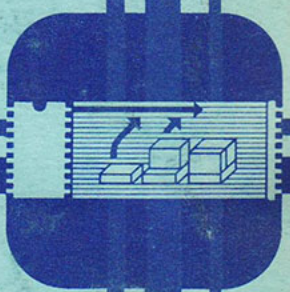
RÉVÉSZ ISTVÁN
ÉS MUNKAKÖZÖSSÉGE

PROGRAM-MODULOK



ORSZÁGOS PEDAGÓGIAI INTÉZET

VORKER®

Programmodulok

Szerkesztette:
RÉVÉSZ ISTVÁN

Szerkesztette:
Révész István

Főszerkesztő:
dr. Szűcs Barna

Sorozatszerkesztő:
Borbola István

A kiadvány szerzője és szerkesztője:
Révész István és munkaközössége

Bírálták:
dr. Puskás Albert
Varjú Károly

ISBN: 963 682 160 7 összkiadás
ISBN: 963 682 163 1

Kiadja az Országos Pedagógiai Intézet Számítástechnikai Programirodája
Felelős kiadó: Szabolcsi Miklós főigazgató

TARTALOMJEGYZÉK

A SOROZAT TAGJAI

Az Országos Pedagógiai Intézet Számítástechnikai Programirodájának gondozásában 1987/88-ban 12 számítástechnikai témakörhöz készült oktatási anyagok C 16-os és C Plus/4-es Commodore házi számítógépekre:

1. Algoritmusok, játékok:

A kiadvány tömören, de közérthetően, szemléletes példákon keresztül világít rá a probléma-algoritmus-program kapcsolatra, elvi és gyakorlati tanácsokat ad a jól felépített programok készítéséhez. A hozzá tartozó kazetta tizenkét, önmagában is alkalmazható, számítástechnikai és matematikai tanulságokkal teli programot tartalmaz. A programok jórészt listázhatók, így példát mutatnak egy színvonalas programozási stílus kialakítására, a gyakorlottabb felhasználóknak lehetőséget adnak arra, hogy más programok készítésénél is hasznosítsák az itt bemutatott programozási fogásokat. *Ajánlott:* ált. isk. 7., 8. osztályosainak és középiskolásoknak.

2. A számítógép és környezete:

Mindazoknak készült, akik szeretnék összekötni saját számítógépüket a külvilággal. 8 db különböző interfész és periféria tervezése kapcsán ismerteti meg az olvasót az interfész tervezés és illesztés fogásaival. Útmutató és 8 kapcsolási rajz, kész nyomtatott áramköri tervrajz. *Ajánlott:* ált. isk. és középiskolai tanároknak, számítástechnikai amatőröknek, 8. osztályos és középiskolai tanulóknak, tanárjelölt főiskolai és egyetemi hallgatóknak.

3. Programmodulok:

A felhasználó 55 db szubrutint, programmodult, ill. önálló felhasználói programot kap. Ezek egy része gépi kódú rutin, melyet az átlagos felhasználó nem tud megírni, de beépítheti munkájába. A felhasználói programok ötletet adhatnak hasonló programok megírásához. Az egyes rutinok, programok alkalmazási lehetőségeinek, működésének leírását útmutató tartalmazza. *Ajánlott:* ált. iskolai tanároknak, tanulóknak, kezdő amatőr programozóknak.

4. Adatkezelés számítógéppel C 16-ra és C Plus/4-re:

A témakör feldolgozását segítő útmutató az adatkezelés, adatfeldolgozás alapfogalmaival és folyamatával foglalkozik. Megismertet az adatfeldolgozó rendszerek fejlesztésével és a rendszerek alkalmazásbavételével. A feldolgozást 5 program és 2 minta-adatállomány segíti. *Ajánlott:* ált. és középiskolai tanároknak, fakultációs tananyagként, továbbképzési anyagként.

5. Szövegfeldolgozás számítógéppel:

A témakör feldolgozásához a felhasználó megkapja a Nemzetközi ABC szövegszerkesztő programot, amely magyar, cirill, német, lengyel, görög karakterkészletet és különleges matematikai jeleket (gyökjel, integráljel, alsó és felső index stb....) is tartalmaz. A szövegek és különleges karakterek nyomtathatók — akár fűzött kézírást imitálva is. A kézírásos nyomtatás főleg alsó tagozatban számíthat sikerre. A szövegszerkesztő minden olyan szolgáltatást tud, amely a tudományos munkához, vagy a tanári munkához kell. A feldolgozást útmutató és bemutató programok is segítik. *Ajánlott:* tanároknak, kutatóknak, íróknak, adminisztrátoroknak, ill. fakultációk anyagaként.

6. Számítógépes grafika C 16-on és C Plus/4-en:

Az útmutató és a mintaprogramok a felhasználói kézikönyvből meg nem tanulható különleges grafikus lehetőségekkel ismertetik meg az olvasót. A többszínű háttérszín üzemmód mellett a bittérképes színes grafikák elkészítésének fortélyait is megtanulhatja a felhasználó. Az útmutató és a mintaprogramok áttanulmányozása után az olvasó saját maga is képes lesz a játékprogramokból ismert színes és mozgó figurák „szellemecskék" programozására. *Ajánlott:* ált. isk. 7., 8. osztályosainak, középiskolásoknak és a grafika programozását most kezdő programozóknak.

7. Hangkeltés C 16-os (és C Plus/4-es) számítógéppel:

Az anyaghoz az útmutató mellett bemutató programok is tartoznak. Ezek áttanulmányozása alkalmassá teszi a felhasználót egy és többszólamú dallamok programozására, az interrupt alatti programozási lehetőségek kihasználására. A programok és dallamok az ált. iskolai ének-zene oktatáshoz igazodnak, így felhasználhatók számítógépes motivációra. *Ajánlott:* ált. isk. 7., 8. osztályosoknak, középiskolásoknak, valamint ált. iskolai énektanároknak, kezdő programozóknak.

8. Számítógépes fogások, trükkök C 16-ra és C Plus/4-re:

Az útmutató és a hozzá tartozó 16 program azok számára mutatja be a C 16-on és a C Plus/4-en alkalmazható fogásokat, akik a BASIC programozás alapjain túljutottak. A közölt rendszerváltozók, memóriacímek és gépi kódú programok megismertetik az olvasót az alkalmazások különleges lehetőségeivel. *Ajánlott:* kezdő programozóknak, általános és középiskolásoknak, tanároknak.

9. Számítógép és video:

Az anyaghoz útmutató, videofilm és bemutató programok tartoznak. A videofilm a hozzá tartozó számítógépes programokkal egy megvalósítható rendszert és egy példát mutat be a képmagnó és a házi számítógép összekapcsolására, az oktatásban történő alkalmazására. Az anyag az elveken túl egy interfész elkészítésével is megismerteti az olvasót. Elsősorban tanároknak, oktatástechnikai szakembereknek ajánlott

10. Számítógéppel vezérelt mérések:

Az útmutató megismerteti az olvasót a legfontosabb méréstani fogalmakkal, a mérések tervezésének, elvégzésének, értékelésének elméleti és műszaki-, méréstani alapjaival. A mérőberendezések és mérési összeállítások jelátvivő funkcionális egységeként a Tudományszervezési és Informatikai Intézetnél kapható Techno-MIR moduláris interfészrendszer berendezéseit mutatja be és használja fel az anyag. A témakörhöz tartozó 21 db professzionális program egyrészt az oktatómunkában és a bemutató kísérletekben alkalmazható számítógép-vezérlésű intelligens műszert szimulál (pl. tároló oszcilloszkópot, 8 csatornás logikai analizátort, digitális multimétert stb....) másrészt az iskolai kísérletekhez szükséges mérési összeállításokat vezérli. *Ajánlott:* általános és középiskolai tanároknak, speciális szakképző intézeteknek, pedagógusképző intézeteknek bemutatási és oktatási célra, pedagógiai intézeteknek és továbbképzéshez.

11. Számítógépes irányítástechnika:

Az anyaghoz útmutató, az útmutatóban ismertetett berendezésekhez 11 db mintaprogram tartozik. Ezek áttanulmányozása megismerteti az olvasót a számítógépes vezérlések és szabályozások alapfogalmaival, a mechatronikus modelleket működtető berendezések (kapcsolómodulok, interfészek) elkészítésével. A mintaprogramok elemzése példát mutat az irányítási algoritmusok programozására. Az irányító rend-

szerek jelátvivő funkcionális egységeként a Tudományszervezési és Informatikai Intézetnél kapható TechnoMIR interfészrendszer elemeit, érzékelő-, erősítő berendezéseknek és beavatkozó szerveknek, berendezéseknek pedig saját készítésű, a felhasználó által is reprodukálható eszközöket használ az anyag. Ezek leírását és elkészítési módját is tartalmazza. *Ajánlott:* az általános és középiskolai technikai tárgy oktatásához, fakultációkhoz, szakköri feldolgozáshoz.

12. Számítógépek összekapcsolása, helyi oktatóhálózatok:

Az anyag egy hazánkban eddig nyomtatásban még meg nem jelent területtel, a Commodore házi számítógépek (és a VIDEOTON TV COMPUTER) összekapcsolásának gyakorlati megvalósításával és e számítógépekből összeállított helyi oktatóhálózatok programozásával foglalkozik. Megismerteti a felhasználóval a számítógépek közötti kapcsolat hardver és softver eszközeit, a kapcsolatteremtés és az adatátvitel alapfogalmait. Megépített és reprodukálható kapcsolásokkal és kész felhasználói programokkal segíti, hogy az anyagot feldolgozó olvasó saját maga is összekapcsolhasson számítógépeket. A megszerzett ismeretek segítségével a minimális elektronikai ismerettel és készüléképítési gyakorlattal rendelkező felhasználó is meg tud valósítani számítógépek közötti adatátvitelt. Az útmutató második része az oktatásban bevált és sikerrel alkalmazott, a szegedi VORKER Kisszövetkezet által kifejlesztett TC-NET oktatóhálózatokkal, azok programozási fogásaival foglalkozik. Az anyag 6 db kapcsolást 1 db nyomtatott áramköri rajzot és 12 db professzionális felhasználói programot tartalmaz. *Ajánlott:* általános és középiskolai tanároknak 3., 4. éves középiskolai tanulóknak, számítástechnikai amatőröknek, középfokú szakképző intézeteknek, tanárképző intézeteknek, pedagógiai intézeteknek, továbbképző intézeteknek stb....

Valamennyi itt felsorolt anyag (útmutató és program) valamint a TC-NET megrendelhető a szegedi VORKER Kisszövetkezettől.

Cím:

VORKER Kisszövetkezet
6724 SZEGED, Zoltán u. 12.
Tel.: H 06-62-26-144
Telex: 82 688

A VORKER Kisszövetkezet vállalja — megrendelés alapján — a felsorolt témakörökben bemutatók, alap- és továbbképzések tartását, szervezését is.

ELŐSZÓ

Az olvasó a szegedi Tarjánvárosi IV. Számú Általános Iskolában fejlesztett 12 pedagógiai program (tanári kézikönyv, szoftver és/vagy beültetési rajzok) egyikét tartja a kezében. A munkában több mint hatvanan vettek részt: általános és középiskolai tanárok, főiskolai, egyetemi oktatók, tudományos intézetek munkatársai).

Az Országos Pedagógiai Intézet Számítástechnikai Programirodája és a szegedi iskola közös vállalkozása összefügg a számítástechnikai alapműveltség körülhatárolására irányuló erőfeszítésekkel. Kétségtelen, hogy a számítástechnikai-informatikai alapismeretek és az alapműveltség kapcsolatának feltárása elsősorban teoretikus jellegű munkát igényel. Az 1970-es években induló tartalmi korszerűsítés tapasztalatai azonban arra hívják fel a figyelmünket, hogy az iskolák nélkül, az iskolai hagyományok és a pedagógiai öntörvényeinek mellőzésével nem hozhatók létre olyan dokumentációk, amelyeket a gyakorlat magáénak érez, s amelyekkel a pedagógusok többsége közösséget vállal.

Az alapműveltség témakörében már eddig is több tanulmány látott napvilágot. Viszonylag kevesebb szó esett azonban arról, hogy miként lehet pedagógiailag szervezett ismeretrendszerré formálni az itthon is és külföldön is már megfogalmazott elképzeléseket (amelyek között meglepően szoros összefüggés lelhető fel).

Az elképzelésekről szóló következő rövid ismertetés nem tekinthető állásfoglalásnak abban a kérdésben, hogy mi a jobb: ha a számítástechnikai-informatikai alapismeretekhez a diákok az *óratervben* különféle tantárgyakban jutnak hozzá; *szakköri vagy fakultációs* programok keretében, esetleg *külön tantárgyként* biztosítják ezt számukra.

A fejlesztők munkáját a következő feltételezések irányították:

1. A számítástechnikai-informatikai ismeretek és az ez iránti érdeklődés nehezen tagolható az eddig megszokott módon, vagyis életkor és iskolatípus szerint. Az elkészült anyagoknak tehát egyaránt szolgálniok kell az általános iskolákat és a középfokú iskolákat.
2. Az iskolák fogadókészsége eltérő (pl. más-más a felszerelés, a tanárok felkészültsége, a szoftverellátottság stb.). Ez aligha teszi lehetővé az eddig megszokott, a mindenki számára kötelező tantervi előírásokat, sémákat. Az igényekhez tehát csak a többféle lehetőség egyidejű bemutatásával kerülhetünk közelebb.
3. A 12 pedagógiai program és ezek járulékai alapul szolgálhatnak egy majdan országos érvényű informatika tantárgy kidolgozásához. A témák ilyen rendszere a már

ma megvalósítható iskolai tevékenységeket tükrözi. Az informatika több más témájáról az iskolák jórésze számára ma még csupán leírás adható (pl. országos, nemzetközi hálózatok). A későbbiekben elképzelhető, hogy az iskolákban is lehetőség lesz az informatika teljes skálájának bemutatására.

4. A 12 rész között biztosíthatók „átjárások": a részek egymáshoz illeszthetők és kombinálhatók egymással.
5. A pedagógiai programokban megfogalmazott elképzelések érvényesítése érdekében az elinduláshoz szükséges a megfelelő segédletek elkészítése.
6. A pedagógiai programokhoz készült járulékok (beültetési rajzok, szoftverkészlet) egyik részének tantárgyi alkalmazásokra kell épülnie (pl. matematika, mérések), másik részük viszont az informatikának a mindennapi életben megjelenő gyakorlatát tükrözze (pl. adatkezelés szövegszerkesztés).

Reméljük, hogy munkánk segíteni fogja a pedagógiai gyakorlatot és ezáltal hozzájárul e téma elméleti kérdéseinek tisztázásához.
Örömmel és tisztelettel fogadjuk munkánk olvasójának, használójának észrevételeit.

Dr. Szűcs Barna

BEVEZETŐ

A számítógépes programozást a legtöbb ember úgy kezdi, hogy meglevő programban változtatást hajt végre, módosít. Ezután megpróbál teljesen önállóan egyszerű, rövid programokat összeállítani. Néhány próbálkozás eldönti az adott egyén további programozói sorsát: ha első munkái jórészt sikerültek, ez további programkészítésre sarkallja; ha nincsenek sikerélményei, a kedve is elmegy a programozástól.

Az útmutatóban bemutatjuk a moduláris programozás előnyeit, a feladatmegoldás folyamatát és — konkrét példán keresztül — egy program elkészítését modulokból. Útmutatónkat igyekeztünk úgy összeállítani, hogy a kezdők — és elsősorban a kezdők — számára nyújtson alapinformációkat, hasznos tanácsokat, de a programozásban jártasabbak is haszonnal forgathassák.

Révész István
e kiadvány szerkesztője

1. A MODULÁRIS PROGRAMOZÁS

Programozási feladataink megoldásában igen hasznosan alkalmazható a **moduláris programozás** és az így előállított **programmodulok**. Ennek az a lényege, hogy a főfeladatot részfeladatokra bontjuk, ezeket önállóan kódoljuk, és az így létrejött modulokból összeállítjuk a kész programot.
A **modul** olyan önálló programrész, amely az egész feladaton belül elkülönített részfeladatként jelenik meg, funkciója elkülönítve is értelmes és a főfeladattól elkülönítve is elkészíthető.
Fenti meghatározásból következik, hogy a moduláris programozás legfontosabb lépése az adott főfeladat *lépésekre* (funkciókra) *bontása*. Ezek a részfeladatok alkotják a modulokat.

1.1. A modulok csoportosítása

A feladat elvégzése szempontjából a moduloknak három csoportja van:
— **vezérlőmodul,**
— **adatmodul,**
— **eljárásmodul.**

1.1.1. Vezérlőmodul

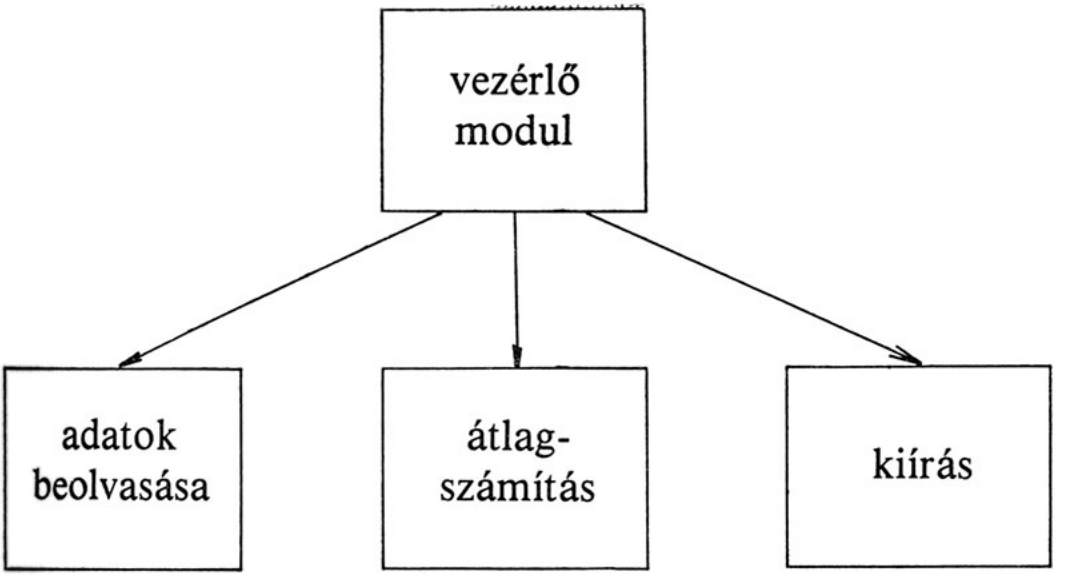
A vezérlőmodul irányítja az egész program vagy a modulok végrehajtását.
A vezérlőmodulban meg kell határozni a végrehajtás sorrendjét, így a vezérlőmodul ennek megfelelően hajtja végre a modulokban meghatározott feladatokat.

Hívás vagy **vezérlésátadás:** a vezérlőmodulból az *alárendelt modul*ba való áttérés.

Figyeljük meg a vezérlőmodul működését az alábbi vázlaton.

Vizsgáljuk ugyanezt egy rövid programon!

Példánkban a vezérlőmodul három alárendelt modult vezérel. Az első modulnak (100—130) öt számértéket kell beolvasnia. A második modul (150—180) kiszámítja a beolvasott öt szám átlagát. A harmadik modul (200—220) végzi a kiíratást.
A vezérlőmodul a fenti sorrendben, egymás után hajtja végre a modulokban meghatározott feladatokat.

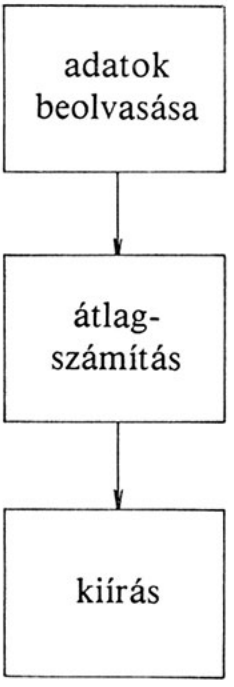


1. ábra

```
1 rem *** 1. lista ***
2 :
3 :
4 :
10 print "S"
20 :
30 rem ** vezerlomodul **
40 print "adatbeolvasas":gosub 100:print"qq"
50 print "atlagszamitas":gosub 150:print"qq"
60 print "kiiratas":gosub 200
70 end
80 :
90 :
100 rem ** adatok beolvasasa **
110 print "kerek ot szamot!"
120 input a1,a2,a3,a4,a5
130 return
140 :
150 rem ** atlagszamitas **
160 a=a1+a2+a3+a4+a5
170 b=a/5
180 return
190 :
200 rem ** kiiratas **
210 print "az ot szam atlaga: ";b
220 return
```

1. lista

Fenti programot természetesen meg lehet oldani egyszerűbben is. Egyszerűbb szerkezetű programoknál nincs szükség vezérlőmodulra, hiszen — ha a modulokat megfelelő sorrendben helyeztük el — a modulok egymást hívják.
Nézzünk erre is egy példát! A 2. ábrán az átlagszámító program blokksémája, a 2. listán a kódolt program látható.



2. ábra

```
1 rem *** 2. lista ***
2 :
3 :
4 :
10 print"S"
20 print"kerek ot szamot! "
30 input a1, a2, a3, a4, a5
40 a= a1+a2+a3+a4+a5
50 b=a/5
60 print"az ot szam atlaga: ";b
```

2. lista

1.1.2. Adatmodul

Az adatmodul a program adatait határozza meg. Adatmodul az 1. listán szereplő 100—130 programsorokból álló rész.
A program számára modulonként is adhatunk meg adatokat, de célszerűbb, ha a program az adatmodult egy helyen — a program elején — tartalmazza.

1.1.3. Eljárásmodul

Az eljárásmodulok a program legfontosabb részei, ezekkel lehet a különböző feldolgozási műveleteket, kiírásokat stb. végrehajtani. Egy programon belül eljárásmodulból van legtöbb. Ezen modulok összessége hajtja végre a tulajdonképpeni feldolgozást.

