TRITON KOMPI

IPT-002 személyi számítógép

BASIC

gépkönyv

triton

IPT-002 SZEMÉLYI SZÁMÍTÓGÉP
BASIC GÉPKÖNYV

TRITON KOMPI

IPT-002 személyi számítógép

BASIC

gépkönyv

triton

SZÁMALK REPRO 84/327

TARTALOM

1. Bevezetés 7
2. Műszaki adatok, szolgáltatások 8
3. A személyi számítógép összeállítása, működési elve, egyéb tudnivalók 10
3.1 A kazettás magnetofon csatlakoztatása a számítógéphez 10
3.2 Nyomtató csatlakoztatása az IPT–002-höz 11
3.3 Televízió csatlakoztatása az IPT–002-höz 11
3.4 Általános tudnivalók 12
3.5 A számítógép üzemmódjai 13
3.6 A változók típusai és lehetséges azonosítóik 15
3.7 Változók megadása 16
4. Az IPT–002 típusú személyi számítógép BASIC nyelvi dialektusának ismertetése 18
4.1 Műveletek 18
4.2 Parancsok 20
4.3 Beviteli és kiviteli utasítások 23
4.4 Adatmegadás 29
4.5 Adattípusok megadása 32
4.6 Aritmetikai függvények 33
4.7 Karakterlánc függvények 36
4.8 Logikai függvények 39
4.9 Programutasítások 40
4.10 Vegyes utasítások 48
5. Szerkesztő üzemmód 50
5.1 A kurzor kívánt pozícióra állítása 50
5.2 Karakterek cserélése 50
5.3 Karakterek törlése 51
5.4 Kilépés a szerkesztő üzemmódból 51
5.5 Karakterek beszúrása (inzertálás) 52
5.6 Beszúrás a programsor végén 52
5.7 A kurzortól jobbra levő összes karakter törlése és beszúrás 53
5.8 Összefoglaló táblázat 53
6. Néhány tanács-gondolat! 54
7. BASIC programkönyvtár 56
8. Általános üzemeltetési és biztonságtechnikai előírások, óvó rendszabályok 89
9. Karbantartási és szervízelőírások 90
10. A műszaki állapot ellenőrzése 91
11. Az M 4001 típusú tápegység csatlakoztatása 92
12. Mellékletek 93

1. BEVEZETÉS

Gratulálunk Önnek, jól választott!

Köszönjük, hogy megvásárolta IPT–002 típusú személyi számítógépünket! Reméljük, hogy termékünk Önnek hasznos segítőtársa, képzett oktatója és egyben vidám szórakoztató partnere is lesz.
Bízunk abban, hogy a gép képességei kiérdemlik az Ön maximális tetszését.

2. MŰSZAKI ADATOK, SZOLGÁLTATÁSOK

Típus:	IPT–002
Programrendszer:	Tárolt programrendszer
Programnyelv:	BASIC
Memória:	Rendszerprogram: 12 Kb (alapgép) RAM tár: 16 Kb (alapgép)
Változók:	
Numerikus:	Egész típusú: (–32768,+32767) Egyszeres pontosságú: (–1.701411E+38, +1.701411E+38) Dupla pontosságú: (–1.701411834544556E+38. +1.701411834544556E+38)
Szöveges:	Karakterlánc: max. 255 karakter
Programsorok lehetséges sorszámai:	0–65529
Programsor hossza:	max. 255 karakter
Kiértékelési módszer:	Balról-jobbra, a matematikai prioritás szabályainak megfelelően.
Beépített matematikai műveletek:	Négy alapművelet, hatványozás, trigonometriai és inverz műveletek, logaritmikus és exponenciális műveletek, négyzetgyökvonás, előjel, abszolutérték, egészrész függvények, véletlenszám-generátor, logikai műveletek.
Külső memória:	Mágnesszalag. Beépített 1 200 / 600 HZ FSK modulációs rendszerű, 600 Baud átviteli sebességű magnóillesztővel rendelkezik. Egyetlen tuchel-csatlakozó szolgál a bemeneti, kimeneti és a magnetofont indító start/stop jelek átvitelére.
Kijelzés:	Közönséges otthoni televíziós készülék, az antennabemenetén keresztül vezérelve. Alfanumerikus és kvázigrafikus karakterek, 32 kar /24 sor. A kiadott jel D szabvány szerinti fekete/fehér TV-jel az OIRT III-as sávra hangolva. Csatlakozó: 75 Ω asszimetrikus.

Nyomtató:	Beépített V. 24-es 200 Baud átviteli sebességű aszinkron interfész.
Működési hőmérséklettartomány:	+15°C – +35°C
Hálózati feszültség:	220 V ± 10 %
Teljesítmény felvétel:	60 VA max
Méretek:	380x220x75 mm
Súly:	

Tartozékok, összeállítás:

Az IPT–002 típusú kiszámítógépet a következő összeállításban hozzuk forgalomba:

1. IPT–002 típusú kiszámítógép	(alapár):
Tartozékai	
– TV antennakábel	1 db
– magnókábel	1 db
– a soros nyomtató interfészkábele	1 db (V. 24 jellegű)
2. M 4001 típusú tápegység	(külön ár)
Tartozékai	
– tápegységkábel	1 db
– biztosító betétek	7 db
3. Felhasználói programkazetta 50 kész programmal	(külön ár)

A fenti tételek, (1,2,3) külön-külön is megrendelhetők.

Opcionális egységek (megrendelésre),

1. 32 Kbyte RAM kártya tárbővítésre (összesen 48 Kbyte RAM, 16 Kbyte program- és display- memória);
2. 16 Kbyte RAM kártya tárbővítésre;
3. Párhuzamos interfész (Centronics típusú) nyomtatóhoz (programmal együtt);
4. Lemezmeghajtó egység (floppy-drive) interfész MF 1800/900 vagy más 5 1/4 inch méretű floppy vezérléséhez (programmal együtt).

3. A SZEMÉLYI SZÁMÍTÓGÉP ÖSSZEÁLLÍTÁSA, MŰKÖDÉSI ELVE, EGYÉB TUDNIVALÓK

3.1 A kazettás magnetofon csatlakoztatása a számítógéphez

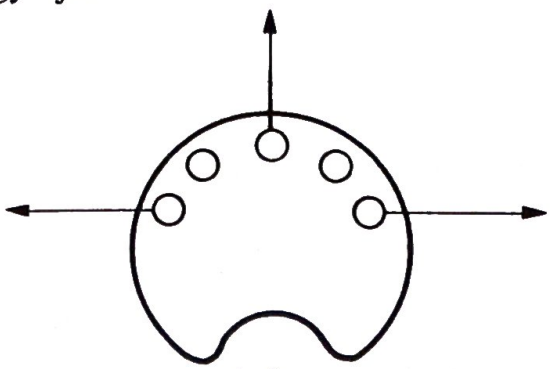
A kazettás magnetofon feladata a számítógép adatainak és programjainak szalagra rögzítése, illetve azok visszatöltése a szalagról a számítógép tárába. Ennek megfelelően csatlakoztatása csak akkor szükséges, ha ezt a funkciót is használni akarjuk.

Az alkalmazandó kazettás magnetofon kiválasztásához a következő szempontok figyelembevételét javasoljuk:

- A hálózati üzem előnyösebb mint a telepes táplálás.
- A számítógép a magnetofon motorjának ki- és bekapcsolására is képes, tehát az automatikus működtetéshez elektromosan vezérelhető start/stop kapcsolós készülék alkalmazása célszerű.
- A digitális jelek rögzítéséhez megfelelőbb ha a felvételi szintet a hangerőszabályozóval állítani lehet, de a szokásos „automata" megoldások sem jelentenek problémát.
- A kazettára felvett programok, adatok gyors és kényelmes megtalálásához jó szolgálatot tehet egy beépített számláló.
- A programkazettákat el kell látni cimkékkel és programlistával, feljegyezve a felvétel azonosítóját és a számláló állását.
- A kazettákat nem szabad kitenni nagyobb hő- és mágneses hatásnak. Sose helyezzék el pl. a gép transzformátora mellé!
- A felvételek készítéséhez használjanak minél jobb minőségű szalagokat!
- Ha van rá mód, akkor használjanak befűzőszalag nélküli kazettát! Ha ez nem megy, ügyelni kell arra, hogy a színes befűző szalagokra nem lehet felvételt készíteni.
- A felvételek biztonságos megőrzéséhez segítséget nyújt az olyan kazetta, melynél az írás engedélyezése/tiltása nem a szokásos „kitördeléssel", hanem elcsúsztatható műanyag retesszel van megoldva.
- A retesz tiltó helyzetében a kazettára írás lehetetlen, ami elsősorban több felhasználó esetében bizonyul praktikusnak, de figyelmeztetése a feledékeny egyedüli felhasználónak is hasznos.
- A gép megvásárlása után tapasztalatszerzés céljából érdemes néhány próbafelvételt készíteni! Kiinduláshoz a

hangerőszabályozó középállása felel meg, az optimális beállítás is többnyire e körül lesz.

– A felvétel készítés és visszajátszás BASIC utasításait a 4. fejezet tárgyalja.



