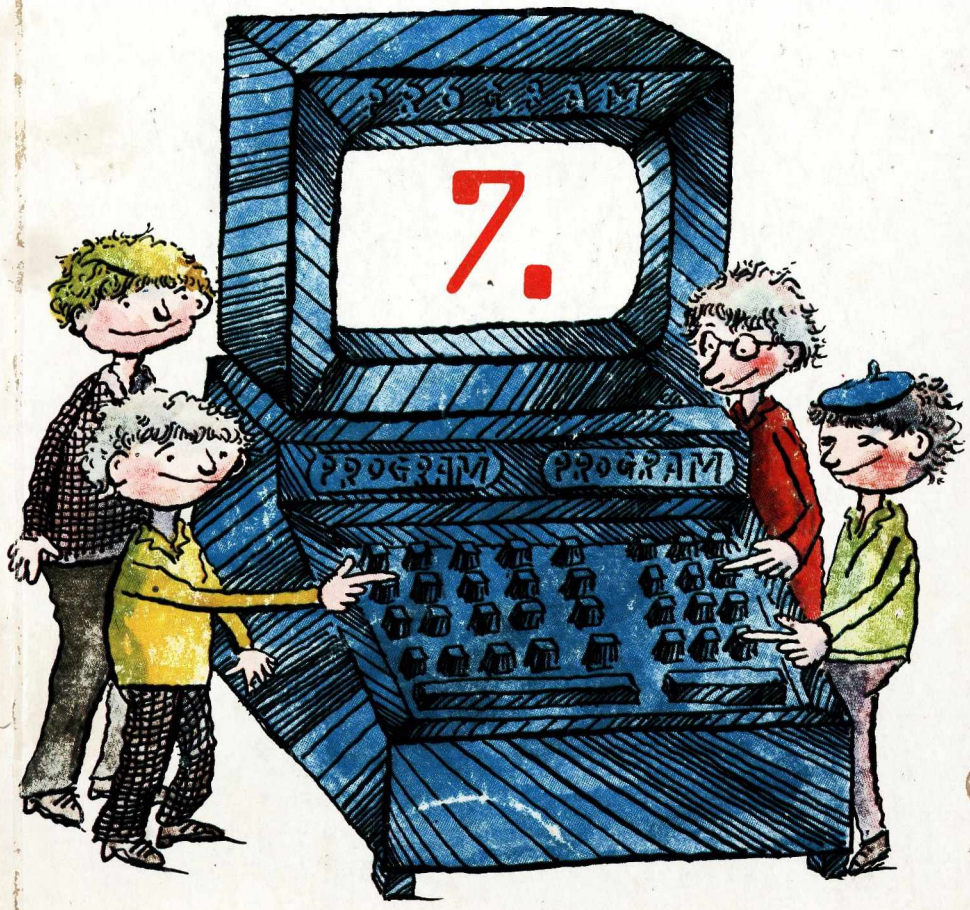
Országos Pedagógiai Intézet
Fővárosi Pedagógiai Intézet Számítástechnikai Sorozat

HT–1080 Z ISKOLAI SZÁMÍTÓGÉP

INFORMATIKA



INFORMATIKA

HT–1080 Z ISKOLAI SZÁMÍTÓGÉP

Szerkesztette: dr. Appel György
Lektorálta: Marosvári Erika

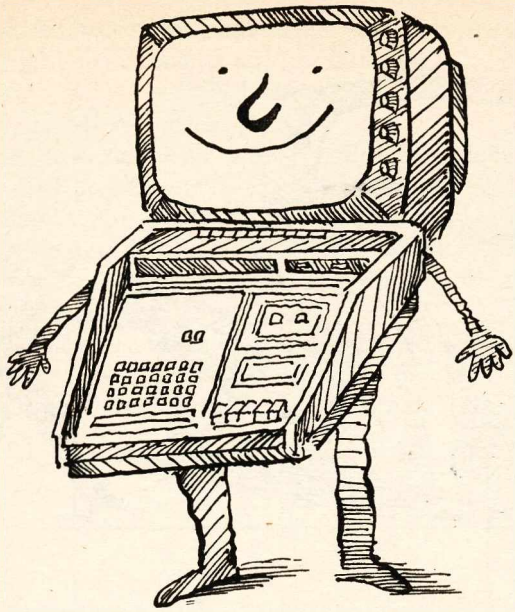
Országos Pedagógiai Intézet és a Fővárosi Pedagógiai Intézet számítástechnikai sorozata
VII.

INFORMATIKA

HT–1080 Z ISKOLAI SZÁMÍTÓGÉP

használata kezdő szakkörök részére
(5–8. osztályosok részére)

IFJÚSÁGI LAP- ÉS KÖNYVKIADÓ VÁLLALAT
1985



Bevezető

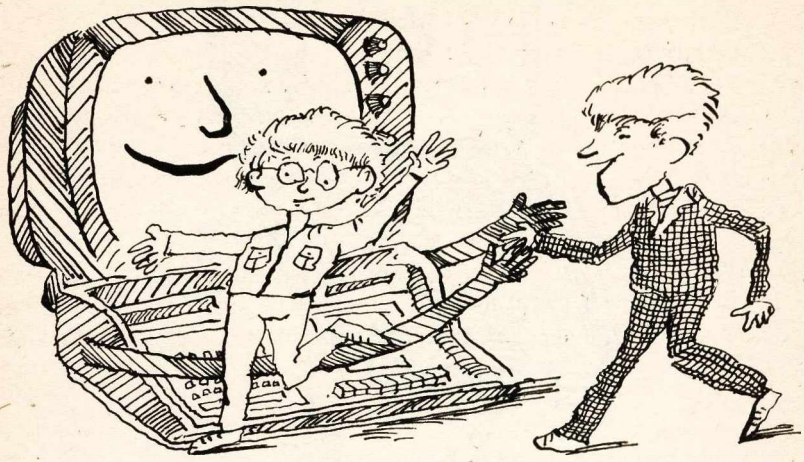
– Örülök, hogy találkoztunk! Először is az ismeretlennek kell bemutatkoznia. Csak egy pár szót magamról:
Nevem: HT–1080 Z iskolai számítógép
Gyártóm: a HÍRADÁSTECHNIKA SZÖVETKEZET
Programnyelvem: a BASIC

Központi egységembe egy Z80-as processzor került elhelyezésre. *Kapacitásom:* 12 K, 1,5 Kbyte-os különleges bővítéssel. Így tudom felismerni a kis- és nagybetűket, a villogó cursort, a gépi kódú monitort, az újraszámozó funkciót. És nem utolsósorban magnetofonos egységem nagy tömegű adat tárolására alkalmas.

Ezenkívül a programozható hanggenerátorom lehetőséget nyújt a programok zenével való színesebbé tételére.

De ne essünk neki rögtön a programozásnak.

Ismerkedjünk meg először néhány számítástechnikai alapfogalommal.



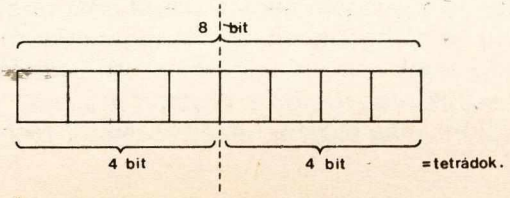
HARDVER: a számítógép állandó mechanikai, elektromos, elektronikus egységeinek, beépített alkatrészeinek összessége.

SZOFTVER: a számítógép működésének szellemi feltétele, ide tartoznak maguk a programok – BASIC-program.

ADAT: a feldolgozandó információ legkisebb, de még önállóan is kezelhető, értelmes része. Tartalmuk alapján két fajtája ismert, a numerikus (számokat és esetleg előjeleket tartalmaz), és az alfanumerikus (betűket, számjegyeket és különleges karaktereket is tartalmazhat).

Bit: a bináris számjegyek, azaz a 0 és az 1-es számjegyek közös neve.

BYTE: 8 bitből álló egység.



Egy byte-tal két tízes számrendszerbeli számjegy, vagy egy karakter (alfabetikus jel vagy speciális jel) ábrázolható.

1 (kilobyte) Kbyte = 1024 byte

1 (megabyte) Mbyte = 1024 Kbyte.

KARAKTER: a karakter az írott közölnivaló, az írott információ legkisebb elemi része, vagyis egy kifejezési eszköz. Karakteren így általában egy olyan írásjelet, jelsorozatot értünk, amely az egyes nyelvek jelkészlete által rögzített betű, számjegy, írásjel vagy speciális jel megjelenítésére, írásbeli kifejezésére szolgál. Önálló jelentése természetesen már nincs.

SZÁMRENDSZER: a számítástechnikában alapvetően fontos, gyakorlati szerepe van. A mennyiségeket a helyiértékes írásmódban különféle (egész) alapszámok hatványainak összegével írjuk fel.

10-es decimális	2-es bináris	8-as oktális	16-os hexadecimális
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27...	11011	33	1B

PROGRAM: a program utasítások sorozata, amelyeket a számítógépnek valamilyen sorrendben, egymás után kell elvégeznie. A program végrehajtására az összes utasítás leírása után egy külön parancs szolgál: a RUN.

PARANCS: a parancs a számítógépet valamilyen tevékenység elvégzésére szólítja fel, amelyet azonnal végre kell hajtania.

UTASÍTÁS: az utasítás a számítógépet valamilyen tevékenység elvégzésére szólítja fel, amelyet akkor kell végrehajtania, ha az utasítássorozatban az adott utasításhoz ért.

Gyakran használt számítástechnikai kifejezések

ASSEMBLER: egy számítógépen futó program, amely az assembly szinten megírt programot átalakítja a gép által érthető kódokká (gépi nyelvre).

CHIP: (morzsa) egy teljes integrált áramkört tartalmazó lapocska.

CIKLUS: utasítás vagy utasítássorozat folyamatos ismétlése mindaddig, amíg a befejező feltétel be nem következik.

CÍM: olyan azonosító kód, amely egyrészt a memóriaregisztereket, másrészt a beviteli–kiviteli eszközöket megkülönbözteti egymástól, ezek kiválasztására is ezt használjuk.

CPU: (Central Processing Unit) a számítógép központi egysége, amely a vezérlő- és az aritmetikai egységet tartalmazza.

EBCDIC: (Expanded Binary Coded Decimal Interchange Code) 8 bites karakterkód, melyet leggyakrabban nagy számítógépekben használnak.

FLAG: (jelzőbit) – feltételkód – olyan bit, amely a számítógépen belül egy bizonyos feltételt jelez. Ezeket leggyakrabban programok elágaztatására használjuk.

FORTRAN: magas szintű (eljárásorientált) programnyelv, amely különösen alkalmas tudományos műszaki problémák megoldására.

IC: (integrált áramkör) teljes áramkör egyetlen chipen.

MAGAS SZINTŰ PROGRAMNYELV: olyan programnyelv, amelyben egy utasítás nem egyetlen gépi utasítást, hanem egy teljes eljárást jelent. A legáltalánosabban elterjedt magas szintű programnyelvek a FORTRAN, BASIC, COBOL, PL/1. A magas szintű programnyelvek olyan fordítóprogramot igényelnek, ame-

lyek az egyes utasításokat gépi utasítások sorozatának fordítják le.

MEMÓRIA: a számítógépnek az adatok és utasítások tárolására szolgáló része. Mindegyik memóriarekesz külön címmel rendelkezik az egyértelmű hivatkozás érdekében.

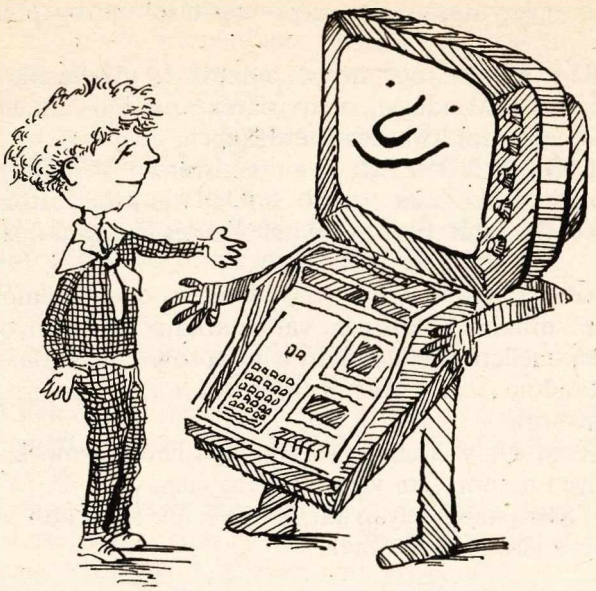
OPERÁCIÓS RENDSZER: olyan rendszerszoftver, amely a számítógép teljes működését vezérli, és feladatai közé tartozik például a memóriakiosztás, bevitel–kivitel vezérlés, megszakítás–feldolgozás.

PROM: (Programable Read Only Memory) olyan memória, amely a normál működés alatt nem változtatható, de bizonyos speciális feltételek mellett a felhasználó is újraprogramozhatja.

RAM: (Random Access Memory) véletlen hozzáférésű írható–olvasható memória.

ROM: (Read Only Memory) csak olvasható memória: tartalma semmilyen módon sem változtatható meg.

VEREMTÁR: (stack) olyan tár, amelyet pl. szubrutin visszatérési címek tárolására használnak.

