SEGÉDKÖNYV
A TV-BASIC

oktatócsomaghoz

Készítette: Hegyi István

S E G É D K Ö N Y V
A T V - B A S I C
oktatócsomaghoz

Készitette: Hegyi István

SEGÉDKÖNYV A TV BASIC OKTATÓCSOMAGHOZ

B e v e z e t ő

Köszöntöm a Kedves Olvasóinkat. Amikor leirom ezt a szót mindjárt kétségeim is támadnak, helyes-e az olvasóink megszólitás, hiszen ugyanennyi erővel Nézőinknek, program felhasználóinknak, vagy tanulótársainknak vagy egyszerre mindegyiknek is szólithatnám Önöket, a TV BASIC sorozat oktatócsomagjának felhasználóit.

Néhány szót a TV BASIC akcióról, amely a SZÁMALK, a Neumann János Számitógéptudományi Társaság és a Magyar Televizió együttmüködésével jött létre, és amely lényegében Magyarországon az első sikeres távoktatási vállalkozás.
A tv adásokból és a TV BASIC tankönyvből több százezren ismerkedtek a számitógép programozás tudományával, és közel tizezren jelezték, hogy vizsgázni is szándékoznak a tanfolyam végén.

Jogos igény, hogy a TV BASIC anyaga, a televiziós sugárzás befejeztével is hozzáférhető legyen a programozással, a BASIC nyelvvel; ismerkedni kivánó egyének és közösségek számára. Ennek az igénynek teszünk eleget, amikor a TV BASIC oktatócsomagot közreadjuk.

AZ OKTATÓCSOMAG HASZNÁLATÁRÓL

A TV BASIC oktatócsomag tartalmazza a televiziós adássorozat 16 müsorát videokazettán, a tankönyvet, az adásokban bemutatott és egyéb minta programokat s mágneses adathordozón, tehát magnokazettán, vagy mágneslemezen közvetleniil a számitógépbe tölthető formában, végül ezt a segédkönyvet.

Az oktatócsomagnak egyéni és kollektiv használata egyaránt lehetséges.
Egyéni tanulásnál a következő munkamenetet javasoljuk.
A segédkönyv egy-egy fejezetéből megismerhetjük az adások témáját, tartalmát. A tanulási tanácsok utalnak arra, hogy van-e szükség előzetes felkészülésre a müsor megtekintése előtt.
Ezt követően nézzük meg az adást, majd olvassuk el a könyv javasolt fejezeteit, válaszoljunk a kérdésekre, próbáljuk megoldani a feladatokat; lehetőleg a gyakorlatban, számitógépen.
Ha megakadtunk, segithet az adás ismételt megnézése, a fogalmak ismételt átnézése, és nem utolsó sorban a kész mintaprogramok gépbetöltése és tanulmányozása. A programok betöltésére a 2.sz. adásban találunk eligazitást.
Igy mentesülünk a programok begépelésének sokszor nem kevés fáradsággal járó munkájától - figyelmünket a lényeg megértésére koncentrálhatjuk.
Másrészt az adathordozón mellékelt programok /pl.notesz, játékok, adatállomány kezelő/ később kialakitandó program könyvtárunk alapját adhatják.

A kollektiv felhasználás esetén szükségessé válik a tankönyv és a számitógépes adathordozón lévő mintaprogramok több példányos beszerzése.

Ilyenkor a segédkönyv a csoport vezetőjének vagy oktatójának munkáját hivatott segiteni. Tehát ő közvetitse a megfelelő instrukciókat a csoport tagjainak, tanulásuk irányitására.
A sorozat valamennyi alkotója,közreműködője,az oktatócsomag valamennyi készitője nevében eredményes tanulást, sikeres és örömteli programozást kiván Önöknek

Hegyi István
szerkesztő

I. A L A P O K

Az adás vázlatos tartalma:

Köszöntjük a kedves nézőket, azokat, akik elhatározták, hogy sorozatunk segitségével megtanulják a BASIC nyelvet.

S ha már elhatározták, nem árt ha megmondjuk mi is az, amivel megismerkednek. A BASIC olyan nyelv, amelynek segitségével a számitógépekkel lehet kommunikálni. A név egyébként rövidités, a Beginner's All-Purpose Symbolic Instruction Code /Kezdők általános célu utasitás-kódja/ angol elnevezés kezdőbetüiből áll.

Mint a nevéből is kiderül, kezdők számára készült és mint ilyen, viszonylag gyorsan elsajátitható. Nem véletlenül választottuk tanfolyamunk témájául. Ez a számitógép-nyelv a magyar származásu John G.Kemény munkájának eredményeként kb. 2o éve indult utjára és azóta a kezdők és a számitógéppel csak hobbyból foglalkozók legnépszerübb, legelterjedtebb programozási nyelve lett.

Manapság nem gyártanak olyan személyi számitógépet,amely BASIC nyelven ne lenne programozható.
Azt igértük a cimben, hogy az alapokkal foglalkozunk. Hogy mégse az egyszeregynél kezdjük, egyezzünk meg, a négy alapmüvelet ismeretét feltételezzük, de ennél sokkal többet nem
A legalapvetőbb kérdés persze az, hogy mi is az, amivel a BASIC nyelven kommunikálunk, vagyis miféle szerkezet a személyi számitógép?

A személyi számitógép egy doboz, amely önmagában, annak ellenére, hogy elég sok mindent tud, szinte semmire sem használható.

A számitógép elvégez bizonyos müveleteket, de szükséges hozzá egy olyan eszköz pl. egy billentyüzet, amivel meg tudjuk mondani neki, hogy mit csináljon, továbbá kell kijelző szerkezet /pl. egy tv-készülék/, ahol a számitógép üzenetei és az eredmények megjelennek.

Ez a minimális eszközkészlet, amivel a személyi számitógépet használni lehet.

Kapcsoljuk be a gépet, amelyet az egyszerüség kedvéért egybeépitettek a billentyüzettel. /Ez a személyi számitógépeknél elég gyakran előfordul./

Kisérelјük meg a kommunikációt a géppel.
Legelöször bekapcsoljuk a számitógépet, ezzel a kommukáció már meg is kezdődött, hiszen a gép OK. felirattal jelzi, hogy müködésre kész.

Kezdjük a társalgást!

Gépeljük be: Jó napot!

Ilyenkor hibajelzést kapunk, ugyanis az ERROR ezt jelenti. Ez angol szó, próbáljuk meg talán angolul. Bár ez egy magyar gép. /PRIMO/ HELLO /A gép válasza ugyanaz, mint az előbb./
Ebből a kis játékból azért megtudhatunk néhány dolgot. A személyi számitógépek BASIC nyelve angol szavakat használ, de rögtön hozzátesszük, hogy nem szükséges angolul tudni ahhoz, hogy valaki a BASIC-et megtanulja. A BASIC nyelv másik igen lényeges tulajdonsága, hogy meglehetősen kötött nyelvtana /szintakszisa/ van. Pl. csak bizonyos szavak használata megengedett: a jó napot és a hello nem tartozik ezek közé.

Nézzünk néhány elfogadott BASIC szót. Például: PRINT, GO TO, READ, INPUT stb.
Nézzük pl. az egyik legegyszerübbet./PRINT 2x2, Alatta megjelenik: 4/ Ebből ugy tünik, még az egyszeregyet sem kell tudnunk, mindig meg lehet kérdezni a géptől.
Persze nem csak az egyszeregyet, hanem ennél bonyolultabb dolgokat is.
Ez az un. kalkulátor vagy parancs üzemmód. Ha a PRINT szó után beirunk egy matematikai kifejezést, akkor a gép azt kiszámitja és az eredményt irja ki.

A PRINT - most már nevezzük igy - paranccsal nemcsak számitások eredménye, hanem szövegek is kiirathatók. Ilyenkor a parancs formája: PRINT "TV BASIC".

A parancsot és utasitásokat mindig RETURN, NEW LINE, ENTER megnyomásával kell lezárni. Innen tudja a gép, hogy más dolgozhat, végrehajtja a parancsot. Ez azért még nem programozás.

A számitógép ugyanis ennél többre is képes, az utasitásokat meg is tudja "jegyezni" vagyis tárolni tudja őket, majd adott sorrendben egymás után végre is tudja hajtani az utasitásokat.
Erről később még részletesen beszélünk.

Bizonyára mindenkit érdekel azonban, hogy mitől képes a gép minderre. Talán okosabbak leszünk, ha belenézünk a gép belsejébe is. Ez jó alkalom arra, hogy megismerjük a számitógép főbb részegységeit.
A gép lelke a napjainkban oly sokat emlegetett mikroprocesszor. A processzor vezérli a számitógépet, elvégzi a müveleteket, figyeli, hogy melyik gombot nyomtuk meg, eldönti, hogy milyen legyen a válasz. Egyszóval ő a főnök.

Ahogyan egy üzemben az üzemvezető nem tud mindent "fejből", ugy a processzor sem tud önmagában semmit pl. a BASIC nyelvről, de a tizes számrendszerről sem.

A tudnivalókat egy másik áramkör - a csak olvasható memoria, az u.n. ROM tároló tartalmazza, innen olvassa ki a processzor, hogy melyik helyzetben mit kell tennie. Az előbbi hasonlattal élve a ROM tárolót nevezhetjük az üzem technológiai utasitásának, vagy ha tetszik, törvénykönyvének.

Egy másik tárolóban a processzor szempontjából változandobb dolgok tárolhatók, ilyenek pl. a tv. kép tartalma, a mi utasitásaink, számok, részeredmények stb. Ez az u.n. RAM tároló, ebbe irni is lehet, nem csak olvasni belőle, mint pl. a főnök jegyzetfüzetébe.

Szükség van még a különböző egységek müködését összehangoló, a külvilággal kapcsolatot teremtő kapu áramkörökre, a tv készülék számára megfelelő jelet létrehozó modulátorra és egy - az áramellátást biztositó - tápegységre.

Mindez eddig meglehetősen egyszerünek tünik, azonban ha belegondolunk, hogy mondjuk a mikroprocesszor, vagy a memoria néhányszor tizezer tranzisztort tartalmaz, egy kissé elbizonytalanodhatnak.

Pedig az alapok - és mi most ezzel foglalkozunk - itt is egyszerüek. Az elektronikus áramkörök elemei számára két állapot létezik: folyik áram vagy nem folyik áram, jelen van egy bizonyos feszültség egy adott ponton vagy nincs jelen. A kétféle állapot időbeni, vagy térbeli konbinációjával - ugyanugy, mint a MORSE-ABC-vel -

kódolhatók a számok, a betük,sőt még a mikroprocesszor utasitásai is.

8 db ilyen kétállapotu valaminek az együttesével összesen 256 különböző lehetőség adódik, ez manapság bőségesen elegendő a kis- és nagybetük, a számok és még sok minden más kódolásához.

Napjainkban elterjedt mikroszámitógépek processzorai lényegében ilyen nyolcelemü jelcsoportokkal végeznek müveleteket.
Tegyük hozzá, óriási sebességgel.
Azt mondják, mindez kapcsolódik a kettes számrendszerhez. Igy igaz. Aki nem hiszi, lapozza utána az általános iskolások matek könyvében, vagy próbálja beleélni magát egy olyan, mondjuk marslakó lelkivilágába, akinek mindössze két ujja van és ettől, amikor számol, mindent kettesével csoportosit. Két tárgyat egy nagyobb kettes csoportba, két kettes csoportot egy négyes csoportba, két nagyobb csoportot egy mégnagyobba von össze.

Könnyü észrevenni, hogy a számok felirásához neki mindössze két számjegyre van szüksége.
Mindez talán kissé fárasztónak tünik, de nem haszontalan pl. annak megértéséhez, hogy milyen módon vezérelhetünk számitógéppel és BASIC programmal diavetitőt, magnetofont, kávéfőzőt, esztergát, köszörüt, robotot, vagy videoképet előállitó szerkezetet.

Ha eddig csak a kiváncsiság ösztönzött volna bennünket a számitógéppel való megismerkedésre, most már gyakorlatibb,mondhatni kényelmi szempontjaink is lehetnek.

Még megéljük, hogy számitógép ugraszt ki bennünket reggel az ágyból, főz kávét, vagy enged fürdővizet, vagy netán kitakaritja a lakást, sőt esetleg ránk is ripakodik, ha a szőnyegre hamuznánk.

Persze itt még nem tartunk, de már ma is vannak olyan pillanatok, olyan munkahelyek, olyan emberek, amikor, ahol és akiknek igen-igen nagy segitséget jelent egy számitógép és nem utolsó sorban egy jól megirt számitógép program.

Pl. tegyük fel, hogy rendkivül sokoldalu, elfoglalt és mellé még egy kicsit feledékeny emberek vagyunk.Mindent felirunk a határidő naplóba, de az időnk jelentős része megy el azzal, hogy végiglapozzuk.
Mi történik, ha mindezt számitógépre visszük? A keresés néhány perc, esetleg másodperc.

Vagy adminisztrátorok vagyunk és különféle mutatók, adatok lepedőnyi táblázatát kell karbantartanunk.
Rendszeresen összeadjuk az egyes oszlopok adatait. Mikor minden stimmel, jön a főnök és közli, hogy néhány adat megváltozott.
Kezdhetjük előlről.
Ilyenkor segit a táblázatkezelő program.
Azon már nem is lepődünk meg, hogy a személyi számitógépek nagy részével zenélni is lehet. Ilyenkor de nem csak ilyenkor nagyon fontos a billentyükezelés.
Erről, vagyis a számitógép kezeléséről lesz szó a következő adásban.

TANÁCSOK A TANULÁSHOZ

Az első - lényegében véve kedvcsináló - adás megtekintése előtt tehát nincs teendőnk.
Utána viszont tanulmányozzuk át a tankönyv bevezetőjét!
Készitsük el az itt szereplő szorzatkitaláló programot.
A 86. és 87. oldal alapján gondoljuk át a PRINT utasitás használatát és próbáljunk meg a képernyő különböző helyére betüket és számokat irni a PRINT utasitás különböző formáinak segitségével.

II. U J J G Y A K O R L A T

Az adás vázlatos tartalma:

Ha kettesben maradnak egy számitógéppel, legelső probléma, hogy miként keltsék életre. Ez a folyamat sajnos nem zajlik olyan gyorsan, mint a müveletek a számitógépben, ez egy-két percet is igénybe vesz.

Legelőször is a tv készüléket kapcsoljuk be, majd az erre rendszeresitett csatlakozó zsinort az antenna helyére dugjuk be, a másik végét pedig a számitógép tv jelzésü kimenetére.
Ezután már csak a számitógépet kell ellátnunk árammal. Ez egy vagy két lépésben történhet, attól függően, hogy a számitógépben beépitett tápegység van vagy pedig külön tápegységet adnak hozzá.

Nem kell félni, általában eltéveszteni sem lehet, mert két egyforma végü csatlakozó nincs. Tehát mindegyik csak a saját helyére megy be.

Ha még azt előttünk valaki nem tette meg, a géptipusnál előirt csatornára hangoljuk a tv-t. /Ez általában a 36-39-es csatorna környékén van./

Ha mindent jól csináltunk és minden berendezésünk hiba nélkül üzemel, valaminek meg kell jelenni a képen. Vagy egy felirat, vagy egy villogó jel mutatja. hogy a számitógép munkára kész. Ez a kezdő számitógépes első sikerélménye.

Kezdő számitógéptulajdonosok, vagy programozók első lépése szokott lenni, hogy könyvből, kiadványból vagy is-

merőseiktől beszereznek egy nem tul hosszu és elég izgalmasnak itélt programot és megkisérlik becsempészni a saját gépükbe.
Ilyenkor találják szemben magukat elöször a számitógép billentyüzetével.
Egy program általában igy kezdődik: 1Ø REM akármi és a bonyodalmak is. Ugyanis a számitógép billentyüzete nagyon hasonló az irogéphez, de nem azonos. A normál irógépen megszoktuk, hogy az l-et az l betü helyettesiti, az O pedig a nullát. Ha itt is ezt a gyakorlatot követjük, a számitógép ezt rossz néven veszi, hiszen számára a lO nem azonos a 1Ø-zel.

Mindenesetre érdemes megjegyezni, hogy a számoknak és betüknek külön billentyüje van a számitógépen.
A számitógépek billentyüzete nagyjából hasonló, általában csak apróbb részletekben térnek el egymástól.
A billentyüket három csoportra oszthatjuk:

- karakterbillentyük: benyomáskor a rájuk rajzolt jel /betü vagy szám/ jelenik meg a képernyön,

- funkconális billentyük: lenyomásukkor valamilyen müveletet végez el a számitógép, pl.: CLS/CLR gomb a törlőrongy szerepét játssza, letörli a képernyőt,

- vegyes használatu billentyük: önállóan használva karakterbillentyük, valamely funkcionális billentyüvel együtt lenyomva a funkció pontositására szolgálnak.

A gépek jelentős részénél a billentyüzet automatikus ismétlési funkcióval is rendelkezik, vagyis ha egy megadott időnél tovább nyomjuk a billentyüt, a gép automatikusan ismétli addig, ameddig a billentyüt nyomva

tartjuk. Tehát nem tanácsos elábrándozni.

A billentyük igénybevételéről nem készült statisztika. de biztos hogy a RETURN, NEW LINE. vagy ENTER feliratu gomb a leggyakrabban használtak egyike.
Ennek megnyomásával értesül ugyanis a számitógép arról, hogy amit egy sorba pötyögtünk, komolyan gondoltuk-e. Ha megnyomjuk ezt a billentyüt. a gép elkezdi feldolgozni és értelmezni az adott sort és tartalmától függően végrehajtja vagy tárolja - vagy hibajelzést ad, ha zöldséget irtunk.

Megnyugtatásul elmondjuk, hogy bármilyen gombot nyomunk meg, bármilyen butaságot követünk el, ezzel a gépet nem tudjuk elrontani.

Klasszikusnak nevezhető játékprogram az u.n. emberevő, amelyben egy emberevőnek nevezett folt a képernyön üldöz bennünket, azaz egy másik foltot vagy csillagot, amelyet mondjuk a billentyüzeten keresztül irányitani tudunk. Próbáljuk ezt a programot beirni a gépbe. Megtalálható a Mikromagazin első számában.

A számitógép a billentyüzeten bevitt utasitásokat tárolni is tudja és ujabb parancsra adott sorrendben végre is tudja őket hajtani.
Honnan tudja a gép, hogy rögtön végrehajtandó parancsról vagy tárolandó, megjegyzendő utasitást közöltünk-e vele?

Ha a sort egy számmal, sorszámmal kezdjük, akkor ezt a sort a gép utasitásként értelmezi, és a RETURN. ENTER, NEW LINE gomb megnyomásakor nem végrehajtja, hanem megvizsgálja. átforditja a maga nyelvére és tárolja. A sor

elején lévő szám azután a végrehajtás sorrendjét is meghatározza, ha a RUN + sorzárás parancsot begépeljük.
A program tehát sorszámmal ellátott utasitások sorozata, amelyet a számitógép végrehajt.

A képernyőn mindig van egy jel, amely megmutatja azt a helyet, ahol a következő bebillentyüzendő karakter meg fog jelenni. Ennek a neve a helyőr, magyarul kurzor.

Három gépnél, a HT-nél, a PRIMO-nál és a COMMODORE-nál megkezdhetjük a gépelést.
Amig még nem nagyon értjük miről van szó, lehetőleg pontos másolásra törekedjünk. /Persze a pontosság később is követelmény./

A SINCLAIR gépeknél kicsit más a helyzet. Itt hiába kisérelj̈ük meg betünként beütni a REM szót a 1Ø-es után az R betü leütése után a gép makacsul a RUN szót irja ki, hiába töröljük ki a DEL billentyüvel és irjuk ujra.

Ennek oka az, hogy a SINCLAIR gépek billentyüi szinte kivétel nélkül funkcionális billentyük is. Hogy éppen milyenek, az attól függ, hogy milyen tipusu az éppen érvényes kurzor.
Létezik K/keyword/, L/letters/, C/capitals/, E/extended/ F/function/, G/graphics/ tipus. Hogy melyik él, azt pedig az előtte bebillentyüzött szöveg, valamint az határozza meg, hogy milyen váltó billentyüt használunk még.

