Dr. Ferenczy Antal

C-64 START !

alapfokon a Commodore 64-ről

gyerekeknek
sztlőknek
nagysztlőknek

LSI Alkalmazástechnikai Tanácsadó Szolgálat

Dr. Ferenczy Antal

C-64 START!

Alapfokon a COMMODORE 64-ről

Ajánljuk:

- gyerekeknek
- szülőknek
- nagyszülőknek

LSI Alkalmazástechnikai Tanácsadó Szolgálat, Budapest, 1986

Írta: dr. Ferenczy Antal

Második átdolgozott kiadás

Lektorálta: Gilicze Lászlóné

TARTALOMJEGYZÉK

BEVEZETÉS

Amikor 1971. júniusában átvettük diplománkat, professzorunk - Mogyoródi József - a búcsúzás ürügyén a következő szavakat mondta: "Jó matematikus holtig tanul.". Az Olvasó plagizálást érezhet, hiszen a köztudott forma így szól: "Jó pap holtig tanul.". Általánosítsuk a jótanácsot!

AZ ÚJRA TÖREKVŐ EMBER HOLTIG TANUL.

Az LSI ATSz gondozásában kiadott könyvecskét e gondolat jegyében nyújtjuk át az Olvasónak. Azoknak, akik akár 70 évesen is vállalják az újat akarás, a tanulás fáradalmait.

Lassan tíz éve oktatom a számítástechnika rejtelmeire a Kertészeti Egyetem mérnökhallgatóit. Itt a számítástechnika, a számítógép az elmúlt tíz év alatt a nemkívánatos, felesleges teherből az egyik legjobban megbecsült témakörré nőtte ki magát. Egyetemünk egészét átfogó mozgalom indult a modern számítástechnikai módszerek és eszközök alkalmazásáért.

Élve az LSI ATSz felkínálta lehetőséggel, olyan bevezető jellegű tankönyvet nyújtunk át az Olvasónak, amely még a legpesszimistább érdeklődő gátlásait is feloldja,s nagyon hamar a számítógép (a C-64) bátor felhasználójává avatja. Félretéve az eddigi tankönyvek precíz - néha precízkedő - hangnemét, hangvételét, szinte "kézenfogva" vezetjük be az Olvasót a C-64 alapismeretekbe.

TANKÖNYVÜNK KÉT DOLOGBAN KÍVÁN ÚJAT NYÚJTANI: HANGVÉTELBEN ÉS FORMÁBAN

- A hangvétel első olvasásra szokatlan lesz, de aki a tudomány megszokott formáit keresi, az bőven talál ízlésének megfelelő tudományos precizitással megfogalmazott tankönyvet, kézikönyvet. (Nekik nem javasoljuk tankönyvünk forgatását!) Az előbb jelzett,és az alábbiakban kifejtett stílus immár öt éve sikereket hoz azokon az előadásokon, gyakorlatokon, konzultációkon, szakkörökön, SINCLAIR-klubokban, amelyeket az Egyetem mintegy 500 hallgatója részére tartottam, illetve vezettem.

- Ury Lászlóval - a Commodore 64 felhasználói kézikönyv (LSI ATSz kiadvány) szerzőjével - közösen dolgoztuk ki a nyelvleckeszerű feldolgozási formát, mely tapasztalataink szerint már ma is bevett szokása a türelmetlen, gyors sikerekre vágyó kezdőknek. Tapasztalataink szerint ugyanis a kezdő programozók - az előkészítő ismertetések, leírások helyett - a mintaprogramok begépelésével kezdik a tanulást,és az ebből levont következtetéseket kamatoztatják első programjaikban. Ezt a gondolatmenetet fogjuk mi is követni a leckék során, amikor is a mintaprogram listája után ismertetjük a várható működést, majd az új elemek részletes leírása után gyakorló feladatokat adunk fel az Olvasónak, s annak tanulságosabb részleteit (megoldásait) közöljük a lecke végén.

Amikor gondolataimat papírra vetem, tanítványaimra gondolok, akik az új, a csodálatos, a misztikus tudást várják tőlem. Tudom, akkor teszek jót Velük, ha lerombolom a misztikumot, s úgy fogalmazok, hogy az egészséges önbizalmat adjon minden tétovázónak, önbizalomhiányban szenvedőnek.

BEVEZETÉS

MINDENKI ÉRTHETI, TUDHATJA, ALKALMAZHATJA

a modern számítástechnika módszereit és eszközeit.

Immár néhány éve előadásaimat, konzultációimat a következő jótanáccsal kezdem:

"Dolgozni csak pontosan, szépen,
Ahogy a csillag megy az égen,
Úgy érdemes."
József Attila

Az Emberhez szóló szavak után engedtessék meg néhány tényszerű megállapítás!

Tankönyvünk a hazánkban leginkább elterjedt C-64 személyi számítógép megismertetését, annak BASIC-nyelvű programozását tűzte ki célul. Felmerült több személyi számítógéptípusra alkalmazható kombinált tankönyv megírásának gondolata is (ilyen a TV-BASIC tankönyve, valamint az Ismerd meg a BASIC nyelvet c. tankönyv), de ezek az összeállítás-típusok éppen a kezdők körében okoznak problémát, hiszen a hibaüzenet nem mindig a programozó hibájára, hanem az esetleges különböző BASIC-megvalósításokra utal.

Tankönyvünk az ismerkedés természetes sorrendjét követi, hiszen az új tulajdonos először azt akarja tudni, hogy hogyan kapcsolja be számítógépét, majd kész programokkal ki akarja próbálni annak lehetőségeit,s csak ezután gondol önálló alkotásra. S ebben is ajánlatos betartani a fokozatosság elvét.

TANKÖNYVÜNK KÉT LÉNYEGES RÉSZRE TAGOLÓDIK:

- az elméleti alapokat ismertető bevezető fejezetekre, amelyek még nem igényelnek állandó géphasználatot,
- a nyelvleckeszerűen feldolgozható programozási leckékre.

A tankönyvben ismertetett,és további C-64 utasítások teljes ismertetését tartalmazza Úry László kétkötetes Commodore 64 programozói segédlete, amely az LSI ATSz gondozásában mindig naprakészen áll a felhasználók rendelkezésére.

Ezúton fejezem ki köszönetemet mindazoknak, akik munkájukkal, jó tanácsaikkal elősegítették a - remélhetőleg sikeres - tankönyv megjelenését.

Budapest, 1986. február

dr. Ferenczy Antal

1. A FELHASZNÁLÓI GÉPHASZNÁLATRÓL

Ha mindennapi munkánk során valamely eszközzel dolgozni szeretnénk, azt mindig elő kell készíteni a munkára. Ha bármilyen berendezést veszünk, annak használatba vétele előtt ajánlatos a használati utasítást figyelmesen elolvasni. Ezek a megállapítások a C-64-re éppúgy érvényesek, mint egy újonnan vásárolt magnetofonra vagy mosógépre.

Ez a fejezet fogja tájékoztatni az Olvasót mindazokról a hasznos információkról, amelyek nélkül a C-64 működtetése akadozhat.

Valamennyi Olvasónknak javasoljuk a "használati utasítás" elmélyült tanulmányozását, hiszen az nagyon sok később jelentkező probléma elkerülését eredményezheti.

1.1 C-64 alapismeretek: konfiguráció, üzembehelyezés

Amikor a C-64 személyi számítógéppel dolgozni akarunk, akkor legalább két berendezést fogunk használni: magát a számítógépet és egy eredménykijelző berendezést. Ezért első lépésként olyan elosztóról kell gondoskodnunk, amely - a biztonsági előírásokat messzemenően betartva - legalább két elektromos berendezés egyidejű üzemeltetését teszi lehetővé.

Azt az üzemeltetési formát, amikor a C-64-et és - eredménykijelzőként - egy fekete-fehér vagy színes képernyős TV-készüléket alkalmazunk minimális számítógépes rendszernek, minimális konfigurációnak nevezzük. Ebből a példából kiindulva nagyon könnyen általánosíthatjuk a számítástechnikusok körében nagyon elterjedt konfiguráció fogalmát, mely a következő:

> A SZÁMÍTÓGÉPET ÉS A VELE ELEKTROMOS KAPCSOLATBAN ÁLLÓ, ADATTÁROLÁSRA VAGY ADATÁTVITELRE ALKALMAS BERENDEZÉSEK ÖSSZESSÉGÉT AKTUÁLIS **KONFIGURÁCIÓ**-NAK NEVEZZÜK.

Tankönyvünkben a következő berendezések alapszintű alkalmazását fogjuk áttekinteni:

- C-64 személyi számítógép,
- eredménykijelző TV-készülék,
- lemezegység (VIC 1541 floppy disk drive),
- mátrixnyomtató (MPS 801 mátrixprinter),
- szalagegység (C2N casette unit).

Természetesen feladattól függően más-más konfigurációra lesz szükségünk. A Commodore kínálatát jól ismerő Olvasó bizonyos berendezéseket hiányolhat. Célunk nem a teljes Commodore berendezéskészlet bemutatása, hanem az aktív számítógéphasználat elősegítése. Ezért hiányoznak a tervezett berendezések közül például a botkormány vagy a különböző kiegészítő programelemeket hordozó kártyák és bővítő egységek. Ezek célszerű leírását Úry László már a Bevezetésben említett munkájában kellő részletességgel tárgyalja.

Bár a C-64 és valamennyi vele elektromos kapcsolatban álló berendezés mérete, tömege és rendeltetése révén alkalmas a szállításra, az alkalmankénti összeállításra, mégis azt javasoljuk, hogy hosszú távra rendezkedjünk be. Tervezzünk meg egy munkaasztalt a rendelkezésünkre álló teljes berendezés-

együttes számára, s azt állítsuk össze úgy, hogy alkalmanként csak a megfelelő berendezések megfelelő sorrendű bekapcsolása legyen a feladatunk. Ezt a tanácsot nem a restség, hanem a berendezések élettartamának megnövelése mondatja velünk. Gyakori meghibásodási lehetőség, hogy az elektromos áramot vagy ezzel együtt információt továbbító kábelek meghibásodnak s ezek javítása, cseréje ma még nem tartozik az egyszerű feladatok közé.

Üzembehelyezési javaslatunkat három részre bontjuk:

- Commodore-munkaasztal tervezése,
- Commodore-munkaasztal összeállítása,
- aktuális konfiguráció üzembeállítása.

Mivel az állandó elhelyezésű módszert javasoltuk, ezért csak az utolsó lépést kell minden alkalommal végrehajtanunk, az első kettőt elegendő akkor, ha a munkaasztalt valamilyen ok miatt módosítani akarjuk.

Commodore-munkaasztal tervezése

Válasszuk ki dolgozószobánk egy nyugodalmas, minden napszakban kényelmes látási viszonyokkal rendelkező részét. (Erre azért van szükség, mert a C-64 alkalmazásához nemcsak a TV-képernyő figyelemmel kisérése szükséges.) Válasszunk olyan asztalt, amelyen kényelmesen elhelyezhető - tőlünk legtávolabb - a legmegfelelőbb látótávolságban egy TV-készülék, - a legközelebb - kényelmes billentyűkezelést biztosító helyen a C-64 személyi számítógép, továbbá könnyen elérhető távolságban a többi berendezések (lemezegység(ek), mátrixnyomtató, szalagegység, stb...). Gondosan ügyeljünk arra, hogy az összeköttetést biztosító kábelek minél kevesebb keresztezéssel, minél egyenesebb formában helyezkedjenek el az asztalon. Kerüljük az egyes berendezések szoros egymás mellé tevését. Különösen a C-64 tápegységét tegyük a többi berendezéstől minél nagyobb távolságra. Az egyes csatlakozókat minél kevesebbszer húzzuk ki, majd dugjuk be újra, mert ezek túl sűrű ismétlése korai kábelszakadáshoz vezethet. Csak akkor folyamodjunk hozzá, ha ez feltétlenül szükséges. A Commodore cég előzékenységének köszönhetően szinte valamennyi kábel mutatja, hogy hová kell őket bedugaszolni. Ha ez a mellékelt Commodore dokumentáció elolvasása után sem sikerülne, akkor forduljunk nagyobb gyakorlattal rendelkező kollégához vagy az illetékes szerviz szakemberéhez. Nagyon előnyös, ha a földelt elosztónk kapcsolóval is rendelkezik, mert ez az állandóra összeállított konfigurációt egy újabb biztonsági kapcsolóval védi a felesleges - sőt káros - áram alatt tartástól.

A Commodore-munkaasztal összeállítása

Ha már kiválasztottuk a megfelelő szobarészt és a kényelmes munkát biztosító asztalt, akkor állítsuk a helyére. Ezután helyezzük el úgy a felhasználandó TV-készüléket, hogy az számunkra megfelelő látótávolságra kerüljön. Ezután mintegy írógéphelyzetbe helyezzük magunk elé a C-64-et. Következő lépésként az antenna-csatlakozót dugaszoljuk be a C-64 hátoldalán a baloldali antenna-csatlakozóba. Ezután a TV antenna-csatlakozóját keressük meg. Ha több választási lehetőség van, akkor válasszuk azt, amely normál üzemmódban az MTV-2 vételéhez szükséges. Ha csatlakozóink mérete vagy rendszere nincs össz-

hangban, akkor ideiglenes megoldásként - amíg megfelelő állandó TV-készüléket nem sikerül beszereznünk - csináltassunk SZAKEMBERREL bemenetátalakítót. Ezután csatlakoztassuk a TV-készüléket az áramelosztóhoz, majd a C-64-et a tápegység segítségével ugyanoda. Ezt úgy tehetjük meg, hogy a hagyományos kábelt az elosztóhoz, a "négyujjú"-t pedig a C-64 jobboldalán a POWER felirat alatt levőhöz csatlakoztatjuk. Ha ezt megtettük, akkor adjunk áramot először az elosztónak, majd a TV-készüléknek, s annak bejelentkezése után az ON feliratú, a C-64 jobboldalán levő kapcsoló segítségével a C-64-nek. Ha mindent helyesen tettünk, akkor a televízió a C-64 ún. vezetékes adásának vételére kész,s a C-64 a billentyűzet jobb felső sarkában levő "varázsszem", a POWER-lámpa segítségével jelzi, hogy munkára kész. Ha korábban a televízión a MTV-2 adását néztük, akkor elképzelhető, hogy már (többé-kevésbé) olvasható a TV képernyőjén a C-64 bejelentkező szövege. Mivel ez nagyon ritkán fordul elő, ezért induljunk ki abból, hogy a TV-n még semmi sem látszik. Ekkor a kereső gomb, majd a finomkereső segítségével a MTV-2-höz közeli hullámsávra hangoljuk TV-készülékünket mindaddig, amíg az alábbi bejelentkező szöveg tisztán olvasható nem lesz:

```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
```

Ha színes televíziót alkalmazunk, akkor a képernyő széle és a felirat világoskék, a képernyő közepe (az ún. háttér) pedig sötétkék lesz.

Ha ez sikerült, akkor folytassuk tovább a Commodore munkaasztal felszerelését! Előbb fordított sorrendben (tehát C-64, televízió, elosztó) kapcsoljuk ki minimális konfigurációnkat és lássunk hozzá a bővítéséhez. Talán felesleges felhívnunk a figyelmet arra, hogy az ezután felsorolt berendezések használata, léte esetleges, tehát csak akkor hiányoznak, ha a feladat speciálisan előírja jelenlétüket, bekapcsolásukat, alkalmazásukat.

Lemezegység:

Talán a leggyakrabban használt kiegészítő berendezés, hiszen ez teszi lehetővé kész programok és adatállományok megbízható tárolását és gyors kezelését. Rendszerbeállítása a következőképpen történik:

- Helyezzük a lemezegységet könnyen elérhető, de ugyanakkor nyugodt környezetbe. (Ez alatt azt értjük, hogy legalább 20-30 cm-re legyen a többi berendezéstől - különösen a C-64 tápegységétől. Ne használjuk semmiféle eszköz (különösen nem lemezek!) tárolóhelyéül.) Világosabban fogalmazva SEMMIT ne tegyünk a tetejére!!!

- A lemezegységhez a Commodore cég két kábelt mellékelt: az egyik a hálózati, a másik az információs kábel. Mindkettő jól mutatja a helyét a rendszerben. A hálózati kábel kétvillás dugóját az elosztóba, másik végét pedig a lemezegység hátoldalának alján levő speciális alakú aljzatba kell beledugaszolni. Az információs kábel egyik végét a C-64 hátoldalán a baloldali dugaszaljba, a másik végét a lemezegység hátoldalán szintén a baloldali felső dugaszaljba kell dugnunk.

- Ha ki akarjuk próbálni ezt a konfigurációt is, akkor a következő sorrendben kapcsoljuk be a rendszert:
  - áramot adunk az elosztónak,
  - bekapcsoljuk a TV-készüléket,
  - a lemezegység hátoldalán levő ON/OFF-kapcsolót bekapcsoljuk, melynek hatására a lemezegység elején bevillan egy piros és folyamatosan ég egy zöld jelző-lámpa (ezek biztosítják a lemezegység író-olvasófejének védelmét),
  - ezután kapcsoljuk be az ON-kapcsoló segítségével a C-64-et, mely az előzőkhöz hasonlóan jelentkezik be.

(A konfiguráció érdemi, tényleges kipróbálására később kerül sor. Itt említjük meg, hogy a lemezegységre utasításainkban 8-as (ritkán 9-es) számmal fogunk hivatkozni.)

Ha valamennyi lépést helyesen hajtottuk végre, akkor a továbblépés (vagy az újrapróbálás) érdekében fordított sorrendben kapcsoljuk ki a rendszert. (Tehát C-64, lemezegység, TV-készülék, elosztó).

Szalagegység:

A Commodore cég speciális magnetofont (szalagegységet) dolgozott ki számítógépes rendszeréhez, olyannyira, hogy az szinte együtt él a C-64-el. Hogy mennyire jogos ez a hasonlat, azt az is bizonyítja, hogy külön áramforrásra nincs is szükség a szalagegység alkalmazásához, mert egyetlen,a C-64-hez csatlakozó kábellel, az ún. köldökzsinórral mind az információtovábbítás, mind az egyszerűbb vezérlési utasítások, mind az áramszolgáltatás kérdését megoldottuk. Ha tehát szalagegységgel kívánunk dolgozni, akkor nem kell mást tennünk, mint a C-64 bekapcsolása előtt a C2N casette unit berendezés speciális csatlakozóját a C-64 hátoldalán levő speciális csatlakozó sínre ráhelyezni. Ezekután az előzőekben leírt módon történik a rendszer be- és kikapcsolása, hiszen a C2N casette unit-berendezés mintegy a C-64 "belső tartozéka".

(Megemlítjük, hogy utasításainkban 1-es számmal fogunk hivatkozni rá.)

Mátrixnyomtató:

A lemezegységhez hasonlóan két csatlakozókábellel felszerelt rendszeregység. Alkalmazását hosszabb eredménytáblázatok, szerkesztett szövegek (ügyiratok, levelek, dokumentációk stb.) elkészítésénél javasoljuk. Rendszerbeállítása a lemezegységnél felsoroltak szerint történhet. Csupán a mátrixnyomtató és a C-64 egymásközti kapcsolata esetleges: ha nem alkalmazunk lemezegységet, akkor kapcsolatuk közvetlen a C-64 hátoldalán levő baloldali speciális dugaszalj segítségével; ha egy vagy több lemezegységet alkalmazunk, akkor a sor végén az utoljára alkalmazott lemezegységhez kapcsoljuk úgy, hogy annak hátoldalán jobbközépen levő dugaszaljba dugjuk a mátrixnyomtató információs kábelét. Ha a kábelek a megfelelő dugaszaljba kerültek, akkor az üzembehelyezés sorrendje most a következő:

- az áramelosztó bekapcsolása,
- a TV-készülék bekapcsolása,
- a mátrixnyomtató bekapcsolása a bal oldalán található kapcsoló segítségével. Ha ezt helyesen tettük, akkor a mátrixnyomtató bal felső sarkában kigyullad a munkaállapotot jelző zöld lámpa és a nyomtatófej néhány betűnyi mozgást végez.
- a lemezegység(ek) bekapcsolása,
- a C-64 bekapcsolása.

(Az itt javasolt bekapcsolási sorrend csak egy a sok közül. Ügyeljünk arra, hogy mindig a C-64-et kapcsoljuk be utoljára s kapcsoljuk ki legelő-

ször. A szabály be nem tartása a memóriában tárolt információ elvesztését, vagy a bekapcsolt berendezések meghibásodását vonhatja maga után.)

(Már itt megemlítjük, hogy a mátrixnyomtatóra utasításainkban 4-es (ritkán 5-ös) számmal hivatkozunk. Ezek beállítását a felhasználó is változtathatja a mátrixnyomtató hátoldalán levő 4-5-T-kapcsoló mozgatásával. A T-állapot a mátrixnyomtató tesztelésére utal. Nagyon hasznos, ha már most elsajátítjuk a festékszalagcsere és a papírbefűzés fortélyait.)

Ha mindent helyesen elvégeztünk, akkor most már bármilyen - a rendelkezésünkre álló berendezés-állományból felépíthető - számítógépes rendszert pillanatok alatt üzembe tudunk helyezni,s utána bármilyen azon elvégezhető feladathoz már csak a megfelelő program vagy programok, illetve a megfelelő adatok szükségesek.

Hogy ezt megtehessük - fordított sorrendben - kapcsoljuk ki a pillanatnyilag bekapcsolt teljes rendszert. (Tehát C-64, lemezegység(ek), mátrixnyomtató, TV-készülék). Aki a szalagegységet hiányolja a felsorolásból, az figyelmesen olvassa el a szalagegységről néhány sorral ezelőtt mondottakat!!!

(Megjegyezzük, hogy a billentyűzet fokozottabb igénybevételét előíró programoknál - ha ez lehetséges - nagyon előnyös a billentyűzet kímélése céljából a botkormány (joystick) alkalmazása.)

Aktuális konfiguráció üzembeállítása

Bár az egyes berendezések ürügyén leírtuk az akkor aktuális üzembeállítási teendőket, most mégis a nyomatékosság kedvéért összefoglaljuk azokat, hogy számítógépes rendszerünket mindig ezen univerzális szabályok figyelembevételével minél tovább sikeresen használhassuk.

BEKAPCSOLÁSI SORREND

0. Annak ellenőrzése, hogy valamennyi használni kívánt berendezés munkára kész állapotban a helyén van-e.
1. Árameloszto bekapcsolása.
2. TV-készülék bekapcsolása.
3. Mátrixnyomtató bekapcsolása (ha van).
4. Lemezegység(ek) bekapcsolása (ha van(nak)). (Ha több van, akkor a bekapcsolás szabálya a mátrixnyomtató felől a C-64 felé haladó sorrendet ír elő.)
5. C-64 bekapcsolása.
6. A bejelentkező TV-kép beállítása finombeállítóval.
7. Ha lemezegységgel dolgozunk, akkor az éppen használni kívánt lemezt csak ezután ajánljuk a lemezegységbe helyezni!!!

KIKAPCSOLÁSI SORREND

1. Vegyük ki a lemezegységből az esetleg benne levő lemezt!!!
2. Kapcsoljuk ki a C-64-et.
3. Kapcsoljuk ki a lemezegységet (ha több van, akkor a C-64-től a mátrixnyomtató felé haladva!).

4. Kapcsoljuk ki a mátrixnyomtatót.
5. Kapcsoljuk ki a TV-készüléket.
6. Kapcsoljuk ki az árameloszto áramkapcsolóját.

Felhívjuk a tisztelt Olvasó figyelmét arra, hogy az előzőekben felsorolt szabályok nem kötelező érvényűek, de ha azokat folyamatosan nem tartjuk be, akkor gyakrabban kell az egyes berendezéseket javíttatni, s egyszer-egyszer az is előfordulhat, hogy a lemezegységben felejtett lemez tartalma elvész.

1.2 Kapcsolatteremtés a C-64-gyel

Ebben a szakaszban a C-64 billentyűzetével és a különböző adatátviteli formákkal ismerkedünk meg.

Ha figyelmesen megnézzük az előttünk álló C-64-et - sokadmagunkkal együtt - jó megjelenésű, angol nyelvterületre specializált táskaírógépre tippelnénk, ha nem tudnánk, hogy egy,az írógépeknél sokkal intelligensebb eszköz van a kezünkben. Ha a bekapcsolás pillanatában vagyunk, akkor a képernyőn az előző szakaszban idézett bejelentkezési szöveg olvasható,

```
**** COMMODORE...
64K RAM...
READY.
```

amely magyar fordításban a következőket jelenti: 64 kilobyte felhasználói munkaterületemből ebben a pillanatban 38911 byte-nyi terület áll a felhasználó rendelkezésére, valamint hogy a C-64 munkára kész.

Ahol a villogó négyzet van, oda fogom írni a felhasználó írásjeleit vagy az én válaszomat.

Az első lefordított magyar mondatban szerepel a kilobyte fogalma. Ez egy memória-mértékegység, amelyet később pontosan definiálunk.

Ha a C-64 beszélni tudna - s ráadásul magyarul -, akkor röviden ezt mondaná minden bekapcsolás után. Az üzenet mélyebb szakmai jelentését később részletezzük. Egyenlőre tekintsük a négysoros üzenetet úgy, mint amikor a kezdő angolul tanuló megtanulja a "HOW DO YOU DO?" bemutatkozó kérdést. Ha elkezdenénk most az üzenet megfejtését, ugyanúgy járnánk, mint az a diák, aki szavanként szótár segítségével lefordítaná a"HOW DO YOU DO?"-t. Ezek előrebocsátása után kezdjünk hozzá a billentyűzet rendszeres, szisztematikus áttekintéséhez.

Ha ránézünk a billentyűzetre, akkor azt látjuk, hogy négy sorban nagyjából egyenletesen helyezkednek el azok a billentyűk, amelyek többsége elhelyezkedésében hasonlít egy szabványos írógéphez. Próbáljuk meg ezt az analógiát, hasonlatot továbbvinni,s mintegy írógépként használva írjuk be a "COMMODORE" szót. Ha ezt megtesszük, akkor a következőt fogjuk tapasztalni:

- Amikor leütjük a C betűt, akkor a bejelentkező szöveg READY. üzenetének R betűje alatt megjelenik a C betű, majd a villogó négyzet a C betű után és az E betű alatt fog megjelenni. (Gondoljuk végig, hogy a billentyű-

zet egyetlen C billentyűjének megnyomása megváltoztatta az eredménykijelző TV képernyőjének tartalmát!).

- Ha a további betűket is begépeljük, akkor a további betűk is szépen sorjában megjelennek a képernyőn,bizonyítva azt, hogy a géppel egyenrangúan mi is előírhatjuk a képernyő aktuális tartalmát. Ekkor a képernyőn a bejelentkezés hat sora után a 7.-ben a COMMODORE felirat,és közvetlenül mögötte a villogó négyzet szerepel.

- Ha az utolsó sorban egyedül "álló" hosszú billentyűt egyszer megnyomjuk, akkor mint az írógépeken,betű írása nélkül tovább tudunk lépni. (Ez a billentyű a szóköz vagy SPACE-billentyű!)

- Ezekután vegyük birtokunkba a számokat. Ezek is, ugyanúgy mint a normál írógépeken, a legfelső sorban helyezkednek el. Stílszerűen a 64 begépelésével próbáljuk ki őket. Ha ezt megtesszük, akkor a képernyő a következő képet fogja mutatni:

```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.

COMMODORE 64
```

Azt a villogó négyzetet, amely lépten-nyomon jelzi beavatkozásunk helyét vagy annak lehetőségét az angol szakirodalom nyomán kurzor (cursor)-nak nevezzük. Most már tetszés szerinti betűt, számot vagy szóközt (az alsó hosszú billentyű eredménye) le tudunk írni úgy, hogy tevékenységünk eredménye a TV-képernyőn látsszék. A betűk és számok helyével ismerkedendő, írjuk be családunk tagjainak teljes nevét és egy adatát (születési évszámát, magasságát, testsúlyát, keresetét stb...). Ha ezt megtesszük, akkor még a legkisebb családban is eljutunk a sor végére. Tegyük fel, hogy minden leütés után figyeljük a képernyő tartalmának változását. Ekkor a sor végén keressük az írógépen megszokott sorváltó billentyűt vagy kart. (Ezt a szerepet a később ismertetésre kerülő **SHIFT-RETURN**-billentyű-kombináció tölti be). Felhívjuk a figyelmet arra, hogy ez csak akkor azonos a sorváltó billentyű szereppel, amikor a C-64 segítségével a képernyőre (mely mintegy helyettesíti az írógéppapírt) írunk. Ha majd a C-64-et, mint igazi számítógépet használjuk, akkor más megoldást fogunk alkalmazni.

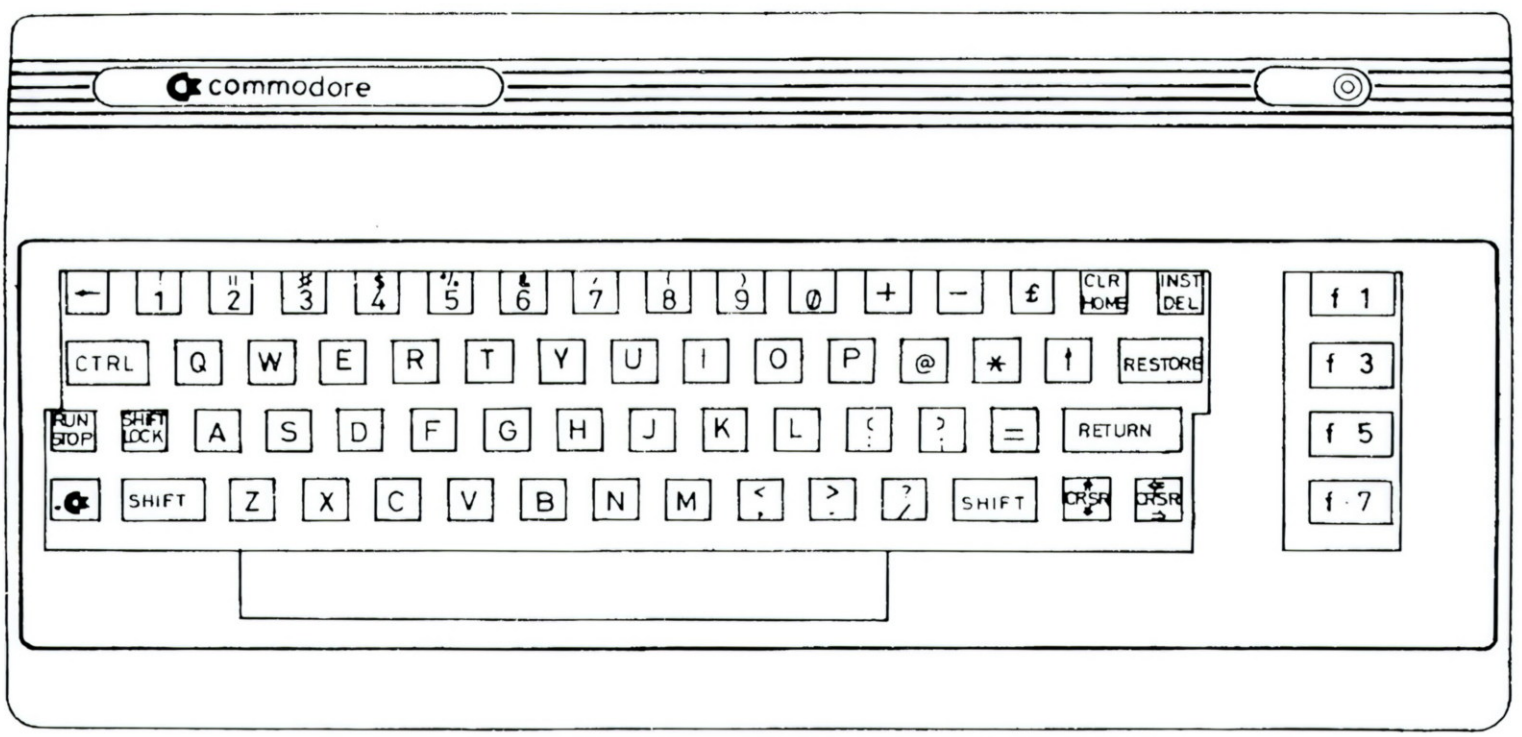
Az eset szemléltetésére tekintsük a következő képernyő-tartalmat:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ():;0123456789
```

Gépeljük be a 8-as utáni pozícióba a 9-es számjegyet. Ekkor a következő képernyőtartalmat kapjuk:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ():;0123456789
```

Ha tovább tanulmányozzuk a C-64 billentyűzetét, akkor - különösen a felső billentyűsoron - nagyon sok olyan billentyűt látunk, amelyek két írásjelet tartalmaznak a billentyű felületén. Amikor például a COMMODORE felirat után a 6-ot és 4-et tartalmazó billentyűt megnyomtuk, akkor a képernyőn a 64 jelsorozat jelent meg. Mit tegyünk, ha & vagy $ jelekre lesz szükségünk? Mit tettünk akkor, amikor írógép volt a kezünkben? Megkerestük a kisbetűről nagybetűre váltó billentyűt s azzal együtt nyomtuk meg a kívánt jelet tartalmazó billentyűt. Ezek a billentyűk az írógépen a bal és jobb alsó sarokban helyezkedtek el. Keressünk most is ilyen billentyű-párt! Szinte az írógépével megegyező helyen találjuk a két **SHIFT** feliratú billentyűt. A billentyűk elhelyezkedése (melyet a következő stílizált C-64 rajz kíván érzékeltetni) sugallja az ötletet, hogy ezek szerepe az írógépnél megszokott felsőjel-hívó szerepet tölti be. Próbaképpen egyszerre nyomjuk le a **SHIFT** és a 4 billentyűket, eredményként a képernyőn $-jel fog megjelenni. Ha a 6-ot nyomjuk meg a **SHIFT**-tel együtt (zsargonnal élve siftelve), akkor a &-jel fog megjelenni. Valamennyi felső állású jelhez ily módon hozzá tudunk férni. Egyetlen kérésünk, hogy a szöveges billentyűket még ne próbálgassuk, az "-et tartalmazó 2-est pedig csak párosával. (Így "" vagy """".) A kérés oka, hogy ezek a billentyű-kombinációk olyan hatást eredményeznek, amellyel kezdetben nehéz megbirkózni.



A C-64 sematikus (vázlatos) billentyűképe

A FELHASZNÁLÓI GÉPHASZNÁLATRÓL

A korábbiakban már említettük, hogy a villogó négyszög neve kurzor (angolul cursor). Ha az angol névből elhagyjuk a magánhangzókat,akkor a **CRSR** betűkombinációt kapjuk. Ha ennek ismeretében felfedezzük a billentyűzet negyedik sorának végén a két **CRSR**-feliratú nyilas billentyűt, akkor nagyon könnyen rájöhetünk arra, hogy ha valamelyik billentyűt egyedül megnyomjuk, akkor az alul szereplő nyíl irányába mozdul írás nélkül a kurzor, ha a **SHIFT** billentyűvel együtt, akkor pedig a felül szereplő nyíl irányába mozdul el írás nélkül a kurzor. A billentyűk begyakorlására javasoljuk, hogy kalandozzunk el a képernyőn. Érdemes a kurzormozgató billentyűk kezelésében nagy gyakorlatra szert tennünk, mert a későbbi programozási, szövegszerkesztési műveletek során előnyös helyzetbe kerül az a felhasználó, aki a képernyőn könnyedén tud helyet változtatni.

Ha már tudunk írni és mozogni a képernyőn, akkor tanuljuk meg azt is, hogy hogyan kell elindulni,s hogyan kell tiszta lapot kezdeni. Ez az első hallásra filozófikusnak,s egyszersmind lehetetlennek tűnő feladat a C-64-en mindössze egy billentyű megkeresését és megnyomását jelenti. Miközben a C-64 kezelésével ismerkedünk - akarva-akaratlanul - némi angol nyelvtudást is fel kell "szednünk". A következő két szó, amelyet meg kell tanulnunk: az otthon jelentésű HOME és a tisztítani jelentésű CLEAR. Ha ezeket figyelembevesszük, s elkezdjük őket keresni, akkor az első sor végén nagyon hamar megtalálhatjuk a **CLR-HOME** feliratú billentyűt. Ha önállóan lenyomjuk a billentyűt, akkor az alsó sorban szereplő **HOME** felirat értelmében a képernyő bal felső sarkába ugrik a kurzor anélkül, hogy a képernyő tartalma megváltozott volna. (Innen mintegy a "kályhától elindulva" bármely képernyőn szereplő írásjelhez eljuthatunk,számlálva a kurzormozgató billentyűk használatát.) Nemcsak az életben, de a programozás során is nagyon gyakran szükség van arra, hogy egy-egy eredménytáblázatot, programlistát tiszta lapon lássunk. Ezt az állapotot a **SHIFT** és a **CLR-HOME** billentyű együttes megnyomásával tudjuk elérni.

Az eddig elmondottak gyakorlására írjunk adatlapot önmagunkról a képernyőre. Ezen soroljuk fel a nevünk, lakásunk, telefonunk, munkahelyünk (iskolánk), foglalkozásunk, havi jövedelmünk, testméreteink stb. adatait,a reá vonatkozó kérdésekkel együtt.

Ha végiggondoljuk az ebben a szakaszban eddig elmondottakat, akkor arra a következtetésre juthatunk, hogy egy-egy billentyű megnyomásának hatása van a képernyő tartalmára. Ez a hatás vagy közvetlenül a megnyomott billentyű tartalmának megjelenésével, vagy közvetve a megnyomott billentyű hatására más képernyő-módosulással fog megvalósulni. Próbáljunk meg ennél összetettebb hatást elérni! A hétköznapi életben is kérdéssel fordulunk azokhoz, akik maguktól nem mondanak semmit. Tegyük ezt a C-64 esetében is. Gépeljük be az üres képernyőre a

HOGY HIVNAK?

szöveget. Ha nem teszünk semmi mást, akkor csak a kérdésünk,és utána a villogó kurzor lesz a képernyőn. Ha sorban kipróbáljuk az eddig még nem használt billentyűket, akkor a **RETURN** billentyű megnyomása után fog "megszólalni" a C-64. Válasza a következő lesz:

```
?SYNTAX  ERROR
READY.
```

Az utolsó sorban megjelenik a kurzor. Ebből azt a következtetést vonhatjuk le, hogy a C-64 a **RETURN** billentyű megnyomására reagál. Valóban így van. A C-64 a **RETURN** billentyű hatására elolvassa a kurzor sorába beírt írásjeleket,s annak tartalmától függő tevékenységet végez el. Esetünkben a HOGY HIVNAK? kérdésre várunk választ, de a C-64 nem ismeri a magyar nyelvet, ezért a "Számomra értelmetlen"-jelentésű ?SYNTAX ERROR üzenetet írja ki, majd a READY. üzenettel jelzi, hogy várja a következő parancsot. Bár tartalmilag sikertelen volt első kísérletünk, mégis sikeres, hiszen szóra, válaszra bírtuk a C-64-et. Mielőtt továbblépnénk,foglaljuk össze felfedezésünket!

A C-64 A **RETURN** BILLENTYŰ MEGNYOMÁSÁVAL BEFEJEZETT JELSOROZATOT ELOLVASSA,ÉS ANNAK TARTALMÁTÓL FÜGGŐEN CSELEKSZIK.

Ezután azt kell megvizsgálnunk, hogy milyen nyelven ért a C-64.Ez a nyelv a BASIC, amelynek szabályait jónéhány évvel ezelőtt rögzítették nemzetközi megállapodásokban,s amelynek nagyon rövid történetét a 4. fejezet előszava tartalmazza majd. A szabályok lényege, hogy egyszerű konstrukciójú utasításokat kell adnunk a számítógépnek. Olyan egyszerűeket, amelyek sem a számítógépnek, sem a nem feltétlenül programozó képzettségű felhasználónak nem jelent komolyan megerőltető feladatot.

Fejlesszük tovább előző ötletünket,s kérdezzük meg a C-64-től, hogy mennyi 2+2. Próbálkozzunk először így:

```
2+2? RETURN
```

Ha begépeljük kérdésünket, akkor az előzőtől eltérően semmi sem fog történni. Próbálkozzunk meg fordított sorrenddel!

```
?2+2 RETURN
```

Meglepődve fogjuk tapasztalni, hogy a C-64 a helyes 4 eredményt fogja szolgáltatni. Ebből levonhatjuk azt a következtetést, hogy a kérdés egyfajta elfogadott formája "? kérdés" sorrend. Ezt megerősítendő próbálkozzunk a következőkkel:

```
?13+25-3 RETURN
?3*3+4*4 RETURN
?1964,09,16 RETURN
```

Ha sorban egymás után végrehajtjuk a fenti utasítás-sort, akkor a következő válaszokat, eredményeket kapjuk.

```
35
25
1964          9             16
```

Próbálkozással eljutottunk odáig, hogy számológépként már tudjuk használni a C-64-et. Ez egyfajta eredmény, de az igazi használattal várjuk meg az elméleti alapok elsajátítását!

1.3 Alapvető parancsok

Ebben a szakaszban azoknak az alapvető feladatoknak a megoldását mutatjuk meg, amelyek nélkül a C-64 csak számológép-feladatokat tudna ellátni. Lehet ezek nélkül is programokat írni, de azt hisszük, hogy .senki sem akar mindig mindent újrakezdeni. Nos, az itt következő C-64 parancsok éppen ezt a problémát oldják meg a kész programok tárolásával, számítógépbe töltésével, illetve lemezegység használata esetén a tartalomjegyzék kiíratásával.

Tartalomjegyzék kiíratása lemezről

A feladat végrehajtásához a következő számítógépes rendszernek (konfigurációnak) kell rendelkezésre állnia:

- C-64 személyi számítógép,
- eredménykijelző TV-készülék,
- lemezegység.

Helyezzük el a lemezegységben a vizsgálni kívánt lemezt a következőképpen:

- Vegyük ki a lemezt a tároló dobozából, majd tasakjából úgy, hogy a jobb felső felénk eső sarkon legyen a lemez márkajele (ha kétoldalas - double sided- lemez második oldalának tartalmára vagyunk kíváncsiak, akkor a lemezt a mutatóujjunk iránya,mint forgatási tengely körül 180 fokkal forgassuk el),
- finoman nyomjuk meg a lemezegység elülső oldalának közepén levő pici műanyag lapocskát, melynek hatására a lemezegység lemez-tartója kinyílik. (Ez a lépés szószerint csak a redőnyzáras lemezegységek esetén igaz, billentyűs készülék esetén tanulmányozzuk a lemezegység használati utasítását!)
- Ezután finoman csúsztassuk be a lemezt a lemezegységbe, majd finoman húzzuk le az előbbi lapocskát úgy, mintha egy kézzel kezelt redőny lenne.

Ha elvégeztük a lemez behelyezését a lemezegységbe, akkor vigyük a kurzort egy üres sor elejére, majd gépeljük be a következő jelsorozatot:

LOAD"$",8 RETURN

A C-64 számára felírt jelsorozat jelentése a következő:

LOAD - utasítás-kulcsszó, amely arra utasítja a C-64-et, hogy valamely berendezésről a memóriába töltsön egy hosszabb jelsorozatot (programot, adatállományt stb.). Az utasítás további paraméterei (a sor további tartalma) mondja meg, hogy mi legyen ez a jelsorozat és hogy ezt a jelsorozatot honnan várja a C-64.

"$" - a betölteni kívánt jelsorozat neve, ahol a kezdő és a záró idézőjel kötelezően mindig jelen van és a kettő közötti ún. azonosító

változik esetről esetre. Esetünkben azért használtunk $-t, mert minden munkára előkészített (megformált) lemez ezen a néven rögzített területén tartalmazza a következő információkat:

- lemez azonosítója,
- lemez verziószáma,
- a lemezen tárolt minden egyes program, adatállomány hosszát, azonosítóját, típusát,
- végül a lemezen még üresen álló helyek mennyiségét.

Megjegyzés: tartalomjegyzéke csak az előkészített, ún. megformált lemezeknek van. (Az újonnan elővett lemeznek még nincs tartalomjegyzéke!)

Mind a tárolt állományok hosszát, mind az üres hely mennyiségét ún. blokkokban adja meg.

,8 - annak az egységnek, berendezésnek az ún. fizikai száma, amelyről az információt be akarjuk olvasni a memóriába. Ez lemezegység esetében leggyakrabban a 8 számot viseli. (További lehetőségeket majd később tárgyalunk.)

RETURN - Itt és a továbbiakban a vastagon írt jelsorozat mindig a C-64 billentyűzetének adott feliratú billentyűjét jelenti. Tehát ebben az esetben a **RETURN** billentyű megnyomása segítségével olvastatjuk el parancsunkat a C-64-gyel.