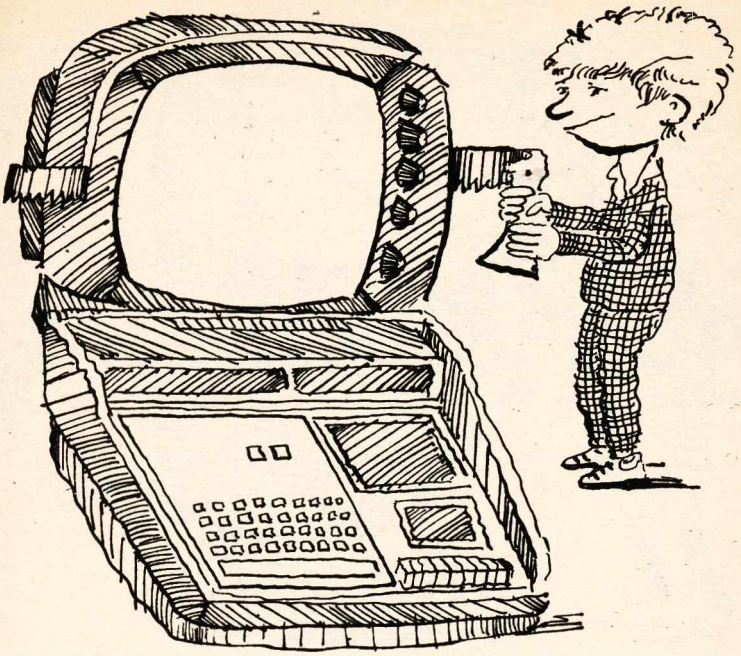


A GÉP BEKAPCSOLÁSA

Nem kell félned, a gépnek semmi baja sem történhet, ha betartod a következő sorrendet.

1. Győződj meg arról, hogy a hálózati kapcsoló ki van-e kapcsolva!
2. Ellenőrizd, hogy a tápegységen feltüntetett feszültségérték megegyezik-e a rendelkezésre álló hálózati feszültség értékével!
3. Dugd be a hálózati csatlakozót egy hálózati aljzatba!
4. Csatlakoztasd a gépet a tévékészülékhez!
5. Kapcsold be először a tévékészüléket, majd a számítógépet!
6. A képernyő bal felső sarkában a „READY ?–" szöveg jelenik meg.
7. Ha a „READY ?–" szöveg nem jelenik meg, térj vissza az 1-es ponthoz.
8. A „READY ?–" megjelenése után nyomd meg a NEW LINE gombot, erre a „READY" szó a képernyő bal alsó sarkába kerül. Ekkor a gép készen áll arra, hogy kívánságaidat teljesítse.

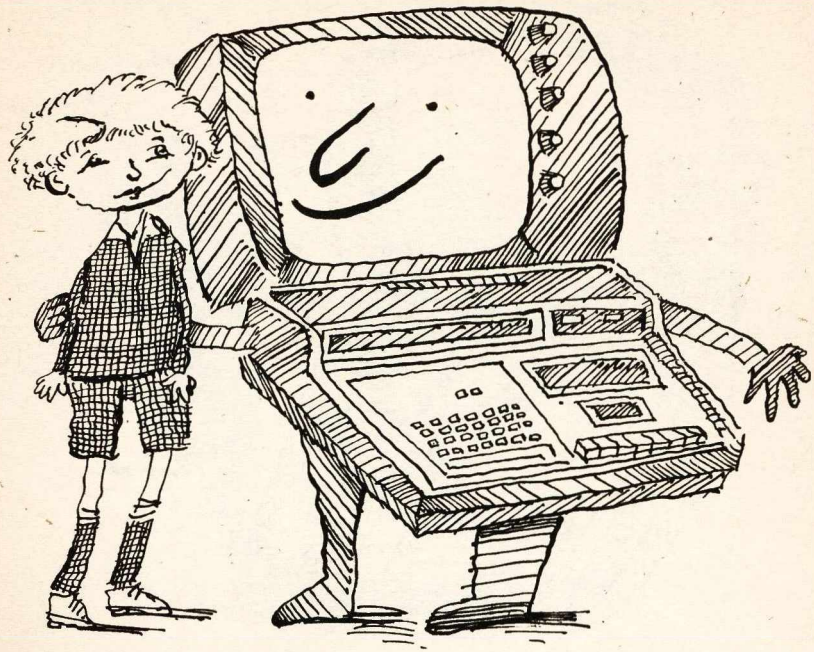
Ha a gépet valamilyen okból kikapcsolod, legalább 15 másodpercet várj az újabb bekapcsolásig!



A BILLENTYŰZET

A kezelőgombok szerepe:

PAGE = benyomására a képernyő jobb oldali részét látod.
F1 = kézi előre- vagy hátratekercseléskor a kazettát leválasztja a számítógépről.
BREAK = megszakítja az éppen futó programot.
NEW LINE = lezárja a parancs vagy adat sorát.
← = törli az utolsónak bevitt karaktert.
↑ = a display megjelenít egy jelet, mely az exp.-jel.
↓ = a cursort a következő sor elejére ugratja.
→ = tabulátor funkció: a cursort 8 betűközzel jobbra viszi.
SHIFT → = hatására a karakterek közé automatikusan szóköz iktatódik be. Ez a NEW vagy CLS paranccsal törlődik.
SHIFT ← = sortörlés



A KÉPERNYŐ KEZELÉSE

VIDEO CUT és PAGE

VIDEO CUT = magyarul: képet vág
PAGE = magyarul: oldal

A **VIDEO CUT** feliratú gomb a gép hátoldalán található. Kiugratott helyzetben a képernyőn 64 karakter látható. A gomb benyomásával a képernyőn 32 karakter látható.

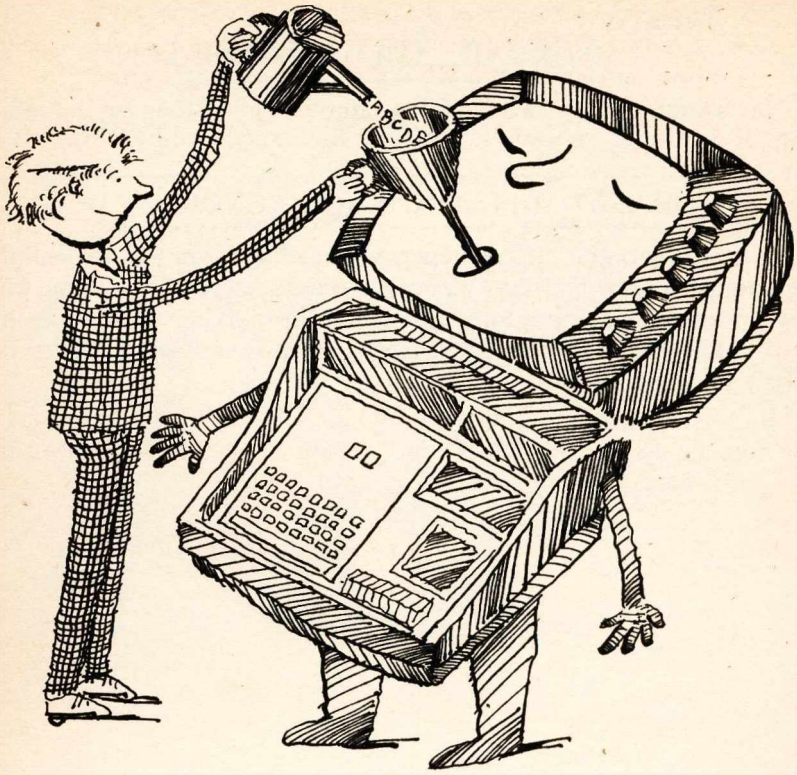
Ha a **PAGE** gombot benyomod, akkor – 32 karakteres formátum esetén – a képernyőnek csak a jobb oldali részét látod.
Például: írjunk be egy mondatot!

```
READY
>ITT ULOK CSILLAMLO SZIKLAFALON.
```

Ha ez kész, nyomd be a **PAGE** gombot, és írj be még egy mondatot:

```
OH, HAT MIFELE ANYAG VAGYOK EN,
```

Ez utóbbi mondat tehát a képernyő másik oldalán látható, mindkét mondat összesen 2-szer 32, azaz 64 karaktert foglal magába. Ha egy sorban akarjuk látni mind a két mondatot, akkor az előbb említett **VIDEO CUT** gomb megnyomására szép sovány betűkkel kiíródik.



A PROGRAM BETÖLTÉSE

A program a kazettán mágneses jelek formájában van rögzítve. A megírt programot kazettára való felvétel után bármikor visszatöltheted a számítógép memóriájába.

A program betöltésénél is tartsd be a helyes sorrendet!

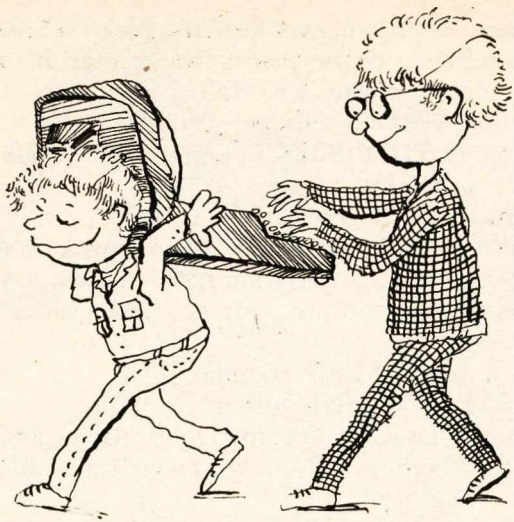
A program betöltését a **CLOAD** parancs szolgálja.

A betöltés szabályai a következők:

1. Helyezd a kazettát a kazettatartóba.
2. Zárd le a kazettatartót.

3. Ha szükséges, tekerd vissza a kazettát a következőképpen:
   - Ha a számítógép be van kapcsolva, nyomd be az **F1** feliratú gombot, majd a magnó **REWING** gombját.
   - Várj, amíg a visszacsévélés befejeződik.
   - Nyomd meg a **STOP/EJECT** gombot, amely leállítja a magnót. Nullázd a számlálót.
   - Engedd fel az F1 gombot.
4. Billentyűzd be a **CLOAD** parancsot. Ha a programnak neve van, akkor: **CLOAD**#–1, „a program neve". Ekkor a gép megkeresi a programnévvel ellátott programot, és innen következik a betöltés.
5. Nyomd be a magnó **PLAY** gombját.
6. Nyomd meg a **NEW LINE** gombot.

A program betöltésekor a képernyő jobb felső sarkában két csillag jelenik meg. Ha a betöltés kész, akkor megjelenik a **READY** üzenet.

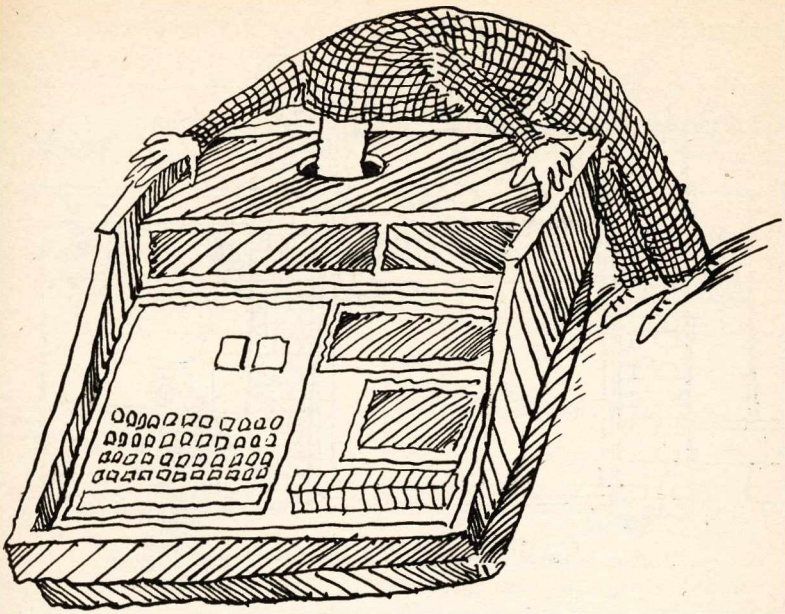


A PROGRAM FELVÉTELE

A program felvételének helyes sorrendje a következő:

1. Nyomd meg az **F1** gombot. (A kazetta melletti piros lámpának is égnie kell.)
2. Nyomd be a **REWING** gombot, és állítsd arra a helyre, ahová a programot fel akarod venni.
3. Nyomd meg a **STOP/EJECT** gombot, hogy megállítsa a szalagcsévélést.
4. Most egyszerre nyomd meg a **PLAY** és a **RECORD** gombot, hogy a felvétel előtt letöröljön egy részt a szalagról.
5. Nyomd meg újra az **F1** gombot. (A kazetta melletti piros lámpának el kell aludnia.
6. Billentyűzd be a **CSAVE** parancsot: **CSAVE#–1**, „a program neve".
7. Nyomd meg egyszerre a **RECORD** és a **PLAY** gombot.
8. Végül nyomd meg a **NEW LINE** gombot.

A kazetta melletti piros lámpa erre automatikusan kigyullad, és a kazettára kerülnek a számítógéptől érkező jelek. Ekkor nem jelenik meg csillag a képernyőn. A felvétel befejeztével a **READY** felirat kerül a display-re.