Korábbi példáinkban is megfigyelhetünk eljárásmodulokat. Az 1. lista átlagszámító modulja (150—180), valamint a kiíró modul (200—220) eljárásmodul. Ugyanígy eljárásmodulnak kell tekintenünk a 2. lista 40-es, 50-es és 60-as sorát is.

FELADAT: *Készítsen programot, mely kiszámítja egy tetszőleges adott mennyiség tetszőleges százalékát!*

1.2. Modulokra bontás

Moduláris programozás során egy feladat megoldásakor a modulokra bontás a legfontosabb lépés. E módszert egy konkrét feladat megoldásával mutatjuk be. Legyen a feladat a napközis térítési díjak havi megállapítása és ezek összesítőjének számítógépes elkészítése.

A feladat — első megközelítésben — két részfeladatra bontható:

— e havi tényleges befizetés/fő
— e havi összesítés.

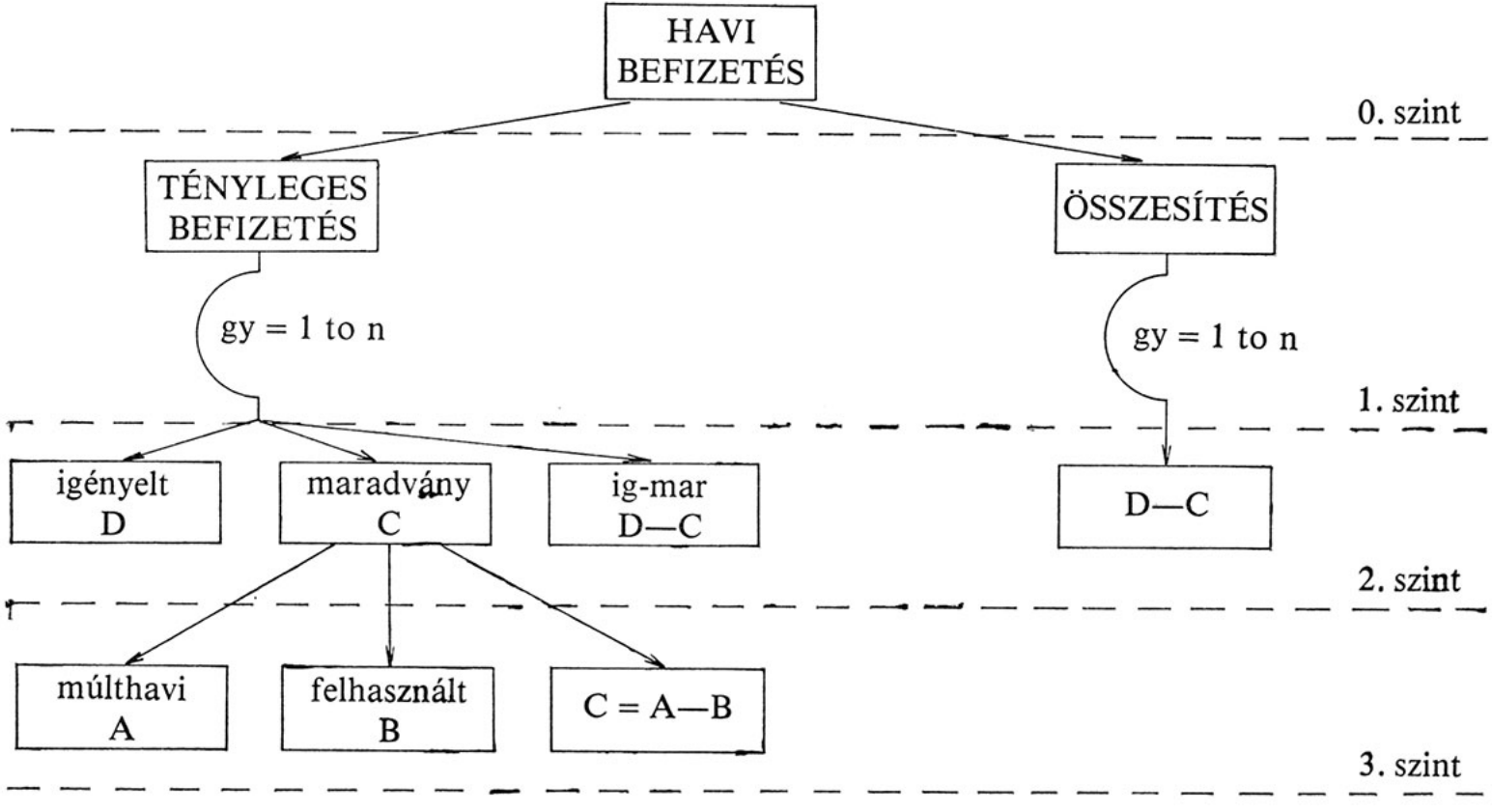
E részfeladatok további funkciókra (feladatokra) bonthatók. Így a következő feladatok jöhetnek létre:

— előző havi tényleges befizetés (A),
— előző havi felhasznált befizetés (B),
— előző havi maradvány (C),
— e havi befizetés (D),
— e havi tényleges befizetés (D−C),
— összesítés (D−C)·n.

A *funkciókra bontást* addig érdemes csinálni, amíg jól elkülöníthető, elemi feladatokat kapunk. Ezeket tekinthetjük *moduloknak*. Ezt követően meg kell határoznunk azt, hogy a modul milyen műveletet végezzen.

Fentiekből megfigyelhető, hogy a módszer lényege a *felülről-lefelé haladás*. Ennek alapján *meghatározzuk a főfeladatot*. Példánkban ez az e havi befizetés. Ezután megvizsgáljuk, milyen részfeladatokat kell végrehajtani, hogy a főfeladatot meg tudjuk oldani. Így megkapjuk az *alárendelt feladatokat*. Ezek további *részfeladatokra* bonthatók és így tovább.

A felülről-lefelé haladás során keletkező részfeladatok *szinteket* alkotnak. Attól függően, milyen összetett a feladatunk, kevesebb vagy több *szint* jön létre egészen addig, amíg egymástól jól elkülöníthető feladatokat nem kapunk. Ezek lesznek programunk moduljai. Ha ezt a lebontást ábrázolással — ún. szerkezeti ábra — kísérjük, akkor jól láthatóvá tehetjük a modulok közötti összefüggéseket is.


HAVI
BEFIZETÉS
0. szint
TÉNYLEGES
BEFIZETÉS
ÖSSZESÍTÉS
gy = 1 to n
gy = 1 to n
1. szint
igényelt
D
maradvány
C
ig-mar
D—C
D—C
2. szint
múlthavi
A
felhasznált
B
C = A—B
3. szint

3. ábra

Az így létrejött szerkezeti ábra alapján világosan látszik a részfeladatok közötti összefüggés. A legalsó — a 3. — szinten levő tényleges és felhasznált befizetésből számítható ki a maradvány (A – B). Ebből és az igényelt befizetésből kapjuk az 1 főre jutó tényleges befizetést (D – C). Ez az 1. szinten található. Ebből és az összesítésből adódik az egész napközis csoport befizetése a tárgyhóban. Ez a 0. szinten van.
Az egyes *szinteken* téglalappal jelölt feladatokat tekintjük **moduloknak**. Ezek mindegyike **eljárásmodul**, de közülük többhöz *adatmodul* is csatlakozik. Az egész programot, a különböző *szinteket* valamilyen módon vezéreljük.
Programunkban — a fentiek alapján — lesz befizetésmodul, maradványmodul, összesítőmodul stb.
FELADAT: *Állítson össze egy napközis díjkiszámító programot!*

1.3. A feladatmegoldás folyamata

Egy feladat számítógépes megoldása során a következő lépéseket szokás megkülönböztetni:

— feladatelemzés,
— programtervezés,
— modultervezés,
— **programkódolás,**
— tesztelés,
— futtatás,
— értékelés.

Az alábbiakban ezekről szólunk.

1.3.1. Feladatelemzés

Bármilyen *program megírását* először *a tervezéssel kezdjük!* A tervezésnél fontos, hogy a feladatot részeire bontsuk, elemezzük. Így kialakulnak a részfeladatok és ezek összefüggései. Ezek meghatározzák a feladatmegoldásig végzendő tevékenységet. A részfunkciókra bontott feladatot megvizsgáljuk, számítógépen megoldható-e.

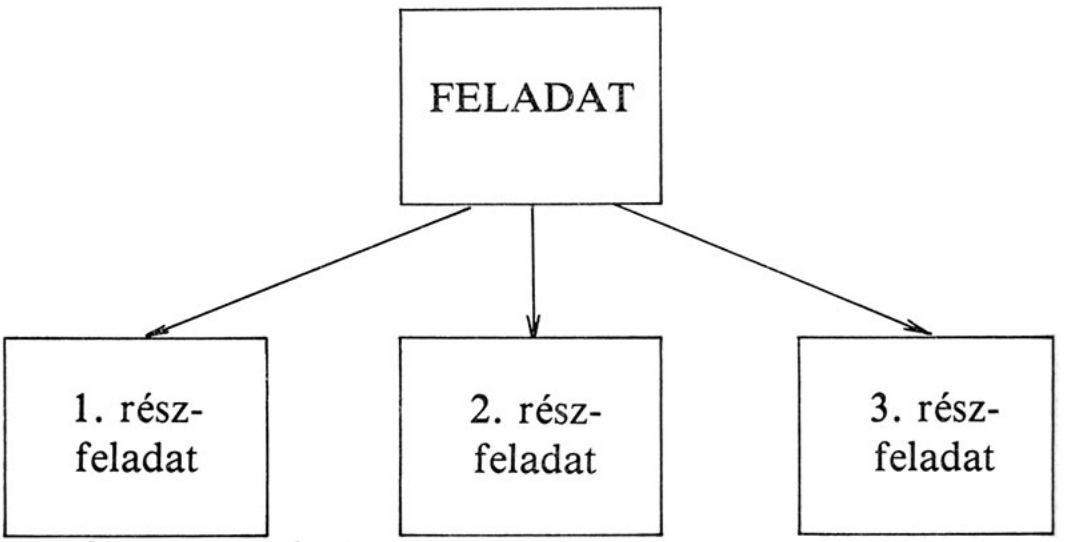
1.3.2. Programtervezés

Az elemzés eredményei alapján tovább finomíthatjuk a program szerkezetét, bonthatjuk fel részfeladatokra, modulokra. A **részfeladatokra bontás**nak több lehetősége van. A következőkben ezeket ismertetjük.

1.3.2.1. Részfeladatokra bontás lehetőségei

A) Egymásutáni lebontás

Egy feladat úgy oldható meg, hogy pl: 3 részfeladatot kell egymás után elvégezni. A 4. szerkezetábrán látható a feladat és a 3 részfeladat. A részfeladatok sorszáma a végrehajtás sorrendjét is jelöli. A folyamatábrán megfigyelhető, hogy ezeket a feladatokat a gép adott sorrendben hajtja végre (5. ábra).



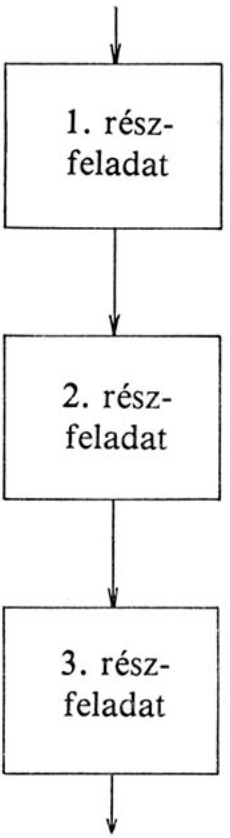
4. ábra

B) Ciklikusan ismétlődő lebontás

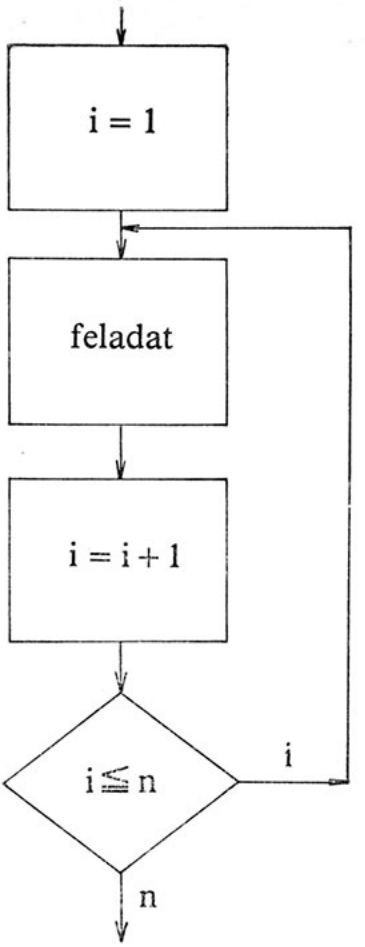
Ilyenkor meghatározott számszor (n-szer) hajtjuk végre ugyanazt a részfeladatot, csak a benne szereplő valamely változó értékét változtatjuk meg minden ciklusban (pl.: növeljük 1-gyel) (6. ábra).
A folyamatábrán jól láthatók az összefüggések. Az első körben $i=1$ és így hajtódik végre a feladat. Ezután i értékét 1-gyel növeljük és megvizsgáljuk, hogy elérte-e n értékét. Ha nem, a feladatot ismét végrehajtjuk a növelt i változóval. Ezt követi i újbóli növelése, vizsgálat és újabb feladatvégrehajtás. Ha i értéke nagyobb, mint n, akkor a program továbbhalad.
Ennél a lebontásnál vezérlésre is szükség van.

C) Feltételtől függő lebontás

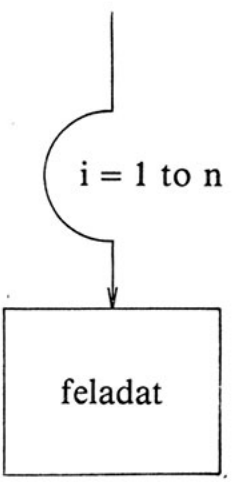
Ez esetben a részfeladatot csak akkor hajtja végre a gép, ha valamely feltétel teljesül (8. ábra).
A 9. ábrán ennek a lehetőségnek a folyamatábrája látható. A részfeladat csak a feltétel teljesülése esetén valósul meg.



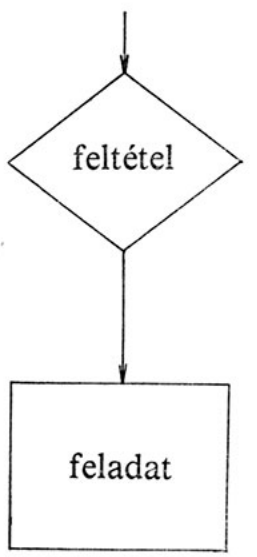
5. ábra



6. ábra



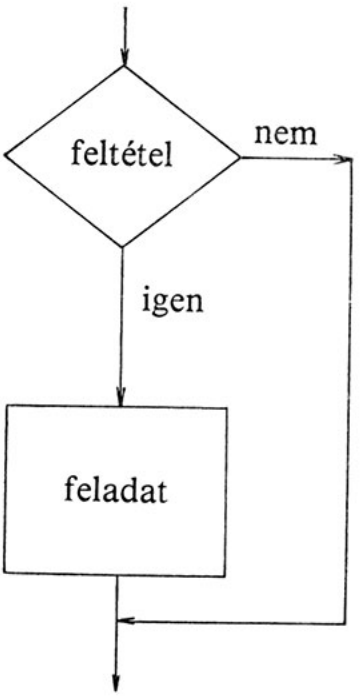
7. ábra



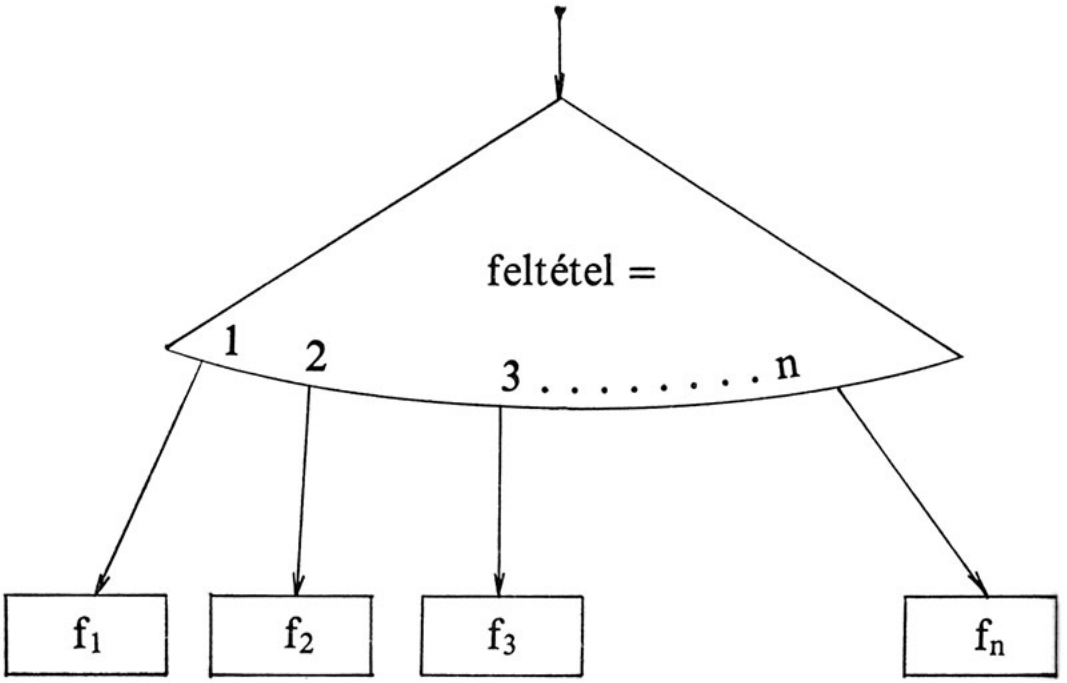
8. ábra

D) Elágazással megoldható lebontás

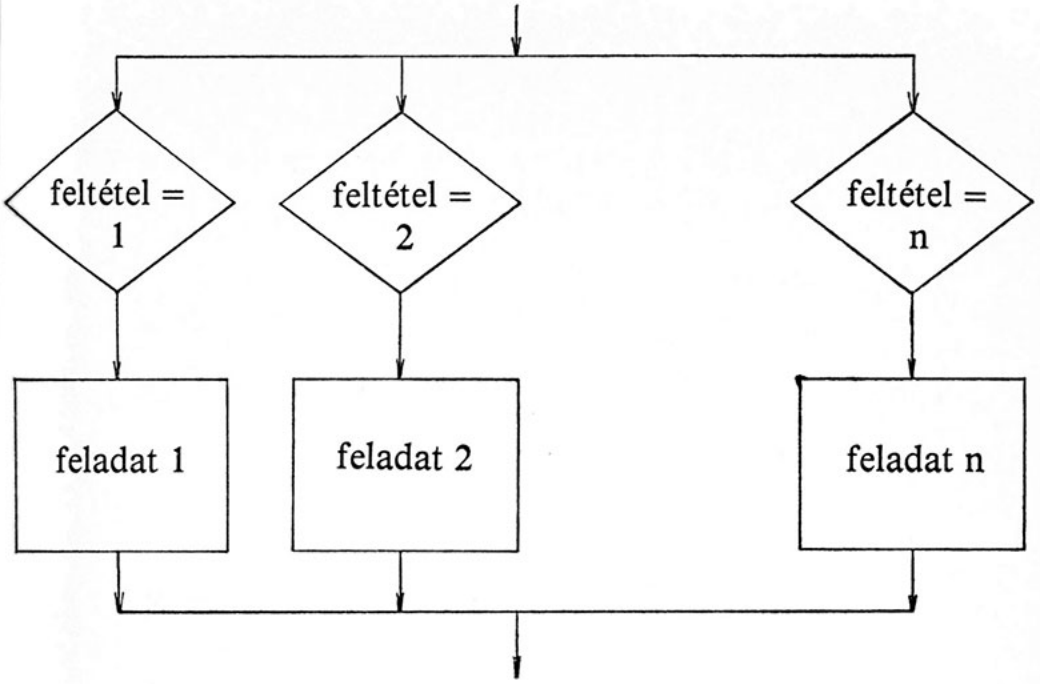
Ilyenkor, egy adott feladatnál 1, 2, 3, ..., *n* feltétel teljesülhet. Attól függően, hogy az 1, a 2, ..., az *n* feltétel teljesül, a program az 1, a 2, ..., az *n* részfeladatot oldja meg (10. ábra).



9. ábra



10. ábra

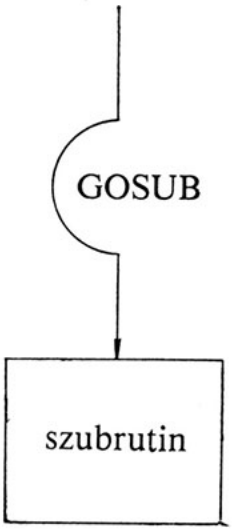


11. ábra

A folyamatábrán (11. ábra) látható, hogy a vezérlőmodul sorra vizsgálja a feltételeket és a feltételek teljesülése esetén a hozzátartozó részfeladatot oldja meg.

E) Szubrutin hívással megoldható lebontás

Ennél a lehetőségnél a feladat megoldásához meghívunk egy szubrutint, mely valamilyen részfeladatot végez (12. ábra).



12. ábra

A részfeladatokra bontás lehetőségeinek valamelyikét felhasználva programunk további funkciókra, modulokra bontható. Ezt a lebontogatást ismételve kaphatunk egészen egyszerű, elemi feladatok elvégzésére alkalmas modulokat.
Hasonló, vagy azonos feladatok megoldására *típusprogramokat (típusmodulokat) is készíthetünk*. A **típusprogram** tulajdonképpen egy kaptafa, melyre az adott, konkrét feladat ráhúzható.
Sok *típusprogramból* **modulkönyvtárat** állíthatunk össze.

