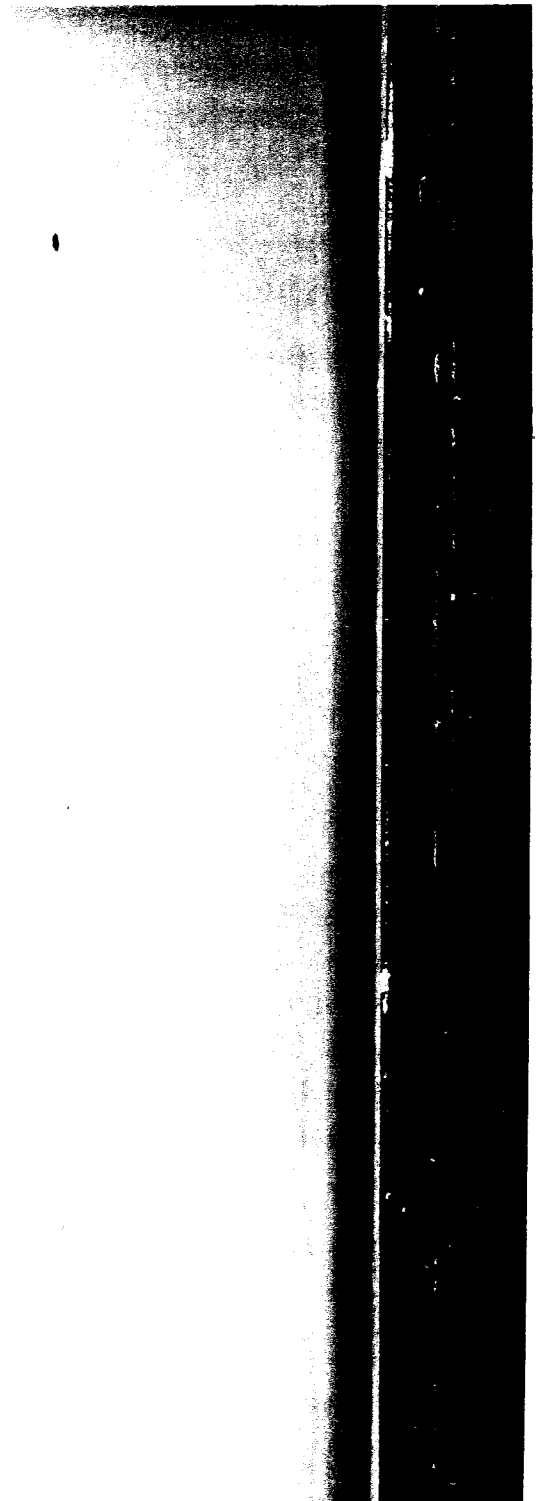
60.– Ft



Dusza Árpád

A HT-1080Z iskolaszámítógép bemutatása programokkal.

DUSZA ÁRPÁD

A HT–1080 Z ISKOLASZÁMÍTÓGÉP BEMUTATÁSA PROGRAMOKKAL

TISZTELT VÁSÁRLÓ!
FELHIVJUK FIGYELMÉT,
HOGY E KÖNYV PROGRAM-
JAÍ C-16, C-64 ÉS
PRIMO SZÁMÍTÓGÉPEKRE
ÍS ELKÉSZÜLTEK.

MEGRENDELÉSHEZ
MINTA
A KÖNYV UTOLSÓ
OLDALÁN TALÁLHATÓ.

Lektorálta:

KŐHEGYI JÁNOS
és
SZÉKELY JENŐ

Szerkesztette:

B. LÁZÁR ISTVÁN

Borító:

BOGÁR GYÖRGY

ISBN 963 03 2263 3

Kiadja:
a Borsod–Abaúj–Zemplén megyei Tanács VB. Művelődési Osztálya
a Miskolc Városi Művelődési Központ gondozásában
Felelős kiadó: BORSOS ÁRPÁD osztályvezető
Készült:
a Borsod–A.–Z. megyei Kórház–Rendelőintézet nyomdájában
85–144. 3000 példány, eng.szám: 54921
Felelős vezető: Lovászik János

TARTALOMJEGYZÉK

ELŐSZÓ

Magyarország minden középiskolájában van legalább egy HT–1080Z típusú személyi számítógép. Tanárok, tanulók ismerkednek vele. Ehhez szeretnék segítséget adni könyvemmel.

Az utasításokat, az utasítások alkalmazását programokkal mutatom be. A programok egyben a számítógép lehetőségeivel is megismertetnek. A programok futtatásával, azok kiegészítésével, módosításával, javításával éppúgy tanulhatunk, mint azok értelmezésével. A programok megértését, új programok irását segíti az alkalmazott algoritmusok leírása, a program szerkezetének és programozási fogások magyarázata.
A leghatékonyabb tanulás természetesen az, ha magunk írjuk a programokat. A könyvben található néhány ötlet esetleg újabbakat szülhet, vagy szebb megoldásokra inspirálhat.

A mintaprogramokkal a program készítésének menetét, az algoritmus tervezését, folyamatát kísérlem meg bemutatni néhány konkrét alkalmazási feladat megoldásának a leírásával.

Nagyon szép és izgalmas feladat a számítógép mellett ülve tapasztalni, hogy mit tud. Jelenleg erre még nem mindenkinek van lehetősége, mert nincs annyi számítógép. Könyvemmel azoknak a tanár kollégáknak szeretnék segíteni, akik inkább hagyják tanítványaikat a számítógépen dolgozni (azon az egyen) és választják a kevésbé „játékos" utat, könyv segítségével szeretnének megismerkedni a HT–1080Z számítógéppel, a programozással.

A tanulók számára is hasznos lehet ez a könyv. Talán gyorsabban jutnak el a számítógép megismerésétől a számítógép alkalmazásáig.

Több mint tíz éve, mióta számítástechnikai ismereteket tanítok, minden évben más tematikával, feladatsorral kísérletezem. Tekintse a tisztelt Olvasó ezt a könyvet is egy kísérletnek.

Miskolc, 1983. november

Ez a könyv 1983–ban íródott. Kiadására végül a Borsod megyei Tanács jóvoltából kerülhetett sor. Ezért köszönettel tartozom.

Miskolc, 1985. szeptember

A szerző

1. ISMERKEDÉS A SZÁMITÓGÉPPEL

A számítógépet ismerni annyit jelent, hogy tudjuk azt, hogy mire használható, és tudjuk kezelni, programozni.
A személyi számítógépek tömeges megjelenésével ezeknek az ismereteknek a megszerzése ma már nem ütközik akadályba. A számítógép emberközelbe került. Ott van az iskolában, a munkahelyeken és a lakásokban. Mindenki számára elérhetővé vált.
Könyvünkben a személyi számítógépek egyik programnyelvéről, a BASIC (ejtsd: bézik) –ről lesz szó, közben kezelési ismeretekre is szert teszünk és megismerkedhetünk a mintaprogramok segítségével néhány alkalmazási feladattal is.

Ahhoz, hogy a programnyelvet megismerjük és alkalmazzuk, nem szükséges, hogy a számítógép belsejét, működését részletesen ismerjük. A számítógép lelke a *központi egység*, de ezenkívül mindenképpen szükséges egy *billentyűzet* és egy *megjelenítő* (TV), hogy kapcsolatot tudjunk teremteni a számítógéppel. Ez így egy minimális kiépítettségű számítógép.

A *központi egységet* a mikroprocesszor, a memória és olyan áramköri elemek alkotják, amelyek az egységek között biztosítják a kapcsolatot. Mindezek általában a billentyűzet alatt találhatók valamilyen „dobozban".

Az ember és a számítógép kapcsolata kétirányú. A *billentyűzettel* mi teremthetünk kapcsolatot a számítógéppel. Ezen gépelhetjük be *parancsainkat* és azt az utasítássorozatot, amivel megadjuk, hogy egy feladat megoldása hogyan történjék, azaz a *programot*.
Amit a számítógép közöl velünk, azt a *megjelenítőn* (display) látjuk, ami legtöbbször egy közönséges TV készülék. Itt jelenik meg az is, amit mi gépeltünk be. A párbeszéd így nyomon követhető a képernyőn.

A programokat mágnesszalagon, vagy mágneslemezen tárolhatjuk. Ehhez *magnetofon* illetve *mágneslemez egység* szükséges, amivel rögzíthetjük a programot és amivel beolvashatjuk. A magnetofon csatlakoztatása az egyszerűbb és egyben olcsóbb megoldás, mert közönséges kazettás magnetofonról van szó. A mágneslemezzel mindez gyorsabb és biztonságosabb.
Nagymértékben megkönnyíti munkánkat, ha a rendszerhez egy *nyomtató* is csatlakozik, amivel kiírathatjuk papírra is a programot és az eredményeket.
Az itt felsoroltak csak a legismertebb egységek, eszközök, amik a számítógéphez kapcsolhatók. Ezeken kívül *rajzgépet, mérőeszközt, fényceruzát, másik számítógépet* is illeszthetünk számítógépünkhöz. A lehetőség szinte kimeríthetetlen. A mi számítógépünk, amit alapul veszünk, olyan, hogy a minimális kiépítettségen kívül egy beépített magnetofon tartozik a rendszerhez. Egy ilyen

számítógéppel már elkezdhetjük a tanulást. A rendszer összeállításához és az egységek kezeléséhez a kézikönyvek, használati útmutatók megadják a szükséges információt, de megkönnyíthetjük helyzetünket, ha valakitől még kérünk egy kis segítséget. Szerencsére egyre többen értenek a számítógéphez. Mi okozhat mégis gondot? Nézzük sorba!

Hogyan kell üzembehelyezni a számítógépet?

Az egyes géptípusok között, de még az azonos típusú, de más időpontban készített gépek között is nagyon nagyok lehetnek az eltérések. Kövessük az előírásokat!
Ebben a könyvben csak a HT–1080Z típusú számítógéphez adunk konkrét útmutatást. Más géptípus használatakor mindig ellenőrizzük azt, amit olvastunk. A bekapcsolás után, amikor megjelenik a képernyőn a READY? kiírás, nyomjuk le a NEW LINE billentyűt! Ekkor a READY szó újból megjelenik, de most már kérdőjel nélkül, a képernyő alján. A READY (kész) kiírás jelzi általában, hogy a BASIC *értelmező program,* az un. interpreter felkészült parancsaink, utasításaink fogadására. A parancs, ill. utasítássorozat begépelése végén, a NEW LINE billentyű lenyomásakor, ez a program értelmezi és végrehajtja azt, amit beírtunk. Mindezt természetesen csak akkor, ha a *BASIC* programozási nyelv szabályainak megfelelően írtuk a számítógépbe.

A BASIC programozási nyelv 1964–ben született, és a személyi számítógépek megjelenésével és elterjedésével most éli virágkorát. Ez az anyelv, amit csaknem minden személyi számítógép „ért". Géptípusok, gépcsaládok BASIC nyelve között azonban vannak eltérések. Erre ügyeljünk, mert előfordul, hogy más gépen ugyanaz a szó mást jelent, vagy eltérő szó szerepel megegyező értelemben. Ezért csak olyan program működésében lehetünk biztosak, amit az adott gépre írtak. Kísérletezni persze érdekes és érdemes.
Egy BASIC programról tudnunk kell, hogy milyen gépen futtatható. A *program futásán* annak végrehajtását értjük. Ebben a könyvben HT–1080Z BASIC programokat találunk.

Amennyiben az egyik gépre írt programok változtatás nélkül futtathatók egy másik gépen, akkor azt mondjuk, hogy ezek a számítógépek *kompatibilisek.* Ilyen például a HT–1080Z és a TRS–80.

Hogyan írjuk be a kész programokat?

Ehhez először meg kell ismernünk a billentyűzetet. A billentyűkön betűket, számokat, különböző jeleket látunk. Akkor, ha két *karakter* van egy

billentyűn, és a felső karaktert akarjuk beírni, a SHIFT billentyűt kell előbb lenyomni, és lenyomva kell tartani, amíg a kiválasztott karakter billentyűjét lenyomjuk. Legegyszerűbb természetesen az, ha kipróbáljuk mi jelenik meg a képernyőn, ha egy billentyűt lenyomunk. Azt a jelet, ami azt jelzi, hogy hol jelenik meg a karakter a képernyőn, *kurzornak* nevezzük.

Vannak olyan billentyűk is a billentyűzeten, amelyek valamilyen funkciót látnak el: mozgatják a kurzort, törölnek egy karaktert, vagy a képernyőt, stb.

Fontos szerepe van a NEW LINE (új sor) billentyűnek. Ezzel zárjuk le a parancsokat és a programsorokat. Keressünk ki a könyvből programokat és azokat gépeljük be! A Mintaprogramok című fejezetből például több program közül választhatunk.
Figyelem! Ügyeljünk arra, hogy a ¤ karakter helyett $–t, Å helyett ↑–t, É helyett @ –t gépeljünk! A nyomtató, amin a kiíratások történtek, a svéd abc–t használta. Ezért van ez az eltérés.

A programok beírásával megtanulhatjuk a billentyűzet használatát és egyúttal bizonyos programozási ismeretekre is szert tehetünk. Többek között megismerhetjük az *utasításszavakat,* láthatjuk az utasítások felépítését, esetleg még egy–két utasítás jelentését, hatását is kideríthetjük. Nagyon hasznos, ha kész programokat vizsgálunk.
Azt mindenesetre vegyük észre, hogy a programsorok mindig pozitív egész számmal kezdődnek. Ezek a *programsorok sorszámai*. Egy programsorban több utasítást is írhatunk. Ekkor az utasításokat kettősponttal választjuk el egymástól.
A begépeléskor a programsorokat *le kell zárnunk*. Ez a NEW LINE billentyű lenyomásával történik. (Egyes számítógépeken ezt a funkciót a RETURN, ENTER billentyűk látják el.) A programsor lezárásával a programsort *tárolja* a számítógép.
Mielőtt a program beírását megkezdenénk, adjunk ki egy NEW (új) *parancsot*. Ez a parancs *törli* az előző programot a memóriából.
Amennyiben kész vagyunk a program begépelésével, a RUN (fuss!) parancscsal *indíthatjuk*, vagy a LIST (listázz!) paranccsal kiírathatjuk a képernyőre a programot. Már félkész programot is érdemes kiíratni. Így meggyőződhetünk arról, hogy mit „vett be" a számítógép, nem tévedtünk–e valahol.
A program futása közben általában kér valamilyen adatot a program. Ezeket nekünk kell begépelni. A program számára az így megadott adatokat *bemenő* (input) *adatoknak* nevezzük. Begépelésük után általában le kell nyomni a NEW LINE billentyűt. A beolvasás csak ekkor történik meg, csak így fut tovább a program.

Amit a számítógép ír ki a képernyőre, azok a *kimenő* (output) adatok. Ez már a programban szereplő utasítások végrehajtásának eredménye.

Van, amikor valamilyen *hiba* miatt leáll a program. Ekkor a képernyőn megjelenik az ERROR szó és mellette a hiba jellegére utaló két betűből álló jelzés, amiknek jelentését a „Melléklet"–ben adjuk meg. Az SN például *szintaktikai* hibát jelent, azaz formai hibát. Ilyenkor nem a programnyelv szabályai szerint írtunk be egy utasítást.

Nehezebb az olyan hibát kijavítani, ami formailag helyes, de logikailag nem. Ez a *szemantikai* hiba. A program nem úgy működik, mint ahogy szeretnénk. Ilyenkor is mi vagyunk hibásak. Valamit kihagytunk, vagy valamit hibásan írtunk be, vagy a jónak mondott program hibás, amit begépeltünk. A számítógépre, mint hibaforrásra, csak legvégül gondoljunk. A számítógép hibája általában „durván" szokott jelentkezni, azaz a gép nem csinál semmit.

Hogyan történik a program javítása?

- Akkor, ha begépeléskor, mielőtt még lezárnánk az utasítássort, észrevesszük a hibát, a ← billentyűvel törölhetjük a begépelt karaktereket.
- Felülírhatjuk a programsort. A hibás sor helyett írjuk le újból, most már helyesen az egész programsort.
- Egy programsor törlésénél is azt használjuk ki, hogy a számítógép mindig a legutóljára begépelt programsort tárolja. Akkor, ha csak a sorszámot gépeljük be, törlődik az ilyen sorszámú programsor.
- A programsorok közé utólag is beírhatunk programsorokat, ha van köztük hely, azaz nem egyesével követik egymást a sorszámok. Programozzunk „szellősen", hogy ha kihagyunk valamit, legyen hová beírni. A könyvben ezért sorszámoztuk tizesével a sorokat.
- A programsoron belül az EDIT paranccsal javíthatunk. Használatát a „Melléklet"–ben megtaláljuk, a parancsok ismertetésénél.

Hogyan tároljuk a programot?

Akkor, ha a begépelt, félig kész, vagy jó programot szeretnénk megőrizni, hogy ne kelljen újból begépelni, a programot rögzítsük mágnesszalagra. A magnetofont állítsuk felvételre és adjuk ki a CSAVE"N" parancsot. Az N lesz a program neve. a HT–1080Z gépen a program neve mindig egyetlen karakter. Ha több karakterből álló nevet adunk a programnak, a gép nem ad hibajelentést, de csak az első karaktert írja a szalagra.

Azt, hogy sikerült–e a felvétel, úgy ellenőrizzük, hogy a kivitt programot a

CLOAD? (a kérdőjelről ne feledkezzünk meg!) pranccsal összehasonlíttatjuk a memóriában lévő programmal. Amennyiben a számítógépben lévő program megegyezik a szalagon rögzített programmal, az ellenőrzés végén a READY kiírás jelenik meg. Ellenőrzéskor a képernyő felső sarkában megjelenik két csillag és a jobb oldali időnként felvillan. Hiba esetén a BAD szót írja ki a számítógép. Az összehasonlításhoz természetesen a szalagot a program elejére kell állítani. Ebben segít az F1 gomb és a fordulatszámláló.

Hogyan olvassuk be a programot?

A szalagot állítsuk oda, ahol a program kezdődik, és kapcsoljuk lejátszásra a magnetofont. Ezek után adjuk ki a CLOAD parancsot. Helyes beolvasás esetén ugyanaz történik, mint amit a felvétel ellenőrzésénél írtunk. A beolvasás befejezését a READY kiírással jelzi a számítógép.

Mielőtt tovább folytatnánk az ismerkedést, érdemes áttanulmányozni a könyv „Mellékletét". Sok hasznos információhoz juthatunk mindazzal kapcsolatban, amiről ebben a fejezetben szó volt, de segíthet a továbbiakban is.

2. ELSŐ PROGRAMOK

Az INPUT, LET, PRINT *utasítások és a változók típusmeghatározása, aritmetikai műveletek*

Miután az előző fejezetben leírtakat nagyvonalakban megértettük és sikerült valamelyik programot lefuttatnunk, biztosan szert tettünk annyi ismeretre, hogy tudjuk a számítógépet kezelni. Ezek után elkezdhetjük a programozás tanulását.

Gépeljük be a következő parancsot!

PRINT 25 / 5–2.5 + 3✱ 12.5E3

A képernyőn ezt látjuk, miután a parancsot végrehajtotta a számítógép:

```
PRINT 25/5-2.5+3*12.5E3
 37502.5
READY
>_
```

A PRINT szó a gép számára egy *parancs*: Nyomtasd ki! A parancsban azt adtuk meg, hogy nyomtassa ki a kifejezés értékét. Ez matematikai jelöléssel a következő:

$$\frac{25}{5} - 2{,}5 + 3 \cdot 12{,}5 \cdot 10^3$$

Ebből látszik, hogy a BASIC nyelvben az alapműveletek jelei: + – ✱ és / . A tizedesvessző helyett tizedespontot használunk.

Háromféle alakban adhatjuk meg a számokat:

egész:	5	–143	4731Ø
tizedespontos:	2.5	–47.56	3.14159
lebegőpontos:	12.5E3	–45E2	1E–3

A lebegőpontos számok a következő számoknak felelnek meg:

$12{,}5 \cdot 10^3$ $\quad -45 \cdot 10^2$ $\quad 10^{-3}$

● Gyakorlásul számítsuk ki a megadott PRINT parancsok begépelésével a következő aritmetikai kifejezések értékét! A PRINT szó helyett ?–t is írhatunk.

$$3{,}5 \cdot 12 + \frac{6}{2}$$

```
PRINT 3.5*12+6/2
 45
READY
>_
```

$$\frac{3}{2} - 12{,}5 \cdot 2{,}3 \cdot 102$$

```
PRINT 3/2-12.5*2.3E2
-2873.5
READY
>_
```

$$2 \cdot 3{,}45 \cdot 3{,}14159$$

```
PRINT 2*3.45*3.14159
 21.677
READY
>_
```

$$\frac{32{,}5 \cdot 46 + 78{,}9}{47{,}3 \cdot 10^4 - 1483{,}5}$$

```
PRINT (32.5*46+78.9)/(47.3E4-1483.5)
 3.33795E-03
READY
>_
```

Ebben a feladatban csak számoltattuk a számítógépet. Egy feladat megoldásához egy utasításra volt szükség, amit parancsként adtunk meg. Most az un. kalkulátor módban végeztettük a számításokat.

Összetettebb feladatok megoldása általában csak több utasítás segítségével történhet. Ekkor már érdemes *programot* írnunk.

Ahhoz, hogy programot írjunk, nem csak a megoldás *algoritmusát* kell ismernünk, hanem tudnunk kell azt is, hogy az utasítások közül melyiket használjuk. Ezért kell ismernünk az utasításokat és tisztában kell lennünk azok jelentésével.

● Számítsuk ki program segítségével egy adott oldalhosszúságú téglalap kerületét és területét!

Mielőtt begépelnénk a következő programot, a NEW paranccsal töröljük ki a memóriát!

```
READY
>10 INPUT A,B
>20 LET K=2*A+2*B
>30 LET T=A*B
>40 PRINT K,T
>_
```

A RUN paranccsal indítsuk el a programot!

Kövessük végig, hogy mit csinál a program és, hogy nekünk mit kell tennünk, hogy a program lefusson!

A program végrehajtása a legkisebb sorszámú utasítással kezdődik. Az INPUT *utasítással kéri* az A és B változó értékeit, amit egy ? kiírással jelez. Az *adatok bevitele* a következőképpen történhet.

— minden adat begépelése után a NEW LINE billentyű lenyomásával jelezzük, hogy kész vagyunk az adat beírásával, vagy
— az adatokat vesszővel elválasztva beírjuk, és csak az utolsó adat beírása után nyomjuk le a NEW LINE billentyűt.

Az első esetben a rendszer két ? kiírással jelzi, ha még nem adtuk meg az INPUT utasításban szereplő összes változónak az értékét.

A hibásan beütött karaktereket addig törölhetjük a ← billentyű lenyomásával, amíg a NEW LINE billentyűt le nem nyomjuk.

```
RUN
? 4.5
?? 3
 15                13.5
READY
>RUN
? 12.5,5.6
 36.2              70
READY
>_
```


Az INPUT utasítás, mellyel a billentyűzetről olvashatunk be adatokat, általános alakja a következő:

INPUT változólista

jelentése: „Olvasd be a billentyűzetről a változólistában szereplő változók értékeit!"

A *változókat* egy vagy két betűvel, vagy egy betűvel és egy számmal jelölhetjük. Lehetséges változók: A, XY, N, Q3, K2, stb.

Az INPUT utasítás *változólistájában* azokat a változókat soroljuk fel — vesszővel elválasztva —, amelyeknek értékét a program írásakor még nem tudjuk, csak a futtatáskor.

Akkor, ha az adatot nem megfelelő formában írtuk be, a rendszer REDO kiírással jelzi a hibát és újból kéri az adatot.

A következő utasítás a 20–as sorszámú LET K=2*A+2*B utasítás lesz, melylyel *kiszámíthatjuk* az A és B változók értékeivel a téglalap kerületét és ezt az értéket veszi fel a K változó. *Tároljuk* az eredményt.

Az *értékadó utasítás* általános alakja:

LET változó = aritmetikai kifejezés

jelentése: „Az egyenlőség bal oldalán álló változó vegye fel az egyenlőség jobb oldalán álló aritmetikai kifejezés kiszámított értékét!"

Az utasítás végrehajtásakor két dolog történik: az aritmetikai kifejezés értékét kiszámítja a rendszer és az értékét az adott változóban tárolja. A változó korábbi értéke ezzel elvész.

Ez történik akkor is, amikor az INPUT utasítással adunk értékeket a változónak. Amennyiben egy változónak LET vagy INPUT utasítással nem adunk értéket és úgy hivatkozunk rá, az értéke nulla lesz.

Az *aritmetikai kifejezések* számokkal és változókkal jelölnek ki müveleteket.

Az aritmetikai műveletek jelei:

1. ↑ a hatványozás jele, amit a ↑ billentyűvel írhatunk be.
2. * a szorzás, / az osztás (ferde vonal)
3. + az összeadás, — a kivonás jele.

A sorszámok a *műveletvégzés sorrendjét* jelentik.

A hatványozás a legmagasabb rendű művelet. Pozitív alap, tizedeskitevőjű hatványozása is megengedett.

Az azonos rendű műveletek végrehajtása balról jobbra történik. A műveletek végrehajtásának ezt a sorrendjét *zárójelekkel* módosíthatjuk. Ennek megfelelően a kerületet a LET K = 2*(A+B) utasítással is kiszámíthatjuk.

Tovább követve a programot a 30–as sorban a téglalap területét számítjuk ki, és az utolsó sorral kiíratjuk a K és a T változók értékeit.

A *kiíró utasítás* alakja:

PRINT kiíratási lista

jelentése: „Írd ki a kiíratási lista elemeinek az értékét!"

A *kiíratási listában* számok, változók és aritmetikai kifejezések szerepelhetnek. Ezt figyelembevéve az előző programot rövidebben is megírhatjuk:

```
10 INPUT A,B
20 PRINT 2*(A+B),A*B
```

Most nem tároltuk a terület és kerület értékét. Kiszámítottuk és kiírattuk. Az INPUT, LET és PRINT utasítások további lehetőségeit mutatjuk be a következő mintaprogrammal.

● Számítsuk ki az R sugarú kör kerületét és területét!

```
>LIST
10 PRINT "A KOR KERULETE ES TERULETE"
20 INPUT "SUGAR";R
30 PI=3.14159
40 K=2*R*PI
50 T=R↑2*PI
60 PRINT "KERULETE";K,"TERULETE";T
READY
>RUN
A KOR KERULETE ES TERULETE
SUGAR? 2.5
KERULETE 15.708                    TERULETE 19.6349
READY
>_
```

– A LET utasításszót elhagytuk. Ezt a HT–1080Z megengedi.

– A PRINT utasítással a változók és kifejezések értékén kívül szöveget is kiírathatunk. A kiírandó *karakterlánc*ot, amit *string*nek nevezünk, idézőjel közé kell tenni. Egy karaktersorozatban mindazok a karakterek szerepelhetnek, amelyek a billentyűzeten előállíthatók, az idézőjelet kivéve. Helyette aposztrófot (felsővesszőt) használjunk.

– Egy karakterláncot az INPUT utasítással is kiírathatunk. Ezt mindig az INPUT utasítás-szó után kell írni, ami után pontosvesszőt kell tenni.

Az INPUT utasítás alakja ekkor:

INPUT "karakterlánc"; változólista

– A PRINT utasításban a *listaelemeket* – szám, karakterlánc, változó, aritmetikai kifejezés – vesszővel, vagy pontosvesszővel választjuk el. *Pontosvessző* esetén az előző kiírást követően történik a megjelenítés, *vessző* esetén a következő *zóna* elején. A képernyőn 16 krakterpozició egy zóna, ezért a kiírás a 0., a 16., a 32. vagy a 48. krakterpozicióban történik.

– Amennyiben az értelmező program számára egyértelmű, hogy egy listaelem befejeződött, nem szükséges az elválasztójeleket kitenni. Ezért helyes a következő utasítás is

60 PRINT "KERULETE"K"TERULETE"T

– A PRINT utasítás befejezésekor soremelés történik. Ez azt jelenti, hogy új sor elejére áll a kurzor. A soremelést letílthatjuk, ha az utolsó listaelem után vesszőt, vagy pontosvesszőt írunk.

O Milyen lesz a kiíratás, ha csak vesszőt és milyen lesz, ha csak pontosvesszőt használunk elválasztójelnek?

O Irassuk ki egymás alá a két végeredményt!

● Határozzuk meg egy kettes számrendszerben megadott négyjegyű szám értékét tizes számrendszerben!

A következő azonossággal – melynek helyességét könnyen beláthatjuk a műveletek elvégzésével – az átalakítás kétféle módját is megadjuk:

```
>LIST
10 INPUT "BINARIS SZAMJEGYEK";A,B,C,D
20 PRINT "DECIMALIS ERTEK"
30 PRINT ((2*A+B)*2+C)*2+D
40 PRINT A*2↑3+B*2↑2+C*2+D
READY
>RUN
BINARIS SZAMJEGYEK? 1,0,1,1
DECIMALIS ERTEK
 11
 11
READY
>_
```

Ebben a programban a *változók értéke* csak *egész* lehet. Ezt figyelembevéve egész változókkal is dolgozhatunk.

```
>LIST
10 INPUT "BINARIS SZAMJEGYEK";A%,B%,C%,D%
20 PRINT "DECIMALIS ERTEK"
30 PRINT ((2*A%+B%)*2+C%)*2+D%
40 PRINT A%*2^3+B%*2^2+C%*2+D%
READY
>RUN
BINARIS SZAMJEGYEK? 1,0,1,0
DECIMALIS ERTEK
 10
 10
READY
>_
```

A % jellel jelöljük az egész változókat. Az A és az A% két különböző változónak számít. Van lehetőség arra is, hogy *kétszeres pontosságú változó*kkal számoljunk. Ezt mutatjuk be a kör kerületének és területének kiszámításával. A kétszeres pontosságú változókat # karakterekkel jelöljük: R#, PI#, KE#, TE#

```
>LIST
10 PRINT "A KOR KERULETE ES TERULETE"
20 INPUT "SUGAR";R#
30 PI#=3.1415931
40 K#=2*R#*PI#
50 T#=R#^2*PI#
60 PRINT "KERULETE";K#
70 PRINT "TERULETE";T#
READY
>RUN
A KOR KERULETE ES TERULETE
SUGAR? 2.5
KERULETE 15.7079655
TERULETE 19.6349628671133
READY
>_
```

```
>RUN
A KOR KERULETE ES TERULETE
SUGAR? 0.00000025
KERULETE 1.57079655D-06
TERULETE 1.9634941894758 55D-13
READY
>_
```

A kétszeres pontosságú eredmények kiírásakor a lebegőpontos alakban az exponens előtt az E betű helyett D betű áll. Pl. 42.834732D3, 37543102D–5.

Az eredmények *pontosságá*nak megitélésében legyűnk körültekintőek, mert nem minden esetben tekinthető minden számjegy pontosnak, amit eredményül kiír a számítógép. A pontosságot a bemenő adatok pontosságán kívül a program által végzett számítások is meghatározzák.

A *kétszeres pontosság* és az *egész változók kijelölését* egyszerűbben is elvégezhetjük a DEFDBL, valamint a DEFINT utasításokkal. Az egyszeres pontosságú változókkal megírt programunk elejére kell írni ezeket az utasításokat, melyekkel megadjuk, hogy milyen betűvel kezdődő változók legyenek kétszeres pontosságúak, vagy egészek.

```
5 DEFDBL K,P–R,T
```

A tízes számrendszerbe átíró programunkat a

```
5 DEFINT A–D
```

utasítás beírásával lehet egész változókra átírni.

Amennyiben nem írunk % vagy # jelet a változó mellé, és nem terjed ki a hatása a változóra a DEFINT vagy DEFDBL utasításoknak, akkor a változó egyszeres pontosságú lesz. A pontosság módosítására azonban szükség lehet, ezért a ! karakterrel, vagy a DEFSNG utasítással *egyszeres pontosságú változó*kat is kijelölhetünk, vagy módosíthatjuk a pontosságot.

A DEFINT, DEFSNG, DEFDBL utasításokat a %, ! és # jelekkel módosíthatjuk.

Hasonlóan definiálhatjuk vagy jelölhetjük a *karakterlánc* (string) *változókat* is, a DEFSTR utasítással vagy a $ karakterrel.

Például: DEFSTR A–B, Z utasítással az A, AB, B1, ZX, Z változók karakterláncok tárolására lesznek alkalmasak.

A $ karakter alkalmazásával is megoldhatjuk ezt: A$, AB$, B1$, ZX$, Z$ jelöléssel.

Egy karakterlánc maximálisan 255 karakterből állhat.

Hosszabb karakterláncok esetén a CLEAR utasítással gondoskodnunk kell a tárolóhelyről. A CLEAR 500 utasítással 500 memóriarekeszt foglalunk le. (A könyvben található listákban a $ karakter helyett a ¤ karakter szerepel.)

3. KIÍRATÁSI FORMÁTUMOK

A PRINT @, PRINTUSING és a CLS utasítások. TAB kiíratási listaelem

A programok használatát, kezelését megkönnyíthetjük, ha a magyarázó, tájékoztató szövegeket áttekinthető formában iratjuk ki a képernyőre és az eredményeket rendezett alakban jelenítjük meg. Ezeknek a követelményeknek megfelelően írtuk át a téglalap kerületét és területét kiszámító programot.

```
10 CLS
20 PRINT@70,"A TEGLALAP OLDALAI"
30 PRINT TAB(20);"A";:INPUT A
40 PRINT  TAB(20);:INPUT "B";B
50 PRINT TAB(12);:INPUT "MERTEKEGYSEG";M$
60 KE=2*A+2*B:TE=A*B
70 PRINT  @320,"A TEGLALAP"
80 PRINT TAB(10);"KERULETE";:PRINT USING "########.##";KE;
:PRINT " ";M$
90 PRINT TAB(10);"TERULETE";:PRINT USING "########.###";TE;
:PRINT  " ";M$;"A2"


      A TEGLALAP OLDALAI
                  A? 2.5
                  B? 5
            MERTEKEGYSEG? CM
A TEGLALAP
          KERULETE      15.00 CM
          TERULETE      12.500 CMA2
READY
>_
```

A *képernyőre* 16 sort írhatunk, és minden sorba 64 karaktert, ami 64 karakterpozíciónak felel meg. A sorokat 0–tól 15–ig sorszámozzuk, a karakterpozíciókat 0–tól 63–ig.

A programban szereplő utasítások értelmezése:

10–es: CLS utasítással *töröljük a képernyőt.*

20–as: PRINT @ 70, utasítással a *képernyő* 70. *karakterpozíciójába állítjuk a kurzort.* A @ (egységár vagy „kukac") karakter helyett É–t írt ki a nyomtató. Ez a képernyő 1. sorának 5. karakterpozíciója lesz. A karakterpozíciók számlálása a 0. sor 0. karakterpozíciótól sorfolytonosan történik.

A pozíciót megadó szám, vagy kifejezés után ,–t kell tennünk, és utána írhatjuk a kiírandó listaelemeket.

30–as: Az előző utasítás végrehajtása után a kurzor a 2. sor elejére került. A TAB(20) kiíratási listaelemmel a kurzort a *sor* 20. *karakterpozíciójába állítjuk.* Itt írja ki, hogy a téglalap melyik oldalát kéri. A PRINT utasítás végén ; van, ezért A értékét ugyanebben a sorban kéri.

40–es: Hasonló az előzőhöz, csak nem a PRINT, hanem az INPUT *utasítással* íratjuk ki, *hogy milyen adatot kér a program.*

50–es: Az INPUT utasítással az M$ *karakterlánc változó* értékét kérjük. Ez egy tetszőleges karaktersorozat lehet. Programunkban: METER, DECIMETER, stb.

60–as: A téglalap kerületének és területének mértékszámát meghatározzuk.

70–es: A képernyő 320. karakterpozícióján kezdjük a kiíratást.

80–as: A PRINTUSING utasítással a KE változót *két tizedesjegy pontossággal íratjuk ki.* Az egész számjegyek kiírására 7 helyet adtunk meg. Ezt a # jelek számával és a pont helyzetével jelölhetjük ki.
A PRINTUSING utasítással vagy csak számokat, változók, kifejezések, vagy csak stringek értékeit írathatjuk ki. *Elválasztójel* lehet a vessző és a pontosvessző, de mindig folyamatosan írja ki a listaelemek értékét.
A formátum kijelölése után pontosvesszőt kell írnunk!
Ujabb PRINT utasítással kiíratjuk a mértékegységet.

90–es: A TE változó kiírása 3 tizedesjegy pontossággal történik.
Hasonlóan, mint az előző utasításban, legvégül kiíratjuk a mértékegységet.

A PRINTUSING utasítást a következő programmal vizsgálhatjuk:

```
10 INPUT "A SZAM";A
20 INPUT "FORMATUM";F$
30 PRINTUSING F$;A
40 GOTO 20
```

A 10–es utasítássorban az F$ karakteres változó értékének megadjuk a kiíratási formátumot (pl. # # # .# #), akkor a kiírás ennek megfelelően történik. A GOTO 10 utasítással elértük azt, hogy a különböző formátumokat folyamatosan kérje a program, mivel a kiíratás után „visszaugrik" 10–es utasítássorra.

A *programot leállítani* a BREAK billentyű lenyomásával lehet.
Próbáljuk ki a következő *formátumok*at!

`####.##` – két tizedes pontosságu kiírás

`**###.##` – *-okat ír a szám elé, kitölti a kiíratási mezőt

`$$####.#` – $ jelet ír a szám elé

`**$####.#` – *-okat és $ jelet ír a szám elé

```
>RUN
A SZAM? 123.4567
FORMATUM? ####.##
 123.46
FORMATUM? **####.##
**123.46
FORMATUM? ¤¤#####.#
  ¤123.5
FORMATUM? **¤####.#
***¤123.5
FORMATUM? ##
%123
FORMATUM? _
```

Pozitív szám esetén:

`+###` – + jel lesz a szám előtt

`###.#+` – + jel lesz a szám után

`-###.##` – – jelet ír a kiíratási mező elé

`###.##-` – csak a számot írja ki

```
>RUN
A SZAM? 123.456
FORMATUM? +###
+123
FORMATUM? ###.#+
123.5+
FORMATUM? -###.##
-123.46
FORMATUM? ###.##-
123.46
FORMATUM? _
```


Negatív szám esetén:

`+###` – a szám elé – jelet ír

`###.#+` – – jelet ír a szám után

`-###.#` – – jelet ír a kiíratási mező elé

`###.#-` – – jelet ír a szám után

```
>RUN
A SZAM? -123.456
FORMATUM? +###
-123
FORMATUM? ###.#+
123.5-
FORMATUM? -###.#
-%-123.5
FORMATUM? -######.#
- -123.5
FORMATUM? ###.#-
123.5-
FORMATUM? _
```

Minden esetben:

- % jelet ír a szám elé, ha kisebb formátumot adunk meg a szám egész részének, mint a szám,
- az utolsó számjegy kerekített érték lesz.

A PRINTUSING utasítással meghatározhatjuk azt is, hogy egy karakterláncból hány karakter íródjon ki.

```
10 INPUT "KARAKTERLANC";A¤
20 INPUT "FORMATUM";F¤
30 PRINT USING F¤;A¤
40 GOTO 20
```

A formátumok és hatásuk:

!	– a karakterlánc első karaktere
%%	– a karakterlánc első két karaktere
% %	– a karakterlánc első három karaktere íródik ki

```
>RUN
KARAKTERLANC? MISKOLC
FORMATUM? !
M
FORMATUM? %%
MI
FORMATUM? % %
MIS
FORMATUM? %
MISKOLC
FORMATUM? _
```

A % közé betűköz (SPACE) karaktereket kell írni. A betűköz karakterek számának a növelésével irathatunk ki több karaktert a karakterláncból.
Az előző programban A$ és F$–al jelöltük a karakterlánc változó. A korábbi programot nem kellett volna átírni, ha a DEFSTR A, F utasítást használjuk. Ezzel az utasítással az A–t és F–t karakterláncváltozónak jelöljük ki.

4. ÁBRÁK A KÉPERNYŐN

REM, READ, DATA, RESTORE, GOTO, ON ERROR utasítások

A PRINT @ utasítással és a TAB kiíratási listaelemmel karakterekből ábrákat alakíthatunk ki a képernyőn. A programokban új utasításokkal is találkozhatunk.
Magyarázó, tájékoztató szöveget a REM *utasítással írhatunk a programba.* A REM szót követő szöveget nem értelmezi a BASIC rendszer.
A következő programmal *–okat jeleníthetünk meg a képernyőn.

```
10 REM CSILLAGOK - PRINT@ VIZSGALATA
20 CLS:REM KEPERNYO TORLESE
30 PRINT@768,:INPUT "SOR(0-15)";S:INPUT "KARAKTERPOZICIO
(0-63)";P
40 PRINT@64*S+P,"*";
50 GOTO 30
```

30–as sor: Az adatokat a 13. sorban kérjük, így az adatbevitellel a képernyőn lévő karakterek nem mozdulnak el. Nem „csúszik" egy sorral feljebb a képernyő. Ezt a PRINT @ 768, utasítással értük el.
Az S változó értéke annak a *sor*nak a *sorszámát* jelenti, ahol a *–ot meg akarjuk jeleníteni. P a soron belül a *karakterpoziciót* határozza meg. A változók lehetséges értékeit jelezzük a programban.

