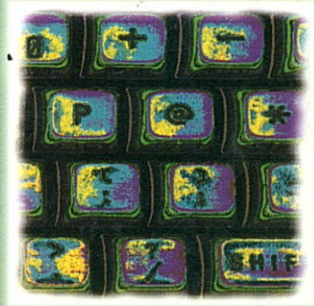
RUZSINSZKINÉ LUKÁCSY MARGIT

SZÁMÍTÁSTECHNIKAI ALAPISMERETEK



NEMZETI TANKÖNYVKIADÓ

Pontus Könyvesbolt és Csomagküldő Szolgálat

Tisztelt Megrendelőnk!

Elnézését kérjük a késedelmes szállításért. Sajnos az Ön által rendelt könyvek eltérő időpontban jelentek meg. Ruzsinszkiné Lukácsy Margit: Számítástechnikai alapismeretek című könyve csak a napokban érkezett meg hozzánk. Tekintettel a magas postai díjakra, szerettük volna megkímélni Önt a kétszeri szállítással jelentkező felesleges kiadástól. Ezért most , egy csomagban küldjük a könyveket.

Tisztelettel

Varga Zoltánné
boltvezető

Ruzsinszkiné Lukácsy Margit

Számítástechnikai alapismeretek

Nemzeti Tankönyvkiadó, Budapest

Lektorok:

HABLICSEKNÉ RICHTER MÁRIA
középiskolai tanár

MADAS PÁL
középiskolai tanár

KNORRNÉ SKAPÉR ÉVA
középiskolai tanár

ISBN 963 18 5329 2

Bevezetés

Az utóbbi években Magyarországon is meggyorsult a számítástechnikai eszközök elterjedése.

A mikroelektronikai ipar rohamos fejlődésének eredményeként megsokszorozódott a számítástechnikai eszközök teljesítménye, és csökkent a beszerzési áruk. A családok számára elérhetővé váltak a számítógépek, így azok beköltöztek az otthonokba.

A fejlődés eredménye az is, hogy a számítógép kezelése ma már nem igényel különleges számítástechnikai szaktudást. A számítógépek eredményes felhasználásának azonban elengedhetetlen feltétele, hogy az élet minden területén legyenek olyan szakemberek, akik a saját területükön képesek kihasználni a számítógépek adta lehetőségeket.

Különböző munkahelyeken találkozhatunk számítógépekkel, ezért nagyon fontos, hogy minél többen közel kerüljenek hozzájuk. Ismerjék meg a gépek kezelését, értsék meg, hogy mit közöl velük a számítógép. Vegyék észre, hogy hiba esetén nem a számítógép romlik el, hanem a program, amit javítani lehet, vagy az adat rossz, amelyet szintén lehet módosítani.

Sok esetben csak a kész programot kell a számítógépbe tölteni, és elég megismerni a program alkalmazását, felhasználási területeit.

Szeretnénk elérni, hogy minden diák tanulja meg az iskolában rendelkezésre álló számítógép üzembe helyezését, használatát, tudja futtatni a számítógéphez kapcsolódó programokat, és tudja javítani a tévesen beírt sorokat.

A számítógépes műveltség megalapozása érdekében röviden végigkövetjük a mai számítógépek kialakulásának történetét.

Megismerkedünk a számítógépek felépítésével, hogy kicsit közelebb kerüljünk hozzájuk, próbáljuk megérteni működésüket.

Olvashatunk az IBM PC számítógépre írt PC DOS operációs rendszer lényegéről, szerepéről, fontosságáról, a programok futtatásának feltételeiről, a PC DOS alapjairól és a legfontosabb parancsairól. Megtanuljuk a programok másolását és a saját lemezünkre történő elhelyezését. Ehhez szükségünk lesz új lemez formázására, illetve a meglévő könyvtárak között történő mozgásra vagy új könyvtár létrehozására.

Nem elégszünk meg azzal, hogy meglévő programokat futtatunk, hanem magunk is írunk néhányat. Ehhez természetesen szükségünk lesz valamilyen programozási nyelvre. Mi a BASIC nyelvet használjuk a Commodore 64 számítógépre, illetve a QBASIC nyelvet az IBM PC-re alkalmazva. A tankönyvben található programokat is így készítettük. A megírt programok kicsit módosítva más számítógépen is felhasználhatók.

A tankönyvünkben található programok többsége kapcsolódik a matematika-tananyaghoz. Lehetőségünk nyílik matematikafeladataink számítógépes megoldására, de egyszerűbb esetekben a zsebszámológépes megoldás módját is megmutatjuk. Ehhez segítséget kaphatunk a Függelékben található táblázatból, amely tartalmazza a keresett program helyét a könyvben.

A számítástechnikában szerzett ismereteinket alkalmazzuk a gyakorlatban is, a példaprogramok alapján próbálkozunk magunk is programírással, részünk lesz abban az örömben, amit az első program elkészítése jelent, amikor a számítógép azt csinálja, amit mi parancsolunk neki.

A lényeg megértését azzal is igyekszünk segíteni, hogy a fejezetben található legfontosabb tényeket röviden összefoglaljuk.

A nagyobb egységek végén kérdéseket, feladatokat is találunk, amelyekre érdemes válaszolni, mert abból kiderülhet, hogy megértettük-e a feldolgozott anyagot, vagy egyes részeit ismét át kell nézni.

Előfordulhat, hogy olyan fogalmakkal találkozik valaki, amelyeket nem ért. Ilyenkor lapozza fel a Függeléket, az ott lévő szótárban legtöbbször megtalálja a fogalom rövid magyarázatát.

A tankönyvben található kötelező anyag mellett az apró betűs részek további kiegészítő ismereteket tartalmaznak. Az irodalomjegyzékben felsorolt könyvek segítségével összetettebb feladatok megoldására is lehetőség nyílik.

Jó munkát kíván

a Szerző

I. A számítógép fejlődésének története

A) A számolást segítő eszközök fejlődése

1. A számolás első jelei és nehézségei

Kezdetben az emberek emlékezetük segítésére kezükbe vettek néhány kavicsot, majd vonalakat húztak a földre, később papírra.

Olvasmány

Példa erre a juhász esete, aki reggel, amikor kiengedte nyáját a karámból, minden birka után egy-egy kavicsot tett egy hordóba. Este, amikor hazaterelte a nyáját, kivett egy kavicsot, amikor beengedett egy birkát. Így tájékozódott a juhok számáról.

A számokat nemcsak ilyen fizikai egységekkel, hanem jelekkel, betűkkel is igyekeztek ábrázolni, de ezekkel a számolás kezdetben nagyon bonyolult volt.

Olvasmány

A számolás nagyon nehézkes lenne, ha napjainkban is a római számokkal, vagy az ókori ékírásos jelekkel kellene a műveleteket elvégeznünk.

Például: a 654 + 319 = 973 *helyett a* DCLIV + CCCXIX = CMLXXIII *jelekkel írnánk le egy összeadást.*

A számolások megkönnyítésére táblázatokat készítettek (a függvénytáblázatban találunk hasonlókat), vagy szavakkal leírták a számolás menetét, tehát algoritmust készítettek, ami már a számítástechnika kezdeti lépéseit jelentette.

Az arab számok bevezetése, a helyiérték felismerése, a tízes számrendszer alkalmazása megszüntette az áttekinthetetlen számolásokat, és lehetővé tette, hogy a tudósok ne a számok jeleinek rajzolgatásával töltsék idejüket, hanem belemélyedjenek a matematika rejtett titkainak megfejtésébe. (A többség továbbra is „hivatásos matematikust" alkalmazott, ha számolást kellett végeznie.)

2. A számolást segítő eszközök

Az egyik legegyszerűbb és legrégebbi számolóeszköz az abakusz, amely körülbelül a pénzzel egyidős (5000 éves), és napjainkban is találkozhatunk kínai változatával, amely huzalokon mozgatható fagolyókból áll.

A skót *Napier* fegyverek kitalálásával foglalkozott, és közben felfedezte és kidolgozta a logaritmust mint számítási eljárást.

Egy angol hivatalnok, *Qughtred* felfedezte, hogy a logaritmust alkotó számsor megjeleníthető egy vonalzószerű eszközön. 1621-ben elkészítette a logarlécet, amelyen a szorzás és az osztás is csak távolságok összege, illetve különbsége volt.

3. Mechanikai számológépek

A logarléc után nem sokat kellett várni az első számolásra alkalmas gép elkészítésére sem.

Wilhelm Schickard matematikusprofesszor 1623-ban olyan eszközt épített, amely fogaskerekei és fogaslécei segítségével összeadott, kivont, szorzott, osztott.

Minderről csak az 1950-es években szereztünk tudomást Kepler hagyatékának rendezgetése közben. Ez az oka, hogy *PASCAL*nak tulajdonították addig az első olyan gépet, amely gépi mechanizmussal végezte az összeadást és a kivonást. Ő azonban csak 1642-ben készült el fogaskerekek segítségével működő összeadógépével. Pascal rájött arra, hogy hogyan lehet a műveletek maradékát a következő helyiértékre átvinni, ezért tíznél nagyobb számokkal is tudott műveleteket végezni.

Leibniz 1673-ban készült el fogaskerekes számológépével, amely szorzásra és osztásra is alkalmas volt, ismételt összeadások és kivonások segítségével.

B) Mechanikus és elektromechanikus számítógépek

1. Az első igazi számítógép

A számítógépek fejlődésének újabb állomását *BABBAGE* munkássága jelentette. 1822-ben egy olyan számítógép működő modelljét mutatta be, amellyel a számításokat könnyebben és pontosabban lehetett elvégezni. Elkészítette a világ első, speciális célú, mechanikus működésű, digitális számítógépét.

A tervrajzok alapján hozzákezdett az úgynevezett differenciális számítógépnek a megépítéséhez, de a kor technikája nem tette lehetővé, hogy elkészítse azt. 1833-tól már más gondolatok foglalkoztatták.

Analitikus számítógépének tervezésekor abból indult ki, hogy az a gép, amely alkalmas egyetlen feladat elvégzésére, az valószínűleg alkalmassá tehető bármilyen matematikailag leírható művelet elvégzésére is. Megállapította, hogy olyan számítógépet kell készíteni, amely kezelője kívánságára bármilyen számítást el tud végezni. Ez a számítógép több önálló egységből épülne fel, és ezek az egységek különböző típusú műveleteket végeznének. Analitikus számítógépe is mechanikus elemekből épült volna, ezért a korabeli műszaki korlátok miatt ez sohasem készült el. Tervével felfedezte a mai számítógépek működésének néhány alapelvét, melyek a következők:

a) A bemeneti egységek segítségével lehetett betáplálni a számokat és a vezérlőutasításokat (külső programozás).
b) A számítási műveleteket egy aritmetikai egység végezte (processzor).
c) A vezérlőegység arról gondoskodott, hogy a megfelelő műveleteket a számítógép megfelelő sorrendben végezze el.
d) A műveletközi eredményeket, adatokat a belső tároló vagy memória raktározza el, hogy szükség esetén újból a gép rendelkezésére bocsássa azokat.
e) Az eredmények megjelenítésére digitális, azaz számjegyeket használó kimeneti egység szolgált.

Babbage gépének működéséhez emberi erőre volt szükség, amit később gőzgéppel akart felcserélni. Terve az volt, hogy a vezérlőutasítások bevitelére alkalmazza a Jacquard által már 1805-től a szövőszéken használt lyukkártyákat.

2. A népszámlálás hatása a számítástechnikára

A számokkal végzett műveletek gépesítése mellett jelentős probléma volt az összeszámlálás automatizálása, a nagy tömegű adatok rendszerezésének, osztályozásának gépesítése is. (A népszámlálásnál például egyenként kell összesíteni, hány férfi és nő, 15 éves vagy 60 éves, házas vagy nőtlen, kereső vagy eltartott van az országban.) Az Amerikai Egyesült Államok Népszámlálási Hivatala az 1890-es népszámlálás adatainak értékelésére pályázatot hirdetett, melyet *HOLLERITH* nyert meg a lyukkártyás feldolgozás ötletével.

Ő alkalmazta először a lyukkártyát statisztikai adatok tárolására, és ezzel megalapozta a gépi adatfeldolgozást.

Eredményeire az egész világon felfigyeltek, és olyan kereslet mutatkozott a gépre, hogy kielégítésére Hollerith létrehozta Washingtonban a Tabulating Machine Company nevű vállalatát, s ennek utóda a ma is ismert International Business Machine Corporation, az IBM.

Olvasmány

Hollerith gépének gyorsaságát mutatja, hogy az 1890-*es népszámlálás adatait (*63 *millió emberét) alig hat héttel a népszámlálás befejezése után már rendszerezték. (Az előző népszámlálását tíz év alatt sem sikerült értékelni.) Hollerith tabulátorát villanymotor hajtotta, és a kártyaosztályozóban is az elektromosságot használta fel.*

3. Az első elektromechanikus, digitális számítógépek

A számítógépek gyors fejlődését gátolta, hogy a velük kapcsolatos kutatásokat titokban tartották, ezért a tudósok egymástól függetlenül dolgoztak. A következőkben bemutatunk néhány közel egy időben történt számítógép készítő kísérletet.

Németországban *Zuse* kezdett hozzá egy működő számítógép tervezéséhez és készítéséhez, amelyet különböző matematikai feladatok megoldására szeretett volna programozni. Már 1936 körül az az ötlete támadt — a világon először —, hogy decimális számrendszer helyett bináris számrendszert kellene alkalmazni. A sorozat következő gépein további újításokat vezetett be.

Zuse a világon először alkalmazott működő számítógépében elektromágneses reléket és kódlyukakkal lyukasztott filmet.

Az Amerikai Egyesült Államokban *Aiken* és társai 1943-ban elkészítették az akkori legkorszerűbb technológia felhasználásával Babbage analitikus számítógépének modern változatát, a Harvard Mark I-et. Ennek az elektromágneses relék voltak a legfontosabb alkotóelemei. Nagyon lassan és zajosan működött.

Olvasmány

A Mark I. néhány jellemző adata:

- 23 *számjegyre* 1 *művelet sebessége: összeadás, kivonás* 0,3 s; *szorzás* 6 s; *osztás* 11,4 s.
- *Hűtéséhez naponta több tonna jég kellett.*
- *Hossza* 16,5 m, *magassága* 2,40 m, *tömege* 6 t *volt.*
- *Közel egymillió alkatrészből állt.*

A háború alatt Angliában a németek rejtjeleinek megfejtésére készítettek el egy újabb számítógépcsaládot, a Colossust. 1943 decemberében kezdett dolgozni az első számítógép, melyben elektroncsövekből kialakított áramköröket alkalmaztak.

C) Elektroncsöves számítógépek

1. Elektronikus számítógép

A mai fogalmaink szerinti első számítógépet 1946 februárjában a pennsylvaniai egyetemen mutatta be *dr. John Mauchly* és csoportja, köztük *Presper Eckert*. Az új számítógép neve ENIAC (Elektronic Numerical Integrator and Computer), amelyben elektroncsövek tették lehetővé a rövid kapcsolási időt. Az ENIAC külső vezérlésű gép volt.

Olvasmány

Az ENIAC hatalmas méreteiről álljon itt néhány adat:

- 30 m *hosszú és* 30 t *tömegű volt (egy termet elfoglalt).*
- 18 *ezer elektroncsövet tartalmazott.*
- 6 *ezer kapcsoló és* 70 *ezer ellenállás volt benne.*
- *Óriási áramfogyasztása miatt* 150 kW *teljesítményű volt a fűtőképessége.*

Az ENIAC működése közben felmerülő problémák:

- *Hatalmas mennyiségű hő keletkezett működése közben, ami nemcsak a mérnököket, matematikusokat tette próbára, hanem a kondenzátorokból is kiolvasztotta a szigetelőanyagot.*
- *Naponta több elektroncső kiégett, ezért csak rövid számolásokat tett lehetővé.*
- *Sok gondot okozott, hogy méreteihez képest kicsi volt a gép belső memóriája.*
- *A külső programvezérlést dugaszolótáblákkal oldották meg.*
- *Elvileg programozható volt, de az átállás más programra nagyon bonyolult feladatot jelentett.*

2. Tárolt programmal működő számítógépek

1946-ban Eckerték csoportjához csatlakozott a magyar származású *NEUMANN JÁNOS* is. A kutatók ekkor kezdték az új számítógép, az EDVAC építését. Neumann azt javasolta, hogy a programot ne külső információhordozón (lyukkártyán, lyukszalagon), hanem magában a gépben, annak központi tárolójában helyezzék el. Ezzel az elképzeléssel elkezdődött a számítógépek történetének egy újabb fejezete.

Olvasmány

Neumann János 1903. *december* 23-*án született Budapesten. Apja bankár volt.* 1914-*ig magánúton folytatta tanulmányait, majd gimnáziumba került.* 1921-*ben érettségizett, de már ekkor számon tartották matematikai munkásságát. Egyszerre két egyetemre járt. Budapesten matematikából doktorált, Zürichben pedig vegyészmérnöki diplomát kapott.* 1927-*ben már a berlini egyetem magántanára volt. Nagyon sokrétű tudományos munkát végzett. Közülük néhány:*

— *kvantummechanika (ma is időtálló),*
— *játékelmélet (az ő nevéhez fűződik),*
— *meteorológia, hidro- és aerodinamika,*
— *számítástechnika,*
— *automataelmélet.*

1955-ben megbetegedett, és 1957. február 8-án meghalt.
Hazánkban róla nevezték el a Számítógéptudományi Társaságot.

A tárolt program előnyei:

— A működés bármely pillanatában a tárolt program bármely része gyorsan hozzáférhető volt a gépet vezérlő áramkörök számára.
— Ki lehetett használni a számítógép nagy feldolgozási sebességét, mert szükség esetén a számítógép maga tudott programot váltani.
— Lehetőséget adott arra, hogy a program már lefutott részei változatlanul vagy akár módosítva aktivizálhatók legyenek.
— Lehetőség nyílt arra is, hogy a program bizonyos feltételek teljesülésétől függően különböző programrészeknél folytatódjon („elágazhatóság").

A Neumann-elv meghatározó szerepet játszik a számítástechnikában, mert elgondolása a ma használt számítógépekben is változatlanul valósul meg.

A Neumann-elv felhasználásával készült el 1952-re az EDVAC, de mégsem ez lett az első tárolt programmal működő gép, mert a MARK-I-en 1948-ban, az EDSAC-on 1949-ben már futtattak tárolt programot. A fejlődés felgyorsulását mutatja, hogy 1951-től már piacra is gyártottak számítógépeket az USA-ban.

D) A tranzisztorok megjelenése, integrált áramkörök, chip

A számítógépek fejlődésében a tranzisztorok megjelenése újabb korszak kezdetét jelentette. Az 1950-es évektől kezdődően a számítógépek fő alkotórészei a tranzisztorokból felépített logikai áramkörök lettek. Segítségükkel lecsökkent a számítógépek mérete (századrésznyire), sokkal kevesebb energiára volt szükség, mert kisebb volt a hőtermelés.

A tranzisztorok hosszú élettartama miatt megnőtt a számítógépek megbízhatósága, megsokszorozódott a műveletek sebessége.

Az 1960-as években már olyan számítógépeket is készítettek, amelyekben integrált áramkörök is voltak: ez a fejlődés újabb lépcsőjét jelentette. Egy körömnyi nagyságú szigetelőlemezkére gombostűfejnyi tranzisztorokat szerel-

tek, úgy, hogy a teljes áramköri egységek elfértek egy négyzetcentiméternyi területen.

Az integrált áramkörök nagyságának csökkenése és összeépítésük eredményezte 1971-től a mikroprocesszorok (chipek) megjelenését. Ezek a mikroprocesszorok lettek a számítógépek lelkei. A chipnek köszönhetjük a személyi számítógépeket, amelyek a felhasználóval közvetlen kapcsolatot tudnak teremteni, egyszerűek, ezért nagyon népszerűek, és az áruk is egyre csökken.

1. A személyi számítógépek csoportosítása

Teljesítőképességük, felhasználhatóságuk alapján a személyi számítógépeket két csoportra oszthatjuk.

— Hobbigépek — főként játékok, egyszerű műveletek, programok, oktatóprogramok futtatására alkalmasak. Olyan készülékekhez csatlakoztathatók, amelyek általában minden háztartásban megtalálhatók (tévé, magnetofon). Ilyen *például* a HT 1080 Z, ZX-81, ZX Spectrum, Commodore-64 számítógép.

— Professzionális számítógépek — alkalmasak vállalati, intézményi feladatok ellátására is. Jobban felszereltek: nagy operatív tárral rendelkeznek, van háttértárolójuk, nyomtatójuk, operációs rendszerük, és alkalmazói programcsomagok is segítik az alkalmazó munkáját. *Például ilyenek*: a Proper, az Apple, az IBM PC, valamint az utóbbival azonos felépítésű (kompatíbilis) gépek.

A Commodore személyi számítógépek sok ember számára hozzáférhetők voltak, ezért széles körben elterjedtek. Az ügyviteli feladatok gépesítésekor kiderültek a Commodore-ok korlátai, ezért sokan cserélték őket IBM gyártmányú vagy azzal kompatíbilis személyi számítógépekre.

2. Az IBM PC-k fejlődése, a felhasználási területek változása

A személyi számítógépek történelmében új korszakot jelentett az IBM által gyártott PC-k (Personal Computer — személyi számítógép) megjelenése, mert a korábbinál sokkal magasabb minőségi és teljesítménybeli követelményeket határoztak meg számukra. Ekkor már nem játék volt a számítógép, amelyen csupán játékprogramokat lehetett futtatni, egyszerűbb számításokat elvégezni, hanem komoly segédeszköze lett alkalmazójának. A felhasználás során betöltött szerepük alapján a PC-k lehetnek:

— *Játékgép* — a kezdeti időszakban csak azért lehetett ilyen gépeket venni, hogy olcsó árával kedvet csináljon a vásárláshoz. Csak mágneskazettás meghajtója volt. (Ma már nem gyártják.)

— *Ügyviteli mikroszámítógép* — egyszínű képernyőjén csak karaktereket jeleníthetünk meg, grafikus jeleket nem. Kifejezetten irodai alkalmazásra készült.
— *Grafikus számítógép* — a műszaki és építészeti tervezéskor, szemléltető ábrák megjelenítésekor sokat segít. Színesgrafikus monitora van.
— *Winchesteres* — lemezes számítógép amelynek legfontosabb tartozéka egy nagy tárkapacitású, gyors hozzáférésű, rögzített mágneslemezegység. A lemez óriási tárolókapacitása alkalmas arra, hogy tárolja kisebb vállalatok üzlettel kapcsolatos összes adatát vagy teljes kutatások adatbázisát és egyéb információk sokaságát. Ezeket az adatokat a felhasználás során közvetlenül a gép rendelkezésére bocsátja, így növeli a feldolgozás sebességét. Ez a változat egy önálló rendszernek, egy teljes értékű ügyviteli számítógépnek tekinthető, melynek teljesítménye megközelíti a nagy számítógépekét.

Olvasmány

A számítógépek történetét generációk szerint is szokták tárgyalni, ezért itt bemutatjuk, hogy az egyes számítógépek melyik generációhoz tartoznak:

0. *generáció — mechanikus és elektromechanikus (relés) számítógépek* (XVIII—XX. *sz.),*
1. *generáció — elektroncsöves számítógépek* (1946–58.),
2. *generáció — tranzisztoros számítógépek* (1958–65.),
3. *generáció — integrált áramkörök alkalmazása* (1965–72.),
4. *generáció — mikroprocesszoros számítógépek* (1970–),
5. *generáció — az emberi agy leképezésén alapulna.*

Kérdések és feladatok

1. Milyen segédeszközöket vettek igénybe a számolás megkönnyítésére a számolások kezdeti időszakában?
2. Milyen mechanikai számológépekről tanultunk, és mi volt a jellemzőjük?
3. Babbage-nek mely elképzelései szolgáltak alapul a mai számítógépekhez?
4. Milyen új elképzeléseket használt fel Zuse gépeinek elkészítéséhez?
5. Fogalmazzuk meg, hogy mi volt a tárolt program lényege!
6. Milyen előnyei voltak a tárolt programnak?
7. Miért jelentett nagy változást a tranzisztorok megjelenése?
8. Mit nevezünk integrált áramkörnek és mit mikroprocesszornak?

II. Zsebszámológépek

A) A számológép és a számítógép kapcsolata

A történeti áttekintésben láthattuk, hogyan törekedtek a tudósok a számolás gépesítésére. Igyekeztek olyan berendezést építeni, amely a munkaigényes számításokat gyorsan, pontosan elvégezte.

A számítógép munkája során több műveletet képes automatikusan elvégezni, mert tárolja a műveletek elvégzéséhez szükséges adatokat, a műveleti sorrendet leíró utasítások sorozatát. A számítógép nemcsak számolásra alkalmas, hanem feltételek vizsgálatára, logikai feladatok megoldására, döntések meghozatalára is. Mindezek nagyon sokrétű felhasználását teszik lehetővé.

A számológép csak számítási feladatok megoldására alkalmas.

A mikroelektronikai ipar kialakulása lehetővé tette a zsebszámológépek kifejlesztését. A legegyszerűbbek csak a négy alapművelet elvégzésére, esetleg százalékszámításra alkalmas egyszerű kalkulátorok. A tudományosnak nevezett gépekkel függvények értékét is számolhatjuk. A legfejlettebbeket programozhatjuk valamilyen nyelven (általában BASIC nyelven), de mégis minőségi különbséget tudunk tenni számológép és számítógép között alkalmazhatósági területük szerint.

B) A zsebszámológépek általános jellemzői

1. Számok beírása és kijelzése

A felhasználó a zsebszámológépbe a számokat és a műveleteket billentyűzettel viheti be. Ezt a billentyűzetet úgy alakították ki, hogyha bal kézzel kezeljük, közben a kapott eredményeket jobb kézzel le tudjuk írni a kijelzőről.

A zsebszámológép bekapcsolása után a kijelzőn megjelenik a 0 szám. Ezután megnyomva valamelyik számbillentyűt, ezt a számot a kijelzőn azonnal láthatjuk.

Többjegyű számok beírásakor a nagyobb helyiértékű számtól kezdve a kisebb helyiértékű felé haladva írjuk be a számokat ugyanúgy, mintha papírra írnánk.

Tizedestört esetén az utolsó egész szám után kell a tizedespont billentyűjét lenyomni. Vigyázat, nem tizedesvessző, hanem tizedespont jelöli a tizedest!

Egyszerre csak nyolc–tizenkét számjegyet fogad el tőlünk a gép.

Például: írjuk be a 357,1 *számot!*

Leütés	Billentyű	Kijelző
1.	3	3
2.	5	35
3.	7	357
4.	·	357.
5.	1	357.1

Ha negatív számot akarunk beírni, akkor a számjegyek beírása után megnyomjuk a +/– előjelváltó billentyűt. Mivel ez a művelet –1-gyel való szorzást jelent, még egyszer lenyomva ismét magát a számot kapjuk.

Folytassuk előző beírásunkat, alkalmazzuk az előjelváltó billentyűt!

Leütés	Billentyű	Kijelző
6.	+/–	–357.1
7.	+/–	357.1

Ha a tizedestört egészrésze 0, akkor a beírást a tizedespont billentyűzésével kezdhetjük, nem kell 0-t írnunk. A tizedespont leütése viszont természetesen elkerülhetetlen, mert elhagyása esetén nem tudnánk olyan számot beírni, melyben az első számjegy 0.

Például: 0,0001 *nem azonos az* 1*-gyel,* 0,0408 *nem azonos a* 408*-cal.*

Új feladat megoldása előtt a kijelző tartalmát törölnünk kell a C, az AC vagy CLR (Clear = töröl) billentyűvel. Ilyenkor a bekapcsolási állapotot kapjuk vissza, tehát a kijelzőn megjelenik a 0. Ha a kapott eredményből akarunk továbblépni, akkor hagyjuk meg a kijelzőn az eredményt.

2. Alapműveletek

A legegyszerűbb számológépen az aritmetikai műveletek elvégzésére öt billentyű áll rendelkezésünkre:

\+ az *összeadás,* – a *kivonás,* × a *szorzás,* : az *osztás,* = az *egyenlőség* jele a billentyűkön.

A pozitív, egész kitevős hatványok számolásakor szorzások egymásutánját végezzük el, ha nincs számológépünkön y^x billentyű. ($5^4 = 5 \times 5 \times 5 \times 5$)

Vannak olyan számológépek, ahol elegendő egyszer beírni a hatvány alapját, és a szorzás jele után csak az egyenlőségjelet kell eggyel kevesebbszer leütni, mint amennyi a hatványkitevő.

Például: 5^4 *billentyűzése lehet:* 5× = = = *és a kijelzőn megjelenik a* 625.
Ebben az esetben gépünk konstansként tárolja az első tényezőt.
Találhatunk a gépen beépített konstansot is. *Például:* Pí = 3,14159.
Azoknál — az igen ritka — gépeknél, ahol nincs egyenlőségjel, találunk egy ENTER feliratú billentyűt.

Ilyen esetben például: a 8+7 *és a* 8×4 *kiszámolása a következő módon történik:*

Leütés	Billentyű	Kijelző	Billentyű	Kijelző
1.	8	8	8	8
2.	ENTER	8	ENTER	8
3.	7	7	4	4
4.	+	15	×	32
Logikája szerint:	8 és 7 összege 15,		8 és 4 szorzata 32.	

Az egyenlőségjelet tartalmazó gépeknél az egyenlőség billentyűzésével zárhatjuk le a nyitott, befejezetlen műveleteket. Hatására megjelenik a kijelzőn az eredmény, és a gép készen áll a soron következő számítások elvégzésére.

Például: az előző feladatunk beírása:

Leütés	Billentyű	Kijelző	Billentyű	Kijelző
1.	8	8	8	8
2.	+	8	×	8
3.	7	7	4	4
4.	=	15	=	32
Logikája szerint:	8 meg 7 egyenlő 15-tel,		8-szor 4 egyenlő 32-vel.	

A beírást folytatjuk. Újabb egyenlőségjel hatására a kijelzőn legutoljára látott szám és a műveleti jel után beírt szám között történik a műveletvégzés.
Például hagyjuk a kijelzőn az eredményt, és folytassuk a műveleteket 8 × 4 + 6 – 2 *beírásával!*

Leütés	Billentyű	Kijelző
5.	+	32
6.	6	6
7.	–	38
8.	2	2
9.	=	36

Az ilyen műveletsort láncműveletnek szoktuk nevezni. Vegyük észre, hogy a +6 után nem ütöttünk egyenlőségjelet, mert a beütött új műveleti jel, a – billentyűzésére is elvégzi gépünk a kijelölt műveletet.

Alkalmazzuk a következő többtagú kifejezés kiszámolására a láncműveletet!

$$9\times3 + 5\times4^3 : 6 = 80.333333$$

Leütés	Billentyű	Kijelző
1.	9	9
2.	×	9
3.	3	3
4.	+	27
5.	5	5
6.	×	5
7.	4	4
8.	×	20
9.	4	4
10.	×	80
11.	4	4
12.	:	320
13.	6	6
14.	=	80.333333

Ha billentyűzés után nem ezt az eredményt kapjuk a kijelzőn, akkor a gépünk nem veszi figyelembe a műveletek elvégzésének szabályos sorrendjét, azaz, hogy először a szorzást, osztást, majd az összeadást, kivonást végezze el balról jobbra haladva. Ilyen esetekben vagy akkor, ha el akarunk térni ettől a sorrendtől, sok gépen zárójeleket alkalmazhatunk.

Ha számológépünkön nincs zárójel, akkor külön-külön ki kell számolnunk a zárójelben lévő kifejezések értékét, ezután papírra leírni vagy memóriába vinni, majd azokkal kell a további műveleteket végezni újra beírással, illetve a memóriából történő előhívással.

A zárójelek lehetővé teszik a számok csoportosítását, a műveleti sorrend kijelölését.

Számítsuk ki $(3\times(-2))-(9:2)$, *valamint* $(8/5+4)\times(2-5/4)$ *értékét!*

Beírás	Eredmény
3×2 +/– – (9:2) =	–10.5
(8:5 + 4) × (2–(5:4)) =	4.2

Ha a tört számlálója vagy nevezője, vagy mindkettő több tagból áll, akkor a számlálót és a nevezőt is külön zárójelbe kell tenni.

Számoljuk ki $[(16\cdot5)+7\cdot(-4)]:[(6:5-1)\cdot3]$ *értékét!*

Beírás	Eredmény
((16×5)+(7×4+/–)) : ((6:5–1)×3) =	86.666667

3. El nem végezhető műveletek

Előfordulhat, hogy olyan művelet elvégzésére adunk utasítást gépünknek, amely értelmetlen. Ebben az esetben hibajelzést kapunk. *Például: a kijelzőn levő szám villog vagy megjelenik egy E és* 0 *stb.*

Ilyen műveletek lehetnek:

— Negatív szám négyzetgyökének vagy logaritmusának számolása.
— Egy szám nullával való osztása.
— Egymás után leütött több műveleti billentyű esetén az utolsót veszi figyelembe vagy nem számol a gép.
— A megengedettnél több zárójel megnyitása.
— Szám után + vagy – jel beütése után = vagy) billentyű lenyomása.

4. A billentyűk másodlagos funkciója

A számológépek többségének olyan billentyűzete van, amelyen a billentyűknek nem csak egy funkciójuk van. Az első funkciót a billentyűn, a másodikat a billentyű fölött láthatjuk. A másodlagos funkciót úgy tudjuk alkalmazni, ha először lenyomjuk a 2nd vagy inv feliratú billentyűt, majd utána azt, amely fölött van a kiválasztott másodlagos funkció.

Például: a gombon x^2 *van, fölötte* x^3
Számoljuk ki 32^2 és 32^3 *értékét!*
Billentyűzés: 32 x^2 = és 32 2nd x^3 =

5. Kijelzés

A zsebszámológépeknél az eredmény különböző formában jelenhet meg a kijelzőn. Gépünk kijelzésének módját rögtön megállapíthatjuk, ha elvégezzük a következő műveletet: 2.48 × 3.652 .

A kijelzőn megjelenhet:

9.05696 — a gép a számolt értéket írja ki maximális számú értékes tizedesjeggyel (repülő tizedesponttal számol).
9.05 — a gép két fix tizedesjeggyel számol, csonkítja a kapott eredményt, elhagyja az utolsó tizedesjegyeket.
9.06 — a gép két fix tizedesjeggyel számol, kerekíti az eredményt.
(A kijelzett tizedesjegyek számát változtathatjuk egyes gépeken.)

Nézzük meg, hogy kerekít vagy csonkít-e a gépünk!

Írjuk be: 7 : 6.

Ha az utolsó számjegy 6, akkor csonkít, ha az utolsó jegy 7, akkor kerekít gépünk.

A számológépünk belső számítási módszeréből adódóan hiba lehet az utolsó tizedesjegyben. Előfordulhat, hogy nem érvényesül a kommutativitás szabálya, azaz eltérő műveleti sorrend esetén nem ugyanazt az eredményt kapjuk.

Például: 0,27 + 10000000 + 0,46 = 10000000,
0,27 + 0,46 + 10000000 = 10000001 *lehet a kijelzőn.*

A kerekítési hibát csökkentik azok a számológépek, melyeknél a kijelzett 8 helyiérték mellett a belső számítás több, például 11 helyiértékkel történik. Az ilyen, úgynevezett védő helyiértéknek köszönhetjük, hogy a gép kerekíti az eredményeket.

Például: 2/3 × 3 = 2 *jelenik meg az* 1.9999999 *helyett.*

Egyes gépeknél, ha számolás közben olyan értéket kapunk eredményül, amelyet nem tud kijelezni gépünk, akkor a kijelzőn a szám normálalakját olvashatjuk. Ez teszi lehetővé, hogy +/– 1 × 10^{-99} és a +/– 9,9999999 × 10^{99} közé eső számtartományban számoljanak ezek a számológépek.

6. Szögfüggvények

Trigonometrikus függvények helyettesítési értékét úgy kapjuk, hogy begépeljük a szögértéket, majd lenyomjuk a sin, cos vagy a tan feliratú billentyűt.

Nagyon sokszor más eredményt kapunk, mint amit a függvénytáblázatból kikeresünk. Ennek az lehet az oka, hogy a fok–perc–másodperc értékekkel megadott szöget nem alakítottuk át tizedesszám formájúra, vagy visszakereséskor tizedesszám-alakot kapunk.

Például: 63° 54′ = 63,9°.

Ha cos alfa = 0.6665, *akkor* alfa=48,2° = 48° 12′.

Ha gépünkön van olyan gomb, amelynek segítségével elvégezhetjük az átalakítást, akkor ennek használatát tanuljuk meg a kezelési útmutatóból.Ha nincs ilyen billentyű, akkor matematikai ismereteink segítségével végezzük el az átalakítást:

1 perc = 1/60 fok,
alfa perc = alfa/60 fok,
alfa másodperc = alfa/3600 fok.

Például: 45 perc = 45/60 fok = 0,75 fok,
42 másodperc = 42/3600 fok = 0,01166 fok.

Visszaalakításkor a tizedesszám értékét szorozzuk 60-nal:

1 fok = 60 perc,
alfa fok = alfa × 60 perc.

Például: 0,56 fok = 0,56 × 60 perc = 33,6 perc.

C) A Texas Instruments számológépei

Eddig a számológépekről általánosságban beszéltünk, olyan tulajdonságaikkal foglalkoztunk, amelyek a legtöbb gépen igazak.

Ismerkedjünk meg kicsit részletesebben a TI-36X SOLAR elnevezésű számológéppel, amely az algebrai számítások mellett statisztikai és kalkulációs műveletek elvégzésére is alkalmas. Ez a számológép egy olyan TEXAS gépcsaládnak a tagja, amelyben megtalálhatjuk a legegyszerűbb kalkulátort (a TI-106-ot), az elemi függvények és a zárójelek alkalmazását lehetővé tevő TI-30-at és egyéb tudományos számológépeket is.

Ez a számológép nem elemmel vagy hálózatról működik, hanem áramforrása olyan szuperérzékeny napelem, amely már egy gyertya fényének hatására is működteti a számológépet.

a) A számológép be- és kikapcsolása

A számológépet nem kell be- és kikapcsolnunk, mert automatikusan bekapcsol, ha a napelemét elegendő fény éri. A kijelzőn megjelenik a 0 szám és a DEG üzemmód jelzése. A számolás megkezdése előtt nyomjuk le az AC/ON billentyűt, hogy mindent töröljünk, ne zavarja meg számolásunkat egy véletlenül a tárban maradt érték.

Ha nem kap fényt számológépünk, akkor kikapcsol. Ezt elérhetjük, ha a számológép fedelét ráhelyezzük a billentyűzetre.

b) Az üzemmódok kijelzése

A számológép kijelzőjén a számokon kívül olvashatunk olyan adatokat is, melyek megmutatják, hogy milyen üzemmódban dolgozunk.

A kijelzések a különböző üzemmódokban a következők lehetnek:

M −5 5 5 5 5 5 5 5 5 5 $^{-55}$

2ND 3RD HYP BIN OCT HEX STAT DEG RAD GRADx r ( )

c) A rövidítések jelentése

M	—	működik a tároló
—	—	negatív előjel, helye a számtól függ
2ND	—	a billentyű második funkciója
3RD	—	a harmadik funkció
HYP	—	hiperbolikus függvény
BIN	—	bináris üzemmód
OCT	—	oktális üzemmód
x	—	polárkoordinátákat váltunk át derékszögű koordinátákká
( )	—	zárójeleket alkalmazunk
HEX	—	hexadecimális üzemmód
STAT	—	statisztikus üzem
DEG	—	fokokban számolja a szöget
RAD	—	radiánban számolja a szöget
GRAD		újfokban adja meg a szöget
r	—	derékszögű koordinátákat váltunk át polárkoordinátákká

Olvasmány

Derékszögű koordináta-rendszerben a koordináták (x, y) egyértelműen megadják egy pont helyét. Polárkoordináta-rendszerben ugyanezt a pontot a nullponttól mért r egységnyi távolsággal és a pozitív tengellyel bezárt szöggel adhatjuk meg.

d) Az adatok bevitele

Az adatok bevitele az eddig tárgyaltaknak megfelelően történik.

— A kijelzőn maximum 10 helyiértékes szám jelenik meg, de gépünk 12 helyiértékkel tud dolgozni. Ezek a 12-jegyű számok 2 szám összegeként hozhatók létre.

Például: 12345678.52887 = 12345678 + 0.52887

Az összeadandók beírása után 12345678.53 lesz a kijelzőn, de a gép 12345678.5289-del számol.

— Egy bevitt vagy kiszámolt szám előjelét a +/– billentyű lenyomásával változtathatjuk meg.

— A kitevő előjelét is megváltoztathatjuk, ha az y^x gomb lenyomása után a +/– -t is lenyomjuk.

— Beépített konstansként megtaláljuk π értékét, melyet a 3rd és a π gombok lenyomásával vihetünk be 12-helyiértékes pontossággal.

e) A törlések lehetőségei

— Ha a kijelzőn lévő számot, rosszul bevitt adatot vagy műveletet akarjuk törölni, akkor a CE/C gombot nyomjuk le!
— Ha mindent törölni szeretnénk, akkor az AC/ON billentyűt nyomjuk le! Ilyenkor az előzőkön kívül törlődik a tárak tartalma, a statisztikai regiszter, az üzemmód visszaáll DEG-re, és változik a kijelzési formátum is.

f) A beviteli hibák javítása

Legegyszerűbb módszer, ha lenyomjuk az AC/ON gombot, így töröljük az összes számítást, így a hibákat is, és újra kezdhetjük a beírást.

A hibásan beírt számot törölhetjük úgy is, ha utána lenyomjuk a CE/C gombot, és a jó szám beírása után nyomjuk csak le az operátorbillentyűt.

Például: 2+4 *beírásakor, ha hibát vétünk,*
2+3 CE/C 4 = *hatására a kijelzőn* 6 *lesz.*

g) A kijelzési formátum

Túl nagy vagy túl kicsi számok bevitelekor alkalmaznunk kell az EE kitevőbeviteli gombot. Segítségével bevihetünk egy maximum 10 helyiértékes mantisszával és $-99 \leq n \leq 99$ kitevővel rendelkező számot.

Például: $4{,}326 \cdot 10^{4}$, beírása: 4.326 EE 4.
$-3{,}25 \cdot 10^{-6}$, beírása: 3.25 +/– EE 6 +/–.

Követelményként szerepelhet, hogy a kijelzőn a tizedespont után mindig meghatározott számú helyiérték íródjon ki. Nyomjuk be a 2nd, a Fix és az n billentyűket, melyekkel jelezzük a számológépnek, hogy kerekítsen *n* számú tizedesjegyre. (Ha a törlést a kijelzőn CE/C-vel végezzük, akkor a következő eredményt is adott, *n* számú tizedessel kapjuk.)

h) Műveleti sorrend

Számológépünk ismeri a matematikában használt műveleti sorrendet vagy prioritást, amelytől zárójelezéssel eltérhetünk. Egyidejűleg maximum 15 zárójelet ágyazhatunk egymásba, és legfeljebb 6 lezáratlan műveletet hagyhatunk.

Például: 6 : 4^2 × 5+3 × cos 60=; *billentyűzése:* 6 : $4x^2$ × 5 + 3 × 60 cos =;
5×6+4×3=42; (5×6+4)×3=102; 5×(6+4)×3=150; 5×(6+4×3) = 90.

Kijelzéskor gépünk felfelé vagy lefelé kerekítést végez az első elhagyott számjegy értékétől függően.

Például: 1 : 3 = 0.333333333; 2 : 3 = 0.666666667.

Az *algebrai függvények* értékét a matematikában megismert módon használja számológépünk.

A trigonometrikus függvények alkalmazása előtt ki kell választanunk, hogy milyen szögmérési egységet használunk, meg kell választanunk a számolási üzemmódot. Bekapcsoláskor a DEG, azaz fok üzemmódban dolgozhatunk, amelyen a 3rd és a DRG gombokkal változtathatunk.

i) Szögek átváltása

30°-ot váltsuk át más egységre! A számítógép DEG üzemmódban van.

A 30, a 3rd és a DRG beütése után 0.523598776 lesz a kijelzőn, radiánban adja a szöget;
újból benyomva a 3rd és a DRG gombokat, 33.33333333, újfokban adja az eredményt;
a 3rd és a DRG ismételt lenyomása után 30, azaz újból a fok értéke lesz a kijelzőn.

A kijelzett szögek trigonometrikus értékét a megfelelő gombok lenyomásával kaphatjuk meg.

Például: cos 56°38′, *beütése* 38 : 60 + 56 cos,
a kijelzőn 0.559192903 *jelenik meg.*

Megkereshetjük a kijelzőn látható trigonometrikus értékhez a megfelelő szöget:

Például: 0.12589 *után a* 3rd *és a* $\cos^{-1}$ *leütésekor megjelenik a* 82.76784495, *a szög értéke.*

j) Százalékjel

A százalékjel segítségével kiszámolhatjuk a felárakat, levonásokat és a százalékokat.

Aritmetikai műveletek után lenyomva a % jelet, az eredmény:
+ százalékláb 3rd % = hatása: n % hozzáadása x-hez,
– százalékláb 3rd % = hatása: n % kivonása x-ből,
× százalékláb 3rd % = hatása: a szám szorzása n %-kal,
: százalékláb 3rd % = hatása: a szám osztása n %-kal.

Példák: 1. 56 38 %-ának a kiszámításakor a billentyűzés
56 × 38 3rd % =; a kijelzőn megjelenik 21.28.
2. 1200 Ft-os áru ára 30 %-kal csökken. Az új ár:
1200 – 30 3rd % = kijelzőn megjelenik: 840.

k) Számítások különböző számrendszerekben

Az alapszámot a DEC (tízes), a HEX (tizenhatos), az OCT (nyolcas), a BIN (kettes) és a 3rd gombok segítségével állíthatjuk be. Ügyeljünk arra, hogy a számokat az üzemmódnak megfelelően írjuk be!

Bináris üzemmód:	a 3rd és a BIN után 0 és 1 számjegyek.
Oktális üzem:	a 3rd és az OCT után 0-7 számjegyek.
Hexadecimális eset:	3rd és HEX után 0-9 számjegyek és AF betűbillentyűk.

Ezekkel a számokkal csak alapműveletet végezhetünk.

Példa: *írjuk át 428-at különböző számrendszerekbe!*
$428_{10} = 110101100_2 = 654_8 = 1AC_{16}$,
428 *után* 3rd *és* BIN *után* 110101100,
3rd *és* OCT *után* 654,
3rd *és* HEX *után* 1AC,
3rd *és* DEC *után* 428 *jelenik meg.*

l) A tárolók felhasználása

A TI-36X SOLAR számológépnek 3 adattárolója van (1, 2, 3). A tár akkor is megőrzi az adatokat, ha kikapcsoljuk a gépet. A tároló használata a futó számításokat nem befolyásolja, de a tárműveletek bármikor elvégezhetők, ha megadjuk a felhasználandó tár sorszámát a megfelelő gombok lenyomása után.

STO n —	a tárológomb lenyomása után a megadott tárolóban tárolódik a kijelzett szám, a korábban tárolt érték automatikusan törlődik, de a kijelző változatlan marad.
RCL n —	a tárolóhívás gomb lenyomása után a megadott tárolóban tárolt adatok megjelennek a kijelzőn, de a tároló tartalma változatlan marad.
3rd és EXC n	gombok lenyomása után gépünk az adott tároló tartalmát felcseréli a kijelzőn lévővel. A kijelzett szám az adott tárolóba, a tárolt érték pedig a kijelzőre kerül.

2nd SUM n gomb lenyomása után a kijelzőn lévő érték hozzáadódik az adott tárban lévő értékhez. Több számítás eredményének összegezését végezhetjük el segítségével.

Figyeljünk arra, hogy összegzés előtt a tár üres legyen, hogy ne adjunk össze nemkívánatos számokat!

Számológépünk más lehetőséget is kínál számunkra, de ezek nem tartoznak szorosan a matematika tananyaghoz, ezért a kezelési utasításból javasoljuk megismerésüket.

Például: a logaritmusszámolást, a hiperbolikus függvényeket, a permutációkat, a kombinációkat, a statisztikai funkciókat, a polárkoordinátákat és a derékszögű koordinátákat, a véletlenszámokat, a komplex számokat.

Példák

Írjuk fel a következő aritmetikai kifejezéseket olyan formában, hogy zsebszámológépünkkel helyesen ki tudjuk számolni az értéküket! Végezzük el a számolást is!

Kifejezés	Beírandó sor	Kijelző
$\frac{5}{6} - 4$	5 : 6 – 4 =	–3.1666667
$8 - \frac{3}{5}$	8 – (3 : 5) =	7.4
$\frac{5}{2} \cdot 12^2$	5 : 2 × 12 x^2 =	360
$26 - 3 \cdot 4$	26 – (3 × 4) =	14
$6 \cdot 8 + 52 - 18$	6 × 8 + 52 – 18 =	82
$\frac{36 \cdot 44}{126 - 18}$	36 × 44 : (126 – 18) =	14.666667
$\frac{57 - 23}{14 + 36}$	(57 – 23) : (14 + 36) =	0.68
$(3 + 7) : \left(4 + \frac{3}{4}\right)^2$	(3 + 7) : (4 + (3 : 4)) x^2 = vagy (3 + 7) : (((4 + (3 : 4)) × (4 + (3 : 4))) =	0.4432133
$12\frac{5}{6} = 12 + \frac{5}{6}$	5 : 6 + 12	12.833333

A törteket általában tizedestörtté kell alakítani, mert a legtöbb gép csak azzal tud számolni.

Kérdések és feladatok

1. Ellenőrizzük a számológépünket, hogy számolás közben figyel-e a műveletek prioritására (azaz elvégzésük sorrendjére)!

Számoljuk ki $5 + 3 \times 4$ értékét!
Ha $5 + 3 \times 4 = 17$, akkor figyelembe veszi,
ha $5 + 3 \times 4 = 32$, akkor nem figyeli a prioritást;
zárójeleznünk kell vagy a kommutativitást figyelembe véve felcserélni a tagokat:
$5 + (3 \times 4) =$ ugyanazt az eredményt adja, mint $3 \times 4 + 5 =$.

2. Hogyan írjuk be a következő számokat?

–238; 0,325; 57,36; –83,12.

3. Számítsuk ki a kifejezések értékét!

$38{,}56 \times 123{,}14$; $77{,}7 : 143{,}26$; $853{,}2^2$;

$\frac{128{,}56 \times 14{,}36}{12{,}5}$; $\frac{543{,}27 \times 181{,}4}{36{,}25 \times 196}$.

4. Végezzük el a következő műveleteket!

	Eredmény
a) $2 \times 12 + 34 : 2 - 63 - 7 \times 2 + 5 \times 6 + 3 \times 9 - 2 \times 11 =$	–1
b) $28 + 56 : 8 - 2 + 12 \times 6 : 4 - 52 : 2 =$	25
c) $(3 + 2 \times 5) \times 3 - (2 + 8 : 4) : 2 + 3 \times 2 =$	43
d) $(9 : 3 - 1) \times 4 + 5 \times (6 \times 20 - 12) : 3 =$	188
e) $12 \times 3 : 4 - 30 : (2 \times 5) + 4 =$	10
f) $3 + (8 - 2) : 3 \times 4 =$	11
g) $(4 - 3) \times 5 - (8 + 4) : 6 : 2 =$	4
h) $(5 \times 4) : 10 + (9 : 3) \times 7 \times 2 =$	44

5. Végezzük el a következő műveleteket!

a) $3{,}5 + 22{,}06 + 184{,}3 + 0{,}026$;

b) $186{,}25 - 9{,}128$;

c) $5204 - 181{,}76$;

d) $12\frac{5}{6} + (4\frac{1}{4} - 1\frac{1}{2})$;

e) $15\frac{3}{8} - (2\frac{3}{4} + 1\frac{4}{5})$;

f) $16\frac{1}{2} - (6\frac{1}{3} - 2\frac{2}{7})$.

6. Számolja ki a kifejezések értékét!

a) $2\frac{5}{6} + 1\frac{1}{3} - \frac{8}{9} \times 2 + 4\frac{1}{2}$;

b) $9\frac{1}{2} : 0{,}5 + 3\frac{1}{3} - 2\frac{4}{5} \times 1{,}2$;

c) $6{,}5 - 3{,}1 \times \frac{1}{5} + 4\frac{2}{3} : 3{,}2 - 0{,}02$;

d) $8\frac{3}{4} + 2{,}1 \times 4{,}02 : 1\frac{1}{2} - 3{,}8 : \frac{4}{5} \times 2{,}1$.

7. Keressen a matematikakönyvben az előzőkhöz hasonló feladatokat, és számolja ki az eredményüket papíron és zsebszámológéppel is!

III. A számítógép felépítése és működtetése

A) A számítógép

Ha a számítógépet segítőtársként, barátként szeretnénk működtetni, akkor nem elég a nyelvét értenünk, kicsit a belsejét, a „lelkét", működésének elvét is ismernünk kell. Kérdés, hogy egyáltalán mi is a számítógép? A legegyszerűbben megfogalmazva azt mondhatjuk, hogy a számítógép egy információátalakító berendezés *(1. ábra)*.

1. ábra

Információnak olyan ismereteket, híreket, tényeket, közléseket tekintünk, amelyek számunkra valamilyen szempontból fontosak. Amikor az információt valamilyen módon megpróbáljuk rögzíteni, akkor adatról beszélünk. A két fogalom összemosódik, nem szoktuk őket szétválasztani.

1. A számítógépek felépítése

A számítógépet két részre szokás felosztani. Egyik a hardver (hardware) — a számítógép műszakilag kialakított teste, amely állandó, a másik a szoftver (software) — a betáplált program, mely cserélhető. Ezek a részek nem különülnek el élesen egymástól, mert egymásra épülnek, és egyik sem működik a másik nélkül.

a) Szoftver

Napjainkban egyre nagyobb szerephez jutnak a kész programok, amelyeket programozók készítenek, a felhasználók csak használják ezeket feladatuk gyors, pontos elvégzéséhez vagy a számítógép működtetéséhez.

A programok rendeltetésük szerint lehetnek:

— Felhasználói programok: ezek egy konkrét feladat megoldására készültek, tehát célprogramok (játékprogramok, nyilvántartások, órarendkészítés, szövegszerkesztés...).

— Rendszerprogramok: ezek teszik lehetővé a számítógép működését, a különböző egységek megfelelő együttműködését, a felhasználó és a gép közti párbeszédet, a perifériaműveleteket...
A rendszerprogramok összessége az operációs rendszer, amely több programból áll, és így biztosítja a hardver működéséhez szükséges feltételeket, segíti más programok futását.

— A fejlesztői programok segítik a programok elkészítését, fejlesztését és tesztelését. Ezeket programozók használják.

b) **Hardver**

A számítógépek hardverét alapgépre, perifériákra és az ezeket összekapcsoló illesztőkre szoktuk felosztani. Az alapgép legfontosabb része a központi egység, melynek munkáját a memória segíti. A perifériák az alapgép környezetében helyezkednek el.

Az adatbevitelt a billentyűzet, az információk megjelenítését a monitor, az adattárolást a mágneslemezegység végzi. Az adatokat nyomtatóval írathatjuk papírra.

Az alapgép és a perifériák valamilyen konkrét kiépítettségét szoktuk konfigurációnak nevezni. Ha a számítógép működéséhez elengedhetetlenül szükséges perifériákat kapcsoljuk az alapgéphez, akkor alapkonfigurációról vagy minimális konfigurációról beszélünk.

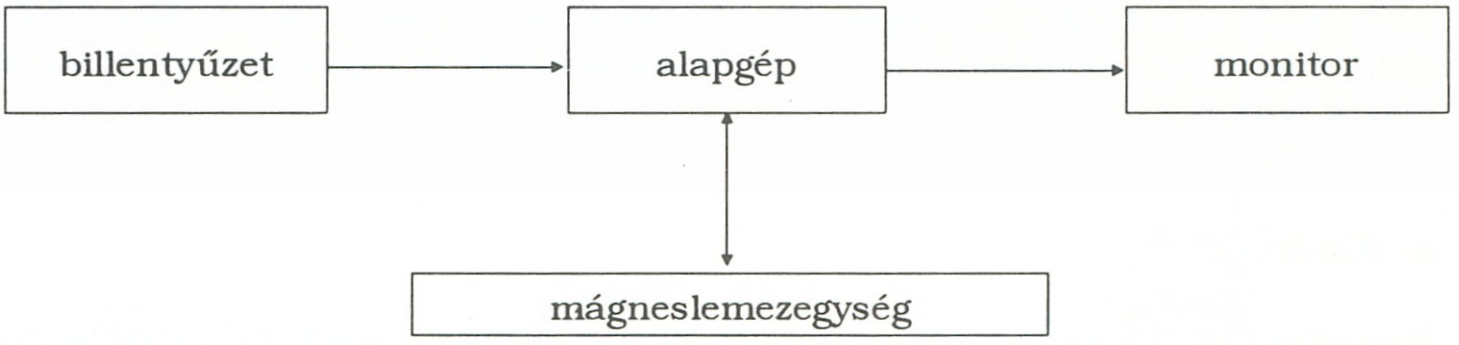
A számítógépek alapkonfigurációja a következő *(2. ábra)*:

— alapgép, a központi egység a vele egybeépített operatív memóriával,

— legalább egy mágneslemezegység — lehet hajlékony — vagy fixlemezes,

— billentyűzet — elsődleges input,

— monitor — elsődleges output.



2. ábra

2. A hardver részei és azok jellemzői

a) Központi feldolgozó egység (CPU)

A számítógép agya a központi feldolgozó egység (CPU = Central Processzor Unit) vagy a processzor. Magában foglalja a vezérlőt, amely összehangolja a számítógép munkáját és a végrehajtót, amely elvégzi az aritmetikai, logikai műveleteket. Feladata, hogy a memóriában tárolt információk alapján a gép állapotán változtasson, módosítsa a memóriában tárolt információ tartalmát. A CPU irányítja a számítógép munkájának menetét, és a számítógép egységei között biztosítja a kapcsolatot.

A processzor munkáját a memóriában tárolt program határozza meg. (Értelmezi a tárból kiolvasott utasításokat, majd ezek alapján vezérlőjeleket állít elő a számítógép egységeinek működéséhez. Fogadja a többi egység jelzéseit és kezelőparancsait.)

b) Operatív memória (OM)

A központi egységgel szoros kapcsolatban van az operatív memória, amely az információk átmeneti tárolására szolgál. Ezek az információk lehetnek:

— adatok, amelyeket a számítógép működése során felhasznál,
— program, amelyben leírtuk, hogy hogyan működjön a számítógép.

Ha egy program lefut, akkor újabbat tölthetünk a helyére, és a felesleges adatokat is újabbakra cserélhetjük.

A memóriának vannak olyan részei, amelyhez nem nyúlhatunk. A ROM (Read Only Memory) csak olvasható memória, mert a benne tárolt információt csak felhasználhatjuk, meg nem változtathatjuk. Ez azoknak a programoknak az állandó tárolására szolgál, amelyeket már a gyártás során beépítettek a számítógépbe, és ezt nem lenne célszerű megváltoztatni, mert gyakran szükség van rájuk, és nélkülük nem működne a számítógép (*például*: BASIC értelmező a korábbi gépek esetén).

A normál vagy véletlen elérésű memória a RAM (Random Access Memory), amely írható és olvasható is. Ide kerül az a program, amelyet éppen futtatni akarunk és a felhasználandó adatok is. A számítógép kikapcsolásakor nullázódik ez a memóriarész és a tartalma elvész. A szükséges adatokat tárolással őrizhetjük meg az újabb felhasználásig.

c) **Perifériák**

Azokat az egyéb eszközöket, amelyek segítségével a számítógép kapcsolatot tart a felhasználóval, perifériáknak nevezzük.

A perifériák a számítógéprendszernek azok az egységei, amelyek a külső adatforgalom lebonyolítását segítik.

— A beviteli egység feladata a számítógép működéséhez szükséges programok és adatok bevitele. Leggyakrabban használt egységek: billentyűzet, fényceruza, egér, botkormány, MODEM, képdigitalizáló.

— A kiviteli egység feladata a feldolgozás során kapott eredményeknek a rendszerből való kihozatala vagy a felhasználó számára olvasható formájú kiírása, megjelenítése, megőrzése. A leggyakrabban alkalmazott egységek: monitor, nyomtató, rajzgép.

— A háttértárak feladata az adatok, programok tárolása, és az esetleges feldolgozás céljára készenlétben tartása. Leggyakrabban a mágneslemezes és a mágnesszalagos tárolókat alkalmazzuk.

Munkájukat a központi egység irányítja.

3. A leggyakoribb perifériák bemutatása

a) **Beviteli egységek**

— *Billentyűzet* (klaviatúra, tasztatúra, keyboard)
A személyi számítógépek beviteli egységei közül elsődleges a billentyűzet. Legtöbbször innen kapja a számítógép az adatokat, mert a billentyűzeten keresztül kerülünk közvetlen kapcsolatba gépünkkel. Láthatjuk, hogy a hagyományos írógép-billentyűzeten (betűk, számok, írásjelek) kívül speciális billentyűket is találunk a klaviatúrán. Érdemes megismerkednünk ezekkel a gombokkal, mert segítenek a beírás javításában, törlésében, a program megállításában. Kellő gyakorlás után gyorsíthatjuk munkánkat. Sok bosszúságot megelőzhetünk, ha nem feledkezünk meg arról, hogy a billentyűzeten megtalálható az 1-es és a 0 szám, nem kell az I betűt és az O betűt használni, mint a régi írógépeken.

— *Fényceruza* (lightpen)
A fényceruza hegyében egy fényérzékeny elem van. Ezt a hegyet kell a képernyőn ahhoz a menüponthoz érinteni, amelyet használni akarunk, vagy a képernyőn lévő rajznak ahhoz a részéhez, amelyet módosítani szeretnénk. Napjainkban főként vonalkódok leolvasására alkalmazzák.

— *Egér* (mouse)
Az egér egy alul nyitott doboz, melynek alján egy forgatható golyó van elhelyezve. Az asztalon mozgatva elfordul a golyó, és a képernyőn ennek megfelelő irányban elmozdul egy speciális jel (*például*: kereszt vagy nyíl). Menü választásakor ezt a jelet kell a megfelelő helyre állítani, majd meg kell nyomni az egéren lévő bal oldali gombot kétszer egymás után, hogy megtörténjen a megfelelő pont kiválasztása.

— *Botkormány* (joystick)
A botkormánynak van egy rúdja, amelynek mozgatásával mozgathatunk a monitoron egy pontot valamilyen irányban. Elsősorban játékprogramok használatakor találkozunk vele.

— *MODEM* (modulátor-demodulátor)
A telefonvonalak adatátviteli minőségének javulásával lehetőségünk lesz térben elkülönülő számítógépek egyidejű kommunikációjára. Ezt a távadatátviteli egység, a modem segítségével valósíthatjuk meg, amely a digitális jeleket analóg jelekké alakítja, hogy azok telefonvonalakon továbbíthatók legyenek.

— *Képdigitalizáló* (Scanner)
A képdigitalizáló segítségével ábrákat, szöveget „olvas" a gép. A papíron levő információt képpontokra bontja, a számítógép számára érthető bitek sorozatára alakítja, és beviszi a tárba.

Megjegyzés
A billentyűzeten kívül felsorolt beviteli egységek használatához megfelelő programra is szükség van.

b) **Kiviteli egységek**

— *Monitor* (display, képernyő)
Személyi számítógépek kiviteli egységei közül a monitor a legfontosabb, mert minden kimenő információ megjelenik a képernyőn, ha a számítógépnek nem adunk egyéb utasítást.
Képernyőre kerülnek:
— a billentyűzeten begépelt szövegek, adatok,
— az adatfeldolgozás eredményei,
— a számítógép üzenetei, *például*: hibaüzenetek, adatok kérése.

A monitor felépítése hasonlít a hagyományos tévékészülékhez, de annál jobb minőségű képet ad. (Egyes számítógépeket tévékészülékhez is csatlakoztathatunk, de a kép minősége rosszabb, és jobban rontja a szemünket is, mint a monitor.) A számítógépet és a monitort vezérlőkártyával kapcsolhatjuk össze. Ha a számítógép és a monitor közös tápfeszültséget hasz-

nál, akkor a monitort csak egyszer kell bekapcsolnunk, mert az az alapgéppel együtt kapcsolódik be.
A monitorok képpontokból állítják elő a karaktereket. Minél több pontot tud ábrázolni gépünk, minél jobb a felbontóképessége, annál élesebb a kép. A megfelelő gombokkal szabályozhatjuk a kép minőségét.

— *Nyomtató* (printer)
A nyomtató az eredmények megőrzését segíti. A papírra írt információk lehetnek grafikus ábrák, adatok és a program listája is. (Papírra nyomtatunk, ezért nagyon figyelnünk kell a papír befűzésére, főként akkor, ha leporellót használunk.)
A nyomtatón az írásminőség beállítását és a papír megfelelő elhelyezését különböző gombok segítik.
A nyomtatók sebessége a központi egységéhez képest kicsi, ezért a nyomtató működése idején a központi egység nem tudna dolgozni. Ennek kiküszöbölésére építettek be a nyomtatók legtöbbjébe saját memóriát, úgynevezett nyomtatópuffert. A központi egységből az információk átkerülnek a nyomtató memóriájába, és a nyomtató saját sebességének megfelelően innen dolgozza fel a kiírandó anyagot.

— *Rajzgép* (plotter)
Számítógép által vezérelhető rajzgép, amely az információkat kétdimenziós grafikus ábrázolásban jeleníti meg a papíron. Főként számítógépes tervezéskor használják.

c) Háttértárak

A perifériák közé soroljuk a háttértárakat is, amelyeken az információkat tároljuk. Alkalmazásukra a következők miatt van szükség:

— Az operatív memória csak a gép kikapcsolásáig őrzi meg tartalmát. Ha szükségünk van a benne tárolt programokra, adatokra, akkor kikapcsolás előtt a számítógéphez kapcsolunk valamilyen mágnesjelekkel működő tárat, és erre átmásoljuk a belső tár tartalmát. Ismételt felhasználáskor az információkat a belső tárba újból be kell töltenünk, amit bármikor megtehetünk.

— A belső tár csak meghatározott mennyiségű információt képes tárolni, ezért az éppen nem szükséges programokat, adatokat, eredményeket a külső tárolóban lehet tartani a program futása idején. Ezzel bővítjük az operatív memória kapacitását.
A háttértárak lehetnek: mágneslemezes tárak, mágnesszalagos tárak, félvezetős tárak stb. A következőkben megismerkedünk az általunk használt tárakkal.

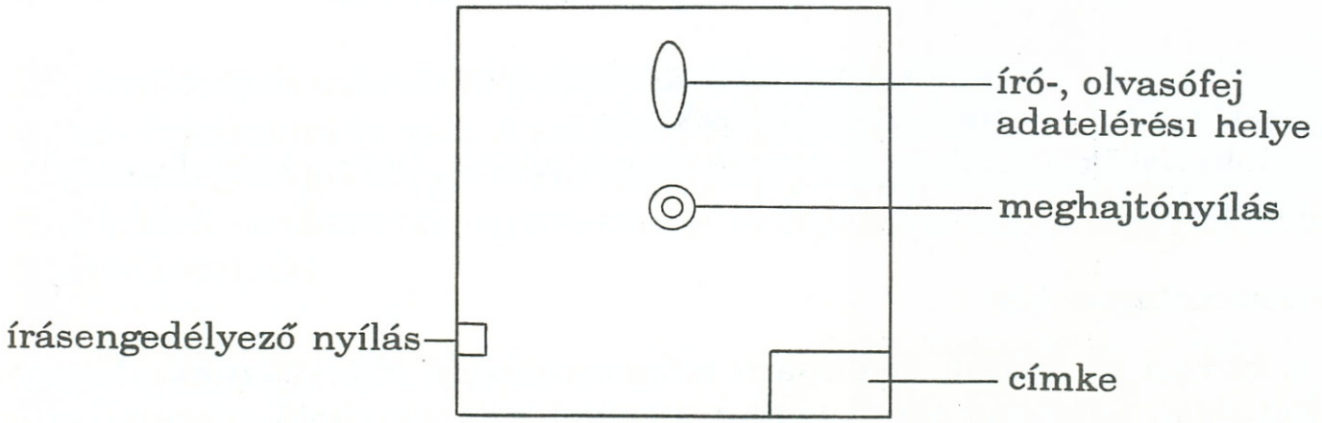
A mágneslemezes háttértárak fő részei:

— *Mágneslemez*, amelyen az információkat mágneses jelek formájában tároljuk. A mágneses állapot az egyesnek, a nem mágneses állapot a nullának felel meg, ezért megegyezik a számítógép kódolási módjával.

— *Meghajtó* (lemezegység, drive), amely:
  — a benne lévő villanymotor segítségével mozgatja a mágneslemezt,
  — a benne lévő elektronikus szerkezet segítségével vezérli az író-olvasó fej sugárirányú mozgását, így felviszi a lemezre az adatokat vagy leolvassa onnan őket.

A lemezes háttértárak által alkalmazott tárolók

— *Hajlékonylemez vagy floppy*
  Ez egy kör alakú, hajlékony, műanyagból készült, jól mágnesezhető anyaggal bevont lemez. Nagyon sérülékeny, ezért a lemezt egy levehetetlen műanyag tok védi, ezzel együtt helyezzük a meghajtóba is. A lemeznek azon részeihez, amelyek a tokból kilátszanak, tilos hozzányúlnunk.
  A *3. ábra* a hajlékonylemezt mutatja be.



3. ábra

A meghajtót általában beépítik a gép házába, de a lemez mozgatható, cserélhető. Kétféle méretű lemezt alkalmaznak, 5,25", illetve 3,5" átmérőjűt (" jel inchben adott méretet jelent, 1 inch = 2,54 cm).

A meghajtóba úgy kell betenni lemezünket, hogy a címke felfelé legyen, és az írásengedélyező ablak bal oldalra kerüljön. Az író-olvasó nyílással befelé, ütközésig kell tolni a lemezt, ekkor egy halk kattanással eltűnik a szemünk elől. Ilyenkor kell elfordítani a lemezegység kilincsét, mellyel biztosítjuk a lemezt a helyén. Előfordulhat, hogy betolás közben megakad lemezünk, ilyenkor húzzuk vissza és próbálkozzunk újra!

A lemez kivétele előtt nézzük meg, hogy világít-e a lemezegység lámpája, mert addig nem vehetjük ki a lemezt, amíg az ki nem alszik.

— *Merevlemez vagy winchester*

A winchestert a mikroszámítógépek tárolókapacitásának növelésére fejlesztették ki. A winchester mágneslemeze olyan mágneses felületű, fémből készült, nem cserélhető lemez, melyet a meghajtóval egybeépítve, légritkított térbe zárva találunk a számítógépben. A meghajtó és az adathordozó egyetlen egységet alkot. Előfordul, hogy több lemezt használunk egy közös tengelyre felfűzve. Leggyakrabban 5,25"-os lemezeket alkalmazunk. Minden lemez mindkét oldalához külön író-, olvasófej tartozik. Ezek a fejek csaknem érintik a lemezt, így egy porszem is óriási károkat okozna, ezért van szükség a hermetikus elzárásra. A winchesterek érzékenyek, ezért érdemes megfelelő gyakorisággal (félévente) kimenteni a winchester tartalmát, újból formattálni, majd visszaírni rá.

— *Cserélhető winchester*

Az adatfeldolgozás során nagy tömegű adat tárolására használjuk. Egyesíti a hagyományos winchester és a floppy előnyeit. Ugyanolyan nagy tárolókapacitású, gyors elérésű, megbízható, mint a merevlemez, de könnyen cserélhető, egyszerűen kiemelhető, mozgatható, mint a hajlékonylemez. Mindezek biztosítják jövőbeni széles körű, gyors elterjedését.

A számítógéphez kapcsolt lemezegységeket betűk segítségével azonosíthatjuk:

A: esetleg B: a hajlékony lemezegységek jele,
C: esetleg D: a fix lemezegységek jele.

A további betűk a hálózati lemezmeghajtót jelölik, illetve különleges célokat szolgálnak.

A mágnesszalagos tár

Ebben a tárban az adatok tárolására mágnesszalagot alkalmazunk. A mágnesszalag olyan műanyag alapú szalag, amelyet mágnesezhető anyaggal vontak be, és általában 9 egymás melletti sávon tároljuk rajta az információkat.