Amikor a LOAD"$",8 jelsorozat után megnyomjuk a **RETURN** billentyűt, akkor a következő játszódik le számítógépes rendszerünkben:

- A C-64 megfelelő adatátviteli alprogramja elolvassa az üzenetet és megvizsgálja, hogy számjeggyel vagy betűvel (esetleg írásjellel például ?) kezdődik-e.
- Ha mint esetünkben az L-betűvel kezdődik az üzenet, akkor addig olvassa az üzenet elejét, amíg egy utasítás-kulcsszót vagy értékadó =-jelet nem talál (ha ez nem sikerül, akkor hibaüzenetet ír és leáll. Lásd például az előző szakasz 2+2? vagy HOGY HIVNAK? utasítás-kísérleteire kapott válaszokat!)
- Ha megtalálta a kulcsszót, akkor megkeresi a hozzá szükséges paramétereket (esetünkben a betöltendő állomány nevét, a "$"-t, továbbá a berendezés fizikai sorszámát, a 8-ast, valamint a betöltési módot eldöntő kódot, mely esetünkben nem szerepel. (Minden utasításnál kétfajta paraméter van: a kötelező és az esetleges.) Esetünkben a LOAD utasítás csak az első paraméter szerepeltetését követeli meg. Ezért a harmadik paraméter hiánya nem okoz fennakadást.
- Mivel az utasítás a formális követelményeknek eleget tesz, a C-64 hozzálát az utasítás végrehajtásához, megkeresi a 8-as jelű fizikai berendezést, majd annak tartalomjegyzék területét. Ezt a tevékenységet jelzi a SEARCHING FOR $ üzenettel.
- Amikor megtalálta a tartalomjegyzéket és azon belül a $-azonosítójú tartalomjegyzéket, akkor ezt és a tartalomjegyzék memóriábatöltésének elkezdését a LOADING üzenettel hozza a felhasználó tudomására.
- Amikor a tartalomjegyzék memóriábatöltése sikeresen befejeződött, akkor ezt a READY. üzenettel és alatta a villogó kurzorral hozza a felhasználó tudomására.
- Mint sejthető a tartalomjegyzék speciális program, így annak tartalmát nem a programoknál megszokott végrehajtási paranccsal, hanem

LIST paranccsal íratjuk ki a képernyőre (vagy majd később a mátrixnyomtatóra). Tehát ahhoz, hogy a tartalomjegyzéket láthassuk, gépeljük be a következő parancsot:

LIST RETURN

Ha ezt az 1541 TEST/DEMO lemez esetén megtesszük, akkor a következő jellegű táblázatot kapjuk:

```
0 "TEST/DEMO 1/85  84 2A"
14    "HOW TO USE"          PRG
8     "HOW PART TWO"        PRG
7     "HOW PART THREE"      PRG
4     "VIC-20 WEDGE"        PRG
1     "C-64 WEDGE"          PRG
4     "DOS 5.1"             PRG
9     "PRINTER TEST"        PRG
6     "DISK ADDR CHANGE"    PRG
12    "VIEW BAM"            PRG
15    "DISPLAY T&S"         PRG
4     "CHECK DISK"          PRG
11    "PERFORMANCE TEST"    PRG
5     "SEQ.FILE.DEMO"       PRG
18    "REL.FILE.DEMO"       PRG
7     "SD.BACKUP.C16"       PRG
7     "SD.BACKUP.PLUS4"     PRG
10    "SD.BACKUP.C64"       PRG
7     "PRINT.64.UTIL"       PRG
7     "PRINT.C16.UTIL"      PRG
7     "PRINT.+4.UTIL"       PRG
13    "UNI-COPY"            PRG
30    "C64 BASIC DEMO"      PRG
35    "+4 BASIC DEMO"       PRG
8     "LOAD ADDRESS"        PRG
7     "UNSCRATCH"           PRG
5     "HEADER CHANGE"       PRG
403 BLOCKS FREE.
```

A kapott táblázatból a következő információkat olvashatjuk le:

- a lemez neve - TEST/DEMO 1/85 ,
- a lemez azonosítója - 84,
- a lemez a 2A jelű rendszerben alkalmazható,
- a lemez első egysége a 14 blokk hosszúságú, HOW TO USE-azonosítójú program (ez utóbbi információt a PRG jelsorozat jelzi),
- az utolsó sor információtartalma, hogy még 403 blokk áll a további tárolandó információk rendelkezésére. (Ez utóbbi nemcsak program, hanem adatállomány is lehet.)

Ha egy kollégánk által használt lemez aktuális (pillanatnyi) tartalmára vagyunk kíváncsiak, akkor az ő lemezével végrehajtva a feladatot, a következő táblázatot kapjuk:

```
0   "ZEBEGENY          " 85 2A
57     "SZOVEGSZERKESZTO" PRG
9      "KEP.1"           PRG
73     "C-64 DB1"        SEQ
79     "C-64 DB2"        SEQ
448  BLOCKS FREE.
```

A kapott tartalomjegyzékből a következő információkat tudjuk leolvasni:

- a lemez neve ZEBEGENY,
- a lemez azonosítója 85,
- a lemez első programja az 57 blokk terjedelmű SZOVEGSZERKESZTO azonosítójú program, melynek ismertetését vagy a lemezhez mellékelt használati utasítás tartalmazza,vagy a lemez tulajdonosa tudja leírni (ennek hiánya esetén magunknak kell találgatni),
- a lemez utolsó két állománya a C-64 DB1, illetve C-64 DB2 azonosítójú 73, illetve 79 blokkot elfoglaló adatállomány,
- a 448 BLOCKS FREE. üzenet azt jelenti, hogy az egy lemezoldal induló 664 blokknyi területéből még 448 blokk nincs elfoglalva.

A vastagon szedett fejléc a valóságban inverzben jelenik meg.

Program betöltése lemezről

A feladat végrehajtásához a következő számítógépes rendszerre (konfigurációra) van szükségünk:

- C-64 személyi számítógép,
- eredménykijelző TV-készülék,
- lemezegység.

Ha a fenti számítógépes rendszer bekapcsolt állapotban van, akkor a tartalomjegyzék kérésekor leírt módon helyezzük be a lemezegység lemez-tartójába azt a lemezt, amelyen elképzeléseink, információink szerint az alkalmazni kívánt program tárolva van. (Ha nem vagyunk benne biztosak, akkor első lépésként a szakasz elején leírt módon kérjük le a lemez tartalomjegyzékét!) Tételezzük fel, hogy ennek eredményeként a következő tartalomjegyzéket kaptuk:

```
0   "VARGESZTES        " 85 2A
72     "SIMON"           PRG
14     "VA1TV"           PRG
16     "VA1VB"           PRG
23     "VA2TV"           PRG
26     "VA2VB"           PRG
45     "VA3VB"           PRG
72     "E-REG"           PRG
62     "CPM"             PRG
41     "UTAZO UGYNOK"    PRG
293  BLOCKS FREE.
```

Mivel a C64 BASIC DEMO azonosítójú programot keressük, próbálkozzunk a TEST/DEMO 1/85 jelű lemezzel! Próbálkozásunk ekkor sikeres lesz. (Ld. 19. oldal)

A C64 BASIC DEMO azonosítójú program a C-64 BASIC-nyelvű programozással megvalósítható lehetőségeket mutatja be. Ismerjük hát meg ezeket a lehetőségeket. Ekkor a következőt kell tennünk: - gépeljük be a következő parancsot:

LOAD"C64 BASIC DEMO",8 **RETURN**

Amikor megnyomjuk a **RETURN** billentyűt, akkor a tartalomjegyzék kérésekor tapasztaltakhoz hasonlóan előbb a lemezegységhez, majd annak tartalomjegyzékéhez fordul (ezt jelzi a SEARCHING FOR C64 BASIC DEMO üzenettel). Ha megtalálta a tartalomjegyzékben a lemezen,akkor hozzálát a memóriába töltéséhez (ezt jelzi a LOADING üzenettel). Ha ezt is sikeresen végrehajtotta, akkor a READY. üzenet és villogó kurzor egyidejű megjelenítésével hozza ezt a felhasználó tudomására.

Program indítása

A feladat csak akkor merülhet fel, ha a memóriában végrehajtásra alkalmas utasítássor (program) van. Ez létrejöhet az előző,vagy ehhez hasonló parancsok eredményeként, vagy pedig úgy, hogy a felhasználó maga alkot a C-64 számára - legalább egy lépésből álló - utasítássort,és ezt akarja az alább ismertetésre kerülő parancs segítségével végrehajtani. Az utasítás (parancs) jellegéből következik, hogy végrehajtásához a C-64-en és az eredménykijelző TV-készüléken kívül csak azokra a berendezésekre van szükségünk, amelyekre a végrehajtásra váró utasítássorozatnak szüksége van.

Ha az előző bekezdésben elmondottak igazak, vagyis a memóriában van egy végrehajtásra váró program (utasítássor), akkor ennek elindításához nem kell mást tennünk, mint a következőt beolvastatni a C-64-gyel:

RUN **RETURN**

Ha ezt megtesszük, akkor a C-64 az utasítások tartalma szerint elkezd dolgozni mindaddig, amíg tennivalót talál,vagy ellentmondást nem tapasztal. Ezt a folyamatot tekinthetjük igazi C-64-hez méltó számítógépes munkafolyamatnak.

A folyamat mélyebb ismertetésére - szükség szerint - később kerítünk sort.

A FELHASZNÁLÓI GÉPHASZNÁLATRÓL

Program betöltése mágnesszalagról

Ezzel a feladattal akkor találkozunk, ha végrehajtani (esetleg fejleszteni) kívánt program magnetofon-kazettán áll rendelkezésünkre. Itt hasonló problémával állunk szemben, mint az a pop-rajongó, akinek rengeteg kazettán rengeteg pop-felvétel áll rendelkezésre. Nagyon pontos regisztrálás nélkül egyszerűen képtelenség rendet tartani a magnetofonkazettán tárolt programok között, hiszen itt nincs a kazetta elején tartalomjegyzék, a programokat semmiféle eszköz, módszer nem védi meg attól, hogy újabb programot (esetleg normális kazettás magnetofonnal) újabb pop-felvételt írjunk féltve őrzött programunk helyére. Ez az eszköz olcsóbb, de egyszersmind lassúbb és veszélyesebb adat- és programtárolási eszköze a C-64-es számítógépes rendszernek, mint a lemezegység. Mégis alkalmazása eléggé elterjedt.

A feladathoz szükséges minimális számítógépes rendszer a következő:

- C-64 személyi számítógép,
- eredménykijelző TV-készülék,
- szalagegység (C2N casette unit).

Ha ez üzemkész állapotban van, akkor helyezzük be a betölteni kívánt programot tartalmazó kazettát a szalagegységbe. Ezután gépeljük be a következő szöveget:

```
LOAD"IMPROVIZACIO",1 RETURN
```

A begépelt parancs idézőjelben szereplő része utal arra, hogy most éppen az IMPROVIZACIO nevű programot szeretnénk a memóriába tölteni. A vessző után szereplő 1 utal arra, hogy a programot a C-64 az 1 fizikai sorszámú szalagegységen keresse. Mivel a Commodore-számítógépek első kiegészítő háttértárolója a szalagegység volt, így azt akkor is feltételezi, ha a parancsból elhagyjuk a ,1 karakterpárt.

Tehát a LOAD utasításban csak akkor kötelező a második paraméter, ha az nem 1-értékű.

Ha a helyes parancsot beolvastatjuk a C-64-gyel, akkor a következő tevékenységsor indul el:

- a C-64 kiírja a képernyőre a következő felszólítást:

```
PRESS PLAY ON TAPE
```

Megjegyzés: Ezt az üzenetet nemcsak a fenti parancs, hanem más módon is pl. a siftelt **RUN-STOP** billentyűpár segítségével előidézhetjük.

- Ez azt jelenti, hogy indítsuk el a C2N berendezést a PLAY billentyű megnyomásával.

Ekkor teljesen kivilágosódik a képernyő s a C2N folyamatosan küldi a kazetta tartalmát a memóriába,

- a C-64 folyamatosan figyeli a C2N-ről jövő jeleket. Ha azok között program (adatállomány) nevet talál, akkor azt kb. 5 másodpercig olvashatóan kijelzi a képernyőre a következő formában:

```
FOUND IMPROVIZACIO
```

Ha ez a keresett program, akkor aláírja a LOADING üzenetet, ha nem ,akkor anélkül, világos képernyő leple alatt tovább olvas,

- ha a betölteni kívánt programot sikerült betöltenie, akkor újra visszakapjuk az eredeti képernyőt READY. üzenettel kiegészítve, továbbá a C2N egységet is automatikusan kikapcsolja.

Ezután a program a RUN paranccsal végrehajtható, a LIST parancscsal pedig a programlistát a képernyőre kérhetjük.

Megjegyzés: Előbbi mondataink sok-sok megjegyzésre adnak okot. Ezekre majd a mágnesszalagos adatállományt is tárgyaló 14. leckénél fogunk visszatérni.

Program tárolása lemezen

Tételezzük fel, hogy van egy programunk, amely megőrzésre érdemes. Legyen ez a következő:

```
10 PRINT CHR$(147)
20 X=1: Y=1
30 DX=1: DY=1
40 POKE 1024+X+40*Y,81
50 X=X+DX: Y=Y+DY
60 FOR T=1 TO 100: NEXT T
70 IF X=0 OR X=39 THEN DX=-DX
80 IF Y=0 OR Y=23 THEN DY=-DY
90 GO TO 40
```

Ha ezt a programot begépelnénk szigorúan karakterhűen, akkor a RUN parancs hatására a következő történne:

- a 10 utasítás hatására üres képernyőt kapnánk,
- a 20-as utasítás-pár induló értéket ad X-nek és Y-nak,
- a 30-as utasítás-pár induló értéket ad DX-nek és DY-nak,
- a 40-es utasítás kirajzol egy képernyőpozícióra egy labdát szimbólizáló korongot,
- az 50-es utasítás-pár kiszámítja az aktuális X-et és Y-t,
- a 60-as utasítás rövid időre leállítja a munkát,
- a 70-es és 80-as utasítás arról gondoskodik, hogy az X- és Y-értékek korláton belül maradjanak. (Ez biztosítja azt, hogy a "labda" a képernyőn maradjon.),
- a 90-es utasítás ismételteti a 40-estől a 80-asig terjedő utasítássort.

Ha valamely Olvasónknak nehezen volt követhető az előző nagyon rövid magyarázat, akkor a következő két lehetőség közül választhat:

- felejtse el a magyarázatot s tekintse úgy a 10-90 sorokat, mint egy kész lehetséges programot, amelyet majd később ismerünk meg működés közben,
- gépelje be soronként az utasításokat - és ha van türelme, akkor próbálja ki minden sor begépelése után.

A programlista ilyen korai bemutatásával pusztán az volt a célunk, hogy elősegítsük a program-fogalom megalkotását és a vele való bánásmódot.

Ezekután oldjuk meg azt a feladatot, hogy a fenti programot tároljuk a ZEBEGENY azonosítójú lemezen (vagy a TEST/DEMO1/85-n). Ezt a következőképpen tehetjük meg:

- ellenőrizzük, hogy megfelelő számítógépes rendszer (C-64, TV-készülék, lemezegység) üzembehelyezéséről gondoskodtunk-e,
- helyezzük a kiválasztott lemezt a lemezegységbe,
- ellenőrizzük (például egy LIST paranccsal), hogy a memóriában van-e a tárolni kívánt program,
- ha minden rendben van, akkor a következő paranccsal hajtassuk végre a feladatot:

```
SAVE"LABDA",8 RETURN
```

Ha a parancsot végrehajtva, a READY. üzenettel bejelentkezik a C-64, akkor két módunk van a tárolás helyes végrehajtásának ellenőrzésére: végrehajtatjuk az ellenőrző programot, vagy egy másik C-64-gyel elolvastatjuk a lemezen feltételezett LABDA nevű programot. Kikapcsolás előtt érdemes a két lehetőség valamelyikével élni, hiszen az esetek többségében többórás programozói munka veszhet kárba.

Program tárolása mágnesszalagon

Tételezzük fel, hogy nem áll rendelkezésünkre lemezegység, csak szalagegység, melyre ugyanazt a labdapattogtató programot szeretnénk tárolás céljából kiírni, mint amelyet az előző parancsnál lemezre tettünk. Mennyiben módosul a feladatunk?

Megjegyzés: A Commodore-cég a mágnesszalagos egység versenyképessége érdekében TURBO TAPE néven olyan programot hozatott létre, mely a kazetta olvasását és írását az eredeti többszörösére gyorsította.

Ekkor lépéseink a következők lesznek:

- szükséges számítógépes rendszer (C-64, TV-készülék, szalagegység) ellenőrzése,
- a kazettára mentendő program ellenőrzése a memóriában (például a LIST parancs segítségével),
- a kazettát a C2N tekercselő billentyűi segítségével a megfelelő számlálóállapotba állítjuk (nem szabad például a kazetta legelejével kísérletezni; nem ajánlatos értékes programot felelőtlenül letörölni, ...stb.),
- ha minden rendben van, akkor a következő parancs begépelésével hajtassuk végre a feladatot,

```
SAVE"LABDA",1 RETURN
```

Ha mindent helyesen tettünk, akkor megjelenik a következő üzenet:

```
PRESS RECORD & PLAY ON TAPE
```

- ha ez a felirat megjelenik a képernyőn, akkor indítsuk el a C2N-t a felvételt és lejátszást vezérlő billentyűk együttes megnyomásával,

- a képernyő teljesen kivilágosodik, mely csak a felvétel befejezése után kapja vissza eredeti színösszeállítását és tartalmát. A felvétel ellenőrzésére a lemez esetében már felsorolt módok állnak rendelkezésünkre.

1.4 Programok végrehajtása

Talán már az eddigiekből is kitűnik, hogy a C-64 utasítások, utasítássorok végrehajtására alkalmas berendezés. Ahhoz, hogy ezt a berendezést minél célszerűbben alkalmazhassuk, elengedhetetlen, hogy már a kezdet kezdetén megismerkedjünk a C-64 programvégrehajtási filozófiájával,gondolatmenetével.

Tételezzük fel, hogy programozni már jól tudó barátunk elkészítette számunkra azt a programot, amely 4 érdemjegyből kiszámítja az átlageredményünket. A következő programlistát tölti be C-64-ünkbe:

```
10 PRINT CHR$(147)"BEVEZETO MINTAPROGRAM"
20 PRINT "MINDIG EGY SZAMMAL"
30 PRINT "ES A (RETURN)-NAL"
40 PRINT "VALASZOLJ!"
50 INPUT " 1. JEGY";J1
60 INPUT " 2. JEGY";J2
70 INPUT " 3. JEGY";J3
80 INPUT " 4. JEGY";J4
90 PRINT "ATLAGEREDMENY=";(J1+J2+J3+J4)/4
100 END

READY.
```

Ha a C-64-en programozó barátunk,vagy a saját programbetöltési tudományunk segítségével rendelkezésünkre áll a fenti program, akkor azt az előző szakaszban már jelzett RUN-parancs segítségével el tudjuk indítani. A program működési mechanizmusának követéséhez nagyon előnyös, ha a program listája valamilyen formában rendelkezésünkre áll, hiszen a képernyőről az első utasítás hatására el fog tünni.

1. lépés:

A RUN parancs begépelése, (csak az a parancs helyes, amikor 3 betűt, az R-t, az U-t és az N-t írjuk be) (a **RUN-STOP** billentyű más célt szolgál!) majd a **RETURN** segítségével a program elindítása.

Tevékenységünk hatására a képernyő letörlődik s a következő jelenik meg rajta:

```
BEVEZETO MINTAPROGRAM
MINDIG EGY SZAMMAL
ES A (RETURN)-NAL
VALASZOLJ!
 1. JEGY?
```

A villogó kurzor a kérdőjel utáni második pozíción várja az első érdemjegyet. Legyen ez mondjuk 3.

2. lépés:

Írjuk be a 3-at és a **RETURN**-nel olvastassuk el a számítógéppel. Ha megtesszük, a következőképpen fejlődik a képernyő tartalma:

```
BEVEZETO MINTAPROGRAM
MINDIG EGY SZAMMAL
ES A <RETURN>-NEL
VALASZOLJ!
 1.  JEGY? 3
 2.  JEGY?
```

Válaszaink legyenek sorban 4, 4, 5. Ha ezeket a második lépéshez hasonlóan begépeljük, akkor a következő eredménytáblázatot kapjuk:

```
BEVEZETO MINTAPROGRAM
MINDIG EGY SZAMMAL
ES A <RETURN>-NEL
VALASZOLJ!
 1.  JEGY? 3
 2.  JEGY? 4
 3.  JEGY? 4
 4.  JEGY? 5
ATLAGEREDMENY 4

READY.
```

Programunkat tehát a C-64 sikeresen végrehajtotta, s eredményként 4-es átlagértéket számított ki nekünk.

Ebből a nagyon egyszerű,fejben is végrehajtható programból azt a következtetést vonhatjuk le, hogy valamennyi jól dokumentált, párbeszédes üzemmódban megírt program nagyon könnyen kezelhető. Amikor majd saját programjainkat írjuk, akkor se feledkezzünk meg erről a tényről.

A C-64-en nemcsak BASIC-nyelvű programokat hajtathatunk végre, hanem más programozási nyelveken megírt programokat is, melyekhez az esetek többségében fordítóprogram használata is szükséges.

1.5 Gyakran használt programok felsorolása

Amikor egy számítógépet alkalmazunk, akkor mindig az a célunk, hogy a megoldandó feladatot minél gyorsabban, minél célszerűbben oldjuk meg. Hiába áll rendelkezésünkre a legmodernebb gépcsoda, ha azt csak számológépként használjuk. Célszerű minél hatékonyabb programokat alkalmazni, mind a konkrét feladatok megoldásánál, mind az azok megoldásához

megírt programok kidolgozásánál. Amikor javaslatot teszünk arra, hogy milyen feladathoz milyen programot válasszon a tisztelt Olvasó, akkor először a programfejlesztést elősegítő programokra, majd a felhasználói programcsomagokra hívjuk fel a figyelmet.

Programfejlesztő programok:

F-COPY 58.5K - az első lemez-karbantartó programok egyike, amely alkalmas lemezek megformázására, tartalomjegyzék készítésére, lemez-tartalom tömörítésére, továbbá program-sorozatok egyik lemezről másikra történő másolására.

COPY 190 - az előzőhöz hasonló program, mely nemcsak lemezről lemezre tud programot másolni, hanem kazettával is tud dolgozni tetszőleges párosításban.

TURBO TAPE - a kazetta vonatkozásában gyors programírást és beolvasást végrehajtó program.

HELP+ - kifejezetten - ahogy nevének magyar megfelelője is mondja - a professzionális programozást SEGÍTŐ, vezérlő program.

SIMON'S BASIC - olyan bővített BASIC-utasításkészlet, mely az alap-BASIC-ben csak nehezen megoldható feladatokat egyszerűen leírhatóvá, vezérelhetővé teszi.

AUSTROCOMP - BASIC-nyelvű programok gyors végrehajtását és kevés helyfoglalását elősegítő program, amely segít a programok titkosításában. (Így védi a programozó szerzői jogát is.)

Felhasználói programok:

EASY SCRIPT - az egyik legjobban elterjedt szövegszerkesztőprogram.

DIGITEXT - az első igazán sikeres magyar nyelvű szövegszerkesztőprogram, mely elsősorban körlevelek szerkesztésénél tesz hasznos szolgálatot.

PRACTICALC - információtáblázatokat feldolgozó programcsomag.

Mivel tankönyvünk feladata a programozói tudás megalapozása, ezért az igazán bőséges C-64 program-választék felsorolásától eltekintünk, s hamarosan rátérünk vállalt feladatunk teljesítésére.

2. SZÁMÍTÁSTECHNIKAI MUNKAFOLYAMAT

Miután vázlatosan megismerkedtünk a C-64 kezelésével, itt az ideje, hogy hozzálássunk a tervszerű felhasználásához. Hogy ezt megtehessük, meg kell tanulni a tervszerű munka elemeit elvben és a gyakorlatban egyaránt. Ezt a fejezetet ezen munkamódszer elsajátítására, illetve elmélyítésére szánjuk.

Első lépésként fogalmazzuk meg a számítástechnikai munkafolyamat lényegét!

AZ OLYAN MUNKAFOLYAMATOT, AMELY SZÁMÍTÁSTECHNIKAI ESZKÖZ VAGY SZÁMÍTÁSTECHNIKAI MÓDSZER (VAGY MINDKETTŐ) SEGÍTSÉGÉVEL AZ ALÁBBIAKBAN FELSOROLT MUNKALÉPÉSEK VÉGREHAJTÁSA UTÁN JUT EL A KITŰZÖTT PROBLÉMA MEGOLDÁSÁHOZ, **SZÁMÍTÁSTECHNIKAI MUNKAFOLYAMAT**-NAK NEVEZZÜK. A MUNKAFOLYAMAT LÉPÉSEI (SZAKASZAI) A KÖVETKEZŐK:

- A MEGOLDANDÓ FELADAT SZABATOS SZAKMAI, FELHASZNÁLÓI MEGFOGALMAZÁSA,
- A MEGOLDANDÓ FELADAT SZABATOS SZÁMÍTÁSTECHNIKAI MEGFOGALMAZÁSA,
- A MEGOLDÓ ELJÁRÁS KIVÁLASZTÁSA,
- A MEGOLDÓ ELJÁRÁS RÉSZLETES MEGTERVEZÉSE (A PROGRAMVÁZLAT ELKÉSZÍTÉSE),
- A SZÜKSÉGES SZÁMÍTÁSTECHNIKAI ESZKÖZ(ÖK) KIVÁLASZTÁSA,
- A PROGRAMOZÁSI NYELV(EK) KIVÁLASZTÁSA,
- A MEGTERVEZETT ELJÁRÁS BEPROGRAMOZÁSA,
- A PROGRAM KIPRÓBÁLÁSÁHOZ SZÜKSÉGES ÚN. TESZTADATOK ÖSSZEÁLLÍTÁSA, ÖSSZEGYŰJTÉSE,
- A PROGRAM KIPRÓBÁLÁSA,
- A TÉNYLEGES FELADAT MEGOLDÁSÁHOZ SZÜKSÉGES ADATOK ÖSSZEGYŰJTÉSE,
- A PROGRAM VÉGREHAJTÁSA A TÉNYLEGES ADATOKKAL,
- A KAPOTT EREDMÉNY(EK) SZAKMAI ÉRTÉKELÉSE,
- SZÜKSÉG ESETÉN MÓDOSÍTÁS(OK), MAJD ÚJBÓLI VÉGREHAJTÁS,
- AZ EREDMÉNYEK ÉS A PROGRAM DOKUMENTÁLÁSA.

Ha figyelmesen végiggondoljuk az előbbi definícióban elmondottakat, akkor nyilvánvaló, hogy pontosan ezeket a dolgokat kell megtennünk s pontosan ilyen sorrendben. Az itt felsorolt fogalmak első hallásra idegenül hangozhatnak, de a későbbi munka érdekében már most meg kell barátkoznunk velük.

Minden számítógépet - így a C-64-et is - úgy kell elképzelnünk, mint az udvarias pincért, aki mindent betű szerint feljegyez,s ezeket a feljegyzéseket mindig szigorúan figyelembe veszi, minden apró mozzanatát ezek a "feljegyzések", információk irányítják.

Fejlesszük tovább hasonlatunkat! Találóbb az a hasonlat, amely szerint a számítógép egy pedáns, rendszerető hivatalnokra vagy adminisztrátorra hasonlít, aki minden információt rendszerez, a megfelelő fiókba tesz (ezek játszák majd a változók szerepét),s különböző tanfolyamokon elsajátította

az információkezelés legmodernebb módszereit (ezek lesznek a programok), melyeket nem a fejében, hanem az információkhoz hasonlóan lejegyzetelt formában tárol valamelyik íróasztal fiókjában,s a hivatalnok (a számítógép) nem tud mást,mint a "fiókok" előírásosan pedáns kezelését.

Ha Mi lennénk a C-64, akkor a memóriát képzeljük el,mint egy nagy kockás számtanfüzetet, amelyben mint az általános iskolás kisdiák, a következő tevékenységeket tudjuk elvégezni:

- bármely kockába adatot beírni,
- bármely kockából adatot kiolvasni,
- alapműveletek eredményét kiszámítani,
- logikai döntéseket hozni,
- az előbbi két művelet eredményét a kijelölt kockába beírni,
- szöveges információkat kezelni a numerikus (számszerű) információkhoz hasonlóan.

Képzeletbeli számítógépünknek két negatív tulajdonsága van:

- csak azt tudja, amire "megtanították",
- csak azt tudja végrehajtani, amire az adott helyzetben érvényes, szabályos parancsa, utasítása van.

Ezen gondolatok előrebocsátása után lássunk hozzá a számítástechnikai munkafolyamat részletes megismeréséhez!

2.1 A megoldandó feladat szabatos szakmai megfogalmazása

A számítástechnikai munkafolyamat elkezdésének elengedhetetlen feltétele a szabatosan, világosan megfogalmazott feladat. Ennek illusztrálására 3 nagyon hasonló feladatot tűzünk ki. Azért ennyit, hogy mindenki az érdeklődésének megfelelő szakmai megfogalmazást választhassa, továbbá annak illusztrálására, hogy különböző problémák vezethetnek ugyanahhoz a számítástechnikai feladathoz. Bármivel kezdünk el foglalkozni, nagyon fontos az érzelmi motiváció, a helyes érzelmi hozzáállás, hiszen ez nagyon sok kezdeti problémán átsegíthet. Ezekután feladataink a következők:

1. megfogalmazás:

Pistikét megkéri az édesanyja, hogy hordja tele hideg vízzel az udvar közepén lévő (egyszerűség kedvéért téglalap alakú) fürdőmedencét. Hány liter vizet kell Pistikének kihordania, ha ismerjük az úszómedence hosszát, szélességét és mélységét dm-ben?

2. megfogalmazás:

Józsi bácsi elment nyugdíjba. Szabadidejében kertészkedéssel fog foglalkozni. Vásárolt egy hétvégi telket, amelynek kertészkedésre alkalmatlan a talaja. Egy kertészkedő barátja azt javasolja neki, hogy vásároljon termőföldet s azzal egyenletesen borítsa be telkét. Józsi bácsi ismeri telkének hosszát és szélességét méterben, a barátja pedig a szükséges vastagságot mondja meg méterben. Mi a teendő?

3. megfogalmazás:

Unokafivérünk házat épít. Már állnak a falak, s a tetőfedéssel is végeztünk. Minden szobát a későbbi céltől függően más-más vastagságban le akar betonozni. Felajánlják neki kész beton szállítását. Ha ismeri egyenként szobáinak hosszát és szélességét méterben, továbbá a szükséges vastagságot méterben, akkor mi a teendő?

Ha egy picit figyelmesebben megvizsgáljuk az előttünk álló feladatokat, akkor nagyon könnyen észrevehetjük a hasonlóságot. Ez a hasonlóság nemcsak azért van, hogy ki-ki az ízlésének legjobban megfelelő feladatot válassza, hanem azért is, hogy érzékeltessük, megmutassuk, hogy nagyon sok szakmailag különbözőnek tűnő feladat számítástechnikailag nagyon hasonló bánásmódot igényel. Ez is egy döntő érv a számítástechnika, a számítógépre megfogalmazott feladatok széleskörű alkalmazhatósága mellett.

Mindhárom esetben egy téglatest 3 meghatározó (a téglatest 3-irányú kiterjedését leíró) adata (a hosszúság, a szélesség, a magasság) ismeretében már nagyon egyszerűen megtudjuk határozni a mindhárom feladatnál kérdezett térfogatot. Csupán arra kell vigyáznunk, hogy a mértékegységeket helyesen válasszuk meg.

Bármilyen bonyolult feladattal állunk szemben, nem kell mást tennünk, mint amit az előző bekezdésben tettünk. Tehát a megoldandó feladat szabatos szakmai megfogalmazása a következőt jelenti:

A FELADATOT FOGALMAZZUK MEG ÚGY, HOGY ABBÓL EGYÉRTELMŰEN KIDERÜLJÖN, HOGY MILYEN INFORMÁCIÓK (ADATOK) FELHASZNÁLÁSÁVAL MILYEN INFORMÁCIÓK (EREDMÉNYEK) MEGISMERÉSE A CÉL.

Ha ezt helyesen megtervezzük, akkor a továbbiakban már nagyon leegyszerűsödnek teendőink.

Gondoljunk csak arra, hogy előző 3 különböző feladatunk a következő 3 lépésre egyszerűsödött:

- 3 adat összegyűjtése,
- a térfogat kiszámítása,
- a kapott eredmény közlése.

Nyomatékosan hangsúlyozzuk, hogy csak jól megfogalmazott feladatokkal szabad a siker reményében foglalkozni. Vagyis csak akkor szabad egy feladatot elvállalni, ha tudjuk, hogy mit kell csinálnunk. A számítástechnika, a számítógép kegyetlenül megbünteti a mellébeszélést, a pongyolaságot, a lezserséget.

2.2 A megoldandó feladat számítástechnikai megfogalmazása

Mielőtt folytatnánk a feladatok megoldását - mintegy előillusztrációként - elmeséljük, hogyan oldatta meg Pistikével édesanyja a feladatot:

Pistike elővett egy üres kockás (négyzethálós) papírlapot. Először édesanyja kérésére felírta a teendőket:

- megmérem a medence hosszúságát, szélességét, mélységét,
- a medence térfogatát megkapom a hosszúság, a szélesség és a mélység szorzataként,

- mondjam meg a végeredményt édesanyámnak,
- készen vagyok a feladattal - kezdhetem a vízhordást,
- a papíron - mivel semmit sem tudok megjegyezni fejben - minden számadatnak helyet jelölök ki a következőképpen:

```
          *********
hosszúság *       *
          *********

          *********
szélesség *       *
          *********

          *********
mélység   *       *
          *********

          *********
térfogat  *       *
          *********
```

Miután Pistike felírta papírjára a teendőket és az adatok és az eredmény felírására szolgáló *-téglalapokat, kimegy az udvarra és hozzálát annak végrehajtásához.

- Megméri a medence hosszúságát dm-ben (tételezzük fel, hogy széttárt tenyere éppen 1 dm-t jelent, s hogy a medence 13 "tenyérnyi"), beírja a kapott 13 értéket a hosszúság *-téglalapjába, ezután megméri a szélességet (legyen ez 7 "tenyérnyi"), beírja az értéket a szélesség *-téglalapjába, majd végül megméri a mélységet (legyen ez 5 "tenyérnyi"), beírja a mélység *-téglalapjába. Ezekután a papírlap alja a következő képet mutatja:

```
          *********
hosszúság *  013  *
          *********

          *********
szélesség *  007  *
          *********

          *********
mélység   *  005  *
          *********

          *********
térfogat  *       *
          *********
```

- Pistike tud fejben számolni, így a második sorban szereplő "utasítás" alapján kiolvassa a hosszúság 13-as és a szélesség 7-es értékét, majd összeszorozza őket, eredményül 91-et kap, ezután kiolvassa a mélység 5-ös

értékét,s ezt összeszorozza a kapott 91-gyel, az eredmény 455 lesz, melyet beír a térfogat *-téglalapjába. Ezekután a papírlap a következő képet mutatja:

```
          *********
hosszúság *  013  *
          *********

          *********
szélesség *  007  *
          *********

          *********
mélység   *  005  *
          *********

          *********
térfogat  *  455  *
          *********
```

- végül megmondja édesanyjának, hogy 455 liter hideg vizet fog kihordani a fürdőmedencébe.

A feladat megoldását leírhattuk volna másképpen is, de ismerve a C-64 működési elvét, inkább egy ahhoz hasonló filozófiájú gondolatmenetet választottunk az esetleges egyszerűség rovására, kárára. Egyszerűbben fogalmazva: Pistike pontosan úgy dolgozott,mintha Ő lett volna a C-64.

A MEGOLDANDÓ FELADAT SZÁMÍTÁSTECHNIKAI MEGFOGALMAZÁSÁN A SZÁMÍTÁSTECHNIKAI TEENDŐK KONKRÉT FELSOROLÁSÁT ÉRTJÜK.

Pistike esetében ezek a teendők a következők voltak:
- a medence méreteinek feljegyzése,
- a medence térfogatának kiszámítása,
- a kapott eredmény közlése.

Egy feladatot akkor tekinthetünk helyesen előkészítettnek, megfogalmazottnak, ha az előzőhöz hasonlóan körvonalaztuk, kijelöltük a teendőket.

2.3 A megoldó eljárás kiválasztása, megtervezése

Amit most részletezni fogunk, az a bemutatott feladaton szőrszálhasogatásnak, felesleges fontoskodásnak tűnhet. Ennek ellenére kérjük az Olvasót az elmondandó gondolatmenet pontos követésére, mert az későbbi bonyolultabb feladatainknál is hasonló lesz.

1. lépés:

Amikor a feladatot Pistike oldotta meg, akkor az egyes adatokat, eredményeket leírta a számítógépet szimbolizáló kockás (négyzethálós) papírlapjára úgy, hogy minden mennyiségnek rajzolt egy többjegyű számot is "elbíró" tég-

lalapot és elé írta a beírandó mennyiség nevét. Hasonlóan kell eljárnunk akkor is, amikor a számítógép (C-64) számára tervezünk. Itt is "téglalapokat" kell lefoglalnunk valamennyi írásban szerepeltetni kívánt mennyiség tárolására. Pistikéhez hasonlóan ekkor is neveket kell adni az egyes "téglalapok"-nak. Amikor ezeket a neveket megválasztjuk, két szempontot kell szem előtt tartani:

- a név segítsen abban, hogy mit akarunk a vele megjelölt "téglalap"-ban tárolni,
- a név ne tartalmazzon felesleges információt.

Az első szempontnak Pistike nevei megfelelnek, hiszen megmondják a tárolni kívánt mennyiségek nevét. Mégis feleslegesen hosszúak, hiszen ahhoz, hogy tudjuk, hogy melyikkel akarunk dolgozni, ahhoz bőségesen elegendők lesznek a kezdőbetűk. Tehát választásunk eredménye a következő emlékeztető (mnemonikus) "névsor":

h, s, m, t

Ezután döntenünk kell arról, hogyan hivatkozzunk rájuk. Ha megfigyeltük azt, hogy mit csinált Pistike az udvaron, miközben meghatározta a kiviendő hidegvíz mennyiségét, akkor kézenfekvő lesz a szakirodalomban elterjedt VÁLTOZÓ (változó mennyiség) fogalom elfogadása, hiszen a megjelölt "téglalap" tartalma folytonosan változhat.

Tehát a 2.1-ben megfogalmazott feladatok megoldásánál a következő változókat fogjuk alkalmazni:

h, s, m, t

A feladat megoldása során játszott szerepe alapján beszélhetünk adat-, segéd- és eredmény-változókról. Feladatunk esetében az első 3 adat-, az utolsó pedig eredmény-változó. Mint láttuk esetünkben nem volt szükség segéd-változó (részeredmények tárolására szolgáló változó) alkalmazására. Vannak olyan feladatok is, mikor a változók többféle feladatot látnak el.

2. lépés:

Ebben a lépésben kell megtalálnunk a feladat megoldásában résztvevő eljárásokat,képleteket. Tehetjük ezt a tudásunk (a fejünk) segítségével vagy a megfelelő szakirodalom (illetve szakemberek) segítségével.

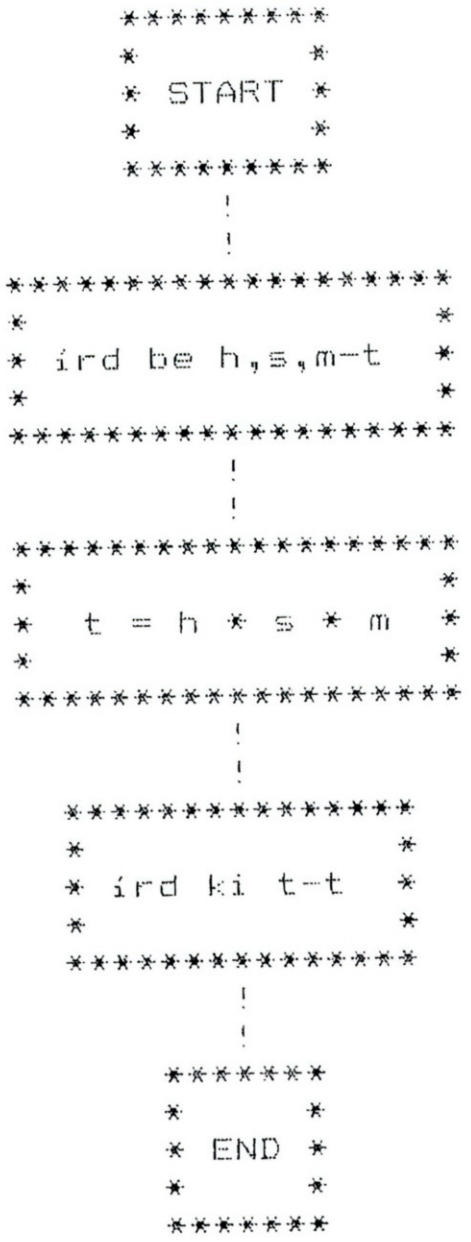
Valamennyi feladatmegfogalmazásunk esetén ez a következő: az aktuális változónevek felhasználásával megfogalmazott térfogat-képletet kell felírnunk, azaz:

t=h*s*m

Első pillantásra feleslegesnek tűnhet a *-ok szerepeltetése szorzójelként, de gondolva a **változónév** előző lépésbeli megválasztási módjára (nincs kikötve, hogy csak egybetűs változónév lehet) jeleznünk kell, hogy mi jelöl egyetlen mennyiséget.

3. lépés:

Most érkezett el a pillanat, hogy összeálljon feladatmegoldó eljárásunk teljes terve,ún. **programvázlata:**



Bár nagyon könnyen követhető a programvázlat, mégis röviden elmondjuk jelentését:

1. *-téglalap - jelzi a program elindítását a START felirattal.

Felkiáltójel-pár - jelzi a következő végrehajtandó utasítást, tevékenységet.

2. *-téglalap - jelzi az adatok beírását a h, s, m változókba.

Felkiáltójel-pár - ugyanúgy, mint az előbb.

3. *-téglalap - kiszámítja a "h * s * m"-szorzatot és beírja a kapott eredményt a t változóba.

Felkiáltójel-pár - ugyanúgy, mint előbb.

4. *-téglalap - kiírja a t változó pillanatnyi értékét (ez a művelet gondoskodik arról, hogy megtudjuk a felsorolt változók(k) tartalmát).

Felkiáltójel-pár - ugyanúgy, mint fent.

5. *-téglalap - az END felirattal jelzi a feladatmegoldás befejezését.

Külön felhívjuk az Olvasó figyelmét arra, hogy csak annak a változónak ismerheti az aktuális, pillanatnyi értékét, amelyet egy 4. *-téglalaphoz hasonló eljárás-lépésben kiíratott.

Amikor a programtervezést végezzük, akkor minden feladat esetében az előbbihez hasonló alapossággal kell előkészítenünk a megírandó számítógépes program vázlatát.

Mielőtt elkészítjük a programvázlat alapján a számítógépes programot, két döntést kell hoznunk.

1. döntés:

A rendelkezésünkre álló számítástechnikai eszközök (számítógépek) közül melyiket alkalmazzuk feladatunk megoldásánál. Tankönyvünk célja és a feladat egyszerűsége miatt nyílvánvaló, hogy a C-64-et választjuk.

2. döntés:

A választott számítógépen (C-64) rendelkezésünkre álló programozási nyelvek közül melyik a legalkalmasabb a feladat megoldására. Az előzőhöz hasonlóan most is alapvető célunk és a feladat egyszerűsége dönt a BASIC választása mellett.

2.4 A programvázlat BASIC-programjának elkészítése

A jól megírt programvázlat és a BASIC-utasításkészlet segítségével elkészítjük a megoldást a C-64-re.

A BASIC-programsorok általános alakja a következő:

sorszám utasítás (vagy utasítások :-tal elválasztva)

ahol az utasítás a következőt jelenti:

utasítás-kulcsszó [paraméter(ek)]

Minden programsort a **RETURN** billentyű megnyomásával kell elolvastatnunk a C-64-gyel.

A programsorok sorrendjét a sorszám szabja meg, melynek csak folyton növekvő voltát (de nem folyamatosságát) kívánja meg a BASIC-interpreter. A program egyfajta átkódolásáról gondoskodik a BASIC-interpreter.

Ezekután feladatunk megoldása a következő:

```
10 REM PISTIKE PELDAJA
20 INPUT H,S,M
30 LET T=H*S*M
40 PRINT T
50 END
```

Ha a fenti öt sort megvizsgáljuk, akkor mindegyik megfelel a programvázlat egy-egy *-téglalapjának, mindegyik tartalmaz sorszámot, utasítás-kulcsszót s - az utolsó kivételével - paraméter(eke)t.

Programozási tanulmányaink előkészítéseként mindegyik sornak szenteljünk néhány szót:

10 sorszámú - a programvázlat START-tartalmú lépésének felel meg. Semmit sem tesz csak mintegy megjegyzi (ennek a kulcsszava a REM), hogy Pistike példájáról van szó. Használata nem lenne kötelező. Pusztán a félreértések elkerülése érdekében alkalmazzuk az első *-téglalap megfelelőjeként.

20 sorszámú - kulcsszava az INPUT, mely arra utasít, hogy az utána felsorolt változók kapjanak értéket. Esetünkben H, S és az M.

30 sorszámú - ez az utasítás ad értéket a T változónak. Ezt jelenti a LET kulcsszó, amelynek a használata a C-64-BASIC-ben nem kötelező. Pusztán a kezdeti analógiák megkönnyítése érdekében alkalmaztuk. Később szinte mindig el fogjuk hagyni a LET kulcsszót.

40 sorszámú - közli a felhasználóval (az Olvasóval) a T változó pillanatnyi értékét, melyre a PRINT kulcsszó utasít.

50 sorszámú - az END paraméter-nélküli kulcsszó jelzi a BASIC-program befejezését.

Most csak ízelítőnek szántuk ezt a BASIC-program magyarázatot. Később fogjuk részletesen bevezetni az Olvasót a programozás rejtelmeibe. Ha mégis ki akarja próbálni az eddig tanultakat, akkor arra vigyázzon, hogy a program begépelésénél minden sort a **RETURN** billentyű megnyomásával olvastasson el a C-64-gyel.

Megjegyzés: Vigyázzon arra, hogy ne használja sorváltás céljára a **SHIFT-RETURN** billentyűpárt. Ügyeljen a sorok túlcsordulására.