1. ábra.

A készülék csatlakozójának bekötését az 1. ábra szemlélteti. Ilyen bekötés szükséges például a BRG, MK–27 típusú magnetofonhoz.

3.2 Nyomtató csatlakoztatása az IPT–002-höz

Az IPT–002 a V. 24-es nemzetközi szabvány szerinti soros interfésszel rendelkezik. 1 byte felépítése:

1 start bit,
7 információs bit,
2 stop bit.

Az adatviteli sebesség 200 Baudra van beállítva. Ha ettől eltérő sebesség szükséges a „sebességkonstans" megváltoztatásával 110Baud, 100Baud, 75Baud, 50Baud-os átvitelt is be lehet állítani. E célból bekapcsolás után hajtsuk végre az alábbi utasításokat!

POKE 12289, 120:	POKE 12290, 4 (50B)
POKE 12289, 251:	POKE 12290, 2 (75B)
POKE 12289, 60:	POKE 12290, 2 (100B)
POKE 12289, 8:	POKE 12290, 2 (110B)

3.3 Televízió csatlakoztatása az IPT–002-höz

Az IPT–002 típusú személyi számítógép a beépített képernyő helyett az otthoni televíziós készüléket is használhatja. Csatlakoztatni a TV készülék antenna bemenetén lehet. A csatornaváltót az OIRT 6-os csatornájára kell beállítani.

Ha a TV már bemelegedett és megjelent a fény a képernyőn, kapcsoljuk be az előzőleg már csatlakoztatott IPT–002 típusú személyi számítógépet. Ha helyes a beállítás és alapgépet alkalmazunk a TV képernyőjén a következőket lehet látni:

```
IPT–002 BASIC
TOT. MEM. (BYTES) 16384
READY
>←
```

A finomhangolóval valamint a fényerő- és kontrasztszabályozóval igyekezzenek a lehető legjobb és legkellemesebb képet létrehozni. A hangerőt nullára kell állítani!

Ha nem a leírtakat tapasztalnák, ellenőrízzék, hogy mindent megfelelően alkalmaztak-e!? Ha a TV készüléken nem utal a működésre semmi látható jel, akkor a TV feltehetően nem alkalmas az OIRT 6–12 csatornák vételére. Ilyenkor vagy egy másik készülék szükséges, vagy bizonyos átalakítás a televízión. E problémájával forduljon az ORION, illetve VT márkaszervizeihez.

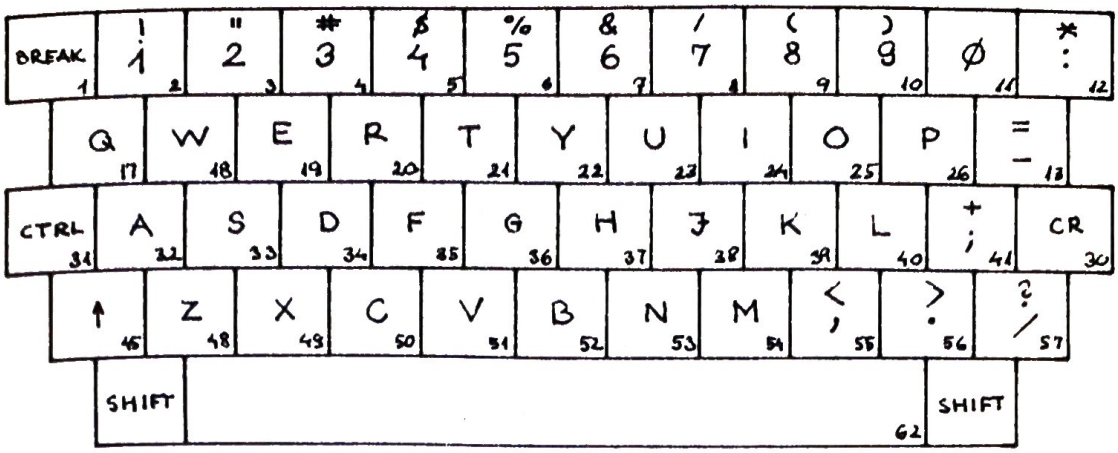
Az IPT–002 típusú személyi számítógép 75 Ω-os asszimetrikus antenna kimenettel rendelkezik; ilyen csatlakozója van pl. az ORION–60, 65, 70, URANUS stb. készülékeknek.

3.4 Általános tudnivalók

Az előzőekben ismertetett csatlakoztatások elvégzése után a számítógép üzemképes. A hálózati kapcsoló bekapcsolása

után a képernyőn megjelenő feliratok tájékoztatnak a gép kiépítettségéről, azaz a memória terjedelméről és az alapgéphez csatlakoztatott egyes bővítésekről. Alapgép esetében az előbb bemutatott ernyőkép látható. A hálózatra kapcsolt állapotot a DC POWER ON feliratú LED világítása jelzi.

A 2. ábra a gép billentyűzetét mutatja be. A CNTL és egy másik nyomógomb **egyidejű** megnyomásával speciális vezérlő funkciókat is lehet aktivizálni. Ezek a következők:



2. ábra.

CNTL –A Break: feladata futó programok megszakítása.

CNTL –I Tabulator: a kurzort (pozíciómutatót) pillanatnyi pozíciójáról a képernyő következő tabulátorpozíciójára helyezi. Egy 32 karakterből álló sorban négy ilyen van a 0, 8, 16, 24-es karakterpozíciókon.

CNTL – H Backstep: visszalépés és egyben az átlépett karakter törlése. Főleg elütéseknél kell használni.

CNTL – X Bevitel végén a közölteket letörli a képernyőről és a kurzort is a bevitel megkezdése előtti pozícióra állítja. A működés a bevitel ismétlésével folytatódhat.

CNTL – B Segítségével a futó program végrehajtása minden üzenet nélkül felfüggeszthető. A képernyő „megmerevedik", lehetőséget ad az esetleges megfigyelésekhez. A felfüggesztett program tetszőleges billentyű megnyomásával folytatható.

CR Carriage return (kocsivissza): az IPT–002 gépen a CR billentyű megnyomásának hatására kezdődik beadott adatok kiértékelése és végrehajtása.

3.5 A számítógép üzemmódjai

A számítógép BASIC nyelven programozható. Ezen a programnyelven a felhasználó által írt programsorok egy-egy sorszámmal azonosíthatók. Az eltérő sorszámú programsorokat a gép a bevitelkor sorbarendezi, majd végrehajtáskor a növekvő sorszámoknak megfelelő sorrendben hajtja végre.

Ha tehát a sor elején megjelenő

>←

karakterek után a felhasználó egy a 0–65529 tartományba eső számmal kezd, majd BASIC utasításokkal fejezi be a karaktersorozatot, pl. így:

```
10 PRINT A+B
```

akkor az mint 10-es azonosítójú programsor tárolódik, közvetlen végrehajtás nélkül. Ezt az üzemmódot **program** üzemmódnak nevezhetjük. Ha azonban a felhasználó pl. a következőket írná,

```
>PRINT 12+3*4-(4-8)  (CR)
```

akkor a gép azonnal közölné a matematikai kifejezés kiszámított értékét:

```
 28
READY
>←
```

A különbség az, hogy a felhasználó a bevitelt ebben az esetben nem egy azonosítóként értelmezhető számmal kezdte. A számítógép mint kalkulátor dolgozott, ezért ezt az üzemmódot **kalkulátor** üzemmódnak nevezhetjük.

Ha **program** üzemmódban tároltuk az elképzelt programot, akkor következhet a program futtatása. A **végrehajtó** üzemmódra a


```
>RUN  (CR)
```

parancs beadásával lehet áttérni.

Persze a végrehajtáskor kiderülhet, hogy az elképzelt program hibás. Erre a BASIC különböző hibaüzenetekekkel reagálhat, melyek listáját és jelentését a 89. oldalon tárgyaljuk. A hibák okai és fajtái sokfélék lehetnek, de minden esetben ki kell azokat javítani. A javítások kényelmes végrehajtása céljából egy ún. **szerkesztő** üzemmód bevezetése is szükségessé vált.

Ezen üzemmód használatát pl. az

```
>EDIT 10 (CR)
```

parancs beadásával lehet kezdeményezni.

Az üzemmódokat később részletesen tárgyaljuk.

3.6 A változók típusai és lehetséges azonosítóik

A számítógép négy különböző típusú változó kezelésére képes:

Numerikus

Egész (integer) típusú számok

– A leggyorsabb feldolgozást biztosítják és a legkisebb tárhelyet igénylik. Cserébe a használható tartomány erős korlátozásával kell fizetni (–32768, +32767).
  Deklarációs karaktere: %

Egyszeres pontosságú
(single) számok

– a (–1.701411E+38, + 1,70411E+38) tartományon belül. Használatuk mind időt mind tárhelyet kevesebbet igényel, mint a dupla pontosságúak. Az elemi függvények mindig ilyen típusú eredményt szolgáltatnak. A hétjegyű pontosság a gyakorlati számítások során elegendőnek bizonyul.
  Deklarációs karaktere:!

