének

```
10 PRINT
20 PRINT "X= 65",
30 PRINT "CHR$(X)=!";CHR$(65);"!"
40 PRINT "X= 49",
50 PRINT "CHR$(X)=!";CHR$(49);"!"
60 PRINT "X= 211",
70 PRINT "CHR$(X)=!";CHR$(211);"!"
80 PRINT "X= 333",
90 PRINT "CHR$(X)=!";CHR$(333);"!"
```

```
X= 65       CHR$(X)=!A!
X= 49       CHR$(X)=!1!
X= 211      CHR$(X)=!♦!
X= 333      CHR$(X)=!
?ILLEGAL QUANTITY  ERROR IN 90
```
85. ábra

egész része. Nem létező kód, vagyis negatív vagy 255-nél nagyobb egész részű paraméter esetén ILLEGAL QUANTITY ERROR hibajelzést kapunk.
Összehasonlítva az eredményt az ASC függvény működését szemléltető mintaprogram eredményével, látható, hogy ez a két függvény egymás inverze, vagyis

CHR$(ASC(X$)) értéke X$
ASC(CHR$(X)) értéke X

Erről a következő mintaprogram futtatásával is meggyőződhetünk:

```
10 PRINT
20 INPUT "X$=";X$
30 PRINT
40 PRINT "X$=";X$,
41 PRINT "CHR$(ASC(X$))=";CHR$(ASC(X$))
50 PRINT
60 INPUT "X =";X
70 PRINT
80 PRINT "X =";X,
81 PRINT "ASC(CHR$(X ))=";ASC(CHR$(X ))
```
86. ábra

LEFT$(XS,N)

Szövegleválasztó függvény. Két paramétere van. Az első bármilyen szöveges érték lehet, a másik olyan numerikus érték, amelynek egész része nem negatív, és nem nagyobb az első paraméterként megadott szöveg hosszánál. A függvényérték az X$ szöveg **első** N darab jeléből álló szöveg.
Ha N értéke negatív, ILLEGAL QUANTITY ERROR hibajelzést kapunk. Ha N értéke

```
10 X$="ABC"
20 PRINT
21 PRINT "HA X$=!";X$;"! AKKOR :"
22 PRINT
30 PRINT "LEFT$(X$,0)=!";LEFT$(X$,0);"!"
31 PRINT "LEFT$(X$,1)=!";LEFT$(X$,1);"!"
32 PRINT "LEFT$(X$,2)=!";LEFT$(X$,2);"!"
33 PRINT "LEFT$(X$,3)=!";LEFT$(X$,3);"!"
34 PRINT "LEFT$(X$,4)=!";LEFT$(X$,4);"!"
```

```
HA X$=!ABC! AKKOR :

LEFT$(X$,0)=!!
LEFT$(X$,1)=!A!
LEFT$(X$,2)=!AB!
LEFT$(X$,3)=!ABC!
LEFT$(X$,4)=!ABC!
```

87. ábra

nulla, akkor a függvényérték üres szöveg lesz. Ha pedig N nagyobb az X$ hosszánál, akkor a függvényérték maga az X$ lesz.

LEN(X$) Hosszúságfüggvény. Paramétere bármilyen szöveges érték lehet. A függvényérték ennek a szövegnek a hossza.

```
10 PRINT
20 INPUT "X$=";X$
30 PRINT
40 PRINT "X$=!";X$;"!",
50 PRINT "LEN(X$)=";LEN(X$)

X$=? A

X$=!A!          LEN(X$)= 1

X$=? AB

X$=!AB!         LEN(X$)= 2

X$=? A B

X$=!A B!        LEN(X$)= 3

X$=?

X$=!!           LEN(X$)= 0
```

88. ábra

Próbáljuk ki a LEN függvényt az „A", „AB", „A B", „ " paraméterekkel: a függvényérték 1, 2, 3 és 0 lesz. Ügyeljünk tehát arra, hogy a szöveg hosszába a szóközök is beleszámítanak. Az üres szöveg hossza természetesen 0.

MID$(X$,M,N) Szövegleválasztó függvény. Három paramétere van. Az első bármilyen szöveges érték lehet, a második olyan numerikus érték, amelynek egész része pozitív, és nem haladja meg az első paraméterként megadott

```
10 X$="ABC"
20 PRINT
21 PRINT "HA X$=!";X$;"! AKKOR:"
22 PRINT
30 PRINT "MID$(X$,1,1)=!";
31 PRINT  MID$(X$,1,1);"!"
32 PRINT "MID$(X$,2,1)=!";
33 PRINT  MID$(X$,2,1);"!"
34 PRINT "MID$(X$,3,1)=!";
35 PRINT  MID$(X$,3,1);"!"
36 PRINT
40 PRINT "MID$(X$,1,2)=!";
41 PRINT  MID$(X$,1,2);"!"
42 PRINT "MID$(X$,2,2)=!";
43 PRINT  MID$(X$,2,2);"!"
44 PRINT
50 PRINT "MID$(X$,1,3)=!";
51 PRINT  MID$(X$,1,3);"!"
52 PRINT
60 PRINT "MID$(X$,2,0)=!";
61 PRINT  MID$(X$,2,0);"!"
62 PRINT "MID$(X$,3,3)=!";
63 PRINT  MID$(X$,3,3);"!"
64 PRINT "MID$(X$,4,2)=!";
65 PRINT  MID$(X$,4,2);"!"
66 PRINT
70 PRINT "MID$(X$,2,-1)=!";
71 PRINT  MID$(X$,2,-1);"!"

HA X$=!ABC! AKKOR:

MID$(X$,1,1)=!A!
MID$(X$,2,1)=!B!
MID$(X$,3,1)=!C!

MID$(X$,1,2)=!AB!
MID$(X$,2,2)=!BC!

MID$(X$,1,3)=!ABC!

MID$(X$,2,0)=!!
MID$(X$,3,3)=!C!
MID$(X$,4,2)=!!

MID$(X$,2,-1)=!
?ILLEGAL QUANTITY  ERROR IN 71
```

89. ábra

szöveg hosszát, a harmadik pedig olyan numerikus érték, amelynek egész része nem negatív, és legfeljebb eggyel haladja meg X$ hosszának és M-nek a különbségét. A függvényérték az X$ szöveg M-edik jelétől kezdődő N jelből álló szöveg.
Ha az N vagy M egész része negatív, ILLEGAL QUANTITY ERROR hibajelzést kapunk. Ugyanez a helyzet, ha az M értéke meghaladja az X$ hosszát. Ha viszont az N értéke nagyobb, mint a fentebb megadott korlát, akkor a függvényérték az a szöveg lesz, amely az X$ M-edik jelétől az X$ végéig tart.

RIGHT$(X$,N)

Szövegleválasztó függvény. Két paramétere van. Az első bármilyen szöveges érték lehet, a másik olyan numerikus érték, amelynek egész része nem negatív, és nem nagyobb az első paraméterként megadott szöveg hosszánál. A függvényérték az X$ szöveg **utolsó** N darab jeléből álló szöveg.

```
10 X$="ABC"
20 PRINT
21 PRINT "HA X$=!";X$;"! AKKOR:"
22 PRINT
30 PRINT "RIGHT$(X$,0)=!";RIGHT$(X$,0);"!"
31 PRINT "RIGHT$(X$,1)=!";RIGHT$(X$,1);"!"
32 PRINT "RIGHT$(X$,2)=!";RIGHT$(X$,2);"!"
33 PRINT "RIGHT$(X$,3)=!";RIGHT$(X$,3);"!"
34 PRINT "RIGHT$(X$,4)=!";RIGHT$(X$,4);"!"

HA X$=!ABC! AKKOR:

RIGHT$(X$,0)=!!
RIGHT$(X$,1)=!C!
RIGHT$(X$,2)=!BC!
RIGHT$(X$,3)=!ABC!
RIGHT$(X$,4)=!ABC!
```

90. ábra

Ha N értéke negatív, ILLEGAL QUANTITY ERROR hibajelzést kapunk. Ha N értéke nulla, akkor a függvényérték üres szöveg lesz. Ha pedig N nagyobb az X$ hosszánál, akkor a függvényérték maga az X$ lesz.

STR$(X)

Szöveges formára alakító függvény. Paramétere bármilyen numerikus érték lehet. A függvényérték az a szöveg, amely a paraméterként megadott adat képernyőre vagy nyomtatóra írásakor jelenne meg. ☞ 91. á. Látható, hogy nem negatív számoknál az előjel helyén álló szóköz bekerül a szövegbe; a számokat a képernyőn elválasztó szóköz azonban nem.

VAL(X$)

Numerikus formára alakító függyvény. Paramétere bármilyen szöveges érték lehet. A függvényérték az a numerikus érték, amely a paraméterként megadott szöveg jeleiből képezhető. ☞ 92. ábra.

```
10 PRINT
11 X=1234567
12 PRINT " X=";X,
13 PRINT "STR$(X)=!";STR$(X);"!"
20 PRINT
21 X=-3.14159
22 PRINT " X=";X,
23 PRINT "STR$(X)=!";STR$(X);"!"
30 PRINT
31 X=1.2E-3
32 PRINT " X=";X,
33 PRINT "STR$(X)=!";STR$(X);"!"

 X= 1234567            STR$(X)=! 1234567!

X=-3.14159             STR$(X)=!-3.14159!

 X= 1.2E-03            STR$(X)=! 1.2E-03!
```

91. ábra

```
10 PRINT
11 X$="1234567"
12 PRINT " X$=!";X$;"!",
13 PRINT "VAL(X$)=";VAL(X$)
20 PRINT
21 X$="-3.14159"
22 PRINT " X$=!";X$;"!",
23 PRINT "VAL(X$)=";VAL(X$)
30 PRINT
31 X$="1.2E-3"
32 PRINT " X$=!";X$;"!",
33 PRINT "VAL(X$)=";VAL(X$)

X$=!1234567!           VAL(X$)= 1234567

X$=!-3.14159!          VAL(X$)=-3.14159

X$=!1.2E-3!            VAL(X$)= 1.2E-03
```

92. ábra

Összehasonlítva az eredményt az STR$ függvényt bemutató program eredményével, tapasztalhatjuk, hogy korrekt numerikus formájú szövegekben a VAL és az STR$ függvények egymás inverzei.