A PROGRAM ELLENŐRZÉSE

A programellenőrzés abban áll, hogy a számítógép a kazettán levő programot összehasonlítja a saját memóriájában levővel. Ha az összehasonlítás során a gép nem talál hibát, a jobb oldali csillag villog, a befejezést **READY** üzenet jelzi. Hiba esetén a **BAD** szó jelenik meg a képernyőn.

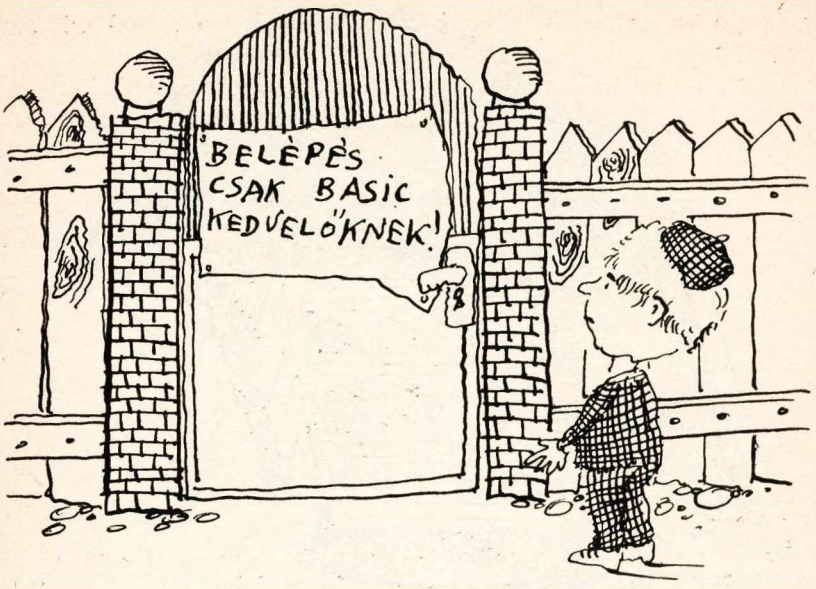
A program ellenőrzését a **CLOAD?** parancs végzi el.

A programellenőrzés elvégeztetésének művelete a következő:

1. Csévéld a szalagot a program elejére.
2. Nyomd meg a magnó **PLAY** gombját.
3. Billentyűzd be a **CLOAD?** parancsot.
4. Nyomd meg a **NEW LINE** gombot.

Ha a képernyőn a **BAD** üzenet jelenik meg, akkor

5. tekercseld vissza a programot, és a **CSAVE** parancs segítségével még egyszer fel kell venni.



A HT–1080 Z PROGRAMOZÁSA

A **CLOAD**, a **CLOAD?**, és a **CSAVE** parancsokat már megismerted. Most pedig a következő parancsok mutatkoznak be:

1. **AUTO**
2. **CLEAR**
3. **CONT**
4. **DELETE**
5. **EDIT**
6. **LIST**
7. **NEW**
8. **RUN**
9. **SYSTEM**
10. **TROFF**
11. **TRON**
12. **RE**

EDIT

Ez a szó magyarul azt jelenti: szerkeszt.

Valóban. Az **EDIT** parancsot szövegszerkesztésnél, javításra tudjuk használni.

Beírandó:

EDIT 50 (ha az 50. programlépést akarom javítani)

EDIT (ha az éppen beírt sort akarom kijavítani)

Ha a programozás közben egy mondatot, szót vagy éppen csak

egy karaktert elrontunk, gépünk különféle módon tudja helyesbíteni tévedésünket.

A H, I, X, L, A, E, Q, D, C, S, K betűk fontos szerepet töltenek be a javítás során. Most megtanuljuk minden egyes „kulcs-betűnek" a jelentőségét.

1. ← **NEW LINE**
E két gomb egyidejű benyomására törlődik a rosszul beírt sor:

Billentyű	Képernyő
PRINT"EGYSZERU JAVITAS NEW LINE	READY >– PRINT"EGYSZERU JAVITAS READY >–

2. **SPACE**
Szerkesztő üzemmódban ha ezt a szóközbillentyűt lenyomod, minden egyes megnyomásra a kép 1 karaktert ír ki, mindaddig, amíg tart a javítandó sor.

Billentyű	Képernyő
10PRINT"SHOOL COMPUTER" NEW LINE EDIT10 NEW LINE SPACE SPACE SPACE ⋮ SPACE	READY >– >10PRINT"SHOOL COMPUTER"– >– >EDIT10– 10– 10P– 10PR– 10PRI– ⋮ 10PRINT"SHOOL COMPUTER"–

Mielőtt azonban továbbmennénk, egy icipici kitérőt teszünk. Megismerkedünk a **RUN** paranccsal, és a **PRINT** utasítással.

RUN

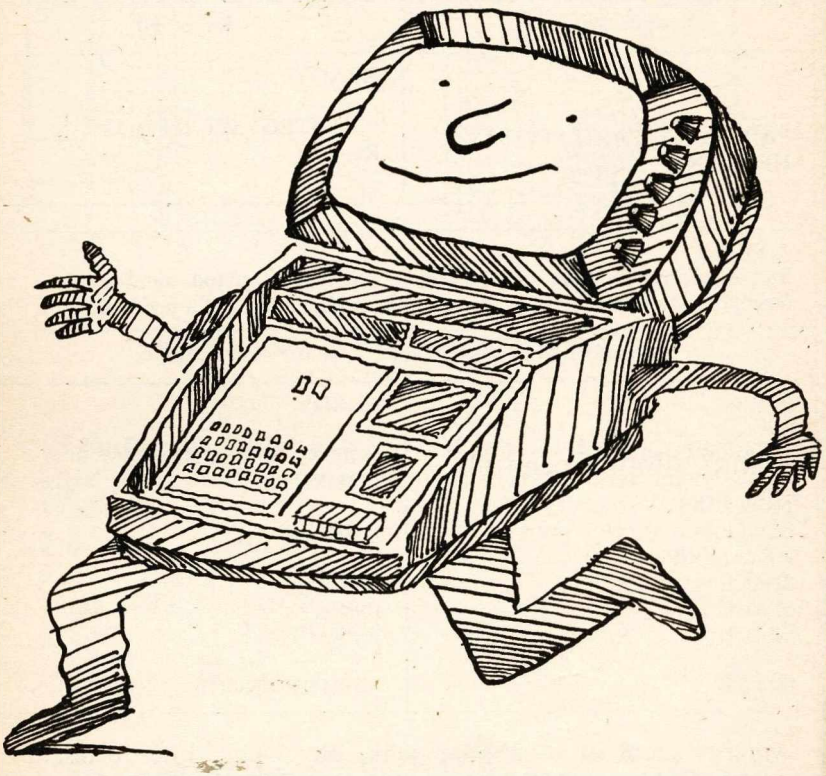
Magyarul azt jelenti: fut, szalad.

Ezt a parancsot kell adni a gépnek akkor, ha a beírt programot futtatni akarjuk.

A program futását a gép a legalacsonyabb sorszámú utasítástól kezdi, vagyis az elejétől. Ha a **RUN** parancs mellé egy program sorszámát írjuk be, akkor a programunk ettől a sorszámtól fut. Például:

a) **RUN** (a program elejétől futtatunk)

b) **RUN 50** (a program 50. programlépésétől futtatunk).



PRINT

Azt jelenti: kinyomtat, kiír.

Ezzel a **PRINT** utasítással különféle változókat, tételt vagy tételsorozatot írhatunk ki a képernyőre.

A képernyőre kiírandó karaktersorozatot mindig idézőjelbe kell írni. A következőképpen: **10PRINT"KARAKTER"**

A **PRINT** utasítás részletesebb megismerésével később foglalkozunk.

Most pedig térjünk vissza az **EDIT** parancs további rejtelmeihez.

3. **H**

Most úgy szeretném kijavítani a beírást, hogy csak a "SHOOL" rész maradjon meg a sorban.

A **H** parancs hatására a cursortól jobbra eső szövegrészt törli a gép.

Billentyű	Képernyő
←	10PRINT"SHOOL COMPUTER"–
←	10PRINT"SHOOL COMPUTER–
←	10PRINT"SHOOL COMPUTE–
←	10PRINT"SHOOL COMPUT–
⋮	⋮
	10PRINT"SHOOL–
H	10PRINT"SHOOL–
SHIFT "	10PRINT"SHOOL"–
NEW LINE	10PRINT"SHOOL"
	>–

4. **I**

Bizonyára észrevettétek, hogy SHOOL szó helytelenül van leírva. Ezért most közé kell szúrnom egy „C" betűt.

Ebben segít az **I** parancs, ugyanis ennek a hatására egy karaktert írhatok be a sorba, és a sor többi része változatlan marad.

Billentyű	Képernyő
	10PRINT"SHOOL"
	>–
EDIT 10	>EDIT 10–
NEW LINE	10–
SPACE	10P–
SPACE	10P–
⋮	⋮
SPACE	10PRINT"S–
I	10PRINT"S–
C	10PRINT"SC
NEW LINE	10PRINT"SCHOOL"
	>–

5. **X**
Most úgy javítom ki a sort, hogy mellé írom a COMPUTER szót.
Az **X** parancs hatására a sor végére tudok beírni, a következő módon:

Billentyű	Képernyő
	10PRINT"SCHOOL" >–
EDIT 10	>EDIT 10
NEW LINE	10–
X	10PRINT"SCHOOL"– 10PRINT"SCHOOL–
COMPUTER	10PRINT"SCHOOL COMPUTER"–
NEW LINE	>–

6. **L**
Erre a parancsra kilistázhatjuk a cursortól jobbra eső részt, azaz az egész sort.

Billentyű	Képernyő
EDIT 10	>EDIT 10
NEW LINE	10–
L	10PRINT"SCHOOL COMPUTER" 10–
L	10PRINT"SCHOOL COMPUTER" 10–
L	10PRINT"SCHOOL COMPUTER" 10–
NEW LINE	10PRINT"SCHOOL COMPUTER" >–

7. **A**
Ha javítás közben meggondolom magam, és mégsem akarok javítani, akkor az **A** parancs hatására töröl minden változtatást, és csak az eredeti szöveget veszi figyelembe.

Billentyű	Képernyő
EDIT 10	>EDIT 10
NEW LINE	10–
SPACE...	10PRINT"SCHOO
A	10–
NEW LINE	10PRINT"SCHOOL COMPUTER" >–

8. **C**
Ezzel a paranccsal egy karaktert kicserélhetünk másik karakterre.

Billentyű	Képernyő
EDIT 10 NEW LINE SPACE... C K NEW LINE	>EDIT 10– >– 10PRINT"SCHOOL – 10PRINT"SCHOOL – 10PRINT"SCHOOL K– 10PRINT"SCHOOL KOMPUTER" >–

9. **E**
Erre a parancsra a gép tárol minden változtatást. Az **E** hatására visszaáll parancs üzemmódba, új programlépésre készen áll.

Billentyű	Képernyő
EDIT 10 NEW LINE E RUN NEW LINE	>EDIT 10 10– >– >RUN SCHOOL KOMPUTER READY >–

10. **Q**
Az előbb láttuk, hogy az **E** parancs tárolta a változtatott szöveget. Ha még a javítást is elrontanám, akkor a **Q** parancs segítségével „hatástalaníthatom".

Billentyű	Képernyő
EDIT 10 NEW LINE SPACE... C C Q RUN NEW LINE	>EDIT 10 10– 10PRINT"SCHOOL – 10PRINT"SCHOOL – 10PRINT"SCHOOL C READY >– >RUN SCHOOL KOMPUTER READY >–

11. **D**

Ha a gépnek **D** parancsot adok szövegszerkesztési módban, akkor a cursortól jobbra eső összes karaktert törli a mondatból. A törölni kívánt karaktereket a gép két felkiáltójel közé teszi.

Billentyű	Képernyő
	10PRINT"SCHOOL KOMPUTER" >–
EDIT 10	>EDIT 10
NEW LINE	10–
SPACE...	10PRINT"SCHOOL –
8	10PRINT"SCHOOL –
D	10PRINT"SCHOOL !KOMPUTER!"
NEW LINE	>–
RUN	>RUN–
NEW LINE	READY >–

12. **S**

Most az a kívánságom, hogy a maradék szóból törölje ki az első „O" betűt. Az **S** parancs hatására a gép megkeresi az O betűt, ahol először szerepel, és azt törli ki.
Figyeld csak!

Billentyű	Képernyő
	READY >–
EDIT 10	>EDIT 10–
NEW LINE	10–
1	10–
S	10–
O	10 PRINT"SCH–
D	10 PRINT"SCH!O!–
NEW LINE	10 PRINT"SCH!O!OL " >–
RUN	>RUN
NEW LINE	SCHOL READY >–

13. **K**

Ha azt kérem a géptől, hogy a szó egy bizonyos betűjétől kezdve töröljön minden további karaktert, ezt a **K** parancs hatásával érhetem el. A gépet arra utasítom, hogy az általam megadott betűt keresse meg. Ez a következőképpen néz ki:

Billentyű	Képernyő
	SCHOL READY >–
EDIT 10	>EDIT10
NEW LINE	10 –
1	10 –
K	10 –
P	10 !PRINT"SCHOL "!–
NEW LINE	>–
RUN	>RUN
NEW LINE	READY >–

Ezzel elérkeztünk a szövegszerkesztés befejezéséhez. Jelenleg a képernyőnk tele van az összevissza javítgatott sorokkal. Ez a további munkánkat zavarhatja. A **NEW** paranccsal törölhetjük az egész képernyőt.