2.5 A szükséges adatok összegyűjtése, a program végrehajtása

Amikor feladatunk megoldása során idáig jutottunk, akkor első lépésként próbáljuk ki programunkat olyan adatokkal, amelyek esetén a végeredmény könnyen fejben ellenőrízhető, vagy pedig előre tudható. Esetünkben ilyen eset, amikor a téglatestünk minden irányban egységnyi hosszúságú. Ekkor mindhárom induló adat 1 lesz. Hajtsuk végre ezekkel az adatokkal a programot. Ez a következő tevékenységeket igényli:

- gépeljük be a RUN parancsot. Ha ezt megtesszük, akkor a következő képernyőrészletet kapjuk:

```
.....................................
RUN
?
```

ahol a kérdőjel utáni második pozícióban villog a kurzor. Ide várja az aktuális adatokat. Ha megnézzük a programlistát, akkor ez a 20-as sor következtében került a képernyőre. E két információ segít a helyes válasz megadásában; ti., hogy 3 adatot, mégpedig a hosszúságot, a szélességet és a mélységet várja tőlünk a C-64. Válaszoljunk ennek megfelelően 3 db vesszővel elválasztott 1-sel a következő formában:

```
.....................................
RUN
? 1,1,1 RETURN
```

Ha ezt megtesszük, akkor a következő eredményt kapjuk:

```
.....................................
RUN
? 1,1,1
  1

READY.
```

Ha feladatunk összetettebb lenne, akkor újabb próbafuttatásokat javasolnánk. Mivel feladatunk nagyon egyszerű, ezért elegendő egy próbafuttatás. Ezekután térjünk át Pistike problémájának, majd sorban a többi hasonló problémának a megoldására.

Pistike példája:

Mérje meg Pistike a medence méreteit. Ezek rendre 13,7 és 5 dm. Ezután hajtassuk végre - a próbafuttatáshoz hasonlóan - a programot Pistike adataival.

Eredményként a képernyő utolsó néhány sora a következőket fogja tartalmazni:

```
.....................................
RUN
? 13,7,5
 455

READY.
```

A kertészkedő nyugdíjas példája:

Józsi bácsi 30 cm vastag termőtalajjal akarja beteríteni frissen vásárolt telkét, mely 30 m széles és 200 m hosszú. Ekkor a végrehajtás után a következő képernyőrészletet kapjuk:

```
.....................................
RUN
? 200,30,0.3
 1800

READY.
```

A betonozó unokafivér példája:

Unokafivérünk először a 4.5 x 4.5 m-es nagyszobát akarja lebetonozni 10 cm vastagon. Ezt a következőképpen oldhatjuk meg:

```
.....................................
RUN
? 4.5,4.5,0.1
 2.025

READY.
```

Ha átnézzük az eredményeket, akkor a következő közös vonásokat fedezhetjük fel:

- Mindegyik végrehajtás 5 sorból áll.
- Mindig a második és harmadik sor tartalmazza a lényeges információkat.
- A második sor a kérdőjel utáni második pozíciótól kezdve tartalmazza a 3 bemenő adatot: mégpedig hosszúság, szélesség, mélység (magasság) sorrendben.
- A hosszúságot a szélességtől, a szélességet a mélységtől (magasságtól) **vessző** választja el.
- Ha valamelyik adatunk tizedes jegyet is tartalmaz, akkor az egész értékeket a törtektől **tizedespont** választja el.
- Pistike példájánál minden induló adatot dm-ben, a többi példában minden adatot m-ben adtunk meg. Így a végeredmény az első esetben dm^3 azaz **l**, az összes többi esetben pedig m^3 lesz.

Foglaljuk össze teendőinket:
- gyűjtsük össze a szükséges induló adatokat,
- indítsuk el programunkat,
- válaszoljunk a C-64 egyszerű kérdéseire (a programlista segítségével, amely megmondja, hogy mely változókat, milyen típusú értékeket vár) vagy az ún. bonyolult kérdésekre (amelyek a kérdést is kiírják),
- a képernyőre kiírt eredményeket írjuk le magunknak a későbbi dokumentálás érdekében.

2.6 A feladat megoldásának értékelése, dokumentálása

Újfent az esetleg türelmetlen felhasználók türelmét kérjük. Példafeladataink nagyon egyszerűek,s ezért kicsit erőltetettnek tűnnek már az eddig mondottak is. Gondoljunk arra, hogy bonyolultabb feladat esetén majd jól fogjuk hasznosítani a mostani megfigyeléseink eredményeit.

Tételezzük fel, hogy 5-soros programunkat PISTIKE néven a TOROKBALINT nevű lemezen tároltuk el.

Először írjuk ide az előbbi példák egy lehetséges dokumentálását:

Pistike példája:

Hány l hidegvizet kell Pistikének az udvaron levő medencébe kihordania ahhoz, hogy tele legyen?

A medence hosszúsága 13 dm.

A medence szélessége 7 dm.

A medence mélysége 5 dm.

A medence térfogata 455 liter.

A térfogatot a TOROKBALINT nevű lemezen tárolt PISTIKE nevű program segítségével számítottuk ki.

A kertészkedő nyugdíjas példája:

Hány m^3 termőtalajt kell Józsi bácsinak vásárolnia ahhoz, hogy eredményesen kertészkedhessen öreg napjaiban?

Józsi bácsi telkének hossza 200 m.

Józsi bácsi telkének szélessége 30 m.

Javasolt termőtalaj-vastagság 0.3 m.

Józsi bácsinak 1800 m^3 termőföldet kell hozatnia.

Feladatunkat a TOROKBALINT lemezen tárolt PISTIKE program segítségével oldottuk meg. A program végrehajtása során az alapadatokat m-ben adtuk meg és a mélység helyett magasságot helyettesítettünk.

SZÁMÍTÁSTECHNIKAI MUNKAFOLYAMAT

A betonozó unokafivér példája:

Hány m^3 előregyártott betont kell unokafivérünknek vásárolnia ahhoz, hogy nagyszobáját 10 cm vastagon "megalapozza"?

A nagyszoba hosszúsága 4.5 m.

A nagyszoba szélessége 4.5 m.

A megkívánt betonvastagság 0.1 m.

A megvásárolandó beton 2.025 m^3.

Feladatunkat a TOROKBALINT lemezen tárolt PISTIKE program segítségével oldottuk meg.

Feladatértékelésünk, dokumentálásunk túlzottnak tűnhet. Gondoljunk egy bonyolultabb, felelősségigényesebb feladatra! Rögtön belátjuk a pontosság, a dokumentálás fontosságát. Megjegyezzük, hogy az esetek többségében ajánlatos pontos dokumentációt készíteni az elkészült programról is - valahogy így -:

Program neve: PISTIKE

Programozási nyelv: BASIC

Tárolási hely: TOROKBALINT nevű lemez.

Alkalmazott eljárás: téglatest térfogata.

Szükséges adatok: a téglatest hossza, szélessége, magassága.

Szolgáltatott eredmény: térfogat.

Adatbevitel formája: hosszúság, szélesség, magasság - mint 3 vesszővel elválasztott számadat beírása.

Adattípus: lebegőpontos (valós) változók.

Eredményközlés: egyszerű - a bemenő adatok mértékegységével összhangban levő - szám.

Ennek ismeretében bárki célszerűen tudja használni a PISTIKE nevű programot.

3. RÖVIDEN A C-64-RŐL

Az eddig elmondottak alapján már el tudjuk képzelni azt, hogy hogyan fest egy számítástechnikai munkafolyamat,s azt is, hogy hogyan kell kezelnünk egyszerű esetekben a C-64-et. Ahhoz, hogy bonyolultabb, összetettebb feladatokat megoldhassunk,mélyebb ismeretekre van szükségünk. Ezt a célt szolgálja az előttünk álló fejezet.

Két területet kell részletesen áttekintenünk: az információábrázolás rendszerét, valamint az alapvető programok működési elvét.

Az elsőre azért lesz szükségünk, hogy értsük és helyesen éljünk a C-64 információábrázolási és feldolgozási lehetőségeivel, illetve korlátaival. Ezt a területet hívják a számítástechnikusok az angol szakirodalom nyomán **hardware**-nek.

A másodikra azért lesz szükségünk, hogy értsük, hogyan dolgozik a C-64. Ez az ismeret fogja elősegíteni azt, hogy mindig helyesen, céljainknak megfelelően adjuk ki parancsainkat. Itt fogjuk ismertetni a mindig, illetve nagyon gyakran használt programok működési elvét. Ezt a területet az angol szakirodalom nyomán **software**-nek fogjuk hívni.

3.1 Hardware alapismeretek

Ha bármilyen eszközt vagy tárgyat használunk, akkor tudjuk legcélszerűbben alkalmazni, ha ismerjük méreteit, lehetőségeit. A C-64 ránézésre akkora, mint egy táskaírógép. Ez is egyfajta méret, amit jó tudni ahhoz, hogy mekkora helyet igényel a munkaasztalon. A felhasználó szempontjából azonban fontosabbak azok a szemmel nem látható méreti paraméterek, amelyek a vele elvégezhető munka nagyságrendjét szabják meg. Ez a méret a hasznos memóriatartalom, amelyet programok és adatok, információk tárolására használhatunk fel. Ahhoz, hogy ezt a fogalmat pontosan érzékeljük,néhány a köznapi ember számára szokatlan fogalmat kell megismernünk. Előző fejezetünkben nem véletlenül taglaltuk Pistike példáját olyan részletesen. Most is ezt vesszük elő az analógia kedvéért. Pistike néhány dolgot tudott (pl. számtani műveleteket fejben elvégezni, írni, olvasni. Ezeket valamikor megtanulta és tudását a fejében tárolta), a feladathoz tartozó összes többi információt leírta egy papírlapra. Pontosan ezt a felosztási rendszert követi a C-64 is. Vannak dolgok (információk, eljárások), amelyet állandóan tud C-64-gyesünk. S vannak amelyeket esetről-esetre tudatosítani kell vele. Ennek a gondolatmenetnek felel meg az a memóriafelosztás, melyben Pistike fejének felel meg a csak olvasható ROM (Read Only Memory) memóriaterület, s Pistike papírlapjának az írásra-olvasásra egyaránt alkalmas RAM (Random Access Memory) memóriaterület. A terület funkcionális felosztása is hasonló a Pistike példájában elmondottakhoz. A ROM tartalmazza az állandóan tudott dolgokat, a RAM pedig az aktuálisan szükséges feladatokat, információkat. Az analógiának lassan búcsút kell mondanunk, hiszen példabeli Pistikénk tudása,s aktuálisan használt papírszeletkéje mindenképpen esetleges méretű. Ha objektív ismeretekre vágyunk, akkor felejtsük el az analógia szubjektív részét, s csak a lényegét, a kétféle hozzáférési, feldolgozási lehetőséget jegyezzük meg. Pistikénél a papírlap mérete s Pistike információbefogadó képessége például nagyban függött attól, hogy mennyire figyelt első osztályos korában

a tanító néni magyarázataira,s hogy mekkora betűkkel, számokkal írja tele papírlapját. Ez emberi felhasználásnál természetes dolog. Ha számítógépekről beszélünk, akkor ilyen szubjektivitást nem engedhetünk meg magunknak.

Ezekután képzeljük el a C-64-et olyan két papírlapként, amelyen kezdettől fogva rögzített az információtárolás rendszere mind méretben, mind koncepcióban. Ezután egy olyan információtárolási koncepciót kerestek,amely megfelel mind a számítástechnikai, mind az elektrotechnikai kívánalmaknak. Ez a rendszer a két különböző elektromos állapot s a neki megfeleltethető 2-es számrendszer. Ezt a koncepciót alkalmazzák az elektronikus számítástechnika kialakulása óta. Az egyik elektromos állapotnak megfeleltetjük a 0-t, a másiknak pedig az 1-t.S ily módon minden kettes számrendszerben leírható információ megvalósítható elektronikus számítógépen. Az ezt realizáló információs egységet nevezzük a számítástechnikában **bit**-nek, mely a számítástechnikusok körében az információábrázolás elemi részecskéje, "atom"-ja, melyet 0-1-értéklehetőséggel szerepeltetünk.

Mivel a legegyszerűbb információs egység is nagyon hosszan elnyúlik a 2-es számrendszerbeli ábrázolás során, ezért a számítástechnikában bevett szokás négyesével összeolvasni az információsorozatokat, mely ezáltal tizenhatos számrendszerbeli számmá alakul át. A számítógépek döntő többsége ebben a számrendszerben közli a felhasználóval a memóriatartalmat. Ezért érdemes részletesebben megismerkedni vele.

3.1.1 Számrendszerek

Amikor az ősember számolt, kezét-lábát igénybevette,s ha a tíz ujja kevésnek bizonyult ezt mindig a barlang falán vagy a homokban jelezte. Valószínűleg ezért alakult ki az egész világon napjainkra a tizes számrendszer,hiszen a tizes váltás ősidőktől kezdve a vérünkben van. Nézzük meg közelebbről az aktuális évszámot, 1986-ot. Ezt az ősember a következőképpen tudta volna azonosítani:

1 nagy rovátka a barlangfalon (mely egyenként 10 közepes rovátkát ér)
9 közepes rovátka a barlangfalon (mely egyenként 10 kis rovátkát ér)
8 kis rovátka a barlangfalon (mely egyenként 10 ujjat ér)
6 ujj (mely egyenként egy évet ér)

Az iskolát járt gyerek ugyanezt így tanulta: 1 db ezresem, 9 db százasom, 8 db tizesem s 6 db egyesem van. Ha megfigyeljük, az ősember egyre nagyobb csomagokban, az iskolás gyerek pedig egyre nagyobb 10-hatványokban (ezer=10^3, száz=10^2, tíz=10^1, egy=10^0) gondolkodik, akkor kézenfekvő az ötlet, hogy bármilyen számrendszerben a felírás alapja az illető számrendszer alapjának hatványai (0-ik, 1-ső 2-dik, 3-dik, ...) s ezeket az ábrázolni kívánt szám értékének megfelelő szorzókkal kell megszorozni. Évszámos példánkban ez így fest:

```
1*10↑3+9*10↑2+8*10↑1+6*10↑0
```

Ehhez hasonló konstrukciót kell végrehajtanunk akármilyen számrendszer esetében. Mielőtt ezt megtennénk, próbáljuk meg megtalálni azt az eljárást,algoritmust, amely bármilyen számrendszer esetében segít a legutóbbihoz hasonló hatványalak feltalálásában.

RÖVIDEN A C-64-RŐL

Próbálkozzunk a következő eljárással: - írjuk fel az ábrázolandó szám abszolút értékét (előjel nélküli értékét) egy üres papírlap bal felső részébe és tőle jobbra húzzunk egy függőleges egyenest:

```
1986 I
     I
     I
     I
     I
     I
```

Példánkban 1986 jelöli az átalakítani kívánt számot, az I-k pedig a függőleges egyenest. Eljárásunkat először a tizes számrendszerrel próbáljuk ki, hogy az eredmény nagyon könnyen ellenőrizhető legyen. Javaslatunk a következő:

- a bal oldali oszlop utolsó elemét osszuk el az új számrendszer alapjával (esetünkben 10-zel),
- a hányados egész részét írjuk az osztandó alá,
- a maradékot pedig az elosztott szám mellé a függőleges egyenes jobb oldalára.

Ha ezt végrehajtjuk, akkor a következő eredményt kapjuk:

```
1986 I 6
 198 I
     I
     I
     I
```

Folytassuk az előző osztást mindaddig, amíg az osztandó el nem fogy. Ha ezt megtesszük, akkor a következő táblázatot kapjuk:

```
1986 I 6
 198 I 8
  19 I 9
   1 I 1
     I
```

Ha a kapott jobboldali oszlopot alulról felfelé olvassuk el, akkor egyrészt visszakapjuk az ábrázolni kívánt számot a 10-es számrendszerben, másrészt - ami ezzel egyenértékű - megkapjuk a 10-hatványok együtthatóit. Próbálkozzunk a 2-es számrendszerben ugyanezzel a számmal és ugyanezzel az eljárással. Ekkor a következő eredményt kapjuk:

```
1986 I 0
 993 I 1
 496 I 0
 248 I 0
 124 I 0
  62 I 0
  31 I 1
  15 I 1
   7 I 1
   3 I 1
   1 I 1
     I
```

Ha a jobboldali eredményt alulról felfelé elolvassuk, akkor a következőt kapjuk:

11111000010(2)

Ha az előző analógiával élünk, akkor ez a következőt jelenti:

1*2↑10+1*2↑9+1*2↑8+1*2↑7+1*2↑6+0*2↑5+0*2↑4+0*2↑3+0*2↑2+1*2↑1+0*2↑0

Ha mindegyik kijelölt hatványozást kiszámítjuk, akkor a következő összeget kapjuk:

1024+512+256+128+64+2=1986

Tehát az eljárás valóban szolgáltatta az 1986 2-hatványokkal történő felírását vagyis a 2-es számrendszerbeli számjegyeket. Erre utal az algoritmus utáni zárt forma zárójelbeli 2-ese is. Mostmár tudjuk oda-vissza olvasni a számokat a 10-es és a 2-es számrendszer között, de ez nagyobb számértékek esetében nagyon munkaigényes,s bizony nagyon-nagy tévesztési lehetőséggel jár. Ezért használjuk helyette a 4-esével összeolvasott formát a 16-os számrendszert. Ezt két úton fogjuk megközelíteni: a 16-tal való osztássorozat segítségével, valamint az összeolvasásos formával. Kezdjük az osztással:

```
1986 I 02
 124 I 12
   7 I 07
     I
```

Ha felírjuk a lassan már megszokott hatványformát, akkor a következőt kapjuk:

7*16↑2+12*16↑1+2*16↑0

RÖVIDEN A C-64-RŐL

Ha a 10-es és 2-es számrendszerben megszokott módon egymás mellé írnánk a hatványegyütthatókat, akkor a következő két lehetőség áll rendelkezésünkre:

071202(16) vagy 7122(16)

Egyik megoldás sem praktikus, mert az első feleslegesen hosszú - a dupla helyfoglalás miatt -, a másik pedig a rendszertelen elhagyások miatt nem egy-egyértelmű megfeleltetés. Ennek kiküszöbölésére a 10-es számrendszer számjegyeit kiegészítették az abc első néhány betűjével a következő módon:

Számérték	16-os számrendszerben
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Így már röviden és egyértelműen felírhatjuk előbbi eredményünket:

7C2(16)

Talán ahhoz a következtetéshez sem kell külön magyarázat, hogy ez tényleg 1986-nak felel meg.

Utolsó észrevételünk, hogy minden számrendszerben az utolsó használt számjegy a számrendszer alapja-1,azaz

2-es számrendszer: 0,1
10-es számrendszer: 0,1,2,3,4,5,6,7,8,9
16-os számrendszer: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Könnyen látható az is, hogy a használt számjegyek száma megegyezik a számrendszer névadójával.

16-tal osztani nagy számok esetében nehézkes, ezért éltek (élünk) a 2-es számrendszer kínálta lehetőséggel. Az ötlet lényege az a megfigyelés, hogy 4 db 2-es számrendszerbeli számjeggyel 2^4 lehetőséget vagyis éppen 16-ot írhatunk fel.

Válasszuk a következő hozzárendelési rendszert:

Számok ábrázolása a 10-es számrendszerben	2-es	16-os
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

A megfeleltetés alkalmazására tekintsük az 1986 2-es számrendszerbeli képét hátulról előre 4-esével csoportosítva. Ha megtesszük, a következőt Kapjuk:

0111 1100 0010(2)

Ha az előző táblázat megfeleltetéseit alkalmazzuk, akkor a következő eredményt kapjuk:

7 C 2

Nagyon könnyű a megfeleltetés és a megfeleltetés eredményének helyességét belátni. A 16-os számrendszerre Úry és a személyi számítógépes szakirodalom bevezető $-jellel hívatkozik. Tegyük ezt mi is!

Tehát:

1986 = 11111000010(2) = $7C2

Az, hogy a számrendszerekkel ilyen sokat foglalkoztunk, ebben a pillanatban felesleges kitérőnek tűnhet. A közeli felhasználás majd hamarosan igazolni fogja a kitérő jogos voltát.

3.1.2 Tártérkép

Már tudunk számokat ábrázolni a különböző számrendszerekben. Tudjuk, hogy a számítógépek legkisebb ábrázolási egysége a 2-es számrendszerbeli számjegynek megfelelő **bit**. Tudjuk, hogy ezeket 4-esével összeolvasva érdemes kezelni. Így kaptuk a 16-os számrendszert. Szinte valamennyi számítógép

RÖVIDEN A C-64-RŐL

- így a C-64 is - két 16-os számrendszerbeli számjegyet kezel együtt, mely 8 **bit**-nek 8 2-es számrendszerbeli számnak felel meg. Ezt az egységet nevezi a nemzetközi számítástechnikai szakirodalom **byte**-nak (ejtsd bájt). Szinte valamennyi információtárolási kapacitást ebben számolunk, jelölünk. Ez 256 különböző információlehetőséget jelent, amely az átlagos igényeket bőségesen kielégíti.

Mielőtt hozzákezdenénk a konkrét információtárolási adatok ismertetéséhez még egy fogalommal kell megismerkednünk. A számítástechnikában a **kilo** előtag mást jelent, mint a hétköznapi életben. Nevezetesen a következőt:

1 **kilo** = 1024 = 2↑10

Ezekután, ha azt írjuk, hogy a C-64 ROM-memóriája 20 kilobyte, akkor senkinek nem okozhat problémát, hogy ez 20 * 1024 byte-ot azaz 20 * 1024 * 8 bitet jelent, mely területről a felhasználó **csak olvashat.** Ezekután gondoljuk meg, mit jelent a RAM-memória 64 kilobyte-ja.

Jó szokásához híven a C-64 is 2-es (illetve 16-os) számrendszerben számol. Így memóriájának minden egyes byte-ját a 16-os számrendszerben számozza meg 0-tól 65535-ig. Szemléltetésül néhány példa a memóriacímzésre:

```
    0. byte = $0000
    1. byte = $0001
   16. byte = $0010
 1024. byte = $0400
 2023. byte = $07E7
```

Mindezekre azért van szükségünk, hogy teljes képet nyújthassunk a memóriafelosztás iránt mélyebben érdeklődő Olvasónak.

A RAM-memória első elérhető byte-ja (címe) a $0000, az utolsó pedig a $FFFF. Ha ezeket az értékeket, mint 16-os számrendszerben felírt számokat visszaolvassuk 10-es számrendszerbe, akkor 0-t és 65535-öt kapunk.

Ahhoz, hogy helyesen gazdálkodjunk a rendelkezésünkre álló memória-területtel,s értsük a C-64 működési elvét, meg kell ismernünk a C-64 vázlatos tárfelhasználását. Igérjük, hogy mellőzzük a kezdő felhasználót nagyon megzavaró részleteket.

ROM-memória-terület:

Mivel a felhasználó csak a benne tárolt információk, eljárások előnyeit élvezi, bőségesen elegendő annyit tudni róla, hogy 20 kilobyte terjedelmű,ami azt jelenti, hogy első byte-ja a $0000, az utolsó pedig a $4FFF címet viseli.

RAM-memória-terület:

Ezt a területet használja a felhasználó. Erről a területről mondja a C-64 bejelentkezéskor, hogy 64 kilobyte-nyi RAM-területemből 38911 byte áll a felhasználó rendelkezésére. Ebből a 38911 byte-ból "gazdálkodik" Pistike, amikor a fürdőmedence térfogatát számítja ki,papírlapon mindent feljegyezve.

Miért nem használhatja a teljes 64 kilobyte-ot?

Azért, mert a C-64 eredményes működéséhez nagyon sok kiegészítő információ tárolására van szükség. Hiába van ROM-területünk, az nem elegendő a teljes működtetéshez.

A RAM-terület felosztása elhelyezkedés és felhasználás szerint
(A táblázat forrása Úry László Commodore 64 I-II c. programozási segédlete)

Memóriaterület	Felhasználás
$0000-$03FF	BASIC-rendszerváltozók
$0400-$07E7	Aktuális képernyőtartalom kódolt formában
$07E8-$07FF	Sprite- és egyéb mutatók
$0800-$9FFF	BASIC-munkaterület (ez indul 38911 byte-tal)
$A000-$BFFF	BASIC ROM kiegészítő
$D000-$DFFF	CHARACTER ROM kiegészítő
$E000-$FFFF	KERNAL ROM kiegészítő

Memóriatérképünk igazán vázlatos volt, de így is jól érzékeltette a C-64 felhasználási módok szerinti tárterület-elosztását. Ebből számunkra egyenlőre csak a BASIC-munkaterület aránya a lényeges, mely elég tekintélyes, de mégiscsak rész. Ezen a területen kell elhelyeznünk, illetve elhelyeztetnünk valamennyi számunkra lényeges utasítást, adatot, részeredményt, eredményt, információt.

Ennek felhasználási rendszerével foglalkozik következő szakaszunk.

3.1.3 Utasítások, változók, tömbök

Ha a programozó rendelkezésére álló terület két végértékét kivonjuk egymásból, akkor 38912-t kapunk, amely majdnem a bejelentkezéskor kiírt érték. Az egy byte-nyi eltérést hamarosan megértjük. Mint már mondottuk, ezen a területen helyezi el a C-64 a felhasználó által megadott valamennyi információt. Ennek elhelyezési rendszere a következő:

- program (utasítás-sorozat) területe,
- változók területe,
- tömbök területe,
- szabad memória-terület,
- karaktersorozatok területe.

Nyílvánvaló, hogy minden program különböző számú utasításból áll, s az sem valószínű, hogy mindig ugyanannyi változóra, tömbre, karaktersorozatra van szüksége az egyes programoknak. Mivel ezek különböző, változó értékek, ezért ezek aktuális értékét időről-időre ismernünk kell. Így merült fel olyan segédváltozók (rekeszek) alkalmazása, amelyek az aktuális helyzetet leírják. Mivel a C-64 64 kilobyte területtel rendelkezik, ezért egy-egy rekesz kezdetének megjelöléséhez 4 16-os számrendszerbeli számra, azaz 2 byte-ra van szükség. Bár az eddigiek alapján is nyilvánvaló, de azért leszögezzük, hogy valamennyi byte egy-egy 16-os számrendszerbeli számmal (a memóriacímmel) azonosítható. Ezeket a 4-jegyű számokat 2 byte-on tároljuk felcserélt sorrendben. Ennek oka talán az, hogy gyakrabban kell dolgoznunk

az elől álló kisebb helyiértékű számokkal. A tényleges értéket úgy számíthatjuk ki, ha az első byte (alsó byte) értékéhez hozzáadjuk a második byte (felső byte) értékének 256-szorosát. Ha majd szükségünk lesz rá, mutatunk rá példát. A BASIC munkaterület elhelyezkedése és felhasználása a következő:

Információhordozó memóriacímek 10-es számrendszerben	16-os	Információtartalom
43-44	$002B-$002C	BASIC-program kezdete (általában $0801)
45-46	$002D-$002E	változóterület kezdete
47-48	$002F-$0030	tömbterület kezdete
49-50	$0031-$0032	szabad memória-terület kezdete
51-52	$0033-$0034	karaktersorozat-terület kezdete
53-54	$0035-$0036	karaktersorozat-terület kezdete
55-56	$0037-$0038	BASIC-terület vége (általában $9FFF)

Ha begépeljük Pistike programját, akkor a következő értékeket olvashatjuk ki az egyes rekeszekből:

Programkezdet:	$0801
Változóterület:	$0841
Tömbök területe:	$085D
Szabad terület:	$085D
Karakter-terület:	$A000

Irassuk ki a $0801-$085C-terület tartalmát byte-onként 10-es számrendszerben. Mielőtt ezt megtennénk, hajtsuk végre a programot Pistike adataival, tehát a 13, 5, 7 adatsorral. A lista eredménye a következő lesz:

```
$0801   23    8   10    0 143   32   80   73
$0809   83   84   73   75   69   32   80   69
$0811   76   68   65   74   65    0   35    8
$0819   20    0  133   32   72   44   83   44
$0821   77    0   49    8   30    0  136   32
$0829   84  178   72  172   83  172   77    0
$0831   57    8   40    0  153   32   84    0
$0839   63    8   50    0  128    0    0    0
$0841   72    0  132   80    0    0    0   83
$0849    0  131   32    0    0    0   77    0
$0851  131   96    0    0    0   84    0  137
$0859   99  128    0    0
```

A programban 5 utasítás és 4 változó szerepel. Keressük meg ezeket! Hagyatkozhatnánk az Olvasó találékonyságára, de eltekintünk ettől. Azt tudjuk,

hogy a program utasításai $0801-től $0840-ig tartanak, mely éppen a $0839-jelű sor 8. elemével zárul. Ha felismerjük a rendszert, akkor a következő átrendezett listát kapjuk:

```
23-8-10-0-143-32-80-73-83-84-73-75-69-32-80-69-76-68-65-74-65-0
35-8-20-0-133-32-72-44-83-44-77-0
49-8-30-0-136-32-84-178-72-172-83-172-77-0
57-8-40-0-153-32-84-0
63-8-50-0-128-0-0-0
```

Minden sor harmadik száma megadja az utasítás sorszámát. Minden negyedik - valószínűleg - a hozzátartozó kisérő ún. felső byte, mely itt mindig 0. Ha ezt a visszafelé olvasást követjük a sor elején is, akkor rendre a következő értékeket kapjuk:

```
1. sor: 8 * 256 + 23 = 2071 = $0817
2. sor: 8 * 256 + 35 = 2083 = $0823
3. sor: 8 * 256 + 49 = 2097 = $0831
4. sor: 8 * 256 + 57 = 2105 = $0839
5. sor: 8 * 256 + 63 = 2111 = $083F
```

Ha memórialistánkban megnézzük a $0817, $0823, $0831, $0839 és $083F byte-ok tartalmát, akkor azt tapasztaljuk, hogy az és a következő byte mondja meg, hogy az éppen elkezdett utasítás meddig tart (melyik byte-on kezdődik a következő). Ez nagymértékben megkönnyíti a BASIC-interpreter (program-értelmező) dolgát. Ezután azt fedezhetjük fel, hogy az utolsó sort leszámítva valamennyi sor egyetlen 0-tartalmú félbyte-tal végződik, az utolsó pedig 3 darabbal. Nyilván ez is a BASIC-interpreter munkáját könnyítő információ. Ezután vizsgáljuk meg az egyes sorokban szereplő még meg nem fejtett számok számát! Ekkor a következő számsort kapjuk:

```
1. sor: 17 szám
2. sor:  7 szám
3. sor:  9 szám
4. sor:  3 szám
5. sor:  1 szám
```

Legfeltűnőbb az utolsó sorbeli 1 árválkodása. Ez a számjegy nyilván az END utasításnak felel meg. Vigyük tovább az analógiát, azaz, hogy minden BASIC-kulcsszónak egyetlen szám felel meg. Ekkor az utasítás paramétereire a következő számok (kódok) maradnak:

```
1. sor: 16 szám
2. sor:  6 szám
3. sor:  8 szám
4. sor:  2 szám
5. sor:  0 szám
```

Arra gondolhatnánk, hogy az egyes számok(kódok) az egyes paramétereket helyettesítik, de számuk ennél lényegesen több, illetve kevesebb. Így hát új analógia (párhuzam) után kell néznünk. Számoljuk meg, hogy Pistike hány billentyűt nyomott meg egy-egy sor paraméterei érdekében. Álljon itt a programlista aláhúzott paraméterlistával:

```
10 REM PISTIKE PELDAJA
   ----------------------
20 INPUT H,S,M
       -------
30 LET T=H*S*M
      ---------
40 PRINT T
       ---
50 END
```

Az első sorban 16, a második sorban 6, a harmadik sorban 8, a negyedik sorban pedig 2 billentyű megnyomása kellett a paraméterek felírásához. Tehát kézenfekvő a következtetés, hogy a BASIC az utasításokat kódjukkal (ún. **token**-ükkel) helyettesíti, míg a paramétereket "betűről-betűre" bekódolja, beírja a memóriájába. Illusztrációként írjuk fel párhuzamosan mindkét listát: felül a C-64-belit, alul pedig a Pistike által írottat:

(Hogy jobban érzékelhessük az analógiát, a C-64 tartalmát 3 sorban egymás alatt (100-10-1 helyiértékkel írjuk):

```
C         1
- 2 1     4     3878877638676676
6 3800    3     203343592096854 50
4

P 10    REM    PISTIKE PELDAJA

C         1
- 3 2     3     374847
6 5800    3     2243470
4

P 20    INPUT H,S,M

C         1       1 1 1
- 4 3     3     38777877
6 9800    6     248223270
4

P 30    LET    T=H*S*M

C         1
- 5 4     5     38
6 7800    3     240
4

P 40    PRINT T
```

```
C        1
- 6 5    2
6 3800   8    000
4

P 50     END
```

Most már ráéreztünk az utasítások ábrázolási(tárolási) módjára. Bár korántsem tekintettük át valamennyi lehetőséget, lépjünk tovább. Ebben a programban 4 változó szerepel H, S, M, T előfordulási sorrendben. A változók tárolására $085D-$0841, azaz 28 byte memóriaterületet használ fel a C-64. Ebből az következik, hogy mindegyik változó 7-7 byte-ot foglal el. Az előfordulási sorrend alapján állítsuk őket párba:

Változó neve	Aktuális értéke	Neki megfeleltetni kívánt memóriabeli szám-hetes
H	13	72-0-132-80-0-0-0
S	5	83-0-131-32-0-0-0
M	7	77-0-131-96-0-0-0
T	455	84-0-137-99-128-0-0

A hét szám közül az első a programban is pontosan a változók megjelölésére szolgált. A második szám is nyilván erre szolgálhat, hiszen mind a négy esetben 0-értékű. Már csak a hátralevő 5-5 számjegy jelentését kell megfejtenünk. Mielőtt ezzel tovább foglalkoznánk, írjuk át 2-es számrendszerbe a négy ábrázolt számértéket (13,5,7,455):

```
13 | 1      5 | 1      7 | 1      455 | 1
 6 | 0      2 | 0      3 | 1      227 | 1
 3 | 1      1 | 1      1 | 1      113 | 1
 1 | 1        |          |        56 | 0
   |                              28 | 0
                                  14 | 0
                                   7 | 1
                                   3 | 1
                                   1 | 1
                                     |
```

Ezek alapján H=1101(2), S=101(2), M=111(2) és T=111000111(2) lesz. Következő lépésként normáljuk őket. Ekkor a következő eredményt kapjuk:

H=0.1101*2↑4
S=0.101*2↑3
M=0.111*2↑3
T=0.111000111*2↑9

Ha mindegyik kitevőhöz hozzáadunk 128-at, akkor rendre 132,131,131,137 adódik, mint a változórekeszek 3-ik byte-jaiban. Most már csak 4 byte tartalmát kell azonosítani. Ehhez írjuk fel a maradék részeket páronként - kibővítve a normál alakot elegendő számú 0-val (a sor elején 2-es számrendszerbeli, az egyenlőségjel után 10-es számrendszerbeli számjegyek szerepelnek):

```
11010000-00000000-00000000-00000000-00000000 = 80-0-0-0
10100000-00000000-00000000-00000000-00000000 = 32-0-0-0
11100000-00000000-00000000-00000000-00000000 = 96-0-0-0
11100011-10000000-00000000-00000000-00000000 = 99-128-0-0
```

A sorok elején szereplő vastag 1-esek, mivel a normál alak velejárói, nem vesznek részt a megfeleltetésben. Ha ezektől eltekintünk, akkor a következő megfeleltetéseket tehetjük:

```
01010000(2) = 80
00000000(2) = 0
00100000(2) = 32
01100000(2) = 96
01100011(2) = 99
10000000(2) = 128
```

Ezen analógiák alapján nyilvánvaló, hogy a C-64 minden valós szám ábrázolására kijelölt változót a következőképpen helyez el a memóriában:

- megkeresi a program utáni első szabad helyet (ha már használtunk változót, akkor az azutáni első szabad helyet),
- legfoglal 7 byte-ot a változó számára,
- az első két byte-ba beírja a változó azonosítójának (nevének) első két karakterét ASCII-kódban (lásd később!),
- kiszámítja a tárolni kívánt érték 2-es számrendszerbeli normál alakját,
- a kitevőhöz hozzáad 128-at s ezt beírja a 3. byte-ba,
- a mantissza első (mindig 1) jegyét elhagyja,s helyébe írja az előjelet szimbolizáló 0-t vagy 1-et (+ illetve -),s a kapott számot hátulról kiegészíti vagy lerövidíti 32 **bináris** (2-es számrendszerbeli) számjegyre,
- az így kapott számot 8-asával elhelyezi a fennmaradó 4 byte-ba.

Mielőtt tovább mennénk,ismerkedjünk meg a valódi valós számok ábrázolási módjával is. Legyen segítségünkre unokafivérünk betonozási problémája. Ekkor

H=4.5 S=4.5 M=0.1 T=2.025

Hogyan ábrázolja ezeket az értékeket a C-64. Ha Pistike programját az ellenőrző memórianyomtatásokkal együtt végrehajtjuk, akkor a következő eredményeket kapjuk:

```
H=131-16-0-0-0
S=131-16-0-0-0
M=125-76-204-204-205
T=130-1-153-153-154
```

Próbáljuk meg eddigi módszerünket alkalmazni és továbbfejleszteni. Az egész részek átalakítása a következő eljárás eredménye:

```
4 I 0       0 I 0       2 I 0
2 I 0         I         1 I 1
1 I 1                     I
  I
```

Írjuk fel az ábrázolni kívánt törtrészeket az egészrészekhez hasonló módon. Ekkor a következő induló táblázatot nyerjük:

```
.5 I          .1 I          .025 I
   I             I                I
   I             I                I
```

Fordítsuk meg az eljárást osztás helyett szorozzunk 2-vel, s az eredmény tört részét a szám alá, az egész részét a túloldalra írjuk. Így:

```
.5 I .        .1 I .        .025 I .
.0 I 1        .2 I 0        .050 I 0
   I             I                I
```

Folytassuk az eljárást mindaddig, amíg:

- a baloldalon .0 nem adódik,
- a jobboldalon ciklikusan ismétlődő szakaszt nem kapunk,
- elérjük a 32. számjegyet.

```
.5 I .        .1 I .        .025 I .
.0 I 1        .2 I 0        .050 I 0
   I          .4 I 0        .100 I 0
              .8 I 0        .200 I 0
              .6 I 1        .400 I 0
              .2 I 1        .800 I 0
              .4 I 0        .600 I 1
   .          .8 I 0        .200 I 1
              .6 I 1        .400 I 0
```

Az első törtrész értéke 0.1(2), a második pedig 0.0001**1001**(2), a harmadik pedig 0.0000**011**0(2), ahol a második és harmadik tört vastagon szedett része ciklikusan ismétlődik. Írjuk fel ezekután adatinkat és azok 2-es számrendszerbeli teljes alakját:

4.5 = 100.1(2) = 0.1001*2↑3

0.1 = 0.00011001(2) = 0.11001*2↑(-3)

2.025 = 10.00000110(2) = 0.10000011O*2↑2

Ha a 128-at hozzáadjuk a kitevőkhöz, akkor rendre 131, 125 és 130 adódik, a mantissza-sorozatok pedig a következők:

10010000-00000000-00000000-00000000 = 16-0-0-0
11001100-11001100-11001100-11001100 = 76-204-204-205
10000001-10011001-10011001-10011001 = 1-153-153-154

Csak a következő analógiákat kell belátnunk:

00010000(2) = 16
01001100(2) = 76
00000001(2) = 1
00000000(2) = 0
11001100(2) = 204
10011001(2) = 153

Pusztán az utolsó byte-ok tartalma nem stimmel, de ezeket az utánuk elhagyott 1-es érték indokolja.

Ehhez hasonló elgondolások alapján, de egyszerűbb rendszerben ábrázolja a C-64 az egész számokat. Itt is 7 -byte-ot foglal le egy-egy egész típusú változó részére, melyet a programban %-ra végződő változónév jelöl, s ezt a 7 byte-ot a következőképpen hasznosítja:

1. byte: a változó nevének első karaktere+128,
2. byte: a változó nevének második karaktere+128,
3-4. byte: bináris (2-es) számrendszerben ábrázolt számérték. Negatív szám esetén a számot hozzáadjuk 65536-hoz és ezt ábrázoljuk, pl. -7 helyett 65536-7=65529-et.
5-7. byte: csak 0-kat tartalmaz.

Például a H%=13-utasítás hatására a következő tartalma lesz a megfelelő 7 byte-nak:

200-128-0-13-0-0-0

Itt H kódja 72, a +128 miatt lesz az első byte tartalma 200, a második pedig 128. Mivel a számérték csak 13, ezért a 3. byte tartalma 0, a negyedik pedig 13 lesz.

Miután megismerkedtünk a hagyományos számítástechnika gyakran alkalmazott változótípusaival - a számokkal -, folytassuk ismeretszerzésünket a karaktersorozatokkal. Ezt az információt két formában tárolja a C-64:

- mindennemű állandót, konstanst a program szerves részeként, karaktersorozatként. Lássunk erre egy egyszerű példát:

```
10 INPUT "A=";A
20 LET B=2*A+2.5
30 PRINT "B=";B
40 END
```

Ha - az ebben a szakaszban már lassan megszokott módon - kinyomtatjuk a program beírásának eredményeként létrejött tárrészletet, akkor a következő eredményt kapjuk:

```
Program     10 INPUT    "A=";A

 C-64               1
   --       1   1    3       366356
tartalom 38  00   3       451495 0

Program     20  LET     B=2*A+2.5

 C-64               1        1 1 1
   --       2   2    3     3675767545
tartalom 98  00   6     2680250063 0

Program     30 PRINT    "B=";B

 C-64               1
   --       4   3    5     3366356
tartalom 28  00   3     2461496 0

Program     40  END

 C-64               1
   --       4   4    2
tartalom 88  00   8     0 0 0
```

Ha megfigyeljük a kapott végeredményt, akkor nyilvánvaló, hogy a program tárolja őket kódolt formában. Figyeljük meg, hogy "A=" kódolt formája 34-65-61-34, 2-é 50, 2.5-é 50-46-53, "B="-é pedig 34-66-61-34 lesz.

Ezekután vizsgáljuk meg azt, hogy hogyan ábrázolja a C-64 azokat a karaktersorozatokat, amelyeket feladatról-feladatra változtatunk. Mintaprogramunk most egy "névfordító" program lesz:

```
10 INPUT A$,B$
20 PRINT B$,A$
30 END
```

Ha a kapott programot byte-ról-byte-ra kilistázzuk, akkor a következő eredményt kapjuk:

```
13-8-10-0-133-32-65-36-44-66-36-0
25-8-20-0-153-32-66-36-44-65-36-0
31-8-30-0-128-0-0-0
65-128-5-251-159-0-0
66-128-4-247-159-0-0


73-77-82-69-86-65-82-71-65
```

Ha soronként összehasonlítjuk a memóriatartalmat programunkkal, akkor az első 3 sor analógiáját pillanatok alatt felfedezhetjük:

- 133 az INPUT kódja, tokenje;
- 153 a PRINT kódja, tokenje;
- 128 az END kódja, tokenje;
- 32 a szóköz kódja;
- 65-36 az A$ változó megfelelője;
- 66-36 a B$ változó megfelelője;
- 44 a vessző kódja;
- az A$ változót a 4. sor, a B$ változót pedig az 5.sor valósítja meg.

Az egyes változók a numerikus (valós és egész típusú) változókhoz hasonlóan 7 byte-ot foglalnak el, melyek felhasználása most a következő:

1-2. byte: a változó neve (a második karakter kódjához a típusjelölő 128 értéket hozzáadja a C-64).

3. byte: a változóban - ebben a pillanatban - tárolt karaktersorozat hossza. (Ennek nagysága miatt van korlátozva 255-ben a karaktersorozat hossza.)

4-5. byte: a karaktersorozat kezdőcíme (byte-ja) a memóriában - a tárfelosztásnál megszokott - fordított sorrendben.

6-7. byte: felhasználatlan (rendszerint 0-tartalmú).

Ha az A$ és B$ kezdőcímeit és hosszát figyelembevéve kiszámítjuk a megfelelő címeket, akkor kaphatjuk meg az utolsó sor tartalmát, mely a kódtáblázat alapján az IMRE és VARGA szavakat tartalmazza. A fordított sorrend oka,hogy a karaktersorozatok helyét a BASIC-terület végétől visszafelé folyamatosan foglalja el.

Valamennyi változótípus esetén (csak alfanumerikus változónév jellemzi a valós típusú változókat, %-végződés az egész típusúakat, $-végződés a karakter-típusúakat) lehetőség van a csoportos adatkezelésre olymódon, hogy egy adatcsoport számára választunk egy alfanumerikus jelsorozatot, mely legfeljebb 2 jelből áll,s e mögé kerek zárójelben megadott sorszám(ok) adják meg, hogy a csoport melyik elemével végzünk műveletet.

Az előző bekezdésben sűrűn használtuk az alfanumerikus jelzőt. Ez azt jelenti, hogy az illető jelsorozat az angol abc valamely betűjével kezdődik s vagy ezen betűk valamelyikével, vagy számjeggyel vagy szóközzel "folytatódik" s legfeljebb véges sok jelből állhat.

Egy adatcsoportot a mindennapi életben is többféleképpen helyezhetünk el. Így van ez a C-64 esetében is. Dolgozhatunk számsorokkal, számoszlopokkal, számtáblázatokkal, sőt többboldalas számtáblázatokkal is. Ezeknek rendre a számítógép egy, két és három dimenziós **tömb**-jei felelnek meg.Ezeket az adatcsoportokat első pillantásra bonyolult formában tárolja, ezért a belső gépi ábrázolás módszerének, filozófiájának ismertetésétől eltekintünk. Itt csak a programozást segítő vázlatos képet érzékeltetjük. Legyen A 6 elemű számsor, akkor elemei a következő elhelyezkedést mutatják a memóriában:

A(0) A(1) A(2) A(3) A(4) A(5)

Ha B egy 3 X 4-es számtáblázat, akkor elemei szemléletesen a következő képpel helyettesíthetők:

B(0,0) B(0,1) B(0,2) B(0,3)
B(1,0) B(1,1) B(1,2) B(1,3)
B(2,0) B(2,1) B(2,2) B(2,3)

Mindkét tömb esetében furcsa jelenség, hogy az elemek sorszámozását 0-val kezdjük. A gép mindig ezt teszi, de a felhasználó "elfeledkezhet" a 0-ik elemekről.

Ezeket az adatcsoportokat helyezi el a C-64 a változók után közvetlenül. Szinte elképzelhetetlen igazi C-64-program alkalmazásuk nélkül.