1.3.3. Modultervezés

Megkülönböztetünk külső és belső tervezést. A *külső tervezés* során a modul külsőségeit: nevét, típusát, környezethez-kapcsolódását tervezzük. Nagyon célszerű, hogy minden modulnak egy bejárata és egy kijárata legyen. Ez azt jelenti, hogy a modul — végrehajtása során — mindig ugyanazon az utasításssoron induljon és ugyanazon fejeződjön be.
A *belső tervezés* során először a kiinduló adatokat kell leírni, vagyis azokat, melyeket a gép tárol, vagy korábban előállított. A tervezés következő lépése az eredményadatok leírása részletesen (pl.: kiírás formája stb.). Ezután kell meghatározni a feldolgozás folyamatát, az adatokon végrehajtandó műveleteket, a sorrendet stb.
Modultervezésnél célszerű a folyamatábra alkalmazása, mivel így a kódolás egyszerűbbé válik.

1.3.4. Programkódolás

Kódolás: a folyamatábrán rögzített műveletek megfogalmazása valamely programozási nyelven.
A kódolást külön-külön, a folyamatábra alapján kell elvégezni. Ennek eredményeképpen jönnek létre az utasítások.
Kódolásnál figyelembe kell venni a programozási nyelv tulajdonságait, sajátosságait.

1.3.4.1. A kódolás módszere

A kódolást célszerű olyan módszerrel végezni, hogy a modulokat egyenként megtervezzük, majd — a feladatnak megfelelően — ezeket a modulokat összefűzzük, vagy valamilyen módon vezéreljük. Itt a feladat jellege szabhatja meg, hogy a modulok milyen módon kapcsolódhatnak egymáshoz. Vegyük figyelembe az 1.3.2.1. pontban megismert dolgokat! Ügyeljünk a modulok közötti helyes kapcsolat létrehozására, mert ha itt hibát vétünk, programunk nem működik, vagy rosszul működik.
A programkód (lista) legyen világos, áttekinthető, következetes és tiszta. Az ilyen programban könnyebben tudunk tájékozódni, hibát keresni, módosítani és javítani.

1.3.4.2. Kódolási konvenciók

Ismerkedjünk meg néhány hasznos kódolási konvencióval! Ezek betartása nem kötelező, de a programozó saját érdeke, hogy e konvenciókat tekintse szabálynak.

1.3.4.2.1. Utasításszámok kódolása

A program azonosítására szolgáló megjegyzéseket 1-től induló és egyesével növekvő utasításszámokkal lássuk el! (3. lista)

```
1 rem  *** 3. lista ***
2 rem
3 rem          **********************
4 rem          *                    *
5 rem          *   alapmuveletek    *
6 rem          *                    *
7 rem          * gyakorlo program  *
8 rem          *                    *
9 rem          **********************
10 rem
11 rem keszitette: vitalis jenone
12 rem             revesz  istvan
13 rem             **** 1987 ****


ready.
```

3. lista

A deklarációkat 110-től 990-ig, tízesével növekedve sorszámozzuk! (**Deklaráláskor adjuk** meg a változók értékeit, dimenzionálunk stb.)

```
100 rem *** 4. lista ***
110 :
120 :
130 :
140 :
150 input "Shany feladat legyen";n: rem n: a feladatok szama
160 h=0: rem h: a helyes valaszok szama
170 for i=1 to n
180 x=int(rnd(1)*100)+1
190 y=int(rnd(1)*100)+1
200 z=int(rnd(1)*10)+1
```

4. lista

A program utasításait 1010-től 9990-ig sorszámozzuk úgy, hogy minden főbb egység önálló ezres számcsoporttal rendelkezzen. A sorszámozás akár egyesével is történhet.

```
200 rem *** 5. lista ***
210 :
220 :
230 :
240 rem *** vezerlomodul ***
250 :
260 b=int(rnd(1)*4)+1
270 if b=1 then 60450
280 if b=2 then 60550
290 if b=3 then 60650
300 if b=4 then 60750
```

5. lista

A GOSUB utasítással hívott modulokat, belső eljárásokat 60 000-től sorszámozzuk ötösével vagy tízesével. Itt is minden hívott eljárás külön százas, vagy ezres számcsoportot kapjon. Az egyes egységeket olyan utasításszám válassza el egymástól, amely két nullára végződik.

```
500 rem *** 6. lista ***
510 :
520 :
530 :
540 rem *** osszeadasmodul ***
550 rem v: valasz a feladatra
560 :
570 print
580 print "]]]]]]]]]]]]"x;"+"; y; "=q"
590 input v
600 if v<>x+y then print "rossz valasz! az osszeg:";x+y:gosub61150:goto60410
610 if v=x+y then print "a valasz helyes": gosub 61150: goto60400
620 :
630 :
640 rem *** kivonasmodul ***
650 rem v: valasz a feladatra
```

6. lista

1.3.4.2.2. Megjegyzések kódolása

Tanácsunk az, hogy csak a legszükségesebb mennyiségű megjegyzés (REM) legyen a programban. Ennek az az oka, hogy a megjegyzések helyet foglalnak a memóriából és ez — kis gépeknél — nem elhanyagolható.
A megjegyzéseknek azonban fontos szerepe is van, mert velük olyan információkat is dokumentálhatunk, melyek kóddal nem rögzíthetők.

További tanácsok:

— Minden programot olyan megjegyzéssel kezdjünk, amely egyértelműen azonosítja azt! (3. lista)
— Mindig adjuk meg a szerző(k) nevét és a készítés dátumát! A dátum alapján programunk elkülöníthető a korábbi változatokból (3. lista).

— A program nagyobb egységeit feltétlenül jelöljük meg önálló megjegyzéssorokkal! (5. és 6. lista)
— A modulokra vonatkozó magyarázatokat a szóban forgó modul elején tegyük meg (6. lista).
— Az adatok deklarálásakor az adatleírás sorban legyen megjegyzés, mely legalább az adat nevét megadja. Ha más, lényeges információ is tartozik az adathoz, célszerű itt dokumentálni.
— Gondosan ellenőrizzünk minden megjegyzést és ne feledkezzünk meg módosításkor, javításkor a karbantartásukról.

1.3.4.2.3. A program külalakja

A program külalakja nagyon fontos, mert befolyásolja a program olvashatóságát nemcsak mások, hanem a magunk számára is.

Tanácsaink a következők:

— Írjuk külön sorba a modulok belépő és kilépő utasításait!
— A BASIC alapszavak közé ne tegyünk szóközeket!
— A műveleti jelek elé és mögé ne tegyünk szóközt!
— A kifejezésekben, változónevekben ne legyenek szóközök!
— A szorosan összetartozó utasítások egy sorba kerüljenek! Pl.: 6000 FOR I=Ø TO 1500: NEXT I.
— Az előző példa alapján: a kettőspont előtt ne legyen, utána viszont legyen szóköz!

A program olvashatóságát, tagoltságát növeli, ha a modulok első REM sora előtt 1—2 üres sort írunk, melyben sorszám és kettőspont szerepel.

1.3.5. Tesztelés

Elkészült programunkat ki kell próbálni és így eldönthetjük, helyesen működik-e. Ha a programban hibát találunk, azt ki kell javítani és a próbát, a tesztet újból el kell végezni. Célszerű először modulonként kipróbálni, majd ha ez sikerrel járt, próbáljuk ki az egész programot. A teljes **program kipróbálása (tesztelése)** legcélszerűbben tervezett adatokkal történhet.

Számunkra megfelelőbb a modulonkénti próba, mert rövidebb, jobban át lehet tekinteni, ezért tesztelése, javítása egyszerűbb, gyorsabb.

Hiba keresésére használhatjuk a TRON parancsot is. Mint ismeretes, a TRON bekapcsolja a nyomkövetést. Ebben az üzemmódban a végrehajtott BASIC utasítások sorszámait az **interpreter** folyamatosan kiírja szögletes zárójelek között a képernyőre. Sok esetben ez is segít, hiszen innen leolvasható az utasítások végrehajtásának sorrendje, és ebből következtethetünk esetleges hibánkra.

Másik hibakereső módszer, amikor a STOP billentyűvel megállítjuk a programot és ilyenkor ki lehet íratni PRINT utasítással a *változó aktuális értékét*. Ebből következtethetünk arra, hogy a program adott pontján helyes-e a kért változó értéke.

1.3.6. Futtatás

A kész, vagy félkész programot, modult a RUN paranccsal futtatjuk. Ha a gép nem ír ki hibaüzenetet, megcsinál mindent, kiírja a READY üzenetet, programunk jól működik.

1.3.7. Értékelés

Tételezzük fel, hogy programunk hibátlan és már többször futtattuk. Az értékelés során meg kell vizsgálnunk, hogy megfelelő eredményt kaptunk-e a megoldás elve helyes volt-e, logikailag megfelelő-e a program. Célszerűen vizsgálhatjuk azt is, hogy a program rövidíthető-e, hiszen a rövidítéssel memóriát nyerhetünk. Ha találunk jobb, biztonságosabb megoldást ugyanarra a feladatra, programunkat célszerű átalakítani.

2. PROGRAMKÉSZÍTÉS PROGRAMMODULOKBÓL

A következőkben azt mutatjuk be, hogyan lehet egy egyszerű oktatóprogramot készíteni modulokból. A mintaprogram készítésénél igyekeztünk betartani a megadott elveket, programozási konvenciókat és a célszerűséget.

2.1. A feladat megoldása

Tételezzük fel, hogy gyakorló programot szeretnénk készíteni az általános iskola 5. és 6. osztályos tanulói számára az alapműveletek gyakorlásához. Szeretnénk, ha a program véletlenszerűen adná a feladatokat; meg lehessen adni a megoldandó feladatok számát; a program azonnal értékeljen és a végén is értékeljen százalékosan és érdemjeggyel. A különböző érdemjegyekhez különböző hangsort „muzsikáljon".

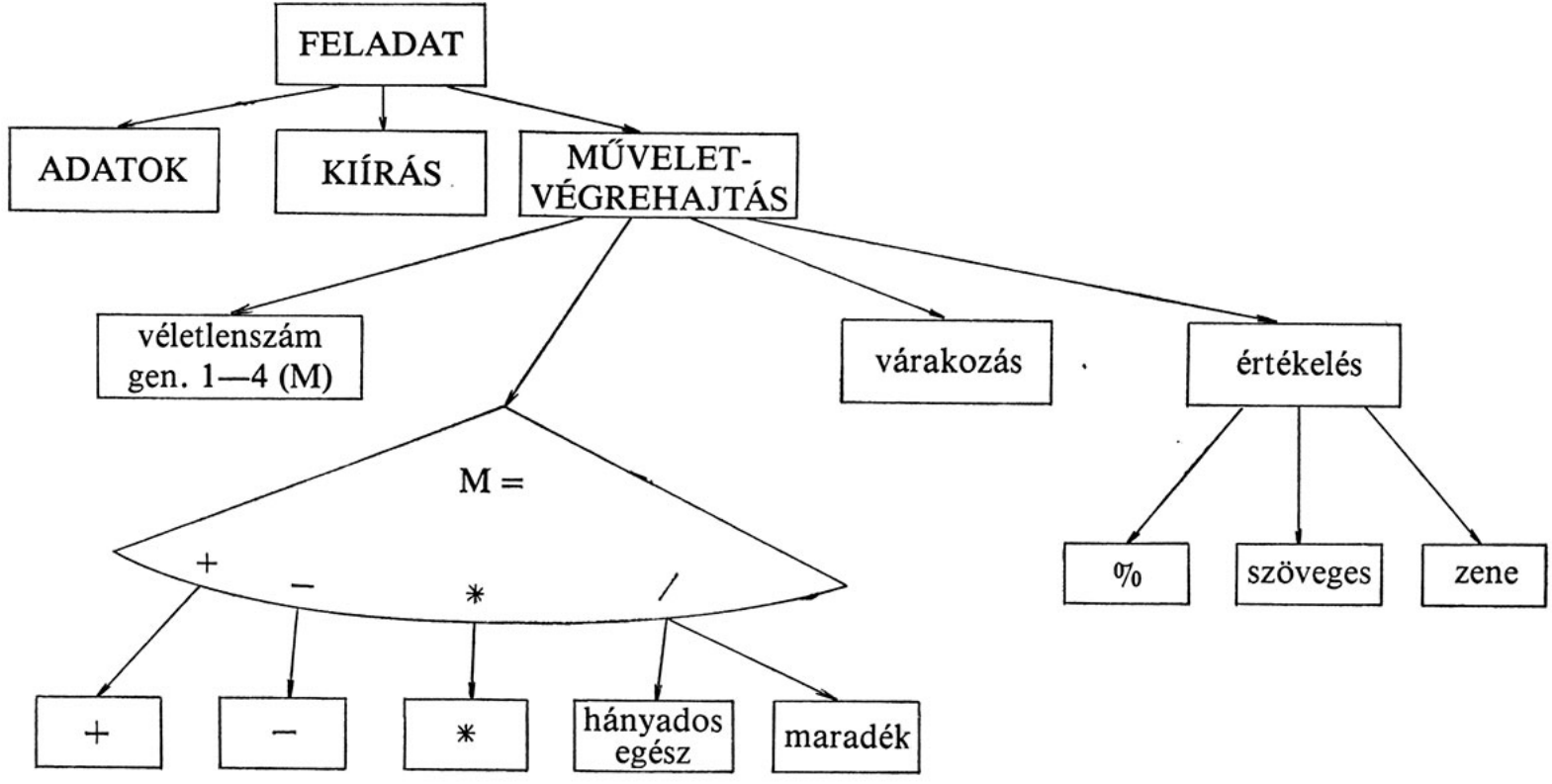
2.1.1. Feladatelemzés

A feladatot elemezve megállapíthatjuk, hogy kell egy olyan **adatmodul,** mely véletlenszerűen állítja elő a számokat, kell egy **vezérlőmodul,** mely szintén véletlenszerűen választja ki a műveleteket; kell összeadás-, kivonás-, szorzás- és osztásmodul; kell továbbá százalékszámító- és értékelőmodul stb.
Az elemzés során határozzuk meg a **főbb változókat:**

N — a feladatok száma,
H — a helyes válaszok száma,
V — a feleletek változója,
S — a százalékos teljesítmény,
X, Y, Z — véletlenszerű számok.

A feladat számítógépen megoldható, didaktikailag célszerű is, mert elsősorban egyénileg használják a tanulók és így önálló gyakorlásra is alkalmas.
Ezek után a feladat **megoldási algoritmusa** a következő lehet:

1. *Meghatározzuk (előállítjuk) az adatokat.*
2. *Véletlenszerűen kiválasztjuk a műveletet.*
3. *Műveletvégzés ellenőrzése, értékelése.*
4. *Ismétlés vagy vége.*


FELADAT
ADATOK
KIÍRÁS
MŰVELET-
VÉGREHAJTÁS
véletlenszám
gen. 1—4 (M)
várakozás
értékelés
M =
+
−
*
/
%
szöveges
zene
+
−
*
hányados
egész
maradék

13. ábra

2.1.2. A program tervezése

Tervünket tovább finomítva azt kapjuk, hogy szükség van egy kiírómodulra, hiszen a tanárt és a tanulót tájékoztatnunk kell. Ugyancsak szükséges, hogy az értékelésnél külön százalékszámító és külön szöveges értékelő modult készítsünk.

Külön érdemes elkészíteni a zenemodult. Szükség van továbbá egy várakozómodulra, mely szubrutinként arra szolgál, hogy a program csak egy billentyű megnyomására fusson tovább.

Tervezőmunkánk végeredményét, a modulokra bontást, a modulok egymáshoz való kapcsolódását az alábbi szerkezetábra szemlélteti (13. ábra).

2.1.3. A modulok tervezése

Célszerű a külső tervezéssel kezdeni. *Vizsgáljuk meg példaként a* **vezérlőmodul** *tervezését!* E vezérlőmodul feladat, hogy véletlenszerűen válasszon műveletet. Ehhez bevezetünk egy B változót, melynek értéke véletlenszerűen változhat 1-től 4-ig. Ezután 4 feltételt szabunk, nevezetesen, ha $B=1$, akkor összeadás, ha $B=2$, akkor kivonás, ha $B=3$, akkor szorzás, ha $B=4$, akkor osztás a művelet. Ennek a modulnak egy bejárata van és mindig egy kijárata, mely a feltételtől függ.

Tulajdonképpen egy másik vezérlés a százalékszámító modul is, hiszen ez az S változó értékétől függően fut az értékelő modul különböző helyeire.

Figyeljük meg mindkét modul listáját!

```
200 rem *** 7. lista ***
210 :
220 :
230 :
240 rem *** vezerlomodul ***
250 :
260 b=int(rnd(1)*4)+1
270 if b=1 then 60450
280 if b=2 then 60550
290 if b=3 then 60650
300 if b=4 then 60750
```

7. lista

```
60900 rem *** 8. lista ***
60910 :
60911 :
60912 :
60920 rem *** szazalekszamito modul ***
60930 :
60940 s=h*100/n
60950 :
60960 if s>90 then 61030
60970 if s>75 then 61090
60980 if s>50 then 61130
60990 if s>33 then 61170
61000 if s>0 then 61210
```

8. lista

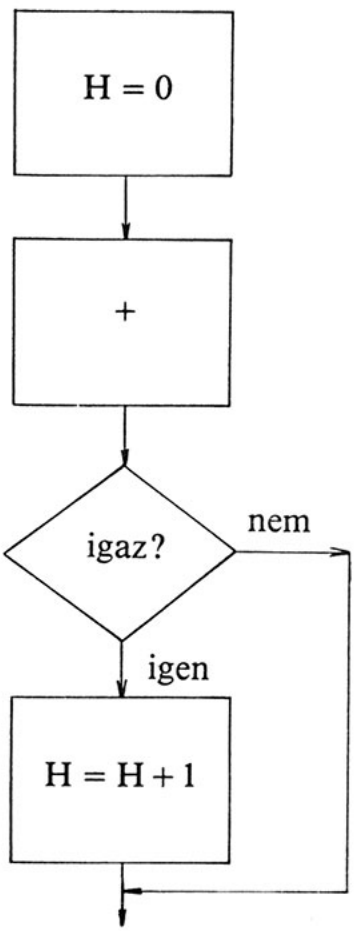
További példaként *vizsgáljuk meg egy* **eljárásmodul,** pl.: az összeadásmodul *tervezését!*

Ennek a modulnak az a feladata, hogy két, véletlenszerűen választott pozitív számból (1 és 100 között) állítson össze egy összeadást, ellenőrizze a választ és annak helyességétől függően folytatódjék a program. Ez utóbbit úgy kell érteni, hogy, ha a válasz helyes, a *H* változó értékét növelje 1-gyel, ha a válasz nem helyes, *H* változó értékét változatlanul hagyva fusson tovább a program.

A modul bejárata a 60 000-es sor, kijárata pedig a 60 050, vagy a 60 060-as sor.

A belső tervezésnél szükség van az *adatmodulban* előállított *X* és *Y* változókra. Szükséges továbbá az eredményadatok **kiírása** helytelen válasz esetén, vagy a helyes válasz jóváhagyó regisztrálása. Ha a válasz rossz, a billentyű-**várakozó modul** közbeiktatásával a gép továbblép, ha a válasz jó, akkor *H* változó (a helyes válasz száma) értékét növeli 1-gyel és úgy lép tovább.

A modul belső működését az alábbi folyamatábra és programlista szemlélteti:



14. ábra

```
500 rem *** 9. lista ***
510 :
520 :
530 :
540 rem *** osszeadasmodul ***
550 rem v: valasz a feladatra
560 :
570 print
580 print "]]]]]]]]]]]]"x;"+"; y; "=q"
590 input v
600 if v<>x+y then print "rossz valasz! az osszeg:";x+y:gosub61150:goto60410
610 if v=x+y then print "a valasz helyes": gosub 61150: goto60400
620 :
630 :
640 rem *** kivonasmodul ***
650 rem v: valasz a feladatra
```

9. lista

Hasonlóan készíthetjük el a kivonás, a szorzás és az osztás moduljait is.

2.1.4. A program kódolása

A folyamatábrával megtervezett programmodulokat megfogalmazzuk BASIC nyelven.

Az előzőleg bemutatott összeadásmodul folyamatábrája alapján könnyen készíthetünk utasítássorokat. A kódolásnál természetesen figyelembe kell vennünk a BASIC nyelv sajátosságait.

Mintaprogramunk úgy készült, hogy először elkészítettük az **eljárásmodulokat** (összeadó-, kivonó-, szorzó-, osztó-, értékelő), majd ezeket vezéreltük. A százalékszámító modul irányítja az értékelő és a zenemodult.

A példaprogramban igyekeztünk betartani a kódolási konvenciókat.

Felhívjuk a figyelmet a példaprogram külalakjára. Itt jól elkülönülnek egymástól a modulok, a részfunkciók. Az elkülönülést szolgálják a csak kettőspontot tartalmazó üres sorok. Igaz, hogy így programunk hosszabb, de a modulok és azok kapcsolata jobban megfigyelhető, szemlélhető.