Előnye, hogy viszonylag olcsó, a berendezésből bármikor kivehető, kicserélhető, könnyen szállítható és tárolható.

Hátránya, hogy az adatokat csak folyamatosan, a tárolási sorrendnek megfelelően tudjuk felhasználni alkalmazásakor.

A mágnesszalagos tárat olyan esetekben alkalmazhatjuk hatékonyan, amikor sok adatot kell sorban feldolgoznunk, majd tárolnunk.

Kisebb számítógépeknél használhatunk mágnesszalag-kazettás tárat. Ennek adathordozója a kazettás magnetofonoknál jól ismert kazetta. A feltekert mágnesszalagot egy szabványnak megfelelő tokban helyezték el, melyet felhasználás közben is benne hagyunk a tokban. (Ilyen kazettával gyakran kerülünk kapcsolatba kedvenc slágereink hallgatása előtt, amikor a magnóba helyezzük.)

Kérdések és feladatok

1. Mit nevezünk számítógépnek?
2. Mit jelent a szoftver és a hardver? Milyen kapcsolat van köztük?
3. Hogyan csoportosíthatjuk a szoftvereket rendeltetésük szerint?
4. Hogyan oszthatjuk fel a hardvert?
5. Ismertessük a számítógépek alapkonfigurációját!
6. Mi a szerepe a központi feldolgozó egységnek?
7. Mi a különbség a ROM és a RAM között?
8. Mi a perifériák feladata?
9. Milyen beviteli egységekről beszéltünk?
10. Milyen kiviteli egységeket ismertünk meg?
11. Miért van szükség a háttértárakra?
12. Milyen részekből épülnek fel a mágneslemezes tárak?
13. Hogyan kell a meghajtóba helyezni hajlékonylemezünket?
14. Milyen előnyei vannak a cserélhető winchesternek?

4. A számítógépek adatábrázolása

A számítógépek számára az információt olyan formájúra kell átalakítani, hogy azokat értelmezni tudják. A számítógép által elérhető jeleket hívjuk kódoknak. A kódok segítségével egyértelműen lefordíthatjuk az információkat (kódoljuk), és a kódolt rendszerből egyértelműen visszakaphatjuk az eredeti információkat (dekódoljuk).

Olvasmány

Természetes számunkra, hogy az emberi beszédet írásjelekkel rögzíthetjük. Nem jut eszünkbe, hogy ez is egyfajta kódolás.

Az általunk ismert első gépi nyelvet Morse, amerikai festő dolgozta ki (1837). *Az ábécéjében minden betűt, számot, írásjelet pontokkal és vonásokkal ábrázolt. A számítógép számára egyszerűbb számokat értelmezni, ezért ezeket alkalmazzuk. A morzejeleket is könnyen átírhatjuk a számítógépen használt úgynevezett bináris kódba.*

A számítógépekben az információk tárolására elektromos áramköröket alkalmaznak. Ez lehetővé teszi, hogy a számítógépek működését a „van feszültség" és a „nincs feszültség" állapotára vezessük vissza. Megállapodás szerint az 1 számjegy a bekapcsolást (van áram), a 0 számjegy a kikapcsolást (nincs áram) jelenti (vagy éppen fordítva). Ez a kettes (bináris) számrendszer használatának felel meg.

A számítógép által használt jelek között a legkisebb egység a bit (binary digit). Két értéke lehet, a 0 vagy az 1. A bit a számítástechnikában az adatok tárolásának, feldolgozásának és átvitelének alapegysége. Egy bittel kevés in-

formációt tudunk továbbítani, ezért összefogunk több bitet, ekkor egy nagyobb egységhez jutunk, a bájthoz (byte-hoz). Ezzel már nemcsak egy jelet, hanem egy adatot is tudunk továbbítani.

Önkényesen megállapodtak abban, hogy

8 bit = 1 byte, 1 kilobyte = 1024 byte.

5. Számrendszerek

Helyiértékes számrendszerben a szám jegyeinek alakja és helye együttesen határozzák meg a szám értékét (számrendszer alapszáma 1 nem lehet).

A tízes számrendszerben a számokról tudjuk, hogy melyik számjegy milyen helyiértékű.

Például: *a 247-ben van:*
2 db *százas* (2×10^2) + 4 db *tízes* (4×10^1) + 7 db *egyes* (7×10^0).

A számokat tehát 10 hatványai szerint építhetjük fel. A tíz hatványai előtt 10-féle szorzótényező szerepelhet a 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A 0-val azt jelöljük, ha 10-nek valamelyik hatványa nem szerepel.

Például: $1020 = 1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 0 \times 10^0$.

A számítógépek nem tízes, hanem általában kettes, nyolcas vagy tizenhatos számrendszert használva működtethetők.

a) Kettes számrendszer

A bináris (kettes számrendszerbeli) ábrázolás alapelve ugyanaz, mint a tízes számrendszerbeli ábrázolási módé. Itt a számokat a 2 hatványai szerint alakítjuk ki. Kettő hatványai előtt csak kétféle szorzótényező szerepelhet: a 0 vagy az 1.

Például: *a tízes számrendszerben felírt 247 alakja a kettes számrendszerben:*
11110111, *jelölése:* $247_{10} = 11110111_2$

Jelentése: $247=1 \times 2^7+1 \times 2^6+1 \times 2^5+1 \times 2^4+0 \times 2^3+1 \times 2^2+1 \times 2^1 + 1 \times 2^0$.

Az átalakítás lépései lehetnek:

— Osszuk el a számot kettővel, és írjuk a szám alá a hányadost, egy másik oszlopba a maradékot! (A maradék 0 vagy 1 lehet.)

— Osszuk el kettővel az előbb kapott hányadost, majd az új hányadost írjuk az előző alá, és a maradékot is az előző maradék alá! Mindezt addig folytatjuk, amíg hányadosként nullát nem kapunk.

— Írjuk fel a maradékokat alulról felfelé haladva, és így megkapjuk a szám bináris alakját.

Például: *a)* *b)*

Hányadosok	Maradékok	Hányadosok	Maradékok
173	1	157	1
86	0	78	0
43	1	39	1
21	1	19	1
10	0	9	1
5	1	4	0
2	0	2	0
1	1	1	1
0		0	

$173_{10} = 10101101_2$ $157_{10} = 10011101_2$

Olvasmány

Törtrészek átalakítását a következő módszerrel végezhetjük:

- *Szorozzuk meg kettővel az adott szám törtrészét, és írjuk külön oszlopba a szorzat egészrészét és törtrészét is!*
- *Szorozzuk meg az előző szorzás után kapott törtrészt ismét kettővel, és a megfelelő oszlopba írjuk az egész- és a törtrészt!*
- *Szorozzuk a törtrészeket mindaddig kettővel, amíg a szorzandó szám törtrésze* 0 *nem lesz. (Általában a* 0*-t csak megközelítjük, mert* 2 *hatványaival nem fejezhető ki a törtrész.)*
- *A szorzás után az egészrészek oszlopában kapott* 0 *vagy* 1 *értékek adják a decimális szám törtrészének bináris alakját, a legmagasabb helyiértéktől kezdve. (A leolvasás fölülről lefelé haladva történik.)*

Például: *a)* *b)*

Egészrészek	Törtrészek	Egészrészek	Törtrészek
0	67	0	125
1	34	0	250
0	68	0	500
1	36	1	000
0	72		
1	44		
0	88		
1	76		
1	52		
1	04		

$0{,}67_{10} = 0{,}101010111...{}_2$; $0{,}125_{10} = 0{,}001_2$.

A két eljárást együtt alkalmazva (külön az egész- és a törtrész) írhatjuk át a racionális számokat tízes számrendszerből kettesbe.

Például: $173{,}67_{10} \approx 10101101{,}101010111_2$

$157{,}125_{10} = 10011101{,}001_2$

Megjegyzés

A tizenhatos számrendszerben a 10–16 *közötti számok is „egyjegyűek", ezeket* A, B, C, D, E, F*-fel jelöljük.*

b) **Számrendszerek közti kapcsolatok**

Tízes (decimális)	Kettes (bináris)	Négyes	Nyolcas (oktális)	Tizenhatos (hexadecimális)
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	10	4	3
5	101	11	5	5
6	110	12	6	6
7	111	13	7	7
8	1000	20	10	8
9	1001	21	11	9
10	1010	22	12	A
11	1011	23	13	B
12	1100	30	14	C
13	1101	31	15	D
14	1110	32	16	E
15	1111	33	17	F
16	10000	100	20	10
17	10001	101	21	11
18	10010	102	22	12
19	10011	103	23	13
20	10100	110	24	14

6. A tárolt adatok szerkezete

Feldolgozás előtt az adatokat különböző szempontok szerint csoportosítjuk. Erre felhasználjuk az adatok jelentését, *például*: nevek, születési időpontok, címek. Ezeket az értelemszerűen kialakított csoportokat könnyebben megtaláljuk, mintha egyenként keresgélnénk a hiányzó adatokat a feldolgozás során.

A legkisebb logikai egység a karakter. Ez lehet egy betű, egy számjegy, egy írásjel, egy műveleti jel.

Több egymás mellé helyezett karakter a mező. A mező egyetlen adat tárolására szolgál. Tartalmazhat csak betűket, *például*: név; csak számokat, *például*: telefonszám; betűket, számokat, írásjeleket, *például*: Kossuth u. 8.

Több egymás utáni mező, melyek között valamilyen logikai kapcsolat van, rekordot alkot. Egy rekord lehet például valakinek a sorszáma, neve, lakóhelye, telefonszáma együtt. Ha ezeket az adatokat több emberről ismerjük, például egy osztály tanulóiról, akkor ezek az összetartozó rekordok egy adatállományt vagy fájlt alkotnak. Az azonos felépítésű rekordok halmazát nevezhetjük fájlnak (file-nak) .

Vannak olyan állományok is, amelyeket nem osztunk rekordokra, például azok, amelyek egy program utasításait tartalmazzák.

Az állomány egy felépítési lehetőségét a *4. ábra* szemlélteti.

1. mező	ÁGI mező	GYŐR mező	56322 mező	rekord	Állomány
2. mező	RITA mező	DABAS mező	35284 mező	rekord	
3. mező	ESZTER mező	VÁMOS mező	58384 mező	rekord	

4. ábra

Feladatok

1. Írjuk fel a következő tízes számrendszerbeli számokat kettes számrendszerben!

a) 348_{10} ; *b)* 76_{10} ; *c)* 873_{10} .

2. Írjuk fel a következő kettes számrendszerbeli számok tízes számrendszerbeli megfelelőjét!

a) 101101_2 ; *b)* 1001110_2; *c)* 1110011_2.

3. Írjuk fel tízes és kettes számrendszerben a következő tizenhatos számrendszerbeli számokat!

a) DAC 26_{16} ; *b)* 1 E 9_{16} ; *c)* FEB04_{16} .

4. Hogyan ismerhető fel egy kettes számrendszerben írt számról, hogy osztható:

a) 2-vel, 4-gyel, 8-cal;
b) 3-mal, 7-tel, 15-tel;
c) 5-tel, 9-cel, 17-tel.

5. Végezzük el a kijelölt átalakításokat!

a) $0{,}425_{10}$ = ... $_2$;
b) $0{,}682_{10}$ = ... $_2$;
c) $0{,}2338_{10}$ = ... $_2$;
d) $56{,}78_{10}$ = ... $_2$;
e) $152{,}315_{10}$ = ... $_2$.

6. Ismertessük a leggyakrabban előforduló adatszerkezeteket!

B) A személyi számítógépek

A számítógép olyan elektronikus berendezés, amely információk feldolgozására képes a betáplált program utasításainak megfelelően. A személyi számítógép olyan általános célra kifejlesztett számítógép (PC), amelyet megtalálunk a háztartásokban. Egy felhasználó számára készült.

A személyi számítógép előnyei:

— könnyen hozzáférhető,
— a program futása közben is lehet vele párbeszédet folytatni,
— egyszerű, ezért nem igényel külön kezelőszemélyzetet,
— alacsony az ára,
— csak én (a felhasználó) vagyok vele kapcsolatban.

1. A számítógép és a felhasználó kapcsolata

A számítógép működésekor a számítógép és a felhasználó között közvetlen munkakapcsolat alakul ki. Ebben az esetben párbeszédes vagy interaktív üzemmódról beszélünk. A felhasználó a billentyűzeten keresztül parancsot ad a gépnek, amelyet a számítógép értelmez, végrehajt, majd a képernyőn keresztül üzenetet ír ki a számára. Erre a felhasználó újból válaszol, és így tovább.

Ha a parancsokat programban soroljuk fel, akkor ezeket a számítógép utasításként kezeli, nem hajtja végre őket azonnal. A program futásakor vizsgálja meg a számítógép az utasításokat, és végrehajtja őket anélkül, hogy a felhasználónak be kellene avatkozni. Ezt a felhasználási módot kötegelt vagy batch üzemmódnak nevezzük.

Munkánk során a két üzemmódot általában vegyesen alkalmazzuk (adatok beírása — párbeszédes; programfutás — kötegelt).

2. Számítógépek kompatíbilitása

A kompatíbilitás helyettesíthetőséget, összeilleszthetőséget, felcserélhetőséget jelent. Az a két számítógép kompatíbilis egymással, amelyen ugyanaz a program változtatás nélkül futtatható, és ugyanolyan eredményre vezet. Különböző gyártmányú, tökéletesen kompatíbilis gépeket nem találunk, mert az újabb géptípusok a korábbiak fejlettebb változatai. Az újabb gépen megírt program tartalmazhat olyan utasítást, amelyet a régebbi típus nem tud végrehajtani, mert az adott utasítás többet kíván, mint amire a gép képes. Ilyenkor a képernyő rendszerint lemerevedik, és a gépet a billentyűzetről nem befolyásolhatjuk. Munkánkat csak a számítógép kikapcsolása, majd visszakapcsolása után folytathatjuk.

A régebbi modelleken készült programokat általában futtathatjuk az újakon, de ez fordítva nem mindig lehetséges. Azt mondhatjuk, hogy az IBM PC gépek felülről kompatíbilisek. Ez nagyon fontos a fejlesztéseknél, mert újabb gépek esetén nem kell újra írni a meglévő programokat.

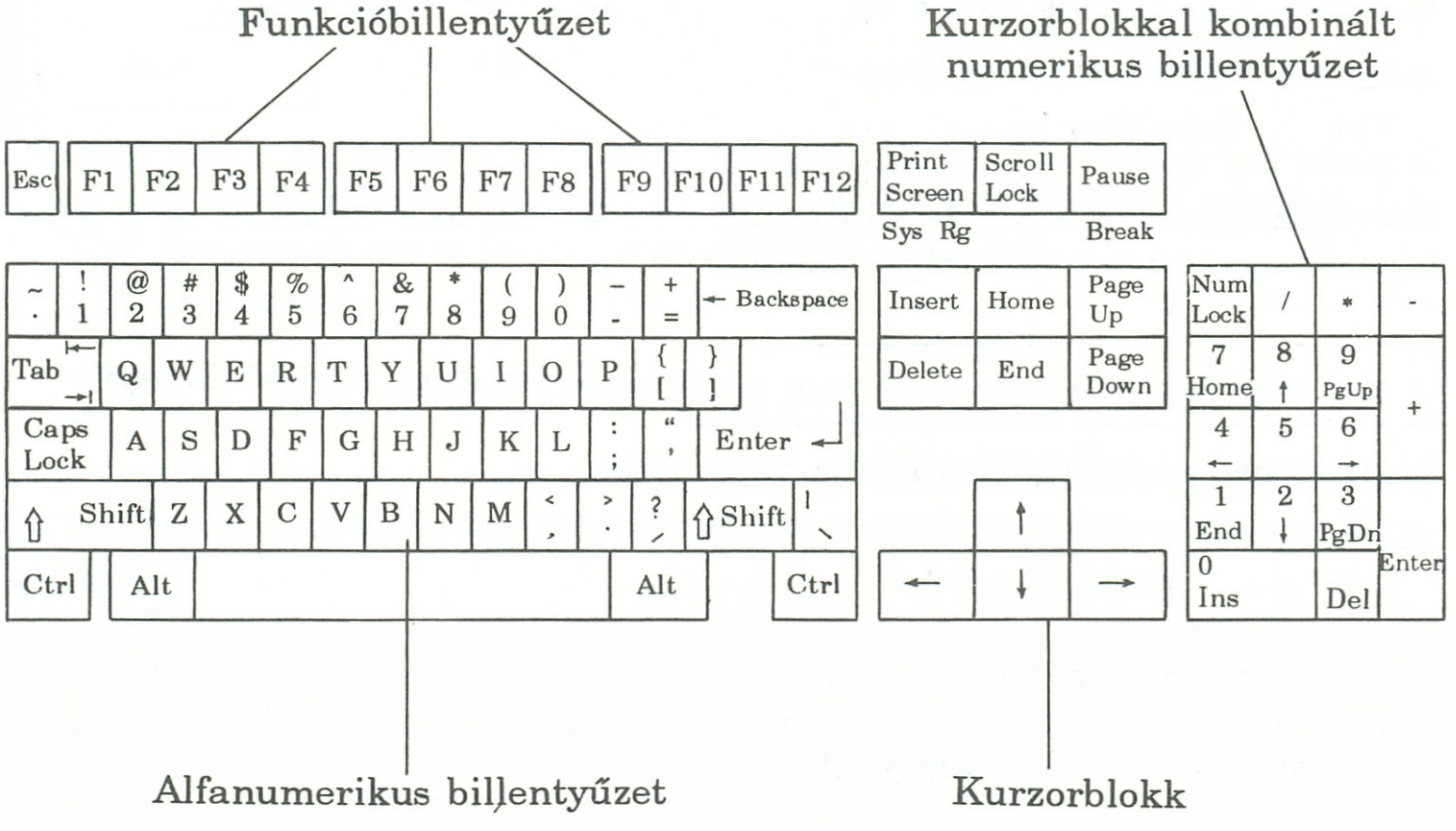
Hazánkban ennek a gépcsaládnak több tagja is elterjedt. A kompatíbilitás sorrendjében és időrendben ezek a következők: IBM PC, IBM PC/XT, IBM PC/AT, 286-os, 386-os, 486-os AT.

C) A PC-k

1. A PC-k billentyűzete

A számítógép használatakor a billentyűzettel kerülünk legtöbbször kapcsolatba. Ismerkedjünk meg vele részletesebben, hogy jól tudjuk használni!

A PC személyi számítógépeken sokféle billentyűzet lehet, de általában kétfélét találunk: 87, illetve 101 gombosat. Az *5. ábra* egy jellegzetes 101 gombos billentyűzetet mutat be.



5. ábra

B) A személyi számítógépek

A számítógép olyan elektronikus berendezés, amely információk feldolgozására képes a betáplált program utasításainak megfelelően. A személyi számítógép olyan általános célra kifejlesztett számítógép (PC), amelyet megtalálunk a háztartásokban. Egy felhasználó számára készült.

A személyi számítógép előnyei:

— könnyen hozzáférhető,
— a program futása közben is lehet vele párbeszédet folytatni,
— egyszerű, ezért nem igényel külön kezelőszemélyzetet,
— alacsony az ára,
— csak én (a felhasználó) vagyok vele kapcsolatban.

1. A számítógép és a felhasználó kapcsolata

A számítógép működésekor a számítógép és a felhasználó között közvetlen munkakapcsolat alakul ki. Ebben az esetben párbeszédes vagy interaktív üzemmódról beszélünk. A felhasználó a billentyűzeten keresztül parancsot ad a gépnek, amelyet a számítógép értelmez, végrehajt, majd a képernyőn keresztül üzenetet ír ki a számára. Erre a felhasználó újból válaszol, és így tovább.

Ha a parancsokat programban soroljuk fel, akkor ezeket a számítógép utasításként kezeli, nem hajtja végre őket azonnal. A program futásakor vizsgálja meg a számítógép az utasításokat, és végrehajtja őket anélkül, hogy a felhasználónak be kellene avatkozni. Ezt a felhasználási módot kötegelt vagy batch üzemmódnak nevezzük.

Munkánk során a két üzemmódot általában vegyesen alkalmazzuk (adatok beírása — párbeszédes; programfutás — kötegelt).

2. Számítógépek kompatíbilitása

A kompatíbilitás helyettesíthetőséget, összeilleszthetőséget, felcserélhetőséget jelent. Az a két számítógép kompatíbilis egymással, amelyen ugyanaz a program változtatás nélkül futtatható, és ugyanolyan eredményre vezet. Különböző gyártmányú, tökéletesen kompatíbilis gépeket nem találunk, mert az újabb géptípusok a korábbiak fejlettebb változatai. Az újabb gépen megírt program tartalmazhat olyan utasítást, amelyet a régebbi típus nem tud végrehajtani, mert az adott utasítás többet kíván, mint amire a gép képes. Ilyenkor a képernyő rendszerint lemerevedik, és a gépet a billentyűzetről nem befolyásolhatjuk. Munkánkat csak a számítógép kikapcsolása, majd visszakapcsolása után folytathatjuk.

A régebbi modelleken készült programokat általában futtathatjuk az újakon, de ez fordítva nem mindig lehetséges. Azt mondhatjuk, hogy az IBM PC gépek felülről kompatíbilisek. Ez nagyon fontos a fejlesztéseknél, mert újabb gépek esetén nem kell újra írni a meglévő programokat.

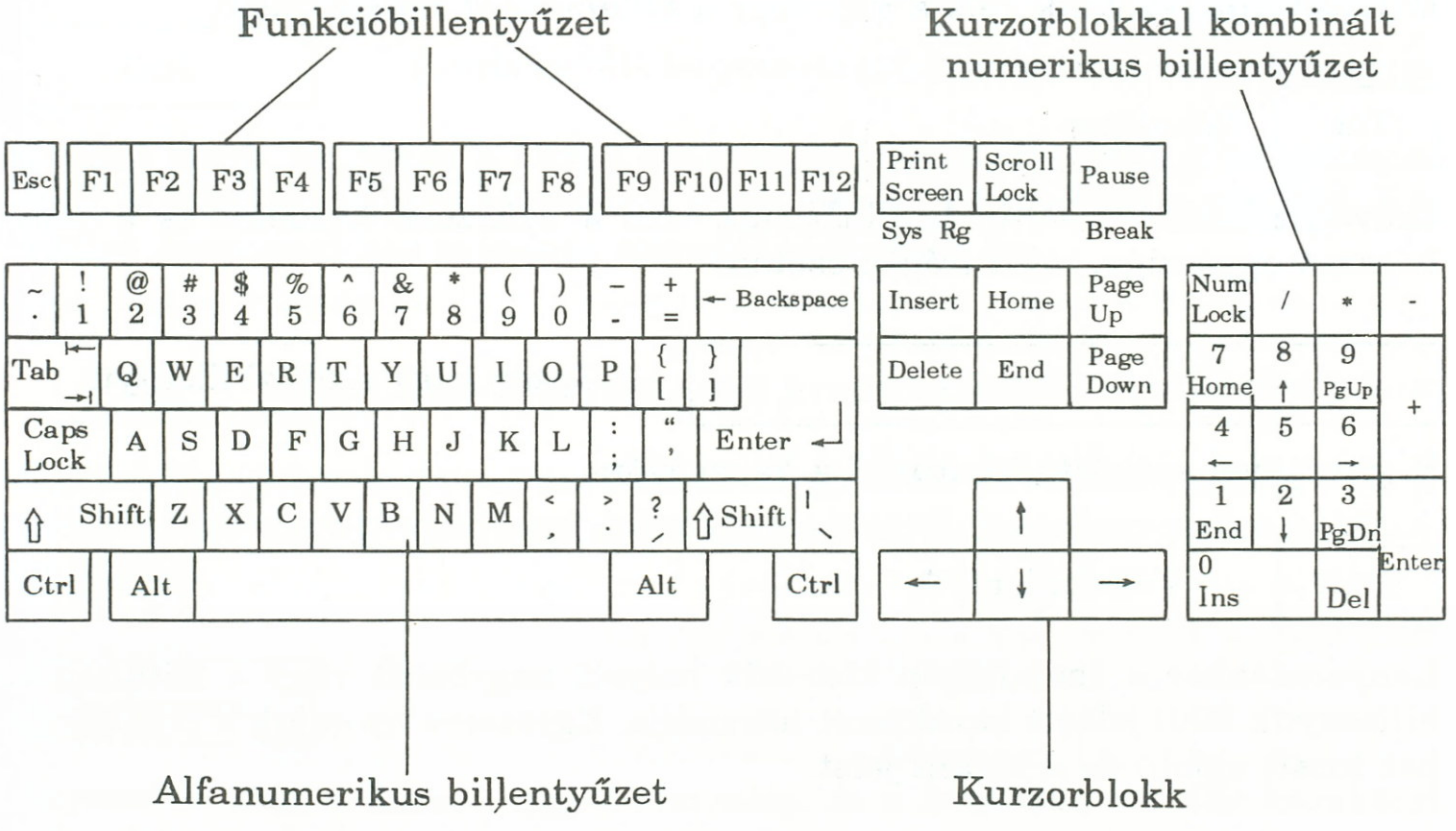
Hazánkban ennek a gépcsaládnak több tagja is elterjedt. A kompatíbilitás sorrendjében és időrendben ezek a következők: IBM PC, IBM PC/XT, IBM PC/AT, 286-os, 386-os, 486-os AT.

C) A PC-k

1. A PC-k billentyűzete

A számítógép használatakor a billentyűzettel kerülünk legtöbbször kapcsolatba. Ismerkedjünk meg vele részletesebben, hogy jól tudjuk használni!

A PC személyi számítógépeken sokféle billentyűzet lehet, de általában kétfélét találunk: 87, illetve 101 gombosat. Az *5. ábra* egy jellegzetes 101 gombos billentyűzetet mutat be.



5. ábra

A billentyűzet kialakításakor figyelembe vették azt, hogy milyen feladatot látnak el az egyes gombok, ezért különböző csoportokban helyezték el őket. A következő egységeket különböztethetjük meg:

a) alfanumerikus billentyűzet (írógép-billentyűzet),
b) vezérlőbillentyűk,
c) kurzorblokkal kombinált numerikus billentyűzet,
d) különálló kurzorblokk,
e) funkcióbillentyűzet.

Üljünk le számítógépünk mellé és rögtön próbáljuk ki a tanultakat!

a) Alfanumerikus billentyűzet

A számítógépen kiterjesztett amerikai írógép-billentyűzetet találunk (betűket, számokat, egyéb jeleket). Ha a kiírásban szeretnénk, hogy a magyar ékezetes betűk jelenjenek meg, akkor ehhez külön programot kell futtatnunk. A billentyűk ismétlésre képesek, vagyis egy billentyű tartós lenyomásakor a számítógép automatikusan ismétli a rajta lévő jelet a lenyomás ideje alatt.

b) Vezérlőbillentyűk

Számítógépünkön olyan billentyűket is találunk, melyek leütésekor látszólag nem történik semmi, mert a képernyőn nem jelenik meg újabb karakter. Ezek láthatatlanul szólnak bele a gép vagy a billentyűzet működésébe.

Tab	Tabulátor

Szövegszerkesztés során beállíthatjuk vele a bekezdés kezdetét és a sorok hosszát ugyanúgy, mint írógép esetén.

	Space vagy szóközbillentyű

Segítségével szóközt „írhatunk" a képernyőre.

Shift	Váltóbillentyű

Lenyomásakor a számítógép kisbetűk helyett nagybetűt vagy a kétállapotú billentyűk felül jelzett karakterét használja. Egyszerre nyomjuk a **Shift**et (lehet kicsit előbb) és a kívánt jelet.

CapsLock

Nagybetűállást rögzítő billentyű

A billentyű lenyomása után a **Shift** lenyomása nélkül nagybetűkkel írhatunk (bekapcsolt állapotát jelzőlámpa és a benyomva maradt gomb mutatja).

Ha ilyen esetben kisbetűvel akarunk írni, akkor le kell nyomni a **Shift** billentyűt. Vigyázzunk, mert a kétállapotú billentyűk lenyomásakor továbbra is az alaphelyzetet, az alul jelölt jelet kapjuk!

Num Lock

A számbillentyűzetet numerikus állásban rögzíti.

Bővebben a következő részben ismertetjük.

Enter

vagy

Return

vagy

Cr

Lezáró billentyű. Lenyomásával jelezzük a számítógépnek, hogy befejeztük az utasításainkat vagy az adatok bevitelét. Dolgozza fel azokat, és így adunk parancsot arra, hogy ugorjék a következő sor elejére.

Alt

A vele együtt lenyomott gomb jelentését megváltoztatja.

Ctrl

A vele együtt lenyomott gomb jelentését megváltoztatja.

Mindkét gomb, az **Alt** és a **Ctrl** a számítógép billentyűzetének a lehetőségeit bővíti, mivel bármelyiket leütve egy gomb más kódot továbbít a gép felé. Különböző programokban más-más szerepet tölthetnek be.

Esc

Általában egy adott programrészből való kilépést jelent.

Megszakíthatjuk vele a program futását. Hogy ez ne történhessék meg, a programozók külön programban gondoskodhatnak arról, hogy lenyomása hatástalan legyen.

Backspace

Lenyomása után a kurzor eggyel visszalép, és a korábban ott álló karaktert törli a képernyőről.

Print Screen	vagy	Prt Scr

A képernyőn lévő szöveg kinyomtatását segíti. A **SHIFT** billentyűvel együtt lenyomva kinyomtatja a képernyő tartalmát, majd a program fut tovább, és a kurzor is a helyére áll. Lenyomása előtt a számítógéphez kell kapcsolnunk egy működőképes nyomtatót, mert különben megakadhat a programunk futása.

c) Kurzorblokkal kombinált numerikus billentyűzet

A billentyűzet jobb oldalán helyeztek el egy kétállapotú, különálló blokkot. Alaphelyzetben a szövegszerkesztést segítő billentyűzetet találjuk.

↑	↓	←	→

Kurzormozgató billentyűk. Valamelyik billentyű lenyomása után a kurzor a nyíllal jelzett irányba mozdul el.

Insert	vagy	Ins	Beszúrást segítő billentyű

A billentyű lenyomása után megjelenik egy villogó négyszög a kurzor helyén. Megkezdhetjük a beszúrást.
(Működése programtól függhet.)

Delete	vagy	Del	Törlést segítő billentyű

A billentyű lenyomása után a kurzortól jobbra lévő karakterek eggyel balra lépnek. A kurzor helyén lévő jelet letörli.

Home	Ugrást előidéző billentyű

Lenyomása után a kurzor alaphelyzetbe, a képernyő bal felső sarkába, az első karakterre ugrik. Hatása az alkalmazott szövegszerkesztőtől is függ. (Előfordul, hogy a kurzor csak a sor elejére ugrik.)

End	Ugrást előidéző billentyű

Lenyomása után a kurzor a szövegegység végére ugrik. Szövegszerkesztőtől függhet a hatása.

Page Up	vagy	Pg Up	és a	Page Down	vagy	Pg Dn

A Pg Up és a Pg Dn billentyűket főként programozók/szövegszerkesztők használják olyan esetekben, ha a program/szöveg több képernyőoldalt foglal el; ezekkel lehet vezérelni a képernyők közötti váltást, azt, hogy visszafelé vagy előrefelé akarunk lépni. Működésük programfüggő.

Olvasmány

A kurzort mozgató billentyűket egy különálló csoportban is megtalálhatjuk. Olyan esetben célszerű őket használni, amikor a numerikus billentyűzeten számokat írunk, mert ilyenkor nem kell visszakapcsolnunk a rögzített állapotból.

A kurzorblokk felső állapotának alkalmazásakor a képernyőn számok jelennek meg. Ha folyamatosan számokat írunk, akkor érdemes ezt az állapotot rögzíteni a Num Lock billentyű segítségével. (A kijelző lámpa világít ebben az esetben is.) A blokkban található billentyűk lenyomása után a számok nullától kilencig, a tizedespont és a legfontosabb műveleti jelek jelennek meg a képernyőn. Táblázatok számokkal történő kitöltésénél meggyorsíthatja az adatok bevitelét.

Itt is választási lehetőségünk van, mert mi dönthetjük el, hogy a numerikus blokk számjegyeit vagy az írógép-billentyűzet számait használjuk. Mindkét esetben ugyanazt az eredményt kapjuk a képernyőn. (Célszerű hozzászokni a kurzorblokk használatához, mert a jobb oldali számokkal lényegesen gyorsabb és könnyebb az adatok bevitele.)

d) Funkcióbillentyűzet

Az F1 ... F12 billentyűk szintén egy csoportot alkotnak, mert ezek mindegyike szabadon programozható. Ha egy programon belül leütjük azt a billentyűt, amelyik programozva volt, akkor az végrehajtja a feladatot a kezelői utasításban leírtaknak megfelelően. Alkalmazhatók egyedül vagy más gombokkal együtt is *(például*: Ctrl-lal).

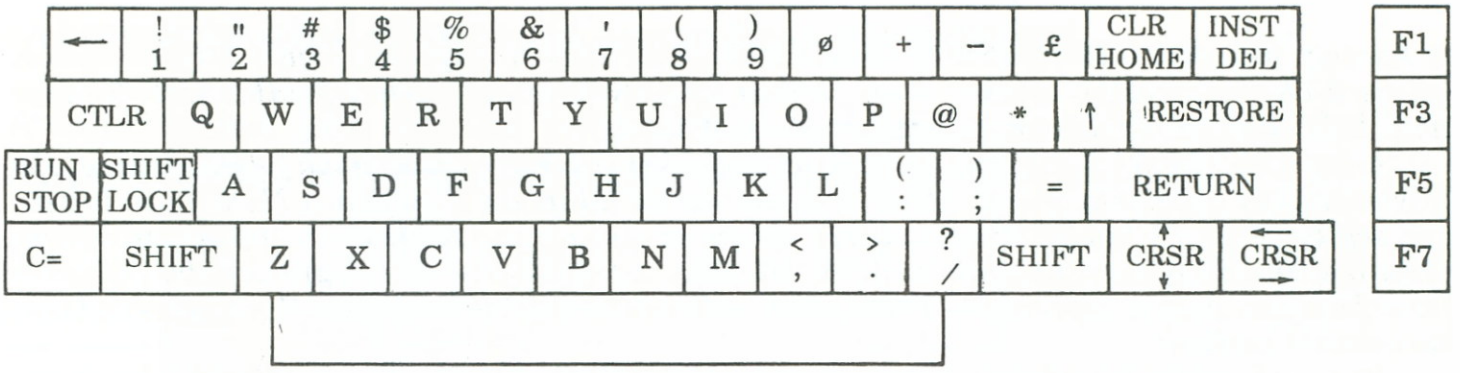
D) A Commodore-64 személyi számítógép

Hazánkban a hobbigépek közül a C-64 terjedt el legjobban. Nagyon sok háztartásban találkozhatunk vele, és az iskolákban is használják még. Ismerkedjünk meg vele!

A BASIC nyelvű programozás a Commodore-64-en is párbeszédes formában zajlik, vagyis a kezelő a futó programba azonnal beavatkozhat, belejavíthat. Ebben a számítógépben megtalálható egy interpreter, amely a megírt BASIC nyelvű programot utasításonként értelmezi, és átalakítja a számítógép számára érthető, gépi kódú utasítások sorozatává.

1. A C-64 számítógép billentyűzete

Személyi számítógépünkkel közvetlenül kapcsolatot teremthetünk az előttünk lévő billentyűzet segítségével *(6. ábra)*. A hagyományos írógép-billentyűzeten (betűk, számok, írásjelek) kívül speciális funkciójú billentyűk is vannak rajta. Érdemes megismerkednünk ezekkel a gombokkal, mert segítenek a beírás javításában, törlésében, a program megállításában stb. Ezeket a billentyűket csak le kell nyomni, nem kell betűnként beírni a parancsokat.



6. ábra

A számítógépen a kiterjesztett amerikai írógép-billentyűzet található (Y és Z betűk a magyar írógéphez képest felcserélve). A számítógép a szóközbillentyű tartós lenyomásakor automatikusan ismétel, a többi billentyűnél nem.

a) A billentyűzet vezérlése

SHIFT (betűváltó billentyű)
A szokványos írógépek betűváltójához hasonlít. Lenyomásakor a számítógép átvált a billentyű tetején felül látható jelre, illetve átvált a billentyű homlokélén jobboldalt látható grafikus jelre.

SHIFT/LOCK
Rögzítő SHIFT gomb. Feloldása újra megnyomással történik. Ez a két váltóbillentyű hasonló funkciójú az írógépnél megszokott kis- és nagybetűváltóhoz.

C= billentyű
Commodore billentyű, a Commodore cég márkajele. Lenyomásakor a számítógép átvált a billentyű homlokélén baloldalt található grafikus jelre.

SHIFT és **C=**
A billentyűk egyidejű lenyomásakor a teljes képernyő átvált a kisbetűs karakterkészletre és vissza.

CTRL
Kontrollbillentyű. Átváltja a beadott karakterek színét a felső sorban a billentyűk élén található színek valamelyikére. Ezt a billentyűt lenyomva kell tartani, amikor leütjük a megfelelő másik billentyűt. Vezérelhető vele a normál és negatív karakterek billentyűzete is.

RVS/ON negatív **RVS/OFF** normál karakterek.

b) **A kurzort vezérlő billentyűk**

(A képernyőn villogó négyzetet nevezzük kurzornak.)

CRSR ↑ ↓
A billentyű lenyomásakor a kurzor lefelé mozog.

CRSR ↑ ↓ és **SHIFT**
A billentyűk együttes lenyomásakor a kurzor felfelé lép.

CRSR → ←
A billentyű lenyomásakor a kurzor jobbra lép.

CRSR → ← és **SHIFT**
A billentyűk együttes lenyomásakor a kurzor balra lép.

INST/DEL
A billentyű lenyomásakor a kurzortól balra eső karakter törlődik. A kurzortól jobbra lévő jelek egy pozícióval balra lépnek.

INST/DEL és **SHIFT**
A billentyűk együttes lenyomásakor a kurzor helye üres lesz, és az utána lévő szöveg jobbra csúszik, így új karaktereket szúrhatunk be.

CLR/HOME
A billentyű lenyomásakor a kurzor alaphelyzetbe, a képernyő bal felső sarkába megy vissza.

CLR/HOME és **SHIFT**
A billentyűk együttes leütésekor törlődik a képernyő, és a kurzor a bal felső sarokba lép.
A kurzormozgató billentyűk automatikus ismétlésűek. A kurzor addig mozog, amíg a billentyűket lenyomva tartjuk.

c) **Egyéb billentyűk feladatai**

RETURN
A billentyű lenyomása jelzi a számítógépnek, hogy a parancsnak vagy utasításnak vége van, olvassa be a tárba a képernyőn található karaktereket.

π
A billentyű lenyomása adja a *Pí* értékét a számítógép számára a lehetséges legnagyobb pontossággal.

RUN/STOP
A billentyű lenyomásával leállíthatjuk a program futását (a képernyőn megjelenik: BREAK IN LINE...).

RUN/STOP és SHIFT
A billentyűk együttes lenyomásakor a program a kazettáról a tárba töltődik, és elindul.

RUN/STOP és RESTORE
A billentyűk lenyomásakor a gép alaphelyzetbe tér.

E) Az operációs rendszer

A számítógép működését és a felhasználói programok futását egy programrendszer teszi lehetővé, amelyet a számítógép operációs rendszerének (Operating System: OS) neveznek.

Az operációs rendszer olyan programok összessége, amelyek biztosítják a kapcsolatot a felhasználó és a számítógép között. Ez a rendszerszoftver összehangolja a számítógép különböző egységeinek működését, lebonyolítja a perifériaműveleteket, ütemezi a programok végrehajtását, értelmezi a kapott parancsot, majd ellenőrzi tartalmilag és formailag, hibaüzenetet ír a képernyőre stb. Nagyon sokféle feladatot lát el ez a működtető rendszer, nélküle nem tudnánk dolgozni, nélküle a gép életképtelen. Az operációs rendszer összekötő kapocs a hardver és a szoftver között. (Az operációs rendszert a hardverhez szokták sorolni.) Felkészíti a számítógépet egy feladat megoldására. Egy általunk megírt program vagy lemezről betöltött program futtatásához szükség van egy, már a gépben lévő program futására is.

1. Az operációs rendszer részei

1. *Vezérlőprogram:* biztosítja a számítógép részeinek együttműködését és a programok futását.
2. *Segédprogramok* (utility program): ismétlődő rutinfeladatokat végeznek. Például, a végrehajtandó programot betöltik az operatív tárba vagy lehetővé teszik programok felvételét a könyvtárba és törlését a könyvtárból. Ezek a programok általában futásra készen a háttértárakon vannak, ahonnan az operatív memóriába hozva végrehajthatók.
3. *Fordítóprogramok:* egy adott nyelven megírt, kódolt programokat a gép által érthető gépi utasításokká alakítják át. (Csak olyan nyelven programozhatunk, amelyhez találunk gépünk operációs rendszerében fordítóprogramot.)

2. A feldolgozási lehetőségek

Az operációs rendszert általában a számítógéppel együtt kapjuk, melynek egy részét az operatív tárban, más részét a számítógép háttértárolóján helyezik el, így bármikor rendelkezésünkre áll. Az operációs rendszer behatárolja, hogy milyen feladatok megoldására és hogyan üzemeltethetjük gépünket. Lehet, hogy a felhasználók csak egymás után használhatják gépüket, de előfordulhat, hogy egyszerre több felhasználó is használhatja ugyanazt a programot *(például*: repülőjegyek nyilvántartása) vagy különbözőt *(például*: multiprogramozás).

Az első esetben egyfelhasználós, valós idejű (real time) feldolgozásról beszélünk. Az adatokat a számítógép azonnal feldolgozza és válaszol a felhasználónak. A felhasználó és a számítógép között közvetlen, párbeszédes kapcsolat van.

A második esetben kötegelt (batch) feldolgozásról beszélünk. Ilyenkor összegyűjtik az adatokat, majd egyszerre dolgozzák fel.

3. A DOS

A személyi számítógépek legelterjedtebb operációs rendszere a DOS (Disk Operating System), a lemezes működtető rendszer. A programjaink többsége mágneslemezen van, ezért a mágneses háttértárak elengedhetetlen részei a számítógépnek. A DOS egyszerre csak egy felhasználót szolgál ki, egyszerre csak egy programot futtat, ezért egyfelhasználós operációs rendszernek nevezzük. (A DOS a számítógép számára mindig rendelkezésre áll valamilyen háttértáron. Kis gépeknél, például: C-64-nél a ROM-ba helyezik el az operációs rendszert.)

Olvasmány

A számítógépek fejlődésével tökéletesedett a DOS *is, mind több lehetőséget biztosítva a felhasználó számára. A különböző állomásokat verziószámokkal jelölik. Az első a* DOS 1.0 *volt, majd a* 2.0 *jelentkezett. A* 3.0*-nak továbbfejlesztett változatai a* 3.10, *a* 3.20. *és a* 3.30. *Újabb változat az* 5.0.

A DOS verziói felülről kompatíbilisek, tehát a régebbi programok futnak az újabb verzióval is. Ezért nem szokták a DOS-t beégetni a ROM-ba, mert lemezről betöltve a fejlettebb változatok jobban kihasználják a hardver lehetőségeit.

a) A DOS részei

1. A DOS magja a ROM BIOS az operatív memória ROM-jában elhelyezkedő, csak olvasható programok együttese, amelyet beépítenek a számítógépbe, ezért annak alkotórészévé válik.
   Részei:
   — bekapcsolási önteszt, amely ellenőrzi a memória és a perifériák működését,
   — betöltő vagy beolvasó rekord, amely betölti a DOS következő részét a memóriába, majd indítja azt,
   — perifériakezelő rekord.
2. A rezidens rész azoknak a programoknak az összessége, amelyek a DOS indítása után betöltődnek a lemezből a tárba, és ott is maradnak, mert gyakran használjuk őket. Tulajdonképpen a ROM BIOS bővítése. Lehetővé teszi újabb perifériák működését is. A betöltő vagy boot rekord a lemez 0. szektorában van mindig, így azonosító nélkül is azonnal megtalálja a DOS. Betöltődése után aktivizálódik, és betölti a DOS rezidens részét.
   A COMMAND. COM a parancsfeldolgozó fájl. A felhasználói programok betöltését és futtatását végzi. Beolvassa, értelmezi, felismeri és végrehajtja a DOS-parancsokat.
3. Rejtett fájlok, melyek a tartalomjegyzék kiíratásakor nem jelennek meg. Ez a véletlen törlés ellen véd.
   — IBMDOS.COM — a háttértárakat vezérli, az alapvető I/0 műveleteket végzi.
   — IBMBIO.COM — a ROM BIOS elkészítése óta elavult részeket javítja.
4. A külső parancsokat tartalmazó fájlok csak szükség esetén kerülnek a tárba. Külső parancs kiadásakor betöltjük a programot, majd futtatjuk. A parancsszó a fájl neve.

b) **Az adatok tárolása**

A logikailag összefüggő adatok halmazát állománynak vagy fájlnak (file) nevezzük. Ezeket az egységeket a tárolás során együtt kezeljük. Az összetartozó információk lehetnek egy program utasításai, adatok vagy bármi, amit az alkalmazó egy logikai csoportba sorol.

Ahhoz, hogy a DOS megtalálja a lemezen a keresett állományt, valahogyan azonosítani kell, nevet kell adni neki. Az állomány azonosítója két részből áll.

1. Fájl név — maximum 8 karakter hosszúságú név, melyben betűk, számok és néhány megengedett karakter lehet. A nevet tetszőlegesen választhatjuk. Célszerű olyan elnevezés, amely utal a fájl tartalmára.
2. Típusjelzés vagy kiterjesztés — maximum 3 karakter hosszúságú, de el is maradhat. Betűt, számot vagy néhány további jelet tartalmazhat. Az operációs rendszer számára fontos jelentése van az .EXE, .COM, .BAT kiterjesztésűeknek; ezeket be tudja hívni és futtatni tudja. A kiterjesztést rendszerint a program rendeli az állománynévhez.

A fájlnevet és a kiterjesztést . választja el egymástól.

Például: COMMAND.COM.

A név megválasztásakor ügyeljünk arra, hogy azok különbözzenek egymástól!

Például: PROGRAM1 *nem azonos* PROGRAM2-*vel.*

A fájlok legfontosabb jellemzőit a lemez tartalomjegyzékében tartjuk nyilván, amelyet bármikor kiírhatunk a képernyőre a DIR parancs segítségével.

F) A számítógép üzembe helyezése

1. PC-k esetén

A számítógép működésénél nagyon fontos a bekapcsolási sorrend, mert ennek betartásával növelhetjük a gép élettartamát.

A bekapcsolási sorrend

— Elhelyezzük a hálózati csatlakozókat, összekapcsoljuk a perifériákat az alapgéppel, majd ellenőrizzük a nyomtatóban lévő papírkészletet.
— Bekapcsoljuk a perifériákat.
— Monitor bekapcsolása (ha külön kapcsolódik).
— Alapgép bekapcsolása.

A ***kikapcsolási sorrend*** ennek a fordítottja.

Olvasmány

Régebbi típusú merevlemezes tároló alkalmazásakor a kikapcsolás előtt az író-, olvasófejet egy megfelelő programmal olyan helyre kell állítani, ahol nem okoz kárt, ha elmozdítás esetén hozzáér a lemezhez. Az újabb merevlemezek maguk végzik el ezt a feladatot. Kikapcsolás után ne feledjük el letakarni a billentyűzetet, mert az a porosodásra nagyon érzékeny.

Ha a gépet kikapcsoljuk, akkor az újrakapcsolásig várjunk 10–15 *másodpercet az önindukció káros hatása miatt.*

A bekapcsolás utáni események

A számítógép és perifériáinak bekapcsolása után

a) a gépbe beépített tesztelőprogram lefut, ez ellenőrzi a memóriát, valamint a géphez kapcsolt egységeket,

b) ha nincs semmi probléma, nincs hibaüzenet, akkor a számítógép megkeresi az operációs rendszert, ami általában a DOS-nak valamelyik változata.

Először az A: jelű floppyról próbálja betölteni a rendszert. Ha a meghajtóban van lemez, de az nem tartalmazza az operációs rendszert, akkor a képernyőn hibaüzenet jelenik meg.

Non — systemdisk or disk error
Replace and strike any key when ready

Jelentése:

Nem rendszerlemez vagy lemezhiba.
Cserélje ki, és nyomjon le egy tetszőleges billentyűt, ha kész!

A hibát könnyen elháríthatjuk, ha olyan lemezt teszünk az A: meghajtóba, amely tartalmazza a rendszert vagy egyszerűen kivesszük a lemezt. Az utóbbi esetben a C: jelű winchesteren keres tovább a gép. Legtöbbször itt meg is találja a rendszert, amit a kezdeti állapot beállításakor vittek fel a lemezre. Megkezdődik a program betöltése, így a gép hamarosan üzemkész állapotba kerül.

c) A DOS betöltése után a számítógép készenléti jel, úgynevezett **prompt** kiírásával a monitoron jelzi a felhasználónak, hogy készen áll a parancsok fogadására. A **prompt** mögött villog egy jel, a kurzor, amely mutatja a következő bebillentyűzendő karakter helyét.

2. Commodore-64 esetén

A bekapcsolás

a) A hálózati csatlakozók elhelyezése és a televízió összekapcsolása a géppel.

b) Televízió bekapcsolása.

c) Ha használunk perifériát
— mátrix printer bekapcsolása,
— mágneslemez-meghajtó bekapcsolása.

d) Számítógép bekapcsolása (hálózati kapcsoló a jobb hátsó oldalon van). A bekapcsolt állapotot egy piros színű világító lámpa mutatja.

A *kikapcsolási sorrend* ennek fordítottja. Ha a gépet kikapcsoljuk, akkor az újrakapcsoláskor várjunk 10–15 másodpercet az önindukció káros hatása miatt.

Vigyázzunk, hogy a gépek csatlakozóihoz ne érjen fémes tárgy *(például:* töltőtoll), mert rövidre zárhat.

A számítógép bekapcsolása után a képernyőn a következő felirat jelenik meg:

```
**** COMMODORE 64 BASIC V2 ****
64 K RAM SYSTEM 38911 BASIC BYTE FREE
READY
■
```

A READY felirat jelzi, hogy a számítógép bevitelre vár. Látunk egy villogó négyzetet is, a „kurzort", amelyet a számítógép kiír minden utoljára beírt karakter mögé. Ahhoz, hogy a számítógép végrehajtsa parancsainkat, utasításainkat, azaz, hogy ezek bekerüljenek a számítógép memóriájába, le kell nyomnunk a RETURN billentyűt, mert ez jelzi a sor végét (új sor következik). A billentyű lenyomása után READY felirat jelenik meg.

Tehát a gép készen áll arra, hogy végrehajtsa utasításunkat.

Olvasmány

A Commodore-64-*et egyes feladatok ellátására mágneses háttértár nélkül is használhatjuk, mert memóriájának ROM részébe beépítették a működtető operációs rendszert. Bekapcsolása után nem kell keresgélni, azonnal rendelkezésre áll, ezért jelenik meg olyan gyorsan a készenléti felirat.*

A C-64 *memóriája* 64 kbyte *kapacitású, amelyből* 38 kbyte *áll a programozó rendelkezésére, mert a többi területen a rendszerprogramok helyezkednek el. A számítógéphez több perifériát is csatlakoztathatunk. Például: kazettás magnót, nyomtatót, floppymeghajtót, botkormányt (joystick).*

A Commodore-64 az egyes perifériákat fizikai egységszám alapján azonosítja
— kazettás egység: 1,
— nyomtató: 4 vagy 5,
— mágneslemez-meghajtó: 8–15 (gyárilag 8).

Ha egységszám nélkül fordulunk a perifériákhoz, akkor a számot a számítógép automatikusan egynek veszi.

a) **A programok betöltése, tárolása és ellenőrzése**

Mielőtt hozzákezdenénk a programok betöltéséhez, illetve felvételéhez, győződjünk meg arról, hogy a kazettás egységet, illetve a lemezes meghajtóegységet helyesen csatlakoztattuk-e a számítógéphez!

A kazettás háttértár használata

— *A program behívása:*
A kazettát megfelelő helyre tekerjük és beírjuk:

LOAD "PROGRAMNÉV", 1 (A magnón minden gombot engedjünk fel!)

Ha elfelejtettük a program nevét, akkor csak a LOAD parancsot gépeljük be, nyomjuk le a RETURN billentyűt! Ekkor az első program automatikusan betöltődik. A képernyőn a következő kiírást látjuk:

PRESS PLAY ON TAPE (Nyomja le a lejátszás gombot!)

Ha lenyomjuk a PLAY gombot, eltűnik a felirat és a kurzor a képernyőről, a szalag forog, megjelenik a következő üzenet:

FOUND programnév (megtalálta a programnevet).

A gép akkor kezdi betölteni a tárba a programot, ha lenyomjuk a C= gombot. A futást megszakíthatjuk a RUN/STOP billentyűvel.
A program betöltését akkor fejezi be a gép, ha a képernyőn újból megjelenik a kurzor és a READY felirat.

— *A program rögzítése*
A számítógépben lévő program tárolásához a következő parancsot kell beírni:

SAVE "PROGRAMNÉV", 1

A programnév tetszőlegesen választott 16 karakternél rövidebb elnevezés lehet.
A RETURN lenyomása után a

PRESS PLAY AND RECORD ON TAPE

(Nyomja le a lejátszás és a felvétel gombot!) kiírás jelenik meg a képernyőn. Az utasítás végrehajtása után eltűnik a felirat és a kurzor is a képernyőről. Új utasítást csak a felvétel befejezése után adhatunk a gépnek.

— *A program rögzítésének ellenőrzése*
A számítógépben lévő és a szalagra felvett programot összeegyeztetni a

VERIFY "PROGRAMNÉV", 1 paranccsal tudjuk.

A lemezhasználat

A lemezt nagyon óvatosan kezeljük! Feleslegesen a védőtokjából ne vegyük ki! Kézzel ne érjünk a védőborító nyílásához! TILOS a meghajtó ki- és bekapcsolása behelyezett lemezzel! A lemez behelyezése és kivétele csak leállt hajtóműnél történjék!

— ***A program behívása***
Nyissuk fel a meghajtóegység ablakát, és a lemezt dugjuk a nyílásba a címkés oldalával felfelé, úgy, hogy a címke a hozzánk közelebbi felén legyen! Minden lemezen található egy kis négyzet alakú bevágás, amelynek most a lemez bal oldalán kell lennie. A meghajtó ajtajának becsukása után írjuk be:

LOAD "PARANCSNÉV", 8

Nyomjuk le a RETURN billentyűt!
A meghajtóegység elindul, megjelenik a következő üzenet:

SEARCHING FOR PROGRAMNÉV

(A program keresése az adathordozón)

LOADING (betöltés)
READY

Ezután a programot a RUN parancs segítségével futtathatjuk.

— ***A program rögzítése***
Gépeljük be a következő parancsot!

SAVE "PROGRAMNÉV", 8

Nyomjuk le a RETURN billentyűt!
Ha a gép a tárolást rendben elvégezte, akkor a következő kiírás jelenik meg a képernyőn:

SAVING PROGRAMNÉV
OK
READY

— ***A program rögzítésének ellenőrzése***

VERIFY "PROGRAMNÉV", 8

A parancs és a RETURN billentyű lenyomásának hatására történik az ellenőrzés.
Ha az új lemezt behelyezzük, utána inicializáljuk (megnyitjuk a lemez parancscsatornáját, majd a CLOSE segítségével zárjuk a megnyitott adatállományt).
Ha meg akarjuk nézni, hogy mi van a lemezen, akkor adjuk ki a

LOAD "$", 8 parancsot.

A LIST parancs kiírja a lemezen rögzített fájlokat.

A nyomtató használata

A nyomtatás indítása az OPEN 4, 4 : CMD 4 : LIST RETURN paranccsal történik, majd a listázás után

```
PRINT 4 RETURN
CLOSE 4 RETURN
```

parancsokat adjuk ki.

A listázás után feltétlenül alkalmazzuk ezt a két parancsot, mert elhagyásuk később hibás működést eredményezhet.

b) **Ismerjük meg a számítógépünket!**

Az előzőekben megismertük a számítógép felépítését, működését és bekapcsolását. Próbáljuk ki a billentyűzetet! Írjunk be egy tetszőleges szöveget! A kurzormozgató billentyűk segítségével mozogjunk a szövegben! Alkalmazzuk a speciális billentyűket, és próbáljunk karaktereket törölni, illetve beszúrni! A billentyűzet helyes alkalmazásával, lehetőségeinek kihasználásával gyorsíthatjuk munkánkat mindkét számítógép esetén.

Kérdések és feladatok

1. Milyen kapcsolatban lehet a felhasználó és a számítógép?
2. Mit jelent a számítógépek kompatibilitása?
3. Mi a vezérlőbillentyűk szerepe?
4. Mit nevezünk kurzornak?
5. Milyen billentyűkkel mozgathatjuk a kurzort?
6. Mit nevezünk operációs rendszernek?
7. Mi a szerepe az operációs rendszernek?
8. Milyen részei vannak az operációs rendszernek?
9. Mit jelent a DOS?
10. Milyen részekből áll a fájl azonosítója?
11. Miért kell ügyelnünk a számítógép üzembe helyezésekor a bekapcsolási sorrendre?
12. Milyen sorrendben kell a bekapcsolást elvégeznünk? Mutassuk be a gyakorlatban is!
13. Milyen szerepe van a RETURN billentyűnek?
14. Miben segít bennünket a C= billentyű?
15. Milyen hatása van a RUN/STOP billentyűnek?
16. Hogyan történik a mágneslemezen tárolt program beolvasása?
17. Próbáljunk ki egy általunk bejátszott játékprogramot!
18. Vegyük fel mágneslemezre az előbbi játékprogramunkat! Milyen lépéseket kell betartanunk?
19. Ellenőrizzük felvett programunkat!
20. Hogyan győződhetünk meg arról, hogy pontosan vettük-e fel a programot?

IV. A PC-k operációs rendszere

Ebben a fejezetben az IBM PC-vel és az általa használt operációs rendszerrel foglalkozunk részletesebben.

A) Az operációs rendszer használata

Az MS-DOS 5.0 operációs rendszert használhatjuk parancssor üzemmódban, és az úgynevezett Shell-program segítségével, amely menüvezérelt felhasználást tesz lehetővé (nem tárgyaljuk).

A parancssor üzemmódban a készenléti jel mögé a felhasználó begépel egy parancsot az operációs rendszer parancsai közül. Az ENTER lenyomása után a számítógép értelmezi a parancsot, és elindítja a parancsnak megfelelő program futását. A parancsok megadásával aktivizáljuk a DOS funkcióit.

B) A készenléti jel

A számítógép bekapcsolása után a képernyőn megjelenő készenléti jel vagy prompt jelzi a felhasználónak, hogy a számítógép készen áll a parancsai fogadására és végrehajtására. Ez a jel a képernyőn az utolsó megkezdett sor elején látható.

Kapcsoljuk be számítógépünket! Ügyeljünk a sorrendre! Ha minden rendben van, akkor alaphelyzetben a prompt mutatja az aktuális meghajtó betűjelét.

A:> _ vagy C:> _

A prompt után villog a kurzor, amely mutatja, hogy oda írhatjuk be a parancsot.

C) A DOS-ban kiadott parancsok formája

A DOS-ban kiadott parancsok formája a következő:

A prompt után: parancsnév ␣ [paraméterek]/[kapcsolók]

— A parancsnak azokat a részeit, amelyeket nem kötelező megadni, [ ] közé szoktuk tenni.

— A drive jele: merevlemez esetén: C: vagy D:
floppy jele: A: vagy B:

— A paraméter lehet: drive jele, fájlnév, alkönyvtár elérési útvonala.

— A kapcsoló(k): betű vagy szám.

— ␣ a szóköz jele, space.

D) Elemi parancsok

Vannak olyan elemi parancsok, amelyeket paraméterek és kapcsolók nélkül alkalmazhatunk. Ismerkedjünk meg először ezekkel a parancsokkal, mert ezek lehetővé teszik számunkra, hogy közelebb kerüljünk a számítógéphez.

1. A készenléti jel (prompt) beállítása

Ha DOS-sal dolgozunk, akkor a készenléti jel állandóan a képernyőn van, ezért érdemes úgy beállítanunk, hogy minél több információt adjon számunkra. Nézzük meg, hogy milyen lehetőségeink vannak a megváltoztatására.

A parancs formája: prompt ␣ [szöveg]

— Ha nem használunk szöveget, akkor az alaphelyzet áll elő. Ez az A:> jel.

— Ha a $ jellel kezdjük a szöveget, akkor nem a beírt szöveget írja ki számítógépünk, hanem annak a funkciója jelenik meg a képernyőn.
Például az aktuális dátum.
Parancs: prompt ␣ $ d
Új prompt: TUE 10-27-1992

— Ha elhagyjuk a $ jelet, akkor a beírt szöveg jelenik meg.
Parancs: prompt ␣ d
Új prompt: d

A $ jellel kezdett szövegbe írt karakterek jelentése:

d	az aktuális dátum kijelzése
t	az aktuális idő kijelzése
$	dollárjel
=	egyenlőségjel
p	aktuális meghajtó és könyvtár kijelzése
v	MS-DOS verziószáma
n	aktuális meghajtó kijelzése
g	„>" nagyobb jel
l	„<" kisebb jel
—	a kurzor a képernyő következő sorának elejére kerül

Példák

Változtassuk meg a készenléti jelet!

Eredeti képernyőkép	*Új képernyőkép*
`A:>PROMPT ␣ $l$d$g`	`<TUE 10-27-1992>`
`<TUE 10-27-1992>PROMPT ␣ $l$t$g`	`<16:47:14.95>`
`<16:47:14.95>PROMPT ␣ $l$d$ ␣ $t$g`	`<TUE 10-27-1992`
`16:47:14.95>`	
`<TUE 10-27-1992`	
`16:47:14.95>PROMPT ␣ $l JÁNOSKA $g`	`<JÁNOSKA>`
`<JÁNOSKA>PROMPT ␣ $n$g`	`A>`

Készítsünk tetszőleges promptot önállóan is!

Nagyon sokféle promptot állíthatunk be, de melyik az, amely számunkra legjobban használható? Általában azt a formát alkalmazzuk, amely biztosítja, hogy a könyvtárak között könnyen kiigazodjunk, ne tévedjünk el, tudjuk, hogy melyik alkönyvtárban vagyunk, és hogyan jutottunk oda. Állítsuk be úgy a készenléti jelet, hogy az aktuális meghajtó és az aktuális könyvtár is benne legyen!

Parancs: prompt ␣ lp$g
Hatása: <A:\>

A készenléti jelből megállapíthatjuk, hogy pillanatnyilag a floppy, az A: meghajtó főkönyvtárában vagyunk.

2. Képernyőtörlés

Amikor a számítógépet elkezdjük használni, legtöbbször zavar bennünket, hogy a képernyőn más is van, mint ami az utolsó parancsunk hatására megjelenik. Az előző kifejezéseket „letörölhetjük" a képernyőről a CLS paranccsal.

A CLS parancs törli a képernyőt, csak a készenléti jel és a kurzor marad meg a képernyő bal felső sarkában, minden más eltűnik. A

C:\>CLS *hatása:* C:\>_

3. A dátum és az idő szerepe, beállítása

Amikor a DOS az operatív tárból a háttértárak valamelyikébe viszi az adatállományt, akkor az állomány neve mellé írja az érvényes rendszerdátumot és a rendszeridőt. Mindez lehetőséget nyújt számunkra, hogy a tartalomjegyzékből megtudjuk az utolsó beírás vagy módosítás idejét. Erre olyan esetekben lehet szükségünk, amikor többen dolgoznak a számítógépen vagy naprakész könyvelést vezetünk. Egyes géptípusoknál minden bekapcsolás után be kell állítani ezeket az adatokat. A legújabbaknál ez már nem jelent gondot, mert működik bennük egy akkumulátorról táplált kvarcóra, amely kikapcsolt állapotban is megőrzi a dátum- és az időadatokat. Ebben az esetben is lehetőségünk van beállításukra, de itt csökken a szerepe.

Megjegyzés
Olyan számítógépeknél, ahol szükség van az idő és a dátum beállítására, érdemes az AUTOEXEC.BAT fájlba beírni ezeket a parancsokat, hogy ne feledkezzünk el róla. Ilyenkor a rendszer indításakor automatikusan kéri tőlünk a gép az adatokat.

Az adatok bevitelénél oda kell figyelnünk a beírás sorrendjére.
Az idő beállításakor a szokásos sorrendet kell követnünk:

óra — 0–23-ig terjedő számok,
perc — 0–59-ig terjedő számok,
másodperc — 0–59-ig terjedő számok.

A TIME parancs megjeleníti a pontos időt, és módot ad a változtatására is.

Parancs: TIME
Hatása: Current time is: 11:32:43 *(Jelenlegi idő.)*
Enter new time: _ *(Adja meg az új időt!)*

A kurzor helyére beírhatjuk a pontos időt, de ha a kijelzett időt elfogadjuk, akkor az **Enter** lenyomása után továbbléphetünk.

Példa

a) *Új időként írjuk be:* 21:06:26
Nézzük meg, hogy elfogadja-e számítógépünk az adatot!
Parancs: TIME
Jelenlegi idő: 9:06:36:72p
Elfogadta az adatokat, de délután 9-nek írta ki.

b) A hétköznapi életben mi sem szoktunk 17 óráról beszélni, hanem például délután 5-ről.
Írjuk be így az adatot! 9:07:16p
Ellenőrzés után láthatjuk, hogy így is elfogadta gépünk az adatot.

A dátum megadásának formáját az határozza meg, hogy a rendszer kiépítésekor milyen ország írásmódját vették figyelembe. Leggyakrabban az amerikai formátumot alkalmazzák.

A sorrend a következő:

mm – dd – [yy]yy; azaz : hónap — nap — év,
mm: a hónap sorszáma 0–12 közötti egész szám,
dd: a nap sorszáma 1–31 közötti egész szám,
yy: az év évszázad nélkül 0–99 közötti egész szám.
(A XXI. században az évszázadot is be kell írni.)

A DATE parancs hatására a képernyőn megjelenik az aktuális dátum, és itt is lehetőségünk van a módosításra.

Parancs: DATE
Hatása: Current date is Thu 11-26-1992
(Jelenlegi dátum.)
Enter new date (mm-dd-yy)_
(Adja meg az új dátumot adott formában!)

Megadhatunk új dátumot, de az **ENTER** lenyomásával változtatás nélkül továbbléphetünk.

Példa

a) *Új dátumként írjuk be születési évünket! Például:* 02-03-87

b) *Nézzük meg, hogy mi történik akkor, ha olyan adatokat írunk be, amelyek nem fordulhatnak elő!*
Például: 23-14-98 *beírása után megjelenik:*
INVALID DATE *(Érvénytelen dátum.)*
Enter new date (mm-dd-yy):_
(Írjon új dátumot!)
02-29-91 *esetén ugyanez a helyzet, mert nem volt szökőév az* 1991. *év.*

Mindaddig nem fogadja el gépünk az adatot, amíg nem a megfelelő határokat adjuk meg.

c) *Figyeljünk a beírásra, mert könnyen lehet december 10-ből október 12. Ezt elfogadja a gép.*

Mindkét parancsot beírhatjuk azonnal az aktuális értékkel együtt. A DOS ellenőrzés után beállítja a megadott értékeket, és nem ír ki üzenetet.

Parancs: TIME ␣ 9:17:16p
Hatása: C:\> _

Ellenőrizzük, hogy elfogadta-e gépünk a beírt parancsot!

Parancs: TIME

Dátum beírásakor hasonlóan járhatunk el.

Parancs: DATE ␣ 04-18-91
Hatása: C:\> _

4. A verziószám lekérdezése

Előfordul, hogy kölcsönkapunk egy programot, amelyet futtatni szeretnénk egy idegen gépen. Ezt csak ugyanabba vagy magasabb verziószámú rendszerbe tartozó DOS alatt tehetjük meg. Ilyenkor a VER parancs alkalmazásával kiírathatjuk az alkalmazott operációs rendszer verziószámát.

Parancs: VER
Hatása lehet: MS-DOS Version 5.00

E) A könyvtárszerkezet

Napjainkban a kész programok floppyn kerülnek a felhasználóhoz. Gyakran előírják, hogy milyen könyvtárszerkezetbe helyezzük őket. Előfordul, hogy winchesterünkre újabb programokat viszünk fel, esetleg régebbieket törlünk. Szükségünk lesz arra, hogy tudjuk módosítani könyvtárunkat.

1. A könyvtárszerkezet leírása

A lemezek kapacitásának növekedése szükségessé tette, hogy a tartalomjegyzéket főkönyvtárra és alkönyvtárakra osszuk fel a fájlkeresés meggyorsítása érdekében. Egy könyvtárban több alkönyvtárat hozhatunk létre, de ezek semmilyen oldalirányú kapcsolatban sem lehetnek egymással; csak ahhoz az egyhez kapcsolódhatnak, amelyikből leágaznak. Ez a garancia arra, hogy a DOS pontosan azonosítani tudja a keresett programot.

Ez a rendszer úgy működik, mint egy valódi könyvtár, amelyből könyvet akarunk kölcsönözni. Kölcsönözzük ki a tankönyvünket! Hogyan találjuk, hol

keressük? Tudjuk, hogy ez szakirodalom, természettudományi jellegű, számítástechnikával foglalkozó tankönyv.

Felvázoljuk gondolatmenetünket a *7. ábrán!*

7. ábra

A vastag vonal mutatja azt az utat, amelyet megtettünk, amíg megtaláltuk a könyvünket. Ugyanígy közlekedünk lemezünkön is. Ott is fel kell sorolni azokat az alkönyvtárakat, amelyeken keresztül eljutunk a keresetthez. Így adhatjuk meg az útvonalat.

Az összes alkönyvtárat tartalmazó kiindulási könyvtárat főkönyvtárnak vagy gyökérkönyvtárnak nevezzük. (Az egész hierarchikus rendszer egy fához hasonlít, ezért szoktuk fastruktúrának is nevezni.) A főkönyvtár jele: \ Az útvonal leírásánál is ezzel a jellel választjuk el az egyes állomásokat. Esetünkben ez a következő:

C:\szakirodalom\természettudomány\számítástechnika\tankönyvek\mi

Vigyázzunk, mert a \, illetve / jel jelentése különböző!

Létrehoztunk egy logikai szerkezetet, amelyben tetszőleges fájlokat helyezhetünk el. *Például a természettudomány könyvtár alkönyvtáraiban lehet a kémia vagy a meteorológia nevű könyvtár is.* A következőkben megtanuljuk, hogy hogyan helyezhetjük el a programokat a lemezünkön.

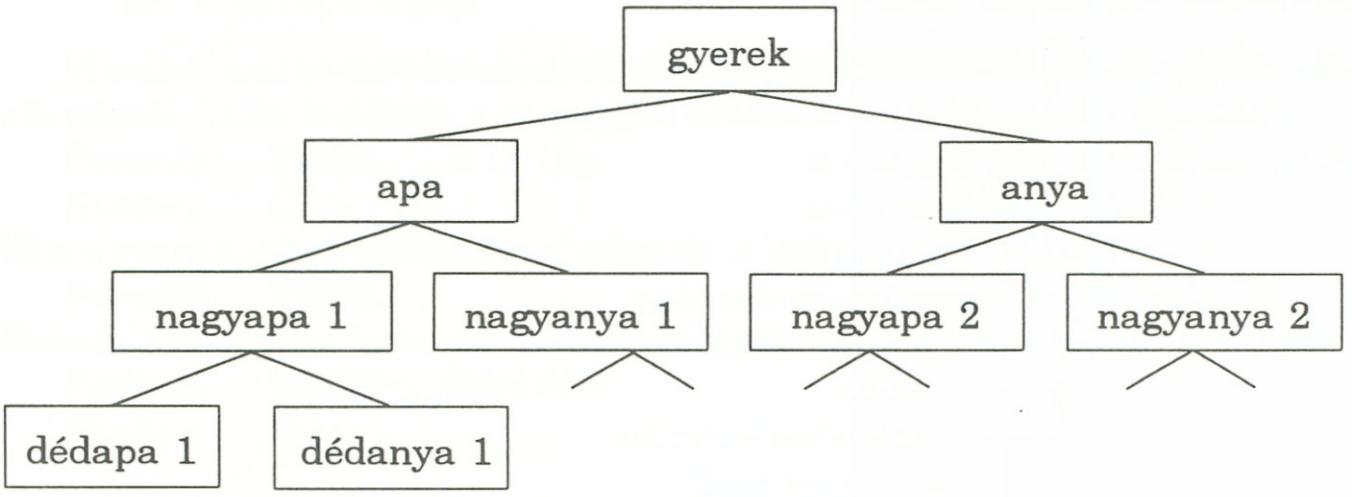
Olvasmány

A könyvtárrendszert a gráfelméletből ismert fához hasonlíthatjuk. Mit hallottunk a gráfról?