Általában minden géptipuson vannak a kurzort mozgató és törlő billentyük is, amelyeknek segitségével egy éppen beirádó soron belüli javitás minden további nélkül meg-

oldható.
COMMODORE-nál a korábban beirt sorokban is. A többi gépnél erre az EDIT parancs szolgál, amely előhivja az elrontott sort, igy elvégezhető a javitás. Fontos megjegyezni, - ugy mint a katonaságnál - az utolsó parancs az érvényes.
Magyarul, ha ugyanolyan sorszámmal ujrairunk vagy javitunk egy sort és azt sorzáró paranccsal lezárjuk, az uj sor felülirja, törli a régit, tehát a régi sor megszünik létezni.

Érdemes megjegyezni, hogy a számitógépgyártók általában a gép használatát oktató programot is mellékelnek a géphez. Ilyenre az adásban is látunk példát.

A program leirása után a LIST paranccsal kaphatjuk visszza a program listáját. Ezután pedig az EDIT parancs segitségével, vagy a sorok ujrairásával eszközölhetünk további javitásokat.

Az igy esetleg nem kis fáradsággal elkészült programnak van egy szépséghibája. Az, hogy ha csak egy pillanatra is megszakad a számitógép áramellátása, azonnal elfelejti a programot és kezdhetjük az egészet elölről. Az effajta balesetekről a programozók rémtörténeteket tudnak mesélni.
Arról a kellemetlenségről nem is beszélve, hogy nem tul élvezetes dolog egy programot mindig ujra begépelni, ha szükség van rá.

Erre is találtak megoldást a személyi számitógépek tervezői és gyártói.
A SAVE /CSAVE/ paranccsal a gép tárolójából a program

magnetofon szalagra. komolyabb készülékeknél mágneses lemezre rögzithető és onnan visszatölthető a memóriába.

A magnetofont az erre a célra mellékelt vezetékpárral kössük össze a számitógéppel. Itt is minden csatlakozónak megvan a párja és a helytelen csatlakozással sem okozhatunk kárt a gépben, legfeljebb bosszuságot magunknak.

Állitsuk a szalagot üres vagy átirható helyre. Irjuk be a SAVE "program név" parancsot. Az idézőjelek közé tett szöveg lesz a szalagon a program neve. Legyen most "EMBEREVŐ".
Inditsuk el a magnetofont "felvétel" állásban. majd nyomjuk meg a sorzáró gombot.

A gépek általában jelzik az átvitel folyamatát,valamint az átvitel végét.
Ilyenkor állitsuk le a magnot.
Ha kiváncsiak vagyunk, hogy miféle jeleket rögzitettünk a magnószalagra, tekerjük vissza a szalagot, szakitsuk meg a magnó kapcsolatát a számitógéppel és játsszuk le a felvételt- és meghalljuk a "rekedt kakas"-t.

Kisérelјük meg most visszatölteni a programot. Álljunk ujra a felvétel elejére. Irjuk be a NEW parancsot, vagy kapcsoljuk ki a gépet. Ekkor a benne lévő programot megsemmisitjük.
Csatlakoztassuk ujra a magnot a géphez. Irjuk most be a LOAD "EMBERVO" /HT-nél CLOAD/ parancsot és inditsuk el a magnot lejátszással. /Figyelem, a SINCLAIR gépeknél más a csatlakoztatás módja felvételnél és lejátszásnál/.
Üssük le a sorzáró billentyüt.

A gépek most is jelzik az átvitelt és a sikeres befejezést.

A gépbe vitt programot a LIST paranccsal láthatóvá tehetjük, feltéve ha minden rendben volt.
Ha nem, jöhet az ujragépelés.
Ezt a kellemetlenséget elkerülendő, egyes gépeken létezik egy VERIFY nevü parancs, amelynek hatására a gép visszajátszáskor összehasonlitja a még a memóriájában lévő programot azzal, amit a magnoról visszaolvasott.
Ha nincs rendben. hibajelzést ad, és a mentést meg lehet ismételni.

Hasonló a programok mentése az u.n. mágneslemezes háttértárolók esetében is pl. a COMMODORE-nál.
A mentő, betöltő ill. ellenőrző parancsok itt a mágneslemezes egység u.n. periféria számával bővülnek, továbbá a lemez tárolási tulajdonságaiból adódóan könnyen megkaphatjuk az egy lemezen tárolt programok u.n. katalogusát.

Ezt a tudományunkat máris kamatoztathatjuk, ugyanis nem csak mi irunk programokat, hanem mások is. Az azonos géptipusok kazettán rögzitett programjai egymás között cserélhetők.
Vagyis egy ZX-en rögzitett program betölthető egy másik ZX-be és igy tovább. Igy csereberével vagy másolással müködő programokhoz juthatunk.

Az igy hozzánk került programok listájának tanulmányozásával értékes ötleteket, megoldásokat leshetünk el.

TANULÁSI TANÁCSOK

Már az adás megtekintése előtt célszerü átolvasni a tankönyv második fejezetét, majd a müsor megtekintése után ujból elővenni főleg azokat a részeket, amelyek az általunk használt géptipusra vonatkoznak.

Ellenőrizhetjük tudásunkat a könyv 48. oldalán lévő kérdések és feladatok megoldásával.
Bőséges manuális gyakorlás is szükséges a billentyük kezelésének elsajátitásához, a programok szerkesztésének, javitásának megtanulásához.

III. F E J G Y A K O R L A T

Az adás vázlatos tartalma:

Ebben az adásban elkészitjük első saját programunkat. Természetesen ne gondoljunk több kilométer hosszu, bonyolult feladatot megoldó programra.
Mielőtt hozzákezdenénk, gondoljuk végig mi mindent várunk egy ilyen számitógépprogramtól.
Elöször is, hogy legyen univerzális, vagyis a feladatot ne csak egyetlen alkalommal, egyetlen szituációban legyen képes megoldani. A másik lényeges dolog pedig, hogy lehetőleg legyen önmüködő. Vagyis a RUN parancs beadásán kicül ne igényeljen a felhasználójától a programozásra vagy a feladatra vonatkozó különleges ismereteket. Szokták ezt ugy is mondani, hogy legyen barátságos. Nézzük mindezt - egy kis jóindulattal - gyakorlásnak nevezhető feladaton.

Az uj mértékrendszer az SI bevezetése óta gyakran vagyunk bizonytalanságban, ha gépkocsimotorok teljesitményéről esik szó. A megszokott LE helyett most a kW a divat, sőt kötelező.

Első programunk ezt az átszámitást fogja elvégezni.
A programkészitès módszerének három fő szakaszát: a feladat elemzését, a program megtervezését és a kódolt program elkészitését már az első programunk megalkotásakor is hasznos megismernünk és elsajátitanunk. Az elemzés és tervezés során a problémát részfeladatokra bontjuk fel.
Esetünkben ez mindössze három részletet eredményez: a beolvasást, az átszámitást és a kiirást.

Követelményeink szerint legyen a program univerzális vagyis lehetőleg ne kelljen külön programot irni Trabantra és Mercedesre. A másik követelmény szerint a program tudassa felhasználójával, hogy mit kell tennie.
Persze nekünk sem árt tudni, ha netán hónapok multán elővesszük a programot, hogy miről szól.

Igy hát az első sort önmagunknak irjuk.
A REM utasitás azt jelenti hogy a sor további része nem tartozik a programhoz, hanem csupán megjegyzés, azaz REM-ark.

Programunk - ha elég intelligens és barátságos - bemutatkozik, vagyis kiirja a képernyőre mit csinál. Erre szolgál a PRINT utasitás.

A PRINT utasitás mint arról már beszéltünk is, mást is tud és egyszerü feladatunk az ugynevezett kalkulátor üzemmódban is megoldható lenne. Azonban kényelmetlen minden ujabb számitás előtt begépeltetni a PRINT szót. Azt szeretnénk ha gépünk ill. programunk ugyanugy átszámolná a motorkerékpár teljesitményadatait mint a repülőgéphajtómüét. Változó adatokkal kellene tehát a feladatot végrehajtani.

A számitógép az adatokat tárolni is képes, méghozzá nagy mennyiségben. A bekapcsolás után a számitógép olyan mint egy nagy üres fiokos szekrény, amelyben raktározni tudja a különböző adatokat.

A fiokoknak hogy hivatkozni tudjunk rájuk ill. a benniik tárolt adatokra, nevet adhatunk. Legyen pl. az egyik fiok neve A. Hogy mi kerüljön a fiokba, azt többféleképpen előirhatjuk a gép számára.

Például igy: LET vagyis legyen A=15. A LET egy ujabb utasitás /vagy ha sorszám nélkül adjuk, parancs/ értéket ad az A-val jelzett fioknak vagyis, ahogyan szaknyelven mondják az A-változónak.

Ezért a LET értékadó utasitás. A változó értékét ki is tudjuk iratni a PRINT utasitással.
Ha meg akarjuk változtatni az A fiok tartalmát, ujabb értékadással megtehetjük.

A gép ilyenkor elfelejti, hogy mi volt korábban az A fiokban és az ujabb értéket helyezi el benne. A számitógépben sok ilyen fiok van, több száz, sőt több ezer.
A géppel folytatott kommunikáció során tehát az adatokra nevükkel is hivatkozhatunk, a gép felismeri őket és előszedi a fiokból a kivánt adatot, sőt bármilyen müveletet hajlandó elvégezni velük és a müvelet eredményét vagy kiirni, vagy egy ujabb /esetleg régi/ változónak értékéül adni.

Itt a dolog már kezd a matematikához hasonlitani, de csak hasonlitani. A fiokban nem csak szám, hanem szöveg is raktározható. Egyetlen megkötés van, a fiok nevének ilyenkor $ jellel kell végződnie.
A PRINT hatására az ilyen fiok tartalma is kiiródik.
Egy kiiró utasitás egyszerre többféle adatot is ki tud irni.

Visszatérünk programunkhoz.
Egy változónak adjuk mondjuk a LO nevet egy másiknak meg a KV nevet és irjuk ezt.

```
30 LET KV = LO  0.736
40 PRINT KV
 0
```

Az eredmény Ø
És ezen nem is nagyon kell csodálkoznunk, hiszen a LO változónak nem adtunk értéket. A legtöbb gép ilyenkor Ø, értéket ad a változónak.
Pótoljuk a mulasztást. Számoljunk például egy Trabanttal.

```
2Ø LET LO=26
```

Ha most egy Zsigulira, vagy egy Volgára akarjuk elvégezni ezt a számitást, akkor a 2Ø-as sorszámu utasitást ki kell cserélnünk. Ez igy nem nagyon kényelmes, mert minden alkalmazáskor át kell irnunk a programot, ami lassu, unalmas és fárasztó.
Ilyen és hasonló problémák megoldására találták ki az un. input utasitásokat. Ahhoz, hogy a gép el tudjon végezni pl. egy számitást, az adatokat meg kell kapnia valahonnan. Egyszerübb személyi számitógépekről lévén szó, az adatok forrása a gép számára a billentyüzet, de lehet pl. a géphez csatlakoztatott periféria: magno, mágneslemez, stb. is.
Áttekintve most a számitógépes feldolgozások, számitások folyamatát, a következőket állapithatjuk meg.
A gép adatokat kap, a benne lévő program elvégzi a számitásokat, a feldolgozást és valamilyen megjelenitőn kiadja a megoldást, az eredményt.
Ez a megjelenitő a személyi számitógépeknél lehet a képernyő, de lehet nyomtató - tehát papir, vagy mágnesszalag magnoszalag is.

De térjünk vissza az input utasitáshoz. Ha a program végrehajtása során a gép egy input utasitáshoz ér, az utasitás hatására a program félbeszakad és a képernyön megjelenik egy kérdőjel. Jelezve, hogy a gép vár tőlünk valamit.

Most kell begépelni azt az adatot, amivel a számitást el kell végezni, pl. mondjuk egy kis Polski Fiatnál 23-at és lenyomni a sorzáró billentyüt.
Ilyenkor ugyanugy müködik, mintha LET utasitással csináltuk volna. A LO jelü fiok tartalma 23 lesz és a program futása folytatódik.
Vagyis programunk esetében a gép elvégzi a számitást és kiirja az eredményt.

Az input utasitás egészen "barátságossá" tehető. Ha ugyanis a változó neve elé idézőjelbe egy szöveget irunk, a gép azt a szöveget a kérdőjel után kiirja a képernyőre. Ennél a mi kis egyszerü programunknál talán nem látszik a jelentősége. De gondoljunk csak utána. Van, - vagy lesz - egy programunk, amely nemcsak egy hanem több bemenő adattal dolgozik. Ha most a felhasználó, akinek ugye nem kötelező részletesen ismerni a program szerkezetét, leül a gép mellé, törheti a fejét, hogy a gép milyen változók értékét kérdezi tőle éppen és milyen sorrendben.
Az INPUT utasitás ilyen formája a SINCLAIR ZX 81-en nincs meg, de itt is segithetünk magunkon egy az INPUT sor elé irt PRINT utasitással.

A program most más szinte teljessé tehető az átszámitás kódolásával.

```
1Ø REM LE - KW ATSZAMITÁS
2Ø PRINT "LE", "KW"
3Ø INPUT "IRD BE A LE-T", LO
4Ø LET KV=LO=Ø.736
5Ø PRINT LO, KV
```

Elinditjuk a programot.

Mikor kéri begépeljük a kivánt adatot, örvendezünk az eredménynek. És azután...
Azután vége, a program leállt.
Igaz, hogy ujabb RUN paranccsal ujra inditható, de a programozók legfőbb erénye a kényelemszeretet, magyarul lustaság.
El lehet kerülni a RUN parancs ujbóli ismételt begépelését, ha az u.n. GO TO utasitást használjuk. Aki valamicskét is tud angolul, talán már sejti is a lényegét.
A GO TO XX hatására a gép, amely eddig sorról, sorra, vagyis az utasitássorszámok növekvő sorrendjében hajtotta végre az utasitásokat, most abbahagyja ezt és a GO TO utasitásban megjelölt sorra ugrik és ott folytatja a végrehajtást.

Irjunk még egy sort a programunk végére.

Inditsuk el. Kéri az adatot. Irjuk be. Kiirja az eredményt, de nem áll le, hanem visszamegy az elejére és ujabb adatot kér. Ha megkapja kiirja az eredményt és ismét visszamegy az elejére. És igy tovább.
A kérdés az hogy a számitógépnek mikor lesz elege ebből az egészből. A válasz: soha. Teszi a dolgát, amig ki nem kapcsoljuk. Ekkor viszont elfelejti a programot és ezt nem szeretnénk.
Szerencsére a gépeken található egy BREAK /megszakitás/ vagy RUN STOP feliratu gomb.
Ha ezt megnyomjuk, a program futása felfüggesztődik, anélkül hogy a gép elfelejtené a programot. Ezt követően két dolgot tehetünk. A CONTINUE feliratu gombbal tovább engedhetjük a programot, vagy RUN-nal ujraindithatjuk.
Programunk most már működik és meg is tudjuk állitani.

TANULÁSI TANÁCSOK

Az adás megnézése előtt tanulmányozzuk a könyv 3. A programozás módszere c. fejezetét, majd a tárgymutató és a 2.sz. függelék felhasználásával a REM, a LET, az INPUT és ujból a PRINT utasitások használatának szabályait.

Gyakorlásként készitsük el a forditott. kilowatt-lóerő átszámitást végző programot és kisérelj̈ünk meg megoldani a tankönyv 69. oldalán olvasható feladatokat.

IV. F O L Y T A T Á S

Az adás vázlatos tartalma:

Az előző adás végéről van tehát egy programunk, amely a lóerőt átszámitja kilowattra és azt vég nélkül hajlandó csinálni. Sőt a gép telhetetlenül követeli az adatokat. A számitógépnek azonban van egy tulajdonsága, amelynek segitségével az előbbinél jóval "udvariasabb" programot irhatunk. A számitógép ugyanis döntésekre is képes. Mit értünk ezen? Eddig olyan programokkal találkoztunk,ahol az utasitásokat szép sorjában kellett végrehajtani, legfeljebb visszaugrottunk a program elejére.
A gép képes arra, hogy egy bizonyos feltétel teljesitése esetén ne a soronkövetkező utasitáson folytassa a végrehajtást, hanem tetszőleges másikon. Nézzünk erre egy példát.
Alakitsunk ismét a programunkon. Az elején ne csak azt irassuk ki, hogy mit kell tenni a felhasználónak,hanem azt is, hogy miként állithatja le a programot.

Irjuk pl.

```
15 PRINT "HA BE AKAROD FEJEZNI,IRJ BE Ø-AT".
```

A program további részében pedig meg kell vizsgáltatnunk, hogy az INPUT utasitás hatására bejövő Ø-e vagy sem. Ha igen, vége.
Ha nem, vissza az elejére. Az erre szolgáló utasitás az IF...THEN utasitás. Lényege már a szavak jelentéséből adódik. Ha ... akkor. Az emlitett feltételvizsgálatot a következőképpen oldhatjuk meg programunkban. Az utolsó GO TO 1Ø sor helyett a következőt irjuk:

```
 9Ø IF LO=Ø THEN STOP
1ØØ GO TO 1Ø
```

Vagyis ha a LO változó értéke, amit az INPUT utasitás nyomán felvesz Ø, a program megáll, egyébként kezdődik elölről és várja a következő adatot.
Érdemes megjegyezni, hogy az ilyen módon megállitott program szintén folytatható a CONTINUE parancs beadásával.

Foglaljuk össze röviden mit is tudunk már.
Aki azt mondja: mindent, ami a programozáshoz szükséges, igaza van. Meglepő, de igy igaz.
Az eddig megismert utasitások segitségével szinte bármilyen komplikált feladat megoldható.

Beszéltünk a változókról, vagyis azokról a fiokokról, amelyek a gép memoriájának részét képezik és amelyek változó adatokat tartalmaznak. A fiokban lévő adat a változó értéke.
A fiok cimkéje a változó neve. Egy változónak a LET vagyis az értékadó utasitás segitségével adhatunk értéket. A változók neve a BASIC nyelvekben tetszőleges szó vagy szöveg lehet. Lényeg, hogy az első karakter betü legyen. Az egyes gépeknél vannak bizonyos megkötések. Ha a változónak szöveges értéket kivánunk adni, a változó neve után $ jelet kell tenni, értékadáskor pedig a szöveget idézőjelbe kell tenni. Megismertünk összesen hat utasitást illetve parancsot. Ha ezeket a szavakat begépeljük, persze megfelelő formai szabályok szerint, a gép azonnal végrehajtja őket a RETURN megnyomása után. Ha viszont sorszámot irunk eléjük, utasitás lesz a nevük, ilyenkor a gép a sorszámok sorrendjében raktározza, tárolja, majd külön parancsra, a RUN-ra sorba végrehajtja őket.

A megismert parancsok:

LET	értékadás
INPUT	adatbevitel
PRINT	kiirás a képernyőre
GO TO	ugrás a megjelölt sorszámu sorra
IF THEN	feltételes parancsvégrehajtás
RUN	a program végrehajtásának inditása

Hogy tovább lépjünk, ismerkedjünk meg egy ujabb parancscsal. Erre akkor van szükség, ha a gépbe uj programot szeretnénk beirni és azt akarjuk, hogy a régit felejtse el a gép.
Ilyenkor használják a NEW parancsot.

A korábban emlitett parancsok mindegyikét programban, utasitásként is kiadhatjuk. Igy van ez a NEW-vel is. Csak éppen nincs sok értelme. Az igy keletkező program öngyilkos program, ugyanis kitörli önmagát.

Ismereteink birtokában ujabb - igaz, nem tul bonyolult feladatnak vághatunk neki. Valószinüleg mindenki látott már szalagnaptárt. Kisérelјük meg ilyet készitеni, persze csak a képernyőre.
A feladat tehát a következő. Irassuk ki függőlegesen egymás alá a napok sorszámait és melléjük azt, hogy a hét melyik napjára esnek.

Kezdjük a dolog könnyebbik végével. Először csak irassuk ki egymás alá a napok sorszámait. Ezt az eddigi ismereteink alapján meg tudjuk oldani, pl. a következő programmal.

```
1Ø I=1
2Ø PRINT I
3Ø I=I+1
4Ø IF I=3Ø THEN GO TO 2Ø
```

Ez igy is megoldás. A fejlett programozási nyelvek, igy a BASIC is azért születtek, hogy megkönnyitsék a programozók munkáját. Az ilyen "számlálós" tipusu feladatokra találták ki a FOR utasitást.