Dupla pontosságú
(double) számok

– azaz 17 jegyű számok a (–1.701411834544556E+ 38, +1.701411834544556E+38) tartományon belül. Használatuk a leghosszabb számítási időt és legnagyobb tárolóhelyet igényli, cserébe a legnagyobb pontosságot kínálja. Csak a négy alapműveletet lehet ennél a típusnál végrehajtani, illetve ilyen típusura lehet alakítani a többi típusú számot (lásd később).
  Deklarációs karaktere: #

Szöveges

Karakterlánc (string)

– azaz legalább 1 és maximum 255 karakterből álló jelsorozat.
  A betűk feldolgozásával, összeillesztésével és az ebben rejlő lehetőségek kihasználásával válik a gép a szuper teljesítményű kalkulátorból igazi számítógéppé.
  Deklarációs karaktere: $

Tömb

– azaz tömb jellegű, dimenzionált változók a fent felsorolt típusok bármelyikében. Segítségükkel könnyen kezelhetők a mátrixok, vektorok, táblázatok stb.

A változók azonosítására (elnevezésére) betűvel kezdődő betű-szám sorozatok szolgálhatnak. A memorizálás és könnyed megkülönböztetés céljából a felhasználó tetszőleges hosszúságú azonosítókat is írhat, de a gép csak az első két karakter alapján tesz különbséget közöttük. Ezért az AB, ABB, ABBA séma szerinti változó elnevezéseket kerülni kell, mert a gép azonosnak fogja értelmezni őket! Az azonos nevű de különböző típusú változókat viszont megkülönbözteti, pl. az alábbi változók

AB%, AB!, AB# AB$

más és más értékkel szerepelhetnek.

Még egy megszorítás vonatkozik a változók azonosítására, éspedig az, hogy a BASIC nyelv kulcsszavainak felhasználása azonosítók céljára nem megengedett. Hibás lenne pl. a KIF azonosító, mert tartalmazná az IF kulcsszót. BASIC kulcsszavainak táblázata a 88. oldalon található.

A megszorítások ellenére 800 felett van a lehetséges azonosítóként felhasználható variációk száma.

3.7 Változók megadása

Előzetes típusmeghatározás nélkül a változó egyszeres pontosságú számként ábrázolódik:

```
pl.  A=12.0
     B= 5.32
```

Az előző pontban ismertetett típusdeklarációs karakterek segítségével, a gép döntését megelőzve, lehet előírni a bevitelkor megadott konstans ábrázolási formáját.


```
Pl.  J%=100
     RO!=.0173345
     D#=2/3
     A$="CICABABA"
```

A megkívánt ábrázolási formáról az alábbi utasítás segítségével lehet meggyőződni:

```
PRINT J%,RO!,D#,AS
100          .01733745
.66666666666666667
CICABABA
READY
>←
```

A bevitelkor lehetőség van számok lebegőpontos (mantissza és exponens alakú) megadására is.

A=1.2E3: B=12.5E–3 csak a negatív előjelet kell megadni, a tizedespont a mantisszában bárhol lehet!

Ha az exponens jelzésére az E karaktert használjuk, akkor egyszeres, ha a D-t, akkor dupla pontosságú számként tekinti majd a gép a beadott konstanst.

ALFA=1E3 egyszeres pontosságú
BETA=2D–3 dupla pontosságú

ábrázoláshoz vezet.

A következő esetekben különböző pontosságú formában számol, majd az eredményt a kívánt típusra alakítja át a számítógép

Művelet		Számolás típusa	Eredmény típusa
SI!=2D1/3D1	(CR)	dupla	egyszeres
IN%=2/3	(CR)	egyszeres	egész

(Jegyezzük meg, hogy osztás céljára az egész típusú ábrázolás alkalmatlan, ezért az osztások csak egyszeres vagy kétszeres pontosságú alakban hajthatók végre!)

D#=SIN(1)	(CR)	egyszeres	dupla

(Az ún. elemi függvények mindig egyszeres pontosságú eredményt képeznek, de elsőként az argentumot is ilyen típusra alakítja a gép.)

AA!=2*32767	egész	egyszeres

A változók ábrázolási formátumát még másféleképpen is befolyásolhatjuk, de ennek tárgyalására csak a kulcsszavak ismertetésénél kerül sor.

Összefoglalva:

A számítógép tárkapacitásával és a gépidővel lehet és érdemes is jól gazdálkodni. Erre nyújtanak módot a különböző típusú változók. Ám ezek használatához néhány szabály ismerete és bizonyos tapasztalat szükséges, melyeket elsősorban próbálkozásokkal és az eredmények megfigyelésével lehet megszerezni.

4. AZ IPT–002 TÍPUSÚ SZEMÉLYI SZÁMÍTÓGÉP BASIC NYELVI DIALEKTUSÁNAK ISMERTETÉSE

Szeretnénk felhívni a figyelmet arra, hogy a további ismertetés nem a BASIC programozási nyelv oktatását célozza, nem a kezdők szempontjából didaktizált tankönyv, hanem pusztán a címben közölt ismertető. A kezdőknek ajánljuk pl.

Bodor–Gerő:
A BASIC programozás technikája, SZÁMALK
Dr. Kocsis A.:
Programozás BASIC nyelven, SZÁMALK,
Lőcs–Sarkadi–Nagy–Szlankó:
BASIC programozási nyelv c. könyvét.

Az ismertetésnél feltételezzük, hogy az olvasó bizonyos programozási képzettséggel rendelkezik, illetve elvárjuk, hogy ha megakad,az általunk ajánlott tankönyvek segítségével igyekszik túljutni a problémán.

A továbbiakban bemutatott mintaprogramoknál az egyes programutasítások közé a könnyebb olvashatóság és jobb áttekinthetőség céljából szóközöket tettünk. A BASIC-nek ezekre nincs szüksége, a tárat feleslegesen foglalja és a végrehajtást is lassítja.

4.1 Műveletek

4.1.1 Aritmetikai műveletek

Az aritmetikai műveletek egy része ismert a matematikából:

+	összeadás	pl.:	PRINT	2+3
–	kivonás		PRINT	3–2
*	szorzás		PRINT	2*3
/	osztás		PRINT	2/3
↑	hatványozás		PRINT	2↑3

Ezekkel a műveleti jelekkel leírt matematikai kifejezések számításakor a prioritás a fenti sorrendnek megfelelő. Tehát pl. a következő sor kiszámításakor elsőként a hatványozás, majd a szorzás, végül az összeadás következik.

```
PRINT 2+4*2↑3     (CR)
 34
READY
>←
```

Ezt nagyon könnyű megjegyezni, mert matematikailag teljesen „természetes". A képleteket, néhány kivétellel, szinte változtatás nélkül írhatjuk át BASIC-be, ami ennek a programozási nyelvnek egyik legnagyobb érdeme.

A fent bemutatott prioritás megváltoztatása, akárcsak a matematikában itt is zárójelzéssel lehetséges.

```
PRINT (2+4*2)↑3  (CR)
 1000
READY
>←
```

4.1.2 Összehasonlító műveletek

A változók összehasonlítására a következő relációs műveletek szolgálnak:

Műveleti jel	Numerikus változók	Szöveges változók
=	egyenlő	azonos
<	kisebb	ábécé sorrendben megelőzi
>	nagyobb	ábécé sorrendben követi
<=	kisebb egyenlő	ábécé sorrendben megelőzi vagy azonos
>=	nagyobb egyenlő	ábécé sorrendben követi vagy azonos
<>	nem egyenlő	nem azonos

4.1.3 Logikai műveletek

OR logikai – VAGY
AND logikai – ÉS
NOT logikai – NEM

A logikai műveletekkel bonyolult feladatok leírása válik könnyűvé, pl:

Ha A egyenlő B-vel és nem nagyobb 3-nál, vagy ha C egyenlő D-vel és kisebb 10-nél akkor E legyen egyenlő 8-al, egyébként legyen 0.

E=–8*((A=B) AND(A<3) OR (C=D) AND (C<10))

A logikai műveletek –1-et eredményeznek ha a feltételek teljesülnek, és 0-át ha nem.

A logikai műveletek prioritása a fenti sorrendben növekvő, azaz legkisebb a VAGY művelet esetében. Ezért nem kellett a két ÉS feltételt zárójelezni a VAGY művelet két oldalán.

4.2 Parancsok

AUTO	EDIT	RUN
CONT	LIST	TROFF
DELETE	NEW	TRON

A U T O *alapsorszám, növekmény*

A programsorok azonosításához szükséges sorszámok automatikus képzésére szolgál. Beadása után megjelenik a sorszám, azaz a gép **program** üzemmódban van, melyből kilépni a CNTL–A karakterrel lehet. A funkció paramétereinek megadásával, illetve elhagyásával a következő kombinációk adódnak:

AUTO	a 10. sortól	10-esével,
AUTO 25,5	a 25. sortól	5-ösével,
AUTO 60	a 60. sortól	10-esével,
AUTO 5,25	az 5. sortól	25-ösével

nőnek a programsorok sorszámai

Az AUTO utasítás hatására megjelenik:

> 10	A gép a 10. programsor bevitelére vár.
> 15*	A csillag azt jelenti, hogy ilyen sorszámmal már van tárolt programsor, az új bevitel tehát törölné az előző tartalmat.