3.2 Software alapismeretek

Ebben az alfejezetben vázlatosan ismertetjük a gyakran használt programok működési elvét. Ismertetésünk 3 lényeges komponensre, szakaszra bontható a C-64 alap operációs rendszerének komponensei, a gyakran használt BASIC-bővítések, továbbá az utasításközlési módozatok (parancs és program üzemmód) ismertetésére.

3.2.1 A C-64 operációs rendszerének programjai

Mint arról már az előzőekben említést tettünk, a C-64 operációs rendszere 3 lényeges elemből áll:

- a képernyőszerkesztő SCREEN EDITOR-ból,
- az adathordozó perifériákkal információcserét lebonyolító KERNAL-rutinokból,
- a BASIC-utasításokat értelmező és végrehajtó BASIC interpreterből.

Ha megelégednénk felületes tudással, akkor ennél többet nem is kellene mondanunk az egyes alapkomponensekről. Nemcsak a szófordulat, de a C-64 hatékony, célszerű felhasználásának célja mondatja velünk el a következő lényeges tudnivalókat:

SCREEN EDITOR:

Ez a C-64 ROM-jába állandóra beégetett utasítássorozat gondoskodik arról, hogy a TV képernyője vagy a monitor "élő" kapcsolatot teremtsen a C-64 és az őt alkalmazó ember között. Ez a program

- tárolja, tartja karban és "sugározza" az aktuális 1000 karaktert - 25 sorban, 40 oszlopban - a felhasználó felé,
- segít a felhasználó utasításainak, adatainak, üzeneteinek nyomonkövethető, könnyű és gyors összeállításában.

Amikor bekapcsoljuk a C-64-et, akkor egy előkészítő program segítségével a C-64 valamennyi memóriacíme alaphelyzetbe kerül, s bejelentkezik a képernyőszerkesztő SCREEN EDITOR az első fejezetben már többször felírt üzenettel. Ekkor a villogó négyzet jelzi, hogy a SCREEN EDITOR kezd el dolgozni.

Már első fejezetbeli próbálkozásaink során megfigyelhettük, hogy a képernyőtartalom különbözőképpen reagál az egyes billentyűk megnyomására. Ezeket a hatásokat a SCREEN EDITOR hajtja végre. A SCREEN EDITOR nemcsak a látható képernyőtartalmat kezeli, hanem azok szemmel nem látható jellemzőit (például az összetartozó sorokat). Hosszú oldalakon ecsetelhetnénk a SCREEN EDITOR feladatait és működését, de erre a programozás kezdeti állapotában nem lesz szükségünk.

KERNAL-rutinok:

A SCREEN EDITOR-hoz hasonlóan ez az operációs rendszer-alkotó is állandó "lakója" a C-64 ROM-jának. Itt a rutin szó gyakran használt eljárás szinonimájaként szerepel. A többes számot pedig a perifériák (adathordozók) sokfélesége indokolja, hiszen minden adathordozótípushoz, minden adatátviteli irányhoz és módszerhez más-más rutinszerű eljárást alkalmazunk. A kezdő felhasználó ezek használatát csak passzívan veszi észre, amikor például lemezhez fordulásnál kipiroslik a lemezegység jelzőlámpája, vagy "kattog" a mátrixnyomtató.

BASIC-interpreter:

Ez a programkomponens is az előzőekhez hasonlóan a C-64 ROM-jában helyezkedik el. Működése a **RETURN**-billentyű megnyomása után indul el. Működése két lényeges fázisra, lépésre bomlik: az értelmezésre és a végrehajtás irányítására. Ennek részletes elemzésére a 3.2.3 szakaszban kerítünk sort.

3.2.2 Bővítési lehetőségek: SIMON'S BASIC, HELP+, TURBO TAPE...

Szemléletesen az ilyen, és ehhez hasonló programbővítéseket a következőképpen kell értelmeznünk:

- első durva hasonlatunk, hogy elküldtük C-64-ünket továbbképző tanfolyamra, de ott nem figyelt, csak jegyzetet kapott, melyet minden ilyenirányú feladat előtt lelkiismeretesen elolvas, majd hozzálát és eredményesen megoldja feladatát,

- **komolyra fordítva a szót** betöltjük a RAM-memóriába az aktuális segédprogramot, s ennek elindítása után (RUN!!!) C-64-esünk lényegesen többféle utasításformát fog megérteni.

A SIMON'S BASIC szinte valamennyi munkaterületen lényeges többletsegítséget tud nyújtani, míg a HELP+ elsősorban programozástechnikai, programszerkesztési segédeszközökben bővelkedik. A TURBO TAPE a kazettán tárolt, vagy tárolni kívánt program gyors elérését hivatott elősegíteni.

3.2.3 Parancs és program üzemmód

Amikor egy általunk helyesnek ítélt jelsorozat végén megnyomjuk a **RETURN**-billentyűt, akkor átadjuk a vezérlést a BASIC-interpreternek, mely a következőket végzi:

- megvizsgálja az első nem szóköz jelet, hogy szám-e (0123456789),
- ha szám, akkor **program**-üzemmódra készül fel,
- ha nem szám, akkor **parancs**-üzemmódra tér át,
- mindkét esetben tovább elemzi a jelsorozatot,
- az első nem-szám-jelig összeolvassa a számjegyeket, melyektől elvárja, hogy 0 és 65535 közé essenek,
- BASIC-alapszót keres ezután, ha talál, akkor helyettesíti az ún. tokenjével vagy kódjával,
- ezután minden nem-BASIC-alapszót(kulcsszót) a megfelelő ASCII-kódjával helyettesít egészen a jelsorozat végéig, amelyet egy 0-értékkel "honorál",
- ha program üzemmódban vagyunk, akkor a sorszám alapján elhelyezi az új utasítást az őt megillető helyre,
- ha parancs üzemmódban vagyunk, akkor átadja a vezérlést a végrehajtó eljárásoknak (rutinoknak),
- az utóbbi két eset bármelyikében véges sok lépés végrehajtása után (kivéve a szándékolt vagy véletlen végtelen ciklus esetén) READY. üzenet kiséretében visszaadja a vezérlést a SCREEN EDITOR-nak.

4. C-64 BASIC

A BASIC a következő angol definicióból képzett mozaikszó:
Beginners All-purpose Symbolic Instruction Code
Az angol definíció magyar fordítása a következő:
Kezdők Általános Szimbólikus Utasításkészlete

A C-64 tervezői kidolgozták a C-64 specialitásait kihasználó BASIC-változatot.

Miután elegendő részletességgel megismerkedtünk a C-64 hardware és software alapfogalmaival, itt az ideje, hogy elinduljunk az aktív géphasználat útján.

Javasoljuk, hogy tankönyvünk ezután következő leckéit a következő rendszerben sajátítsa el:

- egyszerre legfeljebb **egy** teljes lecke anyagát vegye át. Mélyüljön el a lecke anyagának elsajátításában,
- a lecke tanulmányozását a minta-utasítássorozat begépelésével kezdje,
- próbálja ki az utasítássorozatokat a megadott adattal (adatokkal),
- ha már eléggé önálló, akkor próbálkozzon önálló adatválasztással,
- ha már ez is sikerült, akkor próbálkozzon meg az utasítássorozat apróbb, majd komolyabb mértékű módosításával,
- amikor már elegendő tapasztalat gyűlt össze, akkor olvassa el a minta utasítássor után leírtakat, s vesse össze a személyesen tapasztaltakkal,
- ezután következhet az elméletileg szabatos fogalomalkotás,
- ezután javasoljuk a felsorolt gyakorló feladatok megoldását (ügyeljen az esetleg nehézséget okozó sorrendcsere elkerülésére).

Felhívjuk a figyelmet arra, hogy a C-64 csak azt a jelsorozatot fogja feldolgozni, amelyet a **RETURN**-billentyű megnyomásának hatására elolvasott. Tehát minden utasítás után **KELL** a **RETURN**!

1. LECKE: ?-parancs, aritmetikai műveletek és sorrendjük

Minta-parancssorozat:

```
? 2+2
? 20-1
? 1.2+0.9
? 64.4-33.3
? 2*2
? 4.5*4.5
? 39/13
? 33.4/0.2
? 133/0
? 0/133
? 3↑2
```

```
? 13↑3
? 2.5↑2
? 16↑0.5
? 32↑0.2
? 2*2+3*3+4*4
? (45+33)*11
? 10/6+10/6
? 1/(1+1/(1+1/(1+1/2)))
? 1/1+1/1+1/1+1/2
? "COMMODORE 64"
? "1986. APRILIS 18."
? "JANOSHALMA,"+" 1984. JULIUS 2."
```

Parancssorozat magyarázata:

Mind a 23 parancsunk a következő szerkezetet mutatja:

- ?-lel kezdődik, mely mintegy kérdést intéz a C-64-hez,
- valamennyi sorban a ? utáni szóközt egy jelsorozat követi, amely az esetek többségében mindennapi ismereteink segítségével könnyen értelmezhető. Ha sorban végrehajtjuk őket, akkor a következő eredményeket kapjuk:

```
? 2+2                          4
? 20-1                         19
? 1.2+0.9                      2.1
? 64.4-33.3                    31.1
? 2*2                          4
? 4.5*4.5                      20.25
? 39/13                        3
? 33.4/0.2                     167
? 133/0                        ?DIVISION BY ZERO ERROR
? 0/133                        0
? 3↑2                          9
? 13↑3                         2197
? 2.5↑2                        6.25
? 16↑0.5                       4
? 32↑0.2                       2
? 2*2+3*3+4*4                  29
? (45+33)*11                   858
? 10/6+10/6                    3.33333333
? 1/(1+1/(1+1/(1+1/2)))        .625
? 1/1+1/1+1/1+1/2              3.5
? "COMMODORE 64"               COMMODORE 64
? "1986. APRILIS 18."          1986. APRILIS 18.
? "JANOSHALMA,"+" 1984. JULIUS 2."  JANOSHALMA, 1984. JULIUS 2.
```

Ezek alapján próbáljuk meg megfejteni a C-64 műveleti rendszerét!

Első felfedezésünk az lehet, hogy az összeadás (+) és a kivonás (-) műveleti jele ugyanaz, mint amit a hétköznapi életben használunk.

Következő felfedezésünk a tizedesvessző helyett a tizedespont használata (a vessző a számítástechnikusok körében szinte mindig a paraméterelválasztás jele!).

A szorzandó és szorzó mennyiség elválasztására, s egyszersmind összekapcsolására használjuk a * jelet.

Az osztásnál az osztandót az osztótól a / jel választja el.

Amikor egy számértéket többször egymásután meg akarunk szorozni önmagával, akkor beszélünk hatványozásról. Ha például 13-at 3-szor akarjuk megszorozni önmagával, akkor a 13*13*13 kifejezés értékét kívánjuk meghatározni, vagyis 13 harmadik hatványát, melyet a C-64-en a 13↑3 kifejezés valósít meg. Tehát a hatványozás jele a felfelé mutató nyíl.

0-val a C-64 sem tud osztani (lásd a 9. példában a 133/0 osztáskisérletet!).

Az idézőjelben szereplő karaktersorozatokat változtatás nélkül képernyőre írja a C-64.

Karaktersorozatok esetében a +-műveleti jel a karaktersorozatok közvetlen egymás mellé helyezését jelenti.

A kerek zárójelek - a matematikában megszokott módon - a műveletek végrehajtásának megszokott sorrendjét változtatják meg (lásd például a 19., és a 20. parancsokat!).

A leckében megismert új fogalmak:

Kérdező parancs:

Általános alakja: ? kifejezés

Működési elve:

Kinyomtatja a képernyőre a kifejezés aktuális értékét, tartalmát.

Egész számkonstans:

Esetleg előjellel kezdődő számjegyekből álló véges jelsorozat, amelyet a mindennapi gyakorlathoz hasonlóan értelmezünk.

Valós számkonstans:

Esetleg előjeles számjegyeket tartalmazó jelsorozat, amelyben a mindennapi gyakorlattól eltérően tizedespontot használunk. A C-64 nem követeli meg (és nem írja ki) az esetleg hiányzó tört vagy egész részt, de mindkettő hiánya esetén megkövetel egy 0-t, s ezt az eredményközlésnél ki is írja.

Műveleti jelek:

Összeadás	+
Kivonás	-
Szorzás	*
Osztás	/
Hatványozás	↑
Sorrendfelcserélés	()

Karaktersorozatok (szövegkonstansok):

Ha bármilyen jelsorozat "-párok között szerepel, akkor azt a C-64 karaktersorozatként, szövegkonstansként kezeli.

Karaktersorozatok összefűzése:

Ha két vagy több karaktersorozatot a +-jellel kapcsolunk össze, akkor ezt folyamatosan "összeolvassa" a C-64.

Műveletek sorrendje:

1. Hatványozás
2. Szorzás vagy osztás
3. Összeadás vagy kivonás
4. Ha két egyenrangú művelet közül kell választanunk, akkor az elsőt választjuk.

   Ez a sorrend a következőképpen értendő:
   - ha van a kifejezésben hatványozás, és annak végrehajtásához valamennyi adat rendelkezésünkre áll, akkor ezt hajtjuk végre leghamarabb,
   - ha valamennyi végrehajtható hatványozást elvégeztünk, akkor a szorzások vagy osztások következnek, melyek egyenrangúak,
   - ha valamennyi elvégezhető hatványozást, szorzást és osztást elvégeztünk, akkor kerül sor az elvégezhető összeadások, illetve kivonások kiszámítására.

   A műveleti sorrendek szabálya pontosan követi a matematikában megszokott precedencia-szabályt.

   Mivel a karaktersorozatok körében csak egyetlen műveletet definiáltunk, így nincs értelme műveleti sorrendről töprengeni.

Gyakorló feladatok:

1.1 "Kérdezzük ki" a C-64-et, tud-e két számot összeadni az 1-től 100-ig terjedő számkörben.

1.2 "Kérdezzük ki" a C-64-et, tud-e kivonni az 1-től 100-ig terjedő számkörben.

1.3 Írassuk ki a C-64-gyel valamennyi szorzótáblát! (1*1,1*2,...,1*10; sit)

1.4 Számítsuk ki egyetlen paranccsal annak a derékszögű háromszögnek az átfogóját, amelynek két befogója rendre: 3 és 4, illetve 9 és 12.

1.5 Írjuk fel a következő kifejezések BASIC-megfelelőjét!

$$\frac{2*2}{3-1} \qquad \frac{3+23}{23-3} \qquad \frac{2(6-2)}{22-2,2}$$

1.6 Írjuk fel a következő kifejezések matematikai alakját!

```
? 2+3/3-1
? (2+3)/(3-1)
? (2*2+3*3+4*4)↑(1/2)
```

1.7 Írjuk fel azt a parancsot, amely megadja a (11,22,13) koordinátájú vektor hosszát!

1.8 Írjuk fel a következő "emeletes" tört BASIC-megfelelőjét!

$$\cfrac{\cfrac{19-13}{19+13}+\cfrac{19+13}{19-13}}{\cfrac{19-13}{19+13}}$$

1.9 Vizsgálja meg a következő kifejezések végrehajtási sorrendjét!

```
? 2↑3↑4↑5
? 2↑(3↑4)↑5
? (2↑3)↑(4↑5)
? 2↑(3↑(4↑5))
```

A fenti kifejezések közül melyek szolgáltatnak azonos eredményt?

1.10 Mérje meg egy héten keresztül a déli hőmérsékletet, s számítsa ki az átlagát!

1.11 Mérje meg családja tagjainak magasságát, és számítsa ki, hogy átlagosan milyen magas egy családtag!

1.12 Kérdezze meg a család minden tagját, hány forintot keresett ebben a hónapban, majd számítsa ki a család havi jövedelmét!

1.13 Mint tudjuk 8 óra munka, 8 óra szórakozás és 8 óra pihenés az ideális napi felosztás. Kérdezzük meg barátunkat, hogy mennyit dolgozott tegnap, majd számítsuk ki, hogy az ideális 8 óra hány százalékát teljesítette! (Iskola esetén a tanulás a munka!)

2. LECKE: Változók,értékadás

Minta-parancs- és utasítássorozatok:

1. sorozat:

```
A=2*2
? A
```

2. sorozat:

```
A=2*2: ? A
B=3*3: ? B
C=20-1: ? C
```

3. sorozat:

```
10 A=2*2
20 B=3*3
30 C=20-1
40 PRINT A,B,C
```

4. sorozat:

```
10 A=13.4
20 B%=13.4
30 C$="13.4"
40 PRINT A,B%,C$
```

5. sorozat:

```
10 M=3.5
20 A%=12
30 C$="MENNYIT FIZETUNK A KENYERERT?"
40 PRINT C$
50 PRINT A%*M;" FORINTOT"
```

A minta-parancs- és utasítássorozatok magyarázata:

Ha parancs- és utasítássorozatainkat végrehajtjuk, akkor rendre a következő eredményeket kapjuk:

1. sorozat - 1. parancs:

Az A azonosítójú rekesz felveszi az egyenlőségjel jobb oldalán álló aritmetikai kifejezés (esetünkben a 2*2) értékét, vagyis a 4-et, s ezt valós számként ábrázolja benne. A felhasználó semmit sem lát!!!

1. sorozat - 2. parancs:
"Megkérdezi" az A azonosítójú rekesz tartalmát. Ennek hatására válik olvashatóvá a 4-es eredmény!

2. sorozat:
Ez a sorozat 3 parancspárból áll, melyek közül az első mindig értéket ad egy-egy változónak, majd a párja kiírja a képernyőre a kapott eredményt.

Az A tárolórekesz tartalma és a kiírt érték: 4
A B tárolórekesz tartalma és a kiírt érték: 9
A C tárolórekesz tartalma és a kiírt érték: 19

Megjegyzés: Bár már kivánkozik a változó fogalmának megalkotása, várjuk meg a kipróbálásra szánt példák végét!

3. sorozat:
Ha begépeljük utasításainkat, akkor azok megjelennek a képernyőn, de ezenkívül nem történik semmi. Ha látni akarjuk az eredményeket, akkor írjuk be, majd olvastassuk el a **RUN**-parancsot. Ha ezt megtesszük, akkor eredményül a következőt kapjuk:

```
 4            9            19
```

Már előző fejezeteinkben közöltünk programlistákat, melyeket "mint pincér a megrendelt ételeket-italokat" folyamatosan hajt végre a számítógép. Ez a program-üzemmód lényege, melyet a mindennapi életben magunk is lépten-nyomon alkalmazunk, amikor összetettebb feladatok elvégzésére kérjük fel munkatársunkat vagy családunk valamely tagját.

4. sorozat:
Itt is az előző sorozathoz hasonlóan több lépésből álló feladat végrehajtására "tanítjuk meg" számítógépünket.

10-es utasítás: A értéke 13.4 lesz.
20-as utasítás: B% értéke 13 lesz.
30-as utasítás: C$ értéke "13.4" lesz.
40-es utasítás: ennek hatására jelenik meg a képernyőn a következő végeredmény az előző sorozathoz hasonló formában.

```
 13.4        13        13.4
```

5. sorozat:
Esetünkben 5 utasításban, "lépés"-ben írtuk le a teendőket:

10-es utasítás(1. lépés): Az M rekesz vegye fel a 3.5 értéket, mely jelentse azt, hogy 3 és fél kg kenyeret vettünk az ünnepekre.
20-as utasítás(2. lépés): Az A% rekesz vegye fel a 12 értéket, mely jelentse azt, hogy 1 kg kenyér ára 12 forint.
30-as utasítás(3. lépés): C$ vegye fel a "MENNYIT FIZETUNK A KENYERERT?"karaktersorozatot, mint értéket, hogy "párbeszédes", a hétköznapi ember számára is könnyen érthető eredmény jelenhessen meg a képernyőn.

40-es utasítás(4. lépés): A PRINT C$-utasítás hatására megjelenik a képernyőn a "MENNYIT FIZETUNK A KENYERERT?"-kérdés.
50-es utasítás(5. lépés): Végül megjelenik a "42 FORINTOT"-felelet.

A leckében előforduló új fogalmak:

Valós típusú változó:

Azokat a memóriában 7 byte-nyi területet elfoglaló rekeszeket, melyeket valós számértékek tárolására használunk fel, s amelyeket legfeljebb két alfanumerikus írásjellel azonosítunk, valós típusú változónak nevezünk.

Megjegyzés: Nagyon ügyeljünk arra, hogy változónévnek szánt jelsorozat sohase tartalmazzon BASIC-alapszót vagy annak részét. Kezdetben elégedjünk meg az egybetűs változónevekkel.

Változó:

A számítástechnikában minden olyan tárolórekeszt, amelynek tartalma feladatról-feladatra, vagy akárcsak egy feladaton belül is változhat, változónak nevezzük.

Alfanumerikus jelsorozat:

Az olyan jelsorozatot nevezzük alfanumerikusnak, amely betűvel (az angol ABC 26 betűje) kezdődik és betűvel (angol ABC 26 betűje) vagy számjeggyel (0,1,2,3,4,5,6,7,8,9) folytatódik.

Az angol ABC:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

Megjegyzés: A BASIC nem tesz különbséget a kis és nagybetűvel írott változónév között.

Egész típusú változó:

Azokat a memóriában 7 byte-nyi területet elfoglaló rekeszeket, amelyeket legfeljebb két alfanumerikus jellel s egy %-jellel azonosítunk, egész típusú változónak nevezünk.

Megjegyzés: Maga az ábrázolt egész érték csak két byte-ot foglal el, de a formális helyfoglalás hét byte.

Karakter típusú változó:

Azokat a rekeszeket, amelyekben karaktersorozatok tárolásához szükséges információkat tárolunk, s amelyekre legfeljebb két alfanumerikus jellel és egy $-jellel hivatkozunk, karakter típusú változónak nevezzük.

Megjegyzés: Az információábrázolás módját a 3.1.3. szakaszban részletesen elemeztük. A felhasználónak - kezdetben - elegendő azt tudnia, hogy az egyszerű alfanumerikus jelsorozat - valós, a %-jeles - egész, a $-jeles pedig karakter típusút jelöl.

ÉRTÉKADÁS:

(A fogalom kulcsfontosságú szerepet tölt be a számítástechnikában, ezért emeltük ki már a címét is!)

Általános alakja:

Sorszám LET változó = kifejezés

Az általános alak szereplői:

Sorszám - mint minden BASIC-utasításnál - a programban elfoglalt helyet mutatja meg. Ha hiányzik, parancs-üzemmódról van szó.

LET - az értékadó-utasítás BASIC-alapszava, melyet szinte mindig elhagyunk!

Változó - annak a változónak a megnevezése, melynek az értékadás eredményeként új értéket kívánunk adni.

= - az értékadás közvetítő, műveleti jele.

Kifejezés - az értékadás végrehajtásához ezt a kifejezést kell kiértékelnünk, kiszámítanunk.

Működési elve:

Amikor a C-64 rátér az értékadó utasítás végrehajtására, akkor megkeresi a kifejezésben szereplő változók aktuális (pillanatnyi) értékét, értelmezi a konstansokat és a műveleti jeleket, majd a sorrendiség (precedencia-szabály) alapján kiszámítja a kifejezés aktuális értékét. Törli a változó aktuális értékét, majd helyére írja a kifejezés imént meghatározott értékét.

Megjegyzés: Az értékadás bal oldalán álló változónak és a jobboldalon álló kifejezésnek mindig azonos típusúnak, de legalábbis hasonlónak kell lennie.

Változó típusa	kifejezés típusa	minősítése
valós	valós	szabályos
valós	egész	szabályos
valós	karakter	hibát okoz
egész	valós	kerekítési hiba
egész	egész	szabályos
egész	karakter	hibát okoz
karakter	valós	hibát okoz
karakter	egész	hibát okoz
karakter	karakter	szabályos

Kifejezés:

(Mint fogalmaink többsége, ez is fejlődni fog!)
Aktuális ismereteink alapján a következő definíciót fogjuk elfogadni:

Azokat a jelsorozatokat, amelyek különböző típusú (valós, egész, karakter) konstansokat, változók azonosítóit s azokat "összefűző" műveleti jeleket tartalmaznak, kifejezésnek nevezzük.

(Kivételes eseteket leszámítva azonos típusú kifejezés-alkotók (konstansok, változók, műveleti jelek) nem állhatnak egymás mellett!)

Helyes kifejezések:

```
3.14*A+B
AB-CD
(A%+B%)*(C%-D%)
12*33
"KUTYA"+"FULE"
```

Helytelen kifejezések:

```
3A+4C                (Hiányoznak a szorzójelek)
AM/2                 (Itt is a szorzójel hiányzik, ha A - alap, M - magasság)
(-B+-(BB-4AC)↑0.5)/2A (Itt 4 szorzásjel, 1 zárójelpár s a matematikában
                       megszokott +- jelegyüttes felbontása hiányzik.)
```

Kifejezések típusa:

A kifejezések értéke és típusa lépésről-lépésre (műveletenként) alakul, melyek a következő táblázat alapján történnek:

Műveletek / Résztvevők	+	-	*	/	↑
valós,valós	v	v	v	v	v
valós,egész	v	v	v	v	v
valós,karakter	-	-	-	-	-
egész,valós	v	v	v	v	v
egész,egész	e	e	e	v	v
egész,karakter	-	-	-	-	-
karakter,valós	-	-	-	-	-
karakter,egész	-	-	-	-	-
karakter,karakter	k	-	-	-	-

ahol:
v - valós
e - egész
k - karakter
- - nem értelmezett

A "kérdező"-utasítás további általánosítása:

A PRINT ("kérdező") utasítás:

sorszám PRINT output-lista vagy sorszám ? output-lista

sorszám - a programbeli helyét szabja meg

PRINT (?) - a képernyőre írás kulcsszava

output-lista - itt soroljuk fel azokat a változókat, kifejezéseket,amelyeknek tartalmát, értékét meg akarjuk tudni. (Az egyes listaelemeket kiírást vezérlő írásjelek (,;) választják, választhatják el.)

Gyakorló feladatok:

2.1 Írjuk be az N$ változóba a nevünket, az SZ változóba a születési évünket, majd a K változóba írassuk be az 1986-SZ kifejezés értékét, majd "kérdezze meg" az N$;K;" EEVES" kifejezést!

2.2 Írjuk be az A$ változóba autónk márkáját, a B változóba a 100 km-enként elfogyasztott benzin mennyiségét, az A változóba a használt benzin egy literének aktuális árát Ft-ban. Kérdezzük meg hány Ft-ba kerül 100 km megtétele, ha megfeledkezünk a többi költségről (olaj, CASCO, stb...).

2.3 Fejlesszük tovább a 2.2-es feladatot! Írjuk be S-be a tervezett kirándulásunk km-igényét, majd kérdezzük meg a várható benzinköltséget!

2.4 Írjuk be - a geometriában megszokott jelölést követve - az A és B rekeszbe a derékszögű háromszög két befogójának adatait, majd a Pythagoras-tétel felhasználásával határozzuk meg és írassuk ki a C átfogót!

2.5 Ha a másodfokú egyenlet együtthatói rendre A=1, B=-5, C=6, akkor számíttassuk ki a két X-értéket az értékadó, majd kérdező utasítás segítségével! Próbáljunk meg különböző megoldásokat találni!

2.6 Ha vásárolunk 5 kg burgonyát 8.50-es áron, 15 fej salátát, fejenként 3 Ft-ért, s 200 Ft-tal indultunk a boltba, akkor mennyi pénzzel térünk haza.

2.7 Ha a piacra kiviszünk 110 szál rózsát s 230 szál szegfűt, melyeket 33 ill. 17 Ft-ért tudunk eladni. Mennyi pénzt fogunk hazavinni?

2.8 Egy brigád elmegy almát szedni. 13-an vannak. Naponta 50 Ft-ért kapnak étkezést és szállást. Az export almáért kg-onként 3 Ft-ot, a léalmáért kg-onként 0.30 Ft-ot kapnak. Mennyi léalmát kell szedniük ahhoz, hogy meglegyen az ellátás? Ha 777 kg export almát és 1200 kg léalmát szedtek, akkor fejenként mennyi volt a teljesítmény? Ha levonjuk a napi 50 Ft-ot, akkor mennyi marad? Az ideális napi norma 200 Ft "tiszta" kereset. Ennek hány százalékát teljesítették a brigádtagok?

2.9 Petinek 33 Ft-ja van s 4 barátja. Ha igazságosan mindenkinek akar venni a 2 Ft/gombóc fagylaltból, akkor hány gombócot vehet fejenként?

2.10 Andreának 64 szelet csokoládéja van s két barátnője, akik nagyon szeretik a csokoládét. 5 szeletet eltesz az öccsének. A többit igyekszik elosztani hármuk között. Mennyi csokit kap Magdi és Gabi?

2.11 Magdi vizsgára készül. A tananyag 1350 oldal. Naponta 200 oldalt tud elsajátítani s még egy napot az ismétlésre szán. Hány nap kell Magdinak a sikeres vizsgázáshoz?

2.12 Gabi rendezi a klub napi bevételét. A bevétel rendezésénél 5 db 1000 Ft-ost, 30 db 500-ast, 73 db 100 Ft-ost, 22 db 50 Ft-ost, 23 db 20-ast, 34 db fém 10-est számol meg. Mennyi a klub napi bevétele? Mennyi a tiszta bevétel, ha a kiadás 30 000 Ft volt?

2.13 Marika 30 kg szőlőt szedett, Erika 25 kg-ot, Anci pedig 55 kg-ot. Ha az összes mennyiség a 100 %, akkor hány százalékot szedett a 3 lány külön-külön?

3. LECKE: Adatvitel, INPUT-utasítás

Minta-utasítássorozatok:

1. sorozat:

```
10 INPUT A
20 INPUT B
30 C=(A*A+B*B)↑0.5
40 PRINT "A HAROMSZOG ATFOGOJA"C
```

2. sorozat:

```
10 INPUT "MENNYI KENYERET VETTUNK";M
20 INPUT "HANY FT A KENYER ARA";A%
30 PRINT "A KENYERERT ";A%*M;" FORINTOT FIZETTUNK"
```

3. sorozat:

```
10 INPUT "MILYEN AUTOJA VAN";A$
20 INPUT "MENNYIT FOGYASZT 100 KM-EN";B
30 INPUT "HANY FT 1 L BENZIN";A
40 PRINT "AZ ON ";A$;" KOCSIJA 100 KM-EN "
50 PRINT A*B;" FT ERTEKU BENZINT FOGYASZT"
```

4. sorozat:

```
10 INPUT "HANY BARATJA VAN PETINEK";B%
20 INPUT "HANY FT-JA VAN PETINEK";FT
30 INPUT "HANY FT 1 GOMBOC FAGYLALT";G
40 NB%=FT/(G*(B%+1))
50 PRINT "PETI ES BARATAI ";NB%;" GOMBOCOS FAGYLALTOT ESZNEK"
```

5. sorozat:

```
10 INPUT "HANY OLDALAS MAGDI TANKONYVE";T%
20 INPUT "HANY OLDALT TUD EGY NAP MEGTANULNI";N%
30 NN%=(T%+N%-1)/N%
40 NN%=NN%+1
50 PRINT "MAGDINAK A SIKERES FELKESZULESHEZ"
60 PRINT NN%;" NAPRA VAN SZUKSEGE"
```

A minta-utasítássorozat magyarázata:

1. sorozat:
Ha soronként begépeljük a négysoros programot, majd a **RUN**-paranccsal elindítjuk, akkor a képernyőn a következő sorban egy ? jelenik meg. Válaszoljunk erre egy 3-as értéket a **RETURN**-billentyű kiséretében. Ekkor a következő sorban jelenik meg a ?. Ha erre most 4-es értéket felelünk, akkor a következő sor elején megjelenik a "A HAROMSZOG ATFOGOJA 5"-üzenet, majd a READY.-üzenet. S valóban a 3 4-befogójú derékszögű háromszög átfogója 5 egység. Újabb **RUN**-parancs begépelésével újabb derékszögű háromszögek átfogóit számíthatjuk ki a Pythagoras-tétel segítségével.

2. sorozat:
Mivel ez a programunk rövidebb az előzőnél, ezért a 2. sorozat begépelése (beprogramozása) előtt távolítsuk el a memóriából az előző utasítássorozatot. Erre két lehetőségünk van:
- a C-64 kikapcsolásával, majd bekapcsolásával, melyet a **SYS64738**-paranccsal helyettesíthetünk,
- a **NEW**-paranccsal.

Mindkettő kiüríti a C-64 BASIC-programok tárolására fenntartott területét.

Begépelt programunk ezekután "párbeszédes"-üzemmódban "eltársalog" a felhasználóval, programozóval.
10-es utasítás: MENNYI KENYERET VETTUNK? Vegyünk másfél kg-ot. Ezt az 1.5 és a **RETURN** begépelésével tudatosítjuk a C-64-gyel.
20-as utasítás: HANY FT A KENYER ARA? Legyen ez 16 Ft. Ekkor válaszunk 16 és a **RETURN** lesz.
30-as utasítás: A KENYERERT 24 FORINTOT FIZETTUNK - eredmény jelenik meg a képernyőn.

3. sorozat:
10-es utasítás: MILYEN AUTOJA VAN? A válasz legyen S-100
20-as utasítás: MENNYIT FOGYASZT 100 KM-EN? A válasz legyen 8.5
30-as utasítás: HANY FT 1 L BENZIN? Legyen a válasz 20
40-50-es utasítás: AZ ON S-100 KOCSIJA 100 KM-EN 170 FT ERTEKU BENZINT FOGYASZT - üzenet fog megjelenni a képernyőn.

4. sorozat:
10-es utasítás: HANY BARATJA VAN PETINEK? A válasz legyen 4
20-as utasítás: HANY FT-JA VAN PETINEK? A válasz legyen 33
30-as utasítás: HANY FT 1 GOMBOC FAGYLALT? A válasz legyen 2
40-es utasítás: Kiszámítja, hogy fejenként - esetünkben - 3 gombóc jut
50-es utasítás: Megjelenik a képernyőn a "PETI ES BARATAI 3 GOMBOCOS FAGYLALTOT ESZNEK"-üzenet

5. sorozat:
10-es utasítás: HANY OLDALAS MADGI TANKONYVE? A válasz legyen 1250
20-as utasítás: HANY OLDALT TUD EGY NAP MEGTANULNI? Legyen a válasz 200
30-as utasítás: Felfelé kerekítve kiszámítja az egyszeri tanuláshoz szükséges napok számát
40-es utasítás: NN%-hoz hozzáadja az ismétléshez szükséges napot (Az utasítás kissé szokatlan működési elvét lásd alább!)

50-60-as utasítás: Megjelenik a képernyőn a "MAGDINAK A SIKERES FELKESZULESHEZ 8 NAPRA VAN SZUKSEGE"-üzenet.

A leckében előforduló új fogalmak:

Az INPUT-utasítás:

Egy-egy változó típusától függetlenül két módon kaphat új értéket:
- az előző leckében megismert - értékadás és a
- most bemutatott - INPUT adatbeviteli utasítás segítségével.

Általános alakja:

sorszám INPUT karaktersorozat;input-lista

Az alkotóelemek és szerepük:

sorszám - az utasítás programbeli helyét határozza meg.

INPUT - a billentyűzetről adatbevitelt "hívó" BASIC-alapszó, melynek hatására rögtön, vagy az esetleg utána szereplő karaktersorozat képernyőre írása után ?-jelet ír a képernyőre és villogó kurzort - jelezve, hogy adatra (adatokra) vár.

karaktersorozat; - használata esetleges (nem kötelező).(Ha használjuk, akkor ezáltal tudjuk felhívni a program használójának figyelmét az éppen szükséges adat tudnivalóira, amint azt az utolsó 4 mintaprogramban tettük.)

input-lista - egy vagy több változó vesszővel elválasztott felsorolása, mely(ek)nek ebben a programlépésben kívánunk értéket adni.

Az utasítás működési elve:

Amikor a C-64 a megfelelő utasítás végrehajtásához kezd, akkor - ha van - kiírja az INPUT után idézőjelek között szereplő karaktersorozatot, majd a kérdőjelet. Ezután várja a felhasználó válaszát, amelynek adatszámban (vessző az elválasztó-jel) és típusban összhangban kell lenni az input-listán megadott változónévvel.

Általánosított értékadás fogalma:

Eddig olyan értékadó utasításokkal foglalkoztunk, amelyek szinte "megszólalásig" hasonlítottak a matematikában gyakran használt helyettesítési képletekre. Az 5. sorozat 40-es utasításában találkozunk egy olyan értékadó utasítással, amely minden eddigi matematikai ismeretünknek ellentmond. Ez a NN%=NN%+1-utasítás. Gondoljuk végig ezt a lépést mégegyszer az előző leckebeli definíció és a mostani számpélda kapcsán! NN% induló értéke 7. Definíciónk szerint kiszámítjuk az NN%=NN%+1 jobboldali kifejezésének pillanatnyi értékét. Ez 7+1=8. Töröljük NN% pillanatnyi értékét, s a kapott 8-at helyére írjuk.

Gyakorló feladatok:

3.1 Írjon programot, amely magyarázó karaktersorozat kiséretében beolvassa egy háromszög alapját és magasságát, majd kiszámítja, és magyarázó szöveg kiséretében kiírja a háromszög területét a képernyőre!

3.2 Írjon programot, amely magyarázó szöveg kiséretében beolvassa egy autó márkáját, a benzin árát, 100 km-enkénti benzinfogyasztását,a mára tervezett utat. Számítsa ki, és magyarázó szöveggel írassa ki a várható benzinfogyasztást és annak árát!

3.3 Írjon programot, amely egy utasítással beolvas 3 számot, s kiszámítja azok átlagát!

3.4 Írjon programot, amely beolvas két számot, és aritmetikai műveletek segítségével kiszámítja a kisebb és a nagyobb értéket, majd ebben a sorrendben kiíratja!

3.5 Írjon programot, amely az indulási és érkezési időpont, valamint a megtett út ismeretében kiszámítja az átlagsebességet!

3.6 Írjon programot, amely a lejtő magasságának és hosszának ismeretében kiszámítja a lejtő esési százalékát!

3.7 Írjon programot, amely beolvassa a vásárolt termék nevét, mennyiségét, majd egységárát, s kiszámítja az érte kifizetett összeget!

3.8 Írjon programot, amely beolvassa a vásárlás napját, a vásárolt termék nevét és szavatossági határidejét, majd kiszámítja, hogy hány napig használható még a vásárolt termék!

3.9 Írjon programot, amely beolvassa egy tanuló nevét, 5 tantárgyának nevét és a megszerzett érdemjegyet, majd kiszámítja a tanuló átlageredményét!

3.10 Írjon programot, mely beolvassa egy dolgozó nevét és 12 hónapon át a keresetét, majd kiszámítja, hogy mennyit keresett a dolgozó összesen!

3.11 Írjon programot, amely beolvassa a hét minden munkanapján a leszedett alma mennyiségét t-ban, s az azt leszedő napi létszámot, majd számítsa ki, hogy az egyes napokon, illetve egész heti átlagban mennyit szedett egy átlagos dolgozó.

3.12 12 parcelláról zöldpaprikát szedünk le. Megjegyezzük az egyes parcellákról leszedett paprikák számát, illetve tömegét. Számítsuk ki az egyes parcellákra, illetve a 12 parcellára milyen tömegű paprika a jellemző!

3.13 Írjon programot, amely beolvassa egy 4 tagú család tagjainak nevét és havi jövedelmét, majd kiszámítja, hogy ki hány százalékkal részesedik a családi költségvetésből! Tételezzük fel, hogy mindenki mindent bead a közösbe!

4. LECKE: A PRINT-utasítás, egyszerű programok írása

Minta-utasítássorozatok:

1. sorozat:

```
NEW

 10 REM PISTIKE PELDAJA
 20 INPUT "MEDENCE HOSSZUSAGA(DM)";H
 30 INPUT "MEDENCE SZELESSEGE(DM)";S
 40 INPUT "MEDENCE MELYSEGE(DM)";M
 50 T=H*S*M
 60 PRINT "MEDENCE TERFOGATA:";T;"KOB DM AZAZ";T;"LITER"
```

2. sorozat:

```
10 REM PETI FAGYLALT-PROBLEMAJA
20 INPUT "PETI BARATAINAK SZAMA";B
30 INPUT "PETI ZSEBPENZE(FT)";FT
40 INPUT "1 GOMBOC FAGYLALT(FT)";G
50 NF%=FT/(G*(B+1))
60 PRINT "PETI";B;"BARATJANAK ES MAGANAK"
70 PRINT NF%;"GOMBOCOS FAGYLALTOT VEHET "
80 PRINT "A";FT;"FT-NYI ZSEBPENZEBOL"
```

3. sorozat:

```
10 REM AUTOSOK PROBLEMAJA
20 INPUT "AUTOMARKA";A$
30 INPUT "FAJLAGOS BENZINFOGYASZTAS(L/100 KM)";B
40 INPUT "BENZIN EGYSEGARA(FT/L)";A
50 BK=A*B
60 PRINT "AZ ON ";A$;" AUTOJA 100 KM UTAT"
70 PRINT BK;" FT BENZINKOLTSEGGEL TESZ MEG"
```

4. sorozat:

```
10 REM A CSOKIOSZTAS PROBLEMAJA
20 INPUT "ANDREA BARATNOINEK SZAMA";B
30 INPUT "ANDREA CSOKIJAINAK SZAMA";CS
40 INPUT "HANY SZELETET RAK FELRE AZ OCCSENEK";L
50 D%=(CS-L)/(B+1)
60 M%=CS-L-D%*(B+1)
70 PRINT "ANDREA ES BARATNOI FEJENKENT"
80 PRINT D%;"SZELETET KAPNAK"
90 PRINT " ES MEG MARAD";M%;"SZELET"
```

5. sorozat:

```
10 REM ATLAGOS TELJESITMENY KISZAMITASA
20 INPUT "HETFON DOLGOZOTT(FO)";D1
30 INPUT "LESZEDETT ALMA(T)";A1
40 INPUT "KEDDEN DOLGOZOTT(FO)";D2
50 INPUT "LESZEDETT ALMA(T)";A2
60 INPUT "SZERDAN DOLGOZOTT(FO)";D3
70 INPUT "LESZEDETT ALMA(T)";A3
80 INPUT "CSUTORTOKON DOLGOZOTT(FO)";D4
90 INPUT "LESZEDETT ALMA(T)";A4
100 INPUT "PENTEKEN DOLGOZOTT(FO)";D5
110 INPUT "LESZEDETT ALMA(T)";A5
120 PRINT CHR$(147)
130 PRINT "MUNKANAP","LETSZAM","ALMA T","A T L A G"
140 PRINT
150 PRINT "HETFO",D1,A1,A1/D1
160 PRINT "KEDD",D2,A2,A2/D2
170 PRINT "SZERDA",D3,A3,A3/D3
180 PRINT "CSUTORTOK",D4,A4,A4/D4
190 PRINT "PENTEK",D5,A5,A5/D5
200 PRINT
210 HD=D1+D2+D3+D4+D5
220 HA=A1+A2+A3+A4+A5
230 PRINT "A HET",HD,HA,HA/HD
240 PRINT
```

Megjegyzés: Itt, és a további mintaprogramjainkban azért szerepeltetjük a **NEW**-parancsot, hogy ezzel is felhívjuk a figyelmet arra, hogy ennek,vagy az ezzel egyenértékű parancsok alkalmazása feltétlenül szükséges ahhoz,hogy elkerüljük a programok értelmetlen egymásbaépülését. Ehhez hasonlóan ügyeljünk arra is, hogy minden programsort a **RETURN**-billentyű megnyomásával külön-külön olvastassunk el a C-64-gyel!

Az utasítássorozatok magyarázata:

1. sorozat:

10-es utasítás: Ún. megjegyzés-utasítás csak a programlistában szerepel, a program végrehajtása során semmi szerepe nincsen.

20-as utasítás: Az utasítás a "MEDENCE HOSSZUSAGA(DM)?"-üzenet kiírása után várja az aktuális medence hosszát dm-ben.

30-as utasítás: Az utasítás a "MEDENCE SZELESSEGE(DM)?"-üzenet kiírása után várja az aktuális medence szélességét dm-ben.
40-es utasítás: Az utasítás a "MEDENCE MELYSEGE(DM)?"-üzenet kiírása után várja az aktuális medence mélységét dm-ben.
50-es utasítás: Kiszámítja az aktuális medence térfogatát köbdm-ben.
60-as utasítás: Kinyomtatja az idézőjelben szereplő karaktersorozatokat és a közötte felsorolt T változó aktuális értékét a képernyőre.

A program végrehajtása után egy sor kihagyással megjelenik a READY.-üzenet, majd újabb sor elején a villogó kurzor.

2., 3., 4. sorozat:
Az 1. sorozathoz hasonlóan párbeszédes formában old meg egy-egy egyszerű feladatot. A 4 feladat konstrukciója teljesen hasonló.

5. sorozat:
10-es utasítás: Az előző programokhoz hasonlóan megjegyzésben hívja fel a figyelmet az utasítássorozat céljára.
20-110-utasítások: Párbeszédes formában olvastatja be a hét 5 munkanapján dolgozó almaszedők számát és a naponta leszedett alma mennyiségét.
120-as utasítás: A PRINT CHR$(147) olyan utasításkombináció, melynek hatására üres képernyőt kapunk. (Hatása egyenértékű a **CLR**-billentyű megnyomásával. A CHR$-függvénnyel majd a későbbi leckék során ismerkedünk meg részletesen.)
130-as utasítás: Hatására a képernyő következő, második sorában arányosan minden sornegyed elejére kiírja a "MUNKANAP", a "LETSZAM",az "ALMA T" és az "A T L A G"-szöveget.
140-es utasítás: Hatására a harmadik sor üresen marad.
150-190-es utasítás: Mind az 5 utasítás 4-4 információt ír ki a képernyőn soron következő sorba a 130-as sorhoz hasonló elosztásban, mégpedig a munkanap nevét, az aznapi munkáslétszámot, az aznap leszedett alma mennyiségét, majd a két utóbbiból számított fejenkénti átlagot.
200-as utasítás: A 140-eshez hasonlóan üresen hagy egy sort.
210-es utasítás: Kiszámítja, hogy összesen hányan dolgoztak a héten.
220-as utasítás: Kiszámítja, hogy mennyi almát szedtek összesen a héten.
230-as utasítás: A hét napjaihoz hasonlóan kinyomtatja a heti eredményeket.
240-es utasítás: Egy üres sorral választja el eredményünket az utána következő egyebektől.