40–es sor: A * jel megjelenítése.
Akkor, ha a 64 * S + P kifejezés értéke Ø–nál kisebb, vagy 1023–nál nagyobb, a rendszer *hibát jelez:* FC ERROR IN 4Ø . FC a hiba kódja. A kódtáblázatban megnézhetjük, hogy milyen jellegű hiba okozta a program leállását. A hibajelzésből az is kiderül, hogy melyik sorban történt a hiba. Itt a 40–esben.

O Ábrázoljunk valamilyen alakzatot és jegyezzük fel a sorok és a karakterpoziciók sorszámát! Később ezt felhasználjuk.

O Értelmezzük a következő programot! Fejtsük meg, mit csinál!

```
10 I=0
20 CLS
30 PRINT@I,"*";
40 I=I+1
50 GOTO 20
```

O Irassunk a képernyő minden poziciójába # jelet!

● A következő programunk egy ábrát rajzol a képernyőre.

```
10 REM ABRA KARAKTEREKBOL - READ/DATA UTASITAS
20 CLS
40 ON ERROR GOTO 80
50 READ P
60 PRINT@P,"*";
70 GOTO 50
80 PRINT@836,"NINCS TOBB ADAT":END
90 DATA 153,154,155,156,157,158,159,217,219
100 DATA 221,223,281,284,287,345,346,347,348,349,350,351
110 DATA 476
120 DATA 472,480
130 DATA 470,482
140 DATA 412,474,478,540,604,667,669,730,734
150 DATA 793,799
```

```
       *******
       * * * *
       *  *  *
       *******
          *
    * * * * * * *
          *
          *
         * *
        *   *
       *     *
 NINCS TOBB ADAT
READY
>_
```

A DATA (adatok) utasításban megadjuk a képernyő karakterpozicióit, ahol a csillagok megjelennek. A *DATA adatlista elemeinek a beolvasása a* READ *(olvass!) utasítással történik.*
A GOTO *(menj!) utasítás* hatására folyamatosan történik az adatok beolvasása és megjelenítése mindaddig, ameddig az adatlista végére nem érünk. Akkor, ha a beolvasással az adatlista végére érünk a READ utasítással, és nem tud beolvasni adatot, a program hibát jelez.

Az ON ERROR GOTO (ha hiba ugorj!) *utasítás*sal elértük azt, hogy ne hibajelzéssel fejeződjön be a programunk. Ezt az utasítást mindig az elé az utasítás elé kell írnunk, ahol a hiba bekövetkezését várjuk.

O Az ON ERROR GOTO utasítás törlésével milyen hibát jelez a BASIC rendszer?

● Jelenítsük meg többször az ábrát! Irjuk a programba a következő utasítássorokat:

```
30 RESTORE
50 ON ERROR GOTO 20
```

A RESTORE *utasítás* — READ *utasítás a program legelső* DATA *utasítás elemét olvassa be.* Feladatunkban erre az újbóli megjelenítés miatt van szükség. Az ON ERROR GOTO utasítással a program elejére ugrunk: töröljük a képernyőt és az adatok beolvasását kezdjük elölről.
Futtassuk le a programot és azt tapasztaljuk, hogy az ON ERROR GOTO egyszer hatásos, de másodszor már nem. Az ábra csak kétszer jelenik meg és hibával fejeződik be a programunk. A hibát a RESUME (reszjum) utasítással kell „lekezelni". A RESUME utasítás végrehajtása után újból lesz hibafigyelés. Hatására azzal az utasítássorral folytatódik a program, ahol a hiba történt, és ezért az ábra megjelenítése folyamatos lesz. RESUME 20 utasítás hatására a kép törlődik a megjelenítés előtt. A RESUME utasítást írjuk a 80—as sorba, amelyikre ugrunk az ON ERROR GOTO utasítással.
A hibafigyelést az ON ERROR GOTO 0 utasítással szüntethetjük meg.

5. LOGIKAI MŰVELETEK

Az IF–THEN–ELSE *utasítás*

A számítógépet a zsebszámológéptől elsősorban az különbözteti meg, hogy a számítógép *logikai műveletek* végrehajtására is képes. Egy *relációról* el tudja dönteni, hogy az IGAZ, vagy HAMIS. Lássunk erre egy példát:

```
10 REM AZ IF-THEN-ELSE UTASITAS
20 CLS:INPUT "ADJ MEG EGY SZAMOT";X
30 PRINT" AZ AZ ALLITAS,HOGY A BEIRT SZAM NULLANAL KISEBB :";
40 IF X<0 THEN PRINT"IGAZ" ELSE PRINT" HAMIS
```

```
ADJ MEG EGY SZAMOT? 12
 AZ AZ ALLITAS,HOGY A BEIRT SZAM NULLANAL KISEBB : HAMIS
READY
>_
```

```
ADJ MEG EGY SZAMOT? -23.4
 AZ AZ ALLITAS,HOGY A BEIRT SZAM NULLANAL KISEBB :IGAZ
READY
>_
```

Az IF (ha) szó után szereplő X < ∅ reláció X értékétől függően lehet igaz, vagy hamis: és ennek megfelelően két különböző műveletsort végezhet a számítógép. Kiírja az IGAZ és HAMIS szavak közül a megfelelőt.
A számítógép attól lett „okos", hogy a megfelelő műveletet oda írtuk, ahová kell.
A THEN (akkor) szót követően szereplő utasításokat akkor hajtja végre, ha a reláció igaz, az ELSE (egyébként) szó után írt utasításokat pedig akkor, amikor a reláció hamis.

Ha elolvassuk magyarul az utasítást, akkor talán még érthetőbb lesz: *„Ha X kisebb mint ∅, akkor írd ki azt, hogy* IGAZ, *egyébként írd ki azt, hogy* HAMIS!"

Az IF–THE–ELSE utasítás alkalmazásához tudnunk kell a következőket:

– A relációjelek:

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

– A THEN és ELSE szavak után több utasítást is írhatunk.

– Az IF–THEN–ELSE egy utasításon belül is ismétlődhet.

– Akkor, ha a THEN vagy az ELSE szavak után *csak* GOTO utasítás szerepel, elég csak azt a sorszámot megadni, ami a GOTO utasításban szerepelne.

Az előzőek ismeretében és a relációk megfelelő felírásával egy–egy feladatot többféleképpen megoldhatunk.

● Határozzuk meg egy szám abszolútértékét!

```
>LIST
10 REM ABSZOLUTERTEK 1.
20 INPUT  "X=";X
30 PRINT X;"ABSZOLUTERTEKE:";
40 IF X<0 THEN  PRINT  -X  ELSE PRINT X
READY
>RUN
X=? -2.5
-2.5 ABSZOLUTERTEKE: 2.5
READY
>_
```

Amennyiben azt szeretnénk, hogy az X változó értéke a megadott szám abszolútértékével legyen egyenlő, a programunk a következő lehet:

```
>LIST
10 REM ABSZOLUTERTEK 2.
20 INPUT "X=";X:XT=X
30 IF X>0 THEN 40 ELSE X=-X
40 PRINT XT;"ABSZOLUTERTEKE";X
READY
>RUN
X=? 4.56
 4.56 ABSZOLUTÉRTEKE 4.56
READY
>_
```

Az X kezdeti értékét XT változóban tároltuk, mert a végén ki kell íratnunk X kezdő értékét.
A THEN szó után csak a sorszámot adtuk meg, elhagytuk a GOTO szót (30–as sor).
Ugyanezt az eredményt érjük el, ha a relációjelet megváltoztatjuk és átírjuk az utasítást.

30 IF X<0 THEN X = –X

Az ELSE szót itt elhagytuk, mivel ha a reláció hamis, a 40–es utasítássorral folytatódik a program.

● Adott egy két elemből álló számhalmaz. Döntsük el, hogy 3 eleme–e a halmaznak!

```
>LIST
10 INPUT "AZ ELEMEK";A,B
20 IF A=3 THEN 40 ELSE  IF B=3 THEN 40
30 PRINT "3 NEM ELEME A HALMAZNAK":STOP
40 PRINT "3  ELEME A HALMAZNAK"
READY
>RUN
AZ ELEMEK? 2,5
3 NEM ELEME A HALMAZNAK
Break in 30
READY
>RUN
AZ ELEMEK? 3,99
3  ELEME A HALMAZNAK
READY
>_
```


Az IF–THEN–ELSE utasítást helyettesítsük két IF–THEN utasítással.

```
10 INPUT "AZ ELEMEK";A,B
20 IF A=3 THEN 50
30 IF B=3 THEN 50
40 PRINT " 3 NEM ELEME  A HALMAZNAK":STOP
50 PRINT "3 ELEME  A HALMAZNAK"
READY
>RUN
AZ ELEMEK? 4,5
 3 NEM ELEME  A HALMAZNAK
Break in 40
READY
>RUN
AZ ELEMEK? 8,3
3 ELEME  A HALMAZNAK
READY
>_
```

Egy kis matematikai ismerettel egyszerűbben is megoldhatjuk a problémát. A kiírással igy ügyeskedünk.

```
10 INPUT "AZ ELEMEK";A,B
20 IF(A-3)*(B-3)=0 THEN PRINT  "3"; ELSE PRINT "3  NEM";
30 PRINT " ELEME A HALMAZNAK"
```

Logikai műveleteket is végezhetünk a relációkkal.

Logiai műveletek:

NOT	–	tagadás
AND	–	ÉS művelet
OR	–	VAGY művelet

Ezt ismerve az előző feladatot a következőképpen is megoldhatjuk:

```
10  INPUT "AZ ELEMEK";A,B
20 IF A=3 OR B=3  THEN PRINT "3"; ELSE PRINT "3 NEM";
30 PRINT " ELEME A HALMAZNAK"
```

Az AND, OR és NOT logikai műveletek eredményeit adja a következő program. A bemenő adatok legyenek: IGAZ és HAMIS karakterláncok.

```
10 REM LOGIKAI MUVELETEK
20 INPUT "ITELETEK";E1$,E2$
30 PRINT E1$;" VAGY ";E2$;"=";
40 IF E1$="IGAZ" OR E2$="IGAZ" THEN PRINT " IGAZ" ELSE
PRINT " HAMIS"
50 PRINT E1$;" ES ";E2$;"=";
60 IF E1$="IGAZ" AND E2$="IGAZ"  THEN PRINT " IGAZ" ELSE
 PRINT " HAMIS"
70 PRINT "NEM ";E1$;"=";
80 IF NOT E1$="IGAZ" THEN PRINT "IGAZ " ELSE PRINT "HAMIS "
90  PRINT "NEM ";E2$;"=";
100 IF NOT   E2$="HAMIS" THEN PRINT "HAMIS " ELSE PRINT
"IGAZ "


>RUN
ITELETEK? IGAZ
?? HAMIS
IGAZ VAGY HAMIS= IGAZ
IGAZ ES HAMIS= HAMIS
NEM IGAZ=HAMIS
NEM HAMIS=IGAZ
READY
>_
```

A következő programmal bemutatjuk az IF–THEN–ELSE utasítás használatát.

● Készítsünk programot a kétismeretlenes elsőfokú egyenletrendszer megoldásához!

Az egyenletrendszer általános alakja

$$ax + by = c$$
$$dx + ey = f$$

Ekkor a megoldás:

$$x = \frac{ce - bf}{ae - bd}$$

$$y = \frac{af - dc}{ae - bd}$$

Három eset lehetséges:

1. A nevező nem ϕ. Ekkor van megoldás.

2. A nevező ϕ és mindkét számláló ϕ. Ekkor végtelen sok megoldása van az egyenletrendszernek.

3. A nevező ϕ és valamelyik számláló (vagy mindkettő) nem ϕ, akkor nincs megoldása az egyenletrendszernek.

```
10 CLS:PRINT " ELSOFOKU KETISMERETLENES EGYENLETRENDSZER"
20 PRINT "AZ ELSO EGYENLET / A*X+B*Y=C  /PARAMETEREIT KEREM"
30 INPUT "A";A:INPUT "B=";B:INPUT "C=";C
40 PRINT "A MASODIK EGYENLET / D*X+E*Y=F /PARAMETEREIT KEREM"
50 INPUT "D=";D:INPUT"E=";E:INPUT "F=";F
60 Q=A*E-B*D:REM NEVEZO
70 P=C*E-B*F:REM X SZAMLALOJA
80 R=A*F-D*C:REM Y SZAMLALOJA
90 IF Q<>0 THEN PRINT"X=";P/Q:PRINT"Y=";R/Q:END
100 IF PA2+RA2=0 THEN PRINT"VEGTELEN SOK MEGOLDAS VAN":STOP
110 PRINT"NINCS MEGOLDAS"
```

```
 ELSOFOKU KETISMERETLENES EGYENLETRENDSZER
AZ ELSO EGYENLET / A*X+B*Y=C  /PARAMETEREIT KEREM
A? 12.3
B=? -45.6
C=? 123
A MASODIK EGYENLET / D*X+E*Y=F /PARAMETEREIT KEREM
D=? -3
E=? 4.5
F=? -6.78
X=-2.99978
Y=-3.50652
READY
>_
```

```
 ELSOFOKU KETISMERETLENES EGYENLETRENDSZER
AZ ELSO EGYENLET / A*X+B*Y=C  /PARAMETEREIT KEREM
A? 2
B=? 6
C=? -3
A MASODIK EGYENLET / D*X+E*Y=F /PARAMETEREIT KEREM
D=? 4
E=? 12
F=? -6
VEGTELEN SOK MEGOLDAS VAN
Break in 100
READY
>_
```

```
 ELSOFOKU KETISMERETLENES EGYENLETRENDSZER
AZ ELSO EGYENLET / A*X+B*Y=C  /PARAMETEREIT KEREM
A? 2
B=? 5
C=? -12
A MASODIK EGYENLET / D*X+E*Y=F /PARAMETEREIT KEREM
D=? 2
E=? 5
F=? 6
NINCS MEGOLDAS
READY
>_
```


6. PROGRAM VÉGE ÉS ÚJRAINDÍTÁSA

A STOP *és az* END *utasítás. A* CONT *parancs*

Az előző programban a vizsgálatokon kívül, a *program* lehetséges *befejezés*eire hívjuk fel a figyelmet.

A program három különböző helyen fejeződhet be.
A 90–es soron az END *utasítás* hatására, a 100–as soron a STOP utasítás hatására és a 110–es sor befejezése után, mivel nincs több sor a programban. Ekkor a program logikai és fizikai vége megegyezik.

A STOP *utasítás* hatására megáll a program és a képernyőn megjelenik a BREAK IN 100 kiírás, azaz a BASIC rendszer jelzi azt a sort, ahol megállt a program.
Akkor, ha STOP vagy END utasításnál állt le a program,a CONT *parancscsal* továbbindíthatjuk a programot. Ezt megtehetjük akkor is, ha a BREAK billentyű lenyomásával történt programmegszakítás.

7. ARITMETIKAI FÜGGVÉNYEK

A számítógép megkímél a függvénytáblázat használatától is. Az aritmetikai kifejezésekbe beírt *függvények* értékét meghatározza és ezzel további műveleteket tud végezni. A BASIC rendszerünkben használható aritmetikai függvények:

ABS (X) – *abszolútérték függvény*

A függvény értéke: x, ha $x \geq 0$ és
$-x$, ha $x < 0$.

Az abszolútérték függvény nevét három betűvel adjuk meg. A függvény neve után zárójelek közé a *függvény argumentumát* írjuk. Ez lehet egy aritmetikai kifejezés is, melyben függvények is szerepelhetnek.

● Határozzuk meg az intervallum hosszát, ha adottak az intervallum végpontjai!

```
10 REM ABS FUGGVENY
20 CLS:PRINT "INTERVALLUM VEGPONTJAI"
30 INPUT "A";A:INPUT "B";B
40 PRINT "AZ INTERVALLUM HOSSZA";ABS(B-A)

INTERVALLUM VEGPONTJAI
A? 2.56
B? -12.6
AZ INTERVALLUM HOSSZA 15.16
READY
>_
```

SGN (X) – *előjel függvény*

A függvény értéke: 1, ha $x > 0$
0, ha $x = 0$
-1, ha $x < 0$

● Határozzuk meg, hogy egy számsorozatban mennyivel több pozitív szám van mint negatív!

Az 50–es sorban használtuk ki a függvény adta lehetőséget. A változó értéke pozitív szám esetén eggyel nő, negatív szám esetén eggyel csökken.

```
10 REM MENNYIVEL TOBB... -SGN FUGGVENY
20 INPUT "ELEMEK SZAMA";N
30 S=0:PRINT "KEREM AZ ELEMEKET"
40 FOR I=1 TO N
50 INPUT A
60 S=S+SGN(A)
70 NEXT I
80 IF S<0 THEN PRINT ABS(S);"-EL TOBB A NEGATIV SZAM":STOP
90 IF S=0 THEN PRINT "EGYENLO A SZAMUK":STOP
100 PRINT S;"-EL TOBB A POZITIV SZAM"
```

INT (X) – *egészrész függvény*

Matematikai jelölése: $[X]$, és azt a legnagyobb egész számot jelenti, amelyik kisebb, vagy egyenlő x–szel.
A függvényt a következő program segítségével vizsgálhatjuk meg:

```
10 INPUT "ARGUMENTUM ERTEKE";X
20 PRINT "FUGGVENYERTEK";INT(X)
30 GOTO 10

>RUN
ARGUMENTUM ERTEKE? 3.78
FUGGVENYERTEK 3
ARGUMENTUM ERTEKE? -2.7
FUGGVENYERTEK-3
ARGUMENTUM ERTEKE? -5.1
FUGGVENYERTEK-6
ARGUMENTUM ERTEKE? 0.2
FUGGVENYERTEK 0
ARGUMENTUM ERTEKE? _
```

Ilyen programot érdemes mindegyik függvényhez írni.

● Kerekítsünk egy számot adott pontossággal! A pontosságot 10 hatványaival adjuk meg!

```
>LIST
10 REM KEREKITES -INT FUGGVENY
20 INPUT " A SZAM";A
30 INPUT "PONTOSSAG";I
40 PRINT "KEREKITETT ERTEK ";INT(A/10^I+0.5)*10^I
READY
>RUN
 A SZAM? 2.3456
PONTOSSAG? -2
KEREKITETT ERTEK  2.35
READY
>_
```

CINT (X) — *egészrész függvény*

Az INT függvénytől értelmezési tartományban tér el. Az argumentum –32768 és 32767 közötti értékeket vehet fel. Ez az egész változók számábrázolási tartománya.

FIX (X) — *egészérték függvény*

Levágja az X értékének tizedesrészét. Pozitív X esetén azonos az INT függvénnyel.

● Adott két szám. A nagyobb abszolutértékű számot osszuk el a kisebb abszolútértékű számmal maradékosan!

```
10 CLS:PRINT"MARADEKOS OSZTAS"
20 INPUT "A  KET SZAM";A,B
30 IF ABS(A)>ABS(B) THEN 50
40 T=A:A=B:B=T:REM CSERE
50 IF B=0 THEN 80
60 H=FIX(A/B):M=A-H*B
70 PRINT "A HANYADOS";H;TAB(15)"MARADEK";M:END
80 PRINT "NEM OSZTHATO"

MARADEKOS OSZTAS
A  KET SZAM? 34
?? 12
A HANYADOS 2    MARADEK 10
READY
>_
```


H = FIX(A/B) utasítás helyett használhattuk volna a H% = A/B utasítást is, mivel H% egész típusú változó és így nem tudja tárolni az A/B tizedesrészét. Az eddig ismertetett függvények értékei (a CINT–et kivéve) kétszeres pontosságú argumentumok esetén kétszeres pontosságúak lesznek.

SQR(X) — *négyzetgyök függvény*

Az X operandus négyzetgyökét határozza meg egyszeres pontossággal. Az X értéke nem lehet negatív.
A négyzetgyökvonás tört kitevőjű hatványozással is elvégezhető: X↑0.5
Nagyobb gyökkitevő esetén a törtkitevőjű hatványozás, vagy a LOG és az EXP függvények alkalmazásával határozhatjuk meg a gyök értékét:
EXP(0.5 * LOG(X))

● Határozzuk meg kétszeres pontossággal egy szám négyzetgyökét!

Alkalmazzuk bizonyítás nélkül a Newton–féle iterációs eljárást! E szerint a következő sorozat $\sqrt{x}$–hez tart:

$$a_1 = \frac{x}{2}; \qquad a_i = 0{,}5\left(a_{i-1} + \frac{x}{a_{i-1}}\right)$$

A sorozat elemeit kiszámoljuk. A programunk akkor fejeződik be, amikor két egymást követő elem különbségének abszolútértéke kisebb lesz, mint 1E -20.

```
>LIST
10  REM NEWTON FELE ITERACIO
20 DEFDBL A,X
30  INPUT "A SZAM";X
40 A=X/2
50 AT=A
60 A=.5*(A+X/A)
70 IF ABS(A-AT)>1E-20 THEN 50
80 PRINT X;"NEGYZETGYOKE=";A
READY
>RUN
A SZAM? 2
 2 NEGYZETGYOKE= 1.414213562373095
READY
>_
```

LOG(X) – *természetes logaritmus*

A logaritmus alapszáma: e = 2, 71828...
Az argumentum értéke csak pozitív lehet. Egy kifejezés tetszőleges alapú logaritmusának kiszámítása a $\log_a b = \frac{\ln b}{\ln a}$ összefüggés alapján történhet, ahol ln a természetes logaritmust jelenti.

EXP(X) – *exponenciális függvény*

Az e^x függvény értékét adja meg egyszeres pontossággal.

Trigonometrikus függvények:

SIN(X) – sinus függvény
COS(X) – cosinus függvény
TAN(X) – tangens függvény

Az argumentumok tetszőleges pontosságúak lehetnek, de a függvény értékét csak egyszeres pontossággal számolja ki. Az argumentumok értékeit radiánként értelmezi. Fokokat radiánba az $\widehat{\alpha} = \frac{\pi}{180} \cdot \alpha^\circ$ összefüggés alapján számíthatjuk át.

ATN(X) – *ancustangens függvény*

Megadja azt a szöget radiánban, amelyiknek a tangense x.

Egy szög sinusát vagy cosinusát ismerve a következő összefüggéssel számíthatjuk ki a tangensét:

$$\mathrm{tg}x = \frac{\sin x}{\sqrt{1-\sin^2 x}} = \frac{\sqrt{1-\cos^2 x}}{\cos x}$$

A függvény értéke egyszeres pontosságú.

Átalakítást végző függvények:

CDBL(X) – kétszeres pontosságba
CSNG(X) – egyszeres pontosságba

Az argumentumok tetszőleges aritmetikai kifejezések lehetnek. Nagyobb pontosságra történő átalakítás esetén természetesen az értékes számjegyek száma nem változik. Kisebb pontosságú átalakítás esetén kerekítés történik.

● Adott egy háromszög csúcspontjainak koordinátáival. Határozzuk meg a háromszög kerületét, területét, oldalait és szögeit!
Az oldalak kiszámítása után a háromszög egyenlőtlenséget felhasználva megvizsgáljuk, hogy létezik–e ilyen háromszög, majd meghatározzuk a kerületét:

$$K = a + b + c$$

A területet a Heron képlettel számoljuk ki:

$$T = \sqrt{s(s-a)(s-b)(s-c)}, \quad \text{ahol} \quad s = \frac{K}{2}.$$

A cosinus-tétel segítségével megkapjuk az egyik szöget:

$$\cos x = \frac{a^2 + b^2 - c^2}{2ab}$$

A szög meghatározásához az szükséges, hogy a szög cosinusából meghatározzuk a szög tangensét a korábban ismertetett összefüggés felhasználásával. A másik szögét sinustétellel számítjuk ki:

$$\sin y = \frac{c}{b} \sin x.$$

Először a szög sinusából itt is meg kell határoznunk a szög tangensét, mielőtt az ATN függvényt alkalmaznánk.

A harmadik szöget, a háromszög szögeinek összegét ismerve, már kiszámíthatjuk.

```
10 CLS:PRINT "HAROMSZOG OLDALAINAK,SZOGEINEK,KERULETENEK
  ES TERULETENEK        KISZAMITASA"
20 PRINT" KEREM A HAROMSZOG CSUCSÉPONTJAINAK KOORDINATAIT"
30 INPUT "X1";X1:INPUT "Y1";Y1
40 INPUT "X2";X2:INPUT"Y2";Y2
50 INPUT "X3";X3:INPUT "Y3";Y3
60 A=SQR((X3-X2)^2+(Y3-Y2)^2)
70 B=SQR((X3-X1)^2+(Y3-Y1)^2)
80 C=SQR((X2-X1)^2+(Y2-Y1)^2)
90 IF A>C THEN T=A:A=C:C=T
100 IF B>C THEN T=B:B=C:C=T
110 IF A+B<=C OR A+C<=B OR B+C<=A THEN  PRINT"NEM HAROMSZOG"
:END
120 K=A+B+C:S=K/2
130 T=(S*(S-A)*(S-B)*(S-C))^0.5
140 Z=(A^2+B^2-C^2)/2/A/B:IF Z=0 THEN Z=2*ATN(1):GOTO 160
150 Z=SQR(1-Z^2)/Z:Z=ATN(Z):IF Z<0 THEN Z=4*ATN(1)+Z
160 Y=SIN(Z)*B/C:Y=Y/SQR(1-Y^2):Y=ATN(Y)
170 X=4*ATN(1)-Y-Z
180 CLS:PRINT@200," A HAROMSZOG OLDALAI:"
190 PRINTTAB(12);"A=";A:PRINTTAB(12);"B=";B:PRINTTAB(12)
   "C=";C
200 PRINT TAB(10)"KERULETE=";K:PRINTTAB(10)"TERULETE=";T
210 PRINTTAB(7)"SZOGEI:":PRINTTAB(10)"ALFA=";X;" RADIAN= "
   ;X*45/ATN(1);" FOK"
220 PRINTTAB(10)"BETA=";Y;" RADIAN= ";Y*45/ATN(1);" FOK"
230 PRINTTAB(9)"GAMMA=";Z;" RADIAN= ";Z*45/ATN(1);" FOK"
```

```
HAROMSZOG OLDALAINAK,SZOGEINEK,KERULETENEK
  ES TERULETENEK        KISZAMITASA
 KEREM A HAROMSZOG CSUCSÉPONTJAINAK KOORDINATAIT
X1? 0
Y1? 0
X2? 1
Y2? 0
X3? 2
Y3? 1_


          A HAROMSZOG OLDALAI:
             A= 1
             B= 1.41421
             C= 2.23607
           KERULETE= 4.65028
           TERULETE= .5
         SZOGEI:
             ALFA= .32175   RADIAN=   18.4349   FOK
             BETA= .463647   RADIAN=   26.565   FOK
            GAMMA= 2.3562   RADIAN=   135   FOK
READY
>_
```


RND(X) — *véletlenszám*

A rendszerünk RANDOM utasítással állítja elő az első véletlenszámot. Az első ilyen véletlen szám előállítása után az RND függvénnyel kaphatjuk meg a továbbiakat.
Az RND függvény argumentumától függően a függvény értékei a következők lehetnek:

RND(0) — a $[0, 1)$ intervallumból ad meg egy véletlenszámot. Ez a szám egyszeres pontosságú lesz.

RND(X) — az $[1, [x]]$ intervallumból egy egész szám lesz a függvény értéke. Az X 32767-nél nem vehet fel nagyobb értéket, és $X \geq 1$ feltételnek teljesednie kell.

Az argumentumban aritmetikai kifejezések is szerepelhetnek.

● Készítsünk programot, amivel a pénzfeldobást szimulálhatjuk! Határozzuk meg, hogy N dobásból hány százalék lesz „fej"!

Az RND(0) 0 és 1 közé eső véletlenszámokat állít elő. Kivonva 0.5-t a számokból a véletlen számok a $[-0.5, 0.5)$ intervallumba kerülnek, és így várhatóan a számok fele pozitív, a másik fele negatív lesz. Ez történik a pénzfeldobáskor is, ahol vagy „fej" vagy „írás" lehet a dobás eredménye.

```
10 CLS: PRINT " PENZFELDOBAS"
20 RANDOM
30 INPUT "DOBASOK SZAMA";N
40 FOR I=1 TO N
50 VV=RND(0)-0.5:PRINT@78,CHR$(30);I
60 IF VV>0 THEN FE=FE+1
70 NEXT I
80 PRINTN;" DOBASBOL";FE; "FEJ, AZAZ";FE/N*100;"%"
```

```
 PENZFELDOBAS
DOBASOK SZAMA? 100
 100  DOBASBOL 61 FEJ, AZAZ 61 %
READY
>_
```

● „Varázsoljunk" csillagos eget a képernyőre!

A képernyő sorainak sorszámozása 0–tól 15–ig történik. Az RND(16) függvény 1–től 16–ig állít elő számokat, ezért eggyel csökkenteni kell a kapott értéket, hogy valamelyik sorszámot kapjuk. Hasonlóképpen történik a karakterpoziciók előállítása is.
Az INKEY$ függvény egy karaktert olvas be a Q$ stringváltozóba akkor, ha az utasítás végrehajtásáig leütöttük a billentyűt. Erre a sorra azért van szükség, hogy a READY szó, a program befejezését jelölve, ne jelenjen meg a képernyőn, és ne zavarja meg a „csillagos eget".

```
10 REM CSILLAGOK
20 RANDOM
30 INPUT "CSILLAGOK SZAMA";CS:CLS
40 FOR I=1 TO CS
50 S=RND(16)-1:P=RND(64)-1
60 PRINT@64*S+P,"*";
70 NEXT I
80 Q$=INKEY$: IF Q$="" THEN 80
90 CLS
```

A következő „oktató" programban a kérdésekből az RND függvény segítségével választunk ki egyet. Ezzel változatossá tehetjük valamilyen ismeretanyag begyakorlását.

● Készítsünk programot a BASIC utasításszavak magyar jelentésének tanulásához!

```
10 CLS:PRINT"BASIC UTASITASSZAVAK JELENTESE"
20  RANDOM
30 INPUT " A KERDESEK SZAMA";K
40 FOR I=1 TO K
50 RESTORE
60 READ N
70 T=RND(N)
80 FOR J=1 TO T:READ B$,M$:NEXT J
90 PRINT"AZ UTASITASSZO    ";B$
100 INPUT"JELENTESE  ";J$
110 IF J$=M$ THEN PRINT"HELYES":V=V+1 ELSE PRINT "JO VALASZ
  ";M$
120 NEXT I
130 CLS:PRINT" A";K;" VALASZBOL";V;"VOLT HELYES"
140 REM ADATOK SZAMA
150 DATA 11
160 REM UTASITASSZAVAK ES JELENTESUK
170 DATA END,VEGE,DATA,ADATO,IF,HA,LET,LEGYEN
180 DATALIST,LISTAZZ,PRINT,NYOMTASS,READ,OLVASS,RUN,
FUSS,STEP,LEPES,STOP,ALLJ,THEN,AKKOR
```



```
 A KERDESEK SZAMA? 10
AZ UTASITASSZO    END
JELENTESE  ? VEGE
HELYES
AZ UTASITASSZO    READ
JELENTESE  ? OLVASS
HELYES
AZ UTASITASSZO    LIST
JELENTESE  ? LISTAZZ
HELYES
AZ UTASITASSZO    DATA
JELENTESE  ? ADAT
JO VALASZ
  ADATOK
AZ UTASITASSZO    DATA
JELENTESE  ? _
```

O Készítsünk programot, amivel az összeadást lehet gyakorolni!

O Egészítsük ki a következő játékprogramot! Értékelje a játékos ügyességét!

```
10 CLS:PRINT "FELKARU RABLO":PRINT"BARMELYIK
  BILLENTYUT LENYOMHATOD A JELEK KIVALASZTASAKOR"
20 B$(1)="@":B$(2)="#":B$(3)="%":B$(4)="&"
30 INPUT "HANYSZOR JATSZOL";M
40 RANDOM:CLS
50  FOR I=1 TO M:PRINT@65,I;". JATEK"
60  FOR J=1 TO 4:A$(J)=B$(RND(4)):NEXT J
70  PRINT@650,A$(1);TAB(24);A$(2);TAB(38);A$(3);
  TAB(52);A$(4);
80 Q$=INKEY$:IF Q$="" THEN 60
90 FOR T=1 TO 500:NEXT T
1000 NEXT I
```

● Vizsgáljuk meg a következő program segítségével a véletlenszám generátort! Mennyire egyenletes az eloszlás?

```
10 CLS:PRINT" AZ RND ELOSZLASFUGGVENYE"
20 DIM A(100)
30 INPUT "VELETLEN SZAMOK SZAMA";N
40 PRINT"1 ES 100 KOZOTT";N;" DB VELETLEN EGESZ
  SZAMOT ALLITOK ELO"
50 PRINT"VEGUL A KEPERNYON MEGJELENIK,HOGY
  MELYIKBOL MENNYIT"
60 PRINT" ITT TARTOK"
70 RANDOM
80 FOR I=1 TO N:X=RND(100):A(X)=A(X)+1:PRINT@320,I;:NEXT I
90 CLS
```

```
100 FOR I=1 TO 8:FOR J=1 TO 16
110 K=(I-1)*16+J
120 PRINTé(J-1)*64+(I-1)*8,K;"/";A(K);
130 IF K=100 THEN 150
140 NEXT J:NEXTI
150 Q$=INKEY$:IF Q$="" THEN 150
160 CLS
```

```
 1 / 12   17 / 6   33 / 8   49 / 8   65 / 13  81 / 12  97 / 11
 2 / 10   18 / 10  34 / 12  50 / 7   66 / 13  82 / 10  98 / 8
 3 / 9    19 / 12  35 / 11  51 / 10  67 / 8   83 / 3   99 / 8
 4 / 12   20 / 11  36 / 12  52 / 17  68 / 11  84 / 9   100 / 12
 5 / 7    21 / 13  37 / 7   53 / 7   69 / 10  85 / 12
 6 / 6    22 / 11  38 / 14  54 / 7   70 / 7   86 / 9
 7 / 8    23 / 12  39 / 13  55 / 13  71 / 9   87 / 4
 8 / 10   24 / 8   40 / 10  56 / 10  72 / 6   88 / 10
 9 / 15   25 / 12  41 / 9   57 / 10  73 / 9   89 / 14
10 / 7    26 / 9   42 / 10  58 / 15  74 / 12  90 / 9
11 / 10   27 / 7   43 / 5   59 / 18  75 / 6   91 / 11
12 / 5    28 / 9   44 / 9   60 / 8   76 / 10  92 / 12
13 / 14   29 / 11  45 / 9   61 / 13  77 / 11  93 / 6
14 / 14   30 / 11  46 / 18  62 / 10  78 / 13  94 / 15
15 / 7    31 / 8   47 / 12  63 / 5   79 / 7   95 / 8
16 / 12   32 / 6   48 / 8   64 / 11  80 / 12  96 / 10
```


8. TÖBBIRÁNYÚ ELÁGAZÁS

Az ON–GOTO utasítás

Az IF–THEN–ELSE utasítással két irányban történhet az elágaztatás. Két különböző sorszámú utasítással folytatódhat a program, ill. két különböző utasítássorozatot hajthatunk végre.
A következő feladatot csak *több* IF *utasítás*sal oldhatjuk meg, mivel kettőnél több a lehetséges esetek száma.

● Egy adott számról állapítsuk meg, hogy az nulla, negatív vagy pozitív!

```
10 REM TOBBIRANYU ELAGAZAS-1
20 CLS:INPUT " A SZAM";X
30 IF X<0 THEN PRINT "NEGATIV":STOP
40 IF X=0 THEN PRINT "NULLA":STOP
50 PRINT "POZITIV"

 A SZAM? -23
NEGATIV
Break in 30
READY
>_
```

Oldjuk meg a feladatot az ON–GOTO (ha – ugorj!) *utasítás*sal is!

```
10 REM TOBBIRANYU ELAGAZAS-2
20 CLS:INPUT " A SZAM";X
30 ON SGN(X)+2 GOTO 70,60,50
40 END
50 PRINT "POZITIV":STOP
60 PRINT "NULLA":STOP
70 PRINT "NEGATIV"

 A SZAM? 34.5
POZITIV
Break in 50
READY
>_
```

Az ON–GOTO *utasítás alakja:*

ON aritmetikai kifejezés GOTO sorszámlista

jelentése: „Ugorj arra a sorszámra, amit az aritmetikai kifejezés egészrésze a sorszámlistából kijelöl!" Ennek megfelelően, ha a SGN(X) + 2 kifejezés értéke 1, akkor a 70–es, ha 2, akkor a 60–as, ha 3, akkor az 50–es utasítássor végrehajtásával folytatódik a program. Amennyiben a *kifejezés egészrésze nagyobb*

a sorszámlitásban szereplő sorszámok számánál, vagy *nulla*, az ON–GOTO utasítást követő utasítássor végrehajtásával folytatódik a program. Ezt kihasználva helyesen működik a

```
30 ON SGN(X)+2 GOTO 70,60
```

utasítással is a program, ha a 40–es sort töröljük.

O Irjuk be a programba a kiíró utasításokat, ha a

```
30 ON SGN(X)+1 GOTO 50,60
```

utasítást használjuk!

● Készítsünk öröknaptárt!

Az $a = e+h+n+\left[\frac{8(h+1)}{5}\right]+\left[\frac{e}{4}\right]-\left[\frac{e}{100}\right]+\left[\frac{e}{400}\right]$

kifejezésben „e"–vel az évszámot, „n"–nel a napot jelöltük. „h" a hónap sorszáma, ha az kettőnél nagyobb, egyébként 12–vel növelt értéke. Az „a" kifejezés értékét osztani kell 7–tel, és a kapott maradék jelzi, hogy a kérdéses nap a hét melyik napjára esik: ha a maradék 0, akkor vasárnap, ha 1, akkor hétfő, é.i.t.

```
10 CLS:PRINT"OROKNAPTAR"
20 INPUT "EV";E:INPUT "HO";H:INPUT "NAP";N
30 IF H<3 THEN H=H+12:E=E-1
40 A=E+H+N+INT(8*(H+1)/5)
50 A=A+INT(E/4)-INT(E/100)+INT(E/400)
60 M=A-7*INT(A/7)
70 ON M GOTO 90,100,110,120,130,140
80 PRINT"VASARNAP":END
90 PRINT"HETFO":END
100 PRINT"KEDD":END
110 PRINT"SZERDA":END
120 PRINT"CSUTORTOK":END
130 PRINT"PENTEK":END
140 PRINT"SZOMBAT"
```

```
OROKNAPTAR
EV? 1984
HO? 9
NAP? 13
CSUTORTOK
READY
>_
```


O Oldjuk meg a feladatot az IF utasítással is!

● Adott egy pont és egy egyenes! Irjunk programot, amivel eldönthetjük, hogy a pont illeszkedik–e az egyenesre, vagy ha nem, akkor alatta, vagy felette van–e!
Meg kell határozni a ponttal azonos abszcisszájú pontját az egyenesnek. Ennek és az adott pont ordinátáját kell összehasonlítanunk. A pont elhelyezkedését jelzi a két ordinátaérték különbségének előjele, ami lehetőséget ad arra, hogy használjuk az ON–GOTO utasítást.

```
10 CLS:PRINT " PONT ES EGYENES KOLCSONOS HELYZETE"
20  PRINT "AZ A*X+B ALAKU EGYENLET PARAMETEREI":
  INPUT "A";A: INPUT "B";B
30 PRINT "A PONT KOORDINATAI"
40 INPUT "X";X: INPUT "Y";Y
50 Y1=A*X+B
60 ON SGN(Y1-Y)+2 GOTO 70,80,90
70 PRINT "A PONT  AZ EGYENES FELETT VAN":END
80 PRINT "A PONT ILLESZKEDIK AZ EGYENESRE":END
90 PRINT "A PONT AZ EGYENES ALATT VAN"
```

```
 PONT ES EGYENES KOLCSONOS HELYZETE
AZ A*X+B ALAKU EGYENLET PARAMETEREI
A? 1
B? 0
A PONT KOORDINATAI
X? 3
Y? 4
A PONT  AZ EGYENES FELETT VAN
READY
>_
```

A programban „0–ra vizsgáltuk" az illeszkedést, azaz nem tekintettük illeszkedőnek a pontot minimális eltérés esetén sem, ugyanakkor ez adódhatott a *számítások hibája*ként is.
Tekintsük ezért illeszkedőnek a pontot akkor is, ha az ordináták különbsége 10^{-7}–nél kisebb..
Kerekítsük Y1 változó értékét! Igy kell a legkevesebbet változtatnunk a programon.

```
55 Y1 = INT(Y1*1E7+0.5)*1E-7
```

9. CIKLUS

FOR–TO–STEP/NEXT utasítás

● Gépeljük be a következő programot, és indítsuk el!

```
10 CLS
20 I=1
30 PRINT@412,I
40 I=I+1
50 GOTO 30
```

```
                                284
Break in 30
READY
>_
```

A számítógép számlálni fog.
A GOTO utasítással visszaugrunk a program elejére és így újból és újból végrehajtódik a 30–as és a 40–es sorszámú utasítás.
Egy *ciklust* alakítottunk ki, amelyben az I változó értékének kiíratása és növelése ismétlődik. Ezek az *ismételten végrehajtásra kerülő utasítások alkotják a ciklusmagot.*
Ez a ciklus „végtelen" lesz, mert nem adtunk meg semmiféle feltételt, hogy meddig számláljon. A program csak akkor fejeződik be, ha a BREAK billentyű lenyomásával megszakítjuk a futást, vagy ha az I változó értéke 1.701411E+38–nél nagyobb lesz, azaz túlcsordul.