Ha a paraméter által meghatározott szövegben olyan jelek is vannak, amelyek numerikusan értelmezhetetlenek, a numerikussá alakítás csak addig folyik, amíg a gép balról jobb felé haladva el nem éri az első ilyen jelet. Ekkor a függvényérték az addig átalakított jelekből álló numerikus érték lesz.

```
10 PRINT
11 X$="12B34B"
12 PRINT " X$=!";X$;"!",
13 PRINT "VAL(X$)=";VAL(X$)
20 PRINT
21 X$="ABC123"
22 PRINT " X$=!";X$;"!",
23 PRINT "VAL(X$)=";VAL(X$)
30 PRINT
31 X$="-1.234,50"
32 PRINT " X$=!";X$;"!",
33 PRINT "VAL(X$)=";VAL(X$)
```

```
X$=!12B34B!              VAL(X$)= 12

X$=!ABC123!              VAL(X$)= 0

X$=!-1.234,50!           VAL(X$)=-1.234
```

93. ábra

Látható, hogy ha már az első jel sem értelmezhető numerikusan, akkor 0 függvényértéket kapunk.

EGYÉB FÜGGVÉNYEK

FRE(X)

A szabad tárterületet megadó függvény. Paramétere bármilyen numerikus érték lehet. A függvényérték a tár szabad bájtjainak a számát adja meg, függetlenül a paraméter értékétől.

```
10 PRINT
20 PRINT " FRE(0)=";FRE(0)
21 PRINT
30 PRINT " SZABAD TARTERULET=";
31 PRINT FRE(0)-(FRE(0)<0)*65536
32 PRINT
40 X$="----------"
41 Y$="----------"
50 PRINT X$;Y$
60 PRINT " FRE(0)=";FRE(0)
61 PRINT
70 PRINT " SZABAD TARTERULET=";
71 PRINT FRE(0)-(FRE(0)<0)*65536
```

```
FRE(0)=-26906

SZABAD TARTERULET= 38630

FRE(0)=-26920

SZABAD TARTERULET= 38616
```

94. ábra

Figyelem! A FRE függvény a bájtok számát 32 k-ig, azaz 32768-ig pozitív, attól felfelé pedig egyre csökkenő negatív értékekkel adja meg. Ezért ha a szabad terület meghaladja a 32 k mértékét, a FRE negatív függvényértékéhez hozzá kell adnunk 65536-ot. (Vagy a 65536-ból le kell vonnunk a függvényérték abszolút értékét.) A példában egyetlen képlettel adtuk meg a tárterületet, így ez a képlet bárhol felhasználható. A képletben kihasználtuk, hogy a FRE(0) $<$ 0 kifejezés (mint feltétel) értéke -1, ha igaz, és 0, ha nem.

POS(X)

A kurzorpozíciót megadó függvény. Paramétere bármilyen numerikus érték lehet. A függvényérték (a paraméter értékétől függetlenül) az a sorszám, ahányadik pozíciónál tart éppen a kiírás a képernyőn, tehát ahol az akkor folyó PRINT tart, illetve ahol a soron következő PRINT a kiírást folytatná. Minthogy a logikai értelemben vett sor a képernyő következő sorának a végéig tarthat, a függvényérték 0 és 79 közötti értéket vehet fel (a határokat is megengedve).

```
READY.

10 PRINT "<";POS(0);"-----";POS(0);">";
20 PRINT "<";POS(0);"-----";POS(0);">";
30 PRINT "<";POS(0);"-----";POS(0);">";
40 PRINT "<";POS(0);"-----";POS(0);">";
50 PRINT "<";POS(0);"-----";POS(0);">";
60 PRINT "<";POS(0);"-----";POS(0);">";
70 PRINT "<";POS(0);"-----";POS(0);">";
80 PRINT "<";POS(0);"-----";POS(0);">";

READY.
```

95. ábra

A POS függvény használatának elsősorban egy képernyősoron belül van értelme, mert ezzel dönthetjük el, hogy valamely kiírandó szöveg elfér-e még az aktuális sorban vagy sem. ☛ 96. ábra.

SPC(X)

Szóközfüggvény. Paramétere bármilyen numerikus érték lehet, amelynek egész része 0 és 255 között van (a határokat is megengedve). A függvényérték a paraméter egész részének megfelelő számú szóköz. Negatív,

```
10 PRINT
11 PRINT     ,"1","2","3"
12 PRINT "0","0","0","0"
13 PRINT "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
20 PRINT
21 PRINT "84/12/10",
22 PRINT "XV. LAJOS",
23 PRINT "KOVETEL=",
24 PRINT 5400;"FT"
30 PRINT
31 PRINT "84/12/10",
32 PRINT "XVIII. LAJOS",
33 IF POS(0)>20 THEN PRINT ,,,
34 PRINT "KOVETEL=",
35 PRINT 6500;"FT"
40 PRINT
41 PRINT "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
42 PRINT "0","1","2","3"
43 PRINT     ,"0","0","0"
```

96. ábra

illetve 255-nél nagyobb egész részű paraméter esetén ILLEGAL QUANTITY ERROR hibajelzést kapunk. A függvény csak PRINT utasításban vagy parancsban használható, a kiírt adatok közötti üres mezők méretének meghatározására.

```
10 PRINT
20 INPUT "SZOKOZOK SZAMA =";X
30 PRINT
40 PRINT "ALFA";"BETA"
50 PRINT "ALFA";SPC(X);"BETA"
99 GOTO 10
```

97. ábra

TAB(X)

Tabulátorfüggvény. Paramétere bármilyen numerikus érték lehet, amelynek egész része 0 és 255 között van (a határokat is megengedve). Negatív, illetve 255-nél nagyobb egész részű paraméter esetén ILLEGAL QUANTITY ERROR hibajelzést kapunk. A függvény csak PRINT utasításban vagy parancsban és csak képernyőn használható. (Nyomtatón használva ugyanazt eredményezi, mint az SPC.) A függvényérték meghatározza az aktuális képernyősor elejétől számítva azt a pozíciót, ahol a soron következő tétel kiírásának kezdődnie kell. ☛ 98. ábra.
A TAB és az SPC közti különbség tehát az, hogy az SPC a legutóbb kiírt tétel végétől méri a szóközöket, a TAB pedig az aktuális képernyősor elejétől.
Ha a TAB értéke kisebb, mint a POS értéke lenne, azaz ha a függvény által előírt kiírási pozíció a kurzor aktuális helyétől balra van, akkor a gép nem hajtja végre a tabulálást, hanem úgy viselkedik, mintha SPC(0)-t írtunk volna.

```
10 PRINT
20 PRINT "ABCDEFGHI";"***"
30 PRINT
40 PRINT "ABCDEFGHI";TAB(12);"***"
50 PRINT
60 PRINT "ABCDEFGHI";TAB(4);"***"
```

98. ábra

USR(X)

Felhasználó által írt gépi kódú alprogramot aktivizáló függvény. Paramétere bármilyen numerikus érték lehet. Végrehajtásakor a gép átadja a vezérlést arra a címre, amelyet a tár 785. és 786. számú rekesze együttesen határoz meg. (Az elsőben a cím alsó, a másodikban a cím felső bájtjának kell lennie. A programozó dolga, hogy az USR hívása előtt ezt beállítsa, valamint az is, hogy a megadott címre megírja és betöltse a hívni kívánt gépi kódú alprogramot.)
Híváskor a paraméter átadódik a hívott gépi kódú alprogramnak. A függvényérték az alprogram által előállított érték lesz.

SAJÁT FÜGGVÉNYEK

A programozó nemcsak a Commodore BASIC beépített függvényeit használhatja, hanem maga is létrehozhat függvényeket. Ezeknek azonban mind a nevét, mind a paraméterét, mind a függvényérték előállítási módját a programozónak kell definiálnia.
A programozó saját függvénye a következő utasítással definiálható:

DEF

DEF FN*f*(X)=*kifejezés*

Az utasításban a DEF kulcsszó áll elöl, amely meghatározza, hogy függvénydefiniálásról van szó.
Ezt követi a függvény neve. Kizárólag hárombetűs függvényneveket adhatunk meg, amelyeknek első két betűje csak FN lehet. A programozó tehát egy programon belül FNA-tól FNZ-ig összesen legfeljebb 26 saját függvényt definiálhat.
A függvény neve után zárójelben a paraméterét kell feltüntetni. Ez csak numerikus változó lehet. (Nem lehet sem konstans, sem kifejezés!) A saját függvények csak egyetlen paramétert használhatnak, sem többet, sem kevesebbet, azaz paraméter nélküliek sem lehetnek. Paramétert tehát akkor is definiálnunk kell, ha arra a függvényértéket definiáló algoritmusnak nincs szüksége.
Végül az egyenlőségjel után meg kell adnunk a függvényértéket meghatározó algoritmust, kifejezés formájában. Erre a célra bármilyen numerikus eredményű kifejezést, képletet használhatunk. (Szöveges adatok a képlet belsejében szerepelhetnek, de a kifejezés eredményének numerikusnak kell lennie.) Ez a kifejezés tartalmazhat konstansokat, változókat, beépített függvényeket, sőt korábban definiált saját függvényeket is. Bármilyen változók szerepelhetnek benne a paraméteren kívül is, sőt nem is szükséges, hogy a paraméter előforduljon benne.
A következő példában egy FNR nevű saját függvényt definiáltunk. Paramétere a FOK nevű változó. A függvényérték a paraméterként szögfokban megadott szög radiánban, azaz ívmértékben kifejezett értéke lesz.