A gráf olyan pontok halmaza, amelyekből kiinduló egy vagy több vonal bizonyos pontpárokat köt össze.

A fa csúcsokból és a köztük lévő élekből álló olyan rendszer, amelyben bármely két pontot pontosan egy út köt össze, és éleinek száma eggyel kevesebb a szögpontok számánál.

A szerkezet megértéséhez rajzoljuk le családfánkat (8. ábra)!



8. ábra

F) A DOS könyvtárakkal kapcsolatos parancsai

1. A tartalomjegyzék kiíratása

A gép bekapcsolása, majd a rendszerprogramok lefutása után a képernyőn megjelenik a prompt, mely jelzi, hogy gépünk parancsra vár.

A\:>

Ez a jel azt mutatja, hogy az A: meghajtó, vagyis a floppy főkönyvtárában vagyunk. Nézzük meg, hogy mi van a lemezünkön!

Írjuk be a DIR vagy dir parancsot! A számítógép számára mindkettő ugyanazt jelenti, mindegy, hogy kis- vagy nagybetűvel írjuk. Ne feledjük el a parancs végén lenyomni az ENTER billentyűt, mert csak így kezdi értelmezni gépünk a beírt parancsot.

A képernyő képe: A:\>DIR

Az ENTER gomb lenyomása után a képernyőn megjelenik az aktuális könyvtár tartalomjegyzéke, amely a következőkhöz hasonlít:

```
Volume in drive A is AGOTA
Volume Serial Number is OF 36-19CF
Directory of A:\
COMMAND.COM 47845 04-10-92 9:49a
        1 file(s)        47845 bytes
        1094144 bytes free
```

Értelmezzük a képernyőn megjelenő szöveget!

Az A: jelű meghajtóban lévő lemez neve: AGOTA.

A lemez szériaszáma: OF 36-19CF.

Az A: jelű egységben található lemez gyökérkönyvtárának tartalomjegyzéke.

A következőkben a programok felsorolását láthatjuk.
Értelmezzük az oszlopok tartalmát!

1. oszlop:	COMAND	az adatállomány neve,
2. oszlop:	.COM	az állomány kiterjesztése (nem kötelező),
3. oszlop	47845	az állomány hossza byte-okban megadva,
4. oszlop:	04-10-92	az állomány létrehozásának, illetve utolsó módosításának dátuma,
5. oszlop:	9:49a	az állomány létrehozásakor, illetve utolsó módosításakor hány óra volt.

Az elkészülés ideje után találunk egy betűt: **a** — délelőtt, **p** — délután készült a program.

A dátumot ebben a példában a német kiírásnak megfelelően láthatjuk (nap-hónap-év), ez azonban megváltoztatható. A dátumot mi is állíthatjuk.

Például: ha el akarjuk kerülni a péntek 13 vírust, előreállítjuk a dátumot. Ilyenkor az amerikai vagy angol beírási formát alkalmazzuk (hónap-nap-év).

A tartalomjegyzék alkönyvtárat is tartalmazhat. Ilyenkor a méret helyén <DIR> látszik.

Például: TERV < DIR > 09-07-92 10:32a

A tartalomjegyzékünk nemcsak egy-két állományt, alkönyvtárat tartalmazhat, hanem többet is, így a lista legtöbbször nem fér a képernyőre, az elejét nem láthatjuk. A következő kapcsolók segítségével változtathatunk ezen, mert lehetővé teszik számunkra, hogy az egész tartalomjegyzéket lássuk.

DIR / p — hatására egy képernyőnyi lista íródik ki. Ha szeretnénk folytatni a kiíratást, akkor le kell nyomni egy tetszőleges gombot (csak előre lapozható).

DIR / w — hatására tömörített listát kapunk. Számítógépünk öt oszlopba, egymás mellé írja ki az állományneveket és kiterjesztésüket.

DIR / p / w — ezt a két kapcsolót olyan esetben használjuk, ha a tömörített lista sem fér a képernyőre.

Feladat

Tegyél be egy másik lemezt, és írasd ki a tartalmát! Alkalmazd az említett kapcsolókat! Képernyődön megjelenhet például hasonló a Dir/w parancs beírása esetén.

```
FORMAT.COM    BASIC.EXE  TREE.COM     [DOS]  EXAC.COM
[PROBA]       CONFIG.SYS COMMAND.COM [DOSH] [UTIL]
SZOTAR.BAT
```

Lehetőségünk van arra is, hogy olyan állományokat is könnyen megtaláljunk, amelyeknek nem ismerjük pontosan a nevét, csak egyes betűket ismerünk belőle, vagy csak a kiterjesztésére emlékszünk. Ebben a joker vagy helyettesítő karakterek segítenek bennünket.

a) * karakter helyettesíti az összes mögötte álló, még hátralévő karaktert,
b) ? csak azt az egyetlen karaktert helyettesíti, amelynek a helyén áll.

Példák

Tegyük fel, hogy a C: meghajtó főkönyvtárában vagyunk! Írassuk ki az aktuális könyvtár tartalomjegyzékének egyes részeit helyettesítő karakterek segítségével!

Parancs és hatása:

```
DIR ␣ *.EXE
```

Kiírja a főkönyvtár összes EXE kiterjesztésű állományát.

```
DIR ␣ PELDA.*
```

Kiírja a főkönyvtár összes PELDA nevű állományát, mindegy, hogy mi a kiterjesztése.

```
DIR ␣ PEL*.*
```

Kiírja a főkönyvtár összes olyan állományát, amely PEL-lel kezdődik.

```
DIR ␣ P?LDA.*
```

Kiírja a főkönyvtár összes olyan állományát, amely P-vel kezdődik és LDA-val fejeződik be, és közben 1 betű bármi lehet.

```
DIR ␣ *.* vagy DIR
```

Kiírja a főkönyvtár összes állományát, akármi a neve és a kiterjesztése.

A parancs általános formája:

DIR [meghajtó] [\ útvonal] [\ állománynév] [/p] [/w]

Hatására a kijelölt meghajtó adott útvonallal elért alkönyvtárából megjelenik a képernyőn megadott nevű állomány.

Feladat: Gyakoroljuk a különböző kiíratási lehetőségeket! Ellenőrizzük, hogy az előzőekben tárgyalt parancsok hatása megfelel-e a leírtaknak!

(A PELDA nevű állomány nevét helyettesítsük értelemszerűen olyannal, amilyen van a lemezünkön.)

2. Belépés az alkönyvtárba

Eddig a főkönyvtárban voltunk, de szeretnénk átlépni valamelyik alkönyvtárba, hogy így is megnézzük a tartalmát.

A parancs:

CD ␣ útvonal *vagy bővebben* CH DIR ␣ útvonal

A parancs és az alkönyvtár neve között legalább egy szóköz helyet kell kihagynunk (jele:␣).

A főkönyvtárban vagyunk, ezért elhagytuk a jelét. Bármely esetben lehet az útvonal leírását a főkönyvtártól kezdeni. Ha nem az aktuális lemezegységről van szó, akkor a lemezegység nevét is be kell írnunk.

Példa

a) Lépjünk be a DOS nevű alkönyvtárba!
Parancsunk lehet: CD ␣ DOS *vagy* CD ␣ \DOS *vagy* CD ␣ C:\DOS

Az **ENTER** lenyomása után megváltozik a képernyőn a prompt, jelzi, hogy beléptünk az alkönyvtárba.

`C:\DOS>` jelenik meg a képernyőn.

b) Írassuk ki a DOS alkönyvtár COM kiterjesztésű állományait!

```
DIR ␣*.COM
```

A képernyőn megjelenhet:

```
    Volume in drive C is MS-DOS-5
    Volume Serial Number is 18D6-7D36
    Directory of C:\DOS
FORMAT  COM 32911 06-01-91   1:00p
EDIT    COM 413   06-01-91   1:00p
COMMAND COM 47845 06-01-91   1:00p
TREE    COM 6901  06-01-91   1:00p
SYS     COM 13440 06-01-91   1:00p
5 file(s)   101510 bytes
          61780584 bytes free
```

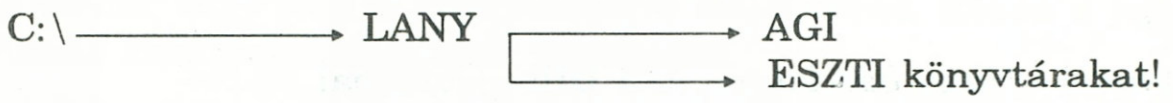
Feladat: A DOS alkönyvtár állományainak felhasználásával gyakoroljuk a kiíratási parancsot!

3. Alkönyvtár létrehozása

Gyakran kerülünk olyan helyzetbe, hogy új, üres alkönyvtárat szeretnénk létrehozni. Megtehetjük ezt, ha pontosan megadjuk a helyhez vezető útvonalat, és ha az adott helyen még nincs ilyen nevű alkönyvtár.

Példák

1. Hozzuk létre a



Alkalmazzuk az MKDIR, röviden MD parancsot, és írjuk be a C:\> után:

```
MD␣LANY
```

Az aktuális könyvtár, a főkönyvtár alatt létrehoztuk a LANY nevű alkönyvtárat. A rendszer abban a könyvtárban maradt, ahonnan elindult. Lépjünk be a LANY nevű alkönyvtárba!

```
CD␣LANY
```
a prompt C:\LANY> lesz.

Hozzuk létre az alatta lévő AGI, illetve ESZTI alkönyvtárat!

```
MD␣AGI
```
a prompt C:\LANY> marad.

```
MD␣ESZTI
```
a prompt C:\LANY> marad.

Írassuk ki a LANY nevű alkönyvtár tartalomjegyzékét!
Parancs: DIR
Az **Enter** lenyomása után a képernyőn megjelenik:

```
.       <DIR>  10-20-92    5:29 p
..      <DIR>  10-20-92    5:29 p
AGI     <DIR>  10-20-92    5:30 p
ESZTI   <DIR>  10-20-92    5:31 p
```
(Természetesen az aktuális dátumok és időpontok jelennek meg)

A rendszer két működtető állományt is ír az alkönyvtárba, hogy tudjon ott dolgozni. Ezt jelzi a ., illetve a .. .

Fontos megjegyeznünk, hogy alkönyvtárat csak létező könyvtárban hozhatunk létre.

2. Hozzuk létre a

C:\ → BARNA → FIU → AKOS könyvtárszerkezetet!

A kiadott parancsok	*A megjelenő prompt*
`md␣barna`	`C:\>`
`cd␣barna`	`C:\BARNA>`
`md␣fiu`	`C:\BARNA>`
`cd␣fiu`	`C:\BARNA\FIU>`
`md␣AKOS`	`C;\BARNA\FIU>`

4. Mozgás az alkönyvtárak között

Lépjünk be az előzőekben létrehozott könyvtárszerkezet AKOS nevű alkönyvtárába is!

CD ␣ AKOS; a prompt C:\BARNA\FIU\AKOS> lesz.

Adjuk ki a CD.. parancsot! A prompt C:\BARNA\FIU> lesz.

Parancsunk hatására egy alkönyvtárral visszaléptünk, tehát a FIU nevű alkönyvtárban vagyunk.

Adjuk ki a CD ␣ \ parancsot!
A prompt C:\> lesz.

Parancsunk hatására rögtön a főkönyvtárba léptünk. Ha közbeeső alkönyvtárba akartunk volna jutni, akkor be kellett volna írnunk azt az útvonalat, ahogyan oda mehetünk.

Példa

A főkönyvtárból lépjünk be a FIU nevű alkönyvtárba!

```
CD␣BARNA\FIU
```

A megjelenő prompt jelzi, hogy végrehajtottuk feladatunkat.

```
C:\BARNA\FIU>
```

5. Alkönyvtárak megszüntetése

A lemezen nemcsak létrehozhatunk könyvtárat, hanem felesleges alkönyvtárat törölhetünk róla az RD paranccsal, ha megadjuk a törölni kívánt alkönyvtárhoz vezető útvonalat. Figyeljünk arra, hogy csak üres alkönyvtárat törölhetünk. Ha az aktuális könyvtárból adnánk ki a törlési parancsot, akkor magunk alatt vágnánk a fa ágát, nem tudnánk hova visszatérni. A főkönyvtárat ezért nem tudjuk törölni.

Egy alkönyvtárat a következő parancs segítségével törölhetünk:

RMDIR ␣ Állománynév , *vagy röviden* RD ␣ Állománynév.

Példák

1. Szüntessük meg az AGI nevű alkönyvtárat!
   A törlés előtt győződjünk meg arról, hogy üres az alkönyvtárunk, lépjünk be és írassuk ki a tartalmát!
   A parancsok:

```
CD␣C:\LANY\AGI
DIR
```

   Ha a képernyőn a fájlnevek helyén csak ., illetve .. van, akkor üres az alkönyvtár, hozzákezdhetünk megszüntetéséhez. Lépjünk vissza az előző, LANY nevű alkönyvtárba, mert csak onnan végezhetjük el a törlést.
   A parancsok:

```
CD ..        – visszalépés,
RD␣AGI       – törlés,
DIR          – listázás.
```

   A tartalomjegyzék megjelenítésével meggyőződhetünk arról, hogy valóban törlődött az AGI nevű alkönyvtár.

2. Töröljük ki a

   C:\ → BARNA → FIU → AKOS könyvtárat!

   Tudjuk, hogy egyik alkönyvtárba sem töltöttünk semmit, üresek.

Parancs	*Prompt*
	C:\>
cd␣barna\fiu	C:\BARNA\FIU>
rd␣Akos	C:\BARNA\FIU>
cd␣..	C:\BARNA>
rd␣Fiu	C:\BARNA>
cd␣..	C:\>
rd␣Barna	C:\>

Feladat: Írassuk ki a főkönyvtár tartalmát, és győződjünk meg róla, hogy töröltük a BARNA nevű alkönytárat! Végezzük el a törlést úgy is, hogy a törlés előtt mindig meggyőződünk róla, hogy az alkönyvtár üres!

Olvasmány

Képzeljük el a könyvtárak megszüntetését, mint egy „igazi" fa darabolását! Felmászunk egy magas fa tetejére, és sorban levágjuk az ágait. Először megnézzük, hogy nincs-e valami az ágon, például egy madárfészek, mert először azt le kell vennünk a kiszemelt ágról. (Ezután lemászunk az ágról, mert ha magunk alatt vágjuk az ágat, akkor nagyot eshetünk az ággal együtt.)

Fontos megjegyeznünk, hogy a megszüntetendő alkönyvtárat először üressé kell tenni, majd ki kell lépni belőle az előző szintre, csak úgy szüntethetjük meg.

6. A könyvtárszerkezet megjelenítése

Az egyes útvonalak leírását segíti, ha szemléletesen ábrázoljuk a fastruktúrát. Lehetőségünk van arra, hogy a TREE parancs segítségével megjelenítsük a képernyőn lemezünk könyvtárszerkezetét. A

```
TREE C:\
```

parancs hatására a C: lemez könyvtárszerkezete jelenik meg.

```
C:
├→ LANY
├→ AGI
└→ ESZTI
```

Ha nem akarjuk az egészet kirajzoltatni, akkor elérési utat is megadhatunk.

```
TREE C:\LANY
```

Feladat: Rajzoltassuk ki lemezünk főkönyvtárának szerkezetét! Keressünk olyan alkönyvtárat, amelynek van újabb alkönyvtára! Lépjünk be ebbe az alkönyvtárba, és írassuk ki csak ennek a szerkezetét!

G) A DOS lemezkezelő parancsai

1. Az aktuális lemezegység megváltoztatása

Munkánk során gyakran előfordul, hogy a használandó programunk nem azon a lemezegységen van, amelyen dolgozunk. Szeretnénk a másik meghajtót aktívvá tenni.

Ilyenkor csak a lemezegység nevét kell beírni kettősponttal a prompt után.

C:	átlépünk a	C:	jelű meghajtóra
A:	átlépünk az	A:	jelű meghajtóra

Az **Enter** lenyomása után a prompt változása mutatja, hogy másik lemezegységen vagyunk.

2. A lemez formattálása

A lemez formázásakor a lemezt alkalmassá tesszük adatok tárolására. Felosztjuk sávokra és szektorokra, közben kijelöljük a tartalomjegyzék helyét. A hibás lemezterületet a DOS lefoglalja, hogy ne tudjunk írni rá. A beírt parancs a lemezről minden adatot és minden programot töröl.

Olvasmány

A lemezeken az adatokat valamilyen logikailag kialakított rendszer szerint helyezzük el, hogy a keresett információkat könnyen megtaláljuk. Az üres lemezeken ki kell alakítani azt az elrendezést, amely meghatározza, hogy az új adatokat hova, milyen formában helyezze el a lemezen a számítógép. Ezt a műveletet szoktuk formattálásnak nevezni. Végrehajtásakor az operációs rendszer egy programja sávokat és szektorokat alakít ki a lemezen. Ekkor a mágneslemezen egymástól egyenlő távolságra lévő koncentrikus körök, sávok alakulnak ki, majd a sávokat egyenlő részekre, szektorokra osztja. (A fixlemezeket ezt megelőzően particionálni is kell, ami azt jelenti, hogy a lemez területét felosztjuk a különböző operációs rendszerek között.)

Formázás előtt nézzük meg, hogy nincs-e a lemezen olyan program vagy adat, amelyet később használni szeretnénk, majd

a parancs: `FORMAT ␣ A:`

Nagyon lényeges, hogy beírjuk a megformázandó lemez betűjelét, mert ha csak a FORMAT-ot írjuk, akkor az aktuális C: meghajtót formázzuk, így a winchestert formázzuk újra, vagyis mindent letörlünk róla.

A parancshoz opciót írva a következők valósulnak meg:

/s — hatására a formázást követően átmásolódnak az operációs rendszer betöltéséhez szükséges fájlok,
/v — hatására a lemeznek nevet adhatunk,
/4 — a lemezt XT formára formázza a gép, ha AT-n formázzuk, akkor is. (A kis kapacitású /DD-s/ lemezekre jellemző 360 kB-os méretre formáz az operációs rendszer.)

Legyen parancsunk: `FORMAT ␣ A: /S/V`

Formattálás közben üzeneteket kapunk a géptől, melyek a következők lehetnek:

1. Felszólít a lemez elhelyezésére. Kiírja:

```
INSERT NEW DISKETTE FOR DRIVE A :
AND PRESS ENTER WHEN READY
```

   Tegye be az A: jelű meghajtóba a lemezt, majd nyomja be az ENTER billentyűt, ha kész!

2. A formattálás befejezését is jelzi a gép. Kiírja:

```
FORMAT COMPLETTE
VOLUME LABEL?
```

   A formázás kész. Adjunk nevet a lemeznek?

3. A név beírása után kiírja a lemez jellemzőit:
   — az összes bájt számát,
   — a felhasznált bájtok számát,
   — a felhasználható bájtok számát.

4. Megkérdezi, hogy folytatjuk-e a formázást:

   `FORMAT ANOTHER (Y/N)?`

   Akar formázni még (Igen/Nem)?

   Válaszunktól függően folytatódik a formázás vagy visszatér a rendszer a DOS-hoz.

5. A merevlemez formattálása előtt a gép megkérdezi tőlünk, hogy valóban formattálni akarjuk-e, mert minden programunk elvész.

3. Az állományok másolása

Előfordul, hogy állományunkról másolatot akarunk készíteni. Szeretnénk a winchesterről a floppyra vagy az egyik alkönyvtárból egy másikba másolni. Ezt a következő parancs segíti:

`COPY␣honnan␣mit␣hova␣milyen névre`

A parancsot csak akkor hajtja végre gépünk, ha vannak olyan meghajtók, amiket megjelöltünk, és az adott alkönyvtárban még nincs ilyen fájl.

A parancs alakjai a következők lehetnek:

a) Szeretnénk a hajlékonylemezünk főkönyvtárának állományát átmásolni fixlemezre:

`COPY␣A:\␣*.*␣C:`

A floppy főkönyvtárának minden állományát átmásoljuk a fixlemezre.

b) A főkönyvtár egyetlen állományát, például a CONFIG . SYS-t másoljuk a lemezre:

`COPY␣C:\␣CONFIG . SYS␣A:`

A C: lemez főkönyvtárából a CONFIG . SYS-t átmásoljuk a floppyra.

c) Nem ismerjük az állomány nevét, csak kiterjesztését, ezért szeretnénk az összes ilyen kiterjesztésű állományt átmásolni a lemezünkre.

`COPY␣C:\␣*.BAT␣A:`

d) Azt tudjuk, hogy a másolandó állomány nevének kezdőbetűje CS. Másoltassuk át az összes ilyen állományt a lemezünkre!

`COPY␣C:\CS*.*`

Ismeretlen jelet, a *-ot írtuk be. Ez egy helyettesítő, idegen szóval joker karakter, amellyel az állománynevek közül egyszerre többre is utalhatunk. Jelentése: „és innentől bármilyen karakterek állhatnak".

Olvasmány

Egy floppy teljes tartalmának egy másik floppyra történő átmásolását megoldhatjuk a DISKCOPY parancs segítségével is.

A parancs: DISKCOPY ␣ *honnan* ␣ *hova*

Elvégezhetjük a másolást egy meghajtón is, ekkor a parancs:

```
DISKCOPY ␣ A: ␣ A:
```

Ilyenkor a számítógép a másolandó állományt egy ideig tárolja a memóriájában. Betöltés után jelzi, hogy cseréljük ki a lemezeket. Másolás után ismét felszólít bennünket a számítógép, hogy cseréljük ki a lemezeket. Folytatja a memóriába töltést a másolandó lemezről stb.

Két meghajtó esetén a parancs:

```
DISKCOPY ␣ A: ␣ B:
```

A parancs szektorról szektorra átmásolja a lemez tartalmát, nem törődik a tartalmával. Ha egy rendezetlen lemezt másol, akkor másolás után is az marad. COPY *parancs hatására a logikailag összetartozó állományok fizikailag egymás után lévő szektorokba kerülnek.*

DISKCOPY *paranccsal formázatlan lemezt is másolhatunk, mert a parancs elvégzi a formázást is.*

4. Az állományok törlése

Az alkönyvtárak megszüntetésekor kikötöttük, hogy csak üres alkönyvtárat törölhetünk. Ha úgy döntünk, hogy megszüntetünk valamilyen állományt, akkor ki kell adnunk a DEL parancsot az állomány azonosítójával együtt.

Például: ha a prompt A:\> , akkor a parancs lehet:

```
DEL ␣ PROBA.BAT
```

Jelentése: töröljük az A: meghajtó főkönyvtárából a PROBA . BAT állományt.

Ha egyszerre több állományt szeretnénk törölni, akkor alkalmazhatjuk a joker karaktereket (*, ?) is.

Például: az A:\> prompt mellett kiadhatjuk a parancsot:

```
DEL ␣ * . EXE
```

A számítógép mindaddig elvégzi a törlést a parancs kiadása után, míg a parancs alakja nem lesz:

```
DEL ␣ * . *
```

Jelentése: töröljük az aktuális könyvtár összes állományát!

Ebben az esetben megjelenik a képernyőn egy kérdés:

```
ARE YOU SURE (Y/N)?
```

(Biztos Ön ebben?)

Ha válaszunk Y, akkor minden állományt kitöröl.
Ha válaszunk N, akkor nem hajtja végre a parancsot.

Fontos megjegyeznünk, hogy a DEL parancs kiadása előtt be kell lépnünk abba a könyvtárba, ahol a törlendő állományunk van. A könyvtár tartalmát belülről is törölhetjük. (A törlést útvonalmegadás használatával is elérhetjük.)

Példák

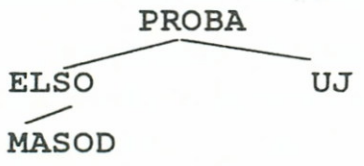
1. Készítsünk a hajlékonylemezünkön egy KOCSI nevű könyvtárat, és ezen belül hozzunk létre egy KEREK nevű alkönyvtárat! Másoljuk át a KEREK nevű alkönyvtárba az AUTOEXEC.BAT állományt a merevlemez főkönyvtárából!

```
A:\
MD␣KOCSI
CD␣KOCSI
MD␣KEREK
CD␣KEREK
COPY␣C:\AUTOEXEC.BAT
```

2. Töröljük az előző könyvtárszerkezetet! (Benne vagyunk a KEREK nevű alkönyvtárban, ezért elvégezhetjük az állomány törlését.)

```
DEL␣AUTOEXEC.BAT
CD ..
RD␣KEREK
CD ..
RD␣KOCSI
```

3. *a)* Hozzuk létre a következő könyvtárszerkezetet az A: lemezünkön, és másoljuk a MASOD nevű alkönyvtárba a merevlemez főkönyvtárából ugyanezen a néven az AUTOEXEC.BAT állományt!




```
MD␣PROBA
CD␣PROBA
MD␣UJ
MD␣ELSO
CD␣ELSO
MD␣MASOD
CD␣MASOD
COPY␣C:\␣AUTOEXEC.BAT
```

b) Rajzoltassuk ki a könyvtárszerkezetet!
Visszalépünk a főkönyvtárba:

```
CD\
TREE
```

Láthatjuk, hogy létrehoztuk a könyvtárat.

c) Lépjünk be a PROBA nevű alkönyvtárba, és írassuk ki a tartalmát!

```
CD␣PROBA
DIR
```

d) Lépjünk be a MASOD nevű alkönyvtárba, és írassuk ki a tartalmát!

```
(A:\PROBA)
CD␣ELSO\MASOD
DIR
```

e) Töröljük lemezünkről ezt a könyvtárszerkezetet!

```
DEL␣*.*
```

(Végrehajtás előtt gépünk megkérdezi, hogy valóban törölni szeretnénk-e.) Y megerősíti törlési szándékunkat.

```
CD..
RD␣MASOD
DIR
CD..
RD␣ELSO
RD␣UJ
CD..
RD␣PROBA
```

f) Győződjünk meg róla, hogy valóban töröltük, amit akartunk!

```
DIR vagy TREE
```

5. A megismert DOS-parancsok

Foglaljuk össze, hogy melyek azok a parancsok, amelyekkel az előző fejezetben megismerkedtünk!

a) **Elemi parancsok**
(általában paraméter és kapcsoló nélküli parancsok)

Parancs és hatása

prompt [szöveg] Az MS-DOS készenléti jelét változtatja meg. Kapcsolók segítségével nagyon sokféle promptot állíthatunk elő.

cls Törli a képernyőt, az üres képernyő bal felső sarkában csak a készenléti jel jelenik meg.

time [óra:[perc:[másodperc]]][a/p]
Megjeleníti a gép pontos idejét, és lehetővé teszi annak módosítását (a – délelőtt, p — délután).

date [mm – dd – yy] [hónap – nap – év]
Megjeleníti a dátumot, és lehetőséget biztosít a módosítására is.

ver A képernyőn megjelenik a DOS verziószáma.

b) **Könyvtárkezelő parancsok**

A parancs és hatása

dir [meghajtó:] [elérési út] [file-név] [/p/w]
A képernyőre írja egy könyvtár állományainak, alkönyvtárainak listáját.

cd [meghajtó:] [útvonal]
Aktuálissá tesz egy megadott könyvtárat, a megadott útvonalon haladva belépünk az adott könyvtárba.

cd.. Visszalépés az alkönyvtárból az előző könyvtárba, a (gyökér felé) főkönyvtár irányába.

cd Visszalépés rögtön a főkönyvtárba.

md [meghajtó:] ␣ elérési út, md ␣ új alkönyvtár neve.
Üres alkönyvtárat hoz létre.

tree Kirajzolja az aktuális könyvtár felépítését.

c) **Lemezkezelő parancsok**

Lemezegység neve, *például* A:
Aktualizálja azt a lemezegységet, amelyen dolgozni szeretnénk.

format ␣ meghajtó: [kapcsolók]
Előkészíti a lemezt arra, hogy használhassa a DOS.

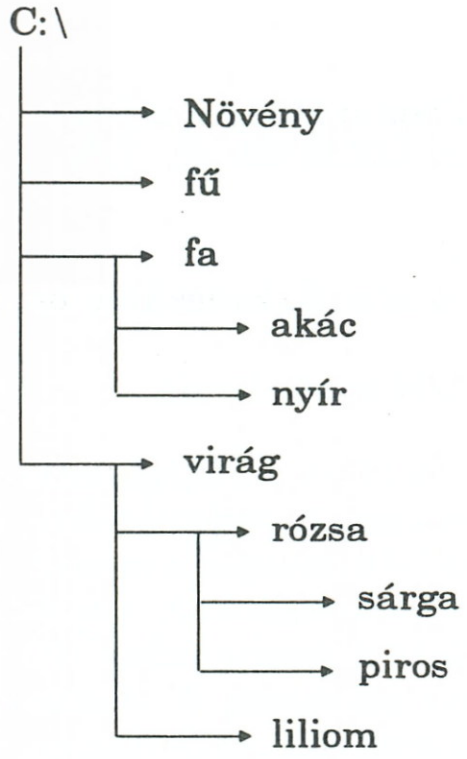
copy ␣ [honnan] ␣ [mit] ␣ [hova] ␣ [milyen névre]
— állományokat másol,
— lehetőséget biztosít a billentyűzetről történő adatfelvételre a lemezen.

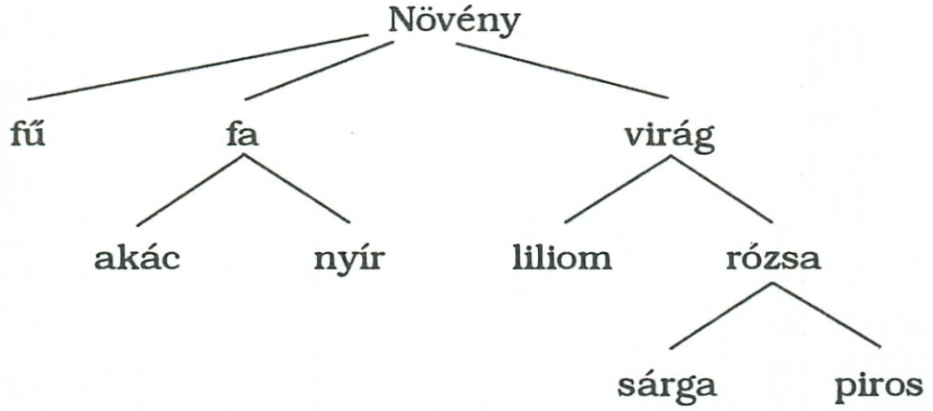
del ␣ [meghajtó] ␣ [elérési út] ␣ file-név
A megnevezett állományokat törli a könyvtárból.

Kérdések és feladatok

1. Milyen alakúak a DOS-ban kiadott parancsok?
2. Hogyan jelezzük a számítógépnek, hogy végeztünk a parancs beírásával?
3. Milyen információkat kaphatunk a készenléti jelből vagy promptból?
4. Állítsuk be a promptot úgy, hogy
   a) a pontos időt mutassa;
   b) az Ön nevét írja ki;
   c) a DOS verziószámát jelezze;
   d) az aktuális meghajtót írja ki;
   e) tartalmazza az aktuális meghajtót és könyvtárt is!
5. Hogyan törölhetjük a képernyőt?
6. Hogyan jeleníthetjük meg a dátumot és az időt?
7. Mire kell figyelnünk új idő, illetve új dátum beírásakor?
8. Állítsuk be a dátumot!
9. Miért fontos a VER parancs?
10. Mi a szerepe a könyvtárrendszernek?
11. Mi történik a DIR és a dir parancs hatására?
12. Hogyan azonosíthatjuk a fájlokat?
13. Mit tartalmaz egy teljes fájllista?
14. Milyen kapcsolókat alkalmazhatunk dir parancs esetén, és alkalmazásuk milyen következménnyel jár?
15. Írassuk ki a főkönyvtár tartalmát úgy, hogy
   a) egyszerre egy képernyőnyi adatot lássunk;
   b) csak a fájl neveket és kiterjesztésüket lássuk;
   c) egyszerre egy képernyőnyi tömörített listát lássunk!
16. A tartalomjegyzékben hogyan ismerhetjük fel, hogy alkönyvtárról vagy fájlról van-e szó?
17. Hogyan léphetünk egyik alkönyvtárból a másikba?
18. Mi a jelentése a következő parancsoknak?
   a) CD␣útvonal
   b) CD.. vagy CD␣..
   c) CD\
19. Hogyan hozhatunk létre alkönyvtárakat?
20. Milyen parancs hatására másolhatjuk át:
   a) hajlékonylemezünk tartalmát fixlemezre;
   b) hajlékonylemezünk egyetlen állományát fixlemezre;
   c) A: lemez E-vel kezdődő állományait C: lemezre;
   d) C: lemez .SYS kiterjesztésű állományait?
   Listázással győződjünk meg róla, hogy másolásunk sikerült!

21. Hozzuk létre a következő könyvtárszerkezetet!





22. Ellenőrizzük, hogy valóban létrehoztuk-e a szerkezetet!
23. Hogyan szüntethetünk meg egy alkönyvtárat?
24. Honnan tudjuk, hogy egy alkönyvtár üres?
25. Milyen parancs hatására jelenik meg a képernyőn a könyvtárszerkezet struktúrája (fa)?
26. Rajzoltassuk ki a 21. feladatban létrehozott könyvtárszerkezetet!
27. Töröljük az előző feladatban létrehozott könyvtárszerkezetet! Törlés előtt vizsgáljuk meg, hogy üres-e a törlendő könyvtár!
28. Hogyan aktivizálhatunk egy lemezegységet?
29. Mit jelent a lemez formattálása?
30. Mire kell ügyelnünk a Format parancs kiadása előtt?
31. Milyen kapcsolókat használhatunk formázáskor?

32. Formázzunk meg egy hajlékonylemezt (A:) úgy, hogy
   a) formázás után a lemezre kerüljenek az operációs rendszer betöltéséhez szükséges fájlok,
   b) nevet adhassunk a lemeznek,
   c) átkerüljenek az operációs rendszer betöltéséhez szükséges fájlok és nevet is adhassunk a lemeznek!

33. Hogyan másolhatunk állományokat?

34. Hogyan törölhetjük az alkönyvtárak tartalmát?

35. *a)* Másoljuk át hajlékonylemezünk főkönyvtárába fixlemezünk összes .EXE és .COM kiterjesztésű állományát!
   b) Töröljük az állományokat a COMMAND.COM kivételével!

V. A BASIC NYELV ALAPJAI

A) A számítógépeken használt nyelvek

A számítógépek a saját gépi nyelvükön dolgoznak. Bitekkel, bájtokkal végeznek műveleteket. A legegyszerűbb feladatokat is nagyon sok lépésre bontják fel. Ezeket a legelemibb lépéseket, utasításokat nevezzük gépi nyelvnek, gépi kódoknak. Az átlagember számára nagyon bonyolult a gépi utasításokat jelölő bináris számsorozatok előállítása, ezért kezdetben ez gátolta a számítógépek terjedését. A probléma kiküszöbölésére hamarosan létrejöttek a közvetítő programnyelvek, amelyek az emberi nyelvhez közelebb állnak, könnyen lefordíthatók, átültethetők az adott számítógéptípusra jellemző gépi nyelvre. A számítógép egy erre a célra megírt program (fordítóprogram) segítségével felismeri a programnyelv egyes szavait, és végrehajtja az ezeknek a szavaknak megfelelő gépi kódú utasítássorozatot.

Olvasmány

A gépi nyelvnél kicsit könnyebbek az ASSEMBLY típusú nyelvek, ahol a gépi utasítások neveit (mnemonik) írhatjuk a számok helyett (pl.: ADD: összegezd, LD: töltsd). Az ASSEMBLY nyelvet éppen úgy, mint az előbb említett közvetítő programnyelven írt programokat le kell fordítani a számítógép nyelvére. Egyes számítógépek már tartalmazzák az assemblernek nevezett fordítóprogramot. (Különböző számítógéptípusok más-más gépi utasításai miatt az assembly nyelvek is gépfüggőek.) Ezek a számítógépek maguk végzik a fordítást.

Hamarosan felmerült az igény gépfüggetlen, univerzális nyelvek kialakítására, amelyek a felhasználói munkát lényegesen egyszerűsítenék. Az ilyen nyelveken írt program már minden olyan gépen futtatható, amelyhez elkészült az adott nyelv fordítóprogramja.

Ismerkedjünk meg néhány nagy hatású, más-más célra készült programozási nyelv elnevezésével és ezek alkalmazási területeivel.

Programnyelv	*Leggyakoribb alkalmazása*
FORTRAN	*a matemetikai, műszaki és tudományos számításokat gyorsította (1954-ben készült el az első változata).*
ALGOL	*a műszaki és tudományos munkát segítette, főként a programozás elméletében játszott szerepet.*
COBOL	*a kereskedelmi és ügyviteli munkát segítette, mert nagy tömegű adat feldolgozását tette lehetővé.*
BASIC	*általános feladatok megoldására alkalmas.*

A mikrogépek szaporodásával más magas szintű nyelvek is ismertté váltak, például a FORTH, PASCAL, LOGO, PROLOG, PL 1, MPROLOG stb.

A különböző nyelveken írt programokat az adott számítógéptípus adott nyelvére írt fordítóprogramja végigolvassa, kifejezésenként értelmezi, majd a saját nyelvére lefordítja. Ezzel természetesen lassul a számítógép munkája, és a fordítóprogram is helyet foglal el a memóriából.

1. A BASIC nyelv jellemzői

A BASIC nyelvet 1964-ben egy magyar származású tudós, John Kemény és kollégái fejlesztették ki Amerikában, kezdők számára, általános feladatok megoldására. A nevét is innen kapta (**B**eginners **A**ll Purpose of **S**ymbolic **I**nstruction **C**ode: kezdők általános célú szimbolikus utasításkódja).

Alapja a FORTRAN nyelv volt, és elsősorban a személyi számítógépek programozására használták. Ez a nyelv viszonylag közelebb áll a beszélt angol nyelvhez, amit kulcsszavai is mutatnak.

Például: PRINT — *írd ki!* INPUT — *olvasd be!* STOP — *állj!* ...

Nagyon előnyös számunkra, hogy kevés utasítása van, így rövid tanulás után már saját programot írhatunk. Ha a programunkban hibát fedezünk fel vagy módosítani szeretnénk, akkor ezt azonnal megtehetjük. Az is kedvező számunkra, hogy számítógépünkön igen sokféle feladatot megoldhatunk e nyelv segítségével. Hosszabb programoknál a nyelv hátránya lehet, hogy kicsit lassú, de ez a sebesség legtöbbször elfogadható számunkra.

2. A QUICK BASIC nyelv

Az MS DOS 5.00 operációs rendszert használó személyi számítógépek programozásakor használhatjuk a Quick BASIC (QBASIC) nyelvet. Ennek a nyelvnek a kifejlesztésekor megpróbálták egyesíteni a BASIC egyszerűségét a magas szintű nyelvek profizmusával, így jött létre ez a Pascalhoz hasonlító nyelv. (A program írásakor nem kötelező használni az utasítások előtt a sorszámozást. Szükség esetén úgynevezett címkével hivatkozhatunk egy adott programsorra. Létrehozhatunk függvényeket, készíthetünk eljárásokat stb.)

Azok a programozók, akik ismerik a BASIC nyelvet, könnyen alkalmazhatják a Quick BASIC nyelvet is.

Tankönyvünkben programjaink többsége mindkét nyelven megtalálható. Olyan esetekben, amikor a programsorokban nem szükséges változtatás ahhoz, hogy QBASIC-ben is fusson programunk, külön nem foglalkoztunk annak átírásával, rábíztuk az olvasóra.

A Quick Basic nyelv lehetőséget teremt számunkra, hogy elsajátítsuk a programozás módszertanát, a strukturált programozást. Ennek rövid ismertetésére a ciklusutasítások után visszatérünk, mert ott példákkal is szemléltethetjük az elmondottakat.

3. Bevezetés a BASIC nyelv alkalmazásába

Ha a számítógéphez fordulunk, akkor adhatunk neki parancsot vagy utasítást. A parancsot a számítógép azonnal végrehajtja.

Például: CLOAD, CSAVE a kazettáról betöltésnél, illetve felvételnél.

A számítógépet az utasítás is valamilyen tevékenység elvégzésére szólítja fel, de ezt a gép csak akkor hajtja végre, ha az utasítások sorozatában (programban) odaér az adott utasításhoz.

a) Az utasítások alkalmazása

A BASIC egyes változataiban, ha programot írunk, akkor az utasításokat számozni kell, mert a számítógép csak a sorszámok alapján tud dolgozni.

Az utasítás előtt egy 0 és 5 jegyű (legfeljebb ötjegyű) közötti egész szám áll, melyet tetszőlegesen választhatunk meg. A gyakorlatban tízesével szoktunk számozni, hogy később két utasítás közé beilleszthessünk egy újabbat, és ne kelljen az összeset átsorszámozni. Az utasításokat nem kell sorban beírni, mert a számítógép maga is növekvő sorrendbe állítja azokat.

Egy utasítássorba több utasítást is írhatunk, de ezeket kettősponttal el kell választani.

Ha a gép két azonos sorszámú utasítást talál, akkor az első utasítás törlődik, és a második marad a programban. Egy sort úgy is törölhetünk, ha a sorszámát beírjuk, és lenyomjuk a RETURN gombot.

b) Az utasítások alakja

Egy utasítás mindig tartalmaz rögzített formában leírandó, többnyire angol szavakat, melyeket bizonyos BASIC nyelv változatokban nyomtatott nagybetűvel kell írnunk. Előfordulhatnak rögzített írásjelek is, például az egyenlőségjel vagy a zárójel.

Vannak változtatható részek, *például a számok, változók, jelek stb.*

Az utasítás eléggé kötött alakja tehát a következő:

sorszám — változtathatatlan rész — változtatható rész

Például: `20 PRINT "LUKÁCSI ESZTER"`

Ha sorszámokat alkalmazunk, akkor a számítógép az utasításokat általában a sorszámok növekvő sorrendjében hajtja végre, de ettől eltérhetünk. *Például:* GOTO utasítással (később megismerjük).

4. Menürendszer

A programozó a program írása közben arra törekszik, hogy a felhasználó számára könnyen kezelhető, érthető programot készítsen. Igyekszik megteremteni, hogy a felhasználónak legyen módja párbeszédet folytatni a számítógéppel, legyen lehetősége arra, hogy maga válassza ki, amit csinálni szeretne. A választékrendszert, amely felsorolja a lehetőségeket, menünek nevezték el.

A programok első menüjét, amellyel a program bejelentkezik, főmenünek nevezik. A program végrehajtása során, egy-egy részfeladat elvégzése után a program futása ismét itt folytatódik. A felhasználónak újból lehetősége van arra, hogy a választékból újat válasszon vagy az előzőt futtassa ismét, esetleg befejezi a munkát a program vége menüpont választásával. Szükséges tehát, hogy legyen egy program vége „fogás" is az „étlapon", hogy ki tudjunk lépni a program futásából.

A főmenü pontjaihoz tartozhatnak úgynevezett almenük, amelyek további választási lehetőségeket biztosítanak. Ezeknek az almenüknek olyan pontot is kell tartalmazniuk, amely segíti a főmenühöz való visszatérést. Ez azért fontos, hogy téves választás esetén ne kelljen lefuttatni az adott részprogramot, illetve az almenü választott pontjának végrehajtása után visszaléphessünk a főmenübe.

5. A QBASIC értelmező alkalmazása

A bekapcsolt számítógépünkön megjelenik a készenléti jel, hozzákezdhetünk munkánkhoz. Tudjuk, hogy ha számítógépünk tartalmazza a QBASIC-hez tartozó fájlokat, akkor azt a DOS alkönyvtárban kell keresnünk. Lépjünk be a DOS alkönyvtárba, és írassuk ki a tartalmát!

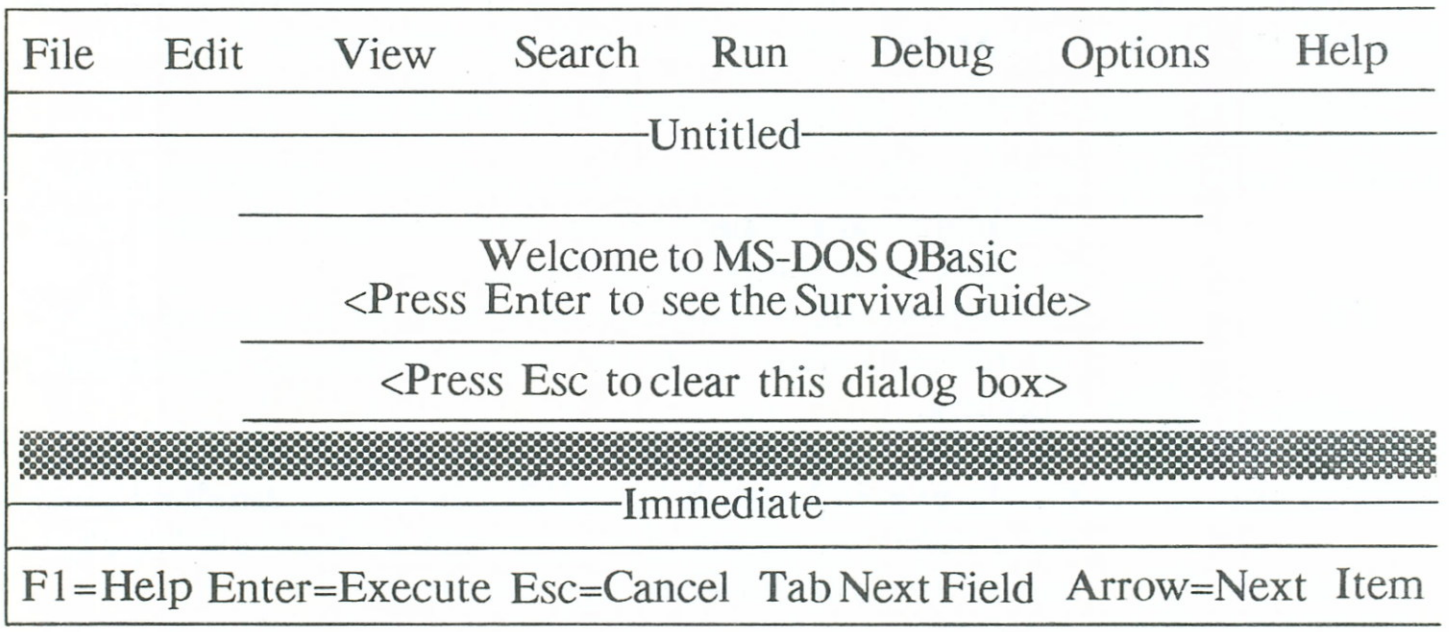
Parancsaink:
```
CD DOS
DIR/p/w
```

A képernyőn megjelenő tartalomjegyzékben keressük meg a QBASIC.EXE nevű fájlt. (Természetesen, ha tudjuk, hogy van ilyen fájl, akkor nem kell kiíratni a DOS tartalmát.)

Ennek futtatásával kapcsoljuk be a QBASIC értelmezőt!

Parancs:
```
QBASIC
```

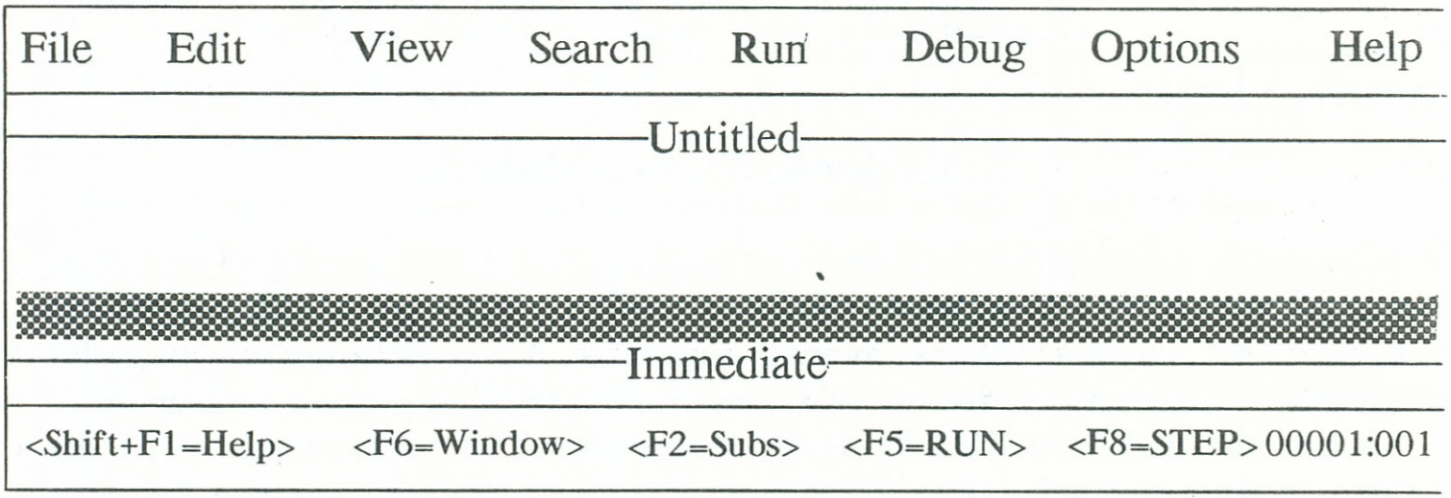
Parancsunk hatására egy új képernyőképet kapunk, melyet a *9. ábra* szemléltet.



9. ábra

A képernyőn kétféle menüt találunk. Az egyik a képernyőn alul elhelyezkedő főmenü, amelyből kiválaszthatjuk, hogy mit csinálunk a számítógépünkkel. A választási lehetőséget a képernyő közepén található világosabb téglalapba (ablakba) írt menüpontok is mutatják. A kurzor az első menüpont alatt villog, és a szöveget közrefogó zárójelek is fehérek. Ez azt jelenti, hogy az **Enter** gomb lenyomása után megjelenik a QBASIC ismertetőjét tartalmazó menü. Mi azonban programozni szeretnénk, párbeszédet akarunk folytatni a gépünkkel, ezért a második pontot aktivizáljuk. Benyomjuk a **Tab** billentyűt. Most a kurzor a második menüpont alatt villog, és itt lettek fehérek a szöveget közrefogó zárójelek.

Az **Esc** billentyű lenyomása után eltűnik a képernyőről a „világító" téglalap, az ablak, és a *10. ábrán* látható képet kapjuk.



10. ábra

Megváltozott a képernyő alsó sorában látható főmenü. Egyik felében a programozás során fontos funkcióbillentyűk szerepét mutatja. A jobb oldali sarokban a számok a kurzor pozícióját mutatják (a képernyő bal felső sarkában villog). Ez a szövegszerkesztés során segíti munkánkat.

Próbáljuk ki, mi történik az egyes gombok lenyomásakor!

Shift + F1 — Megjelenik egy menürendszer, melyből a QBASIC rendszer alkalmazására vonatkozóan választhatjuk ki azt a pontot, amelyet szeretnénk megismerni.

F2 — Választhatunk a szubrutinok közül. Az ablakban megjelenő villogó kurzor mutatja, hogy szerkesztő üzemmódban vagyunk.

F5 és F8 — Látszólag ugyanarra vezet, mert még nincs saját programunk, vagyis kapunk egy sötét képernyőt, amelyen az előzőleg beírt parancsokat és a prompt változásait látjuk, valamint a képernyő alján egy üzenetet:

Press any key to continue.

Tetszőleges billentyű lenyomása után visszakapjuk az üres képernyőt.

F5 — A program futását indítjuk el a megszakítási ponttól.

F8 — A program futását lépésenként, programsoronként hajthatjuk végre.

F6 — A következő ablakra történő átlépést segíti. A kurzor alul villog, az Immediate vonala alatt. Beléptünk a közvetlen ablakba, ami azt jelenti, hogy az ide begépelt parancsainkat (az Enter billentyű lenyomása után) azonnal végrehajtja számítógépünk.

Írjuk be a következő parancsot! PRINT 5 * 6

Az **Enter** *lenyomása után megkapjuk, hogy az eredmény* 30.

(Vegyük észre, hogy megint változott a főmenü!)

6. Képernyők és ablakok

A billentyűk nyomogatása közben észrevettük, hogy két képernyő jelent meg. Az egyik, amelyre a programot írtuk, és ahol a programot alakítjuk, módosítjuk, ezért ezt szerkesztőképernyőnek nevezzük. A másik, amelyen megjelenik programunk futásának eredménye, ezért ezt futóképernyőnek nevezzük.

A két képernyőt az **F4** billentyű lenyomásával váltogathatjuk. A program futása közben a **Ctrl** + **Pause** lenyomásával léphetünk vissza a futóképernyőről a szerkesztőképernyőre, ahol egy világosabb programsor mutatja, hogy

hol szakadt meg a program futása. **F5** lenyomása után innen folytatódik a program végrehajtása.

A futóképernyő felső ablakában helyezzük el a programunkat, melyet az **F5** billentyű lenyomásával futtatunk.

Az alsó ablak csak két sorból áll. Ezt a közvetlen ablakot használhatjuk kalkulátor üzemmódban, vagy olyan esetekben, amikor a programírás közben ellenőrizni szeretnénk, hogy egy adott programsor végrehajtásakor az történik-e, amit szeretnénk. Itt csak 2 sort vehetünk igénybe.

a) Ablakos menürendszer

A képernyő felső részén is találunk egy ablakot, amelyben újabb menüpontok vannak. Munkánk során nemcsak ezekből a menüpontokból, hanem a hozzájuk tartozó almenükből is választhatunk.

Olvasmány

Röviden ismerkedjünk meg a menürendszer szolgáltatásaival, melyek közül mi keveset használunk. Azok, akik többet szeretnének tudni az egyes pontokról, a szakirodalomból elolvashatják, vagy help segítségével a képernyőn is megjeleníthetik a szükséges ismereteket.

Edit	*Menüpont választásakor programszerkesztést végezhetünk.*
View	*Hatására választhatunk a szubrutinok közül, megjeleníthetjük, törölhetjük, módosíthatjuk őket, bekapcsolhatjuk az osztott képernyőt, megjeleníthetjük a kimeneti képernyőtartalmat stb.*
Search	*Segítségével kereshetünk a szövegünkben (betűt, szót, kifejezést, címkét, és lehetőségünk van módosításukra is!).*
Run	*Almenüivel a program futását befolyásolhatjuk, mi választhatjuk ki, hogy a program a futás megszakítása esetén elölről fusson-e vagy a megszakítás helyétől.*
Debug	*Lehetőséget biztosít számunkra a nyomkövetésre, a hibakeresésre. Figyelhetjük az egyes változókat, törölhetjük a változók figyelését, megszakítási pontokat helyezhetünk el stb.*
Options	*Kiegészítést, választást, hardverbővítést segít.*

Programozásunk során a FILE menüpontot használjuk, ezért részletesebben megvizsgáljuk, hogy milyen lehetőséget biztosítanak az almenük.

New Program	A memóriában lévő aktuális program(ok) törlése.
Open Program	Program betöltése az aktuális meghajtóról.
Save	A memóriában lévő program kimentése ugyanazzal a névvel.
Save As	A memóriában lévő program kimentése névváltoztatással.
Print	Az aktív képernyőn kijelölt blokk nyomtatása.
Exit	Kilépés a QUICK BASIC-ből.

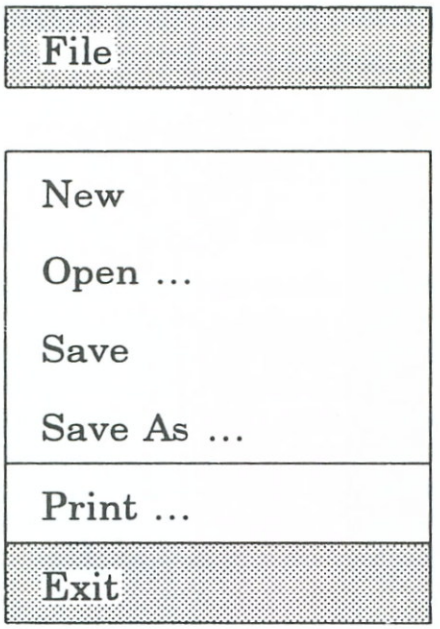
b) **Almenük használatának módja**

A következőkben azokat a lépéseket mutatjuk be, ahogyan aktivizálhatjuk az almenüket. Nézzük meg, hogyan léphetünk ki a QBASIC-ből!

Alt Billentyű lenyomása után a **File** felirat sötét téglalapban villog. Nekünk erre a menüpontra van szükségünk, ezért itt maradunk (másik választásakor a [→] billentyű segítségével a sötét téglalapot a megfelelő helyre állítjuk).

Enter Megjelenik az almenüsor. A fekete téglalapban a **New** kiírást olvashatjuk.

[↓] Mi szeretnénk kilépni, ezért a [↓] billentyűvel a sötét téglalapot az **Exit** sorra állítjuk *(11. ábra)*.



11. ábra

Enter A billentyű ismételt lenyomása után megkapjuk a kiindulási képernyőt, melyről leolvashatjuk, hogy a DOS alkönyvtárban vagyunk.

Előfordulhat, hogy az almenük vizsgálata közben úgy döntünk, hogy továbbfolytatjuk a programozást. Ilyenkor az ESC billentyű lenyomásával állíthatjuk vissza a szerkesztőképernyő eredeti állapotát.

7. Néhány előzetes programozási ismeret

— A számítógép számára minden közleményt a RETURN, más gépeknél az ENTER, esetleg NEW LINE feliratú billentyű lenyomásával fejezünk be.
— Ha END utasítást helyezünk el valahol a programban, akkor annak futása azon a helyen befejeződik, függetlenül attól, hogy a programban esetleg még további sorok következnek.

— Programunk érthetősége érdekében elhelyezhetünk magyarázó szövegeket is a következő formában:

sorszám REM megjegyzés szövege.

Ezek a megjegyzések nem befolyásolják a program végrehajtását.

8. Kalkulátor üzemmód Commodore-64 esetén

Bonyolultabb számolások esetén lehetőségünk van arra, hogy elővegyük zsebszámológépünket, és azzal végeztessük el a műveleteket. A számológép gyorsan, pontosan kiszámolja helyettünk a kifejezés értékét, és az eredmény is azonnal megjelenik a kijelzőn.

Ezeket a számolásokat számítógépünkkel is elvégeztethetjük, ezt is használhatjuk egyszerű feladatmegoldó eszközként, vagyis kalkulátorként. A számítógép azonban csak akkor írja ki a kiszámolt értéket a képernyőre, ha erre külön utasítást adunk neki. Az eredmény megjelenését a PRINT (Írd ki!) parancs segítségével érhetjük el.

Például: PRINT 12 + 24 *A beírás után a sorzáró billentyűt is le kell nyomni, mert ez jelzi a számítógépnek, hogy végeztünk a kifejezés beírásával. A képernyőn megjelenik a 36-os szám.*

Ismerkedjünk meg a számítógép által használt műveleti jelekkel, hogy helyesen tudjuk azokat beírni!

+	összeadás
–	kivonás
*	szorzás (csillag és nem pont vagy x betű)
/	osztás (nem törtvonal és nem kettőspont)
^	hatványozás

(Arra is figyelnünk kell, hogy a tizedest nem vessző, hanem angolszász szokás szerint pont jelöli.)

Példa: *Számítsuk ki egy téglalap kerületét és területét, ha oldalai 25 m és 30 m hosszúak!*

Megoldás: Egy ilyen téglalap területét a T=25*30, kerületét a K=2*(25+30) kifejezésekkel tudjuk kiszámítani.

Ezek értéke a következő parancsok hatására jelenik meg a képernyőn:

```
(1)     PRINT 25*30          <RETURN>  vagy  <ENTER>
(2)     PRINT 2*(25+30)      <RETURN>  vagy  <ENTER>
```

A (2) esetben már zárójel és különböző műveleti jelek is vannak. Itt már figyelembe kell venni az aritmetikai kifejezésekre vonatkozó szabályokat, ugyanúgy, mint számítógép nélküli számoláskor.

A műveletek sorrendjére vonatkozó követelmények

a) A műveletek elvégzésekor először a nagyobb prioritású (elsőbbségű) művelet hajtódik végre.
A prioritások sorrendje a magasabbtól kezdve:

1. előjel,
2. függvények,
3. hatványozás,
4. szorzás, osztás,
5. összeadás, kivonás,
6. relációk (<,>,=,<=,>=,<>),
7. logikai műveletek (AND, azaz és; OR, azaz vagy).

(Ez általában megfelel a hétköznapi értelmezésnek.)

Például: PRINT 64–6*2^3: balról jobbra haladva a számítógép nem a 64-ből veszi el a 6-ot, mivel utána egy magasabb rendű műveletet talál. Továbblép, s még mindig nem a 6-ot szorozza 2-vel; utána újból magasabb rendű műveleti jel áll, ezért továbbmegy. Kiszámolja 2 harmadik hatványát, majd ezt szorozza 6-tal, és így 48-at von ki 64-ből.

b) Azonos prioritású, vagyis egyenrangú műveletek esetén a balról elsőként előforduló értékelődik ki először. (Balról jobbra halad a gép.)

Például: PRINT 24/3*4: azt jelenti, hogy a 24 és a 3 hányadosát, a 8-at kell megszoroznunk 4-gyel, tehát 32-t kapunk.
Hasonlóképpen a (–b+D)/2*a kifejezés értéke nem egyenlő a (–b+D)/2/a vagy a (–b+D)/(2*a) értékével. A második és harmadik a helyesen beírt alak másodfokú egyenlet gyökeinek számolásakor.

c) Előfordul, hogy nem ilyen kiértékelési sorrendet akarunk elérni: ekkor zárójeleket alkalmazunk. Ha egy kifejezést zárójelek közé teszünk, akkor az önállóan értékelődik ki, függetlenül a környezetében lévő műveletek prioritásától.

Például: PRINT 24/(3*4): ekkor a számítógép 24-et 3*4-gyel, vagyis 12-vel osztja (2).

PRINT 24/3/4: az előző eredményhez vezet.

PRINT (64–6)*2^3: ekkor a számítógép az 58-at szorozza meg kettő harmadik hatványával (464).

Feladat: Számoltassuk ki számítógépünkkel az előző kifejezések értékét, majd zárójelezzük másként, és akkor is számítsuk ki az eredményt!

Természetesen több zárójelet is alkalmazhatunk: egymásba skatulyázhatjuk őket. Ilyenkor a legbelső zárójelek közé tett kifejezéseket értékeli ki először a számítógép. A zárójelek között belülről halad kifelé.

Például: PRINT 4*(3*(5+7)–2): a számítógép először elvégzi a belső zárójelben az összeadást, a 12-t megszorozza 3-mal, majd a 36-ból elveszi a 2-t, és a 34-et 4-gyel szorozva megkapja a 136-ot.

Olvasmány

A felesleges zárójel nem hiba. Ha bizonytalanok vagyunk a műveleti sorrendben, akkor zárójelezzünk. Ügyelnünk kell arra is, hogy több műveleti jel ne kerüljön egymás mellé. Két műveleti jel akkor sem lehet együtt, ha az egyik előjel. Zárójelezéssel kerüljük el az ilyen eseteket!

Példák

Írjuk fel a következő matematikai kifejezéseket olyan kifejezés alakjában, hogy a számítógép helyesen számolja ki értéküket!

Matematikai kifejezés	*A megfelelő BASIC-parancs*
$\frac{1}{4} \cdot 19 \cdot 56^2$	PRINT 1/4*19*56^2
$\frac{84 \cdot 187}{159 - 47}$	PRINT 84*187/(159–47)
$36 \cdot (52^2 - 6)^{1/2}$	PRINT 36*(52^2–6)^(1/2)
$(359{,}3 + 56{,}7) \cdot 3{,}4$	PRINT (359.3+56.7)*3.4

9. A PC-k kalkulátorszerű alkalmazása

A QBASIC is lehetővé teszi számunkra, hogy kalkulátorként használjuk számítógépünket. Ebben az esetben a szerkesztőképernyő alsó ablakába írjuk a végrehajtandó parancsot, és az eredményt a futási képernyőn kapjuk meg.

Példa: *Számoljuk ki* 12 + 31 *értékét!*

Megoldás:

Az **F6** lenyomásával elérhetjük, hogy kurzorunk az alsó ablakban villog, gépünk ott várja a parancsokat.

A **PRINT** 12+31 begépelése után nyomjuk le az ENTER billentyűt, és máris olvashatjuk a 43-at a futási képernyőn. Ha újabb parancsot szeretnénk beírni, nyomjunk le egy tetszőleges billentyűt, visszakapjuk a szerkesztőképernyőt.

Feladat: Írjuk be az előző példákban megadott parancsokat változatlan formában, és ellenőrizzük a kapott eredményeket PC számítógépünkön!

Feladatok és kérdések

1. Írjuk fel a következő matematikai kifejezések BASIC megfelelőjét és számoltassuk ki értéküket!

a) $26 \cdot 12^3 \cdot \frac{21}{8}$;

b) $\frac{47 \cdot 16}{57 + \frac{19}{21}}$;

c) $\frac{\frac{39}{12} - 8}{54 - 3^3}$;

d) $\frac{67 + 56}{\frac{13}{7} - \frac{5}{9}} + 4$;

e) $\frac{27 + 52}{(48 - 16)^2}$;

f) $\frac{27 + 52}{48 - 16^2}$.

2. Milyen sorrendben végzi el a számítógép a következő kifejezésekben a műveleteket? Mi lesz a végeredmény?

a) 52+3^4*6–13;

b) 40–24/2^4+52;

c) (40–24)/2^4+52;

d) ((40–24)/2^4+52.

3. Írjuk fel a következő kifejezések matematikai alakját! Számoltassuk ki értéküket!

a) 36*52/(25+13);

b) 43/19–25+16/28;

c) 57–16*5^2;

d) 13–7/3^5+2.

4. Egy 480 Ft-os áru árát 14%-kal emelték. Mennyi lett az áru ára?

(Új ár = régi ár + régi ár * $\frac{14}{100}$)

5. Számítsuk ki egy téglatest felszínét és térfogatát, ha élei 12 cm, 23 cm és 37 cm hosszúak!
6. Milyen programozási nyelvekről beszéltünk? Milyen célra használják azokat?
7. Milyen szerepe van a ROM és a RAM memóriarésznek?
8. Milyen előnyei vannak a BASIC nyelvnek más programozási nyelvekkel szemben?
9. Mi a különbség a parancs és az utasítás között?
10. Hogyan állapítja meg a számítógép, hogy melyik utasítást kell végrehajtania?
11. Hogyan törölhetünk egy utasítást?
12. Milyen alakú egy utasítás?
13. Honnan tudja számítógépünk, hogy végeztünk a programsor beírásával?
14. Hogyan használhatjuk a számítógépünk DOS alkönyvtárában található QBASIC fordítót?
15. Hogyan helyezhetünk el a programban magyarázó szöveget?
16. Melyik parancs segíti a kalkulátorszerű üzemmód használatát?
17. Milyen sorrendben hajtja végre a számítógép a műveleteket?
18. Milyen műveleti jeleket alkalmazhatunk a kifejezések felírásakor?
19. Hogyan változtathatunk a műveletek prioritási sorrendjén?
20. Hogyan dolgozik a számítógép több zárójel alkalmazásakor?
21. Mi a szerepe a funkcióbillentyűzetnek a QBASIC nyelv alkalmazásakor?

B) A programkészítés kezdeti lépései

Az előző fejezetben számítógépünket csak számolásra használtuk, de ennél többet akarunk elérni. A számítógépet szeretnénk összetett feladatok megoldására alkalmazni. Ehhez az szükséges, hogy lépések sorozatára tudjuk bontani a feladatot, más szóval algoritmust tudjunk készíteni a megoldáshoz.

Olvasmány

Olyan feladatok algoritmusáról is beszélhetünk, amelyeket nem érdemes számítógépekkel megoldani. Ilyen előírásokat találhatunk a szakácskönyvekben, mosóporos dobozokon, vagy egyéb használati utasításokon. Viselkedésünket is írott vagy íratlan szabályok határozzák meg, tehát át- meg átszövik napjainkat, de észre sem vesszük, hogy algoritmusok szerint cselekedünk.

1. Az algoritmus, a program és a programozási nyelv fogalmának értelmezése

Az algoritmus olyan eljárás, amely egy feladat véges sok elemi lépésben való megoldására irányul. (Nem minden feladat megoldására adható algoritmus.)

A számítógép egy feladatot csak akkor képes elvégezni, ha elkészítjük a megfelelő algoritmust, és ennek alapján megírjuk a hibátlan programot a gép által értelmezhető nyelven. Ehhez igyekszünk biztosítani az egyértelmű futást a műveletek és sorrendjük pontos, részletes leírásával.

Ha egy algoritmus a számítógép által értelmezhető utasításokból, részlépésekből áll, vagyis valamilyen programozási nyelven írjuk le, akkor már programnak nevezzük. A számítástechnikában *a program precízen megfogalmazott, egyértelműen végrehajtható elemi lépések sorozata. Az utasítások leírására, használatára vonatkozó szabályokat nevezzük programozási nyelvnek.*

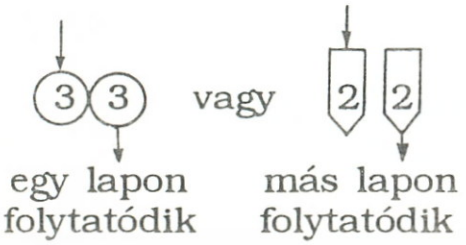
2. A blokkdiagram

Az algoritmus leírására készíthetünk blokkdiagramot, más szóval folyamatábrát.

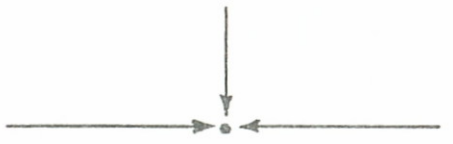
Már az általános iskolában is találkozhattunk ezzel a geometriai jelekből álló jelölésrendszerrel. Az ábrákban matematikai és logikai jelek segítségével jelöljük a soron következő tevékenységet.

Geometriai jelek	*A jelek jelentése*
x : = 3 A – B	Értékadás vagy általános művelet. Ebbe írjuk bele az elvégzendő műveletet matematikai kifejezéssel, vagy szövegesen.
vagy	
Legyen T a két oldal szorzata	
A<B? nem / igen	Vizsgálat és elágazás. Döntenünk kell, mert a továbblépés iránya a vizsgálat eredménye szerint más és más lehet. Logikai jellegű kérdésnél a válasz igen vagy nem. Matematikai döntésnél az előjeltől függően lehet negatív, nulla vagy pozitív a válasz. Itt *például* három út közül választhatunk.
START	A program kezdete.
STOP	A program vége.
X, Y kiírása	Beviteli és kiviteli műveletek (beolvasási és kiírási műveletek).

Folyamatvonal, amely a lépések egymásutánját, sorrendjét jelöli.



Folyamatvonalak csatlakozása. Ha a blokkdiagram méreténél fogva nem fér rá egyetlen lapra, vagy a továbbhaladást mutató nyíl túl távolra nyúlna.
Összetartozó csatlakozókba ugyanazt a betűt vagy számot írjuk.

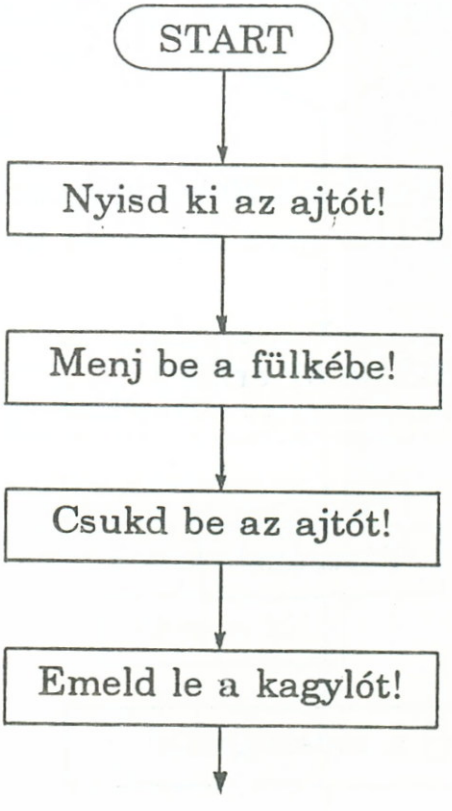


Csatlakozópont: több út összefutása.