Ennek felhasználásával a program a következőképpen fest.

```
1Ø FOR I=1 TO 3Ø
2Ø PRINT I
3Ø NEXT I
```

Ez a program ugyanazt csinálja mint az előbbi. I értékét 1-től egyesével növeli 3Ø-ig és minden I értéket kiir. A kulcsszavak leforditása itt is sokat segit a megértésben. Használható a FOR nevü u.n. ciklusutasitás a következő alakban is:

```
1Ø FOR I=25 TO 1Ø STEP-1
2Ø PRINT I
3Ø NEXT I
```

Ilyenkor a számlálás nem egyesével, hanem a STEP /lépés/ szó után megadott lépésközzel történik.
A ciklusutasitás müködése tehát a következő:

1. A FOR után irt ciklusváltozónak, az egyenlőségjel után következő szám adja a kezdőértékét.
2. Végrehajtja a következő utasitást /utasitásokat/ egészen a NEXT sorig.
3. A STEP után álló lépésköz értékével növeli /negativ szám esetén csökkenti/ a ciklusváltozó értékét.

4. Megvizsgálja, hogy nagyobb-e /negativ lépésköznél kisebb-e/, mint a TO után álló érték és ha nem, végrehajtja a következő utasitást és ismét növeli a ciklusváltozó értékét, ha igen, akkor a NEXT után következő sorral folytatja a végrehajtást.

Most már "csak" ugy kell átalakitani a programot, hogy minden szám mellé egy "szöveget" is irjon.
A hétfő, kedd, szerda, stb. valamelyikét. Ez megoldható lenne mondjuk egy 2Ø PRINT I; N$ formáju sorral.

Kérdés most már, hogy miként kerülnek a napok nevei az N$ szöveges változóba.
Egy megoldást mutat az alábbi program: /COMMODORE 64 számitógépen/

```
  1Ø REM  SZALAGNAPTÁR
  2Ø PRINT "SZALAGNAPTÁR"
  3Ø L=1
  4Ø FORK=1 TO 3Ø
  6Ø IF I=1THENN$="H"
  7Ø IFI=2THENN$="K"
  8Ø IFI=3THENN$ = "SZE"
  9Ø IFI=4THENN$ = "CS"
 1ØØ IFI=5THENN$ = "P"
 11Ø IFI=6THENN$ = "SZO"
 12Ø IFI=7THENN$ = "V":I=Ø
 125 I=I+1:IF 1  7 THEN I=1
 13Ø PRINTK, N$
 14Ø NEXTJ
```

TANULÁSI TANÁCSOK

Az adás előtt érdemes a tankönyvben áttanulmányozni ezeket a fogalmakat, tehát a GO TO, az IF...THEN és a FOR.NEXT utasitások használatának tudnivalóit. Segitségünkre lesz ebben a tankönyv 5. és 8. fejezete, a tárgymutató és a könyv végén található 1.sz. és 2.sz. melléklet, amelyből a környezetünkben található gép specialitásairól is tájékozódhatunk.

Uj programunk a szalagnaptár. Listája az előbbiekben szerepelt. Házi feladatként próbáljunk meg tökéletesiteni a programunkon /például, hogy ne szaladjon el a lista a képernyön/, kérdezze meg, hogy a hétnek melyik napjára esett elseje, hány napos a hónap és eszerint irja ki a listát.
Továbbá megprobálkozhatunk a tankönyv 111. oldalának 1., 3., 4. feladatával és a 165. oldal 4. feladatával is.

V. ELÁGAZÁSOK, CIKLUSOK

Az adás vázlatos tartalma:

Bemutatjuk a házi feladat megoldását. A feladat az volt, hogy alakitsuk ugy át a mult alkalommal összehozott szalagnaptár programot, hogy az először kérdezze meg, hogy milyen napra esett elseje és ennek megfelelően készitse el a napok listáját.
Ennek megoldásához

az INPUT "HÁNYNAPOS A HÓNAP"; HO
a FORK = 1 TO HO javitott és az
INPUT "A HÉT HÁNYADIK NAPJA ELSEJE"; I sorokat

kell a listába beiktatni a I=1 sort pedig törölni.

Bizonyára léteznek egyszerübb és elegánsabb megoldások is. A feladat azonban arra biztosan jó volt, hogy érzékeltesse, egy hétköznapi fogalmaink szerint egyszerü dolgot nem biztos, hogy ugyanolyan könnyü megcsinálni a géppel.

Program-elágazással lényegében már találkoztunk még a lóerő átszámitó programban és a naptár készitésénél. Ez a téma megér még egy misét, annál is inkább, mert ezzel az utasitással kapcsolatban szokott talán a legtöbb hiba előfordulni.

A programkészités hevében legutobb elmaradt egy igen szemléletes példa az elágazások lényegének érzékeltetésére.
Képzeljünk el egy hajdan volt cipészt, magyarul susztert. A suszter kezdetben egyedül dolgozott tehát szépen sorban, maga csinálta végig a cipőkészités valamennyi munkafázisát. Mivel mindent maga csinált és jól csinált, tevékenységének programjában nem voltak elágazások.

Később amikor jobban kezdett menni az üzlet segédeket és inasokat fogadott, s rájuk bizta a munkafázisok elvégzésének egy részét. Mivel ők nem voltak olyan igyekvők bizony selejt is előfordult.
Az egyes munkafázisok után tehát ellenőrző vizsgálatokat kellett beiktatni és hiba esetén a munkát ujra elvégeztetni.

Elágazások megvalósitására a BASIC-ben többféle lehetőség van, a feladat jellege dönti el, hogy éppen melyiket használjuk. Az egyik leggyakrabban használt IF...THEN szerkezetre már láttunk példákat.

Az IF...THEN utasitás hatására valamilyen feltétel teljelülés t vizsgálja meg a gép és attól függően hajtja végre az elágazást A feltétel valamiféle matematikai reláció formájában fogalmazódik meg.

Ha a feltétel igaz, a THEN utáni utasitásra adódik a vezérlés, egyébként a következő sorra. Ha figyelmesen tanulmányozzuk az IF...THEN utasitás szintaktikáját, azt látjuk, hogy az IF után kifejezés 1/reláció/ kifejezés 2 THEN megfogalmazás áll.

Tehát a feltétel vizsgálatát nemcsak változókra és konstansokra, hanem u.n. kifejezésekre is hajlandó elvégezni a gép.
A kifejezések változóknak és konstansoknak müveleti jelekkel való összekapcsolásával keletkeznek. A bonyolultabb kifejezések sok müveletet tartalmaznak, ezért az egyértelmüség kedvéért a müveletek között elsöbbségi rangsort kellett kialakitani.

A BASIC-ben is megengedett a zárójelek használata. Hatásuk ugyanaz, mint a matematikában. Legelőször a legbelső zárójelben lévő müveletek hajtódnak végre. A zárójeles kifejezésen belül, vagy zárójel nélküli kifejezésnél az elsőbbségi sorrend a következő.
Tehát először az hajtódik végre.

1. Hatványozás
2. Szorzás, osztás
3. Összeadás, kivonás
4. Balról jobbra szabály

Ha egy kifejezésben több azonos prioritásu müvelet szerepel, akkor balról jobbra haladva hajtódnak végre.

Létezik a HT gép és a PRIMO esetében IF...THEN...ELSE változata is. Itt a vezérlés a feltétel igaz volta esetén a THEN utáni részre, hamis volta esetén az ELSE utáni részre adódik. A feltételes elágazások alkalmazásának jó néhány trükkje van, ezek közül mutat a müsor néhányat.
Ha a feltétel teljesülése esetén végrehajtandó programsor rövid, egyetlen utasitás vagy parancs érdemes rögtön a THEN után irni, esetleg kettőspontok felhasználásával.

Ha a feltételes elágazás "igen" ágán végrehajtandó prog ramrész vagy modul nagyobb terjedelmü, érdemes GO TO-t használni a THEN után. Ha azonban végiggondoljuk, rájövünk, hogy ez ujabb GO TO-t eredményez. Ugyanis az igen ágon következő programrészből vissza kell ugrani a nem ág utáni részre, vagy a nem ág programrésze után át kell ugrani az előbbi igen ági programrészt.

Pl.
```
3Ø IF A=B THEN GO 12Ø
5Ø
7Ø END
12Ø A=B-A
13Ø I=I+1
14Ø GO  TO 5Ø
```

vagy pl.

```
3Ø IF A=B THEN GO  TO 12Ø
5Ø
6Ø GO TO 15Ø
12Ø A=B-A
13Ø I=I+1
15Ø
```

Elegánsabb és jóval kevesebb hibaforrást rejt magában a következő megoldás.

```
3Ø IF A   B THEN 12Ø
4Ø A=B-A
5Ø I=I+1
   .
   .
   .
11Ø
12Ø
```

Vegyük észre, hogy itt át kellett alakitani a feltételt, pontosan az ellenkezőjére.
Ha mind az igen, mind a nem ágon hosszabb végrehajtandó müveletsor következik, s csak ezután folytatódik a közös programrész, már nem lehet elkerülni az előbb emlegetett átugrást.
A nehézséget, ha utánagondolunk, az okozza. hogy az

egymás mellé rajzolt modulokat a programban csak egymás után lehet elhelyezni.
Gyakori hiba, hogy kifelejtjük a 14Ø sort és a gép A=B esetén végrehajtja M után L modult is.

```
 3Ø IF A=B THEN 14Ø
 4Ø  M
13Ø
14Ø GO  TO 21Ø
15Ø
2ØØ  L
21Ø
```

Azt láttuk, hogy a jobb megoldás érdekében át kellett fogalmazni az elágazás feltételét. Ebben is a bonyolultabb feltételek megfogalmazásában is segitségére van a BASIC a programozóknak. A segitség pedig a logikai müveletek. A BASIC-ben használhatjuk a NOT /nem/, az AND /és/ és az OR /vagy/ logikai müveleteket. Ezeknek értelmezése meglehetősen hasonlit a hétköznapi értelemben használt és vagy nem szavakhoz. Vagyis egy feltétel AND "egy másik feltétel" akkor igaz, ha mindkét feltétel igaz.

"Egy feltétel OR egy másik feltétel" akkor igaz, ha legalább az egyik feltétel igaz. A NOT megfelel a tagadásnak, tehát a NOT feltétel akkor igen, ha a feltétel hamis, forditva. A NOT müveletet már felhasználhattuk volna előbb, a feltétel megforditásához.

Az adásban szereplő példa a logikai müveletekre a Milyen gépet vegyünk a gyereknek cimet viseli.
A logikai müveletek használatával logikai kifejezéseket hozhatunk össze. Ezek nagy segitséget jelentenek bonyolultabb feltételek megfogalmazásához.

A ciklusutasitások önmagukban is tartalmaznak feltételes vizsgálatot. Érdekes feladat kideriteni, hogy a ciklus elején vagy végén szerepel-e ez.
Ha a feladat olyan, hogy egy feltételtől függően nem csak kétfelé lehet elágazni, hanem többfelé is, CASE tipusu szerkezetről beszélünk. Ennek megvalósitása történhet IF THEN utasitások segitségével. Ilyenkor sorban megvizsgálja a feltételek teljesülését és ettől függően ugrik tovább.

A BASIC az ilyen tipusu elágazások megvalósitására tartalmazza az ON...GO TO... tipusu utasitást.
Ez az utasitás lehetővé teszi, hogy a program az A változó vagy kifejezés értékétől függően elágazzon sorszámok valamelyikére. Néhány apróság: ha A=Ø a program a következő sorban folytatódik, ha A mint az elágazások száma, szintén a következő sorban folytatódik a program, ha A nem egész, egészrész /A/t veszi figyelembe. Az ON... GO TO... alkalmazására láttunk egy programot a Rádiótechnika c. lapból. A program a számjegyekkel beirt háromjegyü számot szöveges formában irja ki. Hasznos lehet, ha gyerekeink a számok olvasását tanulják.

TANÁCSOK A TANULÁSHOZ

Az adás megtekintése előtt a tankönyv 5. és 6. valamint 8.. fejezetének a témához kapcsolódó ismereteit tanulmányozzuk. A gyakorláshoz javasoljuk a tankönyv 111.oldalán találhatú 2. 4. és 5. feladat megoldását. Valamint az adás végén emlitett u.n. számolvasó program megtervezését és elkészitését. /A programnak a legépelt háromjegyü számokat szöveges formában kell kiirnia.

MILYEN GÉPET VEGYÜNK A GYEREKNEK
cimü program listája

```
15 PRINT
2Ø INPUT "a gyermek kora", K
25 PRINT K
5Ø IF K  6 OR K   80 THEN PRINT "KORAI"
4Ø IF K  6 AND K  12 THEN PRINT
   "ESETLEG EGY ZX 81-ET"
5Ø IF K   = 12 AND K   18 THEN PRINT
   "C 64-ET VAGY A SPECTRUMOT JAVASOLOM"
6Ø IF K   =18 AND K  8Ø THEN PRINT
   "VALAMI KOMOLYAT, AMIT A PÉNZTÁRCÁJA ENGED"
52 PAUSE 200
63 CLS
7Ø GO  TO 15
```

VI. G R A F I K A

Az adás vázlatos tartalma:

Az a tény, hogy a személyi számitógépek általában a televizió készülék képernyőjét használják fel a bennük tárolt információk, programok, üzenetek megjelenitésére, kézenfekvővé teszi a gondolatot: a számitógép ne csak betüket rójon egymás mellé a képernyőre, hanem rajzoljon oda ábrákat, képeket, netán mozgassa is azokat. Ez a gondolat persze nem ujkeletü, szinte egyidős a számitógépekkel.

A számitógépeket kezdettől fogva használták rajzolásra, rajzológépek vezetésére, főleg a tervezés segitése érdekében.
Persze nem kell ilyen messzire menni. Elég egy pillantást vetni néhány személyi számitógépre irt grafikai programra, vagy ügyesen megirt játékprogramra, hogy felkeltse bennünk az igényt, magunk is szeretnénk ilyet, vagy legalábbis hasonlót csinálni.

A rajzolás megvalósitására legkézenfekvőbb utasitás a PRINT. A PRINT eddig megismert alakja mindig a soron következő szabad helyre, vagy a következő sorba irt. Ezzel is elérhető, hogy a képernyő adott pontján megjelenhessen mondjuk egy szöveg, de ehhez meghatározott számu üres sort és az érvényes soron belül szóközöket kell kiirni, ami bizony kissé nehézkes.
/Az előbb bemutatott egérke ilyen eljárással készült./
Ezen segit a PRINT utasitás PRINT TAB /Y/; ill. egyes gépeknél PRINT TAB /X,Y/, illetve PRINT $ vagy PRINT AT /X,Y/: formája. A PRINT AT /X,Y/: az X-edik sorba és az Y-adik poziciótól kezdve ir.

A számozás a képernyő bal felső sarkában kezdődik. A sorok és oszlopok száma gépenként változik, általában 2o körüli sor és 4o körüli oszlop szokott szerepelni a képernyön.

A tervezéshez célszerü elővenni a kockás papirt. Kezdő programozók első sikerélménye szokott lenni, ha sikerül a nevét a képernyő közepére kiirnia és csillagokkal bekeretezni. Vannak kész programok, amelyek ezt sokkal szebben csinálják, de ezeken aligha lehet a programozást megtanulni.

A csillagok kirakosgatásához érdemes ciklust szervezni. Érdekes, hogy a párhuzamos vonalak egy ciklussal kiirhatók. CIS-sel az egészet most is letörölhetjük. De nézzük meg mi van ha mindig csak az előzőleg kiirt csillagot töröljük le. Törlésen itt azt értjük, hogy szóközt irunk a helyére.
Ilyenkor a csillag végigfut a soron.
Ha a futás tul gyors, időhuzó ál utasitásokat tehetünk a PRINT-ek közé, pl. PAUSE X. Csillagokból vagy más grafikus karakterekből összeállithatunk figurákat. Ezeket hasonló eljárással meg lehet mozgatni.
Észrevehető, hogy ezek a figurák kis karakterekből tevődnek össze.
Nos a gép ilyen előregyártott kis karakterekből tud rajzolni. Természetesen csak eléggé durva rajzokat. Akik finomabbakat akarnak lerajzolni, azok számára adott a lehetőség ezeknek a kis négyzeteknek a "belsejében" lévő 8x8-as bontásu képpontokból összeállitani finomabb grafikákat. Az egyes betük és számok is egy-egy ilyen 8x8-as képelemből álló négyzetbe kerülnek.
A SINLACIR SPECTRUM-nál ehhez jön még az a lehetőség, hogy a szineket is változtatni tudjuk.

A személyi számitógépek legtöbbje a bemutatottnál finomabb grafikai lehetőséget is biztosit tehát.
A PLOT X,Y illetve a SET X,Y utasitásokkal ki lehet gyujtani a képernyő egy pontját.
Az UNPLOT ill. a RESET utasitásokkal pedig el lehet oltani. A koordinátarendszer itt jóval sürübb beosztásu, mint a PRINT utasitás használatánál.

HT-nél 48x128, ZX 81-nél 44x64, SPECTRUM-nál 176x256, PRIMO-nál 192x256 pontra van felosztva a képernyő.
A koordinátarendszer itt azonban más elrendezésü, mint a PRINT utasitás esetében, jobban hasonlit a megszokotthoz. A bal alsó sarokban van a Ø.Ø pont.
A keret helyett most probáljunk meg egy Ø.3 meredekségü átlós egyenest előállitani, mondjuk a Primon.

Mint látjuk, nem elég felvenni változókkal az egyenes egyenletét, az X-et nekünk kell változtatni lépésenként. Hasonló eszközökkel készült a PRIMO bemutató kazettáján található cikk-cakk program.

Kisebb felbontásu gépeken az egyenesből inkább lépcső lesz, de az elkészités igy is nagyon tanulságos.
A SINCLAIR SPECTRUM az emlitettektől eltérő módon is segiti az egyenes, a kör és a köriv rajzolást. Ezt be is mutatjuk. A PLOT X és Y koordináta utasitás a megadott képelemet INK szinüre változtatja. A SPECTRUM-on ezen kivül utasitással változtathatjuk a keret és a háttér szinét is. /INK=tinta, tehát ezzel az utasitással lehet megadni a szöveg vagy grafika szinét./ Az egyenes vonalat rajzoló DRAW utasitás megfelelő adatai: X,Y.

A vonal kezdőpontja az a képelem, ahol a legutobbi PLOT, DRAW vagy CIRCLE utasitás abbahagyta a rajzolást.

/Ezt PLOT poziciónak nevezzük./

A vonal végpontja az a képelem, amely vizszintesen megadott X, függőlegesen a megadott Y távolságra van a kezdőponttól.
A DRAW utasitás tehát csak a vonal hosszát és irányát határozza meg, a kezdőpontját nem
A DRAW utasitásban igy negativ számokat is használhatunk a PLOT utasitásban csak pozitiv számok szerepelhetnek.
A DRAW utasitás nemcsak egyenes vonalak, hanem körivek rajzolására is felhasználható, ha még egy számot megadunk, amely a kör nyilásszögét definiálja. Ez esetben "A" az elfordulás szöge radiánban. Ha az "A" pozitiv, az elmozdulás - kanyarodás - balra, ha negativ, akkor jobbra történik.

A következő utasitás a CIROLE, amely egy teljes kört rajzol. Itt az X és Y határozza meg a kör középpontját, az "R" a kör sugarát adja. /DRAW X,Y, A ill. CIROLE X,Y,R/

A fentiek együttes alkalmazásával már egészen látványos kis grafikát tudunk összehozni.

Talán észrevették, hogy a COMMODORE ebben az adásban még nem jutott szóhoz. Ennek az oka, hogy noha a COMMODORE-nak igen sok és fejlett grafikai lehetősége van, 2oox3oo pont, 16 szine, stb. a BASIC-ből ezek közvetlen utasitással nem érhetők el.

A COMMODORE 64 grafikai lehetőségeivel külön adásban fogunk foglalkozni.