A példa azt is szemlélteti, hogy olyankor érdemes valamilyen saját függvényt definiálnunk, ha valamiféle, egyetlen képlettel megadható értékre a programban sokszor van szükségünk.

Megjegyzés. A híváskor nem kötelező paraméterként ugyanazt a változónevet használnunk, mint a függvény definiálásakor. Az 50-es sor végrehajtásakor a gép tudomásul veszi, hogy a függvénydefinícióban szereplő FOK-nak most az X felel meg. Egyazon programban ugyanazt a függvényt akárhányszor használhatjuk különböző változónevekkel is.

```
10 DEF FNR(FOK)=FOK*π/180
20 PRINT
30 INPUT " FOK=";FOK
40 PRINT
50 PRINT FOK;" =";FNR(FOK);"RADIAN"

FOK=? 20

 20  = .34906585 RADIAN

FOK=? 40

 40  = .698131701 RADIAN

FOK=? 60

 60  = 1.04719755 RADIAN

FOK=? 80

 80  = 1.3962634 RADIAN
```

99. ábra

```
10 DEF FNR(FOK)=FOK*π/180
20 PRINT
30 INPUT " FOK=";X
40 PRINT
50 PRINT X;" =";FNR(X);"RADIAN"
```

100. ábra

Ha a telhasználáskor hibásan írjuk a saját függvény nevét, vagy elfelejtettük a definíciót a programba beírni, UNDEF'D FUNCTION ERROR (definiálatlan függvény) hibajelzést kapunk. Ha a saját függvényünkhöz nem írtunk paramétert, vagy ha az nem változó, vagy ha a függvénynév harmadik jele nem betű, akkor SYNTAX ERROR (helyesírási hiba) hibaüzenetet ír ki a gép. Ha szöveges paramétert definiálunk vagy a helyesen definiált függvényre szöveges paraméterrel hivatkozunk, TYPE MISMATCH ERROR (meg nem engedett típus) hibaüzenetet kapunk. Noha a saját függvény értékét meghatározó képletben szerepelhetnek saját függvények, maga azonban nem szerepelhet benne. Ha

mégis szerepel, a végrehajtáskor OUT OF MEMORY ERROR (a tár megtelt) hibaüzenetet kapunk.
Az érdekesség kedvéért tekintsünk meg egy másik példát:

```
10 DEF FNF(BU)=INT(BU*100/(KJ-KE)+0.5)
20 PRINT "■"
30 INPUT " KM-ORA ELOZO ALLASA=";KE
31 PRINT
32 INPUT " KM-ORA JELEN ALLASA=";KJ
33 PRINT
34 INPUT " BETOLTOTT UZEMANYAG=";BU
40 PRINT
41 PRINT " ═══════════════════════════"
42 PRINT
50 PRINT " FOGYASZTAS =";
51 PRINT FNF(BU);"L/100KM"
60 PRINT
```

101. ábra

A függvény neve FNF. Paramétere a betöltött üzemanyag, azaz BU. A függvényérték a gépkocsi üzemanyag-fogyasztása 100 kilométerre, literre kerekítve.
A program bekéri a kilométeróra-állásokat, valamint az üzemanyag mennyiségét tele tankolás esetén, és megadja a fogyasztást. Jól látható, hogy a függvényértéket megadó kifejezés nemcsak a paramétert használja.
Javasoljuk, az olvasó maga is próbálja ki, hogy ha egy számhoz 0,5-et adunk, és az összeg egész részét vesszük, valóban mindig a szám kerekített értékét kapjuk.
Emlékeztetőül megjegyezzük, hogy a 20-as sorban levő PRINT utáni jel a SHIFT és a CLR gomb együttes megnyomásakor keletkezik, és a hatására a PRINT végrehajtásakor a képernyő törlődik.
Érdemes a programot valósághű és téves adatokkal is kipróbálni. (Példák hibás adatokra: negatív üzemanyag-mennyiség, negatív vagy nulla megtett út stb.) ☛ 102. ábra.
Az ilyen hibák elleni védekezés egyetlen kifejezésbe ritkán építhető be. Esetünkben a program például nyugodtan kiszámolja, hogy ha hátrafelé megyünk, akkor az autónk benzint termel. A hiba kiküszöbölését a függvényhívás előtt kell a megfelelő vizsgálatokkal megoldanunk.
A saját függvények használata tehát egyszerű, ha már definiáltuk őket: ugyanúgy használhatók, mint a BASIC beépített függvényei. Lényeges különbség viszont, hogy a beépített függvények bármelyik programban használhatók, sőt programon kívül, parancsban is. A saját függvény azonban csak abban a programban használható, amelyben a definíciója szerepel. Programon kívül pedig egyáltalán nem használhatjuk őket; a DEF utasítás parancsként nem adható ki.

```
KM-ORA ELOZO ALLASA= 65100
KM-ORA JELEN ALLASA= 65480
BETOLTOTT UZEMANYAG= 40
==========

FOGYASZTAS = 11 L/100KM


KM-ORA ELOZO ALLASA= 72300
KM-ORA JELEN ALLASA= 72620
BETOLTOTT UZEMANYAG=-42
==========

FOGYASZTAS =-13 L/100KM

KM-ORA ELOZO ALLASA= 35600
KM-ORA JELEN ALLAS A= 35250
BETOLTOTT UZEMANYAG= 30
==========

FOGYASZTAS =-9 L/100KM

 KM-ORA ELOZO ALLASA= 43030
 KM-ORA JELEN ALLASA= 43000
 BETOLTOTT UZEMANYAG= 28
 ==========

 FOGYASZTAS =

?DIVISION BY ZERO  ERROR IN 51
```

102. ábra

Ismétlődő adatok (tömbök) feldolgozása

Olvassuk be húsz ember személyi számát, és írjuk ki, hogy hány tízéves és tíz éven aluli van közöttük, hányan vannak tíz és tizenkilenc, hányan húsz és huszonkilenc év közöttiek és így tovább!

Ha egy programozó kezébe ilyen feladat kerül, először is további kérdéseket kell feltennie:

– A személyi szám csak a születési évet tartalmazza, nem az életkort. Az életkor kiszámításához tehát tudni kell, hogy most melyik év van. Ezt a gép magától nem tudja. Ha viszont a programba beírjuk, akkor a program csak az idén fog működni, illetve minden évben át kell írni. Nem volna-e jobb a kezelőtől minden futtatáskor külön adatként megkérdezni a folyó évszámot? Ez azzal az előnnyel is járna, hogy a program nemcsak azt tudná megállapítani, hogy most milyen a kor szerinti megoszlás, hanem azt is, hogy milyen volt vagy lesz egy tetszőleges másik évben – csak azt kellene „folyó évként" begépelni;

– Ha a folyó év is bemenő adat, akkor a program csak ebben az évszázadban működjön? Ez azért lényeges kérdés, mert tudjuk, hogy
  - az e században született nők, illetve férfiak személyi számának első jegye 2, illetve 1,
  - a múlt században születetteké 4, illetve 3,

  de nem tudjuk, mi lesz a jövő században születőké. Arra a kérdésre tehát, hogy tervezzük-e a programunkat arra is, hogy a jövő században születőket is feldolgozhassa, a jövő században is működjön, nemmel célszerű válaszolni, mégpedig részben a fenti bizonytalanság miatt, részben pedig azért, mert a századfordulóig a könyv megírásakor még másfél évtized van hátra, s a mi példaprogramunknál komolyabb programoktól sem várhatnánk ilyen hosszú élettartamot;

– Felkészüljünk-e speciális személyi számú emberekre (Magyarországon tartózkodó külföldiek stb.), akiknek a személyi száma nem az 1–2–3–4 számmal kezdődik? Erre is nemmel válaszolunk, mert ez nem gyakorlati, hanem bemutató példa; az 1–2–3–4-gyel kezdődő személyi számok válogatása bőven elég lesz a bemutatandó programozási lehetőségek szemléltetéséhez, és ahhoz is, hogy az olvasó – ha a programot éppen ilyesmire kívánja ténylegesen alkalmazni – beépítse a most kihagyott kiegészítést;

– Mit jelent a feladat szövegében az „és így tovább"? Nyolc korcsoportot az átlagos várható élettartamig? Tízet, hogy a kilencvenkilenc éveseket is számba vehessük? Tizen-

kettőt, hogy a nótabeli száztizenhat éves barna kislány is beleférjen? Ez az utóbbi látszik a legcélszerűbbnek: válasszunk a programunknak olyan érvényességi tartományt, amelyet gyakorlatilag sohasem léphet túl. Lehet ugyanis, hogy több millió felhasználás közül csak egyetlenegyszer volna szükség arra, hogy a száztíz–száztizenkilenc évesek korcsoportjával is foglalkozzunk, de a program ekkor sem volna képes önmagát kijavítani; leállna, hibát jelezne, ha nem gondolnánk mi magunk korábban erre az igen ritka, de azért nem lehetetlen esetre. A jó programírás egyik titka éppen az, hogy a lehető legkevésbé valószínű eseteket is éppen olyan komolyan kell számba venni, mint a várható leggyakoribbakat. A programhibák döntő hányada sohasem a valószínű, gyakori, tömeges esetek feldolgozásakor következik be.