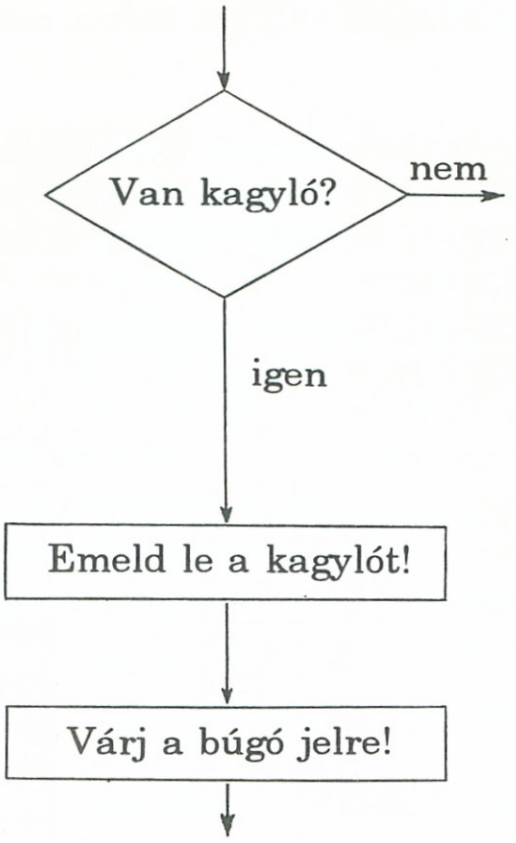
A blokkdiagram elkészítése során célszerű arra törekedni, hogy az összekötő nyilak felülről lefelé és balról jobbra haladjanak, ez könnyebbé teszi az ábránk megértését.

Telefonáló robot tervezése

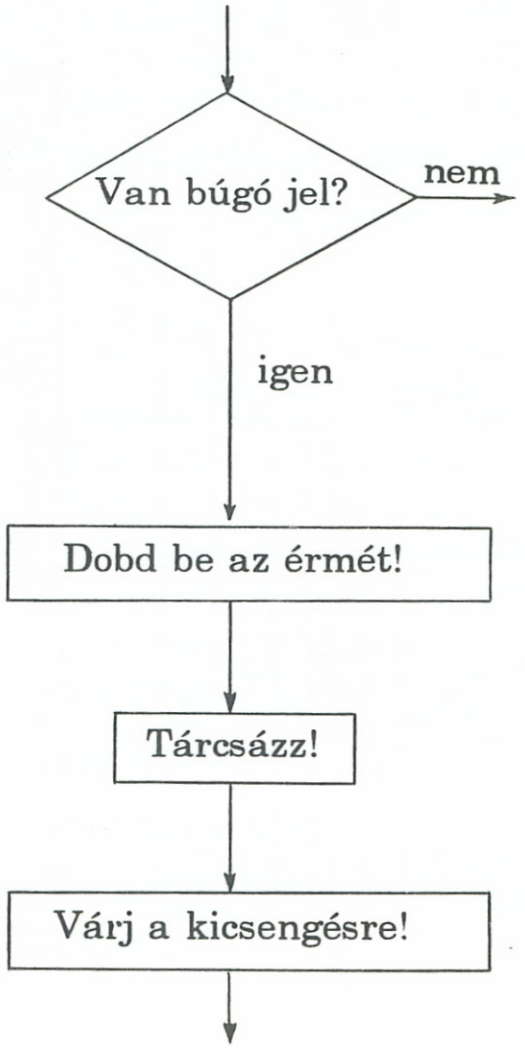
A következőkben az eddigi ismereteinket felhasználva készítünk egy folyamatábrát, amely egy robotot telefonálásra szólít fel. Ez a robot azonban csak azokat a parancsokat hajtja végre, amit mi megadunk neki. Tervezzük meg a telefonálást, ha robotunk ott áll egy telefonfülke előtt egy ötforintossal a kezében. A fülkének ajtaja is van, amely nyitható és zárható.



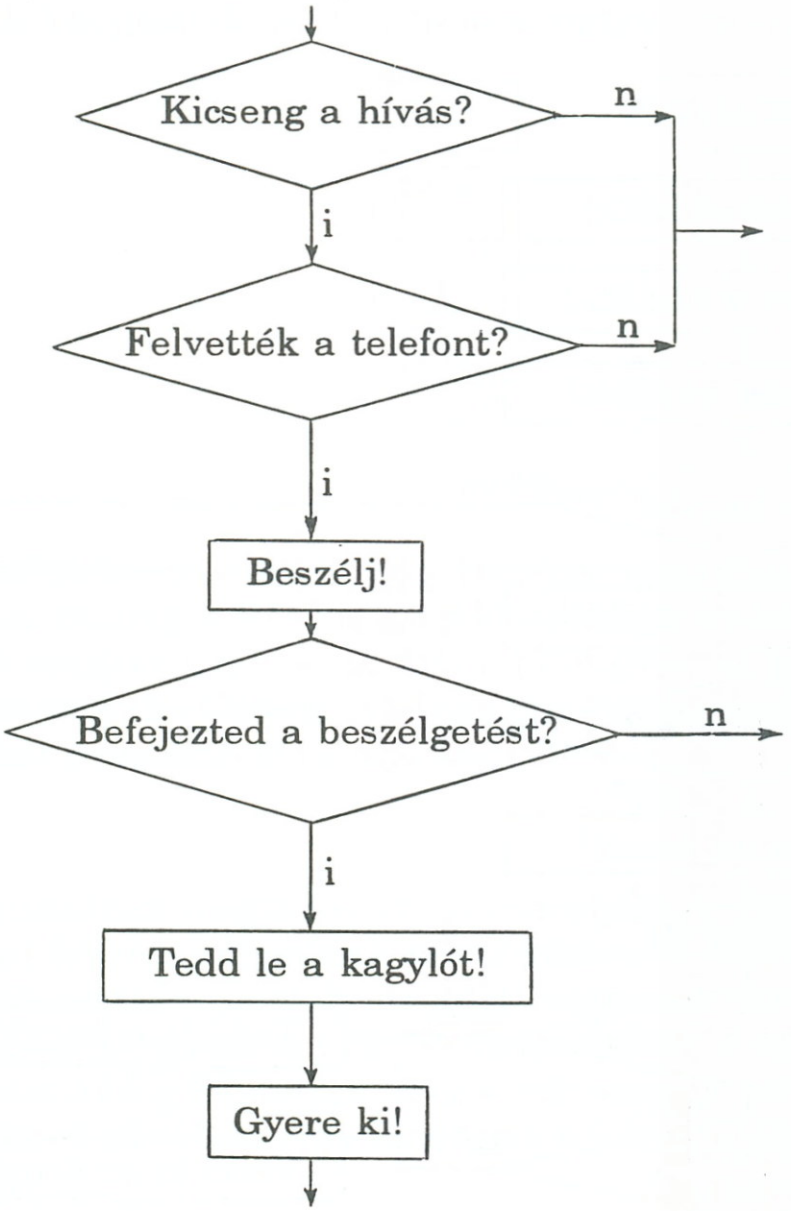
Ha nincs kagyló, akkor mit csinál a robot? Másként kellene megfogalmazni az utasítást.



Most olyan utasítást adtunk, amely robotunkat arra kárhoztatná, hogy örök időkre a fülkében maradjon, ha rossz a telefon. Ismét másként kell fogalmaznunk.



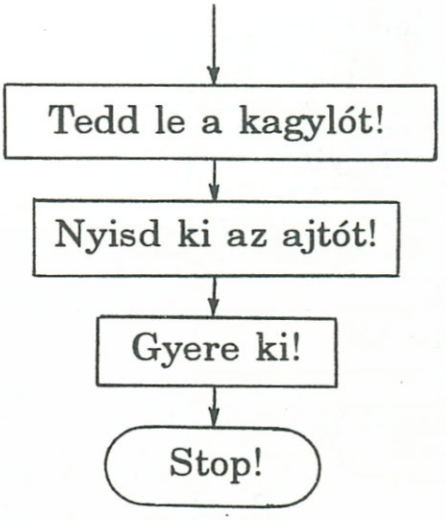
Biztosan mindenki észrevette, hogy itt is két irányban kell továbblépni, mert előfordulhat, hogy nem cseng ki a telefon, mert rossz, vagy kicseng, de akit hívunk, az nincs otthon, nem veszi fel.



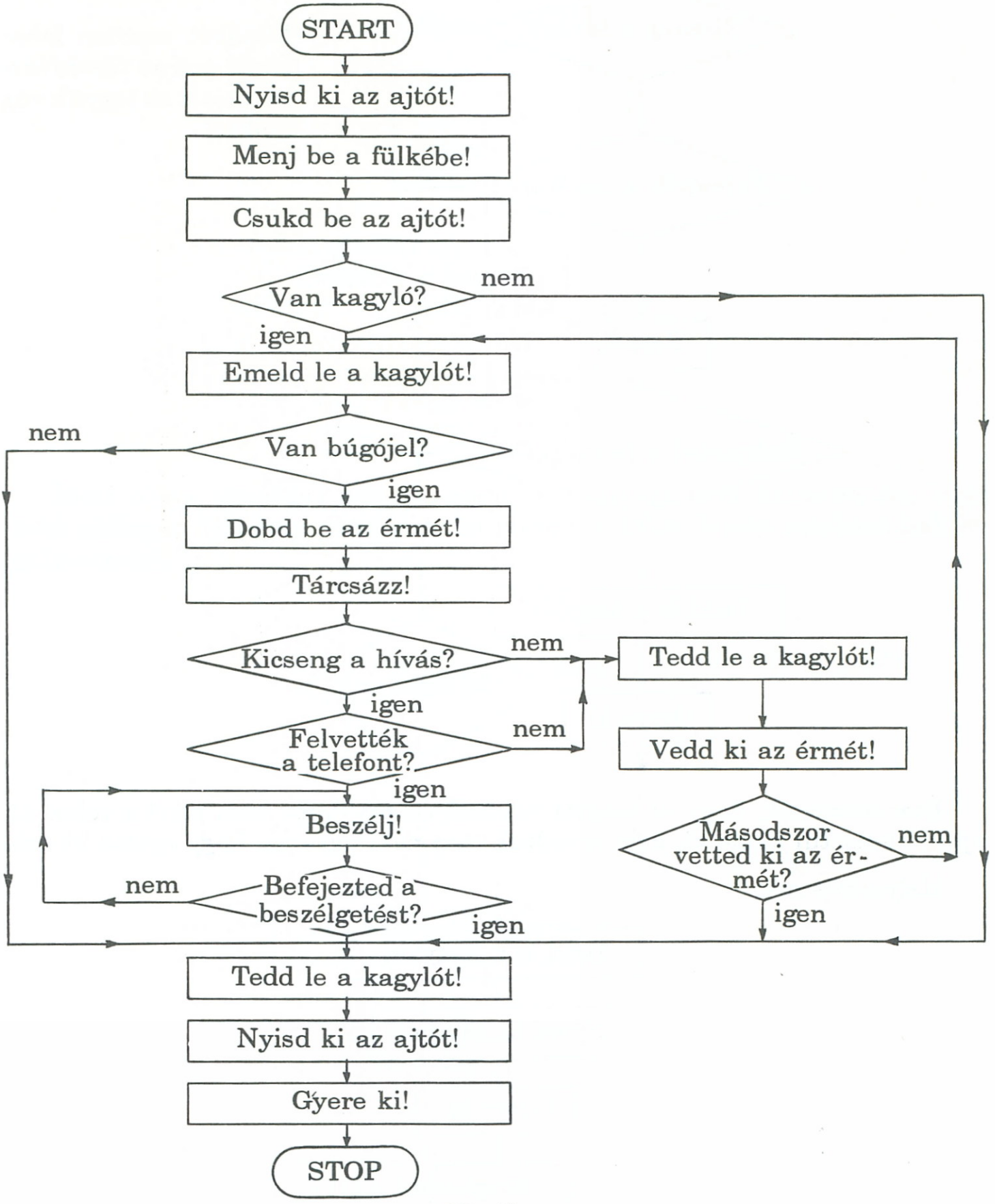
Mindkét esetben lehetőség van az újrahívásra, de ezt se tegyük vég nélkül!

Észrevehetjük, hogy itt megint valamilyen probléma van, mert a robot az ajtón keresztül jönne ki, nem mondtuk meg neki ugyanis, hogy nyissa ki azt.

Helyesbítsünk:

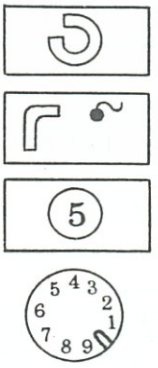


Ábrázoljuk egy folyamatábrán az eddig részleteiben megbeszélt telefonálási eljárásunkat! Ha megadjuk azt is, hogy mi a teendő akkor, ha a kérdésekre nemleges választ kaptunk, teljes blokkdiagramunk a *12. ábrának* megfelelő lesz.



12. ábra

Láthatjuk, hogy milyen körültekintően kell eljárnunk, ha egy algoritmust elkészítünk. Ha jobban körülnézünk a telefonfülkében, akkor ott is találhatunk egy egyszerű algoritmust piktogram vagy szöveg formájában, amelyet olyanoknak készítettek, akik még nem tudnak telefonálni *(13. ábra)*.



Emelje le a kézibeszélőt!

Várja meg a tárcsahangot!

Dobjon be 1 db ötforintost!

Tárcsázzon!

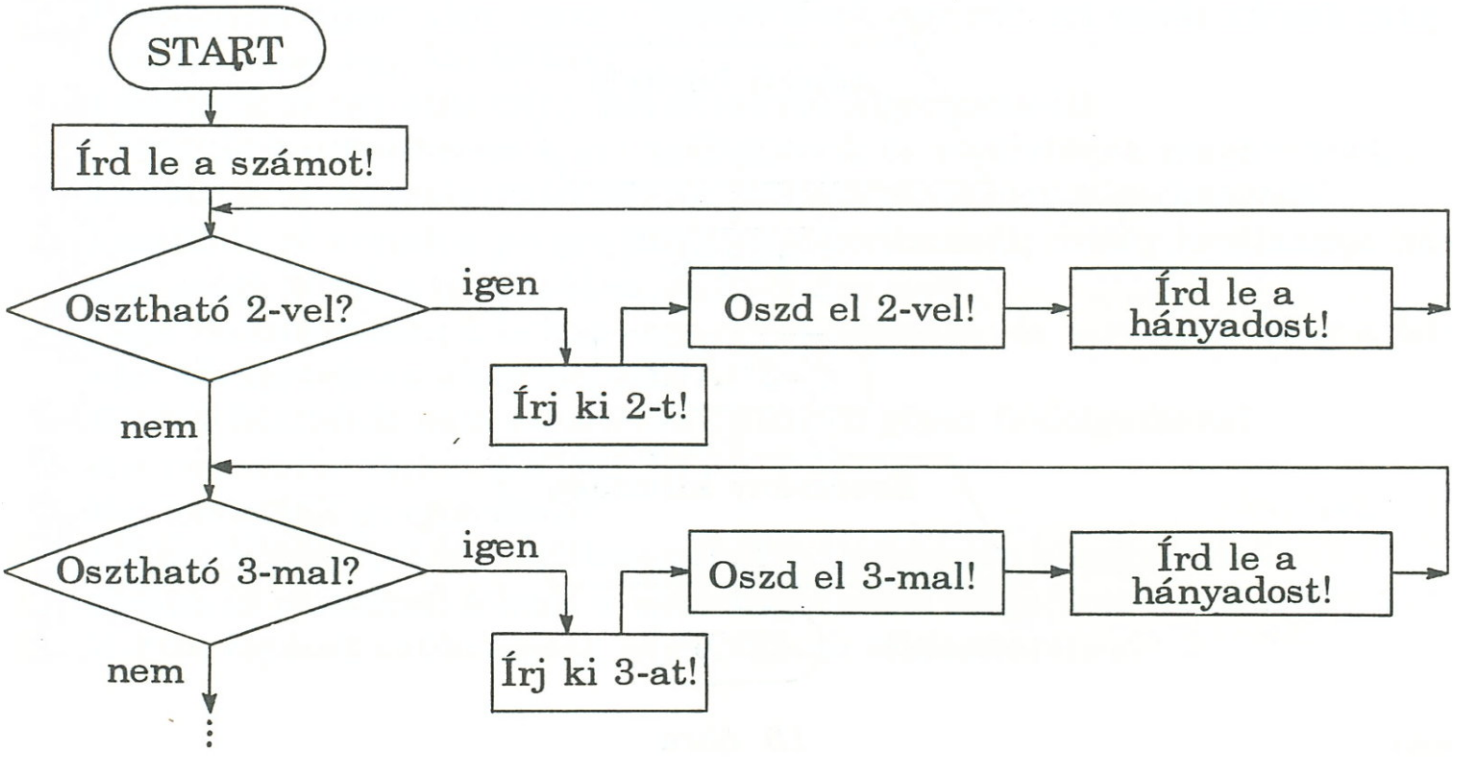
13. ábra

Az előző példa is szemlélteti, hogy az algoritmusok leírására nincsenek megkötések, mindig a feladat megoldásának érthetősége legyen előttünk. Aki nem érti a magyar nyelvet, de érti a piktogramot, az tud telefonálni.

A megismert blokkdiagram-készítési folyamatokat alkalmazzuk néhány matematikai probléma ábrázolására!

Példák

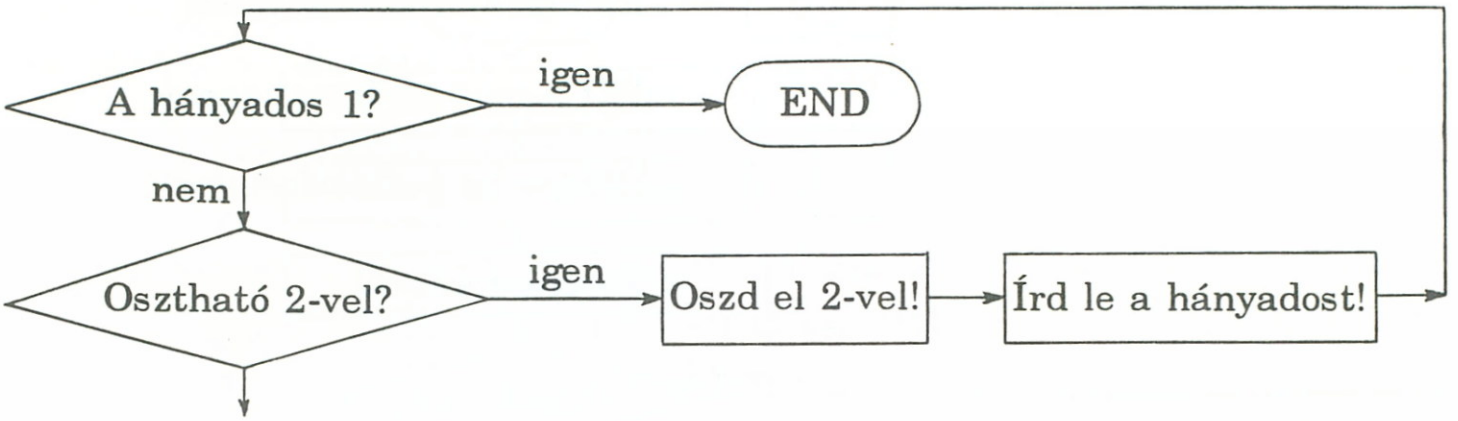
1. Matematikaórán megtanultuk azt az eljárást, melynek során az összetett számokat felírhatjuk prímek szorzataként. Ilyenkor megkerestük azt a legkisebb törzsszámot, amely az adott számnak osztója. Az osztás elvégzése után kapott hányadosnak ismét megkeressük a legkisebb prímosztóját. Ezt az eljárást addig folytatjuk, míg a hányados 1 lesz.
   Készítsük el az előbbi algoritmusnak a folyamatábráját! *(Lásd a 14. ábrát!)*



14. ábra

Blokkdiagramunkon nem ábrázoltuk, hogy a számítógépünknek meddig kell számolni, így egy végtelen hosszú eljáráshoz jutunk. Be kell iktatni az osztások elé egy vizsgálatot, amellyel megállapítjuk, hogy a szám, illetve a hányados 1-e, mert akkor befejeződne a szorzattá alakítás.

Ekkor a következőképpen módosulna az ábra mindegyik sora:

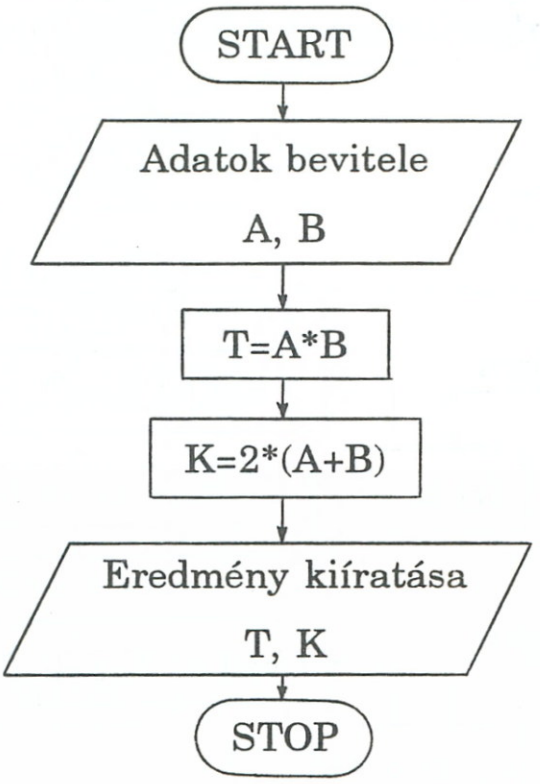


Most már megállna a program futása, de a számítógép „elfelejti" az osztókat, nem tudja megjegyezni az összetett szám prímtényezős alakját, csak saját jegyzeteinkre támaszkodhatunk.

Módosítsuk folyamatábránkat! Vegyünk fel három munkarekeszt! Az egyikben helyezzük el először a számot, majd mindig az aktuális hányadosokat! A másikba tegyük bele a kettőt, majd cseréljük ezt a következő prímszámra! A harmadik tárolóba pedig gyűjtsük a törzstényezőket, amelyeket az eljárás végén kiírathatunk.

Készítsük el a javított blokkdiagramunkat önállóan!

2. Készítsük el a téglalap kerületének és területének kiszámítására vonatkozó blokkdiagramot *(15. ábra)*!



15. ábra

Ezen a folyamatábrán észrevehetjük, hogy a számítógéppel megoldott feladatok műveletsora három fő részből áll.

1. A kiinduló adatok bevitele.
2. A bevitt adatok feldolgozása.
3. A kapott eredmények kiíratása.

A program elkészítésének lépései

Az eddig megbeszéltek alapján foglaljuk össze, hogy a számítógép programjának elkészítéséhez milyen lépésekre van szükség.

1. A megoldandó problémát elemeznünk kell. Meg kell nézni, hogy milyen adatokból, mit szeretnénk meghatározni, milyen lépéseken keresztül érjük el célunkat, és milyen formában szeretnénk megkapni az eredményünket.
2. Elkészítjük a folyamatábrát (nem függ a géptől).
3. A blokkdiagram alapján megírjuk a programot (egy adott gépre szabott leírás).

Olvasmány

Miért teszünk különbséget program és algoritmus között?

Az algoritmus általános formájú, semmilyen gépen nem működik, de könnyen írhatunk belőle bármely számítógépre programot. A programok géphez és programozási nyelvhez kötöttek, így más gépeken csak módosítással működnek.

Feladatok és kérdések

1. Készítsünk olyan algoritmust, amely segít egy robotot adott cikkek megvásárlásában egy közértben!
2. Készítsük el egy sütemény készítésének algoritmusát!
3. Készítsünk algoritmust a kör kerületének és területének kiszámítására!
4. Készítsük el az osztálykirándulás megszervezésének algoritmusát!
5. Készítsük el annak a programnak az algoritmusát, amely kiválasztja három szám közül a legkisebbet, és ki is írja azt!
6. Fogalmazzunk meg feladatot egy robot számára, és tervezzük meg a feladat elvégzésének algoritmusát!
7. Milyen feladatok nem alkalmasak számítógépes feldolgozásra?
8. Mit nevezünk algoritmusnak?
9. Mit nevezünk programnak?
10. Milyen jelöléseket használhatunk a blokkdiagram készítésekor?
11. Milyen fő részekből állnak a számítógépekkel megoldott feladatok?
12. Milyen lépések szükségesek egy program elkészítéséhez?

3. A BASIC-ben használt adatok típusai

Megismerkedtünk számítógépünkkel, néhány parancsot is tudunk használni, és számológépként is működtetjük már.

Kifejezéseinkben eddig csak számokat, úgynevezett állandókat vagy más szóval konstansokat használtunk. Az aritmetikai kifejezésekben azonban találhatunk konstansokon kívül változókat és függvényeket is, melyeket matematikai műveleti jelekkel kapcsolunk össze. Ezeknek a kifejezéseknek is pontosan meg tudjuk határozni az értékét a műveleti sorrend betartásával, ha a kiszámítás időpontjában ismerjük a változók, illetve a függvények argumentumainak értékét.

Például: T = 25 · 30 *helyett* T = A · B-*t írunk. A* T *változó értéke például* 750 m^2 *lesz, ha az oldalak* 25 m *és* 30 m *hosszúak, vagyis az* A *változó értéke* 25, a B *változó értéke* 30 *volt a számítás megkezdése előtt.*
Ha az oldalhosszúság 42 m *és* 13 m, *vagyis* A = 42 m *és* B = 13 m *volt, akkor a terület* 546 m^2 *lesz, és így tovább. (Ezt kalkulátor üzemmódban könnyen ki is számolhatjuk, például* PRINT 25*30.)

Példák

Írjuk fel a következő matematikai kifejezésekhez a megfelelő parancsokat, és számoljuk ki a kifejezések értékét, ha $a = 56$, $b = 193$, $c = -21$!

A számolást végezzük el úgy is, hogy a-nak, b-nek, c-nek más aktuális értéket adjunk!

Matematikai alak	*A gépnek adott parancs*
$a + \frac{b}{c}$	PRINT A + B/C
$\frac{a+b}{c}$	PRINT (A + B)/C
$\frac{a}{b \cdot c}$	PRINT A/(B * C)
$\frac{a \cdot b}{a+b}$	PRINT A * B/(A + B)
$\frac{1}{\frac{1}{a}+\frac{1}{b}}$	PRINT 1/(1/A + 1/B)

Megjegyzés

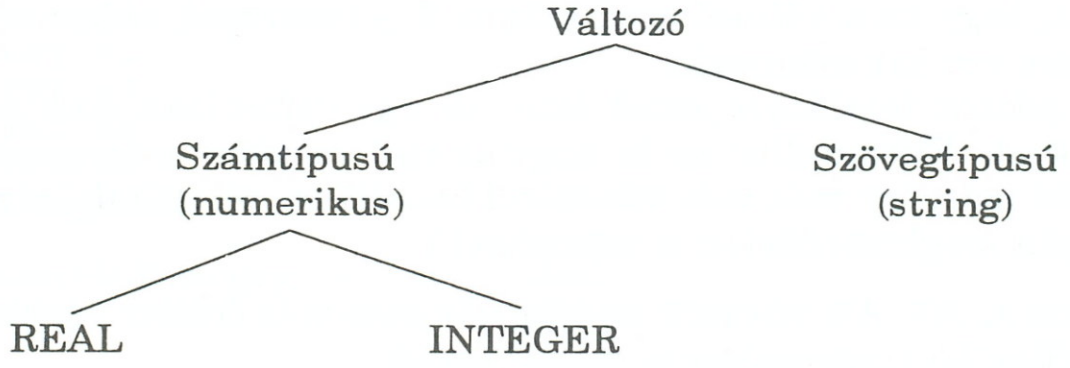
Most minden számolásnál be kellett gépelnünk az egész kifejezést, és ez lassítja a munkánkat. Később visszatérünk feladatunkra, program segítségével egyszerűsítjük majd számolásunkat.

a) A változók típusai és jelölésük

A számítástechnikában a változót nemcsak egy betű jelölheti, hanem egy több betűből álló név is. A változók egy meghatározott értéktartományon belül igen sokféle értéket vehetnek fel.

A változókat megkülönböztethetjük egymástól attól függően, hogy milyen típusú információt tartalmaznak, és milyen műveleteket végezhetünk el velük.

Leggyakrabban használt változótípusok:



— A real típusú változók közül bizonyosak géptől függőek. *Például:* egy adott intervallumon belüli törtek értékeit vehetik fel. Jellemző műveleteinek egyike az egészrész képzése.
— Az integer egészértékeket jelöl, saját művelete *például:* a maradékos osztás.
— A stringváltozók tetszőleges betűket, jeleket és számokat (karaktersorozatot) tartalmaznak.

A változókat azonosítójuk, nevük segítségével különböztethetjük meg egymástól. Az azonosítók végén egy speciális jel, a típusazonosító különbözteti meg a gép számára a változó típusát.

REAL (valós szám):	nincs típusazonosító	*(például:* A, X*)*
INTEGER (egész szám):	%	*(például:* L %*)*
STRING (szöveg):	$	*(például:* B $*)*

Változónévként bizonyos gépeken csak két karaktert használhatunk, amelyek közül az első mindig betű, a második lehet betű is és szám is. Általánosan is igaz, hogy a változók neveinek hossza korlátozott.

Például:	*változónév lehet:*	B	AB	B3	XB	X7	ALMA	*stb.*
	nem lehet:	1	3B	B?	IF	OR	...	*stb.*

Nem tartalmazhat olyan részletet, amely a BASIC nyelv valamelyik szavával megegyezne, mert az egybeesés miatt félreérthető lenne.

A változóneveket választhatjuk három- vagy több-betűsnek is. Ügyelni kell azonban arra, hogy az egyes személyi számítógépek csak az első kettőt veszik figyelembe, és ez hibákra ad lehetőséget (memóriapazarlás is).

Például: a KAPU, KATONA, KALAP

változókat a C-64-es számítógép ugyanannak a KA *változónak értelmezi a program végrehajtása során.*

Egy változónak az értéke számtípus esetén nulla, szövegtípus esetén üres string (nem tartalmaz karaktert), mindaddig, amíg nem adunk neki értéket. Ügyeljünk arra, hogy ha a változó neve mögött %-jel szerepel, akkor a változónak csak egész értéket adhatunk.

A változók adatok tárolására jöttek létre, de egy változóban csak egyféle adatot tárolhatunk. Előfordulhat az is, hogy az első karakter megegyezik, de a típusazonosító más, így más-más változóról beszélünk. (A számítógép a %- és a $-jel alapján megkülönbözteti a változókat.)

Például: az A, A$, A% *változók mindegyike más-más értéket képviselhet. (Később programban is szemléltetjük.)*

Szöveges változók

A numerikus változókkal számokat írhatunk ki, és numerikus műveleteket végezhetünk velük. A szövegtípusú változók pedig szöveget, karaktersorozatot tudnak kezelni. A stringek karakterek sorozatából állnak, melyek betűkből, számjegyekből, egyéb jelekből épülnek fel. A karaktersorozatot mindig idézőjelbe kell tenni.

Például: a LET C$="CUKOR"

Jelentése: a C$ *változó értéke legyen a "CUKOR" szöveg. A szöveges változók alkalmazását a következő program szemlélteti.*

Futtassuk a programot!

```
10 LET A$="AGI"
20 LET B$="1989"
30 LET C$="SIESS!"
40 LET D$="4 db DIO"
50 PRINT B$
60 PRINT D$
70 PRINT A$
80 PRINT C$
```

A képernyőn a futtatás után a következők jelennek meg:

```
1989
4 db DIO
AGI
SIESS!
```

Láthatjuk, hogy az A$ szöveges változó értéke AGI, B$ értéke 1989 és így tovább.

Egy szöveges változóban általában maximálisan 255 karakter hosszúságú szöveget helyezhetünk el.

A QBASIC-ben futtatható programunk:

```
LET A$="AGI"
LET B$="1989"
LET C$="SIESS!"
LET D$="4 db DIO"
PRINT B$
PRINT D$
PRINT A$
PRINT C$
```

F5 lenyomása

Eredményünk megegyezik az előzővel.

Az összeadás és a relációk értelmezése szövegek esetén

A BASIC-ben szöveges változókkal is végezhetünk műveleteket, közöttük is értelmezhetünk relációkat.

a) Az összeadással azonos jel jelöli az egymás után írást vagy összeláncolást, konkatenációt.
A jele tehát +, de operandusai szövegtípusúak, ezért nem lehet őket összekeverni az aritmetikai összeadással.

1.
```
10 LET A$="KECSKE"
20 LET B$="BEKA"
30 PRINT A$+B$
40 PRINT B$+A$
```

Futtatás után a képernyőn megjelenik: KECSKEBEKA
BEKAKECSKE

2.
```
10 LET A$="5"
20 LET B$="8"
30 PRINT A$+B$
40 PRINT B$+A$
```

Futtatás után a képernyő képe: 58
85

Láthatjuk, hogy ez az „összeadás" nem kommutatív, figyeljünk tehát az operandusok sorrendjére!

b) A számítógép el tudja dönteni két szövegről, hogy egyenlőek-e. Ekkor a stringeket jelenként összehasonlítja.

Például: A$ = "ASZTAL"
B$ = "ASZ TAL"

a számítógép számára nem ugyanazt jelenti, ezért nem egyenlők.

c) Szövegekre a kisebb—nagyobb relációt is alkalmazhatjuk. Összehasonlításkor sorra vizsgálja a számítógép a szavakban előforduló betűket, addig, amíg különbözőt nem talált. A számítógép azt a betűt választja kisebbnek, amely az ábécében előbb van. Segítségével *például* névsorba rendezést tudunk megvalósítani.

Például: A$="ANDREA"
B$="ANDRÁS" esetén B$<A$.

Szövegekkel más műveleteket is végezhetünk, de ezekkel most nem foglalkozunk.

A QBASIC-ben futtatható programunk:

```
1.  LET A$="KECSKE"
    LET B$="BEKA"
    PRINT A$+B$
    PRINT B$+A$
    F5 lenyomása
```

```
2. LET A$="5"
   LET B$="8"
   PRINT A$+B$
   PRINT B$+A$
   F5 lenyomása
```

A futtatás eredménye azonos az előzővel.

Kérdések

1. Mit nevezünk változónak, és milyen típusú változókról beszélünk?
2. A következő kifejezések közül melyik lehet a BASIC-ben változónév?

   KB; X; 5C; FORMA; B6; ORVOS; BETEG; 3; C.

3. Milyen típusazonosítókat kell használni az egyes változók megkülönböztetésére?
4. Mit jelent a szöveges változók „összeadása", illetve összehasonlítása?

C) BASIC utasítások

Eddig már találkoztunk néhány BASIC utasítással, de ezekkel részletesebben csak most ismerkedünk meg.

1. Értékadó utasítás

Gyakran előfordul, hogy változóhoz egy konstanst, értékkel rendelkező változót vagy kifejezést rendelünk. Ekkor értékadásról beszélünk.

Alakja a következő:

sorszám LET változó = kifejezés.

Egyes géptípusoknál:

sorszám változó = kifejezés,

ami azt jelenti, hogy a LET *nem kötelező, nélküle is végrehajtja a számítógép az értékadást, de ne éljünk ezzel a lehetőséggel.*

Például:

```
10 LET A = 527.3+6.8
20 LET A% = 17
30 LET A$ = "AUTO"
40 PRINT A,A%,A$
```

Az utasítások végrehajtása után A *értéke* 534.1, A% *értéke* 17, A$ *értéke pedig* AUTO *lesz, amit a képernyőre kiírt* 534.1 17 AUTO *is mutat.*

Ugyanez a felirat jelenik meg a következő program hatására is. (Ellenőrizzük!)

```
10 A = 527.3+6.8
20 A% = 17
30 A$ = "AUTO"
40 PRINT A,A%,A$
```

QBASIC-ben futtatható programunk:

```
LET A = 527.3+6.8
LET A% = 17
LET A$ = "AUTO"
PRINT A,A%,A$
```

Futtassuk programunkat a LET elhagyása után is!

Megjegyzés

Ezeket az értékadásokat parancs üzemmódban is végrehajthatjuk, ugyanezt az eredményt kapjuk. Ellenőrizzük!

A változó értéke tehát az értékadás jobb oldalán álló kifejezés értéke lesz.

Először kiértékelődik az egyenlőség jobb oldalán lévő kifejezés, majd a bal oldalon lévő változó felveszi ezt az értéket, miközben elveszti előző értékét. Ennek lényegét akkor érthetjük meg, amikor a matematikában szokatlan „egyenlőséggel" találkozunk.

Például: a `10 LET X=X+1` .

(Ilyen egyenlőség miatt nem célszerű elhagyni a LET *kulcsszót.)*

Ilyen utasítás hatására a számítógép kiolvassa a tárból X értékét, hozzáad egyet, majd elfelejti az előző értéket, és ezt az új értéket helyezi vissza a tárban lévő X jelű rekeszbe. Ez az utasítás tehát annyit jelent, hogy X új értéke legyen egyenlő az eddigi értékénél eggyel nagyobbal!

Példa

Próbáljuk ki a következő programunkat és magyarázzuk meg, hogy miért ezek az értékek kerültek a képernyőre!

```
10 PRINT A
20 LET A=13
30 PRINT A
40 LET A=A+3
50 PRINT A
```

QBASIC-ben futtatható programunk:

```
PRINT A
LET A=13
PRINT A
LET A=A+3
PRINT A
```

Vigyázzunk az értékadásnál, mert nem mindegy, hogy mit írunk az értékadó egyenlőségjel bal, illetve jobb oldalára!

```
9 LET C=A
10 LET A=B
20 LET B=C
```

utasítások hatására kicserélődik a változók értéke.

Futtassuk programban az előző három sort! Írjuk még be utána a következőket:

```
4 INPUT "KÉREK KÉT SZÁMOT"; A,B
6 PRINT "A="; A
8 PRINT "B="; B
30 PRINT "A="; A
40 PRINT "B="; B
```

QBASIC-ben a programunk:

```
INPUT "KÉREK KÉT SZÁMOT"; A,B
PRINT "A="; A
PRINT "B="; B
LET C=A
LET A=B
LET B=C
PRINT "A="; A
PRINT "B="; B
```

a) Az adatok bevitelét segítő utasítás

Általában egy programot különböző adatokkal többször is szeretnénk használni, esetleg olyan adatokkal dolgozunk, melyeket a program írásakor még nem ismerünk. Ilyenkor az értékadást célszerű minden futás alkalmával újra és újra elvégezni. Ehhez nyújt segítséget az INPUT utasítás. Alakja a következő:

sorszám INPUT azonosítók felsorolása

Az INPUT utasítás után tetszőleges számú azonosítót írhatunk, melyeket vesszővel kell elválasztani.

Amikor programunk futásakor az INPUT utasításhoz érünk, a képernyőn megjelenik egy kérdőjel (?) vagy más jel, mellyel a gép megkér bennünket a változók értékeinek megadására.

Több változó esetén, ha egy adatot gépelünk be, a képernyőn megjelenő ?? jelzi, hogy a gép még adatot vár. A két kérdőjel addig marad a képernyőn, amíg az adatok száma egyenlő nem lesz a változók számával.

IBM számítógépeknél, ha több változónak szeretnénk értéket adni, akkor a megjelenő kérdőjelek után a változók számával egyező számú adat bevitelét fogadja csak el gépünk. Több, illetve kevesebb adat esetén is hibát jelez.

```
REDO FROM START
?
```

Célszerű azonban minden változóhoz saját INPUT utasítást használni, hogy mindegyikhez feltehessük a megfelelő kérdést is, így könnyítve meg a program használatát.

Az adatokat is vesszővel elválasztva kell beírni a gépbe, és az ENTER billentyűvel zárjuk a bevitelüket, vagy a ?? megjelenése után újabb adatot gépelünk be.

Például:

```
10 INPUT A,B,C$,D$
20 PRINT A
30 PRINT B
40 PRINT C$
50 PRINT D$
60 PRINT C$+D$
```

Futtatáskor megjelenik a képernyőn egy kérdőjel, ekkor beírhatjuk *például:* 23, 37, SÁRGA, RÉPA adatokat. A szöveges változó értékét idézőjel nélkül kell beírnunk.

Hibalehetőségek értékadáskor

Ha nem figyelünk eléggé az adatok beírására, akkor hibázhatunk, és ezt a számítógép számunkra valamilyen hibaüzenettel jelzi is. Ezek a következőkből adódhatnak:

— Előfordulhat, hogy több adatot írunk be, mint ahány változó volt az INPUT utasításnál, akkor a képernyőn a következő hibaüzenet jelenik meg:

```
? EXTRA INGORED
```

A kiírás után a számítógép a felesleges adatokat figyelmen kívül hagyva továbblép, és folytatja az utasítások végrehajtását.

— Az IBM számítógép több adat megadása esetén hibát jelez:

```
? REDO FROM START
```

A hibaüzenet megjelenése után megszakad a program futása, újból megjelenik a képernyőn a kérdőjel, a gép várja a jó adatokat.

— Egy másik hibalehetőség, hogy a változók és a bevitt adatok típusa nem egyezik meg, például számtípusú változó esetén szövegtípusú változót adunk meg vagy fordítva. Ekkor a gép hibát jelez, kiírja:

```
? REDO FROM START
?
```

A számítógép nem fogadja el a hibás adattípust, azt elhanyagolja, új értéket kér az INPUT után felsorolt összes változónak.

b) Az értékadás beszédessé tétele

Ha a programot nem mi írtuk, esetleg elfelejtettük, hogy milyen sorrendben írtuk be a változókat, akkor fejtörést okozhat a kérdőjel megjelenésekor az aktuális értékek beírása. Ennek elkerülése érdekében ki kell használnunk azt a lehetőséget, hogy az INPUT utasítást magyarázó szöveggel, kérdéssel láthatjuk el. Alakja a következő:

sorszám INPUT „magyarázó szöveg" ; azonosítók felsorolása

A számítógép az INPUT után idézőjelbe tett szöveget kiírja a képernyőre, így megtudjuk, hogy milyen adatokat kér.

Az idézőjel bezárása után pontosvesszőt kell tenni, ezzel jelezzük, hogy még nincs vége az utasításnak. A változókat vesszőkkel választjuk el egymástól.

Például:

```
10 INPUT A; B helyett áttekinthetőbb
10 INPUT "KEREM A TEGLALAP RÖVIDEBB,
   MAJD A HOSSZABB OLDALAT"; A; B
vagy
10 INPUT "MEKKORA A RÖVIDEBB OLDAL?"; A
20 INPUT "MEKKORA A HOSSZABB OLDAL?"; B
```

IBM PC számítógépeknél az utasítás alakja:

INPUT "üzenet" ; változó

Az üzenet az adatbekérés azonosítását segíti a megjelenő szöveg segítségével.

Pontosvessző helyett vesszőt is használhatunk, de ilyenkor csak a szöveg jelenik meg a képernyőn, a ? nem. Az adatokat bevihetjük a billentyűzetről, melyeket hozzárendelünk a változólistában felsorolt változókhoz. A program mindig annyi és olyan típusú adatot vár, ahány változó szerepel a listában.

Olvasmány

Az INPUT utasítás a szövegtípusú változóban történő beolvasásnál figyelmen kívül hagyja a bevezető szóközöket és a vessző vagy pontosvessző karakternél befejezi az adatbevitelt. Ha szóközöket és írásjeleket is be akarunk olvasni egy változóba, akkor a kérdőjel után az adatokat idézőjelek közé kell tenni.

Például:

```
10 INPUT A$
20 PRINT A$
```

A program futásakor a képernyőn a következő jelenik meg:

```
■ RUN
? "ADATOK: SÜT A NAP"   (Ezt mi írjuk be a megjelenő
                         kérdőjel után.)
ADATOK: SÜT A NAP
READY
■
```

Előfordulhat, hogy INPUT-ra váró programunkat szeretnénk valamilyen okból megállítani. Ezt a C-64 számítógépen csak a RUN/STOP és a RESTORE gombok együttes lenyomásával érhetjük el. Ezek hatására a program futása félbeszakad.

IBM PC számítógépeknél, ha a futási képernyőről szeretnénk átlépni a szerkesztőképernyőre, akkor nyomjuk le a CTRL és a PAUSE billentyűt egyszerre. Képernyőnkön a világos sor mutatja, hogy programunk futása az INPUT utasításnál tart.

2. A képernyő törlésének lehetősége

Programjaink futtatása során a képernyőn jelen maradt minden addig beírt adat, eredmény, az adatok beírása előtt megjelenő kérdőjel stb.

Mindez nehezítette számunkra az eredmény megtalálását.

Szeretnénk elérni, hogy a képernyőn csak a számunkra fontos kiírás maradjon meg, ezért a többit töröljük róla.

A képernyőt a PRINT után együttesen lenyomott SHIFT+CLR/HOME billentyűk segítségével kiadott utasítással vagy paranccsal törölhetjük. Az utasítás végrehajtásakor csak a képernyő tartalma törlődik, a kurzor a képernyő 0. sorának 0. oszlopába áll. Egy LIST parancs segítségével újból megjeleníthetjük programunkat.

IBM PC számítógépeknél a képernyőtörlést a CLS paranccsal végezhetjük el. Hatására a prompt a képernyő felső sarkába kerül a kurzorral együtt. Programban alkalmazva, utasításként, hatására a futási képernyőről törlődik minden, de a program futásának befejezése után, ha lenyomunk egy billentyűt, akkor a szerkesztőképernyőn megjelenik a beírt programunk.

3. A program törlésének módja

a) A programot a NEW parancs kiadásával törölhetjük a memóriából C-64 esetén.

IBM PC számítógépeknél a QBASIC fájl-menüjében megkeressük a New (Program) elnevezésűt, és segítségével törölhetjük a programot a számítógép memóriájából.

A törlés megvalósítása:

ALT	billentyű lenyomásának hatására az ablakos menürendszer FILE elnevezésű tagja egy fekete téglalapba kerül.
ENTER	hatására megjelenik egy újabb menüsor, melyből kiválasztjuk a NEW elnevezésűt.
↓	segítségével ráállunk a menü NEW elemére, itt jelenik meg a fekete téglalap.
ENTER	hatására a számítógép megkérdezi tőlünk, hogy ki akarjuk-e menteni a programot.
TAB	billentyűvel ráállunk az N válaszra.
ENTER	hatására a képernyőről is törlődik minden, csak a kurzor villog a bal felső sarokban.

b) A program törlésének biztos, ámde célszerűtlen módja, ha a számítógépet kikapcsoljuk. Az újbóli bekapcsolással kicsit várni kell, hogy ne okozzunk kárt a gépben.

Olvasmány

IBM PC számítógépeknél nemcsak a gép kikapcsolásával, hanem a gépen található RESET gomb benyomásával is törölhetjük a programot a memóriából. A RESET gomb benyomása után újra töltődik a DOS, de előző programunk nem.

Feladatok

1. Az előző programunkban töröljük a képernyőt az adatok kiírása előtt! 35 PRINT begépelése és a "SHIFT+CLR/HOME" billentyűk együttes lenyomása után futtassuk újból programunkat!
2. Alkalmazzuk az előző programjainkban a képernyőtörlést!

4. A hibák javítása (programszerkesztés)

Előfordul, hogy programírás közben hibázunk, mást gépeltünk be, mint amit akartunk vagy a számítógép jelez hibát, és ezt javítani szeretnénk. Ezért a következőkben a javítás, módosítás lehetőségeivel ismerkedünk meg.

a) Program módosítása

Előfordul, hogy az adott programsort nem akarjuk újragépelni, csak apróbb módosításokat szeretnénk végezni. Ilyenkor elegendő a kurzorvezérlő és a szerkesztőbillentyűk segítségével a megfelelő helyre állítani a kurzort, és máris módosíthatjuk az adott sort.

Csere:
C-64-en először a SHIFT és a CRSR billentyűk segítségével a kurzort a változtatandó sorra visszük, majd megkeressük velük a változtatandó karaktert is. Írjuk be ide a szerintünk megfelelő karaktert, nyomjuk le a RETURN billentyűt, és a módosított sor máris a programtárba került.

IBM PC számítógépeknél a kurzormozgató billentyűk (↓ ↑ → ←) segítségével a kurzort a változtatandó karakterre állítjuk. Benyomjuk az INSERT billentyűt. A változtatandó karakteren egy világos téglalap villog. Írjuk be ide a helyes karaktert, majd ismét nyomjuk le az INSERT billentyűt! Ezzel befejeztük a cserét, ha újabb sorba lépünk.

Beszúrás:
C-64-en vigyük a kurzort a SHIFT és a CRSR billentyűk segítségével a megfelelő helyre, majd nyomjuk le a SHIFT és az INST/DEL billentyűket! Ezzel egy üres helyet hozunk létre, ahová beírhatjuk az adott karaktert.

IBM PC-knél a kurzormozgató billentyűk segítségével a kurzort a megfelelő helyre állítjuk, arra a karakterre, ami elé írni szeretnénk. Most lenyomva a megfelelő billentyűt, az beíródik a megfelelő helyre. A kurzorral más sorra lépve már a módosult sort veszi figyelembe számítógépünk.

Ha elfelejtünk szóközt kihagyni, akkor ugyanezt a módszert kell követnünk, csak most a SPACE vagy szóköz billentyűt kell lenyomnunk.

Törlés:

C-64-en vigyük a kurzort a megfelelő helyre, majd nyomjuk le az INST/DEL billentyűt. Ekkor a kurzor előtti karakter törlődik úgy, hogy a helyén nem marad üres karakter.

IBM PC-knél a kurzort a megfelelő helyre állítjuk, majd lenyomjuk az INSERT billentyűt. A törlendő karakteren egy világos téglalap villog. Nyomjuk le a DELETE billentyűt, melynek hatására a villogó téglalap alatt lévő karakter törlődik úgy, hogy a helyén nem marad üres karakter, mert a törléssel egy időben a kurzortól jobbra lévő karakterek egyet balra „lépnek", tömörödnek.

b) **Javítás újragépeléssel**

— C-64-en a programsor javításának legegyszerűbb módja a sor újragépelése ugyanazzal a sorszámmal. Ilyenkor a régi sor automatikusan törlődik, az új kerül a helyére.

IBM PC számítógépeknél a kurzormozgató billentyűk segítségével a javítandó sorra állunk. Benyomjuk az INSERT billentyűt, és módosított sorunkat újragépelhetjük. Vigyázzunk arra, hogy ha módosított sorunk rövidebb, mint az eredeti, akkor a DELETE billentyűvel törölni kell a felesleges karaktereket. Megszüntetjük az insertállapotot, benyomjuk az INSERT gombot, majd másik sorra lépünk. Most már az új utasítássor az érvényes.

— C-64-en, ha csak egy sorszámot gépelünk be, és lenyomjuk a RETURN gombot, akkor az adott sor törlődik.

IBM PC-knél egy adott sort a CTRL és az Y billentyűk egyidejű lenyomásának hatására törölhetünk a programból.

— C-64-en, ha a program sorszámai közé egy új sorszámú sort illesztünk, akkor azt a gép automatikusan illeszti a megfelelő sorok közé.

IBM PC személyi számítógépeknél, ha a programba új sort szeretnénk illeszteni, üres sort kell létrehoznunk a helyén.

Ezt megtehetjük, ha

1. a kurzort annak a sornak az elejére állítjuk, amely elé új sort akarunk írni, és lenyomjuk a CTRL+N billentyűket. Hatásukra minden sor eggyel lejjebb lép, és meglesz az üres sor (itt is jó az ENTER leütése);

2. a kurzort annak a sornak a végére állítjuk, amely után szeretnénk elhelyezni az új programsort. Lenyomjuk az ENTER billentyűt. Hatására a kurzor a következő sor elején villog, amely üres, mert minden sor eggyel hátrább lépett.

Szerkesztésnél nagyon vigyázzunk az ENTER billentyű használatára, ha a kurzor nincs a sor végén, mert az ENTER lenyomása után attól a karaktertől kezdődően, ahol a kurzor állt, minden sor eggyel lejjebb kerül, tehát nem üres sor keletkezik, hanem a kurzortól jobbra eső karakterek alkotják az új sort.

5. A hibák észlelése

A C-64-en is előfordul, hogy a programsor beírásakor nem mindig vesszük észre a hibát. Előfordul, hogy a kész program átnézésekor sem tűnik fel, hogy valahol eltévesztettünk valamit. A program futtatásakor azonban a számítógép nem lép tovább, ha végre nem hajtható parancsot vagy utasítást talál. Ekkor a számítógép a képernyőre hibaüzeneteket ír számunkra az ERROR (hiba) felirattal, majd megadja a hiba helyét és típusát is. (A legfontosabb hibaüzeneteket magyar jelentésükkel együtt megtaláljuk a Függelékben.)

Az MS-DOS QBASIC-je lehetőséget ad arra, hogy segítséget kérjünk tőle. Az F1 billentyű lenyomása után a képernyőn megjelenik a Help ablakos rendszerű tájékoztatója. Ebből is megtudhatjuk a hiba okát (ha értünk angolul). Az ENTER billentyű lenyomása után javíthatjuk hibánkat.

A QBASIC azzal is segít bennünket, hogy a hibás sor mindaddig világos színűen villog, amíg az ENTER lenyomásával nem tudatjuk gépünkkel, hogy felismertük a hibát.

A QBASIC már az egyes programsorok beírásának befejezése után is jelzi, ha formai hibát vétettünk. A hiba helyén egy világos négyszög villog.

Feladatok

1. Próbáljuk ki a program módosításának lehetőségeit konkrét program futtatása közben!
2. Írjunk hibás programsorokat, hogy a számítógép a várt hibajelzést adja!

6. Az adatok kiíratását segítő utasítás

Láthattuk korábban, hogy a képernyőn való megjelenítést a PRINT utasítás segítségével valósíthatjuk meg. *Az utasítás általános alakja:*

sorszám PRINT kiírandó információk felsorolása

Nézzük meg, hogy milyen formákat valósíthatunk meg kiíratáskor a PRINT segítségével!

a) 10 PRINT — soremelés — az adott sort a képernyőn üresen hagyja, a kurzort a következő sor elejére állítja;

b) 10 PRINT 54 — a képernyőre kiírja az 54-es számot;

c) 10 PRINT 8+7 — a számítógép elvégzi az összeadást, majd kiírja a képernyőre a 15-ös számot;

d) 10 PRINT "szöveg" — a képernyőn az idézőjelek közötti szöveg jelenik meg változatlanul.

A *c)* és *d)* pontban leírt lehetőségek felhasználásával az elvégzendő feladat is megjeleníthető. *Például a*

```
10 PRINT "8+7="; 8+7
```

hatására a képernyőn 8+7=15 jelenik meg.

Egyetlen PRINT paranccsal több változót is kiírathatunk.

e) 10 PRINT A; B; C — a változókat pontosvesszővel választottuk el egymástól. Egy sorban, szorosan egymás mellett (egy pozíció marad üres) megjelenik az A, B és C változó értéke;

f) 10 PRINT A, B, C — a változók közé vesszőt tettünk. A változók értéke egy sorban, 10 karakterenként jelenik meg. A negyedik érték kiírása után a következő sor elején folytatja a kiírást. Így készíthetünk például táblázatokat.

IBM PC-t használva, ha vesszőt alkalmazunk, akkor a kiírás 12 karakterenként jelenik meg ugyanabban a sorban, és 5 érték kiírása után lép a másik sorba.

g) — Ha egy PRINT utasítást pontosvesszővel zárunk le, akkor a következő PRINT utasítás ott kezdi a kiírást, ahol az előző befejezte, vagy a következő INPUT utasítás innen olvas be.

— Ha a PRINT utasítást vesszővel zárjuk le, akkor az ezt követő megjelenítés, illetve beolvasás a következő tabulátorpozíción, azaz az *f)* pontban látottak szerint a következő oszlop elején kezdődik.

— Ha nem teszünk jelet a PRINT utasítás végére, akkor a megjelenítés, illetve beolvasás a következő sor elején kezdődik.

h) Kiíratásra más kulcsszót is használhatunk. A TAB(X) alkalmazásával kijelölhetjük, hogy az adott sor hányadik oszlopában kezdődjön a kiírás.

Alakja a következő:

10 PRINT TAB(8), A — A számítógép az adott sorban a 8. helytől kezdődően kiírja A változó aktuális értékét.

Vigyázzunk, hogy több TAB(X) esetén növekvő pozícióban írjuk a TAB változóit, mert csak így veszi figyelembe a számítógép.

Példák

1. *Futtassuk a következő programrészt és értelmezzük!*

```
48 PRINT TAB(9); "3"; TAB(18); "6"; TAB(27); "9"
84 PRINT TAB(18); "6"; TAB(9); "3"; TAB(27); "9"
```

C-64 esetén a 48. utasítás hatására a soron következő sor 9. oszlopában 3, a 18. oszlopában 6, a 27. oszlopában 9 jelenik meg (IBM PC-n ugyanez). A 84. utasítás hatására a sor 18. oszlopában 6, rögtön utána a 19. oszlopban a 3, és a 27. oszlopban a 9 íródik ki.

IBM PC-nél a 84. utasítás hatására az első sor 18. oszlopában megjelenik a 6, majd a második sor 9. oszlopában a 3, és a 27. oszlopában a 9 íródik ki, mert a kiviteli utasítás 18. pozíciója már a 9 mögött van, ezért új sorban folytatódik a kiírás.

2. *Írjunk programot, melyben megfigyelhetjük, hogy miben különböznek egymástól a kiíratás formái!*

```
10 REM   A KIIRATAS LEHETSEGES FORMAI
20 INPUT A,B,C
30 PRINT A
40 PRINT B
50 PRINT
60 PRINT C
70 PRINT
80 PRINT A;B;C
90 PRINT A,B,C
100  PRINT
110  PRINT A;B;C;
120  PRINT A: PRINT
140  PRINT A,B,C,
150  PRINT A
160  PRINT TAB(12);B;TAB(28);C
```

Ugyanezt a programunkat QBASIC-ben is futtathatjuk, ha elhagyjuk a sorszámozást, akkor is.

7. A program megjelenítése a képernyőn C-64 esetén

C-64 számítógépen programozás közben többször előfordul, hogy módosítani szeretnénk programunkat, de a program listáját nem láthatjuk, mert a program futása után a képernyőről eltűnik. A program listáját a LIST parancs segítségével tehetjük újból láthatóvá. (Természetesen, ha a parancs után lenyomjuk a RETURN billentyűt.)

Hosszabb programok esetén előfordul, hogy a képernyőn éppen azt nem látjuk, amit szeretnénk, mert nem fér rá. Ismerkedjünk meg a LIST parancs különböző lehetőségeivel. Próbáljuk ki őket az előző program segítségével!

Parancs	*Tapasztalat*
LIST	— a teljes program megjelenik a képernyőn.
LIST 30	— csak a 30. sort láthatjuk a képernyőn.
LIST 20–40	— kiírja a 20. sort és a 40 sort is, valamint a köztük lévő utasításokat.
LIST –30	— a képernyőn megjelenik a 30. sor és az összes előtte lévő.
LIST 30–	— ismét megjelenik a 30. sor, de most az összes utána lévőt láthatjuk.
LIST	— az utoljára beírt sor jelenik meg.

Az IBM számítógépeknél szerkesztést segítő billentyűkkel mozoghatunk a programsorok között.

— Ha a program elejére akarunk lépni, akkor nyomjuk le a CTRL+HOME billentyűt.
— Ha a képernyő aljára akarjuk állítani a kurzort, akkor nyomjuk le együtt a CTRL és az END billentyűt.
— Ha a képernyő teteje és az alja között keresünk egy sort, akkor a ↑ vagy a ↓ billentyűkkel lépkedhetünk soronként.

Feladatok

1. A következő feladat lehetőséget ad arra, hogy játsszunk a számítógéppel. Írjunk olyan programot, melyben a számítógép kitalálja, hogy milyen számra gondoltunk. A számítógép utasításai:
Gondolj egy számot! Szorozd meg kettővel! Adj hozzá 6-ot! Szorozd meg 4-gyel! Vonj ki belőle 8-at! Írd be a kapott eredményedet!
Egy lehetséges megoldás C-64 számítógépen:

```
5 PRINT "SHIFT+CLR/HOME"
10 REM SZAM KITALALASA
20 PRINT "GONDOLJ EGY SZAMOT!"
30 PRINT  "SZOROZD MEG KETTOVEL!"
40 PRINT "ADJ HOZZA HATOT!"
```

```
50 PRINT "SZOROZD MEG NEGGYEL!"
60 PRINT "VONJ KI BELOLE NYOLCAT!"
70 PRINT "IRD BE AZ IGY KAPOTT SZAMOT!"
80 INPUT A
85 PRINT "SHIFT+CLR/HOME"
90 PRINT "AZ ALTALAD GONDOLT SZAM:"
100 PRINT ((A+8)/4-6)/2
```

2. Készítsünk az előzőhöz hasonló kitalálós játékokat!
3. A kiíratő utasítás segítségével rajzoljunk a képernyőre!

Például: Tervezzünk karácsonyi üdvözlőlapot!

4. A következő program hatására mi jelenik meg a képernyőn, és miért? Megfelelő szövegek kiíratásával tedd érthetővé a programot!

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT  R
20 PRINT R^2*3.14
30 PRINT 2*R*3.14
```

IBM PC számítógépeknél az **1.** és a **4.** feladatban található programot sorszám nélkül is futtathatjuk, de ügyeljünk arra, hogy az 5-ös sorszámú sort írjuk át CLS-re, mert itt az jelenti a képernyő törlését.

8. Programozási lépések bemutatása körhenger segítségével

Járjuk most végig azt az utat, amelyet eddig megtettünk a következő feladat segítségével:

Számoljuk ki egy egyenes körhenger felszínét és térfogatát, ha alapkörének sugara 10 cm (R), magassága 15 cm (M)!

a) Konkrét adatok ismeretében elegendő kalkulátorként működtetni számítógépünket.

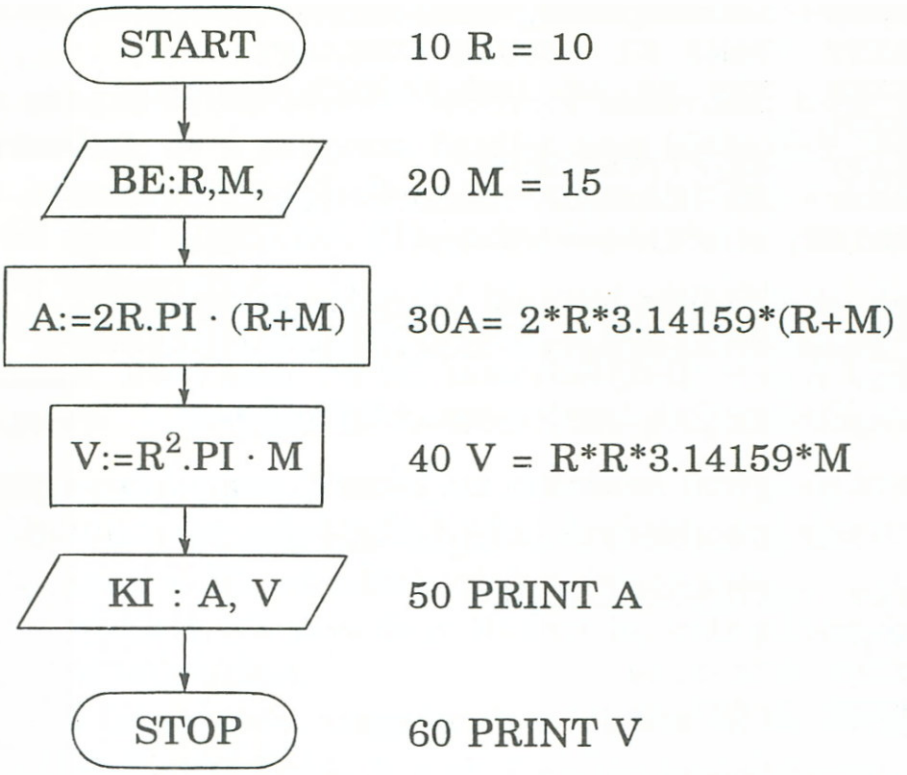
```
PRINT 2*10*3.14*(10+15)
```

A képernyőn megjelenik a felszín számértéke: 1570

```
PRINT 10*10*3.14*15
```

A térfogat számértéke: 4710

b) Ugyanezt az eredményt egy egyszerű program segítségével is megkaphatjuk.



16. ábra

c) Programunk bonyolultabb megoldást kínál a paranccsal történő kiszámolásnál, de több adat esetén egy kis átalakítással jobban használható, mert az R és M értékét nem kell mindig újra beírni a programba.
A program kezdete tehát így alakul:

```
10 INPUT R
20 INPUT M
.
.
.
```

A további sorok változatlanok.

d) Mindegyik kifejezésben szerepel π értéke, amely a számolás során állandó. Nem kell minden esetben begépelni az értéket, ha PI változóval helyettesítjük a kifejezésben és a PI változó értékét megadjuk a program elején.
Programunk a következő lesz:

```
10 INPUT R
20 INPUT M
30 PI = 3.14159
40 A = 2*R*PI*(R+M)
50 V = R*R*PI*M
60 PRINT A
70 PRINT V
```

e) Nem biztos, hogy tudjuk, melyik kiírt eredmény mit jelent, ezért ezt is adjuk meg.

Például:
```
55 PRINT "A FELSZIN:";
65 PRINT "A TERFOGAT:";
```

sorok segítenek ebben. Ugyanezt az adatok bevitelekor is megvalósíthatjuk az egyértelműség érdekében.

Például:
```
5 PRINT "SUGAR?",
15 PRINT "MAGASSAG?",
```

f) Tegyük most mindenki számára érthetővé a programot!

```
10 PRINT "KORHENGER FELSZINENEK ES TERFOGATANAK KISZAMOLASA"
20 PRINT
30 REM ADATOK BEVITELE
40 PRINT "KEREM A HENGER SUGARAT CM-BEN!";
50 INPUT R
60 PRINT "KEREM A HENGER MAGASSAGAT CM-BEN!";
70 INPUT M
80 PI=3.14159
90 REM A FELSZIN ES A TERFOGAT SZAMOLASA
100 A = 2*R*PI*(R+M)
110 V = R*R*PI*M
120 REM AZ EREDMENYEK KIIRATASA
130 PRINT "SHIFT+CLR/HOME"
140 PRINT
150 PRINT "A HENGER SUGARA R="; R; "CM"
160 PRINT
170 PRINT "A HENGER MAGASSAGA M="; M; "CM"
180 PRINT
190 PRINT "A HENGER FELSZINE A="; A; "CM*CM"
200 PRINT
210 PRINT "A HENGER TERFOGATA V="; V;"CM*CM*CM"
220 END
```

Példa

Térjünk vissza egy korábbi feladatunkra is!

A változók bevezetésénél kiszámoltuk több kifejezés aktuális értékét, de minden esetben be kellett gépelnünk a hosszú kifejezést. Írjunk programot, amely az adatok bevitele után elvégzi a kijelölt műveleteket!

```
10 PRINT "NEHANY KIFEJEZES SZAMERTEKE"
20 PRINT
30 INPUT "KEREK HAROM POZITIV SZAMOT!"; A, B, C
40 PRINT
50 D = A+B/C
60 E = (A+B)/C
70 F = A/(B*C)
80 G = A*B/(A+B)
90 H = 1/(1/A+1/B)
```

```
100 PRINT "A+B/C=";D
110 PRINT "(A+B)/C=";E
120 PRINT "A/(B*C)=";F
130 PRINT "A*B/(A+B)=";G
140 PRINT "1/(1/A+1/B)=";H
150 PRINT "AZ A="; A; "B="; B; "C="; C; "ADATOKKAL SZAMOLTAM"
```

Feladatok és kérdések

1. Írjunk programot, amely adott E egységár ismeretében kiszámolja tetszőlegesen megadott M árumennyiség árát!
2. Írjunk programot, amely az idő és a sebesség ismeretében kiszámolja a megtett utat!
3. Írjunk programot, amely kiszámolja egy adott A szám tetszőleges B százalékát! A program megírása előtt készítsünk folyamatábrát!
4. Írjunk „barátságos", azaz jól érthető programot, amely kiszámolja egy A Ft-ba kerülő áru árát, ha
   a) B %-kal emelkedik az ára; *b)* C %-kal csökken az ára!
5. Írassuk ki a térfogat mértékegységeit szöveges változók „összeadásának" segítségével!
6. Hogyan adhatunk egy változónak értéket?
7. Mit jelent a 17 LET A = A – 4 utasítás?
8. Hasonlítsuk össze az INPUT és a LET utasítást!
9. Mi történik, ha a program futásakor az INPUT utasításhoz ér?
10. Melyik hibás a következő utasítások közül?
    a) 18 INPUT A; B$; C
    22 PRINT A; B$; C
    b) 38 INPUT A, B$, C
    47 PRINT "A SZÁMOK" A; C
11. Mi a különbség a PRINT "SHIFT + CLR/HOME" és a NEW parancs között?
    IMB PC-k esetén mi a különbség a CLS és az ablakos menürendszer File pontjának New almenüje között?
12. Milyen lehetőségei vannak a program és a képernyő törlésének?
13. Milyen kiírási formát eredményez a pontosvessző, a vessző, a tab ( )?
14. Mi a különbség a PRINT 8+7 és a PRINT "8+7" parancsok között?
15. Milyen lehetőségeink vannak hibáink javítására?
16. Hogyan jutunk a képernyőn a szerkesztendő helyre?
17. Hogyan cserélhetünk, törölhetünk egy karaktert a programból, és hogyan írhatunk új betűt a meglévők közé?
18. Milyen jelenségek utalhatnak a program hibájára?
19. Hogyan tudjuk megállítani futás közben a programunkat?
20. Hogyan tudunk segíteni a felhasználónak a program futása közben, hogy tudja, a kérdőjel megjelenésekor milyen adatokat kell beírnia?
21. Mit jelent az X=X+1 egyenlőség a számítástechnikában?
22. Miért előnyös a programban konstansok helyett változókkal dolgozni?

9. A program vezérlése

Eddig csak olyan programokkal találkoztunk, amelyekben az utasítások egymásutánja egyértelműen meghatározta végrehajtásuk sorrendjét. A számítógép a legelső utasítástól sorban haladt a következő utasításokon át a program végéig.

Előfordul, hogy ettől a sorrendtől szeretnénk eltérni. Ilyenkor vezérlésátadó utasításokat alkalmazunk, mert segítségükkel elérhetjük, hogy a soron következőnél nagyobb vagy kisebb sorszámú soron folytatódjék a program futása.

a) Feltétel nélküli vezérlésátadás

Ha új adatokkal szeretnénk futtatni előző programjainkat, akkor újra kell indítanunk. Elérhetjük, hogy az eredmények kiíratása után a program elejére térjünk vissza, és a számítógép kérje az új adatokat. Más esetben nem a következő utasításon, hanem előbbin vagy később szeretnénk folytatni a program futását. Ekkor a GOTO utasítást kell alkalmaznunk.

Alakja a következő:

C-64 *esetén:*	sorszám GOTOn	itt n az a sorszám vagy az a cím, ahol a programot folytatni szeretnénk
IBM PC *esetén:*	GOTO	címke (Cím után :-ot kell tenni.)

Például: új:INPUT vagy 10:INPUT

Példa:

A képernyőre szeretnénk sokszor kiíratni, hogy „szeretlek".