Nézzük most a teljes programot!

```
59900 rem *** 10. lista ***
59991 :
59992 :
59993 :
60000 rem
60010 rem
60020 rem         ********************
60030 rem         *                  *
60040 rem         *   alapmuveletek  *
60050 rem         *                  *
60060 rem         * gyakorlo program *
60070 rem         *                  *
60080 rem         ********************
60090 rem
60100 rem keszitette: vitalis jenone
60110 rem             revesz  istvan
60120 rem             **** 1987 ****
60130 :
60140 :
60150 rem *** adatmodul ***
60160 :
60170 input "Shany feladat legyen";n: rem n: a feladatok szama
60180 h=0: rem h: a helyes valaszok szama
60190 for i=1 to n
60200 x=int(rnd(1)*100)+1
60210 y=int(rnd(1)*100)+1
60220 z=int(rnd(1)*10)+1
60230 :
60240 :
60250 rem *** kiiromodul ***
60260 :
60270 print"S"
60280 print"]]]]]]]]]]]]alapmuveletekqq"
60290 print"]]]]]]]]]]gyakorlo  program"
60300 print"qqqqqoldd meg a kovetkezo feladatokat!q"
60310 print"a megoldast ird a kerdojel utan!q"
60320 print"osztasnal elobb a hanyadost, majd"
60330 print"a maradekot ird!"
60340 :
60350 :
60360 rem *** vezerlomodul ***
60370 :
60380 b=int(rnd(1)*4)+1
60390 if b=1 then 60450
60400 if b=2 then 60550
60410 if b=3 then 60660
60420 if b=4 then 60770
60430 :
60440 :
60450 rem *** osszeadasmodul ***
60460 rem v: valasz a feladatra
60470 :
60480 print
60490 print "]]]]]]]]]]]]"x;"+"; y; "=q"
60500 input v
```

10. lista

```
60510 if v<>x+y then print "rossz valasz! az osszeg:";x+y:gosub61440:goto60890
60520 if v=x+y then print "a valasz helyes": gosub 61440: goto60880
60530 :
60540 :
60550 rem *** kivonasmodul ***
60560 rem v: valasz a feladatra
60570 :
60580 print
60590 print "]]]]]]]]]]]]"x;"-"; y; "=q"
60600 input v
60610 if v=x-y then print "a valasz jo": gosub 61440: goto 60880
60620 if v<>x-y then print "valaszod rossz!"
60630 print"a helyes kulonbseg ";x-y: gosub 61440: goto 60890
60640 :
60650 :
60660 rem *** szorzasmodul ***
60670 rem v: valasz a feladatra
60680 :
60690 print
60700 print "]]]]]]]]]]]]"x;"*"; z; "=q"
60710 input v
60720 if v=x*z then print "a valasz jo, a szorzat valoban ";x*z: gosub 61440: go
to 60880
60730 if v<>x*z then print "valaszod rossz!"
60740 print"a helyes szorzat ";x*z: gosub 61440: goto 60890
60750 :
60760 :
60770 rem *** osztasmodul ***
60780 rem v: valasz a feladatra
60790 :
60800 print
60810 print "]]]]]]]]]]]]"x;"/"; z; "=q"
60820 input "a hanyados ";v: input "a maradek ";m
60830 if v=int(x/z) and m=x-z*int(x/z) then 60850
60840 if v<>int(x/z) or m<>x-z*int(x/z) then 60860
60850 print"valaszod helyes ": gosub 61440: goto 60880
60860 print "a valasz rossz! a helyes hanyados "; int(x/z); "]]]a maradek ";x-z*
int(x/z)
60870 gosub 61440: goto 60890
60880 h=h+1
60890 next i
60900 :
60910 :
60920 rem *** szazalekszamito modul ***
60930 :
60940 s=h*100/n
60950 :
60960 if s>90 then 61030
60970 if s>75 then 61090
60980 if s>50 then 61130
```

10. lista

```
60990 if s>33 then 61170
61000 if s>0 then 61210
61010 :
61020 :
61030 rem *** ertekelo modul ***
61040 :
61050 print"Sqqqqq]]]]]ez igen szep munka volt!"
61060 print"q]]]]]teljesitmenyed ";int(s);"%"
61070 print"qq]]]]]osztalyzatod rBjelesDR":goto61260
61080 :
61090 print"Sqqqqq]]]]]munkad jo volt!"
61100 print"q]]]]]teljesitmenyed ";int(s);"%"
61110 print"qq]]]]]osztalyzatod rBjoDR":goto61310
61120 :
61130 print"Sqqqqq]]]]]ezt csinalhattad volna jobban is!"
61140 print"q]]]]]teljesitmenyed ";int(s);"%"
61150 print"qq]]]]]osztalyzatod rBkozepesDR":goto61340
61160 :
61170 print"Sqqqqq]]]]]nagyon sok a hianyossagod!"
61180 print"q]]]]]teljesitmenyed ";int(s);"%"
61190 print"qq]]]]]osztalyzatod rBelegsegesDR":goto61370
61200 :
61210 print"Sqqqqq]]]]]ezt az anyagot nem ismered!"
61220 print"q]]]]]teljesitmenyed ";int(s);"%"
61230 print"qq]]]]]osztalyzatod rBelegtelenDR":goto61400
61240 :
61250 :
61260 rem *** zenemodul ***
61270 :
61280 vol8
61290 sound1,600,30:sound1,688,30:sound1,741,30:sound1,812,60:gosub 61440: goto6
1500
61300 :
61310 vol8
61320 sound1,600,30:sound1,688,30:sound1,741,60:gosub 61440: goto61500
61330 :
61340 vol8
61350 sound1,600,30:sound1,688,30:sound1,741,30:sound1,688,60: gosub 61440: goto
 61500
61360 :
61370 vol 8
61380 sound1,600,30:sound1,688,60:sound1,600,60: gosub 61440: goto 61500
61390 :
61400 vol 8
61410 sound1,741,3?:sound1,688,30:sound1,600,60: gosub 61440: goto 61500
61420 :
61430 :
61440 rem *** varakozo modul ***
61450 :
61460 print "nyomj meg egy billentyut!": getkey a$
61470 return
61480 :
61490 :
61500 rem *** ismetles/vege modul ***
61510 :
61520 print"Sqqqqq]]]]]ujra? (i/n)":input a$
61530 if a$="i" then 60090
61540 if a$="n" then 61550
61550 print"Sqqqqq]]]]]tovabbi jo tanulast kivanok!"
61560 print"qq]]]]]viszontlatasra!"
61570 end
```

10. lista

2.1.5. *A program kipróbálása, javítása*

Minden programot, annak elkészülte után ki kell próbálni, le kell tesztelni. Begépelés után tegyük ezt saját programunkkal is! Néhány próbálkozás után látható, hogy programunk korántsem tökéletes. Jól működik ugyan, de vannak olyan tulajdonságai, amelyek megzavarhatják a futását. Ilyen pl.: az, hogy a feladat kiírása utáni RETURN-t válaszként fogadja és természetesen rossz válaszként. Itt az a megoldás, hogy *előírjuk a gép számára, milyen karaktereket fogadhat el* válaszként. Erre itt most nem térünk ki, legyen *ez a felhasználó számára egy feladat.*

A tesztelés során célszerű kipróbálni végig jó válaszokkal, végig rossz válaszokkal és vegyesen.

A fellelt hibákat ki kell javítani és a tesztet újból el kell végezni.

Programunkban módosításokat is végre lehet hajtani és így akár több programot is összeállíthatunk viszonylag kevés munkával.

Módosítási lehetőségek:

— *Az adatmodul átírásával más határok közötti számokat is előállíthatunk a szükségletnek megfelelően. Akár tizedes törtet is megadhatunk változóként.*
— *Készíthetünk olyan programot is, amely csak egy alapművelet gyakorlására szolgál. Ilyenkor a többi modult egyszerűen elhagyjuk és a vezérlést átírjuk.*
— *A százalékszámító modult is átírhatjuk saját kívánságunknak megfelelő értékekre.*
— *A szöveges értékelésben is alkalmazhatunk más mondatokat, vagy el is hagyhatjuk. Ugyanígy elmaradhat az érdemjegy is. Ilyenkor csak százalékos eredményt adunk meg, de feltétlenül legyen valamilyen értékelés.*

Ez csak néhány javaslat a módosításra. Természetesen ezeken kívül is még számos lehetőség kínálkozik. Javasoljuk a felhasználónak, *próbálkozzon módosításokkal, változtatásokkal.*

2.1.6. *Az eredmények értékelése*

Teszteléskor arra is ügyelnünk és figyelnünk kell, hogy a kapott eredmények számszakilag és logikailag helyesek-e. Vizsgáljuk meg azt is, hogy lehetséges-e programunkat lerövidíteni, vagy más — egyszerűbb — eljárást alkalmazni bizonyos részeknél. Amennyiben rövidíteni tudunk, vagy egyszerűbb eljárást találtunk, feltétlenül alkalmazzuk, hiszen memóriát takarítunk meg vele.

A program végleges változatában csak annyi REM sort hagyjunk, amennyi a jó megértéshez feltétlenül szükséges.

2.1.7. *Dokumentáció*

Ha a fent javasolt eljárással készítettük programunkat, a készítés során maradt néhány írásos anyag is: folyamatábrák, a változók jegyzéke, modullisták stb. Mindezek a program dokumentációját képezik. Amennyiben lehetőségünk van rá, feltétlenül nyomtassuk ki a végleges **programlistát.**

3. A MODULÁRIS PROGRAMOZÁS OKTATÁSA

Jelen útmutató csak *egy* programozási módszert mutat be. Ez a módszer haszonnal alkalmazható mind az általános-, mind a középiskolában. A tanulók könnyen megértik a lényeget: elemekből **(modulokból)** egyszerűen össze lehet építeni egy programot. Új fogalom nem túl sok van, ezeket el lehet sajátítani. Tapasztalataink szerint a moduláris programozás, mint módszer, beépítve a tanmenetbe egy tanév alatt biztonsággal megtanítható és megtanulható.

A módszer oktatása nem igényel előzetes ismereteket, elegendő, ha a BASIC egyszerűbb szabályait ismerik. Az útmutatóban külön tárgyaltuk, de az iskolában történő alkalmazásnál az egyes témarészeket a hozzá kapcsolható gyakorlati feladattal együtt tanítsuk. Az 1. és 2. fejezet feldolgozása hasonló, próbáljuk ezt kihasználni!

Az oktatók figyelmét felhívjuk arra, hogy minden résznél sok-sok gyakorlati példát mutassanak és a tanulókkal sok kis programot, modult írassanak. Jól felhasználhatók az útmutató mellékleteként forgalomba hozott — modulokat tartalmazó — programkazetták.

Törekedjünk arra, hogy a modulok közötti összefüggéseket világosan lássuk és láttassuk egy-egy programon belül, hiszen ez igen megkönnyíti a **kódolást,** a programkészítést.

Sok időt fordítsunk az előzetes tervező munkára, a program szerkezetének helyes kialakítására, az **algoritmus** logikájára, a **folyamatábrák** készítésére.

Külön felhívjuk a figyelmet a *kódolási konvenciókra.* Nem azok szigorú betartását javasoljuk, de mindenképpen szükséges egy következetes rend, ami munkánkat megkönnyíti.

A kész program külalakja is tükrözze igényességünket! Legyen a program jól áttekinthető, világos, egyértelmű! Magunktól is, másoktól is követeljünk meg ilyen munkát!

3.1. Javaslat a „PROGRAMMODULOK" c. útmutató iskolai feldolgozására

A moduláris programozás iskolai feldolgozására — végezetül — az alábbi tematikát javasoljuk:

Szept.	Moduláris programozás	
	Modulok csoportosítása	2 óra
Okt.	Feladatelemzés, folyamatábrák.	
	Modultípusok. Lebontások.	4 óra

Nov.	Példák és gyakorlás.	
	Egyszerű modulok készítése.	4 óra
Dec.	Feladatelemzés, modultervezés	4 óra
Jan.	Modultervezés.	2 óra
Febr.	Kódolás. Kódolási módszerek,	
	konvenciók. Gyakorlati példák.	6 óra
Márc.	Önálló feladat kiadása.	
	A program külalakja.	2 óra
Ápr.	Tesztelés, futtatás, javítás, értékelés.	2 óra
Máj.	Önálló feladat leadása.	
	Elemzés, hibakeresés.	4 óra

E módszertani segédlet a példáival együtt jól szolgálhatja a számítástechnikai oktatás ügyét. Az volt a célunk, hogy tanuló és tanár, kezdő és haladó eredménnyel forgathassa.

4. A KAZETTÁN TALÁLHATÓ PROGRAMOK LEÍRÁSAI

A programkazettán 52 db program található. Ezeket három csoportba lehet sorolni:

1. A programozást segítő modulok (gépi kódú programok, BASIC szubrutinok) Pl.: karakteres hardcopy, képernyő rollozása, scrollozása stb.
2. Ötletadó, illetve bemutató programok, melyekből modulokat lehet kiszedni és felhasználni, de önállóan is működőképesek.
   Pl.: rendezések, dátum-, ill. személyiszám ellenőrző stb.
3. Önálló felhasználói programok, melyek a tanítási órákon is használhatók. Pl.: összevonás, tizedestörtek, Kolombusz hajója stb.

Az önálló felhasználói programok többsége védett, nem lehet másolni és listázni. A legtöbb program azonban listázható, elemezhető.
A programok programozástechnikai színvonalai változók. Az ötletesség és az egyszerű sematikusság keveredik a munkákban. Ennek ellenére a legtöbb programot változatlanul hagytuk, mert elemzésük így tanulságos. A programokat zömmel pedagógusok készítették, de van közöttük tanulói munka is.

A programok betöltéséhez (a kereséshez) elegendő az azonosító első négy karakterét megadni, mert nincs közöttük olyan, amelyik valamelyik másikkal az azonosító első 4 karakterében megegyezne. Így pl.: LOAD"ALAP*" az ALAPMUV. azonosítójú programot keresi meg és tölti be a kazettáról.

1. *Program: Képernyő forgatása (rollozása) alulról felfelé:*

a) *A program azonosítója* (a programfájl neve): KEPE.F.A—F

b) *A program betöltése:* LOAD "KEPE.F.A—F"

c) *Rövid ismertető:*

A program a karakteres képernyő tartalmát forgatja úgy, hogy a képernyőt felülről elhagyó karaktersorok a képernyő alján ismét megjelennek. Forgatás közben a képernyő elmozdulása karaktersoronként történik.
A program a 60080—60090 DATA sorokban elhelyezett gépi kódú rutint a 60140—60160 sorokban szereplő FOR—NEXT ciklussal a helyére tölti. Ezt követően a rutin valamennyi hívása SYS 8192 (SYS DEC("2000")) egy sorral felfelé rollozza a képernyőt.

A forgatást végző képernyőrutin:

```
2
120:  0333
120:  0333
130:  0333                    .opt p,oo,p4
                          ;p2.mintaprogram
150:  0333                    *=   $0333   ;tetszöleges betöltési cím lehet
160:  0333
160:  0333
170:  0333           felso    =    $0c00    ;képernyö bal felsö sarka
180:  0333           also     =    $0fc0    ;képernyö bal alsó sarka
190:  0333
190:  0333
200:  0333
200:  0333 a2 00              ldx  #$00       ;x <= oszlopváltozó
210:  0335 bd c0 0f sor       lda  also,x     ;alsó sor olvasása
220:  0338
220:  0338 9d 00 0c           sta  felso,x    ;felsö sor írása
230:  033b
230:  033b e8                 inx           ;oszlopváltozó növelése
240:  033c
240:  033c e0 28              cpx  #$28       ;elérte-e a 40. oszlopot
250:  033e
250:  033e d0 f5              bne  sor        ;még nem
260:  0340
260:  0340 20 04 df           jsr  $df04      ;scroll le (interpreter rutin)
270:  0343
270:  0343 60                 rts
```

A rutin működése a közölt assembly lista és a lista magyarázatai alapján könnyen megérthető, assemblerrel tetszőleges helyre befordítható.

d) *Bemutató:*

A b) pontban leírtak alapján töltse be a kazettáról a programot. (Vigyázzon a programnév pontos megadására!). A LIST paranccsal listázza képernyőre, majd RUN-nal indítsa el! A működés a képernyőn látható.
A gépi rutin bármelyik saját programjához hozzá tölthető, csak arra kell vigyáznia, hogy olyan területre másolja be, amelyet sem az interpreter, sem pedig a felhasználói program nem használ.

2. *Program: Képernyő forgatása (rollozása) felülről lefelé:*

a) *A program azonosítója* (a programfájl neve): KEPE.F.F.—L

b) *A program betöltése:* LOAD"KEPE.F.F—L"

c) *Rövid ismertető:*

A program a karakteres képernyő tartalmát forgatja úgy, hogy a képernyőt alulról elhagyó karaktersorok a képernyő tetején ismét megjelennek. Forgatás közben a képernyő elmozdulása karaktersoronként történik.
A program a 60080—60090 DATA sorokban elhelyezett gépi kódú rutint a 60140—60160 sorokban szereplő FOR—NEXT ciklussal a helyére tölti. Ezt követően a rutin valamennyi hívása SYS 819 (SYS DEC ("0333")) egy sorral lefelé rollozza a képernyőt.
A forgatást végző képernyőrutin:

```
2
120:   2000
120:   2000
130:   2000                    .opt p,oo,p4
                          ;p1.mintaprogram
150:   2000                    *=   $2000    ;tetszöleges betöltési cím lehet
160:   2000
160:   2000
170:   2000            felso   =    $0c00     ;képernyö bal felsö sarka
180:   2000            also    =    $0fc0     ;képernyö bal alsö sarka
190:   2000
190:   2000
200:   2000
200:   2000 a2 00              ldx  #$00      ;x <= oszlopváltozó
210:   2002 bd 00 0c sor       lda  felso,x   ;felsö sor olvasása
220:   2005
220:   2005 9d c0 0f           sta  also,x    ;alsó sor írása
230:   2008
230:   2008 e8                 inx           ;oszlopváltozó növelése
240:   2009
240:   2009 e0 28              cpx  #$28      ;elérte-e a 40. oszlopot
250:   200b
250:   200b d0 f5              bne  sor       ;még nem
260:   200d
260:   200d 20 f6 de           jsr  $def6     ;scroll fel (interpreter rutin)
270:   2010
270:   2010 60                 rts
```

A rutin működése a közölt assembly lista és a lista magyarázatai alapján könnyen megérthető, assemblerrel tetszőleges helyre befordítható.

d) *Bemutató:*

A b) pontban leírtak alapján töltse be a kazettáról a programot. (Vigyázzon a programnév pontos megadására!) A LIST parancsal listázza képernyőre, majd RUN-nal indítsa el! A működés a képernyőn látható.

A gépi rutin bármelyik saját programjához hozzá tölthető, csak arra kell vigyáznia, hogy olyan területre másolja be, amelyet sem az interpreter, sem pedig a felhasználói program nem használ.

3. *Program: Szöveg mozgatása függőlegesen:*

a) *A program azonosítója* (a programfájl neve): MOZG.MODUL F

b) *A program betöltése:* LOAD"MOZG.MODUL F"

c) *Rövid ismertető:*

A program az A1$ és az A2$ sztringváltozókban megadott szöveget mozgatja a karakteres képernyőn, függőlegesen. Részletes használati utasítását a programlista REM sorai tartalmazzák. Ezért a program betöltését követően először listázza a képernyőre, s olvassa el az útmutatást!

A BASIC rutin a 60290—60320 sorokban található.

d) *Bemutató:*

A program betöltését követően parancs üzemmódban adjon értéket az A1$, A2$ sztringváltozóknak. A szövegek azonos hosszúak legyenek!

Pl.: A1$="MARI"; A2$="PALI" legyen.

GOTO 60000-rel indítsa a programot! A működés a képernyőn látható.

4. *Program: Szöveg mozgatása vízszintesen:*

a) *A program azonosítója* (a programfájl neve): MOZG.MODUL V
b) *A program betöltése:* LOAD"MOZG.MODUL V"
c) *Rövid ismertető:*
Ez a program a "MOZG.MODUL F" programhoz hasonlóan működik. Részletes használati utasítását a programlista tartalmazza.
A mozgató BASIC rutin itt a 60300—60330 sorokban található.
d) *Bemutató:*
A program indítása a "MOZG.MODUL F" programmal azonos módon történik, a működés a képernyőn látható.

5. *Program: Mintaprogram a karakteres képernyőn a karakterenkénti szövegmozgatáshoz:*

a) *A program azonosítója:* (a programfájl neve): SZOVEG MOZG.
b) *A program betöltése:* LOAD"SZOVEG MOZG."
c) *Rövid ismertető:*
A program a karakteres képernyőn BASIC utasításokkal történő szövegmozgatásra mutat példát. A mintaprogramban megadott szövegek azonos hosszúak legyenek!
A betöltést követően ne indítsa el a programot, hanem először nézze át tüzetesen a BASIC program listáját. Ez több jól elkülönített modult tartalmaz.
Az egyes rutinokat könnyű felismerni, mert megnevezéseik csillagsorkeretben vannak. Az értelmezést a REM sorok kommentárjai segítik.
d) *Működtetés:*
RUN-nal indítás után a képernyőről leolvashatók a tennivalók. Először A1$-t kell megadni. A nyilak [↑] a karaktereket jelölik. Ezt követi A2$ definiálása.
Ha a szöveghosszak eltérők, a program nem lép tovább.
A1$ és A2$ (a szövegek) megadása után menüből választhat a megfelelő számbillentyűvel. A programból csak [RUN/STOP] billentyűvel lehet kilépni!