Eldöntöttük tehát a négy kérdést: a programunk bármelyik évben működjön; de csak a folyó évszázadban; csak az 1–2–3–4 számmal kezdődő személyi számokkal foglalkozzon; 12 korcsoport legyen, a nulla–kilenc évesek csoportjától a száztíz–száztizenkilenc évesekéig.
Adatokat számlálni már tudunk: nullázunk egy számlálót (egy erre a célra kitalált változót), és alkalmanként hozzáadunk az értékéhez egyet-egyet.

Vajon most hány változóra lesz szükségünk? Az egyes korcsoportokba tartozók számlálásakor korcsoportonként egyre, vagyis összesen tizenkettőre? Vagy minden egyes személyi szám tárolásához egyre-egyre, azaz összesen húszra? Esetleg személyi számonként és korcsoportonként külön számlálóra, tizenkétszer húszra, vagyis kétszáznegyvenre?
Amikor egy személyi szám feldolgoztunk, vagyis beolvastuk, megállapítottuk belőle az életkort, és megnöveltük a megfelelő számláló értékét, akkor nincs rá tovább szükség ebben a feladatban! Nyugodtan elfelejthetjük, és a helyére, ugyanabba a változóba olvashatjuk be a következő személyi számot.
Ha azonban egy személy például az ötödik korcsoportba tartozik, akkor ahhoz, hogy tudjuk, hány fő tartozik most már vele együtt ebbe a csoportba, tudnunk kell, hogy eddig hányan tartoztak ide. Vagyis tudnunk kell minden egyes korcsoportról az odatartozók számát végig, a program végrehajtásának teljes folyamata alatt. Ehhez tizenkét változó kellene, és – eddigi eszközeink felhasználásával – kb. ugyanannyi feltételvizsgálat.
A logikánk azt diktálja, hogy olyan módszert kellene használnunk, amely lehetővé teszi, hogy a korcsoportok számlálóira ne külön-külön adatnevekkel, hanem a sorszámaikkal utalhassunk: ötödik korcsoport, második korcsoport és így tovább.
Programunk a 103. ábrán látható.
A 10-es sor felteszi a kérdést, kiírja a kérdőjelet, majd beolvassa a folyó évszámot az EV nevű változóba.

DEKLARÁLÁS
DIM

A 20-as sort deklarációs utasításnak hívjuk. Ez közli a géppel, hogy most nem *egy* CSOP nevű adata lesz, hanem *12.* Ezeket a program további részében (tehát a programnak a deklarációs utasítás *végrehajtása* után következő részében) a CSOP adatnévvel és sorszámmal jelölhetjük: CSOP(0), CSOP(1), CSOP(2) és így tovább, CSOP(11)-ig. (A tizenkét elem tehát nem az elsőtől a tizenkettedikig tart, hanem a nulladiktól a tizenegyedikig.)
A 20-as sor végrehajtásakor tehát a gép lefoglal 12 adatnyi helyet a tárban a CSOP adatsorozat, adattömb számára, és ezt meg is jegyzi.

```
10 INPUT "MELYIK EVRE SZAMOLJAK";EV
20 DIM CSOP(11)
110 FO=1
120 PRINT "A";
140 PRINT FO;". FO SZEMELYI SZAMA =";
150 INPUT SZ$
160 SZ=VAL(MID$(SZ$,2,2))+1900
170 IF LEFT$(SZ$,1)>"2" THEN SZ=SZ-100
180 EK=EV-SZ : PRINT , EK;"EVES."
190 CSOP(EK/10)=CSOP(EK/10)+1
200 FO=FO+1 : IF FO<=20 THEN GOTO 120
210 PRINT "D" : PRINT ,"KORCSOPORTOK:"
220 PRINT
230 I=0
240 PRINT I;"-";I+9;"EV:";TAB(25);CSOP(I/10);"FO."
250 I=I+10 : IF I<=110 THEN GOTO 240
```

103. ábra

A 120—150-es sorok hatása csupán egy újabb kérdés megjelenítése és a hozzá tartozó adat beolvasása:

A 1 . FO SZEMELYI SZAMA =?
A 2 . FO SZEMELYI SZAMA =?

Ilyen kérdések jelennek meg, attól függően, hogy a FO értéke éppen mennyi.
Most kezdetben a FO értéke 1, mert a 110-es sorban így állítottuk be. Kiíródik a kérdés, a kezelő beírhatja a választ (szóközök nélkül), és a vezérlés a 160-as sorra kerül.
A matematikai kifejezések kiértékelését a gép a legbelső zárójellel kezdi. Legbelül ez áll:

MID$(SZ$,2,2)

ahol SZ$ tartalma a személyi szám. A függvényérték tehát a személyi szám 2. és 3. jele: a második jeltől kezdődő kételemű szövegrészlet. Tudjuk, hogy ez a születési év utolsó két számjegye.
Ennek a számértékét úgy kapjuk meg, hogy a VAL függvénnyel számmá alakítjuk: az egész fenti kifejezést beírjuk a VAL utáni zárójelbe.
Megvan tehát a születési év utolsó két jegyéből álló szám. Hogyan kapjuk meg ebből a születési évet? Ha valaki például 1928-ban született, eddig azt kaptuk, hogy 28. Hogyan lesz ebből 1928? Nyilván úgy, hogy hozzáadunk 1900-at. Így alakul ki a 160-as sorban SZ értéke. Ismét felhívjuk a figyelmet arra, hogy SZ és SZ$ között semmi kapcsolat nincs, ez két különböző adat.
Eddig rendben vagyunk — kivéve a múlt században születettek esetét. Ha valaki 1897-ben született, személyi száma második-harmadik jegye 97, ehhez adtunk 1900-at, így a születési év szerintünk 1997. A 170-es sor ezt a hibát korrigálja: ha az illető a múlt században született (amit onnan tudhatunk, hogy személyi számának első jegye 3 vagy 4), akkor az eddig kiszámolt születési év százzal több, mint kellene; ezt a százat le kell vonni belőle.

Mivel a LEFT$ függvény szöveget ad eredményül, összehasonlítani is csak szöveggel lehet. A

LEFT$(SZ$,1) > "2"

vagy a

VAL(LEFT$(SZ$,1)) > 2

feltétel egyaránt jó, mert számot számmal vagy szöveget szöveggel összehasonlíthatunk, de keveredve nem.

A 180-as sor számolja ki az életkort a folyó évből és az előbb meghatározott születési évből. Utána informálja a felhasználót, kezelőt, hogy milyen eredményre jutott. Ez nem feltétlenül tartozik a feladathoz, de jó ellenőrzési lehetőség: a kezelő itt esetleg rájöhet, hogy hibát követett el.

A 190-es sor végzi a számlálást. Voltaképpen 12 számlálónk van: minden korcsoportnak egy. Most meg kell növelni eggyel valamelyiknek a tartalmát, mégpedig azét, amelyikbe az a személy tartozik, akinek az életkorát éppen kiszámoltuk.

Ha az illető 15 éves, akkor az egyes, ha 115 éves, akkor a 11-es, ha 5 éves, akkor ugyanannak a CSOP nevű adatcsoportnak a nullás sorszámú elemét kell megnövelnünk.

Észrevehetjük, hogy ebben a speciális esetben a növelendő elem sorszáma éppen úgy keletkezik, hogy az életkor utolsó számjegyét levágjuk. Ezt bonyolult szövegfüggvények helyett a következőképpen is megtehetjük: osztjuk a számot tízzel (így 15-ből 1,5, 115-ből 11,5, 5-ből 0,5 lesz), és vesszük az így kialakult szám egész részét (azaz 1,5-ből az 1-et, 11,5-ből a 11-et, 0,5-ből a 0-t). Ez most éppen a kívánt sorszám!

INDEX, INDEXELÉS

Ha a sorszámként nem egész számokat használunk, a gép automatikusan elvégzi az egész rész képzését. Így bőven elég, ha ennyit mondunk: CSOP(EK/10). Ez azt jelenti, hogy osszuk el az EK életkort tízzel. Az így kapott hányados az a sorszám, ahányadik elemét a CSOP adattömbnek meg kell növelni. Vagyis annak a tömbelemnek kell az eddigi értékéhez egyet hozzáadni. (Ha pedig EK/10 nem egész, akkor az egész része jelenti a sorszámot, a növelni való elem sorszámát.) Ezt a zárójelben levő sorszámot (konstanst, változót vagy kifejezést) nevezzük indexnek, illetve indexkifejezésnek.

Meddig folyjon ez az egész eljárás? Amíg mind a 20 személy adatát fel nem dolgoztuk. Eddig az elsőnél tartunk.

INDEX-LÉPTETÉS

Növeljük meg tehát a FO értékét (200-as sor). Most 2 lesz. Meg kell még ismételni az eljárást? Ha a FO ezzel még nem lépte túl a 20-at, akkor igen, egyébként nem. Vagyis: ha a FO nem lépte még túl a 20-at, akkor kezdjük újra az egészet a 120-as sortól, az új személy személyi számának a megkérdezésétől. (A 140-es sor persze ekkor már az újabb sorszámot fogja kiírni.)

INDEXHATÁR

A gép persze hibát jelezne, ha az adattömb huszonegyedik elemét akarnánk felhasználni, hiszen a tömböt úgy deklaráltuk, hogy a felső indexhatár 20.