```
10 PRINT "SZERETLEK"
20 GOTO 10
30 PRINT "GYŰLÖLLEK"
```

```
10:PRINT "SZERETLEK"
GOTO 10
PRINT "GYULOLLEK"
```

A 30. *számú sor egyszer sem jelenik meg, viszont a képernyőn egymás után íródik ki a* SZERETLEK *szó.*

IBM PC számítógépeknél, ha szeretnénk megállítani a program futását, benyomjuk a PAUSE billentyűt. Ilyenkor a képernyőn látható adatok sem „futnak tovább", nem íródik ki több, „merevvé" válik a képernyő. Leírhatjuk a számunkra szükséges adatokat. Ha tetszőleges, a PAUSE-től különböző billentyűt lenyomunk, a pogram futása ott folytatódik, ahol előzőleg abbamaradt.

Ha szeretnénk visszatérni a szerkesztőképernyőre, akkor nyomjuk le a CTRL és a PAUSE billentyűket egy időben. Visszatértünk programunkhoz, ahol a világos GOTO sor mutatja, hogy ott állt meg a programunk futása. Ha akarjuk, akkor módosíthatjuk utasításainkat, ha nem, akkor nyomjuk le az F5 billentyűt, és ugyanott folytatódik programunk futása, ahol az előzőekben félbeszakadt.

Példa: Írassuk ki a képernyőre a pozitív páros számokat!

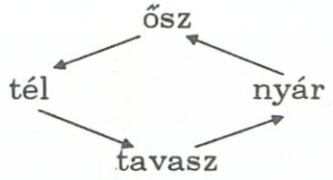
```
10 REM PAROS SZAMOK
20 LET I=2
30 PRINT I
40 LET I=I+2
50 GOTO 30
```

```
LET I=2
UJ:PRINT I
LET I=I+2
GOTO UJ
```

A GOTO utasítással mindkét programunkat végtelenítettük, végtelen sokszori ismétlődést hoztunk létre, azt mondjuk, hogy végtelen ciklust kaptunk. A programnak az ismételten végrehajtott utasítások által alkotott részét nevezzük a ciklus magjának.

Olvasmány

Életünk során mindnyájan találkozunk ciklikus ismétlődésekkel. Ezek egyike az évszakok szabályos visszatérése (17. ábra).



17. ábra

Ez a körfolyamat egy végtelen ciklus, melyben a sorrend nem cserélhető fel, a nyilak iránya nem változtatható. A nyár után ősz következik és nem tavasz, hogy újból nyár legyen.

Feladat: Keressünk mindennapi életünkben ilyen ismétlődő eseményeket!

b) **Feltételes vezérlésátadás** (döntések—elágazás)

A GOTO utasítás előbbi használatakor úgy avatkoztunk bele a program futásába, hogy a program adott helyétől egy másik helyre közvetlen, úgynevezett feltétel nélküli ugrást valósítottunk meg. Programunkat más módon is elágaztathatjuk. Valamely feltétel teljesülésétől vagy nem teljesülésétől tehetjük függővé, hogy milyen úton haladunk tovább. Ehhez nagyon sok esetben relációjeleket alkalmazunk.

Az alkalmazható relációjelek:

<	kisebb,	<=	kisebb vagy egyenlő,
>	nagyobb,	>=	nagyobb vagy egyenlő,
=	egyenlő,	<>	nem egyenlő.

A relációjelekkel összehasonlíthatjuk konstansok, változók és aritmetikai kifejezések, illetve stringek értékét, és ekkor vagy igaz, vagy hamis állításokhoz jutunk.

Például: 10<12 igaz, de 3*4<5*2 hamis.

A logikai értékektől függő vezérlésátadó utasítást nevezzük feltételes vezérlésátadó utasításnak.

Az utasítás a legegyszerűbb esetben a következő alakot veszi föl:

sorszám IF kifejezés relációjel kifejezés THEN utasítás

A kifejezéseket összehasonlítjuk, és ekkor, ha (IF) a két kifejezés összehasonlításával kapott állítás, a reláció igaz, akkor (THEN) a program a THEN utáni utasításokkal folytatódik. Ha a reláció nem teljesül, akkor a THEN-t követő parancsokat a számítógép nem veszi figyelembe és az azokat követő első utasításra ugrik, és azt hajtja végre, ha másként nem rendelkezünk.

Olvasmány

Nézzünk néhány példát az utasítás alkalmazására C-64 használatakor:

```
15 IF A>B THEN GOTO 78 ugyanezt eredményezi a
15 IF A>B THEN 78      utasítás is.
```

Azt jelenti, hogy ha A *nagyobb, mint* B, *akkor a* 15. *sorról a* 78. *sorra ugrik a számítógép a program futásakor, és az ott található utasítást hajtja végre.*

Példák

1. Ha egy kör kerületét, területét számoljuk, az eddigi ismereteink alapján a számítógép akkor is kiszámolja azokat, ha a kör sugara 0 vagy negatív szám volt.

   Tudjuk, hogy ilyen esetben a feladatnak nincs értelme, ezért ezt az esetet ki kell zárnunk.

1. *Írjuk meg így a programunkat! C-64-re vonatkozóan ez a következő lehet:*

```
10 INPUT "A KOR SUGARA:"; R
20 IF R<= 0 THEN 80
30 LET T = R^2*3.14
40 LET K = 2*R*3.14
50 PRINT "A KOR SUGARA:"; R; "CM"
55 PRINT
60 PRINT "A KOR KERULETE:"; K; "CM"
65 PRINT
70 PRINT "A KOR TERULETE:"; T; "CMxCM"
75 GOTO 10
80 PRINT "NEM KELETKEZIK KOR"
90 PRINT"UJ ADATOT KEREK!"
100 GOTO 10
```

2. Programunk QBASIC-ben is futtatható, ha megfelelő címkéket helyezünk el.

```
UJ:  INPUT "A KOR SUGARA:"; R
     IF R<=0 THEN GOTO NEM
     LET T=R^2*3.14
     LET K=2*R*3.14
     PRINT "A KOR SUGARA:"; R; "CM"
     PRINT
     PRINT "A KOR KERULETE?"; K; "CM"
     PRINT
     PRINT "A KOR TERULETE:"; T; "CM*CM"
     GOTO UJ
NEM: PRINT "NEM KELETKEZIK KOR"
     PRINT "UJ ADATOT KEREK!"
     GOTO UJ
```

Ismét végtelen ciklushoz jutottunk. Ha csak egy adattal számolunk, akkor érdemes a terület kiíratása után törölni az ugróutasítást, és F5-tel újraindítani a program futását.

2. Írjunk olyan programot, amely két számot összehasonlít, és kiírja az értékelés eredményét!

1. *C-64 esetén ez a következő lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "KEREM A KET SZAMOT:"; A, B
20 IF A=B THEN PRINT "EGYENLO A KET SZAM" : GOTO 50
30 IF A<B THEN PRINT A; "KISEBB"; B : GOTO 50
40 PRINT A; "NAGYOBB"; B
50 END
```

2. *Programunk QBASIC megfelelője lehet:*

```
INPUT "KEREM A KET SZAMOT!"; A, B
IF A=B THEN PRINT "EGYENLO A KET SZAM": END
IF A<B THEN PRINT A; "KISEBB"; B ELSE PRINT A; "NAGYOBB"; B
```

c) Többirányú elágazás lehetősége QBASIC-ben

A QBASIC nyelv lehetővé tesz többirányú elágazást is.
Az utasítás alakja:

1. IF kifejezés THEN i.utasítás [ELSE h.utasítás]

vagy

2. IF kifejezés THEN
   [i.utasítás]
   ELSEIF kifejezés THEN
   [i.utasítás]
   .
   .
   ELSE [h.utasítás]
   END IF

A kifejezés azt a feltételt tartalmazza, amelyre a vizsgálatot végre akarjuk hajtani. Az i. utasítást akkor hajtja végre számítógépünk, ha a feltétel vizsgálata igaz eredményre vezet. A h. utasítást akkor veszi figyelembe gépünk, ha vizsgálata hamis eredményre vezet. Az első esetben, ha igaz a feltétel, akkor az igaz, ha nem igaz a feltétel, akkor a hamis utasítást hajtja végre számítógépünk. A második esetben végrehajtunk egy utasítássorozatot mindaddig, amíg a feltétel vizsgálatakor igaz eredményhez jutunk, egyébként a hamis utasítás hajtódik végre.

Előző feladatunk megoldásának másik lehetősége:

```
INPUT "KEREM A KET SZAMOT!"; A, B
IF A=B THEN
     PRINT A; "EGYENLO"; B
ELSEIF A<B THEN PRINT A; "KISEBB"; B
ELSEIF A>B THEN PRINT A; "NAGYOBB"; B
END IF
```

Példa:
Készítsünk olyan programot, amely lehetővé teszi, hogy mi döntsük el, hogy a bevitt két számot összeadjuk-e, kivonjuk-e, szorozzuk-e vagy osztjuk-e egymással!

QBASIC-ben erre egy lehetséges program:

```
PRINT "ADJON MEG KET SZAMOT!"
INPUT a, b
PRINT "IRJA BE, HOGY MILYEN MUVELETET VEGEZZEK!"
PRINT "OSSZEADAS, KIVONAS, SZORZAS, OSZTAS"
INPUT a$
IF a$="OSSZEADAS" THEN PRINT "a+b="; a+b:END
IF a$="KIVONAS" THEN PRINT "a-b="; a-b:END
IF a$="SZORZAS" THEN PRINT "a*b=";a*b ELSE PRINT "a/b="; a/b
END
```

10. Ciklusok

A feltételes vezérlésátadást bemutató programjaink közben előfordult, hogy olyan programrészlethez értünk, amelyet a futás során többször szerettünk volna végrehajtani. Az utasítások ismételt végrehajtását ciklusnak, az ismétlődő utasítások sorozatát pedig ciklusmagnak nevezzük. Az ismétlődések számát a ciklusváltozók határozzák meg, ezért a ciklusmagok végrehajtása során feltétlenül szükség van legalább egy ciklusváltozóra. Ügyelnünk kell arra, hogy mielőtt belépünk a ciklusba, a ciklusmagban található változóknak értéket adjunk. Ha ez nem történik meg, akkor a numerikus változók értéke automatikusan 0, a string változóké „üres karakter". (A ciklusmagba kívülről beugrani vagy belülről kiugrani nem szabad, még akkor sem, ha ezt a nyelv lehetővé teszi.)

a) **A ciklusszervező utasítás formái C-64 esetén**

— A programírás egyszerűsítése érdekében a BASIC nyelv egy külön ciklusszervező utasítást is tartalmaz.

Az utasítás alakja a következő:

sorszám FOR ciklusváltozó=kezdőérték TO végérték

⋮ ciklusmag (utasítások sora)

sorszám NEXT ciklusváltozó

Példa: Írjuk át a ciklusszervező utasítás segítségével a pozitív számok kiíratására készített programunkat! (Az első 13 számot írja ki!)

```
10 PRINT "AZ ELSO 13 POZITIV SZAM"
20 FOR I=1 TO 13
30 PRINT I
40 NEXT I
50 PRINT "VEGEZTUNK A KIIRASSAL"
```

Mi történik az adott utasítások hatására a számítógépben? A 20. sor azt jelenti, hogy az I változó értéke először 1 lesz , és innen megy egyesével 13-ig. Az I változó értékét a 30-as sor szerint kiírja a számítógép: megjelenik az 1. A 40. sorban lévő NEXT utasítás hatására az I változó értéke eggyel nő, majd a számítógép megnézi, hogy I értéke nagyobb-e 13-nál (2<13). Ha I értéke nem lépte túl a 13-at, akkor a ciklusmag újra végrehajtódik a FOR utasítást követő sortól (tehát kiírja a 2-t...). Mindez addig folytatódik, amíg I értéke 14 nem lesz, mert ezután a NEXT utáni sort hajtja végre, „kilép a ciklusból", vagyis kiírja: VÉGEZTÜNK A KIÍRÁSSAL.

— Előfordulhat, hogy a ciklusváltozó nem egyesével változik: ekkor meg kell adni a lépésközt, amellyel a ciklusváltozó a NEXT hatására megváltozik.

Az utasítás alakja a következő:

sorszám FOR vált=kezdőért. TO végért. STEP lépésköz

⋮ ciklusmag

sorszám NEXT ciklusváltozó

Példa: Az előző feladatunkat módosítsuk annyival, hogy a pozitív páratlan számokat írassuk ki 18-ig!

```
10 PRINT "PARATLAN SZAMOK 18-IG"
20 FOR I=1 TO 18 STEP 2
30 PRINT I
40 NEXT I
50 PRINT "VEGEZTUNK A KIIRASSAL"
```

Most a NEXT utasítás után I értéke minden végrehajtáskor 2-vel nő, és ezután végzi a hasonlítást a számítógép. A program addig fut, amíg I értéke a 19 értéket fel nem veszi. Ekkor megjelenik a képernyőn, hogy VÉGEZTÜNK A KIÍRÁSSAL.

— Megadhatunk negatív lépésközt is, de ekkor vigyáznunk kell a végérték megadásánál, hogy kisebb legyen a kezdőértéknél. A ciklusváltozó ebben az esetben lépésenként csökken.

Példák

1. Írjunk programot, amely kiíratja a pozitív egész számokat 12-től visszafelé!

```
10 PRINT "VISSZAFELE SZAMOLAS"
20 FOR I=12 TO 1
30 PRINT "I="; I
40 NEXT I
50 PRINT "KIIRTAM A SZAMOKAT"
```

A program futtatásakor azt tapasztaljuk, hogy a „Kiírtam a számokat" kiírása után befejeződik a program végrehajtása. Ennek oka, hogy az automatikus STEP megadása nélküli lépésköz 1, ekkor a ciklusmag első végrehajtása után teljesül, hogy 13>1, tehát végzett a gép.

Javítás: `20 FOR I=12 TO 1 STEP -1`

A többi sor változatlan marad. Programunkat futtatva a várt eredményt kapjuk.

```
VISSZAFELE SZAMOLAS
I=12
I=11
.
.
.
I=1
KIIRTAM A SZAMOKAT
```

2. A következő program segítségével bemutatjuk a ciklusképző utasítás működését a különböző esetekben, ezért nem adjuk meg előre a kezdő- és végértéket sem.

```
10 REM CIKLUSKEPZO UTASITAS MUKODESE
20 INPUT "KEREM A KEZDOERTEKET"; K
30 INPUT "KEREM A VEGERTEKET"; V
40 INPUT "IRJA BE A LEPESKOZT!"; L
50 FOR I=K TO V STEP L
60 PRINT I
70 NEXT I
80 PRINT "KILEPUNK A CIKLUSBOL"; I; "ERTEKNEL"
```

Futtassuk programunkat a következő adatokkal!

a) K=3, V=17, L=1,
b) K=3, V=17, L=2,
c) K=17, V=3, L=–2.

Próbáljuk ki más értékekkel is!

3. Gyakran találkozunk a kamatos kamat fogalmával. Ezt például akkor kaphatjuk, ha pénzünket több évre betesszük az OTP-be. Ekkor az első évben eredeti pénzünk növekszik a kamattal, a következő évben az így megnövelt összeg növekszik és így tovább. Írjunk programot, amely kiírja, hogy egy adott pénzösszeg adott kamatláb mellett mennyi lesz az év végén. Írassuk ki az első 4 évben a megnövelt értéket!

```
10 PRINT "KAMATOS KAMAT SZAMITASA"
20 INPUT "IRJA BE A KEZDO TOKET!"; K
30 INPUT "ADJA MEG A KAMATLABAT!"; L
40 PRINT "A KEZDO TOKE:"; K; "FT"
50 PRINT "A KAMATLAB:"; L; "%"
60 PRINT: PRINT
70 PRINT "EV", "MEGNOVELT ERTEK (FT)" : REM FEJLEC
80 FOR I=1 TO 4
90 S=K*((1+L/100)^I)
100 PRINT I, S
110 NEXT I
120 END
```

b) A ciklusszervező utasítás PC számítógépeken

A QBASIC nyelvben ugyanígy alkalmazhatjuk a ciklusszervező utasítást, ezért előző programjaink ugyanígy futnak IBM PC számítógépen is.

Lehetőségünk van azonban két másik utasítás alkalmazására is. Ismerkedjünk meg ezekkel! A ciklusmag végrehajtása mindkét esetben egy logikai kifejezés értéke alapján történik.

a) A ciklusmagot addig kell végrehajtani, amíg a logikai kifejezés igaz, amíg a feltétel hamissá nem válik. Ekkor befejeződik a ciklus végrehajtása, és a következő utasításon folytatódik a program futása.

— A logikai kifejezés értékét az első végrehajtás előtt is megvizsgáljuk, így előfordulhat, hogy egyszer sem fut a ciklus.

Az utasítás formája:

DO WHILE logikai kifejezés
  utasítások
LOOP

Példák

1. Írjuk át az első 13 szám kiíratását megvalósító programunkat!

```
PRINT "AZ ELSO 13 POZITIV SZAM"
i=1
DO WHILE i<14
     PRINT i
     i=i+1
LOOP
PRINT "VEGEZTUNK A KIIRASSAL"
```

2. Módosítsuk programunkat úgy, hogy tőlünk függjön, honnan kezdődjön a számok kiírása 13-ig!

```
PRINT "HONNAN KEZDJEM A SZAMOK KIIRASAT?"
INPUT i
DO WHILE i<14
     PRINT i
     i=i+1
LOOP
PRINT "VEGEZTUNK"
```

Próbáljuk ki programunk futását, ha a bevitt érték 14 vagy nála nagyobb! Láthatjuk, hogy a képernyőre csak az kerül: VÉGEZTÜNK.

— Ezt az utasítást úgy is alkalmazhatjuk, hogy először végrehajtjuk a ciklusban belüli utasításokat, és azután vizsgáljuk a logikai kifejezés igazságtartalmát. Egyszer mindenképpen lefut a ciklusmag.

Az utasítás alakja:

```
DO
  utasítások
LOOP WHILE logikai kifejezés
```

Feladat: Írjuk át a **2.** *példa* programját a most megismert utasítás felhasználásával!

```
PRINT "HONNAN SZAMOLJAK?"
INPUT i
DO
       PRINT i
       i=i+1
LOOP WHILE i<14
PRINT "VEGEZTUNK"
```

Programunk futtatásakor azt tapasztaljuk, hogy 14 vagy annál nagyobb érték megadásakor kiírja az adott értéket, és csak utána írja, hogy VÉGEZTÜNK.

b) A ciklusmagot addig kell végrehajtani, amíg a logikai feltétel igazzá nem válik. Ekkor befejeződik a ciklus végrehajtása, és a program futása a következő utasításon folytatódik.

— Az első végrehajtás előtt megvizsgáljuk a logikai kifejezés értékét.
Az utasítás alakja:

```
DO UNTIL logikai kifejezés
        utasítások
LOOP
```

Példa: Írjuk át előző programunkat az új utasítással!

```
PRINT "HONNAN KEZDVE SZAMOLJAK?"
INPUT i
DO UNTIL i=14
      PRINT i
      i=i+1
LOOP
PRINT "VEGEZTEM"
```

Programunkat 0≤i<14 esetén futtatva megkapjuk a keresett számsort. i=14 esetén csak a VEGEZTEM kerül a képernyőre.
i>14 esetén végtelen ciklusba kerülünk, mert ilyenkor nem fordulhat elő az egyenlőség. Úgy tudunk kilépni a végtelen ciklusból, ha a Ctrl+Pause billentyűket együtt lenyomjuk. A szerkesztőképernyőn i értékét megváltoztatjuk olyan nagyra, hogy az addig kiírt értéknél nagyobb legyen, *például:* DO UNTIL i=1000. Megállás után visszaállíthatjuk az eredeti állapotot.
(A probléma elkerülhető, ha a DO UNTIL i>=14-et alkalmazzuk).

— Ezt az utasítást is alkalmazhatjuk úgy, hogy először végrehajtjuk a cikluson belüli utasításokat, és csak ezután vizsgáljuk a logikai kifejezést.
Az utasítás alakja:

```
DO
      utasítások
LOOP UNTIL logikai kifejezés
```

Példák

1. Írjuk át előző programunkat az új utasítással!

```
PRINT "HONNAN KEZDJEK SZAMOLNI?"
INPUT i
DO
      PRINT i
      i=i+1
LOOP UNTIL i=14
PRINT "VEGEZTEM"
```

Programunk futtatásakor azt tapasztaljuk, hogy már i=14 esetén is végtelen ciklushoz jutunk, egyébként a futás megegyezik az előzővel.

2. A kamatos kamat számolására írt programunkat írjuk át a most megismert ciklusutasítás felhasználásával!

```
INPUT "KEZDO TOKE:"; k
INPUT "KAMATLAB:"; g
PRINT "EV", "UJ ERTEK"
i=1
DO
      s=k*((1+g/100)^i)
      PRINT i, s
      i=i+1
LOOP UNTIL i=5
END
```

Olvasmány

Ciklusok egymásba építése

— *Találkozhatunk olyan esetekkel is, amikor több ciklussal tudjuk csak megoldani a problémánkat, és ezeket egymásba is építhetjük (skatulyázás). Ezek száma C-64 esetén maximálisan csak kilenc lehet.*

Példa: Írjunk programot, amely kiírja a képernyőre az egyszeregytáblázatot!

1.
```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "EGYSZEREGY"
20 FOR I=1 TO 4
30 FOR J=1 TO 4
40 PRINT I; "*"; J; "="; I*J
50 NEXT J
60 PRINT
70 NEXT I
80 END
```

2. QBASIC-ben futtatott programunk szerkezetéből jobban láthatjuk az egymásba építést.

```
CLS
PRINT "EGYSZEREGY"
FOR i=1 TO 4
     FOR j=1 TO 4
                PRINT i; "*"; j; "="; i*j
     NEXT j
     PRINT
NEXT i
END
```

Nézzük végig, hogyan működik a programunk! A 20. sorban I=1 lesz, ekkor belépünk a másik ciklusba, ahol a 30. sor szerint először J=1. A képernyőre kiíratjuk, hogy 1 × 1=1. A következő lépésben az 50 NEXT J hatására J=2 lesz, I=1 marad. 2<4, ezért a képernyőn megjelenik 1 × 2=2. A következő lépésben

J=3, de I=1 még mindig. Kiíródik 1 × 3=3. Mindez addig folytatódik, amíg kiírja az 1 × 4=4 értéket. A hasonlításkor a számítógép megállapítja, hogy a ciklust végrehajtotta, a következő utasításra lép, és a képernyőn kihagy egy üres sort. 70 NEXT I hatására megnöveli I értékét: I:=2 lesz és most ezzel hajtja végre a belső ciklust, vagyis kiírja: 2 × 1=2; 2 × 2=4; 2 × 3=6 stb. Ezek a lépések addig ismétlődnek, amíg I=4 lesz. A programunk nagyon gyorsan fut, ha I vagy J értéke nagy, ezért a program befejezésekor csak az utolsó táblázatot láthatjuk, nem tudjuk leírni. Segíthetünk ezen, ha a számokat nem egymás alá, hanem egymás után íratjuk ki. Ha a 40. sorban az utasítás végére ;-t írunk, akkor nagyon sűrűn lesznek a számok. Ha vesszőt írunk a végére, akkor nem fér ki a képernyőre az egész.

Próbáljuk lassítani programunk futását! Ezt elérhetjük, ha egy olyan ciklust írunk a belső ciklusmagba, amely nem ír ki semmit, csak végigjárja a számokat például 1-től 1000-ig, és ezzel lassul a futás, leírhatjuk táblázatunkat. Módosítva programunkat:

```
34 FOR K=1 TO 1000
36 NEXT K
```

Foglaljuk össze a FOR/NEXT ciklusképző utasításpár legfontosabb jellemzőit!

Általános alakja:

FOR ciklusváltozó=kezdőérték TO végérték STEP lépésköz
.
. A ciklusmag utasításai
.
NEXT ciklusváltozó

Jelentése:

a) A ciklusváltozó vegye fel a kezdőértéket!
b) A program fusson az azonos ciklusváltozójú NEXT utasításig!
c) Növelje a számítógép a ciklusváltozó értékét az adott lépésközzel!
d) A számítógép vizsgálja meg már a ciklus elején, hogy a ciklusváltozó a kezdő- és a végérték közé esik-e! Ha igen, akkor a *b)* eset szerint fusson tovább, ha nem, akkor a NEXT utáni utasítással folytatódjon a program futása.

Alkalmazási lehetőségek:

— STEP szó kiírása csak akkor szükséges, ha a lépésköz nem 1.
— A lépésköz negatív értéket is felvehet.

Példák

1. Az előző részben a *p*! kiszámítására írt programunkat futtassuk FOR/NEXT segítségével!

   1. *A C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "KEREK EGY SZAMOT, KISZAMITOM A FAKTORIALISAT"; P
20 S=1 : REM A SZORZAT KEZDETI ERTEKE
30 FOR N=1 TO P
40 S=S*N
50 NEXT N
60 PRINT P; "FAKTORIALIS ERTEKE:"; S
```

   2. *QBASIC-beli megfelelője:*

```
CLS
INPUT "KEREK EGY SZAMOT"; P
S=1
FOR N=1 TO P
     S=S*N
NEXT N
PRINT P; "FAKTORIALIS ERTEKE:"; S
```

2. A SZERETLEK szó kiíratására is szervezzünk ciklust a ciklusutasítás segítségével!

   1. *A C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "HANYSZOR AKARJA KIIRNI A SZOT?"; N
20 S=1
30 FOR K=1 TO N
40 S=K
50 PRINT "SZERETLEK"
60 NEXT K L : END
70 PRINT "GYULOLLEK"
```

   2. *QBASIC-beli megfelelője:*

```
CLS
INPUT "HANYSZOR AKARJA KIIRNI A SZOT?"; N
LET S=1
FOR K=1 TO N
     LET S=K
     PRINT "SZERETLEK"
NEXT K L : END
PRINT "GYULOLLEK"
```

3. Készítsünk táblázatot, melyben kiíratjuk a természetes számokat, azok négyzetét és azok reciprokát!

1. *A C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "MEDDIG SZAMOLJAK EL?"; K
20 PRINT "A POZITIV SZAM", "NEGYZET", "RECIPROK"
30 FOR N=1 TO K
40 PRINT N, N*N, 1/N
50 NEXT N
60 PRINT "KIIRTAM A KERT ADATOKAT"
```

2. *A QBASIC-ben ez a következő programmal is megvalósítható:*

```
CLS
INPUT "MEDDIG SZAMOLJAK?"; k
PRINT "SZAM", "NEGYZETE", "RECIPROKA"
LET n=1
DO
        PRINT n, n*n, 1/n
        LET n=n+1
LOOP UNTIL n=k+1
END
```

4. A ciklusutasítás segítségével írassuk ki az első tíz páros számot!
   a) egymás alá;
   b) szorosan egymás mellé;
   c) 16 karakterenként:

```
– 10 FOR I=2 TO 20 STEP 2
  20 PRINT I
  30 NEXT I
– Csak a 20. sor változik 20 PRINT I;
– 20 PRINT I,
```

Feladatok és kérdések

1. Írjunk programot, amely összeadja a páros számokat 1-től N-ig!
2. Írassuk ki az $x \mapsto x+8$ függvény értékeit, ahol x 1 és k közötti egészértékeket vehet fel!
3. Írjunk programot tetszőleges számú szám átlagának meghatározására!
4. Milyen fogalmakkal ismerkedtünk meg a ciklusszervezés közben?
5. Mit jelentenek a következő elnevezések?
   a) ciklus;
   b) ciklusmag;
   c) ciklusváltozó;
   d) kezdőérték;

e) végérték;
f) lépésköz.

6. Milyen alakú a ciklusszervező utasítás, és hogyan működik?
7. Milyen lépésközöket alkalmaztunk?
8. Mire kell ügyelni a kezdő- és végérték megadásánál?
9. Milyen utasítások futása függ logikai feltételek kiértékelésétől?
10. Miben különbözik a program futása, ha a feltételeket a ciklusmag előtt, illetve után értékeljük?
11. Mikor kerülünk végtelen ciklusba?

11. Programozott leállás

A program megállításának egyik lehetősége

Állítsuk meg a két szám összehasonlítására írt programunkat! Ha az 50. sorba beírjuk a GOTO 10 utasítást, akkor megint egy végtelen ciklushoz jutunk. Beszélgessen velünk a program, kérdezze meg, hogy szeretnénk-e tovább folytatni a futtatást! Mindezt megtehetjük, mert szöveges változók között is értelmeztük a relációkat.

A C-64-en futtathatjuk a következő programrészletet:

```
50 PRINT "FOLYTATNI AKARJA?"
60 INPUT "IRJA BE, HOGY IGEN VAGY NEM"; A$
70 IF A$="IGEN" THEN GOTO 10
80 IF A$="NEM" THEN GOTO 110
110 PRINT "BEFEJEZTEM A HASONLITAST"
```

Elértük, hogy programunk megáll, hiszen a 80 utasításról a 110 utasításra ugorva a futás befejeződik. Alkalmazzuk ezt a megoldást az előző végtelen ciklusok esetén is!

Összehasonlító programunk QBASIC-beli megfelelője lehet:

```
ELOL: INPUT "KEREM A KET SZAMOT!"; A, B
    IF A=B THEN PRINT "EGYENLOEK" : GOTO KESZ
    IF A<B THEN PRINT A; "KISEBB" ; B ELSE PRINT A; "NA-
    GYOBB";B
KESZ: PRINT "FOLYTATJA?"
    INPUT A$
    IF A$="IGEN" THEN GOTO ELOL ELSE PRINT "BEFEJEZTEM A
    HASONLITAST"
```

QBASIC-ben ciklust is szervezhetünk a „beszélgető" program megvalósítására:

```
DO
   INPUT "KEREM A SZAMOKAT"; A, B
   IF a=b THEN PRINT "a=b" : GOTO KESZ
   IF a<b THEN PRINT "a<b" ELSE PRINT "a>b"
KESZ:PRINT "AKARJA FOLYTATNI?"
   PRINT "IRJA BE; HOGY IGEN VAGY NEM!"
   INPUT A$
LOOP UNTIL A$="NEM"
PRINT "VEGEZTEM"
```

Példa: Írjunk olyan programot, amely kiírja a számok abszolútértékét!

1. *A C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "KEREK EGY SZAMOT"; A
20 IF A<0 THEN A=-A
30 PRINT "A SZAM ABSZOLUT ERTEKE:"; A
```

Ha azt is ki akarjuk íratni, hogy melyik számról van szó, akkor ezt az új értékadás előtt kell megtennünk. Például: 15 PRINT A; ahol a pontosvessző biztosítja, hogy a második kiíratás A után folytatódik. Kapcsoljunk programunkhoz kérdést feltevő részt, amely kérésünkre megállítja a programot!

2. *QBASIC-ben a következő lehet:*

```
DO
   INPUT "KEREK EGY SZAMOT"; A
   IF A<0 THEN A=-A
   PRINT "A SZAM ABSZOLUT ERTEKE:"; A
   PRINT "FOLYTATNI AKARJA? IGEN VAGY NEM?"
   INPUT A$
LOOP UNTIL A$="NEM"
PRINT "BEFEJEZTEM"
```

12. Programrészek ismétlése

A program megállításának másik lehetősége

Egy kijelölt programrész ismétlését számláló segítségével is megvalósíthatjuk, ha tudjuk, hogy a feladatot hányszor szeretnénk ismételten megoldani. Ilyenkor a programban számoljuk, hogy hányszor végeztük el a feladatot, és ezt az értéket hasonlítjuk össze azzal a számmal, amely az általunk tervezett végrehajtások számát jelzi. Előző példánk felhasználásával írassuk ki 8 szám abszolútértékét!

1. *Egészítsük ki programunkat a következő sorokkal C-64-en:*

```
6 N=8 : REM A KIIRATANDO SZAMOK SZAMA
8 S=1 : REM A SZAMLALO KEZDOERTEKE
40 S=S+1 : REM A SZAMLALO EGYET TOVABBLEP
50 IF S<=N THEN GOTO 10
60 PRINT "A KEPERNYORE IRTAM 8 SZAM ABSZOLUTERTEKET"
```

Általánosabbá tehetjük programunkat a 6 INPUT N utasítással. Alkalmazzuk ezt a megoldást is eddigi programjaink megállítására!

2. *QBASIC-ben futtatható programunk:*

```
INPUT "HANY SZAM ABSZOLUTERTEKET SZAMOLJAM?"; n
LET s=1
DO WHILE s<=n
     INPUT "KEREK EGY SZAMOT!"; a
     IF a<0 THEN a=-a
     PRINT "A SZAM ABSZOLUTERTEKE:"; a
     LET s=s+1
LOOP
PRINT "KESZ"
```

Példák

1. *Írassuk ki a SZERETLEK szót n-szer!*

1. *C-64-en futtatható programunk lehet:*

```
2 PRINT "SHIFT+CLR/HOME"
4 INPUT "HANYSZOR AKARJA KIIRNI A SZOT?; N
5 IF N<=0 THEN GOTO 40
6 S=1
10 PRINT "SZERETLEK"
20 S=S+1
30 IF S<=N THEN GOTO 10
35 END
40 PRINT "GYULOLLEK"
```

2. *QBASIC-ben futtatható programunk:*

```
CLS
LET S=1
INPUT "HANYSZOR IRJAM KI A SZOT?"; N
IF n>0 THEN
DO WHILE S<=n
     PRINT "SZERETLEK"
     LET S=S+1
LOOP
ELSE PRINT "GYULOLLEK"
END IF
```

2. Módosítsuk a páros számokat kiíratő programunkat! A felhasználó döntse el, hogy hányat akar kiíratni!

1. *C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 REM PAROS SZAMOK
12 INPUT "HANY SZAMOT IRJAK KI?"; N
14 IF N<=0 THEN GOTO 48
16 S=1 : REM A SZAMLALO BEALLITASA
20 I=2 : REM AZ ELSO PAROS SZAM
30 PRINT I
40 I=I+2
42 S=S+1 : REM A SZAMLALO EGYET UGRIK
44 IF S≤N+1 THEN 30
46 END
48 PRINT "UJ ADATOT KEREK"
50 GOTO 30
```

2. *QBASIC-ben egy másik lehetséges megoldás:*

```
        CLS
        LET s=1
        LET i=2
ism:    INPUT "HANY SZAMOT IRJAK KI?", n
        IF n<=0 THEN GOTO uj
        DO WHILE s<=n
             PRINT i
             LET i=i+2
             LET s=s+1
        LOOP
        END
uj:     PRINT "UJ ADATOT KEREK": GOTO ism
```

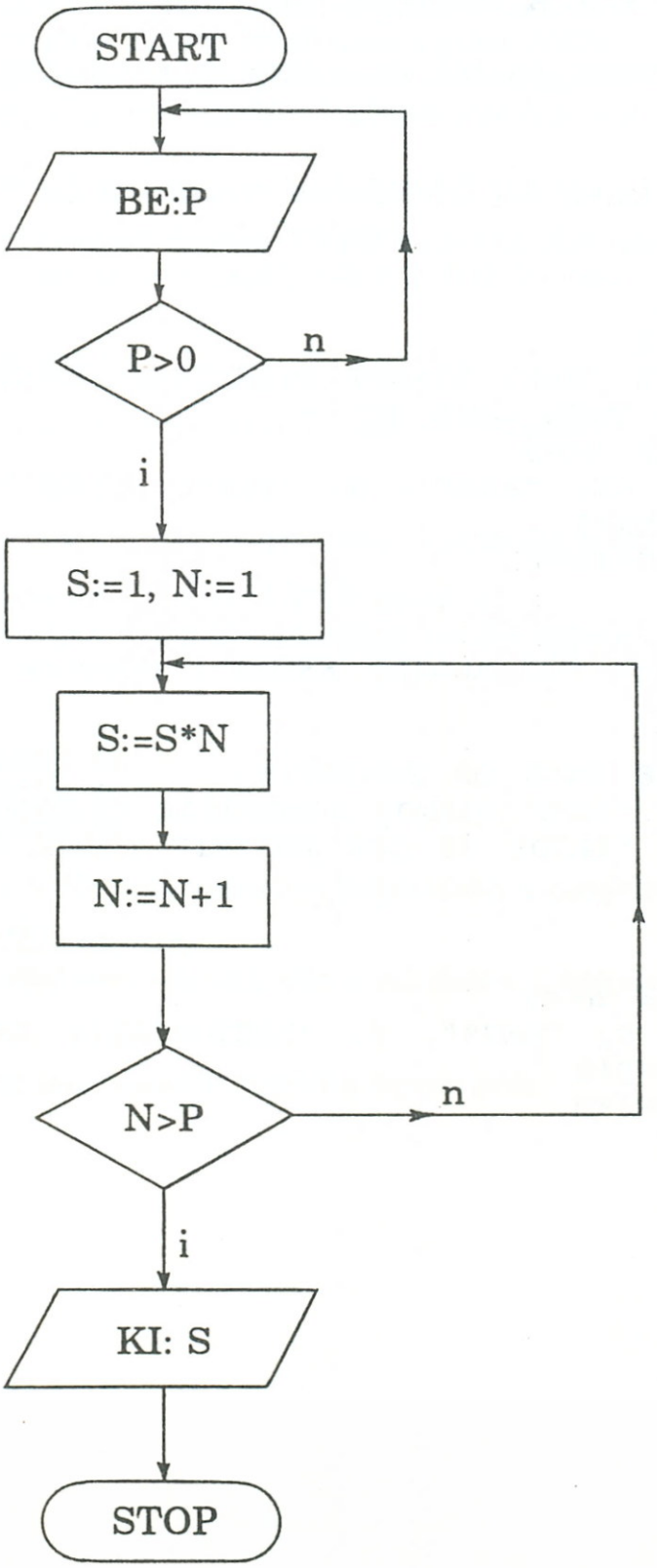
A címkék elhelyezése nagy figyelmet kíván. Valósítsuk meg az adatbevitel ellenőrzését is ciklusutasítással!

```
DO WHILE n<=0
     INPUT "HANY SZAMOT IRJAK KI?", n
LOOP
LET i=2
LET s=1
DO WHILE s<=n
     PRINT i
     LET i=i+2
     LET s=s+1
LOOP
PRINT "KESZEN VAGYOK"
```

Tovább bővíthetjük programunk lehetőségeit, ha hozzáírjuk a kérdező ciklusunkat, hogy akarja-e folytatni.

Előző feladatunkat tovább módosíthatjuk, ha kiírjuk, hogy a páros számok jelennek meg a képernyőn, és ha letöröljük a képernyőről az előzetes kiírásokat.

3. Most szorozzuk össze az első *p* darab természetes számot, amit röviden a *p*! (*p* faktoriális) jellel szoktunk jelölni.
P!= 1*2*3*4*...*P
A 18. ábra blokkdiagramja segíti munkánkat.



18. ábra

1. *C-64-en programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "HANY SZAMOT SZOROZZAK OSSZE?"; P
15 IF P<=0 THEN GOTO 10
20 S=1 : REM A SZORZAT KEZDETI ERTEKE
30 N=1 : REM A SZAMLALO KEZDOERTEKE
40 S=S*N : REM A SZORZAT UJ ERTEKE
50 N=N+1
60 IF N>P THEN PRINT P; "FAKTORIALIS ERTEKE:"; S: END
70 GOTO 40
```

2. *QBASIC-ben ismét két lehetőséget mutatunk be.*

1.
```
CLS
LET N=1
LET SZ=1
5: INPUT "HANY SZAMOT SZOROZZAK OSSZE?"; P
IF P<=0 THEN GOTO UJ
DO WHILE N<=P
  PRINT SZ; "LESZ"; N; "FAKTORIALIS ERTEKE"
  LET N=N+1
  LET SZ=SZ*N
LOOP
END
UJ: PRINT "UJ ADATOT KEREK": GOTO 5
```

2.
```
CLS
DO WHILE P<=0 OR P<>INT(P)
  PRINT "HANY SZAMOT SZOROZZAK OSSZE?"
  INPUT "IRJON BE EGY POZITIV EGESZ SZAMOT!"; P
LOOP
LET N=1
LET S=1
DO WHILE N<=P
  PRINT S; "LESZ"; N; "FAKTORIALIS ERTEKE"
  LET N=N+1
  LET S=S*N
LOOP
```

13. Logikai műveletek

Olvasmány

Gyakran előfordul, hogy a program folytatása egyszerre több feltétel teljesülésétől is függ. Ezekben az esetekben úgynevezett logikai műveleti jeleket használunk. Ezek lehetnek: AND azaz logikai ÉS, OR azaz logikai VAGY, NOT azaz logikai NEM.

A következő példákban konstansok értékének összehasonlításával mutatjuk be a kulcsszavak jelentését, hogy világossá tegyük alkalmazásuk hatását. Valódi programokban ilyen ritkán fordul elő.

Alkalmazási lehetőségeik:

a) *IF egyik feltétel, AND másik feltétel, THEN utasítás. Azt jelenti, hogy ha az egyik feltétel is és a másik feltétel is teljesül, akkor a THEN utáni utasítás hajtódik végre. Például:*

IF 5<8 AND 9>6 THEN 72

Mindkét feltétel teljesül, a program a 72. soron folytatódik.

IF 5>8 AND 9>6 THEN 72

Egyik feltétel hamis, ezért a következő soron folytatódik a program futása.

b) *IF egyik feltétel, OR másik feltétel, THEN utasítás.*
Azt jelenti, ha valamelyik (legalább az egyik) feltétel teljesül, akkor a THEN utáni utasítást kell végrehajtani.
Például:

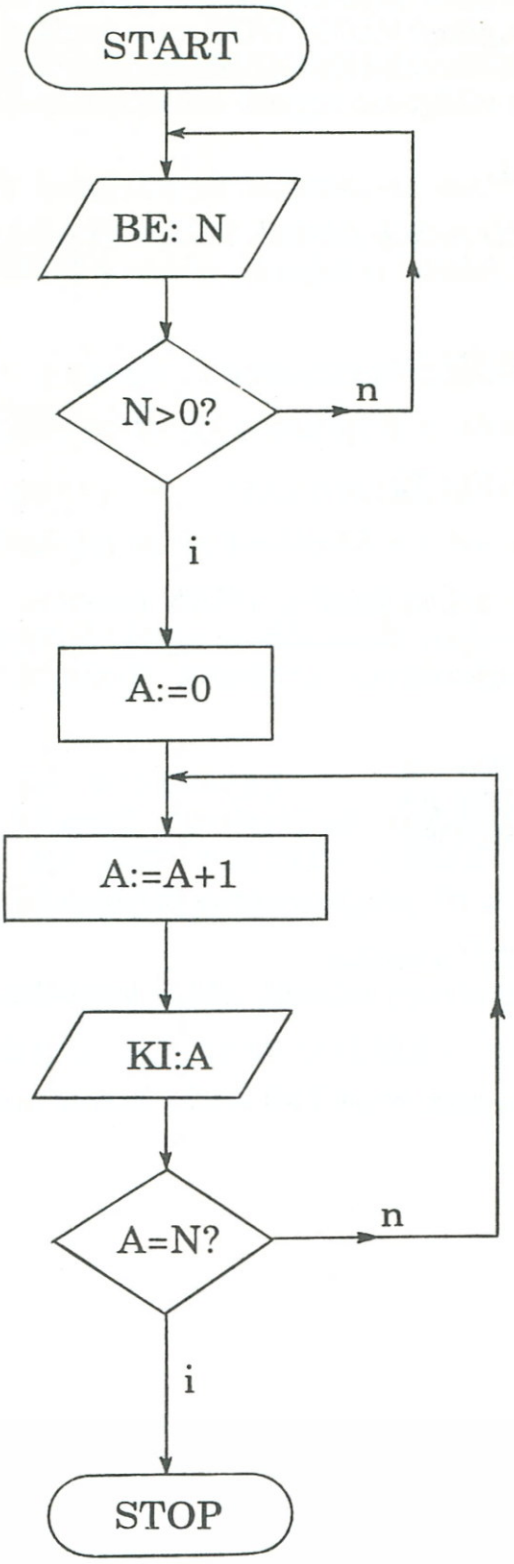
IF 5<8 OR 6<14 THEN 86
IF 5>8 OR 6<14 THEN 86
IF 5<8 OR 6>14 THEN 86

Mindhárom esetben a 86. programsoron folytatódik a program futása.

c) *IF NOT feltétel, THEN utasítás.*
Azt jelenti, ha a feltétel nem teljesül, akkor hatására a kifejezés logikai értéke ellenkezőjére változik.
Például:
NOT 5>9 *értéke igaz lesz, mert* 5>9 *értéke hamis volt.*

Példák

1. Írjunk programot, amely a képernyőre kiíratja a pozitív egész számokat egy előre megadott (N) számig!
Folyamatábra



19. ábra

1. *C-64 esetén a program lehet:*

```
10 PRINT "POZITIV EGESZ SZAMOK KIIRATASA"
20 PRINT "MEDDIG IRJAM KI AZ EGESZ SZAMOKAT?"
30 INPUT N
40 IF N<=0 THEN 20
50 A=0
60 A=A+1
70 PRINT A
80 IF A<=N THEN 60
90 PRINT "BEFEJEZTEM A SZAMOK KIIRASAT"
100 END
```

2. *Egy lehetséges megoldás QBASIC-ben:*

```
CLS
DO WHILE N<=0 OR N<>INT(N)
   PRINT "MEDDIG IRJAM KI A SZAMOKAT?"
   INPUT "IRJON BE EGY POZITIV EGESZ SZAMOT!"; N
LOOP
LET A=0
DO
   LET A=A+1
   PRINT A
LOOP UNTIL A=N
PRINT "KIIRTAM A POZITIV EGESZEKET"; N; "-IG"
```

2. Programunk futása során ugyanazt az eredményt különböző utak bejárásával is megkaphatjuk. Álljon előttünk erre egy példa! Készítsünk programot, amely két számot növekvő sorrendben ír ki!

1. eset

```
10 INPUT "KEREK KET SZAMOT!"; A, B
20 IF A>B THEN C=A : A=B : B=C
30 PRINT "A SZAMOK NOVEKVO SORRENDJE:"; A; B
```

2. eset

```
10 INPUT "KEREK KET SZAMOT!"; A, B
20 IF A<B THEN GOTO 50
30 C=A : A=B
40 B=C
50 PRINT "A SORREND:"; A, B
```

3. eset

Most a feladatot C változó bevezetése nélkül, három kiíratő utasítással oldjuk meg!

Ekkor a program:

```
5 PRINT "SHIFT+CLR/HOME"
10 INPUT "KEREK KET SZAMOT:"; A, B
20 IF A>B THEN PRINT B; "KISEBB, MINT"; A : END
30 IF A=B THEN PRINT "EGYENLOEK": END
40 PRINT A; "KISEBB, MINT"; B
```

Próbáljuk ki a programokat, hogy valóban ugyanazt eredményezik-e!

Feladatok és kérdések

1. Készítsünk programokat, amelyek egy szám abszolútértékét különbözőképpen számolják ki!
2. Írjunk olyan programot, amely bármely két szám között kiíratja az egész számokat!
3. A kiíratott számokat írassuk:
   a) szorosan egymás mellé; *b)* oszloponként.
4. Írjunk programot, amely a hét napjainak sorszámát megadva kiírja a napok nevét!
5. Hogyan érhetjük el, hogy a számítógép ne a soron következő utasítást végezze el?
6. Hogyan csoportosíthatjuk a vezérlőutasításokat?
7. Milyen utasítást alkalmazhatunk:
   a) a feltétel nélküli vezérlésátadásra; *b)* a feltételes vezérlésátadásra?
8. Milyen lehetőségeink vannak a futó program megállítására?

14. Konstans értékek tárolása a programban (DATA-READ)

Az értékadásnak egy másik formája, amikor az adatokat magában a programban adjuk meg DATA utasítással. Ez az utasítás segíti a hosszú adatsorok tárolását. A program futásai során az adatokat nem kell ismételten beírni, mint INPUT utasítás esetén, ezért az adatok többször felhasználhatók. Ismételt alkalmazáskor a konstansokat a DATA utasításból állapítja meg a gép, és ezeket a READ utasítással tudjuk a változókhoz rendelni.

Az értékadást segítő utasításpár alakja a következő:

sorszám$_1$ READ változók nevei
és
sorszám$_2$ DATA adatok

Megjegyzés
Az értékadásnak ezt a formáját a LET értékadó utasítás — általában többszöri — használatával is helyettesíthetnénk, de ha munkánk során sok adattal dolgozunk, akkor áttekinthetőbb a DATA-READ alkalmazása.

A DATA utasításban szereplő adatokat és a READ utasításban lévő változókat vesszővel választjuk el egymástól. (Az utolsó beírt adat, illetve változó után természetesen nem kell vessző.) A DATA utasításban szerepelhetnek számok (előjellel vagy anélkül), szövegek, de kifejezések nem. A DATA utasítás után álló szövegeket nem kötelező minden esetben idézőjelbe tenni, kivéve, ha:

1. A szövegben szóköz van (a szóközt elhagyná).
2. A szövegben kettőspont található (új utasításnak venné a következőket).
3. A szöveg vesszőt tartalmaz (a vessző a szöveg végét jelentené).

Egy program segítségével bemutatjuk a DATA-READ utasítás működését:

```
10 REM A DATA UTASITAS ALKALMAZASA
20 DATA 5.3, -10, AKOS, "KEK EG", 15, "JOTTEM, DE", 24.7
30 READ A, B
40 READ C$, D$
50 READ E%, F$
60 PRINT "A="; A
70 PRINT "B="; B
80 PRINT "C$="; C$
90 PRINT "D$="; D$
100 PRINT "E%="; E%
110 PRINT "F$="; F$
```

Futtatva a programot, a következő értékadások eredményét látjuk:

A=5.3, B=–10, C$=AKOS, D$=KEK EG, E%=15, F$=JOTTEM, DE

Programunk QBASIC-ben változatlanul, és a sorszámok elhagyásával is ugyanígy működik.

a) A DATA-READ utasításpár hatása

— A kiírásból könnyen észrevehetjük, hogy a READ utasítás a változóihoz a DATA utasításban soron következő értéket rendeli. Ugyanez valósulna meg akkor is, ha különböző utasításssorokban lennének az adatok. Ennek igazolására módosítjuk programunkat:

```
20 DATA 5.3, -10, AKOS
25 DATA "KEK EG", 15
28 DATA "JOTTEM, DE", 24.7
```

— Az első feladatunkban több adatot adtunk meg a DATA utasításban, mint amennyit READ-del deklaráltunk. A felesleges adatot (24.7) a gép nem veszi figyelembe. Ha a program egy későbbi helyén egy újabb READ utasítással kérnénk adatot, akkor kiolvasná a soron következőt.

Például:
```
150 READ G
160 PRINT "G="; G
```
hatására a képernyőn G=24.7 jelenik meg.

— Ha tovább folytatnánk a programunkat, és valahol elhelyeznénk egy újabb READ utasítást, akkor a gép hibajelzést adna, mert nem találna több adatot a DATA utasításban.

Próbáljuk ki!
```
220 READ H
230 PRINT "H="; H
```

Elmondhatjuk tehát, hogy az első READ utasítás végrehajtása során annyi adatot olvas ki a DATA utasítás elejéről, ahány változót tartalmaz. A következő READ utasítás innen folytatja a konstansok változóhoz rendelését. A DATA-ban legalább annyi adatnak kell lenni, mint ahány változó a READ utasításokban összesen van (több adat lehet).

— A beolvasás során hibát okozhat az, ha a READ nem megfelelő típusú adatot talál a DATA sorban, például az A-nak a "KEK EG"-gel adnánk értéket. Ügyeljünk arra, hogy a DATA utasításban szereplő adatok típusának sorrendje megegyezzen a READ utasításban szereplő változók típusával, mert ellenkező esetben a számítógép hibát jelez.

Próbáljuk ki!

```
20 READ A$, B
30 DATA 1989, TANEV
40 PRINT B; A$
```

— Hibajelzést kapnánk a következő esetben is, mert a második adat műveletet tartalmaz.

```
10 READ A, B
20 DATA 3.14, 56+37
30 PRINT "A="; A
40 PRINT "B="; B
```

b) A DATA helye a programban

A DATA utasításnak nem kell megelőznie a READ utasítást, és egymás után sem kell lenniük, mert a READ utasítás megkeresi a DATA sort, akárhol helyezzük is el a programban. Ezt az teszi lehetővé, hogy a számítógép összegyűjti a programban található DATA utasításokból az állandó értékeket, és azokat elhelyezi egy előre kijelölt tárhelyre.

Például:

```
10 READ B, C
20 PRINT "B="; B, "C="; C
30 DATA 12
40 A=B+C
50 PRINT "A="; A
60 DATA 28
```

Szokás a DATA sort a program elejére vagy a végére írni az áttekinthetőség javítása érdekében.

Példa

Írjunk programot, amely adott osztályzat esetén kiírja annak nevét!

a) Megoldás:

1. *Commodore 64-en:*

```
10 REM OSZTALYZATOK
20 PRINT "HANYAST KAPTAL?"
30 INPUT N
40 IF N=0 THEN PRINT "NEM FELELTEL?" : END
50 IF N<0 THEN PRINT "IRD BE MEG EGYSZER!" : GOTO 40
60 IF N>5 THEN PRINT "EZ LEHETETLEN" : GOTO 40
70 IF N=1 THEN PRINT N; "=ELEGTELEN" : END
80 IF N=2 THEN PRINT N; "=ELEGSEGES" : END
90 IF N=3 THEN PRINT N; "=KOZEPES" : END
100 IF N=4 THEN PRINT N; "=JO" : END
110 PRINT N; "=JELES"
```

2. *Egy lehetséges megoldás QBASIC-ben:*

```
DO WHILE N<=0 OR N>5 OR N<>INT(N)
   PRINT "HANYAST KAPTAL?"
   INPUT "OSZTALYZAT:"; N
LOOP
IF N=1 THEN PRINT N; "=ELEGTELEN" : END
IF N=2 THEN PRINT N; "=ELEGSEGES" : END
IF N=3 THEN PRINT N; "=KOZEPES" : END
IF N=4 THEN PRINT N; "=JO" : END
PRINT N; "=JELES"
```

Ez kicsit hosszadalmas, főként, ha jelest kapunk, ekkor minden hasonlítást el kell végezni.

b) Megoldás:

1. *Commodore 64-en:*

```
5 PRINT "SHIFT+CLR/HOME"
10 REM OSZTALYZATOK
20 DATA ELEGTELEN, ELEGSEGES
30 DATA KOZEPES, JO, JELES
40 PRINT "HANYAST KAPTAL?"
50 INPUT N
60 IF N=0 THEN PRINT "NEM FELELTEL?" : END
70 IF N<0 THEN PRINT "IRD BE MEG EGYSZER!" : GOTO 40
80 IF N>5 THEN PRINT "EZ LEHETETLEN" : GOTO 40
90 I=1
100 READ X$
110 I=I+1
120 IF I<=N THEN 100
130 PRINT N; "="; X$
```

Számítógépünk annyiszor hajtja végre a READ X$ utasítást, amilyen értékű osztályzatot kapunk. Az első osztályzattól kezdve kiolvassa az adatokat a DATA-ból egészen addig, amíg a kérdéses osztályzat nevét ki nem olvassa, és ezt ki is írja. Programunk lehetőséget ad arra, hogy a DATA sor adatainak kicserélése esetén bármely adatsorból kikeressük az *n.* elemet. Például kiírhatjuk a hónapok nevét, ha beírjuk, hogy hányadikról van szó.

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
DATA NEM FELELTEL, ELEGTELEN, ELEGSEGES
DATA KOZEPES, JO, JELES
  INPUT "AZ OSZTALYZATOM:"; N
DO WHILE N<0 OR N>5 OR N<>INT(N)
  INPUT "KAPOTT JEGYED?", N
LOOP
FOR I=0 TO N
  READ X$
NEXT I
PRINT N; "="; X$
```

15. Az adatok ismételt felhasználása

A programozás során gyakran előfordul, hogy ugyanazokat az adatokat többször szeretnénk felhasználni egy programon belül. Ilyenkor a következő READ utasítás elé RESTORE utasítást kell írnunk: ez teszi lehetővé, hogy az adatok kiolvasása a legkisebb sorszámnál, vagyis a program elejétől folytatódjék.

```
10 DATA 3, 6, 9, 12, 15, 18, 21, 24, 27
20 READ A, B, C, D
30 PRINT A, B, C, D
40 READ E, F
50 PRINT E, F
60 RESTORE
70 READ G, H, I
80 PRINT G, H, I
```

A képernyőn a futás hatására a következő jelenik meg:

```
 3    6    9    12
 15  18
 3    6    9
```

A DATA utasítás hátránya, hogy az adatokat csak sorban tudjuk kiolvasni belőle.

Feladatok és kérdések

1. Készítsünk programot, amely kiírja osztályunk ama tanulójának a nevét, akinek a naplóbeli sorszámát beírjuk!
2. Készítsünk programot, amely kiszámolja, hogy adott munkabér esetén mennyi nyugdíjjárulékot kell fizetni! (Adó számolását is megtervezhetjük.)
3. Készítsünk programot, amely segít a pénztárosnak fizetéskor, hogy az egyes pénzcímletekből hány darabot hozzon a bankból, hogy mindenkinek pontosan tudjon fizetni, és a legnagyobb címletekben fizessen.
4. Gondoljuk végig, hogy a bemutatott programokat hogyan tudnánk megoldani a DATA és a READ utasítások nélkül!
5. Milyen lehetőségeink vannak arra, hogy az adatokat a programban tároljuk?
6. Milyen szerepe van a READ és a RESTORE utasításoknak?
7. Mikor kell a DATA-ban szereplő szövegeket idézőjelbe tenni?
8. Mi történik, ha a DATA-ban
   a) több adatot; *b)* kevesebb adatot
   adunk meg, mint a READ-ban szereplő változók száma?
9. Hol kell elhelyezni a DATA utasítást?
10. Milyen esetekben jelez hibát a számítógép?

D) Véletlenszámok előállítása

Mindnyájan szeretünk játszani a számítógéppel. A játékok során úgy tűnik, hogy a játék menete véletlenszerűen változik, és erre reagálunk az egér vagy a botkormány segítségével, illetve egy megfelelő billentyű leütésével.

Véletlen eseményeket kapunk akkor is, amikor feldobunk egy játékkockát vagy egy pénzérmét, és tippelünk, hogy melyik oldalára esik vissza.

Számolásaink során is szükségünk lehet arra, hogy a képernyőn véletlenszámok jelenjenek meg, melyek között látszólag nem érvényesül semmilyen törvényszerűség sem. *(Például: összeadást vagy kivonást szeretnénk gyakorolni a számítógép segítségével.)* Ilyen esetekben felhasználjuk a számítógépbe beépített RND függvényt, amely véletlenszámok előállítását segíti egy adott intervallumban.

Megjegyzés

A HT számítógépnél RND(X) *kétféle lehet.*
RND(0) 0 *és* 1 *közé eső számokat generál a számítógép (a szám* 0 *lehet, de* 1 *nem).*

1. Az RND függvény alkalmazása C-64-en

Az RND(0) függvény Commodore számítógép esetén felveheti a 0 és az 1 értékét is. Az RND(X) függvényt nem ismeri a Commodore számítógép, így RND(0) felhasználásával generáltathatunk 0 és N közötti egész számokat is. Ehhez szükségünk van egy újabb függvényre, az INT (INTEGER=EGÉSZ) függvényre.

Például:

a) Kockadobás esetén a függvényünk a következő:

```
20 PRINT INT(RND(0)*6)+1
```

b) Az ötös lottó nyerőszámai lehetnek:

```
50 PRINT INT(RND(0)*90)+1
```

c) A hatos lottó eredménye:

```
160 PRINT INT(RND(0))*45)+1
```

Láthatjuk, hogy az RND(0) és az INT függvények segítségével tetszőleges véletlenszámokat állíthatunk elő.

2. Az RND függvény alkalmazása PC-n

Az RND függvény QBASIC-ben egy 0 és 1 közé eső véletlenszámot generál.

A függvény formája: RND [(numerikus kifejezés)]

A függvényünket alkalmazhatjuk argumentum, vagyis numerikus kifejezés nélkül is. Ilyenkor gépünk úgy viselkedik, mintha pozitív számot alkalmaztunk volna argumentumként. A numerikus kifejezés értékével befolyásolhatjuk a kapott véletlenszámot. Ha a beírt szám:

$X<0$,	akkor az megváltoztatja a véletlenszám-generáló ciklus kezdőértékét, így új számokat állít elő, de egy konkrét X értéknél mindig ugyanazt a számot írja ki;
$X=0$,	akkor az ismételten az utolsó számot állítja elő;
$X>0$,	akkor az új számot írja ki, de X megváltoztatásával nem változnak a számok.

Példa: Próbáljuk ki lehetőségeinket a következő program segítségével!

```
PRINT: PRINT
FOR I=1 TO 10
    PRINT RND
NEXT I
```

Módosítsuk programunkat úgy, hogy az RND függvénynek argumentumot is adunk!

a) PRINT RND(7) *vagy* PRINT RND(2)

Tapasztalatunk: mindig ugyanazt a 10 különböző számot kapjuk, ugyanabban a sorrendben, mint RND esetén.

b) PRINT RND(0)

Tapasztalatunk: mind a 10 kiírt szám megegyezik egymással.

c) PRINT RND(–3), *vagy* PRINT RND(–7)

Tapasztalatunk: a kiírt 10 szám azonos, de x értékének változtatásával más és más szám ismétlődik.

A kiírt számok mindegyik esetben 0 és 1 közé estek. Szeretnénk azonban, ha egész számokat kapnánk. Itt is alkalmazhatjuk az INT függvényt.

Példák

Kockadobásnál az utasításunk alakja:

```
PRINT INT(RND*6)+1
```

Lottószámok húzásakor az utasítás alakja:

```
PRINT INT(RND*90)+1
```

1. *Írjunk programot, amely előállít öt lottószámot!*

```
FOR I=1 TO 5
   PRINT INT(RND*90)+1
NEXT I
```

2. *Írjunk programot, amely előállít egy totótippsort!*

```
FOR I=1 TO 13+1
   PRINT INT(3*RND)
NEXT I
```

Olvasmány

Az előző programjaink futtatásakor azt tapasztaltuk, hogy minden egyes újraindításkor ugyanazt az 5, *illetve* 14 *számot kapjuk. Ezzel csak egy szelvényt lenne érdemes kitölteni, mert úgy nő a nyerési esélyünk, ha minden szelvény kitöltése különböző.*

Ha véletlenszerűen új szelvényt akarunk kapni, akkor alkalmazhatjuk a RANDOMIZE *utasítást.*

Az utasítás formája: RANDOMIZE [N]

Ezt az utasítást is alkalmazhatjuk argumentummal, vagyis az N *numerikus kifejezéssel vagy anélkül.*

Segítségével megváltoztathatjuk a véletlenszám-generátor kezdőértékét (változtathatjuk a magszámát). Ha az utasítást numerikus kifejezés, azaz szám nélkül alkalmazzuk, akkor a QBASIC *megszakítja a program futását, és a következő üzenetet írja ki:*

Random-number seed (–3268 to 3268)?
(véletlenszám-mag?)

A kérdésre az adott értékhatárok közötti számmal kell válaszolnunk. Ennek hatására módosul az RND *által generált számsor, de a véletlenszerűsítés elmarad, mert a számsor attól függ, hogy milyen számot írunk be.*

Példa: Alkalmazzuk a RANDOMIZE *utasítást a totótippek előállításakor!*

```
CLS
RANDOMIZE N
FOR I=1 TO 14
    PRINT INT(RND*3)
NEXT I
```

Ha még mindig nem vagyunk elégedettek programunkkal, akkor a numerikus kifejezés helyére írassuk be gépünkkel az aktuális időt! Ezt a TIMER függvénnyel tehetjük meg. Ez az utasítás az éjféltől eltelt időt írja be, másodpercekben, kéttizedes-pontossággal a véletlenszám-generátor kezdőértékének. Most már külső beavatkozás nélkül is különböző „véletlen" számokat kaphatunk, ha alkalmazzuk a

```
RANDOMIZE TIMER
```

sort az utasítások között.

Példák

1. Állítsunk elő 5-5 darab lottószámot 4 lottó kitöltéséhez PC számítógépre!

```
CLS
RANDOMIZE TIMER
FOR I=1 TO 5
   PRINT INT(RND*90)+1
NEXT I
```

2. Az egész számok előállítására szimuláljuk számítógépünkkel a kockadobásokat!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "A DOBOKOCKAVAL A KOVETKEZO ERTEKET DOBTA:"
20 PRINT: PRINT INT(RND(0)*6)+1
30 END
```

Most csak egyetlen érték jelenik meg a képernyőn.

Megkérdezhetjük a játékost, hogy tovább szeretne-e játszani, és ettől függően folytatjuk a programot.

```
30 PRINT "AKARJA FOLYTATNI A JATEKOT?"
40 PRINT "NYOMJA MEG AZ I VAGY AZ N BILLENTYUT!"
50 INPUT A$
60 IF A$="I" GOTO 10
70 END
```

2. *QBASIC-ben egy lehetséges megoldás:*

```
CLS
DO
   RANDOMIZE TIMER
   PRINT "A DOBOKOCKAVAL DOBOTT SZAM :";
   PRINT INT(RND*6)+1 : PRINT
   PRINT "AKARJA FOLYTATNI A JATEKOT?"
   PRINT "NYOMJA MEG AZ I, VAGY AZ N GOMBOT!"
   INPUT A$
LOOP UNTIL A$="N"
PRINT "BEFEJEZTEM"
```

3. Készítsünk programot, amely segíti a szorzótábla használatának gyakorlását!

Megoldási terv:

Bontsuk fel lépésekre feladatunkat!

a) Írjon ki a számítógép a képernyőre két tetszőleges számot, melyek egyike *a*, a másika *b*!

b) Írja ki a számítógép a képernyőre, hogy a tanulónak ezeket a számokat össze kell szorozni fejben, majd beírni a végeredményt a gépbe!

c) Szorozza össze a gép is a két adott számot!

d) Hasonlítsa össze a számítógép a kapott eredményeket!

e) Értékelje a számítógép a tanuló munkáját!

1. *Egy lehetséges program a leírtak alapján C-64-re:*

```
5 PRINT "KET SZAM SZORZATANAK MEGHATAROZASA"
10 A=INT(RND(0)*10)+1
20 B=INT(RND(0)*10)+1
24 PRINT "A="; A
26 PRINT "B="; B
30 PRINT "SZOROZD OSSZE AZ ELOZO KET SZAMOT!"
40 PRINT "IRD BE AZ EREDMENYEDET!"
50 INPUT C : REM A BEIRT SZORZAT
60 D= A*B : REM A SZAMITOGEP SZAMOL
70 IF C=D THEN PRINT "JOL DOLGOZTAL. A KET SZAM SZORZATA
   VALOBAN"; C : GOTO 100
80 PRINT "SZAMOLJ UJBOL!"
90 GOTO 40
100 END
```

Ha nem akarjuk minden szorzásnál újraindítani programunkat, akkor beírhatunk egy újabb sort:

```
100 IF C=D THEN GOTO 10
```

Természetesen ezt az ugróutasítást a 70. sorba is írhattuk volna, kettősponttal elválasztva.

2. *QBASIC-beli programunkat* készítsük el úgy, hogy a felhasználó válaszától függjön, hogy folytatódik-e a program futása, vagy nem!

```
CLS
DO
  RANDOMIZE TIMER
  LET A=INT(RND*10)+1
  LET B=INT(RND*10)+1
  PRINT "A="; A
  PRINT "B="; B
    DO
      PRINT "SZOROZD OSSZE AZ ELOZO SZAMOKAT!"
      PRINT "IRD BE AZ EREDMENYEDET!"
      LET D=A*B
      INPUT C
      LOOP UNTIL C=D
  PRINT "JOL VALASZOLTAL"
  PRINT "FOLYTATNI AKAROD A SZORZAST?"
  PRINT "NYOMD LE AZ I VAGY AZ N GOMBOT!"
  INPUT A$
LOOP UNTIL A$="N"
```

E) Indexes változók

Eddigi tevékenységünk során olyan feladatokkal találkoztunk, ahol kevés adat szerepelt, és egy változóhoz egyidejűleg egyetlen érték tartozott. *Például:*

X=5, *vagy* A$= "ESZTER"

Gyakran találkozunk olyan esetekkel, amikor több, valamilyen módon rendezett, összetartozó értékkel szeretnénk műveleteket végezni. Ilyen adatsort kapunk, ha *például* Eszter ellenőrzőkönyvében a személyi adatait vagy az osztályzatait tekintjük.

Az osztálynapló vagy az ellenőrzőkönyv rubrikái eligazítanak bennünket az adatok között, rendezettséget biztosítanak számunkra, de hogyan kezeljük őket napló nélkül?

Tankönyveinkben, különböző nyilvántartásokban a táblázatos elrendezések segítenek bennünket az áttekintésben. A táblázat soraiban általában ugyanarra az egyedre vonatkozó adatok vannak, amelyek hasonló információ-

hoz juttatnak bennünket *(például Eszter osztályzatai)*. Újabb sorban újabb egyed adataival találkozunk *(például Ágnes osztályzatai)*. Ezek az adatok azonban nemcsak vízszintesen, hanem függőlegesen, oszlopban is rendezettek. *(Például tantárgyak szerint.)* Ezeket az elvárásokat a fejlécben rögzíthetjük.

Táblázataink matematikai általánosításaként speciálisan rendezett halmazt kapunk, melyet tömbnek nevezünk az informatikában. Adatfeldolgozás szempontjából nagyon hasznos az alkalmazásuk, mert ilyenkor az összetartozó értékeket egyetlen névvel azonosítjuk, de elemeit egyértelműen megkülönböztetjük egymástól indexeik segítségével. Bevezetjük tehát az indexes változó elnevezést, amely egy betűből, és az azt követő, zárójelbe tett számból, az indexből (mutatóból, jelzőből) áll. *Például* A(4).

Vigyázat! A(4) nem azonos A4-gyel, mert A4 nem indexes változó!

1. Vektorok

Ha a zárójelben egyetlen érték szerepel, akkor egydimenziós tömbről vagy vektorról beszélünk. Ilyenkor a tömb egyetlen sorból vagy egyetlen oszlopból áll. Az index a vektor elemei közötti sorrendet mutatja. *Például:* A(4) *az* A *vektor* 4. *sorszámú elemét jelöli.* (Az, hogy a vektorokat sornak vagy oszlopnak tekintjük, szubjektív: a gép nem tesz ilyen különbséget.)

Ha az indexes változónál a zárójelben számpárok szerepelnek, akkor kétdimenziós tömbről, ha számhármasok, akkor háromdimenziós tömbről beszélünk stb.

Indexként leggyakrabban egész számot (egész típusú változót) szoktunk használni, de összetett kifejezéseket is alkalmazhatunk. *Például:*

```
A(3*2); B(4+1); C(X); D(3*X).
```

Ilyenkor a számítógép először kiértékeli a zárójelben lévő aritmetikai kifejezést, mert csak ezután tudja, hogy melyik tömbelemmel kell dolgoznia.

Negatív indexet számítógépünk nem fogad el.

2. Tömbdeklaráció

A számítógép memóriájában 10-elemes tömb számára automatikusan van hely. Amennyiben számolásunkban ennél nagyobb tömböt szeretnénk használni, akkor azt előre jeleznünk kell, helyet kell foglalnunk a memóriában. (Ellenkező esetben a számítógép hibát jelez.) Ezt a helyfoglalást nevezzük a tömbök deklarálásának vagy dimenzionálásának, és a DIM A (...) utasítással valósíthatjuk meg.

DIM A (100) egy százelemű tömböt (vektort) deklarál. A tömb nagyságának jelzésére a számítógép nemcsak egy konkrét számot, hanem egy változót is elfogad.

Például: DIM A (N). *(Ezt dinamikus tömbmeghatározásnak nevezzük.)* Mivel ezt csak a BASIC bizonyos változatai fogadják el, ezért célszerű a tömbök dimenzionálását minden esetben elvégezni. (DIM a dimension – kiterjedés, méret – szó rövidítése.)

Egy utasításban több tömböt is deklarálhatunk. *Például:*

```
30 DIM A(2), B(X), C(4).
```

Példák

1. Írjunk programot, amely kiszámolja egy tanuló tanulmányi átlagát tetszőleges számú tantárgy esetén!

   Megoldás:
   A tantárgyak száma 10-nél lehet több, ezért helyet kell foglalnunk a memóriában még az első felhasználás előtt.

   1. *C-64-en futtatható programunk:*

```
10 DIM B(25)
20 PRINT "TANULMANYI ATLAG SZAMITASA"
30 PRINT "HANY TANTARGYBOL KAPTAL OSZTALYZATOT?"
40 INPUT N
50 PRINT "IRD BE AZ OSZTALYZATOKAT!"
55 REM ERDEMJEGYEK BEOLVASASA
60 FOR I=1 TO N
70 INPUT B(I)
80 NEXT I
85 REM BEOLVASOTT OSZTALYZATOK OSSZEADASA
90 S=0
100 FOR I=1 TO N
110 S=S+B(I)
120 NEXT I
130 PRINT "A TANULMANYI ATLAG:"; S/N
```

   Módosítsuk programunkat!

   a) Számoljuk ki az osztály tantárgyankénti átlagát, és ezt írja is ki a számítógép a képernyőre!

   b) Számoljuk ki az osztály átlagát, és írja ezt ki a számítógép a képernyőre!