6. *Program: Mintaprogram a karakteres képernyő négyirányú mozgatásához:*

a) *A program azonosítója:* (a programfájl neve) 4 IRANY+DEMO
b) *A program betöltése:* LOAD"4 IRANY+DEMO"
b) *A program betöltése:* LOAD"4 IRANY+DEMO"
c) *Rövid ismertető:*

A program egy olyan gépi kódú rutint tölt be, amelynek segítségével a karakteres képernyőt a négy fő irányba lehet mozgatni forgatással, vagy eltolással (scrollozással). A kazettán lévő bemutató-program a 1000—1370. DATA sorokban megadott kódokat a 12288 ($3000)-től 12585 ($3129)-ig terjedő címterületre tölti. A program használ 2×40 bájt memória-puffert a 12585 ($3129) — 12666 ($317A)

címterületen, így teljes terjedelme 12288 ($3000)-től 12666 ($317A)-ig 378 ($017A) bájt. A gépi rutin hívása és paraméterezése;

SYS12495, mód, irány

Az egyes paraméterek jelentése és a program működése a DEMO program listájából kiolvasható, vagy a bemutató-program futása közben a képernyőn látható.
Azoknak, akik rendelkeznek ASSEMBLER-rel közöljük a gépi rutin assembly listáját (lásd P3. mintaprogramot). Így a felhasználói programban tetszőleges helyre „befordítható". Ügyelni kell azonban arra, hogy olyan területen helyezzük el a program 378 bájtját, amelyet sem az interpreter [9.], sem pedig saját felhasználói programunk nem használ. Szükséges lehet a BASIC-mutatók [9.] állítása is.
A P3. mintaprogramot sokkal tömörebben és ügyesebben is el lehetett volna készíteni, de nem ez volt a célunk. Bemutatásra és oktatásra szántuk.

```
2
1010:  3000                    .opt p,oo,p4
                        ;p3.mintaprogram
1015:  3000
1015:  3000
1020:  3000                    *=   $3000
1025:  3000            p1      =    $d8        ;1.paraméter
1030:  3000            m1      =    $dc
1035:  3000            m2      =    $de
1040:  3000            m3      =    $dd
1045:  3000            m4      =    $da
1050:  3000            m5      =    $db
1051:  3000            m6      =    $df
1052:  3000            color   =    $053b      ;karakter szine
1053:  3000            albt    =    $d802       ;40-es szorzótábla
1054:  3000            flbt    =    $d81b      ;alsó-felső byte-ok
1055:  3000            sorf    =    $0c00      ;felső karakter-sor
1056:  3000            szif    =    $0800      ;felső szín-sor
1057:  3000            sora    =    $0fc0      ;alsó karakter-sor
1058:  3000            szia    =    $0bc0      ;alsó szín-sor
                        ;** a legfelső karaktersor mentése **
1060:  3000 a2 00      fment   ldx  #$00
1065:  3002 bd 00 0c   v1      lda  $0c00,x   ;a felső sor karakterei
1070:  3005 a4 d8              ldy  p1        ;p1 paraméter
1075:  3007 d0 02              bne  t2        ;p1<>0 -> rollozás
1080:  3009 a9 20              lda  #$20      ;p1=0 scroll
1085:  300b 9d 29 31   t2      sta  puf1,x    ;space a pufferbe
1090:  300e bd 00 08           lda  $0800,x   ;szinterület mentése
1095:  3011 a4 d8              ldy  p1        ;p1 paraméter
1100:  3013 d0 03              bne  t3        ;p1<>0 -> rollozás
1105:  3015 ad 3b 05           lda  color     ;a karakterhely szine
1110:  3018 9d 51 31   t3      sta  puf2,x    ;szín a pufferbe
1115:  301b e8                 inx           ;újabb karakterre
1120:  301c e0 28              cpx  #$28      ;utolsó oszlop-e
1125:  301e d0 e2              bne  v1        ;még van oszlop
1130:  3020 60                 rts
                        ;** a legalsó karaktersor mentése **
1140:  3021 a2 00      ament   ldx  #$00
1145:  3023 bd c0 0f   v3      lda  $0fc0,x   ;karakterek az alsó sorból
1150:  3026 a4 d8              ldy  p1        ;p1 paraméter be
1155:  3028 d0 02              bne  t4        ;p1<>0 igy rollozás
1160:  302a a9 20              lda  #$20      ;scroll-hoz a space
1165:  302c 9d 29 31   t4      sta  puf1,x    ;karakter a pufferbe
1170:  302f bd c0 0b           lda  $0bc0,x   ;szin a pufferbe
1175:  3032 a4 d8              ldy  p1        ;p1 paraméter
1180:  3034 d0 03              bne  t5        ;rollozás
1185:  3036 ad 3b 05           lda  color     ;karakter szine
1190:  3039 9d 51 31   t5      sta  puf2,x
1195:  303c e8                 inx           ;újabb oszlopra
1200:  303d e0 28              cpx  #$28      ;utolsó oszlop-e
1205:  303f d0 e2              bne  v3        ;még nem az utolsó karakter
1210:  3041 60                 rts
                        ;** puffer az alsó sorba **
1220:  3042 a2 00      fvisz   ldx  #$00
1225:  3044 bd 29 31   v2      lda  puf1,x    ;karakterpuffer
1230:  3047 9d c0 0f           sta  $0fc0,x   ;az alsó sorba
1235:  304a bd 51 31           lda  puf2,x    ;szinpuffer
1240:  304d 9d c0 0b           sta  $0bc0,x   ;az alsó sorba
```

```
1245:  3050 e8              inx          ;újabb oszlopra
1250:  3051 e0 28           cpx  #$28     ;utolsó oszlop-e
1255:  3053 d0 ef           bne  v2       ;még nem utolsó
1260:  3055 60              rts          ;vissza a "BASIC"-be
                      ;** puffer a felső sorba **
1270:  3056 a2 00     vissz ldx  #$00
1275:  3058 bd 29 31  v4    lda  puf1,x   ;karakter-puffer
1280:  305b 9d 00 0c        sta  $0c00,x  ;a felső sorba
1285:  305e bd 51 31        lda  puf2,x   ;szinpuffer
1290:  3061 9d 00 08        sta  $0800,x  ;a felső sorba
1295:  3064 e8              inx          ;újabb oszlopra
1300:  3065 e0 28           cpx  #$28     ;utolsó oszlop-e
1305:  3067 d0 ef           bne  v4       ;még nem
1310:  3069 60              rts          ;vissza a "BASIC"-be
                      ; ** mozgatás fel **
1320:  306a 20 00 30  fel   jsr  fment    ;a legfelső sor mentése
1325:  306d 20 f6 de        jsr  $def6    ;scroll fel (interpreter rutin)
1330:  3070 4c 42 30        jmp  fvisz
1335:  3073 20 21 30  le    jsr  ament    ;alsó sor mentésre
1340:  3076 20 04 df        jsr  $df04    ;scroll le (interpreter rutin)
1345:  3079 4c 56 30        jmp  vissz    ;puffer visszatöltése
                      ;** 40-es szorzótábla adatai **
1355:  307c bd 02 d8  tabl1 lda  $d802,x   ;alsó byte
1360:  307f 85 dc           sta  m1
1365:  3081 85 de           sta  m2
1370:  3083 bd 1b d8        lda  $d81b,x   ;felső byte
1375:  3086 85 dd           sta  m3
1380:  3088 29 03           and  #$03
1385:  308a 09 08           ora  #$08
1390:  308c 85 df           sta  m6
1395:  308e 60              rts
                      ; ** mozgatás jobbra **
1405:  308f a2 00     jobb  ldx  #$00
1410:  3091 20 7c 30  v7    jsr  tabl1    ;a 40-es szorzótáblához
1415:  3094 a0 27           ldy  #$27
1420:  3096 b1 dc           lda  (m1),y   ;utolsó karakter-oszlop
1425:  3098 a4 d8           ldy  p1       ;p1. paraméter
1430:  309a d0 02           bne  t6       ;p1<>0 ->rollozás
1435:  309c a9 20           lda  #$20     ;space a scroll-hoz
1440:  309e 85 da     t6    sta  m4
1445:  30a0 a0 27           ldy  #$27     ;utolsó színoszlop
1450:  30a2 b1 de           lda  (m2),y
1455:  30a4 a4 d8           ldy  p1       ;p1. paraméter
1460:  30a6 d0 03           bne  t7       ;p1<>0 ->rollozás
1465:  30a8 ad 3b 05        lda  color    ;karakter szine
1470:  30ab 85 db     t7    sta  m5
1475:  30ad a0 26           ldy  #$26
1480:  30af b1 dc     v6    lda  (m1),y  ;oszlopok másolása
1485:  30b1 c8              iny
1490:  30b2 91 dc           sta  (m1),y
1495:  30b4 88              dey
1500:  30b5 b1 de           lda  (m2),y
1505:  30b7 c8              iny
1510:  30b8 91 de           sta  (m2),y
1515:  30ba 88              dey
1520:  30bb 88              dey
1525:  30bc c0 ff           cpy  #$ff
1530:  30be d0 ef           bne  v6
```

```
1535:  30c0 c8                 iny
1540:  30c1 a5 da              lda  m4
1545:  30c3 91 dc              sta  (m1),y
1550:  30c5 a5 db              lda  m5
1555:  30c7 91 de              sta  (m2),y
1560:  30c9 e8                 inx
1565:  30ca e0 19              cpx  #$19
1570:  30cc d0 c3              bne  v7
1575:  30ce 60                 rts           ;vissza a "BASIC"-be
                     ;** belépés - kiértékelés **
1585:  30cf 20 d8 9d           jsr  $9dd8    ;"," ellenőrzése, egy byte x-be
1590:  30d2 e0 02              cpx  #$02
1595:  30d4 90 03              bcc  t1       ;ha x < 2-nél akkor tovább
1600:  30d6 4c 1c 99           jmp  $991c    ;illegal quantity hiba
1605:  30d9 86 d8     t1       stx  p1       ;1.paraméter tárolása
1610:  30db 20 d8 9d           jsr  $9dd8    ;újabb "," és egy byte beolvasása
1615:  30de e0 04              cpx  #$04
1620:  30e0 b0 f7              bcs  t1       ;ha x>=4 akkor illegal quantity hiba
1625:  30e2 e0 00              cpx  #$00
1630:  30e4 f0 84              beq  fel      ;mozgatás fel (0)
1635:  30e6 e0 02              cpx  #$02
1640:  30e8 f0 89              beq  le       ;mozgatás le (2)
1645:  30ea e0 01              cpx  #$01
1650:  30ec f0 a1              beq  jobb     ;mozgatás jobbra (1)
                     ;** mozgatás balra **
1655:  30ee a2 00              ldx  #$00
1660:  30f0 20 7c 30 v11       jsr  tab11    ;a 40-es szorzótábla adatai
1665:  30f3 a0 00              ldy  #$00
1670:  30f5 b1 dc              lda  (m1),y   ;első karakter-oszlop
1675:  30f7 c4 d8              cpy  p1
1680:  30f9 d0 02              bne  t8       ;p1<>0 így rollozás
1685:  30fb a9 20              lda  #$20     ;space a scroll-hoz
1690:  30fd 85 da     t8       sta  m4
1695:  30ff b1 de              lda  (m2),y   ;első színoszlop
1700:  3101 c4 d8              cpy  p1
1705:  3103 d0 03              bne  t9
1710:  3105 ad 3b 05           lda  color    ;karakter színe
1715:  3108 85 db     t9       sta  m5
1720:  310a c8        v10      iny
1725:  310b b1 dc              lda  (m1),y   ;oszlopok másolása
1730:  310d 88                 dey
1735:  310e 91 dc              sta  (m1),y
1740:  3110 c8                 iny
1745:  3111 b1 de              lda  (m2),y
1750:  3113 88                 dey
1755:  3114 91 de              sta  (m2),y
1760:  3116 c8                 iny
1765:  3117 c0 27              cpy  #$27
1770:  3119 d0 ef              bne  v10
1775:  311b a5 da              lda  m4
1780:  311d 91 dc              sta  (m1),y
1785:  311f a5 db              lda  m5
1790:  3121 91 de              sta  (m2),y
1795:  3123 e8                 inx
1800:  3124 e0 19              cpx  #$19
1805:  3126 d0 c8              bne  v11
1810:  3128 60                 rts           ;vissza a "BASIC"-be
```

```
                        ;-----------------------------------------
1820:  3129 00 00 00 puf1       .byte0,0,0
1822:  312c 00 00 00            .byte0,0,0
1824:  312f 00 00 00            .byte0,0,0
1826:  3132 00 00 00            .byte0,0,0
1830:  3135 00 00 00            .byte0,0,0
1835:  3138 00 00 00            .byte0,0,0
1840:  313b 00 00 00            .byte0,0,0
1844:  313e 00 00 00            .byte0,0,0
1846:  3141 00 00 00            .byte0,0,0
1848:  3144 00 00 00            .byte0,0,0
1850:  3147 00 00 00            .byte0,0,0
1852:  314a 00 00 00            .byte0,0,0
1854:  314d 00 00 00            .byte0,0,0
1856:  3150 00                  .byte0
                        ;-----------------------------------------
1870:  3151 00 00 00 puf2       .byte0,0,0
1872:  3154 00 00 00            .byte0,0,0
1874:  3157 00 00 00            .byte0,0,0
1876:  315a 00 00 00            .byte0,0,0
1880:  315d 00 00 00            .byte0,0,0
1885:  3160 00 00 00            .byte0,0,0
1890:  3163 00 00 00            .byte0,0,0
1894:  3166 00 00 00            .byte0,0,0
1896:  3169 00 00 00            .byte0,0,0
1898:  316c 00 00 00            .byte0,0,0
1900:  316f 00 00 00            .byte0,0,0
1902:  3172 00 00 00            .byte0,0,0
1904:  3175 00 00 00            .byte0,0,0
1906:  3178 00                  .byte0
                        ;-----------------------------------------
```

d) *Működtetés:*

A b) pontban leírtak alapján töltse be a programot. Mielőtt elindítaná nézze meg alaposan a programlistát. Sok a kezelés szempontjából hasznos információhoz juthat. RUN-nal indítva a kódok helyükre töltődnek, s megjelennek a képernyő kezeléséhez szükséges legfontosabb információk.

A bemutató-programban a képernyőt a [CRSR] (kurzor) billentyűkkel lehet elmozdítani.

Felhívjuk a figyelmét arra, hogy a DATA sorokban elhelyezett kódokat nem lehet tetszőleges helyre töltetni, mert közvetlen ugrási címeket tartalmaz a gépi kódú program. Ezért közöltük az assembly listát. Az ez alapján „befordított" kódokból viszont a kazettán található 13. program (azonosítója: "DATAKESZITO") segítségével könnyen készíthet a felhasználó BASIC programjához DATA-sorokat.

7. *Program: Rajzolás a karakteres képernyőre:*

a) *A program azonosítója:* (a programfájl neve): RAJZOLO

b) *A program betöltése:* LOAD"RAJZOLO"

c) *Rövid ismertető:*

A program a kurzor és az inverzkurzor segítségével rajzol a képernyőre, az elkészült ábrákat memóriába menti, vagy a memóriából a képernyőre másolja. Lehetőséget biztosít a képernyő törlésére, inverzbe váltására és a képernyőtartalmak összemásolására.
Jó lehetőséget ad kisgyermekeknek a számítógéppel való ismerkedésre.
A BASIC keretprogram gépi kódú programot tölt a helyére. Ennek rutinjai:

— A karakteres képernyő inverzbe váltása:
Elhelyezkedése: 12000 ($3200)-tól 12826 ($321A)-ig
Paraméterezése: —
Hívása: SYS 12800

— Memóriaterületek mentése:
Elhelyezkedése: 12827 ($321B)-től 12861 ($323D)-ig
Paraméterezése: a rutin olyan memóriaterület 1024 bájtját tudja menteni, amely kezdőcímének alacsonyabb helyiértékű bájtja $00. A magasabb helyiértékű címbájt a

POKE 12830, felső címbájt

utasítással adható meg.
Így a

POKE 12830,12

utasítást követően a rutin a 3072 ($0C00) címmel kezdődő 1024 bájtot másolja át.
A célterület is csak olyan memóriarész lehet, melynek alacsonyabb címbájtja $00. A célcím magasabb helyiértékű címbájtját a

POKE 12838, felső címbájt

utasítással lehet megadni.
Így a

POKE 12836,29

utasítást követően a rutin a 7424 ($1D00) címmel kezdődő területre másolja az 1024 bájtot.
A rutin hívása: (a paraméterezést követően)

SYS 12827

A memóriamásoló rutin működésének megértését segítik a keret BASIC program 60320.—60360. sorai.

— Memória másolása „vagy" kapcsolattal:
Elhelyezkedése: 12862 ($323E)-től 12908 ($326C)-ig

Paraméterezése: A rutin két 1024 bájtos memóriaterületet másol a karakteres képernyőre „vagy" kapcsolattal. Az átmásolandó memóriaterületek kezdőcímeire az előbb leírtak vonatkoznak.
Pl. a

```
POKE 12865,29

POKE 12873,41
```

utasításokat követően a rutin a 7424 ($1D00) és a 10946 ($2900) címekkel kezdődő memóriaterületek „vagy" kapcsolatából keletkező „összemásolt" képet rajzolja a karakteres képernyőre.
Hívása: (a paraméterezést követően)

```
SYS 12862
```

A memóriamásoló rutin működésének megértését segítik a keret BASIC program 60380.—60400. sorai.

Az ismertetett rutin paramétereit a gépi kódú program által elfoglalt memóriaterület adott című bájtjaira kell „beszúrni". Ezért a rutin más területre helyezése körülményes.

A P4. mintaprogram funkciója megegyezik az imént ismertetett rutinnal, de assembly listáját ASSEMBLER-rel [9.] tetszőleges helyre fordíthatja a felhasználó. A paraméterek átadása a 222 ($DE); 223 ($DF) címeken történik. A program működésének megfejtését a lista megjegyzései segítik.

```
2
120:  3200                    .opt oo,p,p4
                ;p4.mintaprogram
140:  3200                    *=   $3200
                ;
160:  3200          lbyt      =    $d8          ;másolási címmutató
162:  3200          hbyt      =    $d9
164:  3200          par1      =    $dc          ;célterület címmutató
166:  3200          parh      =    $dd
174:  3200          lpar      =    $da          ;célterület címmutató
176:  3200          hpar      =    $db
184:  3200          mem1      =    $de          ;célterület címmutató
186:  3200          mem2      =    $df
                ;
                ; ** képernyő inverzbe váltása **
                ;
200:  3200 a9 00              lda  #$00
210:  3202 a0 0c              ldy  #$0c
220:  3204 85 d8              sta  lbyt         ;képernyő címmutató
230:  3206 84 d9              sty  hbyt
240:  3208 a2 04              ldx  #$04         ;4*256 byte másoláshoz
250:  320a a0 00    v2        ldy  #$00
260:  320c b1 d8    v1        lda  (lbyt),y     ;karakter-byte beolvasása
270:  320e 49 80              eor  #$80         ;invertálás
280:  3210 91 d8              sta  (lbyt),y     ;vissza a helyére
290:  3212 c8                 iny               ;újabb byte
300:  3213 d0 f7              bne  v1           ;még nem #$ff
310:  3215 e6 d9              inc  hbyt         ;növelem 256-tal
320:  3217 ca                 dex               ;1*256 byte-tal kevesebb
330:  3218 d0 f0              bne  v2           ;az összes byte még nem inverz
340:  321a 60                 rts               ;vissza a "BASIC"-be
                ;
                ;** memóriaterületek mentése **
                ;
350:  321b a9 00              lda  #$00
360:  321d a4 de              ldy  mem1         ;mentési terület-mutató
370:  321f 85 d8              sta  lbyt
380:  3221 84 d9              sty  hbyt
390:  3223 a9 00              lda  #$00
400:  3225 a4 df              ldy  mem2         ;célterület-mutató
410:  3227 85 da              sta  lpar
420:  3229 84 db              sty  hpar
430:  322b a2 04              ldx  #$04         ;4*256 byte-ot ment
440:  322d a0 00    v4        ldy  #$00
450:  322f b1 d8    v3        lda  (lbyt),y     ;beolvasás
460:  3231 91 da              sta  (lpar),y     ;tárolás
470:  3233 c8                 iny               ;újabb karakter
480:  3234 d0 f9              bne  v3           ;még nem a 255.
490:  3236 e6 d9              inc  hbyt         ;újabb 256 byte
500:  3238 e6 db              inc  hpar         ;a célterületen is
510:  323a ca                 dex               ;1*256 byte-tal kevesebb
520:  323b d0 f0              bne  v4           ;még nem az utolsó
530:  323d 60                 rts               ;vissza a "BASIC"-be
                ;
                ;** másolás 'vagy' kapcsolattal **
                ;
540:  323e a9 00              lda  #$00         ;1. cím-mutató
550:  3240 a4 de              ldy  mem1
560:  3242 85 d8              sta  lbyt
570:  3244 84 d9              sty  hbyt
580:  3246 a9 00              lda  #$00         ;2. cím-mutató
590:  3248 a4 df              ldy  mem2
600:  324a 85 da              sta  lpar
610:  324c 84 db              sty  hpar
620:  324e a9 00              lda  #$00         ;képernyő cím-mutató
```

```
630:    3250 a0 0c          ldy  #$0c
640:    3252 85 dc          sta  parl
650:    3254 84 dd          sty  parh
660:    3256 a2 04          ldx  #$04       ;4*256 byte másolása
670:    3258 a0 00    v6    ldy  #$00
680:    325a b1 d8    v5    lda  (lbyt),y
690:    325c 11 da          ora  (lpar),y   ;a karakterek vagy kapcsolata
700:    325e 91 dc          sta  (parl),y   ;tárolás a képernyőre
710:    3260 c8             iny             ;újabb karakter
720:    3261 d0 f7          bne  v5         ;még nem 255.
730:    3263 e6 d9          inc  hbyt       ;újabb 255 darab
740:    3265 e6 db          inc  hpar
750:    3267 e6 dd          inc  parh
760:    3269 ca             dex             ;1*256 byte-tal kevesebb
770:    326a d0 ec          bne  v6         ;még nem az utolsó
780:    326c 60             rts             ;vissza a "BASIC"-be
```

d) *Működtetés:*

A b) pontban leírtak alapján töltse be a kazettáról a "RAJZOLO" azonosítójú programot. Elindítás előtt tanulmányozza át a program listáját. A RUN-nal indítást követően másolja le a képernyőről a vezérlőbillentyű tartozó funkciókat, hogy könnyen tudja kezelni a programot. A „Billentyűvel tovább" felirat megjelenésekor a [SPACE] és a [RUN/STOP] billentyűk kivételével bármelyikkel továbbléphet.

8. *Program: Alakzatok rajzolása a nagyfelbontású képernyőn*

a) *A program azonosítója:* PIXEL RAJZOLO

b) *A program betöltése:* LOAD"PIXEL RAJZOLO"

c) *Rövid ismertető:*

A program több BASIC rutinnal a nagy felbontású képernyőn rajzolásra mutat példát. Nagy felbontású üzemmódban a képernyő 320×200 elemi képpontot tartalmaz.

Vízszintesen: $0 \leq x \leq 319$; függőlegesen: $0 \leq y \leq 199$

Egy-egy elemi képpontot pixelnek neveznek. Innen a programazonosító: "PIXEL RAJZOLO"

d) *Működtetés:*

A b) pontban leírtak szerint betöltve a program RUN-nal indul. A vezérlést segítő billentyűk funkcióinak leírása a képernyő alján látható.

Példaként rajzoljunk egy négyszöget!