A leckében előforduló új fogalmak:

A PRINT-utasítás:

Általános alakja:

sorszám PRINT output-lista

Az alkotóelemek és szerepük:

sorszám - az utasítás programbeli helyét jelöli ki. Ha elhagyjuk, akkor parancs-üzemmódú végrehajtásról beszélünk.

PRINT - a képernyőre-írás BASIC kulcsszava, mely helyettesíthető a ?-jellel. Ezt tettük eddigi példaprogramjainkban is. A két megadási mód azonosságát úgy bizonyíthatjuk be, hogy ?-jellel helyettesítjük a PRINT-et, majd a LIST paranccsal újra íratjuk a programlistát, s ekkor meglepődve fogjuk tapasztalni, hogy a ? helyén PRINT fog szerepelni.

általánosított output-lista - itt soroljuk fel azokat a karaktersorozatokat, amelyeket, illetve amelyeknek értékét a képernyőn viszont szeretnénk látni. Az egyes információk elhelyezéséről a különböző kiírást vezérlő jelek gondoskodnak (, ; továbbá nyomtatást vezérlő függvények: SPC, TAB, CHR$, stb.).

Az utasítás működési elve:

Megnézi, hogy melyik képernyőpozíció következik kiírásra. Ez általában a következő sor eleje. Ezután megnézi az output-lista első elemét:

- ha ez karaktersorozat vagy változónév, akkor hozzálát az aktuális információ képernyőre írásához,
- ha a soron levő elem vezérlőelem, akkor:
  - , esetén új sornegyedre ugrik,
  - ; esetén helyben marad,
  - más vezérlőelem (pl. CHR$(147)) esetén értelmezi a vezérlőelemet,s ennek megfelelően cselekszik.

Bármilyen output-lista elem végrehajtása után megpróbál áttérni a következőre, s ezt a fentebbi módon végrehajtani. Ha nincs több, akkor rátér a következő utasítás végrehajtására. Ha utoljára volt mit írnia, akkor új sorra vezérli az írásmutatót. Ha vezérlő utasítással zárult az output-lista, akkor a következő PRINT-et ebből a nyomtatási pozícióból indítja.

Képernyőtörlés: PRINT CHR$(147)

Üres sor kihagyása: PRINT

Megjegyzés elhelyezése a programban:

Általános alakja:

sorszám REM karaktersorozat

Az utasítás alkotóelemei és funkciójuk:

sorszám - megszabja a megjegyzés programbeli elhelyezését.

REM - a megjegyzést jelző BASIC alapszó

karaktersorozat - segédeszköz a programlista értelmezéséhez

Gyakorló feladatok:

4.1 Írjon programot, amely üres képernyőre kiírja az év hónapjainak nevét, napjainak számát, továbbá azt, hogy az adott hónapban milyen évszak(ok) van(nak). Minden információ új sornegyedben legyen. Mindegyik felett legyen kiírva a megfelelő "fejléc" pl. hónap, nap, évszak. A negyedéveket üres sorok válasszák el egymástól.

4.2 Írjon programot, amely az üres képernyőre kiírja a család tagjainak nevét, születési helyét és idejét, valamint pillanatnyi havi jövedelmét. Készítsen fejlécet is!

4.3 Írjon programot, amely beolvassa 10 paprikatermő parcella megnevezését, a termőterület nagyságát, az azon megtermelt paprika mennyiségét, majd a kapott adatokból az üres képernyőre készítsen olyan kibővített táblázatot, amely tartalmazza a parcellánkénti átlagtermést is.

4.4 Írjon programot, amely ékezetnélküli vagy távírati formában képernyőre ír egy rövid ismert verset.

4.5 Tervezze meg saját ékezetnélküli névjegy-kártyáját a képernyőre!

4.6 Írja meg egy gyorsbüfé étlapját a képernyőre.

4.7 Írjon esküvői meghívót a képernyőre.

4.8 Írjon hivatalos meghívót valamely társadalmi rendezvényre.

4.9 Írjon levelet hozzátartozójának, melyben valamiért kimenti magát.

4.10 Írjon programot, amely felsorolja, hogy mely napilapokra és folyóiratokra fizethet elő a magyar olvasó.

4.11 Írjon stilizált moziplakátot.

4.12 Írjon stilizált színlapot.

4.13 Írjon stilizált sport-eredménytáblázatot (Olimpia, ladbarúgó bajnokság stb...).

5. LECKE: Döntések, IF...THEN-utasításpár

Mintaprogramok:

1. program:

```
10 PRINT CHR$(147)
20 PRINT "CSALADI POTLEKOT SZAMITO PROGRAM"
30 PRINT
40 INPUT "GYERMEKEK SZAMA";GY%
50 CS=0
60 IF GY%=1 THEN CS=240
70 IF GY%=2 THEN CS=1420
80 IF GY%=3 THEN CS=2530
90 IF GY%=4 THEN CS=3360
100 IF GY%>4 THEN CS=3360+(GY%-4)*240
110 PRINT "A CSALADI POTLEK";CS;"FT"
```

2. program:

```
10 REM FUGGVENYTABLAZAT
20 PRINT CHR$(147)"X-ERTEKEK","PARABOLA"
30 X=-5
40 Y=X*X
50 PRINT X,Y
60 X=X+0.5
70 IF X<=5 THEN 40
```

3. program:

```
10 REM "KIHEGYEZETT" PARABOLA
20 PRINT CHR$(147)"X-ERTEKEK","PARABOLA"
30 X=-1.6
40 IF X*X>1 THEN Y=X*X
50 IF X*X<=1 THEN Y=X
60 IF Y<0 THEN Y=-Y
70 PRINT X,Y
80 X=X+0.2
90 IF X<=1.6 THEN 40
```

4. program:

```
10 REM ELOJEL KIERTEKELO PROGRAM
20 INPUT "KEREK EGY SZAMOT";X
30 PRINT X,
```

```
40 IF X>0 THEN PRINT "A SZAM POZITIV"
50 IF X=0 THEN PRINT "A SZAM NULLA"
60 IF X<0 THEN PRINT "A SZAM NEGATIV"
```

5. program:

```
10 REM GYUMOLCS-OSZTALYOZAS
20 INPUT "A GYUMOLCS ATMEROJE(MM)";D
30 IF D>=65 THEN PRINT "I. OSZTALYU"
40 IF D<65 AND D>=60 THEN PRINT "II. OSZTALYU"
50 IF D<60 AND D>=55 THEN PRINT "III. OSZTALYU"
60 IF D<55 THEN PRINT "OSZTALYOZATLAN"
```

A mintaprogramok magyarázata:

1. program:

Az első 4 utasítás előkészíti a feladat megoldását, s programozástechnikailag semmi újat nem tartalmaz. Az 5. utasítás 0 értéket ad az eredményváltozónak. Az ezt követő utasítások konstrukciója teljesen hasonló - az utolsót leszámítva. Mind az 5 utasítást egy IF ("ha"-jelentésű) BASIC-kulcsszó vezet be, melyet a GY%-változóra vonatkozó egyenlőség vagy egyenlőtlenség követ, majd a THEN ("akkor"-jelentésű) BASIC-kulcsszó után egy CS-változóra vonatkozó értékadás fejez be. Ha elolvassuk ezeket az utasításokat magyarul (pl. az elsőt így "ha GY%=1, akkor CS=240"), akkor máris megfejtettük a feltételes vagy döntési utasítás működési elvét. Az utolsó utasítás mondatba foglaltan megadja az aktuális családi pótlékot.

2. program:

Az első utasítás egy megjegyzés. A második utasítás üres képernyőre kiírja az első sornegyed fölé az "X-ERTEKEK", a második fölé a "PARABOLA"-feliratot. Ún. kezdő értéket ad az X-nek a harmadik utasítás. A negyedik utasítás kiszámítja az aktuális függvény-értéket. Az ötödik utasítás az X,Y-értékpárt kiírja a soron következő sor első két sornegyedébe. A hatodik utasítás megnöveli az X értékét 0.5-del. A hetedik utasítás megkérdezi, hogy X kisebb vagy egyenlő 5-tel. Ha igen, akkor folytassa a negyedik (40-es sorszámú) utasítás végrehajtásával. Ha nem, akkor a hetedik utáni utasítás következne.

3. program:

A 3. program nagyon hasonlít a 2.-hoz, csupán a függvényérték képzése bonyolultabb, összetettebb az előzőnél.

4. program:

A mintaprogram azt mutatja meg, hogy a C-64 bármely valós számnak meg tudja állapítani az előjelét, s ezt a vizsgált számmal együtt egy sorban hozza a programozó tudomására.

5. program:

Az utolsó mintaprogram a gyümölcs mm-ben megadott átmérőjének elolvasása után osztályozza a gyümölcsöt 4 osztályba: I., II., III. és osztályozatlan.

A leckében előforduló új fogalmak:

Relációk

Ha egy BASIC-programban két aritmetikai kifejezést a következő hat reláció-jelkombináció valamelyikével összekapcsolunk, akkor relációt hozunk létre, mely a legegyszerűbb logikai kifejezés. Mint ilyennek, a relációban résztvevő változók aktuális értékétől függően **igaz** vagy **hamis** értéket vehet fel (<,<=,>,>=,=,<>).

Logikai kifejezés:

BASIC-programjainkban a relációk, és a relációk között definiált logikai műveletek által jöhetnek létre logikai kifejezések, melyeknek az előzőekhez hasonlóan **igaz** vagy **hamis** értékei lehetnek.

Logikai műveletek:

NOT - az utána szereplő logikai kifejezés (reláció) értékének tagadása.

AND - az összekapcsolt két logikai kifejezés (reláció) egyidejű **igaz** voltát kívánja meg a végérték **igaz** voltához.

OR - az összekapcsolt két logikai kifejezés (reláció) egyikének bekövetkezését kívánja meg csupán a végérték **igaz** voltához.

Döntési (feltételes) utasítás:

Általános alakja:

sorszám IF logikai kifejezés THEN teendő

Az utasításban szereplő alkotóelemek és szerepük:

sorszám - meghatározza az utasítás programbeli helyét.

IF - az utasítás feltételét "bevezető" BASIC-alapszó.

logikai kifejezés - ennek **igaz** vagy **hamis** voltától függ, hogy végrehajtjuk-e az utasítás további (második felét).

THEN - az utasítás feltételesen végrehajtandó második felét "bevezető" BASIC alapszó.

teendő - lehet konkrét végrehajtandó utasítás(ok) vagy olyan sorszám,amelynél a program végrehajtását a feltétel teljesülése esetén folytatni kell.

Az utasítás működési elve:

A C-64 meghatározza az IF utáni logikai kifejezés aktuális értékét. Ha ez **igaz**, akkor a THEN utáni utasítást hajtja végre. Ha ez **hamis**, akkor a döntési utasítást követő utasítás következik.

Gyakorló feladatok:

5.1 Írjon programot, amely beolvassa egy dolgozó nevét, személyi számát és keresetét, majd eldönti a személyi szám alapján, hogy férfi vagy nő, 30 évesnél fiatalabb-e, és hogy keres-e legalább 3000 Ft-ot.

5.2 Írjon programot, mely beolvassa egy sportoló nevét, születésének évét, majd megállapítja a korcsoportját! (14 évesig serdülő, 20 évesig ifjúsági, 35 évesig felnőtt, utána pedig öregfiú).

5.3 Írjon programot, amely beolvassa egy öttagú család tagjainak nevét és korát, majd állapítsa meg, hogy ki a családban a legidősebb!

5.4 Írjon programot, amely beolvas 3 autómárkát és azt, hogy mennyit fogyasztanak az egyes autók 100 km-en. Válasszuk ki a legtakarékosabbat!

5.5 Írjon programot, amely beolvassa egy tanuló 5 osztályzatát, kiszámítja a tanulmányi átlagát, majd az iskolai szabályok szerint minősíti az eredményt.

5.6 Írjon programot, amely beolvassa egy csoport vizsgaeredményeit, és megszámolja, hogy hányan kaptak jelest! Legyen a csoport létszáma 13!

5.7 Írjon programot, amely beolvas 13 számot, és megszámolja, hogy hány pozitív, negatív és nulla van közöttük!

5.8 Írjon programot, amely beolvassa egy szálloda 13 vendégének nevét és állampolgárságát, majd állapítsa meg, hogy a vendégek közül hányan jöttek Franciaországból, s ez az összlétszám hány százaléka.

5.9 Írjon programot, amely beolvassa egy kollégium lakóinak számát, azok nevét, nemét, haja színét, szeme színét, majd számolja meg a barna hajú, kék szemű lányokat!

5.10 Írjon programot, amely elkészíti 3*X+4, 4*X*X+3*X-3 és a X*X*X-6*X függvények függvénytáblázatát -13 és 13 között X értékét 0.1-enként növelve.

5.11 Írja ki 1848 és 1964 között a 3-mal osztható számokat úgy, hogy egy sorba mindig 4 szám kerüljön.

5.12 Írassa ki a páros számokat 1241 és 1526 között úgy, hogy azok szorosan egymás mellé kerüljenek!

5.13 Írjon programot, amely kiírja az 1222 és 1964 közé eső 13-mal osztható számokat!

5.14 Írjon programot, amely 1986-tól visszaszámlál 1984-ig, s ezeket az értékeket kiírja a képernyőre.

6. LECKE: Programszervező utasítások: GO TO, END, GET

Mintaprogramok:

1. program:

```
10 REM TERULETKISZAMITO PROGRAM
20 PRINT CHR$(147)"TERULETKISZAMITO PROGRAM"
30 PRINT
40 PRINT "ADJA MEG AZ ALAKZAT NEVET!"
50 PRINT
60 PRINT "    1 - KOR"
70 PRINT "    2 - NEGYZET"
80 PRINT "    3 - TEGLALAP"
90 PRINT "    4 - HAROMSZOG"
100 PRINT "    5 - VEGE"
110 PRINT
120 INPUT "VALASZA";V%
130 IF V%=5 THEN END
140 IF V%=4 THEN 380
150 IF V%=3 THEN 310
160 IF V%=2 THEN 250
170 IF V%=1 THEN 190
180 GO TO 20
190 PRINT CHR$(147)"KORT VALASZTOTT"
200 PRINT
210 INPUT "A KOR SUGARA";R
220 IF R<=0 THEN 190
230 T=R*R*3.14159
240 GO TO 490
250 PRINT CHR$(147)"NEGYZETET VALASZTOTT"
260 PRINT
270 INPUT "A NEGYZET OLDALA";A
280 IF A<=0 THEN 250
290 T=A*A
300 GO TO 490
310 PRINT CHR$(147)"TEGLALAPOT VALASZTOTT"
320 PRINT
330 INPUT "A TEGLALAP HOSSZA";A
340 INPUT "A TEGLALAP SZELESSEGE";B
350 IF A<=0 OR B<=0 THEN 310
360 T=A*B
370 GO TO 490
380 PRINT CHR$(147)"HAROMSZOGET VALASZTOTT"
390 PRINT
400 INPUT "AZ A OLDAL";A
410 INPUT "A B OLDAL";B
420 INPUT "A C OLDAL";C
```

```
430 IF A<=0 OR B<=0 OR C<=0 THEN 380
440 IF A+B<=C THEN 380
450 IF A+C<=B THEN 380
460 IF B+C<=A THEN 380
470 S=(A+B+C)/2
480 T=(S*(S-A)*(S-B)*(S-C))↑0.5
490 PRINT
500 PRINT "AZ ALAKZAT TERULETE";T
510 I=1
520 I=I+1
530 IF I<=10000 THEN 520
540 GO TO 20
```

2. program:

```
10 PRINT CHR$(147)"PETI ORARENDJE"
20 PRINT
30 PRINT "MILYEN HET VAN?"
40 PRINT
50 PRINT "    A - A-HET"
60 PRINT "    B - B-HET"
70 PRINT
80 INPUT "VALASZOD";H$
90 IF H$<>"A" AND H$<>"B" THEN 80
100 PRINT
110 PRINT "MILYEN NAP VAN?"
120 PRINT
130 PRINT "   H   - HETFO"
140 PRINT "   K   - KEDD"
150 PRINT "   SZ  - SZERDA"
160 PRINT "   CS  - CSUTORTOK"
170 PRINT "   P   - PENTEK"
180 PRINT
190 INPUT "VALASZOD";N$
200 IF N$="H" OR N$="K" THEN 240
210 IF N$="SZ" OR N$="CS" THEN 240
220 IF N$="P" THEN 240
230 GO TO 190
240 PRINT CHR$(147)"PETI ORARENDJE  ";H$;"-HET  ";
250 IF N$="H" THEN D$="HETFO"
260 IF N$="K" THEN D$="KEDD"
270 IF N$="SZ" THEN D$="SZERDA"
280 IF N$="CS" THEN D$="CSUTORTOK"
290 IF N$="P" THEN D$="PENTEK"
300 PRINT D$
310 IF H$="B" THEN 710
320 IF N$="K" THEN 430
330 IF N$="SZ" THEN 500
340 IF N$="CS" THEN 570
```

```
350 IF N$="P" THEN 640
360 01$="TESTNEVELES"
370 02$="FOLDRAJZ"
380 03$="MATEMATIKA"
390 04$="OROSZ NYELV"
400 05$="FIZIKA"
410 06$="NEMET NYELV"
420 GO TO 1090
430 01$="BIOLOGIA"
440 02$="KEMIA"
450 03$="MAGYAR IRODALOM"
460 04$="TORTENELEM"
470 05$="NEMET NYELV"
480 06$="OSZTALYFONOKI"
490 GO TO 1090
500 01$="ENEK"
510 02$="TESTNEVELES"
520 03$="MATEMATIKA"
530 04$="MAGYAR NYELVTAN"
540 05$="FOLDRAJZ"
550 06$="TECHNIKA"
560 GO TO 1090
570 01$="TESTNEVELES"
580 02$="FIZIKA"
590 03$="MAGYAR IRODALOM"
600 04$="NEMET NYELV"
610 05$="RAJZ"
620 06$="RAJZ"
630 GO TO 1090
640 01$="OROSZ NYELV"
650 02$="BIOLOGIA"
660 03$="MAGYAR NYELVTAN"
670 04$="MATEMATIKA"
680 05$="TORTENELEM"
690 06$="              "
700 GO TO 1090
710 IF N$="K" THEN 820
720 IF N$="SZ" THEN 890
730 IF N$="CS" THEN 960
740 IF N$="P" THEN 1030
750 01$="KEMIA"
760 02$="NEMET NYELV"
770 03$="MATEMATIKA"
780 04$="OROSZ NYELV"
790 05$="TORTENELEM"
800 06$="TESTNEVELES"
```

```
810 GO TO 1090
820 01$="NEMET NYELV"
830 02$="FOLDRAJZ"
840 03$="BIOLOGIA"
850 04$="MAGYAR NYELVTAN"
860 05$="OROSZ NYELV"
870 06$="                "
880 GO TO 1090
890 01$="TECHNIKA"
900 02$="TECHNIKA"
910 03$="FIZIKA"
920 04$="MAGYAR IRODALOM"
930 05$="MATEMATIKA"
940 06$="OSZTALYFONOKI"
950 GO TO 1090
960 01$="FOLDRAJZ"
970 02$="TESTNEVELES"
980 03$="OROSZ NYELV"
990 04$="KEMIA"
1000 05$="TORTENELEM"
1010 06$="NEMET NYELV"
1020 GO TO 1090
1030 01$="MATEMATIKA"
1040 02$="BIOLOGIA"
1050 03$="NEMET NYELV"
1060 04$="ENEK"
1070 05$="MAGYAR NYELVTAN"
1080 06$="MATEMATIKA"
1090 PRINT
1100 PRINT "ORA","TANTARGY"
1110 PRINT
1120 PRINT " 8- 9",01$
1130 PRINT
1140 PRINT " 9-10",02$
1150 PRINT
1160 PRINT "10-11",03$
1170 PRINT
1180 PRINT "11-12",04$
1190 PRINT
1200 PRINT "12-13",05$
1210 PRINT
1220 PRINT "13-14",06$
1230 PRINT
1240 GET T$: IF T$="" THEN 1240
1250 IF T$<>"V" THEN 10
1260 END
```

A mintaprogramok magyarázata:

1. program:

Mint a 10-es megjegyzés és a 20-as bejelentkező utasítás is utal rá,a program a matematikában gyakran előforduló területszámítási feladatok megoldására készült. A 30-110 utasítássor írja ki a képernyőre a választható síkidomokat és azok választási módiát (pl. az 1-válasz kört jelent). A felhasználó válaszát a 120-as utasítás tölti be a V% változóba. Ezután a 130-180 döntési utasítássor választja ki a megfelelő teendőt V% értékétől függően. Ha V% 5, akkor befejezzük a programot. Ezt jelenti az END alapszó. Minden más esetben a THEN utáni sorszám mondja meg, hogy hol folytassuk a program végrehajtását. Ha V% nem veszi fel az 1,2,3,4,5 értékek egyikét sem, akkor a 180-as utasítás a program elejére küld a GO TO 20-utasítással. A feldolgozott 4 eset a következő:

KÖR:

A 190-es utasítás kiírja a választást, a következő üres sort hagy, majd beolvassa a kör sugarát. Ezt követi egy adatellenőrzés: ha a sugár nem pozitív, akkor újat kér. A 230-as utasítás számítja ki a terület értékét, majd áttér a 490-es utasításra.

NÉGYZET:

A 250-es utasítás kiírja a választást, a következő üres sort hagy, majd beolvassa a négyzet oldalának hosszát. Ezt követi egy adatellenőrzés: ha az oldal hossza nem pozitív, akkor új adatot kér. A 290-es utasítás számítja ki a négyzet területét, majd átirányít a 490-es utasítás végrehajtására.

TÉGLALAP:

A 310-es utasítás kiírja a választást, a következő üres sort hagy, majd két lépésben beolvassa a téglalap két oldalát. Ezt követi az adatok ellenőrzése: csak pozitív adatokat fogadunk el.'A 360-as utasítás számítja ki a téglalap területét. Majd itt is a 490-es sor "következik".

HÁROMSZÖG:

A 380-as utasítás kiírja a választást, a következő üres sort hagy, majd a következő 3 utasítás beolvassa a 3 oldal hosszát. Ezt követi az oldaladatok ellenőrzése 4 lépésben. A 470-es utasítás a háromszög félkerületét,majd a 480-as a Héron-képlet segítségével a területét számítja ki.

A terület kiírása és újabb alakzat kérése:

A 490-es sor üres sort hagy, az 500-as sor pedig kiírja a képernyőre az éppen kiszámított területet. Az 510-530-as utasítássor 1-től 10000-ig elszámlál, s ezalatt - mintegy 10 másodpercig - szemlélhetjük az aktuális terület eredményét,mielőtt újabb alakzat területének kiszámítására térnénk át. Ezt az 540-es utasítás fogja elérni.

2. program:

10-es utasítás: bejelentkezik a program - leírva végrehajtásának célját.
20-70-es utasítássor: kiírja a képernyőre a ma már országosan elterjedt választási lehetőséget, hogy A- és B-hét váltja egymást.
80-as utasítás: beolvassa az aktuális hétválasztást a H$ változóba.
90-es utasítás: ha nem jól válaszoltunk, akkor új hetet kérdez (Ezt jelenti a THEN utáni 80-as sorszám.)
100-180-as utasítássor: a hét kiválasztása után a nap kiválasztása következik. Az itteni lehetőségeket sorolja fel ez az utasítássor.
190-es utasítás: ez olvassa be az N$ változóba a kiválasztott napot.
200-230-as utasítássor: gondoskodik arról, hogy csak helyes válasz maradhasson az N$ változóban.
240-300-as utasítássor: gondoskodik arról, hogy üres képernyőn megjelenjen eddigi választásunk eredménye.
310-es utasítás: ha B-hetet választottunk, akkor elküld a 710-es utasítás végrehajtására.
320-as utasítás: ha A-hét keddjét választottuk, akkor elküld az órarendet kitöltő 430-490 utasítássorra.
330-as utasítás: ha A-hét szerdáját választottuk, akkor elküld az órarendet kitöltő 500-560 utasítássorra.
340-es utasítás: ha A-hét csütörtökjét választottuk, akkor elküld az órarendet kitöltő 570-630 utasítássorra.
350-es utasítás: ha A-hét péntekjét választottuk, akkor elküld az órarendet kitöltő 640-700 utasítássorra.
710-1080-as utasítássor: ugyanazt teszi, mint az A-hét esetében a 320-680 utasítássor. A 690-es sor megfelelőjére itt nincs szükség, hiszen itt már rögtön következik az aktuális napi órarend kiírása a képernyőre.
1090-1230-as utasítássor: kiírja a képernyőre az aktuális napi órarendet.
1240-es utasítás: feltételes várakozást hoz létre.
1250-es utasítás: ha a feltételes várakozást nem a v-billentyű megnyomásával szakítottuk meg, akkor Peti órarendjének újabb lapját kívánjuk megismerni.
1260-as utasítás: a program befejezése.

A leckében előforduló új fogalmak:

A GO TO-utasítás:

Általános alakja:

sorszám GO TO sorszám

vagy

sorszám IF logikai kifejezés THEN GO TO sorszám

Az alkotóelemek és azok szerepe:

sorszám - mindig a program egy utasításának sorszáma.

GO TO - az a BASIC alapszó, amely az utána szereplő sorszámú utasítás végrehajtását kívánja meg a GO TO-t tartalmazó utáni helyett.

Megjegyzésl: A GO TO-t tartalmazó második alak ismerősnek tűnhet, hiszen az csupán egy bővített döntési utasítás, ahol kihangsúlyozzuk, hogy a GO TO utáni sorszámú utasítás következik.

Megjegyzés2: A C-64 elfogadja a GOTO alakot is.

Az END-utasítás:

Általános alakja:

sorszám END

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét jelöli ki.

END - BASIC alapszó, amely a program végrehajtásának befejezését idézi elő.

A GET-utasítás:

Általános alakja:

sorszám GET változónév

Az utasítás alkotói és szerepük:

sorszám - az utasítás programbeli helyét határozza meg.

GET - BASIC alapszó, amelynek hatására a billentyűzeten megnyomott billentyű értékét beteszi az utána szereplő változóba. (Nem kell **RETURN**!)

változónév - ez fogja tartalmazni a beolvasott egyetlen jelű(!) értéket.

A feltételes várakozás:

Általános alakja:

sorszám GET karakter-változó: IF karakter-változó="" THEN sorszám

AZ UTASÍTÁS LÉNYEGE, HOGY A KÉTSZER SZEREPLŐ FOGALMAK (SORSZÁM, KARAKTER-VÁLTOZÓ) AZONOSAK!

Az utasítás működési elve:

A karakter-változó induló értéke "". Így amikor az utasításhoz ér a program, akkor a feltétel teljesülése miatt saját magát fogja ismételni mindaddig, amíg valamelyik billentyű megnyomása után a karakter-változó más értéket nem kap. Ekkor a feltétel nem teljesül, s a program a következő utasítással folytatódik.

Javasoljuk, hogy programjainkban alkalmazzuk a lehetséges válaszokat felsoroló ún. menűket, mert ezzel lényegesen megkönnyítjük a programjainkat felhasználók munkáját.

Gyakorló feladatok:

6.1 Írjon programot, amely az 1. mintaprogramhoz hasonlóan ki tudja számítani a kocka, a téglatest, a tetraéder, a gömb felszínét!

6.2 Írjon programot, amely a hazánkat környező országok közül (Ausztria, Csehszlovákia, Jugoszlávia, Románia, Szovjetunió) kiválaszt egyet, és annak 5 lényeges adatát a képernyőre írja! (Használjuk fel a 2. mintaprogram megfelelő szerkesztési ötleteit!)

6.3 Írjon programot, amely kívánságra kiírja az 5 NB I-es labdarúgó-csapat leggyakoribb összeállítását!

6.4 Írjon programot, amely kiírja családja (csoportja, osztálya) tagjainak hat legfontosabb adatát! Természetes, hogy a 2. mintaprogramhoz hasonlóan mindig csak egy személy (a kiválasztott) adatai legyenek a képernyőn!

6.5 Írjon programot, amely a feltételes várakoztatást is felhasználva kiírja a képernyőre az X*X+2, az X*X-2*X+3 és a 6*X-3 függvények értéktáblázatát 3-tól 7-ig 0.1-es lépésközzel!

6.6 Általánosítsuk a 6.5-ös feladatot úgy, hogy az első(a), az utolsó(b) behelyettesítési értéket, valamint a lépésközt(dx) a billentyűzetről olvassuk be.

6.7 Írjon programot, amely "lapozható" étlapot varázsol a képernyőre!(Előételek, aperitifek, levesek, frissen sültek, készételek, desszertek, frissítők, italok, gyümölcsök stb...).

6.8 Írjon programot, mely választás után kiírja néhány író leghíresebb 7 művét!

7. LECKE: Intelligens INPUT- és PRINT-utasítások

Mintaprogramok:

1. program:

```
10 PRINT "■SORSZERKESZTES"
20 A$="A": B$="B": C$=" "
30 PRINT X$;A$,B$
40 X$=X$+C$: I=I+1
50 IF I<5 THEN 30
60 I=0: X$=""
70 A$="A": B$="B": C$="."
80 PRINT X$;A$,B$,A$,B$
90 X$=X$+C$: I=I+1
100 IF I<5 THEN 80
110 I=0: X$=""
120 A$="A": B$="B": C$="."
130 PRINT X$;A$;B$,A$;B$,C$
140 X$=X$+C$: I=I+1
150 IF I<5 THEN 130
160 I=0: X$=""
170 A$="A": B$="B": C$="."
180 PRINT A$;X$;B$,A$;B$,C$
190 X$=X$+C$: I=I+1
200 IF I<5 THEN 180
210 END
```

2. program:

```
10 REM KEPERNYORE IRAS ADOTT POZICIOBAN
20 PRINT "■"
30 PRINT " FERENCZY  ANTAL"
40 PRINT " BUDAORS"
45 PRINT " --------"
50 PRINT " SZELES II.1"
60 PRINT " 2040"
70 PRINT
80 I=I+1
90 IF I<5 THEN 70
100 PRINT ,,"SZELID ANNA"
105 PRINT ,,"**********"
110 I=0
120 PRINT
130 I=I+1
140 IF I<6 THEN 120
```

```
150 PRINT ,,"        ZEBEGENY"
160 PRINT ,,"        ---------"
170 PRINT ,,"        REMENY U. 13"
180 PRINT
190 PRINT ,,"        2135"
200 GET A$: IF A$="" THEN 200
210 END
```

3. program:

```
10 PRINT "■   F U G G V E N Y T A B L A Z A T"
20 PRINT
30 PRINT "       A FUGGVENY   F(X)=(1+1/X)↑X"
40 PRINT
50 PRINT "       X              F(X)"
60 PRINT
70 X=1.3
80 Y=(1+1/X)↑X
90 PRINT "     ";X,"       ";Y
100 X=X+0.1
110 IF X<=3 THEN 80
120 GET V$: IF V$="" THEN 120
130 END
```

4. program:

```
10 REM UDVOZLOKARTYA KIIRASA KEPERNYORE
20 PRINT "■"
30 PRINT: PRINT: PRINT
60 PRINT "  KELLEMES KARACSONYI UNNEPEKET"
70 PRINT
80 PRINT ,"     ES"
85 PRINT
90 PRINT ,"■■BOLDOG UJ EVET"
100 PRINT: PRINT: PRINT
110 PRINT ,"    KIVAN"
120 PRINT: PRINT: PRINT
140 PRINT ,"■■■SOK-SOK SZERETETTEL"
150 PRINT: PRINT
160 PRINT ,,,"TONI"
170 GET S$: IF S$="" THEN 170
180 END
```

5. program:

```
10 REM MINI TELEFONKONYV
20 PRINT "■OLVASSA BE 9 ISMEROSE NEVET"
30 PRINT ,"ES TELEFONSZAMAT!"
40 PRINT
50 INPUT "1. NEV,TELEFON:";N0$,T0$
60 INPUT "2. NEV,TELEFON:";N1$,T1$
70 INPUT "3. NEV,TELEFON:";N2$,T2$
80 INPUT "4. NEV,TELEFON:";N3$,T3$
90 INPUT "5. NEV,TELEFON:";N4$,T4$
100 INPUT "6. NEV,TELEFON:";N5$,T5$
110 INPUT "7. NEV,TELEFON:";N6$,T6$
120 INPUT "8. NEV,TELEFON:";N7$,T7$
130 INPUT "9. NEV,TELEFON:";N8$,T8$
140 PRINT
150 INPUT "HOGY HIVJAK ONT";N$
160 PRINT "■■■";N$;" MINI TELEFONKONYVE"
170 PRINT " ---------------------------------------- "
180 PRINT " NEV",,"TELEFONSZAM"
190 PRINT " ---",,"----------"
200 PRINT
210 PRINT " ";N0$
220 PRINT "■",,T0$
230 PRINT
240 PRINT " ";N1$
250 PRINT "■",,T1$
260 PRINT
270 PRINT " ";N2$
280 PRINT "■",,T2$
290 PRINT
300 PRINT " ";N3$
310 PRINT "■",,T3$
320 PRINT
330 PRINT " ";N4$
340 PRINT "■",,T4$
350 PRINT
360 PRINT " ";N5$
370 PRINT "■",,T5$
380 PRINT
390 PRINT " ";N6$
400 PRINT "■",,T6$
410 PRINT
420 PRINT " ";N7$
430 PRINT "■",,T7$
440 PRINT
450 PRINT " ";N8$
460 PRINT "■",,T8$
470 GET T$: IF T$="" THEN 470
480 PRINT "■"
490 END
```

A mintaprogramok magyarázata:

1. program:

10-es utasítás: Az üres képernyőre kiírja "SORSZERKESZTES" szöveget. Az utasítás újdonsága a képernyőtörlés kérésének újszerű módja: az " után együtt nyomjuk meg a SHIFT és a CLR billentyűt.

20-as utasítás: Az A$, B$ és C$ karaktertípusú változók felveszik a felsorolt értékeket. Az utasítás újdonsága, hogy egy sorszámmal 3 értékadó utasítást adtunk meg. Az utasítások elválasztásáról a : gondoskodik.

30-as utasítás: Kiírja a képernyőre az X$, az A$ és a B$ aktuális értékét úgy, hogy X$ és A$ a ; hatására szorosan egymás mellé kerüljenek, a B$ pedig az előtte levő vessző hatására a második sornegyed elejére kerül. Mivel X$ kezdeti értéke "", ezért először az A a sor elejére kerül.

40-es utasítás: Ez az értékadás-pár X$ karaktereit megtoldja egy szóközzel (hiszen ez C$ tartalma), az I értékét pedig megnöveli 1-gyel.

50-es utasítás: Olyan feltételes utasítás, mely I értékétől függően a 30-as vagy a következő (60-as) utasítással "folytattatja" a programot. Ha a 30-assal, akkor az X$ változása miatt más és más nyomtatási képet kapunk, az A betű jobbra vándorol.

60-as utasítás: I és X$ kezdeti értékének (0,"") visszaállításával előkészíti az újabb kiírási formát.

70-es utasítás: A 20-ashoz hasonlóan kezdeti értéket ad A$-nak, B$-nak és C$-nak (A, B, .).

80-as utasítás: Képernyőre írja X$, A$, B$, A$ és B$ értékét. A kiírás helyét az egyes változókat elválasztó írást vezérlő jelek határozzák meg.

90-es utasítás: Ugyanaz mint a 40-es.

100-as utasítás: Nagyon hasonló az 50-eshez.

110-150-es utasítások: Újabb kiírási kombinációt mutatnak be.

160-200-as utasítások: Újabb kiírási kombinációt mutatnak be.

Figyeljük meg az egyes kiírási utasítások (30, 80, 130, 180) alakját és a hatásukra létrejött sorokat. Egy-egy sor is szemléletesen illusztrál, de pontosabb fogalmat nyerünk a , és a ; írásvezérlő szerepéről, ha mindig 5 hasonló sorból vonjuk le következtetésünket.

2. program:

A levélpostai forgalomban előírt formát utánozzuk az előző programban megismert képernyőtörlés (SHIFT és CLR), az üres sor írása (PRINT), a sornegyed-váltás (,) és az "-jelek között szerepeltetett szóköz segítségével.

3. program:

Ez a program egy bonyolultnak tűnő függvény értéktáblázatát írja ki a képernyőre a képernyőtörlés (SHIFT és CLR), a sornegyed-váltás (,) és a "-jelek között szerepeltetett szóközök segítségével.

4. program:

Ez a program egy karácsonyi üdvözlőkártyát varázsol a képernyőre. A program újdonsága a 80-as és 120-as sorban az "-jel után a balra mutató **CRSR** billentyű beprogramozása, melynek hatására a jelzett szövegek kiírása 1 illetve 3 pozícióval korábban kezdődik.

5. program:

Az első 15 sor párbeszédes formában információkat kér a felhasználótól. A 160-as utasítás a képernyőtörlés (**SHIFT** és **CLR**), lefelé és jobbra mutató **CRSR** "-jel utáni beprogramozásával eléri, hogy a MINI TELEFONKONYV tulajdonosának neve a keret (egy sor és egy oszlop) kihagyásával legyen kiírva. Az utasítás további része semmi újat nem tartalmaz. A program utolsó újdonsága (220,250,280,... sorok), hogy a telefonszámok fix kezdő pozícióját úgy érjük el, hogy látszatra külön sorban nyomtatjuk, de ez a külön sor a **CRSR** felfelé mutató nyilával "kezdődik", amely azt eredményezi, hogy az előző sor egyes részeit újra írjuk.

A leckében előforduló új fogalmak:

Képernyőre írást vezérlő jelek:

A kurzor mozgató (vezérlő) jelek használatát már a billentyűzet megismerése közben kitapasztalhattuk. Ezek a jelek nemcsak a programírást, hanem a program végrehajtását is megkönnyíthetik, ha őket a megfelelő helyen a szövegkonstans tagjaként (az "-jelek közé írt jelek közé) beprogramozzuk. Hatásuk a végrehajtás során pontosan az lesz, mint a normál üzemmódban.

Jelkombináció	Hatás
karaktersorozaton belül "-jelek között:	
HOME	A kurzort a bal felső sarokba viszi
SHIFT és **CLR**	Törli a képernyőt és a kurzort a bal felső sarokba állitja
CRSR-nyilak	A kurzort a megadott irányba eggyel elléptéti
a változó-lista elemei között:	
,	A kurzort a következő sornegyed elejére viszi
;	A következő pozícióra állítja a kurzort.

Az idézőjelek között szerepeltetett szóköz teljes értékű írásjel!!!
INPUT-utasításban pontosan annyi értéket kell beolvastatnunk, mint amennyi változónevet felsoroltunk.

Gyakorló feladatok:

7.1 Írjon programot, amely felváltva A betűket és szóközöket ír folyamatosan az üres képernyőre!

7.2 Írjon programot, amely "lépcsősen" 20 soron keresztül kiírja a képernyőre a nevét!

7.3 Írjon programot, amely az üres képernyőre minden második sor 11. pozíciójától kezdve kiírja az "Alma a fa alatt, nyari piros alma" szöveget.!

7.4 Írjon programot, amely üres képernyőre minden harmadik sor 6. pozíciójától kezdve kiírja a "Beethoven: IX. szimfonia" szöveget!

7.5 Írjon programot, amely táblázatot készít hazánk 19 megyéjéről! A fejléc a második sorban, a megyék neve a 3., a megyeszékhelyek neve a 19. pozíción szerepeljen. Gondoskodjék a táblázat zavartalan olvasásáról a feltételes várakozó utasítás segítségével!

7.6 Írjon programot, amely a képernyő keretét (első és utolsó sor, első és utolsó oszlop) üresen hagyja, s eggyel beljebb *-okból álló keretet rajzol! Gondoskodjon arról, hogy a képernyőt a program vége ne rontsa el!

7.7 A PRINT utasítás segítségével rajzoljon dominót a képernyőre!

7.8 A PRINT utasítás segítségével a lehető legnagyobb méretben írja ki a képernyőre a "C-64" feliratot!

7.9 Eddigi eszközei felhasználásával készítse el egy lakószoba stilizált tervrajzát!

7.10 Fejlessze tovább tervrajzát egy egyszerű lakássá!

7.11 Szervezze meg, hogy a képernyőn végigfusson egy tetszőleges szöveg!

7.12 Készítsen el egy stilizált Magyarország térképet a képernyőn!

7.13 Készítsen olyan programot, amely menű alapján 4-5 esetből választja ki a képernyőn stilizáltan ábrázolni kívánt tárgyat, s annak rajzát a feltételes várakozás segítségével tetszés szerinti ideig szemlélni engedi.

8. LECKE: Ismétlődő programrészek, ciklusszervezés

Mintaprogramok:

1. program:

```
10 PRINT "■SZAMOK","NEGYZETUK","KOBUK""
20 PRINT
30 FOR I=1 TO 19
40 PRINT "   ";I,"   ";I*I,"    ";I*I*I
50 NEXT I
60 GET V$: IF V$="" THEN 60
70 PRINT "■"
80 END
```

2. program:

```
10 A$="          "
20 B$="         "
30 C$="        "
40 D$="       "
50 E$="      "
60 F$="     "
70 G$="    "
80 H$=""
90 PRINT "■    SZAMOK    NEGYZETUK    KOBUK"
100 PRINT
110 FOR I=1 TO 19
120 K=1
130 FOR J=1 TO 3
140 K=K*I
150 X$=A$
160 IF K>=10 THEN X$=B$
170 IF K>=100 THEN X$=C$
180 IF K>=1000 THEN X$=D$
190 IF K>=10000 THEN X$=E$
200 IF K>=100000 THEN X$=F$
210 IF K>=1000000 THEN X$=G$
220 IF K>=10000000 THEN X$=H$
230 PRINT X$;K;
240 NEXT J
250 PRINT
260 NEXT I
270 GET V$: IF V$="" THEN 270
280 PRINT "■"
290 END
```

3. program:

```
10 REM CSILLAG PIRAMIS
20 PRINT "■":A$="*"
30 FOR X=0 TO 11
40 PRINT ,,X$;A$
50 A$=A$+"**"
60 X$=X$+"■■"
70 PRINT
80 NEXT X
90 GET V$: IF V$="" THEN 90
100 END
```

4. program:

```
10 REM KETSZERES CIKLUS
20 A$="*": PRINT "■"
30 FOR I=0 TO 23
40 Y$=""
50 FOR J=0 TO 38
60 PRINT "■";X$;Y$;A$
70 Y$=Y$+"■■"
80 PRINT "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■SOR=";I,"OSZLOP=";J;
90 NEXT J
100 X$=X$+"■"
110 NEXT I
120 GET V$: IF V$="" THEN 120
130 PRINT "■"
140 END
```

5. program:

```
10 PRINT "■": J=1
20 FOR X=0 TO 300
30 FOR Y=0 TO I
40 PRINT "■■";
50 NEXT Y
60 PRINT "* *  *  * * "
70 I=I+J
80 IF I=18 THEN J=-J
90 IF I=0 THEN J=-J
100 NEXT X
```

A mintaprogramok magyarázata:

1. program:

10-es utasítás: Hatására az üres képernyő első sorában megjelenik a SZAMOK, NEGYZETUK, KOBUK-felirat az első sor első, második és harmadik sornegyedének elején.
20-as utasítás: Üres sor kiírása.
30-as utasítás: Az ún. **ciklusfej**, amely először egyenértékű az I=1 értékadással, később pedig az I=I+1 értékadást és az aktuális I>19 feltétel megvizsgálását jelenti. Ha ez utóbbi teljesül, akkor a NEXT I utáni utasítással folytatja a program végrehajtását. Ha nem, akkor a ciklusfej utáni utasítás következik.
40-es utasítás: A megadott forma szerint kiírja az aktuális I, I*I és I*I*I értékét.
50-es utasítás: A **ciklus záró utasítás**, mely megkeresi a hozzá tartozó ciklusfejet (Esetünkben ez a 30-as). A FOR és NEXT utasításpár között elhelyezkedő, többszöri ismétlésre kerülő utasítás(oka)t nevezzük a **ciklus mag**jának.
60-as utasítás: A már megszokott késleltető utasítás, amellyel elérjük,hogy zavartalanul szemlélhessük a program végrehajtásának eredményét.
70-es utasítás: Letörli a képernyőt.
80-as utasítás: Befejezi a program végrehajtását.

2. program:

10-80-as utasítás: Különböző hosszúságú szóköz konstansokat állít be.
90-es utasítás: Az előző program 10-es utasításához hasonlóan üres képernyőre kiírja a SZAMOK, NEGYZETUK, KOBUK szöveget.
100-as utasítás: Üres sort hagy ki.
110-es utasítás: Az előző program 30-as utasításához hasonlóan megszervezi, hogy a következő utasításokat a NEXT I-t tartalmazó 260-asig bezárólag 19-szer hajtsa végre I=1,2,...,19 értékek mellett.
120-as utasítás: Kezdőértéket ad K-nak.
130-as utasítás: Megszervezi, hogy a 240-es NEXT J-ig bezárólag 3-szor hajtsa végre az utasításokat (minden I érték mellett!) J=1,2,3.
140-es utasítás: Aktualizálja a kiírandó K értékét.
150-220-as utasítás: Egy döntéssorozat eredményeként beállítja X$-ba, hogy hány szóköz előzze meg az aktuális K-érték kiíratását.
230-as utasítás: Az aktuális K érték helyiértékes kiíratása.
240-es utasítás: A J-szerinti ciklus befejezése.
250-es utasítás: A megkezdett képernyősor befejezése.
260-as utasítás: Az I-szerinti ciklus befejezése.
270-es utasítás: Feltételes várakozás.
280-as utasítás: Képernyőtörlés.
290-es utasítás: Program befejezése.