● Számláljunk N természetes számig (N > 0)!

```
10 REM SZAMLALAS -1
20 CLS:INPUT "MEDDIG";N
30 I=1:REM KEZDOERTEK
40 PRINT@412,I
50 I=I+1:REM MODOSITAS
60 IF I<=N THEN 40:REM DONTES
```

Ebben a programban a 40–es és az 50–es utasítássor alkotja a ciklusmagot. Ezeknek az utasításoknak a végrehajtása ismétlődik mindaddig, ameddig az I változó értéke nagyobb nem lesz N változó értékénél.
Ezt vizsgáljuk az IF utasítással a 60–as utasítássorban. Itt *döntünk* arról, hogy a ciklusmagban szereplő utasításokat végrehajtsuk–e még egyszer, vagy kilépünk a ciklusból.

Ez a döntés I változó értékétől függ, melynek értéke a program futása során változik (50–es sor). Ez lesz a *ciklusváltozó*nk, melynek kezdőértékét még a ciklusba lépés előtt meg kell adnunk (30–as sor).
Egy *ciklus* létrehozásában, *szervezés*ében segíthet az, ha megválaszolunk a következő kérdésekre:

- A változók milyen értékekkel lépnek a ciklusba? Mennyi a *kezdőérték*ük?
- Melyik változó értéke *módosul* a ciklusmagon belül és hogyan?
- Milyen utasítások alkotják a *ciklusmagot?*
- Milyen *feltétel* teljesedésekor fejeződik be a ciklus?

● Irassuk ki az [a, b] –ban található egész számokat!

```
10 CLS:PRINT" A,B ZART INTERVALLUM ZEGESZ SZAMAI-1"
20  INPUT "A=";A:INPUT "B=";B
30 IF A>B THEN T=A:A=B:B=T
40 A=-INT(-A):B=INT(B):E=A
50 IF B<A THEN PRINT "NINCS EGESZ SZAM AZ INTERVALLUMBAN"
  :END
60 PRINT E;
70 E=E+1
80 IF E<=B THEN 60
```

```
 A,B ZART INTERVALLUM ZEGESZ SZAMAI-1
A=? -2.45
B=? 6.7
-2 -1  0  1  2  3  4  5  6
READY
>_
```

A 30–as sorban kicseréljük a változók értékeit, ha esetleg nem a megfelelő sorrendben adnánk meg az adatokat. Az A változó értéke így nem lesz nagyobb a B változó értékénél.
A cseréhez egy „segédváltozót" kell használnunk, hogy ne veszítsük el A, vagy B értékét. Ez a T változó lesz.
A 40–es sorban A értéke A kezdő értékénél nem kisebb, legkisebb egész szám, B értéke pedig B kezdőértékénél nem nagyobb, legnagyobb egész szám lesz. Ezzel megkaptuk az intervallum legkisebb és legnagyobb egész számait.
Az 50–es sorban azt vizsgáljuk meg, hogy egyáltalán van–e egész szám az intervallumban, mert ha B változó értéke kisebb lesz mint A változóé, akkor A és B zárt intervallumban nincs egész szám. Ekkor a ciklusba be sem kell lépnünk.
A 60–as és 70–es utasítássorok alkotják a ciklusmagot. Ismételt végrehajtá-

sokkal E értékeit mindaddig kiírathatjuk, ameddig nagyobb nem lesz B értékénél.

● Határozzuk meg két szám legkisebb közös többszörösét!

A feladatot megfogalmazhatjuk a következőképpen is, konkrét számokkal:

Két gyerek felfelé ugrál a lépcsőn. Az egyik minden 3. lépcsőre, a másik minden 4. lépcsőre ugrik. Akkor, ha mindig az a gyerek ugrik, amelyik lejjebb áll, hányadik lépcsőn állnak először együtt? Általánosan: legyen A és B értéke egyenlő azzal, hogy hány lépcsőt ugrik egy–egy gyerek. A1 és B1 értéke pedig mutassa azt, hogy hányadik lépcsőn állnak.

Az A1 és B1 változókat kell összehasonlítanunk ahhoz, hogy eldöntsük: ugyanazon a lépcsőn állnak–e? Ez egyenlőség esetén következik be.

Egyenlőtlenség esetén a kisebb értékű változó értékét kell növelni, és újból meg kell vizsgálnunk, hogy egyenlőek–e.

Az A1 és a B1 változók értékei A ill. B változók értékeinek többszörösei lesznek. Egyenlőség csak a többszörösök esetében fordulhat elő. Azt, hogy a legkisebb többszöröst kapjuk meg, azzal értük el, hogy mindig a kisebb értékű változó értékét növeljük.

```
10 CLS:PRINT "LEGKISEBB KOZOS TOBBSZOROS"
20  INPUT "A KET SZAM";A,B
30 A1=A:B1=B
40 IF A1=B1 THEN PRINT A;"ES";B;"LEGKISEBB KOZOS
 TOBBSZOROSE";A1:END
50 IF A1<B1 THEN A1=A1+A ELSE B1=B1+B
60 GOTO 40

LEGKISEBB KOZOS TOBBSZOROS
A KET SZAM? 12
?? 21
 12 ES 21 LEGKISEBB KOZOS  TOBBSZOROSE 84
READY
>_
```

A három mintaprogram mindegyikében megtalálhatjuk a *ciklus legfontosabb elemeit,* részeit:

– előkészítés: a változók kezdőértékeinek megadása
– ciklusmag: ismételten végrehajtandó programrészlet,
– ciklusváltozó értékének módosítása,
– döntés: annak vizsgálata, hogy végre kell–e még egyszer hajtani a ciklust, vagy befejeződhet.

Mi az amiben különböznek?

– A számlálást végző programban a ciklus végrehajtására legalább egyszer mindenképpen sor kerül, ha a feltételnek megfelelő értéket adunk N–nek. A másik két programnál már előfordulhat, hogy egyszer sem kell belépni a ciklusba.
– Az első két programban a ciklusváltozó értékének változását előre meg tudjuk határozni. A ciklusmag végrehajtása során mindig megadott értékkel változik.
– Az első és második programban mielőtt belépnénk a ciklusba, már ismert a ciklusváltozó kezdő és végértéke.

Amennyiben egy ciklus a két utóbbi tulajdonsággal rendelkezik, azaz

– ismert a ciklusváltozó kezdő és végértéke, és
– megadható, hogy a ciklusváltozó milyen lépésközzel változzon, a FOR–TO–STEP/NEXT *utasítással is megvalósíthatjuk a ciklust.*

Az utasítás hatásával, működésével a következő program segítségével ismerkedhetünk meg:

```
10 CLS:PRINT " CIKLUSKEPZO UTASITAS"
20  INPUT "KEZDOERTEK";K
30 INPUT "VEGERTEK";V
40 INPUT "LEPESKOZ";L
45 REM CIKLUS
50 FOR X=K TO V STEP  L
60    PRINT X;:REM CIKLUSMAG
70 NEXT  X

 CIKLUSKEPZO UTASITAS
KEZDOERTEK? -5
VEGERTEK? 3
LEPESKOZ? 2
-5 -3 -1  1  3
READY
>_

 CIKLUSKEPZO UTASITAS
KEZDOERTEK? 3
VEGERTEK? -5
LEPESKOZ? 12
 3
READY
>_
```

```
 CIKLUSKEPZO UTASITAS
KEZDOERTEK? 2.5
VEGERTEK? 13
LEPESKOZ? -1.5
 2.5
READY
>_

 CIKLUSKEPZO UTASITAS
KEZDOERTEK? 13
VEGERTEK? -3.6
LEPESKOZ? -1.5
 13  11.5  10  8.5  7  5.5  4  2.5  1 -.5 -2 -3.5
READY
>_
```

A számítógépnek a FOR–TO–STEP *és a* NEXT *utasítások*kal – amit tekinthetünk egy utasításnak is, mert mindig párban kell hogy szerepeljenek – a *következő utasítássorozatot adtuk:*

– A *ciklusváltozó*, X értéke legyen egyenlő a *kezdőérték*kel, K értékével, és hajtsa végre az utasításokat, amelyeket az azonos ciklusváltozójú NEXT utasításig talál, azaz a ciklusmag utasításait.
– Változtassa a ciklusváltozó értékét a *lépésközzel*, L értékével.
– Amennyiben L > ∅ esetén a K≤X≤V egyenlőtlenség, vagy ha L<∅, akkor a K≥X≥V egyenlőtlenség teljesedik, ahol a V a *végérték*, a ciklusmagot végrehajtódik, egyébként a NEXT utasítást követő utasítással folytatódik a program.

A kezdő és végértékeket és a lépésközt tetszőleges aritmetikai kifejezésekkel is megadhatjuk.
A lépésközt meghatározó kifejezés értékétől függ, hogy a ciklusváltozó értéke növekedni vagy csökkenni fog–e. A kezdőértékkel egyszer mindenképpen végrehajtódik a ciklus, ezért a FOR *utasítás előtt vizsgálatot kell végeznünk*, ha előfordulhat az az eset, hogy egyszer sem kell végrehajtani a ciklust.

O Vizsgáljuk meg, hogy milyen értékeket vesz fel a ciklusváltozó a kezdő és végérték, valamint a lépésköz különböző értékeinél:

● A „számláló" program:


```
10 REM SZAMLALAS - 2
20 CLS: INPUT "MEDDIG SZAMLALJAK";N
30 FOR I=1 TO N STEP 1
40 PRINT@ 412,I
50 NEXT I
```

Akkor, ha a lépésköz 1, a következő alakban is megadható a FOR utasítás:

```
30  FOR I=1 TO N
```

● Az [a, b] egész számai:

```
10 CLS:PRINT "A,B ZART INTERVALLUM EGESZ SZAMAI -2"
20  INPUT "A=";A:INPUT "B=";B
30 IF A>B  THEN T=A:A=B:B=T
40  A=-INT(-A):B=INT(B):E=A

50 IF B<A THEN PRINT "NINCS EGESZ  SZAM AZ
  INTERVALLUMBAN":END
60 FOR E=A TO B
70 PRINT E;
80 NEXT E

A,B ZART INTERVALLUM EGESZ SZAMAI -2
A=? 8.9
B=? -3
-3 -2 -1  0  1  2  3  4  5  6  7  8
READY
>_
```

Ennél a feladatnál, mielőtt a FOR ciklusba belépnénk, megvizsgáltuk, hogy végre kell–e hajtani legalább egyszer a ciklust.
Ez a feladat megoldható a következőképpen is:

```
10 CLS:PRINT "A,B ZART INTERVALLUM EGESZ SZAMAI -3"
20  INPUT "A=";A:INPUT "B=";B
30 IF INT(A)=INT(B)AND INT(A)<>A AND INT(B)<>B THEN
   PRINT "NINCS EGESZ SZAM AZ INTERVALLUMBAN":END
40 IF A<B THEN K=INT(A):V=INT(B) ELSE  K=INT(B):V=INT(A)
50 FOR E=K TO V:PRINT E;:NEXT E

A,B ZART INTERVALLUM EGESZ SZAMAI -3
A=? 3.78
B=? 3.1
NINCS EGESZ SZAM AZ INTERVALLUMBAN
READY
>_
```

A második megoldásnál a ciklust egy sorba írtuk, ami megengedett. Mielőtt a ciklusba lépnénk itt is megvizsgáljuk, hogy van–e egész szám az intervallumban (30–as sor). A 40–es sorban az első és az utolsó egész számot határozzuk meg, miközben figyelembe vesszük azt, hogy az intervallum végpontjait nem biztos, hogy növekvő sorrendben adtuk meg.

● Határozzuk meg egy természetes szám összes osztóit!

```
10 CLS:PRINT " OSSZES OSZTO -1"
20  INPUT "N=";N
30 FOR  I=1  TO N-1
40 IF N/I=FIX(N/I) THEN PRINT I;
50 NEXT I
60  PRINT N;

 OSSZES OSZTO -1
N=? 60
 1  2  3  4  5  6  10  12  15  20  30  60
READY
>_
```

A 40–es sorban, ami a FOR ciklus magja, vizsgáljuk az oszthatóságot 1–től N–ig.
Ez a megoldás elég lassú lesz. Sok felesleges vizsgálatot végzünk, mivel N/2-től N–ig biztosan nincs osztója N–nek.
Ezt figyelembevéve a megoldás:

```
10 CLS:PRINT" OSSZES OSZTO -2"
20 INPUT "N=";N
30 FOR I=1 TO N/2
40 IF N/I=FIX(N/I) THEN PRINTI;
50 NEXT I
60 PRINTN;

 OSSZES OSZTO -2
N=? 48
 1  2  3  4  6  8  12  16  24  48
READY
>_
```

Javíthatunk a programunkon, azaz meggyorsíthatjuk, ha észrevesszük azt, hogy ha megtaláltunk egy osztót, akkor tulajdonképpen már két osztót ismerünk: ha I osztója N–nek, akkor N/I is az lesz. A vizsgálatot ekkor csak N négyzetgyökéig kell végeznünk.
Meg kellett azt is oldani, hogy ha N négyzetszám, ne írja ki kétszer a négyzetgyökét, azaz I–t és N/I–t is (40–es sor).


```
10 CLS:PRINT "OSSZES OSZTO -3"
20  INPUT "N=";N
30 FOR I=1 TO SQR(N)
40 IF N/I=FIX(N/I)THEN IF I*I=N THEN PRINT I
   ELSE PRINT I,N/I
50 NEXT I

OSSZES OSZTO -3
N=? 12
 1                12
 2                6
 3                4
READY
>_
```

Az egymásba épített IF–THEN–ELSE utasítások helyett inkább használjunk több IF utasítást.

```
10 CLS:PRINT "OSSZES OSZTO -4"
30 CLS:INPUT "N=";N
40 FOR I=1 TO SQR(N)
50 IF N/I<>FIX(N/I)THEN 80
60 IF I*I<>N THEN PRINT I;
70 PRINT N/I;
80 NEXT I

N=? 120
 1  120  2  60  3  40  4  30  5  24  6  20  8  15  10  12
READY
>_
```

Vegyük észre, hogy a ciklusmag végrehajtását a ciklusváltozó egy adott értékével úgy fejezhetjük be, ha a NEXT utasítást végrehajtjuk. A 40–es sorról a 70–es sorra ugrunk, ha I nem osztója N–nek. A mintaprogramokban láthattuk, hogy a ciklusváltozó kezdő– és végértékét, valamint a lépésközt aritmetikai kifejezésekkel is megadhatjuk. A *ciklusváltozó* lehet egész, vagy egyszeres pontosságú, *de nem lehet kétszeres pontosságú változó*.

Az előző programot kétszeres pontosságu változókkal a következőképpen oldhatjuk meg:

```
10 CLS:PRINT "OSSZES OSZTO -5"
20 DEFDBL I,N
30 INPUT "N=";N
40 I=1
50 IF N/I<>FIX(N/I)THEN 80
60 IF I*I<>N THEN PRINT I;
70 PRINT N/I;
80 I=I+1:IF I*I<=N THEN 50

OSSZES OSZTO -5
N=? 1024
 1  1024  2  512  4  256  8  128  16  64  32
READY
>_
```

● Szorozzuk össze 1–től N–ig a természetes számokat!

A szorzat értékes számjegyeinek száma egyszeres pontosság esetén maximum 7 lehet, ezért 10^7–nél nagyobb szorzatokat már nem iratjuk ki.

```
10 CLS:PRINT" N FAKTORIALIS"
20  INPUT  "N=";N
30 P=1:REM ELOKESZITES
40 FOR I=1  TO N
50 P=P*I
60 IF P>1E7  THEN 90
70 NEXT I
80 PRINT N;"!=";P:END
90 PRINT  N;"! NAGYOBB, MINT 10^7"

 N FAKTORIALIS
N=? 7
 7 != 5040
READY
>_
```

Két dolgot figyeljünk meg:

– A 30–as utasítássorban P változónak a kezdőértéke 1 lesz. A szorzatot csak így tudjuk előállítani.
– A 60–as utasítássorban azt vizsgáljuk, hogy a szorzat értéke nagyobb–e, mint 10^7. Amennyiben igen, kiugrunk a ciklusból. A ciklusból kiugorhatunk, de nem ugorhatunk a ciklusba. Az első végrehajtásra kerülő utasításnak a FOR–TO–STEP utasításnak kell lennie, amikor a ciklusba lépünk.

Két ciklus *egymásba ágyazását* vizsgáljuk meg a következő mintaprogrammal.

● Legyen: $A = \{x \in \mathbf{N} / a \leq x \leq b\}$ és
$B = \{y \in \mathbf{N} / c \leq y \leq d\}$

Képezzük A X B–t!

```
10 CLS:PRINT "DIREKT SZORZAT"
20 PRINT "A<B ES C<D LEGYEN"
30  INPUT "A=";A:INPUT "B=";B
40 INPUT "C=";C:INPUT "D=";D
50 PRINT "A x B=<";
60 FOR X=A TO B
70 FOR Y=C  TO D
80 PRINT " (";X;",";Y;")";
90 NEXT Y
100 NEXT X
110 PRINT ">"

DIREKT SZORZAT
A<B ES C<D LEGYEN
A=? 1
B=? 2
C=? 3
D=? 4
A x B=< ( 1 , 3 ) ( 1 , 4 ) ( 2 , 3 ) ( 2 , 4 )>
READY
>_
```

Az A halmaz elemeit a külső ciklus (a ciklusváltozó: X), B halmaz elemeit a belső ciklus (a ciklusváltozó: Y) állítja elő.
Az A halmaz első elemét párosítjuk a B halmaz összes elemével, majd a másodikat, és így tovább. *Ciklusok egymásba ágyazása*kor a külső ciklus teljes egészében tartalmazza a belső ciklust. Összekapcsolva az összetartozó (azonos ciklusváltozójú) FOR és NEXT utasításokat, a vonalak nem keresztezhetik egymást.

● Határozzuk meg egy függvény zérushelyét intervallum felezés módszerével!

Ezzel a mintaprogramunkkal azt szeretnénk bemutatni, többek között, hogy miképpen szervezzük a ciklust, ha nem használhatjuk a FOR–TO–STEP/NEXT utasítást.
Az „intervallum felezés" több ismert módszer közül az egyik, a függvények zárushelyének a közelítő meghatározására.

Kiinduláshoz ismernünk kell azt az intervallumot, amelyben a zérushelyet megtalálhatjuk. Legyen a zérushelyet tartalmazó intervallum két végpontja: a és b. Folytonos függvények esetében annak eldöntéséhez, hogy az intervallumon belül a függvénynek van–e zérushelye, azt kell megvizsgálnunk, hogy az intervallum végpontjaihoz tartozó függvényértékek *ellentétes előjelűek–e*. Amennyiben előjelet vált a függvény, akkor valahol metszenie kell az x tengelyt. (Esetleg több helyen is, de ezzel nem foglalkozunk.)
Az intervallum *felezéspontjához* tartozó függvényértéket kiszámítjuk és megvizsgáljuk, hogy az intervallum kezdőpontjához tartozó függvényértékkel *azonos előjelű–e*: amennyiben igen, akkor a zérushely az intervallum második, egyébként az első felében lesz. Így kapunk egy újabb intervallumot, – de ez már kisebb lesz – amelyben a zérushely megtalálható. Ezt az eljárást ismételjük mindaddig, ameddig az intervallum hossza kisebb nem lesz egy adott értéknél.
Programunkkal az $y = x^2 - 2$ függvény zérushelyét határozhatjuk meg.

```
10 CLS:PRINT "ZERUSHELY FELEZESSEL"
20  INPUT "AZ INTERVALLUM VEGPONTJAI";A,B
30 INPUT "PONTOSSAG";EP
40 YA=A*A-2:YB=B*B-2
50 IF YA*YB)=0 THEN 20
60 C=(A+B)/2: PRINT£266,C
70 YC=C*C-2
80 IF YA*YC=0 THEN PRINT "ZERUSHELY=";C:END
90 IF YA*YC)0 THEN A=C ELSE B=C
100 IF ABS(B-A))EP THEN 60
110 PRINT "ZERUSHELY=";FIX((A+B)/2/EP)*EP

ZERUSHELY FELEZESSEL
AZ INTERVALLUM VEGPONTJAI? 0
?? 2
PONTOSSAG? 0.001
           1.41504
ZERUSHELY= 1.414
READY
>_
```

A programban a függvényértékek szorzatának előjeléből állapíthatjuk meg, hogy megegyeznek–e az előjeleik. A 80–as sorban azt vizsgáljuk, hogy a felezéssel esetleg nem „találtuk–e el" a zérushelyet.
Más függvények zérushelyeinek a kiszámításához a 40–es és a 70–es sorokat kell csak átírni.
Kétszeres pontossággal is meghatározhatjuk a függvény zérushelyét, ha kétszeres pontosságú változókat használunk.

Ezt a

```
15 DEFDBL A–Z
```

utasítássorral érhetjük el.

● Határozzuk meg más módszerrel is a számok négyzetgyökét!

```
10 CLS:PRINT "NEGYZETGYOK KOZELITES "
20 DEFDBL A,P,X,I:INPUT"MELYIK SZAMNAK A NEGYZETGYOKET
 KERESSUK";A
30 INPUT "MILYEN PONTOSSAGGAL";P
40 PRINT "VALAMELYIK BILLENTYUT NYOMJUK LE,AKKOR FUT
 TOVABB A PROGRAM
50 X=0
60 I=1
70 X=X+I
80 PRINT"X=";X;TAB(40);"XÄ2";X*X
90 IF X*X<A THEN 70
100 X=X-I
110 I=I/10
120 IF INKEY$="" THEN 120
130 IF A-X*X>P THEN 70
140 PRINT A;"NEGYZETGYOKE ";X
150 PRINT"PONTOSSAG= ";P

X= 1.73                          XÄ2 2.9929
X= 1.74                          XÄ2 3.027600000000001
X= 1.731                         XÄ2 2.996361000000001
X= 1.732                         XÄ2 2.999824000000001
X= 1.733                         XÄ2 3.003289000000001
X= 1.7321                        XÄ2 3.000170410000001
X= 1.73201                       XÄ2 2.999858640100001
X= 1.73202                       XÄ2 2.999893280400001
X= 1.73203                       XÄ2 2.999927920900001
X= 1.73204                       XÄ2 2.999962561600001
X= 1.73205                       XÄ2 2.999997202500001
X= 1.73206                       XÄ2 3.000031843600001
 3 NEGYZETGYOKE  1.73205
PONTOSSAG=  1D-04
READY
>_
```

Egész, tized, század, stb. értékekkel növeljük az X változó értékét és azt vizsgáljuk, hogy az X értékének négyzete nagyobb–e, mint a szám (A).
Amennyiben nagyobb lesz, visszalépünk, levonjuk a növekedést X értékéből (90–es sor) és vesszük az I értékének tizedrészét, amivel növelni fogjuk újból az X értékét. A program akkor fejeződik be, ha az adott pontosságot elértük.

10. INDEXES VÁLTOZÓK

DIM, CLEAR *utasítások*

Egy számsorozat elemeit a következőképpen jelöljük: $a_0, a_1, a_2, a_3, \ldots$. Ez a jelölésmód lehetővé teszi azt, hogy a *sorozat* bármely elemét az indexével, azaz egy nem negatív egész számmal adjuk meg. Erre a BASIC programozási nyelvben is van lehetőség. Az A(0), A(1), A(2), A(3), . . . *indexes változók* alkalmasak egy számsorozat elemeinek tárolására. Jelentősége akkor válik nyílvánvalóvá, ha tudjuk azt, hogy az *indexek*et, tetszőleges aritmetikai kifejezéssel is megadhatjuk:

A(2*I), IJ(K%), X$(L%), D%(I+5)

A kifejezés egész része lesz a változó indexe, ami meghatározza azt, hogy melyik elemről van szó. Ez az érték nem lehet negatív. Az *indexet* meghatározó aritmetikai kifejezés értékét a programban *változtathatjuk* és így egy bizonyos műveletet a sorozat bármely elemével végrehajthatunk úgy, hogy nem kell minden elemre külön felírni ugyanazt az utasítást.

● Adott egy N elemű számsorozat. A sorozat elemeit olvassuk be egy indexes változóba és írassuk ki fordított sorrendben!

```
10 REM INDEXES VALTOZOK
20 INPUT "AZ ELEMEK SZAMA";N
30 FOR I=1 TO N
40 READ A(I):PRINT A(I);
50 NEXT I
60 PRINT
70 FOR I=N TO 1 STEP -1
80 PRINT A(I);
90 NEXT
100 DATA 12,43,6,8787,34,-42,43,34,22,-34

>RUN
AZ ELEMEK SZAMA? 8
 12  43  6  8787  34 -42  43  34
 34  43 -42  34  8787  6  43  12
READY
>_
```


A READ A(I) és a PRINT A(I) utasításokkal a sorozat minden elemét be tudtuk olvasni ill. ki tudtuk írni azáltal, hogy az I értékét a ciklus-utasítással változtattuk.

Az *egyindexes változók*at *vektorok*nak ill. *egydimenziós tömbök*nek is nevezzük. Az indexes változók használata előtt a programban gondoskodnunk kell arról a memóriaterületről, ahol a változók értékeit tárolja a rendszer. Ezt a DIM utasítással adhatjuk meg. 11 elemig a helyfoglalás automatikus, de ha az elemek száma ettől több, akkor a helyet nekünk kell biztosítanunk. Meg kell adni a DIM *utasítás*ban az elemek maximális számát.

41 elemet olvashatunk be a programba, ha kiegészítjük a programunkat a

15 DIM A(40)

utasítással.

● Adott egy, legfeljebb 100 elemből álló számsorozat. Tároljuk a számsorozat elemeit az A vektorban! Válogassuk ki a számsorozatból a nem egész számokat, a páros ill. a páratlan egész számokat, és ezeket helyezzük el a B, C ill. D vektorokba!

A, B, C és D vektor elemeinek számát a kiíratáskor ismernünk kell, ezért az IB, IC és az ID változókkal számláljuk a kiírandó vektorok elemeinek számát. A kiíratás előtt vizsgáljuk meg értéküket, mivel a FOR ciklus egyszer mindenképpen végrehajtódik.

```
10 REM VEKTOROK  - VALOGATAS
20 DIM A(100),B(100),C(100),D(100)
30 INPUT "A SOROZAT ELEMEINEK SZAMA";N
40 FOR I=0 TO N-1:PRINT I;".-";:INPUT A(I):NEXT I

50   IB=0:IC=0:ID=0: REM INDEXEK
60 REM VALOGATAS
70  FOR I=0 TO N-1
80 IF  A(I)<>FIX(A(I)) THEN B(IB)=A(I):IB=IB+1:GOTO 100
90 IF A(I)/2=FIX(A(I)/2) THEN  C(IC)=A(I):IC=IC+1  ELSE
  D(ID)=A(I):ID=ID+1
100  NEXT I
110  REM KIIRATAS
120 CLS:PRINT "NEM EGESZ  SZAMOK"
130 IF IB=0 THEN 150
140 FOR I=0 TO IB-1:PRINT B(I);:NEXT I
150 PRINT :PRINT "PAROS SZAMOK"
160  IF IC=0 THEN 180
170 FOR I=0 TO IC-1:PRINT C(I);:NEXT I
180 PRINT:PRINT "PARATLAN SZAMOK"
190 IF ID=0 THEN END
200 FOR I=0 TO ID-1:PRINT D(I);:NEXT I
```

O Oldjuk meg a Többirányú elágazás c. fejezet feladatait indexes változók alkalmazásával!

● Készítsünk statisztikát az osztály dolgozatairól! Számláljuk össze az osztályzatokat és számítsuk ki az osztályátlagot!
Ezzel a programmal azt mutatjuk be, hogy az indexes változókkal hogyan oldhatunk meg egy szétválogatási feladatot. A különböző osztályzatú dolgozatok számlálását indexes változóval végezzük úgy, hogy az osztályzatokat használjuk indexként. A vektornak azt az elemét növeljük eggyel, amelyben az indexnek megfelelő osztályzatokat számláljuk (80–as sor).
A DIM utasítást nem kell használnunk, mert 11 elemig a rendszer biztosítja a helyet.

```
10 CLS:PRINT " DOLGOZATOK STATISZTIKAJA"
20 INPUT "A DOLGOZATOK SZAMA";N
30 DIM B(5)
40 PRINT"ERDEMJEGYEK"
50 FOR J=1 TO N
60 INPUT E
70 IF E<1 OR E>5 THEN PRINT"NEM OSZTALYZAT":GOTO 60
80 B(E)=B(E)+1
90 NEXT J
100 REM ATLAGSZAMITAS
110 S=0:FOR J=1 TO 5
120 S=S+J*B(J)
130 NEXT J
140 REM KIIRATAS
150 CLS:FOR I=5 TO 1 STEP -1 :READ A$:PRINTA$,B(I)
;" DB":NEXT I
160 PRINT"OSZTALYATLAG";:PRINTUSING"###.##";S/N
170 DATA JELES,JO,KOZEPES,ELEGSEGES,ELEGTELEN
```

```
 DOLGOZATOK STATISZTIKAJA
A DOLGOZATOK SZAMA? 6
ERDEMJEGYEK
? 3
? 4
? 1
? 6
NEM OSZTALYZAT
? 3
? 5
? 4_
JELES              1  DB
JO                 2  DB
KOZEPES            2  DB
ELEGSEGES          0  DB
ELEGTELEN          1  DB
OSZTALYATLAG  3.33
READY
>_
```


● Irjunk programot, ami TOTO tippeket ad!
Tároljuk el indexes változókba a tippeket, azaz X, 1 és 2–t, majd állítsuk elő az RND függvénnyel ezek indexeit.

```
10 CLS:PRINT " TOTO TIPPEK"
20 RANDOM
30 A$(1)="1 ":A$(2)="2 ":A$(3)="X "
40 FOR I=1 TO 13:PRINT A$(RND(3));:NEXT I
```

```
 TOTO TIPPEK
2 X X 2 2 X X 1 2 1 1 2 X
READY
>_
```

● Adott N db szám. Keressük ki közülük a legkisebbet!
A számokat egy vektorban tároljuk. Az index értékének változtatásával egy ciklussal megoldhatjuk, hogy minden elemet összehasonlítsunk azzal az elemmel, amely előzőleg a legkisebbnek adódott. Először a legkisebb szám természetesen a sorozat első eleme lesz, ezért az L változó kezdő értéke 1. Értéke csak akkor változik, ha az összehasonlítások során kisebb elemet talál a program. A ciklus végén a legkisebb elem indexét adja az L értéke.
A DIM utasítással eddig a *program írásakor* adtuk meg az elemek maximális számát. A határ korlátozott volt. Amennyiben egy vektor elemeinek száma *futtatásonként más és más,* éljünk azzal a lehetőséggel, hogy a *helyfoglalás* a tényleges elemszám ismeretében történhet. A DIM A(N) utasítással N számú elem helyét biztosíthatjuk. A DIM utasítás előtt már kell, hogy N változónak értéke legyen.

```
10 CLS:PRINT " A LEGKISEBB ELEM KIKERESESE":
PRINT"AZ ADATOK A DATA ADATLISTABAN VANNAK"
20 READ N: DIM A(N)
30   FOR I=1 TO N:READ A(I):NEXT I
40 L=1:REM A LEGKISEBB INDEXE
50 FOR I=2 TO  N
60 IFA(L)>A(I) THEN L=I:REM UJ INDEX
70 NEXT I
80 PRINT "A LEGKISEBB ELEM";A(L)
90 DATA 10,12.5,437,-45,0,7435.5,53.2,-432,434,712,543
```

```
 A LEGKISEBB ELEM KIKERESESE
AZ ADATOK A DATA ADATLISTABAN VANNAK
A LEGKISEBB ELEM-432
READY
>_
```

● Készítsünk programot, amivel abc sorrendbe rendezhetünk szavakat!

Miután kiválasztottuk a sorozat legnagyobb elemét – ezt csinálta előző programunk – csak azt kell megoldanunk, hogy a sorozatnak ezt az elemét már ne választhassuk ki újból, ha a sorozatot ismét végigvizsgáljuk. Helyezzük el a legnagyobb elemet a helyére, és a vizsgálatot csak a következő elemtől végezzük.

Numerikus értékek tárolása meghatározott számú memóriarekeszben történik: egész változó 5, egyszeres pontosságú változó 7, kétszeres pontosságú 11 rekeszt igényel (névvel együtt).

Karakterláncok (stringek) hossza változhat. Maximálisan 255 karaktert tudunk tárolni egy változóban. Ez indokolja, hogy a karakterláncok részére a memóriahelyről külön is kell gondoskodnunk.

CLEAR kifejezés – *memória törlés*

Hatása: törli a változók értékeit és a kifejezés egészrészének megfelelő számú memóriarekesz tartalmát, amiket lefoglal a stringek, stringműveletek számára.

CLEAR utasítás előzze meg a beolvasó, a tipusdefiniáló és a DIM utasításokat. Legjobb, ha ezzel kezdjük a programot. Program közben csak akkor használjuk, ha minden változó értékét törölni akarjuk.

```
10 CLS:PRINT "SORBARENDEZES A LEGNAGYOBB KIVALASZTASAVAL"
20 READ N
30 DEFSTR A,T:DIM A(N)
40 FOR I=1 TO N: READ A(I): PRINT A(I):NEXT I
50 FOR I=1 TO N-1
60 FOR J=I+1 TO N
70 IF A(I)>A(J) THEN T=A(I):A(I)=A(J):A(J)=T
80 NEXT J
90 NEXT I
100 FOR I=1 TO N:PRINTé74+64*(I-1),A(I);:NEXTI
110 DATA 13,EMESE,ERIKA,LACI,AGI,ZSUZSI,MARI
120 DATA ERZSI,KATI,EVI,LAJOS,PETI,ISTVAN,ILDI

SORBARENDEZES A LEGNAGYOBB KIVALASZTASAVAL
EMESE      AGI
ERIKA      EMESE
LACI       ERIKA
AGI        ERZSI
ZSUZSI     EVI
MARI       ILDI
ERZSI      ISTVAN
KATI       KATI
EVI        LACI
LAJOS      LAJOS
PETI       MARI
ISTVAN     PETI
ILDI       ZSUZSI
READY
>_
```


● Az eratoszteneszi szita módszerével keressük meg 200–ig a prímszámokat!

Számítógép nélkül ez úgy történik, hogy 200–ig leírjuk a számokat és először kihúzzuk a számok közül 2 többszöröseit, majd 3 többszöröseit, stb., vagyis a prímszámok többszöröseit. Azt nekünk nem kell tudni ekkor, hogy egy szám prímszám–e. Elég ha azt látjuk, hogy ha nincs kihúzva, akkor a többszöröseit törölni kell. A vizsgálattal elég a legnagyobb szám $\sqrt{N}$–ig elmenni, mert ennél nagyobb szám többszöröseit már biztosan nem találjuk a számok között.

Példaként 20–ig bemutatjuk az eljárást:

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~

11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

Ezek után csak le kell olvasni azokat a számokat, amelyek megmaradtak.

Programmal egy vektorban elhelyezett számokkal kell ugyanezt elvégeznünk. Azt kell megvizsgálnunk, hogy a vektor következő elemét töröltük–e már, azaz az értéke nulla–e. Amennyiben nem, ki kell nulláznunk a szám többszörösének megfelelő indexű elemeket, mivel az index azonos a változó értékével.

A képernyőn követhetjük az eljárást. A kiíratásnál azt kellett megoldani, hogy a számok mindig ugyanolyan mezőszélességet foglaljanak el, hogy áttekinthető legyen a képernyő.

A program végére egy várakozó ciklust írtunk, hogy a kép változatlan maradjon mindaddig, amig le nem nyomtunk egy billentyűt.

```
10 REM ERATOSZTENESZI SZITA
20 N=200
30 DIM A(N):CLS
40 FOR I=2 TO N:A(I)=I:PRINTUSING"#####";I;:NEXT I
50 FOR I=2 TO N/2
60 IF A(I)=0 THEN 100
70 FOR J=2*I TO N STEP I
80 A(J)=0:PRINTé(J-2)*4,"     ";
90 NEXT J
100 NEXT I
110 PRINTé836,"SZITALAS KESZ.USS LE EGY BILLENTYUT!"
120 Q$=INKEY$:IF Q$="" THEN 120
130 REM KIIRATAS
140 CLS:PRINT" A PRIMSZAMOK ";N;"-IG"
150 FOR I=1 TO N:IF A(I)<>0 THEN PRINTUSING"#####";A(I);
160 NEXT I
```

```
  2   3     5     7           11    13          17
     19           23                29    31
                37          41    43          47
                53                59    61
     67           71    73                79
     83                 89                      97
          101   103         107   109         113
                                         127
    131                137   139
          149   151               157
    163         167               173
    179   181                           191   193
          197   199
  SZITALAS KESZ.USS LE EGY BILLENTYUT!
```

```
A PRIMSZAMOK  200 -IG
  2   3   5   7  11  13  17  19  23  29  31  37  41  43  47  53
 59  61  67  71  73  79  83  89  97 101 103 107 109 113 127 131
137 139 149 151 157 163 167 173 179 181 191 193 197 199
READY
>
```

O Tömörítsük a prímszámokat! Helyezkedjenek el egymás után a vektorban!

Egy *táblázat* adatait szeretnénk tárolni és azokkal műveletet végezni. Ehhez a táblázat elemeinek egyértelmű megjelölése szükséges. Akkor, ha megadjuk a táblázatban szereplő elem *sorát* és *oszlopát*, a meghatározás egyértelmű. A BASIC nyelvben ezt *kétindexes változók*kal valósíthatjuk meg. Ezeket *mátrixok*nak, vagy *kétdimenziós tömb*öknek is szokás nevezni.

Például: A(I,J) X$(2*K,15) I%(5,4)

Az első index a sor, a második az oszlop sorszámát adja. Használatukkor az egyindexes változóknál leírtak érvényesek.

● Adott egy N elemű ponthalmaz. Határozzuk meg a pontok távolságát!

A pontokat koordinátájukkal adjuk meg. A távolságokat Pitagorasz–tételének alkalmazásával számítjuk ki. A pontok párosítása két, egymásba ágyazott ciklussal történik.


```
10 CLS:PRINT "PONTOK TAVOLSAGA - PONTOK KOORDINATAI
 A DATA ADATLISTABAN(10 DB)"
20 INPUT"A PONTHALMAZ ELEMEINEK SZAMA";N
30 DIM X(N),Y(N),T(N,N)
40 FOR I=1 TO N:READ X(I),Y(I):NEXT I
50 REM TAVOLSAGOK
60 FOR I=1 TO N
70 FOR J=1 TO N
80 T(I,J)=SQR((X(I)-X(J))^2+(Y(I)-Y(J))^2)
90 NEXT J
100 NEXT I
110 REM KIIRATAS
120 CLS:FOR I=1 TO N:FOR J=1 TO N
130 PRINTUSING"######.##";T(I,J);
140 NEXT J
150 PRINT
160 NEXT I
170 REM PONTOK KOORDINATAI
180 DATA 0,3,4,0,0,0,1,1,2,2,-2,2,-2,-2,5,5,12,-3
190 DATA 12.3,4.5,23,123
```

```
     0.00     5.00     3.00     2.24     2.24
     5.00     0.00     4.00     3.16     2.83
     3.00     4.00     0.00     1.41     2.83
     2.24     3.16     1.41     0.00     1.41
     2.24     2.83     2.83     1.41     0.00
READY
>_
```

O A táblázat szimmetrikus. Oldjuk meg úgy is a feladatot, hogy a táblázatnak csak az egyik felét írja ki a programunk!