Ha az eljárás végrehajtása a következő személyi számra is megtörtént, a FO értéke megint

növekszik, és így tovább, míg el nem érünk a huszadik személyig. Az ő személyi számát is beolvassuk és feldolgozzuk, majd megnöveljük a FO értékét, amely így 21 lesz. A 200-as sorban megkérdezzük, kisebb vagy egyenlő-e még, mint 20. Ez a feltétel most már nem teljesül, tehát a GOTO végrehajtása nem következik be, vagyis a programvégrehajtás a soron következő, azaz a 210-es sorral folytatódik.
A 210-es sor törli a képernyőt, majd kiírja az eredménytáblázat címét. A 220-as sor egy üres sort ír a képernyőre, hogy ne folyjon össze annyira a cím a táblázattal.
A 230–250-es sorok összetartoznak. A 230-as beállítja az I nevű, eddig nem használt változó értékét 0-ra. A 240-es sor hatására most ez jelenik meg a képernyőn:

0–9 EV. . . . FO

ahol a . . . helyén az a szám áll, ahány 0 és 9 év közötti személy volt a csoportban. A 0 azért jelenik meg, mert ez az I értéke. A 9 azért, mert ez az I+9 értéke. A létszám azért, mert ez a CSOP(I/10) értéke, amint ezt korábban megbeszéltük.
A 250-es sor (a 200-ashoz hasonlóan) továbbszámlál, mégpedig ezúttal tízesével, és visszaküldi a vezérlést mindaddig, amíg I túl nem lépi az utolsó korcsoport alsó évszámát, a 110-et. Az utolsó kiírt sor tehát a 110–119 évesek sora lesz, ahogy akartuk.

FOR
TO
NEXT

A 110-es sort persze kicserélhetjük a szabványos tevékenységszerkezetek tárgyalásakor megismert új utasításra:

110 FOR FO=1 TO 20

és a 200-as sort a

200 NEXT FO

sorra, továbbá a 230-as sort a

230 FOR I=0 TO 110 STEP 10

STEP

sorra, valamint a 250-est a

250 NEXT I

sorra; programunk ugyanúgy működik, mint korábban. Az IF-es megoldást azért mutattuk be, mert így a végrehajtásnak ezek a részletei is jobban láthatók.
A tömbökkel kapcsolatban tudni kell, hogy nemcsak olyan tömbök léteznek, mint amilyenek a mintaprogramunkban voltak. A

DIM TABLA (20,14)

utasítás például olyan tömböt hoz létre, amelynek 21 sora és 15 oszlopa van, összesen tehát TABLA(0,0)-tól TABLA(20,14)-ig 315 eleme.
Háromdimenziós tömbök is létezhetnek; ezeket még könnyű is térben elképzelni. Használhatunk azonban ennél több dimenziót is, amíg csak a tár be nem telik. Így például a gép elfogadja a

DIM NAGY(1,1,1,1,1,1,1,1,1,1,1,1)

deklarációt is; más kérdés, hogy ritka lehet az a feladat, amely ilyen tömb használatát kívánja meg.

Az egyes dimenziókban is akkora lehet a kiterjedés, amekkorát csak a tár megenged. Így például a gép elfogadja a

DIM SOK(7776)

deklarációt is.
A sokdimenziós, illetve a sokelemű tömbök esetében persze a további adatok, illetve maga a program által felhasználható tárkapacitás csökken.
Szövegekből is képezhetünk tömböket, például a

DIM SZOVEGEK$(5)

deklaráció olyan tömböt hoz létre, amelynek hat eleme van, mindegyik egy-egy szöveg.
Ha a feladat nem olyan, hogy a nulladik elemnek (nulladik sornak, oszlopnak) kifejezett értelme van, inkább érdemes kihasználatlanul hagyni, és az egyes sorszámú elemekkel kezdeni a munkát, minthogy felhasználjuk a nulladikat is valamilyen sorszám-transzformációval, és aztán belezavarodjunk saját programunkba.
Attól, hogy a programunkban szerepel egy tömb, még lehet ugyanolyan nevű egyszerű változónk is. Tehát ugyanúgy, ahogy a numerikus és a szöveges változók azonos nevei nem zavarják egymást, az egyszerű változók és a tömbök nevei sem. A programunkban nyugodtan lehet egy időben X nevű és X$ nevű egyedi adat, ugyanakkor X nevű és X$ nevű tömb is. (Más kérdés, hogy célszerű-e az ilyen névazonosság.) Csak egy *tilos:* hogy legyen például egy X nevű egydimenziós és ugyanakkor egy X nevű kétdimenziós tömb is. Ha egy tömböt deklaráltunk, ugyanaz a név még egy tömbdeklarációban nem fordulhat elő. Ez annyira így van, hogy ha például egy DIM utasítást egy ciklus belsejébe írunk véletlenül, akkor az utasítás másodszori végrehajtásakor hibajelzés jelenik meg, és a programfutás megszakad. Ugyanolyan névre való deklarációt a gép akkor sem enged meg, amikor pedig nincs ellentmondás, mert mind a kétszer ugyanazt akarjuk.

Ha DIM utasítás nem volt a programban, s egy változónevet mégis úgy használunk, mintha tömbnév volna, például ha a programban megjelenik az

55 X(3)=0

utasítás, akkor a gép úgy viselkedik, mintha lett volna egy

54 DIM X(10)

utasítás is. Tehát a BASIC ilyenkor automatikusan előírja magának, hogy létezzen egy X nevű tömb összesen 11 elemmel (a nulladiktól a tizedikig). Ugyanezt több dimenzióban is megteszi: a tömbelemek ilyenkor is minden dimenzióban 0-tól 10-ig számozódnak. Ha ezek után mégis végrehajtódik egy DIM, az persze most már hibának számít.
Befejezésül két apróság, amellyel szebbé tehetjük a fejezet mintaprogramját:

130 IF FO=1 OR FO=5 THEN PRINT "Z" ;

Ez az utasítás azt eredményezi, hogy a program nem úgy kérdez, hogy mennyi A 1 . FO, A 2 . FO stb. személyi száma, hanem az 1. és az 5. fő sorszáma előtt rendesen kiírja az AZ névelőt.

140 PRINT FO;" ■. FO SZEMELYI SZAMA=,,;

Itt csak egy új jel, az osztott négyzet jelent meg a pont előtt. Hatása az, hogy a program visszalép a szám (a FO értéke) után automatikusan, letilthatatlanul megjelenő szóközre, és a pontot oda írja majd ki; a sorszámnevet jelző pont tehát közvetlenül a szám után jelenik meg és nem egy pozícióval arrébb. Az osztott négyzetet a megfelelő kurzormozgató gombnak, a bal felé mutató nyílnak az idézőjel utáni begépelésével illesztettük a szövegbe. Tapasztalatunk szerint egy-egy program felhasználói megítélésében (ha a programot nem kizárólag a saját használatunkra írtuk) sok szerepe van az ilyen apróságoknak, szépítgetéseknek. Ha a megírásra fordítottunk néhány órát, érdemes ugyanannyi percet még az ilyen részletekkel is eltölteni.

Ezzel lezárhatjuk az elemi ismereteknek azt a körét, amely megalapozza az alkalmazáshoz elengedhetetlen programozói logika megértését. Ennek elsajátítása és kellő begyakorlása lehetővé teszi a Commodore-gép „műkedvelői" szinten való használatát, egyszerű programok megírását, és alapot nyújt a magasabb szintű tudnivalók befogadására.

Irodalom

Commodore 64 Bedienungshandbuch. COMMODORE GmbH, Frankfurt.
Commodore 64 felhasználói kézikönyv. NOVOTRADE Rt.
Commodore 64 Micro Computer Handbuch. COMMODORE GmbH, Frankfurt.
Commodore Portable SX–64 Color Computer User's Guide. COMMODORE Ltd. Wayne, 1983.
Commodore 1541 Disk Drive User's Guide. COMMODORE Ltd. Westchester, 1982.
Commodore VIC–1541 floppy disk felhasználói kézikönyv.
Floppy Disk VC 1541 Bedienungshandbuch. COMMODORE GmbH, Frankfurt, 1983.
GP–100VC Benutzer Handbuch. SEIKOSHA CO. Ltd.
GP–100VC Graphic Printer Owner's Manual. SEIKOSHA CO. Ltd., Tokyo.
MPS–801 Dot Matrix Printer User's Manual. COMMODORE BM, INc., 1983.
MPS–801 Grafikdrucker Bedienungshandbuch. COMMODORE GmbH, Frankfurt, 1983.
VC 1541 Floppy Disk Bedienungs Handbuch. COMMODORE GmbH, Frankfurt.
VIC 1541 Floppy Disk User's Manual. COMMODORE Inc., 1982.
1530 Datassette Unit Operating Instructions. COMMODORE Ltd., 1983.
Dr. Bakó András: Commodore 64 mikrogép programozása. Budapest, 1983.
Bodor Tibor: Commodore 64 felhasználói segédlet. INTRONIK, Budapest, 1984.
Bodor Tibor: Commodore 64 lemezkezelés (hallgatói segédlet). INTRONIK, Budapest, 1984.
Bodor–Gerő: Aircomp–16 BASIC programozási kézikönyv. PERSONAL, Budapest, 1984.
Bodor–Gerő: Aircomp–16 BASIC referenciakártya. PERSONAL, Budapest, 1983.
Bodor–Gerő: A BASIC programozás technikája. SZÁMALK, Budapest, 1983.
Bodor–Gerő: Bevezetés a korszerű FORTRAN-programozásba I–II. SZÁMALK, Budapest, 1983.
Bodor–Gerő: Segédkönyv a „MIKROGÉPEK" című oktatófilm-, videokazetta- és diafilm-sorozathoz. TII–OOK–DIA–MAFILM, Budapest, 1985.
Gerő Péter: Commodore 64 BASIC (hallgatói segédlet). INTRONIK, Budapest, 1984.
Halász Árpád: Alapismeretek a Commodore–64 mikroszámítógép használatához.
Lángos István: A Commodore 64 mikrogép kezelése és programozása. COMPORGAN, Budapest.
Dr. Mérey András: Adatszerkezetek. SZÁMOK, Budapest, 1979.
Dr. Ury László: Commodore 64 BASIC felhasználói kézikönyv I–II. LSI ATSZ, Budapest, 1984.
Dr. Ury László: Commodore 64 információs kártya. LSI ATSZ, Budapest.