TANÁCSOK A TANULÁSHOZ

A grafikai utasitáskészlet az egyes géptipusokon igen eltérő, ezért azt javasoljuk, hogy a rendelkezésre álló géptipus grafikai lehetőségeivel probáljunk megismerkedni a gépkönyvek segitségével.

Gyakorló feladatként megfelel ha a képernyő közepére felirjuk saját nevünket, és előbb csillag karakterekkel majd egyenes vonalakkal bekeretezzük.
Amelyik gépen erre lehetőség van, probáljuk körvonalakból keretet készitenil

VII. A PROGRAMOZÁS MÓDSZERE

Az adás vázlatos tartalma:

Ebben az adásban a programozás módszerével foglalkozunk. Érdemes egy pillanatra felidézni a számitógéppel való kommunikáció folyamatát. Kezdetben van a probléma. Az ember megfogalmazza a kivánságot. Hogy korábbi példánknál a naptárnál maradjunk: jó lenne ha a gép megmondaná milyen napra esik mondjuk szeptember 9-e. A gép számára a szeptember 9-e egy adat, ennek felhasználásával elvégez egy müveletsort, majd kiadja az eredményt.

Szükség van valamilyen módszerre, amelynek segitségével a programozási problémák megoldhatók. Az általunk javasolt módszer a következő - a módszer egyik alapelve: a programkészitést papiron, ceruzával kell kezdeni. Toljuk félre a gépet és legelőször is fogalmazzuk meg pontosan és lehetőleg irásban,

a/ hogy mit szeretnénk eredményül kapni /ez lehet szám, szöveg, táblázat, grafikon, rajz stb./

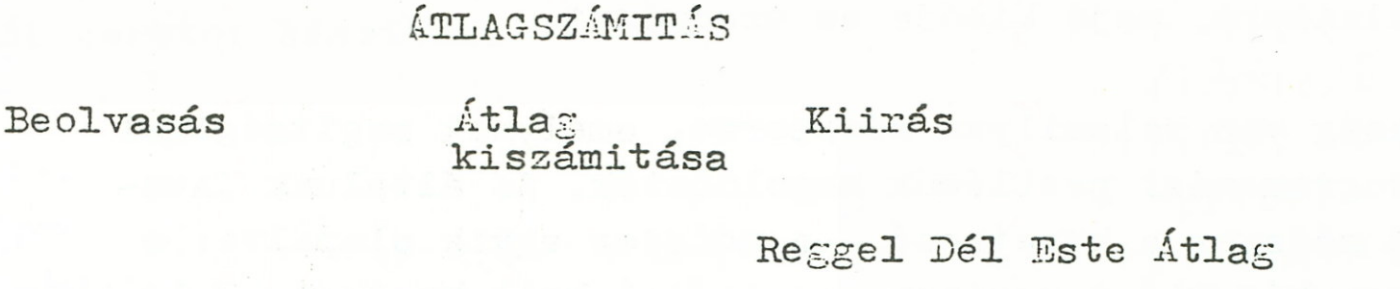
b/ hogy milyen kiinduló /bemenő/ adatokat akarunk megadni.

A programozás módszerének elemeit egy nagyon egyszerü feladaton mutatjuk be. A készitendő program feladata, hogy a reggeli a déli és az esti hőmérséklet legépelése után irjon ki a képernyőre a beirt hőmérséklet értékével arányos hosszuságu csillagokból álló vizszintes vonalat. Számitsa ki a napi középhőmérsékletet és ezt is ábrázolja vonal formában.

A program elkészitése során megismerjük a programkészités javasolt módszerének lépéseit: a feladat részfeladatokra bontását. a részfeladatok elemzését, esetleg ujabb részekre bontását majd kódolását. a részprogramokból a teljes feladat programjának összeállitását, tervezését.

A középhőmérséklet számitó program u.n. hierarchia diagramja, amely jól szemlélteti a feladat részfeladatokra bontását és a részfeladatok egymásra épülését.

Ábra



Ezután következik a program kiprobálása. Begépeljük a programot és megkisérreljük lefuttatni. Ha fut és jó eredményeket ad, minden jó. Ha nem, következik a programozók gyötrelme, a hibakeresés. Egy hibalehetőség példaképpen: SYNTAX ERROR - helyesirási-hiba - rosszul gépeltük vagy irtuk a programot. Egyéb hibajelzésekről a gépkönyvek adnak részletes tájékoztatást. Még egy hibalehetőség, amely figyelmet érdemel. Se kép, se hang a képernyön illetve a hangszóróban. Ilyenkor nagyon valószinü, hogy igen hosszu, esetleg végtelen ciklust hajt végre a gép, többnyire a mi hibánkból.

Ilyenkor a BREAK segit, no meg a hibakeresés, melyre pl. akkor is szükség van ha az eredmény nem az, amit vártunk. Ilyenkor ujra végig kell magunkat rágnunk a programon. Segit a hiba felderitésében, ha modulok végére PRINT

utasitásokat iktatunk ideiglenesen be és kiiratjuk a modul kimenő adatát. Igy kiderül, melyik modulban van a hiba.
A legjobb tanács amit adni lehet, hogy ne veszitsük el a türelmünket.
A programkészités módszeréről további hasznos és részletes tudnivalókat találunk a tankönyvben.

TANÁCSOK A TANULÁSHOZ

A tankönyv részletesen foglalkozik a programozás módszerével. Ezeknek a részeknek az áttanulmányozása mindenképpen hasznos. A programozást megtanulni csak önálló munkával, programok elkészitésével lehet.

Javasoljuk, hogy válasszanak ki egy egyszerübb feladatot és azt próbálják meg elejéről a végéig önállóan megoldani, programjait elkésziteni a bemutatott módszerrel.

VIII. SZUBRUTINOK

Az adás vázlatos tartalma:

A programozó legfőbb erénye a lustaság. Pontosabban szólva a programozó irtozik attól, hogy ha ugyanazt a dolgot kétszer vagy többször kell csinálni. Az ilyen "gépies" feladatokat igyekszik a gépre háritani. Ennek elérése a BASIC nyelv több lehetőséget is tartalmaz. Ilyen volt a már megismert ciklusutasitás és ilyen lehetőséggel foglalkozunk ebben az adásban is.

Emlékszünk még LE-KW átszámitó programunkra. Egészitsük ki most ugy, hogy a képernyőre kiirt üzenetei még "barátságosabbak" és szebbek legyenek. Ehhez elöször is jobban szét kellene választani a sorokat.
Tehát először is törölje a képernyőt, aztán irjon egy szép fejlécet, aztán hagyjon 3 sort üresen és minden átszámitás után hagyjon ki 5 sort, csak ugy kezdje a következőt. Mindez megoldható a programban elhelyezett PRINT utasitásokkal, amelyek után nem irunk semmi kiirnivalót.

8 db PRINT utasitás beirásával megusztuk, de talán nem is kell mondani, hogy ennél összetettebb feladatok is vannak. Nem mintha ez a néhány PRINT beirás olyan megerőltető lenne, de hát a programozóban felmerül a kérdés: nem lehetne mindezt egyszerübben, kevesebb irkálással megoldani?
Ilyen és hasonló ismétlődő feladatok megoldásához nyujtanak lehetőséget a szubrutinok.

Nevüket alapprogramnak szokták forditani. Olyan önálló program részek, amelyeket a gyakran használt utasitásokból alakitunk ki.

A BASIC-ban ezt két utasitással tudjuk megvalósitani. Ezek: a GOSUB és a RETURN.
A GO SUB szubrutinhivó utasitás hasnnlit a GO TO-ra,azaz a program végrehajtás máshol folytatódik egészen addig, amig egy RETURN utasitást nem talál a számitógép. Ekkor visszatér a GO SUB utáni sorra.
A korábban emlegetett "suszter" bácsira gondolva, az analógia a következő. A mester a gyakran ismétlődő müveletek elvégzésére gépeket vásárol. A cipőkészitités különböző fázisain segédek dolgoznak. Pl. a varrógépet ha szükséges mindegyik használhatja. Ha befejezte a varrást, visszamegy a saját helyére folytatni a munkát.

A GO SUB annyival több mint a GO TO, hogy egy programon belül a szubrutint akárhányszor felhasználjuk, ugy mondjuk hivhatjuk. A RETURN után ugyanis nem egyetlen konkrétan megadott cimre tud visszaugrani, mint a GO TO esetében, hanem mindig a meghivás után következő sorra.

Alakitsuk át szubrutinunkat univerzálisabbra, ugy hogy tetszőleges számu sort irathassunk vele. Ezt ugy érhetjük el, hogy a PRINT utasitás végrehajtására ciklust szervezünk.

```
2ØØ FOR I=1 TO  3
21Ø PRINT
22Ø NEXT 1
23Ø RETURN
```

Ez még mindig 3 sort ir ki de ha a 2ØØ. sort kicseréljük a következőre:

```
2ØØ FOR I=1 TO N
```

Ha most az N értékét a ciklus lefutása előtt mindig beállitjuk az általunk óhajtott számra, /ti. ahány üres sort akarunk/, akkor szubrutinunkkal tetszőleges számu üres sort irathatunk.

Az előbbi program akkor igy fest:

```
PRINT fejléc
LET N=3: GO SUB 2ØØ
INPUT
LET N=5: GO SUB 2ØØ
```

Az előbbiekből az is a tanulság, hogy a szubrutinokat célszerü nem csak az adott feladatra, hanem annál kicsit többre is alkalmassá tenni. Igy esetleg máskor és máshol is felhasználható lesz.

Az ilyen "többet tudó" univerzálisabb szubrutinoknak a meghivás előtt mindig meg kell adni, melyek azok a konkrét értékek, amelyekkel a tevékenységet végrehajtják. Erre a célra ugyanazokat a változókat kell felhasználnunk, amelyeket a szubrutin is használ. Ezt nevezik paraméterátadásnak. Az előbbi példában a szubrutin bemenő paramétere N volt. Ha a szubrutin olyan hogy valamilyen értéket kiszámol a bemenő paraméterek alapján, a kapott eredményt vissza kell adni a főprogramnak. Ez a kimenő paraméter átadása. Hogy az előbbi példából valami haszon is származzék és hogy az általánosnak felhasználható.
A szubrutinokat tetszőleges helyre tehetjük a programban, csak arra kell vigyázni, hogy véletlenül rá ne fusson a vezérlés.

Pl. Előbbi programunkban, ha a szubrutin első sora előtt nincs GO TO vagy a STOP, akkor a szubrutin hivás nélkül is végrehajtódhat.
A szubrutinoknak, amint ez ránézésre is megállapitható, több belépési pontja is lehet.
Igy például kiinduló feladatunk is megoldható egy több belépésü ponttal rendelkező szubrutinnal.
Következik egy könnyebb, de annál tanulságosabb feladat...
Mondjuk egy kacsalábon forgó palota megrajzolása.
Az előbbiekből tanulva, ill. már tudatosan keressük az ismétlődő tevékenységeket, amelyek szubrutin irásra alkalmasak.

Például észrevehetjük, hogy sok párhuzamos és a képernyő szélével is párhuzamos egyenest kell rajzolni.
Használjuk fel a korábbi adásban megismert kis program elvét, amely csillagokból párhuzamos egyeneseket rajzol.
Ezt alakitsuk át, hogy tetszőleges helyre, tetszőleges hosszban lehessen egyenes rajzolni.
Nézzük mik lesznek az adatok.

C hányadik sor
D hányadik oszlop — bal alsó pont
A hány pont egy sorban
B hány sorral feljebb rajzolja a következő pontot
A szubrutin:

```
1ØØ FOR I=C TO C+A
11Ø SET /I,D/
12Ø SET /I,D+B/
13Ø NEXT I
14Ø RETURN
```

A függöleges párhuzamosokat kiiró szubrutin a szerepel felcserélésével adódik.

```
2ØØ FOR K=D TO D+B
21Ø SET /C,K/
22Ø SET /C+A,K/
23Ø NEXT K
24Ø RETURN
```

Már van 2 szubrutinunk. Ez már sokmindenre használható. Rajzolhatunk vele keritést, stb. és felhasználjuk őket egy másik szubrutinhoz, amely téglalapot rajzol.
A háztető felrajzolásához szükséges két pontot /X1, Y1 és X2, Y2 koordinátákkal/ összekötő szubrutin:

```
29Ø LET M=/Y2-Y1/ /X2-X1/
3ØØ FOR L=Ø TO X2-X1
31Ø SET /X1+L, Y1+M-L/
32Ø NEXT L
33Ø RETURN
```

Összefoglalva a szubrutin előnyeit.

1. Röviditi a programot
2. Általánositjuk a feladatot
3. Szinte uj parancsokat definiálhatunk szubrutinok segitségével. Erre a PRINT AT megvalósitásánál láttunk példát, de ugyanugy a DRAW, a CIRCLE mellé már van egy téglalap utasitásunk is, kicsit más szintaktikai /használati/ szabályokkal. de ugyanolyan jól használhatóan. A szubrutinok segitségével elönyösen készithetők a részfeladatokra bontott u.n. moduláris programok.

Az összegyüjtött szubrutinok nagyon megnövelik a programozónak és gépének lehetőségeit.

TANÁCSOK A TANULÁSHOZ

A szubrutinok használatára vonatkozó tudnivalók a tankönyv 6. és 7. fejezetében találhatók. Az adás megtekintése előtt a 6. fejezet végén szereplő alapfogalmakat feltétlenül érdemes tanulmányozni.

Gyakorlásként nagyon tanulságos a müsorban szereplő egyenes és téglalaprajzoló szubrutinok elkészitése és kiprobálása. Ezenkivül érdemes megkisérelni az eddig megismert feladatok egy részének szubrutinként való kódolását, valamint a házi feladatként szereplő előjegyzési naptár program elkészitését.

IX. FÜGGVÉNYEK I.

Az adás vázlatos tartalma:

Az adás elején szerepel a korábbi házi feladat megoldása. A program listája:

```
  5 REM   NAPTAR
 1Ø FOR N=8 TO 11
 3Ø GOSUB 2ØØ
 4Ø GOSUB 3ØØ
 5Ø NEXTN
 55 FOR I=Ø TO 20: PRINT AT I,12;"X": NEXT I
 6Ø STOP
2ØØ PRINT AT 21,0: "MIT CSINÁL"N";"TOL";N+1"-IG?"
21Ø INPUTA$
22Ø RETURN
3ØØ PRINT AT /N-8/ 5, 2;M; "-", N+1, AT /N-8/
    5,15, A$
4ØØ FOR I=Ø TO 31: PRINT AT /N-8/ 5+2,1; " ":
    NEXT I
41Ø RETURN
```

A függvények matematikai fogalmáról, azt hiszem mondhatjuk igy, mindenki hallott már. Most viszont nézzük a dolognak egy kissé sajátos számitástechnikai megközelitését. Visszatérve vesszőparipánkhoz: a programok általános szerkezetéről a következőt mondhatjuk. A programmal adatokat közlünk, amelyeket feldolgoz és ujabb adatokat, amely eredményt szolgáltat.

A feldolgozási eljárás tehát ha minden bemenő adathoz egyértelmüen rendelünk kimenő adatot, függvényt.

A függvényt szemléletesen tehát tekinthetjük olyan gépnek, mondjuk darálónak, amelybe egy vagy több valamit beteszünk kijön belöle egy másik valami. Vannak olyan feldolgozási eljárások, amelyeket a programozók és felhasználók gyakran használnak.
Ilyen pl. a naptár példánál megismert "egész rész képzés".

Az igen gyakran szereplő részfeldolgozások egy részét, kis programok formájában beépitették gépbe. Ezért aztán elegendő a nevét kiirni és utána esetleg zárójelben odairni a bemenő adatot és a gép szolgáltatja az eredményt.
A PRINT INT /5.673/ utasitásban elöször az egészrész képzés programja fut le, előállitja az eredményt és azt irja ki. A bemenő adat azonban nemcsak szám lehet.
Pl.: PRINT LEN /"TV-BASIC"/
Ez pl. a cimfestők kedvenc utasitása, itt a bemenő adat egy szöveg a LEN nevü beépitett függvény a szöveg hosszát adja eredményül. Ezekből a beépitett programokból, vagyis függvényekből gépeink általában több tucatot ismernek. Pl.:

```
PRINT RIGHT$ /"MALAC FARKA",5/=FARKA
```

Ennek a függvénynek pl. két bemenő adata van, egy szöveg és egy szám. Kimenő adata egy szöveg, a bemenő 5 utolsó karaktere.

A következő függvénynek három bemenő adata van.

```
PRINT MID$ /"MIKROSZAMITOGEP",6,7/=SZAMITO
```

A kiment 7 betü a 6.-tól kezdve.
Ennek a függvénynek egy érdekes alkalmazása a következő programocska.

```
10 INPUT A$
20 FOR I=1 TO LEN A$
30 PRINT MID$ / A$, I,1/
40 NEXT I
```

```
RUN
? DARABOLO
D
A
R
A
B
O
L
O-
```

Ez tényleg darabol, ha azonban kicsit átalakitjuk:fordit.

```
1Ø INPUT A$
2Ø FOR I-LEN A$ TO 1 STEP-1
3Ø PRINT MID$ / A$,I, 1/
4Ø NEXT I
RUN
? FORDIT
TIDROF
```

A függvények egy másik csoportja a matematikai függvények. Ezeknek irásmódja hasonlit a hétköznapi irásmódhoz. Aki ismeri, ugyanugy használhatja őket mind eddig. aki viszont nem ismeri, annak érdekes ismerkedési módot ajánljunk.

Az adásban bemutatott programmal matematikai függvények képe rajzolható a képernyőre.
Ezután, ha nekiállunk a függvények paraméterein változtatni, érdekesebbnél érdekesebb ábrákat kaphatunk, közben megismerhetjük a függvények természetét.

A matematikai függvények közül a szögfüggvényeket kell egy kissé máshogy kezelni mint megszoktuk. A szögfüggvények változóit ugyanis radiánban kell megadni.

Az átszámitás könnyen elvégezhető, de nem szabad róla megfeledkezni. A szinusz és koszinusz függvény azért is érdemel külön figyelmet mert segitségükkel tudunk a képernyőre kört rajzolni.
Az ellipszist előállitó program a körétől alig különbözik.

További kis átalakitással pedig érdekes görbéket, az egymásra merőleges rezgések összetétekor keletkező Lissajous görbéket kapunk. Sőt mi több, a szöggfügvények segitségével tudunk órát szerkeszteni a képernyőre.
A bemutatott programok listáját megtaláljuk a függelékben.

Az eddig tárgyalt függvények két csoportot alkottak, az egyiknél a bemenő adat szám, a másiknál a bemenő adat szöveg. Előfordul, azonban, hogy egy szövegen hasonló átalakitást szeretnénk csinálni, mint egy számon, vagy forditva.
Pl. a 12355-nek csak az utolsó három jegyét akarom megtartani. Ez matematikai eszközökkel is megtehető, de kezelhetőbb lenne a RIGHT$ /12355,3/
Ha beirjuk a számot, hibajelzést kapunk. Irhatjuk azonban hogy

```
PRINT RIGHT$ /STR$/12355/,3/=355
```

A STR$ függvény bemenő adata szám, eredménye szöveg. Nem szabad elfelejteni, hogy az eredmény, a "355" ilyenkor szintén szöveg, ahhoz, hogy ismét számként használhassuk a VAL függvényt kell használni.

```
A=VAL /RIGHT$ /STR$/ 12355/,3
? A=355
```

Ebből az előbbi példából is jól láthattuk, hogy a függvények tetszés szerint használhatók a BASIC utasitásokban, minden esetben, ha az eredmény tipusa /szöveg-szám/ megfelelő. Összefoglalva áttekintjük a függvényeket.

Három szöveg-szöveg tipusu van, a LEFT$, a RIGHT$ és a MIDS/,. Kettő szöveg-szám tipusu, a VAL /érték/ és a LEN /hossz/. Egy szám-szöveg tipusu a STR$. Szám-szám tipusuak példája a matematikai függvények.

TANÁCSOK A TANULÁSHOZ

Az adásban bemutatott egyszerü programok begépelésén és tanulmányozásán kivül, javasoljuk a függvényrajzoló program elkészitését.
Átirása másik gépre kissé bonyolultabb feladat, de ezzel is megprobálkozhatunk.