O Irjuk át a programot! A táblázatnak csak meghatározott elemét írja ki! Ezt lehessen kérni.
Kettőnél több indexű változókat is használhatunk a BASIC rendszerünkben. Ezeket a változókat *többdimeziós tömb*öknek is nevezik.

● Készítsünk programot, amivel megkönnyíthetjük egy szálloda ügyintézését!

A feladat elég összetett, ezért egyszerűsítsük le. Ez azért is szükséges, mivel a memóriánk kapacitása korlátozott.
A szálloda vendégeinek adatait három hónapos időintervallumban tároljuk. A szállodánk két emeletes és emeltenként három szoba van.
Az adatokat egy *négyindexes változó*ban helyezzük el, ahol az

első index a hónap sorszáma,
a második a napot,
harmadik az emeletet,
a negyedik a szobaszámot jelenti.

Programunkkal csak arra vállalkozunk, hogy a szálloda vendégeinek adatait az időpont, emelet és szobaszám ismeretében lekérdezhessük.
Az adatokat a DATA adatlistában helyezzük el, hogy kipróbálhassuk, jól működik–e a programunk. Alkalmazva a programot az adatokat INPUT utasítással is bekérhetjük, de mód van arra is, hogy mágnesszalagról olvassuk be. Erről azonban később lesz szó.

```
10 CLS:PRINT "A SZALLODA LAKOI - HONAP <=3;NAP(=31;
EMELET(=2,SZOBASZAM(=3":PRINT "A PROGRAM A 'VEGE'
 SZO BEGEPELESEKOR ALL LE
20 CLEAR 100  :DIM LA¤(3,31,2,3)
30 READ H,NA,E,SZ,A¤
35 IF H=0 THEN 60
40 LA¤(H,NA,E,SZ)=A¤
60 INPUT "HONAP";H¤:IF H¤="VEGE" THEN END ELSE H=VAL(H¤)
70 INPUT "NAP";NA:INPUT "EMELET";E:INPUT "SZOBA";SZ
80 IF LA¤(H,NA,E,SZ)=""THEN PRINT"NEM FOGLALT" ELSE
 PRINT"LAKOJA  - ";LA¤(H,NA,E,SZ)
90 PRINT "EGY BILLENTYUT USS LE!"
100 Q¤=INKEY¤:IF Q¤="" THEN 100
110 GOTO 60
120 REM A LAKOK ADATAI
130 DATA 1,12,1,1,KIS JOZSEF,2,15,2,3,NAGY PAL,2,
12,0,3,SAS AMAL,3,20,2,1,FEKETE PETER
140 DATA 0,0,0,0,0¤
```

O Hogyan kell módosítanunk a programot több szálloda esetében?

11. MŰVELETEK KARAKTEREKKEL

String függvények

Az eddigiekben a karakterláncokról (stringekről) csak annyi szó esett, hogy a stringeket tároló változókat a numerikus változóktól a $ jellel különböztetjük meg. A$, XIS, vagy a DEFSTR utasítással jelöljük ki: DEFSTR A–B, T
Figyelem! Az a nyomtató, amivel a könyv programjait kiírattuk a $ jel helyett a ¤ jelet használja.
A PRINT utasítással kiírt szöveg tulajdonképpen egy string, amit idézőjelek közé írunk:

```
PRINT "TERULET" ; T
```

Idézőjelek közé kell tennünk az értékadó utasításban is:

```
A$ = "ISKOLA"
```

Ezekben az esetekben az idézőjelet kivéve minden billentyűzettel előállítható és kiírásban megjelenő karakter szerepelhet a stringben: számjegyek, betűk, írásjelek, műveleti jelek.
Az INPUT és READ (DATA) utasításokkal történő értékadásoknál nem kell idézőjelet használni.
A stringváltozók értékei karakterenként egy–egy memóriarekeszt foglalnak le. A karakterek kódjait tárolják a rekeszek. A kódok decimális értékeit a Mellékletben megtaláljuk.
Két string *összehasonlításakor* a reláció logikai értéke a karakterek kódjainak értékétől függ. A kódok összehasonlítása történik balról jobbra haladva mindaddig, ameddig valamelyik string végére nem érünk.
Csak betűket tartalmazó stringek esetében, azt mondhatjuk, hogy az a string kisebb, amelyik az abc sorrend szerint előbbre van. Ez adódik abból, hogy a betűk kódjai az abc szerint növekednek. A nagybetűk kódjai kisebbek, mint a kisbetűké.
A + jel a stringeknél is egy műveletet jelez, a *stringek összefűzését*.
Két stringből egy stringet hozhatunk létre. Például: A$ = "EFGH", B$ = "PQR", akkor az A$ + B$ értéke EFGHPQR lesz.
Több olyan műveletet végezhetünk a stringekkel, amit függvény alakban adhatunk meg. Ezek a *stringfüggvények*.
A stringfüggvények értékei lehetnek:

```
számok – LEN("SYZ")
         ASC(A$)
         VAL("143.56")
```

stringek — CHR$(65+I)
LEFT$(A$, 3)
RIGHT$("ABCD", 2)
MID$(A$, 3, N)
STR$(2*K+I)
STRING$(5,65)

A függvények nevei után írt $ jel az, ami jelzi, hogy a függvény értéke string lesz. Az argumentumok, ahogyan ezt a példák is mutatják, aritmetikai kifejezések és stringek, string változók, vagy olyan stringfüggvények, melyeknek értéke string.

A következő programmal bemutatjuk az alábbi stringfüggvényeket.

LEN (string) — *a string hossza*

A függvény értéke megadja az argumentumban szereplő string hosszát. Azt, hogy hány karakterből áll.

LEN ("ABCDEF") függvény értéke 6 lesz.

MID$ (string, kifejezés 1, kifejezés 2) — *string része*

A függvény értéke az argumentumban szereplő string egy része lesz. Az első aritmetikai kifejezés egész értékének megfelelő sorszámú karaktertől kezdődően annyi karakter, amennyi a második aritmetikai kifejezés egész értéke. MID$ ("ABCDEFG", 3, 4) függvény értéke CDEF lesz.

ASC (string) — *string első karakterének kódja*

ASC("ABC") értéke 65 lesz. Ez az A karakter kódja.

● Határozzuk meg egy string karaktereinek kódját!

A stringből ki kell emelnünk a karaktereket. Ezt a MID$ függvénnyel oldhatjuk meg. A karakterek helyét, sorszámát egy ciklusváltozóval jelöljük ki, ami 1—től a string karaktereinek számáig változik, azaz LEN(X$) értékéig. A kódokat ABC(A$) függvény értékének kiírásával kapjuk meg.

```
10 CLS:PRINT " KARAKTEREK ES KODJAIK"
20 REM LEN,MID$, ASC
30  INPUT "KEREM A KARAKTERSOROZATOT";X$
40 PRINT "KARAKTEREK ASCII KODOK"
50 FOR I=1 TO LEN(X$)
60 A$=MID$(X$,I,1)
70 PRINT TAB(4);A$;TAB(16);ASC(A$)
80 NEXT I
```



```
 KARAKTEREK ES KODJAIK
KEREM A KARAKTERSOROZATOT? MISKOLC
KARAKTEREK ASCII KODOK
    M              77
    I              73
    S              83
    K              75
    O              79
    L              76
    C              67
READY
>_
```

O Próbáljuk ki, hogy milyen karaktereket írhatunk a stringekbe!

O Számoljuk össze, hogy egyes karakterek hányszor ismétlődnek egy stringben! Az ismétlések számát tároljuk indexes változóban.

Az ABC függvény inverzének tekinthetjük a CHR$ függvényt, amivel a karakter kódjából a karaktert állíthatjuk elő.

CHR$ (kifejezés) — *kódhoz tartozó karakter*

A függvény értéke az a karakter lesz, amelynek kódja az aritmetikai kifejezés egész értéke. A CHR$(65) függvény értéke az A karakter.

Az argumentumban szereplő kifejezés egész értéke 0 és 255 közötti érték lehet.

● Irassuk ki a kódokhoz tartozó karaktereket!

```
10 REM KODTABLAZAT
20 CLS:FOR I=32 TO 191
30 X$=STRING$(4-LEN(STR$(I)),32)
40 PRINT@Y*64+6*X,X$;I;CHR$(I);
50 Y=Y+1:IF Y=16 THEN Y=0:X=X+1
60 NEXT I
70 IF INKEY$="" THEN 70
```

O Vizsgáljuk meg a „nem megjelenő" karakterek hatását. Ezek a 32—nél kisebb kódszámú karakterek! Hatásukat a Mellékletben közöljük.

O Irassuk ki a grafikus karaktereket!
A legkisebb kódszámú grafikus karakter kódszáma 128.

O Milyen változás történik, ha a BASIC rendszerünket a 12288–as címmel indítjuk? (SYSTEM parancs)

● Bontsunk fel egy mondatot szavakra!

A szavakat a betűköz választja el egymástól, amit kódja alapján azonosíthatunk. Az első megoldásban a MID$ függvénnyel emeljük ki a szavakat alkotó karaktereket és ezeket összegyűjtjük az SZ$ változóban.

```
10 CLS:PRINT "MONDAT SZAVAI - KARAKTEREK OSSZEFUZESE"
20 INPUT "IRJ BE EGY MONDATOT!";M$
30 SZ$=""
40 FOR I=1 TO LEN (M$)
50 SZ$=SZ$+MID$(M$,I,1)
60 IF MID$(M$,I,1)=" " THEN PRINT SZ$:SZ$=""
70 NEXT I
80 PRINT SZ$

MONDAT SZAVAI - KARAKTEREK OSSZEFUZESE
IRJ BE EGY MONDATOT!? TANULOM A BASIC NYELVET
TANULOM
A
BASIC
NYELVET
READY
>_
```

A string elején és végén álló karaktereket a LEFT$ ill. a RIGHT$ függvényekkel választhatjuk le a stringről.

LEFT$ (string, kifejezés) — *string baloldali része*

A függvény értékének megfelelő számú karakter a string elejéről. LEFT$ ("ABCDEF", 3) értéke ABC karaktersorozat lesz.

RIGHT$ (string, kifejezés) — *string jobb oldali része*

A függvény értéke az aritmetikai kifejezés egészértékének megfelelő számú karakter a string végéről.
RIGHT$ ("ABCDEF", 3) értéke DEF.

Az előző feladatot úgy is megoldhatjuk, hogy a szavakat a LEFT$ és RIGHT$ függvényekkel választjuk le a mondatokról.


```
10 REM  LEFT$, RIGHT$
20 CLEAR 500:INPUT "IRJ BE EGY  MONDATOT!";M$
30 L=1
40 FOR J=1 TO LEN(M$)
50 X$=MID$(M$,L,1)
60 IF X$=CHR$(32) THEN PRINT LEFT$(M$,L):
M$=RIGHT$(M$,LEN(M$)-L):L=0
70 L=L+1
80 NEXT J
90 PRINT M$


>RUN
IRJ BE EGY  MONDATOT!? IROK EGY PROGRAMOT
IROK
EGY
PROGRAMOT
READY
>_
```

L változóban számláljuk a mondat elején álló szó karaktereinek számát és a PRINT LEFT$(M$,L) utasítással írjuk ki a szót.
Miután kiírattuk a szót az M$ változó értékét módosítjuk. Csak azokat a szavakat hagyjuk meg, amelyeket még nem irattunk ki. Ezt végzi el az M$ = RIGHT(M$, LEN(M$) – L) utasítás.

● Számítsuk át tízes számrendszerbe egy tíznél kisebb alapú számrendszerben megadott számot!
A feladatot stringfüggvények segítségével oldjuk meg. Ehhez ismernünk kell két újabb függvényt:

STR$ (kifejezés) — *szám átalakítása stringgé*

A függvény értéke egy numerikus string lesz, ami alakilag az aritmetikai kifejezés értékével egyezik meg.
Az STR$(345.6) függvény értéke 345.6 lesz. . Pozitív szám esetén az előjel helyett betűköz karakter van.

VAL (numerikus string) — *string átalakítása számmá*

A függvény értéke a numerikus stringgel megadott szám lesz.
A VAL("123.8") értéke 123.8 lesz.

Írjuk át tizes számrendszerbe 324_5–t. $324_5 = 3 \cdot 5^2 + 2 \cdot 5 + 4$, amit felírhatunk $(3 \cdot 5 + 2) \cdot 5 + 4$ alakban is. Az utóbbi kifejezés értékét a következő algoritmussal számítjuk ki:

Az első számjegyet szorozzuk az alapszámmal és a szorzathoz hozzáadjuk a következő számjegyet. Ezután a kapott összeget szorozzuk az alapszámmal és a szorzathoz hozzáadjuk a következő számjegyet. Ezt így folytathatjuk több számjegy esetén is.

```
10 CLS:PRINT " ATSZAMITAS TIZES SZAMRENDSZERBE"
:REM  VAL,STR$
20 INPUT "0< ALAPSZAM <10";G:INPUT "A SZAM";S
30 S$=STR$(S):S=0
40 N=LEN(S$):IF N=1 THEN 90
50 FOR I=1 TO N-1
60 SJ=VAL(MID$(S$,I,1)):IF SJ>=G THEN 110
70 S=(S+SJ)*G
80 NEXT I
90 S=S+VAL(MID$(S$,N,1))
100 PRINT "TIZES SZAMRENDSZERBEN";S:END
110 PRINT "ILYEN SZAMJEGY NINCS A";G;
" ALAPU SZAMRENDSZERBEN"
```

```
 ATSZAMITAS TIZES SZAMRENDSZERBE
0< ALAPSZAM <10? 8
A SZAM? 1206
TIZES SZAMRENDSZERBEN 646
READY
>_
```

A számjegyek meghatározásának egyik módja, hogy a STR$ függvénnyel az átirandó számot numerikus stringgé alakítjuk és abból a MID$ függvénnyel vesszük ki a számjegyeket.
A számjegyeket a MID$ függvénnyel választjuk ki az S$ változóból és azokat a VAL függvénnyel alakítjuk át számokká, hogy számolni tudjunk velük. Az S változóban gyűjtjük az összeget, amit az alapszám és a számjegyek ismeretében állíthatunk elő.

O Irjunk programot tíznél nagyobb számrendszerben adott szám átírására is!
A kilencnél nagyobb számjegyeket jelöljük az ABC betűivel, és ezek kódjait használjuk a számításhoz!

● Egy darázsfészekben maximálisan 15 darázs tartózkodhat. A darazsak véletlenszerűen ki–be röpdösnek. Határozzuk meg, hogy hány alkalommal volt a fészekben 0, 1, 2, . . . 15 darázs!
Azt tudjuk, hogy minden darázs azonos valószínűséggel röpülhet, ezért, ha kint több darázs van, mint a fészekben, akkor nagyobb a valószínűsége annak, hogy berepül darázs, minthogy ki.
Az A vektorban számláljuk egyes esetek előfordulásának számát. Mivel mindig

egy darázs repülhet ki, vagy be, csak két eset lehetséges: vagy az eggyel kisebb, vagy az eggyel nagyobb indexű elem értékét kell növelni. Ezt úgy érjük el, hogy az I változónak 1, vagy –1 értéket adunk, azt figyelembe véve, hogy D számú darázs van a fészekben a 15–ből. Annak valószínűsége, hogy a következő alkalommal kirepül a darázs D/15 lesz.

```
10 CLEAR 200:DIM A(15):CLS:PRINT"DARAZSAK -
 NORMALELOSZLAS GRAFIKONJA"
20 INPUT "KEZDETBEN MENNYI VAN A FESZEKBEN 15-BOL";D
30 CLS:RANDOM:A(D)=1
40 PRINT@64*D+2,:PRINTUSING"##";D;:PRINT" "
;STRING$(A(D),42);
50 I=SGN(RND(0)-D/15)
80 D=D+I:A(D)=A(D)+1:IF A(D)=60 THEN 100
90 GOTO 40
100 IF INKEY$="" THEN100 ELSE CLS
```

```
 1 *
 2 *
 3 *****
 4 *********
 5 ****************
 6 ******************************
 7 **************************************************
 8 ************************************************************
 9 ****************************************************
10 ************************************
11 ****************
12 ******
13 *
```

Az előfordulások szemléltetésére grafikont is készítünk a STRING$ függvénnyel.

STRING$ (kifejezés, karakter, v. kódja) — *több azonos karakter egy stringben*

A függvény értéke egy string lesz, amiben a megadott, vagy kódjával adott karakter a kifejezés egész értékének megfelelő számban szerepel.
STRING$(5,"X") és STRING$(5,88) függvények értéke egyaránt XXXXX lesz.

12. GRAFIKA

CLS, SET, RESET utasítások és a POINT függvény

A képernyőre 16 sorba, soronként 64 karaktert írhatunk ki. Amint ezt a korábbiakban láttuk, a karakterekből ábrákat alakíthatunk ki a képernyőn. Az ábrákat „fínomíthatjuk", ha *grafikus karaktereket* használunk. Ezeknek a karaktereknek a kódja 128 és 191 közé esik és megjelenítésük a PRINT CHR$(X) utasítással történik.

● Irassuk ki a következő programmal a grafikus karaktereket és azok kódját!

```
10 REM GRAFIKUS KARAKTEREK
20 CLS:FOR X=128 TO 191
30 PRINT@510,CHR$(X),X
40 IF INKEY$="" THEN 40
50 NEXT
```

A grafikus karakterek *egy karakterpozició hat pontjá*ból állnak össze. Ezt a pontmátrixot 3 sor és 2 oszlop pontsor alkotja, melyben a pontok világítanak, vagy nem. Ez azt jelenti, hogy a *képernyőn* 48 pont jeleníthető meg egymás alatt és 128 egymás mellett.
Tekintsük a képernyőt egy olyan koordinátarendszer részének, melyben ábrázolni tudjuk a $0 \leq X \leq 127$, $0 \leq Y \leq 47$ koordinátájú pontokat. Az eltérés a megszokott elrendezéstől csak annyi, hogy az y tengely lefelé mutat. A (0;0) pont a képernyő bal felső sarkában, a (0;47) pont a képernyő bal alsó sarkában jeleníthető meg a SET, vagy törölhető a RESET utasítással.

Grafikus utasítások:

SET (aritmetikai kifejezés, aritmetikai kifejezés) — *pont megjelenítés*
RESET (aritmetikai kifejezés, aritmetikai kifejezés) — *pont törlés*

Az argumentumok egészrésszei adják a pont koordinátáit. Megengedett értékeik, SET(X,Y) vagy RESET(X,Y) utasítások argumentumait figyelembe véve:

$0 \leq X \leq 127$

$0 \leq Y \leq 47$

O Vizsgáljuk meg a pontok elhelyezkedését a képernyőn a következő programmal!

```
10 CLS:PRINT " PONT MEGJELENITES - TORLES"
20 CLEAR 100:PRINT "KEREM A PONT KOORDINATAIT
-0<=X<=127,0<=Y<=47";:INPUT X,Y
30 PRINT "MEGJELENITES(M), VAGY TORLES (T)";
40 IF X<0 OR X>127 OR Y<0 OR Y>47 THEN CLS: END
50  REM MEGJELENITES
60 Q$=INKEY$: IF Q$="" THEN 60 ELSE IFQ$="T" THEN 90
70 CLS: SET(X,Y)
80 FOR I=1 TO 2000: NEXT: GOTO 20
90 REM TORLES
100 CLS
110  REMFOR I=0 TO 47:FOR J=0 TO 127:SET(J,I):NEXT:NEXT
120 FOR I=1 TO 16:PRINT STRING$(64,191);:NEXT
130 PRINT STRING$(63,191);
140 FOR I=45 TO 47:FOR J=126 TO 127:SET(J,I):NEXT:NEXT
150 RESET (X,Y)
160 FOR I=1 TO 2000:NEXT:GOTO 10
```

A koordináták megadása után jelezni kell, hogy megjelenítést, vagy törlést akarunk—e vizsgálni. Ez az M, ill. a T karakter leütésével történik.
A pont törlése előtt a képernyő minden pontját megjelenítjük, hogy világos legyen az egész képernyő. A 191—es kódszámú grafikus karakter a karakterpozició mind a 6 pontját kijelzi. Igy gyorsabb, ezért választottuk ezt a megoldást. A képernyő utolsó karakterpozíciójában történik csak pontonként a megjelenítés (140—es sor), mivel, ha ide karaktert írunk, automatikusan soremelés történik. A 120, 130 és 140—es utasítássorok helyett beírhatjuk a következő utasítássort, és összehasonlíthatjuk a két megoldást:
110 FOR I=0 TO 47:FOR J=0 TO 127:SET(I,J):NEXT:NEXT

● Rajzoljunk a képernyőre és jelenítsük meg a kép negatívját!

```
10 CLS:PRINT "RAJZ ES  NEGATIVJA":PRINT "A J,B,L,F
 BILLENTYUKKEL MOZGATHATJUK A PONTOT":PRINT"A SPACE
BILLENTYUVEL VALTHATUNK RAJZOLASRA,VAGY TORLESRE":
PRINT "A 'V' BILLENTYUVEL JELEZZUK A RAJZOLAS BEFEJEZESET"
20 PRINT"KEZDO POZICIO":INPUT "X=(0-127)";X:
INPUT "Y=(0-47)";Y
30 IF X<0 OR Y<0 OR X>127 OR Y>47 THEN 20
40 CLS:SET(X,Y):I=1
50 X1=X:Y1=Y
60 P$=INKEY$:IF P$="" THEN 60
70 IF P$="V" THEN 150
80 IF P$=" " THEN I=-I :GOTO 60
90 IF P$="F"  THEN Y=Y-1:IF Y=-1 THEN Y=Y+1:GOTO 60
100 IF P$="L" THEN Y=Y+1:IF Y=48 THEN Y=Y-1:GOTO 60
110 IF P$="B" THEN X=X-1:IF X=-1 THEN X=X+1:GOTO 60
120 IF P$="J" THEN X=X+1:IF X=128 THEN X=X-1:GOTO 60
130 IF I=1 THEN SET(X,Y):RESET(X1,Y1) ELSE RESET(X,Y)
:SET(X1,Y1)
140 GOTO 50
150 REM NEGATIV KEP
160 FOR X=0 TO 127
170 FOR Y=0 TO 47
180 IF POINT(X,Y) THEN RESET(X,Y) ELSE SET(X,Y)
190 NEXT Y:NEXT X
200 Q$=INKEY$:IF Q$="" THEN 200
210 CLS
```

Egy pontot kell mozgatni a képernyőn, amivel vagy rajzolhatunk, vagy törölhetünk pontokat.
A pont mozgatását a J (jobb), B (bal), L (le) és F (fel) billentyűk lenyomásával oldjuk meg. A lenyomott karaktereknek megfelelően változtatjuk a pont koordinátáit. Növeljük, vagy csökkentjük a mozgó pont koordinátáit tartalmazó változók értékeit.
Például L hatására Y = Y + 1 utasítással egy ponttal lejjebb kerülünk.
Azt is jeleznünk kell, hogy azt a pontot, ahonnét elmozdultunk törölni kell-e, vagy nem. Ezt egy változó értéke jelzi (1), amit a betűkhöz billentyű lenyomásával módosíthatunk.
Gondolnunk kell arra is, hogy a pont ne kerüljön le a képernyőről.
A rajzolás befejezését a V billentyű lenyomásával jelezhetjük és ezek után végighaladunk a képernyőn, és előállíthatjuk a kép negatívját. A pontok állapotát a POINT *függvény* értéke jelzi, ami lehet –1, vagy 0, aszerint, hogy világít–e a pont vagy sem.

POINT (aritmetikai kifejezés, aritmetikai kifejezés) – *a pont állapota*

Az argumentumokban szereplő aritmetikai kifejezések egészrészei által megadott koordinátájú pont állapotától függően a függvény értéke:

– 1, ha a pont világít,
0, ha a pont nem világít.

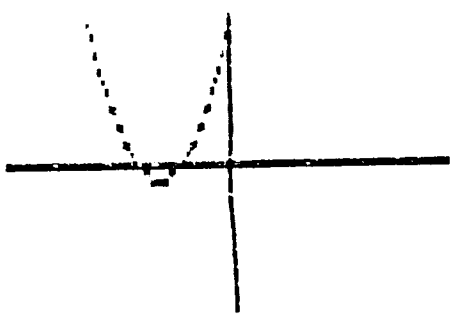
A numerikus értékek egyben logikai értékeket is jelentenek:

– 1 – IGAZ
0 – HAMIS

A pontok negatívját a 180–as sorban állítjuk elő.

● Ábrázoljuk a képernyőn, egy koordináta-rendszerben, a másodfokú függvényt!

```
10 REM MASODFOKU FUGGVENY ABRAZOLASA
20 CLS:PRINT"AZ A*(X-U)^2+V =Y FUGGVENY KIRAJZOLASA"
30 REM  ** PARAMETEREK BEVITELE **
40 INPUT "A=";A:INPUT "U=";U:INPUT "V=";V
50 REM *** A KOORDINATA TENGELYEK KIRAJZOLASA ***
60 CLS:FOR X=0 TO 127:SET(X,24):NEXT X
70 FOR Y=0 TO 47:SET(64,Y):NEXT Y
80 REM  *** FUGGVENY KIRAJZOLAS ***
90 FOR X=0 TO 127
100 FX=X-64:FX=FX/2
110 Y=A*(FX-U)^2+V:Y=-Y/4+24
120 IF Y<0 OR Y>47 THEN 140
130 SET(X,Y)
140 NEXT X
150 Q$=INKEY$:IF Q$="" THEN 150
```



Miután megadjuk az $a(x-u)^2 + v$ függvény paramétereit, kirajzoljuk a koordináta tengelyeket. Az origó a (64,24) pontba kerül, ezért a függvényünket transzformálnunk kell, hogy a szokásos formában kapjuk meg a görbét.
Az X értéke 0–tól 127–ig változik. Ez a képernyő pontjainak abszcisszái. A koordinátarendszerünkben, ha a pontok távolságát egymástól X irányban egységnyinek vesszük, a megjelenítés a –64 abszcisszájú pontokkal kezdődik,

ezért FX = X–64, és az FX–hez tartozó függvényértékeket kell meghatároznunk. X irányban a szomszédos pontok távolsága 2 egység lesz, ha az FX=FX/2 utasítással széthúzzuk a függvényt.
A függvényértékeket az Y = –Y + 24 utasítással vihetjük át koordinátarendszerünkbe, ha egységnyinek választjuk a pontok távolságát.
Legyen két szomszédos pont távolsága az Y tengelyen 4 egység, akkor az Y = –Y/4 + 24 utasítást kell használnunk.

● Határozzuk meg programmal egy adott összetételű grafikus karakter kódszámát!

A grafikus karakter hat képpontjának állapotát megadjuk. I és N billentyűk lenyomásával jelezhetjük, hogy világítson a pont, vagy sem. A pontmátrix pontjainak megadása után a programmal sorban megjelenítjük a grafikus karaktereket és pontonként összehasonlítjuk a megadott karakterekkel. Az azonos karaktereknél megáll a program.

```
10 CLS:PRINT " GRAFIKUS JELEK KODJAI"
20 PRINT "KEREM A GRAFIKUS JEL 3 X 2 -ES MATRIXAT"
30 PRINT"VILAGIT?(I-N)"
40 FOR I=1 TO 3:FOR J=1 TO 2
50 QQ=INKEY$:IF QQ<>"I" AND QQ<>"N" THEN 50
60 IF QQ="I" THEN SET(20+J,20+I) ELSE RESET(20+J,20+I)
70 NEXT J:NEXT I
80 FOR K=128 TO 191
90 PRINT@471,CHR$(K)
100 FOR I=1 TO 3:FOR J=1 TO 2
110 PRINT@384,"A KOD: ";K;
120 IF POINT(20+J,20+I)<>POINT(45+J,20+I) THEN 150
130 NEXT J:NEXT I
140 PRINT@668,:END
150 NEXT K
```

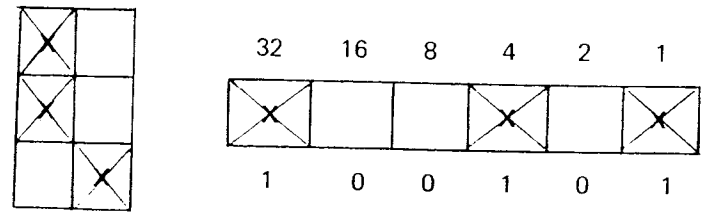
Többször lefuttatva a programot, észrevehetjük, hogy a pontokkal tulajdonképpen binárisan számolunk.
Számozzuk meg a pontokat és helyezzük el azokat egy sorban!

1	2
3	4
5	6

6	5	4	3	2	1

Ebben az elrendezésben már jól látszik a szabály.

Határozzuk meg a következő grafikus karakter kódját!



A világító pontokat 1–el, a sötéteket 0–val jelöltük. Kaptunk egy bináris számot, ami 37–tel egyenlő a tízes számrendszerben. Az első grafikus karakter kódja 128. Ettől számítva a karakterünk a 37. lesz. A kódja tehát 128 + 37 = 165.

13. SZUBRUTIN

GOSUB, ON GOSUB és a RETURN utasítások

Összetett programok írásakor, a feladat elemzésekor alaposan gondoljuk végig, hogy nem bonthatjuk–e fel jól körülhatárolt részekre a feladatot. Ezeket oldjuk meg külön–külön, majd építsük be ezeket a programba.
Egy–egy részfeladat megoldására írjunk *szubrutin*okat:
A szubrutin olyan programrészlet, ami

- önálló feladatot old meg,
- a programból bárhonnét hívható a GOSUB utasítással
- és a RETURN utasítással fejeződik be.

GOSUB cím — *szubrutinhívás*
Az utasítás hatására a program az utasításban szereplő címen folytatódik. „Hívjuk" a szubrutint. A szubrutin végét a RETURN utasítással jelezzük.

RETURN — *visszatérés a szubrutinból*
Hatására ott folytatódik a program, ahonnét a „hívás" történt: a GOSUB utasítást követő utasítással.

Pontokból szakaszokból és ellipszis ívekből alakítsunk ki egy fejet!

A feladat három részfeladatra bontható.
Pont megjelenítése

```
10 INPUT X,Y:CLS
20 GOSUB 100:END
100 REM PONT ABRAZOLASA
110 IF X<0 OR X>127 OR Y<0 OR Y>47 THEN RETURN
120 SET(X,Y):RETURN
```

A szubrutinnak az X és Y változó értékét kell megadni. Ezek a szubrutin *bemenő paraméterei.*
A szubrutin megvizsgálja, hogy megjeleníthető–e a pont, és ha igen, akkor megjeleníti. Ennek megfelelően a szubrutinból két helyről is lehetséges a visszatérés.
Ezt a szubrutint felhasználjuk a *szakasz* pontjainak megjelenítéséhez is.

A szakasz *bemenő paraméterei* a két végpont koordinátái lesznek: X1, Y1, X2, Y2.

A szubrutin által *használt változók:*

YP	–	függőleges szakasz pontjainak ordinátái
YD	–	szakasz meredekségének abszolútértéke
L	–	a pontok kirajzolása melyik végpontnál kezdődik
YV	–	ordináták változása (csökken vagy nő)
K9	–	pontok számlálása

Kimenő paraméterek: X és Y megjelenítendő pont koordinátái.
A szubrutint próbáljuk ki mielőtt beépítenénk egy nagyobb programba!

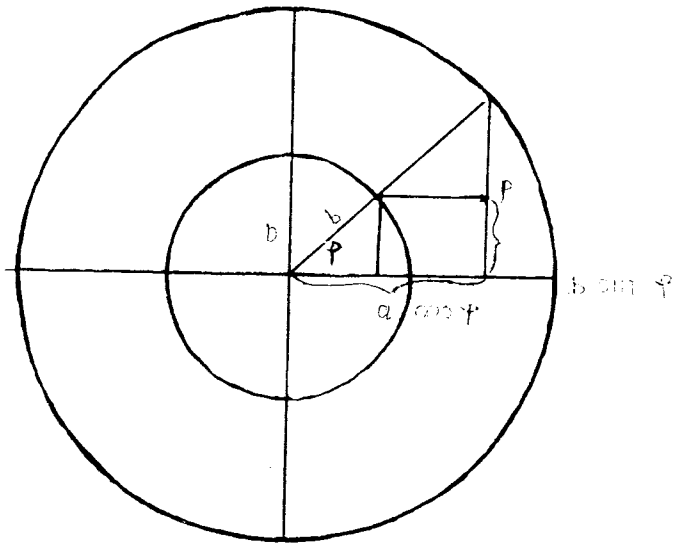
```
10 REM SZAKASZ ABRAZOLASA
30 CLS:INPUT "X1";X1:INPUT "Y1";Y1
40 INPUT "X2";X2:INPUT "Y2";Y2
50 GOSUB 200:END
100 REM PONT ABRAZOLASA
110 IF X<0 OR X>127 OR Y<0 OR Y>47 THEN RETURN
120 SET(X,Y):RETURN
200 REM SZAKASZ ABRAZOLASA
210 IF X1=X2 THEN FOR YP=Y1 TO Y2 STEP SGN(Y2-Y1):
  SET(X1,YP):NEXT :RETURN
220 YD=ABS((Y2-Y1)/(X2-X1)):L=SGN(X2-X1):YV=SGN(Y2-Y1):K9=0
230 FOR PX=X1 TO X2 STEP L
240 X=PX:Y=Y1+YV*K9*YD:GOSUB 100:K9=K9+1
250 NEXT:RETURN
```

Ezek után írjuk meg az *ellipszisek ívét* kirajzoló szubrutint!
A szubrutin bemenő paraméterei:

A	–	ellipszis nagytengelye
B	–	ellipszis kistengelye
U; V	–	a középpont koordinátái
IK	–	az ív kezdete
IV	–	az ív vége

A teljes ívet 100 egységnek vesszük. A Ø pontja az ívnek 3 óra óraállásának felel meg. A teljes ellipszist IK = Ø és IV = 100 értékek esetén rajzolja ki a szubrutin.
A kimenő paraméterek: X és Y a megjelenítendő pont koordinátái.
A program az ellipszis pontjainak szerkesztésén alapszik.
A 0 pont köré a ill. b sugarú köröket rajzolunk, ahol a a nagytengely, b a kistengely. Az O–ból húzott félegyenes és a körök metszéspontjai: Q ill. R.

Ezeken a pontokon keresztül a tengelyekkel párhuzamos egyeneseket raizolunk, és ezek metszéspontja P, az ellipszis egy pontja.



A P pont koordinátáit α, a és b ismeretében a következőképpen számíthatjuk ki, ha a középpont az origóban van:

$x = a \cdot \cos \alpha$ $\qquad$ $y = b \cdot \sin \alpha$

Miután megírtuk és kipróbáltuk ezt a szubrutint is, megírhatjuk a *főprogramot.*
Ebből a programrészből hívjuk a szubrutinokat.

A szubrutinok bemenő paramétereinek értékét a DATA adatlistából olvassuk be.

Átsorszámozva a következőképpen néz ki a programunk:


```
10 REM PONT,PONT VESSZOCSKE
20 CLS
30 READ X,Y:GOSUB 90:READ X,Y:GOSUB 90
40 READ X1,Y1,X2,Y2:GOSUB 120
50 READ A,B,U,V,IK,IV:GOSUB 180
60 READ A,B,U,V,IK,IV:GOSUB 180
70 READ A,B,U,V,IK,IV:GOSUB 180
80 PRINT@866,:END
90 REM PONT ABRAZOLAS
100 IF X<0 OR  X>127 OR Y<0 OR Y>47 THEN RETURN
110 SET(X,Y):RETURN
120 REM SZAKASZ ABRAZOLAS
130 IF X1=X2 THEN FOR YP=Y1 TO Y2 STEP SGN(Y2-Y1):
SET(X1,YP):NEXT YP:RETURN
140 YD=ABS((Y2-Y1)/(X2-X1)):L=SGN(X2-X1):YV=SGN(Y2-Y1)
:K9=0
150 FOR PX=X1TO X2 STEP L
160 X=PX:Y=Y1+YV*K9*YD:GOSUB 90:K=K+1
170 RETURN
180 REM ELLIPSZIS
190 PI=3.14:IK=IK/100*2*PI:IV=IV/100*2*PI
200 FOR FI=IK TO IV STEP SGN(IV-IK)*PI/50
210 X=U+A*COS(FI):Y=V-B*SIN(FI):GOSUB 90
220 NEXT FI
230 RETURN
240 REM ADATOK
250 DATA 40,17,60,17
260 DATA 50,20,50,25
270 DATA 20,20,50,20,50,100
280 DATA 20,8,50,20,0,50
290 DATA 6,3,50,30,90,60
```

READY

Több szubrutin közül az ON GOSUB utasítással választhatjuk ki a megfelelőt.

ON aritmetikai kifejezés GOSUB sorszámlista — elágazó szubrutinhívás

Hatása: hívja azt a szubrutin, amelyet az aritmetikai kifejezés egészrésze a sorszámlistából kijelöl. A sorszámlistában a szubrutinok kezdő sorszámait adjuk meg.

Amennyiben az aritmetikai kifejezés egészrésze ϕ, vagy nagyobb, mint ahány sorszám szerepel a listában, az ON–GOSUB utasítást követő utasítás végrehajtásával folytatódik a program.

O Készítsünk öröknaptárt a következő szubrutin felhasználásával!

```
100 REM OROKNAPTAR SZUBRUTIN
110 REM PARAMETEREK: E=EV , H=HONAP , N=NAP
120 IF H<3 THEN H=H+12:E=E-1
130 A=E+H+N+INT(8*(H+1)/5)
140 A=A+INT(E/4)-INT(E/100)+INT(E/400)
150 M=A-7*INT(A/7)
160 ON M+1 GOSUB 180,190,200,210,220,230,240
170 RETURN
180 PRINT "VASARNAP":RETURN
190 PRINT "HETFO":RETURN
200 PRINT "KEDD":RETURN
210 PRINT "SZERDA":RETURN
220 PRINT "CSUTORTOK":RETURN
230 PRINT "PENTEK":RETURN
240 PRINT "SZOMBAT":RETURN
```

● A következő példaprogramunk egy *stringfüggvény*t mutat be, amit szubrutinnal valósítunk meg. A szubrutin bemenő paraméterei M$ és B$. A szubrutinnal azt határozzuk meg, hogy B$ *megtalálható–e* M$*–ben*, azaz a B$ értékének megadott karaktersorozatot tartalmazza–e M$.
A szubrutin két kimenő paramétere S és L. Akkor, ha talált M$–ben B$ értékével azonos karaktersorozatot, az S értéke 1 lesz, egyébként 0.
Az L értéke azt jelzi, hogy hányadik karaktertől kezdődik az azonosság.

```
10 CLS: PRINT "STRING KERESES STRINGBEN"
20 INPUT "AMELYBEN KERESSUK";M$
30 INPUT "AMIT KERESUNK";B$
40 GOSUB 80
50 PRINT M$;"-BEN";
60 IF S=1 THEN PRINT L;"  KARAKTERTOL"; ELSE
  PRINT "  NEM ";
70 PRINT "TALALHATO ";B$:END
80 REM KERESO SZUBRUTIN
90 S=0:FOR L=1  TO LEN(M$)-LEN(B$)+1:
IF  MID$(M$,L,LEN(B$))=B$ THEN S=1:RETURN
100 NEXT L: RETURN
```

```
STRING KERESES STRINGBEN
AMELYBEN KERESSUK? BUDAPEST
AMIT KERESUNK? EST
BUDAPEST-BEN 6   KARAKTERTOLTALALHATO EST
READY
>_
```


14. ADATTÁROLÁS MÁGNESSZALAGON

A programok mágnesszalagon történő tárolásáról már volt szó. A CSAVE utasítással írattuk ki a programot és a CLOAD utasítással olvassuk be.
BASIC programnyelven a PRINT utasítással adhatunk parancsot az adatkiírásra és az INPUT utasítással olvashatunk be adatokat. Az utasításban jeleznünk kell azt is, hogy melyik magnetofont használjuk. Ennek jelzése a CSAVE és a CLOAD utasításoknál használtakkal azonos. Ezután kell írnunk a változólistát.