3. program:

10-es utasítás: Megjegyzés.
20-as utasításpár: Képernyőtörlés és A$ kezdőértékének megadása.
30-as utasítás: Annak megszervezése, hogy a következő utasításokat 12-szer (I=0,1,2,...,11) hajtsa végre a C-64.

40-es utasítás: Az X$ és A$ aktuális tartalmának kiíratása.
50-es utasítás: Az A$ tartalmának megtoldása két csillaggal.
60-as utasítás: Az X$ tartalmának megtoldása egy balra mozgató kurzorjellel.
70-es utasítás: Üres sort hagy.
80-as utasítás: Az X-szerinti ciklus befejezése.
90-es utasítás: Feltételes várakozás.
100-as utasítás: A program befejezése.

4. program:

10-es utasítás: Megjegyzés.
20-as utasításpár: A$ értéket kap, s képernyőt törlünk.
30-as utasítás: I-szerinti ciklust szervez meg, mely 0-tól 23-ig változik - jelezve, hogy melyik sorba fogunk írni.
40-es utasítás: Y$ - az oszlopmutató - induló értékként ""-t kap.
50-es utasítás: Megszervezi a J-szerinti ciklust, amely megmutatja, hogy melyik oszlopba szánjuk a csillagot.
60-as utasítás: A bal felső sarokba lép, majd onnan lép a megfelelő helyre az X$ és Y$ segítségével, majd kiíratja a megfelelő értéket.
70-es utasítás: Y$-t megtoldja egy jobbra vezérlő kurzormozgató jellel.
80-as utasítás: A bal felső sarokból 24-et lefelé lépve kiírja a "SOR="-üzenet után az I, az "OSZLOP=" után pedig J aktuális értékét, majd a sor-emelés megakadályozását szolgálja a ; szerepeltetése.
90-es utasítás: Lezárja a J-szerinti ciklust.
100-as utasítás: Megtoldja X$ értékét egy lefelé mozgató kurzor jellel.
110-es utasítás: Befejezi az I szerinti ciklust.
120-as utasítás: Feltételes várakozás.
130-as utasítás: Törli a képernyőt.
140-es utasítás: Befejezi a programot.

5. program:

A program formailag nem tartalmaz semmi újat, csak a kapott eredmény lesz mozgalmas.

A leckében előforduló új fogalmak:

A FOR...TO...STEP, NEXT utasításpár:

Általános alakja:

sorszám FOR változónév=kezdőérték TO végérték STEP lépésköz

sorszám(ok) utasítás(ok)

sorszám NEXT változónév

Az utasítássor alkotói és szerepük:

sorszám - az utasítás programbeli helyét határozza meg.

FOR - 'tól'-jelentésű BASIC alapszó, mely mintegy "bevezeti" a ciklust.

változónév - olyan ún. egyszerű változó, amely értékével és annak változásával vezérli az utána következő utasítássor (ciklusmag) végrehajtását.

= - értékadó egyenlőségjel.

kezdőérték - formailag aritmetikai kifejezés, amelynek kiszámított értékét veszi fel először a ciklus végrehajtását vezérlő változó.

TO - 'ig' jelentésű BASIC alapszó, mely az utána következő aritmetikai kifejezés aktuális értékével "mondja meg", hogy legfeljebb milyen értékig változzon a ciklust vezérlő változó.

STEP - 'lépés' jelentésű BASIC alapszó, amely megadja, hogy a ciklust vezérlő változó milyen mértékben változzon. (Ha mint példáinkban az utána szereplő paraméterrel együtt elhagyjuk, akkor értéke 1, vagyis a ciklusváltozó mintegy számolja, hogy hányadik végrehajtásnál tartunk.

lépésköz - aritmetikai kifejezés, amelynek aktuális értéke szabja meg, hogy mennyivel változzon a ciklusváltozó értéke az adott értékhatárok között (kezdőérték-végérték).

NEXT - 'következő' jelentésű BASIC alapszó, melynek hatására a program megkeresi azt a legközelebbi FOR-TO(-STEP)-utasítást, amelyet a NEXT után felsorolt változónévvel elkezdett.

A ciklus végrehajtásának működési elve

A program a FOR-TO(-STEP)-utasítás első végrehajtásával "lép be" a ciklusba, melynek során a ciklusváltozó felveszi az utána felírt aritmetikai kifejezés értékét, majd rátér a következő utasítás végrehajtására, mely a ciklusmag első - az esetek többségében egyetlen - eleme. Ezután sorban végrehajtja az utasítás(oka)t mindaddig, amíg nem találkozik azzal a NEXT utasítással, amely a ciklusváltozóra hivatkozik. Ekkor visszatér a ciklust definiáló FOR-TO(-STEP)-utasításra. Megnöveli a ciklusváltozó értékét a valamilyen módon megadott növekménnyel. Ezután megvizsgálja, hogy a kapott érték a kezdőérték és végérték által megadott értékhatárok közé esik-e. (Persze a végértéket is megengedi.) Ha igen, akkor a FOR-TO(-STEP) utáni,ha nem, akkor a NEXT utáni utasítással folytatódik a feladat.

Megjegyzés: Vigyázzunk arra, hogy a ciklusváltozó értékét csak a FOR-TO(-STEP)-utasítás változtathatja meg. Cikluson kívül ne használjuk a ciklusváltozó régi értékét, mert ez nagyon sok programhiba oka lehet. Bonyolultabb, összetettebb feladatok megtervezésénél kövessük a "skatulya"-elvet, ahogy ezt a 2., 4., és 5. programban magunk is tettük.

Javaslat: Ebben a leckében a számítástechnika egyik legbonyolultabb fogalmát vezettük be, mely szinte valamennyi feladatban szerepet játszik. Javasoljuk, hogy csak akkor lépjen tovább, ha ezt a fogalmat, ezt a "fegyvert" teljes egészében elsajátította. Az elmélyedésre a következők valamelyikét javasoljuk: oldja meg az itt következő feladatokat kétféle módon:

- az IF-THEN utasítással,
- majd a FOR-TO(-STEP), NEXT utasítás-párral.

Ha ennél a résznél akadnak problémái,akkor feltétlenül kérje szakember konzultációját!!!

Gyakorló feladatok:

8.1 Írjon programot, amely 13-tól 21-ig kiírja a számokat, majd az összegüket.

8.2 Írjon programot, amely kiírja szorosan egymás mellé a háromszámjegyű páros számokat!

8.3 Írjon programot, amely kiszámítja a négyszámjegyű, 3-mal osztható egész számok összegét!

8.4 Írjon programot, amely kiszámítja az első 13 egész szám szorzatát, vagyis a 13!-t (13 faktoriálist)

8.5 Írjon programot, amely 10 eredménytáblázat formájában kiszámítja, és táblázatonként feltételes várakozással képernyőre írja 1-től 10-ig a számok szorzatát (A szorzó táblákat.)

8.6 Írjon programot, amely kiírja az X*X-2, az 5*X-6 és a 9*X*X-7*X-4 függvények értéktáblázatát 0-tól 10-ig, 0.5-es lépéssel.

8.7 Írja meg a 8.6-os feladatot úgy, hogy a lépésköz 0.01 legyen, hogy a program ügyeljen a helyiértékre, s hogy alkalmas helyre beépített várakozás segítse a szükséges függvényértékek leolvasását.

8.8 A 4. program sorfolytonosan tölti fel a képernyőt csillagokkal. Írjon olyan programot, amely oszlopfolytonosan labdákkal tölti fel a képernyőt. Gondoskodjon arról is, hogy a folyamat lassabban hajtódjon végre.

8.9 Írjon programot, amely beolvassa egy osztály létszámát, majd a tanulók magasságát, s kiszámítja az átlagos testmagasságot.

8.10 Írjon programot, amely beolvassa 33 db őszibarack átmérőjét, majd kiszámítja az I. osztályú (65 mm feletti) barackok átlagos átmérőjét és darabszámát, valamint a százalékos részesedését!

8.11 Írjon programot, mely 86-tól 64-ig -0.5 lépésközzel - lehetőség szerint helyiértékhelyesen - kiíratja a 6*X-7 és a 9*X*X-8*X-2 függvények értéktáblázatát!

8.12 Írjon programot, amely üres képernyő második sorának elejére 33, harmadik sorának elejére 32 sit. a 22-ik sorig csillagot írat ki, majd gondoskodik arról, hogy zavartalanul szemlélhessük a kapott ábrát!

8.13 Írjon programot, amely hat, a képernyőn elhelyezett 7 * 7-es labdanégyzetet rajzol!

9. LECKE: Egydimenziós tömbváltozók

Mintaprogramok:

1. program:

```
10 REM SZAMOK NEGYZETOSSZEGE
20 PRINT "■SZAMOK NEGYZETOSSZEGE 1-18 KOZOTT"
30 PRINT
40 PRINT "SORSZAM    ERTEK      NEGYZET"
50 PRINT
60 DIM N(18)
70 FOR I=1 TO 18
80 N(I)=I*I
90 NEXT I
100 FOR I=1 TO 18
110 PRINT "   ";I;"■■.",I,N(I)
120 NEXT I
130 S=0: REM NEM FELTETLENUL KELL
140 FOR I=1 TO 18
150 S=S+N(I)
160 NEXT I
170 PRINT
180 PRINT "NEGYZETOSSZEG",S
190 GET V$: IF V$="" THEN 190
200 PRINT "■": END
```

2. program:

```
10 REM ATLAGOS CSOPORT-TESTMAGASSAG
20 PRINT "■ATLAGOS CSOPORT-TESTMAGASSAGOT SZAMITUNK"
30 PRINT "A CSOPORT LETSZAMA: 16 FO"
40 PRINT
50 PRINT "KERJUK A 16 ADATOT CM-BEN!"
60 PRINT
70 DIM M(16)
80 FOR I=1 TO 16
90 INPUT M(I)
100 NEXT I
110 S=0
120 FOR I=1 TO 16
130 S=S+M(I)
140 NEXT I
150 S=S/16
160 PRINT "■ATLAGOS TEST-MAGASSAG A CSOPORTBAN"
170 PRINT
180 PRINT "SORSZAM    MAGASSAG"
```

```
190 PRINT
200 FOR I=1 TO 16
210 PRINT "  ";I;"■.";M(I)
220 NEXT I
230 PRINT
240 PRINT "A T L A G";S
250 GET V$: IF V$="" THEN 250
260 PRINT "■": END
```

2/a. program:

```
10 REM ATLAGOS CSOPORT-TESTMAGASSAG
20 PRINT "■ATLAGOS CSOPORT-TESTMAGASSAGOT SZAMITUNK"
30 PRINT "A CSOPORT LETSZAMA: 16 FO"
40 PRINT
50 PRINT "KERJUK A 16 ADATOT CM-BEN!"
60 PRINT
70 DIM M(16)
80 FOR I=1 TO 16
90 PRINT I;"■. MAGASSAG";
100 INPUT M(I)
110 PRINT
120 NEXT I
130 S=0
140 FOR I=1 TO 16
150 S=S+M(I)
160 NEXT I
170 S=S/16
180 PRINT "■ATLAGOS TEST-MAGASSAG A CSOPORTBAN"
190 PRINT
200 PRINT "SORSZAM   MAGASSAG"
210 PRINT
220 FOR I=1 TO 16
230 PRINT "  ";I;"■.";M(I)
240 NEXT I
250 PRINT
260 PRINT "A T L A G";S
270 GET V$: IF V$="" THEN 270
280 PRINT "■": END
```

3. program:

```
10 REM NAPI ATLAGTELJESITMENY
20 PRINT "■NAPI ATLAGTELJESITMENY KISZAMITASA"
30 PRINT
40 INPUT "MUNKANAPOK SZAMA";N
```

```
50 DIM A(N),D(N)
60 PRINT "SORSZAM   ALMA(T)   DOLGOZO(FO)"
70 PRINT
80 FOR I=1 TO N
90 PRINT "  ";I;"■.";
100 INPUT A(I)
110 PRINT "■";,
120 INPUT D(I)
130 PRINT
140 NEXT I
150 PRINT "■MUNKANAP  DOLGOZO  ALMA   ATLAG"
160 PRINT
170 FOR I=1 TO N
180 PRINT "  ";I;"■.",D(I),"■";A(I),"■■■";A(I)/D(I)
190 GET T$: IF T$="" THEN 190
200 NEXT I
210 PRINT "■": END
```

4. program:

```
10 REM MINI TELEFONKONYV 2
20 PRINT "■OLVASSA BE 9 ISMEROSE NEVET"
30 PRINT ,"ES TELEFONSZAMAT!"
40 PRINT
50 DIM N$(9),T$(9)
60 FOR I=1 TO 9
70 PRINT I;"■. NEV,TELEFON:";
80 INPUT N$(I),T$(I)
90 NEXT I
100 PRINT
110 INPUT "HOGY HIVJAK ONT";N$
120 PRINT "■ON■";N$;" MINI TELEFONKONYVE"
130 PRINT " ---------------------------------------------- "
140 PRINT " NEV",,"TELEFONSZAM"
150 PRINT " ---",,"------------"
160 PRINT
170 FOR I=1 TO 9
180 PRINT " ";N$(I)
190 PRINT "■",,T$(I)
200 PRINT
210 NEXT I
220 GET T$: IF T$="" THEN 220
230 PRINT "■"
240 END
```

5. program:

```
10 REM MINOSEGI ELOSZLAS VIZSGALATA
20 PRINT "■MINOSEGI ELOSZLAS VIZSGALATA"
30 PRINT
40 INPUT "TERMESZTETT NOVENY";N$
50 PRINT
60 INPUT "TERULETEK SZAMA";K
70 PRINT
80 DIM NT$(K),T1(K),T2(K),T3(K)
90 PRINT "TERULET","1. OSZT.","2. OSZT.","3. OSZT."
100 PRINT
110 FOR I=1 TO K
120 INPUT N$(I),T1(I),T2(I),T3(I)
130 NEXT I
140 GET A$: IF A$="" THEN 140
150 PRINT "■SZAZALEKOS EREDMENYTABLAZAT"
160 PRINT ,"TERMESZTETT NOVENY:",N$
170 PRINT "TERULET       I. O.    II. O.   III. O."
180 PRINT
190 FOR I=1 TO K
200 S=T1(I)+T2(I)+T3(I)
210 S1=100*T1(I)/S
220 S2=100*T2(I)/S
230 S3=100*T3(I)/S
240 PRINT N$(I)
250 PRINT "■","■■";S1
260 PRINT "■",,"■■";S2
270 PRINT "■",,,"■■";S3
280 GET T$: IF T$="" THEN 280
290 NEXT I
300 PRINT
310 PRINT "VEGEZTUNK?"
320 GET V$: IF V$="" THEN 320
330 PRINT "■": END
```

Mintaprogramok magyarázata:

1. program:

10-es utasítás: Megjegyzés, amely az őt követő program céljára utal.
20-es utasítás: Az előző megjegyzést az üres képernyőre írja.
30-as utasítás: Üres sort hagy.
40-es utasítás: Az elkövetkezendő táblázat fölé magyarázó szöveget ír.
50-es utasítás: Üres sort hagy.
60-as utasítás: 18 (valójában 19) valós szám elhelyezésére (ábrázolására) foglal helyet N közös névvel.
70-es utasítás: Megszervezi, hogy a következő utasítást 18-szor (I=1,2,3,...,18 mellett) hajtsa végre.

80-as utasítás: Olyan értékadó utasítás, amely az I*I szorzat kiszámítása után megkeresi a 60-as utasításban definiált N adatmező I-ik elemét, s ebbe a tárolórekeszbe (változóba) beleírja I*I értékét.
90-es utasítás: Befejezi az I-szerinti ciklust.
100-as utasítás: Újabb I-szerinti ciklust szervez.
110-es utasítás: Két üres pozíció után kiírja I aktuális értékét, majd közvetlenül mögéje tesz egy pontot. (Ezt idézi elő az idézőjelben a pont előtt szereplő kurzort balra vezérlő jel). Ezután I újabb kiírása következik, majd az N adatterület (egydimenziós tömb) I-ik elemének kinyomtatására kerül sor.
120-as utasítás: Befejezi az I szerinti ciklust.
130-as utasítás: Elővigyázatosságból nulla értéket adunk S-nek.
140-es utasítás: A már többször alkalmazott ciklusszervező utasítás.
150-es utasítás: Az S változóba összegzi "lépésről-lépésre" az N adatterület elemeit.
160-as utasítás: Befejezi az I-szerinti ciklust.
170-es utasítás: Üres sort hagy.
180-as utasítás: Kiírja a "NEGYZETOSSZEG" után S aktuális értékét.
190-es utasítás: Feltételes várakozás.
200-as utasítás: Képernyőtörlés után befejezi a program végrehajtását.

2. program:

10-es utasítás: A program célját leíró megjegyzés.
20-as utasítás: Üres képernyőre kiírja ugyanezt.
30-as utasítás: Kiírja a tervezett csoportlétszámot.
40-es utasítás: Üres sort hagy.
50-es utasítás: Felkéri a felhasználót, hogy cm-ben adja meg adatait. (fontos a későbbi helyes végeredmény érdekében, hogy betartsuk a mértékegységekre vonatkozó kéréseket.)
60-as utasítás: Üres sort hagy.
70-es utasítás: 16 magasságadat tárolására definiálja (lefoglalja) az M azonosítójú adatterületet, ahol valós típusú értékeket tárolhatunk.
80-as utasítás: Megszervezi az I szerinti ciklus végrehajtását (I=1,2,3,...,16)
90-es utasítás: Az M adatterület soron következő I-ik változójába beolvassa a billentyűzetről a soron következő magasságadatot.
100-as utasítás: Befejezi az I szerinti ciklust.
110-es utasítás: Előkészíti (kinullázza) az S változót, hogy oda gyűjthessük a magasságadatok összegét.
120-as utasítás: Az összegző ciklus megszervezése.
130-as utasítás: Az összegző utasítás.
140-es utasítás: Ciklus vége.
150-es utasítás: Átlagképző utasítás
160-as utasítás: Üres képernyőre kiírja a leíró szöveget.
170-es utasítás: Üres sort hagy.
180-as utasítás: Az alapadatok fölé feliratot ír.
190-es utasítás: Üres sort hagy.
200-as utasítás: Megszervezi a 16 alapadat kiíratását vezérlő ciklust.
210-es utasítás: Az előző programban már megmutatott módon kiírja a soron levő adat sorszámát, majd az értékét.

220-as utasítás: Befejezi a ciklust.
230-as utasitás: Üres sort hagy.
240-es utasítás: Magyarázó szöveggel együtt kiíratja a csoportra jellemző átlagos magasságot.
250-es utasítás: Feltételes várakozás.
260-as utasítás: Képernyőtörlés, majd program vége.

2/a. program:

A program szinte teljes egészében megegyezik az előzővel. Pusztán a 90-es beolvasó utasítást helyettesítettük 3 utasítással, melyek tartalmilag pontosan ugyanazt teszik, mint a 2. program megfelelő része. A mostani 90-es közérthetően kéri a soron levő adatot. A mostani 100-as utasítás beolvassa a soron levő adatot. A mostani 110-es pedig az éppen aktuális sort fejezi be. A program további része betű szerint azonos a 2. programmal.

3. program:

10-es utasítás: Megjegyzés, amely a program célját írja ki.
20-as utasítás: Kiírja az üres képernyőre a program célját.
30-as utasitás: Üres sort hagy.
40-es utasítás: Beolvassa az N változóba a munkanapok számát.
50-es utasítás: Az előző utasításban nyert információ - a munkanapok számának ismeretében - a naponta leszedett alma mennyiségének tárolására lefoglalja az A tömböt, a naponta dolgozó létszám tárolására pedig a D tömböt. (A deklarációk zárójelében szerepelt N értékét már ismerjük!!)
60-as utasítás: A további beolvasást megkönnyítendő, kiírjuk a sorszám és a várt adatok nevét és mértékegységét fejlécként.
70-es utasítás: Üres sort hagy.
80-as utasítás: Megszervezi a beolvasást végző ciklust.
90-es utasítás: Előkészíti a beolvasást.
100-as utasítás: Pozícionáltan beolvassa az aktuális leszedett almamennyiséget.
110-es utasítás: A felfelé mutató kurzor-jel és a sornegyedváltást vezérlő vesszők segítségével pozíciónáljuk az aktuális dolgozólétszám beolvasását.
120-as utasítás: Ez végzi az előbb jelzett létszám beolvasását.
130-as utasítás: Üres sort hagy.
140-es utasítás: Befejezi a ciklust.
150-es utasítás: Üres képernyőre kiírja a szerepeltetni kívánt adat, ill. eredményoszlopok magyarázó fejlécét.
160-as utasítás: Üres sort hagy.
170-es utasítás: Megszervezi az eredménytáblázat kiíratását végrehajtó ciklust.
180-as utasítás: Kiíratja az aktuális sorszámot, almamennyiséget, dolgozólétszámot, átlagteljesítményt - a ",", a ";" és a balra mutató kurzorjel segítségével elősegítve a minél esztétikusabb eredménytáblázatot.
190-es utasítás: Feltételes várakozás.
200-as utasítás: Befejezi a ciklust.
210-es utasítás: Képernyőtörlés, majd program vége.

4. program:

10-es utasítás: Megjegyzésben utal a program céljára, s egyben a 2-es számmal arra, hogy ezt a feladatot a tömbváltozók és a ciklusok nélkül egyszer már hosszadalmasan megoldottuk.
20-40-es utasítások: A már említett programhoz hasonlóan bejelentkezik mostani programvariánsunk is.
50-es utasítás: Definiál két 9-elemű szöveg tárolására alkalmas tömböt: az N$-t a telefontulajdonos neve, a T$-t pedig a telefonszáma számára.
60-as utasítás: Megszervezi az adatpárokat beolvasó ciklust.
70-es utasítás: Párbeszédes formában előkészíti az adatpárok beolvasását.
80-as utasítás: Beolvassa vesszővel elválasztott karaktersorozat-párként a telefontulajdonos nevét és telefonszámát.
90-es utasítás: Befejezi a beolvasó ciklust.
100-as utasítás: Üres sort hagy.
110-es utasítás: Beolvassa a MINI TELEFONKONYV tulajdonosának nevét.
120-160-as utasítások: A már említett első programvariánshoz hasonlóan előkészíti a telefonkönyv kiíratását.
170-es utasítás: Megszervezi a kiíró ciklust.
180-as utasítás: Kiírja a soronlevő telefontulajdonos nevét.
190-es utasítás: Ugyanebbe a sorba beírja a telefonszámát is.
200-as utasítás: Üres sort hagy.
210-es utasítás: Befejezi a ciklust.
220-as utasítás: Feltételes várakozás.
230-as utasítás: Képernyőtörlés.
240-es utasítás: Program vége.

5. program:

10-es utasítás: Megjegyzés, mely informál a program céljáról.
20-as utasítás: Üres képernyőre kiírja ugyanezt.
30-as utasítás: Üres sort hagy.
40-es utasítás: Beolvassa az N$ karakter-típusú változóba a termesztett növény nevét.
50-es utasítás: Üres sort hagy.
60-as utasítás: Beolvassa a K valós típusú változóba a területek számát.
70-es utasítás: Üres sort hagy.
80-as utasítás: Négy feladatra szabott (K elemű) tömböt (NT$, T1, T2, T3) deklarál a terület nevének, az ott termett első-, másod- és harmadosztályú termés mennyiségének tárolására.
90-es utasítás: A beolvasás megkönnyítésére kiírja az adatok elvárt sorrendjét.
100-as utasítás: Üres sort hagy.
110-es utasítás: Megszervezi a K terület adatainak beolvasását végző ciklust.
120-as utasítás: Beolvassa az adatokat.
130-as utasítás: Befejezi a beolvasási ciklust.
140-es utasítás: Feltételes várakozás.
150-es utasítás: Üres képernyőre kiírja az eredménytáblázat első sorát.
160-as utasítás: Kiírja a termesztett növényt.
170-es utasítás: Kiírja a táblázatok oszlopainak feliratát.

180-as utasítás: Üres sort hagy.
190-es utasítás: Megszervezi az eredménytáblázatot kiszámító és kiíró ciklust.
200-as utasítás: Kiszámítja a soron levő terület össztermését.
210-230-as utasítások: Kiszámítják az egyes minőségek százalékos részesedését.
240-es utasítás: Kiírja a sor elejére a soron levő terület nevét.
250-es utasítás: A felfelé és jobbra mutató kurzorjel és a sornegyedváltó ","-t felhasználva kiíratjuk az elsőosztályú termés részesedését.
260-as utasítás: A felfelé és balra mutató kurzorjel és a sornegyedváltó ","-t felhasználva kiíratjuk a másodosztályú termés százalékos részesedését.
270-es utasítás: A felfelé és balra mutató kurzorjel és a sornegyedváltó ","-t felhasználva kiíratjuk a harmadosztályú termés százalékos részesedését.
280-as utasítás: Feltételes várakozás.
290-es utasítás: Befejezi a ciklust.
300-as utasítás: Üres sort hagy.
310-es utasítás: Megkérdezi, hogy végeztünk-e.
320-as utasítás: Feltételes várakozás.
330-as utasítás: Képernyőtörlés és program vége.

A leckében szereplő új fogalmak:

Egydimenziós tömbök

Azokat az adatcsoportokat, amelyek egyidejű elhelyezést és azonos feldolgozást igényelnek több változót tartalmazó adatterületen, az ún. egydimenziós tömbben tároljuk. Ez olyan fogalom, mint nyelvtani tanulmányaink során a közös főnév, mely mintegy közös nevet ad az illető fogalomnak, aztán egy jelzővel vagy más ismérvvel választjuk ki a konkrét elemet.Itt a közös főnév szerepét a tömb neve játssza, a kiválasztó eszköz pedig a név után zárójelben szerepeltetett konkrét vagy szimbólikus sorszám, amely megmondja, hogy hányadik elemmel kívánunk foglalkozni. Általában a program elején szoktuk megadni a felhasználni kívánt adatcsoportok azonosítóját, típusát és elemszámát, majd a program további részében már csak hivatkozunk az éppen felhasználni kívánt elemre.

Tömbök deklarációja:

Általános alakja:

sorszám DIM tömbváltozó neve(tervezett elemszám)

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét határozza meg.

DIM - BASIC alapszó, amely utal arra, hogy utána adatcsoportok számára foglalunk helyet a memóriában.

tömbváltozó neve - az adatcsoport közös neve, mely pontosan ugyanazoknak a szabályoknak kell, hogy eleget tegyen, mint az eddig megismert változók.

()- az utasítás paraméter-elválasztó része.

tervezett elemszám - nem negatív egész értéket felvevő aritmetikai kifejezés, amely meghatározza az adatcsoport tárolására szánt "rekeszek" számát.

1. megjegyzés: A BASIC sajátossága, hogy minden adatcsoportnak van 0-ik eleme. Ez egyes matematikai eljárásoknál nagyon előnyös, de az esetek többségében zavaró lehet használatának erőltetése.

2. megjegyzés: Programjainkban a területkijelölés után a tömbváltozónévvel és az utána zárójelben szerepeltetett konkrét vagy szimbólikus sorszámmal tudunk hivatkozni az adatcsoport bármely elemére. Szigorúan tilos az adatcsoport nem definiált elemét használni (pl. 10 DIM A(13) után 20 A(14)=3-at írni!!!) Ha ezt mégis megtennénk, akkor a végrehajtás során a 20. utasítás végrehajtása helyett a ?BAD SUBSCRIPT ERROR hibaüzenetet írja ki a C-64. (Lásd Úry kézikönyv II. kötetét!)

3. megjegyzés: A BASIC interpreter a nem definiált tömböket 10 eleműnek tekinti, de javasoljuk, hogy kezdetben csak olyan adatcsoportokat, tömböket használjunk, amelyeket definiáltunk, deklaráltunk.

Gyakorló feladatok:

9.1 Írjon programot, amely egy 13 elemű A tömbbe beolvas adatokat, s kiszámítja ezek összegét és átlagát!

9.2 Írjon programot, amely beolvassa az adatok számát, definiál egy ennyi elemű B tömböt, majd beolvassa az adatokat. Készítsen belőle egy esztétikus adattáblázatot is!

9.3 Írjon programot, amely beolvassa a család tagjainak számát, majd az egyes családtagok havi jövedelmét. Számítsuk ki, hogy az egyes családtagok a családi jövedelem hány százalékát szolgáltatják.

9.4 Írjon programot, amely beolvassa 11 parcella területét, a rajta megtermelt paprika mennyiségét. A program írja át egy harmadik tömbbe parcellánként a termésátlagokat, majd készítsen egy esztétikus eredménytáblázatot!

9.5 Egy könyvtárban 33 könyv van. Mindegyikről tudjuk az íróját, a címét és az oldalszámát. Írjon programot, amely beolvassa ezt a 33 adathármast, s utána kívánság szerint kilistázza a vastag (>=500 oldal),a vékony (<=100 oldal) és a közepes könyveket!

9.6 Írjon programot, amely beolvassa környezete (osztály, csoport, család, munkahely stb...) tagjainak nevét, születési helyét és idejét. Majd kívánságra kiírja az 1964-ben Budapesten születetteket.

9.7 Próbálja meg fejleszteni az előző programot (tárolja külön a család- és utónevet), s próbálja meg megszervezni, hogy a kiválasztás szempontjai változtathatók legyenek.

9.8 Írjunk programot, amely beolvassa könyvtárunk állományát (író neve, mű címe), majd külön táblázatba kiíratja az egyes szerzők könyvtárunkban meglevő műveit.

9.9 Írjunk programot, amely beolvassa ismerőseink nevét, címét és telefonszámát. A program második fele név alapján keresse ki a megadott személy címét és telefonszámát.

9.10 Anyagbeszerzőnk 13 tételt vásárol. Írjunk programot, amely beolvassa minden tételnél a vásárolt áru nevét, egységárát, a vásárolt mennyiséget. A program számítsa ki, hogy mennyit fizettünk az egyes tételekért és összesen. A program egy esztétikus eredménytáblázattal informáljon a számítások eredményéről.

9.11 Egy családban 5 gyerek jár iskolába. Pénteken a szülők ki akarják számítani az egyes gyerekek heti teljesítményét. Ezen a héten 19 jegyet kaptak. A gyerekek neve: Andrea, Imre, Magdolna, Melinda és Péter. Olvassuk be a 19 adatot adatpáronként (Ki mit kapott?), majd számítsuk ki az egyes gyerekek heti átlageredményét!

9.12 Fejlesszük tovább úgy a programot, hogy akkor se legyen probléma, ha valamelyik gyerek egész héten nem felelt.

9.13 Írjon programot, amely megszervezi egy tetszőlegesen hosszú adatsor beolvasását, javítását és kilistázását - esetlegesen nagyon egyszerű feldolgozását. Az egyes lépések, funkciók közül a már korábban megismert menű-módszerrel válasszunk!

10. LECKE: Gyakran használt adatcsoport-feldolgozó eljárások

Mintaprogramok:

1. program:

```
10 REM MAXIMUM KERESO PROGRAM
20 PRINT "■MAXIMUM KERESO PROGRAM"
30 PRINT
40 INPUT "ADATOK SZAMA";N
50 DIM X(N)
60 PRINT
70 PRINT "SORSZAM   ADAT"
80 PRINT
90 FOR I=1 TO N
100 PRINT "  ";I;"■.",
110 INPUT X(I)
120 NEXT I
130 XA=X(1)
140 FOR I=2 TO N
150 IF X(I)>XA THEN XA=X(I)
160 NEXT I
170 PRINT
180 PRINT "AZ ADATSOR MAXIMUMA=";XA
190 GET V$: IF V$="" THEN 190
200 PRINT "■"  END
```

2. program:

```
10 REM SZELSOERTEKET SZAMITO PROGRAM
20 PRINT "■MAXIMUMOT ES MINIMUMOT"
30 PRINT
40 PRINT "     SZAMITO PROGRAM"
50 PRINT
60 INPUT "     ADATOK SZAMA";N
70 PRINT
80 PRINT "SORSZAM   ADAT"
90 PRINT
100 DIM X(N)
110 FOR I=1 TO N
120 PRINT "  ";I;"■.",
130 INPUT X(I)
140 NEXT I
150 XA=X(1)
160 XI=X(1)
170 FOR I=1 TO N
180 IF X(I)>XA THEN XA=X(I)
190 IF X(I)<XI THEN XI=X(I)
```

```
200 NEXT I
210 PRINT
220 PRINT
230 PRINT "A LEGKISEBB  ERTEK=";XI
240 PRINT
250 PRINT "A LEGNAGYOBB ERTEK=";XA
260 GET V$: IF V$="" THEN 260
270 PRINT "□": END
```

3. program:

```
10 REM KI A LEGMAGASABB?
20 PRINT "□LEGMAGASABBAT KIVALASZTO PROGRAM"
30 PRINT
40 INPUT "CSOPORT LETSZAMA";N
50 PRINT
60 DIM N$(N),CM(N)
70 PRINT "SORSZAM   NEV","■■MAGASSAG<CM>"
80 PRINT
90 FOR I=1 TO N
100 PRINT "  ";I;"■.",
110 INPUT N$(I)
120 PRINT "□",,,
130 INPUT CM(I)
140 NEXT I
150 FOR I=1 TO N
160 IF CA<CM(I) THEN CA=CM(I): K=I
170 NEXT I
180 PRINT
190 PRINT
200 PRINT N$(K);" A LEGMAGASABB."
210 PRINT
220 PRINT "AKI";CA;"CM MAGAS."
230 GET V$: IF V$="" THEN 230
240 PRINT "□"
250 END
```

4. program:

```
10 REM SORBARENDEZES NAGYSAG SZERINT
20 PRINT "□SORBARENDEZES NAGYSAG SZERINT"
30 PRINT: PRINT
40 INPUT "  RENDEZENDO ADATOK SZAMA";N
50 PRINT
60 DIM W(N)
70 PRINT "SORSZAM   ADAT"
```

```
80 PRINT
90 FOR I=1 TO N
100 PRINT "  ";I;".";
110 INPUT W(I)
120 NEXT I
130 PRINT
140 PRINT "KEZDHETJUK A SORBAALLITAST?"
150 GET T$: IF T$="" THEN 150
160 PRINT "■■■■■■■■■■■■■■■■■■■■■■■■■TURELEM! DOLGOZOM!"
170 FOR I=1 TO N-1
180 WA=W(I): A=I
190 FOR J=I+1 TO N
200 IF W(J)<WA THEN WA=W(J): A=J
210 NEXT J
220 W(A)=W(I)
230 W(I)=WA
240 NEXT I
250 PRINT "■A RENDEZETT ADATOK NOVEKVO SORRENDBEN"
260 PRINT
270 PRINT
280 PRINT "SORSZAM     RENDEZETT ADATOK"
290 PRINT
300 FOR I=1 TO N
310 PRINT I,W(I)
320 GET T$: IF T$="" THEN 320
330 NEXT I
340 GET V$: IF V$="" THEN 340
350 PRINT "■"
360 END
```

4/a. program:

```
10 REM SORBARENDEZES ABC SZERINT
20 PRINT "■SORBARENDEZES ABC SZERINT"
30 PRINT: PRINT
40 INPUT "  RENDEZENDO SZAVAK SZAMA";N
50 PRINT
60 DIM W$(N)
70 PRINT "SORSZAM    KARAKTERSOROZAT"
80 PRINT
90 FOR I=1 TO N
100 PRINT "  ";I;".";
110 INPUT W$(I)
120 NEXT I
130 PRINT
140 PRINT "KEZDHETJUK A SORBAALLITAST?"
150 GET T$: IF T$="" THEN 150
160 PRINT "■■■■■■■■■■■■■■■■■■■■■■■■■TURELEM! DOLGOZOM!"
170 FOR I=1 TO N-1
```

```
180 WA$=W$(I): WA=I
190 FOR J=I+1 TO N
200 IF W$(J)<WA$ THEN WA$=W$(J): WA=J
210 NEXT J
220 W$(WA)=W$(I)
230 W$(I)=WA$
240 NEXT I
250 PRINT "■A RENDEZETT SZAVAK ABC SORRENDBEN"
260 PRINT
270 PRINT
280 PRINT "SORSZAM    KARAKTERSOROZAT"
290 PRINT
300 FOR I=1 TO N
310 PRINT I,W$(I)
320 GET T$: IF T$="" THEN 320
330 NEXT I
340 GET V$: IF V$="" THEN 340
350 PRINT "■"
360 END
```

5. program:

```
10 REM *** HAZI TELEFON KONYVET LETREHOZO
20 REM *** ES KARBANTARTO PROGRAM
30 PRINT "■■■■■■■■■HAZI TELEFONKONYVET LETREHOZO
40 PRINT "■■■■■■     ------------------------------------"
50 PRINT
60 PRINT "         ES KARBANTARTO PROGRAM"
70 PRINT "         ----------------------"
80 PRINT: PRINT
90 INPUT "    TULAJDONOS NEVE";T$
100 PRINT: PRINT
110 INPUT "    TELEFONTULAJDONOSOK SZAMA";N
120 DIM T1$(N),T2$(N),T3$(N)
130 PRINT "■■■■■■■■■■■■■■■BARMELY BILLENTYU"
140 PRINT "■■■■■■■MEGNYOMASAVAL TOVABBLEPHET!"
150 GET Z$: IF Z$="" THEN 150
160 FOR I=1 TO N
170 T1$(I)="*** ***"
180 T2$(I)="*** ***"
190 T3$(I)="*** ***"
200 NEXT I
210 PRINT "■■■■■■■■■■TEVEKENYSEG-LISTA"
220 PRINT "         --------------------"
230 PRINT
240 PRINT
250 PRINT "■  1 - TELEFONKONYV FELTOLTESE"
260 PRINT "■  2 - TELEFONKONYV KILISTAZASA"
270 PRINT "■  3 - TELEFONKONYV JAVITASA"
280 PRINT "■  4 - KERESES A NEV SZERINT"
290 PRINT "■  5 - KERESES A CIM SZERINT"
300 PRINT "■  6 - KERESES A TELEFON SZERINT"
```

```
310 PRINT "■  7 - ADATRENDEZES"
320 PRINT "■  8 - PROGRAM VEGE"
330 PRINT
340 INPUT "■       VALASZA ";V%
350 IF V%<1 OR V%>8 THEN 340
360 IF V%=1 THEN 440
370 IF V%=2 THEN 640
380 IF V%=3 THEN 770
390 IF V%=4 THEN 1000
400 IF V%=5 THEN 1140
410 IF V%=6 THEN 1280
420 IF V%=7 THEN 1410
430 IF V%=8 THEN 1600
440 PRINT "■TELEFONKONYV FELTOLTESE"
450 PRINT "■",,,"■■■■■■■";T$
460 PRINT
470 PRINT "(VESSZOT NE HASZNALJON!!!)"
480 PRINT
490 PRINT "SORSZAM   NEV"
500 PRINT ,"CIM(TELEPULES UTCA H'SZ ISZ)"
510 PRINT ,"TELEFONSZAM"
520 PRINT
530 FOR I=1 TO N
540 PRINT "  ";I;"■.",
550 INPUT T1$(I)
560 PRINT ,
570 INPUT T2$(I)
580 PRINT ,
590 INPUT T3$(I)
600 PRINT
610 NEXT I
620 FOR V=0 TO 3000: NEXT V
630 GO TO 210
640 PRINT "■■■";T$;" TELEFONKONYVE"
650 PRINT " ---------------------------------------- "
660 PRINT
670 FOR I=1 TO N
680 PRINT: PRINT
690 PRINT " ELOFIZETO NEVE: ";T1$(I)
700 PRINT
710 PRINT " CIME: ";T2$(I)
720 PRINT
730 PRINT " TELEFONSZAMA: ";T3$(I)
740 GET V$: IF V$="" THEN 740
750 NEXT I
760 GO TO 210
770 PRINT "■TELEFONKONYV JAVITASA"
780 PRINT
790 PRINT "TELEFONTULAJDONOSOK SZAMA:";N
800 PRINT
```

```
810 INPUT "MELYIKET AKARJA JAVITANI";K%
820 IF K%<0 OR K%>N THEN 810
830 PRINT
840 PRINT "REGI ADATOK:"
850 PRINT
860 PRINT "ELOFIZETO NEVE: ";T1$(K%)
870 PRINT "CIME: ";T2$(K%)
880 PRINT "TELEFONSZAMA: ";T3$(K%)
890 PRINT
900 PRINT "UJ ADATOK:"
910 PRINT
920 PRINT "ELOFIZETO NEVE:    ";T1$(K%)
930 INPUT "■■■■■■■■■■■■■■■■■■■■";T1$(K%)
940 PRINT "CIME:    ";T2$(K%)
950 INPUT "■■■■■■■■";T2$(K%)
960 PRINT "TELEFONSZAMA:    ";T3$(K%)
970 INPUT "■■■■■■■■■■■■■■■■■■";T3$(K%)
980 FOR I=1 TO 1000: NEXT I
990 GO TO 210
1000 PRINT "■KERESES■AZ ELOFIZETO NEVE■SZERINT"
1010 PRINT: L=0
1020 INPUT "KERESETT ELOFIZETO";T1$
1030 FOR I=1 TO N
1040 IF T1$(I)<>T1$ THEN 1100
1050 PRINT "■■■ELOFIZETO NEVE: ";T1$(L)
1060 PRINT "CIME: ";T2$(L)
1070 PRINT "■■■ELOFIZETO NEVE: ";T1$(L)
1080 GET V$: IF V$="" THEN 1080
1090 L=I
1100 NEXT I
1110 IF L=0 THEN PRINT "■■■■■NEM TALALTUK AZ ELOFIZETOT!"
1120 GET V$: IF V$="" THEN 1120
1130 GO TO 210
1140 PRINT "■KERESES■AZ ELOFIZETO CIME■SZERINT"
1150 PRINT: L=0
1160 INPUT "KERESETT CIM";T2$
1170 FOR I=1 TO N
1180 IF T2$(L)<>T2$ THEN 1240
1190 PRINT "■■■ELOFIZETO NEVE: ";T1$(L)
1200 PRINT "CIME: ";T2$(L)
1210 PRINT "TELEFONSZAMA: ";T3$(L)
1220 GET V$: IF V$="" THEN 1220
1230 L=I
1240 NEXT I
1250 IF L=0 THEN PRINT "■■■   NEM TALALTUK MEG A CIMET!"
1260 GET V$: IF V$="" THEN 1100
1270 GO TO 210
1280 PRINT "■KERESES■AZ ELOFIZETO TELEFONSZAMA■SZERINT"
1290 PRINT: L=0
1300 PRINT "KERESETT TELEFONSZAM";T3$
1310 FOR I=1 TO N
1320 IF T3$(I)<>T3$ THEN 1370
1330 PRINT "■■■ELOFIZETO NEVE: ";T1$(L)
1340 PRINT "CIME: ";T2$(L)
```

```
1350 PRINT "TELEFONSZAMA: ";T3$(L)
1360 I=I
1370 NEXT I
1380 IF L=0 THEN PRINT "NEM TALALTUK A TELEFONSZAMOT!"
1390 GET V$: IF V$="" THEN 1390
1400 GO TO 210
1410 PRINT "ELOFIZETOK NEVSORBA RENDEZESE"
1420 PRINT "TURELEM!!!   DOLGOZOM!!!"
1430 IF N=1 THEN 210
1440 FOR I=1 TO N-1
1450 T1$=T1$(I): T2$=T2$(I): T3$=T3$(I): K=I
1460 FOR J=I+1 TO N
1470 IF T1$<T1$(J) THEN 1510
1480 IF T1$=T1$(J) AND T2$<T2$(J) THEN 1510
1490 IF T1$=T1$(J) AND T2$=T2$(J) AND T3$<T3$(J) THEN 1510
1500 T1$=T1$(J): T2$=T2$(J): T3$=T3$(J): K=J
1510 NEXT J
1520 T1$(K)=T1$(I)
1530 T2$(K)=T2$(I)
1540 T3$(K)=T3$(I)
1550 T1$(I)=T1$
1560 T2$(I)=T2$
1570 T3$(I)=T3$
1580 NEXT I
1590 GO TO 210
1600 PRINT "PROGRAM VEGE"
1610 FOR I=0 TO 10000
1620 NEXT I
1630 PRINT ""
1640 END
```

Mintaprogramok magyarázata:

1. program:

10-es utasítás: A program célját leíró megjegyzés.
20-as utasítás: A program célját írja az üres képernyőre.
30-as utasítás: Üres sort hagy.
40-es utasítás: Beolvassa az adatok szamát.
50-es utasítás: Az adatok aktuális számának ismeretében lefoglalja az X tömböt, melyben N(+1) valós szám tárolására lesz lehetőségünk.
60-as utasítás: Üres sort hagy.
70-es utasítás: Az adatok beolvasását segítő fejlécet írja ki.
80-as utasítás: Üres sort hagy.
90-es utasítás: Megszervezi az adatok beolvasását végző ciklust.
100-as utasítás: A soron levő adat beolvasását megkönnyítendő kiírja a sorszámot (ezt segíti a balra mutató kurzorjel), majd a következő sornegyedre áll.
110-es utasítás: Beolvassa a soron levő adatot az X tömb megfelelő rekeszébe.
120-as utasítás: Befejezi a beolvasást végző ciklust.

130-as utasítás: A maximumkeresés első lépéseként félreteszi XA-ba az első tömbelemet, hiszen egy elem közül maga az elem a legnagyobb.
140-es utasítás: A teljes indukció gondolatát követve megszervezi a 2., 3., n. elem vizsgálatát végző ciklust.
150-es utasítás: XA tartalmazza az eddigi maximumot. Ezen akkor és csak akkor kell változtatnunk, ha nála nagyobbat találtunk. Pontosan ezt teszi utasításunk.
160-as utasítás: Befejezi a maximumkereső ciklust.
170-es utasítás: Üres sort hagy.
180-as utasítás: Kiírja a maximumkeresés végeredményét.
190-es utasítás: Feltételes várakozás.
200-as utasítás: Képernyőtörlés és program vége.