NEW

Ez a parancs töröl minden programsort, és nullázza az összes változót. Hatására a képernyő bal felső sarkában megjelenik a **READY** üzenet, várva a további parancsot, utasítást.
Begépelési formája:

Billentyű	Képernyő
NEW NEW LINE	>NEW– READY >–

DELETE

Ez a parancs a következőképpen törli a sort, vagy akár az egész programsorokat:

DELETE 5 (csak az 5-ös sort törli)

DELETE 5–50 (5-től 50-ig minden sort töröl)

DELETE –60 (a program elejétől a végéig minden sort töröl).

Remélem, hogy a szövegmódosítás már kitűnően megy! Ezek után játsszunk egy kicsit a számítógépünkkel!

Írjunk egy egészen icipici programot! Megtanuljuk azokat az utasításokat, amelyek segítségével felépíthetjük nyúlfarknyi programunkat.

Ilyen utasítás a **PRINT**.

Erről az előbbiekben már szó volt, most viszont bővebben áttekintjük.

PRINT

Mit lehet kiíratni a **PRINT** utasítással? A következőket:

a) numerikus változókat (3.14159, 345, 1.23, −3,4)

b) szöveges változót (A $, B $)

c) indexes változókat (A(1), A(9), B(2), B(5))

Akkor vegyük sorba.

a) *Egyszerű, numerikus változó*

A változónév az angol ABC bármelyik betűje lehet, vagy két betűje, vagy egy betű és egy egyjegyű szám: AB, A, A2.

Billentyű	Képernyő
10 AB= 567:A=345:A2= 3*5+8 NEW LINE 20 PRINT AB;A;A2 NEW LINE RUN NEW LINE	READY >10AB=567:A=345:A2= 3*5+8_ >10AB=567:A=345:A2=3*5+8 >20 PRINT AB;A;A2;_ >20 PRINT AB;A;A2; >RUN 567 345 23

b) *Szöveges, stringes változó*

Jele: $. A numerikus változóval számokat, értékeket írattunk ki, most pedig szöveget, karaktersorozatot tudok kezelni. A stringek karakterláncok, tehát betűkből, számjegyekből, jelekből állhatnak. A stringek tárolására használt változókat stringváltozónak nevezzük. A karaktersorozatot mindig idézőjelbe kell tenni. Nézzünk egy példát:

Billentyű	Képernyő
10 A $ = "PIFF ES A HT" NEW LINE 20 PRINT A$ NEW LINE RUN NEW LINE	READY >10 A$="PIFF ES A HT"_ >10 A$="PIFF ES A HT" >20 PRINT A $_ >20 PRINT A$ >RUN PIFF ES A HT READY

c) *Indexes, tömbös változó*

A tömb egy rendezett adatlista. Ez állhat szám- és stringtömbökből egyaránt. Egy tömbben azonban csak szám, vagy csak string állhat.

A tömb lehet egydimenziós:

A 1	150
A 2	250
A 3	350

Billentyű	Képernyő
10 A (1) = 150 20 A (2) = 250 30 A (3) = 350 40 PRINT A(1); A(2); A(3); RUN NEW LINE	READY >10 A (1) =150 >20 A (2) =250 >30 A (3) =350 >40 PRINT A(1); A(2); A(3); >RUN 150 250 350 READY >–

(A **NEW LINE** gomb lenyomását itt már nem írtam ki, és a későbbiekben sem fogom feltüntetni!)

És lehet kétdimenziós:

	0	1	2
A 0	150		
A 1		250	
A 2			350

Billentyű	Képernyő
10 A (0,0) =150 20 A (1,1) =250 30 A (2,2) =350 40 PRINT A (0,0); A (1,1); A (2,2) RUN	READY >10 A (0,0) = 150 >20 A (1,1) = 250 >30 A (2,2) = 350 >40 PRINT A (0,0); A (1,1); A (2,2) >RUN 150 250 350

Nézzük meg, mit kell tenni akkor, ha 10-nél nagyobb tömböt szeretnék használni. A gép max. 10-ig fogadja el ilyen formában az indexes változót. Tehát 10-nél nagyobb dimenzió megadása esetén a program megáll, és BS (a megadottnál nagyobb dimenzió) jelű hibát jelez.

Ilyen dimenzió használatánál a DIM utasítást kell használni.
A DIM utasítás *jelentése:* méret, kiterjedés.

A 11	150
A 12	250
A 13	350

Billentyű	Képernyő
5 DIM A(13) 10 A(11) = 150 20 A(12) = 250 30 A(13) = 350 40 PRINT A(11); A(12); A(13) RUN	READY 5 DIM A(13) 10 A(11) = 150 20 A(12) = 250 30 A(13) = 350 40 PRINT A(11); A(12); A(13) RUN 150 250 350

Két dimenziónál:

	11	12	13
A 11,	150		
A 12,		250	
A 13,			350

Billentyű	Képernyő
5 DIM A(13,13) 10 A(11,11)=150 20 A(12,12)=250 30 A(13,13)=350 40 PRINT A(11,11); A(12,12); A(13,13) RUN	READY > 5 DIM A(13,13) >10 A(11,11)= 150 >20 A(12,12)= 250 >30 A(13,13)= 350 >40 PRINT A(11,11); A(12,12); A(13,13) >RUN 150 250 350

PROGRAM! Tömb kiíratása:

```
10 DIM A (6,12)
20 FOR I= 0 TO 6
30 A (I,0)=I
40 FOR J=0 TO 12
50 A(0,J)=J
60 PRINT A (I,J) ;
70 NEXT J:PRINT : NEXT I
```

Stringes tömb kiíratásánál ugyanígy kell eljárni. Túl nagy szám esetén, vagy stringes tömbök alkalmazásánál a program futtatásakor **OM,** vagy **OS** jelű hiba jelentkezik.

OM = Nincs több tárolóhely.

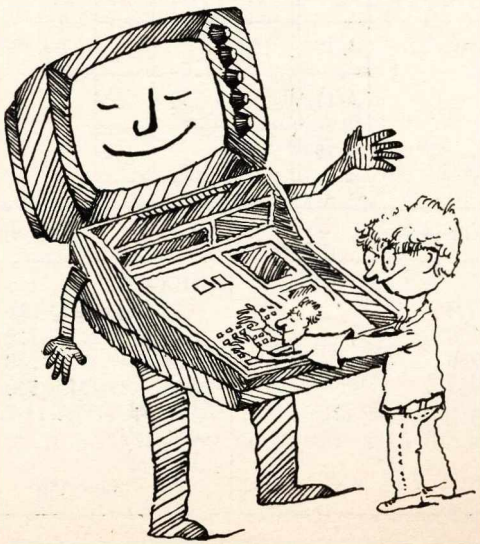
OS = Nincs több hely stringek részére.

Ilyenkor előre kell helyet foglalni a számítógép tárában. Hogyan?

Amikor a gépet bekapcsoljuk, és lenyomjuk a NEW LINE gombot, abban a pillanatban 50 byte szabadul fel, azaz minden byte-ba 0-t rendel. Amikor ez az 50 byte kevés, akkor a **CLEAR** utasítást kell használni.

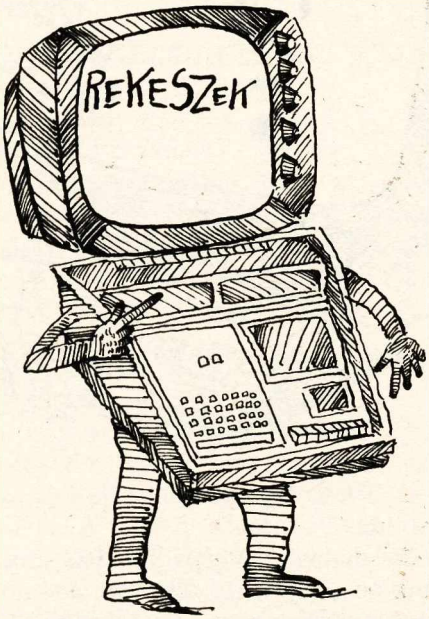
A program elejére kell beírni, hogy: **10 CLEAR 1000.**

Erre az utasításra a gép 1000 byte-ot tesz szabaddá a program számára. (A későbbiekben már nem táblázatos formában írom a programsorokat!)



LET = Értékadó utasítás

Eddig nagyon sokat beszéltem a változókról, változók kiíratásáról. De mit is jelent valójában az a szó, hogy *változó?* A gép memóriája apró, ún. kis rekeszekre van osztva. Képzelj el egy fiókos szekrényt, valahogy így. Legegyszerűbben úgy tudjuk megjegyezni, hogy mi van benne, ha a fiókokat megcímezzük, azaz *változónévvel* látjuk el. A gép számára ez azért kell, hogy a program futásánál majd ezekkel a változókkal hivatkozhassunk a fiókok tartalmára.
Nézzük meg a következő kifejezést:

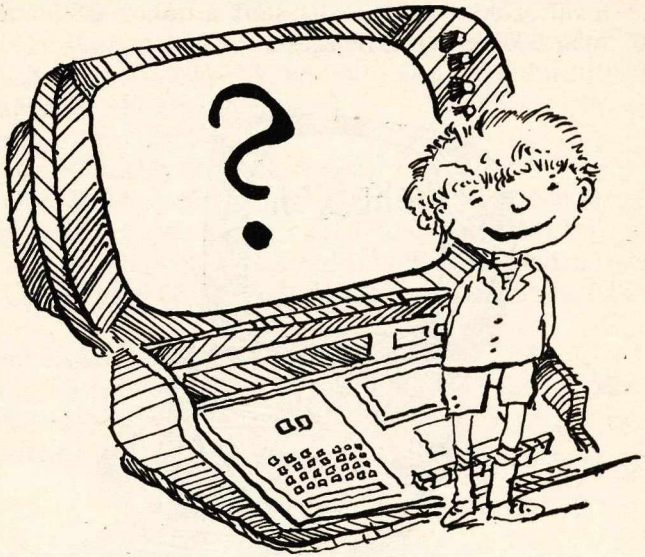


Y	=	5 * 6 / 2

változó legyen egyenlő aritmetikai kifejezés

Amikor a gép egy ilyen utasítást kap, az egyenlőségjel jobb oldalán lévő kifejezést kiszámolja, és az „Y" változóhoz rendeli, egyetlen értékként. Írjuk be a gépbe:

```
10 Y = 5 * 6 / 2    és írassuk ki a képernyőre.
20 PRINT Y
RUN
15
```



INPUT = adatbevitel

Az INPUT utasítás hatására a gép addig vár, ameddig valamilyen adatot nem viszünk be a gépbe. Ez lehet szám, vagy akár szöveg is. A gép várakozását a képernyőre kitett kérdőjellel jelzi. Minden további **INPUT** hatására két kérdőjel (? ?) jelenik meg a képernyőn.

A 20. oldalon azt mondtam, hogy „írjunk egy icipici programot". Nos, most már tudunk annyit, hogy programot készíthetünk a kör kerületének kiszámítására.

PROGRAM!

```
10 INPUT R
```

– Bekérem a sugár értékét „R" változóval jelölve.

20 K=2*R*3.14159 – „K" változóval jelöltem azt a „fiókot", amely a kerület értékét fogja tartalmazni.

30 PRINT K – Kiíratom a képernyőre a K-fiók tartalmát.

RUN – Futtatom a programot.

? 87 – Várja a gép a sugár értékét.

546.637 – Beírom, hogy pl.: 87

A kör kerülete tehát: 546.637

Most ugyanezt a programot készítsük el, de valamivel szebben!

```
10 INPUT "MENNYI A SUGAR="; R
20 K=2*R*3.1415927
30 PRINT"A KOR KERULETE"; K; "CM"
RUN
MENNYI A SUGAR=? 87
A KOR KERULETE 546.637 CM
```

Ugyanazt a programot láttuk most is, csak szebb kiírással. A program írásánál mindig a szebb formákhoz ragaszkodj!

A szöveg mindig az INPUT után következik, és mindig idézőjel közé kell tenni a kiírandó karaktersorozatot. Az idézőjel után mindig pontosvesszőt kell rakni, és csak így következhet a változó!

DATA – READ – RESTORE

Adatbevitel forma.

3 féleképpen tudunk már adatot kezelni.

az első: az értékadás volt,

a második: az INPUT utasítás segítségével, és most a

harmadik: a DATA utasítás használatával.

Data azt jelenti: adatok. **DATA**-val szintén lehet bevinni numerikus és szöveges adatokat. Írjunk egy példát:

10 READ A,B, A$, B$ – *READ utasítás kiolvassa* a DATA-ban lévő értékeket. A READ után mindig valamilyen változó áll!

20 DATA 5,8,ALMA,KÖRTE – Megadom az értékeket. A változók között csak vessző állhat!

30 PRINT A,B, A$, B$ – Kiíratom a numerikus és a szöveges adatokat a képernyőre.

Ha a **READ** utasításban több változót jelölök meg, mint amennyi a **DATA**-ban van, OD jelű hibát jelez a gép.

OD = nincs több adat.

Ha a DATA-ban van több, azt figyelmen kívül hagyja.

```
RUN                              – futtatom a programot.
 5  8  ALMA  KÖRTE
```

Most nézzük meg, mit jelent a RESTORE utasítás.