Beépített magnetofon esetében:

```
PRINT# –1,A,X$
INPUT# –1,A,X$
```

Csatlakoztatott magnetofon használatakor:

```
PRINT# –2,K(I),Y
INPUT# –2,K(I),Y
```

A PRINT # –1 utasítás végrehajtása előtt nekünk kell gondoskodnunk, hogy a szalagot üres helyre állítsuk és a magnetofont felvételre kapcsoljuk. Az F1 billentyű legyen felengedett állapotban (ne égjen a lámpa)!
A PRINT # –1 utasítás végrehajtásakor a magnetofon bekapcsol és az *adat a mágnesszalagra kiíródik*. Az adatlista végére érve a magnetofon kikapcsol, majd a következő kiírásnál újból bekapcsol. A kiírás akkor is megtörténik, ha nincs felvételre állítva a megnetofon. Ilyenkor adatokat vesztünk.
Egy utasítással kiírt adatok előtt mindig egy kis üres rész marad ki, ezért lehetőleg egy utasítással több adatot irassunk ki. Így gyorsíthatjuk a kiírást, ill. a beolvasást.
A *beolvasáskor* nekünk kell gondoskodnunk arról, hogy a szalagot oda állítsuk, ahol a megfelelő adat van. Azt olvassa be, amit talál, és addig vár az olvasással, ameddig be nem kapcsoljuk lejátszásra a magnetofont.
A beolvasáskor fellépő hibákat az ON ERROR GOTO és a RESUME utasításokkal szűrjük ki.
Gyorsíthatjuk a programunkat, ha az adatokat stringekben írjuk ki. Ezt mutatjuk be a következő programunkkal. A képernyő tartalmát íratjuk ki és olvassuk be.

```
10 CLS:CLEAR 300:PRINTTAB(12);"RAJZOLAS KEPERNYORE -
 TAROLAS MAGNON"
20 PRINT "A PONTOT A 'NYIL' BILLENTYUKKEL
 MOZGATHATJUK":PRINT "RAJZOLAS  - TORLES VALTAS A
 SPACE BILLENTYUVEL TORTENIK"
30 PRINT"A RAJZOLAS BEFEJEZESET A 'V' BILLENTYUVEL
 JELEZZUK - ELOTTE       KAPCSOLD FELVETELRE A MAGNOT!"
40 X=0:Y=0:INPUT "HANYADIK SORTOL-SORIG(0-47)
 TAROLJUK";K,L:INPUT "HANYADIK OSZLOPTOL-OSZLOPIG(0-127)
TAROLJUK";M,N
50 CLS:SET(X,Y):I=1
60 X1=X:Y1=Y
70 P$=INKEY$:IF P$="" THEN 70
80 IF ASC(P$)=86 THEN 160
90 IF ASC(P$)=32 THEN I=-I:GOTO 70
100 IF ASC(P$)=91 AND Y<>0 THEN Y=Y-1
110 IF ASC(P$)=10 AND Y<>47 THEN Y=Y+1
120 IF ASC(P$)=8 AND X<>0 THEN X=X-1
130 IF ASC(P$)=9 AND X<>127 THEN X=X+1
140 IF I=1 THEN SET(X,Y):RESET (X1,Y1) ELSE
 RESET(X,Y):SET(X1,Y1)
150 GOTO 60
160 REM KIIRATAS
170 IF INKEY$="" THEN 170 ELSE PRINT#-1,K,L,M,N
180 FOR I=K TO L
190 T$="":FOR J=M TO N
200 IF POINT(J,I) THEN X$="1" ELSE X$="0"
210 T$=T$+X$:RESET (J,I)
220 NEXT J:PRINT#-1,T$:NEXT I
230 REM BEOLVASAS
240 CLS:PRINT "ADATKIIRAS KESZ!"
250 PRINT "CSEVELD VISSZA A MAGNOT,ES NYOMJ LE
 EGY BILLENTYUT!":CLEAR
260 PRINT "ALLIITSD LEJATSZASRA A MAGNOT,ES NYOMJ LE
 EGY BILLLENBTYUT!":CLEAR
270 ON ERROR GOTO 360
280 IF INKEY$="" THEN 280 ELSE INPUT#-1,K,L,M,N
290 CLS:FOR I=0 TO L-K
300 IF INKEY$<>"" THEN CLS:GOTO 250
310 INPUT#-1,T$
320 FOR J=M TO N
330 IF MID$(T$,J+1,1)="1" THEN SET(M+J,K+I)
340 NEXT:NEXT
350 IF INKEY$="" THEN 350 ELSE END
360 :RESUME 250
370 PRINT"CSEVELD VISSZA A SZALAGOT AZ ADATSOR
 ELEJERE!"
```

A képernyőre ábrát rajzolhatunk a pont mozgatásával. A SPACE billentyűvel jelezhetjük, hogy a pont „nyomot hagyjon", vagy törölje azokat a pontokat, ahol volt. A rajz befejezését a „V" billentyű leütésével jelezzük, miután a magnetofont felvételre állítottuk.

A 200–as utasítássorban vizsgáljuk a képpontok állapotát, és ezt 1, ill. 0 karakterekkel jelezzük, amelyeket a T$ változóban gyűjtünk össze (210–es sor). A képsor végére érve kiíratjuk a mágnesszalagra a T$ változó értékét (220–as sor). Az így kivitt adatokat olvashatjuk be, és jelentetjük meg az ábrát, ha a beolvasás megkezdése előtt az adatsor elejére állítjuk a szalagot, és lejátszásra kapcsoljuk a magnetofont.

A 310–es sorban olvassuk be a képpontok állapotát jelző karakterlánc–változó értékét. A J ciklusváltozójú ciklusban bontjuk karaktereire, és ha a karakter „1", akkor megjelenítjük a pontot.

15. HANGGENERÁTOR

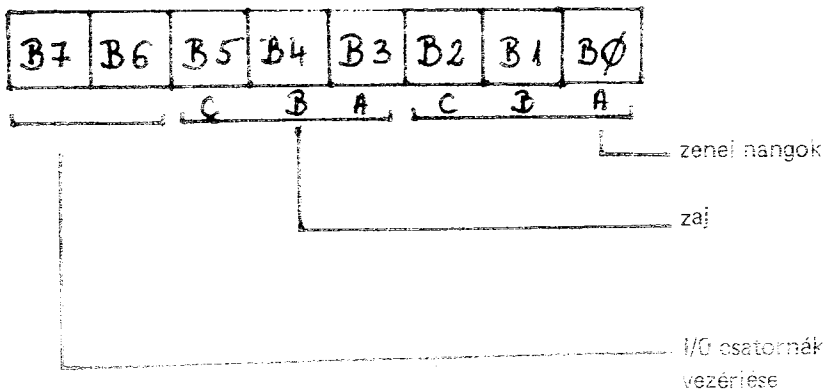
Gépek zajából, hangjából sokmindent megtudhatunk. Füllel figyelhetjük, hogy hogyan dolgoznak. Figyelemmel kísérhetjük, hogy mit csinál a gép és észrevehetjük, ha rendellenesen működik.
A számítógépünkbe hanggenerátort építettek be, melynek működtetésével tudatosan élhetünk azzal a lehetőséggel, hogy hangjelek útján is informálódjon a használó, az, aki a programot működteti. A programba beírt utasítások, melyekkel hangokat állítunk elő (generálunk), jelzik egy–egy műveletsor végrehajtását, valamilyen esemény megtörténtét. A hangjelekkel felhívhatjuk a használó figyelmét a szükséges műveletek végrehajtására.
A hanggenerátor elsődleges szerepe ez, de zene megszólaltatására is lehetőség van, valamint a játékokat is „élőbbé" lehet tenni.
A *hanggenerátor vezérlését* 14 db 8 bites regiszter (tároló) végzi. A regiszterek tartalma határozza meg, hogy mi történik.
15. és a 16. regiszter a *digitális input–output regiszterek*. 8 bites digitális jel beolvasása történhet valamilyen perifériáról, ill. arra történhet kivitel.
Az OUT 31,R *utasítással kijelöljük* azt a *regisztert*, amelyiknek az értékét módosítani akarjuk. Az R értéke 0–tól 13–ig változhat, ha a hanggenerátort akarjuk működtetni. Az OUT 30,T *utasítással megadjuk* az előzőleg kijelölt *regiszter értékét*. A T lehetséges értékeit, azaz egyes regiszterek értékeit, valamint a regiszterek szerepét a következő táblázatban adjuk meg:

regiszterek	funkciója	lehetséges értékük
	„A" hanggenerátor periódusidejének	
RØ	– fínom beállítása	Ø – 255
R1	– durva beállítása	Ø – 15
	„B" hanggenerátor periódusidejének	
R2	– fínom beállítása	Ø – 255
R3	– durva beállítása	Ø – 15
	„C" hanggenerátor periódusidejének	
R4	– fínom beállítása	Ø – 255
R5	– durva beállítása	Ø – 15
R6	Zajgenerátor periódusidejének beállítása	Ø – 31
R7	Keverő	Ø – 255

regiszterek	funkciója	lehetséges értékük
R8	„A" hanggenerátor hangereje	Ø – 15
	– időben változó hangerősség	16
R9	„B" hanggenerátor hangereje	Ø – 15
	– időben változó hangerősség	16
R1Ø	„C" hanggenerátor hangereje	Ø – 15
	– időben változó hangerősség	16
	Időben változó hangerősség periódusidejének	
R11	– fínom beállítása	Ø – 255
R12	– durva beállítása	Ø – 255
R13	Hangerősség időbeni változásának módja	Ø – 15
R14	I/O csatorna	
R15	I/O csatorna	

Három hanggenerátorunk van, melyek mindegyikével előállíthatunk állandó, vagy időben szabályosan változó zenei hangokat. A zajt a zajgenerátor szolgáltatja.
A következő négy OUT utasítással már megszólaltathatjuk az „A" hangot:
OUT 31,Ø:OUT 30,241:OUT 31,8:OUT 30,15:OUT 31,7:OUT 30,62
Az „A" hanggenerátor periódusidejét a fínom beállító 241–es értékével adtuk meg, a hangerőt 15–ös szintre állítottuk. Ezeket az értékeket a Ø–ás és a 8–as regiszterbe helyeztük el.
A 7–es regiszterbe 62–t írtunk és ezzel az „A" zenei csatornán az „A" hanggenerátor hangját engedélyeztük.
A hang megszüntetése a regiszterek nullázásával, vagy a (számítógépünk hátoldalán található) RESET nyomógomb benyomásával történhet. Az utóbbit mindig használhatjuk, ha a regisztereket alaphelyzetbe akarjuk állítani.
A *7–es regiszter a keverő* szerepét tölti be. A regiszter bitjeinek az értékétől függ, hogy engedélyezzük, vagy letiltjuk az egyes csatornákon a zenei hangokat, vagy a zajt.

A bitek szerepe:



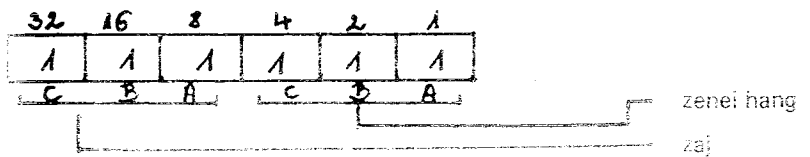
A 6 kisebb helyiértékű bitek értékével vezéreljük a hanggenerátort, a B6 és a B7 bitekkel az input—output csatornákat. Amennyiben az utóbbiak értéke 1 az R14 és az R15-ös regiszterek *bemenetként*, ha az értékük 1, akkor *kimenetként* működnek.
A hanggenerátor vezérlése úgy történik, hogy ha a bit értéke 0, akkor *engedélyezi* az ábrán megadott csatornákon a zenei hangot, vagy zajt, ha 1, akkor *letiltja*. A hanggenerálást a B6 és a B7 bitek értéke nem befolyásolja.
Az előzőek alapján a 7—es regiszter értékeitől függően a következők történnek:

R7 = 63 minden csatornát letiltunk
R7 = 62 „A" csatornán engedélyezzük a zenei hangot
R7 = 55 „A" csatornán engedélyezzük a zajt
R7 = 60 „A" és „B" csatornán engedélyezzük a zenei hangokat
R7 = 56 mindhárom csatornán engedélyezzük a zenei hangot, stb.

Hangot természetesen csak akkor hallunk, ha a hanggenerátorokon beállítjuk a hangokat és megadjuk a hangerősséget.
A következő program segítségével megvizsgálhatjuk a keverő működését:

```
10 REM KEVERO
20 DATA 162,1,117,1,76,1,20,63,15,10,16,0,27,14,0,0
30 FOR I=0 TO 15:OUT 31,I:READ IJ: OUT 30,IJ:NEXT
40 CLS:INPUT "A KEVERO REGISZTER ERTEKE(0-63)";KE
50 OUT 31,7:OUT 30,KE:REM KEVERO BEALLITASA
60 FOR I=1 TO 5000:NEXT
70 OUT 31,7:OUT 30,63:REM LETILTAS
80 GOTO 40
```


Az „A", „B" és „C" regiszterekbe „0", „8", és „F" hangerőt írjuk.
Az „A" és „B" hanggenerátorok hangereje különböző lesz: 15, 10. A „C" hanggenerátor időben változó erősségű hangot állít elő (R10 = 16.). A változás periódusideje 1 mp lesz (R11 = Ø, R12 = 27).
A zajgenerátor periódusideje 20. (R6 = 20)
A program kezdetekor a keverőn letiltjuk a hangot (R7 = 63) és ezt tesszük a 70—es sorban is, amikor kilépünk a várakozási ciklusból (60—as sor).
A keverő értékének beállításához egy kis segítség:



Letiltáskor mind a hat bitben 1 van, azaz a regiszter tartalma 63. Ebből kell levonni annak a bitnek a helyiértékét, amelybe azt szeretnénk, ha 0 kerülne, azaz engedélyezzük a megfelelő hangot.

Hangok előállítása (generálása) azt jelenti, hogy

zenei hangoknál megadjuk:
- a periódusidőt (RØ, R1, R2, R3, R4, R5)
- a hangerőt (R8, R9, R1Ø)

zajnál:
- a periódusidőt (R6)
- a hangerőt (R8, R9, R1Ø)

hangerő szabályos változásánál:
- a periódusidőt (R11, R12)
- a változás módját (R13)

Hangok előállításához az is hozzátartozik, hogy megadjuk meddig hallatszon a hang. Ezt a hang engedélyezése és letiltása közé írt *várakozási ciklus* ciklusszámával állíthatjuk be.

A következő program alkalmas arra, hogy könnyen kipróbáljuk a hanggenerátort. A regiszterek értékét és a ciklusszámot változtatva különböző hangokat állíthatunk elő, így megtervezhetjük programjainkhoz a hangeffektusokat.

```
10 REM REGISZTEREK VIZSGALATA
20 DIM RM$(13),R(13):RM$(0)="A FINOM":RM$(1)=
"A DURVA":RM$(2)="B FINOM":RM$(3)="B DURVA":
RM$(4)="C FINOM":RM$(5)="C DURVA"
30 RM$*6)="ZAJGENERATOR":RM$(7)="KEVERO":RM$(8)=
"A HANGERO":RM$(9)="B HANGERO":RM$(10)="C HANGERO"
40 RM$(11)="HANGERO VALTOZAS - FINOM":RMG$(12)="HANGERO
 VALTOZAS - DURVA"
50 RM$(13)="HANGERO VALTOZAS TIPUSA"
60 CLS:GOSUB 150:GOSUB 160
70 PRINT£14*64,CHR$(30);"REGISZTER(R3) V.
 CIKLUSSZAM(N100)";:INPUT BE$
80 IF ASC(BE$)=62 THEN I=VAL(RIGHT$(BE$,LEN(BE$)-1))
 ELSE 110
90 PRINT£14*64,CHR$(30);"R";I;"REGISZTER TARTALMA";
:INPUT T:R(I)=T
100 GOSUB 160:GOTO 70
110 IF ASC(BE$)=78 THEN N=VAL(RIGHT$(BE$,LEN(BE$)-1))
 ELSE 70
120 FOR K=0 TO 13:OUT 31,K:OUT 30,R(K):NEXT
130 FOR J=1 TO N:NEXT
140 GOSUB 150:GOTO 70
150 FOR RE=0 TO 15:OUT 31,RE:OUT 30,0:NEXT:RETURN
160 REM REGISZTEREK TARTALMANAK KIIRASA
170 CLS:FOR J=0 TO 13:PRINT£J*64,"R";J;TAB(10);RM$(J)
;TAB(45);R(J);:NEXT
180 RETURN
```

```
R 0       A FINOM                          171
R 1       A DURVA                          0
R 2       B FINOM                          0
R 3       B DURVA                          0
R 4       C FINOM                          0
R 5       C DURVA                          0
R 6       ZAJGENERATOR                     0
R 7       KEVERO                           58
R 8       A HANGERO                        15
R 9       B HANGERO                        0
R 10      C HANGERO                        0
R 11      HANGERO VALTOZAS - FINOM         0
R 12      HANERO VALTOZAS - DURVA          0
R 13      HANGERO VALTOZAS TIPUSA          0
REGISZTER(R3) V. CIKLUSSZAM(N100)? _
```

Indítsuk el a programot, és adjuk meg, hogy melyik regiszternek az értékét akarjuk módosítani. Például: R7 begépelésével a 7–es regisztert kijelöltük és ekkor be kell gépelni a regiszter új értékét. Azt, hogy milyen értékeket adhatunk meg, azt a táblázatból kinézhetjük.

A hang időtartamát a következőképpen adhatjuk meg: N2000. Ekkor a várakozó ciklus ciklusszáma 2000 lesz.
Ezzel a programmal is kipróbálhatjuk a keverőt. Irjuk be a regiszterbe az előző program DATA adatlistájában szereplő értékeket, és a 7–es regiszter értékét változtassuk.

A hangmagasság vizsgálata:
R7 = 62, R8 = 15, és változtassuk az R0 és R1 regiszterek értékét!

A hangerősség vizsgálata:
R0 = 100, R7 = 62 és változtassuk az R8 regiszter értékét!

A zajgenerátor vizsgálata:
R7 = 55, R8 = 15 és változtassuk az R6 regiszter értékét!

A hangerő változás periódusidejének vizsgálata:

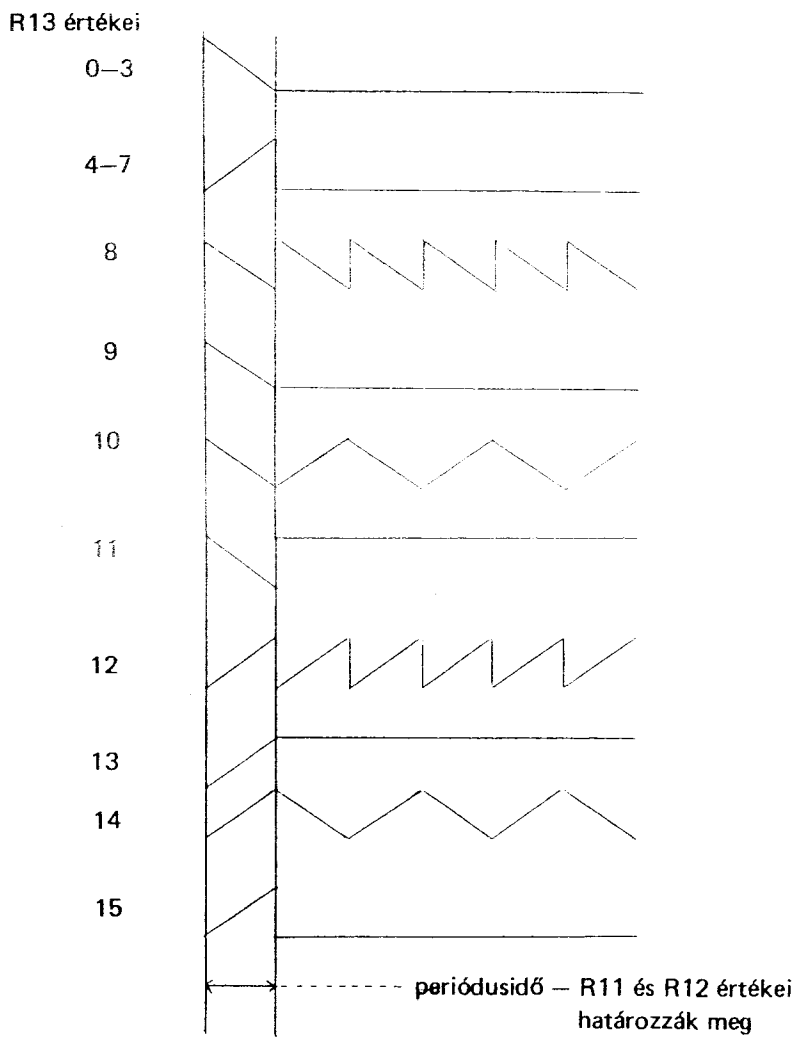
zenei hanggal R0 = 100, R7 = 62, R8 = 16
zajjal R6 = 20, R7 = 55, R8 = 16 és változtassuk az R11 és az R12 regiszterek értékét.

A hangerő változás időbeni lefolyásának vizsgálata:

zenei hanggal
R0 = 100, R7 = 62, R8 = 16, R12 = 100

zajjal
R6 = 20, R7 = 55, R8 = 16, R12 = 100 és
változtassuk az R13 regiszter értékét.

IDŐBENI VÁLTOZÁS



A *zenei programok* készítéséhez nyújthat segítséget a következő két program. Az elsővel *skálázhatunk*, a másodikkal egy *oktávval* változtathatjuk a kezdő hang magasságát.

```
10 CLS:PRINT " SKALAZAS"
20 INPUT " A HANGOK IDEJE";N
25 PRINT" A LEGFELSO BILLENTYUSORT HASZNALD"
30 OUT 31,8:OUT 30,15:REM A CSATORNA HANGEREJE
40 OUT 31,7:OUT 30,62:REM CSATORNA KIVALASZTAS A
 KEVERON -(A)
50 DIM HF(12),HD(12)
60 REM ALAPHANGOK ERTEKEI
70 FOR I=0 TO 11:READ HD(I),HF(I):NEXT I
80 REM BILLENTYUZET LEKERDEZESE
90 HH$=INKEY$:IF HH$="" THEN 90
100 IF (ASC(HH$)<58 AND ASC(HH$)>47) OR HH$=":"
 OR HH$="-" THEN 110 ELSE END
110 IF HH$=":" THEN H=10:GOTO 140
120 IF HH$="-" THEN H=11:GOTO 140
130 H=ASC(HH$)-48
140 OUT 31,0:OUT 30,HF(H):OUT 31,1:OUT 30,HD(H)
150 FOR I=1 TO N:NEXT I
160 OUT 31,0:OUT 30,0:OUT 31,1:OUT 30,0
170 GOTO 90
180 REM ZENEI HANGOK - A,C,CISZ,D,DISZ,E,F,FISZ,G,
GISZ,AISZ,H
190 DATA 0,241,1,162,1,139,1,117,1,96,1,76,1,58,1,
40,1,23,1,8,0,235,0,222
```

```
10 CLS:PRINT "OKTAV"
20 OUT 31,7:OUT 30,63:REM HANG LETILTAS
30 OUT 31,8:OUT 30,15:REM A CSATORNA HANGEREJE
40 DB=0
50 INPUT "DURVA BEALLITO=0,FINOM BEALLITO";HA:FB=HA
60 OUT 31,7:OUT 30,62:REM AZ A CSATORNA KIVALASZTASA
70 REM HANG BEALLITAS - A CSATORNAN
80 OUT 31,0:OUT 30,FB:OUT 31,1:OUT 30,DB
90 PRINT "DURVA BEALLITO";DB,"FINOM BEALLITO";FB
100 FOR I=1 TO 500:NEXT I
110 HA=2*HA:DB=INT(HA/256):FB=HA-DB*256
120 IF DB<256 THEN 80
130 OUT 31,8:OUT30,0
```

Végül azoknak egy kis segítség, akik a I/O csatornákat akarják használni. A következő programmal a R14–ről olvassunk be adatot és a beolvasott adat értékének kétszeresét visszük az R15–re. Az R7 regiszter B6 és B7 bitjeinek

értéke határozza meg, hogy az R14 és az R15 regisztereket mire akarjuk használni, ezért először az R7 értékét adjuk meg.
Az adatbeolvasás az INP utasítással történik abból a regiszterből, amelyet előzőleg kijelöltünk. A beolvasott adattal végrehajtjuk a műveletet, és az eredményt rávisszük az output csatornára.

```
10 REM I/O CSATORNAK
20 REM R14-INPUT;R15-OUTPUT
30 OUT 31,7:OUT 30,64
40 REM ADATBEOLVASAS
50 OUT 31,14:A=INP(31)
60 A=2*A:REM MUVELET
70 REM ADATKIVITEL
80 OUT 31,15:OUT 30,A
990 GOTO 40
```


16. MEMÓRIA

PEEK, POKE, MEM, VARPTR és USR *utasítások*

A számítógép működésének megértéséhez közelebb jutunk, ha megismerkedünk a memória felépítésével, funkcióival.

A memória:

- tárolja a BASIC interpretert,
- a BASIC programunkat,
- a BASIC programban használt változók nevét és értékeit,
- a képernyőn megjelenő karakterek kódjait,
- a gépi nyelven írt programokat, szubrutinokat,
- és jelzi a billentyűzet állapotát.

A memória 8 számjegyű bináris számok tárolására szolgáló *rekeszekből* áll. Egy rekesz 8 bitet, azaz 1 byte (bájt)—ot tud tárolni. A rekeszekre címükkel lehet hivatkozni. A *cím*, a rekesz sorszámát jelenti, amit 2 byte—tal adhatunk meg. Binárisan a rekeszek címei 0 és 1111 1111 1111 1111 közötti értékek lehetnek. Ez 64 . 2^{10} rekesznek felel meg, azaz 64 K—nak. A jelenlegi kiépítettsége 32 K. Ennyi a memóriánk kapacitása.
A bináris számábrázolás helyett a 16—os (hexadecimális) vagy a 10—es (decimális) számrendszert használjuk. A bináris számokat úgy alakítjuk át könnyen hexadecimális számokká, hogy a legkisebb helyiértékű számtól kezdődően négyes csoportokra osztjuk a számjegyeket és úgy olvassuk le az értéküket:

bináris alak:	1101	1001	1110	0011
decimális értékük:	13	9	14	3
hexadecimális alak:	D	9	E	3

A két decimális számjeggyel felírható hexadecimális „számjegyek" jelölése:

10 — A
11 — B
12 — C
13 — D
14 — E
15 — F

Hexadecimális szám bináris alakba történő átírásakor ezt az eljárást kell visszafelé követni. A későbbiek során a memória címekre decimálisan és hexadecimálisan is szükségünk lehet, ezért a *memória felosztását*, azt, hogy milyen funkciót lát el a memória egy–egy területe, decimálisan és hexadecimálisan is megadjuk. A tárterületeket a legkisebb és a legnagyobb címekkel jelezzük.

címek hexadecimális	decimális	funkció
0000	0	BASIC interpreter tárolása
2FFF	12287	Csak olvasható tár (ROM)
3000	12288	BASIC kiterjesztés és a monitor program
37FF	14335	Csak olvasható tár (EPROM)
3800	14336	Billentyűzet állapotának jelzése
3BFF	15359	
3C00	15360	Képernyőn megjelenő karakterek kódjainak tárolása
3FFF	16383	
4000	16384	BASIC programok, változók értékei, gépi nyelvű szubrutinok tárolása (RAM)
7FFF	32767	
8000	32768	Bővítési lehetőség (RAM)
FFFF	65535	

BASIC programból a PEEK függvénnyel és a POKE utasítással érhetjük el a memória rekeszeket.

PEEK (cím) — *kiolvasás a memóriából*

Az argumentumban megadott című rekesz tartalmát kapjuk meg. A címet decimálisan kell megadni.

PRINT PEEK (12900) — kiírja az 12900–as című rekesz tartalmának decimális értékét.

A = PEEK (4785) — A változó értékeként tároljuk a 4785–ös című rekesz tartalmát.

POKE adat, cím — *beírás a memóriába*

Az utasításban szereplő adatot beírja a megadott címre. A címet és az adatot decimális alakban kell megadni.

POKE 15480,65 — A 15480–as című rekesz tartalma 65 lesz.

● A memória vizsgálatához írjunk egy programot, amivel megadott címtől kezdődően kiírathatjuk a rekeszek tartalmát. A kezdőcímet lehessen megadni decimálisan és hexadecimálisan is és a rekeszek címeit és tartalmát jelenítsük meg mindkét alakban!

A programunkban szükség lesz decimális–hexadecimális és hexadecimális–decimális átalakításra is. Kezdjük ezért a feladatot a két szubrutin megírásával!

A szubrutinokhoz írtunk egy rövid *főprogramot*, hogy kipróbálhassuk: jól működik–e a szubrutin.

A szubrutinok elé írt *magyarázatok* nemcsak a szubrutin megértését, hanem a szubrutin alkalmazását is megkönnyítik.

```
10 CLS:PRINT"MEXADECIMALIS SZAM ATALAKITASA DECIMALIS
 SZAMMA"
20 PRINT"HEXADECIMALIS SZAMJEGYEK:1,2,3,4,5,6,7,8
,9,A,B,C,D,E,F"
30 INPUT"HEXADECIMALIS SZAM";T$
40 GOSUB 60
50 PRINT"DECIMALIS ALAKJA: ";T:END
60 REM H - D  SZUBRUTIN
70 REM BEMENO PARAMETER:T$
80 REMSZUBRUTIN VALTOZOI:I,K,J$
90 REM KIMENO PARAMETER:T
100 K=LEN(T$):T=0:FOR I=1 TO K:J$=MID$(T$,I,1)
110 IF ASC(J$)>47 AND ASC(J$)<58 THEN J=VAL(J$)
120 IF ASC(J$)>64 AND ASC(J$)<71 THEN J=ASC(J$)-55
130 T=T+J*16^(K-I):NEXT
140 RETURN

MEXADECIMALIS SZAM ATALAKITASA DECIMALIS
 SZAMMA
HEXADECIMALIS SZAMJEGYEK:1,2,3,4,5,6,7,8
,9,A,B,C,D,E,F
HEXADECIMALIS SZAM? A8F2
DECIMALIS ALAKJA:  43250
READY
>_
```

A hexadecimális számjegyek miatt mindenképpen string változóban kell megadnunk a hexadecimális számot:T$. A számjegyek leválasztása a számról a 100–as utasítássorban történik. A számjegyek kódjainak vizsgálatával átírjuk tízes számrendszerbe (110–es és 120–as sorok), hogy végül a megfelelő helyiérték szerint összegezzük (130–as sor).

```
10 CLS:PRINT"DACIMALIS SZAM ATALAKITASA HEXADECIMALIS
 ALAKBA"
20 INPUT "DECIMALIS SZAM";DS
30 GOSUB 60
40 PRINT"HEXADECIMALIS SZAM:";HS$
50 END
60 REM D-H SZUBRUTIN
70 REM BEMENO PARAMETER:DS
80 REM SZUBRUTIN VALTOZOK:HH,HS
90 REM KIIMENO PARAMETER:HS$
100 HS$=""
110 HH=INT(DS/16):HS=DS-HH*16
120 IF HS<10 THEN HS$=CHR$(HS+48)+HS$ ELSE
 HS$=CHR$(HS+55)+HS$
130 DS=HH:IF HH<>0 THEN 110 ELSE  RETURN

DACIMALIS SZAM ATALAKITASA HEXADECIMALIS
 ALAKBA
DECIMALIS SZAM? 32194
HEXADECIMALIS SZAM:7DC2
READY
>_
```

A hexadecimális számjegyeket a legkisebb helyiértéktől kezdődően úgy kapjuk meg, hogy 16–al osztjuk a decimális számot (DS) és a maradék lesz a hexadecimális számjegy. A számjegyek nagyságát vizsgálva kódoljuk a két decimális számjeggyel felírható hexadecimális számokat (120–as sor).
A továbbiakban a kapott hányadossal végezzük el a maradékos osztást 16–al és megkapjuk a következő számjegyet, amit a HS$ változóhoz láncolunk. Ezt a két szubrutint építjük be programunkba, mellyel a memória vizsgálatát végezzük.


```
10 REM MEMORIA VIZSGALATA
20 CLEAR 1000:DIM A$(10):CLS
30 PRINT"D ILL. H BETUVEL JELEZZUK A SZAM VEGEN,HOGY
 MILYEN SZAMRENDSZERBEN ADJUK MEG A KEZDOCIMET (PL.:3A45H)"
40 INPUT"KEREM A KEZDOCIMET DECIMALIS,VAGY HEXADECIMALIS
 ALAKBAN";T$
50 IF RIGHT$(T$,1)="H" THEN T$=LEFT$(T$,LEN(T$)-1)
:GOSUB 150:C=T ELSE C=VAL(LEFT$(T$,LEN(T$)-1))
60 CLS: PRINT"NYOMJ LE EGY BILLENTYUT! - V-VEL
 BEFEJEZHETED":PRINTTAB(10);"HEXADECIMALIS";TAB(40);
"DECIMALIS"
70 PRINTTAB(8);"CIM";TAB(20);"ERTEK";TAB(35);"CIM"
;TAB(50);"ERTEK"
80 FOR I=0 TO 10:A$="" :NEXT
90 DS=C:GOSUB210:C$=HS$:DS=PEEK(C):GOSUB210
100 IF LEN(HS$)=1 THEN HS$="0"+HS$
110 A$(10)=C$+STRING$(10,32)+HS$+STRING$(10,32)+
STR$(C)+STRING$(13,32)+STR$(PEEK(C))
120 PRINT@256," ":FOR I=0 TO 9:PRINT STRING$(8,32);A$(I)
:NEXT
130 FOR I=0 TO 9:A$(I)=A$(I+1):NEXT
140 W$=INKEY$:IF W$="" THEN 140 ELSE IF W$="V"
 THEN END ELSE C=C+1:GOTO 90
150 REM H-D SZUBRUTIN
160 K=LEN(T$):T=0:FOR I=1 TO K:J$=MID$(T$,I,1)
170 IF ASC(J$)>47 AND ASC(J$)<58 THEN J=VAL(J$)
180 IF ASC(J$)>64 AND ASC(J$)<71 THEN J=ASC(J$)-55
190 T=T+J*16^(K-I):NEXT
200 RETURN
210 REM D-H SZUBRUTIN
220 HS$=""
230 HH=INT(DS/16):HS=DS-HH*16
240 IF HS<10 THEN HS$=CHR$(HS+48)+HS$ ELSE
 HS$=CHR$(HS+55)+HS$
250 DS=HH:IF HH<>0THEN 230 ELSE RETURN
260 D
```

```
NYOMJ LE EGY BILLENTYUT! - V-VEL BEFEJEZHETED
          HEXADECIMALIS                    DECIMALIS
        CIM          ERTEK          CIM              ERTEK


        16EC          D0          5868              208
        16ED          55          5869              85
        16EE          54          5870              84
        16EF          C3          5871              195
        16F0          4C          5872              76
        16F1          4F          5873              79
        16F2          53          5874              83
        16F3          45          5875              69
        16F4          CC          5876              204
        16F5          4F          5877              79
```

Ø		címtől az interpretert,
16896	–	től a BASIC programunkat,
1536Ø	–	tól a képernyő tartalmát irathatjuk ki, stb.

Az utóbbinál ellenőrizhetjük a kiírás helyességét is, mivel a képernyőn lévő karakterek decimális kódjait írja ki a programunk.
A képernyő segítségével láthatóvá tehetjük a memóriába történő beírást is. A megadott kódszámú karakter megjelenik a képernyő adott karakterpoziciójában. Az történik, mint a PRINT @ utasítás esetén.
A 60–as sorba azt is írhatjuk a következő programban, hogy

CLS:PRINT@P , CHR$(K)

```
10 CLS:PRINT"BEIRAS A MEMORIABA - KEPERNYON MEGJELENITES"
20 PRINT"A 'V' BILLENTYU LENYOMASAVAL FEJEZODIK
 BE A PROGRAM"
30 PRINT"MAS BILLENTYU HATASARA TOVABB FUT"
40 INPUT "A KEPERNYO KARAKTERPOZICIOJA";P
50 INPUT "A KARAKTER KODJA";K
60 CLS:POKE 15360+P,K
70 Y$=INKEY$:IF Y$="" THEN 70 ELSE IF Y$<>"V" THEN 10
```

A memória állapotáról ad információt a MEM *utasítás* is. A BASIC programunk tárolására felhasználható még szabad memóriaterület nagyságát adja meg. Azt, hogy hány szabad rekesszel rendelkezünk.

A rendszer elindításakor kiadott PRINT MEM utasítással kiírathatjuk, hogy maximálisan milyen nagyságú memóriaterület áll a BASIC programunk rendelkezésére (16 K).

A BASIC programban használt változók értékeinek helyét a memóriában a VARPTR *utasítás*sal határozhatjuk meg.

A következő program 30–as utasítássorában a T változó értéke lesz az a cím, ahol az X$ változó hossza tárolódik, azaz a *string*ben szereplő karakterek száma. Ezt olvassuk ki a PEEK utasítással. A 40–es utasítássorban az XS elhelyezkedésének kezdőcímét állítjuk elő a T+2 és a T+1 című rekeszek értékeivel. A két byte tartalmából állítjuk elő a címet decimálisan.

Ennek a két adatnak az ismeretében íratjuk ki egy ciklussal az X$ karaktereit tartalmazó rekeszek címeit (90–es sor), a rekeszek tartalmát, azaz a karakterek decimális kódjait (100–as sor) és a karaktereket (11Ø–es sor).


```
10 REM STRINGEK TARCIMEI
20 CLS:INPUT "KEREM A STRINGET";X$
30 T=VARPTR(X$):N=PEEK(T)
40 C=PEEK(T+2)*256+PEEK(T+1)
50 PRINT@128,"A BEGEPELT KARAKTEREK TARCIMEI"
60 PRINT@384,"A KARAKTEREK KODJAI"
70 PRINT@640,"A KARAKTEREK"
80 FOR I=0 TO N-1
90 PRINT@192+8*I,C+I;
100 PRINT@448+8*I,PEEK(C+I);
110 PRINT@704+8*I,CHR$(PEEK(C+I));
120 NEXT

KEREM A STRINGET? DEBRECEN

A BEGEPELT KARAKTEREK TARCIMEI
 31985    31986    31987    31988    31989    31990    31991    31992


A KARAKTEREK KODJAI
 68       69       66       82       69       67       69       78


A KARAKTEREK
D        E        B        R        E        C        E        N
READY
>
```

Numerikus változók értékeinek tárolására felhasznált rekeszek száma:

egész	– 2
egyszeres pontosságú	– 4
kétszeres pontosságú	– 8

A C = VARPTR(X) *utasítás*sal (90–es sor) a legalacsonyabb helyiértékű rekesz címét határozzuk meg. A címek növekvő sorrendjében helyezkednek el az egyre magasabb helyiértékű byte–k. Az egyszeres és kétszeres pontosságú változók kitevője a 4., ill. a 8. rekeszben található. A programmal megadott pontosságú változók értékét íratjuk ki byte–onként, a byte–ok csökkenő helyiértéke szerint, balról jobbra. Legvégül a kitevőt. A rekeszek tartalmát binárisan is megjelenítjük, hogy vizsgálni lehessen a számábrázolás módját.

```
10 CLS:PRINT"A NUMERIKUS VALTOZOK TAROLASA"
20 INPUT "MILYEN PONTOSSAGU
VALTOZOKAT AKARSZ VIZSGALNI -%,!,#";V$:CLS
30 IF V$="%" THEN DEFINT X:T=1 ELSE
 IF V$="#" THEN DEFDBL X:T=7 ELSE T=3
40 PRINT@0,STRING$(39,32);:PRINT@0,"";:
INPUT "A VALTOZO ERTEKE";X:
PRINT@40,STRING$(20,30);:PRINT@40,X;CHR$(31)
50 FOR I=T TO 0 STEP -1
60 IF I=T THEN IJ=T ELSE IJ=T-I-1
70 IF V$="%" THEN IJ=1-SGN(I)
80 PRINT@133+IJ*7,"T";I;
90 C=VARPTR(X):H=258+IJ*7:
PRINT@H,STRING$(4,32);:PRINT@H,C+I;
100 K=PEEK(C+I):H=387+IJ*7:PRINT@H,K;
110 FOR J=1 TO 8:B(J)=0:NEXT
120 J=1
130 B(J)=K-2*INT(K/2):K=INT(K/2)
140 J=J+1:IF K<>0 THEN 130
150 PRINT@64*(7+IJ),"T";I;"-";:
FOR L=8 TO 1 STEP -1:PRINTB(L);:NEXT
160 NEXT I
170 GOTO40
```

```
A VALTOZO ERTEKE? ■                                   2

     T 1     T 0

   17708   17707

     0       2
T 1 - 0   0   0   0   0   0   0   0
T 0 - 0   0   0   0   0   0   1   0
```

```
 A VALTOZO ERTEKE? ■                                  -2

     T 1     T 0

   17708   17707

    255     254
 T 1 - 1   1   1   1   1   1   1   1
 T 0 - 1   1   1   1   1   1   1   0
```



```
A VALTOZO ERTEKE? ■                                   3

     T 2     T 1     T 0     T 3

   17709   17708   17707   17710

     64      0       0       130
T 2 - 0   1   0   0   0   0   0   0
T 1 - 0   0   0   0   0   0   0   0
T 0 - 0   0   0   0   0   0   0   0
T 3 - 1   0   0   0   0   0   1   0
```

```
A VALTOZO ERTEKE? ■                                  -3

     T 2     T 1     T 0     T 3

   17709   17708   17707   17710

     192     0       0       130
T 2 - 1   1   0   0   0   0   0   0
T 1 - 0   0   0   0   0   0   0   0
T 0 - 0   0   0   0   0   0   0   0
T 3 - 1   0   0   0   0   0   1   0
```

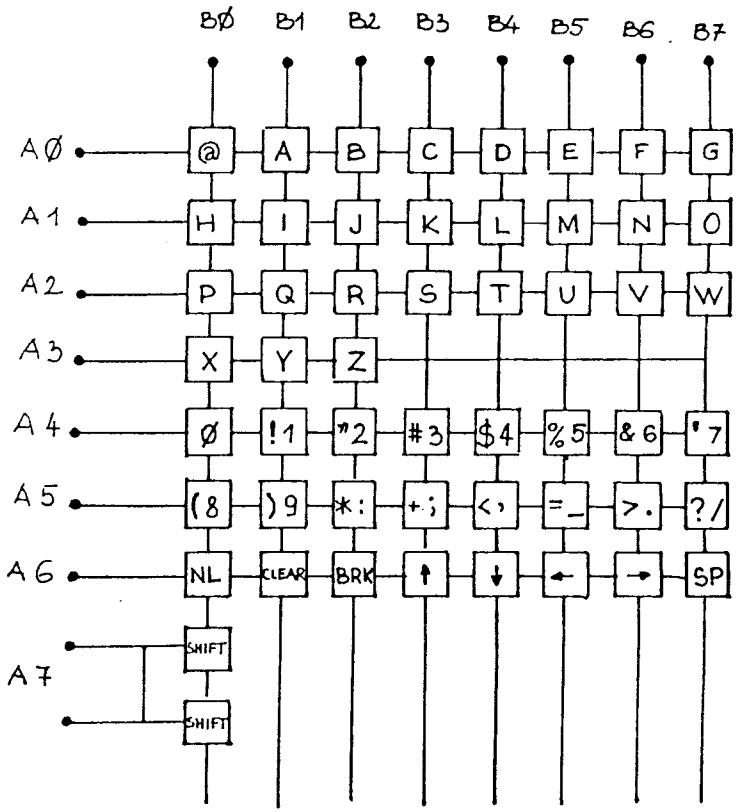
```
A VALTOZO ERTEKE? ■                                   3

     T 6     T 5     T 4     T 3     T 2     T 1     T 0     T 7

   17713   17712   17711   17710   17709   17708   17707   17714

     64      0       0       0       0       0       0       130
T 6 - 0   1   0   0   0   0   0   0
T 5 - 0   0   0   0   0   0   0   0
T 4 - 0   0   0   0   0   0   0   0
T 3 - 0   0   0   0   0   0   0   0
T 2 - 0   0   0   0   0   0   0   0
T 1 - 0   0   0   0   0   0   0   0
T 0 - 0   0   0   0   0   0   0   0
T 7 - 1   0   0   0   0   0   1   0
```

A memória egy része a 38ØØH(14336D) címtől a 3BFFH(15359D) címig a *billentyűzet állapotá*ról ad képet. A memóriarekeszek értékéből megállapíthatjuk azt, hogy melyik billentyűt, vagy billentyűket nyomtuk le.
A billentyűzet–mátrix a rendszer logikájának megértéséhez nyújt segítséget.