A RUN-nal indított program ún. üres módban jelentkezik be. Ez azt jelenti, hogy a rajzoló kurzort a kurzormozgató billentyűk segítségével tetszőleges irányban elmozdíthatjuk anélkül, hogy „nyomot hagyna" a képernyőn. Vigyük a villogó kurzort a képernyő bal oldalának felső harmadába. Váltsuk át a módot ún. rajzolási módba, majd az F2 funkciósbillentyű segítségével „tegyünk ki" egy pontot. (Ez lesz a négyszög bal felső sarka.) Ismét visszaváltva üres módba vigyük a kurzort a képernyő jobb oldalának alsó harmadába. Nyomjuk meg az F6 funkciós-

billentyűt! A program e jobb alsó sarokpontot összeköti a már korábban megjelölt bal felső csúcsponttal. Hasonlóan lehet kört és szakaszt rajzolni. F2-vel a kívánt helyre kell mozgatni az ún. gépi pixelkurzort, a kurzormozgató billentyűkkel a villogó kurzort be kell állítani, majd a megfelelő funkciósbillentyűvel el kell végezni a pixelkurzor és a villogó kurzor összekötését.

9. *Program: Karakteres képernyő nyomtatása* (szubrutin)

a) *A szubrutin azonosítójú:* KARAKTERES HCOPY
b) *A szubrutin betöltése kazettáról:* LOAD "KARAKTERES HCOPY"
c) *Rövid ismertető:*
Ez a szubrutin a karakteres képernyő tartalmát nyomtatja ki. Képes kvázigrafikus karakterek (táblázatok) nyomtatására is.
d) *Működtetés:*
A szubrutint fűzzük (MERGE; APPEND) vagy írjuk saját BASIC programunkhoz a 60050. sortól kezdődően. Programunk valahányszor meghívja e modult (GOSUB 60050) mindannyiszor kinyomtatja a karakteres képernyő tartalmát.

10. *Program: Kazettás BASIC programok összefűzése (I.):*

a) *A gépi kódú program azonosítója:* 3F16
b) *A program betöltése kazettáról:* MONITOR-ban: L"3F16"
c) *Rövid ismertető:*
Ez a gépi kódú rutin a memória 16128 (03F00)-től 16384 ($3FFF)-ig terjedő területén helyezkedik el. Két, a kazettás egységről egymás után beolvasott BASIC programot fűz egybe a beolvasás sorrendjében. Ha a második program sorszámai nagyobbak az első legnagyobb sorszámnál, akkor a rutin az összefűzött programokat átsorszámozza. Ellenkező esetben a RENUNBER-t kell használni.
d) *Használata:*
Monitor üzemben a b) pontban ismertetett módon töltsük be a kazettáról a gépi kódú programot. Indítása: G3F16-tal, vagy SYS DEC ("3F16")-tal történik. Ezt követően megjelenik a képernyőn a felirat: keresse ki a modult! (Kész = i.) A kazettás egység számlálójának segítségével megkeresve a modult, nyomjuk le az [i] billentyűt ha kész vagyunk. Az első BASIC program betöltődik. Ezt követően ismét megjelenik a felirat. Az újabb modul megkeresése után ismételten lenyomva az [i] billentyűt, a következő BASIC programot hozzáfűzi az előzőhöz és az összefűzött programokat átsorszámozza a rutin. A kapott program kimenthető. Kettőnél több BASIC program össze fűzését nem lehet a rutinnal megoldani. Mivel a program $3F00—$3FFF között helyezkedik el, ezért csak olyan BASIC programok összefűzésére alkalmas, amelyek együttes mérete 16125 bájt, vagy annál kisebb. Ezt a gondot oldja meg a P5. programlistán közölt assembly leírás. Megfelelő ASSEMBLER-rel tetszőleges helyre helyezhető (így a C Plus/4-es BASIC RAM-jának a végére is);

```
2
120:   3f00                    .opt oo,p,p4
                    ;p5.mintaprogram
140:   3f00                    *=   $3f00
                    ;
142:   3f00          setlfs    =    $ffba
143:   3f00          setnam    =    $ffbd
144:   3f00          load      =    $ffd5
145:   3f00          bsout     =    $ffd2
                     ;** az első program beolvasása **
160:   3f00 a9 01    elso      lda  #$01      ;logikai file-szám
170:   3f02 aa                 tax           ;eszköz (magnó)
180:   3f03 a8                 tay           ;parancs (1)
190:   3f04 20 ba ff           jsr  setlfs   ;logikai file létrehozása
200:   3f07 a9 00              lda  #$00     ;nincs filenév
210:   3f09 20 bd ff           jsr  setnam   ;filenév felállítása
220:   3f0c a9 00              lda  #$00     ;beolvasás mutató
230:   3f0e 20 d5 ff           jsr  load     ;"KERNAL LOAD"
240:   3f11 86 2d              stx  $2d      ;az utolsó beolvasott
250:   3f13 84 2e              sty  $2e      ;byte címe (mutató)
260:   3f15 60                 rts
                     ;** belépési cím  --  főprogram **
270:   3f16 20 bb 3f           jsr  keper    ;szöveg a képernyőre
280:   3f19 20 00 3f           jsr  elso     ;első program beolvasása
290:   3f1c 38                 sec           ;"BASIC"-kezdet mutató állítása
300:   3f1d 8a                 txa
310:   3f1e e9 02              sbc  #$02
320:   3f20 85 2b              sta  $2b
330:   3f22 98                 tya
340:   3f23 e9 00              sbc  #$00
350:   3f25 85 2c              sta  $2c
360:   3f27 20 bb 3f           jsr  keper    ;újabb szöveg a 2. prg. előtt
370:   3f2a a9 01              lda  #$01     ;logikai file-szám
380:   3f2c aa                 tax           ;eszköz a magno (1)
390:   3f2d a0 00              ldy  #$00     ;parancs (0)
400:   3f2f 20 ba ff           jsr  setlfs   ;logikai file létrehozása
410:   3f32 a9 00              lda  #$00     ;nincs filenév
420:   3f34 20 bd ff           jsr  setnam   ;filenév felállítása
430:   3f37 a5 2d              lda  $2d      ;beolvasás előtt a ré-
440:   3f39 38                 sec           ;gi filevéget 2-vel le-
450:   3f3a e9 02              sbc  #$02     ;jebb kell állítani
460:   3f3c aa                 tax
470:   3f3d a5 2e              lda  $2e
480:   3f3f e9 00              sbc  #$00
490:   3f41 a8                 tay
500:   3f42 a9 00              lda  #$00     ;újabb beolvasás magnóról
510:   3f44 20 d5 ff           jsr  load     ;"KERNAL LOAD"
520:   3f47 86 2d              stx  $2d      ;betöltési cím vége
530:   3f49 84 2e              sty  $2e
540:   3f4b a9 73              lda  #<meleg  ;"BASIC" meleg indítás átírása
550:   3f4d 8d 00 03           sta  $0300
560:   3f50 a9 3f              lda  #>meleg
570:   3f52 8d 01 03           sta  $0301
580:   3f55 a9 50              lda  #$50
590:   3f57 85 03              sta  $03      ;az új kezdet az átszámozáshoz
600:   3f59 a9 50              lda  #$50
610:   3f5b 85 04              sta  $04
620:   3f5d a0 02              ldy  #$02     ;a 2 byte-os cím vagy sorszám
630:   3f5f b1 2b              lda  ($2b),y
640:   3f61 85 14              sta  $14
650:   3f63 c8                 iny
660:   3f64 b1 2b              lda  ($2b),y
670:   3f66 85 15              sta  $15
680:   3f68 a9 08              lda  #$08
690:   3f6a 85 05              sta  $05      ;növekmény sor
700:   3f6c a9 00              lda  #$00
```

```
710:  3f6e 85 06          sta  $06
720:  3f70 4c 0d ac       jmp  $ac0d     ;átsorszámozó rutinra ("KERNAL")
730:  3f73 a9 86    meleg lda  #$86      ;"BASIC" ready módba
740:  3f75 8d 00 03       sta  $0300
750:  3f78 8d 01 03       sta  $0301
760:  3f7b a9 10          lda  #$10      ;"BASIC"-kezdet $1001
770:  3f7d 85 2c          sta  $2c
780:  3f7f a9 01          lda  #$01
790:  3f81 85 2b          sta  $2b
800:  3f83 a9 ab          lda  #<ujra    ;"BASIC" indítás újra
810:  3f85 8d 00 03       sta  $0300
820:  3f88 a9 3f          lda  #>ujra
830:  3f8a 8d 01 03       sta  $0301
840:  3f8d a9 0a          lda  #$0a
850:  3f8f 85 03          sta  $03       ;a növekmény $0a = 10
860:  3f91 a9 00          lda  #$00
870:  3f93 85 04          sta  $04
880:  3f95 a0 02          ldy  #$02      ;a kezdőcím
890:  3f97 b1 2d          lda  ($2d),y
900:  3f99 85 14          sta  $14
910:  3f9b c8             iny
920:  3f9c b1 2d          lda  ($2d),y
930:  3f9e 85 15          sta  $15
940:  3fa0 a9 0a          lda  #$0a      ;a növekmény $0a = 10
950:  3fa2 85 05          sta  $05
960:  3fa4 a9 00          lda  #$00
970:  3fa6 85 06          sta  $06
980:  3fa8 4c e0 ab       jmp  $abe0     ;az átsorszámozó rutinra
990:  3fab a9 86    ujra  lda  #$86      ;"BASIC" újra
1000: 3fad 8d 00 03       sta  $0300
1010: 3fb0 8d 01 03       sta  $0301
1020: 3fb3 c6 e3          dec  $e3
1030: 3fb5 f0 03          beq  e1
1040: 3fb7 4c ba 3f       jmp  e1
1050: 3fba 60       e1    rts          ;vissza a "BASIC"-be
                  ;** szöveg a képernyöre **
1060: 3fbb a0 00    keper lda  #$00
1070: 3fbd a9 00          lda  #$00
1080: 3fbf 85 ca          sta  $ca       ;kurzor oszlopa
1090: 3fc1 b9 d3 3f v1    lda  iras,y    ;szöveg a képernyöre
1100: 3fc4 20 d2 ff       jsr  bscut     ;"PRINT" a képernyöre ("KERNAL")
1110: 3fc7 c8             iny           ;újabb karakter
1120: 3fc8 c0 27          cpy  #$27      ;az utolsó is kiíródott-e
1130: 3fca d0 f5          bne  v1        ;még nem
1140: 3fcc a5 c6    v3    lda  $c6       ;billentyűkód beolvasása
1150: 3fce c9 21          cmp  #$21      ;i betü-e
1160: 3fd0 d0 fa          bne  v3        ;nem
1170: 3fd2 60             rts
                  ;** tárolt szöveg **
1190: 3fd3 4b 45 52 iras  .byt $4b,$45,$52   ;k e r
1200: 3fd6 45 53 53       .byt $45,$53,$53    ;e s s
1210: 3fd9 45 20 4b       .byt $45,$20,$4b    ;e   k
1220: 3fdc 49 20 41       .byt $49,$20,$41    ;i   a
1230: 3fdf 20 4d 41       .byt $20,$4d,$41    ;  m a
1240: 3fe2 47 4e 4f       .byt $47,$4e,$4f    ;g n ó
1250: 3fe5 4e 20 41       .byt $4e,$20,$41    ;n   a
1260: 3fe8 20 4d 4f       .byt $20,$4d,$4f    ;  m o
1270: 3feb 44 55 4c       .byt $44,$55,$4c    ;d u l
1280: 3fee 54 20 21       .byt $54,$20,$21    ;t   !
1290: 3ff1 20 28 4b       .byt $20,$28,$4b    ;  ( k
1300: 3ff4 45 53 5a       .byt $45,$53,$5a    ;é s z
1310: 3ff7 3d 49 29       .byt $3d,$49,$29    ;= i )
1320: 3ffa ff ff ff       .byt $ff,$ff,$ff
1330: 3ffd ff ff 00       .byt $ff,$ff,$00
```

11. *Program: Több kazettás BASIC program összefűzése (II.):*

a) *A program azonosítója:* LINKT

b) *A program betöltése kazettáról:* LOAD"LINKT"

c) *Rövid ismertető:*

Ez a BASIC program kazettás BASIC programokat (vagy modulokat) másol egymás mögé.

A programrészek sorszámai nem fedhetik át egymást!

A "LINKT" program részletes használati utasítása a programlista 200-tól 520-ig terjedő REM soraiban megtalálható.

d) *Használata:*

Töltse be a "LINKT" programot. A 170. sorban javítsa ki a REM 3,0 megjegyzést POKE 3,0-ra. Így a 170. sor helyesen:

```
170 PRINT CHR$(14)+CHR$(9): POKE 3,0: POKE4071,143
```

(E sort eredetileg azért „rontottuk el" REM-mel, hogy a felhasználó próbálkozásai során a LINKT program ne vesszen el! Próbálja RUN-nal elindítani a programot úgy, hogy a 170. sort kijavította. Kezdheti elölről a munkát, ha a kazettásegység nincs előkészítve.) Az előkészített és az összefűzendő programokat tartalmazó kazettát helyezze az adatmagnóba, majd a programot RUN-nal indítsa el. A program egymás után kéri a programrészeket a kazettáról, majd elkészíti az összefűzött programot.

Néhány fontos dologra azonban ügyelni kell!

— Az első program kötelezően az 1000. sorszámmal kell, hogy kezdődjön!

— Az összemásolandó programrészek mindegyikének az első utasítása kötelezően:

```
SYS 34892: POKE 43,1: POKE 44,16
```

— Az 500. sorban n-nek az összemásolandó programok számát kell értékül adni.

— Az 510. DATA sorban pontosan meg kell adni az összemásolandó modulok fájlneveit (azonosítóit).

12. *Program: Több, lemezen tárolt BASIC program összefűzése:*

a) *A program azonosítója:* LINKD

b) *A program betöltése kazettáról:* LOAD"LINKD" (betöltés után másolja lemezre!)

c) *Rövid ismertető:*

A program működése és kezelése értelemszerűen megegyezik a 11. sorszámú, LINKT azonosítójú programmal.

Betöltése után nézze át a LINKD program listáját és a REM sorokban a használati utasítást.

d) *Használata:*
Megegyezik a LINKT programéval.

13. *Program: gépi kódú program visszafejtése DATA sorokba:*

a) *A program azonosítója:* DATAKESZITO
b) *A program betöltése kazettáról:* LOAD"DATAKESZITO"
c) *Rövid ismertető:*
A program a memória tetszőleges területén elhelyezett gépi kódú programot visszafejt DATA sorokba. Így e programok BASIC programmal is betölthetők lesznek. A "DATAKESZITO" elvégzi az adatok hexadecimális — decimális átalakítását, és tetszőleges sorszámú és lépésközű sorokba helyezését.
d) *Használata:*
Gépelje be a P6. sz. mintaprogramot, majd ASSEMBLER-rel fordítsa le és helyezze el a 12288($3000)—12386($3062) címterületen.

```
2
110:   3000                    .opt oo,p,p4
                    ;p6.mintaprogram
120:   3000                    *=   $3000
                    ;
                    ;** karakteres hardcopy **
                    ;** c-16-ra c plus/4-re **
                    ;
200:   3000          open      =    $ffc0
202:   3000          chkout    =    $ffc9
204:   3000          clrchn    =    $ffcc
206:   3000          chrout    =    $ffd2
208:   3000          stop      =    $ffe1
210:   3000          close     =    $ffc3
                    ;
230:   3000          store     =    $67      ;átmeneti tároló
232:   3000          temp1     =    $71      ;képernyö címe l-byte
234:   3000          temp2     =    $72      ;képernyö címe h-byte
236:   3000          fnlen     =    $ab      ;filenév hossza
238:   3000          lf        =    $ac      ;logikai fileszám
240:   3000          sa        =    $ad      ;másodlagos cím
242:   3000          dn        =    $ae      ;egységszám
244:   3000          fnpoint   =    $af      ;filenév mutató
                    ;
250:   3000 a9 04               lda  #$04     ;printer egységszám a mutatóba
252:   3002 85 ae               sta  dn
254:   3004 a9 04               lda  #$04     ;logikai fileszám a mutatóba
256:   3006 85 ac               sta  lf
258:   3008 a9 00               lda  #$00     ;filenév mutató (nincs filenév)
260:   300a 85 af               sta  fnpoint
262:   300c a9 00               lda  #$00     ;képernyö "RAM" kezdöcíme
264:   300e a0 0c               ldy  #$0c
266:   3010 85 71               sta  temp1
268:   3012 84 72               sty  temp2
270:   3014 85 ab               sta  fnlen    ;a = 0 · (nincs filenév)
272:   3016 85 ad               sta  sa       ;nincs másodlagos cím
274:   3018 20 c0 ff            jsr  open
276:   301b a6 ac               ldx  lf       ;nyomtató kijelölése outputra
278:   301d 20 c9 ff            jsr  chkout
280:   3020 a2 19               ldx  #$19     ;max. 25 képernyösor
284:   3022 a9 0d    sorcikl    lda  #$0d     ;chr$(13) soremelés a nyomtatón
288:   3024 20 d2 ff            jsr  chrout   ;chr$(13) karakter kiküldése
290:   3027 20 e1 ff            jsr  stop     ;stop billentyü lekérdezése
292:   302a f0 2e               beq  exit     ;ha stop akkor vége
294:   302c a0 00               ldy  #$00     ;0. oszlop
296:   302e b1 71    oszlopc    lda  (temp1),y
300:   3030 85 67               sta  store    ;a képernyökódból "ASC-II"-kód
302:   3032 29 3f               and  #$3f
304:   3034 06 67               asl  store
308:   3036 24 67               bit  store
310:   3038 10 02               bpl  ellenor1   ;bpl * + 4
312:   303a 09 80               ora  #$80
314:   303c 70 02    ellenor1   bvs  ellenor2   ;bvs * + 4
316:   303e 09 40               ora  #$40
318:   3040 20 d2 ff ellenor2   jsr  chrout   ;karakter a nyomtatóra
320:   3043 c8                  iny
322:   3044 c0 28               cpy  #$28     ;utolsó oszlop-e a sorban
324:   3046 d0 e6               bne  oszlopc   ;ha nem, újabb oszlopra
326:   3048 98                  tya
328:   3049 18                  clc
330:   304a 65 71               adc  temp1    ;újabb oszlopra állás
332:   304c 85 71               sta  temp1
334:   304e 90 02               bcc  ellenor3   ;túlcsordulás van
336:   3050 e6 72               inc  temp2
338:   3052 ca          ellenor3 dex
340:   3053 d0 cd               bne  sorcikl
```

```
344:   3055 a9 0d            lda  #$0d     ;soremelés
346:   3057 20 d2 ff         jsr  chrout   ;vezérlőkarakter kiküldése
350:   305a 20 cc ff exit    jsr  clrchn   ;csatornák törlése
352:   305d a5 ac            lda  lf
354:   305f 20 c3 ff         jsr  close    ;nyomtató file lezárása
356:   3062 60               rts
```

Ezt követően töltse be a "DATAKESZITO" programot, majd RUN-nal indítsa el.

A program először annak a memóriaterületnek a kezdőcímét kérdezi, ahol a gépi kódú program található. Címként decimális számot kell megadni. Esetünkben a kezdőcím 12288.

A DATA sorokba fejtendő memóriaterület utolsó bájtjának címe (végcím:) 12386. Ezt követi az első DATA-sor ún. kezdő sorszámának megadása. Legyen ez a 200. Majd a programsorszámok közötti lépésköz 5. A szükséges kezdeti adatok megadását követően a "DATAKESZITO" program DATA-sorokba fejti vissza a gépi kódú, tárolt programot. DELETE60000— paranccsal töröljük a "DATAKESZITOT", s a P7. mintaprogram alapján írjunk betöltőt a DATA sorokhoz.

A P7. mintaprogram is a "DATAKESZITO"-vel készült.

```
100 rem p7. mintaprogram
110 rem ** karakteres hardcopy **
120 rem töltődés :12288-tól 12386-ig
130 :
140 rem    hívása: sys 12288
150 :
160 for i=12288 to 12386
170 read a:poke i,a
180 next i
190 end
195 :
200 data 169,004,133,174,169,004,133,172
205 data 169,000,133,175,169,000,160,012
210 data 133,113,132,114,133,171,133,173
215 data 032,192,255,166,172,032,201,255
220 data 162,025,169,013,032,210,255,032
225 data 225,255,240,046,160,000,177,113
230 data 133,103,041,063,006,103,036,103
235 data 016,002,009,128,112,002,009,064
240 data 032,210,255,200,192,040,208,230
245 data 152,024,101,113,133,113,144,002
250 data 230,114,202,208,205,169,013,032
255 data 210,255,032,204,255,165,172,032
260 data 195,255,096

ready.
```

14. *Program: Közvetlen input adatok beírása a DATA sorokba:*

a) *A program azonosítója:* INPUT TO DATA

b) *A program betöltése kazettáról:* LOAD"INPUT TO DATA"

c) *Rövid ismertető:*

Aki próbálkozott már nagymennyiségű adat DATA sorokba írásával, az tudja, hogy milyen hálátlan feladat. Figyelni kell a maximális megengedett sorhosszat (ez C Plus/4-nél 88 karakter), a vesszőket, az utasításszót stb. Ezt küszöböli ki a program. Az inputként bevitt adatokat automatikusan DATA sorba írja.

d) *Használata:*

A programnak a RUN-nal indítást követően a DATA sorok kezdő sorszámát kell megadni. Ez nem lehet 0, vagy 55000-nél nagyobb. A lépésköz megadása a programsorok növekedésének mértékét jelenti. Javasoljuk, hogy a lépésköz 2 legyen.

A képernyőn folyamatosan nyomon követhető a DATA-sorban még elhelyezhető karakterek száma (hossza). Az egyes adatok után nem kell vesszőt írni, azt a program automatikusan elhelyezi.

A DATA-sorok szerkesztéséből a "." (pont)-tal lehet kilépni. A felesleges programsorok a DELETE paranccsal törölhetők.

15—18. *Programok: Cirill betűk megjelenítése:*

a) *A programok azonosítója:* CIRILL 1; közvetlenül utána a kazettán: CIR 1
CIRILL 2; közvetlenül utána a kazettán: CIR 2

A "CIRILL 1" azonosítójú programmal csak a karakteres képernyőre lehet cirill betűket írni, a "CIRILL 2"-vel pedig MPS 801-es, MPS 803-as CITIZEN 120 D vagy EPSON nyomtatókra is.

b) *A program(ok) betöltése kazettáról:* LOAD"CIRILL 1" vagy
LOAD"CIRILL 2"

c) *Rövid ismertető:*

A programok bemutatási célra készültek. Segítségükkel a cirill ABC nagybetűit lehet a képernyőn és/vagy a nyomtatón megjeleníteni.