CNTL–A: READY

><

Ha a megjelent sorszámra a programozó csupán egy (CR) billentyűvel reagál, akkor az AUTO funkció továbblép, tárolás nem történik.

C O N T

Ha valamely program futása pl. egy STOP utasítás vagy egy a billentyűzetről származó CNTL–A karakter hatására megszakadt, akkor a CONT parancs segítségével lehet

folytatni. A program felfüggesztett állapotában meg lehet vizsgálni, vagy akár módosítani a változókat, keresve illetve javítva ezzel a hibákat.

Fontos tudnivaló, hogyha a felfüggesztett programon is változtatunk, akkor az már nem folytatható a CONT paranccsal. Ilyen változtatás csak az EDIT parancs segítségével végezhető; utána a programot a RUN vagy a GOTO utasítással kell elölről indítani.

D E L E T E *sorszám–sorszám*

Feladata a paraméterben megadott programsorok törlése a számítógép tárából.

A paraméterek megadásával/kihagyásával a következő kombinációk adódnak.

DELETE 10	Törli a 10. programsort,
DELETE 10–70	Törli a 10. programsortól kezdődően a 70. programsorig az eddig tárolt BASIC programot.

E D I T *sorszám*

A parancs segítségével a számítógép **szerkesztő** üzemmódba kerül. Ez az üzemmód sokfajta lehetőséget kínál a sorszámmal megadott programsorok megváltoztatására, módosítására és kijavítására; ismertetésére a későbbiekben kerül sor.

L I S T *sorszám–sorszám*

A parancs hatására a számítógép kiírja a paraméterben megadott programsorokat. A paraméterek kombinálásával a következő variációk lehetségesek:

LIST	A paraméterek elhagyásával a gépben tárolt összes programsor kiíródik. A CNTL–B gomb itt is hatásos.
LIST 30	A 30. programsor íródik ki.
LIST 10–90	A megadott programsorok kiíródnak.
LIST –50	A megadott programsorig írja ki a tároltakat.
LIST 100	A megadott programsortól kezdődően írja ki a programsorokat.
LLIST	Hatása azonos a LIST parancséval, de most nem a képernyőre, hanem a nyomtatóra viszi ki az adatokat.

N E W

A parancs törli az összes beírt programot, változókat és adatokat, valamint törli a képernyőt is.

R U N *sorszám*

A parancs hatására a számítógép a **végrehajtás** üzemmódba kerül, és a megadott sorszámtól kezdődően fogja futtatni a programot. A paraméter elhagyása a legkisebb sorszámú programsort jelenti. A RUN hatására az összes változó nulla értéket vesz fel. Lásd még GOTO utasítást is!

RUN	futtatás a program elejétől.
RUN 200	futtatás a 200. programsortól.

A RUN parancs programból is kiadható, de számítani kell a változók törlésére!

T R O F F

Trace off – kikapcsolja a nyomkövető funkciót, lásd TRON-t!

T R O N

Trace on hatására bekapcsolódik a nyomkövető funkció. Segítségével meg lehet figyelni egy program végrehajtását, ami hasznos lehet pl. hibakeresésnél. Minden egyes sor végrehajtását a sorszám kiiratásával tudatja a számítógép. A kiírt sorszámok és a programlista birtokában egyszerűbbé válik a még oly csökönyös hibák felderítése is.

Például:

```
10 FOR A=1 TO 5
20 B=B+A
30 NEXT
40 PRINT "SZUMMA=" ; B
TRON
RUN
```

. . . . . .

```
<10><20><30><10><20><30><10><20>
<30><10><20><30><10><20><30><40>
SZUMMA = 15
READY
```

A CNTL–B nyomógomb itt is hatásos.

A TRON utasítás programból is kiadható, s így a vizsgálódás a ténylegesen megfigyelendő sorokra korlátozható.

4.3 Beviteli és kiviteli utasítások

PRINT	INPUT#
LOCATE	INP
PRINT TAB	OUT
PRINT USING	CLOAD?
LPRINT	CLOAD
INPUT	CSAVE
PRINT #	MERGE

P R I N T *változó , változó . . . . . .*
Az utasítás a konstansok, változók, karakterláncok, kifejezések kiszámítására és a képernyőn való kiírására szolgál.

```
PRINT (2+3*2)↑2
 64
READY
>←

A=PI: B=3*2: C=5/2: D=2*2*A  (CR)
PRINT  A,B,C,D               (CR)
 3.14159          6
 2.5              12.56636
READY
>←
```

Ha a felsorolt változók között vessző van, úgy a kiíratás mindig a következő tabulátorpozíciótól kezdődik. Pontosvessző alkalmazásakor a kiíratás folytatólagosan, azaz a karakterláncoknál szóköz nélkül, a numerikus változóknál egyetlen szóköz beiktatásával történik.

```
A$="CICA": B$="BABA": A=1: B=2
PRINT  A$;B$; A; B;
CICABABA 1 2
READY
>←
```

A PRINT utasítás önmagában alkalmazva egy soremelés-sel azonos hatású.

Végrehajtás üzemmódban, ha a PRINT utasítást követő tétellista pontosvesszővel zárul, akkor a kinyomtatást követő soremelés elmarad. Segítségével roppant hatásos képek kijelzése oldható meg egyszerű módon.

A PRINT utasítás tételei között megadható közvetlenül karakterlánc is, pl:

```
PRINT "KETTŐ A NÉGYZETEN=" ; 2↑2
```

L O C A T E *(kifejezés, kifejezés)*
Az utasítás hatására a kurzor a két kifejezés aktuális értéke által meghatározott X illetve Y koordinátára kerül. Az oszlop kijelölése 0–31-ig, a soré 0–23-ig megengedett.

```
ERASE
FOR A%=1 TO 20
LOCATE (A%,A%): PRINT "*": NEXT
```

A funkció nagyban egyszerűsíti a bonyolult ábrák felrajzolását.

P R I N T T A B*(kifejezés)*
Az utasítás hatására a kurzor a soron belül a kifejezés aktuális értéke által meghatározott pozicióra kerül. A kifejezés aktuális értékének a (0–31) tartományon belül kell lennie. A tabulátor mint a PRINT módosítója többször is előfordulhat a PRINT-et követő listában. Használatát próbálja ki az alábbi egyszerű programmal:

```
10 DEFINT A,B: ERASE: B=31
20 FOR A=1 TO 14
30 PRINT TAB(A) "*" ; TAB(B–A) "*"
40 NEXT :PRINT TAB(15) "OK"
```

Próbálja ki a programot a 30. sor következő módosításával is!

```
30 PRINT TAB(A)"*" ;    TAB(B–A)"*" ;
```

Meglepő a változtatás csekélysége és az eredmény változásának nagysága.

Ha a kiiratásnál a kurzor már túlhaladta a TAB funkcióval kívánt pozíciót, akkor a TAB funkciót figyelmen kívül hagyja és a kiiratás a pillanatnyi pozíciótól kezdődik.

P R I N T U S I N G *karakterlánc*
Ezzel a PRINT módosítóval lehet előírni a kiírandó numerikus és karakteres változók formátumát.
Numerikus változóknál a következő karakterek kombinációival lehet operálni, figyeljük meg a kerekítéseket is.

Jel	Megjegyzés		
#	Minden egyes jel egy számjegyet jelöl	##	12
.	Tizedespont, az egész és tizedes jegyek előírásához	##.##	12.3
+	Előjel kiírható akár a szám előtt, akár utána	##.+	12.3+
		+##.#	+12.3
–	Csak a negatív előjelet írja ki, de nem elől, ahol egyébként is kiírná.	##.#–	12.3–
**	A # jelek által kijelölt helyet töltse ki csillaggal, ha oda nem kerülhet értékes szám.	**###.#	*12.3
↑↑↑↑	Exponenciális formátum	#.#↑↑↑↑	1.2 E+01
		##.##↑↑↑↑	1.23 E+02
,	Az egész rész angol-amerikai szokások szerinti 3-as tördeléséhez a vesszőt a tizedespont előtt és az első # után bárhol el lehet helyezni.		643,246,123.19

Ha az USING-ot követő karakterláncot más karakterekkel is kiegészítjük, azok végrehajtáskor a megjelölt helyen kiíródnak.

```
FOR A=1E3 TO 9E3 STEP 1E3
PRINT USING "BER"= ####.##FT;A
NEXT
```

Ha a változó az előírt számú számjeggyel nem írható ki, akkor a lehetséges formátum hibás kiírását a % jel előzi meg.