A házi feladatként szereplő "e" betüket számoló program önálló megoldása már tudásellenőrző mérce lehet önmagunk számára.

A függvények témájához a tankönyv 8. fejezetében találunk rövid áttekintést.

Az egyes függvényutasitások müködésének tanulmányozásához pedig a könyv végén található táblázat ill. a rendelkezésre álló géptipus gépkönyvének fellapozása ad segitséget.

X. F Ü G G V É N Y E K II.

Az adás vázlatos tartalma:

Elöszőr a legutóbbi házi feladatnak - az "e" betük számolásának megoldása. Ime a lista

```
10 REM * E-SZAMOLo *
40 INPUT "KEREM A SZÖVEGET"; A$
50 LET GY=0
60 FOR I=1 TO LEN /A$/
70 B$=MID$ /A$/,I,1/
80 IF B$="e" THEN LET GY=GY+1
90 NEXT I: PRINT
100 PRINT GY
```

A beépotett függvények száma a számitógépeken mint emlitettük, több tucat, de korántsem annyiféle felhasználói lehetőség előfordulhat.
Nincs például olyan beépitett függvény, amely mondjuk egy lakatos kisiparosnak, aki kertkapukat készit, mindig kiszámitaná, hogy adott méretü kapuhoz mennyi szögvasat, vagy csövet kell venni és leszabni.
Pedig erre igazán gyakran szükség lenne. Ezen segitenek az u.n. felhasználó által definiálható függvények. DEF FN x =
A jobb oldalon tetszőleges x-nek kifejezése állhat. Ezután az igy definiált függvényre ugyanugy hivatkozhatunk, ugyanugy használhatjuk, mint a belső függvényeket.
Amelyik gépen nincs meg ez a lehetőség, ott szubrutinok irásával segithetünk magunkon. A felhasználó által definiált függvények néhány tipikus alkalmazása:

- Fok - radián átszámitás

- Pénznem átváltása
- Mértékegységek átváltása, stb.

Vitatkozhatunk róla, hogy az RND az utasitás, függvény-e vagy sem. Bizonyos értelemben függvény, mivel számitásokat végez és eredményt ad, viszont a számitások kiinduló bemenő értékét nem mi adjuk, hanem a gép.

Az RND egy intervallumba eső véletlen számot ad. Ez az intervallum egyes gépeknél a /Ø;1/ intervallum, más gépeknél elő lehet irni az intervallum határait. A keletkezet szám Ø lehet, az intervallum felső határértékével egyenlő nem lehet.

Az RND által kiadott számokban semmi szabályosságot nem látunk, pont ezért mondjuk véletlennek.

Az igazság az, hogy ezek nem véletlen számok, csak nagyon sok számból válogat a gép. Igy aztán álvéletlen szám a helyes megnevezés.

Könnyen nyerhetünk bármilyen intervallumba eső véletlen számot azokon a gépeken is, ahol nem lehet előirni az intervallum határait.
1ØØ * RND Ø és 1ØØ közötti 5ø + 5Ø * RND 5Ø és 1ØØ közötti véletlen számot ad.
A véletlen klasszikus megtastesitője a kockadobás. Próbáljuk meg ezt programmal utánozni.
6 * RND Ø és 6 közötti véletlen számot ad, de nekünk egész számra van szükségünk. INT /6*RND/
Viszont Ø és 5 közé eső egész véletlen számokat ad, hiszen az INT függvény lefelé kerekit.
A végső megoldás: 1+ INT /6*RND/

A véletlen szám előállitásának fomtos szerepe van általában a játékoknál, de bizonyos folyamatok modellezésénél is. Sőt arról is meggyőződhetünk, hogy a tanulást segitő programokban is helye van. Ilyen példa ugyanis található a TV BASIC tankönyvében a szorzótábla gyakorló program.

Mint minden program, ez is tökéletsithető. A számitógppel való foglalatoskodás ugy tünik, önmagában örömet okoz a gyerekeknek, a programmal való tanulás már olyannyira örömteli lehet, hogy már észre sem vesszük, hogy tanulunk.

Egy pillanatra térjünk vissza a véletlen ill. álvéletlen számokhoz. Ha kifejezetten ez a szándékunk, a RANDOMIZE /x/ utasitással elérhetjük, hogy a gép véletlen számok sorozatát mindig ugyanattól a tagtól inditja. Ez segithet például a program tesztelésénél, amikor nem előnyös, ha minden alkalommal más számokkal dolgozik a program.

Amiről még beszélünk, nem kapcsolódik szorosan a függvények témaköréhez, inkább az adatbevitelhez, bár nevezhető függvénynek is.
Az INKEY$-ről van szó. Az adatbevitelt eddig az INPUT utasitással bonyolitottuk. Ennek azonban több hátránya is van.

1. A képernyőn mindig ott a kérdőjel, ami nem mindig esztétikus. Képzeljünk magunk elé egy ragyogó grafikáju játékot, ahol a billentyüzettel kell irányitani valamit. Szóval a kérdőjel ilyenkor nem kivánkozik a képernyőre.
2. Minden esetben nyomni kell a sorzáró billentyüt.
3. Vessző és kettőspont nem vihető be a szövegben sem.

4. Hibás bevitel esetén hibaüzenetet ir ki a gép és a bevitelt meg kell ismételni. A hibaüzenet a képernyőn marad, ami sokszor nem kivánatos.

Létezik azonban másfajta beviteli utasitás is az INKEY$ ill. a COMMODORE-nál a GET

```
2Ø A$= INKEY$
```

Ennek a sornak a hatására az A$ változóba 1 betü /jel/ kerül, az amelyet az utasitás végrehajtásakor éppen bebillentyüzünk.

Egy rövid példaprogram:

```
1Ø A$=INKEY$
2Ø PRINT A$
3Ø GO TO 1Ø
```

Ha futtatjuk a programot, a lista elöszőr kifut a képből. Végtelen ciklus van. Ha lenyomunk egy billentyüt, a betü megjelenik, de rögtön ki is fut a képernyőről. Ennek magyarázata az, hogy amikor egy billentyüt sem nyomunk le a gép nem vár, hanem A$-ba egy Ø hosszuságu szöveget helyez A$=" " és ez nyomtatja ki a PRINT mindig uj sorba.

Megváltoztatjuk a programot, hogy csak akkor irjon ki, ha A$ nem üres.

Az INKEY$ utasitás /GET/ használata általánosan elterjedt:

1. Olyan esetekben, amikor a program választási lehetőséget kinál. Példa erre pl. egy raktárkezelő program, amely a képernyőn elöszőr megjeleniti a MENÜT;
2. Ugyancsak nagyon gyakran szerepel az INKEY$ játék programban az előbbi esettel ellentétben nem vár a gép és ha nem nyomtuk le időben a billentyüt, boszszankodhatunk.

Két utasitással foglalkozunk még, amelyek közül csak

az egyik egyik nevezhető függvénynek,a másik nem.
A PEEKn utasitás kikeresi a számitógép memóriájának n. sorszámu rekeszét és ennek tartalmát teszi a kijelölt változóba ill. kiirja.

Ennek az utasitásnak a forditottja a POKE m. n, amely a memoria m-edik rekeszébe tölti n értéket /N csak Ø és 225 közötti egész szám lehet./

TANÁCSOK A TANULÁSHOZ

Az adásban bemutatott utasitások leirását a könyv 8. 11. és 15. fejezetében találhatjuk meg, és természetesen az összefoglaló táblázatban.

Gyakorlásként érdemes néhány felhasználói függvényt elkésziteni. Az INKEY$ utasitás használatára legkézenfekvőbb példa a házi feladatként is szereplő rajzoló program, ez több-kevesebb változtatással minden géptipuson elkészithető.

XI. T Ö M B Ö K

Az adás vázlatos tartalma:

Gyakran szükséges, hogy a számitógép memóriájában sorban tárolt bizonyos adatokhoz a sorszámuk alapján is hozzá tudjunk férni. Tipikus példa erre mondjuk egy sportverseny eredményeinek nyilvántartása.
Ilyen és hasonló problémák megoldása tette szükségessé a BASIC-ban az ugynevezett tömbök használatát. Egy tömb azonos nevü változókból áll. A tömb elemeit a közös név után tett sorszámmal, vagy sorszámokkal különböztetjük meg egymástól. Tömböt alkothat a hónapok napjainak száma,

H/1/,	H/2/,	H/3/,	stb.
31	28	31	stb.

A hét napjainak nevei

NS/1/,	NS/2/
hétfő	kedd

vagy mondjuk egy osztály tanulóinak tanulmányi átlagai.

A/1/,	A/2/,	stb.

Térjünk vissza a változókkal kapcsolatos korábbi hasonlatunkhoz, mely szerint a gép tárolója egy fiókos szekrény, a névvel ellátott fiók a változó.
A tömb ebben a szekrényben egy olyan fiók, amely sorszámozott további fiokokra oszlik.
Ha a fiók neve F, a rekeszeké F/1/ F/2/ stb.
A fiók a tömb, a rekeszek az indexes vagy tömb változók.

A játékot folytatni is lehet. Pl. Az F/1-es alfiókot további alalalfiókra bonthatjuk. Ezeknek neve F/1,1/, F/1,2/, F/1,3/, f/1,4/ stb.
Ugyanezt megcsinálhatjuk a többi alfiókkal.
F/2,1/, F/2,2/, F/3,1/, F/3,2/ stb.
és máris előttünk áll egy kétdimenziós tömb.

És ha már belejöttünk, miért ne folytassuk.
Egyetlen fiókban már egy egész raktári polcrendszert hoztunk létre és mellesleg egy háromdimenziós tömböt.

Az eljárást lehet folytatni, elképzelni már egy kicsit nehezebb. Pedig ha utánanézünk a gépkönyvben, a gépek általában nem korlátozzák a definiálható tömbök dimenzióját. Elvileg akár száz dimenziós tömböt is meghatározhatunk.

Természetes, hogy egy ilyen polcrendszer elég tekintélyes helyet elfoglal a gép memóriájában.

A tömböket általában a program elején, de mindenesetre felhasználásuk előtt definiálni kell. Erre szolgál a DIM utasitás.
Pl. DIM A/3,5,2/ egy 3x5x2, azaz harminc rekeszből álló polcrendszert ad meg.
Ahogy a változók lehetnek szövegesek, ugyanugy létez-

nek string tömbök is. A konvenció ugyanez, a tömb nevét egy $ karakterrel kell kiegésziteni a végén.

Egy egyszerü feladat, amelyből a tömb változó használatának előnyei kiderülhetnek.
Van mondjuk egy érettségi előtt álló középiskolai osztály. Tegyük fel, hogy mi vagyunk az osztályfőnökök és

számitógépre visszük az osztálynaplót.
Egy kétdimenziós tömbben tároljuk a gyerekek tanulmányi eredményét az egyes érettségi tantárgyakból.
A tömb ismeretében sok mindent nagyon egyszerüen ki lehet programmal számitani.
Pl. az egyes gyerekek tanulmányi átlageredményét az érettségi tárgyakból, - az egyes tárgyak átlageredményét az osztályben, - az egész osztály összetett átlageredményét.
Hogy a program valóban osztálynaplóként müködjön, az szükséges, hogy ne csak az egyes tantárgyakból elért osztályzatok átlagát tároljuk. Ha minden egyes érdemjegyet tárolunk,nagyon helyigényes.
A megoldás: egy tömbben tároljuk az átlagosztályzatokat tanulónként és tantárgyanként. Egy másik ugyan ilyen méretü tömbben tárpljuk, hogy az előbbi átlag hány érdemjegy átlaga.
Igy ha mondjuk Kovács Pisti matekból ötösre felel, ki tudom számitami az uj átlagosztályzatát.
/Régi átlag X osztályzatok száma + uj osztályzat, osztva osztályzatok száma + 1./

Ha a gyerekekre nem számokkal akarok hivatkozni, egy külön egydimenziós stringtömbben kell tárolni a tanulók neveit.
A tömbök alkalmazásának bemutatására szerepel az adásba a sporteredményeket nyilvántartó program, amely tömbben tárolja a versenyzők neveit, és elért eredményeiket. Ennek a programnak az elkészitése során merül fel a sorbarendezés szükségessége. A sorbarendezés lényegét egy lehetséges egyszerü algoritmussal is bemutatja az adás.

Kézenfekvő a gondolat, hogy az ugynevezett táblás játékok pillanatnyi állását kétdimenziós tömbökben ábrázoljuk.
Erre is látunk példát az adás végén az u.n. játék bemutatásakor. A program végleges formájának elkészitése a nézők feladata.

A bemutatott programok az oktatócsomag mellékletben ill. adathordozók is megtalálhatók.

TANÁCS A TANULÁSHOZ

A tömbökkel kapcsolatos tudnivalókat a tankönyv 9. fejezetében találhatják meg egy egyszerü példa kapcsán kifejtve.

Gyakorlásként egy naptár /ill. öröknaptár/kiiró program elkészitését, valamint a torpedó program megtervezését és kódolását javasolhatjuk.

```
 1 REM * EREDMENY NYILVÁNTARTÁS *
 3 INPUT " A RESZTVEVŐK SZAMA"; V
 5 DIM N$ /V/, E/V/
 1Ø REM * ELŐKESZITES
 15 PRINT "O NEVBEIRÁS"
 2Ø FOR R=1 TO V
 4Ø PRINT
 5Ø PRINT "RAJTSZAM:"; R;...NEV";
 6Ø INPUT N$ /R/
 7Ø NEXT R
1ØØ REM * EREDMENYBEIRÁS *
11Ø INPUT "RAJTSZAM, EREDMENY:"; R,e/R/
14Ø IF R>O THEN 12Ø
```

```
2ØØ REM * RENDEZES *
21Ø C=Ø
22Ø FOR I=2 TO V
23Ø IF E/I-1/= E/I/ THEN 28Ø
24Ø M=E/I/ : M$ = N$ /I/
25Ø E/I/=E/I-1/ : N$ /I/ = N$/I-1/
26Ø E/I-1/=M : N$ /I-1/= M$
27Ø C=1
28Ø NEXT I
29Ø IF C<>O THEN 21Ø
3ØØ REM * KIIRÁS *
31Ø PRINT "HELYEZES NEV PONT"
32Ø FOR R=1 TO V
33Ø PRINT
34Ø PRINT R; TAB/1Ø/; N$/R/; TAB/31/; E/R/
35Ø NEXT R
```

XII. J Á T É K P R O G R A M O K

Az adás rövid tartalma:

Ha a személyi számitógépeknek a legtöbb embert érintő alkalmazását keressük, valószinüleg a játékot kell az első helyen emlitenünk. Valóban a müszaki és gazdasági számitások, adatfeldolgozási feladatok mellett a számitógép szórakoztató, de mindenesetre képességfejlesztő szerepet is játszik.

A számitógép azért kedvelt játszótárs, mert változatos játékokat kinál, a játékos akciójára gyorsan válaszol, értékeli annak eredményeit és végül:nem csal. A számitógépes játékok és a BASIC nyelv - noha nem minden játék program irodott ezen a nyelven - erősen összekapcsolódtak. Ennek az az oka, hogy nagyon sokan vannak, akik csupán szórakozásból - egyéb komolyabb feladataik mellett - BASIC nyelvü játékprogramot irnak. Többek között ezért is szinte kötelező, hogy egy BASIC tanfolyam témái között a játékprogramozás is szerepeljen.

Mindemellett a játékok általában sok ötletet igényelnek, sok alternativát kezelnek, sok funkciót valósitanak meg, tehát jó erőprobát jelentenek a tanuló programozók számára. Nem véletlen tehát, hogy mi is foglalkozunk velük.
Szót ejtünk az adásban arról, hogy egyáltalán milyen tipusu játékprogramok léteznek, s ezek közül melyik valósitható meg BASIC nyelven.

Másrészt arról, hogy mi a játékprogramok jellegzetessége, miben különböznek más programoktól.
Többféle játéktipus létezik. A csoportositásnál kiindulási szempont elöszőr maga a játék. Léteznek u.n. kaland tipusu játékok.

Második csoport az akció tipusú játékok, amelyek ügyességet, reflexet igényelnek és fejlesztenek. Ezekben el kell szaladni, ugrani, lőni a megfelelő pillanatban.

Léteznek u.n. stratégiai játékok, ezek valemely egyébként is közkedvelt játék számitógépes megvalósitásai.

A szimulációs játékok: vagy valamilyen fizikai folyamatot szimultálnak, pl. a repülést illetve a repülőgépvezetést, vagy valamilyen helyzetet utánoznak: ilyenek a jégkorong, a foci, stb.

A játékok kimenetele itt a játékostól és a véletlentől függ és a sok variációs lehetőség és a véletlen események miatt válik érdekessé. A fokozatosságot a játék. programozásban is be kell tartani.
Viszont léteznek aránylag eszközökkel megvalósitható, mégis igen izgalmas játékok.
Jó példa erre a szunyog!

A játék lényege: a képernyőn megjelenik egy szoba ággyal és a minket megszemélyesitő figura. Véletlenszerüen megjelennek a szunyogok a szobában. Ezeket kell adott idő alatt összefogdosni, majd az ágyhoz menni és lefeküdni aludni. Ha nem sikerül összefogdosni a szunyogokat, azok nem hagynak aludni, ami kellemetlen no

meg pontlevonással jár.
Amennyiben lehet egy ilyen rövidke program modulszerkezetéről beszélni, a program a következő modulokból áll:
- Kiirja röviden a játékszabályokat / ha nem fér a memóriába, ez elhagyható/.
- Felrajzolja a szobát és az ágyat.
- Véletlenszerüen generálja a szunyogokat
- Billentyü figyeléssel megoldja a figura mozgatását. Itt külön probléma a falnak ütközés esete.
- Figyeli a szunyogok elkapását.
- Megvizsgálja az eredményt.
- Az eredmény alakulását esetleg rosszmáju megjegyzésekkel kiséri.

Ezután nézzük meg milyenek a jó játékprogram funkciói és tulajdonságai.
A gép egyrészt játszik, tehát személyt, helyzetet, eseményt helyettesit, szimulál.
Másrészt adminisztrál, megjeleniti az eredményt, a játéktáblát, jelzi a győzelmet, a nyereményt, stb.

A tulajdonságok:
- Jó megjelenés, ha szövegben érintkezik a gép a felhasználóval, a szöveg áttekinthető, világos legyen.
- Ugyanugy látható legyen a játéktér, a tábla, az eredmény megjelenése.
- Fejlett társalgási nyelv. Egy jó játékprogramnak illik bemutatkozni. Ismertetni a szabályokat. Esetleg megjegyezni a partner nevét. Mindent jeleznie kell, ha vár, hogy miért, hogy mire vár, ha léptünk, válaszoltunk, hogy lépésünket megértette. Ha rosszul léptünk figyelmeztet, stb.

- Fontos: A játékprogramban nem fordulhat elő gépi hibajelzés. Gondoljuk el milyen illuzióromboló, ha egy szépen szerkesztett kép közepén megjelenik:

? DIVISION BY ZERO ERROR

Milyen feladatokat jelent egy játék megvalósitása a programozó számára.

- Meg kell ismerni /ki kell találni/ a játék algoritmusát és azt gépre kell vinni. Ez tartalmazhatja a játék szabályait, milyen feltételek mellett ki nyer.
- Véletlen generálással elő kell állitani valamely kezdő vagy közbeeső helyzetet.
- Érzékelni és jelezni kell a célbaérés tényét.
  Meg kell oldani a játék figuráinak mozgatását a képernyőn. Ez, mint a grafikánál már láttuk, kitörléssel és ujrarajzolással megvalósitható.
- Mérni kell az időt, számlálni az eredményt, ez általában ciklusutasitással történik.