Írjuk be a következő programot:

```
10 READ A
20 DATA 500
30 PRINT A
40 READ B$
50 DATA MIKULAS
60 PRINT B$
RUN
500
MIKULAS
```

Most változtassuk meg a programot azzal, hogy 35-ös programlépésre beírunk egy **RESTORE** utasítást.

```
35 RESTORE
RUN
23
23
```

Láthatod, hogy ebben az esetben nem írta ki a mikulást, mivel – és ez a **RESTORE** lényege – visszaállítottam a kezdőértéket, és így csak az első **READ** utasítást hajtotta végre. Kétszer írta ki, mert mind a két **READ** utasításra reagált, csak az első értékét vette másodszor is.

Készítsük el még egyszer a kör kerületét számoló programot úgy, hogy a **DATA**-ban előre megadjuk a sugár értékét.

PROGRAM!

```
10 FOR I=1T07
20 READ R
30 K = 2* R * 3.1415927
40 DATA 2,4,98,198,45,87,23
50 PRINT „A KOR KERULETE"; K
60 NEXT I
RUN
A KOR KERULETE 12.5664
```

A KOR KERULETE 25.1327
A KOR KERULETE 615.752
A KOR KERULETE 1244.07
A KOR KERULETE 282.743
A KOR KERULETE 546.637
A KOR KERULETE 144.513

Most írjuk be a RESTORE utasítást a 25. programlépésre.

25 RESTORE

Ha futtatod a programot, újra láthatod, hogy a RESTORE utasítás hatására a gép 7-szer írja ki egymás után a legelső értéket, a 12.5664-et.

GOTO – Feltétel nélküli vezérlőutasítás

GOTO = menj -hoz, -hez, -höz, vagyis valamelyik programlépésre.

Azért nevezzük feltétel nélküli utasításnak, mert minden feltétel nélkül elküldhetünk egy utasítást valamelyik programsorra. Nézzük meg egy függvény kiszámításának programját:

```
10 INPUT X                    – Bekérem az X értéket.
20 Y = 2* X + 5               – A gép kiszámolja Y értékét.
30 PRINT "x=";X,"Y=";Y        – Kiíratom az X és Y változókat.
40 GOTO 10                    – Visszaküldöm a programot
                                a program elejére, a 10-es sor-
                                számra.
```

```
RUN
? 5
X = 5      Y = 15
? 7
X = 7      Y = 19
?
```

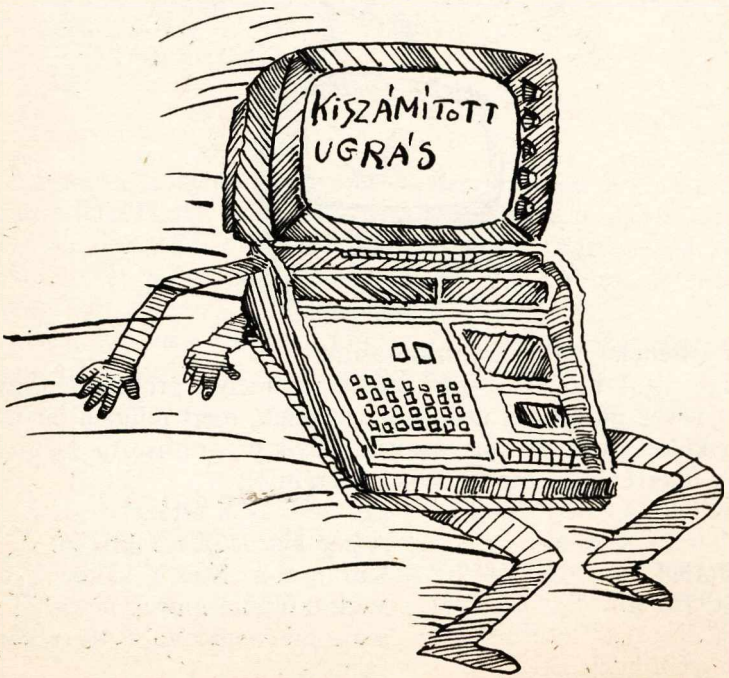
A gép a végtelenségig kérdezné az adatot, ha nem szakítanám meg a BREAK billentyűvel. A képernyőn ez jelenik meg:

```
BREAK IN L0
READY
```

Ebben a pillanatban visszaállt a gép parancs üzemmódba. Minden esetben, ha a gép végtelen ciklusba kerül, így lehet újra folytatni a programozást.

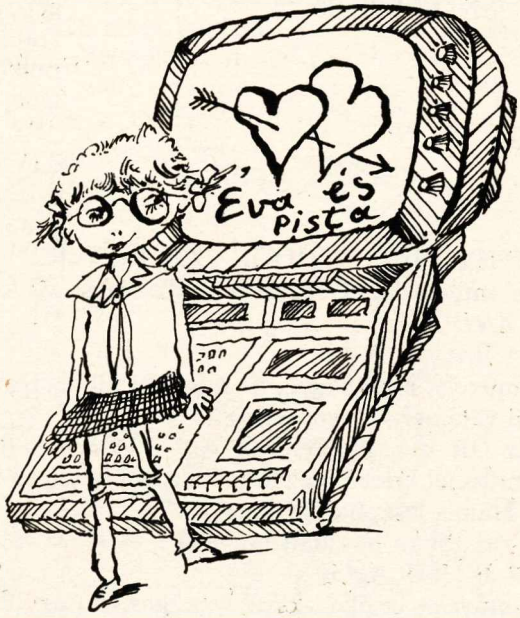
A **GOTO** utasításnak van egy speciális formája, az **ON n GOTO** utasítás. Ez ún. kiszámított ugrás, vagyis előre megadhatom, hogy a program milyen sorrendben hajtsa végre az utasításokat.

PROGRAM!



```
10 INPUT „IRJ BE EGY SZAMOT 1 ES 5 KOZOTT"; A
15 ON A GOTO 20,30,40,50,60
20 PRINT"EGY":END
30 PRINT"KETTO":END
40 PRINT"HAROM":END
50 PRINT"NEGY":END
60 PRINT"OT":END
```

A program futásakor beírok 1 és 5 között egy számot, és a program válaszol arra, betűvel, hogy milyen számot adtam be. (Ez a program megjelent az ÖTLET 1983. november 17-i számában.)



IF – Feltételes vezérlőutasítás

IF = ha. AZ IF utasításhoz szorosan kapcsolódó utasítás a **THEN** = akkor, **ELSE** = különben.

Azért nevezzük feltételes utasításnak, mert az utasítás végrehajtása feltételtől függ. Mik lehetnek ezek a feltételek? Valamely reláció vagy logikai kifejezés.

A gép a következő relációjeleket ismeri:

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

Írjunk egy példát IF utasítás alkalmazásával:

```
10 INPUT „IRJ BE EGY SZAMOT";A
```
– Bekérek egy számot.

```
20 IF A < 0 THEN PRINT „EZ NEGATIV
   SZAM":END ELSE 30
```
– Megvizsgálja a gép, hogy a beírt szám kisebb-e nullánál. Ha igen, kiírja, hogy „EZ NEGATIV", ha nem, a következő programlépésre ugrik.

```
30 PRINT"HT-1080Z SCHOOL-COMPUTER"
```

A gép logikai műveletek elvégzésére is alkalmas. Mi lehet ez?
Az egyik az *AND = és*
Például: A = B AND A<B

A logikai művelet értéke igaz, ha mindkét állítás igaz. Hamis, ha a kettő közül valamelyik nem teljesül.

A másik az *OR = vagy*. Például: A = B OR A<B

A logikai művelet értéke igaz lesz, ha a két feltétel közül valamelyik teljesül. Hamis lesz, ha egyik sem.

A harmadik ritkábban használt művelet a *NOT = nem*
Például: A = B NOT A<B

A logikai művelet értéke akkor lesz igaz, ha egyik feltétel sem teljesül, és akkor lesz hamis, ha mindkét feltétel teljesül.

PROGRAM!

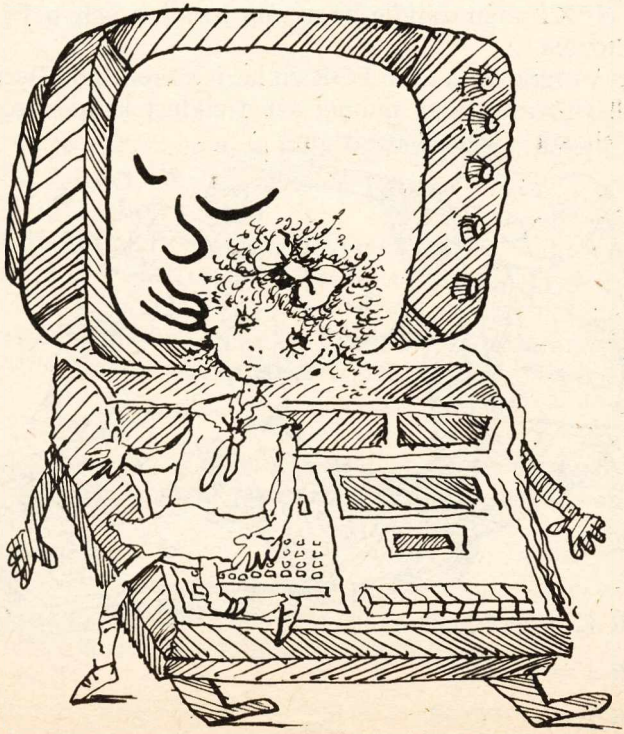
3 szám sorbarendezése.

```
10 INPUT"MI A HAROM SZAM"; A,B,C,
20 IF A<B THEN D=B: B=A: A=D
30 IF B<C THEN D=C: C=B: B=D
40 IF A<B THEN D=B: B=A: A=D
50 PRINT A;B; C
60 GOTO 10
```

FOR, TO, NEXT és STEP ciklusutasítások
FOR = -tól
TO = -ig
NEXT = ciklus lezárása
STEP = lépj

Ciklusnak nevezzük az egy vagy több művelet ismételt végrehajtását a programban. A ciklus lefutása a FOR és a NEXT utasítás hatására történik. Ezzel az utasítással meghatározhatom, hogy a program hányszor hajtsa végre a ciklust.

A 29. oldalon írtunk egy programot függvény kiszámítására. Ott mindig valamilyen értéket kellett beadnom a gépbe és úgy adta ki az Y értékét. Most ezt a programot úgy módosítjuk, hogy a gép magától vegyen fel értéket, például 1-től 10-ig, és írja ki a képernyőre az X, valamint az Y értékét.



PROGRAM!

```
10 FOR X = 1 TO 10
20 Y = 2* X + 5
30 PRINT "Y";Y, "X";X
40 NEXT X
```

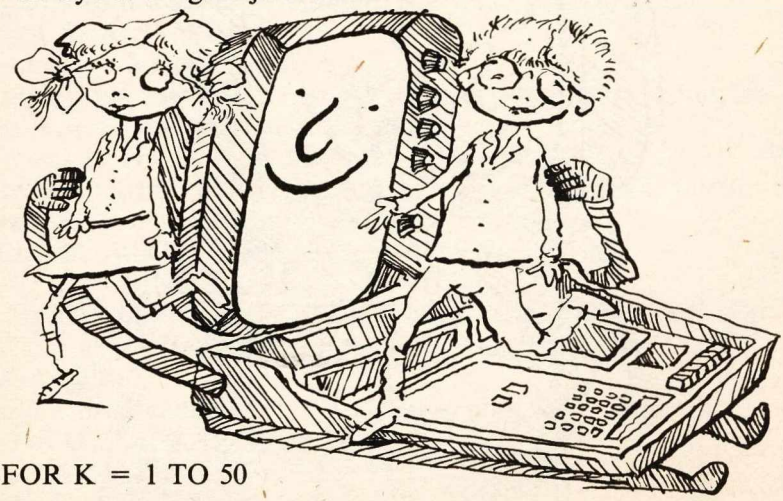
RUN hatására a gép kiírja 1-től 10-ig az Y és X értékét.

Most írjuk ki úgy a függvényt, hogy nem 1-től 10-ig, hanem 10-től 1-ig vegye fel az értéket. Ebben az esetben úgy módosul a programunk, hogy STEP − 1-et írok a 10-es programlépésre. Ilyen esetben a STEP után mindig valamilyen negatív számnak kell állnia, hiszen 10-től 1-ig csak úgy jut el, ha mínusz értékkel lépkedünk.
Javítsd ki a 10-es programsort; így:

```
10 FOR X = 10 TO 1 STEP − 1
```

és futtasd a programot.

- A FOR utasítás után mindig valamilyen változónak kell állnia.
- A NEXT után mindig az a változó áll, amely a FOR ciklus változója.
- Egy programban több FOR ciklus is szerepelhet. Ilyenkor arra kell vigyázni, hogy mindig azt a ciklust kell először lezárni, amelyiket a legutoljára írtam.



```
FOR K = 1 TO 50
FOR J = 1 TO 3
FOR B = 1 TO 78
```

```
NEXT B
NEXT J
NEXT K
```

CIKLUSOK EGYMÁSBA ÁGYAZÁSA.

PROGRAM!

```
 10 PRINT " FAKTORIALIS SZAMITAS"
 20 PRINT
 30 N=0
 40 F=1
 50 PRINT" KEREM A SZAMOT!"
 60 PRINT
 70 INPUT N
 80 IF N>33 THEN GOTO 160
 90 FOR K= N TO 1 STEP -1
100 F = F * K
110 NEXT K
120 PRINT
130 PRINT N; "-NAK A FAKTORIALISA=="; F
140 FOR I= 1 TO 1000: NEXT I
150 GOTO 10
160 PRINT"EZT MAR NEM TUDOM!"
170 FOR I= 1TO 1000:NEXT I
180 GOTO 10
```

A üres FOR ciklus – FOR I=1TO1000 – a programban szünetet jelent. Más gépeken többnyire PAUSE utasítás.