   2. *Programunkat sorszám nélkül is ugyanígy futtathatjuk QBASIC-ben.*

2. Készítsünk programot, amely a képernyőre írja egy 21 fős osztály névsorát!

1. *C-64-en futtatható programunk:*

```
10 DIM A$(21)
20 FOR I=1 TO 21: REM TANULOK NEVENEK BEIRASA
30 PRINT "A(Z)"; I; ". TANULO NEVE"
40 PRINT
50 INPUT A$(I)
60 NEXT I
70 REM NEVSOR NYOMTATASA
75 PRINT "AZ I/C OSZTALY NEVSORA" : PRINT
80 FOR I=1 TO 21
90 PRINT
100 PRINT A$(I)
110 NEXT I
```

2. *QBASIC-ben is változatlanul futtathatjuk programunkat.*

Olvasmány

Az előző esetben a számítógépbe a kész névsort írtuk. Sokszor azonban csak a nevek adottak, és szeretnénk, ha azokat a számítógép névsorba rendezné. (Emlékezzünk az előzőekre, hogy stringváltozóknál a gép a betűk ábécében elfoglalt helyét vizsgálja. Összehasonlításkor az a betű kisebb, amelyik előbb található az ábécében.)

Erre alkalmazhatjuk a következő rendezési programot:

```
10 REM ADATOK BEVITELE
20 PRINT "HANY TANULO VAN?"
30 INPUT N
40 DIM A$(N)
50 FOR I=1 TO N
60 PRINT "A(Z)"; I; ". TANULO NEVE"
70 INPUT A$(I)
80 NEXT I
100 REM NEVSORBA RENDEZES
110 FOR J=N TO 1 STEP-1
120 FOR I=1 TO J-1
130 IF A$(I)>A$(I+1) THEN K$=A$(I+1) : A$(I+1)=A$(I) :
    A$(I)=K$
140 NEXT I
150 NEXT J
200 REM A TANULOK NEVSORANAK KIIRASA
210 PRINT "A NEVSOR"
220 FOR I=1 TO N
230 PRINT A$(I)
240 NEXT I
```

A névsorba rendezést cserével végeztük el, tehát az egyenlőtlenség teljesülése esetén a kisebb értéket K$ változóba helyezzük, majd a kisebb értéket kicseréljük a nagyobbal. Ezután az A$(I) értéke eggyel hátrébb csúszik. A K$ segédváltozó abban segít, hogy ne vesszen el a nagyobb érték.

A vektorok helyfoglalása a tárban

A számítógépben az egyszerű változókhoz hasonlóan az indexes változók is tárcellákat jelölnek ki, amit a *20. ábrán* szemléltetünk:

a) Indításkor

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

20. ábra

b) Értékadás után, ha B(0) = –13 : B(1) = 5.3
B(3) = 72 : B(5) = –2.1, akkor a tárbeli kép a *21. ábrán* látható.

B(0)	–13
B(1)	5.3
B(2)	0
B(3)	72
B(4)	0
B(5)	2.1

21. ábra

Láthatjuk, hogy ha nem adunk értéket a numerikus tárbeli változónak, akkor megmarad a kiinduló értéke, a 0 (karakterváltozó esetén az alapérték a szóköz, „üres").

Előfordulhat, hogy a memóriából szeretnénk törölni a benne található összes egyszerű és tömbváltozót, de a programot változatlanul akarjuk hagyni (a RUN parancs hatására mindez automatikusan megvalósul).

Összefoglalva: a tömbök dimenzionálásakor a következő szempontokat kell szem előtt tartanunk:

— Ugyanazt a tömböt egy programban csak egyszer szabad definiálni.

— A többszöri meghatározás elkerülése érdekében célszerű valamennyi DIM utasítást a program elejére elhelyezni. Ez azért is jó, mert a DIM utasításnak előbb kell lennie a programban, mint a tömbelemekre való hivatkozásnak.
— Ha a számítógép tárolókapacitása kicsi, akkor igyekezzünk elkerülni a dinamikus tömbmeghatározást.
— Célszerű a programban szereplő tömbök méretét, felépítését az algoritmus készítésekor megtervezni.
— Csak az azonos típusú adatcsoportokból képezhetünk tömböket a tárban.
— Túl nagy tömbök deklarálása előtt memóriahelyet kell felszabadítanunk.

Példák:

Írjunk programot, amely elkészíti egy totó tippsorait!

Definiálnunk kell egy háromelemű tömböt, amelyben elhelyezzük a szöveggel megadott tipplehetőségeket, hogy a tippsort könnyen olvasható formában kapjuk meg. Lépéseinket a *22. ábrán* látható folyamatábra szemlélteti.

1. *Az algoritmus egy lehetséges leírása C-64-en futtatható programmal:*

```
5 PRINT "SHIFT+CLR/HOME"
10 DIM T$(3)
20 T$(1)="1"
30 T$(2)="2"
40 T$(3)="X"
50 PRINT "A HETI TOTOEREDMENYEK"
60 I=1
70 X=INT(RND(0)*3)+1
80 PRINT T$(X)
90 I=I+1
100 IF I<14 THEN GOTO 70
110 PRINT "ELLENORIZZE SAJAT TIPPJEIT!"
120 PRINT "REMELEM, NYERT"
130 END
```

Természetesen más megoldást is találhatunk. Például ciklusképző utasítás segítségével programunk a 60. sortól a következő lehet:

```
60 FOR I=1 TO 14
70 X=INT(RND(0)*3)+1
80 PRINT T$(X)
90 NEXT I
```

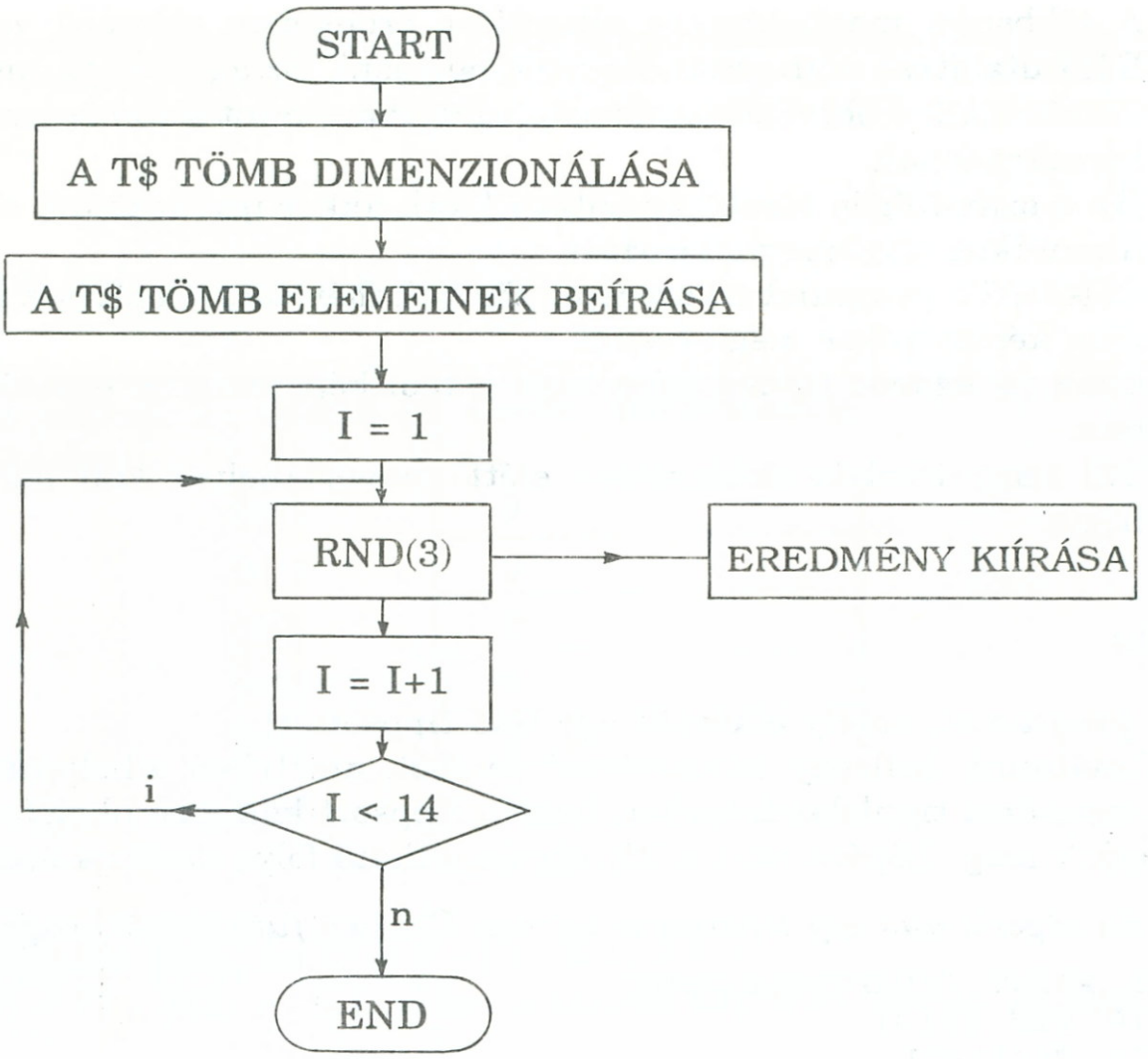
Nem kell a 100. sor, de a többi maradhat.

Változtassuk a kiíratási formákat!

a) Szorosan egymás mellé írja a számítógép a tippsort a képernyőre!

b) Helyközzel írja ki az eredményt a képernyőre a számítógép!

c) Táblázatszerűen kapjuk meg az adatokat.



22. ábra

2. *Programunk QBASIC-beli megfelelőjében alkalmazzunk ciklusutasítást, és a folytatás a felhasználótól függjön!*

```
DIM t$(3)
t$(1)="1"
t$(2)="2"
t$(3)="X"
PRINT "HETI TOTOEREDMENYEK"
DO
     FOR I = 1 TO 14
          RANDOMIZE TIMER
          X = INT(RND*3)+1
          PRINT t$(X)
     NEXT I
     PRINT "AKAR UJABB TIPPSORT?"
     PRINT "NYOMJA LE AZ I VAGY AZ N GOMBOT!"
     INPUT a$
LOOP UNTIL a$="N"
PRINT "KIIRTAM A TIPPEKET"
```

3. Többdimenziós tömbök

Az előzőekben már jeleztük, hogy indexként nemcsak egy értéket adhatunk meg, hanem többet is, vesszővel elválasztva. Ilyenkor többdimenziós tömböket kapunk.

Ha egy tömböt két számmal (aritmetikai kifejezéssel) definiálunk, akkor mátrixokról szoktunk beszélni. Ilyenkor egy sorokból, oszlopokból álló táblázatot kell elképzelnünk. Dimenzionáláskor az első szám a sorindex, a második szám az oszlopindex. (Ezt hagyományosan így szokták tekinteni, de gondolhatnánk fordítva is.)

Például: DIM B(3, 5) *egy* 3 *sorból és* 5 *oszlopból álló táblázatot állít elő.*

Könnyen megkaphatjuk egy elemét, ha azt a sorindex és az oszlopindex megadásával kijelöljük.

Például: B(3, 2) *azt jelenti, hogy kérjük a* 3. *sor* 2. *elemét.*

(Sok BASIC-változat a sorokat és az oszlopokat egyaránt 0*-tól számozza, így a fenti* DIM *utasítás valójában egy* 4*-szer* 6*-os mátrixot hoz létre.)*

Olvasmány

A mátrix helyfoglalása a tárban
Az egydimenziós tömbökhöz hasonlóan nézzük meg a mátrix tárbeli képét. Esetünkben a 23. ábrán látható rácsot vagy táblázatot kell elképzelnünk:

	0	1	2	3	4	5
0						
1						
2						
3						

23. ábra

A tanulmányi és tantárgyi átlag számolásának lépései

Készítsünk programot, amely kiszámítja *a* számú tanuló tanulmányi átlagát, ha *b* tantárgyból kapnak osztályzatot! Számítsuk ki az osztály tantárgyankénti átlagát is! Mindezt írjuk ki a képernyőre !

Lépésenként kövessük nyomon a program írását, mert később más feladatok megoldása esetén is alkalmazhatjuk mindezt kisebb módosításokkal!

Programunk megtervezése

1. *Adatok bevitele*

   A tanulók adatait helyezzük el egy kétdimenziós tömbben, ahol az első szám a tanuló sorszámát, a második szám a tantárgy sorszámát jelenti, ahogyan ezt a *24. ábra* szemlélteti.

   Például:

	1. tantárgy	2. tantárgy	3. tantárgy	4. tantárgy
1. tanuló	A(1,1)	A(1,2)	A(1,3)	A(1,4)
2. tanuló	A(2,1)	A(2,2)	A(2,3)	A(2,4)
3. tanuló	A(3,1)	A(3,2)	A(3,3)	A(3,4)

24. ábra

Az osztálynaplóból kiírt konkrét adatokkal a *25. ábra* mutatja a táblázat kitöltését.

	Matematika	Magyar	Történelem	Fizika
Kis Ági	4	3	5	3
Nagy Eszti	3	4	4	2
Tóth Ákos	5	4	3	4

25. ábra

A *24.* és *25. ábra* táblázata alapján könnyen nyomon követhetjük a számítógép működését.

Amikor az adatokat bevisszük, akkor az előbbiek szerint megadott értéket írjuk be, tehát a 2. tanuló (Nagy Eszti) a 3. tantárgyból (történelem) 4-es osztályzatot kapott. A számítógép ezt elraktározza, vagyis A(2,3) = 4.

Mivel az adatokat egy mátrix segítségével helyezzük el a tárban, ezért az A mátrixot dimenzionálnunk kell. *Például:*

DIM A(20,6), *ha maximálisan* 20 *tanulót, legfeljebb* 6 *tantárgyból osztályoznak. (Ha valaki valamely tárgyból nem kap osztályzatot, akkor* 0-*t írunk be!)*

Az adatok bevitelekor szükségünk van:
— A tanulók számára: *a.*
— Az osztályozott tantárgyak számára: *b.*
— A tanulók adataira (két egymásba ágyazott ciklus kell a beolvasáshoz).

2. *Adatok feldolgozása*
Vizsgáljuk meg, hogy milyen feladatot kell megoldani programunknak!
a) Tanulónként ki kell számítani a tanulmányi átlagot.
b) Tantárgyanként ki kell számítani az osztály átlagát.

Az összegek kiszámításakor két egymásba ágyazott ciklust használunk, hogy tanulónként összeadjuk az osztályzatokat. A kapott összegeket a D(I) vektorban tároljuk, ha tanulmányi átlagot, és C(J) vektorban, ha tantárgyankénti átlagot akarunk számítani. Ezt a két vektort is definiálni kell a program elején!

3. *Eredmények kiíratása*
Ismét gondoljuk végig, hogy mi az, amit a képernyőn látni szeretnénk.
a) Írja ki programunk a képernyőre a vizsgált tanuló sorszámát, osztályzatait, és tanulmányi átlagát!
b) Írja ki programunk a képernyőre a vizsgált tantárgy sorszámát, a belőle kapott osztályzatokat és az osztály tanulmányi átlagát!

A kiíratást az előbbiekhez hasonlóan két egymásba ágyazott ciklus segítségével oldhatjuk meg.

A leírtak alapján a következő BASIC nyelvű programot készíthetjük C-64-re:

```
5 PRINT "SHIFT+CLR/HOME"
10 REM ATLAGSZAMITAS
15 REM TOMBOK DIMENZIONALASA
20 DIM A(20,6): REM TANULOK OSZTALYZATAI
40 DIM C(6): REM OSZT. OSSZ. TANTARGYANKENT
50 DIM D(20): REM OSZTALYZATOK OSSZEGE TANULONKENT
60 REM ADATOK BEOLVASASA
70 PRINT "HANY TANULO VAN?"
80 INPUT A
90 PRINT
100 PRINT "HANY TARGYBOL VAN OSZTALYZAT?"
110 INPUT B
120 PRINT
130 FOR I=1 TO A: REM TANULOK ADATAI
140 PRINT
150 PRINT "A(Z)"; I; ".TANULO OSZTALYZATAIT IRD BE!"
160 PRINT
170 FOR J=1 TO B
```

```
180 PRINT "A(Z)"; J; ".TANTARGYBOL KAPOTT OSZTALYZAT"
190 INPUT A(I,J)
200 PRINT
210 NEXT J
220 NEXT I
230 REM AZ ADATOK FELDOLGOZASA
240 REM TANULONKENT AZ OSZTALYZATOK OSSZEGE
250 FOR I=1 TO A
260 D(I)=0
270 FOR J=1 TO B
280 D(I)=D(I)+A(I,J)
290 NEXT J
300 NEXT I
310 REM TANTARGYANKENT AZ OSZTALYZATOK OSSZEGE
320 FOR J=1 TO B
330 C(J)=0
340 FOR I=1 TO A
350 C(J)=C(J)+A(I,J)
360 NEXT I
370 NEXT J
380 REM AZ ATLAGOK KIIRATASA
390 PRINT "AZ I/C OSZTALY TANULOINAK ATLAGA"
400 FOR I=1 TO A
410 PRINT
420 PRINT "A(Z)"; I; ".TANULO OSZTALYZATAI"
430 PRINT
440 FOR J=1 TO B
450 PRINT A(I,J)
460 NEXT J
470 PRINT
480 PRINT "A(Z)"; I; ".TANULO TANULMANYI ATLAGA";
490 PRINT D(I)/B
500 PRINT
510 NEXT I
520 PRINT "AZ I/C OSZTALY TANTARGYI ATLAGAI"
530 FOR J=1 TO B
540 PRINT
550 PRINT "A(Z)"; J; ".TANTARGY OSZTALYZATAI"
560 PRINT
570 FOR I=1 TO A
580 PRINT A(I,J)
590 NEXT I
600 PRINT
610 PRINT "A(Z)"; J; ".TANTARGYI ATLAG:"; C(J)/A
620 PRINT
630 NEXT J
```

Programunk futtatásakor sajnos a képernyőn keveset látunk, mert nem fér rá a kiírás.

Módosítási lehetőségek

a) Az első kiíratás után megállítjuk a programot, majd a CONT parancs beírásával folytatjuk a futást, ha az eredményeket rögzítettük.

```
512 PRINT "A CONT PARANCS HATASARA FOLYTATOM"
514 STOP
516 PRINT "SHIFT+CLR/HOME"
```

b) Másik lehetőség, ha egy sorba íratjuk ki a részeredményeket. Próbáljuk ki!

c) Egy újabb esetben táblázatos kiírást választunk.
Töröljük a 430, 480, 490, 500, 560, 610, 620 *sorokat!*
Módosítsuk a 450 PRINT A(I,J); *és az* 580 PRINT A(I,J); *sorokat!*
Utolsóként ; *jelet írunk.*
Új sorokat írunk:

```
395 PRINT "SORSZAM", "ATLAG", "OSZTALYZATOK"
420 PRINT I, D(I)/B,
525 PRINT "SORSZAM", "ATLAG", "OSZTALYZATOK"
550 PRINT J, C(J)/A,
```

Táblázatunk még mindig nem tetszik, mert az átlagok különböző hosszúságú értékek, ezért az osztályzatok kiírása nem ugyanott kezdődik. Egyezzünk meg abban, hogy a végeredményt 2 tizedesjegy-pontossággal íratjuk ki. Erre felhasználjuk az egészrészfüggvényt, amelyet már korábban is alkalmaztunk.
A módosított sorok:

```
420 PRINT I, INT (D(I)/B*100+0.5)/100,
550 PRINT J, INT (C(J)/A*100+0.5)/100,
```

(A + 0.5 a kerekítés miatt kell.)

d) A kiírást végezzük el a TAB függvény segítségével is!

Programunkat tovább szépíthetjük, ha definiálunk egy szöveges változókat tartalmazó tömböt, amely tartalmazza a tanulók nevét:
DIM A$(20), és egy másikat, amely a tantárgyakat foglalja magában:
DIM B$(6), és sorszám helyett ezeket a szavakat íratjuk ki. Természetesen módosul az adatok beolvasása is, mivel a tanulók nevét és a tantárgyakat is be kell olvasni.

Módosítsuk programunkat a szöveges változókat tartalmazó tömbök segítségével!

Futtatás közben előfordulhatott, hogy véletlenül nyilvánvalóan hibás értéket ütöttünk be, például negatív osztályzatot vagy tanulószámot, esetleg 0-t vagy többet, mint amit a tömb definiálásakor előre terveztünk (alkalmazhatunk dinamikus deklarálást). Ilyenkor a gépünk általában nem jelzett hibát. Ezeket az eseteket is kizárhatjuk egy ciklusutasítás segítségével.

Szépítsük tovább programunkat!

Ez a program is változatlanul futtatható QBASIC-et használó számítógépen is.

Példa:

Írjunk olyan programot, amely a hét napjainak sorszáma alapján megadja, hogy melyik napról van szó!

Az adatokat, a hét napjait helyezzük el egy adatsorba a DATA utasítás segítségével!

Definiálnunk kell egy 7 elemű tömböt, amibe bekerülnek a hét napjai. Megoldásunkkal zárjuk ki a téves adatok beírásának lehetőségét is!

1. *Egy C-64-en futtatható program:*

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "NAPOK NEVENEK MEGHATAROZASA SORSZAMOK ALAPJAN"
20 PRINT
30 DATA HETFO, KEDD, SZERDA, CSUTORTOK
40 DATA PENTEK, SZOMBAT, VASARNAP
50 DIM N$(7)
60 FOR I=1 TO 7 : REM NAPOK NEVENEK BEOLVASASA A TOMBBE
70 READ N$(I)
80 NEXT I
90 PRINT
100 REM KONKRET SORSZAMU NAP NEVENEK MEGHATAROZASA
110 PRINT "KEREM A NAP SORSZAMAT!"
120 INPUT N
130 IF N<1 THEN GOTO 110
140 IF N>7 THEN GOTO 110
150 PRINT "A(Z)"; N;".NAP NEVE:";N$(N)
160 PRINT
170 PRINT "AKARJA FOLYTATNI?"
180 PRINT "NYOMJA MEG AZ I VAGY AZ N BILLENTYUT!"
190 INPUT V$
200 IF V$="I"  THEN 110
210 IF V$="N" THEN PRINT "BEFEJEZTUK A MUNKAT"
220 END
```

Ezt a programrészletet felhasználjuk olyan esetekben, amikor sorszám alapján tudjuk megkülönböztetni az összetartozó adatokat, és ezeket nevesíteni akarjuk.

Például: Az osztálynaplóban a 12. *tanuló* LUKÁCSI ESZTER.

2. *Programunk QBASIC-beli megfelelője a következő lehet:*

```
CLS
DATA Hetfo, Kedd, Szerda, Csutortok
DATA Pentek, Szombat, Vasarnap
DIM N$(7)
FOR I=1 TO 7
  READ N$(I)
NEXT I
DO
  PRINT
```

```
    DO
      PRINT "KEREM A NAP SORSZAMAT!"
      INPUT N
    LOOP WHILE N<1 OR N>7
    PRINT "A(Z)"; N; ".NAP NEVE:"; N$(N)
    PRINT
PRINT "HA FOLYTATNI AKARJA, NYOMJA LE AZ I VAGY AZ N GOMBOT!"
INPUT V$
LOOP UNTIL V$="N"
PRINT "VEGEZTEM A KIIRASSAL"
```

4. Az indexes változók alkalmazásának előnyei

1. Indexes változók segítségével ki tudjuk fejezni az adatok összetartozását.
2 Nem kell minden változóhoz új nevet kitalálnunk.
3. Ha tömbelemek mindegyikével ugyanazt a műveletet szeretnénk elvégezni, akkor elegendő egy ciklust alkalmazni.

   Például kiíratásnál:

```
10 FOR I=1 TO 100
20 PRINT A(I,25)
30 NEXT I
```

4. Sok adat feldolgozását segíti, mert az összetartozó adatokra egyetlen névvel hivatkozhatunk. A feldolgozás során csak az index változik.
5. Az összetartozó adatok egymás után, rendszerezetten találhatók a számítógép memóriájában. Így a tömbelemeket a gép gyorsabban érheti el.
6. Csökkenti a programlista méretét, és ezáltal gyorsítja a program végrehajtását is.

Olvasmány

A BASIC nyelv bizonyos változataiban csak egy- és kétindexű tömböket használhatunk. Nem ritka azonban ezek általánosításaként többindexű tömbök megvalósítása sem.

Összefoglalás

Az indexes változók bemutatásával befejeztük az adatok bevitelét segítő módszerek megismerését, melyek a következők voltak:

a) LET
b) INPUT
c) DATA-READ

Mindegyik állhat magában vagy ciklusban, attól függően, hogy egyszerű vagy tömbváltozó értékét adjuk-e meg.

Feladatok és kérdések

1. Írjunk programot, amely néhány pénzfeldobást helyettesít, és összeszámolja, hány fej és hány írás volt a feldobások után!
2. Készítsünk programot, amely a raktárban lévő áruknak kiszámítja az értékét az ár és a mennyiség ismeretében. Számolja ki a program az összes mennyiséget és az összes értéket is! *Segítségül egy táblázat:*

	Ára Ft/kg	Mennyisége(kg)	Értéke
Kávé			
Tea			
Cukor			
Liszt			
Só			

3. Készítsünk programot, amely egy termék kódja alapján kiírja, hogy milyen áruról van szó, és megjeleníti jellemzőit is *(például: ár, mennyiség).*
4. Hogyan állíthatunk elő 0 és 1 közé eső véletlenszámokat?
5. Hogyan állíthatunk elő 1 és x közé eső véletlen egész számokat?
6. Mit nevezünk tömbnek az informatikában?
7. Mit jelent az indexes változó elnevezés?
8. Mikor beszélünk vektorról, illetve mátrixról?
9. Milyen alakúak lehetnek az indexek?
10. Mit jelent a tömbök dimenzionálása, és hogyan valósíthatjuk ezt meg?
11. Milyen szempontokat kell figyelembe venni a tömbök dimenzionálásakor?
12. Az indexes változók alkalmazásának milyen előnyeiről beszéltünk?
13. Milyen módszerekkel tudunk adatokat bejuttatni a számítógépbe?

F) Önálló programozási egységek, szubrutinok

1. Alprogramok alkalmazása

Eddigi feladataink során találkoztunk olyan esetekkel, amikor a program több helyén is ugyanazokat az utasításokat kellett végrehajtani. *(Például az osztály tanulóinak tanulmányi, illetve az osztály tantárgyi átlagának kiírásakor alkalmazott táblázatok elkészítése.)*

Ilyenkor mindegyik helyre leírtuk a szükséges programrészletet. Rövid részek esetén ez csak néhány sor, de összetett feladatok esetén ez már nagyon lassú és áttekinthetetlen lehet.

Az ismétlésnek egy formáját, amikor a műveletsort közvetlenül egymás után kell többször végrehajtani, már megismertük a ciklusok szervezésekor. Előfordul azonban, hogy ugyanazt a részprogramot az algoritmus különböző helyein kell ismét végrehajtanunk.

1. A BASIC nyelv biztosítja számunkra, hogy a többször használt programrészeket elegendő a programban egyszer szerepeltetni, és ha a program futása közben szükségünk van erre a részre, akkor csak hivatkoznunk kell rá. Ezt a többször felhasznált programrészletet nevezzük eljárásnak vagy szubrutinnak. Ezeket — minden utasításukat beleértve — több alkalommal is végrehajtathatjuk. Az eljárások azonban akkor is jó szolgálatot tehetnek, ha csak egyszer hívjuk őket: használatukkal programunk szerkezete világosabb, áttekinthetőbb lehet.

   Minden programban van egy kitüntetett eljárás, a vezérlő eljárás, amely a részeljárások végrehajtását irányítja: ezt főprogramnak hívjuk. Ehhez kapcsolódhatnak az önálló programrészek, a szubrutinok, amelyeket a főprogram egységként tud meghívni a program végrehajtása során.

   A szubrutin hívását a főprogramban elhelyezett GOSUB utasítás segítségével oldhatjuk meg. Ezzel a vezérlésátadással jelezzük, hogy az alprogramot szeretnénk végrehajtani. (GOSUB a GO (menj) és a SUBROUTINE (alprogram) szavak összevonásából jött létre.)

   Az utasítás alakja:

   sorszám 1 GOSUB sorszám 2

   sorszám 1 — a vezérlésátadó utasítás helye a főprogramban (a GOSUB utasítást tartalmazó programsor sorszáma);
   sorszám 2 — a szubrutin kezdősorának sorszáma.

Az utasítás hatására a program futása a sorszám 2. soron folytatódik, majd annak befejezése után, melyet a RETURN utasítás zár le, visszatérünk a hívás utáni első utasításra a főprogramba, vagyis a program végrehajtása a GOSUB utasítás utáni utasításon folytatódik.

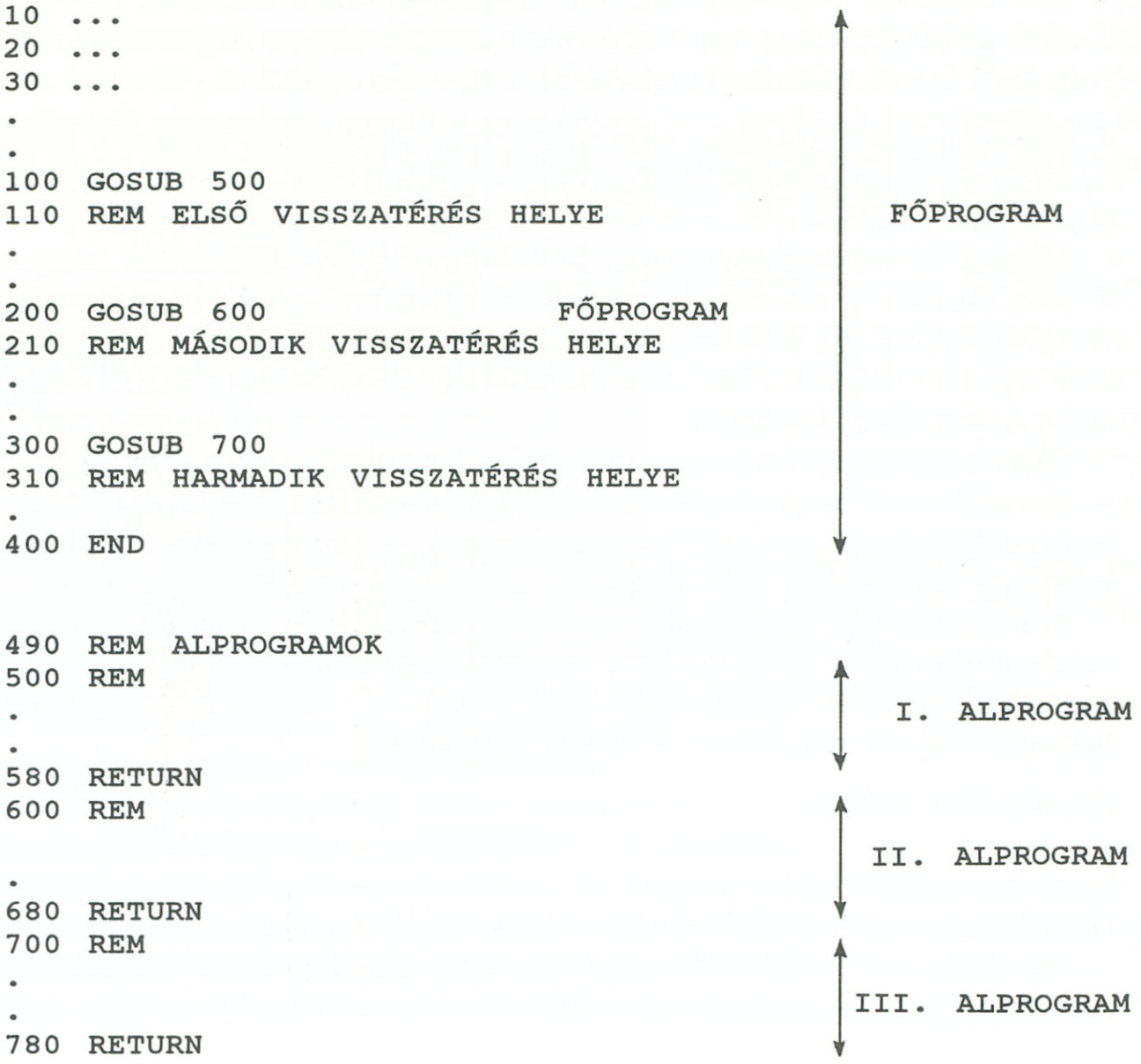
2. A QBASIC nyelv is ismeri ezt az utasítást.

   Az utasítás formája: GOSUB név

Ez a név megegyezik annak a szubrutinnak a nevével, melyet szeretnénk végrehajtani. A szubrutin neve lehet sorszám vagy betűkből álló név is. (A QBASIC-ben lehetőségünk van a főprogramtól független szubrutinkészítésre is, de kezdő programozóknak gondot jelenthet az összes változó megtalálása, melyet a főprogramban meg kell adni előre, ezért ezt az utasítást nem tárgyaljuk.)

A megbeszélteket a *26. ábra* szemlélteti.

Mondjuk el részletesen az utasítássorok jelentését az ábra alapján!



26. ábra

Amikor a program a 100-as sorszámú utasításhoz ér, akkor a vezérlés átadódik az 500-as sorszámú utasításra, és mindaddig ott folytatódik, amíg az 580-as sorszámú RETURN utasításhoz nem ér, melynek hatására a vezérlés visszaadódik a 110-es sorra. A 200-as sorszámú utasítás hatására a vezérlés átadódik a 600-as sorra, és így tovább.

Vázlatunkban három szubrutin szerepelt, melyek mindegyikét csak egyszer hívtuk meg. Természetesen több eljárást is elhelyezhettünk volna a főprogram végére, illetve többször is futtathattuk volna őket.

2. A szubrutinok elhelyezése a programban

A BASIC nyelv alapváltozatában a részfeladatok, a modulok általában nem különülnek el élesen egymástól, amelynek legfőbb oka az, hogy nincs szubrutinnyitó utasítás. A szubrutin kezdetét nem jelzi semmi. (QBASIC-ben lehetőségünk van olyan utasítás alkalmazására is, amely teljesen elkülöníti az egyes eljárásokat a főprogramtól.)

A programot olvasó személy nem látja egy adott programsorról, hogy az egy szubrutin nyitósora, mert a szubrutin kezdősorának sorszámát csak a főprogram hívóutasításából olvashatjuk ki. Ezt mindenképpen célszerű kiküszöbölnünk, olyan módon, hogy a program olvasója számára a REM végre nem hajtandó utasítással üzenetet közvetítünk. Például beírjuk, hogy itt egy szubrutin kezdődik, vagy szöveggel arra is utalhatunk, hogy milyen feladat megoldására alkalmas az adott programrész. A REM utasítás lehetőséget teremt számunkra a program tagolására, logikai egységeinek elkülönítésére.

Az alprogram végét egyértelműen jeleznünk kell egy RETURN (vissza) utasítással. A RETURN szó után nem kell (és nem is szabad) beírni azt a sorszámot, ahol a program futása folytatódik, mert a RETURN utasítás hatására a vezérlés automatikusan visszakerül a szubrutint hívó GOSUB utasítást követő utasításra, márpedig GOSUB utasítás több is lehet. A szubrutin úgy hajtódik végre, mintha a részprogramot hívó GOSUB utasítás helyén lennének az alprogram utasításai.

A program áttekinthetősége érdekében a szubrutinokat célszerű a főprogram után elhelyezni. (Hiszen a program futása amúgy is a legkisebb sorszámú programsorral kezdődik.)

Olvasmány

Ha a program futásánál fontos szempont a gyorsaság, esetleg más fontos ok miatt, lehetséges a program olyan szerkesztése, ahol a szubrutinok a program elején helyezkednek el. Ilyenkor arra ügyeljünk, hogy a program indításakor a vezérlés a főprogramhoz kerüljön! Például főprogramunk a 600*-as sornál kezdődik. Ekkor a programot a* RUN 600*-zal vagy* GOTO 600*-zal indíthatjuk, illetve a program első sorában egy ugrást helyezünk el:*

```
10 GOTO 600
```

Szubrutinok nemcsak a főprogramból, hanem más szubrutinokból is hívhatók. Bonyolultabb programoknál az egyes eljárásokon belül is vannak ehhez rendelt szubrutinok, melyek az eljárás ismétlődő feladatait végzik el. Ezek együttesen adják a program felépítését, tervezését külön szakemberek, programtervezők végzik.

A vezérlő eljárásnak nemcsak az elejét kell világosan meghatároznunk, hanem a végét is. Le kell zárnunk a program futását egy STOP vagy egy END utasítással, különben a programfutás a program végére helyezett eljárások közül a legkisebb sorszámúval folytatódnék.

A program szokásos felépítése a következő:

```
10 REM FOPROGRAM KEZDETE
.
.
.
90 END : REM BEFEJEZODOTT A PROGRAM FUTASA
100 REM AZ ELSO ELJARAS
.
.
.
190 RETURN : REM VISSZA A FOPROGRAMBA
```

Ha lehetőségünk van END utasítás használatára, akkor azt alkalmazzuk, ha a program megfelelően fut le. A STOP utasítást csak hibajelzés esetén alkalmazzuk leállásra. Ne feledjük el ezeknek az utasításoknak a kiírását, mert elhagyásuk esetén a program tovább fut, végrehajtja az első eljárás utasításait is, és a RETURN utasításhoz érve hibajelzést kapunk.

Olvasmány

Leállíthatjuk a főprogram futását úgy is, hogy a főág befejezése után egy GOTO *utasítással a program fizikai végén lévő* END *utasításhoz ugratunk.*

A program felépítése a következőképpen módosul:

```
10 REM FOPROGRAM KEZDETE
.
.
90 GOTO 1000 : REM A PROGRAM VEGERE UGRUNK
100 REM Az ELSO ELJARAS
.
.
190 RETURN : REM VISSZA A FOPROGRAMBA
.
.
1000 END : REM BEFEJEZODOTT A PROGRAM FUTASA
```

3. A szubrutinok használatának előnyei

1. Lehetőséget teremtenek a program tagolására logikai szekezete szerint.
2. Logikailag zárt egységet alkotó részek, melyek maguk is programoknak tekinthetők.
3. A program bármely részéből hívhatók.
4. Lehetővé teszik egyes programrészek ismétlését.
5. Akárhányszor felhasználhatjuk őket a program futása során.
6. Az adott programrészt csak egyszer kell kódolni.

Példák

1. Elméleti ismereteinket egy nagyon egyszerű program futtatásával próbáljuk ki, amely a következők szerint működik:
   a) A főprogram a legkisebb sorszámtól kezdődik. Bekéri a felhasznált adatokat, és hívja a szubrutinokat.
   b) Az egyik szubrutin számolja ki két megkérdezett szám szorzatát! (Írja is ki, hogy mit csinál!)
   c) A másik szubrutin számolja ki két megkérdezett szám összegét, hívja meg az első szubrutint, majd írja ki a két szám összegét és szorzatát!
   d) A harmadik szubrutin folytasson velünk párbeszédet a program folytatása ügyében!

1. *Ezeket a feltételeket például a következő program is teljesíti C-64 esetén:*

```
10 REM FOPROGRAM
20 PRINT "KEZDODIK A FOPROGRAM" : PRINT
30 PRINT "KEREK KET SZAMOT!"
40 INPUT A,B
50 GOSUB 200
60 PRINT "FOLYTATODIK A FOPROGRAM FUTASA"
70 GOSUB 300
80 PRINT "AKAR TOVABB SZAMOLNI?"
90 GOSUB 400
100 IF V$="IGEN" THEN 30
110 PRINT "BEFEJEZTEM A MUNKAT"
120 END
200 REM AZ ELSO SZUBRUTIN
210 PRINT "AZ ELSO SZUBRUTINT HAJTOM VEGRE"
220 PRINT A; "*"; B; "="; A*B
230 PRINT "ELVEGEZTEM A SZORZAST"
240 RETURN
300 REM A MASODIK SZUBRUTIN
310 PRINT "MOST A MASODIK SZUBRUTIN FUT"
320 PRINT A; "+"; B; "="; A+B
330 GOSUB 200
340 PRINT "VISSZATERTEM A MASODIK SZUBRUTINBA"
360 PRINT "KIIRTAM AZ OSSZEGET ES A SZORZATOT"
370 RETURN
400 REM A HARMADIK SZUBRUTIN
410 PRINT "VALASZOLJON A KOVETKEZO SZAVAK EGYIKEVEL!"
420 PRINT : PRINT "IGEN" : PRINT
430 PRINT "NEM"
435 INPUT V$
440 RETURN
```

A program futtatása előtt írjuk le a végrehajtási sorrendet a sorszámok segítségével!

2. *A QBASIC-ben futtatott programunk esetén a feladat d) részét készítsük el* LOOP UNTIL *utasítással is!*

2. Egy bankban helyezzünk el p Ft-ot k %-os kamatra! Kíváncsiak vagyunk, hogy mennyi pénzt kapunk 2, 3, 5 és 7 év múlva, ha évente levonják a kamatadót.

A kamatadó, amit pénzünk kamatozása után levonnak, csökkenti a nyereségünket, tehát k %-os kamat helyett csak az a %-kal csökkentett kamatot, a K–K/100*A összeget kapjuk meg.

A vizsgált évek nem szabályosan változnak, ezért a kamatos kamat számolását szubrutin segítségével valósítjuk meg.

```
10 PRINT "KISZAMITOM A KAMATOS KAMATOT 2, 3, 5 ÉS 7 EVRE"
20 REM BEMENETI ADATOK
30 PRINT "A KOVETKEZO ADATOKAT KEREM:" : PRINT
40 PRINT "MENNYI PENZT HELYEZETT EL A TAKAREKBAN?"
50 PRINT
60 INPUT P : PRINT
70 PRINT "A KAMATLAB? (SZAZALEKBAN)" : PRINT
80 INPUT K
84 PRINT "A KAMATADO? (SZAZALEKBAN)" : PRINT
86 INPUT A
100 REM KEPERNYO TERVEZES ES SZAMITAS
110 PRINT "KAMATOS KAMAT SZAMITASA" : PRINT
120 PRINT "A KEZDOBETET:"; P; "Ft" : PRINT
130 PRINT "A KAMATLAB:"; K; "%" : PRINT
140 PRINT " A KAMATADO:"; A; "%" : PRINT
150 N=2
160 GOSUB 300
170 N=3
180 GOSUB 300
190 N=5
200 GOSUB=300
210 N=7
220 GOSUB 300
230 END
300 REM KAMATOS KAMATOT SZAMITO ELJARAS
310 K=K-K/100*A : REM ADOZAS UTANI KAMAT
320 S=P*((1+K/100)^N)
330 REM EREDMENY KIIRASA
340 PRINT
350 PRINT N ; "EVIG KAMATOZOTT A PENZEM"
360 PRINT "A PENZEM JELENLEG:"; S; "Ft"
370 RETURN
```

Kérdések

1. A részprogramok megismétlésének milyen módjaival foglalkoztunk? Mi a lényeges különbség a két módszer között?
2. Mit neveztünk eljárásnak, és hogyan kapcsolódott a főprogramhoz?
3. Hogyan valósíthatjuk meg a szubrutin hívását?
4. Hogyan kezdődik egy szubrutin, és hogyan fejeződik be?
5. Hol folytatódik a program futása egy eljárás futásának befejezése után?
6. Honnan valósíthatunk meg szubrutinhívást?
7. Hányszor hívhatunk egy szubrutint?
8. Hol kezdődik egy program futása, és mikor fejeződik be?
9. Hogyan akadályozhatjuk meg, hogy egy program futása a szubrutinnal kezdődjék? Mit kell tennünk, hogy a program futása ne valamelyik szubrutin végrehajtásával fejeződjön be (hibajelzéssel)?
10. Mit jelent a szubrutinok egymásba skatulyázása? Hogyan történik ilyenkor a program végrehajtása?
11. Milyen sorrendben hajtja végre a számítógép a következő utasításokat? Írjuk le az utasítások sorszámát a végrehajtás sorrendjében!

```
10  A=3  :  I=1
20  PRINT "A KEZDOERTEK:"; A
30  A=A+1
40  IF  A=20  THEN  100
50  GOSUB  80
60  GOTO  30
70  REM  LEPTETO  SZUBRUTIN  KEZDODIK
80  I=I+1
90  RETURN
100  PRINT  A*5
110  END
```

A program futtatása nélkül írjuk le a kiíratási képet!
A megoldást ellenőrizzük a program futtatásával!

G) BASIC függvények

Programozásunk során szükségünk lehet a matematikában megszokott függvények alkalmazására. Szeretnénk, ha számítógépünk (a zsebszámológéphez hasonlóan) megkímélne bennünket a függvénytáblázatok használatától. Erre van lehetőségünk, mert a BASIC nyelv is tartalmaz beépített függvényeket, melyek közül néhányat ismertetünk.

1. A négyzetgyök kiszámítása

A négyzetgyök kiszámítása (nem negatív számok esetén) az SQR függvény segítségével történik.

Például C-64 esetén ez a következő lehet:

```
10 PRINT "SZAMOK NEGYZETGYOKENEK KISZAMOLASA"
20 PRINT "KEREM A SZAMOT"
30 INPUT A
40 LET B=SQR (A)
50 PRINT "A BEVITT SZAM A="; A
60 PRINT
70 PRINT "A SZAM NEGYZETGYOKE B="; B
```

2. Egészrészfüggvény

Az adott számnál nem nagyobb egészek közül a legnagyobbat adja az INT (X) függvény.

Például:
```
INT (2.8)=2
INT (2.1)=2
INT (-4.3)=-5
```

(Előfordulhat, hogy egyes BASIC-változatokban negatív szám esetén rosszul működik.)

3. Trigonometrikus függvények

Használatuk esetén vigyáznunk kell, hogy a szögértéket radiánban adjuk meg. (Ha az X szög fokban adott, akkor az $X*3.14/180^0$ képlet segítségével számoljuk át radiánba).

Például: Készítsünk függvénytáblázatot, amely szögek szinuszát számolja ki, és egyfokonként kiírja azt!

```
10 PRINT "FUGGVENYTABLAZAT SIN X KISZAMOLASARA"
20 PRINT "SZOG"; "A SZOG SZINUSZA"
30 FOR I=1 TO 90
40 S=SIN(I*3.14/180)
50 PRINT I, S
60 NEXT I
```

Ugyanígy számolhatjuk a COS(X) és TAN(X) függvényt is.

4. A logaritmusfüggvények

A logaritmusfüggvények közül a BASIC-ben a természetes alapú logaritmust, LOG(x)-et számolhatjuk ki. „e"-től különböző alapú logaritmust is számolhatunk a $\log_b(x) = \log_e(x)/\log_e(b)$ összefüggéssel, ahol b az (1-től különböző, pozitív) alap.

5. Abszolútérték-függvény: ABS(X)

```
10 PRINT "ABSZOLUTERTEK-SZAMOLAS"
20 PRINT "KEREM A SZAMOT" : PRINT
30 INPUT X
35 Y=ABS(X)
40 PRINT "A MEGADOTT SZAM: X="; X
50 PRINT "A SZAM ABSZOLUTERTEKE: IXI="; Y
```

6. Véletlenszámot előállító függvény

RND(X) (Ezt már korábban megvizsgáltuk.)

7. Egyéb függvények

Egyéb függvényeket is értelmezhetünk, például amely az előző függvényeket felhasználva további műveletek eredményét adja meg. *Például:*

```
PRINT 3*A+ABS(SIN(X))-4
```

Lehetőség van arra is, hogy saját függvényt definiáljunk. Az ilyen függvények nevének FN-nel kell kezdődni, és általában még egy betű azonosítja. Ezeket felhasználás előtt külön kell definiálni. *Például:*

```
FNG(A)      a saját függvényünk
100 DEF FNG(A)=2*A-4*SIN(A)/7+5
```

Olyan esetekben szoktuk ezt használni, amikor a program futása során többször van szükségünk egy függvény értékének kiszámolására. A használható függvényekről a számítógép gépkönyve tájékoztat. A függvények köre erre a célra szolgáló eljárásokkal tetszőlegesen bővíthető.

Kérdések

1. Milyen szerepe van a beépített függvénynek a programozás során?
2. Milyen beépített függvényekről tanultunk?
3. Függvényt határoznak-e meg a következő kapcsolatok?

```
COS(X)
ABS(-2*X)
SQR(32)
```

4. Milyen értékeket kapunk a következő esetekben?

```
Y=INT(1/2)
Z=SQR(X)
```

5. Mire kell ügyelnünk trigonometrikus függvényeknél a szögek megadásakor?
6. Hogyan számoljuk ki egy szám tízes alapú logaritmusát?
7. Hogyan használhatjuk az RND függvényt 1-nél nagyobb egész számok előállítására?
8. Hogyan ismerhetjük fel,hogy az adott függvényt a program írója hozta létre?

VI. A számítógép használata matematikai problémák megoldására

Az előző fejezetekben megtanultuk azokat az alapvető programozási ismereteket, amelyek segítségével matematikai feladatokat is megoldhatunk.

Ebben a fejezetben keressünk közösen megoldásokat egy-egy matematikai probléma számítógépes megoldására!

Az eddig megírt programjainkban nem használtunk ékezetes betűket, de van arra lehetőség, hogy egy szoftver segítségével a magyar ábécének megfelelő betűket alkalmazzunk. A következő programokat így készítettük, így érthetőbbé váltak a beírásaink.

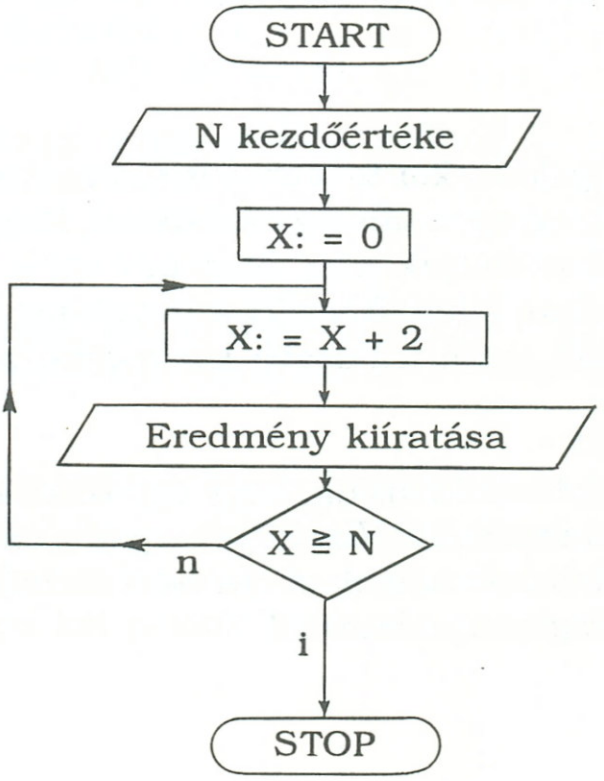
Példák

1. Készítsünk programot, amely kiíratja az egymás után következő pozitív páros számokat

   a) N = 33-ig; *b)* tetszőleges értékig!

   Megoldás:

   A feladat megoldásának algoritmusát a *27. ábra* folyamatábrája szemlélteti:



27. ábra

1. *Egy lehetséges program az algoritmus leírására C-64-re vonatkozóan:*

a)
```
10 PRINT "POZITÍV PÁROS SZÁMOK KIÍRÁSA"
20 PRINT "A 33-IG ELŐFORDULÓ POZITÍV PÁROS SZÁMOK"
30 N=33
40 X=0
50 X=X+2
60 PRINT X
70 IF X>=N THEN END
80 GOTO 50
```

b) Változások a következő sorokban jelentkeznek:

```
20 PRINT "MEDDIG ÍRJAM KI A POZITÍV PÁROS SZÁMOKAT?"
30 INPUT N
```

Lerövidíthetjük programunkat, ha feltételes vezérlésátadás helyett ciklusutasítást alkalmazunk.

Változások:

```
40 FOR I=2 TO N STEP 2
50 PRINT I
60 NEXT I
70 END
```

2. *QBASIC-beli program lehet:*

```
PRINT "POZITÍV PÁROS SZÁMOK KIÍRÁSA"
PRINT "MEDDIG ÍRJAM KI A POZITÍV PÁROS SZÁMOKAT?"
INPUT N
X=2
DO WHILE X<N
  PRINT X
  X=X+2
LOOP
END
```

Újabb feladataink megoldásakor továbbra is számokkal foglalkozunk. Megnézzük, hogy mikor osztható egy szám egy másikkal, hogyan tudunk prímeket előállítani, és hogyan állapíthatjuk meg, hogy egy szám prím-e.

Ennyi ismeret birtokában elkészíthetjük a törzstényezőre bontás programját, és megkereshetjük két szám legkisebb közös többszörösét és legnagyobb közös osztóját is.

2. Írjunk programot, melyben eldöntjük, hogy egy természetes szám osztója-e egy másik természetes számnak! Vizsgálja meg a számítógép, hogy a bevitt adatok valóban természetes számok-e! Ha nem természetes számot írunk, akkor kérjen a számítógép új adatokat!

1. *A C-64-en futtatható programunk lehet:*

```
10 PRINT "KÉREK KÉT TERMÉSZETES SZÁMOT!"
20 PRINT "MEGVIZSGÁLOM, HOGY OSZTÓJA-E EGYIK A MÁSIKNAK"
30 INPUT "AZ OSZTANDÓ:"; A
40 IF A<=0 THEN 140
50 IF A<>INT (A) THEN 140
60 INPUT "AZ OSZTÓ:"; B
70 IF B<=0 THEN 180
80 IF B<>INT (B) THEN 180
90 K=A/B
100 L=INT(K)
110 IF L*B=A THEN PRINT B; "OSZTÓJA" ; A; "-NAK" : END
120 PRINT B; "NEM OSZTÓJA"; A; "-NAK"
130 END
140 PRINT "A BEÍRT ADAT NEM TERMÉSZETES SZÁM"
150 PRINT "ÚJ ADATOT KÉREK"
160 GOTO 30
170 END
180 PRINT "TERMÉSZETES SZÁMOT KÉREK"
190 GOTO 60
200 END
```

A megoldásnál felhasználtuk, hogy egy szám akkor osztója egy másiknak, ha a hányados egészrésze megegyezik a hányadossal, tehát a hányados egészrészének és az osztónak a szorzata megadja az osztandót.

2. *A QBASIC-ben írt programunkban* igyekezzünk elkerülni a feltétel nélküli vezérlésátadó utasítást, hogy ne kelljen címkét alkalmaznunk!

```
PRINT "KÉREK KÉT TERMÉSZETES SZÁMOT!"
PRINT "MEGVIZSGÁLOM, HOGY OSZTÓJA-E EGYIK A MÁSIKNAK"
DO
  PRINT "ÍRJA BE AZ ELSŐ TERMÉSZETES SZÁMOT!"
  INPUT "AZ OSZTANDÓ:",A
LOOP UNTIL A>=0 AND A=INT(A)
DO
  PRINT "A MÁSIK TERMÉSZETES SZÁM?"
  INPUT "AZ OSZTÓ:",B
LOOP UNTIL B>=0 AND B=INT(B)
K=A/B
I=INT(K)
IF I*B = A THEN PRINT B; "OSZTÓJA"; A;"-NAK"
  ELSE PRINT B; "NEM OSZTÓJA"; A;"-NAK"
END
```

3. Osztáskor gyakran kerülünk olyan helyzetbe, hogy a hányados nem lesz egész szám. Előfordul, hogy csak a hányados egészrészére vagyunk kíváncsiak, és a maradékot valami másra szeretnénk felhasználni. Írjunk programot, amely elvégzi két pozitív egész szám maradékos osztását!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "KÉREK KÉT TERMÉSZETES SZÁMOT"
20 INPUT "AZ OSZTANDÓ:"; A
30 IF A<=0, THEN 130
40 INPUT "AZ OSZTÓ:"; B
50 IF B<=0 THEN 40
60 K=A/B
70 L=INT(K)
80 M=A-L*B
90 PRINT "A HÁNYADOS"; L : PRINT
100 PRINT "A MARADÉK"; M : PRINT
110 PRINT A; "="; L; "*"; B; "+"; M
120 END
130 PRINT "TERMÉSZETES SZÁMOT KÉREK"
140 GOTO 20
```

A megoldásnál alkalmaznunk kellett azt a gondolatot, hogy a két szám hányadosának egészrésze adja a maradékos osztás hányadosát. Ennek ismeretében a maradékot egy kivonás segíségével kaphatjuk meg. Alkalmazzuk eljárásunkat abban az esetben, amikor szögperceket akarunk fokká és perccé alakítani, hogy a függvénytáblázatból ki tudjuk keresni a megfelelő szögfüggvényeket. Ilyenkor az A-val jelzett szögperc értékét kell maradékosan elosztanunk B=60-nal.

Például 134' = 2°14'

2. *QBASIC-ben írt programunkban alkalmazzuk a* LOOP WHILE *utasítást! (Programunk első része megegyezhetne a 2. példában leírt programmal.)*

```
PRINT "MARADÉKOS OSZTÁST VÉGZÜNK"
DO
  PRINT "ÍRJA BE AZ ELSŐ TERMÉSZETES SZÁMOT"
  INPUT "AZ OSZTANDÓ:";A
LOOP WHILE A<=1 OR A<>INT(A)
DO
  PRINT "A MÁSODIK TERMÉSZETES SZÁM?"
  INPUT "AZ OSZTÓ:",B
LOOP WHILE B<=1 OR B<>INT(B)
LET K=A/B
LET I=INT(K)
LET M=A-I*B
PRINT "A HÁNYADOS:"; I
PRINT
PRINT "A MARADÉK:"; M
PRINT
PRINT A; "="; I; "*"; B; "+"; M
END
```

4. Gyakran előfordul, hogy a bevitt adatokat kerekítjük, csupán kéttizedes-pontossággal adjuk meg. Ilyen esetekben elegendő az eredményt így megadni.

Készítsünk olyan programot, amely egy osztás eredményeként kapott értékeket két tizedesre kerekíti, és természetesen kizárja a 0-val történő osztás lehetőségét!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "OSZTÁS UTÁNI KEREKÍTÉST VÉGZEK"
20 INPUT "KÉREM AZ OSZTANDÓT" ; A
30 INPUT "KÉREM AZ OSZTÓT" ; B
40 IF B=0 THEN 100
50 C=A/B
60 D=INT ((C*100+0.5))/100
80 PRINT "A KEREKÍTETT EREDMÉNY:"; D
70 PRINT "A PONTOSABBAN SZÁMOLT HÁNYADOS:"; C
90 END
100 PRINT "A 0-VAL VALÓ OSZTÁSNAK NINCS ÉRTELME"
110 PRINT "ÚJ OSZTÓT KÉREK"
120 GOTO 30
```

2. *QBASIC-beli programunk lehet:*

```
CLS
PRINT "OSZTÁS UTÁNI KEREKÍTÉST VÉGZEK"
INPUT "KÉREM AZ OSZTANDÓT!"; A
DO
  INPUT "KÉREM AZ OSZTÓT!"; B
LOOP WHILE B=0
LET C=A/B
LET D=INT((C*100+0.5))/100
PRINT "A KEREKÍTETT EREDMÉNY:"; D
PRINT "A PONTOSABBAN SZÁMOLT HÁNYADOS:"; C
END
```

5. Egy számról úgy állapíthatjuk meg, hogy törzsszám-e, hogy sorba vesszük a számokat egészen az eredeti szám négyzetgyökéig, és megvizsgáljuk azok oszthatóságát az eredeti számmal. Ha ez a hányados nem egyezik meg a hányados egészrészével, azaz a vizsgált szám nem osztható az éppen soron következő számmal, akkor a vizsgált szám törzsszám.
Döntsük el egy pozitív számról, hogy prím-e?

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "ÍRJ BE EGY POZITÍV EGÉSZ SZÁMOT!"
20 PRINT "MEGVIZSGÁLOM; HOGY PRÍM-E"
30 INPUT N
35 IF N<=1 OR N<>INT(N) THEN 120
40 IF N=3 OR N=2 THEN 90
50 I=1
60 I=I+1
70 IF N/I=INT(N/I) THEN 110
```

```
80 IF I<INT (SQR(N))+1 THEN 60
90 PRINT N; "PRÍMSZÁM"
100 END
110 PRINT N; "NEM PRÍMSZÁM"
115 END
120 PRINT "ÚJ SZÁMOT KÉREK" : GOTO 30
```

2. *QBASIC-beli programunkban ügyeljünk a címkézésre!*

```
    PRINT "SZÁMOK VIZSGÁLATA"
 30:INPUT N
    IF N=3 OR N=2 THEN 90
    IF N<=1 OR N<>INT(N) THEN 120
    I=1
 60:I=I+1
    IF N/I=INT(N/I) THEN 110
    IF I<INT(SQR(N))+1 THEN 60
 90:PRINT N; "PRÍMSZÁM"
    END
110:PRINT N; "NEM PRÍMSZÁM"
    END
120:PRINT "ÚJ SZÁMOT KÉREK": GOTO 30
```

Programunkat ciklusutasítással is megoldhatjuk, ezzel elkerüljük a címkék alkalmazását.

```
CLS
PRINT "SZÁMOK VIZSGÁLATA"
INPUT "KÉREK EGY SZÁMOT"; N
IF N=2 OR N = 3 THEN PRINT N; "PRÍM" :END
IF N<=1 OR N<>INT(N) PRINT "ÚJ SZÁMOT KÉREK": END
I=1
DO
    I=I+1
    IF N/I=INT(N/I) THEN PRINT N;"NEM PRÍM": END
LOOP WHILE I<INT(SQR(N))+1
PRINT N; "PRÍM"
END
```

6. Írjunk programot, amely egy adott számig kiírja az összes prímszámot!

1. *C-64-en futtatható programunk lehet:*

```
 10 PRINT "KIÍROM A PRÍMSZÁMOKAT EGY MEGADOTT HATÁRIG"
 20 PRINT "MEDDIG ÍRJAM KI A PRÍMEKET?"
 30 INPUT N
 40 FOR I=1 TO N
 50 J=1
 60 J=J+1
 70 IF I/J=INT(I/J) THEN 100
 80 IF J<INT(SQR(I))+1 THEN 60
 90 PRINT I; "PRÍMSZÁM"
100 NEXT I
```

2. *QBASIC-ben futtatható programunk:*

```
PRINT "KIÍROM A PRÍMSZÁMOKAT EGY ADOTT HATÁRIG"
PRINT "MEDDIG ÍRJAM KI A PRÍMEKET?"
INPUT N
FOR I=3 TO N
   J=1
60:J=J+1
        IF I/J=INT(I/J) THEN 100
        IF J<INT (SQR(I))+1 THEN 60
        PRINT I; "PRÍMSZÁM"
100:NEXT I
```

Az előzőhöz képest megváltoztattuk a ciklus kezdőértékét, azért, hogy 3-tól kezdje kiíratni a számokat, így nem jelent problémát, hogy a 2 is prím. Ezt külön kiírathatjuk.

7. Bontsunk törzstényezőire egy beadott számot! Ehhez csak azt kell megvizsgálnunk, hogy az adott szám mely számokkal osztható és hányszor. Az adatok tárolására egy tömböt használunk.

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "ÍRJ BE EGY POZITÍV EGÉSZ SZÁMOT!"
20 PRINT "KIÍROM PRÍMEK SZORZATAKÉNT"
30 DIM A(100) : REM EBBEN A TÖMBBEN ŐRIZZÜK A PRÍMEKET
40 INPUT N
50 IF N<0 OR N<>INT(N) THEN 40
60 M=N
70 S=0 : REM A TÖMB AKTUÁLIS ELEMÉNEK HELYE
80 I=1
90 I=I+1
100 IF M/I=INT (M/I) THEN S=S+1 : A(S)=I : M=M/I : GOTO 80
110 IF I<M THEN 90
120 IF S=1 THEN PRINT N ; "PRÍMSZÁM" : END
130 PRINT "A SZÁM TÖRZSTÉNYEZŐI:"
140 FOR I=1 TO S
150 PRINT A(I);
160 NEXT I
```

2. *QBASIC-beli programunkat készítsük el önállóan a 6. példa alapján! Ahol lehet, alkalmazzunk ciklusutasítást!*
   Például az adatok bevitelekor:

```
DO
  INPUT N
LOOP WHILE N<0 OR N<>INT(N)
```

8. Két pozitív egész szám legnagyobb közös osztójának és legkisebb közös többszörösének meghatározására különböző eljárásokat ismerünk. Matematikaórán megismerkedtünk a törzstényezős felbontáson alapuló módszerrel, de az algoritmus számítógépen nehezebben megvalósítható. Ismer-

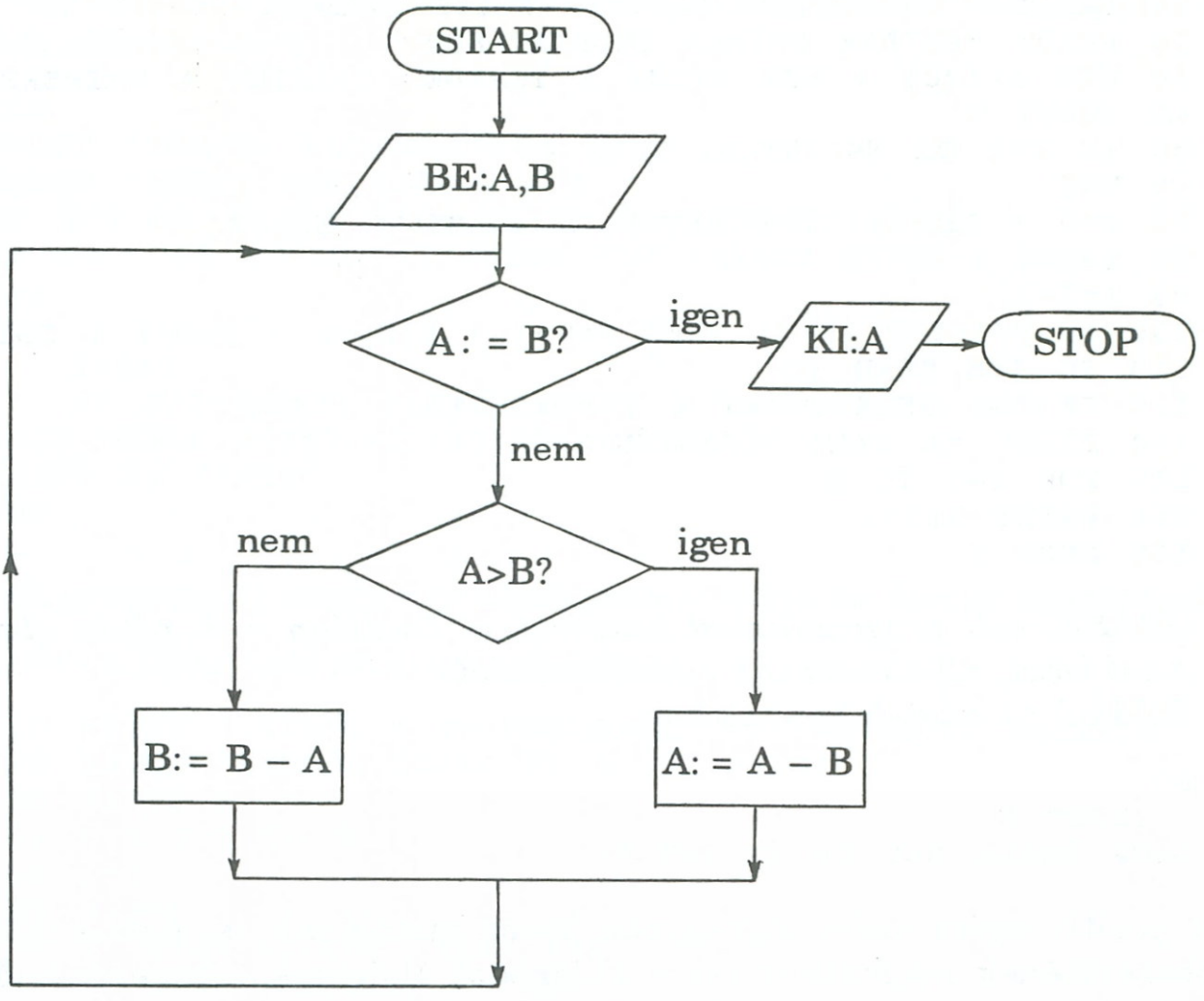
kedjünk meg most újabb lehetőségekkel, melyeket könnyen programozhatunk!

Az első megoldásban azt az alapgondolatot használjuk fel közös osztó keresésekor, hogy ha egy c szám osztója a-nak és b-nek is, akkor osztója $a–b$-nek is. A két szám különbsége viszont mindig kisebb a nagyobbik számnál.

Eljárásunk során a nagyobb szám helyett a két szám különbségét tekintjük mindaddig, amíg a különbség 0 nem lesz. Ha a különbség 0, akkor a két szám egyenlő, és ezek a számok megadják a legnagyobb közös osztót. (A számolás során a vizsgált két szám értéke csökken, ezért véges sok lépésben el kell jutnunk a legnagyobb közös osztóhoz.)

Legkisebb közös többszörös meghatározásához felhasználjuk, hogy két szám legkisebb közös többszörösének és legnagyobb közös osztójának szorzata megegyezik a két szám szorzatával.

Programunkban zárjuk ki a rossz adatokat, vagyis kérjen a számítógép új számokat, ha nem megfelelőket írtunk be. Ne csak a legnagyobb közös osztót, hanem a legkisebb közös többszöröst is írjuk ki! Ezeket a lépéseket a *28. ábra* blokkdiagramja nem tartalmazza, de programunk így teljesebb lesz.



28. ábra

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "KÉREK KÉT POZITÍV EGÉSZ SZÁMOT" : PRINT
20 PRINT "MEGHATÁROZOM A LEGNAGYOBB KÖZÖS OSZTÓT" : PRINT
30 PRINT "ÉS A LEGKISEBB KÖZÖS TÖBBSZÖRÖST": PRINT
40 INPUT "A SZÁMOK"; A, B
50 IF A<=0 OR B<=0 THEN 100
60 IF A<>INT(A) OR B<>INT(B) THEN 100
65 LET S=A*B
68 PRINT "A MEGADOTT KÉT SZÁM:"; A; "ÉS"; B
70 IF A=B THEN 110
80 IF A>B THEN LET A=A-B : GOTO 70
90 LET B=B-A : GOTO 70
100 PRINT "HIBÁS ADATOT KAPTAM, ÚJAT KÉREK" : GOTO 40
110 LET O=A
120 LET T=S/O
130 PRINT "A LEGKISEBB KÖZÖS TÖBBSZÖRÖS:"; T: PRINT
140 PRINT "A LEGNAGYOBB KÖZÖS OSZTÓ"; O: PRINT
```

2. *QBASIC-ben ugyanígy futtatható programunk, de írjuk meg ciklusok segítségével is!*

```
PRINT "KÉREK KÉT POZITÍV EGÉSZ SZÁMOT"
PRINT "MEGHATÁROZOM A LEGNAGYOBB KÖZÖS OSZTÓJUKAT"
PRINT "ÉS A LEGKISEBB KÖZÖS TÖBBSZÖRÖSÜKET"
DO
    INPUT "A SZÁMOK:"; a, b
LOOP WHILE a<=0 OR b<=0 OR a<>INT(a) OR b<>INT(b)
LET s=a*b
PRINT "A MEGADOTT KÉT SZÁM:"; a; "ÉS" ; b
DO
    IF a>b THEN LET a=a-b ELSE LET b=b-a
    LET o=a
    LET t=s/o
LOOP WHILE a<>b
PRINT "A LEGKISEBB KÖZÖS TÖBBSZÖRÖS:"; t
PRINT "A LEGNAGYOBB KÖZÖS OSZTÓ:";o
```

9. A legnagyobb közös osztó keresésére egy újabb eljárás az euklideszi algoritmus, melynek lényege a következő:

Ha $a \geq b$, akkor:
1. a-t maradékosan elosztjuk b-vel:
(A: = BH + M; O<=M<B)

2. Ismételjük az eljárást, amíg $m \neq 0$; a következő lépésekben:
A: = B; B: = M
A: = BH + M, vagyis az előző osztót elosztjuk az előző maradékkal.