Talán feltünt, hogy a pédák között nem volt szó olyan játékokról, mint pl. a repülésszimuláció vagy a létrásemberke.
A képeken a mozgás látványát az biztositja, hogy a gép másodpercenként 15-2o-szor változtatja az emberke helyzetét, közben figyeli, hogy nem esett-e lyukba, nem ütközött-e, stb.
A repülésszimulációnál a kép csak kb. o.5 másodpercenként változik, de teljes egészében ujrarajzolódik és közben a gép 1o-15 differenciát egyenlettel számol. Ilyen sebességü számitást és képrajzolást BASIC program nem tud végezni. Ezt bizonyitja egy egyszerü, játéknak is tekinthető programocska.

```
1Ø FOR I=1 TO 999
```

```
2Ø IF AND /Ø/-Ø.5 THEN PRINT "/": GO TO 4Ø
3Ø PRINT " "
4Ø NEXT I
5Ø GET AØ; IF AØ=" "THEN GO TO 5Ø
RUN
```

A gép minden karakter helyére vagy / jelet rajzol. A képernyő teleirása a COMMODORE számitógépen 14 másodpercig tart, ami egy repülésszimulációnál bizony lassu lenne. A játékprogramoknak gazdag irodalma és piaca van, de sokkal izgalmasabb ujat irni.

TANULÁSI TANÁCSOK

A játékprogramozás nem vizsgaanyag, ennek ellenére jó erőpróba lehet bárkinek egy nem túl bonyolult játékprogram elkészitése. Mindenképpen hasznos egy játékprogram müködésének megértése, szerkezetének tanulmányozása. Itt az alkalom, hogy a sorozat elején szereplő emberevő programot bonckés alá vegyük, megértéséhez minden ismeret a birtokunkban van.
A korábbi adásban feladott és ebben az adásban bemutatott TORPEDÓ c. játékprogram ugyancsak szerepel a mellékletben ill. adathordozón.
A müsorban illusztrációként bemutatott gyári készitésü profi játékprogramok természetesen védettek és igy nem lehetnek részei oktatócsomagunknak.

XIII. GRAFIKAI LEHETŐSÉGEK

Az adás vázlatos tartalma:

A számitástechnikai eredményei közül - különösen kivülállok a számitógépes grafikát tartják a leglenyügőzőbbnek.
A legutóbbi időkig, a személyi számitógépek elterjedéséig a számitógépes grafikát olyan területeken alakalmaztak elsősorban, ahol a kutatás, a tervezés vagy a gyártás alapja rajz volt.
Napjainkban a számitógépes grafika alkalmazása terjedőben van az ügyvitel területén is, ahol a grafikonok, oszlop- és kördiagaramok, statisztikai térképek sokkal áttekinthetőbb és könyebben elemezhető formában tartalmazzák az eredményeit, mint a korábban megszokott táblázatok.
Ma tehát a grafika az ember-gép kapcsolata+ emberibbé tételének eszköze is. A sorozatban többek között ezért kapott hangsulyt.

Mint már emlitettük, a BASIC nem grafikai alkalmazásokra kifejlesztett programozási nyelv, a különböző géptipusok grafikai lehetőségei még annyira sem egységesek, mint a BASIC egyéb területei. A gyártók egyéni elképzelései és érdekei itt sokkal jobban érvényesülhettek, mint a többé-kevésbé szabványositott BASIC-ban.
A gép műszaki adottságai eléggé meghatározók. A különkülön megjelenithető grafikus elemek száma, a felbontás, a müveletvégzés sebessége, valamint az, hogy az előállitott kép szines-e vagy sem, eleve eldöntik az alkalmazási lehetőségeket. Természetesen ötletességgel, türelem-

mel és sok munkával itt is sok mindent pótolni lehet, de pl. a szineket nem.
Az adásban a teljesség igénye nélkül felsorakoztatunk néhány grafikai lehetőséget, effektust, amelyek BASIC programokban is alkalmazhatók.
Keretezésről, a képernyő szélével párhuzamos vonalak rajzolásáról korábban már szó esett. Viszont ugyancsak gyakran szükség van pl. grafikon készitésekor két pont összekötésére egyenes szakasszal.
Ennek programkészlete bár nem tul bonyolult, kicsivel több utángondolást igényel, mint a keretezés. Pl egy pont koordinátája csak egész lehet, a gép kerekit, ebből bonyodalmak adódnak. A gyengébb grafikáju gépeken, mint a ZX 81. és a HT, érdemes a lépésközt megnövelni és inkább szaggatotr vonalat rajzolni.Igy kevésbé látszik lépcsőzetesnek.

A pontösszekötő programot érdemes minél általánosabb formában szubrutinként megirni.
A körrajzolásról szintén esett szó a szögfüggvények kapcsán. Csak néhány kiegészités.
- Szintén érdemes belőle szubrutint csinálni /ha nincs eleve a gépen/.
- A durvább felbontásu gépeken itt is inkább hagyjunk ki pontokat, a szaggatott kör kevésbé látszik lépcsősnek.
- Néhány érdekes effektus.
  Több kört rajzolunk azonos középponttal, növekvő sugárral = céltábla.
  Ugyanez, ha az előző kisebb sugarú kört kitöröljük = felfujódó léggömb.
- Ha az előző kört kitöröljük, következő fázisban ismét berajzoljuk, hullámszerü látványt kapunk..

Az adásban ugyancsak bemutatunk egy sokszög rajzoló programot, amely kellően nagy oldalszám esetén már kört rajzol. Listája az egyenesrajzoló programéval együtt megtalálható a mellékletben.
Szinte valamennyi gépen találhatunk olyan karaktereket, amelyek nem betük, nem számok, nem irásjelek, hanem valamilyen grafikus ábrát jelenitenek meg. Ezek kombinálása, egymás mellé rakása sok esetben segit valamely bonyolultabb figura, ábra kialakitásában.
A SINCLAIR és a PRIMO számitógépeken ezen túlmenően lehetőség van u.n. felhasználási grafikus karakterek definiálására. Ezeket a SPECTRUM esetében egy 8x8-as rácsmezőben magunk tervezhetjük meg. A SPECTRUM-nál az igy megtervezett karaktert 8db bináris szám formájában vihetjük be a gépbe és ettől fogva a következő kikapcsolásig használhatjuk. Ahol a háttértől eltérő szint akarunk ott 1-et irunk, ahol egyezőt, ott Ø-át.

A PRIMO esetében a bemutató kazettán található karekter nevü programmal végezhetjük el a definiálást igen kényelmesen.
A magunk deffiniálta karakterek felhasználásával érdekes hatásokat érhetünk el. Rakjunk össze pl. egy gépkocsit. Keréknek kétféle karaktert is definiáljunk, a kerék két különböző forgási fáziásnak megfelelőt. Ha ezeket megfelelő ütemben váltogatjuk, ugy látszik, mintha a kerék forogna.
Másik példa: ha velemilyen játékprogramhoz labirintust akarunk készitени, érdemes un. "sövény" karaktert definiálni.
Egy karakteren vagy soron belül az előtér ill. háttér szin váltogatásával, villogtatásával figyelemfelkeltő hatást érhetünk el.Ennek egy felhasználói területe az

u.n. menü, amikor egy felhasználói program a felhasználó választásától függően más-más részfeladatot old meg. Ilyenkor a választás jelzésére megfelelő a villogtatás. Az egész képernyő háttérszinének váltogatása esetleg hanghatással kombinálva a robbanás, az összeomlás, az összeütközés illuzióját kelti, amint erre a játékprogramban jócskán láttunk példát.

A számitógépes grafika népszerüségének növekedésével egyre szaporodnak a grafikát segitő készen kapható segédprogramok és eszközök is. Ezek közül a Cambell nevü, a különböző stilusu és tipusu betükkel való irást teszi lehetővé. Mi is használtuk egy müsorban.
A Melbourne Draw a rajzkészitést könnyiti meg.

A COMMODORE-hoz kapható grafikai segédletek közül a PANDA egy programból /kazettán/ és egy berendezésből áll.

Ez egy palatábla szerü kis szerkezet egy kis pálcikával és két gombbal. A program indulásakor felrajzolja a menüt, vagyis hogy milyen tipusu vonalak, idomok rajzolhatók és milyen szinüre szinezhetők. A két gomb közül az egyikkel visszahivható a menü, a másikkal véglegesithető a megrajzolt ábrakészlet.

A képen diszken tárolhatók és visszatölthetők. Mint ez is itt, amelyet profi grafikus készitett kb. 15 perc alatt.

Az adásban szó esik a READ és a DATA utasitásokról is egy morse kódoló program kapcsán.
A grafikáról azért két dolgot nem árt negemliteni.
Az egyik: a finom felbontású, pláne szines grafikához sok memória kell.

A másik: a nagyfelbontású mozgó grafikához sok-sok művelet másodpercenként, vagyis nagy sebességü számitógép kell. Ezért aztán nem kell csodálkoznunk, ha kis mikrógépünkkel nem tudjuk leutánozni mondjuk a Csillagok háborúja c. film jeleneteit.
Persze a fejlődés itt sem áll meg. Nap, mint nap jelennek meg az egyre gyorsabb kapacitásu, ugyanakkor egyre kisebb méretü számitógépek.
És teszik, ha nem is valósággá, de láthatóvá a lehetetlent.

TANULÁSI TANÁCSOK

A grafikai lehetőségekkel kapcsolatban elsősorban az egyes géptipusok gépkönyveire támaszkodhatunk.
Mindenképpen érdemes elkezdeni egy grafikai szubrutin gyüjtemény kiépitését, kezdve a pontrajzolástól, az egyenes, a sokszög húzásán át a körrajzolásig, "satirozó" vagy szinező programokig, vagy az oszlopdiagram készitéséig.

EGYENESRAJZOLÓ PROGRAM

```
3ØØ INPUT "KEZDŐPONT"; X1, Y1
31Ø INPUT " VEGPONT"; X2, Y2
32Ø IF X2=X1 AND Y1=Y2 THEN SET /X1,Y1/: GOTO
    59Ø
33Ø IF X1=X2 THEN L=Ø : GO TO 5ØØ
34ø IF Y1=Y2 THEN M=Ø : GO TO 4ØØ
35Ø M= /Y2-Y1/ /X2-X1/
36Ø IF ABS/M/ 1 THEN L=1/M : GO TO 5ØØ
4ØØ Y=INT //X-X1/*M+Y1+0.5/
42Ø SET/X,Y/
43Ø NEXT X
44Ø GO TO 59Ø
5ØØ/ FOR Y=Y1 TO Y2 STEP SGN/Y2-Y1/
51Ø X= INT /Y-Y1/* L+X1+0.5/
52Ø SET /X,Y/
53Ø NEXT Y
59Ø STOP /RETURN/
```

SOKSZÖG/KÖR/RAJZOLÓ PROGRAM

```
 9ØØ INPUT "KOZÉPPONT"; XK,YK
 91Ø INPUT "SUGAR"; R
 92Ø INPUT "OLDAL"; N
 93Ø Xé=XK+R:Y1=YK
 94Ø FOR FI=O TO 36Ø STEP 36Ø N
 95Ø  F2=F1*PI/18Ø
 96Ø X2=INT/XK+R*COS/F2//
 97Ø Y2=INT/YK+R*SIN/F2//
 98Ø GOSUB 32Ø
 99Ø X1=X2:Y1=Y2
1ØØØ NEXT FI
1Ø1Ø STOP
```

```
  1 REM ж MORSE ж
  2 PRINT "O" /CLS/
  5 DIM A$ /26/
 1Ø FOR I=Ø TO 25
 2Ø READ A$ /I/
 3Ø NEXT I
 4Ø INPUT "KEREM A SZOVEGET"; T$
 5Ø K=LEN/T$/
 6Ø FOR J=1 TOK
 7Ø M$ MID$/T$,J,1/
 8Ø PRINT A$/ASC/M$/-ASC/"A"//;" ";
 9Ø NEXT J
 1ØØ PRINT: PRINT: PRINT
 9ØØ REAM ADATOK
1ØØØ DATA.-, -..., -.-., -.., ., ..-., --.
1Ø1Ø DATA ...., .., .---, -.-, .-.., --
1Ø2Ø DATA -., ---, .--, --.-, .-., ..., -
1Ø3Ø DATA ..-, ...-, .--, -..-, -.--, --.
```

XIV. GRAFIKA A COMMODORE-ON

ADATÁLLOMÁNYOK

Az adás vázlatos tartalma:

A személyi számitógépek rajzolási lehetőségeiről megállapitottuk, hogy általában itt a legnagyobb az eltérés az egyéb gépek között. Az is kiderült, hogy a BASIC viszonylagos lassusága nem kedvez a grafikus megjelenitésnek, főleg ha mozgást is szeretnénk a képernyőre varázsolni.

A COMMODORE gép ebből a szempontból jelentősen különbözik a többitől. A BASIC-ből közvetlenül kezelhető grafikája nincs is. Viszont rendelkezik egy rendkivül érdekes és hatásos folt /Sprite/ rajzolási technikával, amely mozgó ábrák megjelenitését is lehetővé teszi. A COMMODORE rajzolási technikájának elsajátitásához azonban mélyebben, jobban bele kell bujnunk a számitógépbe, jobban meg kell ismernünk a szerkezetét, hiszen mint mondtuk, nincsenek külön grafikus BASIC utasitások. A meglevőket kell felhasználnunk, amelyek nyilván nem rajzolásra tervezettek.

Emlékezzünk vissza, hogy amikor a változókról beszéltünk, a számitógép memóriáját egy fiókos szekrényhez hasonlitottuk. A fiók felirata volt a változó neve, a fiókban tárolhattuk a különböző adatokat, számokat és betüket.

Nos, ha tovább szeretnénk vinni a hasonlatot, azt kell mondanunk, hogy a fiókok tovább oszthatók rekeszekre. Egy rekeszbe egy karaktert, betüt, számot vagy irásjelet tudunk elhelyezni.

A rekesz, vagy byte /bájt/, a legkisebb memória elem, amelyhez közvetlenül hozzá tudunk férni. A rekeszeket a gép a fiókfelosztástól függetlenül megszámozza Ø-tól a COMMODORE esetében 65536-ig. Ezek a számok a memória-rekesz cimei. A memória legkisebb cimezhető része tehát a byte. Amikor programunkban kijelölünk egy változót, a gép feljegyzi ennek nevét, majd hozzárendel ehhez, a változó tipusától függően megfelelő számu bájtot és feljegyzi ezek cimét a memóriában.

A számitógép elektronikus eszköz lévén, nyilván senki sem gondol arra, hogy ezekbe a rekeszekbe a betü képét vagy rajzát helyezi el a gép, hanem inkább valamiféle kódját.
1 byte 8 bitből áll, egy bit pedig egy két állapotu áramkori elemnek felel meg.

Amikor a memória egy byte-jába irunk, a gép a byte bitjeit állitja megfelelő helyzetbe, az adatnak megfelelően. A byte kiolvasásakor pedig azt nézi meg, milyen állapotban vannak az egyes bitek.
Ha csak egy bit tartalmát akarjuk vizsgálni vagy módositani, kénytelenek vagyunk az egész byte-tal foglalkozni, t.i. az a legkisebb hozzáférhető egység.

A tár egy byte-jába - amint azt már emlitettük - a POKE BASIC utasitással lehet valamilyen adatot beirni. Formája:

x POKE cim, tartalom.

A cim Ø-65535-ig terjedhet, mármint a COMMODORE-nál. A tartalom egy max. 255 értékü tizes számrendszerbeli szám. A byte-ba természetesen ennek a számnak a kettes számrendszerbeli megfelelője kerül.
A 255 pl. csupa egyes értékü bitet, a Ø csupa Ø értékü

bitet, a 2-es hátulról a második helyen 1-es, a többin Ø-át jelent. Igy a POKE-kal megoldható a tár egyes bitjeinek megfelelő beállitása. Amint látni fogjuk, erre szükség is lesz a grafikánál. A tár valamelyik byte-ját a PEEK függvénnyel olvashatjuk ki. A cimtartomány ugyanaz mint előbb és a kiolvasott érték is. Először POKE-kal beviszünk egy számot egy cimre, majd PEEK-kel kiolvashatjuk.
Legyen pl POKE 1524,83
PEEK
Ilyenkor a képernyő tartalma megváltozik.

Igy szerzett tapasztalatainkból arra lehet következtetni, hogy a gép a memória egy részét használja a képernyő tartalmának és a vele való tevékenységeknek a tárolására.

Ez a többi gépnél is igy van, csak idáig nem kényszerültünk foglalkozni vele. Alaphelyzetben 1624-2Ø23-ig tárolódik a képernyőt alkotó 25x4Ø=1ØØØ karakter helyi tartalma, 55296-56295 pedig ezek szinre vonatkozó adatainak kódja.

Van a COMMODORE-on másik grafikai megjelenitési mód is. Ehhez a képernyőt vizszintes irányban 32Ø, függőleges irányban pedig 2ØØ pontra bontja fel. A pontokat ugyszintén a képmemória byte-ok feltöltésével lehet ábrázolni.
Ezen kivül meghatározhatunk max. 8 db u.n. sprite-ot, vagy foltot. Egy folt a képernyőn vizszintes irányban 24, függőleges irányban pedig 21 pontból áll /5Ø4/. A folt a képernyőn bárhova elhelyezhető és áthelyezhető /mozgás/. A képernyőn egyszerre legfeljebb 8 folt jelenithető meg, de ezek ki és bekapcsolható, azaz eltüntethetők és visszaállithatók.

A foltokat mind vizszintes, mind függőleges irányba kétszeresére lehet nagyitani.
A folt minden pontjának megfelel a memóriában egy bit. Ha a bit értéke Ø, a pont szine a háttér szinével egyezik meg, vagyis nem látható. Ha 1, akkor ettől eltérő. Ezt a tényt lehet felhasználni a folt alakjának a megtervezésére. Egy folt 5Ø4 pontja tehát 63 byte-ot foglal le a memóriában összesen. Az egyszerüség kedvéért ezt még kiegészitik 64-re.

Hogy ez a 64 byte-os egység hol található a memóriában, azt az u.n. folt mutató byte-ok adják meg. Ezekből ezek szerint tehát 8 db-nak kell lenni.

A foltmutatóban lévő szám határozza meg, hogy a tár hányadik 64 byte hosszuságu darabjában található a folt meghatározása. Akárhová természetesen nem tehetjük, erre vannak megkötések. A POKE 2Ø46,192 pl. 12288-as cimtől helyezi el a folt adatait.
A foltokat az 53269 cimü byte megfelelő bitjeinek 1-re állitásával lehet megjeleniteni /bekapcsolni/, kikapcsolni ugyanezeknek a biteknek Ø-ra állitásával.

Ha pl. a harmadik foltot akarjuk bekapcsolni 8-at kell irni az 53269 cimü byte-ba.
A foltokat 16 különböző szinben lehet megjeleniteni. Minden folthoz tartozik még egy szin byte, amelynek tartalma meghatározza a folt szinét.

Külön byte, az 53277-ik ill. a 53271. határozza meg a folt méreteinek vizszintes ill. függőleges irányu kétszerezését, hasonló elven mint a bekapcsolásnál.

A folt helyének megadása a legfelső sor és a bal oldali oszlop helyének meghatározásával lehetséges. Függőlege-

sen 25Ø, vizszintesen 343 pozicióba helyezhető a folt, tehát részben kerülhet a képernyő fölé és mellé. Ilyenkor csak egy része látható.

Vizszintes irányban 1 byte nem elegendő a 343 különböző helyzet tárolására, ehhez 9 bitre van szükség. A 8 folt 9. bitjei egy különálló byte-ot alkotnak.

Ezenkivül megoldható még a sprite-ok átfedése és ütköztetése is.

A foltok meghatározása hasonlóan történik a felhasználó által definiált karaktereknél. Mivel biteket utasitással nem tudunk állitani, egyszerre kell egy egész byteot. Ehhez pedig a byte értékét 1Ø-es számrendszerben kell megadni a POKE utasitáshoz.

A módszer a következő:
Veszünk egy kockás papirt és felrajzoljuk a folt határait, belerajzoljuk a byte-ok határait és az egyes bitek fölé irjuk a kettes számrendszerbeli helyzetüket. Ahol az alaptól eltérő szint akarunk, oda 1-et irunk. A megfelelő helyi értékeket byte-onként összeadjuk. Az igy kapott számokat POKE-oljuk a sprite memória megfelelő cimü rekeszébe.
Hogy a karakterdefiniálással való hasonlóság teljes legyen, készitsük el sprite-ok felhasználásával a TV BASIC feliratot. A már emlitett tulajdonságokból adódóan a COMMODORE-hoz meglehetősen sok grafikai segédprogram létezik. Ilyen az u.n. sprite editor is, amelynek müködését bemutatjuk.