Először a mátrix sorait külön–külön vizsgáljuk. A billentyűzet–mátrix *sorai*ban szereplő karakterek billentyűinek lenyomását a következő memóriarekeszek érzékelik. (A címeket decimális alakban adjuk meg.)

AØ – 14337
A1 – 14338
A2 – 1434Ø
A3 – 14344
A4 – 14352
A5 – 14368
A6 – 144ØØ
A7 – 14464

Ezek az értékek szerepelnek a következő program DATA adatlistájában, ahonnét beolvassuk a T vektorba (40–es sor).
Az A vektorba olvassuk be a PEEK utasítással ezeknek a rekeszeknek az értékeit (50–es sortól a 80–as sorig).
A B vektorban tároljuk az előző állapotot. Mivel a program elején a B vektort nulláztuk, a memóriarekeszek tartalmának kijelzésére csak akkor kerül sor, ha leütünk egy billentyűt, vagy a továbbiakban, ha más billentyűt vagy billentyűket is lenyomunk.

```
10 REM BILLENTYUZET ERZEKELESE -1
20 CLS: PRINT "NYOMJ LE EGY BILLENTYUT"
30 FOR I=1 TO 8:B(I)=0:NEXT
40 FOR I=1 TO 8:READ X:T(I)=X:NEXT
50 A(1)=PEEK(T(1)):A(2)=PEEK(T(2))
60 A(3)=PEEK(T(3)):A(4)=PEEK(T(4))
70 A(5)=PEEK(T(5)):A(6)=PEEK(T(6))
80 A(7)=PEEK(T(7)):A(8)=PEEK(T(8))
90 FOR I=1 TO 8:IF A(I)<>B(I) THEN 110 ELSE NEXT
100 GOTO 50
110 FOR I=1 TO 8:B(I)=A(I):NEXT
120 CLS: PRINT "  CIM";TAB(18);"B I T E K";TAB(36)
;"DECIMALIS ERTEK"
130 FOR I=1 TO 8:PRINT T(I);: GOSUB 150:PRINTéI*64+40,
A(I):NEXT
140 GOTO 50
150 REM REGISZTER BITJEI
160 T=A(I):FOR J=1 TO 8
170 PRINTéI*64+3*(8-J)+8," ";
180 IF T/2=FIX(T/2) THEN PRINT "  0"; ELSE PRINT"  1";
190 T=INT(T/2):NEXT:RETURN
200 REM A BILLENTYUZET REGISZTEREINEK CIMEI
210 DATA 14337,14338,14340,14344,14352,14368,14400,14464
```

CIM	BITEK								DECIMALIS ERTEK
14337	0	0	0	0	0	0	0	0	0
14338	0	0	0	0	0	0	0	0	0
14340	0	0	0	0	0	0	0	0	0
14344	0	0	0	0	0	0	0	0	0
14352	0	0	0	0	0	0	0	0	0
14368	0	0	0	0	0	0	0	0	0
14400	0	0	0	0	0	0	0	0	0
14464	0	0	0	0	0	0	0	0	0

Valamelyik memóriarekesz értékének a megváltozásakor a 110–es utasítássorral folytatódik a program, ahol tároljuk a memóriarekeszek újabb értékeit. Ezek után megkezdődik a memóriarekeszek címének és értékeinek kiírása (130–as sor).
A 150–es utasítássorral kezdődő szubrutinnal a rekeszek tartalmát bináris alakban íratjuk ki, hogy lássuk melyik bit értéke lesz 1. Ez jelzi, hogy a billentyűzet–mátrix melyik oszlopában található az a karakter, amelyiknek billentyűjét lenyomtuk. BØ a legkisebb, B7 a legnagyobb helyiértékű bitet jelzi az ábrán.

Mi a szerepük azoknak a rekeszeknek, amelyek az előző programban vizsgált rekeszek között helyezkednek el?
Vonjuk ki a rekeszek címeiből 14336–t: 2 hatványait kapjuk. Egy 8 bites szám bitjeit jelöljük A7, A6, . . . , AØ–al, ahogyan a billentyűzet–mátrix sorait is jelöltük. Legyen azoknak a biteknek az értéke 1, amely bitnek megfelelő sort vizsgálni szeretnénk. Például a billentyűzet–mátrix 3. sorának vizsgálata esetén az A2 = 1, azaz a számunk értéke 4 lesz.
Adjuk ezt 14336–hoz: 14340–t kapunk, azaz annak a memóriarekesznek a címét, amit az előző programban a 3. sor vizsgálatához adtunk meg.

Vizsgáljuk most az *1., 3. és 5. sort*.
Ekkor AØ = 1, A4 = 1, és a számunk 49 lesz.
14385 + 49 = 14385, azaz a 14385 című memóriarekeszt kell vizsgálnunk, ha az 1., 3. és 5. sorban megadott karakterek billentyűjének lenyomását szeretnénk érzékelni.
Ezt a számítást követjük a következő programban is, ahol az igen (I) és nem (N) válaszokkal jelöljük ki azt, hogy melyik sorokat vizsgáljuk. A válaszokat egy bináris szám számjegyeinek tekintjük (I = 1, N = Ø).
Az így kapott számot kell hozzáadnunk 14336–hoz, a memóriaterület kezdőcíméhez (70–es sor), és megkapjuk annak a rekesznek a címét, amelyiknek értéke jelzi, hogy az AØ, A4 vagy az A6 sorok valamelyik karakterének billentyűjét lenyomjuk.


```
10 REM BILLENTYUZET ERZEKELESE - 2
20 CLS:PRINT"A BILLENTYUZET-MATRIX VIZSGALT SORAI
 - VALASZ: I-N"
30 SZ=0:FOR I=0 TO 7:PRINTI;".SOR ";
35 S$=INKEY$:IF NOT(S$="I" OR S$="N") THEN 35
40 PRINTS$:IF S$="I" THEN SZ=SZ+2^I
50 NEXT
60 C=14336+SZ
70 X=PEEK(C)
80 IF X<>0 THEN PRINT@780,"CIM=";C,"ERTEKE=";CHR$(30);X
90 GOTO70
```

```
A BILLENTYUZET-MATRIX VIZSGALT SORAI - VALASZ: I-N
 0 .SOR I
 1 .SOR N
 2 .SOR I
 3 .SOR I
 4 .SOR N
 5 .SOR I
 6 .SOR N
 7 .SOR I

              CIM= 14509          ERTEKE= 64
```

A billentyűzet állapotát jelző memóriarekeszek vizsgálatával *több billentyű egyidejű lenyomását is figyelni tudjuk*. Ezt mutatjuk be majd egy játékprogram segítségével.
Ügyességi számítógépes játékoknál a számítógép gyorsasága, például egy pont mozgásának sebessége, meghatározó. A BASIC programnyelven írt programok esetében a végrehajtandó utasítások számának csökkentésével tudunk ezen javítani. Ennek vannak korlátai.
További javítást a BASIC programban beépített *gépi nyelvű szubrutinok*kal érhetünk el. A gépi nyelvű szubrutin hívása az USR *függvénnyel történik.*

USR (egész kifejezés) — gépi nyelvű szubrutin hívása

A *szubrutin kezdőcímét* az USR függvény előtt kell megadni. A szubrutin első utasításának címét a POKE utasítással a 16526 és a 16527–es című rekeszekbe írjuk. A 16526–os címre a kisebb helyiértékű byte–ot.
A szubrutin kezdőcíme legyen 32432 (7EBØH). Ez két byte. A két byte értéke: 126(7EH) ill. 176(BØH).

A *cím kijelölése* tehát a következő:

```
POKE 16526 , 176 : POKE 16527 , 126
```

A címen kívül meg kell adni a gépi nyelvű *szubrutin bemenő paramétereit* is. Egyik lehetőség, hogy ezeket a POKE utasítással írjuk a memóriába a szubrutin hívása előtt.

A kimenő paraméterek elhelyezéséről a szubrutinban kell gondoskodnunk. A gépi nyelvű szubrutinból való visszatérés után a *kimenő paraméterek* értékeit a PEEK függvénnyel tudjuk a BASIC programban átvinni. A gépi nyelvű szubrutinunkat a RET utasítással kell lezárni!

Két byte összegzését végző szubrutinunk az előzőek figyelembevételével a következő lesz:

cím			utasítások	
hexadecimális	*decimális*	*szimbolikus*	*hexad.*	*decimális*
7EBØ	32432	LD A,Ø	3E	62
7EB1	32433		ØØ	Ø
7EB2	32434	ADD A,Ø	C6	198
7EB3	32435		ØØ	Ø
7EB4	32436	LOAD(7EAFH),A	32	5Ø
7EB5	32437		AF	175
7EB6	32438		7E	126
7EB7	32439	RET	C9	201

A két bemenő paramétert a 32433 és a 32435—ös rekeszekbe helyezzük el. A kimenő paraméter, a két byte összege, a 32431—es rekeszbe kerül.
A szubrutin argumentumának ebben az esetben nincs szerepe, ezért bármilyen változót, vagy konstanst megadhatunk.

Az összegzést csak 255—ig tudja elvégezni a szubrutinunk, mivel a túlcsordulást nem vizsgáljuk. A következő programba majd ezt is beépítjük.
A szubrutin utasításkódjait a program elején a POKE utasításokkal írjuk a memóriába.

A gépi nyelvű szubrutinok számára a számítógép bekapcsolásakor foglalhatunk le memóriaterületet. A *? megjelenésekor decimális alakban beírhatjuk azt a címet, ahonnét védeni akarjuk a BASIC programtól a memóriát.


```
10 CLS:PRINT"KET BYTE OSSZEGZESE-1"
20 INPUT  " A KET BYTE ERTEKE DECIMALISAN";A%,B%
25 IF A%>255 OR B%>255 THEN 20
30  REM  A SZUBRUTIN BEIRASA
40 POKE 32432,62:POKE 32433,00:POKE 32434,198:
POKE 32435,0
50 POKE 32436,50:POKE 32437,175:POKE 32438,126:
POKE 32439,201
60 REM PARAMETEREK BEIRASA
70 POKE32433,A%:POKE32435,B%
80 REM SZUBRUTIN HIVAS
90 POKE 16526,176:POKE 16527,126
100 Y%=USR(I%)
110 PRINT"A KET BYTE OSSZEGE";PEEK(32431)

KET BYTE OSSZEGZESE-1
 A KET BYTE ERTEKE DECIMALISAN? 21
?? 34
A KET BYTE OSSZEGE 55
READY
>_
```

A memóriarekeszek a BASIC nyelvű programból és a gépi nyelvű szubrutinból egyaránt elérhetők. Ez adja az egyik lehetőséget a *paraméterek átadására* ill. *átvételére*.
Tekintsük ezek után az USR szubrutinhívó utasítást egy olyan függvénynek, melynek argumentumát a zárójelben szereplő egész aritmetikai kifejezés értéke adja. Ezt a két byte—os értéket egy rendszerszubrutin hívással a HL *regiszterpárba* írhatjuk:

CALL ØA7FT

A gépi nyelvű szubrutinunk kimenő paramétere az USR függvény értéke lesz, ha azt a szubrutin végén a HL regiszterbe írjuk és elugrunk a ØA9A címre!
JP ØA9AH

A két byte összegezését végző szubrutinunk ekkor a következő:

cím		utasítások		
hexadecimális	decimális	szimbólikus	hexad.	dec.
7EBØ	32432	CALL ØA7FH	CD	2Ø5
7EB1	32433		7F	127
7EB2	32434		ØA	1Ø
7EB3	32435	LD A,H	7C	124
7EB4	32436	ADD A,L	85	133
7EB5	32437	LD L,A	6F	111

cím		utasítások		
hexadec.	decimális	szimbólikus	hexadec.	dec.
7EB6	32438	LD H,Ø	26	38
7EB7	32439		ØØ	Ø
7EB8	32440	JR NC,CIM	3Ø	48
7EB9	32441		Ø1	1
7EBA	32442	INC H	24	36
7EBB	32443	CIM:JP ØA9AH	C3	195
7EBC	32444		9A	154
7EBD	32445		ØA	1Ø

Ebben a programban már a túlcsordulást is vizsgáltuk, így két tetszőleges byte-ot tudunk adni. Az eredményt két byte–tal adjuk meg.

```
10 REM KET BYTE OSSZEGZESE-2
20 REM OSSZEGZO SZUBRUTIN BEOLVASASA
30 CLS:READ C
40 READ X:IF X<256 THEN POKE C,X:C=C+1:GOTO 40
50 REM FOPROGRAM
60 PRINT"KET BYTE OSSZEGZESE"
70 INPUT "A KET BYTE ERTEKE DECIMALISAN";A%,B%
80 IF A%>127 THEN A%=A%-256
90 POKE 16526,176:POKE 16527,126
100 PRINT"A KET BYTE OSSZEGE";USR(256*A%+B%)
110 REM KEZDOCIM ES SZUBRUTIN UTASITASAINAK DECIMALIS
 KODJAI
120 DATA 32432,205,127,10,124,133,111,38,0,48,1,36,
195,154,10
130 DATA 256

KET BYTE OSSZEGZESE
A KET BYTE ERTEKE DECIMALISAN? 67
?? 34
A KET BYTE OSSZEGE 101
READY
>_
```

A szubrutin bemenő paramétere két byte–on ábrázolható. Ezt írjuk be a HL regiszterpárba, ahol a H legmagasabb helyiértékű bitje az előjelet jelzi. Ezt figyelembevéve az A% változó értékét át kell alakítanunk, ha az nagyobb, mint 127, azaz a legnagyobb helyiértékű bit értéke 1.
Ekkor a két byte–ot csak negatív számként tudjuk bemenő paraméterként a HL regiszterpárba beírni (80–as sor).

Az egész változók tárolásának módját a következő programmal összpontosítjuk.
A tárolásra felhasznált két byte bitjeinek értékét egy gépi nyelvű szubrutinnal határozzuk meg.

cím		utasítások		
hexadec.	decimális	szimbólikus	hexadec.	dec.
7ØØØ	28672	CALL ØA7F	CD	205
7ØØ1	28673		7F	127
7ØØ2	28674		ØA	1Ø
7ØØ3	28675	LD A,Ø	3E	62
7ØØ4	28676		ØØ	Ø
7ØØ5	28677	LD E,Ø	1E	3Ø
7ØØ6	28678		ØØ	Ø
7ØØ7	28679	LD D,Ø	16	22
7ØØ8	2868Ø		ØØ	Ø
7ØØ9	28681	ADD A,L	85	133
7ØØA	28682	JR NZ , IDE	2Ø	32
7ØØB	28683		Ø2	2
7ØØC	28684	LD A,H	7C	124
7ØØD	28685	LD E,D	5A	9Ø
7ØØE	28686	IDE:AND A,E	A3	163
7ØØF	28687	LD L,A	6F	111
7Ø1Ø	28688	LD H,Ø	26	38
7Ø11	28689		ØØ	Ø
7Ø12	2869Ø	JP ØA9AH	C3	195
7Ø13	28691		9A	154
7Ø14	28692		ØA	1Ø

Az A% egész változó értékét beírjuk a 28678 és a 28680–as memóriarekeszbe. Ezt a két byte–ot hasonlítjuk össze a HL regiszter értékével, ahová 2 hatványait írjuk. Ekkor mindig csak egy bitnek az értéke lesz 1.

```
10 REM EGESZ VALTOZOK ERTEKENEK TAROLASA
20 REM SZUBRUTIN BEOLVASASA
30 CLS:READ C
40 READ X:IF X<256 THEN POKE C,X:C=C+1:GOTO 40
50 REM FOPROGRAM
60 INPUT "AZ EGESZ VALTOZO ERTEKE";A%
70 K%=VARPTR(A%)
80 POKE 28678,PEEK(K%):POKE 28680,PEEK(K%+1)
90 PRINT"BINARIS ALAKJA";
100 X%=-32768
110 POKE 16526,0:POKE 16527,112
120 FOR I=15 TO 0 STEP -1
130 IF USR(X%)=0 THEN PRINT0; ELSE PRINT1;
140 X%=SGN(X%)*X%/2%
150 NEXT
160 REM SZUBRUTIN CIME ES UTASITASAI
170 DATA 28672,205,127,10,62,0,30,0,22,0,133,32,2,124
180 DATA 90,163,111,38,0,195,154,10
190 DATA 256

AZ EGESZ VALTOZO ERTEKE? 5
BINARIS ALAKJA 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
READY
>_
```

A szubrutin decimális alakban történő beírása nehézkes, mivel a hexadecimális kódokat át kell alakítanunk decimális kódokká.
Használjuk fel a korábban írt hexadecimális–decimális átalakító BASIC szubrutint, és a gépi nyelvű szubrutin utasításait ennek felhasználásával írjuk a memóriába.

Ezt a megoldás egy *játékprogram*ban mutatjuk be, ahol gépi nyelvű szubrutinnal vizsgáljuk a billentyűzetet, a „nyíl" billentyűket.
Ezek lenyomásával irányíthatjuk egy, a képernyőn megjelenő karakterek mozgását, ügyelve arra, hogy ne „menjen le" a karakter a képernyőről.

A gépi nyelvű szubrutin feladata, hogy a nyíl billentyűk lenyomásakor a következő értékeket állítsa elő:

←	lenyomásakor	−1-et,
→	lenyomásakor	1-et,
↑	lenyomásakor	−64-et,
↓	lenyomásakor	64-et.

Több billentyű lenyomásakor a megadott értékek összegét. Ezt az értéket használjuk fel a BASIC programban a karakter mozgásirányának megadásához, ahhoz, hogy a karakter melyik szomszédos karakterpozicióba kerüljön.
A gépi nyelvű szubrutin szimbólikus utasításai és hexadecimális kódjai a következők:

szimbólikus utasítások		*hexadecimális kódjuk*
LD B,0	....	06
		00
LD A,(3840H)	....	3A
		40
		38
AND 8H	....	E6
		08
JR Z,CIM	....	28
		04
LD A,B	....	78
ADD A,0C0H	....	C6
		C0
LD B,A	....	47
CIM:LD A,(3840H)	....	3A
		40
		38
AND 10H	....	E6
		10
JR Z,Cim1	....	28
		04
LD A,B	....	78
ADD A,40H	....	C6
		40
LD B,A	....	47
CIM1:LD A,(3840H)	....	3A
		40
		38
AND 20H	....	E6
		20
JR Z,CIM3	....	28
		04
LD A,B	....	78

szimbólikus utasítások		*hexadecimális kódjuk*
ADD A,FFH	....	C6
		FF
LD B,A	....	47
CIM:LD A,(384ϕH)	....	3A
		4ϕ
		38
AND 4ϕH	....	E6
		4ϕ
JR Z,CIM4	....	28
		ϕ4
LD A,B	....	78
ADD A,1	....	C6
		ϕ1
LD B,A	....	47
CIM4:LD A,B	....	78
LD(7FFFH),A	....	32
		FF
		7F
RET	....	C9

A kimenő paramétert a 7FFF(32767D) című memóriarekeszbe helyezzük. A 130–as sorban hívjuk a szubrutint. Mivel bemenő paramétere nincs a szubrutinnak, ezért az USR utasítás előtt csak a szubrutin címét kell kijelölnünk (120–as sor).
A kimenő paramétert, a 32767–es rekesz tartalmát kiolvassuk és beírjuk az I% változóba (130–as és 140–es sor).
Az I% változó értéke határozza meg, hogy a karakter hová mozdul el. A karaktert megjelenítjük az új pozicióban és töröljük az előző helyen (100–as sor). Az új pozíciót egyben tároltuk is: X1 = X.
Akkor, ha a 32767–es rekesz tartalma nem változik – nem nyomtunk le újabb billentyűt –, a pont tovább mozog mindaddig, ameddig nem változtatjuk meg az irányát, vagy le nem „megy" a képernyőről. Ezt vizsgáljuk a 80–as utasítássorban. A pont gyors mozgását a gépi nyelvű szubrutin beépítésével értük el.
A szubrutin beírása a memóriába a 40–es sorból hívott és a 160–as sortól kezdődő BASIC szubrutinnal történik. A DATA adatlistában hexadecimális alakban beírt kezdőcimet és gépi nyelvű utasításokat a 200–as utasítássorral kezdődő szubrutin decimális alakban adja meg.

Ezeket a POKE C, T utasítással visszük a memóriába (190–es sor).

```
10 REM PONT ORZES - BILLENTYU ERZEKELES
20 CLS:PRINT"a NYIL  BILLENTYUK LENYOMASAVAL LEHET A
 PONTOT A KEPERNYON      TARTANI."
30 PRINT" VARJ EGY KICSIT!"
40 GOSUB160
50 A(1)=1:A(2)=-1:A(3)=64:A(4)=-64:RANDOM:
I%=A(RND(4)):B=VARPTR(I%)
60 CLS:X=550:X1=550
70 X=X+I%
80 IF X<0 OR X>1023 OR X/64=INT(X/64)THEN CLS:
PRINT"FOLYTATOD(I-N)?";ELSE 100
90 W$=INKEY$:IF W$="" THEN 90 ELSE IF W$="I" THEN 50
 ELSE IF W$="N" THEN CLS:PRINT"KOSZONOM AZ ERDEKLODEST!"
:END ELSE 90
100 POKE 15360+X,42:POKE 15360+X1,32:X1=X
110 IF PEEK(14400)=0 THEN 70
120 POKE16526,0:POKE16527,127
130 A=USR(1):IF PEEK(32767)>128 THEN POKE B+1,255
 ELSE POKE B+1,0
140 POKE B,PEEK(32767):GOTO 70
150 GOTO 80
160 REM GEPI NYELVU  SZUBRUTIN BEIRASA
170 READ T$:GOSUB 200:C=T
180 READ T$:IF T$="VEGE" THEN RETURN ELSEGOSUB 200:POKEC,T
190 C=C+1:GOTO  180
200 REM H - D  SZUBRUTIN
210 K=LEN(T$):T=0:FOR I=1 TO K:J$=MID$(T$,I,1)
220 IF ASC(J$)>47 AND ASC(J$)<58 THEN J=VAL(J$)
230 IF ASC(J$)>64 AND ASC(J$)<71 THEN J=ASC(J$)-55
240 T=T+J*16^(K-I):NEXT
250 RETURN
260 REM GEPI NYELVU SZUBRUTIN KEZDOCCIME
270 DATA 7F00
280  REM GEPI NYELVU SZUBRUTIN UTASITASKODJAI
290 DATA 06,00,3A,40,38,E6,08,28,04,78,C6,C0,47
300 DATA 3A,40,38,E6,10,28,04,78,C6,40,47
310 DATA 3A,40,38,E6,20,28,04,78,C6,FF,47
320 DATA 3A,40,38,E6,40,28,04,78,C6,01,47,78,32,FF,7F,C9
330 DATA VEGE
```

17. MONITOR PROGRAM

SYSTEM parancs

A SYSTEM parancs hatására kilépünk a BASIC rendszerből. A képernyőn megjelenő *? kijelzés után kell beírnunk decimális alakban azokat a kezdőcímeket, ahonnét indítani akarjuk a BASIC rendszert, a monitor programot, vagy más, általunk írt gépi szintű programot. A cím elé / jelet kell írni!
/ 2Ø489 begépelése után a képernyő bal felső sarkában kiíródik a READY? szó. Ugyanaz a lehetőség adott, mint amikor a számítógépet bekapcsoljuk. Ekkor *foglalhatjuk le a memória egy részét* a gépi nyelven írt szubrutinjaink számára. Decimális alakban meg kell adni azt a legnagyobb memóriacímet, ameddig a memóriát használhatja a BASIC rendszerünk a BASIC nyelven írt programok és a programban használt azonosítók értékeinek tárolására.
Amennyiben csak a NEWLINE billentyűt nyomjuk le, nem történik helyfoglalás. Mindkét esetben ezzel a BASIC rendszert is elindítottuk.
Plusz lehetőségekhez jutunk, ha újból kiadjuk a SYSTEM parancsot és a 12294 *címmel indítjuk a* BASIC *rendszert*. Használhatjuk a kisbetűket és hatásos lesz a RE parancs, amivel a BASIC program utasításssorait tudjuk újrasorszámozni.
A 12299 *címtől elindított* BASIC *rendszer*ben az előzőeken kívül, ha valamelyik billentyűt lenyomva tartjuk, a karakter kiírása ismétlődik, ami megkönnyíti az EDIT utasítás használatakor a kurzor megfelelő pozicióba állítását.
Akkor, ha villogó kurzort szeretnénk használni, a 12288–*as címet* kell megadni. Ekkor az eddig ismertetett lehetőségek mind adottak. A kurzor villogását letiltani, vagy megengedni a SHIFT és BREAK billentyűk egyidejű lenyomásával lehet.
A MONITOR *program* független a BASIC rendszerünktől. A 12710-es címtől indítható, és a mikroprocesszor regiszterek, valamint a memóriarekeszek tartalmának vizsgálatát, módosítását és gépi szintű programok kipróbálását teszi lehetővé.
A MONITOR program elindításával megjelennek a képernyőn a központi egység regisztereinek aktuális értékei.
A *memória tartalmának vizsgálat*át a D betű és egy hexadecimális alakban megadott memóriacím begépelésével indíthatjuk.
A D2AØØ prancsra a 2AØØ című rekesszel kezdődően 16 rekesz tartalmát olvashatjuk le a képernyőről hexadecimális alakban.
A ↓ billentyű hatására újabb 16 rekesz tartalmát jeleníthetjük meg, amelyeknek a címe nagyobb. Kisebb című rekeszek tartalmának megjelenítését a ↑ billentyű lenyomásával érhetjük el. Más billentyűt lenyomva újból a regiszterek tartalma jelenik meg és ekkor adhatunk ki újabb parancsot.
Az R billentyű lenyomása után a *regiszterek tartalmát módosíthatjuk*. A regiszterek értékét egymás után kiírja, és ezután megadhatjuk az új értéket, vagy az X billentyű lenyomásával jelezhetjük, hogy nem változtatunk egy–egy regiszter értékén.
A *memóriába adatokat*, vagy *gépi nyelvű utasításkódokat írhatunk* be hexadecimális alakban. A beírás kezdőcímét az M karakter után hexadecimális alakban adhatjuk meg. M7AØØ parancs hatására a 7AØØ című rekesztől kezdődően folyamatosan megadhatjuk a rekeszek értékét. Korábbi értéküket kijelzi a program. Ezzel a módszerrel bármilyen gépi nyelven írt programot, szubrutint beírhatunk a memóriába a 3CØØ címtől kezdődően.
A BASIC rendszer használata esetén a memória nagyobb című rekeszeit használhatjuk erre a célra.
A memóriába beírt gépi nyelven készült programjainkat ki is próbálhatjuk a G paranccsal, amelyben meg kell adnunk a programunk kezdőcímét és azt a címet, ameddig fusson a program.
G7AØØ, 7A45 paranccsal a 7AØØ kezdőcímtő *indítjuk a programunkat* és a MONITOR programba csak akkor térünk vissza, ha a 7A45 címre futott a programunk. Ezt jelzi a PC regiszter értéke is a visszatéréskor. Sikeres visszatérés esetén újabb indítást végezhetünk, esetleg más címeket megadva, mivel a memória tartalma nem változik.
Abban az esetben, ha valamilyen hiba folytán nem jutunk el a visszatérési címig, a RESET gomb benyomásával állíthatjuk alaphelyzetbe a számítógépet.
A MONITOR program biztosította lehetőségek sokat segíthetnek abban, hogy megismerkedjünk a Z80 mikroprocesszor utasításkészletével, és elsajátítsuk a gépi nyelvű programozást.
Az előző fejezet példaprogramjaival ehhez kívánunk segítséget nyújtani. Ennél többre ebben a könyvben nem vállalkozhatunk.
A BASIC rendszerbe visszatérni a B paranccsal lehet.

18. MINTAPROGRAMOK

18.1. VADÁSZAT

A képernyő bal szélén megjelennek különböző magasságban a „vadállatok" szerepét betöltő karakterek és vízszintesen mozognak. Bármelyik billentyű leütésével alul a képernyő közepén elhelyezett „íjból" kilőhetünk egy „nyilat". Egy felkiáltójel mozog felfelé a képernyőn. Ez lesz a „nyílvessző". Ha ügyesek vagyunk, úgy indítjuk a „nyílvesszőt", hogy a két karakter találkozzon. Ekkor eltaláltuk a „vadállatot".

```
10 REM  VADASZAS
15 FOR I=0 TO 13:OUT 31,I:READ RE:OUT 30,RE:NEXT
20 CLS:INPUT "HANY VADAT AKARSZ LONI";L:T=0:CLS
30 PRINT" BARMELYIK BILLENTYU LEUTESEVEL LOHETSZ.":
PRINT@65,"MOST IS USS LE EGY BILLENTYUT ES INDUL A JATEK.""
40 W$=INKEY$:IF W$="" THEN 40
50 CLS:FOR I=1 TO L
60 Y=RND(10):V=RND(32)+32:G=14
70 PRINT@992,"^";
75 OUT31,13:OUT30,4:OUT 31,7:OUT 30,31
80 FOR J=1 TO 63
90 YV=(Y-1)*64+J
100 PRINT@YV-1,CHR$(32);:PRINT@YV,CHR$(V);
110 IF Q$<>"" THEN 130
120 Q$=INKEY$:IF Q$="" THEN 170
125 OUT31,13:OUT 30,0:OUT 31,7:OUT 30,30
130 IF G=0 THENPRINT@32,CHR$(32);:GOTO 170
140 YG=(G-1)*64+32:PRINT@YG+64,CHR$(32);:PRINT@YG,CHR$(33);
150 IF YG=YV OR YG=YV+1 THEN PRINT@YV,"***":OUT31,7
:OUT 30,0:FOR K=1 TO 200:NEXT K:OUT 31,7:OUT30,63:
T=T+1:GOTO 180
160 G=G-1
170 NEXT J
180 CLS:Q$="":NEXT I
190 PRINTL;"VADBOL";T;"-T ELTALALTAL"
500 DATA 200,2,0,0,0,0,5,63,16,15,16,0,60,0
```

A 60–as sorban véletlenszám függvénnyel képezzük, hogy milyen magasságban mozogjon a karakterünk, hanyadik sorban (Y) és a karakter kódját (V). G változó értéke azt mutatja, hogy a „nyílvessző" hanyadik sorban lesz. G kezdőértéke az „íj" helyzetét adja, amit a 14. sorban helyeztünk el.
A 100–as utasítássortól kezdődik az a ciklus, amelyben a „vadállatokat" mozgatjuk. A ciklusváltozó értéke (J) adja meg, hogy a sor hanyadik karakterpozíciójában van a „vad".

A 110–es sorban a képernyő karakterpozícióját határozzuk meg a PRINT@ utasításhoz, amivel kíratjuk a karaktert (120–as sor). Azért, hogy egymás után több „nyílvesszőt" ne lőhessünk ki, a 140–es sort átugorjuk mindaddig, ameddig a „vad" ki nem szalad a képernyőről, azaz a J ciklusváltozójú ciklus befejezéséig. Ekkor töröljük Q$ változó értékét (210–es sor), és indul a következő „vad".

A 160–as sortól kezdődően vizsgáljuk a „nyíl" és a „vad" helyzetét. A G változó értéke jelzi, hogy a „nyíl" hanyadik sorban van. G = 0 esetén csak a „vad" halad tovább (160–as sor), mert a „nyílvessző" lekerült a képernyőről.

A 170–es sorban YG változó értéke megadja a „nyíl" karakterpozícióját a képernyőn, és azt megjeleníti, az előzőt törli. A 180–as sorban vizsgáljuk a találatot. A „vad" két pozíciójánál is találatot jelez a programunk, YV és YV+1-nél is.
A találatot három csillag jel kiírásával jelezzük és egy kis várakozási ciklus után a T változó értékét növeljük. Ezzel számoljuk a találatokat, amit a játék befejezésekor kíratunk.

A játékot hangeffektusokkal kívántuk élőbbé tenni. A 20–as sorban olvassuk a DATA adatlistából a hanggenerátor regisztereinek értékét. Az „A" hanggenerátoron egy zenei hang periódusidejét állítottuk be (R0 = 200, R1 = 2). Ezzel és a zajgenerátor együttes megszólaltatásával jelezzük a „nyíl" kilövését. Az „íj" húrjának és a távolódó nyílvessző hangját időben gyengülő hanggal érzékeltetjük.
Az „A" és a „C" csatornákat időben változó hangjelek megszólaltatására tettük alkalmassá azáltal, hogy a 8–as és a 10–es regiszterekbe 16–ot írtunk.
A 150–es soron kiválasztjuk az időben változó jel alakját (R13 = 0) és engedélyezzük az „A" csatornán a zenei hangot, a „C"–n a zörejt (R7 = 30).
Időben változó a „vad" hangja is. Itt erősödik a hang és megszűnik. Ekkor csak a „C" csatornán jelenik meg az időben változó zaj (R13 = 4, R7 = 31). Ezt a hangot a 90–es sorban indítjuk.

A 180–as sorban a találatot jelezzük. Engedélyezzük az összes hanggenerátor működését (R7 = 0). Ekkor a „B" hanggenerátoron is megszólal a zaj 15–ös hangerősséggel (R9 = 15), amit még a program elején állítottuk be. Ez a hang szól a várakozó ciklus befejezéséig. Ekkor letíltjuk a hangokat (R7 = 63).

18.2. MORSE

A program bemenő adata egy tetszőleges karaktersorozat. A karaktersorozat betűinek morse jeleit megjeleníti a képernyőn és ezt hangjelekkel is kijelzi.

```
10 CLS:PRINT "MORSE"
20 DIM KD$(26)
30 REM **** MORSE JELEK BEOLVASASA  ****
40 FOR I=0 TO 26:READ KD$(I):NEXT I
50 OUT 31,8:OUT 30,15:OUT 31,0:OUT 30,162:OUT 31,1:
OUT 30,1
60 OUT 31,7:OUT 30,63
70 INPUT "KEREM A SZOVEGET";SZ$
80 REM ****  KARAKTERENKENT KODOLUNK  ****
90 FOR I=1 TO LEN(SZ$)
100 KA$=MID$(SZ$,I,1):KK=ASC(KA$)-64
110 IF (KK<>-32 AND KK<1) OR KK>26 THEN 200
120 IF KK=-32 THEN KK=0
130 FOR J=1 TO LEN(KD$(KK))
140 PV$=MID$(KD$(KK),J,1)
150 IF PV$="." OR PV$="/"THEN N=20 ELSE N=40
160 PRINT PV$;:IF PV$="/" THEN 170 ELSE OUT 31,7:OUT 30,62
170 FOR T=1 TO N:NEXT T
180 OUT 31,7:OUT 30,63
190 NEXT J:PRINT "/";
195 NEXT I
200 REM MORSE KODOK
210  DATA /,.-,- ...,-.-.,-..,.,..-.,--,.....,.,.---,
-.-,.-..,  --,-.
220 DATA ---,.--.,--.-,.-.,...,-,..-,...-,----,-..-,-.--,
--..,--.-
```

```
MORSE
KEREM A SZOVEGET? MEGYUNK A BUKKBE
--/./--./-.--/..-/-./-.-///.-///- .../../-/-./-./-./- ..././
READY
>_
```

Az angol abc 26 betűjének morse kódjait és a szünetjelnek megfeleltetett / jelet a KD$ vektorba olvassuk be a DATA adatlistából (40—es sor). Kijelöljük az A hanggenerátor hangerősségét és a frekvenciáját. Az utóbbinál a fínom és a durva beállítást is meg kell adni (50—es sor). A keverő—regiszterbe beírt értékkel letíltjuk a hanggenerátorokat (60—as sor).
A beolvasott karaktersorozatot karakterenként vizsgáljuk. A karakter ismeretében előállítjuk a karakter kódját.
A kódból levonva 64—et megkapjuk a KD$ vektornak azt az elemét, amelyben a karakter morse jelét tároljuk. Előbb azonban megvizsgáljuk, hogy betű vagy szünetjel karakterről van—e szó. A vizsgálatot a kódszám alapján végezzük. Akkor, ha KK nem kisebb mint 1 és —32-vel sem egyenlő, valamint 20 nál sem nagyobb, akkor betűről van szó. —32-t úgy kapunk, hogy a szünetjel karakter kódjából is kivontunk 64—et. A szünetjel esetén az index KK = 0 és így a / jelet íratjuk ki.
A morse jelek a KD$ vektorban vannak. Ezeket is karakterenként vizsgáljuk. Megállapítjuk, hogy . vagy — jelet kell—e kiíratnunk és ennek megfelelő hangjelet generálunk. Kijelöljük a hangjel idejét. Ezt az N változó értéke határozza meg. N értékétől függ a várakozási ciklus hossza (150—es sor). A kiíratás után engedélyezzük azt, hogy az A hanggenerátor zenei hangot adjon. A várakozó ciklus végrehajtásáig szól a hang, majd letíltjuk a 180—as soron.