Szöveges változók esetén a következő lehetőségek állnak rendelkezésre:

a) ! jel mint módosító

Segítségével a szöveges változó legelső karakterét lehet leválasztani.

```
A$="KUKA"
PRINT USING"!" ; A$
K
READY
```

Többszörös kifejezés természetesen itt is megengedett.

```
A$="AUTOMATIC"
B$="FREQUENCY"
C$="CONTROL"
PRINT  USING"!";A$;B$;C$
AFC
READY
>←
```

b) % jel mint módosító

Segítségével a kifejezések között felsorolt szöveges változók elejéről egynél több karakter is leválasztható.
Az utasítás alakja:

```
USING  "%␣␣␣␣ %";
        szóköz
```

A leválasztandó karakterek száma megegyezik az idézőjelek közötti szóközök és % jelek számával.
A USING utasítást követő karakterláncot nem csak közvetlenül, hanem közvetve is meg lehet adni.

```
U$=" #.# #  ↑↑↑↑ ":  A=123
PRINT  USING  U$;A
1.23E+02
READY
>←
```

L P R I N T
Hatása azonos a PRINT utasításával, különbség az, hogy az adatok a sornyomtatón íródnak ki.

I N P U T *változó, változó, változó* . . . . . .
Az INPUT és PRINT segítségével lehetővé válik a dialógus a gép és a kezelő személy között.

Az INPUT utasítás hatására a program futása felfüggesztődik és a felhasználó különböző változókat adhat meg a billentyűzetről.

10 INPUT A1%, A2%, A3%, A4%,
Ha vezérlés a 10. sorra adódik, egy kérdőjel jelenik meg. A válasznál csak konkrét konstansokat lehet megadni; kifejezések megadása **nem megengedett**!

A bevitelt elvégezhetjük egyenként vagy akár csoportosan is.

Futtassuk le a fenti 10-es programsort

? 10 (CR) ?? 20, 30 (CR) ?? A B C D (CR)	Az első bevitel után két kérdőjel figyelmeztet, hogy a gép további adatokat vár. Csoportos bevitelnél a számokat vesszővel kell elválasztani.
REDO ?? 40, 50 (CR)	Elfogadhatatlan bevitel visszautasításakor a REDO üzenetet kapja a felhasználó.
EXTRA IGNORED	Ha a tétellista által kívánt adatnál több adat bevitelét kísérli meg a felhasználó, úgy az EXTRA IGNORED üzenet kiírása után folytatódik a program (a többletet figyelmen kívül hagyja).

Az INPUT utasítás adatainak rendeltetéséről a következőképpen lehet tájékoztatást adni:

```
10 INPUT "ELLENÁLLÁS"=; R
20 INPUT "FESZÜLTSÉG"=; U
30 PRINT "ÁRAMERŐSSÉG"=; U/R
```

Az INPUT utasítást csak programból lehet kiadni.

P R I N T# *adat, adat, "karakterlánc"*. . . . .
Az utasítás a változólistában felsorolt változókat, kifejezések aktuális értékét, idézőjelek között közölt karakterláncokat a megadott sorrendben kazettára rögzíti.

PRINT # VX, VY, X0, 0, G, "FERDE HAJITÁS"

Megjegyzés: A PRINT és INPUT utasítások végrehajtásához nem automatikus szintbeállítós, ALC-s, jó minőségű magnóra van szükség!

I N P U T # *változólista*
Az utasítás a PRINT utasítás inverze. Hatására az előzőekben kazettára rögzített változókat a számítógép beolvassa a kazettáról. Fontos, hogy a változólista teljesen azonos legyen a rögzítéskor megadottal!!

Az INPUT utasítást csak programból lehet kiadni.

I N P (*port*)
Az utasítás a készülékhez bővítésként kapcsolt bemenő portról 1 byte behozatalát teszi lehetővé. A zárójelben paraméterként a port I/O címét kell megadni.

```
A=INP(255)
```

O U T *port, kifejezés*
Az utasítás a készülékhez bővítésként kapcsolt kimenő portra 1 byte kivitelét teszi lehetővé. Első paraméterként a port I/O címét kell megadni, második praméter maga az adat.

OUT 255, (A+B) / C

C L O A D? *"állománynév"*
Az utasítás lehetőséget nyújt a tárban levő és a kazettára rögzített BASIC programok összehasonlítására.
Amint a számítógép megtalálja az adott állománynévvel (file-névvel) azonosított felvételt a szalagon, bitről-bitre összehasonlítja a felvételt a tár tartalmával. Ha bármilyen különbség lelhető fel, akkor erről a "FD"-ERROR üzenet tájékoztatja a felhasználót, de a magnót csak az állomány végén állítja le!

Ha állománynév nélkül alkalmazzuk a CLOAD? utasítást, akkor az a tár tartalmát a szalagon található első felvétellel fogja összehasonlítani.

A CLOAD? utasítás ellentétben a CLOAD-dal nem törli a tár tartalmát.

Az utasítás felhasználható egy adott állomány (felvétel) végének megkeresésére is!

C L O A D *"állománynév"*
Az utasítás a kazettára rögzített BASIC programot a gép tárába tölti. A szalag kívánt pozíciójáról előzőleg a felhasználónak kell gondoskodnia.

Az utasítás végrehajtásakor elsőként a tár pillanatnyi tartalmát törli a gép, majd bekapcsolja a magnó motorját és megindul a keresés a megadott azonosítójú felvétel után.

Ha a gép egyező nevű felvételt talál, akkor azt a tárba tölti. Ha tehát a szalag elejére csévélünk és a kazettán bárhol is létezik a betölteni kívánt program felvétele, akkor azt előbb-utóbb sikeresen beolvashatjuk a tárba.

Az olvasott felvételek azonosítója a képernyő jobb felső sarkában jelenik meg. Ha a CLOAD utasítást állománynév nélkül adjuk ki, úgy az első megtalált állományt fogja beolvasni.
A CLOAD utasítás végrehajtása után a gép leáll!

C S A V E *"állománynév"*
Az utasítás a tárban levő programot kazettára rögzíti. A rögzítendő program azonosítása céljából az idézőjelek között megfelelő állománynevet kell megadni. Kikötés, hogy az állománynév max. 8 karaktere között az idézőjel nem

szerepelhet. A kazetta kivánt pozíciójáról a felhasználónak kell gondoskodnia. A biztonságos rögzítés szempontjából ajánlatos egy programot többször is felvenni.

```
FOR A=1 TO 3: CSAVE "NEWTON": NEXT
```

Rögzítéskor a magnó motorjának indítása és megállítása automatikus.

A 8 karakternél rövidebb állományneveket a gép szóközökkel egészíti ki!

4.4 Adatmegadás

Az adatok közlésére, felhasználására, illetve azok szervezésére az alábbi utasítások szolgálnak.

CLEAR
DATA
DIM
LET
READ
RESTORE

CLEAR *kifejezés*

Az utasítás hatására a számítógép a kifejezés aktuális értékének megfelelő tárterületet rezervál a szöveg típusú változók számára, továbbá az összes változónak nulla értéket ad.

A kifejezés nem vehet fel negatív értéket, illetve nem haladhatja meg a maximális memóriaértéket.

Argumentum nélkül használva 100 byte áll a szöveges változók rendelkezésére. A számítógép bekapcsolásakor is ekkora terület lesz automatikusan kijelölve.

Ha a karakterlánc-műveletek közben túllépnénk a karakterek számára kijelölt területet, akkor OS(out of string) hibajelzést fogunk kapni.

CLEAR MEM/4	A teljes tár egynegyedét jelöltük ki a karakterláncok számára.
CLEAR 100	A bekapcsoláskor automatikusan végrehajtott 100 byte-os rezerválással ekvivalens utasítás.
CLEAR 0	Ha a program egyáltalán nem használ szöveges változókat, úgy ezzel az utasítással a teljes memória a programok és egyéb adatok számára használható fel.

D A T A *adatlista*

A DATA utasítás után különböző típusú numerikus konstansokat és szöveges változókat adhatunk meg, egymástól vesszővel elválasztva. Kifejezéseket nem szabad alkalmazni!

Az így kapott adattömb soron következő elemeit a READ utasítással lehet elérni!

Az utasításpár segítségével nehezen vagy egyáltalán nem megadható függvénykapcsolatokat is könnyedén megvalósíthatunk.

Jól felhasználható pl. ábrák kiegészítéséhez, ahol az X;Y koordinátákat és az így megadott pontra irandó karaktereket a DATA utasítás adatlistájában felsorolt adatok szolgáltatják.

Ha több adatot kívánunk olvasni, mint amennyi a programban található valamennyi DATA utasítás összes eleme, akkor OD (out of data) hibajelzést fogunk kapni.
Lásd még READ, RESTORE utasításokat!

D I M *név (dim 1, dim 2, ...dim n)*

Az utasítás segítségével *n* dimenziós, megadható mélységű mátrixot alakíthatunk, melynek elemeit kényelmesen, a rendezők felsorolásával érhetjük el.

a)
```
DIM  A (10, 10)
S=0
FOR  I=1  TO  10
FOR  J=1  TO  10
S=S+A(I, J)
NEXT  J,I
```

A program a 10 x 10 mátrix 100 elemének az összegét állítja elő

b)
```
DIM   B  (20)
DIM  C%(3,3)
DIM  D$(16)
DIM  E#(10,10,)
```

Egyszeres pontosságú
Egész
Karakteres
Dupla pontosságú

} tömbök létrehozása

L E T *változó = kifejezés*
változó = kifejezés

Általános értékadó utasítás. A LET kiírása nem szükséges, csak más BASIC nyelvi dialektusokkal való kompatibilitásra szolgál.