3. Az utolsó nullától különböző maradék a legnagyobb közös osztó.

1. *C-64 esetén a program elején az adatok bekérése megegyezik az előzővel. A 70-es sortól módosul.*

```
70 IF A>=B THEN 90
80 LET C=A : LET A=B : LET B=C : REM TÁRAK ÉRTÉKCSERÉJE
90 LET H=A/B
100 LET L=INT(H)
110 LET M=A-B*L
120 IF M=0 THEN 150
130 LET A=B : LET B=M
140 GOTO 90
150 LET O=B
160 LET T=S/O
170 IF O=1 THEN PRINT "RELATÍV PRÍMEK" : END
180 PRINT "LNKO:"; O; "LKKT:"; T
190 END
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
PRINT "KÉREK KÉT POZITÍV EGÉSZ SZÁMOT!"
PRINT "MEGHATÁROZOM A LEGNAGYOBB KÖZÖS OSZTÓJUKAT!"
PRINT "ÉS A LEGKISEBB KÖZÖS TÖBBSZÖRÖSÜKET"
DO
  INPUT "A SZÁMOK:"; a,b
LOOP WHILE a<=0 OR b<=0 OR a<>INT(a) OR b<>INT(b)
LET s=a*b
PRINT "A MEGADOTT KÉT SZÁM:"; a; "ÉS"; b
DO WHILE a<b
  LET c=a: LET a=b: LET b=c
LOOP
LET h=a/b
LET e=INT(h)
LET m=a-b*e
DO WHILE m<>0
  LET a=b: LET b=m
LOOP
LET o=b
LET t=s/o
IF o=1 THEN PRINT "RELATÍV PRÍMEK"
  ELSE PRINT "LNKO:"; o;"LKKT"; t
END
```

10. A számrendszerek tanulásakor láttuk, hogy a különböző számrendszerekben megadott számokat átírhatjuk közös számrendszerbeli számmá. Ez a konvertálás „kézi" számolással kicsit nehézkes, próbáljunk programot írni a maradékos osztás segítségével.

Számolásunk során a következő algoritmust hajtjuk végre:

— Az adott számot elosztjuk a számrendszer alapszámával. (Tekintjük a hányados egészrészét, és megállapítjuk a maradékot.) Ha a hányados

0, akkor készen vagyunk. Ellenkező esetben megjegyezzük a maradékot. (Egy tömb nulladik elemeként tartjuk nyilván.)
— Eljárásunkat mindaddig folytatjuk, amíg a hányados nem lesz egyenlő 0-val. (Az előző hányadost osztjuk az alappal, és a maradékot beírjuk a tömb következő elemének.)
— Az osztások végén kiíratjuk a tömb elemeit fordított sorrendben.

Írjunk programot, amely egy tízes alapú számrendszerbeli számot nyolcas alapú számrendszerbe konvertál!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "KÉREK EGY TÍZES SZÁMRENDSZERBELI"
20 PRINT: PRINT "SZÁMOT; KIÍROM NYOLCAS"
30 PRINT: PRINT "SZÁMRENDSZERBEN"
40 INPUT N
45 B=N
50 I=0
60 H=INT(N/8)
70 M(I)=N-8*H
80 N=H
90 I=I+1
100 IF N<>0 THEN 60
110 PRINT "SHIFT+CLR/HOME"
120 PRINT B "SZÁM NYOLCAS SZÁMRENDSZERBELI"
130 PRINT "ALAKJA:";
140 FOR J=I-1 TO 0 STEP-1
150 PRINT M(J);
160 NEXT J
```

2. *Programunk QBASIC-beli megfelelője a következő lehet:*

```
CLS
PRINT "KÉREK EGY TÍZES SZÁMRENDSZERBELI"
PRINT : PRINT "SZÁMOT KIÍROM NYOLCAS"
PRINT : PRINT "SZÁMRENDSZERBEN"
INPUT n
LET b=n
LET i=0
DO
  LET h=INT(n/8)
  LET m(i)=n-8*h
  LET n=h
  LET i=i+1
LOOD WHILE n<>0
PRINT "A(z)"; b; "SZÁM NYOLCAS SZÁMRENDSZERBELI"
PRINT : PRINT "ALAKJA:";
FOR j=i-1 TO 0 STEP-1
  PRINT m(j);
NEXT j
```

A 10. példában konkrétan megadtuk a számrendszerek alapszámát, így feladatunk megoldását csak módosítással alkalmazhatjuk más esetekben.

11. Írjunk programot, amely tetszőleges pozitív, tízes számrendszerbeli egész számnak kiírja tetszőleges alapú számrendszerbeli megfelelőjét!

1. *C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "KÉREK EGY TÍZES SZÁMRENDSZERBELI": PRINT
20 PRINT "EGÉSZ SZÁMOT; ÁTÍROM TETSZŐLEGES": PRINT
30 PRINT "SZÁMRENDSZERBE (1<N<10)" : PRINT
40 INPUT "A SZÁM:"; SZ
50 IF SZ<=0 OR SZ<>INT(SZ) THEN 200
60 INPUT "AZ ÚJ SZÁMRENDSZER ALAPJA:"; N
70 IF N<=1 ORN>9 OR N<>INT(N) THEN 230
80 PRINT "AZ ADOTT"; SZ; "SZÁM"; N; "SZÁMRENDSZERBELI ALAK-
   JA:"
90 IF SZ<N THEN PRINT SZ : END
100 DIM M(16)
110 I=0
120 H=SZ/N
130 L=INT(H)
140 M(I)=SZ-L*N
150 IF L>0 THEN 260
160 FOR J=I TO 0 STEP-1
170 PRINT M(J);
180 NEXT J
190 END
200 PRINT "ÚJ ADATOT KÉREK; MERT NEM"
210 PRINT "TERMÉSZETES SZÁMOT ÍRT BE"
220 GOTO 40
230 PRINT "AZ ALAP CSAK 1 ÉS 9 KÖZÉ ESHET"
240 GOTO 60
260 SZ=L
270 I=I+1
280 GOTO 120
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "KÉREK EGY TÍZES SZÁMRENDSZERBELI": PRINT
PRINT "EGÉSZ SZÁMOT, ÁTÍROM TETSZŐLEGES": PRINT
PRINT "SZÁMRENDSZERBE (1<n<10)" : PRINT
DO
   INPUT "A TERMÉSZETES SZÁM:";sz
LOOP WHILE sz<=0 OR sz<>INT(sz)
DO
   PRINT "AZ ÚJ SZÁMRENDSZER ALAPJA:"
   INPUT "(1<n<10)"; n
LOOP WHILE n<=1 OR n>9 OR n<>INT(n)
PRINT "AZ ADOTT"; Sz; "SZÁM"; n; "SZÁMRENDSZERBELI ALAKJA:";
IF sz<n THEN PRINT sz : END
DIM m(100)
LET i=0
DO
   LET h=sz/n
```

```
   LET k=INT(h)
   LET m(i)=sz-k*n
   LET sz=k
   LET sz=i+1
LOOP WHILE k<>0
FOR j=i TO 0 STEP-1
   PRINT m(j);
NEXT j
END
```

Azt is láttuk, hogy nemcsak tízes számrendszerből alakíthatunk más számrendszerbe számokat, hanem ez fordítva is megvalósítható. Adott bináris számot átalakíthatunk tízes számrendszerbe.

12. Egy legfeljebb nyolcjegyű bináris szám általános alakja:
$A * 2^7 + B * 2^6 + C * 2^5 + D * 2^4 + E * 2^3 + F * 2^2 + G * 2 + H$, ahol az ábécé betűi a 0 vagy 1 számjegyeket helyettesítik, ezek értékétől függ az adott szám tízes számrendszerbeli értéke. Érdemes ezt az általános alakot beírni a programba, és az együtthatókat változóként bevinni, mert akkor bármely szám átírására alkalmazhatjuk. Használhatunk a bináris szám értékének kiszámításához egy másik képletet is, amelyet kiemeléssel kaphatunk.

Az ((((((A * 2 + B) * 2 + C) * 2 + D) * 2 + E) * 2 + F) * 2 + G) * 2 + H formula segítségével egy azonos alakú utasításokból álló utasításrendszerhez is eljuthatunk.

Írjunk programot, amely legfeljebb 8-jegyű (8 bites) bináris számnak kiírja decimális megfelelőjét! (Az eljárás bővíthető.)

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "KÉREK EGY 8 BITES BINÁRIS SZÁMOT"
20 PRINT: PRINT "KIÍROM TÍZES SZÁMRENDSZERBELI"
30 PRINT: PRINT "MEGFELELŐJÉT"
40 INPUT "KETTŐ HATVÁNYAINAK EGYÜTTHATÓI:";A,B,C,D,E,F,G,H
50 LET N =A*2^7+B*2^6+C*2^5+D*2^4+E*2^3+F*2^2+G*2+H
60 LET T =(((((((A*2+B)*2+C)*2+D)*2+E)*2+F)*2+G)*2+H
70 PRINT "SHIFT+CLR/HOME"
80 PRINT "A(z)"; A; B; C; D; E; F; G; H; "SZÁM TÍZES"
90 PRINT: PRINT "SZÁMRENDSZERBELI ALAKJA:"; T
100 PRINT: PRINT "MÁSKÉPPEN SZÁMOLVA:"; N
```

A kiírásból láthatjuk, hogy a két számolási eljárás ugyanazt a decimális számot eredményezi, így elegendő programunkban az egyiket szerepeltetni. (50-es vagy 60-as sor.)

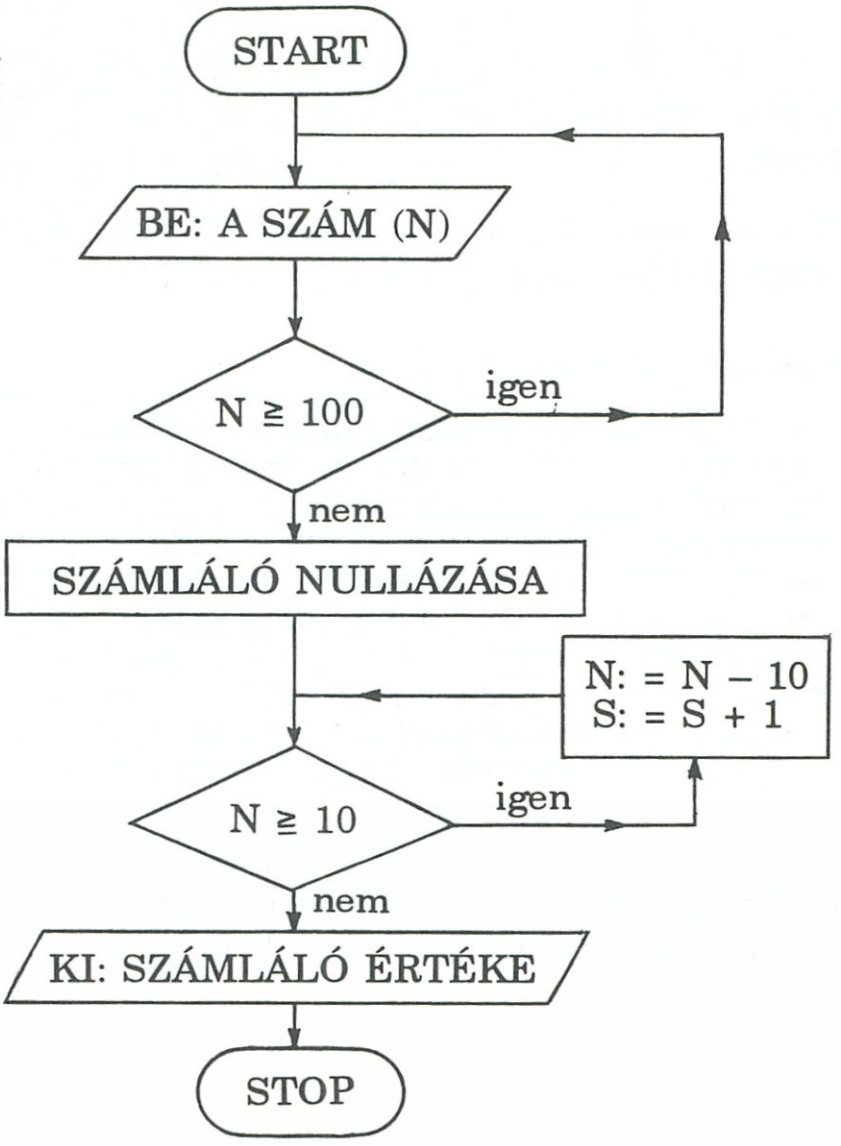
2. *Programunk a sorszámok elhagyásával QBASIC-ben is futtatható.* Természetesen megoldhatjuk feladatunkat zárójelek nélkül is, de ilyenkor több utasítást használunk. Egy lehetséges megoldás:

```
LET T=A
LET T=T*2+B
LET T=T*2+C
LET T=T*2+D
LET T=T*2+E
LET T=T*2+F
LET T=T*2+G
LET T=T*2+H
```

Próbáljuk ki ezt a lehetőséget is! Az előző feladatok segítségével írjunk programot, amely tizedestörtet kettedes törtté alakít!

13. A számrendszerek közti konverzióra a számok helyiértékét használjuk fel. Készítsünk olyan algoritmust, melynek segítségével a számokat helyiértékük szerint vizsgálhatjuk *(29. ábra)*!

Írjunk programot, amely megszámolja, hogy adott kétjegyű számban hány darab tízes van. A kiírásban az egyesek száma is jelenjen meg!



29. ábra

Folyamatábránkat bővíthetjük, és akkor vizsgálhatjuk nagyobb számok esetén, hogy hány százas, ezres, tízezres stb. van bennük.

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "MEGSZÁMOLOM; HOGY HÁNY TÍZES VAN"
20 PRINT "A BEÍRT SZÁMBAN"
30 INPUT "KÉREK EGY 100-NÁL KISEBB SZÁMOT"; N
40 IF N>=100 THEN 120
50 S=0
60 IF N>=10 THEN GOTO 90
70 PRINT S; "DB TIZES VAN A SZÁMBAN"
75 PRINT N; "DB EGYES VAN A SZÁMBAN"
80 END
90 N=N-10
100 S=S+1
110 GOTO 60
120 PRINT "KÉTJEGYŰ SZÁMOT KÉREK"
130 GOTO 30
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "MEGSZÁMOLOM, HOGY HÁNY TÍZES VAN"
PRINT "A BEÍRT SZÁMBAN"
DO
    PRINT "ÍRJ BE EGY SZÁMOT! 9<n<100"
    INPUT n
LOOP WHILE n>=100 OR N<10
LET s=0
DO
    LET n=n-10
    LET s=s+1
LOOP WHILE n>=10
PRINT s; "DB TÍZES VAN A SZÁMBAN"
PRINT n; "DB EGYES VAN A SZÁMBAN"
END
```

14. Írjunk programot, amely ötjegyű számot vizsgál helyiértékek szerint! A kapott eredményeket tároljuk egy vektorban, ahonnan a számolás befejezése után kiírhatjuk.

(Készítsünk önállóan táblázatos kiíratást is, ahol a fejlécben szerepel a tízesek, egyesek kiírás, és alatta jelennek meg a megfelelő értékek!)

1. *C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "HÁNY JEGYŰ SZÁMOT ÍRUNK FEL?" : PRINT
20 INPUT N
30 DIM S(10)
40 FOR I=1 TO N : REM VEKTOR NULLÁZÁSA
50 LET S(I)=0
```

```
60 NEXT I
70 PRINT "KÉREK EGY SZÁMOT (0<K<100000)" : PRINT
80 PRINT "KIÍROM, HOGY MELYIK HELYIÉRTÉKEN": PRINT
90 PRINT "MILYEN SZÁM ÁLL"
100 INPUT K
105 LET SZ=K
110 IF K>=100000 THEN 190
120 IF K>=10000 THEN 210
130 IF K>=1000 THEN 240
140 IF K>=100 THEN 270
145 IF K>=10 THEN 300
150 PRINT "A(Z)"; SZ; "SZÁM MÁS ALAKJA:"
160 PRINT SZ; "="; S(1); "*10000+"; S(2); "*1000+";
    S(3); "*100+"; S(4); "*10+"; K; "*1"
170 PRINT SZ; "="; S(1); "DB TÍZEZRES+"; S(2); "DB EZRES
    +"; S(3); "SZÁZAS+"; S(4); "DB TÍZES+"; K; "DB EGYES"
180 END
190 PRINT "ÖTJEGYŰ SZÁMOT KÉREK"
200 GOTO 100
210 LET S(1)=S(1)+1
220 LET K=K-10000
230 GOTO 120
240 LET S(2)=S(2)+1
250 LET K=K-1000
260 GOTO 130
270 LET S(3)=S(3)+1
280 LET K=K-100
290 GOTO 140
300 LET S(4)=S(4)+1
310 LET K=K-10
320 GOTO 145
```

Most már programunk további bővítésével tetszőleges számot elemezhetünk a helyiérték szerint. Írjuk meg programunkat a maradékos osztásnál megtanult módszer segítségével is! (Az osztó a 10 000, az 1000 stb. számok lesznek. A hányados egészrésze adja az osztók számát, a maradék pedig az új osztandó lesz.)

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "HÁNY JEGYŰ SZÁMOT ÍRUNK FEL?"
INPUT n
DIM s(10)
DO
    PRINT "KÉREK EGY SZÁMOT (0<k<100000)": PRINT
    PRINT KIÍROM, HOGY MELYIK HELYIÉRTÉKEN": PRINT
    PRINT "MILYEN SZÁM ÁLL"
    INPUT k
    LET sz=k
LOOP WHILE k<0 OR k>=100000
LET s(1)=0
DO WHILE k>=10000
```

```
    LET s(1)=s(1)+1
    LET k=k-10000
LOOP
LET s(2)=0
DO  WHILE k>=1000
    LET s(2)=s(2)+1
    LET k=k-1000
LOOP
LET s(3)=0
DO  WHILE k>=100
    LET s(3)=s(3)+1
    LET k=k-100
LOOP
LET s(4)=0
DO  WHILE k>=10
    LET s(4)=s(4)+1
    LET k=k-10
LOOP
PRINT "A(Z)"; sz; "SZÁM MÁS ALAKJA:"
PRINT sz; "="; s(1); "*10000+"; s(2); "*1000+"; s(3);
      "*100+"; s(4); "*10+"; k; "*1"
PRINT sz; "="; s(1); "DB TÍZEZRES+"; s(2); "DB EZRES +";
s(3); "DB SZÁZAS+"; s(4); "DB TÍZES+"; k; "DB EGYES"
END
```

A tízes számrendszer helyiértékének megfelelően felírt alak lehetőséget ad egy olyan program megírásához, amely segít a pénztárosoknak címletezni. Tehát kiírja, hogy ahhoz, hogy a fizetési borítékokba pontosan a megfelelő összeget tegyék, melyik pénzegységből hány darabot kell hozni a bankból.

Írjuk meg önállóan ezt a programot!

15. Függvények ábrázolásakor gyakran előfordul, hogy ábrázolás előtt táblázatot kell készítenük, mert csak az összetartozó értékek ismeretében tudjuk megrajzolni függvényünk képét.

Számolásunk meggyorsítására gyakran segítségül hívjuk zsebszámológépünket. Ilyenkor azonban a képletet mindig újra be kell gépelnünk.

Vizsgáljuk meg, hogy milyen lehetőségeink vannak függvények összetartozó értékeinek kiszámolására! Írjuk meg a hozzá tartozó programot is!

A vizsgált függvényünk legyen $x \rightarrow (x+5)/(x^2-9)$.

a) Határozzuk meg a függvény értékét, ha $x = 4$. A

```
PRINT (4+5)/(4*4-9)
```

parancs hatására a képernyőn megjelenik a keresett érték.

b) Határozzuk meg a helyettesítési értéket tetszőleges x esetén, hogy csak a változó új értékét kelljen beírni, a függvényt nem.

C-64 esetén futtatható programunk lehet:

```
10 PRINT "MEGHATÁROZOM AZ (x+5)/(x*x-9) FÜGGVÉNY"
20 PRINT : PRINT "HELYETTESÍTÉSI ÉRTÉKÉT"
30 PRINT : PRINT "MILYEN ÉRTÉKNÉL SZÁMOLJAK?"
40 INPUT x
50 LET y=(x+5)/(x*x-9)
60 PRINT "A(z)"; x; "HELYEN A FÜGGVÉNY ÉRTÉKE ="; y
```

Programunkat minden futtatáskor újra kell indítanunk, de ezen is javíthatunk.

c) Változtassuk meg programunkat, hogy a futása folytatódjon az eredmény kiírása után is!

A *b)* pontban írt programunkat kell folytatni a következőképpen:

```
65 PRINT : PRINT : PRINT
70 GOTO 10
```

(Végtelen ciklusunkat csak a RUN/STOP + RESTORE billentyűk lenyomásával állíthatjuk meg.)

Ha tudjuk, hogy hány értéket akarunk kiíratni, akkor elérhetjük, hogy a program magától megálljon.

d) Számláló segítségével állítsuk meg programunkat, ha megkaptuk a megfelelő számú függvényértéket!

C-64-en futtatható programunk lehet:

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "FÜGGVÉNYÉRTÉKET SZÁMOL" : PRINT : PRINT
20 PRINT "HÁNY HELYEN ADJAM MEG A" : PRINT
30 PRINT "HELYETTESÍTÉSI ÉRTÉKET?"
40 INPUT N
50 LET S=0
60 PRINT "A VÁLTOZÓ ÉRTÉKE?"
70 INPUT X
80 LET Y=(X+5)/(X*X-9)
90 PRINT X; "HELYEN A FÜGGVÉNY ÉRTÉKE:"; Y
100 LET S=S+1
110 IF S<N THEN 60
120 PRINT "KIÍRTAM"; N; "SZÁMÚ FÜGGVÉNYÉRTÉKET"
130 END
```

e) Számláló nélkül, a FOR-TO-NEXT utasítással ugyanezt az eredményt érjük el, ha előző programunknak megváltoztatjuk az 50., a 100. és a 110. sorát.

```
50 FOR X=1 TO N
100 üres sor marad
110 NEXT X
```

Programunk QBASIC-beli megfelelője lehet:

```
INPUT n
FOR x=1 TO n
    INPUT "A VÁLTOZÓ AKTUÁLIS ÉRTÉKE:"; x
    LET y=(x+5)/(x*x-9)
    PRINT x; "HELYEN A FÜGGVÉNY ÉRTÉKE:"; y
NEXT x
PRINT "KIÍRTAM"; n; "SZÁMÚ FÜGGVÉNY ÉRTÉKET"
END
```

Most már megáll a programunk, de még mindig problémát jelenthet, hogy x értékét minden alkalommal meg kell adnunk. Gyakran előfordul, hogy a változók értékei egymást követő természetes számok. Ilyenkor ismét egyszerűsíthetjük a felhasználó dolgát.

f) Írjunk programot, melyben az egyesével emelkedő változók értékét a számítógép automatikusan veszi figyelembe!

C-64-en futtatható programunk lehet:

```
10 PRINT "FÜGGVÉNYÉRTÉKET SZÁMOLOK" : PRINT
20 PRINT "A 7 ÉS 12 KÖZÖTT"
30 FOR X=7 TO 12
35 LET Y=(X+5)/(X*X-9)
40 PRINT X; "HELYEN AZ ÉRTÉK:"; Y
50 NEXT X
60 END
```

g) Írjunk programot, melyben a változók értéke 1-től különböző számmal változik! Előző programunknak a 30. sora változik, ahova a lépésközt is be kell írnunk.

```
30 FOR X = 7 TO 12 STEP 2
```

h) Általánosítsuk tovább programunkat! Tegyük lehetővé, hogy a program futása közben adjuk meg a ciklusutasításhoz a kezdő- és végértéket is.

C-64-en futtatható programunk lehet:

```
10 "FÜGGVÉNYÉRTÉKET SZÁMOLOK" : PRINT
20 PRINT "KÉREM; ÍRJA BE A KÖVETKEZŐ ÉRTÉKEKET!"
30 PRINT
40 INPUT "KEZDŐÉRTÉK="; K
50 INPUT "VÉGÉRTÉK="; V
60 INPUT "LÉPÉSKÖZ="; L
70 FOR X=K TO V STEP L
75 LET Y=(X+5)/(X*X-9)
80 PRINT "A(Z)"; X; "HELYEN AZ ÉRTÉK:"; Y
90 NEXT X
100 END
```

Most már programunk minden olyan esetben jól működik, amikor a nevezőben nem szerepel ismeretlen. Esetünkben a program futtatása közben előfordulhat, hogy a számítógép hibaüzenet kiírása mellett leáll, ha a törtkifejezés értékét számolja. Programunknál ez akkor fordulhat elő, ha $x=3$, vagy $x=-3$ értékkel számolunk. Ilyenkor a nevező 0, és a nullával való osztást nem értelmezzük.

i) Javítsuk tovább programunkat, hogy olyankor se álljon le futása, ha a nevező 0 lenne! (Vizsgáljuk meg, hogy a nevező mikor lenne 0, és abban az esetben, ha a megadott változóértéknél 0 lenne a helyettesítési érték, akkor kérjen új értéket a számítógép.)

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "FÜGGVÉNYÉRTÉKEK KISZÁMOLÁSA": PRINT
20 PRINT "ADOTT HATÁROK KÖZÖTT": PRINT
30 INPUT "KEZDŐÉRTÉK="; K
40 INPUT "VÉGÉRTÉK="; V
50 INPUT "LÉPÉSKÖZ="; L
60 FOR X=K TO V STEP L
70 LET N=X*X-9
80 IF N<>0 THEN 120
90 PRINT "0-VAL NEM LEHET OSZTANI"
100 PRINT "A KÖVETKEZŐ X HELYEN SZÁMOLOM AZ ÉRTÉKET"
110 GOTO 140
120 Y=(X+5)/N
130 PRINT "A(Z)"; X; "HELYEN AZ ÉRTÉK:"; Y
140 NEXT X
150 END
```

Még mindig nem vagyunk elégedettek programunkkal, mert nem tetszik a kiíratási kép. Írassuk egymás mellé a független változót és a függvény értékét! Ekkor két hosszú számoszlop jelenik meg a képernyőn a következő utasítás hatására:

```
130 PRINT X,   Y
```

(Próbáljuk ki a ; hatását is!)

Jó lenne fejlécet is készítenünk. Ha a ciklusban helyezzük el a fejléc tartalmát, akkor minden alkalommal kiírja a gépünk, ahogy ezt a

```
125 PRINT "X", "Y"
```

utasítás eredményezi. Ezt a sort töröljük, és írjuk be helyette:

```
55 PRINT "X", "Y"
```

Vizsgáljuk meg, hogy hogyan módosíthatnánk még a programot, és alkalmazzuk tetszőleges, hasonló szabályokkal adott függvény összetartozó értékeinek számolására tetszőlegesen kicsi lépésközzel is, hogy pontosabban tudjuk megrajzolni a függvény képét!

2. *Programunk QBASIC-beli megfelelője lehet:*

```
INPUT n
FOR x=1 TO n
     DO
       INPUT "A VÁLTOZÓ AKTUÁLIS ÉRTÉKE?"; x
     LOOP WHILE x=3 OR x=-3
     LET y=(x+5)/(x*x-9)
     PRINT "A(Z)"; x; "HELYEN A FÜGGVÉNY ÉRTÉKE:"; y
NEXT x
```

Általánosabb formája a következő:

```
PRINT "FÜGGVÉNYÉRTÉKEK KISZÁMOLÁSA": PRINT
PRINT "ADOTT HATÁROK KÖZÖTT": PRINT
INPUT "KEZDŐÉRTÉK=", k
INPUT "VÉGÉRTÉK=", v
INPUT "LÉPÉSKÖZ=", l
FOR x=k TO v STEP l
     DO
       PRINT "AZ x ÉRTÉKE NEM LEHET 3 VAGY -3"
       PRINT x; "MOST AZ x ÉRTÉKE; ADJA MEG A KÖVETKEZŐT!"
       INPUT "LEGYEN x ÉRTÉKE="; x
     LOOP WHILE x=3 OR x=-3
     LET y=(x+5)/(x*x-9)
     PRINT "A(Z)"; x; "HELYEN A FÜGGVÉNY ÉRTÉKE:"; y
NEXT x
END
```

Számítógépünket felhasználhatjuk trigonometrikus függvények helyettesítési értékének kiszámolására is, és segítségével függvénytáblázatot is kiírathatunk a képernyőre.

16. Készítsünk táblázatot, melyben kiírjuk 10 fokonként egy 0° és 90° közötti szög szinuszát, koszinuszát, és magát a szöget is! Természetesen nem szabad elfelejtenünk, hogy a számítógép csak radiánban megadott szöggel tud számolni, ezért biztosítanunk kell az átváltást.

(L=π*L°/180° radián)

```
10 PRINT "SIN ÉS COS SZÖGFÜGGVÉNYTÁBLÁZAT"
20 PRINT
30 PRINT "X FOK", "SIN X", "COS X"
40 PRINT
50 FOR X=0 TO 90 STEP 10
60 Y=3.1415926*X/180
70 PRINT X, SIN(Y), COS(Y)
80 NEXT X
```

Lehetőségünk van arra is, hogy növeljük vagy csökkentsük a lépésközt. Próbáljuk meg 5 fokonként, majd 1 fokonként kiíratni az értékeket! Ilyenkor az 50. sort újra kell írni az új lépésközzel. Általánosítsuk programunkat, hogy csak az adatot kelljen beírni, programsort ne! Írassunk fejlécet is, hogy tudjuk, melyik oszlopban milyen eredmény jelenik meg! (Ügyeljünk arra, hogy fokokban adjuk meg a kezdő- és a végértéket!)

17. Módosítsuk a 16. példánkat!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "SIN ÉS COS SZÖGFÜGGVÉNYTÁBLÁZAT"
20 PRINT
30 PRINT "MILYEN HATÁROK KÖZÖTT ÍRJAM KI AZ ÉRTÉKEKET?"
40 PRINT
50 PRINT "ÍRJA BE A KEZDŐÉRTÉKET!"
60 INPUT K
70 PRINT "KÉREM A VÉGÉRTÉKET!"
80 INPUT V
90 PRINT "MILYEN LÉPÉSKÖZT HASZNÁLJAK?"
100 INPUT L
110 IF K>V AND L>=0 THEN 310
120 IF K<V AND L<=0 THEN 310
130 S=0
135 PRINT "X", "SIN X", "COS X"
140 FOR X=K TO V STEP L
150 IF S<>0 THEN 190
170 PRINT
180 S=S+2
190 Y=3.1415926*X/180
200 PRINT X, SIN(Y), COS(Y)
210 S=S+1
220 IF S<20 THEN 290
230 PRINT
240 PRINT "HA KIÉRTÉKELTE, NYOMJON LE EGY BILLENTYŰT!"
250 GET A($)
260 IF A$="" THEN 250
270 PRINT "SHIFT+CLR/HOME"
280 S=0
290 NEXT X
300 END
310 PRINT "ROSSZ ADATOT ÍRT BE"
320 PRINT "ÚJ ADATOT KÉREK"
330 GOTO 30
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "SIN ÉS COS SZÖGFÜGGVÉNYTÁBLÁZAT": PRINT
PRINT "MILYEN HATÁROK KÖZÖTT ÍRJAM KI AZ ÉRTÉKET?"
PRINT
DO
  DO
    PRINT "ÍRJA BE A KEZDŐÉRTÉKET!"
    INPUT K
    PRINT "KÉREM A VÉGÉRTÉKET!"
    INPUT V
    PRINT "MILYEN LÉPÉSKÖZZEL SZÁMOLJAK?"
    INPUT L
    LOOP WHILE K>V AND L>=0
LOOP WHILE K<V AND L<=0
```

```
PRINT "X". "SINX", "COSX"
FOR X=K TO V STEP L
  LET Y=3.1415926*X/180
  PRINT X, SIN(Y), COS(Y)
NEXT X
PRINT "BEFEJEZTEM A SZÁMOLÁST"
END
```

18. Függvénytáblázatunk kiíratásakor problémát jelenthet, hogy túl sok jegyet ír ki a számítógép, és azok nem a megfelelő helyre kerülnek. Kerekítsük eredményünket négytizedes pontosságra, és tervezzük meg a kiíratást TAB(X) függvény segítségével!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "SHIFT+CLR/HOME"
20 PRINT "SZÖGFÜGGVÉNYTÁBLÁZAT:"
30 INPUT "KEZDŐÉRTÉK"; K
40 INPUT "VÉGÉRTÉK"; V
50 INPUT "LÉPÉSKÖZ"; L
60 LET PI=3.1415926
70 PRINT "SHIFT+CLR/HOME"
80 PRINT TAB (10); "X"; TAB (20); "SIN(X)"; TAB (30);
   "COS(X)"
90 FOR X=K TO V STEP L
100 LET X=PI*X/180
110 LET Y=SIN(X)
120 LET Y=INT(Y*10000+0.5)/10000
130 LET C=COS(X)
140 LET C=INT(C*10000+0.5)/10000
150 LET X=X*180/PI
160 PRINT TAB (10); X; TAB (20); Y; TAB (30); C
170 NEXT X
180 END
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "NÉGYJEGYŰ SZÖGFÜGGVÉNYTÁBLÁZATOT KÉSZÍTEK"
PRINT
DO
  DO
    PRINT "ÍRJA BE A KEZDŐÉRTÉKET!"
    INPUT K
    PRINT "KÉREM A VÉGÉRTÉKET!"
    INPUT V
    PRINT "MILYEN LÉPÉSKÖZZEL SZÁMOLJAK?"
    INPUT L
    LOOP WHILE K>V AND L>=0
LOOP WHILE K<V AND L<=0
LET PI=3.1415926
PRINT TAB(10); "X"; TAB(20); "SIN(X)"; TAB(30); "COS(X)"
FOR X=K TO V STEP L
```

```
    LET X=PI*X/180
    LET Y=SIN(X)
    LET Y=INT(Y*10000+0.5)/10000
    LET c=COS(X)
    LET c=INT(c*10000+0.5)/10000
    LET X=X*180/PI
    PRINT TAB(10); X; TAB(20); Y; TAB(30); c
NEXT X
END
```

19. Matematikai problémáink megoldása során előfordul, hogy ismerjük egy derékszögű háromszög oldalainak arányát, de nem ismerjük egyik hegyesszögét sem.

Írjunk programot, amely szögfüggvényértékből megkeresi a hozzá tartozó szögértéket!

A szögérték meghatározásához egy újabb függvényre, az arcus tangens=ATN-re van szükségünk. Ez a függvény is mutatja, hogy közvetlenül csak tangens szögfüggvény esetén kaphatjuk meg a szögeket radiánban kifejezve. (Ezek az értékek –Pí/2 és Pí/2 közé esnek, a többit nekünk kell ezekből meghatároznunk.)

```
10 PRINT "KÉREM A SZÖG TANGENSÉNEK ÉRTÉKÉT"
20 PRINT "MEGHATÁROZOM A HOZZÁ TARTOZÓ SZÖGET"
30 INPUT Y
40 LET X=ATN(Y)
60 PRINT "A SZÖG TANGENSE:"; Y: PRINT
70 PRINT "A HOZZÁTARTOZÓ SZÖG RADIÁNBAN:"; X: PRINT
80 LET X=X*180/3.14159
90 PRINT "A SZÖG FOKBAN KIFEJEZVE:"; X
100 END
```

Programunk a sorszámok elhagyásával változtatás nélkül futtatható QBASIC-ben.

Keressünk összefüggést a tangens szögfüggvény és a többi között, hogy minden esetben megkaphassuk a keresett szöget adott pozitív értékekből.

Elevenítsük fel matematikai ismereteinket:

$\operatorname{tg} x = \sin x/\cos x,$	ha	$\cos x \neq 0$
$\sin^2 x + \cos^2 x = 1,$	ezért	$\sin x = (1-\cos^2 x)^{1/2}$
		$\cos x = (1-\sin^2 x)^{1/2}$

Mindezeket felhasználva kapjuk, hogy

$\operatorname{tg} x = \sin x/(1-\sin^2 x)^{1/2}$, illetve $\operatorname{tg} x = (1-\cos^2 x)^{1/2}/\cos x$.

A definícióból adódik, hogy

$\operatorname{tg} x = 1/\operatorname{ctg} x.$

Adott sin x értéke. Hegyesszögű háromszög esetén keressük meg x értékét!

($0 \leqq \sin x \leqq 1$)

Ilyenkor külön kell vizsgálnunk a $\sin x = 1$ értékét, mert akkor tg x-nek nem lenne értelme.

1. *C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "KÉREM A SZÖG SZINUSZÁNAK ÉRTÉKÉT!"
20 PRINT "MEGHATÁROZOM A HOZZÁ TARTOZÓ SZÖGET"
30 INPUT Y
40 IF Y>1 OR Y<0 THEN 150
50 IF Y=1 THEN 180
60 LET T=(Y/SQR(1-Y*Y))
70 LET X=ATN(T)
80 PRINT "SHIFT+CLR/HOME"
90 PRINT "A SZÖG SZINUSZA:"; Y: PRINT
100 PRINT "A SZÖG TANGENSE:"; T: PRINT
110 PRINT "A SZÖG RADIÁNBAN:"; X: PRINT
120 LET X=X*180/3.14159
130 PRINT "A SZÖG FOKBAN:"; X
140 END
150 PRINT "SZINUSZ ILYEN ÉRTÉKET NEM VEHET FEL"
160 PRINT " ÚJ ÉRTÉKET KÉREK"
170 GOTO 10
180 PRINT "SIN X=1 ESETÉN"
190 PRINT "A SZÖG RADIÁNBAN: PI/2="; PI/2
200 PRINT "A SZÖG FOKBAN:90"
210 END
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
PRINT "KÉREM A SZÖG SZINUSZÁNAK ÉRTÉKÉT!"
PRINT "KIÍROM A SZÖG TANGENSÉT, ÉS"
PRINT "MEGHATÁROZOM A HOZZÁ TARTOZÓ SZÖGET"
DO
  INPUT Y
LOOP WHILE Y>1 OR Y<=0
IF Y=1 THEN PRINT "SIN X=1 ESETÉN 90 FOKOS A SZÖG": END
LET T=(Y/SQR(1-Y*Y))
LET X=ATN(T)
PRINT
PRINT "A SZÖG SZINUSZA:"; Y: PRINT
PRINT "A SZÖG TANGENSE:"; T: PRINT
PRINT "A SZÖG RADIÁNBAN MEGADVA:"; X: PRINT
LET X=X*180/3.14159
PRINT "A SZÖG FOKBAN KIFEJEZVE"; X
END
```

Módosítsuk programunkat, hogy a cosinus értékből és cotangens értékből is meg tudjuk határozni a keresett hegyesszöget. A kereséseket egy programban is összefoglalhatjuk, ha írunk az elejére egy menüt, amely választási lehetőséget ad attól függően, hogy milyen szögfüggvénnyel dolgozunk. Írjunk ilyen programot is! Kiterjeszthetjük a kereséseket előjeltől függően hegyesszögnél nagyobb szögekre is, valamint figyelembe vehetjük a periódusokat is hasonlítások segítségével. Alkalmazzuk trigonometrikus függvényeinket derékszögű háromszög hiányzó adatainak kiszámolására!

20. Írjunk programot, amely egy derékszögű háromszög egyik befogója, és a vele szemközti hegyesszög ismeretében kiszámolja a többi oldalt és szöget is!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "SHIFT+CLR/HOME"
20 PRINT "KÉREM A HÁROMSZÖG BEFOGÓJÁT"
30 INPUT A
40 IF A<=0 THEN 210
50 PRINT "KÉREM A VELE SZEMKÖZTI SZÖGET FOKBAN"
60 INPUT S
70 IF S<=0 OR S>=90 THEN 240
80 LET X=S*3.14159/180
90 LET B=A/TAN(X)
100 LET Z=90-S
110 LET C=A/SIN(X)
120 PRINT "A DERÉKSZÖGŰ HÁROMSZÖG OLDALAI:"
130 PRINT "A="; A
140 PRINT "B="; B
150 PRINT "C="; C
160 PRINT "A DERÉKSZÖGŰ HÁROMSZÖG SZÖGEI:"
170 PRINT "ALFA="; S; "FOK"
180 PRINT "BÉTA="; Z; "FOK"
190 PRINT "GAMMA=90 FOK"
200 END
210 PRINT "HÁROMSZÖG OLDALA CSAK POZITÍV LEHET; ÚJ ADATOT
    KÉREK"
230 GOTO 30
240 PRINT "A HEGYESSZÖG 0° ÉS 90° KÖZÉ ESIK"
250 PRINT "ÍRJON BE ÚJ ADATOT"
260 GOTO 60
270 END
```

A derékszögű háromszög átfogóját a két befogó ismeretében a Pitagorasz-tétel segítségével is kiszámolhattuk volna.

Ekkor írjuk be: 110 C=SQR(A*A+B*B)

Számolásunkat ellenőrizhetjük is a Pitagorasz-tétel segítségével.

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
DO
  PRINT "KÉREK EGY POZITÍV SZÁMOT"
  PRINT "EZ LESZ A HÁROMSZÖG ADOTT BEFOGÓJA"
  INPUT A
LOOP WHILE A<=0
DO
  PRINT "KÉREM A VELE SZEMKÖZTI HEGYESSZÖGET"
  PRINT "FOKBAN MEGADNI (0<=S<=90)"
  INPUT S
LOOP WHILE S<=0 OR S>=90
LET X=S*3.14159/180
LET B=A/TAN(X)
LET Z=90-S
LET C=A/SIN(X)
PRINT "A DERÉKSZÖGŰ HÁROMSZÖG OLDALAI:": PRINT
PRINT "A="; A: PRINT
PRINT "B="; B: PRINT
PRINT "C="; C: PRINT
PRINT "A DERÉKSZÖGŰ HÁROMSZÖG SZÖGEI:" : PRINT
PRINT "ALFA="; S; "FOK": PRINT
PRINT "BÉTA="; Z; "FOK" :PRINT
PRINT "GAMMA=90 FOK": PRINT
```

21. Más esetben a derékszögű háromszögnek csak az oldalai adottak. Ilyenkor is kiszámolhatjuk a hiányzó adatokat.

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "SHIFT+CLR/HOME"
20 PRINT "KÉREM A KÉT BEFOGÓT"
30 INPUT A, B
40 IF A<=0 OR B<=0 THEN 190
50 LET T=A/B
60 LET S=ATN(T)
70 LET C=SQR(A*A+B*B)
80 PRINT "A DERÉKSZÖGŰ HÁROMSZÖG OLDALAI:"
90 PRINT "BEFOGÓK: A="; A, "B="; B
100 PRINT "AZ ÁTFOGÓ: C="; C
110 PRINT "AZ EGYIK HEGYESSZÖG RADIÁNBAN:"; S
120 LET S=S*180/3.14159
130 LET Z=90-S
140 PRINT "A SZÖGEK FOKBAN KIFEJEZVE:"
150 PRINT "ALFA="; S; "°"
160 PRINT "BÉTA="; Z; "°"
170 PRINT "GAMMA=90°"
180 END
190 PRINT "A HÁROMSZÖG OLDALA CSAK POZITÍV LEHET"
200 PRINT "ÚJ ADATOT KÉREK"
210 GOTO 30
220 END
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
DO
  PRINT "KÉREM A HÁROMSZÖG KÉT BEFOGÓJÁT!"
  PRINT "a>0 ÉS b>0"
  INPUT a, b
LOOP WHILE a<=0 OR b<=0
LET t=a/b
LET s=ATN(t)
LET c=SQR(a*a+b*b)
PRINT "A DERÉKSZÖGŰ HÁROMSZÖG OLDALAI:"
PRINT "BEFOGÓK: a="; a, "b="; b
PRINT
PRINT "AZ ÁTFOGÓ: c="; c: PRINT
PRINT "AZ EGYIK HEGYESSZÖG RADIÁNBAN:"; s
LET s=s*180/3.14159
LET z=90-s
PRINT: PRINT "A SZÖGEK FOKBAN KIFEJEZVE:"
PRINT: PRINT "ALFA="; s; "FOK"
PRINT: PRINT "BÉTA="; z; "FOK"
PRINT: PRINT "GAMMA=90 FOK"
END
```

Írjunk önállóan olyan programot, mely az átfogó és a befogó ismeretében számolja ki a hiányzó adatokat!

22. A szögfüggvények ismerete a háromszögek területének kiszámításában is segít bennünket, mert nem szükséges kiszámolnunk a háromszög magasságát, hogy az ismert T=a*m/2 képletet alkalmazzuk. Alkalmazhatjuk a T=a*b*sin gamma/2 képletet.

Írjunk programot, amely kiszámolja egy háromszög területét, ha ismerjük két oldalát és az általuk bezárt szöget!

1. *C-64-en futtatható programunk lehet:*

```
5 PRINT "SHIFT+CLR/HOME"
10 PRINT "KÉREM A HÁROMSZÖG OLDALAIT"
20 INPUT A, B
30 IF A<=0 OR B<=0 THEN 120
40 PRINT "KÉREM AZ ADOTT SZÖGET FOKBAN"
50 INPUT S
60 IF S>=180 OR S<=0 THEN 150
70 LET S=S*3.1415926/180
80 LET T=A*B*SIN(S)/2
90 PRINT "A HÁROMSZÖG TERÜLETE:"
100 PRINT T; "TERÜLETEGYSÉG"
110 END
120 PRINT "A HÁROMSZÖG OLDALA CSAK POZITÍV LEHET"
130 PRINT "ÚJ ADATOT KÉREK"
140 GOTO 10
150 PRINT "A SZÖG CSAK 0° ÉS 180° KÖZÖTTI"
160 PRINT "ÉRTÉK LEHET"
170 PRINT "ÚJ ADATOT KÉREK"
180 GOTO 50
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
CLS
DO
  PRINT "KÉREM A HÁROMSZÖG OLDALAINAK HOSSZÁT!"
  PRINT "a>0 ÉS b>0"
  INPUT a, b
LOOP WHILE a<=0 OR b<=0
DO
  PRINT "KÉREM AZ ADOTT SZÖGET FOKBAN!"
  PRINT "0<SZÖG<180"
  INPUT s
LOOP WHILE s>=180 OR s<=0
LET s=s*3.1415926/180
LET t=a*b*SIN(s)/2
PRINT "A HÁROMSZÖG TERÜLETE:": PRINT
PRINT t; "TERÜLETEGYSÉG"
END
```

Programunk lehetőséget biztosít arra, hogy kiszámoljuk a szabályos sokszögek területét. Ilyenkor a sokszög középpontjából egybevágó háromszögekre bontjuk a sokszöget. A sokszög köré írható kör sugarai adják a háromszögek szárait, az általuk bezárt szög S = 360/N. A sokszög területe a háromszög területének N-szerese lesz.

```
T=N/2*R*R*SIN(360/N)
```

Írjuk meg a leírtak alapján a folyamatot megvalósító programunkat!

23. A beépített függvények között találkoztunk a logaritmusfüggvénnyel is, de alapszáma nem a megszokott 10, hanem *e* = 2,1828 volt. Ezért a program megírása során ezt is figyelembe kell venni.

Írjunk programot, amely tetszőleges szám tízes alapú logaritmusát kiírja a képernyőre!

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "SHIFT+CLR/HOME"
20 PRINT "KÉREK EGY SZÁMOT!"
30 PRINT "KIÍROM A TÍZES ALAPÚ LOGARITMUSÁT"
40 INPUT N
50 IF N<=0 THEN 90
60 LET X=LOG(N)/LOG(10)
70 PRINT N; "SZÁM TÍZES ALAPÚ LOGARITMUSA=LG"; N; "=";X
80 END
90 PRINT "CSAK POZITÍV SZÁM LOGARITMUSÁT ISMEREM"
100 PRINT "ÚJ ADATOT KÉREK"
110 GOTO 20
```

2. *Programunk QBASIC-beli megfelelője lehet:*

```
PRINT "SZÁMOK TÍZES ALAPÚ LOGARITMUSÁT SZÁMOLOM"
DO
  PRINT "KÉREK EGY POZITÍV SZÁMOT"
  INPUT n
LOOP WHILE n<=0
LET x=LOG(n)/LOG(10)
PRINT "A(z)"; n; "SZÁM TÍZES ALAPÚ LOGARITMUSA;"
PRINT "LG"; n; "="; x
END
```

24. Másodfokú egyenlet megoldása közben előfordul, hogy nem vagyunk kíváncsiak pontosan a gyökökre, csupán arra, hogy hány van belőlük (például egyenes és kör kölcsönös helyzetének vizsgálatakor).

Írjunk programot, amely egy másodfokú egyenlet együtthatóinak ismeretében kiírja a gyökök számát! (Az egyenlet általános alakja: $Ax^2 + Bx + C = 0$.)

1. *C-64-en futtatható programunk lehet:*

```
10 PRINT "A DISZKRIMINÁNS SEGÍTSÉGÉVEL"
20 PRINT "KIÍROM A MÁSODFOKÚ EGYENLET"
30 PRINT "GYÖKEINEK SZÁMÁT": PRINT
40 PRINT "ABC SORRENDBEN KÉREM AZ EGYÜTTHATÓKAT"
50 INPUT A, B, C
60 LET D=B*B-4*A*C
70 IF D<0 THEN PRINT "NINCS VALÓS GYÖK": END
80 IF D=0 THEN PRINT "EGY VALÓS GYÖK VAN": END
90 PRINT "KÉT VALÓS GYÖK VAN"
```

2. *Programunk QBASIC-ben változtatás nélkül futtatható.*

VII. A programok ellenőrzése

A) A program helyességének vizsgálata

Programírás közben észrevettük, hogy milyen sok alkalommal hibázhatunk. Elegendő egyes helyett l betűt, vagy 0 helyett O betűt írni, esetleg lemarad egy zárójel, és máris nem működik programunk, kereshetjük a hibát.

Az előforduló hibákat két nagy csoportra szokták osztani.

Formai (szintaktikai) hibák

A programot az adott nyelv formai előírásainak megfelelően kell megírni, mert így biztosíthatjuk az információk egyértelműségét, így érti meg számítógépünk. Hibás sor leírása után közvetlenül is jelezhet számítógépünk, de a program futása közben feltétlenül kiírja a hiba jellegét.

Például hibás gépelés: `INPUT A;B`

hiányzik az a sor, amelyre hivatkozunk a GOTO utasítással.

Tartalmi (szemantikai) hibák

Nemcsak formailag, hanem *logikailag* is ellenőriznünk kell programunkat megfelelő adatokkal, hogy eredményként azt kaptuk-e, amit szeretnénk.

Programunkra csak akkor mondhatjuk, hogy helyesen *működik,* ha

— a program *minden érvényes* bemeneti adatra lefut (túl kicsi, túl nagy), és helyes eredményt ad;
— *nem fogad el rossz* bemeneti adatokat (pl. kör sugara nem lehet negatív érték).

Ezek az elvárások teszik szükségessé, hogy a programot a *hibás adatbevitel ellen* is védjük. Előfordulhat, hogy a program felhasználója nem érti meg az útmutatásainkat, vagy véletlenül hibás adatot ír be. Ha ilyenkor téves eredményt kapunk vagy rosszul fut a programunk, akkor a *programot kell javítanunk.* Be kell építenünk olyan utasításokat, amelyek megakadályozzák hibás adatbevitel esetén a program futását, amennyiben ez lehetséges.

(Elérhetjük, hogy negatív sugár megadása esetén új adatot kér számítógépünk, de nem tudunk tenni semmit, ha 5,23 helyett 5,238-et vagy 4,23-ot írunk be adatként.)

A programozó, miután elkészítette programját, kipróbálja, ellenőrzi, megismeri tulajdonságait, vagyis *teszteli* azt. Ehhez azonban jól kell megválasztanunk adatainkat, hogy lehetőleg minden olyan értékre futtassuk a programot, amely hibához vezethet.

A tesztelés során felmerülő *hibákat azonnal* ki kell *javítanunk*, majd az egész *folyamatot újra kell* kezdenünk, nehogy javítás közben másutt jelentkezzék hiba.

Most már valószínű, hogy jól fut programunk, de ezt még több gyakorlati feladat esetén ellenőrizni kell, hogy felhasználói környezetben is teljesíti-e az elvárásokat. Ilyenkor történik a *program érvényesítése.* Hosszabb futási idő után, amikor a tapasztalataink alapján módosítottuk a programot, ezáltal végleges formába öntöttük, mondhatjuk, hogy készen vagyunk, *hitelesítjük* programunkat.

B) A program tesztelésének módszerei

A program helyességének vizsgálatakor láthattuk, hogy programunk még nincs készen akkor, ha a kódolást befejezzük. Keresnünk kell a hibalehetőségeket, ellenőriznünk, tesztelnünk kell azt. Kétféle módszert követhetünk eljárásunk során.

Statikus módszer

Ha ezt a módszert alkalmazzuk, akkor a programot *nem futtatjuk,* hanem a program szövegét vizsgáljuk. A szintaktikai hibákat viszonylag könnyen felfedezhetjük, javíthatjuk, de a tartalmi hibákat csak az algoritmus logikájának végiggondolásával találhatjuk meg. Jól segíthet bennünket például egy *táblázat,* amelyet a *változókról* készítünk, melyben leírjuk, hogy melyik változónak mi a neve, a jelentése és a típusa, valamint milyen sorszámú utasítások esetén kap új értéket. Mindez lehetővé teszi a típuskeveredés kizárását vagy azt, hogy egy változót más-más dolog jelölésére alkalmazzunk.

Dinamikus módszer

Ezt az eljárást alkalmazva *futtatjuk* a programot, melyet a *szintaktikai hiba* észlelése azonnal megállít. A gép hibaüzenete segít bennünket a hiba megkeresésében és javításában.

A logikai hibákat a kiadott eredmények alapján észlelhetjük: nem az előre kiszámolt eredmény jelenik meg a képernyőn.

Arra, hogy minden adatot kipróbáljunk, nincs lehetőségünk, ezért nagyon fontos a bemenő adatok jó kiválasztása.

Előfordul, hogy nem ismerjük a program szerkezetét, ilyenkor csak kívülállóként vizsgálódunk a következőképpen:

— Ha a program két bemenő adatra ugyanúgy működik, akkor elég az egyikre kipróbálni, mert egyik helyes futása esetén valószínű, hogy a másik adattal is helyes eredményt kapunk. Az adatokat ekvivalenciaosztályokba sorolhatjuk az előző feltételeknek megfelelően.
— Ha ugyanazon osztály elemeit nem azonos módon kezeli a számítógép, akkor tovább kell bontani ezt az osztályt.
— Ekvivalenciaosztályokat *bemenő és kimenő adatokra is* készítenünk kell, hogy lefedjük az érvényes és érvénytelen adatokat.

Példa

Készítsünk bemeneti adatokat tartalmazó ekvivalenciaosztályokat az $Ax^2 - Bx + C = 0$ alakú, legfeljebb másodfokú egyenlet megoldásához *(30. ábra)!*

1.	2.	3.	4.	5.	6.
A = 0	A = 0	A = 0	A ≠ 0	A ≠ 0	A ≠ 0
B = 0	B = 0	B ≠ 0	B = 0	B ≠ 0	B ≠ 0
C = 0	C ≠ 0	C ≠ 0	C = 0	C ≠ 0	C = 0

30. ábra

Ezekhez a bemenő adatokhoz a következő kimenő adatok tartozhatnak:

— két valós gyök van,
— egy valós gyök van,
— nincs valós gyök.

Most már hozzákezdhetünk a tesztesetek összeállításához, ahol vegyük figyelembe a következő elvárásokat:

— Készítsünk külön tesztesetet *minden* érvénytelen ekvivalenciaosztályra!
— Érvényes ekvivalenciaosztályok esetén pedig olyan eseteket kell választanunk, hogy lehetőleg lefedjük az ekvivalenciaosztályokat (minél többre kipróbáljuk).
— Egyeztessük a bemenő és a kimenő adatok osztályait!
— Feltétlenül szerepeltessük az ekvivalenciaosztály határain lévő adatokat!

A másik esetben *ismerjük* a program kódját, ismerjük annak *logikai menetét*, így erre alapozzuk tesztelésünket.

Minden elképzelhető adatot itt sem tudunk kipróbálni hosszabb programok esetén, ezért itt is jól meg kell terveznünk a futtatásokat. A tesztadatok kiválasztásakor a következő elveket tartsuk szem előtt:

— A program minden utasítását *legalább egyszer* érintse valamelyik futtatás (utasítások egyszeri lefedésének elve).
— Az elágazások *minden* igaz és hamis *ágát* is legalább egyszer be kell járni (döntések lefedésének elve).
— A döntésekben szereplő *minden feltételt* legalább egyszer hamis, illetve igaz értékekkel is le kell futtatnunk, és a határeseteket is figyelembe kell venni (feltétellefedés elve).

Teszteseteinket mindhárom elv figyelembevételével kell kialakítanunk.

Tesztelést nemcsak a program elkészülte után, hanem *programírás közben* is végezhetünk. Már ekkor feltérképezhetjük a bemenő és kimenő adatok ekvivalenciosztályait, gondolnunk kell a döntéslefedés és a feltétellefedés elvére is. Folyamatosan készíthetjük a változótáblát is.

A tesztelést nemcsak a program írójának, hanem idegennek, kívülállónak is érdemes elvégezni, mert esetleg kivédhetjük azt, hogy ugyanolyan hibába essünk.

Bármikor és bármilyen módon is tesztelünk, gondoljuk végig a tesztesetek eredményét, és hasznosítsuk belőle a lehető legtöbbet, mert ezzel a próbák számát csökkenthetjük!

Kérdések és feladatok

1. Hogyan csoportosíthatjuk a programírás közben előforduló hibákat? Mi a különbség a csoportok között?
2. Mikor mondhatjuk, hogy programunk helyesen működik?
3. Hogyan védhetjük programjainkat a hibás bemeneti értékek ellen? Miért van szükség erre?
4. Miért kell körültekintően elvégeznünk programunk tesztelését?
5. Milyen tesztelési módszereket alkalmazhatunk? Mi a lényeges különbség az eljárások között?
6. Hogyan kell megválasztanunk a tesztelés adatait?
7. Milyen eljárást követünk olyan program tesztelésekor, amikor
   a) nem ismerjük a program szerkezetét;
   b) ismerjük a program logikai menetét?

C) A strukturált programozásról

Tanulásunk során megismerkedtünk néhány algoritmust leíró módszerrel, és az algoritmus alapján elkészítettük BASIC, illetve QBASIC nyelvű programunkat.

A két program formája között is észrevehettünk lényeges különbséget. A BASIC nyelv alkalmazásakor az utasítások előtt mindig állt egy szám, amelyre bármikor hivatkozhattunk a programban. Legtöbbször a GOTO utasítás alkalmazásakor használtuk ki ezt a lehetőséget.

Óvakodjunk a feltétel nélküli vezérlésátadás alkalmazásától, próbáljunk más megoldást keresni, mert könnyen áttekinthetetlenné, zavarossá, esetleg hibássá teheti programunkat. Ha mégis alkalmazzuk a GOTO utasítást, akkor ügyeljünk arra, hogy

— ne ugorjunk ki egy szerkezeti elem belsejéből. Ha nem akarjuk folytatni, akkor a kilépési sorra ugorjunk, és úgy lépjünk ki!
— ne ugorjunk be egy programozási egység belsejébe. Csak a belépési pontra álljunk, és úgy lépjünk be!

A QBASIC nyelvben nem kötelező sorszámot alkalmazni, de itt is van GOTO parancs. Ebben az esetben az adott programsorra egy úgynevezett címkével hivatkozhatunk.

QBASIC-beli programozáskor programjaink áttekinthetőbbek lehetnek, mivel jól tagolhatjuk őket, felbonthatjuk programjainkat kisebb logikai egységekre, alprogramokra. Ezeknek az eljárásoknak csak egy bemenete és egy kimenete van.

A számítógépek programozásának ezt a módszerét, ahol a programot apróbb logikai egységekre bontjuk fel, strukturált programozásnak nevezzük. Különválaszthatjuk az egyes részprogramokra vonatkozó, és a program egészére érvényes szimbólumokat. Ez a fajta programozás lehetővé teszi, hogy egyes programrészeket különböző programozók írjanak, illetve egyes programrészeket más programokban is felhasználjunk.

A könnyebb olvashatóság érdekében érdemes kialakítani egy olyan rendszert, ahol bekezdések vannak, az egyes alprogramoknak eleje és vége van, miközben a számítógép műveleteket végez. Érdemes az összetartozó egységeket különböző bekezdésekkel szemlélteni.

Az elmondottakat jól szemlélteti a ciklusutasítás.

```
Ciklusfeltétel
    utasítások
Ciklus vége
```

Példa

Készítsük el az első 51 természetes szám összegére vonatkozó algoritmust!

```
Program
    S=0
    Ciklus I=1-től 51-ig
            S=S+I
    Ciklus vége
    Ki S
Program vége
```

Programja lehet:

```
LET S=0
FOR I=1 TO 51
    S=S+I
NEXT I
PRINT S
```

QBASIC-beli programjainknál már megfigyelhettük, hogy törekedtünk erre a strukturált leírásra. Érdemes megismerkednünk vele, mert könnyen olvasható, áttekinthető, és könnyen átírható a programunk más programnyelvre.

D) Vírusok

A nyolcvanas évek közepén figyeltek fel a felhasználók először néhány megmagyarázhatatlan jelenségre. Például a betűk lepotyogtak a képernyő legalsó sorába, vagy átsétált egy katicabogár egyik oldalról a másikra. Más esetekben teljesen megsemmisültek egyes adatállományok, vagy eltűnt a feldolgozását segítő program, összekeveredett a tartalomjegyzék stb. Mindezt az úgynevezett számítógépes vírus okozta. Ez olyan „kórokozó" program vagy programrészlet, amely reprodukálja magát. Ez azt jelenti, hogy önmaga másolásával szaporodásra képes, így könnyen sokszorozódik. Terjedéséhez „élő" környezet, vagyis futó program szükséges.

Van olyan, amely már a program betöltésekor támad, megfertőzi a betöltő szektort és a rejtett állományokat. Mások valamilyen végrehajtható program futásakor aktivizálódnak. Megfertőzik a hordozóprogramot vagy a hozzá tartozó részeket.

A sok kellemetlenség elkerülése érdekében nagyon fontos a vírusfertőzés megelőzése. Tudjuk, hogy a vírusok programok, ezért valakiknek meg kellett írni őket. Egyes vélemények szerint a szoftvergyártók találták ki őket, hogy megelőzzék az illegális másolást. Nagyon fontos tehát, hogy idegen, bizonytalan eredetű programot, lemezt ne használjunk.

Az ismert vírusok ellen védelmet nyújtanak a vírusfelismerő- és vírusmegelőző programok, amelyek jelzik a felhasználó számára a vírus jelenlétét, felhívják a figyelmet a kiirtásukra, és arra is alkalmasak lehetnek, hogy megakadályozzák a vírus aktivizálását.

Ha nem sikerül a megelőzés, és vírusos lett a rendszer, akkor segítségünkre állnak a vírusirtó programok. Ezek megtalálják a vírust, és kitörlik az állományból. A vírus által okozott kárt azonban magunknak kell megoldanunk.

Feladatok

1. Melyik parancs és melyik utasítás a következő programsorok közül? Ellenőrizzük döntésünket a számítógépünkkel!

a) `A=7*3 : PRINT A`
b) `NEW`
c) `80 LIST`
d) `LIST 80`
e) `120 PRINT 7*3`
f) `PRINT "SHIFT+CLR/HOME"`
g) `LET A=3*B`
h) `30 LET C=7+4`

2. Milyen hibák vannak a következő programban? Javítsuk ki őket, és futtassuk a programokat!

a)
```
10 INPUT A; B;
20 INPUT "ÍRJ B E KÉT SZÁMOT!"
30 A=B<2
40 B=3A
50 PRINT A; B
```
b)
```
10 A=12
20 36=B
30 PRINT A*B
```
c)
```
10 INPUT "KÉREM AZ ADATOKAT" A; B
20 PRINT "A*B="A.B
```

3. Mi jelenik meg a képernyőn a következő programok futtatásakor? Magyarázzuk meg az eredményt!

a)
```
10 LET A=54
20 LET B=87
30 LET B=A
40 PRINT B
```

b)
```
10 A=54
20 B=87
30 A=B
40 PRINT A
```

4. Írjuk fel a következő matematikai kifejezések BASIC megfelelőjét!

a) $x = \dfrac{75\,(a-4b)+3c}{2a+b\cdot d}$;

b) $y = \dfrac{(5a-2b)^2}{3a\cdot 4b}$;

c) $v = \dfrac{3a+2b^2\cdot 4(3a-b)}{a+8b}$;

d) $z = \dfrac{4a-b+5c^3}{4ab^2}$.

5. Írjuk fel a következő kifejezések BASIC megfelelőjét, és határozzuk meg a kifejezések értékét, ha

a) $x = \dfrac{a+b}{c+d}$;

b) $y = a + \dfrac{7b}{2c}$;

c) $v = 4a + \dfrac{5b}{2c+d}$;

d) $z = \dfrac{2a-3b}{c^3-4d}$.

6. Mi jelenik meg a képernyőn a következő program futása után, és mi a listázás után?