PRINT@ és TAB

Ezeket az utasításokat akkor használjuk, ha például egy sort nem a képernyő bal felső sarkában akarok megjelentetni, hanem a képernyő közepén.

Képernyőbeosztás:

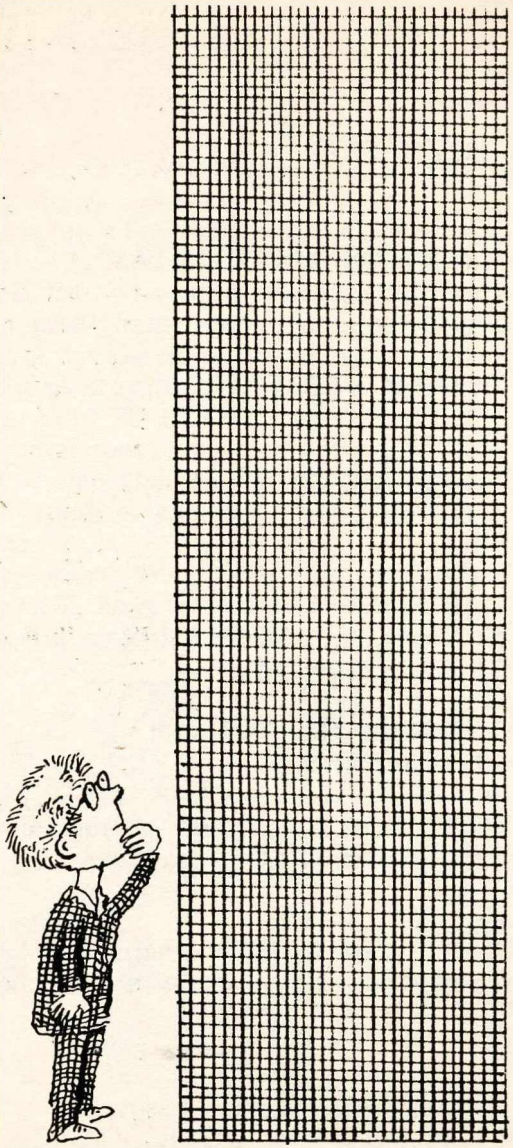
PRINT@

Meghatározza, hogy melyik *sorba* írjam a szöveget.

Beírási formája: 10 PRINT@64*5 "UGYES VAGY"

PRINT TAB

Meghatározza, hogy melyik *oszlopban* kezdjem a szöveget.



Beírási formája: 10 PRINT TAB (12) " SZEP "
A képernyőn tehát összesen: 64 karakter, és 15 sor fér el.

SET – RESET

SET utasítás hatására a gép kirak egy pontot a 127 × 47-es koordinátán belül megadott pont valamelyikére.

Ha tizedesszámot írsz be, azt a gép automatikusan egész számnak tekinti.

Tegyünk ki a képernyő közepére egy pontot:

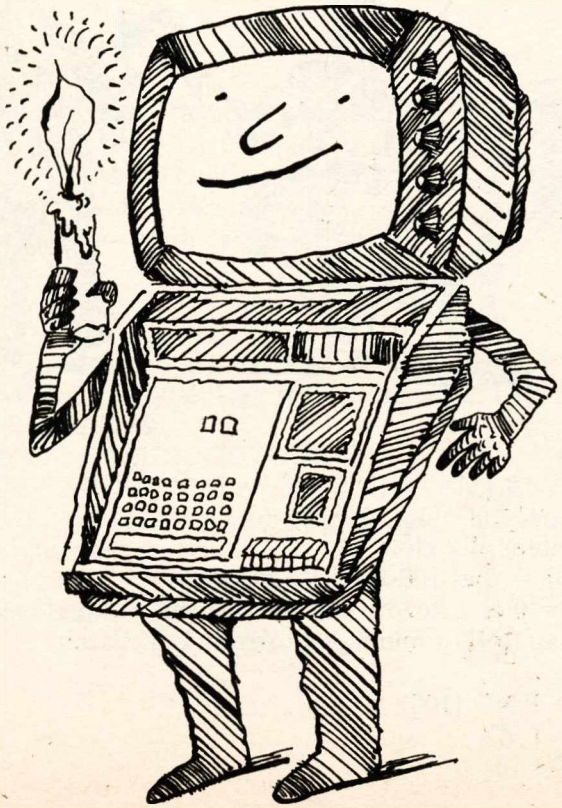
10 SET (34,20) – mindig két koordinátát kell megadni!

RESET utasítás eloltja a pontot.

Írjuk be ezt is:

20 RESET (34,20) – eloltja azt a pontot amelyiket kigyújtotta.

30 GOTO 10 – visszaszalad a program elejére, s állandóan ismétli ezt a folyamatot. Így a képernyőn egy villogó pont jelenik meg.

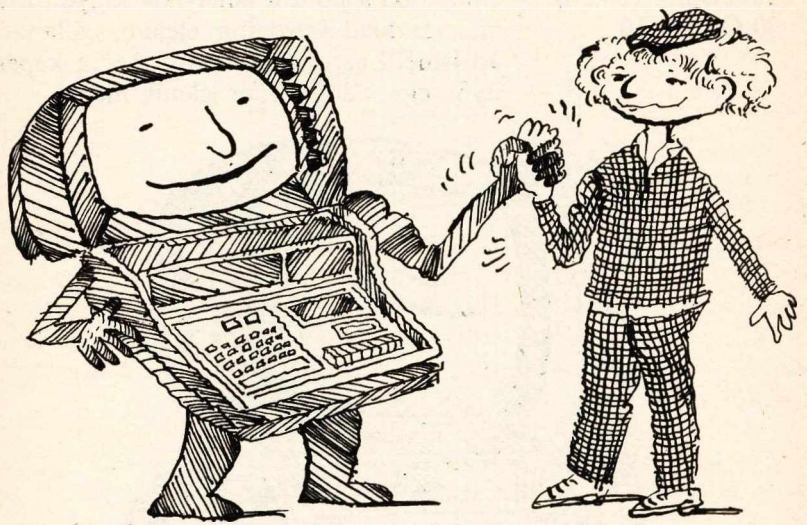


PROGRAM!

Koordináta kirajzolása

```
10 CLS                    – Képernyő törlése
20 FOR I=0TO60:SET (I,23) :NEXT I
30 FOR K=47TO3 STEP-1:SET (23,K): NEXT K
40 GOTO 40
```

A 40 GOTO 40-es programlépés hatására a képernyőn nem jelenik meg a READY felirat, így nem törlődik le a rajzból egyetlen sor sem.



RND

Véletlenszámot állít elő.

RND (100) = 0 és 100 között generálszámokat

RND (0) = 0 és 1 között állít elő véletlenszámokat

Beíráskor az RND-t mindig változóval kell ellátni:

Például:

```
10 ZX= RND (100)
20 PRINT ZX
30 GOTO 10
```

```
10 HT= RND (0)
20 PRINT HT
30 GOTO 10
```

PROGRAM! A program dob két kockával.

```
10 CLS
20 X = RND (6)
30 Y = RND (6)
40 PRINT"A"; Y; "-t ES A ";Y ; "-t dobtam "
```

PROGRAM! A program véletlenszerűen betölt egy meghatározott területet a képernyőn, azaz kifehéríti.

```
10 CLS
20 A = RND (21)
30 B = RND (20)
40 SET A,B
50 GOTO 20
```

INKEY $

Ez az utasítás figyeli a billentyűzetet, vagyis a billentyűzetről beolvas egy karaktert. Ezt a funkciót akkor használjuk, ha azt akarjuk, hogy a program csak egy meghatározott billentyű lenyomása esetén menjen tovább. Nézzünk egy ilyen példát: a program addig vár, amíg nem nyomom meg az „M" betűt, és akkor kiírja, hogy „fekete macska".

```
10 IF INKEY $ = " M " THEN 20 ELSE 10
20 PRINT "FEKETE MACSKA"
```

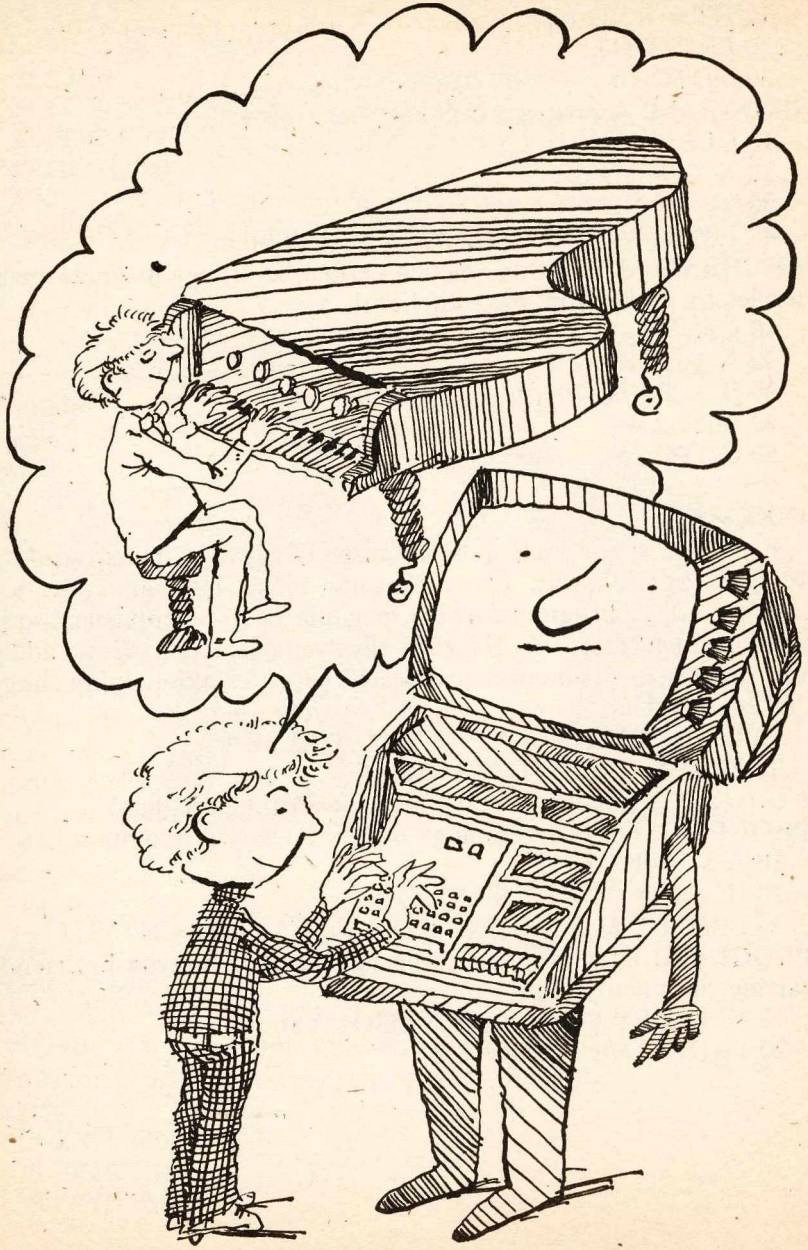
Az INKEY $ mindig IF utasítás kíséretében használható.

PROGRAM! Kiírja a gép, hogy melyik billentyűt nyomtam meg.

```
10 A $ = INKEY $
20 IF A$ = " " THEN GOTO 10
30 PRINT " MEGNYOMTAD A "; A $ "BILLENTYUT"
```

PROGRAM! Ez a program pedig figyelmeztet, hogyha hozzáérek valamelyik billentyűhöz.

```
10 IF INKEY $ = " " THEN GOTO 10
20 PRINT" NE PISZKALJ!"
```



GOSUB RETURN ON n GOSUB – Szubrutin (alprogram) utasítások.

A szubrutin egy főprogramhoz tartozó kis program, amelyre a főprogram bármely pontjáról átadható a vezérlés. Nagyon sokszor készítünk olyan programot, amelyben olyan részfeladat van, ami a programban többször előfordul. Ilyen esetben célszerű – a folyton ismétlődő műveleteknél – egy alprogramra, szubrutinra hivatkozni, ahol végrehajtódik ez a feladat.

A szubrutin programokra a **GOSUB** utasítással kell hivatkozni, azaz elküldöm arra a programlépésre, ahol a szubrutin kezdődik. A szubrutin kis program befejezését a **RETURN** utasítással kell befejezni. **RETURN** utasítás hatására tehát befejeződik a szubrutin végrehajtása, és a program áttér, vagyis visszatér a főprogramra.