A=12+3*4–(4–8)+A↑3	36
A=A+A/2	54
C$="MA"	MA
C$=C$+C$	MAMA
D$=INKEY$	Az éppen megnyomott billentyű karaktere.

```
E=MEM/2                                  A szabad memória fele
F=(A=B)*1"+(A>B)*2+(A<B)*4               F = -1 ha A = B
                                         F = -2 ha A > B
                                         F = -4 ha A < B
```

R E A D *változólista*

Az utasítás a DATA utasítás változólistájában megadott numerikus konstansok és karakterláncok sorrendi elérésére szolgál.

A program futásakor az első READ utasítás hatására, a programsorszámok alapján elsőnek számító DATA utasítás listájának első eleme lesz felhasználva. A következő READ végrehajtásakor ugyanezen DATA utasítás soronkövetkező elemét olvassuk be.

Ha az egymást követő olvasásokkal az első DATA utasítás változólistájának összes elemét felhasználtuk, akkor automatikusan a következő DATA utasítás első elemére lépünk tovább.

Ha többet olvasnánk, mint amennyit a DATA utasítások lehetővé tesznek, úgy OD(out of data) hibajelzést fogunk kapni.

```
10 DATA  1, 2, "ELSŐ DATA VÉGE"
20 READ  A: PRINT A
30 READ  B, C$, D$
40 PRINT B; C$; D$;
50 DATA  "MÁSODIK DATA ELEJE", 3
60 READ  E
70 PRINT E
80 READ  F
90 END
   futásakor a következőket kapjuk
   1
   2 ELSŐ DATA VÉGE MÁSODIK DATA ELEJE
   3
   OD ERROR IN 80
   READY
```

Lásd még DATA és RESTORE utasításokat!

R E S T O R E

A RESTORE hatására a következő READ utasítás, bárhonnan olvasott előzőleg, a legelső DATA utasítás legelső elemétől kezdődően fog olvasni. Segítségével lehetőség van a korábban már használt konstansok ismételt beolvasására.

Ha az előző mintaprogramban 75. sorként egy RESTORE utasítást alkalmaznánk, hibaüzenet nélkül egy 1-est olvasnánk be az F jelű változóba.

4.5 Adattípusok megadása

A változók típusai és lehetséges azonosítóik, valamint a változók megadása fejezetben leírtakon kívül az alábbi utasítások könnyíthetik meg a programozási munkát.

CINT	DEFDBL
CDBL	DEFINT
CSNG	DEGSNG
	DEFSTR

C I N T *(kifejezés)*
Az utasítás a kifejezés aktuális értékét, vagy az akként szereplő más típusú numerikus változót egész típusúra alakítja át.

Az egész típusú változó kisebb helyet és rövidebb feldolgozási időt igényel, tehát használata – ha az a pontossági igényeknek is megfelel – felettébb célszerű.

Az átalakított értéknek a – 32768, + 32767 tartományba kell esnie, egyébként OV (overflow) hibát okoz.
Lásd az INT (X) függvényt is!

C D B L *(kifejezés)*
Az utasítás a kifejezés aktuális értékét, vagy az akként szereplő más típusú numerikus változót dupla pontosságúra alakítja át.

A dupla pontosságú változó 17 számjeggyel számol és pontossága kárpótolhatja a programozót a nagyobb tárigényért és a hosszabb műveleti időért.

A dupla pontosságú változókon azonban csak a négy alapművelet végezhető el ténylegesen. Minden más művelet előbb egyszeres pontosságura konvertálódik, majd a művelet elvégzése után vissza, csakhogy ez nem növeli meg az értékes jegyek számát!

C S N G *(kifejezés)*
Az utasítás a kifejezés aktuális értékét, vagy az akként szereplő más típusú numerikus változót egyszeres pontosságura alakítja át.

Az egyszeres pontosságú változó a középút az egész és a dupla pontosságuak között. Az összes aritmetikai függvény közvetlenül egyszeres pontosságú számábrázolással hajtódik végre, és 7 értékes számjegyet szolgáltat.

D E F D B L *betű, betű-betű*
Az utasítás hatására a paraméterként megadott betűvel (vagy a betűhatárok és a közöttük lévő betűkkel) kezdődő azonosítójú változókat automatikusan dupla pontosságuként

kezeli a gép, hacsak külön típusdeklaráló karaktert nem alkalmaz a felhasználó.

Az utasítás segitségével a típusdeklaráló karakterek elmaradhatnak, ami egyrészt csökkenti a hibalehetőségeket, másrészt ezek gyakori szükségessége esetén csökkenti a tárigényt és kényelmesebbé teszi a programozói munkát.

DEFDBL A, B, E–K, Z	A, B, E, F, G, H, I, J, K, Z betűvel kezdődő azonosítójú változók dupla pontosságúak lesznek

D E F I N T *betű, betű-betű*

Az utasítás hatására a paraméterként megadott betűvel (vagy a betűhatárok és a közöttük lévő betűkkel) kezdődő azonosítójú változókat automatikusan egész típusukként kezeli a gép, hacsak külön típusdeklaráló karaktert nem alkalmaz a felhasználó.

D E F S N G *betű, betű-betű*

Az utasítás hatására a paraméterként megadott betűvel (vagy a betűhatárok és a közöttük lévő betűkkel) kezdődő azonosítójú változókat automatikusan egyszeres pontosságukként kezeli a gép, hacsak külön típusdeklaráló karaktert nem alkalmaz a felhasználó.

DEFSNG A–D, K–M A 1%=1	A, B, C, D, K, L, M betűvel kezdődő azonosítójú változók egyszeres pontosságúak lesznek, kivétel A1%

D E F S T R *betű, betű-betű*

Az utasítás hatására a paraméterként megadott betűvel, (vagy a betűhatárok és a közöttük lévő betűkkel) kezdődő azonosítójú változókat automatikusan karakterláncnak tekinti a gép, hacsak külön típusdeklaráló karaktert nem alkalmaz a felhasználó.

4.6 Aritmetikai függvények

A következő függvények esetében, ha van argumentum az mindig zárójelben van, és az eredmény primer formában egyszeres pontosságukként keletkezik.

A B S	E X P	P I	S I N
A T N	F I X	R A D	S Q R
C O S	I N T	R A N D O M	T A N
D E G	L N T	R N D	
	L O G	S G N	

A B S (*kifejezés*)
A kifejezés aktuális értékének az abszolut értékét szolgáltatja.

A T N (*kifejezés*)
A kifejezés aktuális értékének az arkusz tangesét szolgáltatja. A kifejezés értékét alapállapotban, vagy a RAD utasítás végrehajtása után radiánban, DEG után pedig fokban értelmezi.

C O S (*kifejezés*)
A kifejezés aktuális értékének a koszinuszát szolgáltatja. A kifejezés értékét alapállapotában, vagy a RAD utasítás végrehajtása után radiánban, DEG után pedig fokban értelmezi.

D E G
Végrehajtása után az összes szögfüggvény argumentumát illetve az inverz szögfüggvények eredményét fokban kell értelmezni.

E X P (*kifejezés*)
A kifejezés aktuális értékének megfelelő hatványra emeli a természetes alapot (e^X).

F I X (*kifejezés*)
Pozitív számok esetén a kifejezés aktuális értékének INT értékét adja. Negatív számok esetén a kifejezés aktuális értékének INT + 1 értékét adja.

I N T (–3.5)=–4; I N T (3.5) = 3
F I X (–3.5)=–3; F I X (3.5) = 3

I N T (*kifejezés*)
A kifejezés aktuális értékének egész részét szolgáltatja, azaz levonja a törtrészt.
Tetszőleges, egyébként ábrázolható számhatárok között alkalmazható.

L N T (*kifejezés*)
A kifejezés aktuális értékének természetes logaritmusát szolgáltatja.

L O G (*kifejezés*)
A kifejezés aktuális értékének tizes alapú logaritmusát szolgáltatja.

P I
A Ludolf-féle számot (3.14159) szolgáltatja.

R A D
Végrehajtása után az összes szögfüggvény argumentumát illetve az inverz szögfüggvények eredményeit radiánban kell értelmezni.
Bekapcsoláskor, tehát alapállapotban a gép a szögfüggvényeket radiánban értelmezi.

R A N D O M
A függvény feladata a véletlenszám-generátor kezdőértékének véletlenszerű alapbeállítása.
Olyan programokban, ahol az RND függvény használata alapvető, a program elején célszerű kiadni.

R N D (*kifejezés*)
A függvény végrehajtása után egy véletlenszámot nyer a programozó.

RND (0) egy véletlenszámot generál a (0–1) tartományban. A kapott szám egyszeres pontosságú

RND (*kifejezés*)
egy véletlenszámot generál a + 1 és a kifejezés aktuális értékének egész része között. A kapott szám egész típusú, tehát + 32768-nál kisebb!