Egy betöltő és egy karakter-definiáló programból állnak. DATA sorokban tartalmazzák a karakter-definíciókat. Ezeket tetszés szerint meg lehet változtatni, így nemcsak cirill betűket, de más nemzeti ABC-k különleges betűit is ki lehet a programmal alakítani. A kazettán a betöltő programot közvetlenül a definiáló program követi.

d) *A programok használata:*

Először a "CIRILL 1", vagy "CIRILL 2" azonosítójú programok valamelyikét kell betölteni. Ez átdefiniálja a BASIC mutatókat és az F1 funkciós billentyűt. Felszabadítja a helyet az új karakterkészletnek. (Indítása RUN-nal). Ezt követően F1-re betöltődik a kazettáról a "CIR 1", vagy "CIR 2" azonosítójú program, melynek funkciói:

— A BASIC mutatók állítása az F2 és F3 funkciós billentyűkkel.
— Az átdefiniált karakter ROM (ill. RAM) mutató visszaállítása az F1 billentyűvel hiba után. (Hiba vagy a MONITOR-ból kilépés esetén a képernyő „összeomlik". Ekkor kell megnyomni F1-et!)
— A nagybetűs karakterkészlet átmásolása a RAM-ba, majd a megfelelő grafikus karakterek átdefiniálása. (Lásd a $10C0-tól $14FF-ig terjedő területet. 16 * 256 + 192 = 4288 = $10C0)
(A programot RUN-nal kell indítani!)
A programok kifejezetten bemutatási célra készültek, ezért BASIC-ből definiálják a nyomtatható karaktereket. Így a nyomtatás sebessége rendkívül lassú. Az éppen definiált karakterek a képernyő bal felső sarkában megjelennek. Minden egyéb tudnivalót a képernyő tartalmaz.

19—22. *Programok: Matematikai jelek megjelenítése:*

a) *A programok azonosítói:* MATEMATIKA 1;
közvetlenül utána a kazettán MAT 1
MATEMATIKA 2;
közvetlenül utána a kazettán: MAT 2
A "MATEMATIKA 1" azonosítójú programmal csak a karakteres képernyőre lehet matematikai jeleket írni, a "MATEMATIKA 2"-vel pedig a 15—18. programoknál felsorolt nyomtatókra.

b) *A program(ok) betöltése kazettáról:* LOAD"MATEMATIKA 1"; vagy
LOAD"MATEMATIKA 2"

c) *Rövid ismertető:*
Ezek a programok is bemutatási célra készültek. Segítségükkel matematikai jeleket lehet megjeleníteni a képernyőn és/vagy a nyomtatón.

d) *Használatuk:*
Teljes mértékben megegyezik a 15—18. programoknál leírtakkal, annyi különbséggel, hogy itt matematikai jelek definiálódnak.
Minden szükséges információt a képernyő tartalmaz.

23. *Program: Ellenőrzött input:*

a) *A program azonosítója:* INPUT RUTIN (szubrutin)

b) *A szubrutin betöltése:* LOAD"INPUT RUTIN" vagy APPEND programmal

c) *Rövid ismertető:*
Az "INPUT RUTIN" ellenőrzött inputot valósít meg. Megvizsgálja, hogy a bevitt karakter elfogadható-e? Ha igen, újabbat olvas, ha nem, hibajelzést ad.

d) *Használata:*
A közölt rutin szubrutinként használható. Hívása előtt meg kell adni a bemeneti paramétereket. Ezek:

bb:= a beolvasandó adatmező hossza. (Ha bb=5, akkor max. 5 hosszúságú karakterláncot fogad el a rutin. Kilépni [RETURN]-rel lehet)

ba:= a beolvasandó adat típusa. (Ha ba=1, akkor alfabetikus, ha ba=2, akkor alfanumerikus és ba=3 esetén numerikus.)

Az adott hosszúságú és típusú adat a rutin b$ sztringváltozójában adódik további felhasználásra.

A szubrutin paraméterezés utáni hívása GOSUB 60000.

Lokális változók: b, b$, q$, ba, bb, t.

24. *Program: Személyiszám ellenőrzése*

a) *A programmodul azonosítója:* SZEMELYISZAM ELL

b) *A szubrutin betöltése:* LOAD"SZEMELYISZAM ELL" vagy APPEND programmal

c) *Rövid ismertető:*
A közölt program által tartalmazott szubrutin (lásd. P8. mintaprogramot) az sz$ sztringváltozóban szóközmentesen megadott személyiszámból előállítja az X, k9, q9 kimenő paramétereket, melyek alapján a program eldönti, hogy létező, vagy hibás személyiszámról van-e szó.

d) *Használata:*
A szubrutin hívása előtt az sz$ sztringváltozóban helyezze el szóközmentesen a vizsgálandó személyiszámot. Ellenőrizze, hogy a sztring hossza 11-e? Ha igen: hívja a szubrutint (GOSUB 6310). Ezt követően vizsgálja meg a kimenő paramétereket! Döntsön a P8. mintaprogram 60200–60290. soraiban közölt feltételek szerint.

25. *Program: Dátum ellenőrzése*

a) *A programmodul azonosítója:* DATE

b) *A szubrutin betöltése:* LOAD"DATE" vagy APPEND-del

c) *Rövid ismertető:*
A program az időszámításunk utáni dátumok helyességét vizsgálja. Kifejezetten bemutatási célra készült, ezért a vizsgálat ún. feltétel sorait nem vontuk össze.

d) *Használata:*
Töltse be a programot, majd RUN-nal indítsa el. Ezt követően vizsgálja meg az alábbi P9. mintaprogramot!

```
60000 rem **    p8. mintaprogram     **
60005 :
60006 :
60010 rem ******************************
60020 rem *" Személyiszám ellenőrzése"*
60030 rem *"   S z u b r u t i n     "*
60040 rem ******************************
60050 :
60060 rem ******************************
60070 rem *     bemeneti változó :     *
60080 rem *    sz$   (személyiszám)    *
60090 rem ******************************
60100 :
60110 rem ******************************
60120 rem *    lokális változók:       *
60130 rem *  a9,b9,c9,d9,e9,f9,g9,h9   *
60140 rem *  i9,j9,k9, x, y,   q9      *
60150 rem ******************************
60160 :
60170 rem ******************************
60180 rem *     kimeneti változók:     *
60190 rem *        x, k9, q9           *
60200 rem *----------------------------*
60210 rem *ha x=0      ->nemlétező sze-*
60220 rem *              mélyiszám     *
60230 rem *ha k9=q9    ->jó, ill. létező*
60240 rem *és k9*q9<>0  személyiszám   *
60250 rem *                            *
60260 rem * ha k9<>q9 ->rosszul mega- *
60270 rem *              dott személyi-*
60280 rem *              szám          *
60290 rem ******************************
60300 :
60310 a9=val(left$(sz$,1))
60320 b9=val(mid$(sz$,2,1))
60330 c9=val(mid$(sz$,3,1))
60340 d9=val(mid$(sz$,4,1))
60350 e9=val(mid$(sz$,5,1))
60360 f9=val(mid$(sz$,6,1))
60370 g9=val(mid$(sz$,7,1))
60380 h9=val(mid$(sz$,8,1))
60390 i9=val(mid$(sz$,9,1))
60400 j9=val(mid$(sz$,10,1))
60410 k9=val(right$(sz$,1))
60420 x=1*a9+2*b9+3*c9+4*d9+5*e9+6*f9+7*g9+8*h9+9*i9+10*j9
60430 y=x/11
60440 q9=x-int(y)*11
60450 return
```

```
60000 rem **    p9. mintaprogram     **
60005 :
60010 rem ******************************
60015 rem *"   Dátum ellenőrzése      "*
60020 rem *"      (Szubrutin)         "*
60025 rem ******************************
60030 :
60035 rem ******************************
60040 rem *   bemeneti paraméterek:    *
60045 rem * a := az évszám             *
60050 rem * b := a hónap (számmal!)    *
60055 rem * c := a nap                 *
60060 rem *----------------------------*
60065 rem * s1:= 1. segédváltozó (első*
60070 rem *      hívás előtt s1=0 )    *
60075 rem ******************************
60080 :
60085 rem ******************************
60090 rem * lokális változók, tömbök :*
60095 rem * d(12) = a hónapok utolsó   *
60100 rem *          napjai            *
60105 rem *   i    = ciklusváltozó      *
60110 rem *  dt    = 2. segédváltozó    *
60115 rem ******************************
60120 :
60125 rem ******************************
60130 rem *   kimeneti paraméterek      *
60135 rem * s1<>0 az első hívás után   *
60140 rem * dt = 0 ,ha rossz a dátum   *
60145 rem * dt = 1 ,ha jó a dátum      *
60150 rem *----------------------------*
60155 rem * a rutin hívása előtt dt=3 *
60160 rem * kell, hogy legyen!         *
60165 rem ******************************
60170 :
60175 if s1<>0 then 60195
60180 dim d(12):s1=33
60185 data 31,28,31,30,31,30,31,31,30,31,30,31
60190 for i=1 to 12:read m : d(i)=m: next i
60195 if a<=0 then dt=0 : return
60200 if b<1 or b>12 then dt=0 : return
60205 if c<1 then dt=0 : return
60210 if b=2 and c=29 then 60220
60215 if d(b)<c then dt=0:return : else dt=1:return
60220 if a<>4*int(a/4) then dt=0 : return
60225 if a=100*int(a/100) then 60230:else dt=1:return
60230 if a<>400*int(a/400) then dt=0:return : else dt=1:return
```

A 60035—60165. REM sorokból a használathoz szükséges minden információt megtudhat.
(A "DATE" program és az itt közölt P9. mintaprogram nem azonos!)

26. *Program: Adatok beolvasása karakterenként:*

a) *A program azonosítója:* SPEC GETRUTIN

b) *A program betöltése:* LOAD"SPEC GETRUTIN"

c) *Rövid ismertető:*

A program egy szokásos GET-rutin működtetésére mutat példát. A GET parancscsal és az ismertetett rutinnal (P10. mintaprogram) karakterenként lehet adatokat beolvasni.

```
377 rem ** p10. mintaprogram  **
378 :
379 :
380 rem "ellenőrzött get rutin"
390 :
400 get a$
410 if a$="" then 430
420 goto 480
430 print chr$(rv);" ";chr$(157);chr$(146);:get a$
440 if a$<>"" then goto 480
450 ck=ck+1
460 if ck=m then ck=0:rv=154-rv
470 goto 400
480 ifa$=chr$(20)ando<>0theno=o-1:h=len(b$):b$=left$(b$,h-1):print" ";chr$(20);
490 if a$=r$ or o=om then print" ":o=0:om=0:return
500 if f=1 then if a$<"0" or a$>"9" goto 400
510 if f=2 then if a$<"0" or (a$>"9" and a$<"a") or a$>"f" goto 400
520 if f=3 then if a$<"a") or a$>"f" goto 400
530 if o<om then b$=b$+a$:printa$;
540 o=o+1:if o=om then print:o=0:om=0:return
550 goto 400
```

d) *Használata:*

A program kifejezetten bemutatási célra készült. Feladata valójában nem a számrendszerek közötti átváltás, hanem a karakterek beolvasásának ürügyén a GET-rutin működésének bemutatása.

400. sor: Karakter beolvasása a billentyűzetpufferből

410. sor: Ha, üres, ugrás a 430. sorra

430. sor: A szimulált space kurzor előző állapotának inverzbe váltása, a kurzor balra vissza egy pozícióval, majd a kurzor reverse állapotának kikapcsolása a GET előtt.

440. sor: Ha a billentyűzetpufferből nem jött karakter, ugrás a 450. sorra.

450. sor: A szimulált kurzor villogását előállító ciklus *ck* ciklusváltozójának növelése.

460. sor: Ha a ciklusváltozó elérte maximális értékét (m), akkor 0 alapértékre állítás, a kurzor reverse kapcsoló (*rv*) ellenkezőjére állítása és a 470. sorban vissza a 400. sorra. (PRINT CHR$(18)): reverse kapcsoló be; PRINT CHR$(146) reverse kapcsoló ki.)

Ha az rv=18 volt, akkor rv=164—rv alapján rv=146 lesz. rv=146 esetén rv új értéke rv=164—rv alapján: rv=18.

420. sor: Ha a 410. sorban kiderül, hogy a billentyűzetpufferből nem üres karakter került beolvasásra, akkor a program a 480. sorra ugrik.

480—550. sorok: A beolvasott karakter vizsgálata, mezővé fűzés (480. sor a DEL szimulálása, 490. sor maximális hossz és RETURN vizsgálat, 500—520 sorok "hexa karakterek" vizsgálata, 530. sor összefűzés...)

A 400—470. sorok a programból kimenthetők és a felhasználó saját programjába beépíthetők. A felhasználónak a saját kívánalmai szerinti feltétel vizsgálatról magának kell gondoskodnia.

27. *Program: Egész számok összeadása, kivonása:*

a) *A program azonosítója:* OSSZEVONAS

b) *Betöltése:* LOAD"OSSZEVONAS"

c) *Rövid ismertető:*

A program 5. osztályos tanulóknak készült és az előjeles egész számok összeadásának és kivonásának gyakoroltatását segíti.

Négy különböző feladattípusból 29 db-ot ad és a munka végén értékel.

d) *Használata:*

5., 6. osztályosok matematika óráin a gyakorláshoz ajánljuk. Minden a kezeléshez szükséges ismeretet a képernyő tartalmaz.

A programban található „sejtelmes zenét" a P11. mintaprogram (szubrutin) alapján más programokba is be lehet építeni.

```
5000 rem ** p11. mintaprogram **
5002 rem **  sejtelmes zene   **
5004 :
5006 vol7
5008 for k=1 to 3
5010 for i=1 to 2
5012 sound1,262,8
5014 sound2,7,5
5016 sound1,643,16
5018 sound3,1000,4
5020 sound1,262,8
5022 sound2,516,5
5024 sound1,643,8
5026 sound2,770,6
5028 sound1,262,8
5030 sound2,516,8
5032 sound1,643,16
5034 sound3,1000,4
5036 next i
5038 for i=1 to 2
5040 sound1,169,8
5042 sound2,7,4
5044 sound1,685,16
5046 sound3,1000,4
5048 sound1,169,8
5050 sound1,685,8
5052 sound2,739,4
5054 sound2,881,3
5056 sound2,953,2
5058 sound1,169,5
5060 sound1,685,16
5062 sound3,1000,4
5064 next i
5066 next k
5068 sound1,262,25
5070 sound2,7,25
5072 for i=1 to 10
5074 sound1,600+i*30,1
5076 next i
5078 for i=7 to 1 step -1:voli
5080 vol i
5082 sound3,1000,10
5084 next i
5086 :
5088 return
```

28. *Program: Mértékegységek átváltásának gyakorlása:*

a) *A program azonosítója:* MERTEKEGYSEG
b) *Betöltése:* LOAD"MERTEKEGYSEG"
c) *Rövid ismertető:*

A program általános iskolás tanulóknak készült és a 10 többszöröseivel való szorzást és osztást gyakoroltatja, majd a mértékegységek átváltásának gyakoroltatására tér át. Az átváltandó mennyiséghez nem adja meg, hogy milyen mértékegységre kell átváltani.

Ezt a megoldást tudatosan választotta a programozó, mert így a tanuló az összes, általa ismert mértékegységgel próbálkozhat.

Pontot azonban csak akkor kap, ha eltalálja, hogy a „gép melyikre gondolt".

d) *Használata:*

5., 6. osztályosok matematika óráin a mértékegységek átváltásának gyakoroltatásához ajánljuk. Minden a kezeléshez szükséges ismeretet a képernyő tartalmaz.

A program $D800-ról $4800-ra másolja a nagybetűs karakterkészletet, majd átdefiniálja a szükséges karaktereket. A karakterkészlet mozgatását a P12. mintaprogram végzi.

```
4 rem ** p12. mintaprogram **
5 rem ** karakterkészlet $4800-ra **
6 rem ** karakter-ram mutató$4800-ra **
7 :
8 for i=0 to 44
9 reada$:a=dec(a$)
10 pokedec("4350")+i,a
11 next i
12 sys dec("4350")
13 data a0,04,a2,ff,bd,00,d4,9d,00,44,ca,d0,f7,ad,00,d4,8d,00,44,ee,56,43
14 data ee,59,43,ee,5f,43,ee,62,43,88,d0,e0,a9,c0,8d,12,ff,a9,44,8d,13,ff,60
18 printchr$(8)
19 print"S"
```

A P13. programrészlet a karakterek átdefiniálását végzi a RAM-ban (az új helyükön).

```
2531 rem **   p13. mintaprogram   **
2532 rem ** karakterek átdefiniálása **
2533 :
2540 for i=0 to 7:read a:poke 18232+i,a:next i
2550 for i=0 to 7:read a:poke 18256+i,a:next i
2560 for i=0 to 7:read a:poke 18424+i,a:next i
2570 for i=0 to 7:read a:poke 18416+i,a:next i
2580 for i=0 to 7:read a:poke 18400+i,a:next i
2590 for i=0 to 7:read a:poke 18408+i,a:next i
2600 for i=0 to 7:read a:poke 18280+i,a:next i
2610 for i=0 to 7:read a:poke 18224+i,a:next i
2620 for i=0 to 7:read a:poke 18144+i,a:next i
2630 for i=0 to 7:read a:poke 18168+i,a:next i
2640 for i=0 to 7:read a:poke 18240+i,a:next i
2643 for i=0 to 7:read a:poke 17408+i,a:next i
2647 for i=0 to 7:read a:poke 17656+i,a:next i
2648 for i=0 to 7:read a:poke 17624+i,a:next i
2649 for i=0 to 7:read a:poke 17640+i,a:next i
2700 data 0,60,126,15,255,126,90,219
2710 data 60,126,24,219,255,189,195,60
2720 data 60,126,219,126,60,36,36,195
2730 data 0,0,0,192,240,48,24,24
2740 data 0,0,0,3,15,12,24,24
2750 data 24,24,48,240,192,0,0,0
2760 data 24,24,12,15,3,0,0,0
2770 data 112,8,16,32,120,0,0,0
2780 data 112,8,48,8,112,0,0,0
2790 data 24,40,72,120,8,0,0,0
2800 data 0,0,0,8,8,0,0,0
2803 data 135,67,61,1,0,0,0,0
2807 data 192,192,0,0,0,0,0,0
2808 data 225,194,188,128,0,0,0,0
2809 data 3,3,0,0,0,0,0,0
2810 return

ready.
```

29. *Program: Műveletek tizedestörtekkel (1):*

a) *A program azonosítója:* TIZEDESTORTEK

b) *A program betöltése:* LOAD"TIZEDESTORTEK"

c) *Rövid ismertető:*

Először a számok helyét kell meghatározni a számegyenesen, majd a tizedes-

törtek betűvel leírt értéke alapján a számértékekkel történő leírást lehet gyakoroltatni.

Végezetül közönséges törteket kell átírni tizedestört alakra.

d) *Használata:*

5. osztályosok matematika óráin gyakoroltatásra ajánljuk. A kezeléshez szükséges ismereteket a képernyő tartalmazza.

A program RUN-nal indítható.

30. *Program: Rendezési eljárások:*

a) *A program azonosítója:* RENDEZESEK

b) *A program betöltése:* LOAD"RENDEZESEK"

c) *Rövid ismertető:*

A program különféle rendezési eljárások gyűjteménye. A felhasználónak BASIC programjában nem kell megírni a kívánt eljárást, elegendő ha a megfelelőt e programból átveszi. Segítségével szomszédcserés, buborék, Shell—Metzner és gyors rendezésre van lehetőség.

d) *Használata:*

A rendezési eljárások gyorsaságának bemutatására is felhasználható a program, mert az egyszer megadott rendezendő elemkészletet megjegyzi, ha a felhasználó azt kívánja.

A működéshez szükséges információkat a képernyő tartalmazza. A minta keretprogram max. 100 elemű 3 jegyű számokból álló számhalmazokat tud rendezni. Kérjük a felhasználót, hogy a program használata előtt nézze át a megjegyzésekkel és magyarázatokkal ellátott programlistát!

31. *Program: Természetes számok prímtényezőkre (törzstényezőkre) bontása:*

a) *A program azonosítója:* PRIMT./

b) *A program betöltése:* LOAD"PRIMT./"

c) *Rövid leírás*

A prímszám olyan természetes szám, amely nem bontható fel nála kisebb természetes számok szorzatára, vagyis nincs 1-en és önmagán kívül más osztója. A számelmélet alaptétele szerint minden 1-nél nagyobb természetes szám vagy prímszám — sorrendtől eltekintve — egyértelműen felbontható prímszámok szorzatára. (Ez a törzstényezős felbontás.) A felbontható természetes számok összetett számok; a felbontásban szereplő prímszámok a törzstényezők.

Ugyanaz a törzstényező természetesen többször is előfordulhat a felbontásban. A program a számelmélet alaptétele szerinti törzstényezőkre bontásra mutat példát.

d) *Használata:*

Felhasználható önálló (ellenőrző) programként, de megfelelően átalakítva a prímeket előállító szubrutinként is.

Minden kezelési információ megtalálható a képernyőn.
Futtatás előtt célszerű a magyarázatokkal ellátott programlistát áttanulmányozni!

32. *Program: Közönséges törtek egyszerűsítése:*

a) *A program azonosítója:* EGYSZERUSITES
b) *A program betöltése:* LOAD"EGYSZERUSITES"
c) *Rövid leírás:*
A programmal p/q alakban megadott közönséges törteket lehet egyszerűsíteni, a számláló és a nevező összes közös osztóját megkeresni.
d) *Használata:*
Felhasználható önálló (ellenőrző) programként, de megfelelően átalakítva a közös osztókat előállító szubrutinként is.
Minden kezelési információ megtalálható a képernyőn.
Futtatás előtt célszerű a magyarázatokkal ellátott programlistát áttanulmányozni!