Írjunk egy programot szubrutin alkalmazásával 3 szám legnagyobb közös osztójának kiszámítására.

```
10 PRINT"A","B","C",          – Kiíratjuk a fejlécet.
"LNKO"
20 READ A,B,C                 – A READ utasítás beolvassa a
                                DATA-ban lévő adatokat,
                                melyek 3 oszlopban fognak
                                megjelenni.
30 X=A                        – Betesszük az X változóba a
                                DATA-ból beolvasott adatok
                                közül.
40 Y=B                        – Az Y változóba betesszük a
                                READ által beolvasott más-
                                odik értéket.
50 Z=C                        – A Z változóba betesszük a
                                READ által beolvasott har-
                                madik értéket.
60 D=G                        – A D változóba tesszük a szub-
                                rutinban kiszámolt értéket,
                                amit G-vel jelölünk.
70 GOSUB 200                  – Most elküldöm a programot a
                                200-as programlépésre, ahol
                                egy szubrutin program hajtó-
                                dik végre.
80 PRINT A,B,C,G              – Kiíratom a beolvasott érték-
                                eket, és a G-ben levő megol-
                                dást, amely a 4. oszlop lesz.
```

```
90 GOTO 20
```

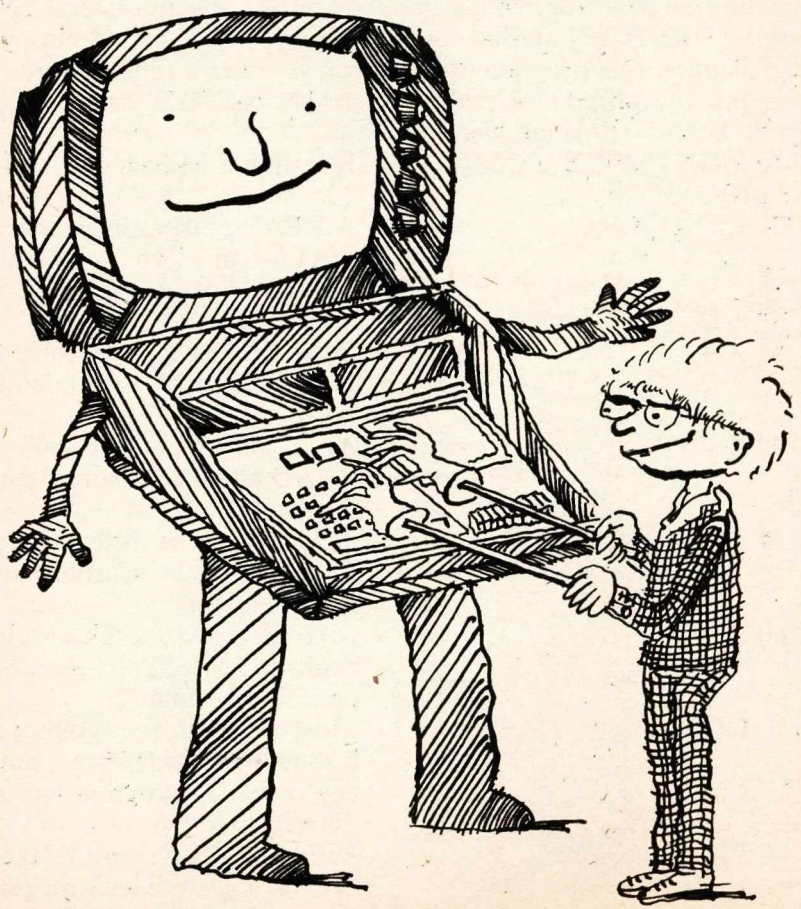
– Visszaküldöm a 20-as programlépésre, így mindaddig ismétli a programrész végrehajtását, amíg beolvasható adatot talál a DATA-ban.

```
100 DATA 60,90,30
```

– Az első oszlop adatai.

```
120 DATA 1298,234,567
```

– A második oszlop adatai.



130 DATA 456,987,35	– A harmadik oszlop adatai.
200 Q=INT(X/Y)	– Itt kezdődik a szubrutin. X és Y tartalmát elosztjuk, ennek az egészrészét (INT) vesszük, és a Q változóhoz rendeljük.
210 R=X–Q*Y	– R változóba kiszámolja, hogy mennyi a maradék.
220 IFR=0 THEN 300	– Ha a maradék nulla, a 300-as programlépésre ugrik, és az LNKO-t beteszi a G-változóba.
230 X=Y	– Ha a feltétel nem teljesül, az X változó tartalmazza a nagyobb számot, és
240 Y = R	az Y változó fogja tartalmazni R értékét.
250 GOTO 200	– Mindaddig végzi a ciklust, amíg a maradék 0.
300 G=Y	– És az LNKO-t beteszi a G változóba.
310 RETURN	– Visszaugrik a főprogram a GOSUB utáni programlépésre, a 90-esre.

ON n GOSUB

Ez az utasítás hasonló az ON n GOTO utasításhoz, azzal a különbséggel, hogy a programvezérlést nem a főprogramban kell végrehajtatni, hanem a szubrutinban.

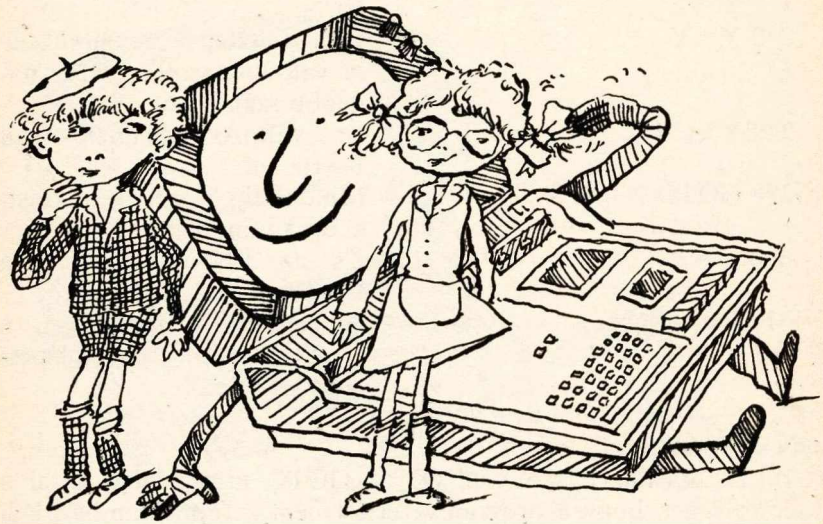
Írjunk feladatot ON n GOSUB használatára:

```
 10 ?"FELADAT ON GOSUB HASZNALATARA"
 20 ?"1-es = EGER"
 30 ?"2-es = KUTYA"
 40 ?"3-as = ELEFANT"
 50 INPUT "Ird BE 1, 2, VAGY 3"; N
 60 ON N GOSUB 160, 100, 150
 70 END
100 ?"SZUBRUTINBAN= EGER":RETURN
150 ?"SZUBRUTINBAN= KUTYA":RETURN
160 ?"SZUBRUTINBAN= ELEFANT":RETURN
```

REM

Ez csupán megjegyzés egy programban, amely a programozó és a felhasználó számára fontos, a programban futtatáskor sohasem kerül végrehajtásra.

```
10 REM MOST KIIROM, HOGY KEDVES VAGY
20 PRINT"KEDVES VAGY"
RUN
KEDVES VAGY
READY
```



SYSTEM

Ez a parancs a számítógépet monitor üzemmódba állítja át. Így lehet írni kisbetűt, vagy pedig átsorszámozni a program sorszámát. Egyelőre csak ezt a két formát mutatom meg, de jegyezd meg, hogy ebben az üzemmódban lehet gépi kódú programokat írni.

Hogyan kell írni *kisbetűket?*

Gépeld be a következőt:

SYSTEM	– Ezután egy kérdőjel, és egy csillag jelenik meg a képernyőn.
* ? /12288	– Írj be egy "/" jelet, és azt az ötjegyű számot, amelyet kisbetű írásakor kell alkalmazni. 12288.
NEW LINE	– És ez jelenik meg a képernyőn:

NEW KEYBOARD ROUTINE ENABLE READY és egy villogó cursor. Ha zavar a villogás, nyomd le a **SHIFT** és a **BREAK** billentyűt egyszerre!

Ezután folytathatod a programozást.

Ha a kisbetűkre vagy kíváncsi, nyomd le a **SHIFT** billentyűt, és a kívánt karaktergombot: Így az *A helyett a betű* jelenik meg a képernyőn.

Hogyan kell átsorszámozni a programot?

A RE utasítás csak ebben a SYSTEM üzemmódban használható. Írjunk egy példát, és használjuk a kisbetűket is:

```
1 PRINT"Tanuld meg az utasitasokat"
2 PRINT"Ne szomorkodj,"
3 PRINT"Ha nem erted"
4 PRINT"azonnal"
```

Sorszámozzuk át.

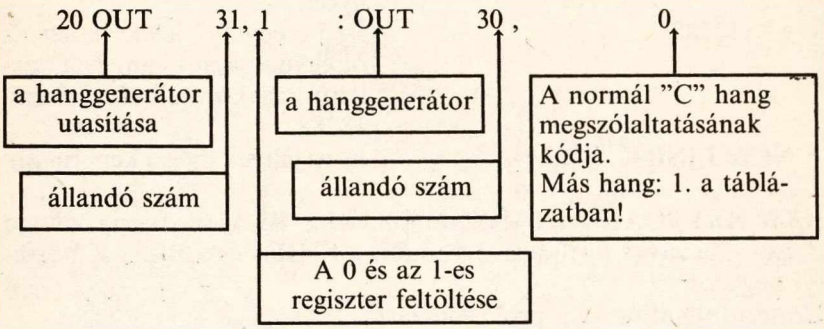
```
RE
LIST
10 PRINT"Tanuld meg az utasitasokat"
20 PRINT"Ne szomorkodj,"
30 PRINT"Ha nem erted"
40 PRINT"azonnal"
```

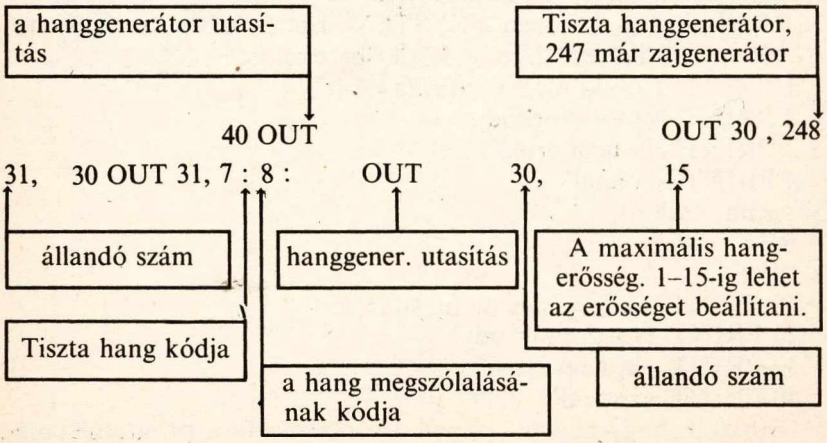
Láthatod, hogy nagyon szépen átsorszámozta a programlépéseket. A RE utasítás mindig 10-zel sorszámoz.

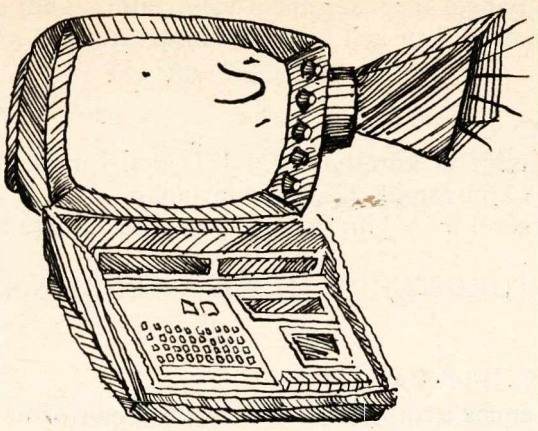
OUT hanggenerátor

A HT-gépen 3 csatornán, A,B,C, szólaltathatunk meg hangot. Az információ a gép memóriájába a regiszterein keresztül jut. Összesen 14 regisztere van, 0-tól 13-ig.
Mégis hogyan szólal meg egy hang a gépen?
Részletes leírást a regiszterekről most nem kapsz, de egy rövid ismertetővel megtudhatod, hogyan kell hangot kicsalni a gépből.

```
10 OUT 31, 0 : OUT 30 , 200
```







Ennek a programnak a futásakor megszólal ugyan a C-hang, de csak akkor lehet leállítani, ha a gép hátoldalán lévő **RESET** gombot benyomom. Most úgy módosítjuk a programot, hogy egy üres FOR-ciklust iktatunk be, majd legvégül nullázzuk a 2 regisztert.

Így:

50 FOR I= 1 TO 500 : NEXT I – 500 mp szünet.
60 OUT 31, 0 : OUT 30, 0 – 0 a 0-ás regiszter.
70 OUT 31, 1 : OUT 30, 0 – 0 az 1-es regiszter.

A hang most csak rövid ideig hallható. Hosszabbítani tehát a FOR-ciklussal lehet.

	Zenei hang kódok			
	I. oktáv	II. oktáv	III. oktáv	IV. oktáv
C	32 3	144 1	200 0	100 0
Cisz	247 2	123 1	190 0	95 0
D	200 2	100 1	178 0	89 0
Disz	155 2	77 1	167 0	83 0
E	128 2	61 1	160 0	80 0
Eisz				
F	88 2	44 1	150 0	75 0
Fisz	50 2	25 1	141 0	70 0
G	21 2	11 1	133 0	66 0
Gisz	244 1	250 0	125 0	63 0
A	224 1	240 0	120 0	60 0
Aisz	194 1	225 0	112 0	56 0
H	171 1	213 0	107 0	53 0
Hisz				

Tehát, ha a hangot meg szeretném változtatni, akkor a 10-es és a 20-as programlépésben kell kijavítanom az utolsó 2 számot, ha szükséges.