18.3. SZÁMLÉTRA

Két játékos felváltva mond egy—egy számot, az előzőtől legalább 1—gyel, legfeljebb L—lel nagyobbat. A játékot az nyeri, aki ezen a számlétrán „fellépegetve" először kimondja az előre meghatározott számot, például 100—at.
A programot gépeljük be, és a számítógép partnerünk lesz a játékban.

```
20 CLS:PRINT "SZAMLETRA"
30 INPUT "MEDDIG JUSSUNK EL";N
40 INPUT "HANYASAVAL";L
50 M=0:REM A LETRA LEGALSO FOKA
60 PRINT "AKARSZ KEZDENI(I/N)?";
70 A$=INKEY$:IF A$="" THEN 70 ELSE PRINT
80 IF A$="I" THEN 140
90 REM A GEP VALASZA
100 V=N-M-INT((N-M)/(L+1))*(L+1)
110 IF V=0 THEN V=RND(L):REM NINCS NYERO LEPES
120 M=M+V:PRINT "A SZAMOM:";M
130 IF M=N THEN PRINT "NYERTEM":END
140 REM A JATEKOS VALASZA(ELLENORZESSEL)
150 INPUT "A SZAMOD:";M1
160 IF M<M1 AND M1<M+L+1 THEN 180
170 PRINT "NE CSALJ!":GOTO 150
180 M=M1:IF M<N THEN 100
190 IF M>N THEN PRINT"NE SZORAKOZZ VELEM!TUDOM,HOGY";
200 PRINT"NYERTEL"
```

```
SZAMLETRA
MEDDIG JUSSUNK EL? 10
HANYASAVAL? 3
AKARSZ KEZDENI(I/N)?
A SZAMOD:? 2
A SZAMOM: 4
A SZAMOD:? 3
NE CSALJ!
A SZAMOD:? 6
A SZAMOM: 8
A SZAMOD:? 10
NYERTEL
READY
>_
```

A nyerő stratégia L = 1Ø és N = 1ØØ esetén az, ha a játék végén mi mondjuk ki a 89–et. Ekkor ellenfelünk legfeljebb 99–et mondhat – azaz nem éri el a 100–at, de legalább 9Ø–et kell mondania, és akkor már mi eljuthatunk a 1ØØ-ig.
Ugyanilyen indoklással beláthatjuk, hogy mi nyerünk, ha korábban 78–at, 67–et, stb. mondtunk. A legkisebb nyerőszám az 1. A nyerő stratégiát ismerve a kezdő játékos biztosan nyerhet.
A legkisebb nyerő számot megkapjuk, ha N–t osztjuk L + 1–el és vesszük a maradékot. Ez lesz V változó értéke. Ezt a számot kimondva L + 1–es lépésekkel eljutunk N–ig, és utoljára mi mondjuk ki N értékét. Igy írtuk meg a programot is.
V = Ø esetén, amikor N osztható L + 1–el a programlépést véletlenszám függvénnyel állítja elő, mert nem nyerő az állás.

18.4. SORBARENDEZÉS

A sokféle rendező algoritmus közül mi az un. „buborék módszert" választottuk. Az algoritmus a következő: a számsorozat szomszédos elemeit összehasonlítjuk, és ha nem a megfelelő sorrendben vannak, akkor kicseréljük a két elemet. A rendezésnek akkor lesz vége, azaz a számsorozat elemei csökkenő vagy növekvő sorrendben helyezkednek el, ha végigvizsgálva a számsorozatot a szomszédos elemek a helyükön vannak, és nem kell egyiket sem kicserélni szomszédjával.
A program megírása előtt gondoljuk végig, milyen részfeladatokat kell megoldani.

– Meg kell adni egy számsorozatot!
  Az elemeket INPUT utasítással is bekérhetjük, de használhatjuk a DATA utasítást is, ahonnét a READ utasítással olvassuk be az adatokat.
– Tárolni kell ezt a számsorozatot!
  Egy vektorba (egyindexes változókba) olvassuk be. Ez legyen A(N). A DIM utasítással le kell foglalnunk a vektornak a helyét a memóriában. N a vektor maximális indexét jelenti.
– A számsorozat szomszédos elemeinek a vizsgálatát egy ciklussal oldjuk meg.
– Legyen növekvő a sorrend!
– Mikor fejeződik be a program?
  Mielőtt a vizsgálatot elkezdjük egy változónak értéket adunk (V = Ø), aminek az értéke csak a csere során változhat (V = 1). A ciklusból kilépve ezt vizsgáljuk.

– Ki kell iratnunk a rendezett számsorozatot!
  A kiíratástól függ, hogy végül csökkenő, vagy növekvő számsorozat jelenik–e meg a képernyőn.

Az algoritmust a következő programmal próbálhatjuk ki:

```
10 REM SORBARENDEZES
30 REM *** ADATBEOLVASAS ***
40 CLS: PRINT "BUBOREKOK":READ N
50 DIM A(N)
60 FOR I=1 TO N:READ A(I):PRINT @64*I+30,A(I):NEXT I
70 REM *** RENDEZES ***
80 V=0
90 FOR I=N-1 TO 1 STEP -1
100 IF A(I)<=A(I+1) THEN 150
110 T=A(I):A(I)=A(I+1):A(I+1)=T
130 PRINT  @64*I+30,CHR$(30);A(I);:
PRINT@64*(I+1)+30,CHR$(30);A(I+1);
140 V=1
150 NEXT I
170 IF  V<>0 THEN 80
180 PRINT @832,
190 REM *** ADATOK ***
200 DATA 13,31,54454,2131,-3112,32.454,3232,
-334,3232,-43334,434334,0,0.00001,387
```

```
BUBOREKOK
                                -43334
                                -3112
                                -334
                                 0
                                 1E-05
                                 31
                                 32.454
                                 387
                                 2131
                                 3232
                                 3232
                                 54454
                                 434334

READY
>_
```

A program növekbő sorrendbe írja ki a számsorozatot.
Az algoritmust szeretnénk személetesebbé tenni. A cseréket kiíratjuk, de csak azt a két számot írjuk át, amelyek helyet cseréltek. Ezt úgy érjük el, hogy fölülírjuk a számokat a képernyőn.
Ugyanarra a helyre írunk a PRINT @ a utasítással. A CHR$ (3Ø) kiírásával

a sort töröljük, mert az előző számból maradhatnak karakterek, ha az hosszabb volt.
Módosítsuk, ill. írjuk be a következő sorokat!

```
60 FOR I=1 TO N:READ A(I):PRINT@64*I+30,A(I):NEXT I
130 PRINT@64*I+30,CHR$(30);A(I);:PRINT@64*(I+1)+30
  ,CHR$(30));A(I+1);
180 PRINT@832
```

Az utóbbi sor beírásával nemcsak a korábbi kiírást írtuk felül, de elértük azt, hogy a program befejezésekor ne változzon a képernyőn a számok helyzete. A READY még a képernyőre kerül. A cseréket nem nagyon tudjuk követni. Ezért írjuk be a következő sort:

```
120 Q$=INKEY$:IF Q$="" THEN 120
```

Mindaddig nem lép tovább a program, amíg le nem ütünk egy billentyűt, azaz a **Q$** értéke egy „üres karakter". Most, ha lépésenként haladunk, szépen látszik, hogy a negatív számok hogyan „haladnak" felfelé. Mint a „buborékok"!
Vegyük észre, hogy minden ciklus után egy szám a végleges helyére kerül. Először a legkisebb.
Gyorsítsuk a programot! Azt az elemet, amelyik már a „helyén van" nem kell vizsgálnunk, ezért a ciklusváltozó végértékét növelni kell eggyel. Legyen a végérték a K változó.
Ezzel a számok rendezése megoldott. A cserék, természetesen csak korlátozott elemszám esetén követhetők a képernyőn.

● Rendezzünk abc sorrendbe szavakat, azaz karakterláncokat (stringek)!

Karakterlánc változókat kell használnunk!
Az A(N) és a T változókat kellene átjelölni A$(N)–re és T$–ra. Ilyenkor nagyon hasznos a DEFSTR utasítás.

Adatoknak megfelelnek a számok is, de a kétkedők kedvéért írjunk be olyan karakterláncokat, amelyeket betűk alkotnak.


```
10 REM SORBARENDEZES
20 DEFSTR A,T
30 CLS:PRINT" BUBOREK - NYOMJ LE EGY BILLENTYUT!":READ N
40 DIM A(N)
50 FOR I=1 TO N:READ A(I):PRINT@64*I+30,A(I):NEXT I
60 V=0
70 FOR I= N-1 TO 1 STEP -1
80 IF A(I)=<A(I+1) THEN 130
90 T=A(I):A(I)=A(I+1):A(I+1)=T
100 Q$=INKEY$:IF Q$="" THEN 100
110 PRINT@64*I+30,CHR$(30);A(I);
:PRINT@64*(I+1)+30,CHR$(30);A(I+1);
120 V=1
130 NEXT I
140 IF V<>0 THEN 60
150 PRINT@832,
160 REM   *** ADATOK ***
170 DATA 13,SZAVAK,BETUK,LABDA,SZILVA,EMLEK,JATEK,
BUBOREK,ALMA,MISKOLC,BUKK,FA,BASIC,HT-1080Z
180 DATA 13,31,2131,-3112,32.454,3232,-334,3232,
-43334,54454,434334,32.56,0.00001,-387
```

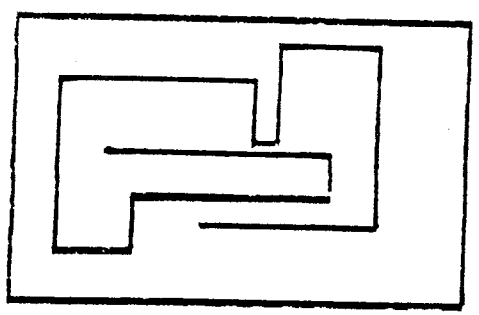
```
 BUBOREK - NYOMJ LE EGY BILLENTYUT!
                                ALMA
                                BASIC
                                BETUK
                                BUBOREK
                                BUKK
                                EMLEK
                                FA
                                HT-1080Z
                                JATEK
                                LABDA
                                MISKOLC
                                SZAVAK
                                SZILVA

READY
>_
```

18.5. HELYFOGLALÁS A KÉPERNYŐN

A képernyőn egy pont mozog. Az útvonalát a pontok megjelenítésével jelezzük. A „nyíl" billentyűk segítségével úgy kell mozgatnunk a pontot, a mozgás irányának megváltoztatásával, hogy minél több olyan pontot érintsen, ahol még nem volt, és ne ütközzön a „falba".

```
10 REM HELYFOGLALAS A KEPERNYON
20 CLS: PRINT@200," A NYIL BILLENTYUKKEL IRANYITHATOD
 A PONTOT":PRINT@340,"NYOMJ LE EGY BILLENTYUT!"
30 FOR I=0 TO 15: OUT 31,I:READ RE: OUT 30,RE:NEXT
40 DATA 162,1,117,2,0,0,10,63,16,16,15,0,5,12,0,0
50 IF INKEY$="" THEN 50
60 CLS: RANDOM
70 IR(1)=8:IR(2)=9:IR(3)=10:IR(4)=91:L=0
80 FOR I=0 TO 127:SET(I,0):SET(I,47):NEXT
90 FOR I=0 TO 47:SET(0,I):SET(127,I):NEXT
100 X=RND(80)+20:Y=RND(20)+15:P=IR(RND(4)):Q$=CHR$(P)
110 Q$=INKEY$:IF Q$="" THEN  130
120 Q=ASC(Q$):IF Q=8 OR Q=9 OR Q=10 OR Q=91 THEN
 P=Q:OUT 31,7:OUT 30,61
130 IF P=8 THEN X=X-1
140 IF P=9 THEN X=X+1
150 IF P=10 THEN Y=Y+1
160 IF P=91 THEN Y=Y-1
170 IF POINT(X,Y) THEN 200 ELSE SET(X,Y):L=L+1
180 OUT 31,7:OUT 30,62
190 GOTO 110
200 CLS:PRINT@330,L;"PONTOT  FOGLALTAL EL"
210 OUT 31,7:OUT 30,31:FOR I=1  TO 100:NEXT:
OUT 31,7:OUT 30,63:OUT 31,13:OUT 30,0
```



Az IR vektorban helyezzük el a „nyíl" billentyűk kódjainak értékét. Miután a „falat" kirajzoltatjuk (80–as, 90–es sor), véletlenszám függvénnyel előállítjuk a pont kezdő koordinátáit, valamint a pont mozgásának irányát a P = IR (RND(4)) utasítással (1ØØ–as sor).
A 11Ø–es sorban kérjük az irányváltozást meghatározó billentyű lenyomását. Akkor, ha az utasítás végrehajtásakor nem nyomunk le billentyűt a pont tovább mozog. A 13Ø–as sortól kezdődik a pont újabb koordinátáinak meghatározása a mozgás irányától függően. A 12Ø–es sorban azt is megvizsgáljuk,

hogy nem nyomtunk–e le más billentyűt, mint a „nyíl" billentyűket. Ha igen, akkor a pont tovább mozog az eredeti irányban.

A 17Ø–es sorban vizsgáljuk azt, mielőtt megjelenítenénk a pont ujabb pozícióját, hogy nem ütközött–e a falnak, vagy nem került–e olyan helyre, ahol már volt. Azt nézzük meg, hogy a pont világít–e.

A programban hangokat is generálunk. A 3Ø–as sorban olvassuk be a hangregiszter értékeit. Az „A" és a „B" hanggenerátort zenei hangra állítottuk be, amiknek a hangja időben változni fog.

A „C" hanggenerátorral zajt állíthatunk elő állandó hangerősséggel. A 12Ø-as sorban a pont mozgását kísérő hangot engedélyezzük. A 18Ø–as sorban az irányváltozást jelezzük a B hanggenerátor mélyebb hangjával.

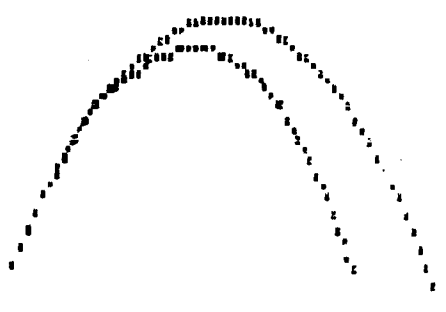
A játék egy ütközés hangjával ér véget, amit a C hanggenerátor zajának rövid idejű megszólaltatásával érünk el (21Ø–es sor). A várakozó ciklus végén a hangot letiltjuk.

18.6. FERDE HAJITÁS

Ferdén elhajított test a gravitációs tér hatására parabola pályán mozog, ha a közegellenállást nem vesszük figyelembe, egyébként un. ballisztikus pályán. Ezt a két görbét jeleníti meg a következő program a DATA adatlistában szereplő értékek esetén.

Amennyiben a 13Ø–as utasítássorból kitöröljük a REM szót, a hajítás szimulálását láthatjuk.

```
10 CLS:PRINT " FERDE HAJITAS":PRINT"NYOMJ LE EGY
 BILLENTYUT. AZ ADATOK A DATA ADATLISTABAN VANNAK.
20 IF INKEY$="" THEN 20 ELSE CLS
30 READ M,G,K,U1,U2,X,Y,T1
40 X1=X:Y1=0
50 U=SQR(U1*U1+U2*U2)
60 A2=-G-K*U2*U/M
70 A1=-K*U1*U/M
80 V2=U2+A2*T1
90 V1=U1+A1*T1
100 Y=Y+(U2+V2)/2*T1
110 X=X+(U1+V1)/2*T1
120 XX=INT(X+0.5):YY=INT(Y+0.5):XT=INT(X1+0.5):
YT=INT(Y+0.5)
130 REMIF YT<48 ANDYT>=0 AND XT<128 AND XT>=0 THEN
 RESET(X1,47-Y1)
140 IF YY<48 AND YY>=0 AND XX<128 AND XX>=0 THEN
 SET(XX,47-YY)
150 T=T+T1
160 U1=V1:U2=V2
170 X1=X:Y1=Y
180 IF Y>= 0 THEN 50
190 ON ERROR GOTO 200:GOTO 30
200 IF INKEY$="" THEN 200 ELSE CLS:END
210 REM KEZDO PARAMETEREK
220 DATA 1,10,0,20,30,0,0,0.1
230 DATA 1,10,0.002,20,30,0,0,0.1
```



A mozgást Δt időintervallumokra osztjuk, és úgy tekintjük, mintha Δt ideig egyenletesen gyorsulva mozogna, azaz a közegellenállásból származó erő nem változna ezalatt. Legyenek a gyorsulás–vektor koordinátái: a_1, a_2, a Δt időintervallum kezdetén, a sebességvektor koordinátái: u_1, u_2.

A közegellenállásból származó gyorsulás koordinátái (a_1', a_2'), a gyorsulás és a sebesség vektor háromszögeinek hasonlóságát felhasználva:

$$a_1' = |\underline{a}| \frac{u_1}{|u|} \qquad a_2' = |\underline{a}| \cdot \frac{u_2}{|u|}$$

Mivel $|\underline{a}| = -k\frac{|\underline{u}|^2}{m}$, ahol m a test tömege, k arányossági tényező,

$$a_1' = -k\frac{|\underline{u}| \cdot u_1}{m} \qquad a_2' = -k\frac{|\underline{u}| u_2}{m}$$

Figyelembe véve a gravitációt, az elhajított test gyorsulásának koordinátái:

$$a_1 = -k\frac{|\underline{u}| u_1}{m} \quad ; \quad a_2 = g - k\frac{|\underline{u}| u_2}{m}$$

A mozgást egyenletesen változónak tekintjük a Δt időintervallum alatt, ezért az időintervallum végén a sebességvektor koordinátái:

$$v_1 = a_1 \cdot \Delta t + u_1 \quad ; \quad v_2 = a_2 \Delta t + u_2$$

A helyvektor koordinátáihoz hozzáadva a Δt idő alatt bekövetkezett elmozdulást, megkapjuk a pont újabb helyzetét:

$$x' = x + \frac{u_1 + v_1}{2} \Delta t \quad ; \quad y' = y + \frac{u_2 + v_2}{2} \Delta t$$

A következő Δt időintervallumhoz tartozó sebességet és helyvektort is hasonlóan számítjuk ki az előzőekben meghatározott V_1, V_2 és X', Y' értékekből.

A programban igyekeztünk az eddig használt jelöléseket használni, hogy áttekinthetőbb legyen.

A DATA adatlistából olvassuk be: M, G, K, U1, U2, X, Y és TT változók értékeit. TT–vel Δt jelöltük.

X1 és Y1 változókban tároljuk az előzőleg megjelenített pontot, amit a 130–as sorban törlünk, ha a REM szót elhagyjuk. Ekkor csak egy pontot látunk mozogni, mintha valóban elhajítanánk egy testet.

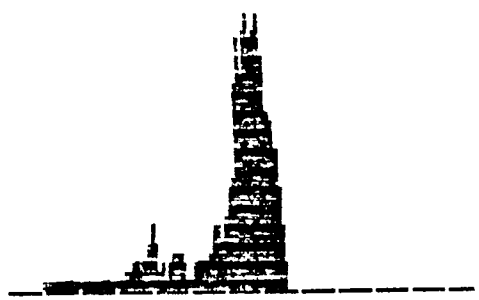
Az XX, XY, XT és YT változók a képernyő koordinátáinak értékeit tartalmazzák. A számítás az X és Y változókkal történik és minden lépésben kerekítjük azokat.
A vizsgálatokkal elértük azt, hogy a számítás akkor is folytatódjon, ha nem lehet megjeleníteni az értékeket a képernyőn. A DATA elemeinek módosításával – U2 növelésével – láthatunk olyan esetet, hogy „kimegy" a pont a képernyőből, majd „visszatér".
A DATA adatlistából egymás után kiolvassa a program a kezdeti adatokat, és a képernyőn több hajítást is bemutathatunk, elvégezhetjük az összehasonlítást.
Vizsgálhatjuk, hogy hogyan módosul a test pályája, ha változtatunk a bemenő paramétereken.
Programunkkal a közegellenállás hatását mutatjuk be.
A 230–as DATA adatlistában a K változó értéke legyen Ø és U1 és U2 értékeit cseréljük fel. Ekkor meggyőződhetünk arról, hogy a hajítás távolsága megegyezik a két esetben, mivel ekkor a hajítási szögek pótszögek.

18.7. DARAZSAK

A stringfüggvények ismertetésekor már bemutattuk a „darazsak röptetését", azaz a normáleloszlás grafikonját. A grafikont karakterekkel rajzoltuk ki és csak 15 darázs ki–be repülését tudtuk vizsgálni.
A grafikus megjelenítés nagyobb lehetőséget biztosít. Több „darazsat" tudunk „röptetni" és a „röptetések" száma is nagyobb lehet, mivel a megjelenítés finomabb.
A 60–as utasítássorral a vízszintes tengelyt rajzoltuk ki. Az A vektorban tároljuk az azonos állapotok előfordulásának számát. Az indexek jelzik a darazsak számát. Az A vektor elemeinek értéke mutatja: hány alkalommal fordult elő az az eset, hogy az index értékével azonos számú darázs volt a fészekben. Az A vektor elemeinek értékét függőleges oszlopokkal ábrázoljuk. Az oszlopok magasságát a 100–as utasítássorral növeljük.
A 110–es utasítássorban állítjuk elő az RND függvénnyel azt, hogy be–, vagy kirepüljön a „darázs". Minél több „darázs van a fészekben" annál valószínűbb, hogy a következő repülés kifelé történik. A D/N hányados értéke határozza meg, hogy mi fog történni nagyobb valószínűséggel.

A „röptetés" akkor fejeződik be, ha valamelyik állapot 46 szor már előfordult. Ezt vizsgáljuk a 120–as utasítássorban.

```
10 CLS:PRINT "DARAZSAK - NORMALELOSZLAS GRAFIKON"
20 DIM A(126)
30 PRINT@200,"A DARAZSAK SZAMA OSSZESEN(MAX:126)";
:INPUT N
40 PRINTTAB(12);:INPUT "MENNYI VAN A FESZEKBEN";D:CLS
50 REM  *****  BEOSZTAS  *****
60 FOR I=0 TO 127:SET(I,46):IF I/10=FIX(I/10) THEN
 RESET(I,46)
70 NEXT I
80 REM *** REPTETES,GRAFIKON  ***
90 RANDOM:A(D)=1
100 SET(D,46-A(D))
110 I=SGN(RND(0)-D/N):D=D+I:A(D)=A(D)+1
120 IF A(D)<46 THEN 100
130 Q$=INKEY$:IF Q$="" THEN 130
140 CLS
```



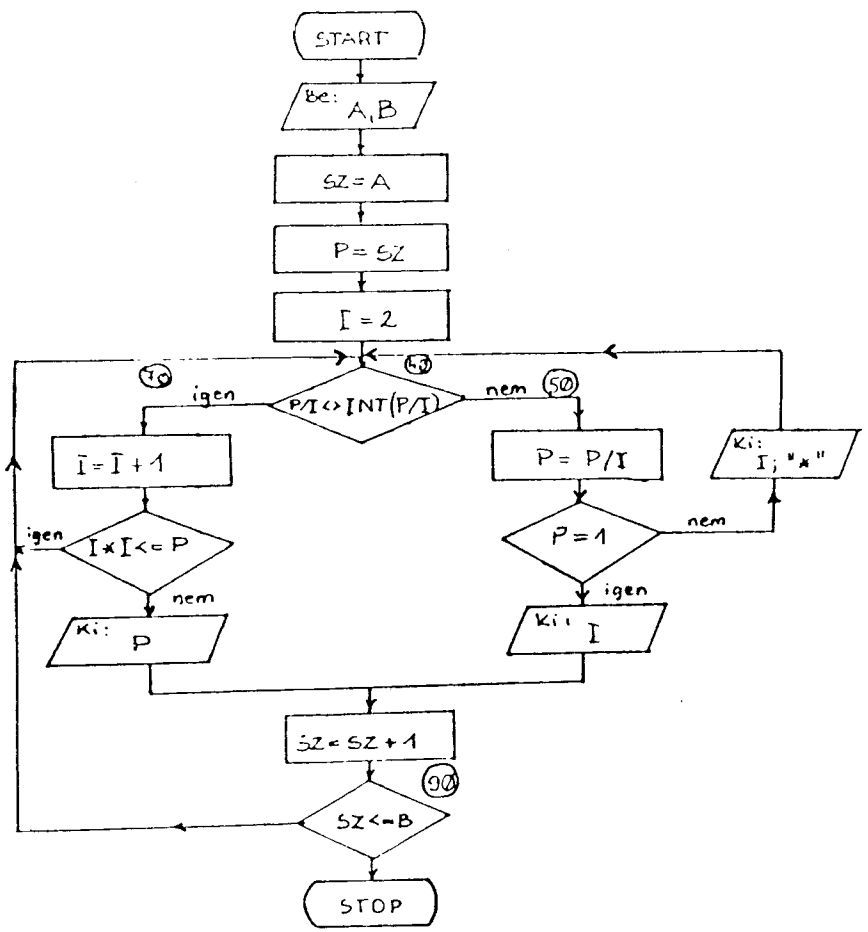
18.8. PRIMTÉNYEZŐS FELBONTÁS

Alkalmazzuk a prímtényezős felbontás szokásos módszerét! Ez pontokba szedve a következő:

1./ Az osztandó legyen a felbontandó szám!
2./ Az osztó értéke legyen 2!
3./ Az osztandó egész többszöröse az osztónak?
igen: 4. pont
nem: 7. pont

4./ Az osztó prímtényező lesz!
5./ Az új osztandó a hányados lesz!
6./ Az osztandó értéke 1 ?
igen: vége
nem: 3. pont
7./ Az osztó értékét növeljük 1–el!
8./ Az osztó négyzete nagyobb, mint a szám?
nem: 3. pont
igen: az osztandó lesz a prímtényező
9./ vége.

Ezzel egy szám prímtényezős felbontásának algoritmusát megadtuk. Bővítsük feladatunkat: A–tól B–ig minden egész számot írjunk fel prímtényezők szorzataként. Ekkor az algoritmusunk folyamatábrája a következő:



A folyamatábrával áttekinthetővé tehetjük az algoritmust és a programozási nyelv jelölésrendszerével leírt műveletek megkönnyítik a programírást. Nagyon hasznos, ha a program megírása előtt folyamatábrát készítünk. Ezzel tervszerűbbé tehetjük munkánkat. Nagy gyakorlattal kell azonban rendelkezni ahhoz, hogy a program tervezésekor készített folyamatábra egyben a program dokumentációjaként is jó legyen. Az utóbbi esetben fontos követelmény, hogy ugyanolyan szerkezetű legyen, mint a program. A programírás során született ötleteket nem szabad figyelmen kívül hagyni, de gondolni kell arra, hogy a folyamatábra is módosuljon. Folyamatábránk végleges alakja is csak a program megírása után született meg.

A változók jelentése:

A: a legkisebb
B: a legnagyobb szám, amelyet prímtényezők szorzataként írunk fel
S: a szám, aminek prímtényezős felbontása történik
P: osztandó
I: osztó

```
5 CLS:PRINT "SZAMOK PRIMTENYEZOS FELBONTASA -1"
10 DEFDBL A-S:INPUT "METTOL";A:INPUT "MEDDIG";B
20 SZ=A
30 P=SZ:PRINT SZ;"=";:I=2
40 IF P/I<>INT(P/I) THEN 70
50 P=P/I:IF P=1 THEN PRINT I:GOTO 90
60 PRINT I;"*";:GOTO 40
70 I=I+1:IF I*I<=P THEN 40
80 PRINTP
90 SZ=SZ+1:IF SZ<=B THEN 30

SZAMOK PRIMTENYEZOS FELBONTASA -1
METTOL? 1983
MEDDIG? 1993
 1983 = 3 * 661
 1984 = 2 * 2 * 2 * 2 * 2 * 2 * 31
 1985 = 5 * 397
 1986 = 2 * 3 * 331
 1987 = 1987
 1988 = 2 * 2 * 7 * 71
 1989 = 3 * 3 * 13 * 17
 1990 = 2 * 5 * 199
 1991 = 11 * 181
 1992 = 2 * 2 * 2 * 3 * 83
 1993 = 1993
READY
>_
```

Programunk minden számmal megvizsgálta az oszthatóságot. Ezt elég lenne csak a prímszámokkal elvégezni.

● Keressük ki a prímszámokat az eratosztenészi szita módszerével! Ezt a feladatot már megoldottuk egy korábbi programmal. Ezt felhasználjuk, és a prímszámokat az A vektorban helyezzük el. Ez lesz a programunk első része. A prímtényezős felbontást végző programrészlet a 180-as sorszámú utasítássorral kezdődik.

```
10 REM PRIMSZAMOK KIKERESESE ERATOSZTENESZI SZITAVAL
20 REM PRIMTENYEZOS FELBONTAS
30 FOR I=0TO15:OUT31,I:READ RE:OUT 30,RE:NEXT
40 CLS:PRINT" SZAMOK PRIMTENYEZOS FELBONTASA"
50 INPUT "METTOL";A:INPUT  "MEDDIG";B
60 IF B<10 THEN N=5 ELSEN=SQR(B)*2:DIM A(N)
70 REM SZITALAS
80 FOR I=2 TO N:A(I)=I:NEXT
90 FOR I=2 TO N/2 : IF A(I)=0 THEN 110
100 FOR J=2*I TO N STEP I:A(J)=0:NEXT
110 NEXT
120 REM TOMORITES
130 J=4:FOR I=5 TO N
140 IF A(I)<>0 THEN A(J)=A(I):J=J+1
150 NEXT
160 SZ=A
170 OUT 31,13:OUT 30,4:OUT 31,7:OUT 30,62
180 P=SZ:PRINTSZ;"=";:I=2
190 IF P/A(I)<>INT(P/A(I)) THEN 220
200 P=P/A(I):IF P=1 THEN PRINT A(I):GOTO 240
210 PRINTA(I);"*";:OUT 31,13:OUT 30,4:OUT 31,7
:OUT 30,61:GOTO 190
220 I=I+1:IF A(I)*A(I)<=P THEN 190
230 PRINTP:OUT 31,13:OUT 30,4:OUT 31,7:OUT 30,61
240 SZ=SZ+1:IF SZ<=B THEN 170
250 DATA 117,1,162,1,0,0,0,63,16,16,0,0,5,4,0,0
```

```
 SZAMOK PRIMTENYEZOS FELBONTASA
METTOL? 1993
MEDDIG? 2003
 1993 = 1993
 1994 = 2 * 997
 1995 = 3 * 5 * 7 * 19
 1996 = 2 * 2 * 499
 1997 = 1997
 1998 = 2 * 3 * 3 * 3 * 37
 1999 = 1999
 2000 = 2 * 2 * 2 * 2 * 5 * 5 * 5
 2001 = 3 * 23 * 29
 2002 = 2 * 7 * 11 * 13
 2003 = 2003
READY
>_
```


18.9. PÍ ÉRTÉKÉNEK KÖZELÍTÉSE

Számolás-igényes feladatok megoldásakor természetes, hogy eszünkbe jut a számítógép. Ilyen feladat a pí értékének meghatározása is.
Már a program megírása előtt fel kell mérnünk lehetőségeinket, a kétszeres pontosságú számábrázolás is csak 16 számjegyig teszi lehetővé a pí értékének meghatározását. Nagyobb pontosságot csak úgy érhetnénk el, ha programokat írnánk, melyekkel megoldanánk a nagyobb pontosságú adatok tárolását és azokkal a műveletvégzést.
A számoláshoz a számítógépnek is időre van szükség. Ezt az eljárás kiválasztásakor figyelembe kell venni. Példaprogramjainkkal két módszert mutatunk be a sok közül. Talán a két legkönnyebben programozhatót. Az időtényező figyelembevételének szükségességét ez a két program is jól érzékelteti.
Wallis (1659) a pí értékére a következő közelítést adta:

$$\frac{\pi}{2} = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdots$$

```
10 REM PI - WALLIS,1659
20 CLS:PRINT"PI KOZELITESE"
30 DEFDBL A-Z
40 INPUT "PONTOSSAG";EP
50 A=2:B=1:P=1:L=1
60 PT=P
70 P=P*A/B
80 K=(PT+P)/2
90 L=-L:IF L=1THEN A=A+2 ELSE B=B+2
100 PRINT@200,"2*OSSZEG= ";2*P;TAB(42);"A=";A;TAB(50)"B=";B
110 PRINT@250,"SZAMTANI KOZEP= ";K+KT;
120 IF ABS(KT-K)> EP THEN KT=K:GOTO 60
130 PRINT@334,"PI= ";INT((K+KT)/EP+0.5)*EP
```

```
PI KOZELITESE
PONTOSSAG? 0.0001

        2*OSSZEG=  3.159894156314577      A= 90    B= 91
  SZAMTANI KOZEP=  3.141643352105137
             PI=  3.1416
READY
>_
```

Leibniz (1673) közelítése a következő:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

```
10 REM PI KOZELITESE  -  LEIBNIZ,1673
20 DEFDBL A-Z:CLS :PRINT"PI KOZELITESE":
INPUT "PONTOSSAG";EP
30 I=1:PT=1
40 ST=S
50 S=S+1/I
60 PRINTé200,"4*A SOROZAT OSSZEGE = ";4*S;TAB(50);"I=";I
70 P=2*(S+ST):PRINTTAB(8);"PI KOZELITESE =";P
80 I=-SGN(I)*(ABS(I)+2):IF ABS(P-PT)>EP THEN PT=P:GOTO 40
90 PRINTé328,"PI=";EP*INT(P/EP+0.5)

PI KOZELITESE
PONTOSSAG? 0.001

        4*A SOROZAT OSSZEGE =  3.171888735237148  I= 65
        PI KOZELITESE = 3.141119504467917
        PI= 3.141
READY
>_
```

18.10. MI LEGYEN A KÖNYV CÍME?

Nem könnyű egy könyvnek címet adni. Szavak, félkész mondatok jutnak eszünkbe. Össze tudjuk szedni a szavakat, melyekkel valamit szeretnénk tudatni, de a cím összeállítása nem mindig sikerül.
Hívjuk segítségül a számítógépet! Megadjuk a kulcsszavakat, amik jellemzik a könyvet, és állítsuk elő az összes lehetséges összetételt! Válasszuk ki a kulcsszavak halmazának összes részhalmazát!
Legyen $H = \{A, B, C, D\}$. Állítsuk elő az üres halmaz kivételével a H halmaz összes részhalmazát!
A problémát szemléletesebben úgy is megfogalmazhatjuk, hogy hányféleképpen lehet egy sakktáblán elhelyezni 1, 2, 3, ill. 4 bábút úgy, hogy egy sorban és egy oszlopban csak egy bábú legyen, és nem tekintjük különböző esetnek, ha ugyanazokat az oszlopokat foglalják el, tehát az AB és BA állást.

	Ø	A	B	C	D
1	x	0	–		
2		x	–	0	
3			x	–	0
4				x	

A bábúkat helyezzük el az X–el jelölt mezőkön. A programban ezt a 6Ø–as utasítássorral oldjuk meg. RH vektor elemeinek értéke jelzi, hogy hányadik oszlopban vannak a bábúk.
A 7Ø–es utasítássorban az első sorban lévő bábút léptetjük jobbra mindaddig, amíg le nem lépünk a tábláról, azaz az értéke: N + 1. Ezt vizsgáljuk a 90–es utasítássorban. Ekkor visszalépünk a sor elejére és a következő sorban lépünk egyet jobbra a bábúval. A ciklussal minden sorban megvizsgáljuk a „lelépést".
A 12Ø–as sorban azt vizsgáljuk, hogy hány bábú lépett már el a kezdő pozicióból, azaz mennyi a részhalmazok elemeinek száma. A 130–as sorral elérjük azt, hogy az egy elemű részhalmazokat vizsgálat nélkül kiírja a program.
A 140–es sorban lévő ciklus csak akkor fut végig, ha a sorszámok növekedésével az oszlopok sorszámai is növekednek, azaz, ha egy bábúval belépünk a következő sor bábújának oszlopába, vagy túllépünk rajta, akkor nincs kiíratás, a program a 70–es sorral folytatódik.
A program akkor fejeződik be, ha a H halmazt is kiírta, mint részhalmazt. Az ábrán 0 jellel jelölt állást kiírja, a – jellel jelzettet nem írja ki a program.

```
10 CLS:PRINT " MI LEGYEN A KONYV CIME? -
 HALMAZ OSSZES RESZHALMAZA"
20 CLEAR 100:DIM KS$(20)
30 FOR I=1 TO 20:READ KS$(I):IF KS$(I)="VEGE" THEN
 40 ELSE NEXT
40 N=I-1
50 REM RESZHALMAZOK
60 FOR I=1 TO N:RH(I)=I-1:NEXT
70 RH(1)=RH(1)+1
80 FOR I=1 TO N-1
90 IF RH(I)=N+1 THEN RH(I)=I:RH(I+1)=RH(I+1)+1
100 NEXT
110 REM KIIRATAS
120 FOR I=1 TO N:IF RH(I)>=I THEN M=I:NEXT
130 IF M=1 THEN 150
140 FOR I=1 TO M-1:IF RH(I)<RH(I+1) THEN NEXT ELSE 70
150 FOR I=1 TO M:PRINTKS$(RH(I));"    ";:NEXT
```

```
160 PRINT:IF M<N THEN 70 ELSE PRINT "LEHET GYORSITANI
 A PROGRAMON!":END
170 REM KULCSSZAVAK
180 DATA ISKOLA,SZAMITOGEP,BASIC,TANULAS,100 PROGRAM
,HT-1080Z,MIT TUD
190 DATA VEGE
```

A program futtatása során a számítógép a következő címeket „sugallta":

Iskolaszámítógép BASIC programozási nyelve
Mit tud az iskolaszámítógép?
HT–1080Z, az iskolaszámítógép
BASIC programozási nyelv tanulása az iskolaszámítógépen
100 BASIC program az iskolaszámítógéphez
Számítógép az iskolában

Azt a problémát végül, hogy mi legyen a könyv címe, a számítógép nem oldotta meg, de valamit segített.

19. MELLÉKLET

19.1. *Kezelési parancsok*

AUTO– *automatikus sorszámozás*

AUTO — a sorszámok: 10, 20, . . .
AUTO 2,3 — a sorszámok: 2, 5, 8, . . .
AUTO 3 — a sorszámok: 3, 13, 23, . . .

– A sorszámok után egy * jelzi, ha már van ilyen sorszámú utasítás.
– A BREAK billentyű lenyomásával léphetünk ki ebből a módból.

CLEAR – *memória törlés*

Programban is használjuk. Részletes ismertetése az utasításoknál.

CLOAD – *beolvasás magnetofonról*

CLOAD — a beépített magnetofonról beolvassa a következő utasítást
CLOAD*–1, "JATEK" — a beépített magnetofonról beolvassa a J nevű programot
CLOAD*–2, "BETU" — a csatlakoztatott magnetofonról beolvassa a B nevű programot

– A megnetofont lejátszásra kell állítani.
– Helyes beolvasáskor a két * jel közül a másodiknak időnként villognia kell.
– Ha a második nem villog: a RESET gomb benyomásával állítsuk alaphelyzetbe a rendszert és a jelszint beállítása után kezdjük előlről a beolvasást.

CLOAD?*–1, "X" — ellenőrzi az X nevű tárolt programot a beépített magnetofonon
CLOAD? — ellenőrzi a következő programot, amit a szalagon talál a beépített magnetofonon
— közvetlenül a kiíratás után kell a vizsgálatot elvégezni, amikor a program még változatlanul a memóriában van.

CONT – *megszakított program végrehajtásának folytatása*

CONT – folytatódik a program futása, ha a STOP utasítással, vagy a BREAK billentyű lenyomására állt le a program

CSAVE – *program felvétel mágnesszalagra*

CSAVE"E" – az E nevű program rögzítése a beépített magnetofonon történik.
CSAVE 1,"EMSE" – az előző utasítással azonos
CSAVE#-2,"ERA" – az E nevű program rögzítése a csatlakoztatott magnetofonon történik.
– A magnetofont felvételre kell állítani.

DELETE – *utasítások törlése*

DELETE 22–60 – a 22–es sorszámtól a 6Ø–as sorszámmal bezárólag törli az utasítássorokat
DELETE–30 – a 3Ø–as utasítássorig törli a programot
DELETE 40 – törli a 40–es utasítássort.
– Egy utasítássort a sorszám beírásával is törölhetünk.