Már ebben az adásban elkezdik az előadók a következő téma az adatállományok ismertetését:

Az adatállományban /nem a számitógépben, hanem valamilyen külső adathordozón pl. mágneslemezen/ nagy mennyiségü adatot tárolhatunk. Az adatállományok szerkezete hasonlit a tömbökéhez. Az egy dologhoz, valamihez tartozó adatcsoportot nevezzük rekordnak, az adatok a rekord mezői.

A rekordok alkotják az adatállományt. Az adatállományok kezelésének lépései: létrehozás, feldolgozás, módositás. Ennek során a rekordokkal kell müveleteket végeznünk, előbb azonban kapcsolatot kell teremteni az adatállománnyal, azaz meg kell nyitni. A lépések tehát: megnyitás, rekord, müveletek, lezárás. Tehát a müveletek végeztével az állományt le is kell zárni.

Az adásban egy egyszerü anyagnyilvántartó adatállomány létrehozása kapcsán ismerjük meg ennek a folyamatnak a lépéseit.

Az adatállományt létrehozó program listája COMMODORE számitógépre

```
 1Ø REM ¤LÉTREHOZÁS
 2Ø REM ¤MEGNYITÁS¤
 7Ø OPEN2,8,2, "Ø:ANYAG,S,W"
 8Ø REM ¤ADATBEOLVASÁS¤
 9Ø PRINT "O" /CLS/
1ØØ INPUT "AZONOS ITO? ";A
11Ø PRINT "     12345678Ø1234567Ø"
12Ø INPUT "NEV:  ";NØ
13Ø INPUT "MENNYISEGIEGYSEG /KG,DB,M/:"; MØ
14Ø INPUT "MENNYISEG? ":M
15Ø INPUT "ELSZÁMOLÓ ÁR:";AR
2ØØ REM ¤IRAS AZ ALLOMANYRA"
21Ø PRINT ¤2,A;",";NØ;",";MØ;",";M;","AR
```

```
23Ø INPUT "FOLYTATJA? /I/N/:"F$
24Ø IF F$   "N" THEN 8Ø
26Ø REM xLEZARASx
27Ø CLOSEZ
28Ø END
```

TANÁCSOK A TANULÁSHOZ

Az adás témájával nyilván csak azoknak érdemes foglalkozni, akik COMMODORE gépen szándékoznak grafikai programokat készíteni.
Az anyag részletes leirása a tankönyvben is megtalálható. Meg kell jegyezni, hogy a COMMODORE grafikájához meglehetősen sok segédprogram létezik.
Ha egyszer-kétszer a lényeg megértéséhez érdemes is egy-egy sprite kódolását elvégezni, mindenképpen célszerü legalább egy sprite-editor program beszerzése.

Az adatállományok témaköréhez, amely ugyancsak bővebb kifejtést nyert a tankönyvben, ajánljuk a 13. fejezet 9. példájának tanulmányozását is.

XV. A D A T Á L L O M Á N Y O K

Az adás vázlatos tartalma:

Általában a számitástechnikának, de a személyi számitógépeknek is egyik fontos alkalmazási területe a nagy mennyiségü adat tárolása, kezelése és feldolgozása. Gondolhatunk itt akár egy raktár anyagnyilvántartására, egy adott körzet lakosságának pl. egészségügyi adataira éppugy, mint helyfoglalás adminisztrálására.
A nagy mennyiségü összetartozó adat tárolására két mód van. A programon belül - mint láttuk, például DATA sorokban, vagy változókban, tömbökben.
Ilyen módszerrel azonban csak max. néhány száz egyed adatait lehet tárolni.

Ezzel szemben, a programtól független, a programon kivüli tárolás esetén elvileg végtelen, de a gyakorlatban is igen sok adat tárolására nyilik lehetőség. Az adatokat ilyenkor mágnesszalagon vagy mágneslemezen tároljuk, igy korlátot csak ezen tárolóeszközök kapacitása jelent. Ez mindenesetre nagyságrendekkel nagyobb, mint ha a programban kellene adatokat tárolnunk.
Adatokat a programtól függetlenül u.n. adatállományokban lehet tárolni. Az adatállomány összetartozó adatok valamilyen rendezettségben tárolt gyüjteménye.
Ez a függetlenség azt is jelenti, hogy a meglévő adatállomány feldolgozására bármikor készithetünk uj programokat.

Foglaljuk össze az adatállományokkal kapcsolatos tudnivalókat. A programok mágneses adathordozókra történő kimentésénél láttuk, hogy az azonositáshoz szükséges volt nevet adni a programnak. Ez itt is igaz, az adat-

állomány azonositásához nevet kell neki adni, hogy a többi közül ki lehessen választani a szükségeset. A továbbiakban maradunk az előző adásban emlitett példánál és leltározzuk a tv studiót. Az adatállomány neve ANYAG. A dolog természetéből következik, hogy az adatállománynak a raktárban található anyagok nevét, mennyiségét, árát, értékét kell tartalmaznia.

Mint tudjuk, egy anyaghoz kapcsolódó információk együttesének neve rekord. A rekord tartalmazhat nem az anyaghoz, hanem a tároláshoz kapcsolódó információt is. Ilyen pl. a rekord sorszáma.
Esetünkben egy rekord a következő adatokat tartalmazza:

anyagazonositó /szám/
anyagnév /betü/
mennyiség egység /db. méter, kg/
mennyiség
elszámolási ár

Az adatállományban tehát ezek az adatok szerepelnek, minden a raktárban megtalálható anyagra külön-külön.

Egy adatállománnyal kapcsolatban háromféle müvelet szokott előfordulni:

- létre kell hozni /ez már megtörtént/
- feldolgozást kell végrehajtani a benne lévő adatok felhasználásával /pl. összesitès, leltár/
- módositani kell.

Mielőtt azonban az adatállomány rekordjaival bármit kezdenénk, kapcsolatot kell teremteni a gép és az állomány között. Ezt nevezik az állomány megnyitásának. Csak azután tudjuk a rekordokat a számitógépbe beolvasni és adataival számolást végrehajtani.

A feldolgozás végén ezt a kapcsolatot meg is kell szüntetni, vagyis az állományt le kell zárni.
A megnyitásra és zárásra vonatkozó utasitások COMMODORE-nál és PRIMO-nál.

```
OPEN i,8,j,"Ø:név,S,W"
i=logikai szám
8=lemezegység
j=csatornaszám
S=szekvenciális
W=irás
```

PRIMO:

```
OPEN "file név" olvasásra
CREATE "file név" irásra
```

A megnyitott állományból az INPUT utasitással olvashatunk be egy rekordot és a PRINT utasitással irhatunk ki egy rekordot. Az állományt mindkét gépnél a CLOSE utasitással zárhatjuk le.

Egy ilyen számitógépes rendszertől a következőket szokták elvárni:

- kivánságra irja ki a raktárban lévő anyagok készletét a nyomtató,
- anyag be- és kivételezés esetén a változásokat vezesse át az állományon és adminisztrálja a be- és kivétel tényét.

Mindegyik tevékenység egymástól függetlenül bármikor bekövetkezhet. Tehát a részfeladatoknak külön-külön is futtathatónak kell lenni a gépen. Ilyenkor két megoldás létezik. Az egyik: külön programot irunk mindegyikre és ezeket mindig külön betöltjük, amikor szükséges. A másik megoldás szerint egyetlen programot használunk, amely menüben /étlapon/ kinálja a felhasználónak mindhárom szolgáltatást.

Pl.: Készletjelentés /1/
Bevételezés /2/
Kiadás /3/
Befejezés /4/

A tevékenységek után irt számok begépelésével választja ki a felhasználó, hogy melyik szolgáltatást kéri. Mindegyik után természetesen vissza kell térni a menühöz.

Most nézzük milyen programozási feladatokat kell egy ilyen vagy hasonló adatállománynál ellátni.
Legelőször is létre kellett hozni az állományt. Ehhez irásra meg kell nyitni. A létrehozást nyilván ciklikus tevékenységként terveztük meg.
Elképzelhető, hogy a program mindig megkérdezi a felhasználót, hogy akar-e még adatokat felvenni, ha nem, lezárja az adatállományt. Az is megoldál lehet, hogy előre megkérdezzük a felhasználót, hogy hány adatot akar felvenni az adatállományba. Ezután a ciklusban számlálni kell a felvett rekordokat és ha a végére értünk, ugyancsak lezárni a file-t.

A menü program a képernyő törléséből, kiirásából, a felhasználó válaszának kezeléséből és ettől függően feldolgozó modulok valamelyikének hivásából áll.
A befejezést, mivel egyetlen CLOSE müveletből áll, nem célszerü külön modulként kezelni, beépithető a MENÜ programba.

Következik a feldolgozás megtervezése.
Furcsa módon a legbonyolultabbnak látszó feladat, a készletjelentés elkészitése a legegyszerübb. Mi a teendő? Legelőbb is a felhasználónak ki kell irni a szolgáltatás cimét /KESZLETJELENTES/, hogy meggyőződhessen róla, jó gombot nyomott-e meg. Ezután természetesen meg

kell nyitni olvasásra a már elkészült állományt.
Ez az OPEN i,8,j,"Ø ANYAG" utasitással lehetséges.
A megnyitott állományból egy xINPUTx i,A,B,C, ...utasitással lehet egy rekordot beolvasni. A rekord adatait ki kell iratni, esetleg a belőlük kiszámolt további mennyiségeket /érték, stb./. Ezután ujra beolvasás stb. Ezt is nyilván ciklikusan érdemes csinálni. Természetesen figyelni kell, mikor van vége az állománynak. Mellékes probléma, de nem lényegtelen, fejlécet is kell csinálni. Lehetőleg minden lapra.
Ha az állomány elejétől a végéig minden rekordot beolvastunk és még egyet akarunk olvasni, hibajelzés keletkezik, ami megnehéziti pl. a menü programba való visszatérést.
A problémán az segit, hogy a COMMODORE 64 gépen van egy ST nevü állapot /statusz/ byte.

Ha ennek tartalma 64 lesz, az utolsó rekordot beolvastuk. A beolvasás után tehát meg kell vizsgálni ST tartalmát. Ha 64, vissza a beolvasáshoz, ha 64, vége a beolvasásnak, jöhetnek a befejező müveletek.

A KÉSZLETJELENTÉST KÉSZITŐ PROGRAM LISTÁJA

```
 3Ø REM xKESZLETx
 6Ø REM xMEGNYITASx
 7Ø OPEN 2,8,2, "ANYAG,S,R"
 8Ø REM xADATBEOLVASASx
 9Ø PRINT "O" /CLS/
 95 INPUT x2,A,NØ,MØ,M,AR
1ØØ PRINT "AZONOSITO .... NEV"
12Ø PRINT "EGYSEG MENNYISÉG"  ÉRTÉK"
135 ER=MxAR
14Ø PRINT MS,M,ER
22Ø PRINT: PRINT: PRINT
```

```
23Ø IF ST=64 THEN 26Ø
24Ø GET AØ$: IF AØ$ = " " THEN 24Ø
25Ø GO TO 95
26Ø REM xLEZARASx
27Ø CLOSE2
28Ø END
```

Az adatállomány módositása a soros állomány esetén csak az állomány ujrairásával valósitható meg. A változatlan rekordokat egyszerüen átmásoljuk egy másik állományba, a változásnak megfelelően módositjuk a rekordokat, majd ezt is felirjuk az uj állományba.

A MÓDOSITÓ PROGRAM LISTÁJA

```
 1Ø REM xMODOSITASx
1ØØ REM xAZONOSITO BEOLVASASx
11Ø INPUT "AZ ANYAG AZONOSITOJA:";AA
12Ø REM xALLOMANYOK MEGNYITASAxx
13Ø OPEN 2,8,2, "ANYAG,S,R"
135 OPEN 3,8,3, "ATMENETI,S,W"
14Ø INPUT x 2, A,NØ$,MØ$,M,AR
15Ø UJ=ST
16Ø IF A=AA THEN 32Ø
165 PRINT x 3,A,;",";NØ$;"?";MØ$;",";M;",";AR
17Ø IF UJ  64 THEN 14Ø
18Ø PRINT "NINCS ILYEN AZONOSITO"
19Ø CLOSE 2, CLOSE 3
2ØØ GO TO 11Ø
32Ø REM xMENNY BEOLVx
325 PRINT "AZONOSITO ... NEV"
326 PRINT A,NØ$
327 PRINT: PRINT "EGYSEG MENNYISEG AR"
328 PRINT MØ$,M,AR : PRINT
33Ø INPUT "A KIVETT MENNYISEG:"; KM
```

```
43Ø MM=M-KM
45Ø REM *MODOSITAS*
46Ø PRINT *3,A;",";NØ;",";MS;2,";MM;",";AR
5ØØ INPUT *2,A,NØ,MØ,M,AR
51Ø UJ=ST
52Ø PRINT *3,A;",";NØ;",";MØ;",";M;",";AR
53Ø IF UJ   64  THEN 5ØØ
54Ø CLOSE 2, CLOSE 3
55Ø OPEN 15,8,15
56Ø PRINT * 15,"SØ:ANYAG"
57Ø PRINT *15,"RO:ANYAG=AMENETI"
58Ø CLOSE  15
```

TANULÁSI TANÁCSOK

Az adatállományok témaköréből a tv-adás csak izelitőt ad. A könyv már részletesebben tárgyalja a témát, de itt is csak az adatállománynak az u.n. szekvenciális tipusa szerepel. Már ezzel is sokféle feladat oldható meg, azonban pl. a gépkönyvekből vagy más leirásokból érdemes más tipusu adatállományok kezelésével is megismerkedni. /Pl. közvetlen elérésü, vagy relativ/. /dr. Ury László, Lángos István stb./

XVI. V I Z S G Á R A K É S Z Ü L Ő K N E K

Ez a müsor azoknak szólt, akik végignézték a tanfolyamot, végigrágták magukat a könyvön, esetleg már számitógép mellé is jutottak és elhatározták, hogy vizsgára jelentkeznek.
Azok a tanácsok azonban amelyek ebben az adásban elhangzottak, részben vagy egészben nemcsak a vizsgára, hanem általában a programozás tanulására is jórészt érvényesek.
Ezekből foglalunk össze az alábbiakban néhányat.
A külön erre a célra megjelent könyvön, a tv adásokon kivül, a gyakorlati munkához elengedhetetlen annak a gépnek, ill. a BASIC nyelvének a leirása, amelyen a kezdő vizsgázó dolgozni tud.
Ezen tul nem érdemes az irodalmat halmozni.
Saját elkövetett vagy elkerült hibáiból lehet a tanuló programozónak a legtöbbet tanulnia.

Az adásban felvetődő további kérdéskörök, tanácsok:

- Irjunk világosan, áttekinthetően. Azon felül, hogy magunknak előnyös, a program későbbi felhasználója is ember, könnyitsük meg a dolgát.
- Részesitsük előnyben az áttekinthető megoldásokat az elegánsakkal, trükkösökkel szemben.
- Hagyjuk a gépre a mechanikus számolást.
- Ismétlődő részekre használjunk szubrutint.
- Kerüljük a szükségtelen GO TO-kat. Programunkat lehessen felülről lefelé olvasni.
- A változónevek legyenek világosak, lehetőleg a dolog lényegére utalók.
- Az elágazásokat szorosan kövesse a velük kapcsolatos tevékenység.

- A legtöbb probléma tul nagy ahhoz, hogy egy lépésben megirhassuk. Kisebb részekre kell bontani. Modularizálni kell. Tanfolyamunk könyvében erre nagyon jó példákat láttunk.
- Ha ugy érezzük rosszul fogtunk hozzá egy programhoz, inkább kezdjük ujra, ne toldozgassuk.
- Gyakori hiba, hogy a változóknak nem adnak kezdőértéket.
- Ha rossz a program és hibát keresünk, ne álljunk meg az első megtalált hibánál. Javitáskor ne vigyünk ujabb hibát a programba.
- Legyünk tekintettel arra, hogy a FOR ciklusok általában egyszer lefutnak, akármilyenek is legyenek a ciklusváltozó és a határok értékei.
- Egésznek szánt változókat ne definiáljunk lebegőpontosra még tévedésből sem.
  Lebegőpontos számokkal nem lehet jól számlálni.
- Hibakeresési módszerek és lehetőségei a TRACE nyomonkövetése, modulonkénti tesztelés.
- Kerüljük a ciklusból ki-be ugrást.
- Gondoljunk a felhasználó által elkövethető hibák jelzésére és kezelésére.

1

spectrum

```
 5 REM EMBEREVO
 6 LET x1=1
10 LET y1=1
15 LET x2=30
20 LET y2=20
25 IF INKEY$="v" THEN  LET y1=y1-1.5
30 IF INKEY$="c" THEN  LET y1=y1+1.5
35 IF INKEY$="m" THEN  LET x1=x1-1.5
40 IF INKEY$="n" THEN  LET x1=x1+1.5
45 LET x2=x2+SGN (x1-x2)
50 LET y2=y2+SGN (y1-y2)
55 CLS
60 PRINT AT y1,x1;"*";AT y2,x2;"?"
65 IF INT (x1+.5)<>x2  OR INT (y1+.5)<>y2 THEN  GO TO 25
70 PRINT AT y1,x1;"RESZVETEM";AT y1-1,x1;"?"
75 PAUSE 100
80 RUN
```

2

spectrum

```
 5 REM EMBEREVO
 6 LET x1=1
10 LET y1=1
15 LET x2=30
20 LET y2=20
25 IF INKEY$="v" THEN  LET y1=y1-1.5
30 IF INKEY$="c" THEN  LET y1=y1+1.5
35 IF INKEY$="m" THEN  LET x1=x1-1.5
40 IF INKEY$="n" THEN  LET x1=x1+1.5
45 LET x2=x2+SGN (x1-x2)
50 LET y2=y2+SGN (y1-y2)
55 CLS
60 PRINT AT y1,x1;"*";AT y2,x2;"?"
65 IF INT (x1+.5)<>x2  OR INT (y1+.5)<>y2 THEN  GO TO 25
70 PRINT AT y1,x1;"RESZVETEM";AT y1-1,x1;"?"
```

3

spectrum

```
 1 REM
 2 REM
 3 REM
 4 REM
 5 REM
10 REM **LE-KW atszamitas**
20 CLS
30 PRINT "LE-KW"
40 INPUT "KEREM A LOERO ERTEKET";LO
50 LET KW=LO*0.736
60 PRINT LO;" LOERO",KW;" KILOWATT"
```

4

spectrum

```
10 REM **LE-KW**
20 CLS
30 PRINT "LE-KW"
35 PRINT "Leallitas 0"
40 INPUT "Kerem a loero erteket";LO
45 IF LO=0 THEN  STOP
50 LET KW=LO*0.736
60 PRINT LO;" LOERO",KW;" KILOWATT"
70 GO TO 40
```

5

spectrum

```
10 REM szorzotabla
20 LET a=INT (10*RND)+1
30 LET b=INT (10*RND)+1
40 PRINT "MENNYI  ";a;"*";b;"?": PRINT : INPUT c: PRINT c
60 IF c=a*b THEN  PRINT "HELYES": PRINT : PRINT : PRINT : GO TO 10
70 PRINT "EJNYE-EJNYE": PRINT : PRINT : GO TO 40
```

6

spectrum

```
 1 REM kor
10 INPUT "kozeppont?";x,y
20 INPUT "sugar?";r
30 FOR n=0 TO 2*PI STEP PI/90
40 PLOT x+r*COS n,y+r*SIN n
50 NEXT n
60 PAUSE 100
65 CLS
70 CIRCLE x,y,r
100 IF INKEY$="" THEN  GO TO 100
110 LOAD *"m";1;"ellipszis"
```

7

spectrum

```
 1 REM ellipszis
10 INPUT "kozeppont?";x,y
20 INPUT "nagytengely?";a
25 INPUT "kistengely?";b
30 FOR n=0 TO 2*PI STEP PI/90
40 PLOT x+a*COS n,y+b*SIN n
50 NEXT n
60 PAUSE 100
70 CIRCLE x,y,a
80 IF INKEY$="" THEN  GO TO 80
100 LOAD *"m";1;"lissajous"
```