2. program:

10-es utasítás: A program célját ismertető megjegyzés.
20-50-es utasítások: A program célját üres képernyőre író programsorok.
60-as utasítás: Beolvassa a feldolgozni kívánt adatok számát.
70-90-es utasítás: Üres sorok között a beolvasást elősegítő felíratot ír ki.
100-as utasítás: Lefoglalja az X valós tömböt N(+1) számérték tárolására.
110-140-es utasítások: Az előző program megfelelő utasításaihoz hasonlóan gondoskodnak a feldolgozandó adatsor beolvasásáról.
150-es utasítás: A maximum keresésének előkészítéseként X első elemét választjuk maximumnak.
160-as utasítás: A minimum keresésének előkészítéseként X első elemét választjuk minimumnak.
170-es utasítás: Megszervezi a további (2., 3., N.) elemek vizsgálatát végző ciklust.
180-as utasítás: Megvizsgálja, hogy a soron levő elem nagyobb-e, mint az eddigi maximum. Ha igen, akkor XA ezt az értéket veszi fel.
190-es utasítás: A 180-ashoz hasonló elvet alkalmazza a minimum keresésénél.
200-as utasítás: Befejezi a maximumot, minimumot kereső ciklust.
210-250-es utasítások: Üres sorok és megfelelő kisérőszöveg mellett kiírja az adatsor legkisebb és legnagyobb értékét.
260-as utasítás: Feltételes várakozás.
270-es utasítás: Képernyőtörlés és program vége.

3. program:

10-es utasítás: A program célját leíró megjegyzés.
20-30-as utasítások: A program célját üres képernyőre író utasítások.
40-es utasítás: A csoport létszámát beolvasó utasítás.
50-es utasítás: Üres sort hagy.
60-as utasítás: A csoportlétszám ismeretében két tömböt foglal le: egyet a csoport tagjai névsorának (N$), egyet pedig az egyes tagok magasságadatának tárolására (CM).
70-es utasítás: A beolvasás megkönnyítése érdekében kiírt fejléc, melyben kétszer szerepel a kurzort balra mozgató jel.
80-as utasítás: Üres sort hagy.
90-es utasítás: Megszervezi az adatpárok (név, magasság) beolvasását végző ciklust.

100-as utasítás: Előkészíti a már megszokott módon az I-ik név beolvasását.
110-es utasítás: Beolvassa az I-ik nevet.
120-as utasítás: A felfelé és balra mutató kurzormozgató jel segítségével előkészíti a soron levő tag magasságadatának beolvasását.
130-as utasítás: Beolvassa a soron levő magasságadatot.
140-es utasítás: Befejezi a beolvasó ciklust.

Megjegyzés: Mivel magasságot keresünk, ezért feltételezhetjük, hogy már az első lépés eredményes lesz a keresés során. Így engedhetjük meg a CA=0 induló magasságot, továbbá a K=0 induló sorszámot.

150-es utasítás: Megszervezi a legmagasabbat (legnagyobbat) kereső ciklust.
160-as utasítás: Megnézi, hogy a soron levő magassága CM(I) nagyobb-e mint az eddigi legmagasabb. Ha igen, akkor az értéket CA-ba, az indexet pedig K-ba teszi félre.
170-es utasítás: Befejezi a kereső ciklust.
180-220-as utasítások: Párbeszédes formában közlik a felhasználóval a program végeredményét.
230-as utasítás: Feltételes várakozás.
240-es utasítás: Képernyőtörlés.
250-es utasítás: Program vége.

4. program:

10-es utasítás: Megjegyzés formájában közli a program célját.
20-30-as utasítások: Üres képernyőre kiírja a program célját.
40-es utasítás: Beolvassa a rendezendő adatok számát.
50-es utasítás: Üres sort hagy.
60-as utasítás: Lefoglalja a rendezendő adatok számára a megfelelő elemű W tömböt.
70-80-as utasítások: A két sor kiírásával megkönnyíti az adatok beolvasását.
90-120-as utasítások: A lassan már megszokott módon beolvassa a rendezendő adatokat.
130-150-es utasítások: Kiírással és feltételes várakozással módot ad a rendezendő adatok ellenőrzésére.
160-as utasítás: A rendezés időtartamára kiírja az üzenetet az üres képernyő közepére.
170-240-es utasítások: Ez a két egymásba skatulyázott ciklus nagyság szerint növekvő sorrendbe állítja a beolvasott adatokat. (Az eljárást majd a lecke fogalmainál ismertetjük részletesebben.)
250-290-es utasítások: Kiírják az eredménytáblázat fejlécét.
300-330-as utasítások: A feltételes várakozást is felhasználva kiírják a rendezett adatokat.
340-es utasítás: Feltételes várakozás, amely lehetővé teszi az eredmény tanulmányozását.
350-es utasítás: Képernyő törlése.
360-as utasítás: Program vége.

4/a. program:

A 4. programhoz hasonlóan adatokat rendez ABC szerint sorrendbe.

5. program:

Eddigi ismereteinket összegzi az 5. mintaprogram. Talán a javítandó adatoknál a régi adat "alákészítése" jelenthet némi újdonságot, de ez is pusztán néhány ügyesen megválasztott kurzormozgató jel kérdése.

A leckében előforduló új fogalmak:

Teljes indukció

Olyan bizonyítási, keresési, számítási eljárás, amely egy adott elemszámra (pl. 1-re) megadja a feladat megoldását, továbbá azt, hogy ha eggyel több elemünk van, akkor hogyan kapjuk meg ennek a feladatnak a megoldását az előző felhasználásával.

Maximumkeresés:

Legyen N elemünk, melyek közül keressük a legnagyobb értéket. Ha N=1, akkor nyilvánvaló, hogy ez a legnagyobb. Legyen ez mostani indukciós eljárásunk induló megoldása. Mint ilyet, tegyük félre egy könnyen elérhető helyre. Ezután vegyük sorba a hátralevő elemeket. Ha jobbat (nagyobbat) találunk a félretettnél, akkor tegyük azt félre. Ha az eljárást követjük, akkor a félretett helyen mindig az addigi maximális érték lesz. (Ha arra is kíváncsiak vagyunk, hogy melyik elem lesz maximális, akkor a pillanatnyi maximális elem sorszámát is tegyük félre.)

Minimumkeresés:

A feladat és megoldása szinte teljes egészében megegyezik az előzővel. Csak a vizsgálat során nem nagyobbat, hanem kisebbet kell keresnünk.

Nagyság szerinti sorbarendezés:

(Erre a feladatra az eljárások sokaságát dolgozta ki a matematika és a számítástechnika. Mi itt egy klasszikus módszert alkalmazunk, amely a minimumkeresések és sorrendcserék sorozatával jut el a megoldáshoz.)

Ha N elemünk van, akkor N-1 lépésben jutunk el a feladat megoldásához. (Itt is igaz, hogy N=1 esetén megvan a megoldás, hiszen 1 elem mindig a helyén van.)

Az első végrehajtással az első, a második eljárással a második, az utolsóelőtti eljárással pedig az utolsóelőtti elem kerül a helyére. (Ha az utolsóelőtti elem is a helyén van, akkor az utolsó sem lehet máshol, csak a helyén.)

Kiindulva abból, hogy a K-ik lépés előtt az első K-1 elem a helyén van, megkeressük a K., K+1.,...N. elem minimumát és ezt felcseréljük a K. elemmel. (A minimumkeresést már előbb leírtuk, az elemcsere pedig egyszerű értékadó utasítások sorozata, amint azt a 4. és 4/a. programok is illusztrálják számok, illetve karaktersorozatok (szövegkonstansok) esetében.)

Gyakorló feladatok:

10.1 Írjon programot, amely beolvassa egy 7 tagú család tagjainak nevét és testsúlyát, s kikeresi a család legkönnyebb tagját.

10.2 Írjon programot, amely beolvassa 6 autó típusát, az eddigi ráfordított költséget, s az általuk megtett kilométereket. Válassza ki a program azt a típust, amellyel eddigi adataink szerint a leggazdaságosabb közlekedni.

10.3 Írjon programot, amely megadja egy gazdaság különböző növények termesztésére felhasznált területeinek nagyságát (ha), az azokon megtermelt termések mennyiségét (t), az illető termények felvásárlási egységárát. Keressük meg, hogy melyik növény hozta ha-nként a legtöbb jövedelmet.

10.4 Ha van kedvünk, fejlesszük tovább előző programunkat úgy, hogy ismerve az egyes növényekre jellemző munkabéreket, anyagköltségeket, továbbá a termésekre jutó szállítási, értékesítési költségeket, melyik növény termesztése lesz fajlagosan a legjövedelmezőbb.

10.5 Ha ismerjük 5 család havi jövedelmét és a családtagok számát, akkor rendezzük a családokat az egy főre jutó jövedelem szerint növekvő sorrendbe.

10.6 Fejlesszük tovább előző programunkat úgy, hogy a fennálló szociális kategóriák szerint minősítse a család anyagi helyzetét.

10.7 A 10.5-ös feladat adatai alapján számítsuk ki az egy főre jutó jövedelemátlagot, majd keressük meg, írassuk ki és számláljuk meg az ennél kevesebb jövedelmű családokat.

10.8 Írjunk programot, amely egy üzem napi létszámadatait olvassa be egy hónapon keresztül. Keresse meg, hogy mi volt a maximális létszám,hány napon keresztül dolgozott ezzel a létszámmal, végül hány napot dolgozott a maximumhoz képest legalább 80%-os létszámmal.

10.9 Írjon programot, amely egytizedes pontossággal megbecsüli, hogy az X*X-2 függvény a -4 és 4 közötti szakasz mekkora részén lesz pozitív.

10.10 Számlálja meg, hogy az 1/(N*N) kifejezés (N=1,...,1000 mellett) hányszor esik 0.1 és 0.00001 közé.

10.11 Írjon programot, amely beolvas 19 pozitív számot, kiszámítja a számtani és mértani átlagot.

10.12 Írjon programot, amely az előző program felhasználásával leszámlálja, hogy 19 adat közül hány szám nagyobb a számtani, és hány szám nagyobb a mértani átlagnál.

10.13 Írjon programot, amely beolvassa 17 nap termelési eredményét s az azt létrehozó dolgozók létszámát. Számítsa ki az egy főre jutó napi átlagteljesítményt, majd számolja meg, hogy hány napon teljesítettek ennyit, vagy ennél többet fejenként.

11. LECKE: A POKE utasítás és alkalmazása

Mintaprogramok:

1. program:

```
10 POKE 53280,8
20 POKE 53281,7
30 PRINT "J"
40 X=0: Y=0
50 DX=1: DY=1
60 POKE 1024+X+40*Y,81
70 FOR B=0 TO 50: NEXT B
80 POKE 1024+X+40*Y,32
90 X=X+DX
100 Y=Y+DY
110 IF X=0 OR X=38 THEN DX=-DX
120 IF Y=0 OR Y=24 THEN DY=-DY
130 GO TO 60
```

2. program:

```
10 POKE 53280,8
20 POKE 53281,7
30 PRINT "J"
40 FOR I=1024 TO 1063:
50 POKE I,127
60 POKE I+960,127
70 NEXT I
80 FOR J=1024 TO 1984 STEP 40
90 POKE J,127
100 POKE J+39,127
110 NEXT J
120 X=1: Y=1
130 DX=1: DY=1
140 POKE 1024+X+40*Y,81
150 FOR B=0 TO 50: NEXT B
160 POKE 1024+X+40*Y,32
170 X=X+DX
180 Y=Y+DY
190 IF X=1 OR X=38 THEN DX=-DX
200 IF Y=1 OR Y=23 THEN DY=-DY
210 GO TO 140
```

3. program:

```
10 POKE 53280,8
20 POKE 53281,7
30 PRINT "■"
40 FOR I=0 TO 24
50 FOR L=0 TO 39
60 POKE 1024+L+40*I,64+L
70 NEXT L
80 NEXT I
90 FOR J=55296 TO 56295 STEP 40
100 FOR L=0 TO 39
110 POKE J+L,K
120 NEXT L
130 K=K+1
140 IF K=16 THEN K=0
150 NEXT J
160 GO TO 90
```

4. program:

```
10 PRINT "■"
20 FOR I=0 TO 15
30 FOR J=0 TO 15
40 POKE 53280,I
50 POKE 53281,J
60 PRINT "■"
70 FOR K=1 TO 11
80 PRINT "■■ALMA A FA ALATT, NYARI PIROS ALMA"
90 NEXT K
100 SZ=0
120 FOR L=55296 TO 56295
130 POKE L,SZ
140 SZ=SZ+1
150 IF SZ=16 THEN SZ=0
160 NEXT L
170 FOR E=0 TO 100: NEXT E
180 NEXT J
190 NEXT I
200 GET T$: IF T$="" THEN 200
210 IF T$<>"V" THEN 10
```

4/a. program:

```
10 PRINT "■"
20 FOR I=0 TO 15
30 FOR J=0 TO 15
40 POKE 53280,I
50 POKE 53281,J
60 PRINT "■"
70 FOR K=1 TO 11
80 PRINT "■■ALMA A FA ALATT, NYARI PIROS ALMA"
90 NEXT K
100 SZ=0
120 FOR L=55296 TO 56295
130 POKE L,SZ
140 SZ=SZ+1
150 IF SZ=15 THEN SZ=0
160 NEXT L
170 FOR E=0 TO 100: NEXT E
180 NEXT J
190 NEXT I
200 GET T$: IF T$="" THEN 200
210 IF T$<>"V" THEN 10
```

4/b. program:

```
 10 PRINT "■"
 20 FOR I=0 TO 15
 30 FOR J=0 TO 15
 40 POKE 53280,I
 50 POKE 53281,J
 60 PRINT "■"
 70 FOR K=1 TO 11
 80 PRINT "■■ALMA A FA ALATT, NYARI PIROS ALMA"
 90 NEXT K
100 SZ=0
110 FOR L=0 TO 24
120 FOR M=0 TO 39
130 POKE 55296+M+40*L,SZ
140 NEXT M
150 SZ=SZ+1
160 IF SZ=16 THEN SZ=0
170 NEXT L
180 FOR E=0 TO 100: NEXT E
190 NEXT J
200 NEXT I
210 GET T$: IF T$="" THEN 210
220 IF T$<>"V" THEN 10
```

5. program:

```
10 REM ************************************
20 REM * KERTESZETI NOVENYRENDSZERTAN *
30 REM ************************************
40 POKE 53280,8: POKE 53281,13
50 PRINT "■■■■           K E R T E S Z E T I"
60 PRINT "■■■■■■■■■■   N O V E N Y R E N D S Z E R T A N"
70 PRINT "■■■■■■■■■■■■■■■■          T A N P R O G R A M"
80 PRINT "■■■■■■■■■■■■■■■■■■■■■■■■                1 9 8 6"
90 FOR I=1024 TO 1063
100 POKE I,127
110 NEXT I
120 FOR I=1064 TO 1944 STEP 40
130 POKE I,127: POKE I+39,127
140 NEXT I
150 FOR I=1984 TO 2023
160 POKE I,127
170 NEXT I
180 POKE 1106,88
190 POKE 1141,88
200 POKE 1906,88
210 POKE 1941,88
220 FOR T=1194 TO 1212 STEP 2
230 POKE T,88
240 NEXT T
250 FOR T=1347 TO 1379 STEP 2
260 POKE T,88
270 NEXT T
280 FOR T=1514 TO 1532 STEP 2
290 POKE T,88
300 NEXT T
310 POKE 1678,88
320 POKE 1681,88
330 POKE 1685,88
340 POKE 1688,88
350 GET T$: IF T$="" THEN 350
360 PRINT "■"
```

Mintaprogramok magyarázata:

1. program:

A program a közismert labdapattogtatást végzi.
10-es utasítás: A POKE utasítás, amely az első paraméterrel megjelölt byteba a második paraméter aktuális értékét teszi, narancssárgára festi a képernyő fel nem használt keretét.
20-as utasítás: Sárgára festi a képernyőnek azt a területét, ahol dolgozur'
30-as utasítás: Törli a képernyőt.

40-es utasítás: A labda aktuális helyzetét megadó X és Y változók felveszik a 0 értéket.
50-es utasítás: A labda pillanatnyi mozgásirányát megadó DX és DY változók felveszik az induló 1 értékeket.
60-as utasítás: A keret és a képernyő színéhez hasonlóan a képernyő aktuális tartalmát is módosíthatjuk a POKE-utasítással. A képernyő aktuális 1000 pozícióját az 1024-2023. byte-okon tartalmazza a C-64. Ha X jelöli a bal oldali képernyőszéltől mért távolságot és Y pedig a felsőszéltől mért távolságot, akkor az 1024+X+40*Y konstrukció éppen az X,Y pozíciónak megfelelő byte-sorszámot adja meg. A labda kódja a 81. Ha ezt megadjuk, akkor a megfelelő képernyőpozícióban egy tintaszínű, besatírozott korong jelenik meg. Természetes, hogy balra legfeljebb 39-et, lefelé legfeljebb 24-et léphetünk. (Sajnos a C-64 nem védi ki a paraméter-túllépést, ezért saját érdekünkben vigyázzunk arra, hogy a POKE-utasítás mindkét paramétere értékhelyes legyen!!!)
70-es utasítás: Egy 50-szer lefutó ún. üres ciklus, amelynek az a szerepe, hogy lelassítsa a program futását.
80-as utasítás: A 60-as utasításban kirajzolt labda helyére szóközt tesz, azaz töröl.
90-es utasítás: A labda X-koordinátáját megnöveli DX-szel.
100-as utasítás: A labda Y-koordinátáját megnöveli DY-nal.
110-es utasítás: Ha a vízszintes irányban elérte valamelyik engedélyezett végértéket, akkor ellenkezőjére változtatja a vízszintes irányú sebességet.
120-as utasítás: Ha a függőleges irányban eléri valamelyik engedélyezett végértéket, akkor a függőleges irányú sebességét ellenkezőjére változtatja.
130-as utasítás: Az eljárást a 60-as utasításnál folytatja.

2. program:

Ez a program egyfajta módosítása az előzőnek. A képernyőt sakktábla-elemekkel keretezzük be (40-110-es utasítások), majd az így lecsökkentett területen mozog az előző rendszerhez hasonlóan a labda.

3. program:

A program első 3 utasítása megegyezik az előző kettőével. Az ezt követő 5 utasítás két egymásba skatulyázott ciklusban a képernyő 25 sorába (ezt jelenti a 0-24 ciklusszervezés) kiír 40 grafikus jelet a POKE felhasználásával (a 64-103 kódúakat).
90-150-es utasítások: Eddig a POKE-utasítást keret és képernyő színezésére, valamint különböző karakterek megjelenítésére alkalmaztuk. Most ebben a ciklusban azt mutatjuk meg, hogy a már kiírt írásjel színét írásjelenként is megváltoztathatjuk, ha a memória 55296-56295. byte-jait a POKE-utasítás segítségével megváltoztatjuk. Itt is élhettünk volna az előzőekben már megszokott konstrukcióval. (Felhívjuk az Olvasó figyelmét arra, hogy az érték a fontos és nem a konstrukció. Az csak bizonyos esetekben szemléletesebb és megbízhatóbb.)
160-as utasítás: Ismételteti az előző színezési eljárást.

4. program:

Mind a három programvariáns színezési esettanulmány.

5. program:

Egy korábban elkészült oktatási segédeszköz bevezető sorainak idézésével mutatjuk be eddigi képernyőkezelő utasításaink lehetőségeit.

A leckében előforduló új fogalmak:

A POKE-utasítás:

Általános alakja:

sorszám POKE cím,érték

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét szabja meg. (Az utasítást nagyon gyakran használjuk programon kívül parancs üzemmódban is.)

POKE - az utasítást megadó BASIC alapszó, amely az első paraméterrel megadott helyre (címre) beírja a második paraméterrel megadott értéket.

cím - a memória egy címe, ahová az értéket írjuk. A cím formálisan 0 és 65535 közé kell, hogy essen. Persze a programozónak kell ügyelnie arra,hogy csak olyan memóriacím értékét módosítsa, amely céljainak megfelel.

érték - mivel az értéket egy byte-os egész számként tárolja, ezért 0 és 255 közöttinek kell lennie. Erre a paraméterre is igaz, hogy a programozó "iszsza meg a levét", ha valami szabálytalan értéket akar adni a megcímzett memóriarekesznek.

Keret- és képernyőszín módosítása:

Ha a POKE-utasítás első paramétere 53280, akkor ún. keretszínt módosítunk. Ha a POKE-utasítás első paramétere 53281, akkor ún. képernyő- vagy háttérszínt módosítunk.

Mindkét esetben a második paraméter 0 és 15 közötti egész értéket vesz fel,s ennek megfelelően a végrehajtás pillanatától kezdve a megfelelő színezésben lesz látható a képernyő megfelelő része. Bár az értéktáblázat szinte valamennyi kézikönyvben (így Úry Lászlóéban is) benne van, mégis közöljük.

Számérték	Szín	Számérték	Szín
0	fekete	1	fehér
2	vörös	3	ciánkék (türkisz)
4	bíbor (lila)	5	zöld
6	kék	7	sárga
8	narancs	9	barna
10	rózsaszín	11	szürkel
12	szürke2	13	világoszöld
14	világoskék	15	szürke3

Képernyőre karakter kiírása:

Az első program leírása során már elmondottuk, hogy a képernyő egyes pozícióinak karaktertartalmát a C-64 a memória 1024-2023-as című területein tárolja. Így ezeket a megfelelő című byte-ok tartalmával is meg tudjuk változtatni. Ennek előnye, hogy a PRINT utasítástól eltérően a sor vagy sornegyed további és megelőző részeit változatlanul hagyja. Alkalmazhatunk direkt címzést vagy az 1. programban már ismertetett konstrukciót. Az egyes kiírni kívánt írásjelek kódját megtalálhatjuk az egyes kézikönyvekben (pl. Ury László kézikönyv II. kötetében), vagy a később ismertetésre kerülő programozástechnikai függvények segítségével a C-64 is kiírja majd nekünk.

Képernyőre írt karakterek színezése:

A POKE-utasítással megváltoztattuk a képernyő s a keret színét. Joggal merül fel az igény, hogy a kiírt karakterek színét is meg tudjuk változtatni. Erre több lehetőségünk van. Itt csak azokat a lehetőségeket soroljuk fel, amelyekkel már most rendelkezünk:

- a **CTRL** és a Commodore-billentyű, valamint a számjegyeket tartalmazó billentyűk (1,2,3,4,5,6,7,8) együttes megnyomása megváltoztatja a villogó kurzor és az ezután leirt karakterek színét, a kiválasztott billentyűkombinációnak megfelelő színre,
- a memóriaterület 55296-56295-ös byte-sorozata tartalmazza az egyes memóriapozícióba kiírt írásjelek színét. (Ez az ún. színmemória.) Ha itt egy-egy byte tartalmát megváltoztatjuk, akkor ezáltal módosíthatjuk a már kiírt karakter színét, akár pozíciónként is, ahogy ezt a megfelelő mintaprogramok illusztrálják.

Gyakorló feladatok:

11.1 Írjon programot, amely üres képernyőre kirajzolja a nemzeti zászlót.
11.2 Írjon programot, amely üres képernyőre egy stilizált sakktáblát varázsol.
11.3 Fejlessze tovább az előző programot úgy, hogy egy labdát írjon minden olyan mezőre, amelyen induló állásban bábú áll.
11.4 Írjon programot, amely kívánság szerint változtatja a keret és a háttér színét.
11.5 Írjon programot, amely a környező országok stilizált zászlóit rajzolja a képernyőre.
11.6 Írjon programot, amely színes, hangulatos üdvözlőkártyát rajzol a képernyőre.
11.7 Írjon programot, mely kedvenc versét, a vers hangulatát visszaadó díszítésekkel együtt a képernyőre írja.
11.8 Írja az üres képernyő közepére a labda (81) segítségével a C-64 szöveget, majd váltsa annak színét a szivárvány minden színében.
11.9 Írjon programot, amely oszlopfolytonosan (oszloponként) változtatja az üres képernyő színét.
11.10 Írjon programot, amely sakktáblaszerűen kitölti a képernyőt 88-as kódú ábrákkal.
11.11 Írjon programot, amely a 83-as kódú jelből hozzá hasonló ábrát hoz létre a képernyőn.
11.12 Rajzoljon stilizált szemet a képernyőre az eddig megismert grafikai jelek be-POKE-olásával.
11.13 Próbálja meg úgy továbbfejleszteni az előző programot, hogy szeme "könnyezzen"!

12. LECKE: Függvények

Mintaprogramok:

1. program:

```
10 REM FUGGVENY TABLAZAT
20 PRINT "■";TAB(4)"X";TAB(14)"1/X";TAB(24)"1/X↑2"
30 PRINT
40 FOR X=5 TO 7 STEP 0.1
50 Y1=1/X
60 Y2=1/X↑2
70 E1=INT(1000*Y1+0.5)/1000
80 E2=INT(1000*Y2+0.5)/1000
90 PRINT TAB(2)X;TAB(13)E1;TAB(24)E2
100 NEXT X
110 GET V$: IF V$="" THEN 110
120 PRINT "■": END
```

2. program:

```
10 PRINT "■"
20 FOR I=0 TO 22
30 PRINT SPC(I);"ZEBEGENY  1985"
40 FOR L=0 TO 50: NEXT L
50 NEXT I
60 FOR L=0 TO 99: NEXT L
70 FOR K=22 TO 0 STEP -1
80 PRINT TAB(K)"SZARVAS  1986"
90 FOR L=0 TO 50: NEXT L
100 NEXT K
110 FOR L=0 TO 99: NEXT L
120 GO TO 20
```

3. program:

```
10 REM ROHANO PATAK
20 X=INT(19*RND(1)+10)
30 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒";TAB(X)"♠";TAB(30)"▒▒▒▒▒▒▒▒▒▒▒▒▒"
40 FOR S=0 TO 99: NEXT S
50 GO TO 20
```

4. program:

```
10 REM KANYARGO PATAK
20 X=INT(8*RND(1)+2)
30 PRINT "                                                    "
40 P=12+5*SIN(L)
50 PRINT "■";TAB(P)"◠◠◠◠◠◠◠◠"
60 PRINT "■";TAB(X)"♠"
70 PRINT "■";TAB(39-X)"♠"
80 L=L+0.45
90 FOR K=0 TO 199: NEXT K
100 GO TO 20
```

5. program:

```
10 REM EGYSZERU FUGGVENYABRA
20 FOR X=0 TO 6283 STEP 0.1
30 H=INT(20+19*SIN(X))
40 FOR K=1 TO H
50 PRINT "*";
60 NEXT K
70 PRINT
80 NEXT X
```

A mintaprogramok magyarázata:

1. program:

10-es utasítás: Megjegyzés, mely a program célját közli.
20-as utasítás: Üres képernyőre az első sor 4-es, 14-es és 24-es jelű oszlopától kezdve kiírja a sorszámok után idézőjelben szereplő karaktersorozatokat.
30-as utasítás: Üresen hagyja a második sort.
40-es utasítás: Megszervezi a függvényértékek kiszámítását és kiíratását végző ciklust.
50-60-as utasítások: Kiszámítják az aktuális függvényértékeket.
70-80-as utasítások: Három tizedesjegyre kerekítik a kiszámított függvényértékeket.
90-es utasítás: A TAB-utasítás segítségével kiírják az X-értéket és a két helyettesítési értéket a képernyőre.
100-as utasítás: Befejezi az X-változó vezérelte ciklust.
110-es utasítás: Feltételes várakozás.
120-as utasítás: Képernyőtörlés és program vége.

2. program:

10-es utasítás: Törli a képernyőt.
20-as utasítás: Megszervezi az SPC-függvényt bemutató ciklust.
30-as utasítás: Bemutatja, hogy mit jelent, ha a sor elejétől I pozíciót kihagyunk.
40-es utasítás: 50-ig számlál, s ezzel késlelteti a program túl gyors végrehajtását.
50-es utasítás: Befejezi az SPC-függvényt bemutató ciklust.
60-as utasítás: További számlálással lassítja a program túl gyors végrehajtását.
70-es utasítás: Visszafelé számlálva (lépve) szervezi meg a TAB-függvényt bemutató ciklust.
80-as utasítás: Illusztrálja a TAB-függvény működését.
90-es utasítás: Késlelteti a program végrehajtását.
100-as utasítás: Befejezi a TAB-függvényt bemutató ciklust.
110-es utasítás: Számlálással késlelteti a program gyors végrehajtását.
120-as utasítás: Visszatér a 20-as utasításra.

3. program:

10-es utasítás: Ismerteti a program célját.
20-as utasítás: Az RND- és INT-függvények segítségével "gondol" egy egész számot 10 és 28 között.
30-as utasítás: A sor két szélére a Commodore-billentyű és a +-billentyű segítségével létrehozott grafikus jeleket ír, közé pedig a TAB segítségével "siftelt" A-kat.
40-es utasítás: Lassítja a program végrehajtását.
50-es utasítás: Visszatér a 20-as utasításra.

4. program:

10-es utasítás: Közli a program célját.
20-as utasítás: Az INT- és RND-függvények segítségével "gondol" egy egész számot 2 és 9 között.
30-as utasítás: A szóköz és az előző programban már szerepelt pepita minta segítségével előfesti a sort.
40-es utasítás: A SIN-függvény alkalmazásaként P felvesz egy 7 és 17 közötti valós számot.
50-es utasítás: A felfelé mutató kurzor, majd a tabuláló P segítségével az előző pepita alapra a P-ik pozíciótól kezdve "ráfesti" 7-szer a "siftelt" UI-kombinációt.
60-70-es utasítások: A "csinosítást" befejezendő sor eleje és vége táján egy "siftelt" A-t és S-et helyeznek el.
80-as utasítás: A SIN-függvény argumentumát (változóját, L-t) megnöveli 0.45-tel.
90-es utasítás: Programkésleltető számlálás.
100-as utasítás: Visszalépés a 20-as utasításra.

5. program:

10-es utasítás: Közli a program célját.
20-as utasítás: Megszervezi a függvényábrázolást illusztráló ciklust.
30-as utasítás: Az INT- és a SIN-függvények segítségével létrehoz egy 1 és 38 közötti egész számot.
40-es utasítás: Megszervezi a nyomtatás H-szori végrehajtását.
50-es utasítás: Kiír egy *-ot, majd a ";" hatására "ottmarad".
60-as utasítás: Befejezi a ciklus végrehajtását.
70-es utasítás: Befejezi az éppen aktuális sor írását.
80-as utasítás: Befejezi a függvényábrázolást vezérlő fő ciklust.

A leckében előforduló új fogalmak:

A TAB-függvény:

Általános alakja:

sorszám PRINT TAB(pozíció)kiírandó információ

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét szabja meg.

PRINT - a képernyőre írást "kiváltó" BASIC alapszó.

TAB - BASIC alapszó, mely soron levő képernyősor zárójelben szerepeltetett pozíciójánál kezdi az azután felsorolt információ képernyőre írását.

pozíció - olyan egész értéket felvevő aritmetikai kifejezés, amely meghatározza az első kiírni kívánt karakter helyét az aktuális képernyősorban.

kiírandó információ - tartalmát a neve megadja.

Megjegyzés: Ha az adott képernyősorban már elértük vagy meghaladtuk a kért pozíciót, akkor a következő sor megadott pozícióján kapjuk meg a kiírandó információ első írásjelét. A pozíció értelemszerűen 0 és 39 közé eső egész szám.

Az INT-függvény

Olyan értékadást végző művelet, amely elhagyja az argumentumában (zárójelében) szereplő aritmetikai kifejezés tört részét. (Mielőtt negatív számértékekre alkalmaznánk, próbáljuk ki az ottani viselkedését is! Ez kissé eltér az imént mondottaktól.)

Az INT-függvény alkalmazása kerekítésre:

Tegyük fel, hogy egy számítás eredményeként 5 tizedesjegy pontosságú nem negatív eredményeink vannak (pl. 1.23456). További számításainkhoz viszont csak a három tizedesjegyre kerekített érték szükséges. Mi a teendő? Eddigi iskolai tanulmányaink alapján példánk kerekített eredménye 1.235. Hogyan tudjuk a kerekítési szabályt megtanítani a C-64-nek. A gondolatmenet:

Művelet	Példa	Érték
Szorzás	1000*1.23456	1234.56
Hozzáadás	1234.56+0.5	1235.06
Egészképzés	INT(1235.06)	1235
Osztás	1235/1000	1.235

A fenti táblázatból ki-ki könnyedén megkonstruálhatja az 1. program 70-80-as utasításaiban szereplő kerekítési utasításokat.

Az SPC-függvény:

Általános alakja:

sorszám PRINT SPC(üres helyek száma)kiírandó információ

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét szabja meg.

PRINT - a képernyőreírást "kiváltó" alapszó.

SPC - BASIC alapszó, amely az utána szereplő zárójelben megadott számú üres helyet hagy íratlanul az utoljára kiírt információ és a most soron levő között.

üres helyek száma - egész értéket felvevő aritmetikai kifejezés.

kiírandó információ - tartalma nyilvánvaló.

Grafikus jelek a képernyőn:

Ha a PRINT-utasítás karaktersorozataiban a **SHIFT** és a Commodore billentyűk segítségével grafikus jeleket írunk, azok a megfelelő képernyőpozíción megjelennek. (Lásd a grafikus jelek táblázatát pl. Úry László kézikönyvének II. fejezetében.)

Az RND-függvény:

Olyan függvény, amely a [0,1) intervallumban gondol egy számot, melyet nagyon sok céllal felhasználhatunk. A véletlen szám statisztikai jellemzői: egyenletes eloszlást követ, s az egymás után "gondolt" számok kielégítik a matematikai statisztika függetlenségi kritériumait.

A SIN-függvény:

A matematikában megismert sin x -függvény. Egyetlen specialitása, hogy a szög értékét radiánban, ívhosszban kell megadni! (A további függvények is hasonlóak. Részletes definíciójukat megtalálhatjuk Úry László kézikönyvének 4.1 alfejezetében.)

Gyakorló feladatok

12.1 Írjon programot, amely a tizedes pontossággal kinyomtatja a sin x , cos x (COS(X)) függvénytáblázatát a [0,2.2] intervallumban, ha a szögívhossz tizedenként nő.

12.2 Írjon programot, amely kétféle módon írja ki a tg x táblázatát az [1,2] intervallumban 5 századonként:
- tg x = sin x/cos x,
- tg x = TAN(X)

12.3 Írjunk programot, amely kinyomtatja az első 23 pozitív egész szám természetes alapú logaritmusát, 6 tizedes pontossággal (ln x=LOG(X)).

12.4 Írjon programot, amely az előző függvény inverzét (az e^X-et (EXP(X)) íratja ki 2 tizedes pontossággal a 0.1, 1.9 értékhatárok között tizedenként.

12.5 Írjon programot, amely a képernyőre véletlenszerűen 19 labdát tesz. A labda kódja 81.

12.6 Írjon programot, amely "gondol" 13 véletlen egész számot 13 és 21 között.

12.7 Írjon programot, amely "kihúzza" a heti lottószámokat. A program továbbfejlesztésében gondoljon a számismétlések elkerülésére!

12.8 Írjon programot, amely minden sorba véletlenszerűen 3 esőcseppet hullat el.

12.9 Írjon programot, amely hóesést imitál!

12.10 Írjon programot, amely helyiértékre helyesen kiírja a 3 tizedesjegyre lekerekített mérési eredményeket!

12.11 Írjon programot, amely a POKE-utasítás segítségével elkészíti a SIN-függvény relatív (arányos) függőleges ábráját egy perióduson keresztül.

12.12 Írjon programot, amely elkészíti X*X*X függvény relatív ábráját a -2,2 értékek között.

12.13 A POKE-utasítás felhasználásával készítse el 40 tetszőleges adat nagysági viszonyait megadó relatív ábrát.

13.LECKE: Karaktersorozat-kezelő függvények

Mintaprogramok:

1. program:

```
10 REM SZOVEG HOSSZ STATISZTIKA
20 INPUT "SZOVEG SZAVAINAK SZAMA";N
30 PRINT
40 PRINT "SORSZAM   KARAKTERSOROZAT      HOSSZ"
50 PRINT
60 FOR I=1 TO N
70 PRINT TAB(2)I,
80 INPUT A$
90 PRINT "",,LEN(A$)
100 S=S+LEN(A$)
110 NEXT I
120 PRINT
130 PRINT "A SZOVEG SZAVAINAK ATLAGOS HOSSZA";
140 A=S/N
150 PRINT INT(100*A+0.5)/100
160 GET V$: IF V$="" THEN 160
170 PRINT ""
180 END
```

2. program:

```
10 REM ARANYOS ROVATKITOLTO
20 INPUT "NEVE";N$
30 INPUT "SZEMELYI SZAMA";SZ$
40 INPUT "SZULETESI HELYE";H$
50 INPUT "ANYJA NEVE";A$
60 INPUT "ISKOLAI VEGZETTSEGE";I$
70 INPUT "MUNKAHELYE";M$
80 INPUT "FOGLALKOZASA";F$
90 INPUT "FIZETESE";FT
100 PRINT "A DOLGOZO ADATLAPJA"
110 PRINT ""
120 PRINT TAB(2)"NEVE:";
130 K=LEN(N$)
140 PRINT TAB(7+(30-K)/2)N$
150 PRINT
160 NE$=LEFT$(SZ$,1)
170 EV$="19"+MID$(SZ$,2,2)
180 HO$=MID$(SZ$,4,2)
190 NA$=MID$(SZ$,6,2)
200 SO$=RIGHT$(SZ$,4)
```

```
210 PRINT TAB(2)"SZEMELYI SZAMA:
    ";NE$;" ";EV$;" ";HO$;" ";NA$;" ";SO$
220 PRINT
230 K=LEN(H$)
240 PRINT TAB(2)"SZULETESI HELYE:";TAB(19+(18-K)/2)H$
250 K=LEN(A$)
260 PRINT
270 PRINT TAB(2)"ANYJA NEVE:";TAB(15+(22-K)/2)A$
280 K=LEN(I$)
290 PRINT
300 PRINT TAB(2)"ISKOLAI VEGZETTSEGE:";TAB(22+(15-K)/2)I$
310 K=LEN(M$)
320 PRINT
330 PRINT TAB(2)"MUNKAHELYE:";TAB(13+(24-K)/2)M$
340 K=LEN(F$)
350 PRINT
360 PRINT TAB(2)"FOGLALKOZASA:";TAB(15+(22-K)/2)F$
370 P=10
380 FOR I=1 TO 10
390 IF FT>=10↑I THEN P=P-1
400 NEXT I
410 PRINT
420 PRINT TAB(2)"FIZETESE:";TAB(10+P)FT
430 GET V$: IF V$="" THEN 430
440 PRINT "□"
450 END
```

3. program:

```
10 PRINT CHR$(147)TAB(7)"---------------------------"
20 PRINT CHR$(18)TAB(7)"A CHR$-FUGGVENY BEMUTATOJA"CHR$(146)
30 PRINT TAB(7)"---------------------------"
40 FOR I=1 TO 4
50 PRINT
60 NEXT I
70 FOR I=1 TO 3
80 PRINT CHR$(28)TAB(5)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
90 NEXT I
100 PRINT
110 FOR I=1 TO 3
120 PRINT CHR$(5)TAB(5)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
130 NEXT I
140 PRINT
150 FOR I=1 TO 3
160 PRINT CHR$(30)TAB(5)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
170 NEXT I
180 GET V$: IF V$="" THEN 180
190 PRINT CHR$(147)
200 END
```

4. program:

```
10 REM TINTASZIN BEMUTATO
20 PRINT CHR$(147)
30 PRINT TAB(10)"TINTASZIN BEMUTATO"
40 PRINT TAB(10)"00000000000000000"
50 PRINT: PRINT: PRINT
60 DATA 5,FEHER
70 DATA 28,PIROS
80 DATA 30,ZOLD
90 DATA 31,KEK
100 DATA 129,NARANCS
110 DATA 144,FEKETE
120 DATA 149,BARNA
130 DATA 150,ROZSASZIN
140 DATA 151,SZURKE1
150 DATA 152,SZURKE2
160 DATA 153,VILAGOSZOLD
170 DATA 154,VILAGOSKEK
180 DATA 155,SZURKE3
190 DATA 156,BIBOR
200 DATA 158,SARGA
210 DATA 159,CIAN
220 FOR I=1 TO 16
230 READ SZ,S$
240 PRINT CHR$(SZ)"A TINTA SZINE: ";S$;TAB(28)"KODJA=";SZ
250 NEXT I
260 GET V$: IF V$="" THEN 260
270 PRINT CHR$(147)
280 END
```

5. program:

```
10 REM FENYJATEK
20 PRINT CHR$(147)
30 FOR I=1 TO 7
40 PRINT
50 NEXT I
60 PRINT SPC(13)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
70 PRINT SPC(13)"▒                 ▒"
80 PRINT SPC(13)"▒                 ▒"
90 PRINT SPC(13)"▒                 ▒"
100 PRINT SPC(13)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
110 PRINT CHR$(145)CHR$(145)CHR$(145)SPC(15);
115 PRINT CHR$(28)"F"CHR$(5)"E"CHR$(30)"N";
```

```
120 PRINT CHR$(28)"Y"CHR$(5)"J"CHR$(30)"A";
130 PRINT CHR$(28)"T"CHR$(5)"E"CHR$(30)"K"
140 FOR C=1 TO 1000
150 P=1024+1000*RND(1)
160 P=INT(P)
170 IF PEEK(P)<>32 THEN 150
180 POKE P,42
190 POKE 54274+P,15*RND(1)
200 NEXT C
```

Mintaprogramok magyarázata:

1. program:

10-es utasítás: Megyjegyzés, mely közli a program célját.
20-as utasítás: Beolvassa a megvizsgálni kívánt szöveg hosszát (szavainak számát).
30-as utasítás: Üres sort hagy.
40-es utasítás: A beolvasás megkönnyítésére fejlécet ír.
50-es utasítás: Üres sort hagy.
60-as utasítás: Megszervezi a program beolvasó és feldolgozó ciklusát.
70-es utasítás: Kiírja a soron levő szó sorszámát, majd sornegyedet vált.
80-as utasítás: Beolvassa a soron levő szót A$-ba.
90-es utasítás: A LEN-függvény segítségével kiszámítja az A$-ban tárolt szó (karaktersorozat) hosszát, majd azt az előző sor utolsó negyedének elejére írja.
100-as utasítás: Az A$ karaktersorozatának hosszát hozzáadja S-hez.
110-es utasítás: Befejezi a ciklust.
120-as utasítás: Üres sort hagy.
130-as utasítás: Kiírja a képernyőre a végeredményt bevezető szöveget.
140-es utasítás: Kiszámítja az átlagos szóhosszat.
150-es utasítás: Kiírja a képernyőre a két tizedesre kerekített végeredményt.
160-as utasítás: Feltételes várakozás.
170-es utasítás: Képernyő törlése.
180-as utasítás: Program vége.

2. program:

10-es utasítás: Megjegyzés, mely közli a program célját.
20-as utasítás: Képernyőtörlés után beolvassa a dolgozó nevét.
30-as utasítás: Beolvassa a dolgozó személyi számát.
40-es utasítás: Beolvassa a dolgozó születési helyét.
50-es utasítás: Beolvassa a dolgozó anyjának nevét.
60-as utasítás: Beolvassa a dolgozó iskolai végzettségét.
70-es utasítás: Beolvassa a dolgozó munkahelyét.
80-as utasítás. Beolvassa a dolgozó foglalkozását.
90-es utasítás: Beolvassa a dolgozó fizetését.
100-as utasítás: Megkezdi az eredménytáblázat kiírását.

110-es utasítás: Három üres sort hagy.
120-as utasítás: Kiírja a nevet bevezető szöveget.
130-as utasítás: Kiszámítja a dolgozó nevének hosszát.
140-es utasítás: A TAB-függvény és a benne szerepeltetett kifejezés segítségével úgy írja ki a dolgozó nevét, hogy előtte és utána ugyanannyi hely maradjon üresen.
150-es utasítás: Üres sort hagy.
160-as utasítás: A LEFT$-függvény segítségével leválasztja a személyi számból az első karaktert.
170-es utasítás: A MID$-függvény segítségével a személyi számból kiolvassuk a születési év utolsó két számjegyét. Elé írjuk a születési év első két számjegyét.
180-as utasítás: Az előző utasításhoz hasonlóan "kiolvassa" a dolgozó születési hónapját.
190-es utasítás: Az előző utasításhoz hasonlóan "kiolvassa" a dolgozó születési napját.
200-as utasítás: A RIGHT$-függvény segítségével a személyi szám végéről "leszedi" a dolgozó "sorszámát".
210-215-ös utasítás: Bevezető szöveg után kiírja a dolgozó tagolt, bővített személyi számát.
220-as utasítás: Üres sort hagy.
230-360-as utasítások: A dolgozó nevéhez hasonlóan bevezető szöveg után arányosan pozícionáltan kiírják a dolgozó adatait.
370-es utasítás: Beállítja P induló értékét.
380-400-as utasítás: A dolgozó fizetésének nagyságrendjétől függően csökkenti P értékét.
410-es utasítás: Üres sort hagy.
420-as utasítás: Bevezető szöveg után pozícionáltan írja ki a dolgozó fizetését.
430-as utasítás: Feltételes várakozás.
440-es utasítás: Képernyőtörlés
450-es utasítás: Program vége.

3. program:

10-es utasítás: A CHR$-függvény egy korábban már alkalmazott formájával üres képernyő első sorába a TAB-függvénnyel megadott pozíciójától kezdve "-"-jelekből álló vonalat ír ki.
20-as utasítás: A CHR$-függvény egy további alkalmazásaként ún. inverz módon írja ki a feliratot (tintaszín alapon háttérszínnel), majd visszaváltott az alapszínre.
30-as utasítás: A 10-es utasításhoz hasonlóan aláhúzza az imént kiírt szöveget.
40-60-as utasítások: Egy négyszer végrehajtásra kerülő ciklusban négy sort kihagy.
70-90-es utasítások: A CHR$-függvény újabb alkalmazásaként háromsoros piros csíkot rajzol a képernyőre a Commodore és +-billentyű-kombináció segítségével.
100-es utasítás: Üres sort hagy.
110-130-as utasítások: A 70-90 sorozathoz hasonlóan fehér csíkot rajzol a képernyőre.