Ha tehát nem egész típusú pl. (0–100) intervallumba eső véletlenszámra van szükség, akkor következő megoldás lehetséges.

VS=RND (0) * 100

S G N *(kifejezés)*
A kifejezés aktuális értékének az előjelét szolgáltatja:

pozitív számokra	+ 1,
zérusra	0,
negatív számokra	–1.

S I N *(kifejezés)*
A kifejezés aktuális értékének a szinusát szolgáltatja. A kifejezés értékét alapállapotban (bekapcsolás után), vagy a RAD utasítás után radiánban, a DEG után pedig fokban értelmezi.

S Q R (*kifejezés*)
A kifejezés aktuális értékének a négyzetgyökét szolgáltatja. Használata gyorsabb mint a vele egyébként ekvivalens (*kifejezés*) ↑ (1/2) utasítás. Itt jegyezzük meg, hogy az X végrehajtása is gyorsabb mint az X ↑ 2-é.

T A N *(kifejezés)*
A kifejezés aktuális értékének a tangensét szolgáltatja. A kifejezés értékét alapállapotban, (bekapcsolás után) vagy a RAD utasítás végrehajtása után radiánban, a DEG után pedig fokban értelmezi.

4.7 Karakterlánc függvények

ASC	MID$
CHR$	RIGHT$
FRE	STR$
INKEY$	STRING$
LEFT$	VAL
LEN	

A S C (*szöveges kifejezés)*
A zárójelben megadott szöveges kifejezés első karakterének ASCII kódját szolgáltatja.

```
a) PRINT  ASC ("A")
    65
   READY
   >←
```



```
b) A$ = "ORA"
   PRINT  ASC (A$)
    79
   READY
   >←
```

c) PRINT ASC (RIGHTS (A$, 3)) (példa kifejezésre)

C H R S *(kifejezés)*
A CHRS utasítás az ASC inverzeként fogható fel. A kifejezés aktuális értékének megfelelő karaktert szolgáltatja. Ennek megfelelően a kifejezés aktuális értékének a (0,255) intervallumba kell esnie.
A karakterkészlet kiíratását a következő programmal érhetjük el.

```
FOR A = 32 TO 95
PRINT CHR$ (A): NEXT
```

F R E *(szöveg)*
A függvény a szöveg számára fenntartott szabad tárterület nagyságát szolgáltatja.

```
CLEAR 200
PRINT FRE (A$)
 200
READY
>←
```

I N K E Y $
A függvény a billentyűzet éppen lenyomott gombjának megfelelő karakterláncot szolgáltatja. Ha nincs lenyomott gomb, úgy egy NULL láncot kap a felhasználó.

A függvény azért hasznos, mert a program futását nem kell felfüggeszteni, mint pl. INPUT esetében, továbbá nem jelenik meg a beadott adat sem a képernyőn.
Kiválóan felhasználható pl. olyan játékok esetén, ahol a képernyőn valamilyen mozgás utánzása folyik és a mozgás irányát pedig a billentyűzetről befolyásoljuk.

L E F T $ (*szöveges kifejezés, n*)
A függvény a megadott szöveges kifejezés első karakterét szolgáltatja.

Használatával tehát az adott szöveg bal oldaláról konkrét számmal (*n*) illetve kifejezéssel megadott számú karaktert lehet leválasztani.

```
A$="TA": B$="KARO "
PRINT LEFT$ (A$ + B$, LEN (A$ + B$−1)
TAKAR
READY
>←
```


Ezzel azonos eredményt számolna a PRINT LEFT$ ("TAKARO", 5) utasítás végrehajtása is.

L E N (*szöveges kifejezés*)
A függvény a szöveges kifejezés eredményéül adódó szöveg hosszát, azaz karaktereinek számát szolgáltatja.

```
10 INPUT "SZÖVEG = " ; A$
20 C=LEN (A$)
30 FOR B=1 TO C
40 PRINT LEFT$ (A$, B)
50 NEXT
60 GOTO 10
70 END
```

A fenti példában különböző hosszúságú szövegeket lehet megadni, melyekkel a program azonos módon képes bánni. Ütse be a programot és futtasa le!

Próbálja ki úgy is, hogy a 40. sor végén egy pontosvesszőt is alkalmaz (;).

M I D $ (*szöveges kifejezés, p, n*)
A függvény a megadott szöveges kifejezés *p*-edik pozíciójától számolt *n* darab karakterét szolgáltatja.
Mind a pozíció (*p*)mind a darabszám (*n*) is lehet kifejezés. A függvénnyel tehát az adott szöveg közbülső karaktereit lehet kiválasztani.

R I G H T $ (*szöveges kifejezés, n*)
A függvény a megadott szöveges kifejezés jobbról számított *n* darab karakterét szolgáltatja. Segítségével tehát az adott szöveg hagyományosan értelmezett utolsó *n* elemét lehet leválasztani. Az *n* lehet kifejezés is!

```
A$="CICA"
PRINT RIGHTS (A$, 3)
ICA
READY
>←
```


Megjegyzés: ugyanerre az eredményre vezetne ebben az esetben a PRINT MIDS (AS, 2, 3) utasítás is.

S T R $ (*kifejezés*)
A függvény a kifejezés aktuális értékének megfelelő – a kiszámolt értéket leíró – szöveget hozza létre. Feladata tehát a numerikus érték olvasható karakterlánccá való alakítása.

V A L (*szöveg*)
A függvény az STR$ inverzeként fogható fel. Feladata a karakteresen rendelkezésre álló numerikus információt a belső számábrázolásnak megfelelő (bináris) formátumra alakítani.

S T R I N G $ (*kifejezés, karakter)*
A függvény a kifejezés aktuális értékének megfelelő számú, paraméterként megadott karakterből álló karakterláncot alkot.

```
PRINT  STRING (16, " – " )
----------------
READY
>←
```

Rendkívül hasznos grafikák, táblázatok stb, készítésénél. A karaktert szöveges kifejezéssel is meg lehet adni!

4.8 Logikai függvények

A logikai függvények előállítása az alábbi három alapvető művelet segítésével lehetséges:

A N D
O R
N O T

Az operátorok két 16 bites előjeles egész típusú – vagy ilyen típusra átalakított – operandusok között 16 bites, bitenkénti logikai műveletet végeznek. A negatív számok kettes komplemens ábrázolásúak.
Ha az átalakítandó számok nem esnek bele a
(– 32768, + 32767)
tartományba, az operátorok használata nem lehetséges, amire FC (function) hibajelzés utal.

*op*l AND *op*2
16 bites bitenkénti logikai ÉS művelet.

250 AND 15
mert

```
1 1 1 1 1 0 1 0
0 0 0 0 1 1 1 1
---------------
0 0 0 0 1 0 1 0   10 dec
```

*op*1 OR *op*2
16 bites bitenkénti logikai VAGY művelet.

162 OR 10
mert

```
1 0 1 0 0 0 1 0
0 0 0 0 1 0 1 0
---------------
1 0 1 0 1 0 1 0   170 dec
```

NOT *op*
A művelet az operandus kettes komplemensét állítja elő.

NOT 0 — 1 dec
mert 0000 0000 0000 0000$_2$=0$_{10}$
1111 1111 1111 1111$_2$=-1$_{10}$

A tárgyalt műveletek egyik lehetséges felhasználása a különféle egyszerű, illetve bonyolult logikai függvények létrehozásán kívül az INP, OUT utasításokkal elérhető bemenő illetve kimenő portok állapot(státusz) regiszterei-nek kezelése és azok bitjeinek megfelelő kiértékelése.

A műveletek felhasználhatók a többféle feltételt alkalmazó feltételes utasításokban a következő jelentéssel:

AND – ÉS
OR – VAGY
NOT – NEM

a) IF (A>3 AND A<15) AND NOT A=9 THEN C=1 ELSE C=2
olvasva: ha A nagyobb 3-nál, és kisebb 15-nél de nem egyenlő 9-el, akkor C egyenlő eggyel, egyébként C=2

b) A feltételek teljesülése (–1)-es érték átadását jelenti.

Q = (A=B) AND (C=D) Q = –1, ha a feltételek teljesülnek, ha nem Q = 0
Q = NOT C<0 Q = –1, ha a C pozitív vagy nulla, ha negatív Q = 0

4.9 Programutasítások

END
FOR TO STEP NEXT
GOTO
GOSUB
IF THEN ELSE
ON
STOP
REM
RESUME
RETURN

E N D
Hatására a program futása befejeződik. Használata csak akkor szükséges, ha a felhasználó programjának logikai vége nem azonos a fizikai végével.

```
10  REM  " 1. PROGRAM"
20
90  PRINT  "1. PROGRAM EREDMÉNYE:"
100 END
110 REM  "2. PROGRAM"
```

A mintapéldákban a program több, külön is futtatható részből áll. Hogy az első program, a 90. sor lefutása után ne haladhasson a 110. sor felé, szükséges a 100. sor END utasítása.

FOR TO STEP NEXT

FOR *változó = kifejezés* TO *kifejezés* STEP *kifejezés*
: *utasítások* : NEXT *változó*

Ez az utasítás a BASIC nyelv ciklusutasításának általános alakja.