CHR $ és ASC

CHR $ segítségével kiírathatom a HT által ismert karakterek kódjait. Ezek a kódszámok 32–191-ig tartanak.

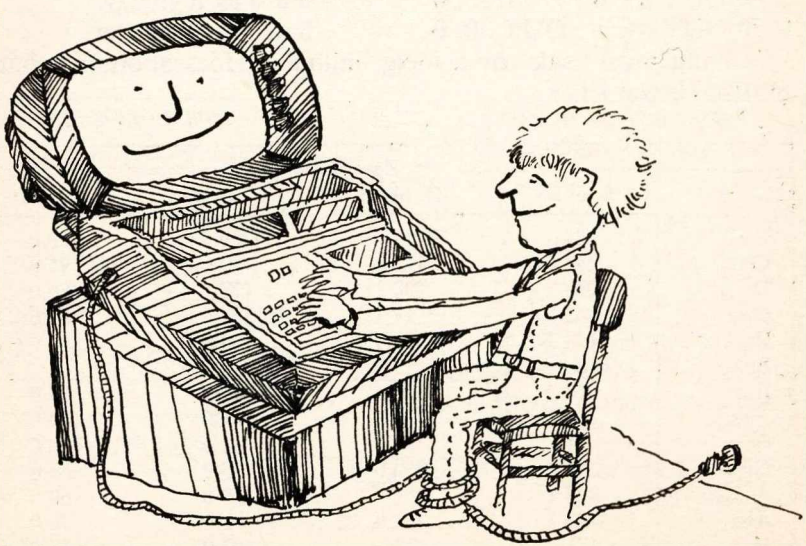
CHR $ segítségével most kiíratok a képernyőre egy idézőjelbe tett szöveget.

```
10 PRINT "OKOS"CHR$ 34 "KIS GEP"CHR$ 34"VAGY HT."
RUN
OKOS "KIS GEP" VAGY HT.
```

Az **ASC** pedig ennek a fordítottja, azaz egy meghatározott karaktert lehet kiíratni a képernyőre.

Például:

```
10 PRINT ASC " + "
RUN
43
```



A következő *PROGRAM!* Kiírja a gép összes karakterét.

```
10 FOR I = 32 TO 191
20 FOR K = 1 TO 500 : NEXT K
30 PRINT CHR $ I
40 PRINT
50 NEXT I
60 PRINT I
```

STRING $

Egy adott karakterből n hosszúságú karaktert állít elő.

```
10 PRINT STRING $ 20, "+"
RUN
+ + + + + + + + + + + + + + + + + + + +
```

Aritmetikai függvények

ABS X = Kiszámítja az X abszolút értékét.

ATN X = Kiszámítja az X radiánban vett arcus tangensét. A fokhoz meg kell szorozni 57,29578-cal.

COS X = Kiszámolja a radiánban megadott X koszinuszát. A fokhoz meg kell szorozni az X-et 0.0174533-mal.

INT X = Pozitív számnál csak az egész részt veszi figyelembe, negatív számnál kerekít egészre.

```
pl.: 100 I=INT −3.9:?I
RUN
−4
```

SIN X = Kiszámolja a radiánban megadott X szinuszát. A fokhoz meg kell szorozni X-et 0.0174533-mal.

SQR X = Kiszámolja X négyzetgyökét.

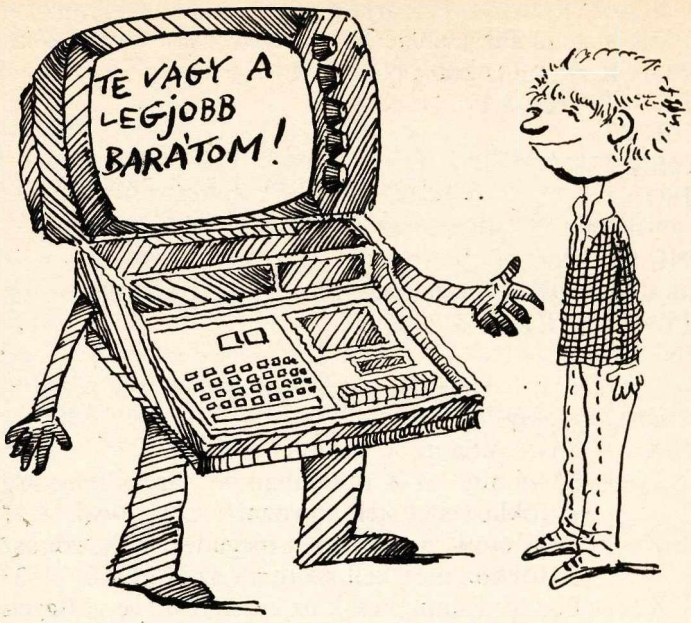
TAN X = Kiszámolja a radiánban megadott X tangensét. A fokhoz meg kell szorozni 0,0174533-mal.

LOG X = Kiszámolja X természetes alapú logaritmusát.

A HT SCHOOL COMPUTER ennél még több utasítást ismer, de ezekkel ebben a könyvben nem foglalkozunk.

És, ha ezeket már mind tudod, túl vagy a „kezdő" lépcsőn.

További jó tanulást kívánok a haladó szakkörben!



Hibaüzenetek leírása

Hibajelzés	Magyarázat
NF	FOR nélküli NEXT; NEXT utasítást használtunk megfelelő FOR nélkül, ez a hiba akkor is felléphet, ha egymásba ágyazott ciklusokban NEXT változókat felcseréltünk.
SN	Szintaktikai hiba; valamilyen hibás beírás eredménye, nyitva hagyott zárójel, érvénytelen jel, vagy rosszul írt utasítás.
RG	GOSUB nélküli RETURN: olyan RETURN utasítást talált a gép, amelyhez nincs megfelelő GOSUB.
OD	Nincs több adat: a READ vagy az INPUT utasításnak nem áll több adat rendelkezésére, mert elfelejtettünk DATA utasítást adni, vagy pedig mert az adatlistáról a gép már az összes adatot beolvasta.

FC	Nem megengedett függvényutasítás, megpróbáltunk egy műveletet olyan paraméterrel végrehajtani, amelylyel az nem lehetséges.
OV	Túlcsordulás, egy kiszámított érték vagy adat (abszolút értékben) túl nagy, így a gép ezt nem tudja tovább kezelni.
OM	Nincs több tárolóhely, az összes tárolóhely már vagy betöltött vagy előre lefoglalt. Az ok: túl nagy értéke lehet a tömbdimenziónak, vagy egymásba ágyazott ugró utasítások, mint GOTO, GOSUB, FOR-NEXT. Az is lehet, hogy egyszerűen túl hosszú a program.
UL	Definiálatlan sor: megpróbáltunk egy nem létező sorra hivatkozni.
BS	A megadottnál nagyobb dimenzió, megpróbáltunk olyan tömbelemet hívni, amelynek dimenziója nagyobb a DIM utasításban megadottnál.
DD	Dimenzió felülírás, megpróbáltuk egy olyan tömb dimenzióját megadni, amelynek dimenzióját már korábban meghatároztuk (DIM utasítással módosíthatjuk, úgy, hogy a DIM-et a program elejére tesszük).
/0	Nullával való osztás.
ID	Nem megengedett, közvetlen felhasználás, az INPUT utasítást parancs üzemmódban akartuk végrehajtani.
TM	Típuskeveredés, megpróbáltunk nem stringváltozóba stringet írni.
OS	Nincs több hely stringek részére.
LS	A string túl hosszú, egy string csak 255 karakterből állhat.
ST	Túlságosan összetett stringművelet, a műveletet összetevőire kell bontani.
CN	A CONT utasítást nem tudja végrehajtani, CONT-ot adtunk például END vagy EDIT után.

NR	NO RESUME; a program végét a hibafelismerési üzemben érte el a gép.
RW	ERROR nélküli RESUME; a gép RESUME-ot talált, mielőtt ON ERROR GOTO-t hajtott volna végre.
UE	Nem kiírható hiba; megpróbáltunk egy hibát nem létező ERROR kóddal szimulálni.
MO	Hiányzó operandus, megpróbáltunk egy olyan műveletet végrehajtani, ahol az operandusok egyike hiányzik.
FD	Hiányos file; a külső forrásból (pl. kazettás magnetofonról) történt adatbeolvasás hibás volt, vagy az adatok nem voltak sorrendben stb.

ASCII karakterkódok.

ASCII: (American Standard Code for Information Exhange: az információcsere egy szabványos amerikai kódja). Kommunikációs célokra kidolgozott olyan bináris kód, amely a kis- és nagybetűket, számokat, írásjeleket és speciális vezérlő karaktereket tartalmazza.

32	space	41	)
33	!	42	*
34	"	43	+
35	#	44	,
36	$	45	–
37	%	46	.
38	&	47	/
39	'	48	ø
40	(	49	1

50	2	63	?
51	3	64	@ É
52	4	65	A
53	5	66	B
54	6	67	C
55	7	68	D
56	8	69	E
57	9	70	F
58	:	71	G
59	;	72	H
60	<	73	I
61	=	74	J
62	>	75	K

76	L	89	Y
77	M	90	Z
78	N	91	[↑
79	O	92	\ ö
80	P	93	I á
81	Q	94	⌒ ü
82	R	95	_
83	S	96	\ é
84	T	97	a
85	U	98	b
86	V	99	c
87	W	100	d
88	X	101	e

102	f	115	s
103	g	116	t
104	h	117	u
105	i	118	v
106	j	119	w
107	k	120	x
108	l	121	y
109	m	122	z
110	n	123	{
111	o	124	| ö
112	p	125	} á
113	q	126	~ ü
114	r	127	#

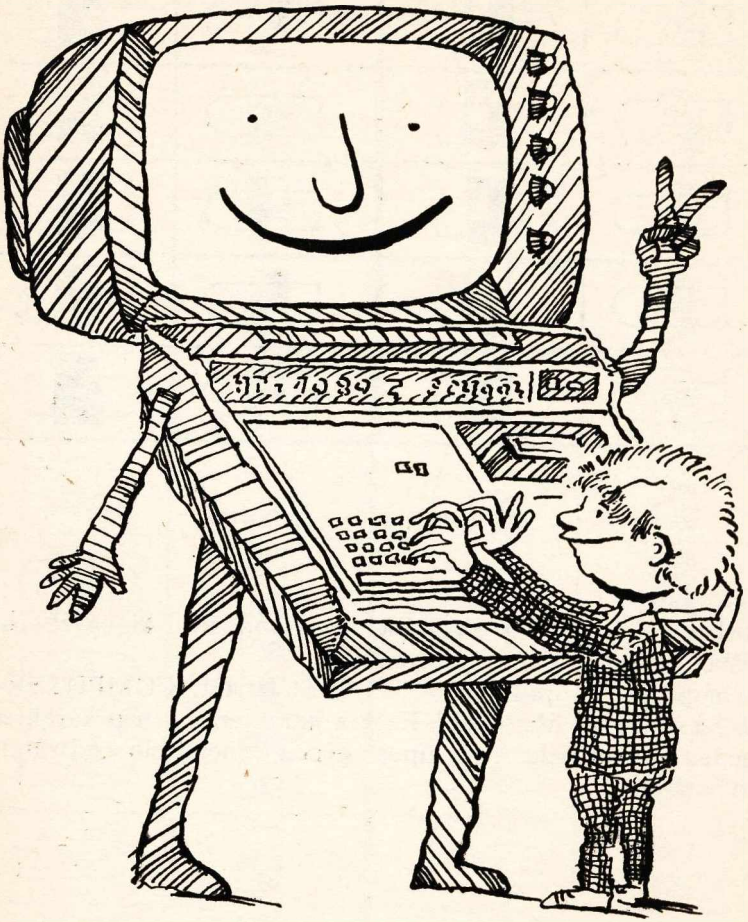
128		141	
129		142	
130		143	
131		144	
132		145	
133		146	
134		147	
135		148	
136		149	
137		150	
138		151	
139		152	
140		153	

154		167	
155		168	
156		169	
157		170	
158		171	
159		172	
160		173	
161		174	
162		175	
163		176	
164		177	
165		178	
166		179	

180		186	
181		187	
182		188	
183		189	
184		190	
185		191	

Megjegyzés:

- Kisbetűket a gép csak SYSTEM üzemmódban ír ki, egyébként a kisbetűk kódszámára is nagybetűt ír.
- Néhány újabb típusú HT–1080 Z SCHOOL COMPUTER-en „ü,é,á,ö" betűk írhatók ki. Ezek a karakterek a régi karakterek mellett szerepelnek. Az új típusú gépen a megfelelő kódszámmal hívható elő.



JEGYZET

JEGYZET

JEGYZET

Tartalomjegyzék

Felelős szerkesztő: BŐCS FERENC
Címlapgrafika: DALLOS JENŐ
Műszaki vezető: BAKOS JÁNOS
Műszaki szerkesztő: DÁVID ÉVA
Megjelent az Ifjúsági Lap- és Könyvkiadó Vállalat gondozásában
1985-ben 5 A/5 ív terjedelemben
Felelős kiadó: DR. PETRUS GYÖRGY igazgató
Szedte a Nyomdaipari Fényszedő Üzem
85-1126 — Dabasi Nyomda, Budapest — Dabas
Felelős vezető: BÁLINT CSABA igazgató

Ára: 23,—Ft

IFJÚSÁGI LAP- ÉS KÖNYVKIADÓ VÁLLALAT