Az angol szójegyzék és a kislexikon meghatározásait a The University English Dictionary (University Books, London), a The Illustrated Heritage Dictionary and Information Book (Houghton Mifflin Company, Boston) és a Computer Dictionary (Howard W. Sams and Co., Inc., Indianapolis) alapján állítottuk össze.

Kislexikon

A könyvben használt fontosabb alapfogalmak kiemelésével azoknak akarunk segíteni, akiknek ezek a fogalmak még nem egészen maguktól értetődőek.
A kislexikonban felsorolt fogalmak értelmezése, magyarázata a tárgymutató által meghatározott helyeken természetesen a könyv szövegrészében is megtalálható, ezért itt csak a tömör definíciójukat adjuk meg.

adatállomány

Általános értelemben: azonos rendeltetésű, formájú és tartalmú információk összessége. Specifikusan: papíron, lemezen vagy kazettán meghatározott rendben és formában tárolt adatok halmaza.

adatbevitel

Információk, adatok betöltése egy külső adattároló eszközről, például billentyűzetről, lemezes vagy kazettás egységről a központi egység tárjába.
Szokásos szinonimák: begépelés, beolvasás, olvasás, betöltés, adatbekérés, input.

adatblokk

Egymást követő információk, adatok olyan sorozata, csoportja, amely adatátviteli szempontból egyetlen megbonthatatlan egységet képez.
A Commodore szóhasználata blokknak nevezi a lemez egy adott sávjának egy adott szektora által elfoglalt lemezterületet és az ezen elhelyezkedő adatmennyiséget is.

adatcsatorna

Olyan elektronikus berendezés, amely lehetővé teszi, azaz vezérli és végrehajtja az adatátvitelt a számítógépes rendszer két vagy több eleme, például a központi egység és a lemezegység vagy a központi egység és a nyomtató stb. között.

adathordozó

Olyan anyag, melyre adatok rögzíthetők (pl. papír, lemez, mágnesszalag stb.).

adatkiírás

Információk, adatok kivitele a központi egység tárjából egy külső adattároló eszközre, például képernyőre, lemezre, papírra, kazettára.
Szokásos szinonimák: kiírás, írás, nyomtatás, adatkivitel, megjelenítés, output.

adattároló eszköz

Olyan berendezés, amely képes egy meghatározott adathordozóra adatokat felvinni és/vagy onnan adatokat lehozni (pl. nyomtató, magnetofon, lemezegység).

alprogram

Olyan utasítássorozat, amely önálló funkciót lát el, és amely a végrehajtás során a vezérlést külön utasítás hatására veszi át, illetve adja vissza.
Szokásos szinonimák: szubrutin, alárendelt eljárás.

alternatív szerkezet

Olyan programelem, amely két egymást kizáró funkciót lát el úgy, hogy ezek közül az egyik egy adott feltétel bekövetkezése, a másik pedig ugyanezen feltétel be nem következése esetén hajtódik végre.
Szokásos szinonima: vagylagos szerkezet.

ASCII

Amerikai szabványos kód, amelyet eredetileg numerikusan kódolt információk (szöveg) telexgépen való továbbítására hoztak létre, de később a számítógépes információtárolásban is alkalmazták.

beépített függvény

Olyan függvény, amely szerepel a gép gyárilag elkészített függvénykészletében, és így bármely programban külön definiálás nélkül bárhol használható.
Szokásos szinonimák: gyári függvény, alapfüggvény, belső függvény.

ciklus

egy ciklusmagból és az ennek ismétlését megszervező utasításokból álló utasítássorozat.

ciklusmag

Olyan utasítássorozat, amelynek végrehajtása egy feltételtől függően egymás után többször is megismétlődhet.

ciklusváltozó

Olyan (általában numerikus) változó, amelynek bizonyos értékei meghatározzák egy léptetéses ciklus végrehajtását.

egységszám

A számítógép külső egységéhez, például nyomtatóhoz, lemezegységhez, képernyőhöz, kazettás egységhez, billentyűzethez stb. gyárilag beállított vagy előre hozzárendelt egyedi szám, melynek segítségével a központi egység az adott egységeket egyértelműen azonosítani tudja.

elöltesztelő ciklus

Olyan ciklus, amelyben a ciklusmag végrehajtása előtt vizsgáljuk meg a ciklusmag ismételt végrehajtását előíró feltételt, azaz a ciklusmag előtt döntjük el, hogy azt újra végre kell-e hajtani.

esetszétválasztás

Olyan programelem, amely több egymást kizáró funkciót lát el úgy, hogy azok közül egy összefüggő feltételrendszer egy-egy részfeltételének bekövetkezésétől, illetve be nem következésétől függően legfeljebb egy hajtódik végre.

feltételes szerkezet

Olyan programelem, amely egy meghatározott funkciót lát el, ami azonban csak egy adott feltétel teljesülése esetén hajtódik végre.

hátultesztelő ciklus

Olyan ciklus, amelyben a ciklusmag végrehajtása után vizsgáljuk meg a ciklusmag ismételt végrehajtását előíró feltételt, azaz a ciklusmag után döntjük el, hogy azt újra végre kell-e hajtani.

indexléptetés

Olyan ismétlődő tevékenység, melynek során egy vagy több tömb valamely indexe egy adott kezdőértékről indulva minden egyes ismétlődés során egy meghatározott értékkel megnövelődik.

képernyősor

A képernyő egy sora, amely meghatározott számú betűhelyet tartalmaz. (A Commodore esetén ez a képernyő méretétől függetlenül mindig 40. Figyelem! A képernyősor nem azonos a programsorral, amely a Commodore-on két képernyősort is elfoglalhat.)
Szokásos szinonima: sor.

központi egység

A számítógépnek az a része, amely az utasítások tárolására, azok végrehajtására, valamint a végrehajtás vezérlésére alkalmas. A Commodore-nál ez gyakorlatilag maga a mikrogép. Minthogy a billentyűzettel egybe van építve, magát az egész dobozt is szokás központi egységnek nevezni.

kulcsszó

Olyan szó, amelynek alapján a gép egyértelműen fel tud ismerni és meg tud különböztetni egy adott utasítást vagy utasításrészt, és amely éppen emiatt soha nem hiányozhat az utasításból, soha nem használható más célra vagy más szövegösszefüggésben, mint amit az utasítás megkíván. Tulajdonképpen a kulcsszó határozza meg az utasítás vagy utasításrész funkcióját.

kurzor

Négyzet alakú villogó jel a képernyőn, amely mindig azon a pozíción áll, ahová a legközelebb megjelenítendő jel fog kiíródni. A kurzor akkor is így működik, ha a jelzése – pl. amikor a program ír a képernyőre – nem látható.

lemezegység helyreállítása

Bizonyos hibák hatására a lemezegységet vezérlő berendezés „megzavarodik", nem képes a feladatát ellátni. Meghatározott parancs hatására azonban a lemezegység alapállapotba, azaz olyan állapotba kerül, mintha most kapcsoltuk volna be.

lemez formázása

Az a folyamat, melynek során a gyári új lemezre felvitetjük azokat a jeleket, amelyekre a Commodore lemezegységének szüksége van ahhoz, hogy a lemezt kezelni tudja.

lemez tartalomjegyzéke

Magán a lemezen levő olyan nyilvántartás, amely megadja, hogy a lemezen hol, milyen terjedelemben, milyen típusú, milyen névvel azonosított állomány helyezkedik el. A tartalomjegyzéket a lemezegységbe épített lemezkezelő program automatikusan hozza létre, és szükség szerint automatikusan módosítja.

lemeztérkép

Magán a lemezen levő olyan nyilvántartás, amely megadja, hogy a lemez mely blokkjai szabadok, és melyek foglaltak. A lemeztérképet a beépített lemezkezelő program a tartalomjegyzékhez hasonlóan automatikusan kezeli.

léptetéses ciklus

Olyan ciklus, amelyben a ciklusmag ismétlését egy változó, a ciklusváltozó értékei határozzák meg úgy, hogy a ciklusmag először végrehajtódik a ciklusváltozó kezdőértékére, majd a végrehajtás után a ciklusváltozó mindig megnövelődik egy adott számértékkel, a növekményértékkel, és a ciklusmag ismételten végrehajtódik mindaddig, amíg a ciklusváltozó értéke meg nem halad egy előre megadott végértéket.

Szokásos szinonimák: ciklusváltozóval vezérelt ciklus, FOR utasítással szervezett ciklus, FOR-ciklus, FOR—NEXT ciklus.

parancscsatorna

Különleges csatorna, amely adatforgalom lebonyolítására nem használható. Ezen a csatornán keresztül küldhetők el a lemezegységnek szóló lemezkezelő és állománykezelő parancsok, illetve ezen keresztül kaphatók meg a lemezegység üzenetei.

programsor

Sorvége (RETURN) jeltől sorvége (RETURN) jelig tartó, sorszámmal kezdődő jelsorozat, amelynek hossza nem haladhatja meg a két képernyősort.