33. *Program: Prímszámok előállítása 0-tól n-ig:*

a) *A program azonosítója:* PRIMSZAMOK 0—N
b) *A program betöltése:* LOAD"PRIMSZAMOK 0—N"
c) *Rövid ismertető:*
A program célja annak az eljárásnak a bemutatására, amelynek segítségével a 0 és a N természetes számok közé eső valamennyi prímszám előállítható.
d) *Használata:*
A 32. programnál leírtakhoz hasonlóan.

34. *Program: Százalékszámítás:*

a) *A program azonosítója:* SZAZALEKSZAMITAS
b) *A program betöltése:* LOAD"SZAZALEKSZAMITAS"
c) *Rövid ismertető:*
A program feladatok megoldásainak ellenőrzését segítheti. Kifejezetten bemutatási célra készült.
Meghatározható segítségével az alap, a százalékláb és a százalékérték.
d) *Használata:*
A 32. programnál leírtakhoz hasonlóan.

35. *Program: A legnagyobb közös osztó és a legkisebb közös többszörös meghatározása:*

a) *A program azonosítója:* EUKLIDESZI ALG.
b) *A program betöltése:* LOAD"EUKLIDESZI ALG."
c) *Rövid ismertető:*
A program kifejezetten bemutatási célra készült. Az Euklideszi algoritmus a ma-

radékos osztásnak olyan végesszámú egymás utáni végrehajtása, amelynek segítségével a legnagyobb közös osztó meghatározható. Ha elvégezzük az Euklideszi (maradékos) osztást az a és a $b \neq 0$ egész számok között (vagyis megkeressük azokat a q_1 és r_1 egész számokat amelyekre igaz), akkor:

$$a = b \cdot q_1 + r_1 \quad (0 \leq r_1 < |b|).$$

Ezután b és r_1 között elvégezve a maradékos osztást

$$b = r_1 \cdot q_2 + r_2 \quad (0 \leq r_2 < r_1) \ldots \text{ és így tovább.}$$

Az r_i maradékok monoton csökkenése miatt végesszámú lépés után bekövetkezik, hogy

$$r_{n-1} = r_n \cdot q_{n+1} + r_{n+1} \quad (0 \leq r_{n+1} < r_n)$$

$$r_n = r_{n+1} \cdot q_{n+2}.$$

Ahol r_{n+1} éppen az "a" és a "b" legnagyobb közös osztója.

d) *Használata:*
Az előzőekhez leírtak szerint. Minden kezelési információ megtalálható a képernyőn.

36. *Program: Számelméleti feladatok megoldása:*

a) *A program azonosítója:* SZAMELMELET
b) *A program betöltése:* LOAD"SZAMELMELET"
c) *Rövid ismertető:*
A program az alábbi számelméleti feladatok megoldásának algoritmusait tartalmazza:
— prímszámvizsgálat (eldönti egy természetes számról, hogy prím-e)
— prímszám keresése (adott intervallumban megkeresi az összes prímet)
— prímtényezők (egy természetes szám összes törzstényezőit határozza meg)
— lnko és lkkt (két egész szám legnagyobb közös osztóját és legkisebb közös többszörösét határozza meg)
— binomiális együtthatók (a binomiális tételben szereplő binomiális együtthatókat határozza meg az

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \ldots (n-k+1)}{1\ 2 \ldots k}$$

képlet alapján.)

d) *Használata:*
A program listájából látható, hogy az eljárások szubrutinjai külön-külön is kiszedhetők, ill. felhasználhatók. Kérjük az Olvasót a programlistát gondosan tanulmányozza át.

37. *Program: Alapműveletek* (+, −, *, :) *gyakorlása:*

a) *A program azonosítója:* ALAPMUV.
b) *A program betöltése:* LOAD"ALAPMUV."
c) *Rövid ismertetés:*
A program kifejezetten bemutatási célra készült. A moduláris „programépítkezésre" mutat példát. Mindazonáltal a matematikai alapműveletek gyakoroltatására is alkalmas, a nyújtott teljesítményt értékeli.
d) *Használata:*
RUN-nal indul. A képernyő tartalmazza a működtetéshez szükséges információkat.

38. *Program: Számok átváltása tetszőleges számrendszerből tetszőleges számrendszerbe:* (max. 35-ös számrendszert tud kezelni!)

a) *A program azonosítója:* SZAMRENDSZ.ATVAL
b) *A program betöltése:* LOAD"SZAMRENDSZ.ATVAL"
c) *Rövid leírása:*
A program max. 35-ös számrendszerek között tud átváltani. A főprogram a 60060-as és a 60290-es sorok között helyezkedik el. Az átváltó modul a 60110—60250 sorokban található. A 60340—66590. sorok más programokba is beépíthető ellenőrzött (adatbeviteli) GET-rutint tartalmaznak.
d) *Használata:*
A program RUN-nal indul. Minden szükséges kezelési és felhasználói információ megtalálható a képernyőn.

39. *Program: Tetszőlegesen elhelyezett koordinátarendszerbe függvénygrafikon rajzolása:*

a) *A program azonosítója:* KOORD
b) *A program betöltése:* LOAD"KOORD"
c) *Rövid leírás:*
A programmal tetszőlegesen elhelyezett koordinátarendszerbe lehet parabolát, vagy egyenest rajzolni. A koordinátarendszer kezdőpontja a képernyőn a kurzormozgató billentyűkkel jelölhető ki és a [SPACE] billentyűvel rögzíthető.
d) *Használata:*
Önálló programként működik, de megfelelő átalakítással programmodulként is használható.

40. *Program:* Determináns kifejtése:

a) *A program azonosítója:* DETERMINANS KIF.
b) *A program betöltése:* LOAD"DETERMINANS KIF."
c) *Rövid leírás:*

A program max. 9-ed rendű determinánst tud kifejteni. A determináns elemeit a billentyűzetről lehet bevinni.

d) *Használata:*
A „papír — ceruza" módszerrel kifejtett determináns-értékek ellenőrzésére használható.

41. *Program: LISSAJOUS-görbék megjelenítése a grafikus képernyőn:*

a) *A program azonosítója:* LISSAJOUS

b) *A program betöltése kazettáról:* LOAD"LISSAJOUS"

c) *Rövid leírás:*
A Lissajous-görbék a Descartes koordinátarendszerben ábrázolt függvénygrafikonok, melyekben mindkét változó egy paraméter periodikus függvénye és az egyik változó frekvenciája egész számú többszöröse a másikénak. Az ábrázolást a kisebbik frekvenciájú változó egy periódusára elegendő elvégezni.

d) *Használata:*
Adjunk a=3; b=2; $\varphi=60$ értéket; vizsgáljuk a képernyőn megjelenő görbét. Különböző paraméterű merőleges rezgések összetevését vizsgálhatjuk a programmal.

42. *Program: Számtani sorozattal kapcsolatos feladatok megoldásának gyakorlása:*

a) *A program azonosítója:* SZAMTANI SOROZAT

b) *A program betöltése kazettáról:* LOAD"SZAMTANI SOROZAT"

c) *Rövid leírás:*
A program feladatokat ad a számtani sorozattal kapcsolatos számítások megoldásának gyakorlására. Ismerteti a szokásos jelöléseket, majd az általános iskolában tanult képleteket (ha azt a felhasználó kéri). Ezt követően különböző feladatokat ad. A feladatsor végén értékel.

d) *Használata:*
A feladatok a képernyőn jelennek meg, a gyereknek a füzetben kell számolnia. A program az eredményt a billentyűzetről várja. A helyes megoldást is közli, ha a begépelt adat hibás volt.

43. *Program: $y=a\cdot x+b$ alakban megadott függvények metszéspontjának meghatározása:*

a) *A program azonosítója:* LIN—EGY.R.

b) *A program betöltése kazettáról:* LOAD"LIN—EGY.R."

c) *Rövid ismertetés:*
A program $y=a\cdot x+b$ alakban megadott függvények metszéspontját határozza meg numerikusan és grafikusan is.

d) *Használata:*
A betöltött programot RUN-nal indítsuk. Adjuk meg a következő két lineáris függvényt: $y=2\cdot x-1$; $y=-3\cdot x+2$. A megoldást a grafikus képernyőn kapjuk. A program kifejezetten bemutatási célra készült, a kezeléséhez szükséges információkat a képernyőre írja.

44. *Program: $y=f(x)$ alakban megadott valós függvények ábrázolása:*

a) *A program azonosítója:* FUGGV.NAGYITAS
b) *A program betöltése a kazettáról:* LOAD"FUGGV.NAGYITAS"
c) *Rövid leírás:*
A programmal tetszőleges y=f(x) alakban megadott függvényt lehet ábrázolni, vagy transzformálni.
d) *Használata:*
RUN-nal indítást követően adjuk meg az

$$y = \frac{1}{x \cdot \sin(x \cdot 2\pi)}$$

függvényt. A program a függvénygörbét a grafikus képernyőre rajzolja.
Érdekes az

$$y = e^{\frac{1}{\sin(x)}}$$

függvény grafikonja is.

45. *Program: Magyarország „vaktérképe"*

a) *A program azonosítója:* MO VAKTERKEPE
b) *A program betöltése kazettáról:* LOAD"MO VAKTERKEPE"
c) *Rövid ismertető:*
A program Magyarország vaktérképét rajzolja a karakteres képernyőre ún. kvázigrafikus karakterekkel.
d) *Felhasználható:*
Tetszőleges BASIC programban szubrutinként.
Betöltés után nézze át a programlistát, majd RUN-nal indítsa a programot!

46. *Program: Európa fővárosai:*

a) *A program azonosítója:* FOVAROSOK
b) *A program betöltése kazettáról:* LOAD"FOVAROSOK"
c) *Rövid leírás:*
A program Európa fővárosainak tanítását, a szerzett ismeretek begyakoroltatását segíti. 7. osztályos tanulóknak készült.

d) *Használata:*
A RUN-nal indítást követően a program alkalmazásához és kezeléséhez szükséges minden információ megjelenik a képernyőn. A feladattípusok között a megfelelő számbillentyű lenyomásával lehet választani.

47. *Program: Óceánok, földrészek és földrajzi neveik:*

a) *A program azonosítója:* KOLOMBUSZ HAJOJA

b) *A program betöltése kazettáról:* LOAD"KOLOMBUSZ HAJOJA"

c) *Rövid ismertető:*
A programok a földrészek, óceánok meghatározását, földrajzi neveik helyesírásának gyakoroltatását segíti.

d) *Használata:*
A program indítását követően a felhasználáshoz és működtetéshez szükséges valamennyi információ megtalálható a képernyőn. Választani a megfelelő számbillentyű lenyomásával lehet.

48. *Program: Sorsolás számítógéppel:*

a) *A program azonosítója:* TOMBOLA

b) *A program betöltése kazettáról:* LOAD"TOMBOLA"

c) *Rövid ismertető:*
A programmal négyféle tombolajegy (piros, kék, zöld és fekete) sorsolható.
Ha valamelyik típusból egy sem kelt el, akkor az eladott tombolajegyek száma: Ø kell hogy legyen.
A programmal mindaddig lehet sorsolni, ameddig legalább egy még ki nem húzott tombolajegy van valamelyik típusból. Természetesen a sorshúzást hamarabb is abba lehet hagyni.

d) *Használata:*
A program indítását követően a felhasználáshoz és a működtetéshez szükséges valamennyi információ megtalálható a képernyőn.

49. *Program: Többfordulós versenyek értékelése:*

a) *A program azonosítója:* VERSENY

b) *A program betöltése kazettáról:* LOAD"VERSENY"

c) *Rövid ismertető:*
A program a legkülönbözőbb versenyek eredményeinek vezetésére és többfordulós versenyek esetén (is) a végső sorrend megállapítására és kinyomtatására alkalmas (a nyomtatás többször is elvégezhető).

d) *Használata:*
Megegyezik a 48. programnál leírtakkal.

50—52. *Program: Tornaversenyek pontszámainak feldolgozása:*

A programkazettán egymást követően 3 db tornaversenyt értékelő program található. Ezek azonosítói, betöltésük és blokk-számuk:
50. prg. LOAD"CSB.ES EGYENI" blokk-száma: 94
51. prg. LOAD"1. FORDULO" blokk-száma: 100
52. prg. LOAD"2. FORDULO" blokk-száma: 103
Tornaversenyértékelő programok kezelési útmutatója:
Egy-egy tornaversenyen minden csapat minden versenyzője, s az egyéni bajnokságon induló versenyzők is minden tornaszeren bemutatott gyakorlataikra értékelő pontszámot kapnak. Így a versenyen nagyszámú adatot kell rögzíteni, és a verseny végén az eredményhirdetéshez feldolgozni.
A számítógép használatával e munkát könnyebbé, pontosabbá tehetjük.
Nagy előny, hogy a verseny végén azonnal rendezett eredménylistánk lesz.
Tornaversenyeken a pontszámokat a SPORTPROPAGANDA nyomtatványain kell vezetni. A programok a hivatalos nyomtatványok fejrovataihoz igazodnak.
A tisztelt felhasználó három programot kap.
Mindegyik program azonos elven működik. A verseny típusától függően kell kiválasztani a megfelelőt.
A programok futtatása csak minimális számítástechnikai ismereteket kíván, viszont a tornasport adminisztrációs szabályaiban jártasnak kell lenni.
A biztonságos használathoz elegendő az első program részletes ismertetése és a másik kettőben az eltérések kiemelése.

1. Csb. és egyéni

Betöltése: LOAD"CSB.ES EGYENI"

Univerzális felépítésű, használható mind női, mind férfi versenyen, és együttesen is. Maximálisan 42 csapat és 42 egyéni versenyző pontszámait tudjasfolyamatosan fogadni. Betöltés után rövid tájékoztató szöveg jelenik meg. Lapozás után bekéri a futtatáshoz szükséges alapadatokat. (Versenyzők száma, neve stb.) A beírásokat a verseny megkezdése előtt időben kell elkezdeni, hogy az indulásra készen legyünk. Ezután a vezérlés automatikusan az első főmenüre kerül. Ez a program alaphelyezete. Minden befejezett akció után ide tér vissza, és innen ágaztatható el a csapatbajnokságra, vagy az egyéni versenyre, a [C], ill. az [E] billentyű megnyomásával.
Mindkét versenytípus egy újabb menüvel jelentkezik be, ahol a képernyőmaszkon látható betű, ill. szám megnyomásával választhatunk az egyes tornaszerek, vagy szolgáltatások közül. Ezeket a verseny alatt bármikor használhatjuk.

Pontok beírása csapatversenyen

A tornaszereknél működő versenybíróságoktól behozott jegyzőkönyvekből történik. Például az 5-ös rajtszámú csapat végzett ugráson. Nyomjuk meg az [U] betűt. A képernyőn megjelenik az UGRAS felirat, és a gép kéri a rajtszámot. Ezután kiírja a csapat nevét, kéri a jegyzőkönyvön lévő hat versenyző érvényes pontszámát. Egy csa-

patban hat versenyző indulhat, de csak a legjobb öt pontszám vehető figyelembe a csapateredményben. A gép kikeresi a legkisebb pontot, ezt kiejtve összesít, és kiírja a csapateredményt, amely a jegyzőkönyvre rávezethető. A képernyő két részre osztott, alól kiírja a legkisebb pont értékét, valamint folyamatosan összegzi a szerenként gyűjtött pontokat.
Így az utolsó szer pontszámainak beírása után rögtön kitölthető a jegyzőkönyv összesített csapateredmény rovata.
[RETURN] után a vezérlés visszakerül a főmenüre. További beírásoknál hasonló módon kell eljárni.

A program szolgáltatásai csapatversenyen:
A verseny végén, vagy verseny közben bármikor a gép több szolgáltatását vehetjük igénybe. Ezek a képernyőmaszkon látható megfelelő számbillentyűvel elérhetők:
1. A versenyjegyzőkönyv horizontális összesítését végzi. Az egyes versenyzők összpontszámát jelzi ki, valamint az összetett csapateredményt.
2. A kettes billentyűvel a szerenkénti helyezések kérhetők. Egymás után minden szer eredménye kiíródik rendezve, csapatnevekkel és az elért pontszámokkal együtt. Az eredmények nyomtathatók.
3. Meghívja a végeredmény kijelzést. (Verseny közben a pillanatnyi helyezéseket.) Ez is kinyomtatható.
4. Az összesítő jegyzőkönyvet nyomtatja ki a versenybíróság részére. Táblázatos formában látható minden csapat szerenkénti és összeredménye.

Pontszámok beírása egyéni versenyen:
A tornaszer kiválasztása, a rajtszám beírása ugyanúgy történik mint a csapatversenyen. A gép most megkérdezi az előírt és a szabadon választott gyakorlatok pontértékeit. A két pont összege azonnal megjelenik, s a jegyzőkönyvre rávezethető. A képernyő itt is két részre osztott. Az alsó részen halmozottan jelennek meg a szab. vál., és az összpontszám. Az utolsó szer beírása után a jegyzőkönyv megfelelő rovatai azonnal kitölthetők.

A program szolgáltatásai egyéni versenyen:
1. Az egyes billentyű képernyőre kéri az egyéni jegyzőkönyv formátumát, összes beírt adatával. Plusz szolgáltatás az átlagpont kiszámítása. Kérésre a jegyzőkönyv kinyomtatható.
2. Szerenkénti helyezések hívása.
3. Végeredmény hívása.
4. A négyes billentyű összesítő jk.-t nyomtat.

Hibajavítás:
A program a torna versenyszabályaihoz igazodik.
A megengedett 10 pontnál nagyobb értéket nem fogad el. Egy rossz beütést [RETURN] előtt az [INST/DEL] billentyűvel törölhetünk.
Sorléptetés utáni javításhoz a menükön keresztül újra hívjuk a tornaszert és a csa-

patot (vagy az egyéni versenyzőt), és a pontokat elölről kezdve jól beírhatjuk. Fontos! A gép mindig a beírt adatokat raktározza el a memóriába. Ha javítás, vagy ellenőrzés céljából meghívunk egy csapatot, egyéni versenyzőt, a már előzőleg beírt pontokat a képernyő bal szélén láthatjuk. Ha megtekintés után az F1-el visszatérünk a menüre ezek értéke nem változik meg. Ha viszont [RETURN]-el „üresen" lépkedünk át a pontszámokon, azok felülíródnak 0 (zéró) értékkel. Változtatás nélkül mindaddig visszaléphetünk a menüre, míg az erre utaló felirat a képernyő alján látható.

Csb. és egyéni első forduló:
Betöltése: LOAD"1. FORDULO"
Több versenyrendezési formában szokásos kétfordulós bajnokságok kiírása. A két verseny együttes eredménye dönti el a helyezéseket. Az 1. forduló eredményeit célszerű gépi úton tárolni, és a másodiknál felhasználni.
E program a főmenüben látható utasításra szalagra menti az adatokat. [HELP]
A program kezelése hasonló módon történik mint a csapat és egyéni bajnokságé.

Csb. és egyéni második forduló:
Betöltése: LOAD"2. FORDULO"
Szalagról beolvassa az első forduló adatait. A csapatok és versenyzők az 1. forduló rajtszámait kapják, ezért a beírásoknál ezt használja fel azonosításra. A 2. fordulóban azonban új rajtszámokat sorsolnak ki, ezért a régi számokat írjuk a jk.-ek sarkára, s ezeket használjuk. A szolgáltatások ugyanazok. Eltérés, hogy az összesítő jk.-ből kettő készít: egyet a 2. fordulóról, és egy összevontat az 1—2. fordulóról. Itt plusz versenyzők indítására nincs lehetőség.

FELHASZNÁLT IRODALOM

1. ALCOCK, Donald: Ismerd meg a BASIC nyelvet!
   Műszaki Könyvkiadó Budapest, 1983.
2. BAUMANN, Rüdeger: BASIC Eine Einführung in das Programmieren.
   Klett Stuttgart, 1980.
3. Bodor Tibor—Gerő Péter: A BASIC programozás technikája.
   Számalk Budapest, 1983.
4. Dusza—Varga: A BASIC nyelvű programozás ábécéje.
   Műszaki Könyvkiadó Budapest, 1985.
5. Dr. Hámori Miklós: Tanulás és tanítás számítógéppel.
   Tankönyvkiadó Budapest, 1984.
6. Dr. Kocsis András: Programozás BASIC nyelven I—II.
   Számalk Budapest, 1983.
7. Dr. Kocsis András: TV-BASIC.
   Számalk Budapest, 1984.
8. Szlávi—Zsakó: Módszeres programozás.
   Műszaki Könyvkiadó Budapest, 1986.
9. Dr. Ury László: Commodore C 16, C 116 BASIC és felhasználói kézikönyv.
   LSI Budapest, 1986.

88-3023 — Szegedi Nyomda
Felelős vezető: Surányi Tibor igazgató

A VORKER Kisszövetkezet ajánlja a Tisztelt Megrendelők figyelmébe az alábbi szoftver termékeket és — szolgáltatásokat.

1. C 16, C Plus/4, C 64 és Spectrum gépekre kifejlesztett oktató és játékprogramokat, amelyek Magyarországon a legolcsóbbak. Ezek közül az idegen és magyar nyelv, a matematika, a kémia, a biológia és a fizika tantárgyi oktatóprogramok segítik a pedagógusok és a tanulók munkáját.

2. Egy- és kétszemélyes játékok, valamint totóprogramok segítik a szabadidő jobb eltöltését.

3. IBM kompatibilis gépekre vállaljuk a munkaügyi és bérszámfejtő rendszerünk adaptálását.

Különleges hardver ajánlatunk: a világon egyedülálló fejlesztésünk a TC-NET+4 számítógép-interfész, amellyel 16 db C 16 illetve C Plus/4 számítógépet kapcsolhatunk lokális hálózatba. Ez két fontos célt szolgál: egyrészt az összes gép ilyen módon használhat egyetlen mágneslemezt és nyomtató egységet, másrészt megvalósítja számítástechnikai vonalon azokat az előnyöket, amelyeket a nyelvoktatás területén egy nyelvi labor biztosít. Ily módon számítástechnikai kabinet hozható létre kis anyagi ráfordítással.

Várjuk érdeklődésüket!

VORKER® Vállalatközi
Organizációs Ipari Szolgáltató
és Kereskedelmi Kisszövetkezet
Szeged, Pf: 711.
Telex: 82-688
Telefon: (62) 26-144
(62) 25-479