8

spectrum

```
10 REM lissajous
20 CLS
30 INPUT "fazisszogelteres-rad?";y
40 INPUT "a frekvenciak aranya?";x
45 PRINT "fi=";y,"F1/F2=";x
50 LET z=x*100: LET q=x*40
60 FOR t=z TO 10 STEP -1
70 LET w=(3.14*t)/q
80 LET a=70*SIN (w)+120
90 LET b=40*SIN (w*x-3.14*y)+90
100 PLOT a,b
110 NEXT t
120 IF INKEY$="" THEN  GO TO 120
130 LOAD *"m";1;"ora"
```

9 spectrum

```
 10 REM ora
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
 40 NEXT n
 50 IF INKEY$="" THEN  GO TO 50
 60 FOR t=0 TO 200000
 70 LET a=t/30*PI
 80 LET sx=80*SIN a: LET sy=80*COS a
200 PLOT 128,88: DRAW  OVER 1;sx,sy
210 PAUSE 42
220 PLOT 128,88: DRAW  OVER 1;sx,sy
400 NEXT t
```

lo "irog"

```
  5 IF INKEY$<>"" THEN  GO TO 5
  6 IF INKEY$="" THEN  PRINT "*": GO TO 6
  7 PRINT INKEY$;
  8 GO TO 5
 10 CLS : LET a=1: LET b=1: LET s=1
 80 IF INKEY$="" THEN  PRINT AT a,b;"*": PAUSE 100: PRINT AT a,b;" ": GO TO 8
 90 LET a$=INKEY$
100 IF a$="s" THEN  GO TO 190
110 IF a$="t" THEN  GO TO 10
120 IF a$="z" THEN  LET a=a+1
124 IF a$="n" THEN  LET a=a-1
125 IF a>=20 THEN  LET a=1
126 IF a<=1 THEN  LET a=20
140 IF a$="h" THEN  LET b=b-1
150 IF a$="i" THEN  LET b=b+1
151 IF b>=30 THEN  LET b=1
152 IF b<=1 THEN  LET b=30
155 IF a$="v" AND s=1 THEN  LET s=2: GO TO 160
156 IF a$="v" AND s=2 THEN  LET s=1
160 IF s=1 THEN  PRINT AT a,b;a$
170 IF s=2 THEN  PRINT AT a,b;" "
175 LET b=b+1
180 GO TO 90
190 REM nd
```

11 spectrum

```
  1 REM fuggveny-rajzolo"
 10 PLOT 0,87: DRAW 255,0
 20 PLOT 127,0: DRAW 0,175
 30 INPUT "intervallum";s
 35 INPUT "fuggveny";e$
 37 LET t=0
 40 FOR f=0 TO 255
 50 LET x=(f-128)*s/128: LET y=VAL e$
 60 IF ABS (y)>87 THEN  LET t=0: GO TO 100
 70 IF NOT t THEN  PLOT f,y+88: LET t=1: GO TO 100
 80 DRAW 1,y-regiy
100 LET regiy=INT (y+.5)
110 NEXT f
```

```
  2 REM **** SZUNYOG ****
  5 LET N=0
 10 DIM A(20,2)
 20 FOR I=1 TO 20
 30 FOR J=1 TO 2
 40 LET A(I,J)=INT (RND*(J*10+12))
 50 NEXT J
 60 NEXT I
 70 FOR I=1 TO 20
 80 PRINT AT A(I,1),A(I,2);"*"
 90 NEXT I
100 PRINT AT 15,20;"AGY"
110 LET X=INT (RND*22)
120 LET Y=INT (RND*32)
130 IF X=15 AND (Y=20 OR Y=21 OR Y=22) THEN  GO TO 110
140 PRINT AT X,Y;"O"
145 LET N=0
146 LET S2=0
150 LET N=N+1
152 PRINT AT 0,0;"IDO=";N
160 IF INKEY$<>"5" THEN  GO TO 170
161 IF Y=0 THEN  GO TO 170
163 PRINT AT X,Y;" "
164 LET Y=Y-1
166 PRINT AT X,Y;"O"
170 IF INKEY$<>"6" THEN  GO TO 180
171 IF X=21 THEN  GO TO 180
173 PRINT AT X,Y;" "
174 LET X=X+1
176 PRINT AT X,Y;"O"
180 IF INKEY$<>"7" THEN  GO TO 190
181 IF X=0 THEN  GO TO 190
183 PRINT AT X,Y;" "
184 LET X=X-1
186 PRINT AT X,Y;"O"
190 IF INKEY$<>"8" THEN  GO TO 200
191 IF Y=31 THEN  GO TO 200
193 PRINT AT X,Y;" "
194 LET Y=Y+1
196 PRINT AT X,Y;"O"
200 IF X=15 AND Y=20 THEN  GO TO 220
210 IF X=15 AND (Y=21 OR Y=22) THEN  PRINT  FLASH 1;"OSSZETORTED MAGAD":
215 GO SUB 150
220 FOR a=0 TO 31
230 FOR b=0 TO 21
240 IF SCREEN$ (b,a)="*" THEN  PRINT AT 0,20; FLASH 1;"CSALTAL": STOP
250 NEXT b
260 NEXT a
270 PRINT  FLASH 1;"GRATULALOK"
```



```
READY.

10 REM LE-KW ATSZAMITAS
20 PRINT"J LE-KW"
30 INPUT"KEREM A LOERO ERTEKET";LO
40 LET KW=LO*0.736
50 PRINT LO;"LOERO";KW;"KILOWATT"

READY.
```

14

```
READY.

1 INPUT"HANY NAPOS A HONAP";H
5 INPUT"A HET HANYADIK NAPJA ELSEJE";I
10 FOR K=1 TO H
12 IF I=1 THEN N$="H"
13 IF I=2 THEN N$="K"
14 IF I=3 THEN N$="SZE"
15 IF I=4 THEN N$="CS"
16 IF I=5 THEN N$="P"
17 IF I=6 THEN N$="SZO"
18 IF I=7 THEN N$="V"
19 I=I+1:IF I>7 THEN I=1:T=TI
20 PRINTK,N$
25 IF T+50>TI THEN GOTO 25
30 NEXTK

READY.
```

15

FORDITÓ ÉS DARABOLÓ

```
10 INPUT"□";A$
20 FOR I=1 TO LEN(A$)
30 PRINT MID$(A$,I,1)
40 NEXT I
50 END
100 INPUT"□";A$
110 FOR I=LEN(A$) TO 1 STEP -1
120 PRINT MID$(A$,I,1);
130 NEXT I
140 END

READY.
```

16

```
READY.

10 REM******
20 REM*EZ A PROGRAM "E"BETUKET SZAMLAL*
30 REM******
40 INPUT"KEREM A SZOVEGET !";A$
50 LET GY=0
60 FOR I=1 TO LEN(A$)
70 B$=MID$(A$,I,1)
80 IF B$="E"THEN LET GY=GY+1
90 NEXT I:PRINT
100 PRINT GY

READY.
```

READY.

17/1

```
10 REM*****************
20 REM*                 *
30 REM*   KOZHOM        *
40 REM*                 *
50 REM*                 *
60 REM***** REGGELI HOM *****
70 PRINT"□"
80 INPUT"REGGELI HOMERSEKLET: ";RH
90 PRINT
200 REM***** DELI HOM *****
210 INPUT"DELI HOMERSEKLET: ";DH
220 PRINT
400 REM***** ESTI HOMERSEKLET *****
420 INPUT"ESTI HOMERSEKLET: ";EH
430 PRINT
550 REM***** ATLAG *****
560 AH=(RH+DH +EH)/3
600 REM***** KIIRAS *****
610 PRINT"□"
620 PRINT"A REGGELI HOMERSEKLET: ";RH
625 PRINT
630 FOR I=1 TO RH
640 PRINT"*";
650 NEXT I
660 PRINT: PRINT
670 PRINT"A DELI HOMERSEKLET: ";DH
675 PRINT
680 FOR I=1 TO DH
690 PRINT"*";
700 NEXT I
710 PRINT: PRINT
720 PRINT"AZ ESTI HOMERSEKLET: ";EH
730 FOR I=1 TO EH
740 PRINT"*";
750 NEXT I
760 PRINT: PRINT
800 PRINT"A KOZEP HOMERSEKLET: ";AH
810 PRINT
820 FOR I=1 TO AH
830 PRINT"*";
840 NEXT I
850 PRINT: PRINT
860 PRINT"       V E G E"
870 END

READY.
```

```
10 REM*****************
20 REM*                 *
30 REM*   KOZHOM        *
40 REM*                 *
50 REM*****************
60 REM***** REGGELI HOM *****
70 PRINT"□"
80 INPUT"REGGELI HOMERSEKLET: ";RH
90 PRINT
100 IF RH<1 OR RH>40 THEN GO TO 120
110 GO TO 160
120 REM* THEN AG *
130 PRINT"HIBAS ERTEK, UJRA KEREM!"
140 PRINT
150 GO TO 80
160 REM* IF VEG *
200 REM***** DELI HOM *****
210 INPUT"DELI HOMERSEKLET: ";DH
220 PRINT
230 IF DH<1 OR DH>40 THEN GO TO 250
240 GO TO 390
250 REM* THEN AG *
360 PRINT"HIBAS ERTEK, UJRA KEREM!"
370 PRINT
380 GO TO 210
390 REM* IF VEG *
400 REM***** ESTI HOMERSEKLET *****
420 INPUT"ESTI HOMERSEKLET: ";EH
430 PRINT
440 IF EH<1 OR EH>40 THEN 460
450 GO TO 500
460 REM* THEN AG *
470 PRINT"HIBAS ERTEK, UJRA KEREM!"
480 PRINT
490 GO TO 420
500 REM* IF VEG *
550 REM***** ATLAG *****
560 AH=(RH+DH +EH)/3
600 REM***** KIIRAS *****
610 PRINT"□"
620 PRINT"A REGGELI HOMERSEKLET: ";RH
625 PRINT
630 FOR I=1 TO RH
640 PRINT"*";
650 NEXT I
660 PRINT: PRINT
670 PRINT"A DELI HOMERSEKLET: ";DH
675 PRINT
680 FOR I=1 TO DH
690 PRINT"*";
700 NEXT I
710 PRINT: PRINT
720 PRINT"AZ ESTI HOMERSEKLET: ";EH
730 FOR I=1 TO EH
740 PRINT"*";
750 NEXT I
760 PRINT: PRINT
800 PRINT"A KOZEP HOMERSEKLET: ";AH
810 PRINT
820 FOR I=1 TO AH
830 PRINT"*";
840 NEXT I
850 PRINT: PRINT
860 PRINT"      V E G E"
870 END

READY.
```

```
READY.

5 POKE650,128
10 PRINT"■":POKE53281,13:POKE53280,13
20 B$="Q":PRINT B$;"■■";
30 GETA$
40 IFA$="■"THENPRINT" ■■■";:GOTO20
50 IFA$="■■"THENPRINT" ";:GOTO20
60 IFA$="■"THENPRINT" ■■";:GOTO20
70 IFA$="■■"THENPRINT" ■■■";:GOTO20
80 IFA$="■"THENPRINT"─";:GOTO20
90 IFA$="■"THENPRINT"─■■■";:GOTO20
100 IFA$="■"THENPRINT"■■■";:GOTO20
110 IFA$="■■"THENPRINT"■■■";:GOTO20
120 IFA$="■"THENPRINT"■■";:GOTO20
130 IFA$="■"THENPRINT"■■■■";:GOTO20
140 IFA$="■"THENPRINT"■■■■";:GOTO20
150 IFA$="■"THENPRINT"■■";:GOTO20
160 IF A$<>""THENB$=A$:PRINTB$;
170 PRINTB$;"■■";
200 GOTO30

READY.
```



```
READY.

1 REM ****EREDMENY-NYILVANTARTAS
5 DIM N$(100),E(100)
10 REM ****ELOKESZITES
15 PRINT"■ *** NEVBEIRAS ***"
20 FOR I=1 TO 100
30 E(I)=0
40 PRINT
50 PRINT"RAJTSZAM:";I;"  NEV:";
60 INPUTN$(I)
70 NEXT I
100 REM **** EREDMENYBEIRAS
110 PRINT"■ *** EREDMENYBEIRAS ***"
120 PRINT
130 INPUT"RAJTSZAM,EREDMENY:";I,E(I)
140 IF I>0 AND I<101 THEN 120
300 REM **** KIIRAS
310 PRINT"■ *** EREDMENYEK ***"
315 PRINT"HELY.          NEV
320 FOR I=1 TO 100
330 PRINT I;TAB(6);N$(I);TAB(30);E(I)
340 NEXT I

READY.
```

```
READY.

1 REM ****EREDMENY-NYILVANTARTAS
3 INPUT"A RESZTVEVOK SZAMA";V
5 DIM N$(V),E(V)
10 REM ****ELOKESZITES
15 PRINT"□ *** NEVBEIRAS ***"
20 FOR R=1 TO V
40 PRINT
50 PRINT"RAJTSZAM:";R;"  NEV:";
60 INPUTN$(R)
70 NEXT R
100 REM **** EREDMENYBEIRAS
110 PRINT"□ *** EREDMENYBEIRAS ***"
120 PRINT
130 INPUT"RAJTSZAM,EREDMENY:";R,E(R)
140 IF R>0 THEN 120
300 REM **** KIIRAS
310 PRINT"□ *** EREDMENYEK ***"
315 PRINT"HELY.          NEV
320 FOR I=1 TO V
330 PRINT I;TAB(6);N$(I);TAB(30);E(I)
340 NEXT I

READY.
```



```
READY.

1 REM ****EREDMENY-NYILVANTARTAS
3 INPUT"A RESZTVEVOK SZAMA";V
5 DIM N$(V),E(V)
10 REM ****ELOKESZITES
15 PRINT"□ *** NEVBEIRAS ***"
20 FOR R=1 TO V
40 PRINT
50 PRINT"RAJTSZAM:";R;"  NEV:";
60 INPUTN$(R)
70 NEXT R
100 REM **** EREDMENYBEIRAS
110 PRINT"□ *** EREDMENYBEIRAS ***"
120 PRINT
130 INPUT"RAJTSZAM,EREDMENY:";R,E(R)
140 IF R>0 THEN 120
200 REM RENDEZES
210 C=0
220 FOR I=2 TO V
230 IF E(I-1)=>E(I) THEN 280
240 M=E(I)       :M$=N$(I)
250 E(I)=E(I-1) :N$(I)=N$(I-1)
260 E(I-1)=M     :N$(I-1)=M$
270 C=1
280 NEXT I
290 IF C<>0 THEN 210
300 REM KIIRAS
310 PRINT "□ HELYEZES          NEV          PONT"
320 FOR R=1 TO V
330 PRINT
340 PRINT R;TAB(6);N$(R);TAB(31);E(R)
350 NEXT R

READY.
```

```
READY.

10 REM ******************************
20 REM *                            *
30 REM *     T O R P E D O          *
40 REM *                            *
50 REM ******************************
60 REM
70 REM ** TABLA KESZITES **
80 REM
90 DIM T$(10,10)
100 FOR I=1 TO 10
110 FOR J=1 TO 10
120 T$(I,J)="."
130 NEXT J
140 NEXT I
150 REM
160 REM ** VELETLENSZAMOK **
170 REM
180 X=INT(RND(1)*8+1)+1
190 Y=INT(RND(1)*8+1)+1
200 REM
210 REM ** ALAKZAT ELHELYEZES **
220 REM
230 T$(X,Y)="O"
240 T$(X-1,Y)="O"
250 T$(X+1,Y)="O"
260 T$(X,Y-1)="O"
270 T$(X,Y+1)="O"
280 REM
290 REM ** JATEK **
300 REM
310 S=0
320 PRINT "KEREM A SOR,OSZL SZAMOT";
330 INPUT X,Y
340 IF X<0 OR X>10 THEN PRINT "HIBAS SORSZAM":GOTO  320
350 IF Y<0 OR Y>10 THEN PRINT "HIBAS OSZLOPSZAM":GOTO  320
370 IF X=0 OR Y=0 THEN GOTO 430
380 IF T$(X,Y)="." THEN PRINT "NEM TALALT":T$(X,Y)="+":GOTO 320
390 IF T$(X,Y)="+" OR T$(X,Y)="*" THEN PRINT "KORABBI CELZAS":GOTO 320
400 PRINT "■TALALT!■"
410 T$(X,Y)="*"
420 S=S+1
430 IF S<5 THEN GOTO 320
440 IF S=5 THEN                                   PRINT "GRATULALOK, 5 TALALAT!"
450 REM
460 REM ** KIIRATAS **
470 REM
480 FOR I=1 TO 10
482 PRINT:PRINT
500 FOR J=1 TO 10
510 PRINT " ";T$(I,J);
520 NEXT J
530 NEXT I
540 PRINT:PRINT:PRINT
545 REM
550 END
READY.
```

```
5 REM KOR RAJZ
10 X1=50:Y1=50
20 X2=200:Y2=50
30 Y VAL 50
40 FOR X=X1 TO X2
50 SET (X,Y)
60 NEXT X
70 STOP
100 X1=50:Y1=50
110 X2=50:Y2=180
120 X=X1
130 FOR Y=Y1 TO Y2
140 SET (X,Y)
150 NEXT Y
160 STOP
200 X1=50:Y1=30
210 X2=150:Y1=130
220 X=X1:Y=Y1
230 X=X+1:Y=Y+1
240 SET (X,Y)
250 IF X<X2 THEN 230
260 STOP
300 INPUT "KEZDOPONT";X1,Y1
310 INPUT "VEGPONT";X2,Y2
320 IF O1=X2 AND Y1=Y2 THEN SET (X1,Y1):GO TO 590
330 IF X1=X2 THEN L=0: GO TO 500
340 IF Y1=Y2 THEN M=0: GO TO 400
350 M=(Y2-Y1)/(X2-X1)
360 IF ABS(M)>1 THEN L=1/M:GOTO 500
400 FOR X=X1 TO X2 STEP SGN(X2-X1)
410 Y=INT((X-X1)*M+Y1+0.5)
420 SET(X,Y)
430 NEXT X
440 GOTO 590
500 FOR Y=Y1 TO Y2 STEP SGN(Y2-Y1)
510 X=INT((Y-Y1)*L+X1+0.5)
520 SET(X,Y)
530 NEXT Y
590 STOP
600 INPUT"KEZDOPONT";X1,Y1
610 INPUT"VEGPONT";X2,Y2
620 M=(Y2-Y1)/(X2-X1)
630 FOR X=X1 TO X2
640 Y=INT((X-X1)*M+Y1+0.5)
650 SET(X,Y)
660 NEXT X
670 STOP
900 INPUT"KOZEPPONT";XK,YK
910 INPUT"SUGAR";R
920 INPUT"OLDAL";N
930 X1=XK+R:Y1=YK
940 FOR FI=0 TO 360 STEP360/N
950 F2=FI*PI/180
960 X2=INT (XK+R*COS(F2))
970 Y2=INT (YK+R*SIN(F2))
980 GO SUB 320
990 X1=X2:Y1=Y2
1000 NEXT FI
1010 STOP
1100 X=110:Y=110:N=40
1110 FOR R=10 TO 100 STEP 10
1120 GO SUB 930
1130 NEXT R
1140 STOP

READY.
```

```
1 REM ***  MORSE  ***
2 PRINT "■"
3 PRINT " SZOVEG ATALAKITASA  M O R S E  JELEKKE "
4 PRINT:PRINT
5 DIM A$(26)
10 FOR I=0 TO 25
20 READ A$(I)
30 NEXT I
40 INPUT "KEREM A SZOVEGET";T$
45 PRINT:PRINT
50 K=LEN(T$)
60 FOR J=1 TO K
70 M$=MID$(T$,J,1)
80 PRINT A$(ASC(M$)-ASC("A"));" ▮ ";
90 NEXT J
100 PRINT:PRINT:PRINT
1000 DATA .-,-...,-.-.,-..,.,.-.,--.
1010 DATA ....,..,.---,-.-,.-..,--
1020 DATA -.,---,.--.,--.-,.-.,...,-
1030 DATA ..-,...-,,--,-..-,-.--,--.
READY.
```

Készült az MTV házinyomdájában
1985-ben, 100 példányban
Táskaszám: 85161
Vezető: Modla László