Szokásos szinonima: sor.

random állomány

A Commodore-gépek sajátos adatállománya, amelynek adatblokkjai a lemezen elszórva helyezkednek el, és amelynek szervezési módja lehetővé teszi az adatblokkok közvetlen elérését.

Kezelését könyvsorozatunk egy előkészületben levő kötete részletesen ismerteti.

saját függvény

Olyan numerikus függvény, amelynek nevét, paramétereit, valamint a függvényértéket meghatározó kifejezést a programozónak kell definiálnia. Ennek megfelelően a saját függvény csak abban a programban használható, amelyben definiálva van, ott is csak a definiálása után.

Szokásos szinonimák: programozói függvény, utasításfüggvény.

szekvenciális szerkezet

Olyan programelem, amely egy meghatározott funkciót lát el úgy, hogy az mindenképpen végrehajtódik.

Szokásos szinonimák: feltétlen szerkezet, imperatív szerkezet.

tápegység

Az a berendezés, amelytől a számítógép az energiaellátását közvetlenül kapja. A Commodore tápegysége a szükséges tápfeszültséget a hálózati áramból állítja elő.

tömb

Azonos típusú adatokból álló adatcsoport, melynek elemeire, az egyes adatokra ugyanazzal a névvel, a tömb nevével hivatkozunk, de a tömbelemeket a tömbben elfoglalt helyüket, pozíciójukat egyértelműen meghatározó indexek segítségével azonosítjuk.

Szokásos szinonimák: adattömb, egydimenziós tömb = vektor, kétdimenziós tömb = táblázat = mátrix.

véletlenszám

Valamilyen eljárással előállított olyan számérték, amely semmilyen összefüggésben nincs az ugyanezzel az eljárással előállított megelőző számértékkel.

vezérlés

Az a tevékenység, melynek során egy utasítás végrehajtása után a gép áttér egy másik utasítás végrehajtására.

vezérlésátadás

Olyan tevékenység, melynek során valamilyen utasítással közvetve vagy közvetlenül meghatározzuk, hogy valamely utasítás végrehajtása után a gép melyik másik utasítás végrehajtására térjen át.

Angol–magyar kisszótár

Azoknak, akik a számítástechnika kapcsán kívánnak megismerkedni az angol nyelvvel, kigyűjtöttük a könyvben előforduló angol szavakat, és megadtuk a kiejtésüket, a szótár szerinti jelentésüket, valamint a számítástechnikai, elsődlegesen a Commodore szerinti értelmezésüket.
A kiejtés meghatározásánál nem a hivatalos angol kiejtésmódot vettük figyelembe, hanem a hazai számítástechnikai környezetben leginkább elterjedt, megmagyarosodott szakzsargont, amely a folyamatos magyar beszédbe beépülve is könnyen kiejthető, és nem hangzik modorosan.

A

ABS (abs): az abszolútérték függvény neve
ADVANCE (advansz): halad, előrelép
— PAPER ADVANCE: papírtovábbítás
AND (end): és
— logikai konjunkció műveleti jele
ATN (a-té-en): az arkusztangens függvény neve
ASC (a-es-cé): a karakterek ASCII kódját megadó függvény neve
ASCII = American Standard Code for Information Interchange (a-es-cé-í-í): információátvitel céljára létrehozott amerikai szabványos kód

B

BAD (bed): rossz, hibás
BLK = black (blek): fekete
BLU = blue (blú): kék
BREAK (bréjk): megszakít, megszakítás
— a program futásának megszakítása akár szándékosan, akár hiba következtében

C

C = Commodore (komodor):
— a cég emblémájával jelölt vezérlőgomb, bizonyos billentyűfunkciók kiválasztására

CHR = character (karakter): jel
— a gép jelkészletének valamelyik eleme
CLOSE (klóz): becsuk, lezár
— valamely adatállomány lezárására használható utasítás
CLR = clear (klír): tisztít
— a képernyőt törlő billentyű
CMD = command (komand): parancs
— a kiírást a képernyőről a nyomtatóra átirányító parancs
COPY (kopi): másol, másolat
— adatállományok lemásolására használható parancs
COS (kosz): a koszinuszfüggvény neve
CRSR = cursor (kurzor):
— a kiírás pozícióját meghatározó jel a képernyőn
CTRL = control (kontrol): vezérel, vezérlés
— vezérlőgomb, bizonyos billentyűfunkciók kiválasztására
CYN = cyan (cián): zöldeskék, türkiz

D

DEF = definition (definísn): definíció, meghatározás
— saját függvényeket definiáló utasítás
DEL = delete (dilit): elhagy, kihagy, töröl
— egy jelnek a képernyőről való törlésére szolgáló gomb
DEVICE (divájsz): eszköz
— külső adattároló eszköz, pl. lemezegység, kazettás egység stb.
DIM = dimension (dimensn): dimenzió
— tömböt definiáló utasítás
DIRECT (dájrekt): közvetlen
— ILLEGAL DIRECT ERROR: tilos közvetlen adatbevitel, azaz az INPUT utasítás parancsként nem adható ki
DIVISION (divísn): osztás
— DIVISION BY ZERO: nullával osztás
DRIVE (drájv): meghajtó
— tulajdonképpen a lemezegység

E

EJECT (idzsekt): kivet, kidob
— a kazetta kivételére szolgáló billentyű
END (end): vége
— program végét jelző utasítás
ERROR (eror): hiba
EXISTS (egziszc): létezik
— FILE EXISTS: az állomány már szerepel a lemezen

EXP = exponential (ekszp): az e^x függvény neve

EXTRA (eksztra): valamin túli, valamit meghaladó
- fölösleges, fölösleg, többlet
- EXTRA IGNORED: a fölösleg elhagyva

F

FFWD = fast forward (faszt forvard): gyors előrehaladás
- a kazetta előrecsévélésére szolgáló gomb

FILE (fájl): állomány
- külső adathordozón, pl. lemezen vagy kazettán levő adatállomány vagy program
- FILE NOT FOUND: az állomány nem található
- FILE EXISTS: az állomány már szerepel a lemezen

FN = function (ef-en, fánksn): függvény
- a saját függvények nevének első két betűje

FOR (for): valami szerint
- egy adott ciklusváltozó adott kezdőértéke, végértéke és növekményértéke szerint léptetéses ciklust szervező utasítás kulcsszava

FOUND (fáund): talált, találva
- annak jelzése, hogy a keresett állományt a gép megtalálta a külső adathordozón, kazettán

FUNCTION (fánksn): függvény

G

GOSUB = go subroutine (gószáb, gószub): menj a megadott szubrutinra
- vezérlésátadás a megadott sorban kezdődő alprogram első utasítására

GOTO = go to (gótu): menj a megadott sorra
- vezérlésátadás a megadott sor első utasítására

GRN = green (grín): zöld

H

HOME (hóm): otthon, haza
- a kurzort a kiindulási helyére, azaz a képernyő bal felső sarkába mozgató gomb

I

IF (if): ha
- a feltételes utasítás kulcsszava

IGNORED (ignórd): figyelmen kívül hagy, mellőz, elhagy
- EXTRA IGNORED: a fölösleg elhagyva

ILLEGAL (ilígöl): nem megengedett, érvénytelen
- ILLEGAL DIRECT ERROR: tilos közvetlen adatbevitel, nem megengedett INPUT parancs

INITIALIZE (inisiölájz): kezd, kezdőértéket beállít
- a lemezegység alapállapotát helyreállító parancs

INPUT (input): bemenet, bevitel
- adatok bevitelére, beolvasására szolgáló utasítás

INST = insert (inzert): beilleszt, inzertál
— a képernyőn levő jelek közé új jel beszúrását lehetővé tevő gomb

INT = integer (intidzsör): egész
— a törtből egész számot létrehozó függvény

L

LEFT (left): bal, bal oldali
— a szöveg bal oldaláról jelsorozatot kiemelő, leválasztó függvény

LEN = length (len, lengsz): hosszúság
— a szöveg hosszát, karaktereinek számát megadó függvény

LET (let): legyen
— az értékadó utasítás nem kötelező kulcsszava

LIST (liszt): lista, listáz
— program vagy lemez-tartalomjegyzék kiírására utasító parancs

LOAD (lód): betölt
— program vagy tartalomjegyzék betöltése a tárba lemezről vagy kazettáról
— LOAD ERROR: a betöltés sikertelen volt

LOADING (lóding): betöltés
— a betöltés folyamatban van

LOCK (lok): zár, rögzít
— SHIFT LOCK: a váltógomb hatását rögzítő, illetve oldó gomb, váltózár

LOG (log): a természetes alapú logaritmusfüggvény neve

M

MEMORY (memori): tár, tároló
— OUT OF MEMORY (aut ov memori): a tár megtelt, kimerült a tárkapacitás

MID = middle (mid, midl): közép, közepe valaminek
— a szöveg közepéből jelsorozatot kiemelő, leválasztó függvény

MISMATCH (mizmecs): nem egyezik, nem illeszkedik, eltér
— TYPE MISMATCH: a műveletben részt vevő adatok típusa nem egyezik meg, vagy egy függvény paraméterének a típusa nem felel meg a függvénynek

N

NEW (nyú): új
— a tárban levő programot törlő parancs, amely ezáltal új program begépelését teszi lehetővé
— gyári új lemez formázását, illetve már formázott lemez törlését előíró kulcsszó

NEXT (nekszt): következő, soron következő
— a léptetéses ciklus végét jelző utasítás

NOT (not): nem
— üzenetekben használatos tagadószó
— a logikai negáció műveleti jele

O

OK (oké): minden rendben
— valamely művelet (általában ellenőrzés) sikerességét jelző üzenet