140-es utasítás: Üres sort hagy.
150-170-es utasítások: a 70-90 sorozathoz hasonlóan zöld csíkot rajzol a képernyőre.
180-as utasítás: Feltételes várakozás.
190-es utasítás: Letörli a képernyőt.
200-as utasítás: Program vége.

4. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Képernyő törlése.
30-as utasítás: Kiírja a program címét.
40-es utasítás: Siftelt W-kel aláhúzza a címet.
50-es utasítás: 3 üres sort ír.
60-210-es utasítások: Programbeépített adatként tartalmazza a lehetséges tintaszínek CHR$-kódját, valamint nevét.
220-as utasítás: Megszervezi a tintaszínek bemutató ciklusát.
230-as utasítás: A 60-210 DATA-területről beolvassa az aktuális szín CHR$-kódját, valamint nevét.
240-es utasítás: Az aktuális tintaszínnel kinyomtatja a tintaszín nevét és kódját.
250-es utasítás: Befejezi a ciklust.
260-as utasítás: Feltételes várakozás.
270-es utasítás: Képernyőtörlés.
280-as utasítás: Program vége.

5. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Törli a képernyőt.
30-50-es utasítások: 7 sort kihagyó ciklus.
60-100-as utasítások:Az SPC-függvény és a Commodore-+-kombináció segítségével téglalapot rajzol a képernyőre.
110-115-ös utasítás: A CHR$-függvény segítségével 3 sort lép felfelé, majd az SPC segítségével 15-öt jobbra, hová piros-fehér-zöld betűkkel kiírja előbb a "FEN"-betűhármast.
120-as utasítás: Az előzőhöz csatolja az "YJA"-t.
130-as utasítás: Befejezi a kiírást a "TEK"-kel.
140-es utasítás: Megszervezi a ciklust, amely 1000 csillagot rajzol ki a képernyőre.
150-es utasítás: "Gondol" egy számot 1024 és 2023 között.
160-as utasítás: Képezi P egész részét.
170-es utasítás: Megnézi a P-ik memóriarekesz tartalmát a PEEK-függvény segítségével. Ha ez különbözik szóköztől, akkor újat gondol.
180-as utasítás: Kiír egy csillagot.
190-es utasítás: Kiszínezi a csillagot.
200-as utasítás: Befejezi a ciklust.

A leckében előforduló új fogalmak:

A LEN-függvény:

Általános alakja:

LEN(karaktersorozat vagy ilyen típusú változó)

Működési elve:

Megszámolja, hogy az aktuális karaktersorozat hány karakterből áll.

A LEFT$-függvény:

Általános alakja:

LEFT$(karaktersorozat vagy ilyen típusú változó, egész típusú konstans(változó vagy aritmetikai kifejezés)).

Működési elve:

A függvény az első paraméterével megadott karaktersorozat elejéről a második paraméterrel megadott számú karaktert választ ki.

A MID$-függvény:

Általános alakja:

MID$(karaktersorozat vagy ilyen típusú változó, egész konstans vagy ilyen típusú változó, egész konstans vagy ilyen típusú változó)

Működési elve:

Kiválasztja az első paraméterrel megadott karaktersorozatból a második paraméterrel megadott sorszámú helytől a harmadik paraméterrel megadott számú karaktert.

A RIGHT$-függvény:

Általános alakja:

RIGHT$(karaktersorozat vagy ilyen típusú változó, egész konstans vagy ilyen típusú változó)

Működési elve:

A függvény az első paraméterrel megadott karaktersorozat végéről a második paraméterrel megadott számú karaktert "kiolvassa".

A CHR$-függvény:

Általános alakja:

CHR$(egész konstans vagy ilyen típusú változó)

Működési elve:

Az argumentum értékének megfelelő karakterrel, vezérlőjellel dolgozik tovább a program. (Néhány példát a mintaprogramok adnak. A bővebb listákat a kézikönyvek tartalmazzák. Pl. Úry László kézikönyvének II. kötetében.)

A DATA-utasítás:

Általános alakja:

sorszám DATA különböző típusú konstansok

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli szerepet szabja meg.

DATA - BASIC-alapszó, mely utal arra, hogy az utána vesszőkkel elválasztva felsorolt karaktersorozatokat a program alkalmas helyen konstans adatsorozatként fel fogja használni.

különböző típusú konstansok - tartalma nyilvánvaló.

A READ-utasítás:

Általános alakja:

sorszám READ változók felsorolása

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét határozza meg.

READ - BASIC-alapszó, amely azt jelenti, hogy az utána felsorolt változók a programban elhelyezett DATA-utasításoktól várják (olvassák le) az adatokat.

A PEEK-függvény:

Általános alakja:

PEEK(memóriacím)

Működési elve:

Kiolvassa a zárójelében szereplő memóriacím aktuális tartalmát.

Gyakorló feladatok:

13.1 Írjon programot, amely a képernyő első sorába kiírja az ABC betűit, valamint a számjegyeket (0-9), s a PEEK segítségével "nyomozza" ki az egyes karakterek ún. képernyőkódját.

13.2 Írjon programot, mely egy karaktersorozatként beolvasott (legalább két részből álló) nevet alkotóelemeire bont.

13.3 Írjon programot, amely megszámolja, hogy egy beolvasott vers (pl. Petőfi: Egy gondolat bánt engemet) hány e betűt tartalmaz.

13.4 Írjon programot, amely egy 13 tagú névsorból (a teljes nevek alkotnak egy sorozatot) kikeresi az Andreákat.

13.5 Írjon programot, amely világoskékre festi a képernyő aktuális tartalmát.

13.6 Írjon programot, amely "kis szövegszerkesztő"-ként beolvas egy szöveget (szósorozatot), majd úgy írja ki a képernyőre, hogy minden szó között egy hely legyen, s a sorvég szót ne törjön meg.

13.7 Fejlessze tovább úgy szövegszerkesztőjét, hogy minden új mondat (egyszerűség kedvéért nagybetűvel kezdődő szó) új sor 4. pozícióján kezdődjön.

13.8 Írjon programot, amely a beolvasott rövid szavakat (legfeljebb 3 betű) zöld színnel, a közepes hosszúakat sárgával és a nagyon hosszúakat (legalább 13 betű) pirossal írja ki a képernyőre.

13.9 Próbáljon meg szótagoló programot szerkeszteni.

13.10 Írjon programot, amely egy adott szövegből kigyűjti a benne előforduló magánhangzók gyakoriságát s azt, hogy az összes magánhangzó hány százaléka "a"-betű.

13.11 Írjon programot, amely egy adott szöveg valamennyi magánhangzóját "e"-betűre cseréli.

13.12 Írjon programot, amely egy szöveg valamennyi mássalhangzóját "*"-gal helyettesíti.

13.13 Véletlenszerűen helyezzen el 13 csillagot a képernyőn, s indítsa el úgy a pattogó labdát, hogy az a csillagokról is helyesen (a fizikai törvényeknek megfelelően) pattanjon vissza.

14. LECKE: Adatállományok kazettán és lemezen

Mintaprogramok:

1. program:

```
10 REM ADATKIVITEL KAZETTARA
20 OPEN 13,1,1,"TELEFONKONYV"
30 PRINT CHR$(147)"TELEFONKONYV KAZETTAN"
40 PRINT
50 INPUT "TELEFONTULAJDONOSOK SZAMA";N
60 PRINT: PRINT#13,N
70 FOR I=1 TO N
80 INPUT "■■TULAJDONOS NEVE";N$
90 PRINT#13,N$
100 INPUT "■■PONTOS CIME";C$
110 PRINT#13,C$
120 INPUT "■■TELEFONSZAMA";T$
130 PRINT#13,T$
140 PRINT
150 NEXT I
160 CLOSE 13
170 END
```

2. program:

```
10 REM ADATBEOLVASAS KAZETTAROL
20 OPEN 11,1,0,"TELEFONKONYV"
30 DIM N$(5),C$(5),T$(5)
40 FOR I=1 TO 5
50 INPUT#11,N$(I),C$(I),T$(I)
60 NEXT I
70 CLOSE 11
80 PRINT "■■■■■■■■■■■■KAZETTAN TAROLT TELEFONKONYV"
90 PRINT
100 PRINT "■■■■■■■■■■■■LAPOZAS BARMELYIK BILLENTYUVEL"
110 FOR I=1 TO 5
120 GET T$: IF T$="" THEN 120
130 PRINT CHR$(147)
140 PRINT "■■■■■■■■■■■■■■■■"N$(I)
150 PRINT "■■■■■■■■■■■■■■■■"C$(I)
160 PRINT "■■■■■■■■■■■■■■■■TELEFON: "T$(I)
170 NEXT I
180 GET V$: IF V$="" THEN 180
190 PRINT CHR$(147)
200 END
```

3. program:

```
10 REM RAKTARNYITO PROGRAM
20 INPUT "■RAKTAR AZONOSITOJA";R$
30 PRINT
40 OPEN 12,8,12,"@:"+R$+",S,W"
50 INPUT "CIKKEK SZAMA";N
60 PRINT#12,N
70 FOR I=1 TO N
80 PRINT: PRINT I;"■. CIKK",
90 INPUT C$
100 PRINT#12,C$
110 PRINT#12,M
120 NEXT I
130 CLOSE 12
140 END
```

4. program:

```
10 REM RAKTARKEZELO PROGRAM
20 PRINT "■■■■■■■■■■■RAKTAR KEZELO PROGRAM"
30 INPUT "■■■■■■■■A RAKTAR AZONOSITOJA";R$
40 OPEN 9,8,9,R$+",S,R"
50 INPUT#9,N
60 PRINT
70 PRINT "■■■■■■■■CIKKEK SZAMA:";N
80 PRINT "■■■■■■■■KEZDHETJUK A MUNKAT?"
90 GET M$: IF M$="" THEN 90
100 DIM C$(N),M(N)
110 FOR I=1 TO N
120 INPUT#9,C$(I),M(I)
130 NEXT I
140 CLOSE 9
150 PRINT CHR$(147)
160 PRINT "■■A RAKTAR PILLANATNYI HELYZETE"
170 PRINT "■■■CIKK NEVE          MENNYISEGE"
180 PRINT
190 FOR I=1 TO N
200 PRINT: PRINT TAB(2)C$(I);TAB(23)M(I)
210 GET T$: IF T$="" THEN 210
220 NEXT I
230 INPUT "■■■■VALTOZTATNI KIVANT CIKK";C$
240 IF C$="VEGE" THEN 340
250 K=0
```

```
260 FOR I=1 TO N
270 IF C$=C$(I) THEN K=I
280 NEXT I
290 IF K=0 THEN 230
300 INPUT "■■■VALTOZTATNI KIVANT MENNYISEG";M
310 IF -M>M(K) THEN 230
320 M(K)=M(K)+M
330 GO TO 150
340 PRINT "■■■■■■■■■■■■PROGRAM VEGE"
350 PRINT "■■■■■■■■■■■■TURELEM! KIIROM LEMEZRE!"
360 OPEN 12,8,12,"@:"+R$+",S,W"
370 PRINT#12,N
380 FOR I=1 TO N
390 PRINT#12,C$(I)
400 PRINT#12,M(I)
410 NEXT I
420 CLOSE 12
430 PRINT CHR$(147)
440 END
```

5. program:

```
10 REM RAKTARBOVITO PROGRAM
20 PRINT "■■■■■■■■■■RAKTARBOVITO PROGRAM"
30 INPUT "■■■■■■■■REGI RAKTAR NEVE";R$
40 OPEN 7,8,7,R$+",S,R"
50 INPUT#7,N
60 DIM C$(N),M(N)
70 FOR I=1 TO N
80 INPUT#7,C$(I),M(I)
90 NEXT I
100 CLOSE 7
110 INPUT "■■■■■■■■UJ RAKTAR NEVE";U$
120 INPUT "■■■■■■■■UJ CIKKEK SZAMA";NU
130 OPEN 8,8,8,"@:"+U$+",S,W"
140 PRINT#8,N+NU
150 FOR I=1 TO N
160 PRINT#8,C$(I)
170 PRINT#8,M(I)
180 NEXT I
190 PRINT "■UJ CIKKEK NEVE      MENNYISEGE"
200 PRINT
210 FOR J=1 TO NU
220 INPUT C$
230 PRINT "■",,
240 INPUT M
250 PRINT#8,C$
260 PRINT#8,M
270 NEXT J
280 CLOSE 8
290 END
```

Mintaprogramok magyarázata:

1. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: "TELEFONKONYV" néven adatterületet nyit (definiál) kazettán.
30-as utasítás: Üres képernyőre kiírja a program feladatát.
40-es utasítás: Üres sort hagy.
50-es utasítás: Beolvassa a telefontulajdonosok számát.
60-as utasítás: Üres sort hagy, s kiírja az előző adatot kazettára.
70-es utasítás: Megszervezi azt a ciklust, mely a billentyűzetről beolvasott adatokat kazettára írja.
80-as utasítás: Beolvassa a soron levő telefontulajdonos nevét.
90-es utasítás: Kiírja a kazettára.
100-as utasítás: Beolvassa a pontos címét.
110-es utasítás: Kiírja a kazettára.
120-as utasítás: Beolvassa a telefonszámot.
130-as utasítás: Kiírja a kazettára.
140-es utasítás: Üres sort hagy.
150-es utasítás: Befejezi a ciklust.
160-as utasítás: Befejezettnek nyilvánítja (bezárja) a "TELEFONKONYV" nevű kazettás adatterületet.
170-es utasítás: Program vége.

2. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Megkeresi a kazettán a "TELEFONKONYV" nevű adatterületet, hogy arról adatokat olvasson le. Ha sikerül, akkor a terület (adatállomány, file) megnyitásával előkészíti azt.
30-as utasítás: Mivel ez a program az előző párja, ezért 5 elemre számít (tudniillik ennyi adatsort írtunk rá), s ennek megfelelően deklaráljuk a tömböket.
40-es utasítás: Megszervezi az adatbeolvasó ciklust.
50-es utasítás: Beolvassa a kazettáról az összetartozó adathármasokat (név, cím, telefon).
60-as utasítás: Befejezi a ciklust.
70-es utasítás: Befejezettnek nyilvánítja az adatállomány olvasását. (Bezárja a file-t.)
80-as utasítás: Üres képernyőn a beolvasott telefonkönyv kiírását "vezeti be" a C-64.
90-es utasítás: Üres sort hagy.
100-as utasítás: Megadja a lapozás (a továbblépés) módját.
110-es utasítás: Megszervezi a képernyőre író (listázó) ciklust.
120-as utasítás: Feltételes várakozás.
130-160-as utasítások: Üres képernyőre kiírják egy-egy telefontulajdonos adatait.
170-es utasítás: Befejezi a ciklust.
180-as utasítás: Feltételes várakozás, hogy az utolsó telefontulajdonos adatait is zavartalanul szemlélhessük.
190-es utasítás: Képernyő törlése.
200-as utasítás: Program vége.

3. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Beolvassa a létrehozni kívánt raktár-adatterület nevét (azonosítóját).
30-as utasítás: Üres sort hagy.
40-es utasítás: A lemezen megnyitja írásra (kitöltésre) a 20-as utasítással beolvasott néven az adatterületet.
50-es utasítás: Beolvassa a raktárban előforduló cikkek számát.
60-as utasítás: Kiírja a lemezre a várható cikkek számát.
70-es utasítás: Megszervezi a cikklista beolvasását és lemezre írását végző ciklust.
80-as utasítás: Előkészíti a soron levő cikk nevének beolvasását.
90-es utasítás: Beolvassa a cikk nevét.
100-as utasítás: Kiírja lemezre a cikk nevét.
110-es utasítás: Kiírja lemezre a cikk induló (0) mennyiségét.
120-as utasítás: Befejezi a ciklust.
130-as utasítás: Befejezettnek nyilvánítja az adatterület írását. (lezárja a file-t).
140-es utasítás: Program vége.

4. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: A program "bejelentkező" sora.
30-as utasítás: Beolvassa a raktár (a feldolgozni kívánt adatterület) azonosítóját.
40-es utasítás: Olvasásra megnyitja a lemezen az előző lépésben megadott nevű adatterületet (adatállomány, file).
50-es utasítás: Beolvassa a lemezről a nyilvántartásban szereplő cikkek számát.
60-as utasítás: Üres sort hagy.
70-es utasítás: Kiírja a képernyőre a lemezről beolvasott információt.
80-as utasítás: Egy költői kérdés(, amelyre nem várunk igazi választ.)
90-es utasítás: Feltételes várakozás.
100-as utasítás: A lemezről beolvasandó információk tárolására két megfelelő elemszámú egydimenziós tömböt deklarálunk.
110-es utasítás: Megszervezi az adatbeolvasó ciklust.
120-as utasítás: Beolvassa lemezről a soron levő cikk nevét és raktározott mennyiségét.
130-as utasítás: Befejezi a ciklust.
140-es utasítás: Befejezettnek nyilvánítja az adatterület olvasását (lezárja a file-t).
150-es utasítás: Képernyőt töröl.
160-180-as utasítások: Előkészítik a pillanatnyi raktárkészlet képernyőre írását.
190-es utasítás: Megszervezi a raktárkészletet képernyőre író ciklust.
200-as utasítás: Kiírja a soron levő cikk nevét és raktározott mennyiségét.
210-es utasítás: Feltételes várakozás.
220-as utasítás: Befejezi a ciklust.
230-as utasítás: Beolvassa annak a cikknek a nevét, amelynek a mennyiségét meg akarjuk változtatni.

240-es utasítás: Ha az előző lépésben semmit sem akarunk megváltoztatni, akkor a "VEGE" beolvasásával elérjük, hogy ezt az állapotot tekintsük véglegesnek, s annak további sorsáról a 340-es utasítással kezdődő utasítássor intézkedik.
250-es utasítás: Előkészíti a megváltoztatni kívánt cikk előkeresését.
260-as utasítás: Megszervezi a cikket kereső ciklust.
270-es utasítás: Ha megtalálta, akkor a sorszámát félreteszi a K változóba.
280-as utasítás: Befejezi a ciklust.
290-es utasítás: Ha a keresés sikertelen volt, akkor a 230-as utasítással újat kér.
300-as utasítás: Megkérdezi a változtatni kívánt mennyiséget (Az előjelet a raktár szempontjából nézzük: kiadás = -, bevétel = +).
310-es utasítás: Ha a kért mennyiség meghaladja a készletet, akkor nem fogadja el a kérést, s helyette másikat kér.
320-as utasítás: Regisztrálja (jóváírja) a készletváltozást.
330-as utasítás: Visszatér a 150-es utasításra, hogy az aktuális helyzet kiírása után új kéréseknek tehessen eleget.
340-es utasítás: Üres képernyőre kiírja a "PROGRAM VEGE"-üzenetet.
350-es utasítás: Felhívja a figyelmet arra, hogy még ezután kerül sor az aktuális raktárkészlet lemezre írására.
360-as utasítás: Írás céljából megnyitja (deklarálja) az imént olvasott adatterületet.
370-es utasítás: Kiírja a lemezre a raktározható cikkek számát.
380-as utasítás: Megszervezi a lemezre író ciklust.
390-400-as utasítások: Kiírják az aktuális cikk nevét és mennyiségét.
410-es utasítás: Befejezi a ciklust.
420-as utasítás: Bezárja az adatterületet a lemezen.
430-as utasítás: Képernyőt töröl.
440-es utasítás: Program vége.

5. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Üres képernyőre írja a program célját.
30-as utasítás: Beolvassa a raktár azonosítóját.
40-es utasítás: Olvasásra megnyitja a megjelölt adatterületet a lemezen.
50-es utasítás: Beolvassa az eddig raktározott cikkek számát a lemezről.
60-as utasítás: Két egydimenziós tömböt deklarál az eddigi raktárkészlet tárolására.
70-es utasítás: Megszervezi a régi raktárkészlet beolvasását végző ciklust.
80-as utasítás: Beolvassa a lemezről a soron levő cikk nevét és aktuális mennyiségét.
90-es utasítás: Befejezi a beolvasó ciklust.
100-as utasítás: Befejezettnek nyilvánítja a régi adatterület beolvasását.
110-es utasítás: Beolvassa az új raktár nevét, mely megegyezhet a régivel is (de ekkor fennáll a régi adatok elvesztésének a veszélye a program befejezéséig).
120-as utasítás: Az új cikkek számának beolvasása.
130-as utasítás: Az új raktárat tartalmazó adatterület megnyitása lemezen.
140-es utasítás: A megnövelt cikkszám kiírása a lemezre.
150-es utasítás: A régi raktárkészletet lemezre író ciklus megszervezése.

160-170-es utasítások: A régi cikkek és mennyiségeik lemezre vitele.
180-as utasítás: Befejezi a ciklust.
190-200-as utasítások: Előkészítik az új cikkek és mennyiségeik beolvasását.
210-es utasítás: Megszervezi az új cikkek beolvasását és lemezre vitelét végző ciklust.
220-240-es utasítások: Beolvassák az aktuális adatpárt.
250-260-as utasítások: Kiírják a lemezre a megfelelő adatpárokat.
270-es utasítás: Befejezi a ciklust.
280-as utasítás: Bezárja az új raktárat tartalmazó adatállományt.
290-es utasítás: Program vége.

A leckében előforduló új fogalmak:

Adatállomány (file, adatterület):

A fogalmat folyamatosan definiáljuk (fogalmazzuk meg, "építjük fel"). Azokat a jelsorozatokat, amelyeket a C-64 egy lépésben dolgoz fel, **adatok-nak** nevezzük. Azok az adatsorozatok (rendszerint vesszővel elválasztott adatok), amelyeket az adatátvitel (beolvasás billentyűzetről, lemezről vagy kazettáról; kiírás képernyőre, kazettára, lemezre, mátrixnyomtatóra) során egy lépésben kezel, alkotják a **fizikai rekord**-ot. Azok az adatsorozatok, amelyek logikailag alkotnak hasonló egységet (pl. egy dolgozó neve, címe, telefonszáma, stb...), alkotják a **logikai rekord**-ot. Azokat az adatsorozatokat, amelyek egy-egy feladat adatait vagy eredményeit adják, nevezzük **adatállomány**-nak (file, adatterület). Bár valamennyi adathordozón előfordulhatnak, a megszokás általában csak a mágneses adathordozókat tekinti "igaziak"-nak.

Kazettás adatállomány:

Magnetofonkazettán rögzített olyan jelsorozat, melynek első adatai tartalmazzák az adatállomány azonosítóját, majd a felvitel és feldolgozás sorrendjében, vesszővel elválasztva a teljes adatsorozatot, végül az adatállomány végét jelző jelsorozatot.

Soros adatállomány lemezen:

Az előző adatállományhoz hasonló módon a lemezen is tárolhatunk egyszerű adatállományokat. (Már ekkor kiütközik a lemez azon előnyös tulajdonsága, hogy egyszerre több, rajta elhelyezett adatállomány kezelhető. S ekkor még további adatszervezési lehetőségeiről nem is szóltunk. Bővebben lásd Úry László kézikönyvének "A lemezegység használata" c. fejezetét.)

Adatállomány megnyitása:

Kazettás adatállomány esetén:

Általános alakja:

sorszám OPEN logikai szám,1,mód,"adatállomány azonosítója"

Az utasítás alkotóelemei és szerepük:

sorszám - meghatározza az utasítás programbeli helyét.

OPEN - BASIC-alapszó, amely utal arra, hogy adatállomány létrehozásához vagy feldolgozásához kezdünk hozzá.

logikai szám - olyan pozitív egész konstans vagy változó, amely utal az éppen feldolgozni kívánt adatállományra.

1 - a kazettás egység "beépített" fizikai azonosítószáma.

mód - a feldolgozás módját megadó egész szám. (Ha mód=1, akkor most fogjuk kazettára írni az adatállományt. Ha mód=0, akkor most fogjuk feldolgozni (a kazettáról beolvasni) az ott elhelyezett adatokat.)

Soros adatállomány lemezen:

Általános alakja:

sorszám OPEN logikai szám,8,csatornaszám,"adatállomány azonosítója,S,mód"

Az utasítás alkotóelemei és szerepük:

sorszám - megadja az utasítás programbeli helyét.

OPEN - BASIC-alapszó, amely jelzi, hogy a továbbiakban egy adatállomány létrehozásával vagy feldolgozásával kívánunk foglalkozni.

logikai szám - olyan pozitív egész konstans vagy változó, amely egyetlen számértékkel utal a feldolgozni kívánt adatállományra.

8 - a lemez egyik "beépített" fizikai száma,

csatornaszám - pozitív egész szám vagy változó, amely megadja, hogy a 13 rendelkezésünkre álló (2-14) csatorna közül melyiket használjuk adatállományunk kezelésére.

adatállomány azonosítója - karaktersorozat, mely megadja az adatállomány nevét. (Ha az adatállomány írásakor az azonosítót a @:-jellel vezetjük be, akkor ezzel mintegy engedélyezzük az esetleg ugyanilyen néven már létrehozott adatállomány felülírását.)

S - a hozzáférési módot adja meg - nevezetesen azt, hogy az adatokat sorban egymás után írjuk vagy olvassuk.

mód - a feldolgozási módot adja meg. (Ha mód=R, akkor a már létrehozott adatállományból fogjuk sorban egymás után kiolvasni az adatokat. Ha mód=W, akkor most fogjuk létrehozni az adatállományt, most fogjuk sorban egymás után kiírni az adatokat.)

Adatállomány olvasása:

Általános alakja:

sorszám INPUT#logikai szám,változó-lista

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét adja meg.

INPUT#logikai szám - BASIC-alapszó, amely arra utal, hogy az utána megadott logikai számú berendezésről akarjuk leolvasni az adatokat.

változó-lista - megadja, hogy mely változók kapjanak értéket az utasítás eredményeként.

Adatállomány írása:

Általános alakja:

sorszám PRINT#logikai szám,bővített változó-lista

Az utasítás alkotóelemei és szerepük:

sorszám - az utasítás programbeli helyét adja meg.

PRINT#logikai szám - BASIC-alapszó, amely megadja, hogy az utána következő logikai számmal azonosított berendezésre kívánjuk kiírni a bővített változó-lista tartalmát.

bővített változó-lista - vezérlő karakterekkel és szövegkonstansokkal bővített változó-lista. (A bővítésre azért van szükség, mert az olvasásnál mindig vesszőig olvas az INPUT#-utasítás.)

Adatállomány lezárása:

Általános alakja:

sorszám CLOSE logikai szám

Az utasítás alkotóelemei és szerepük:

sorszám - megadja az utasítás programbeli helyét.

CLOSE - BASIC-alapszó, mely lezárja az utána megadott sorszámú adatállományt.

Gyakorló feladatok:

14.1 Írjon programot, amely beolvassa egy csoport valamennyi tagjának személyi adatait (név, születési hely és idő, anyja neve, pontos lakcíme, munkahelye, foglalkozása), majd ezekből az adatokból "CSOPORT" néven adatállományt hoz létre kazettán vagy lemezen.

14.2 Írjon programot, amely az előző feladat eredményeként létrehozott adatállomány tartalmát tetszetős formában képernyőre listázza.

14.3 Írjon programot, amely az előbbi adatállományt "UJ CSOPORT" néven további néhány dolgozó személyi adatával bővíti.

14.4 Írjon programot, amely "KADERLAP" néven hoz létre bővített adatállományt, ahol az egyes dolgozók szakmai képzettségével, iskolai végzettségével, munkában eltöltött idejével, jövedelmével bővítjük az egyes dolgozókra vonatkozó tárolt információkat.

14.5 Írjon programot, amely az előző adatállományok tetszés szerinti adatának memóriabeli, majd adatállománybeli módosítását lehetővé teszi.

14.6 Írjon programot, amely a 14.4-es programmal létrehozott adatállományból képernyőre listázza az 1986-ban jubiláló (törzsgárda-fokozatot <10-15-20-25-30 év> elérő) dolgozókat.

14.6 Írjon programot, amely "HAZI KONYVTAR '86" azonosítóval lemezen tárolja otthoni könyvtára köteteinek szerzőjét, címét, kiadóját, kiadásának évszámát, terjedelmét, műfaját, témáját.

14.7 Írjon programot, amely házi könyvtárunkat a szokásos sorrend szerint sorbarendezi, majd a régi néven lemezre viszi.

14.8 Írjon programot, amely különböző típusú kereséseket hajt végre az adatállományon, s a kapott eredményt képernyőre viszi.

14.9 Írjon programot, amely egy gazdaság tábláiról az alapadatokat (azonosító, terület, termelt növény, költségráfordítás, felhasznált munkaerő, gép, szerves és szervetlen műtrágya, megtermelt termés, az általános érvényes egységár) tartalmazza.

14.10 Írjon programot, amely a gazdaság összesítő adatait kiszámítja az előbbi adatállomány felhasználásával (összes terület, összes felhasználások, összes termés, összes árbevétel).

14.11 Írjon programot, amely egy-egy növény esetében végzi el az előbbi összegzéseket.

14.12 Készítsen programot, mely megkeresi, hogy melyik növény, melyik táblán hozta a legnagyobb átlagtermést.

14.13 Írjon programot, amely a pillanatnyi információk alapján meghatározza, hogy abszolút és relatív értelemben melyik növény volt a leggazdaságosabb.

15.LECKE: A mátrixnyomtató alkalmazása

Mintaprogramok:

1. program:

```
10 REM FUGGVENYTABLAZAT A MATRIXNYOMTATON
20 OPEN 3,4
30 PRINT#3,,,"FUGGVENYTABLAZAT"
40 PRINT#3
50 PRINT#3,,,"================"
60 FOR I=1 TO 3: PRINT#3: NEXT I
70 PRINT#3,"    A L F A      SIN           COS"
80 PRINT#3
90 FOR AL=0 TO 60
100 X=AL*π/180
110 Y=SIN(X)
120 Z=COS(X)
130 Y=INT(10000*Y+0.5)/10000
140 Z=INT(10000*Z+0.5)/10000
150 P1=7: IF AL>=10 THEN P1=6
160 IF AL=0 THEN PRINT#3,TAB(P1)AL;CHR$(16)"17"Y;CHR$(16)"30"Z
170 IF AL>0 THEN PRINT#3,TAB(P1)AL;CHR$(16)"15"Y;CHR$(16)"28"Z
180 NEXT AL
190 PRINT#3
200 CLOSE 3
210 END
```

2. program:

```
10 REM RAKTARKESZLET EREDMENYLISTAJA MATRIXNYOMTATON
20 PRINT CHR$(147)
30 INPUT "■■■■■■■      RAKTAR AZONOSITOJA";R$
40 OPEN 7,8,9,R$+",S,R"
50 INPUT#7,N
60 DIM C$(N),M(N)
70 FOR I=1 TO N
80 INPUT#7,C$(I),M(I)
90 NEXT I
100 CLOSE 7
110 INPUT "■■■■      DATUM";D$
120 OPEN 4,4: PRINT#4: PRINT#4
130 PRINT#4,TAB(21)CHR$(14);"RAKTARKESZLET";CHR$(15)
140 PRINT#4,TAB(21)CHR$(14);"============";CHR$(15)
150 PRINT#4: PRINT#4: PRINT#4
160 PRINT#4,TAB(8)"HELY: ";R$;CHR$(16)"45";"DATUM: ";D$
170 FOR I=1 TO 5: PRINT#4: NEXT I
180 PRINT#4,"          CIKK MEGNEVEZESE          MENNYISEGE(T)"
190 PRINT#4
200 FOR I=1 TO N
210 PRINT#4
```

```
220 IF M(I)>=1000 AND M(I)<10000 THEN PRINT#4,
    TAB(9)C$(I);CHR$(16)"35"M(I)
230 IF M(I)>=100 AND M(I)<1000 THEN PRINT#4,
    TAB(9)C$(I);CHR$(16)"36"M(I)
240 IF M(I)>=10 AND M(I)<100 THEN PRINT#4,
    TAB(9)C$(I);CHR$(16)"37"M(I)
250 IF M(I)>=0 AND M(I)<10 THEN PRINT#4,
    TAB(9)C$(I);CHR$(16)"38"M(I)
260 NEXT I
270 PRINT#4
280 CLOSE 4
290 END
```

3. program:

```
10 REM TELEFONKONYV A MATRIXNYOMTATON
20 OPEN 11,1,0,"TELEFONKONYV"
30 INPUT#11,N
40 DIM N$(N),C$(N),T$(N)
50 FOR I=1 TO N
60 INPUT#11,N$(I),C$(I),T$(I)
70 NEXT I
80 CLOSE 11
90 OPEN 5,4
100 PRINT#5,,CHR$(14);"HAZI TELEFONKONYV";CHR$(15)
110 PRINT#5
120 PRINT#5,,CHR$(14);"--------------------";CHR$(15)
130 PRINT#5
140 PRINT#5
150 PRINT#5,"  TULAJDONOS NEVE                ";
152 PRINT#5,"PONTOS CIME                    ";
155 PRINT#5,"TELEFONSZAMA"
160 PRINT#5
170 FOR I=1 TO N
180 PRINT#5
190 PRINT#5,TAB(3)N$(I);CHR$(16)"25";C$(I);CHR$(16)"55";T$(I)
200 NEXT I
210 PRINT#5
220 CLOSE 5
230 END
```

4. program:

```
10 REM FUGGVENYGORBE A NYOMTATON
20 OPEN 6,4
30 FOR AL=0 TO 65
40 P=35+33*SIN(AL*π/33)
50 PRINT#6,TAB(P)"O"
60 NEXT AL
70 PRINT#6
80 CLOSE 6
90 END
```

5. program:

```
10 REM FEJLODESI STATISZTIKA
20 PRINT CHR$(147)
30 DIM N$(19)
40 DIM ME(19)
50 DIM KF(19)
60 DIM MF(19)
70 FOR I=1 TO 19
80 PRINT "■KOVETKEZIK A "I"■. DOLGOZO"
90 INPUT "■DOLGOZO NEVE";N$(I)
100 INPUT "■MUNKABALEPES EVE";ME(I)
110 INPUT "■KEZDO FIZETES";KF(I)
120 INPUT "■MOSTANI FIZETES";MF(I)
130 NEXT I
140 OPEN 13,4
150 PRINT#13,CHR$(14)"  DOLGOZOK FEJLODESI STATISZTIKAJA"
160 PRINT#13: PRINT#13
170 PRINT#13,"  ==============================";CHR$(15)
180 FOR I=1 TO 3: PRINT#13: NEXT I
190 PRINT#13,TAB(4)"DOLGOZO NEVE    KEZDES EVE EVES FEJLODES"
200 PRINT#13
210 FOR I=1 TO 19
220 F=(MF(I)-KF(I))/(1986-ME(I))
230 F=INT(100*F+0.5)/100
240 PRINT#13,TAB(4)N$(I);CHR$(16)"23"ME(I);CHR$(16)"33"F
250 NEXT I
260 CLOSE 13
270 END
```

.

Mintaprogramok magyarázata:

1. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Munkára előkészíti a 3-as logikai számú, 4-es fizikai számú mátrixnyomtatót.

30-as utasítás: 20 üres hely kihagyása után kiírja a mátrixnyomtatón a "FUGGVENYTABLAZAT" karaktersorozatot.
40-es utasítás: Üres sort hagy a papíron.
50-es utasítás: Aláhúzza az előbb kiírt karaktersorozatot.
60-as utasítás: 3 üres sort emel.
70-es utasítás: Kiírja függvénytáblázatunk "fejléc"-sorát.
80-as utasítás: Üres sort hagy a papíron.
90-es utasítás: Megszervezi a függvény-értékeket kiszámító és kinyomtató ciklust.
100-as utasítás: Átszámítja a fokban megadott szöget radiánba (ívhosszba), ahol a szorzásjel után a PI szerepel.
110-es utasítás: Kiszámítja a SIN(X)-értéket.
120-as utasítás: Kiszámítja a COS(X)-értéket.
130-as utasítás: 4 tizedesjegyre kerekíti az Y értékét.
140-es utasítás: 4 tizedesjegyre kerekíti a Z értékét.
150-es utasítás: Pl értékét AL nagyságrendjétől függően állítja be.
160-170-es utasítások: AL értékétől függően más és más formában írják ki a függvényértékeket.
180-as utasítás: Befejezi a ciklust.
190-es utasítás: Üres sort hagy a papíron.
200-as utasítás: Befejezettnek nyilvánítja a mátrixnyomtató használatát. (Lezárja a mátrixnyomtatóra definiált adatállományt.)
210-es utasítás: Program vége.

2. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Képernyőt töröl.
30-as utasítás: Beolvassa a feldolgozni kívánt raktár azonosítóját.
40-es utasítás: Megnyitja az előző lépésben megadott nevű csatornaszámmal, soros feldolgozásra(S), s olvasásra(R).
50-es utasítás: Beolvassa a lemezről a raktárban előforduló cikkek számát.
60-as utasítás: A cikkszám ismeretében két egydimenziós tömböt foglal le.
70-es utasítás: Megszervezi az adatok beolvasását végző ciklust.
80-as utasítás: Beolvassa a soron levő adatpárt (cikk azonosítója és mennyisége).
90-es utasítás: Befejezi az adatbeolvasó ciklust.
100-as utasítás: Bezárja a lemezen levő adatállományt.
110-es utasítás: Beolvassa az aktuális dátumot a billentyűzetről.
120-as utasítás: Megnyitja a mátrixnyomtatóra szánt adatállományt 4-es logikai és fizikai számmal.
130-as utasítás: A következő sor 21. pozíciójától kezdve a CHR$(14) vezérlőkarakter segítségével duplaszéles betűkkel kiírja a "RAKTARKESZLET" feliratot. A CHR$(16)"45" hatására a 45. pozíciótól kezdve kinyomtatja a szükséges információkat.
170-es utasítás: 5 üres sort hagy a papíron.
180-as utasítás: Kinyomtatja a táblázat oszlopainak fejléc-rovatát.
190-es utasítás: Üres sort hagy a papíron.
200-as utasítás: Megszervezi az eredménytáblázatot kinyomtató ciklust.
210-es utasítás: Üres sort nyomtat.
220-250-es utasítások: M(I) értékétől függően más és más nyomtatási formával éri el, hogy a raktározott mennyiséget helyiérték-helyesen nyomtassa ki.

260-as utasítás: Befejezi a ciklust.
270-es utasítás: Üres sort nyomtat.
280-as utasítás: Befejezettnek nyilvánítja a nyomtatást.
290-es utasítás: Program vége.

3. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Megnyitja olvasásra a kazettán(1,0) "TELEFONKONYV" néven tárolt adatállományt 11-es logikai számmal.
30-as utasítás: Beolvassa a tényleges logikai rekordok (név, cím, telefonszám) számát.
40-es utasítás: Ennek ismeretében definiál 3 megfelelő elemszámú tömböt.
50-es utasítás: Megszervezi az adatbeolvasó ciklust.
60-as utasítás: Beolvassa a soronlevő adathármast.
70-es utasítás: Befejezi a beolvasó ciklust.
80-as utasítás: A 11-es logikai számú adatterület lezárásával befejezettnek nyilvánítja az adatok beolvasását.
90-es utasítás: Megnyitja 5-ös logikai és 4-es fizikai számmal a mátrixnyomtatót.
100-as utasítás: Az előző programhoz hasonlóan a CHR$(14)-CHR$(15) vezérlőkombináció segítségével kinyomtatja a táblázat címét.
110-es utasítás: Üres sort hagy a papíron.
120-as utasítás: Aláhúzza a feliratot.
130-140-es utasítások: Üres sort hagynak a papíron.
150-155-ös utasítások: Kinyomtatja a táblázat oszlopainak fejléc rovatát.
160-as utasítás: Üres sort hagy a papíron.
170-es utasítás: Megszervezi a nyomtató ciklust.
180-as utasítás: Üres sort hagy.
190-es utasítás: A TAB-függvény és a már alkalmazott CHR$(16) segítségével kinyomtatja az aktuális adathármast.
200-as utasítás: Befejezi a ciklust.
210-es utasítás: Üres sort hagy.
220-as utasítás: "Bezárja" a nyomtatót.
230-as utasítás: Program vége.

4. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: 6-os logikai és 4-es fizikai számmal munkára előkészíti a nyomtatót.
30-as utasítás: Megszervezi a nyomtatást végző ciklust.
40-es utasítás: Kiszámítja a nyomtatási pozíciót. (A szorzásjel után PI van.)
50-es utasítás: A soron levő sor P-ik pozíciójára "siftelt" W-t ír.
60-as utasítás: Befejezi a ciklust.
70-es utasítás: Üres sort hagy.
80-as utasítás: Befejezettnek nyilvánítja a nyomtatást.
90-es utasítás: Program vége.

5. program:

10-es utasítás: Megjegyzés, amely közli a program célját.
20-as utasítás: Letörli a képernyőt.
30-60-as utasítások: Definiálják a szükséges tömböket.
70-es utasítás: Megszervezi a beolvasó ciklust.
80-as utasítás: Üres képernyőre kiírja a soron levő dolgozó sorszámát.
90-120-as utasítások: Beolvassák a soron levő dolgozó aktuális adatait.
130-as utasítás: Befejezi a beolvasó ciklust.
140-es utasítás: Megnyitja a mátrixnyomtatót.
150-es utasítás: Duplaszéles címet ír.
160-as utasítás: Két sort kihagy.
170-es utasítás: Aláhúzza az imént kiírt szöveget.
180-as utasítás: 3 sort kihagy.
190-es utasítás: Kinyomtatja a táblázat oszlopainak fejléc rovatait.
200-as utasítás: Üres sort hagy.
210-es utasítás: Megszervezi az eredményeket kiszámító és kinyomtató ciklust.
220-as utasítás: Kiszámítja az egy évre jutó átlagos fizetésnövekedést.
230-as utasítás: Két tizedesjegyre kerekíti az előbb kiszámított eredményt.
240-es utasítás: Kinyomtatja az aktuális eredménysort.
250-es utasítás: Befejezi a ciklust.
260-as utasítás: Befejezettnek nyilvánítja a nyomtatást.
270-es utasítás: Program vége.

A leckében előforduló új fogalmak:

A mátrixnyomtató használata:

Ha a mátrixnyomtató segítségével programjaink eredményeit dokumentálni kívánjuk, akkor a következő utasítássorozatot kell alkalmaznunk:

sorszám OPEN logikai szám,fizikai szám
sorszám CMD logikai szám
sorszám PRINT#logikai szám,output-lista
sorszám PRINT#logikai szám
sorszám CLOSE logikai szám

Az utasítássorozat elemei és szerepük:

sorszám - az aktuális utasítás programbeli helyét határozza meg.

OPEN - BASIC-alapszó, amely egy adatállomány megnyitását idézi elő.

logikai szám - egész konstans vagy változó, amellyel az adatállományra hivatkozunk.

fizikai szám - ez mátrixnyomtató esetében szinte mindig 4 (esetleg 5)

CMD - BASIC-alapszó, mely az elsődleges output-berendezés hovatartozásáról dönt. Ez általában a képernyő, s ettől példáinkban sem tértünk el. Ezért nem alkalmaztuk a CMD-utasítást.

PRINT#logikai szám - a kazettához és lemezhez hasonlóan ez hajtja végre a tényleges adatátvitelt (a nyomtatást). (Ebből annyit írunk, ahány sort akarunk látni a papíron.)

CLOSE - ez is pontosan úgy működik, mint kazetta és lemez esetén a megfelelő párja.

Mátrixnyomtatót vezérlő jelsorozatok:

Jelsorozat	Hatása
TAB(K) vagy SPC(K)	K pozíciót kihagy.
CHR$(16)"23"	a 23. pozícióra kezdi nyomtatni a kiírandót.
CHR$(14)	az ezután felsorolt jeleket duplaszéles betűkkel írja
CHR$(15)	az ezután felsorolt jeleket normál méretben írja

Megjegyzés: Felsorolásunk csak néhány lehetőséget sorol fel. Sokkal teljesebb listát kapunk, ha fellapozzuk a kézikönyveket. Nyilvánvaló, hogy vezérlés nélkül normál módon dolgozik a mátrixnyomtató.

Gyakorló adatok:

15.1 Írjon programot, amely egy legfeljebb 66-soros szöveget 13 példányban kinyomtat.

15.2 Írjon programot, amely egy meghívólevelet adott névtömb alapján személyre szóló megszólítással vezet be.

15.3 Írjon programot, amely 0.1 és 15.5 között egy tizedes lépésközzel 3 tizedes pontossággal kinyomtatja a LOG(X) és az EXP(X) függvények értéktáblázatát.

15.4 Irjon programot, amely beolvassa egy osztály tagjainak (dolgozóinak) nevét, születési helyét, személyi számát, lakcímét, majd külön táblázatban kinyomtatja a férfiak s nők többi adatát.

15.5 Fejlessze úgy tovább az előző programot, hogy keresse ki a Budapesten született és vidéken lakó lányokat, s egy táblázatban csak őket szerepeltesse.

15.6 Írjon programot, mely az előbbi adatbázisból kiemeli a katonaköteles férfiakat.

15.7 Írjon programot, amely stilizált tabellát (labdarúgás, jégkorong, kézilabda, stb...) ír ki a mátrixnyomtatóra.

15.8 Készítsen érdeklődési körének megfelelő témakörben (sport, kultúra, fogyasztás) az európai országokat rangsorba állító táblázatot.

15.9 Készítsen olyan programot, amely egy számítás eredményét kívánság szerint a képernyőre vagy mátrixnyomtatóra írja.

15.10 Készítse el a magyar zászló szimbólikus képét a mátrixnyomtatón. (Ha nincs ötlete: P - piros, F - fehér, Z - zöld legyen.)

15.11 A grafikus jelek segítségével tervezzen minél összetettebb mértani alakzatot a mátrixnyomtatóra.

15.12 Készítse el kedvenc növénye stilizált rajzát a mátrixnyomtatón.

15.13 Készítse el kedvenc figurája stilizált rajzát a mátrixnyomtatón.