```
10 A=24
20 B=32
30 PRINT A
40 C=10
50 PRINT A, C
60 PRINT B
```

7. Írjunk programot, amely a négyzet oldalának ismeretében kiszámolja, és a képernyőre mindenki számára érthetően kiírja a négyzet területét és kerületét, ha

a) előre adott a négyzet oldalának hossza,
b) futás közben kéri a gép a négyzet oldalhosszát.

8. Írjunk programot, amely futás közben bekér egy tetszőleges számot, megvizsgálja annak előjelét, és kiírja a képernyőre, hogy pozitív, negatív számról vagy a 0-ról van-e szó.

9. Írjunk beszélgető programot, ahol a számítógép vásárlás előtt felteszi a kérdéseket: Mit akarsz vásárolni a boltban? Mennyi az egységáruk? Hányat akarsz venni az egyes árukból? Mennyi pénzed van?

A képernyőn jelenjék meg: Mennyit fizetünk az egyes termékekért? Mennyit fizetünk összesen? Mennyi pénzt kapunk vissza, vagy mennyivel maradunk adósak (esetleg semmi pénzünk sem marad).
(Oldjuk meg a feladatot először egy áru esetén!)

10. Készítsünk programot, amely megkérdezi tálaláskor, hogy
Hány tányérunk van?
Hányan akarnak ebédelni?
Hány kanalunk, villánk, késünk van?
Döntse el a számítógép, hogy elég-e a tányér és az evőeszköz az ebéd elfogyasztásához! Mennyi lesz felesleg, vagy újabbakat kell-e szerezni stb.

11. Írjunk programot, amely három szakasz hosszának ismeretében eldönti, hogy szerkeszthető-e azokból háromszög, és kiírja a döntését a képernyőre!

12. Írjunk programot, amely a program futása közben beadott három szakasz hosszának ismeretében eldönti, hogy lehet-e derékszögű, egyenlő szárú vagy egyenlő oldalú háromszöget alkotni belőlük!

13. Írjunk programot, amely kiszámolja az $x^2 - 3x + 4$ függvény értékeit!

a) tetszőlegesen megadott x értékek esetén;
b) ha x értéke szabályos lépésközönként változik!
(Számoljunk törtértékű lépésközökkel is, hogy pontosan tudjunk rajzot készíteni a görbéről!)
Készítsünk táblázatot a kiíratáshoz!

14. Írjunk programot, amely a következőket jeleníti meg a képernyőn!

A	5*A	A*A	A*A*A
3	15	9	27
6	30	36	216
9	45	81	729

15. Futás közben megadott adatok alapján határozzuk meg egy

a) kocka felszínét és térfogatát,
b) négyzet alapú hasáb felszínét és térfogatát,
c) téglatest felszínét és térfogatát!
Készítsünk megoldásunk előtt blokkdiagramot is!

16. Határozzuk meg program segítségével két befogó ismeretében a derékszögű háromszög területét és kerületét!
Készítsünk folyamatábrát!

17. Írjunk programot, amely futás közben bekéri az adatokat és

a) 3 számot kiír növekvő sorrendben,
b) 3 nevet kiír ábécé sorrendben!

18. Készítsünk programot, amely százalékszámításkor két adat ismeretében kiszámolja a harmadikat!

a) Az a alapnak kiszámolja s százalékát;
b) s százalékhoz tartozó e érték alapján kiszámolja az a alapot.
c) Meghatározza, hogy hány százaléka az e érték az a alapnak.

19. Készítsünk valutabeváltó programot!

a) Számoljuk ki, hogy a vásárolt valutáért hány forintot fizetünk!
b) Számoljuk ki, hogy meglévő pénzünkért mennyi valutát kaphatunk különböző pénznemekben!

20. Készítsünk programot, amely szótárként működik. Például, a program adja meg a BASIC parancsszavak magyar jelentését!

21. Írjunk programot, amely csillagkarakterek segítségével megjelenít egy B betűt!

22. A véletlenszám-generátorral állítsuk elő rendszámok számjegyeit! (Nálunk háromjegyű számokat tartalmaznak a rendszámok.)

23. Készítsünk programot, amely 1 és 100 közötti véletlenszámokat állít elő! Ki kell találnunk azt a számot. Írja ki a számítógép, ha nagyobbra vagy kisebbre gondoltunk!

24. Készítsünk programot, amely kiír egy kérdést és rá három választ! El kell döntenünk, hogy melyik a helyes. A számítógép a beírt válaszunk alapján kiírja, hogy helyesen válaszoltunk-e.

25. Írjunk programot, amely egy számot törzstényezőire bont, és a tömbben tárolt prímszámokat a végén kiírja!

26. Készítsünk programot, amely kiszámolja a naponkénti átlaghőmérsékletet 1 héten keresztül, ha mindennap 4-szer mérjük meg a hőmérsékletet!

27. Írjunk programot, amely egy tízes számrendszerben adott számot kettes számrendszerben jelenít meg! (A maradékokat őrizzük egy vektorban!)

28. Írjunk programot, amely n db áru ára közül kikeresi a legolcsóbbat és a legdrágábbat! (Az árakat olvassuk be egy vektorba!)

29. Készítsünk programot, amivel meghatározhatjuk b számú élelmiszerbolt napi és heti bevételének összegét! Számoljuk ki az élelmiszerboltok bevételeit hosszabb időn keresztül! Nézzük meg, hogy mely napokon a legnagyobb, illetve a legkisebb a forgalom!

Függelék

1. A számítógépek hibaüzenetei

A hibák észlelése

A programsor beírásakor nem mindig vesszük észre a hibát. Előfordul, hogy a kész program átnézésekor sem tűnik fel, hogy valahol eltévesztettünk valamit. A program futtatásakor azonban a számítógép nem lép tovább, ha végre nem hajtható parancsot vagy utasítást talál. Ekkor a számítógép a képernyőre hibaüzeneteket ír számunkra ERROR (hiba) felirattal, majd megadja a hiba helyét és típusát is.

Nemcsak a parancsokban lehetnek hibák, hanem a perifériák is lehetnek rosszak, ezért az írás és olvasás során is felléphetnek problémák. Ezeket is jelzik számunkra egyes számítógéptípusok a megjelenő üzenetekben.

A COMMODORE-64 hibaüzenetei

A következőkben megismerkedünk a C-64 hibaüzeneteivel, amelyek tartalmazzák a hiba okát is. Itt részletesebben találkozunk az okok leírásával is.

Hiba jelzése	*Hiba oka*
DEVICE NOT PRESENT	Nem jelen lévő készülék. Olyan bemeneti vagy kimeneti készülékre hivatkoztunk, amely nem áll rendelkezésre.
DEVISION BY ZERO	Nullával való osztás, ez nem megengedett.
EXTRA INGORED	Felesleges, visszautasítva. Az INPUT utasításra válaszként túl sok adatot gépeltünk be. Csak a szükségeseket veszi figyelembe.
FILE NOT FOUND	Nem találta az adatállományt.
FILE NOT OPEN	Nincs nyitva az adatállomány.
FILE OPEN	Adatállomány nyitva. Nem kell még egyszer nyitni.
ILLEGAL DIRECT	Illegális parancs. Például az INPUT utasítást parancsként használtuk.

LOAD	Betöltés. Valamilyen probléma van a szalagon lévő programmal.
NEXT WITHOUT FOR	FOR nélküli NEXT-et használtunk, vagy az alkalmazott változók nem egyeznek.
OVER FLOW	Túlcsordulás. A megengedett legnagyobb számnál 1.7014 188 4 E + 38 nagyobbat használtunk.
REDO FROM START	Bemeneti adathiba. Például az INPUT utasításban karakteradatot gépeltünk, és a gép számot vár. Helyes érték begépelése esetén a program magától folytatódik.
RETURN WITHOUT GOSUB	GOSUB nélküli RETURN utasítást használtunk.
STRING TOO LONG	Túl hosszú szöveg. 255 karakternél hosszabb.
? SYNTAX ERROR	Szintaktikus hiba. Olyan utasítást írtunk, amelyet a C-64 nem ismer. Például rosszul gépeltünk utasítást.
TYPE MISMATCH	Típus nem megfelelés. Például szöveg helyett számot használtunk.
UNDEF'D STATEMENT	Definiálatlan utasítás. Például olyan sorszámon akarjuk folytatni a futást, amelyet előzőleg nem adtunk meg.
VERIFY	Összehasonlítás. A szalagon vagy lemezen lévő program nem egyezik a tárban lévővel.
BAD DATA	Hibás adat. Például a számítógép egy megnyitott adatállományból numerikus adatot várt, és szöveg típusú adatot kapott.
BAD SUBSCRIPT	Hibás index. A program egy olyan tömbelemre hivatkozik, amely kívül esik a DIM-ben megadott határon.

A hibaüzenetek teljes listáját megtalálhatjuk a szakkönyvekben.

Feladat

Írjunk hibás programsorokat, hogy a számítógép a várt hibajelzést adja!

A DOS leggyakrabban megjelenő hibaüzenetei

A perifériákkal kapcsolatos hibaüzenetek:

1. Hiba adat olvasásakor:

   `<TYPE> ERROR READING <DEVICE>`
   `ABORT, RETRY, IGNORE?`
   <típus> hiba az olvasás során <egység>
   befejezni, ismételni, átlépni ?

2. Hiba adat írásakor:

   `<TYPE> ERROR WRITING <DEVICE>`
   `ABORT, RETRY, IGNORE?`
   <típus> hiba az írás során <egység>
   befejezni, ismételni, átlépni?

   A hibás egység neve jelenik meg az <egység> helyén.
   Például: A:, B:, C:, D:, PRN:, LTP1:, LTP2:, CON:, COM1:,

A <típus> jelentheti a következő hibákat:

BAD COMMAND	Hibás parancs a perifériának. A programot csak a parancs kijavítása után lehet újra futtatni.
DATA	Adathiba. A lemezen megsérült rész található. Formattáljuk újra!
DISK	Lemezhiba.
DRIVE NOT READY	A lemezegység nincs készenléti állapotban. Előfordulhat, hogy nincs a lemezegységben lemez, vagy a bent lévő lemez nincs megfelelően formázva, esetleg nincs bezárva a lemezegység ajtaja, vagy egyáltalán nem kapcsoltuk be az egységet. Vegyük ki a lemezt, és tegyük be újra!
GENERAL FAILURE	Ismeretlen hiba. A rendszer nem tudta azonosítani a hibát. Legtöbbször formattálatlan lemez esetén jelenik meg ez a jelzés.
NO PAPER	A papír elfogyott. A nyomtatóban nincs papír vagy a nyomtató nincs bekapcsolva.
NON DOS DISK	Nem DOS lemez. A lemezen nincs operációs rendszer.
NOT READY	Nem kész. A berendezés nem működőképes. Előfordulhat, hogy a nyomtató még nyomtat, vagy a

	lemezegységben nem formázott lemez van, vagy nincs bezárva a lemezegység ajtaja.
READ FAULT	Olvasási hiba. A DOS rosszul olvasta az adatokat.
SECTOR NOT FOUND	A szektor nem található meg. Rossz a lemez, hibás területek vannak rajta.
WRITE PROTECT	Írásvédelmi hiba. Írásvédett lemezre akartunk írni. Ha írni akarunk a lemezre, akkor távolítsuk el a lemezen lévő írásvédelmi bevágásról a ragasztót!
WRITE FAULT	Íráshiba. A DOS nem tudta az adatokat az adathordozóra írni.

Ha a megjelenő üzenetek a lemezmeghajtóra vonatkoznak, akkor a lemezt nem szabad lecserélni. A rendszer addig várakozik, amíg a felhasználó nem adja a következő válaszok egyikét.

A — Abort, azaz programfutást megszakítani. A rendszer kilép a programból, és visszatér a DOS-hoz.
R — Retry, azaz újra megpróbálja a műveletet elvégezni. A hiba okának kijavítása után a rendszer ismét elvégzi a hibát okozó műveletet.
I — Ignore, azaz figyelmen kívül hagyja a hibát. A rendszer átlépi a hibát, és folytatja a program végrehajtását. Ilyen esetben fontos adatok veszhetnek el.

Ha a hibaüzenet olyankor jelenik meg, amikor nincs lemez az A: meghajtóban, és nem is akarjuk, hogy az aktuális meghajtó az A: legyen, akkor célszerű az I választ adni. Az ezután megjelenő üzenet után megjelenő prompt mögé írjuk be a használni kívánt meghajtó betűjelét.

A rendszer működése közben megjelenő üzenetek:

Acces denied	Az elérés tilos. Érvénytelen fájlműveletet akartunk elvégezni. Például csak olvasható állományba írni akartunk.
Are your sue (Y/N) ?	Biztos Ön benne? Ha egy DEL paranccsal törölni akarjuk egy könyvtár összes állományát, akkor a számítógép ezzel figyelmeztet bennünket, nehogy véletlenül töröljünk mindent. Y beütése és ENTER lenyomása után végrehajtódik az eredeti parancs.

Attempted write-protect violation	Kísérlet történt az írásvédelem megsértésére. Távolítsuk el az írásvédelmi bevágásról a ragasztást, ha írni akarunk rá. Az éppen formázás alatt álló lemezre sem lehet írni.
Bad command or file name	Helytelen parancs vagy állománynév. Érvénytelen a beírt DOS parancs, írjuk be újra! Esetleg a fájl neve volt rosszul beírva, vagy nincsenek a lemezen a megadott állományok. Ügyeljünk arra, hogy a külső parancsok megadása esetén megadjuk az útvonalat!
Bad or missing command interpreter	Helytelen vagy hiányos parancsértelmező. Az a lemez, ahonnan betöltődött a DOS, nem tartalmazza a COMMAND.COM állományt. Másoljuk át egy háttérlemezről ismételten a COMMAND.COM állományt.
Cannot find system files	A rendszerfájlok nem találhatók, hiányoznak a lemezről. Másoljuk rá.
Cannot load COMMAND, system halted	A COMMAND.COM nem betölthető, rendszer leállítva. Előfordulhat, hogy megsérült a lemez. A rendszert újból el kell indítani.
Current drive in no longer valid	Az aktuális lemezegység hibás. A használni kívánt lemezegység helyett másikat kell használnunk. Írjuk be ennek a jelét!
Disk boot failure	Boot hiba. A DOS betöltésekor hiba történt. Formázzuk újra a lemezt!
Disk full-write not completed	A lemez megtelt — a kiírás nem fejeződött be.
Disk not compatible	A lemez nem kompatíbilis.
Disk unsuitable for system disk	A lemez rendszerlemezként nem használható. Nem tudjuk formázni a lemezt, így a DOS rendszert nem tudjuk rajta tárolni, csak adatokat.
Drive letter must be specified	Az egység betűjelét meg kell adni. Ha alkalmazzuk a FORMAT parancsot, akkor szóköz után a meghajtó azonosítóját is meg kell adni. *Például* FORMAT A:
Duplicate filename or file not found	Duplikált vagy nem található állomány. Egy állománynak olyan nevet akartunk adni, ami már van a könyvtárban, vagy az az adatállomány, amelyet szeretnénk átnevezni, nincs a lemezünkön.

Enter current Volume Label for Drive d (Press ENTER for none)	Kérem a D: egység kötetazonosítóját (nyomja le az ENTER gombot, ha nincs). Írjuk be a D fixlemezegység nevét!
Entry error	Parancsmegadási hiba. Javítsuk ki az utolsó parancsba beírt szintaktikai hibát!
Error in .EXE file	Hiba van az EXE állományban.
Error loading operating system	Hiba történt az operációs rendszer betöltésének folyamán. Rosszul töltöttük be a DOS-t. Tartaléklemezről töltsük újra az operációs rendszert!
Error writing to device	Hiba történt az egységre íráskor. Az adott perifériára nem tudta kiírni a kívánt számú bájtot a DOS.
Errors on list device indicate that it may be off-line. Please check!	A listázó egység nincs bekapcsolva, vagy off-line állapotban van. Kérem ellenőrizni! Nézzük meg, hogy bekapcsoltuk-e a nyomtatót!
File cannot be copied onto itself	Az állomány önmagára nem másolható. Ha egy állományt másolni akarunk az eredeti lemez tartalomjegyzékébe, akkor meg kell változtatnunk a nevét, vagy másik lemezre másoljuk!
File creation error	Állomány-előállítási hiba. Ez előfordulhat, ha megtelt a lemez, vagy nem tudunk felülírni egy meglévő állományt, mivel olyan állományt akartunk változtatni, amely védett, csak olvasásra érhetjük el.
File not found	Az állományt nem találja. Lehetséges, hogy rosszul adtuk meg az elérési útvonalat, vagy az állomány azonosítóját.
Format failure	Formázási hiba. A lemezt nem tudjuk használni, mert lemezhibás.
Format another (Y/N)?	Formáz egy másikat (igen/nem)? Választhatunk, hogy az ENTER lenyomása után újabb lemezt formázzon-e gépünk, vagy befejezze a műveletet.
Format complete	A formázás befejeződött.
Format not supported on Drive D:	Nem lehet megformázni a D: egységet. Az adott lemezegység nem formattálható.
Incorrect DOS version	Rossz DOS verziót alkalmaztunk. A kiadott parancs nem felel meg az alkalmazott DOS előírásainak.

Incorrect numbers of parameters	Rossz a paraméterek száma. A hibát az okozhatja, hogy kifelejtettünk egy paramétert, vagy a paraméterek közé nem tettünk szóközt.
Insert new diskette for drive d and press ENTER when ready	Helyezzen be egy új lemezt az egységbe, majd nyomja meg az ENTER billentyűt, ha elkészült! Az új formázatlan lemez behelyezése után lenyomott ENTER billentyű hatására megkezdődik a lemez formázása.
Insufficient disk space	Nincs elég hely a lemezen. Nem tudjuk az adott állományt erre a lemezre elhelyezni, mert kevés a szabad hely.
Insufficient memory	Kevés a memória. A rendszerhez tartozó memóriában kevés a szabad terület, nem tudjuk végrehajtani a parancsot a memóriában.
Invalid date	Rossz a dátum. Helytelen formátumban írtuk be a dátumot, vagy érvénytelen értéket adtunk meg.
Invalid directory	Érvénytelen tartalomjegyzék. Rosszul adtuk meg a könyvtárhoz vezető útvonalat.
Invalid disk change	Nem szabad a lemezt kicserélni. Vigyázat! Olyan esetben cseréltük ki a lemezt, amikor azon még nyitott állományok voltak. Tegyük vissza a lemezt!
Invalid drive in search path	Helytelen meghajtóra hivatkoztunk a keresési úton. A PATH parancsban érvénytelen lemezegységet adtunk meg, ezért egy parancskeresés közben írja ezt az üzenetet a DOS.
Invalid drive or file name	Érvénytelen lemezegység vagy állománynév.
Invalid drive specification	Érvénytelen lemezegységet adtunk meg.
Invalid file name or file not found	Érvénytelen állománynév vagy nem létező, hibás állománynév. Olyan állományt szerettünk volna átnevezni vagy kiíratni, amelynek hibás volt a neve, vagy egyáltalán nem létezett a könyvtárban.
Invalid media or track 0 bad disk unusable	A lemez fizikailag nem megfelelő. Előfordulhat, hogy a lemez 0. sávja hibás, ezért nem lehet formázni. Más esetben a lemezegység és a lemez nem kompatíbilis.

Invalid number of parameters	Érvénytelen számú paramétert adtunk meg. A parancsban túl sok vagy túl kevés paramétert adtunk meg.
Invalid path	Érvénytelen útmegadás. Valamelyik parancsban a megadott elérési út helytelen, vagy nem megengedett karakterek vannak benne, vagy hosszabb adott karakterszámnál.
Invalid path, not directory or directory not empty	Helytelen útmegadás, nem könyvtár, vagy a megadott könyvtár nem üres. Olyan könyvtárat adtunk meg, amelyik nem létezik, vagy amelyik nem üres. Csak olyan könyvtárat lehet törölni, amely üres.
Invalid path or file name	Érvénytelen útmegadás vagy állománynév. Olyan elérési utat adtunk meg, amelyik nem létezik, vagy hibás, vagy az állományazonosító sem létezik.
Invalid time	Érvénytelen időt adtunk meg. Olyan időpontot adtunk meg, amely nem helyes, vagy nem a megfelelő formátumú.
Memory allocation error. Cannot load COMMAND, system halted	Memória lefoglalási hiba miatt a parancsprocesszort nem lehet betölteni, a rendszer leállt. Hibát követtünk el a DOS betöltésekor. Töltsük be újra!
Missing operating system	Hiányzó operációs rendszer. A DOS hiányzik a merevlemezről, ezért onnan nem tölthető be. Másoljuk át a merevlemezre a DOS-t!
-More-	Több. A következő oldal megjelenítéséhez nyomjon le egy billentyűt!
Name of list device [PRN]	A nyomtatóegység azonosítója [PRN]. Lehet még: LPT1:, LPT2:. A nyomtatás megkezdése után megjelenő üzenet. Az ENTER billentyű lenyomása után egy párhuzamos kapcsolóval rendelkező nyomtatóra kerül a kiírandó szöveg.
Non free file handles. Cannot start command.com, exiting	Nincs szabad állománykezelő. A command.com nem indítható, kilépés. A parancsprocesszort nem lehet újból behívni, mert pillanatnyilag túl sok a nyitott állomány.
No path	Nincs elérési útvonal megadva. A DOS nem tudja elvégezni az adott parancsokat az aktuális könyvtárban.

Non-System disk or disk error. Replace and strike any key when ready	Nem rendszerlemez vagy hibás a DOS lemez. Cseréljük ki a lemezt, majd nyomjunk le egy tetszőleges billentyűt! Vegyük ki az A: meghajtóból a lemezt, és indítsuk újra a rendszert. (Tehetünk egy DOS-t tartalmazó lemezt az A: meghajtóba, majd így indítjuk újra a rendszert.)
Parameter not compatible with fixed disk	A megadott paraméter nem használható fixlemez esetén.
Path not found	Az elérési útvonal nem létezik.
Path too long	Túl hosszú az elérési útvonal, 63 karakternél hosszabb utat írtunk le.
Read error, COUNTRY.SYS	Olvasási hiba történt a COUNTRY.SYS állomány olvasásakor. Előfordulhat, hogy tönkrement a használt lemez. Próbálkozzunk biztonsági másolattal!
Reinsert diskette for drive x: and strike ENTER when ready	Helyezze vissza az X: egységbe a lemezt, és nyomja le az ENTER billentyűt, ha kész. Olyan esetben, ha a kiadott parancs nem tudja beolvasni a memóriába az összes rendszerfájlt, akkor ismét be kell tenni a lemezt, és újra indítani az ENTER lenyomásával.
Resident part of print installed	A PRINT parancs állandóan a tárban lévő része installálva van. Megtörtént a PRINT parancs első futása. A memóriába került egy program, amely feldolgozza a következő PRINT parancsokat.
Syntax error	Szintaktikus hiba. Formai hiba van a megadott parancsban. Ellenőrizzük!
System transferred	A rendszer átvitele megtörtént. Az újonnan formázott lemezre rákerültek a rendszerfájlok.
Terminate job (Y/N)?	Befejeződjön a program futása (I/N)? Ha a DOS parancsot teljesít, és megnyomjuk a Ctrl-Break billentyűket, akkor választhatunk, hogy be akarjuk-e fejezni a feldolgozást (igen), vagy a következő parancs végrehajtását szeretnénk (nem).
Track 0 bad-disc unusable	A lemezen a 0. sáv nem használható. A lemez használhatatlan, mert nem lehet elhelyezni rajta a boot-rekordot, a tartalomjegyzék rekordot, valamint az állományleíró táblázatot.

Unrecognized command in CONFIG.SYS	Nem felismerhető parancs a CONFIG.SYS-ben. Javítsuk ki a hibásan megadott parancsot, majd indítsuk újra a rendszert.
Warning! All data on non-removable disk drive x will be lost proceed format (Y/N)?	Figyelem! Az összes adat elveszik az X: fixlemezegységről, megformázzam (I/N)?. Csak akkor történik meg a formázás, ha igennel válaszolunk.

A QBASIC nyelv alkalmazásakor a képernyőn megjelenő hibakódok értelmezése

Kód	*A hiba szövege*	*A magyar jelentése*
1.	Next without for	for nélküli next
2.	Syntax error	formai hiba
3.	Return without gosub	return gosub nélkül
4.	Out of data	nincs több adat
5.	Illegal function call	hibás függvényhívás
6.	Overflow	túlcsordulás
7.	Out of memory	betelt a memória
8.	Undefined line number	nem definiált sorszám
9.	Subscript out of range	tartományon kívüli sorszám
10.	Duplicate definition	már egyszer definiáltuk
11.	Division by zero	nullával való osztás
12.	Illegal direct	nem megengedett direkt parancs
13.	Type mismatch	típus-összeférhetetlenség
14.	Out of string space	a stringterület betelt
15.	String to long	túl hosszú a karakterlánc
16.	String formula too complex	túl bonyolult a stringkifejezés
17.	Cannot continue	nem folytatható
18.	Undefined user function	definiálatlan függvény
19.	No resume	nincs resume
20.	Resume without error	resume hiba nélkül
21.	Unprintable error	nem nyomtatható hiba
22.	Missing operand	hiányzik az operandus
23.	Line buffer overflow	sor-buffer túlcsordulás
24.	Device timeout	készülék-időtúllépés
25.	Device fault	készülékhiba
26.	For without next	next nélküli for
27.	Out of paper	kifogyott a papír
28.	Unprintable error	nem nyomtatható hiba
29.	While without wend	wend nélküli while

30.	Wend without while	while nélküli wend
31–32.	Unprintable error	nem nyomtatható hiba
33.	Duplicate label	megismételt címke
34.	Unprintable error	nem nyomtatható hiba
35.	Subprogram not defined	nem definiált szubrutin
36.	Unprintable error	nem nyomtatható hiba
37.	Argument-count mismatch	argumentum-összeférhetetlenség
38.	Array not defined	nincs definiálva tömb
39.	Case else expected	else nélküli case
40.	Variable required	hiányzik a változó
41–49.	Unprintable error	nem nyomtatható hiba
50.	Field overflow	mezőtúlcsordulás
51.	Internal error	belső hiba
52.	Bad file number	rossz fájlszám
53.	File not found	nem található fájl
54.	Bad file mode	hibás a fájlmegnyitási mód
55.	File already open	már megnyitott fájl
56.	Field statement active	már aktív a field utasítás
57.	Device I/O error	I/O eszköz hiba
58.	File already exists	már létező fájl
59.	Bad record lenght	hibás rekordhossz
60.	Unprintable error	nem nyomtatható hiba
61.	Disk full	megtelt a lemez
62.	Input past end	fájl-vége utáni input
63.	Bad record number	hibás rekordszám
64.	Bad file name	hibás fájlnév
65–66.	Unprintable error	nem nyomtatható hiba
67.	Too many files	túl sok fájl
68.	Device unavailable	nem létező készülék
69.	Communication buffer overflow	betelt a kommunikációs buffer
70.	Disk write protected	írásvédett lemez
71.	Disk not ready	nem áll készen a lemezegység
72.	Disk media error	lemezanyag hiba
73.	Advenced Feature	továbbfejlesztett lehetőség
74.	Rename across disks	átnevezés más lemezre
75.	Path/file acces error	útvonal/fájl hozzáférési hiba
76.	Path not found	az útvonal nem található
77.	Unprintable error	nem nyomtatható hiba

2. A leggyakrabban előforduló fogalmak jelentése

Abakusz: Az ókorban még nem ismerték a helyiérték fogalmát, ezért a homokba vagy négyszögletes táblára párhuzamos vonalakat húztak, ezzel jelezték a számok nagyságrendjét. A rájuk helyezett kavicsok vagy jelek pedig a mennyiséget. Napjainkban is megtaláljuk kínai változatát, a huzalokon mozgatható fagolyókkal (japán neve: szorobán).

Adat (data): Ismert tények, hírek, fogalmak valamilyen módon rögzített alakja. Az információ megjelenési formája. Ez az ábrázolási mód teszi lehetővé az adatok feldolgozását, tárolását, átadását.

Adatállomány (data file): Egymással logikai kapcsolatban lévő adatok rendezett halmaza, melyet valamilyen szempontból egységnek tekinthetünk, és a feldolgozás során együtt kezelünk. Általában rekordokból épül fel. *Például*: személyi adatok rekordjaiból.

Adatátvitel (data transmission): Az adatok elektronikus úton történő továbbítása számítógép vagy telefonvonalak segítségével.

Adatbázis (data base): Egymással logikai kapcsolatban lévő adatállományok halmaza, amely egy adatfeldolgozó rendszer működéséhez elengedhetetlenül fontos adatokat tartalmazza, *például:* bérfeldolgozáskor a személyi nyilvántartás, a béradatok, a levonások, az adóbefizetések stb. állományai.

Adatbevitel (data input): A folyamat során a számítógépes rendszerbe valamilyen adat kerül.

Adatfeldolgozás (data processing): Adatok gyűjtése és rögzítése a számítógép számára feldolgozható formában. Adatok kezelése, melynek során az adatokkal műveleteket végzünk. Adatok megjelenítése a kívánt formában (tartalmazza az adatgyűjtést és az adatkezelést).

Adathordozó (data medium): Az adatok tárolására, számítógépbe történő bevitelére, kijövő adatok rögzítésére szolgáló eszköz. *Például*: mágneslemez, mágnesszalag, lyukkártya stb.

Adatkivitel (data output): A folyamat során a számítógépes rendszerből valamilyen adatok kerülnek ki. Megjelennek géppel olvasható formában, vagy az alkalmazó által felhasználhatóan.

Adatmező (data field): Egyetlen adat tárolására felhasznált hely. *Például:* személyi nyilvántartásnál lehet név mező, születési adatok mező, lakcím mező stb.

Adatrögzítés (data recording): Az adatok géppel olvasható adathordozóra történő felvitele.

Adatszerkesztés (data editing): Az adatok megjelenési formájának módosítása, változtatása hibás beírás esetén, vagy azért, hogy kijelzéskor jól olvasható, áttekinthető legyen.

Adatvédelem (data protection): A tárolt adatok biztonságos, bármikor történő felhasználhatóságát, valamint illetéktelen hozzáférések kiküszöbölését jelenti. Megakadályozza az adatok megváltoztatását vagy megsemmisítését.

Akkumulátor (accumulator): Az a regiszter, amelyben a műveletek eredménye keletkezik.

Aktuális érték (actual value): Az az érték, amellyel egy adott változó, adott időpillanatban rendelkezik.

Aktuális könyvtár (actual directory): Egy adott lemez adott könyvtára, amelyben az állományokkal kapcsolatos műveleteket éppen végezzük. A lemezen beállíthatjuk, hogy melyik legyen ez, és ekkor nem kell mindig leírni a teljes elérési utat.

Aktuális meghajtó (actual driver): Az a meghajtó, amelyen a lemezkezelő műveletek során éppen dolgozunk. Beállíthatjuk, hogy melyik legyen az, így nem kell mindig megadni, hogy melyik lemezen dolgozunk.

Aktualizálás (updating): Adott állománynak vagy egy részének módosítása, naprakésszé tétele.

Alacsony szintű programozási nyelv (low-level language): A gép számára közvetlenül érthető utasításokat ad. *Például*: a gépi vagy assembly nyelv.

Alapkonfiguráció (configuration): A számítógép működéséhez elengedhetetlenül szükséges perifériák és az alapgép összekapcsolása.

Alapszám (radix): Helyiértékes számrendszerekben az a szám, amelyet megfelelő hatványra kell emelni a helyiérték sorszámának megfelelően.

Alfanumerikus billentyűzet (alphanumerical keyboard): Olyan billentyűcsoport, amely betűkből, számokból és funkcióbillentyűkből áll.

Algoritmus (algorithm): Egy feladat megoldásának egymást követő lépéseit leíró szabályok összessége.

Alkalmazói program (application program): Egy konkrétan megfogalmazott feladat megoldására készített számítógépes program. A felhasználónak csak az aktuális adatokkal futtatni kell a kész programot. Felhasználói programnak is nevezhetjük.

Alkalmazói szoftver (application software): Az alkalmazói programok összessége.

Alkönyvtár (subdirectory): Egy adott könyvtáron belül az összetartozó fájlok csoportja. Ha egy könyvtárban több adatállomány van, akkor ezeket különböző alkönyvtárakba sorolhatjuk.

Állomány (file): A számítógépet működtető programok és a feldolgozásra váró adatok logikailag összefüggő rendszere.

Almenü: Olyan menü, amely egy-egy részfeladat megoldására nyújt választási lehetőséget.

Analóg (analog): A folytonosan változó fizikai mennyiségeket folytonosan ábrázoló eszközök jelzője. *Például*: az óramutató folyamatos mozgása mutatja az idő múlását.

Aritmetikai és logikai egység–ALU (arithmetic and logic unit): A számítógép központi vezérlő egységének az a része, amely végrehajtja az aritmetikai és logikai műveleteket.

Aritmetikai kifejezés (arithmetic expression): Olyan kifejezés, amely számértékű konstansokból, változókból és aritmetikai műveleti jelekből épül fel, egy adott programozási nyelv szabályai szerint. Az értéke tetszőleges szám lehet.

ASC II (American Standard Code for Information Interchange): Amerikai szabványos információcsere-kód. Világszabvány, amely az információk nemzetközi cseréjét teszi lehetővé karakterek kódolása segítségével.

Assembler (assembler): Olyan fordítóprogram, amely az assembly nyelvű programból a gép számára érthető, gépi nyelvű programot állít elő.

Assembly nyelv (assembly language): Olyan alacsony szintű nyelv, mely közel áll a gépi nyelvhez, de nem bináris formában dolgozik, hanem kulcsszavakat használ. *Például*: a műveleti jeleknél ADD = összead.

Azonosító (identifier): Adatot vagy adathalmazt megnevező karakter vagy szöveg.

Bájt (byte): Általában 8 bitből álló információ egysége, melyet önkényesen választottak.

BASIC (Beginners All-purpose Symbolic Instruction Code): Kezdők általános szimbolikus utasításkódja. Kezdők számára kifejlesztett magas szintű programozási nyelv, melyet főként a mikroszámítógépeknél alkalmaznak.

BASIC értelmezőprogram (BASIC interpreter): A magasszintű nyelven írt programot utasításonként fordító, értelmező, az értelmezett utasításban előírt műveleteket rögtön elvégző program. A végrehajtás előtt ellenőrzi az utasítást, hiba esetén hibaüzenetet ír a képernyőre, és csak a hiba kijavítása után hajtja végre az utasítást. Lehetővé teszi, hogy a kezelő a program futása közben beavatkozzon a futásba.

Beépített konstans (built constans): Zsebszámológépeknél a könnyebb, gyorsabb, pontosabb számolás érdekében egyes állandók értékét beégették a memóriájába, így alkalmazásukkor azokat csak elő kell hívni!
Például: pí = 3.1415927 vagy e = 2.7182818

Bemeneti vagy beviteli egység (input device): Olyan periféria, amelynek segítségével bevihetjük az adatokat a számítógépbe. *Például*: háttértároló, billentyűzet.

Bemenő adat (input data): A számítástechnikai rendszerbe feldolgozásra bevitt adat.

Beszúrás (insert): Egy meglévő szövegbe karakterek vagy sorok elhelyezése, esetleg az adatállományba új rekord beillesztése.

Betöltő rekord (boot record): Olyan rekord, amely a tárba kerülése után aktivizálódik, majd betölti a DOS következő tárolóban lévő részét. Azonosítója nincs, de könnyű megtalálni, mert a 0. sorszámú szektorban van.

Bevitel (input): Az adatoknak a perifériákról a központi egységbe juttatása.

Beviteli-kiviteli (input-output): Az adatok bevitelét és kivitelét egyaránt segítő eszközök jelzője. *Például:* perifériák.

Billentyűzet vagy klaviatúra vagy tasztatúra (keyboard): Valamilyen rendszer szerint elhelyezett billentyűk halmaza, amelynek segítségével a számítógépbe adatokat juttathatunk, és működtethetjük a számítógépet.

Bináris (binary): Számok, karakterek, üzenetek stb. ábrázolása két különböző, egymást kizáró állapot segítségével. *Például:* 0 és 1 számjegy, vagy van áram és nincs áram stb.

Bináris kód (binary code): Az adatok kettes számrendszerbeli ábrázolási módja. Csak 0 és 1 számjegyet alkalmazunk.

Bit (binary digit): Az angol kifejezésből származó rövidítés. A bináris kód legkisebb eleme, ezért azonos egy bináris számjeggyel. Értéke 0 vagy 1 lehet. Az adatok tárolásának, feldolgozásának az alapegysége a számítástechnikában.

Blokkdiagram vagy folyamatábra (block diagram): Egy feladat megoldási menetének, lépéseinek grafikus ábrázolása megállapodás alapján készített geometriai jelekkel.

Botkormány (joystick): Bemeneti periféria, amelyet általában mikroszámítógépes játékoknál használunk. Rúdjának mozgatásával a monitoron különböző irányokban mozgathatunk egy pontot.

Boole-algebra (boolean algebra): A Boole angol matematikus által felépített olyan kételemű algebra, amely az igaz és hamis állításokra épül.

Célprogram vagy felhasználói program vagy alkalmazói program (application program): Egy konkrét alkalmazói feladat megoldására készített program. *Például:* játékprogramok, szövegszerkesztők, órarendkészítő stb.

Ciklus (loop): A program valamely részének egy adott feltétel teljesülésétől függő egymás utáni végrehajtása, megismétlése.

Ciklusmag (cycle): A ciklusban ismétlődő utasítások halmaza, amely a ciklusnak része.

Ciklusutasítás (loop statement): A ciklusszervezést segítő utasítás. *Például:* FOR NEXT.

CP/MC (Control Program for Microprocessors): Olyan operációs rendszer, amelyet a 18080/Z80-as mikroszámítógépeken használhatunk.

Csak olvasható tár — ROM (red only memory): A memóriának az a része, amelyhez nem nyúlhatunk. A benne tárolt információt felhasználhatjuk, de meg nem változtathatjuk.

Csatolóegység vagy interfész (interface): Olyan hardver eszköz, amely lehetővé teszi két funkcionális egység összekapcsolását. Az aktuális jeleket egy másik készülék vagy program számára érthetővé tevő vagy átalakító berendezés.

Csillag * (asterisk): Lehet: 1. A szorzás műveleti jele. 2. Helyettesítő karakter. Fájlok megadásakor egy teljes szót helyettesíthet. *Például:* nevét vagy kiterjesztését.

Csip vagy áramköri lapka vagy morzsa (chip): Néhány négyzetmilliméter felületű, vékony, négyszög alakú, felvezető lemez. A felületén valamilyen áramkört valósítottak meg. Kialakíthatnak teljes integrált áramkört vagy áramkörrendszert is.

Decimális szám (decimal numeral): Tizes számrendszerbeli szám.

Deklaráció (declaration): Adott változó jellemzőinek közvetlen vagy közvetett módon történő megadása. *Például:* típusa, nagysága, mérete.

Dekódol (decode): Kódolt jelet visszaalakít.

Dekódoló (decoder): Olyan eszköz, amellyel a kódolt adatot visszaalakíthatjuk a kódolás előtti állapotnak megfelelő formába.

Demoduláció (demodulation): Az eredeti jelet, információt visszaállítjuk a vevőkészülékben az adóból érkező megváltoztatott vivőhullámból.

Digitális vagy számjegyes (digital): A változó mennyiségeket számjegyekkel felírt formában megjelenítő eszköz jelzője. *Például:* a karóra kijelzőjén megjelenő számok mutatják az időt.

Dimenzió vagy mérés, nagyság (dimension): A tömbök mérete. Megmutatja, hogy egy tömbön belüli elem helyének megadásához hány indexet kell megadni.

Direkt mód, közvetlen vagy parancsmód (direct mode): Ebben az üzemmódban a számítógép csak parancsot fogad el, és azt azonnal végre is hajtja.

Dollárjel $ (dollar sign): A BASIC nyelvben a változó nevének utolsó karaktereként a szöveges változó azonosítója.

DOS (Disk Operating System): Lemezes operációs rendszer. Azoknak a programoknak az összefoglaló neve, amelyek kapcsolatot teremtenek a számítógép és a perifériák, valamint a számítógép és a felhasználó között. Tehát az operációs rendszer olyan programok összessége, amelyek a számítógépet működtetik, vezérlik a programok és az adatok kezelését, és igyekeznek a számítógép erőforrásainak maximális kihasználására.

Egér (mouse): A mikroszámítógépekhez kapcsolható periféria, amelyet egy sík lapon mozgatva, mozgatni tudjuk vele a képernyőn megjelenő kurzort. Az egéren lévő nyomógombok lehetővé teszik számunkra, hogy a számítógép számára igen–nem utasítást adjunk.

Egészrész (integer part): Olyan egész szám, amely egy valós számhoz legközelebb áll, de annál nem nagyobb. Jelölése: [ ]. *Például:* [4,2]=4, [4,8]=4, [–5,3]=–6.

Egész változó (integer variable): Olyan változó, amelynek csak egész értékeket adhatunk.

Egyfelhasználós operációs rendszer : Olyan operációs rendszer, amely egyszerre csak egy felhasználót szolgál ki, egyszerre csak egy program futtatását teszi lehetővé.

Egységszám (decive number): Olyan szám, amely a számítógép perifériáinak azonosítását teszi lehetővé. Készülékszámnak is szokták nevezni.

Elágazás (branch): Olyan esetekben, amikor a program folytatása bizonyos feltételek teljesülésétől függ és ilyenkor két vagy több lehetséges irányban haladhatunk, elágazásról beszélünk.

Elágazási pont (branch point): Olyan pontja a programnak, ahonnan bizonyos feltételek teljesülésétől függően a program futása több irányban folytatódik.

Elágaztató utasítás (branch instruction): Olyan utasítás, amely valamely feltétel teljesülése, illetve nem teljesülése esetén biztosítja, hogy a program futása különböző irányokban folytatódjék.

Elektromágneses relé (electromagnet relay): (Jelfogó) Olyan készülék, amely tekercsből és érintkezőkből épül fel. A tekercs gerjesztésekor mágneses tér jön létre, melynek hatására az érintkezők áramkört zárnak vagy bontanak.

Elektroncső (electronic tube/valve): Olyan egyenirányító vagy erősítő eszköz, amely zárt, légüres térben elhelyezett elektródokból áll. (A tér lehet üvegbura vagy fémcső, az elektródok lehetnek a katód, az anód és a rács.)

Elérési út (access path): Az az útvonal, amelyen egy adott könyvtárból eljutunk egy másik könyvtárba (az érintett könyvtárak felsorolása érintési sorrendben).

Eljárás (procedure): Olyan része a programnak, amely meghatározott célú tevékenységet végez el. (*Például:* egy eljárás során árat emelünk, másik eljárás árat csökkent, a harmadik összesíti a napi bevételt.)

Eljáráshívás : Olyan utasítása a programnak, mellyel a program futása az adott nevű eljáráson folytatódik.

Eljárásnév (procedure name): Olyan azonosító, melynek segítségével az eljárásra hivatkozhatunk, programon belül hívhatjuk az eljárást.

Előjelkarakter (sign character): Olyan karakter, amely az előjelpozíción áll, és így megadja, hogy az adott szám pozitív vagy negatív.

Előjelváltó billentyű: Olyan billentyű a zsebszámológépen, amely megváltoztatja a kijelzőn megjelenő szám előjelét, mivel (–1)-gyel való szorzást jelent.

Előrelapozás: Olyan esetekben, amikor a képernyőre nem fér rá a kiírandó szöveg, akkor lehetőségünk van arra, hogy egy képernyőnyit kiírassunk, majd a szükséges adatok megismerése után egy billentyű lenyomásának hatására újabb képernyőnyi adat jelenik meg.

Értékadás (assign): Olyan hozzárendelés, amikor a változónak típusának megfelelő értéket adunk.

Értékadó utasítás (assigning instruction): Olyan utasítás, amellyel egy változóhoz értéket rendelünk. (*Például:* LET x=5.)

Értelmezőprogram (interpreter): Olyan program, amely egy magasszintű programnyelven megírt programot utasításonként lefordít a gép nyelvére, és ezt végre is hajtja a következő utasítás beolvasása előtt. (A program futása és fordítása nem válik szét.)

Fájl (file): (Állomány, adatállomány) A számítógépben tárolt információk és programok rendezett összessége, melyek egymással logikai kapcsolatban állnak.

Fájlnév (filename): (Állomány neve) Olyan elnevezés, amellyel megjelöljük az egyes állományokat, hogy kereséskor a számítógép könnyen megtalálja azokat.

Fastruktúra: Olyan gráf, amelyben bármely két pontot pontosan egy út köt össze, és éleinek száma eggyel kevesebb a szögpontok számánál.

Felbontás (resolution): Olyan fontos jellemzője a számítógépes grafikának, amellyel a képélességet adják meg. Ilyenkor megadják az egymástól megkülönböztethető képelemek számát, adott hosszúságra vonatkoztatva. A felbontás függőleges és vízszintes irányban különböző lehet, ezért mindkét adatot megadják.

Feldolgozás sebessége (process speed): Olyan érték, amely megmutatja, hogy a bevitt adatokból milyen gyorsan kaphatjuk meg a kívánt eredményt.

Fejlesztői program: Olyan program, amely segíti a programozást, a programok tesztelését és fejlesztését.

Felfelé kompatibilitás (upward compatible): Olyan jellemzője a gyártott géptípussorozatnak, amely biztosítja, hogy a később gyártott típusokon futnak a korábbi típusokon futó programok módosítás nélkül, de fordítva ez nem mindig igaz.

Felhasználói program (application program): (vagy célprogram vagy alkalmazói program) Olyan program, amelyet egy konkrét feladat számítógépes megoldására írnak meg.

Feltétel (condition): Olyan logikai kifejezés, amely logikai műveletekből és változókból épül fel.

Feltételes elágazás (conditional branch): Olyan elágazás, amikor a program futása bizonyos feltételek teljesülésétől függően folytatódik.

Feltételes ugrás (conditional jump): Olyan összetett utasítás, amely lehetővé teszi, hogy a program futása az adott feltétel teljesülésétől függően a soron következő utasításon vagy egy másik utasításon folytatódjon. (IF THEN ELSE.)

Feltétlen ugrás (unconditional jump): Olyan utasítás, amely elősegíti, hogy a program végrehajtása ne a soron következő utasításon folytatódjon, hanem azon, amit ez az utasítás kijelöl. (GOTO)

Fényceruza (ligh pen): Olyan ceruzához hasonlító eszköz, amelyet a megjelenítőhöz kapcsolunk. Segítségével rajzolhatunk a képernyőre, vagy egyes elemeket kiválaszthatunk a képernyőn.

Fixlemez, rögzített lemez, nem cserélhető lemez (fixed disk): Olyan merev mágneslemez, amelyet egybeépítettek a lemezmeghajtó egységgel, és a környezettől elzártak.

Floppi diszk (floppy disk), hajlékonylemez: Olyan hajlékony mágneslemez, amelyet védőtokba helyeztek, és azzal együtt használják.

Folyamatábra (flowchart): Olyan ábra, amely szemléletesen mutatja egy feladat megoldásának lehetőségeit, ahol az adatok közti kapcsolatokat, műveleteket megállapodás szerinti jelképekkel rajzolják le.

Fordítóprogram (compiler): Olyan számítógépes program, amely az egyik programnyelvről a másik programnyelvre alakítja át az adott programot.

Formattálás (formatting) vagy formázás: Olyan eljárás, melynek során a mágneslemezeket alkalmassá tesszük arra, hogy adatokat tároljunk rajtuk vagy olvassunk le róluk.

Ilyenkor helyezzük el a lemezen azokat a jeleket, amelyek gyorsítják az adatok keresését. (Vigyázat, ilyenkor a mágneslemezről minden eddigi információ törlődik!)

Forráslemez (source disk): Olyan lemez, amelyről az eredeti adatállományt átmásoljuk valamilyen más adattárolóra.

Főkönyvtár vagy gyökérkönyvtár (main directory): Olyan kiindulási, kezdeti könyvtár, amely tartalmazza az összes állományt.

Főmenü (main menu): Olyan menü, amely a programban első helyen szerepel. Általában ezzel jelentkezik be a program, és a részfeladatok végrehajtása után ugyanide tér vissza.

Főprogram (main program): Olyan programrészlet, amely meghatározza a szubrutinok végrehajtási sorrendjét.

Funkcióbillentyűzet (functional keyboard): Olyan elkülönülő része a billentyűzetnek, amely a sajátos gépi funkciókat megvalósító billentyűket tartalmazza. Ezekhez a billentyűkhöz programok segítségével különböző jelentéseket rendelhetünk.

Futtatás (run): Olyan cselekvés, amellyel végrehajtjuk az elkészült programot.

Gépi adatfeldolgozás (machine data processing): Olyan adatfeldolgozás, amelyet számítógéppel végzünk.

Gépi kód (machine code): Olyan alakja az utasításoknak, amelyet a számítógép közvetlenül értelmezni tud.

Hajlékonylemez (floppy disk): Olyan hajlékony mágneslemez, amelyet védőtokba helyeztek el, és ezzel együtt alkalmaznak.

Hajlékonylemezes tároló (floppy disk unit): Periféria, ahol hajlékonylemezek segítségével őrizhetjük meg a szükséges adatokat.

Hardver (hardware): A számítógép alkatrészeinek és az összes gépi eszközöknek az együttese (keményáru).

Háttértároló (external storage, backfround storage): Az olyan programrészeket, adatokat, amelyek nem vesznek részt adott időpontban a feldolgozásban, itt tárolhatjuk. Segítségével növelhetjük az operatív tár kapacitását, mert a háttértároló tartalmát bármikor felhasználhatjuk.

Helyettesítő karakter vagy joker (wildcard): Olyan helyettesítő jel, amely bármely állománynév egy vagy több betűjét helyettesítheti.

Hexadecimális (hexadecimal): Olyan számrendszer, melynek alapszáma a 16. Számoláskor a 0 ... 9 számjegyeket és az A ... F betűket alkalmazza.

Hibajelzés (error indication): A számítógépnek a jelzése, amellyel figyelmezteti a felhasználót, hogy rosszul használja az adott programot, vagy arra, hogy hiba van a programban, esetleg az adatokban.

Hibaüzenet (error message): Olyan hibajelzés, melynek során a számítógép konkrétan megnevezi, kiírja a képernyőre a hiba okát.

Hozzáférési idő vagy elérési idő (access time): Az az időtartam, amely az adat kérése és megérkezése között telik el.

IBM-PC vagy IBM személyi számítógép (IBM Personal Computer): Olyan személyi számítógép, amelyet az IBM cég gyárt.

IC vagy integrált áramkör (integrated circuit): Áramkör, melynek elemeit egy áramköri lapkára szerelik, és hermetikusan elzárják a külvilágtól. Az áramkör elemei között az összeköttetést csak a tönkretételükkel szakíthatjuk meg.

Időosztás (time sharing): Olyan üzemmód a számítógépen, melynek hatására több folyamat látszólag egyidejűleg hajtódik végre.

Illesztőegység vagy interfész (interface): Előírások együttese, amelyek lehetőséget teremtenek két funkcionális egység összekapcsolására. Az összekapcsolást megvalósító hardvereszközt csatoló vagy illesztő egységnek is nevezik.

Index (index):

A) Jelkép, amely pontosan meghatározza egy adott halmaz valamelyik részét.

B) Olyan szám, amely azonosítja a tömbváltozó elemeit.

Indexes változó (indexed variable): Adategyüttes, amely azonos tulajdonságú adatokból áll, és az indexeik pontosan meghatározzák az adatok helyét.
Információ (information): Olyan adat, amely valamilyen új ismeretanyagot tartalmaz.
Inicializálás (initialization): Eljárás, melynek során a változóknak kezdeti értéket adunk.
Initialize: Kezdőértékkel ellát. Valamilyen alaphelyzet beállítása. A kifejezést a lemez formázására is szoktuk használni IBM számítógépek esetén.
Input vagy adatbevitel: Eljárás, melynek során a számítógépes rendszerbe adat kerül.
Interpreter vagy értelmezőprogram (lásd értelmezőprogram)
Insert: Olyan billentyű, amely elősegíti, hogy két karakter közé beszúrjunk, beillesszünk egy harmadikat. Lenyomása után a két karakter között egy üres hely keletkezik, ahova beírhatjuk az újat.
Író-olvasó fej (read-write-head) vagy mágnesfej: Olyan elektromágnes, amely a mágneses adathordozón lévő adatokat leolvassa, esetleg törli, vagy új adatokat ír rá.
Írásengedélyező nyílás (read/write slot): Olyan hosszúkás nyílás a hajlékony mágneslemez védőborítóján, melyen keresztül az író-olvasó fej hozzáérhet a lemezhez.

Joker karakter (lásd helyettesítő karakter)
Joystick (lásd botkormány)

Kalkulátor (lásd számológép)
Karakter (character): Megállapodás alapján használt jel, amelyet adatok ábrázolására alkalmaznak. *Például*: számjegyek vagy betűk.
Képdigitalizáló (scanner) vagy dekódoló vagy letapogató: Olyan adatbeviteli eszköz, melynek segítségével ábrákat, szövegeket olvas a számítógép. A papíron levő információkat képpontokra bontja, majd a számítógép számára érthető bitek sorozatára alakítja és beviszi a tárba.
Képernyőtörlés (clear screen): A képernyőn található adatokat eltávolítjuk, és ezzel lehetőségünk nyílik újabb karakterek kiíratására.
Készenléti jel (prompt): Jelzés a képernyőn, amely a számítógép bejelentkezésekor megjelenik, hogy tudtunkra adja, minden rendben, kezdhetjük a munkát. A DOS annak a meghajtónak a betűjelét is kiírja a promptba, amelyen megtalálta a DOS-t.
Kétállapotú billentyű: Olyan billentyű, amelyen az egyik karaktert az adott billentyű és a SHIFT billentyű együttes lenyomásával írhatjuk le.
Kezdeti értékadás (preset): Eljárás, melynek során a program futásakor használt változókat értékkel töltjük fel.
Kifejezés (expression): Olyan formula, amelyet műveleti jelekkel összekapcsolt konstansokból és változókból építünk fel, hogy adott kiindulási értékekből új értéket számoljunk ki.
Kijelző (indicator): Olyan eszköz, amely valamilyen folyamat eredményét előre meghatározott módon, látható formában mutatja meg.
Kimenő adat (output data): Olyan adat, amely a számítógépes feldolgozás során keletkezik, és valamilyen módon megjelenik a felhasználó számára.
Kimentés (save): Olyan eljárás, melynek során a fontos, későbbi időpontban ismét felhasználandó adatokat átmásoljuk valamilyen mágneses adathordozóra.
Kiterjesztés vagy típusjelzés: A DOS állomány azonosítójának a második része, amely azonos típusú állományok esetén ugyanaz lehet. Az állomány nevétől .(pont) választja el. *Például:* COMMAND.COM esetén a .COM
Kiviteli egység (output device): Olyan periféria, amelyen megjelennek a számítógépből kiviendő adatok.
Klaviatúra (lásd billentyűzet)
Kód (code): Olyan szabályok összessége, amelyek egyértelműen meghatározzák az adatok ábrázolási módját. Legtöbbször megadnak egy táblázatot, amelyből kiolvashatjuk, hogy egy az adott elemhez milyen kódot rendelhetünk. *Például:* az L betű kódja 12.
Kódolás (to code, to encode): Olyan tevékenység, melynek során adott táblázat segítségével ábrázolják az adatokat. Abban az esetben is kódolásról beszélünk, ha a feladat

megoldásához elkészített algoritmus alapján valamilyen programnyelven megírjuk a programot.

Kompatíbilis (compatible): Olyan tulajdonság, amely összeilleszthetőséget, felcserélhetőséget, helyettesíthetőséget jelent. Akkor beszélünk számítógépek kompatibilitásáról, ha az egyiken futó program változtatás nélkül futtatható a másikon is.

Konfiguráció (configuration): Olyan kiépítettséget jelent, amely megmutatja, hogy az alapgéphez milyen perifériákat kapcsoltunk, hogyan építettük fel a rendszerünket.

Konstans (constans): Olyan adat, amelynek értéke egy programon belül állandó.

Könyvtár (directory): Olyan logikai szerkezet, amely az állományok nyilvántartását végzi, a legfontosabb jellemzőinek segítségével (azonosító, nagyság, létrehozás vagy módosítás dátuma, ideje).

Kötegelt feldolgozás (batch processing): Az adatfeldolgozás lehetősége, melynek során a feldolgozandó adatokat meghatározott időpontokban dolgozzák fel. A közbeeső időben csak gyűjtik az adatokat.

Központi feldolgozó egység (CPU) (Central Processing Unit): Olyan egység, amely a számítógép összes tevékenységét irányítja, a gép agya. A számítógép perifériák nélküli része, alapvető egysége.

Kurzor (cursor): A képernyőn villogó jel, amely megmutatja, hogy hol jelenik meg az általunk begépelt karakter. Azt is jelzi számunkra, hogy a számítógép elfogadja tőlünk az adatot vagy parancsot, ha beírjuk.

Kurzorblokk (cursorblock): A billentyűzetnek egységként kezelt különálló része, amelyet elsősorban szövegszerkesztés közben használunk.

Kurzorpozicionálás (cursor move): A kurzormozgató billentyűk segítségével a kurzort a kívánt helyre állítjuk a képernyőn.

Külső információhordozó: Olyan eszköz, amelyen az információkat a számítógépen kívül tárolni tudjuk. *(Például:* lyukkártya, lyukszalag.)

Lemez (disk) vagy mágneslemez: Adathordozó, amely egy mágnesezhető anyaggal bevont körlap, és ezen az információkat mágneses jelek segítségével rögzíthetjük.

Lemezegység (disk unit) vagy mágneslemezegység: Olyan perifériája a számítógépnek, amely lehetővé teszi a mágneslemez alkalmazását.

Lemezmeghajtó (disk drive) vagy meghajtó: Olyan rendszer, amely mozgatja a mágnesszalagot vagy mágneslemezt. Ebben találjuk meg az író-olvasó fejeket és az ezeket vezérlő áramköröket is.

Lemezes működtető rendszer (lásd Disk Operating System = DOS)

Lezáró billentyű vagy sorzáró billentyű: Olyan funkcionális billentyű, amelynek lenyomása lezárja a beírt sort, ezáltal felszólítja a számítógépet, hogy értékelje a beírt sort, és hajtsa végre a kiadott parancsot, vagy tárolja az adatot. A billentyű felirata gépenként más és más lehet. *Például:* ENTER, RETURN, NEWLINE, CR.

Lézernyomtató (laser printer): A nyomtatandó képet lézersugár segítségével egy szelén henger palástjára írja. Az érintett terület ezáltal sztatikusan feltöltődik, ezért a henger elfordulása után a festékkazettából festék ragad rá, amely később a papírra tapad. A papírra egy hőelem égeti rá a festéket. Nagyon jó minőségű képet kapunk.

Lineáris program (linear program): Olyan program, amelyben az utasítások sorrendje megegyezik a végrehajtás sorrendjével.

Lista (list): Olyan adatok összessége, amelyet valamilyen rendszer szerint rendeztünk. Listának szoktuk nevezni a kinyomtatott vagy képernyőre kiírt programot is.

Lokális hálózat (local area network, LAN): Számítógéphálózat, amelynek elemei földrajzilag közel vannak egymáshoz (legfeljebb 1-2 km távolságra). Így kapcsolhatjuk össze az iskolában a mikroszámítógépeket.

Logarléc (slide-rule): Számtani műveletek elvégzésére alkalmazott eszköz, amelyen az adatokat távolsággal jelenítjük meg.

Lyukkártya (punch card): Olyan kártya, amelyen az adatokat különböző helyeken elhelyezett lyukak segítségével ábrázoljuk.

Lyukszalag (punch tape): Olyan szalag, amelyen az adatokat lyukak segítségével ábrázoljuk.

Mágnesfej (magnetic haed) vagy író-olvasó fej.
Mágneslemez (magnetic disk) vagy lemez.
Mágneslemezegység (magnetic disk unit) vagy lemezegység.
Mágnesszalag (magnetic tape): Olyan szalag, amelyet mágnesezhető anyaggal vontak be, és adattárolásra használhatunk.
Másodlagos funkció (secondary function): A billentyűkön felül jelzett karaktert a SHIFT billentyű és az adott billentyű egyidejű lenyomásával érhetjük el. Az ilyen tulajdonságú billentyűk több lehetőséget adnak számunkra.
Mátrix nyomtató (matrix printer): Olyan nyomtató, amely az egyes karaktereket pontok segítségével állítja elő. A nyomtatást karakterenként végzi, így sokféle grafikus ábrát nyomtathatunk vele. Javíthatjuk a nyomtatás minőségét, ha egy soron többször végigmegyünk.
Meghajtó (egység) (drive) vagy lemezmeghajtó.
Megszakítás (interrupt): Olyan esemény, amikor beavatkozunk a program futásába, valamiért megállítjuk a futást, de lehetőséget biztosítunk arra, hogy későbbi időpontban újraindítsuk.
Megjelenítő (display): Képernyő, amelyen vizuálisan megjelennek a program során felhasznált, illetve eredményként kapott adatok, programok, üzenetek, hogy a felhasználó ezeket elolvashassa. Kapcsolatot teremt a számítógép és az alkalmazója között.
Memória (memory) vagy központi tár: Homogén tároló, amely tartalmazza a számítógép pillanatnyi működéséhez szükséges adatokat és programokat. Nincs elkülönített rész az adatok és a programok számára.
Menü (menu): Képernyőn megjelenő felsorolás, amely megmutatja, hogy milyen lehetőségeink vannak a program felhasználására. A felhasználó maga döntheti el, hogy milyen tevékenységet választ.
Merevlemezes tároló (winchester): Nagy kapacitású háttértár, amelybe a mágneses felületű lemezeket légmentesen zárva, egybeépítenek a meghajtóval.
Mikroprocesszor (mikroprocessor): Olyan egység, amely elvégzi a program utasításai által előírt műveleteket. Ez a mikroszámítógép központi egysége. Feladatai közé tartozik a számítógép működésének vezérlése, a perifériákkal történő kapcsolattartás, adatforgalom lebonyolítása stb.
Mikroszámítógép (microcomputer): Általános célra kifejlesztett, olcsó számítógép, amelyet háztartásokban, játékprogramok futtatására, esetleg tanulásra használnak, de folyamatirányítóként az iparban is megtalálunk. Feldolgozóegysége egy mikroprocesszor.
Modem (modem-MOdulator DEModulator): Olyan készülék, amely lehetővé teszi, hogy az adatokat elektromos jelek formájában telefonvonalon keresztül továbbítsuk, majd az adatátvitel befejezése után visszaalakítja a számítógép számára érthető formájúvá. Távadatátviteli egységnek is tekinthetjük, mivel az adatok nagyobb távolságra történő továbbítását teszi lehetővé.
Multiprogramozás (multiprogramming): Olyan működése a számítógépnek, amikor látszólag egy időben több program fut egy számítógépen.
Műveleti precedencia (precedence of operation) vagy műveletek prioritása: Olyan szabály, amely meghatározza, hogy a különböző műveleteket milyen sorrendben kell elvégeznünk. *Például:* legelőször a hatványozást, majd a szorzást vagy osztást és végül az összeadást vagy kivonást kell elvégeznünk. A műveleti sorrendet zárójelezéssel megváltoztathatjuk, mert először a zárójelben lévő kifejezés értékét kell meghatározni.

Nem cserélhető lemez (fixed disc) vagy merev mágneslemez: Olyan mágneslemez, melyet a lemezegységgel egybeépítenek és a környezettől elzártan működtetnek.
Nem felejtő tár (nonvolatile storage): Olyan tárolóeszköz, melynek tartalma a tápegység feszültségének kikapcsolása után is megmarad.

Normálalak (normal form): A számot egy 1 és 10 közé eső szám és 10 hatványának szorzataként állítjuk elő. *Például:* $680 = 6{,}8 \cdot 10^2$.

Numerikus (numeric): Adatok, amelyeket számokkal írhatunk le.

Numerikus billentyűzet (numerical keyboard): Különállóan elhelyezkedő billentyűcsoport a számítógép billentyűzetén, amelyeket lenyomva számokat vihetünk be a számítógépbe.

Nyomtató (printer): Kiviteli egység, amely a kívánt információkat papírra írja.

Nyomtatópuffer: Olyan memória, amelyet a nyomtatóban helyeznek el. Alkalmazásakor az adatok a központi egységből nem a nyomtatóra, hanem a nyomtatópufferbe kerülnek, és innen dolgozza fel őket a nyomtató a központi egység segítsége nélkül. Ezért fordulhat elő, hogy a nyomtató még dolgozik, és a központi egység már újabb feladat elvégzésével foglalkozik.

Offline: Olyan tevékenység, amely nincs közvetlen kapcsolatban a számítógéppel. A számítógéphez tartozó berendezés jellemzője, amely jelzi, hogy nem közvetlenül vezérli a számítógép, hanem szükség van közvetítő adathordozókra is.

Oktális (octal): Adat, melyet nyolcas számrendszerbeli számként adtunk meg.

Olvasható-írható tár (random access memory) vagy RAM: A memóriának az a része, ahova az a program kerül, amelyet éppen futtatni akarunk, és ide kerülnek azok az adatok is, amelyekre szükség van a program futása közben. A számítógép kikapcsolása után a tartalma elveszik.

Olvasható tár (read only memory) vagy nem törölhető tár vagy ROM: A memóriának az a része, amelynek a tartalmát használhatja a mikroprocesszor, de a benne tárolt információkat nem változtathatja meg. Tartalma a számítógép kikapcsolása után is megmarad, így ismételt bekapcsolás után rendelkezésünkre áll. Itt tárolódnak a számítógép működéséhez feltétlenül szükséges programok, adatok.

Online: Olyan eszköz jelzője, amely közvetlen kapcsolatban van a számítógéppel, ezért a gép közvetlenül vezérli.

Operációs rendszer (operating system): Olyan programok összessége, amelyek vezérlik a programok végrehajtását, lebonyolítják a perifériaműveleteket, összehangolják a számítógép különböző egységeinek működését és elősegítik a számítógép és a felhasználó közti párbeszédet. A számítógéppel végzett munkánk leegyszerűsödik az operációs rendszernek adott parancsok sorozatára.

Operatív tár (operating memory) vagy operációs memória: A központi egység része, amely tartalmazza az éppen futó programot, a hozzá szükséges adatokat és az operációs rendszer szükséges részeit.

Parancs (command): Olyan utasítás, amelyet a számítógép azonnal értelmez és végrehajt. *Például:* RUN.

Parancsmód (command mode) vagy közvetlen üzemmód vagy direktmód: A számítógép üzemmódja, amikor csak parancsot fogad el, amit azonnal végre is hajt.

Párbeszédes üzemmód (interactiv mode): Munkamódszer, melyben a számítógép és a felhasználó felváltva küldi egymásnak az üzeneteket. *Például:* hibajelzés, adatkérés, adatbevitel.

Partíció (partition): A memória vagy merevlemezes tároló folytonos területe, amely külön logikai egységként kezelhető.

PC (Personal Computer) vagy személyi számítógép: Olyan mikroszámítógép, amely általában egy felhasználó számára készült.

Periféria (perifherial unit): Olyan egység, amely a számítógépes rendszer külső adatforgalmát végzi. A számítógép és a felhasználó közti kapcsolattartás eszköze. A beviteli perifériákon keresztül jutnak az adatok, a programok a számítógépbe, míg a kiviteli perifériákon jelennek meg az eredmények és a számítógép üzenetei.

Plotter (lásd rajzgép)

Printer (lásd nyomtató)

Prioritás (priority) (lásd műveleti sorrend, vagy műveleti precedencia)

Processzor (processor) vagy központi feldolgozó egység: Olyan egység, amely értelmezi és végrehajtja a megadott parancsokat.
Program (program): Utasítássorozat, amely meghatározza egy probléma megoldásának lépéseit, és ezt a számítógép által végrehajtható formában jeleníti meg.
Programbelövés (debugging) vagy tesztelés: Olyan eljárás, melynek során a kész programot megfelelő próbaadatokkal futtatjuk, hogy megtaláljuk és kijavítsuk az előforduló hibákat.
Prompt (prompt) vagy készenléti jel: Jelzés a képernyőn, amely tudatja a felhasználóval, hogy a számítógép készen áll a parancsok fogadására és azok végrehajtására, kezdődhet a munka.
Programlista (program list): Kinyomtatott információ, amely a számítógépprogramot tartalmazza.
Puffer (buffer): A perifériák tárolója.

Rajzgép (plotter): Kiviteli periféria, amely az adatokat kétdimenziós, grafikus formában ábrázolja.
RAM vagy írható- olvasható tár vagy véletlen elérésű tár regiszter (register): Áramköri egység, amelyben meghatározott méretű adatot tárolhatunk. Kiemelt szerepű tároló.
Rekord (record): Olyan adatok halmaza, amelyek egymással valamilyen kapcsolatban állnak, ezért egységként kezelhetjük őket. *Például:* a személyi adatok.
Rendszerprogram vagy rendszerszoftver vagy működtető rendszer vagy operációs rendszer: Olyan programok összessége, amelyek összehangolják a számítógép hardverének működését, és biztosítják a felhasználó és a számítógép közti elemi kommunikációt.
ROM vagy olvasható tár vagy nem törölhető tár rögzített mágneslemezegység vagy merevlemezes tároló rutin (routine): Számítógépes program vagy programrészlet, amelyet a program futása közben többször felhasznál.
Sáv (track): Koncentrikus kör a mágneslemez felületén, amelyen az információt tárolja. Adatok olvasása vagy rögzítése esetén egyetlen olvasó- vagy írófejjel érintkezik.
Segédprogram (utility): Olyan program, amely segíti a számítógépi program tesztelését, ellenőrzését vagy végrehajtását.
Shift (lásd váltó)
Soremelés (line feed): Nyomtatás közben végzett tevékenység, amikor egy gomb lenyomása után egy sorral lejjebb folytatódik az információk kiírása.
Sorgörgetés (scroll): Olyan művelet, melynek során a képernyőn lévő információkat egy sorral feljebb vagy lejjebb csúsztatjuk. Ebben az esetben eltűnik a mozgás irányába eső első sor.
Sornyomtató (line printer): A papírra egyszerre egy teljes sort nyomtat, ezért csak karakterek nyomtatására alkalmas, grafikus ábrákéra nem.
Soros hozzáférésű tároló (serial-access storage): A benne található információkat csak egymás után, sorban tudjuk kiolvasni. Ha egy keresett adat előtt másokat találunk, akkor el kell olvasni az összes megelőzőt, hogy megkapjuk a kívánt információt. *Például:* mágnesszalagos tároló.
Sorszám (line number) vagy utasítássorszám: Olyan szám, amelyet egyes programnyelvekben az utasítássorok elé írunk, hogy ezzel jelezzük az utasítások végrehajtásának sorrendjét. (Ez a sorszám lehet az utasítások címkéje is.)
Sorzáró billentyű (lásd lezáró billentyű)

Számbillentyűzet (lásd numerikus billentyűzet)
Számítógép (computer): Adatfeldolgozásra alkalmas gép, amely egy előre elkészített program alapján, emberi beavatkozás nélkül, egyedül végzi a műveletek sorozatát.
Számítógép-hálózat (computer network): Számítógépek összekapcsolt rendszere, amelyben az egyes munkaállomások csak valamilyen hozzáférési eljárással érhetik el a hálózat kijelölt gépein lévő erőforrásokat. Tehát a hálózaton belül az egyes felhasz-

nálók nem férhetnek hozzá a hálózat gépein lévő minden adathoz. Az egyes szintek adataihoz jelszavak beírásával juthatunk.

Számológép (calculator): Elektromechanikus vagy elektronikus gép, amely matematikai műveletek elvégzésére alkalmas. A fejlettebb változatok programozhatók, illetve képesek néhány adat tárolására is.

Százalékjel (percent sign): Olyan karakter, amely a BASIC nyelvben az egész változót jelöli a változó mögé írt utolsó karakterként.

Szektor (sector): Olyan körcikk a mágneslemezen, amely azonos középponti szöghöz tartozik. A szektor tehát a sávnak egy olyan szelete, amelyen meghatározott számú (512) bájt található.

Szemantikai hiba (semantical error): Olyan hiba, amely az adatok tartalmából vagy hibás összefüggéseiből adódik.

Személyi számítógép (lásd PC)

Szerkesztőprogram (editor): Programozási eszköz, amely segít a programok begépelésekor és javításakor.

Szintaktikai hiba (syntax error): Olyan hiba, amely az adott programnyelv szabályainak helytelen alkalmazásából adódik. Adódhat az adatok, vagy programsorok hibás beírásából. *Például:* számítógépünk egész értékű adatot vár, és mi törtet írunk be.

Szoftver (software): Olyan programrendszer, amely a számítógépet működteti. Magában foglalja a számítógépes programokat, eljárásokat, szabályokat, és az ezekhez kapcsolódó dokumentációkat is.

Szóköz (space): Egykarakternyi hely a képernyőn, amely üres. Ha a karakterek között betűközt akarunk írni, akkor nyomjuk le ezt a billentyűt. A tárban elfoglal egy karakternyi helyet, a kiírási képen pedig azt tapasztaljuk, hogy a kurzor egy hellyel jobbra lép. Jelölésére jelet alkalmazhatunk.

Szóközbillentyű (space kay): Lenyomása után a kurzor egy hellyel jobbra lép, és helyén üres karakter marad.

Szövegszerkesztés (text editing): Olyan program, melynek segítségével szöveges állományokat hozhatunk létre, vagy módosíthatjuk a meglévők tartalmát.

Szubrutin (subroutine) vagy alprogram: Olyan programrészlet, amelyet a program futása közben egységként kezel, és szükség szerint többször is felhasznál, esetleg újabb programokban is alkalmazhatjuk.

Tabulátor (tab): Nyomógomb a billentyűzeten, melynek segítségével beállíthatjuk a beírandó szöveg hosszát. A billentyű lenyomása után a kurzor a tabulátorbeállításnak megfelelő helyre áll.

Tár (storage, memory): A számítógép egysége, amelyben elhelyezhetjük az információkat, ahol megőrizhetjük őket, és szükség esetén kivehetjük belőle, hogy felhasználjuk.

Tárkapacitás (storage vagy memory capacity): Az a legnagyobb adatmennyiség, amelyet betölthetünk a tárba.

Tárolt program: Felfedezése Neumann János nevéhez fűződik. Javasolta, hogy az adatokhoz hasonlóan a futó programot is a számítógépben tárolják.

Tartalomjegyzék (directory) vagy könyvtár: Mágneslemezre rögzített jegyzék, amely tartalmazza a rajta tárolt adatállományok nevét, méretét, típusát.

Távadatátviteli egység (lásd MODEM)

Távfeldolgozás (teleprocessing): Olyan adatfeldolgozási módszer, ahol a számítógép és a perifériák földrajzilag különböző helyeken vannak elhelyezve. Az adatátvitelt hírközlő csatornák végzik. *Például:* telefonvonalak, műholdas összeköttetések.

Terminál (terminal): Olyan periféria, amely lehetővé teszi, hogy a felhasználó kétirányú kapcsolatot tarthat fenn a számítógéppel.

Tizedespont (decimal point): Jel, amely az egész számot és a hozzátartozó tizedes törtet össszekapcsolja. Matematikában a tizedesvesszőt használjuk ilyen célra.

Tesztelő program: Olyan program, amely egy adott program hibáinak kiszűrését, tesztelését segíti.

Tömb (array): Azonos tulajdonságú adatokat tartalmazó információk összessége, amelyeket indexeik egyértelműen megkülönböztetnek egymástól.

Tömbdeklarálás (array declaration): Olyan szám megadása, amely megmutatja, hogy az adott tömbbe maximálisan hány adat helyezhető el.
Törlés (erase): Eljárás, melynek során az adathordozóról eltávolítjuk az adatokat, hogy helyette újabbakat írjunk rá.
Törlést segítő billentyű: Lenyomása után törlődik az a karakter, amelyen a kurzor állt (del = delete).
Tömörített lista: Rövidített lista, amellyel kiírathatjuk a képernyőre egy adott alkönyvtár tartalmát úgy, hogy csak az állománynevek szerepeljenek benne. Az öt oszlopban elhelyezkedő nevek rövid áttekintést adnak az alkönyvtár tartalmáról, és így ötször annyi név fér a képernyőre, mint egyéb kiíratáskor. Ezt a **dir/w** parancs idézi elő.
Túlcsordulás (overflow): Olyan jelenség, amely akkor fordul elő, ha egy művelet eredménye nem fér el az adott tárhelyen.

Ugrás (jump): Olyan programlépés, amikor a program futása nem a számok által előírt módon folytatódik.
Ugrást előidéző billentyű: Lenyomása után a kurzor a szöveg elejére vagy végére ugrik, és ott folytathatjuk a beírást.
Utasítás (instruction vagy statement): Olyan információ, amely a programban műveleteket jelöl ki, és a programon belül egy önálló egységet alkot.
Útvonal (path): Könyvtárak láncolata, amely pontosan leírja, hogy hogyan juthatunk el a kívánt alkönyvtárba.
Üres utasítás (blank instrucktion): Csak a következő utasításra történő továbblépést idézi elő.
Üzemkész állapot (in working order): A DOS betöltődött, a rendszer bejelentkezik, a felhasználó elkezdheti a munkát. A képernyőn megjelenik a prompt, és mögötte ott villog a kurzor.

Váltóbillentyű (shift): Segítségével felcserélhetjük a billentyűzeten található karaktereket.
Változó (variable): Olyan adat, amely bármilyen értéket felvehet bizonyos értékhatárok között. Az aktuális értékét mindig a program futása közben adjuk meg.
Végtelen ciklus: Olyan ciklus, amely csak külső beavatkozás hatására fejeződik be.
Véletlenszám-generátor (random number generator): Olyan algoritmus, amely valamilyen kezdőértéktől indulva, bizonyos értékhatárok között egymástól látszólag független számokat állít elő.
Vezérlésátadó utasítás (branch statement): Hatására a vezérlés átadódik egy másik utasításra, így a program futása nem a soron következő utasításon folytatódik.
Vezérlőbillentyűk (control key): Olyan billentyűk, amelyek leütése után a képernyőn nem jelenik meg látható karakter, hanem egyéb, a gép vagy a billentyűzet működését befolyásoló hatást fejt ki. *Például:* a sorzáró ENTER.
Vírus (vírus): Önműködően terjedő „kórokozó", amely károsítja a gépen található programokat és magát a számítógépet is.

Winchester vagy merevlemezes tároló.

A könyvünkben található programok

Tartalom

Nemzeti Tankönyvkiadó
A kiadásért felel: dr. Ábrahám István igazgató
Raktári szám: 53 281
Felelős szerkesztő: Zankó Istvánné
Műszaki vezető: Héjjas Mária termelési igazgatóhelyettes
Műszaki szerkesztő: Hülber Péter
Terjedelem: 22.16 (A5) ív
Kiadás száma, éve: első kiadás, 1993
TA–5447–II/C–7 9395
Készült a Borsodi Nyomda Kft.-ben
Felelős vezető: Ducsai György ügyvezető igazgató

Ezt a könyvet eredményesen lapozhatják azok, akik

- önállóan, külső segítség nélkül szeretnének számítástechnikai ismereteket szerezni,
- a tanításhoz keresnek segédanyagot,
- a zsebszámológépek alkalmazásának lehetőségeit tanulják,
- nem ismerik a Commodore 64 és az IBM PC kompatibilis számítógépeket,
- ismerik a C–64-et és a PC-t,
- nem ismerik a BASIC nyelvet és a QUIK BASIC-et,
- jól ismerik a BASIC nyelvet, de nem találkoztak még a QBASIC-kel,
- a PC-k operációs rendszeréről, az MS-DOS 5.0-ról szeretnének többet tudni,
- programokat akarnak másolni könyvtárukba,
- nem értik a számítógépek hibaüzeneteit,
- a számítástechnikai fogalmak rövid, pontos meghatározásait keresik.

Egyszóval mindenki találhat magának olyan részt, melyet hasznosítani tud, és érdeklődését kielégíti.

* * *

Kezdők számára készült:

Fekete Sándorné: *Számítógép-kezelés az alapoktól*

Tankönyv és feladatgyűjtemény az MS-DOS 5.00-ról

című könyve is.

Segítségével a PC-vel rendelkezők önállóan is megkezdhetik a saját gépükkel való ismerkedést. Gyakorlófeladatainak megoldása is benne van a könyvben, így ellenőrizhetik is önmagukat.

518 Ft (áfával) 53 281