ON (on): valamire vonatkozóan, valamitől függően
— egy változó értékétől függően végrehajtható esetszétválasztást elrendelő utasítás kulcsszava

OPEN (ópen): nyit, nyitás, megnyitás
— adatállományokat megnyitó utasítás

OR (or): vagy
— a logikai diszjunkció műveleti jele

OVERFLOW (overfló): túlcsordulás
— OVERFLOW ERROR: valamely adat értéke meghaladta azt a mértéket, ami a tárban még ábrázolható

P

PAPER (péjpör): papír

PEEK (pík): bekukucskál, rápillant
— adatkiolvasás a tár egy meghatározott helyéről

PLAY (pléj): játék, lejátszás
— a kazettát lejátszásra elindító gomb

POKE (pók): beledöf, megpiszkál
— adatbevitel a tár egy adott helyére

POS = position (pozísn): pozíció, helyzet
— a kurzor pillanatnyi helyzetét megadó függvény neve

POWER (pauer): erő, erőforrás
— tápfeszültség
— a tápfeszültség csatlakozóját vagy kapcsolóját megjelölő felirat

PRESENT (prezent): jelen van, kéznél van
— NOT PRESENT: arra utaló üzenet, hogy valamely használni kívánt külső egység nincs a géppel összekötve, vagy nincs bekapcsolva

PRESS (presz): nyom, megnyom
— PRESS PLAY ON TAPE: nyomja meg a lejátszást elindító gombot a kazettás egységen

PRG = program (program): program
— a program jelzése a lemez tartalomjegyzékében

PRINT (print): nyomtat, nyomtatás
— adatkivitel, kiírás a tárból valamely külső adathordozóra, pl. képernyőre, papírra, lemezre, kazettára

PUR = purple (pörpl): bíbor, lila

Q

QUANTITY (kvóntiti): mennyiség
- általában paraméterként használt kifejezés, változó vagy konstans értéke
- ILLEGAL QUANTITY: értelmezési tartományon kívül eső paraméterérték

R

READY (redi): kész, készen áll
- üzemkész, készenlétben áll
- NOT READY: nem üzemkész

RECORD (rekord): rögzítés, felvétel
- a kazettát felvételre elindító gomb
- PRESS RECORD PLAY ON TAPE: nyomja meg egyidejűleg a felvételt és a lejátszást elindító gombot a kazettás egységen

RED (red): vörös

REDO (redú): megismétel, újra végrehajt
- hibás adatbevitel megismétlésére utaló üzenet
- REDO FROM START: újra gépeld be a teljes adatot

REM = remark (rem, rimark): megjegyzés

RENAME (rinéjm): új nevet ad valaminek
- lemezen levő állományok nevének megváltoztatására szolgáló parancs

RESTORE (risztór): helyreállít, megjavít
- a gép hiba előtti állapotát helyreállító gomb

RETURN (ritörn): visszatér, visszaküld
- a sorvége gomb jelzése, kocsivissza
- a sorvége gomb megnyomásával keletkező jel
- a vezérlést a hívott alprogramból, szubrutinból a hívó eljárásba visszaadó utasítás

REWIND (rivájnd): visszacsévélés
- a kazetta visszacsévélését bekapcsoló gomb

RIGHT (rájt): jobb, jobb oldali
- szöveg jobb oldaláról jelsorozatot kiemelő, leválasztó függvény

RND = random (rendom): véletlen, véletlenszerű
- a véletlenszámot előállító függvény neve

RUN (rán): fut, futás
- a program végrehajtását elindító parancs; voltaképpen: rajt!
- a kazettáról a program betöltését és elindítását előidéző gomb

RVS = reverse (rivörz): átvált, ellenkezőjére cserél
- a képernyő alapszínének és a kiírás színének felcserélésére szolgáló gomb
- RVS ON: a színcsere bekapcsolása
- RVS OFF (rivörz of): az eredeti színek visszaállítása

S

SAVE (széjv): megment, tartalékol
— a programot lemezre vagy kazettára kimentő parancs

SAVING (széjving): kimentés
— a program kimentése folyamatban van

SCRATCH (szkrecs): kivakar, levakar, töröl
— állományok lemezről való letörlésére használható parancs

SEARCHING (szörcsing): keresés, kutatás
— a betöltendő program keresése a szalagon vagy a lemezen folyamatban van

SGN = sign (szájn): jel, előjel
— az előjel- (szignum) függvény neve

SHIFT (sift): váltás
— a billentyűzetet az írógép alsó, illetve felső állásának megfelelő állapotba átállító váltógomb

SIN (szin): a szinuszfüggvény neve

SPC = space (szpéjsz): szóköz
— a kiírandó szövegben szóközöket elhelyező függvény

SQR = square root (es-kú-er, szkver rút): a négyzetgyökvonó függvény neve

STATUS (sztéjtösz, státusz): állapot
— az állomány végének elérését jelző változó

STEP (sztep): lépés
— a léptetéses ciklus lépésközét, növekményparaméterét meghatározó kulcsszó

STOP (stop, sztop): állj
— a program futását megszakító, leállító gomb
— az ugyanezt a hatást kiváltó utasítás

STR = string (es-té-er, sztring): füzér, jelsorozat
— szöveg
— a numerikus értéket szöveggé átalakító függvény

SUBSCRIPT (szábszkript): index
— valamely tömbelem tömbön belüli pozícióját meghatározó érték
— BAD SUBSCRIPT: az index értéke túllépte az indexhatárokat

SYNTAX (szinteksz): szintaxis, mondattan
— nyelvi helyesség
— SYNTAX ERROR: nyelvtani szabályokat sértő, pl. helyesírási hiba, formailag hibásan megfogalmazott, leírt parancs vagy utasítás

T

TAB (tab): tabulátor
— a képernyőre kiírandó szöveg tabulálására használható függvény

TAN (tan): a tangens függvény neve

TAPE (téjp): szalag, mágnesszalag

— kazetta, kazettás egység

THEN (den): akkor

— a feltételes utasítás kulcsszava a feltétel teljesülése esetén végrehajtandó utasítás, utasítások előtt

TO (tu): -ig

— a léptetéses ciklus végparaméterét meghatározó kulcsszó

TYPE (tájp): típus

— adattípus

— TYPE MISMATCH: összeférhetetlen típusú adatok fordultak elő egy kifejezésben vagy egy függvény paraméterében

U

UNDEF'D = undefined (ándifájnd): nem definiált, definiálatlan, meghatározatlan

— UNDEF'D FUNCTION: definiálatlan, ismeretlen függvény

USR = user (júzer): felhasználó

— a felhasználói függvény neve

V

VAL = value (veljú): érték

— numerikus érték

— a szöveget numerikus értékké átalakító függvény neve

VALIDATE (velidéjt): érvényesít, megalapoz

— a lemez tömörítését elrendelő parancs

VERIFY (verifáj): ellenőriz, bizonyít

— a tárban és a lemezen vagy kazettán levő program összehasonlítását elrendelő parancs

— VERIFY ERROR: az összehasonlított programok nem egyeznek meg

VERIFYING (verifájing): ellenőrzés, egybevetés

— a program ellenőrzése folyamatban van

W

WHT = white (vájt): fehér

Y

YEL = yellow (jeló): sárga

Tárgymutató

Tárgymutatónk a könyvben előforduló összes BASIC utasítást és kulcsszót tartalmazza. Ezenkívül szerepelnek benne az alapvető mikrogépes és programozástechnikai fogalmak is. A feltüntetett oldalszámok a címszó első előfordulási helyét adják meg, valamint azokat a helyeket, ahol az a korábbiaktól eltérő szövegösszefüggésben vagy a korábbiakhoz képest bővebb információkkal ellátva szerepel.

A

J

K

L

M

N

O

P

R

S

SZ

T

Tibor Bodor—Péter Gerő

COMMODORE PROGRAMMING IN PRACTICE

INTRODUCTION

The book is the first volume of the serial introducing to the handling, programming and use of the COMMODORE—64 computer.
It presents the usage of the keyboard, the screen, the casette unit, the disk drive, the printer on an elementary level.
In addition to this, the book introduces into the basic principles of programming, and describes the fundamental statements of the COMMODORE—64 BASIC language.
The book is basically written for beginners, who are not familiar with the COMMODORE, the BASIC, and even the programming principles as well.
All facilities covered by the book are based upon the personal experience of the authors.
Since the book presents tested and executable full programs instead of program parts as illustrations, the practical experience of the authors can be very easily reproduced, and gaining such practice, the Reader will joyfully use the COMMODORE—64.

Тибор Бодор—Петер Геро

ПРАКТИКА ПРОГРАММИРОВАНИЯ ПЕРСОНАЛЬНОЙ МИКРО—ЭВМ ТИПА С64 ФИРМЫ COMMODORE

Основные знания

Эта книга является первым томом серии, представляющей управления, программирования, применения ЭВМ С64.

В соответствии с этим она излагает на элементарном уровне управление, применение некоторых средств, так клавиатуры, наколителя на кассетной магнитной ленте, печатающего устройства.

Кроме этого введет читателя в основные принципы программирования, представляет базовый набор команд версии языка Бейсик на С64.

Книга предложена в первую очередь начинающим пользователям, которые незнакомы ни как с машиной, а так и с языком Бейсик, даже с основами программирования.

Возможности, изложенные в книге, обоснованы личными опытами авторов.

Поскольку в книге находятся не отдельные отрывки из программ а проверенные, готовые к выпольнению комплектные программы, читатели имеют возможность для приобретения тех опытов, с помощью которых может использовать С64 с радостью в своей работе.

Ára: 55,– Ft