EDIT – *utasítássorok törlése*

Editor parancsmódban maradunk:
– *kurzor mozgatása*
SPACE – jobbra egy karakterpozicióval
5SPACE – jobbra öt karakterpozicióval
← – balra egy karakterpozicióval
6 ← – balra hat karakterpozicióval
– *kilratás*
L – kiírja a sort
– *keresés*
SK – az első K betű elé áll a kurzor
3SV – a harmadik V betű elé áll a kurzor
– *törlés*
D – törli a kurzor mögött álló karaktert
3D – három karaktert töröl a kurzortól jobbra

3KZ – a kurzortól jobbra a harmadik Z karakterig törli a sort
KA – a kurzortól jobbra az A karakterig törli a sort (A–t már nem)
– *csere*
CW – a kurzor helyén álló karakter helyére a W karaktert írja
2CXY – a kurzor helyétől jobbra lévő két karaktert az XY karakterekre cseréli
– *módosítás befejezése*
A – a módosítást kezdhetjük elölről
Q – a módosítás befejeződik és az utasítássor változatlan marad
NEWLINE – a változtatások érvényesítésével fejeződik be a módosítás

Editor parancsmódból szövegmódba kerülünk:
I – ahol a kurzor áll
H – a kurzor helyétől kezdődően törli a sor végét
X – a kurzor a sor módosítása nélkül a sor végére áll

Szövegmódban a módosítási lehetőségek:
– új karaktert írhatunk be
– a ← billentyű lenyomásával karaktereket törölhetünk
SHIFT ↑ – átváltás szövegmódból editor parancsmódba

LIST – *listázás képernyőre*

LIST – kiírja a programot
LIST 35 – kiírja a 35–ös utasítássort
LIST 2Ø–5Ø – a 2Ø–as sorszámú utasítássortól az 50–esig kiírja a programot
LIST–6Ø – a 6Ø–as utasítássorig kiírja a programot
LIST 60– – a 60–as utasítássortól listázza a programot

A kiírást a SHIFT és @ billentyűk együttes lenyomásával megállíthatjuk és bármelyik billentyű lenyomásával folytathatjuk.
LLIST – program kiírása sornyomtatóra

NEW – *program törlése*

„Törli" a programot és a képernyőt.
A program és az adatok változatlanul a gépben maradnak, de a rendszer–változókat átállítja. Ezért az interpreter nem találja meg a programot és az adatokat.
– Mielőtt új programot írnánk a gépbe, ezt a parancsot kell kiadni.
– A NEW parancs után a POKE17129,1 pranccsal, és a Ø programsor törlésével a program még kilistázható, de nem futtatható.

RE – *ujrasorszámozás*
Az eredeti sorszámokat átírja.
RE – az új sorszámok: 1Ø, 2Ø, 3Ø . . .
RE2 – az új sorszámok: 2, 12, 22, . . .
RE3,5 – az új sorszámok: 3, 8, 11, . . .
RE4 – az új sorszámok: 10, 14, 18, . . .
Ezt az utasítást csak akkor használhatjuk, ha a BASIC rendszerünket a 12288-as, 12299–es vagy a 12294–es címről indítjuk a SYSTEM paranccsal. Ezt megtehetjük akkor is, ha a program már a gépben van.

RUN – *program indítása*
RUN – a legkisebb sorszámú utasítássor végrehajtásával kezdődik a program futtatása
RUN 3Ø – a 3Ø–as utasítássortól indul a program.

SYSTEM – *rendszerprogramok indítása*
✳ ? kiírással kéri a rendszerprogramok kezdőcímét
/ cím begépelésével indul a rendszerprogramunk. A címet decimális számmal kell megadni.
/ 2Ø489 gépi szintű programok részére lefoglalt memória legkisebb címének a beírásával, vagy a NEWLINE billentyű lenyomásával indíthatjuk a BASIC rendszert.

A BASIC *rendszer bővítése:*
/ 12294 – kisbetűket írhatunk a SHIFT és a betűket jelző billentyű egyidejű lenyomásával
– a RE utasítást használhatjuk
/ 12299 – az előző lehetőségek érvényesek, és ha lenyomva tartjuk a billentyűt, a karakter kiírása ismétlődik

/ 12288 – az előző lehetőségek érvényesek és megjelenik a villogó kurzor. A villogást a SHIFT és a BREAK billentyűk egyidejű lenyomásával tílthatjuk le, ill. engedélyezhetjük.
/ 12710 – *monitor program* indítása
– A központi egység regisztereinek és a memóriarekeszek tartalmának vizsgálatát, módosítását, gépi szintű programok írását, futtatását teszi lehetővé.
– Kiírja a központi egység regisztereinek értékét hexadecimális számokkal.

Lehetséges parancsok:

D – memóriatartalom vizsgálata
D234F – a 234F címtől kezdődően 16 memóriarekesz tartalmát kiírja
– a ↑ billentyű lenyomásával kisebb, a ↓ billentyűvel nagyobb címről íratható ki a tartalom
– Bármelyik másik billentyűt lenyomva befejeződik a memória vizsgálat.
R – regiszterek tartalmának módosítása
– Egymás után kiírja és kéri a regiszterek értékeit. Miután mind a 12 regiszternek értéket adunk, visszatérünk monitor üzemmódba.
M – beírás a memóriába
M3D12 – a 3D12 címtől kezdődően megváltoztatjuk a memóriatartalmat. A legkisebb megadható cím 3CØØ.
– Az X billentyű lenyomásával fejezhetjük be a beírást.
G – Gépi szintű program indítása.
G7ØØE, 71D2 – a 7ØØE címtől kezdődik a gépi szintű utasítások végrehajtása. A 71D2 cím elérése után a monitor program folytatódik.
B – visszatérés a BASIC rendszerbe.

TROFF – *nyomkövetés kikapcsolása*
TRON utasítást hatástalanítja.

TRON – *nyomonkövetés*
A program futásakor kiíródik a végrehajtott utasítássorok sorszáma. Hibakereséskor, program ellenőrzésekor használjuk. Programba is beírható.

19.2. UTASITÁSOK

CLEAR kifejezés – *memória törlés*

A kifejezés egészrészének megfelelő számú byte–ot lefoglal stringműveletek részére és törli a változók értékét.
CLEAR – törli a változók értékeit és a GOSUB visszatérési címeket.
CLEAR 520 – törli a változókat és 520 byte–ot lefoglal a stringműveletek számára.
– A RUN automatikusan CLEAR 50–et hajt végre.

CLS – *képernyő törlése*

DATA adatlista – *adatok elhelyezése programban*

Az adatlistában számok és stringek lehetnek. Az adatlista elemeit vesszővel választjuk el.
DATA 23.456, "ADAT", 2356, "EVE", MISKOLC, "KISS, SZÖKE"
– Az adatok beolvasása a READ utasítással történik.
– A beolvasást a RESTORE utasítással lehet elölről kezdeni.

DEFDBL betűlista – *kétszeres pontosságú változók kijelölése*

DEFDBL A,B,E–H – Az A, B, E, F, G, H betűkkel kezdődő változók kétszeres pontosságúak lesznek.

A kétszeres pontosságú változók jelölésére a # jelet is használhatjuk:
A#, A1#

DEFINT betűlista – *egész változók kijelölése*

DEFINT I–N,T – Az I,J,K,L,M,N és a T betűvel kezdődő változók egész változók lesznek.

Az egész változókat a % jellel is kijelölhetjük: X1%, A%(I)

DEFSNG betűlista – *egyszeres pontosságú változók kijelölése*

DEFSNG X,Y, A–C – Az Y,Y, A,B és C betűkkel kezdődő változók egyszeres pontosságúak lesznek.

Egyszeres pontosságú változók kijelölésére a ! jelet is használhatjuk: R!, C3!(15)
A változók alaphelyzetben egyszeres pontosságúak.

DEFSTR betűlista – *string változók kijelölése*

DEFSTR R–T, A,G – Az R,S,T és a G betűkkel kezdődő változók lesznek.

A string változók jelölésére a $ jelet is használhatjuk: A$, X$(I), LI$

DIM tömblista – *tömbök dimenzinálása*

A tömblista elemeivel megadjuk a tömbök dimenzióját és az indexek maximális értékét.

DIM A(1ØØ), X(5,1Ø, 4,12) – Az A vektor 1Ø1 elemű lehet. Az X többdimenziós tömb, indexeinek értéke legfeljebb 5, 1Ø,4, ill. 12 lehet.

DIM Y(2 * N + 1,5Ø) – Az Y mátrix elemeinek első indexe nem lehet nagyobb 2 * N + 1 értékénél, a második indexe 5Ø–nél.
Az N változónak korábban értéket kell adni.

– A tömbök elemeinek indexe az indexben szereplő kifejezés egészértéke. Ez nem lehet negatív.

END – *a program logikai vége*

Vége a programnak. Ezt a READY szó kiírásával jelzi a rendszer.

ERROR hibakód – *hiba szimulálás*

A rendszer úgy reagál az utasításra, mintha a megadott kódszámú hiba bekövetkezett volna.
A hibák kódjait a 19.9 fejezetben adjuk meg.
– ON ERROR GOTO utasítás tesztelésénél használjuk.

FOR–TO–STEP / NEXT – *ciklus utasítás*

FOR változó = kifejezés TO kifejezés STEP kifejezés
.
.
.
NEXT változó

A FOR és a NEXT utasítások közé írt utasításokat ismételten végrehajtja. Az ismétlések számát, a ciklusváltozó értékét a kifejezések értékei határozzák meg.

```
FOR X = 4 TO 26 STEP 3
  .
  .
  .
NEXT X
```

— A ciklusmag végrehajtása X = 4,7,1ϕ,. . . 25 értékekkel történik.

```
FOR I = N TO 1 STEP –1
  .
  .
  .
NEXT I
```

— Az I változó N,N–1, . . . 2,1 értékeket veszi fel a ciklusmag végrehajtásakor.

```
FOR I = 1TO M
INPUT A (I)
NEXT I
```

— M számú adatot olvas be. A ciklusváltozó értékének változását nem kell megadni, ha az 1.

```
FOR J = 1 TO 200:NEXT
```

— Üres ciklus. Várakozásra használjuk.

— A ciklusváltozó nem lehet kétszeres pontosságú.
— A NEXT utasításban, ha kiírjuk, ugyanannak a változónak kell szerepelni, mint a FOR utasításban.
— A NEXT utasításban nem kell a ciklusváltozót kijelölni.
— A NEXT utasítással kell befejezni a ciklusmag végrehajtását.
— A ciklusba nem lehet kívülről „beugrani". A FOR utasítással kezdődik a ciklus.
— A ciklusból kiugorhatunk.
— A ciklusmag a változó kezdőértékével egyszer mindig végrehajtódik.
— Több ciklust is egymásba építhetünk.

GOSUB sorszám — *szubrutin hívás*

A megadott sorszámú utasítással folytatódik a program, majd a szubrutin végét jelző RETURN utasítás hatására a GOSUB utasítást követő utasítás végrehajtására kerül sor.

GOSUB 3ϕϕ — A 3ϕϕ–as sortól kezdődő szubrutin végrehajtása következik.

A szubrutinból is hívhatunk szubrutinokat.

GOTO sorszám — *ugró utasítás*

Az utasításban megadott sorszámú utasítással folytatódik a program.

GOTO 125 — A 125–ös sorszámú utasítással folytatódik a program végrehajtása.

IF–THEN–ELSE — *logikai döntés*

IF állítás THEN utasítások v. sorszám ELSE utasítások v. sorszám.

Állítás lehet egy reláció, relációkkal végrehajtott logikai műveletsor, vagy a POINT utasítás.

Akkor, ha az *állítás igaz*, a THEN szó után szereplő utasítások végrehajtására kerül sor, vagy az adott sorszámú utasítással folytatódik a program. Akkor, ha az *állítás hamis*, az ELSE után írt utasítások végrehajtására kerül sor, vagy az adott sorszámú utasítással folytatódik a program végrehajtása.

Akkor, ha az ELSE szót nem használjuk, a következő sorral folytatódik a program, ha az állítás hamis.

```
IF A <ϕ THEN B = –A ELSE B = A
```

— Ha A értéke negatív, akkor B = –A, ha A = 0 v. pozitív, a B = A utasítás végrehajtása történik.

```
IF A = 3 OR B = 4 THEN 50
```

— Ha az A értéke 3, vagy a B értéke 4, az 50–es utasítássorral folytatódjon a program, egyébként az IF után *következő* sorral.

```
IF A=1 THEN IF B=1 THEN 30 ELSE 40
```

— H az A értéke 1, és a B értéke is egy, akkor a 30–as utasítássor következik. A 40–es utasítássorra akkor jutunk, ha A=1 és B nem egyenlő 1–el. Ha A nem egyenlő l-el, a következő sorral folytatódik a program.

IF X=Ø THEN A=1 : GOTO 5Ø — X egyenlő nulla esetén A értéke 1 lesz és az 5Ø–es sorral folytatódik a program.

IF POINT (X,Y) THEN RESET (X,Y) ELSE SET (X,Y) — Akkor, ha az X,Y koordinátájú pont világít a képernyőn, törli azt, egyébként megjeleníti.

5Ø IF INKEY$ = " " THEN 5Ø — Mindaddig várakozik a program, ameddig le nem ütünk egy billentyűt.

INPUT beolvasási lista — *adatbeolvasás*

Kéri az utasításban felsorolt változók értékeit.

INPUT A,B$ — kéri az A és a B$ változók értékeit

A beolvasási listában magyarázó szöveg és változók szerepelhetnek. A változók értékeit a billentyűzetről, vagy magnetofonról olvassa be.

INPUT "OLDALAK" ; A,B — Kiírja a képernyőre azt, hogy OLDALAK és kéri A és B változók értékeit a billentyűzetről.

INPUT # –1,X,Y,Z — Az X,Y és a Z változók értékeit olvassa be az 1-es számú magnetofonról.

— ? kiírással jelzi a program, hogy adatot vár.

— A begépelt adatok közé vesszőket gépeljünk és legvégül a NEWLINE billentyűt nyomjuk le.

— Csak közvetlenül az INPUT utasítást követően helyezhető el magyarázó szöveg. Utána ;–t kell írni.

LET változó = kifejezés — *értékadás*

Kiszámítja a jobb oldalon álló kifejezés értékét és azt elhelyezi a bal oldalon álló változóba.

A változó és a kifejezés aritmetikai vagy string tipusú lehet. A tipusuknak meg kell egyezni.

A LET utasításszó elhagyható.

A = 4 * B + C — A B változó értékének 4–szereséhez hozzáadjuk C változó értékét, és ez lesz az A változó értéke.

R$ = "ABC" + K$ — Ha pl. a K$ = "XY"–nal, akkor R stringváltozó értéke ABCXY lesz.

— Akkor, ha az aritmetikai kifejezés nagyobb pontosságú, mint a változó, értékes számjegyeket veszíthetünk.

— A stringváltozókban maximum 255 karaktert helyezhetünk el.

LPRINT kiíratási lista — *kiíratás sornyomtatóra*

A PRINT utasításnál leírtak érvényesek.

ON ERROR GOTO sorszám — *hibafigyelés*

Hiba esetén az adott sorszámú utasítás végrehajtásával folytatódik a program.

ON kifejezés GOSUB sorszámlista — *feltételes szubrutinhívás*

A kifejezés egészértéke meghatározza, hogy a sorszámlista hányadik elemével jelölt szubrutin hívására kerül sor.

ON X GOSUB 2Ø,3Ø,6Ø,45 — Ha az [X] értéke 1, akkor a 2Ø–as sorszámmal kezdődő szubrutint hívja, ha 2, akkor a 3Ø–as, ha 3, akkor a 6Ø–ast, ha 4, akkor a 45–öst.

A szubrutin végrehajtása után, vagy ha [X] értéke Ø, vagy 4–nél nagyobb az ON–GOSUB utasítást követő utasítással folytatódik a program.

— A kifejezés értéke nem lehet negatív, és 255–nél nagyobb.

ON kifejezés GOTO sorszámlista — *feltételes ugrás*

A kifejezés egészértéke határozza meg, hogy a sorszámlista hányadik elemével jelölt sorszámon folytatódjon a program.

ON X GOTO 1ØØ,6Ø,85,35 — Ha az [X] értéke 1, akkor a 1ØØ–as, ha 2, akkor a 6Ø–as, ha 3, akkor a 85–ös és ha 4, akkor a 35–ös utasítássorral folytatódik a program.

Akkor, ha az [X]értéke Ø, vagy nagyobb mint 4, az ON–GOTO–t követő utasítást hajtja végre a program.

— A kifejezés értéke nem lehet negatív, és 255–nél nagyobb.

OUT kódszám, adat — *adatkivitel regiszterekbe*

A kódszámmal adott regiszterbe ír egy byte–ot, amit decimálisan kell megadni.

OUT 31,6 — A 31–es regiszterbe ír 6–ot. Ezzel kiválasztotta a hanggenerátor 6–os regiszterét.

OUT 30,15 — A 31—es regiszter értékével meghatározott alregiszterbe ír 15—öt. Ezzel beállítja a zajgenerátor periódusidejét.

POKE cím, adat — *beírás a memóriába*

A megadott memóriacímre beírja az adatot. A címet és az adatot decimálisan kell megadni. Megadhatjuk az adatot olyan kifejezéssel is, amelynek az értéke nem negatív és 256—nál kisebb.

POKE 15000,B — B változó egészrészét beírja a 15000—es című memóriarekeszbe.

PRINT kiíratási—lista — *kiíró utasítás*

Kiírja a kiíratási—lista elemeinek értékét.

Listaelem *lehet:* konstans, kifejezés, TAB(kifejezés)

TAB(kifejezés) — hatására a kifejezés értékének megfelelő karakterpozicióba állítja a kurzort az adott sorban, ha azt még el nem hagyta.

A listaelemeket vesszővel, vagy pontosvesszővel választjuk el.

Pontosvessző hatása: folyamatos a kiírás

Vessző hatása: következő zónába ír

PRINT "KERULET";KE;TAB(40), "TERULET";TE — Kiírja azt, hogy KERULET és a KE változó értékét. A 40—es karakterpoziciótól pedig azt, hogy TERULET és a TE változó értékét.

PRINT @ kifejezés, kiíratási lista — *kurzor elhelyezése*

A kifejezés értéke megadja, hogy a kiíratás a képernyő hányadik karakterpoziciójában kezdődjön. A kifejezés 0 és 1023 közötti értékeket vehet fel, mivel 16 sor van, egy sorban pedig 64 karakter.

PRINT @ I*64, "FOLDES" — Az I. sor elejére írja, hogy FOLDES

PRINTUSING string; kiíratási lista — *kiíratási formátum megadása*

A string értékével megadott formátummal írja ki a lista elemeit.

PRINTUSING "##.#";A,B — A és B változó két egész és egy tizedes értékét írja ki.

PRINTUSING A$; A,B — Ugyanaz történik, mint az előző példában, ha A$ = "## .#"

A = "**##.#" — * jeleket ír a szám elé

A = "$$##.#" — $ jelet ír a szám elé

A = "$$**##.#" — * jeleket és $ jelet ír a szám elé

PRINTUSING B$; C$ — Ha C$ = "ABCD" és B$ = "!" a kiírás A, ha B$ = "%%" a kiírás AB, ha B$ = "% %" akkor ABC.

RANDOM — *véletlenszám generátor*

Beállítja a véletlenszám generátor kezdőértékét. A véletlenszámot az RND függvénnyel hívhatjuk.

— Irjunk a programunk elejére RANDOM utasítást, ha az RND függvényt használjuk!

READ változólista — *adatbeolvasás a DATA adatmezőből*

A változólista elemeinek értékét a DATA adatmezőből olvassa be.

READ A%,X,A(I) — Az A%, X és az A(I) változók értékeit a DATA soronkövetkező adatai adják.

— RESTORE utasítással kezdhetjük előlről az adatbeolvasást.

REM karakterek — *megjegyzés*

A REM utasításban tetszőleges karaktersorozatot, pl. magyarázó szöveget elhelyezhetünk. Ez csak a programlistában szerepel, a rendszer nem értelmezi, a következő REM EZ EGYSZOVEG sorral folytatódik a futás.

RESET (kifejezés, kifejezés) — *képpont törlés*

A kifejezések egészértéke a képernyő egy pontjának koordinátáit adják. Az utasítás hatására az adott koordinátájú pont törlődik (sötét lesz).

RESET (15,I) — (15,I) koordinátájú pont törlődik.

— A koordináták lehetséges értékei:

$0 \leq X \leq 127$ $0 \leq Y \leq 47$

RESTORE — *beolvasás a* DATA *adatmező elejéről*

A RESTORE utasítást követő READ utasítás a legkisebb sorszámú DATA adatmezőből olvassa be az adatokat.

RESUME sorszám — *hibakezelő rutin vége*

Az ON ERROR GOTO utasítás hatására történő ugrás után csak a RESUME utasítás végrehajtásával folytatódhat a program. Előtte meg kell szüntetni a hiba okát.

RESUME — Ott folytatódik a program, ahol a hiba történt.
RESUME 4Ø — A 4Ø—es sorszámú utasítássorral folytatódik a program.
RESUME NEXT — A hibás sort követő utasítássoron folytatódik a program.

RETURN — *szubrutin vége*

Hatására a GOSUB vagy az ON GOSUB után következő utasítás végrehajtásával folytatódik a program.

SET (kifejezés, kifejezés) — *képpont megjelenítés*

A kifejezések egészértékei a képernyő grafikus pontjainak koordinátáit adják. Az utasítás hatására az adott koordinátájú pont világos lesz.

SET(X,Y) — Az (X,Y) koordinátájú pontot megjeleníti.

A koordináták lehetséges értékei:

$Ø \leq X \leq 127$ $Ø \leq Y \leq 47$

— A (0;0) pont a képernyő bal felső sarkában van.

STOP — *a program leállítása*

Megáll a program. A rendszer ezt a BREAK IN sorszám kiírással jelzi.

— A CONT utasítás hatására folytatódik a program végrehajtása.
— A program „belövésekor" is használjuk, a hiba helyének behatárolására.

19.3. ARITMETIKAI FÜGGVÉNYEK

A függvények argumentumát X—el jelöltük, ami tetszőleges aritmetikai kifejezés lehet.

ABS(X) — *abszolutérték függvény*

A függvény értéke: X, ha $X \geq Ø$ és —X, ha $X < 0$

ATN(X) — *arcustangens függvény*

A függvény értéke az a szög radiánban, amelyiknek a tangense X.

CDBL(X) — *kétszeres pontosságú ábrázolás*

A függvény értéke: X értéke kétszeres pontossággal.

CINT(X) — *egészrész függvény*

A függvény értéke: [X], azaz X értékénél nem nagyobb legnagyobb egészszám.

Értelmezési tartomány: $-32768 \leq X \leq 32767$

COS(X) — *cosinus függvény*

A függvény értéke: cos x

X értéke radiánban adja meg a szöget.

CSNG(X) — *egyszeres pontosságú ábrázolás*

A függvény értéke: X értéke egyszeres pontossággal.

EXP(X) — *exponenciális függvény*

A függvény értéke e^X, ahol e = 2,71828

FIX(X) — *egészérték függvény*

A függvény értéke: a szám egész része

INT(X) — *egészrész függvény*

A függvény értéke: [x], azaz x értékénél nem nagyobb legnagyobb egészszám.

LOG(X) — *természetes alapú logaritmus függvény*

A függvény értéke: ln x, azaz a logaritmus alapszáma, e = 2,71828

Értelmezési tartomány: $X > 0$

RND(X) — *véletlenszám függvény*

Értelmezési tartomány: $Ø \leq X \leq 32767$

A függvény értéke:

$[X] = Ø$ esetén a $[Ø,1)$ intervallumból egyszeres pontosságú, $X > 1$ esetén az $[1,[X]]$ intervallumból választ ki egy véletlen egész számot.

$(0 \leq X < 32768)$

SGN(X) — *előjel függvény*

A függvény értéke: 1, ha X > ∅
∅, ha X = ∅
−1, ha X = ∅

SIN(X) — *sinus függvény*

A függvény értéke: sin x
X értéke radiánban adja meg a szöget.

SQR(X) — *négyzetgyök függvény*

A függvény értéke: $\sqrt{X}$
Értelmezési tartomány: X ≥ ∅

TAN(X) — *tangens függvény*

A függvény értéke: tg x
X értéke radiánban adja meg a szöget.

19.4. STRING MŰVELET ÉS FÜGGVÉNYEK

A példákban szereplő változók értékei:

A$ = "ABCDE"
B$ = "XYZ"
V$ = "1543"
K = 65 (65 az A karakter kódja)

\+ — *stringek összefűzése*

C$ = A$ + B$ — C$ értéke: ABCDEXYZ

ASC string — *a string első karakterének kódja*

L = ASC(A$) — L értéke: 65

CHR$ kifejezés — *a kifejezés értékének megfelelő kódszámú karakter*

C$ = CHR$(K) — C$ értéke: A

LEN (string) — *a string hossza*

N = LEN(A$) — N értéke: 5

LEFT$ (string, kifejezés) — *a kifejezés értékének megfelelő számú karakter a string elejéről*

C$ = LEFT$(A$, 3) — C$ értéke: ABC

MID$ (string, kifejezés, kifejezés) — *a string adott elemétől adott számú karakter kiválasztása.* Az első kifejezés értéke azt adja meg, hogy hányadik karaktertől, a második azt, hogy mennyit.

C$ = MID$(A$,2,3) — C$ értéke: BCD

RIGHT$ (string, kifejezés) — *a kifejezés értékének megfelelő számú karakter kiválasztása a string végéről.*

C$ = RIGHT$(A$,3) — C$ értéke: CDE

STR$ (kifejezés) — *az aritmetikai kifejezés értékét stringként adja meg.*

C$ = STR$(654.65) — C$ értéke: 654.65

STRING$ (kifejezés, karakter—konstans vagy kódja) — *a kifejezés egészértékének megfelelő számban előállítja a karaktert.*

C$ = STRING$(4,"A") — C$ értéke: AAAA
C$ = STRING$(5,65) — C$ értéke: AAAAA

VAL (numerikus string) — *tetszőleges stringet számként próbál értelmezni.*

N = VAL(V$) — N értéke: 1543
N = VAL(A$) — N értéke: Ø

19.5. EGYÉB FÜGGVÉNYEK

INKEY$ — *egy karakter beolvasása*

V$ = INKEY$ — Az utasítás végrehajtásakor lenyomva tartott billentyű karakterét elhelyezi a string–változóba.
— A karakter nem jelenik meg automatikusan a képernyőn.

INP(kódszám) — *adatbeolvasás regiszterekből*

INP(31) — hang és I/O regiszterek értékének beolvasása.
Az OUT 31,R utasítással kijelölt regiszter értékét olvassa be.
OUT 31,7 : OUT 3Ø,127 : OUT 31,15 : X = INP(31)
— Az R15 regiszter lesz az INPUT csatorna, és beolvassuk az értékét.

MEM — *memória szabad területe*

Értéke megadja a memória szabad byte–jainak számát.

PRINT MEM — Kiírja, hogy hány byte szabad még a memóriából.
PRINT MEM($) — A szabad (nem–string) terület.
PRINT MEM(A$) — A szabad string–terület nagyságát írja ki. Az argumentumban szereplő kifejezés tipusától függően.

PEEK (cím) — *olvasás a memóriából*

Kiolvassa az adott című memóriarekesz tartalmát.
A címet decimálisan kell megadni és az eredményt is decimálisan kapjuk.
PRINT PEEK(3ØØØ) — Kiírja a 3ØØØ–es cím tartalmát.

POINT (kifejezés, kifejezés) — *képpont vizsgálat*

A kifejezésekkel a képernyő koordinátáit adjuk meg.
Ha a képernyő (X,Y) koordinátájú pontja világít, az értéke –1, ha nem, 0 lesz. Ez megfelel IGAZ, vagy HAMIS logikai ítéletnek.

IF POINT(X,Y) THEN RESET(X,Y) ELSE SET(X,Y)

— Ha világít az (X,Y) koordinátájú pont törli azt, ha nem, megjeleníti a pontot.
— A koordináták lehetséges értékei:
Ø ≤ Y ≤ 47 ; Ø ≤ X ≤ 127

USR (egész kifejezés) — *gépi szintű szubrutin hívása*

Az egész kifejezéssel a szubrutin paraméterét adjuk meg.

VARPTR (változó) — *változó helye a memóriában*

A változó értékét tartalmazó memóriarekeszek címeit határozhatjuk meg az utasítás segítségével.

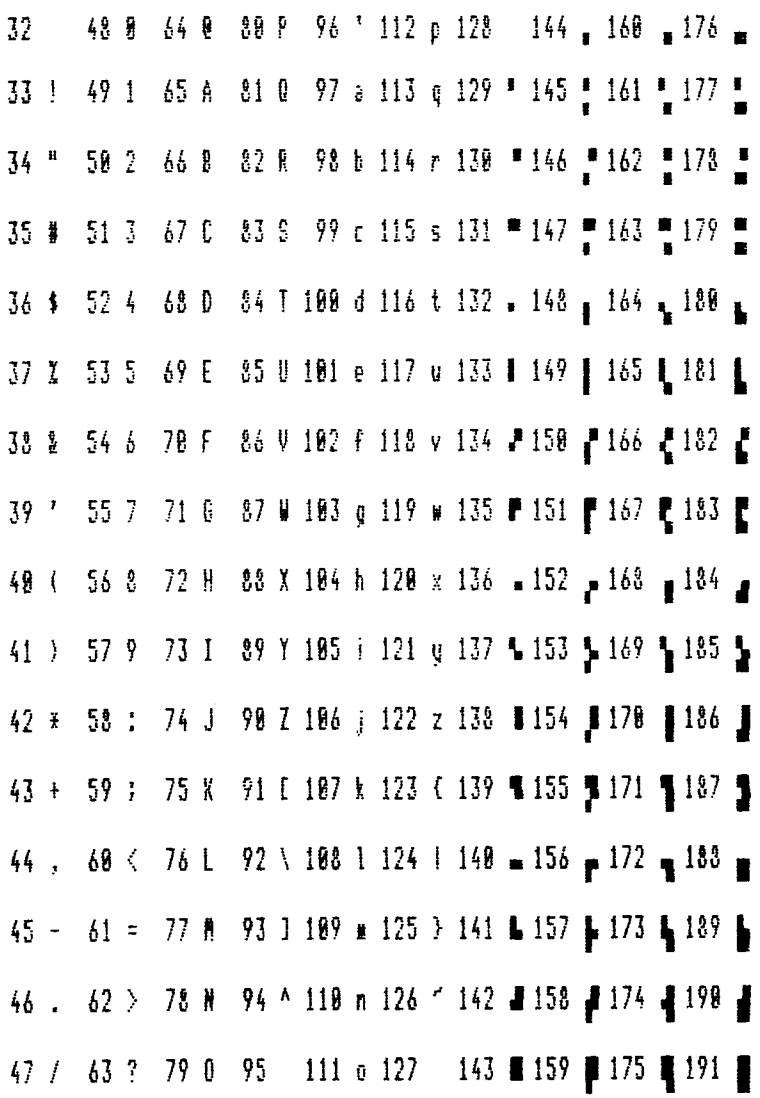
19.6. KARAKTEREK ÉS ASCII KÓDJUK

32		48	0	64	@	80	P	96	`	112	p	128		144		160		176	
33	!	49	1	65	A	81	Q	97	a	113	q	129		145		161		177	
34	"	50	2	66	B	82	R	98	b	114	r	130		146		162		178	
35	#	51	3	67	C	83	S	99	c	115	s	131		147		163		179	
36	$	52	4	68	D	84	T	100	d	116	t	132		148		164		180	
37	%	53	5	69	E	85	U	101	e	117	u	133		149		165		181	
38	&	54	6	70	F	86	V	102	f	118	v	134		150		166		182	
39	'	55	7	71	G	87	W	103	g	119	w	135		151		167		183	
40	(	56	8	72	H	88	X	104	h	120	x	136		152		168		184	
41	)	57	9	73	I	89	Y	105	i	121	y	137		153		169		185	
42	*	58	:	74	J	90	Z	106	j	122	z	138		154		170		186	
43	+	59	;	75	K	91	[	107	k	123	{	139		155		171		187	
44	,	60	<	76	L	92	\	108	l	124	|	140		156		172		188	
45	-	61	=	77	M	93	]	109	m	125	}	141		157		173		189	
46	.	62	>	78	N	94	^	110	n	126	~	142		158		174		190	
47	/	63	?	79	O	95	_	111	o	127		143		159		175		191	

Tabulátor karakterek: 192–től 255–ig.

PRINT " * " : CHRS(N+192) : " * " — N darab szóköz a két * karakter között.

BASIC rendszer bővítése esetén:

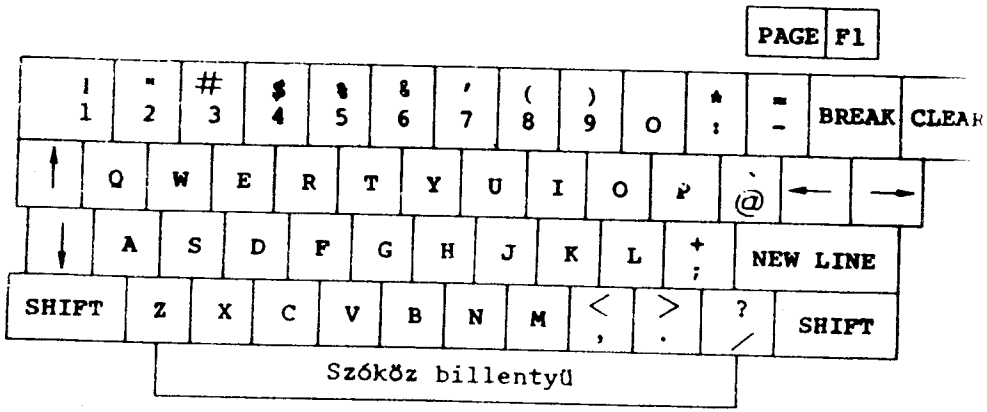


19.7. VEZÉRLŐ KARAKTEREK

A kurzor mozgatására, kiírási mód megválasztására, a képernyő adott részének törlésére van lehetőség.

kód	PRINT CHR$ (kód) hatása
8 vagy 24	– egy karakterpozicióval balra lép a kurzor
10 – 13	– új sorban folytatódik a kiírás
23	– a kiírás „ritkítva" történik A kiírt karakterek között betűköz karakterek lesznek.. – Alapállapotot a CLEAR billentyű lenyomásával, vagy a PRINT CHR$(28) utasítással állíthatjuk vissza.
25	– egy karakterpozicióval jobbra lép a kurzor
26	– egy sorral lejjebb áll a kurzor A közbeeső karakterek nem törlődnek.
27	– egy sorral feljebb áll a kurzor A közbeeső karakterek nem törlődnek.
28	– a képernyő első karakterpoziciójába áll a kurzor
29	– a kurzor a sor elejére áll
30	– a kurzor állásától jobbra törli a sort
31	– a kurzor állásától kezdődően törli a képernyőt

19.8. BILLENTYŰK ÉS FUNKCIÓJUK



← — a kurzortól balra egy karaktert töröl

→ — a kurzort 8 karakterpozicióval jobbra viszi

↓ — a kurzort egy sorral lejjebb viszi

↑ — a hatványozás jele. A képernyőn [jel jelenik meg

SHIFT — felső karakterek legépelésekor lenyomva kell tartani

betűköz — / SPACE / billentyű a legalsó, hosszú billentyű

BREAK — program leállítás

CLEAR — képernyő törlés

NEWLINE — parancsok, utasítássorok és INPUT adatok lezárása

19.9. HIBAÜZENETEK ÉS HIBAKÓDOK

Jel	Kód	Hiba jellege, forrása
BS	9	Az index nagyobb, mint amit a DIM utasításban megadtunk
CN	17	CONT utasítással nem folytatható a program végrehajtása
DD	10	Kétszer szerepel ugyanaz a változó a DIM utasításban
FC	5	A változó az értelmezési tartománynak nem eleme (negatív index, negatív szám négyzetgyöke, a SET utasítás argumentuma nem megengedett).
FD	22	Hibás adatbeolvasás magnóról
ID	12	Nem megengedett direkt utasítás (INPUT sorszám nélkül)
LS	15	A string 255 karakternél hosszabb
MO	21	Operandus hiányzik
NF	1	FOR nélküli NEXT utasítás, vagy nem megfelelő ciklusváltozó a NEXT utasításban
NR	18	Nincs RESUME utasítás az ON ERROR GOTO utasítás után
OD	4	Nincs adat. A READ, vagy az INPUT utasítás számára nem adtunk meg adatot, vagy nem eleget
OS	14	Nincs hely a karakterláncok (stringek) részére. Használjuk a CLEAR utasítást.
OM	7	Nincs több tárolóhely: memória vagy betelt, vagy a CLEAR utasítással lefoglaltuk, sok az egymásba ágyazás (GOSUB, FOR–NEXT), túl hosszú a program.
OV	6	Túl– vagy alulcsordulás. Egy szám, vagy egy kifejezés értéke nagyobb, vagy kisebb, mint a megengedett értékhatár.
RG	3	GOSUB nélküli RETURN utasítás
RW	19	A RESUME utasítás nem az ON ERROR GOTO után következik.
SN	2	Szintaktikus hiba: utasításszó hibás leírása, nem megengedett karakter használata, rossz zárójelzés, stb.
ST	16	Túlságosan összetett stringművelet
TM	13	Karakterláncok (stringek) és numerikus adatok vegyes használata
UE	20	Nincs olyan hibakód, amit az ERROR utasításban használunk
UL	8	Nem létező sorra hivatkoztunk
/Ø	11	Kisérlet nullával való osztásra

20. PROGRAMOK JEGYZÉKE

A COMMODORE és PRIMO számítógépekre elkészített programcsomagok programjait * –gal jelöltük.

21. IRODALOMJEGYZÉK

ABC–s füzetek
Budapest, ELTE, 1982.

D. Allcock: Ismerd meg a BASIC nyelvet
Budapest, Műszaki Könyvkiadó, 1983.

BASIC *iskolásoknak*
Budapest, KFKI, 1983.

BASIC *kézikönyv*
Budapest, Hiradástechnikai Szövetkezet, 1983.

C. Brown: Beszélgessünk a számítógépről!
Budapest, Műszaki Könyvkiadó, 1975.

Csákány A.: Mit tud a zsebszámológép?
Budapest, Műszaki Könyvkiadó, 1978.

Csákány A. – Dr. Vajda: Játékok a számítógéppel
Budapest, Műszaki Könyvkiadó

Dusza Á.: Kis Programozók Baráti Köre: BASIC (munkafüzet, feladatlap, útmutató)
Budapest, TIT, 1979.

Dusza Á.: A BASIC programozási nyelv (jegyzet)
Budapest, OPI, 1981.

Hámori M.: Ismerkedés a komputerrel
Budapest, Tankönyvkiadó, 1973.

Kisszámítógépes oktató rendszerek
Budapest, KFKI, 1980.

J. Lambert – M. Simon – Verrept: Építsünk logikai gépet
Budapest, Műszaki Könyvkiadó, 1978.

Lőcs–Sarkadi–Nagy–Szlankó: A BASIC programozási nyelv
Budapest, Műszaki Könyvkiadó, 1976.

Obádovics–Szelezsán: Bevezetés a programozásba
Budapest, Tankönyvkiadó, 1973.

Dr. Perge I.: A számítástechnika alapjai
Budapest, Tankönyvkiadó, 1979.

Dr. Poronyi G.: 33 BASIC program
Pécs, Baranya Megyei Pedagógiai Intézet, 1983.

Szelezsán J. – Vadász P.: Az elektronikus számítógép programozása
Budapest, Közgazdasági és Jogi Könyvkiadó, 1974.

Utassy–Mersich–Székely: A HT–1080Z iskolaszámítógép programozása
Budapest, Kandó K. Villamosipari Műszaki Főiskola, 1983.

Zámborszky Ferenc: Zárt formában nem megoldható problémák gépi feldolgozása (feladatgyűjtemény)
Miskolc, Pedagógus Továbbképzési Intézet, 1983.

A könyv és a könyv programjai megrendelhetők az alábbi címen:
MISKOLC VÁROSI MŰVELŐDÉSI KÖZPONT
Miskolc, Széchenyi u. 30. 3530 Pf.: 324
Telefon: 46 / 87–844
Számlaszám: 270–98002 OTP B.A.Z. Megyei Igazgatósága
641–473218 Városi Művelődési Központ, Miskolc
Készpénzért megvásárolhatók a „Molnár Béla" Úttörőházban,
Miskolc, Győri kapu 27. 3531

A HT–1080Z számítógéphez készített kazettán a könyv minden programja megtalálható. A COMMODORE és PRIMO számítógépekre átírt programokat * –gal jelöltük a „Programok jegyzéke" c. fejezetben.

MEGRENDELŐLAP
(minta)

Megrendeljük a Borsod–Abaúj–Zemplén megyei Tanács V.B. Művelődésügyi Osztálya kiadásában megjelent
DUSZA ÁRPÁD: A HT–1080Z ISKOLASZÁMÍTÓGÉP BEMUTATÁSA PROGRAMOKKAL című könyvet.
Ára: 60.– Ft. Megrendelt példányszám: . . . . .

Megrendeljük a könyv programjait tartalmazó:
HT–1080Z programcsomagot kazettán – Ára: 1.500.– Ft Db.: . . .
COMMODORE–16 programcsomagot kazettán vagy lemezen Ára: 1.300.– Ft Db.: . . .
COMMODORE–64 programcsomagot kazettán vagy lemezen Ára: 1.300.– Ft Db.: . . .
PRIMO programcsomagot kazettán Ára: 1.300.– Ft Db.: . . .

Megrendelő neve: . . . . . .

Címe: . . . . . .

Megrendelés száma: . . . . . .

Kelt: . . . . . PH aláírás