Az egyenlőségjel jobb oldalán álló kifejezés aktuális értéke a ciklusváltozó kezdeti értéke, míg a végső értéke pedig a TO kulcsszó mögött álló kifejezés aktuális értéke lesz.

A kezdeti és a végső érték között minden egyes ciklusban a STEP mögött álló kifejezés aktuális értékének megfelelő értékkel növekszik a ciklusváltozó értéke. Ha a STEP és a mögötte álló kifejezés elmarad, úgy a növekményt a BASIC automatikusan + 1-nek tekinti.

A ciklusutasítás és a NEXT utasítás között lévő szimbólikusan jelölt utasításokat addig fogja a gép végrehajtani, míg csak a ciklusváltozó növekményekkel figyelembe vett aktuális értéke meg nem haladja az előírt végértéket.

Számítsuk ki az első 100 természetes szám összegét:

```
FOR  A%=1  TO  100:  B%=B%+A%
NEXT A%: PRINT B%
  5050
READY
>←
```

Ebben az esetben amikor az A% ciklusváltozó értéke 101-re emelkedett, a ciklus végetért. Ha a STEP mögötti kifejezés negatív értékű, a ciklus akkor ér véget, ha a ciklusváltozó aktuális értéke kisebbé válik mint a megadott végső érték.

A fenti példával azonos eredményt szolgáltat a következő utasítássor is:

```
FOR A%=100 TO 1 STEP – 1
B%=B%+A%: NEXT A%: PRINT B%
 5050
READY
>←
```

Minthogy mindig csak az aktív FOR ciklust lehet NEXT-el zárni, felesleges a NEXT után megadni a vonatkozó ciklusváltozót. Hiszen megadása csökkenti a szabad tárat, növeli a feldolgozási időt, eltévesztése pedig csak hibákat okozhat!

A ciklusváltozók megadása a NEXT után csak a programírás stádiumában, a készülő program áttekinthetősége miatt célszerű. „Belőtt" programokból a futási idő csökkentése érdekében érdemes a ciklusváltozót elhagyni.

A futási idő szempontjából optimális, ha a ciklusváltozó és a határadatok illetve a növekmény egész típusú szám. Ez persze nem jelenti tetszőleges egyszeres és kétszeres pontosságú számok előírásának a kirekesztését.

A ciklusokat egymásba lehet skatulyázni.

Pl. az A (I,J,K) és a B (I,J,K) háromdimenziós mátrixok összeadását az alábbi ciklusokkal lehet elvégezni:

```
10 FOR K = 1 TO D3                    3.dimenzió
20 FOR J = 1 TO D2                    2.dimenzió
30 FOR I = 1 TO D1                    1.dimenzió
40 C(I,J,K) = A(I,J,K) + B (I,J,K)
50 NEXT
60 NEXT
70 NEXT
```


A feldolgozás szempontjából itt viszont célszerűbb az 50. 60. 70. sor alábbi összevonása:

```
50 NEXT I,J,K
```

Ezzel a szükséges tárkapacitás és feldolgozási idő is csökkenthető.

G O T O *sorszám*

A feltétel nélküli vezérlésátadó utasítás hatására a sorszámnak megfelelő BASIC programsorra adódik át a vezérlés. Különböző feltételes utasításokkal kombinálva a GOTO feltételes vezérlésátadó utasítássá alakítható.!

Nemlétező programsorra való utalás UL (undefined line) hibát okoz!

A GOTO utasítás a programok indítására is használható. A különbség a RUN és közötte, hogy ekkor a gép

előzetesen nem törli az összes változó értékét, és így a már kiszámolt értékek átadhatók a programnak.

G O S U B *sorszám*

A program különböző helyein előforduló és azonos funkciókat megvalósító programsorokat célszerű szubrutinná alakítani. A szubrutinok különböző helyről való hívására a GOSUB, a hívás helyére való visszatérésre pedig a RETURN utasítás szolgál (lásd 47. oldal).

A GOSUB utasítás tehát feltétel nélküli szubrutinhívó utasítás, amit persze feltételes utasításokkal kombinálva feltételes szubrutinhívásra is felhasználhatnak.

Nem létező programsorra való utalás UL (undefined line) hibát okoz!

Példaként álljon itt a Newton–Raphson féle iteráció programja. A 200. sorban megadott szubrutin tartalmazza a megoldandó egyenlet nullára redukált alakját. Példánkban a SIN (X) = COS (X) egyenletet oldjuk meg.

```
10  INPUT "KÖZELITÉS =" ; C
15  A=1E–3:  DEG
20  X=C:  GOSUB  200
30  Z=Y:  X=A+C
40  GOSUB  200
50  L=C:  C=D–A*Z*(Y – Z)
60  IF  ABS(D–C)=1E–6  THEN  20
70  PRINT  "GYÖK=";  C
80  END
200  Y=SIN  (X)–COS (X)
210  RETURN
```

Közelítésként adjunk 30 fokot! A program rövid idő alatt meghatározza a 45 fokot, az egyenlet jól ismert megoldását.

A fenti program alkalmas olyan egyenletek megoldására is, amelyeket analitikusan már nem lehet megoldani.
Például próbálkozzunk az $X^X=10$ egyenletet kielégítő X érték meghatározásával!
X értéke 2 és 3 között van, hiszen:

$2^2 = 4$ $\qquad$ $3^3 = 27$

```
200  Y = X↑X – 10
```

Futtassuk le a programot! Gondoljuk meg, hogy egyébként ezt az egyszerű egyenletet nem tudnánk megoldani! A megoldás: 2.506184146

IF *feltétel* THEN *utasítássorszám* ELSE *utasítássorszám*
A fenti sor a BASIC nyelv feltételes utasításának általános alakja.
Ennek az általános alaknak különböző variációit példákon mutatjuk be.

a) 10 IF A=B PRINT "EGYENLŐ"

ha A≠B akkor semmit sem nyomtat ki, a feltétel vizsgálata után a következő programsor hajtódik végre.

b) 10 IF A THEN 200

Jelentése: ha A pozitív ugorj a 200. sorra, egyébként hajtsd végre a következő programsort.

Az IF A tehát azonos az IF A>=0 feltétellel, rövidebb, de nem olyan magától értetődő.
10 IF A<0 THEN A=ABS(A)
Ha A negatív, akkor annak abszolut értékét kell venni.

Ez a programsor csak oktatási célokra szolgál. Célszerű a 10. sort az A=ABS (A) utasítással helyettesíteni!

Bár az IF utasítás utáni feltétel kiértékelésével egyértelmű, hogy teljesüléskor a programsoron belül, nem teljesüléskor pedig a sorszám szerint következő programsorban van a folytatás, ezt hangsúlyozni lehet a THEN utasítás alkalmazásával. Emellett a THEN, ha sorszámra alkalmazzuk, azonos egy GOTO utasítással. A mintapéldák ezt is bemutatták.

Az IF THEN ELSE hármas egyértelműbben és konkrétan írja elő a feltétel teljesüléskor és nem teljesüléskor végrehajtandó utasításokat. Ez a hármas nem von bele más sorszámú programsorokat, tehát egyszerűsíti a programozói munkát.

A GOSUB utasításnál leírt Newton–Raphson iterációs program 60–80. sora az IF THEN ELSE felhasználásával a következőképpen módosítható:

```
60 IF ABS(D-C)>=1E-6 THEN 20 ELSE PRINT "GYÖK"=";C
END
```

Az így módosított program a tárat kevésbé terheli és futása is gyorsabb!

Az IF THEN ELSE feltételes utasítások egymásba is skatulyázhatók, azaz megengedett a THEN után újabb IF . . . . . .használata, de ezt csak gyakorlottabb programozóknak ajánljuk!

O N

a) ON *kifejezés* GOTO *sorszám* 1, *sorszám* 2, *sorszám n* . . .
Az utasítás a kifejezés aktuális értékétől függő *n* irányú szétágazás megvalósítására alkalmas.
Elsőként a kifejezés aktuális értékének egész részét számolja ki a gép, majd a GOTO utáni címlista ennyiedik elemét tekinti tényleges ugrási címnek. A címlistának közvetlenül a GOTO után álló eleme az 1-es sorszámú.

Ha a kifejezés aktuális értékének egész része a változólista egyetlen elemére sem utal, mert pl. nulla, vagy nagyobb mint *n*, úgy a vezérlés a következő sorra adódik át. Amennyiben negatív szám adódik, úgy hibajelzést kap a felhasználó!
Mintapéldánkban egy csillagot mozgathatunk a képernyő területén fel-le, jobbra-balra az F, L, J, B billentyűkkel! Csak ezek a billentyűk lehetnek hatásosak és a képernyőt elhagyni nem lehet!

```
1 ERASE: K=16: X=16: Y=12
3 ON ENROB GOTO 200
5 GOTO 110
10 A$=INKEY$: IF A$=THEN 10
25 REM "FEL.......LE.....BALRA....
   JOBBRA....."
30 B=(A$=F)*1+(A$=L)*2+(A$=B)*3+(A$=J)*4
40 ON B GOTO 60, 70, 80, 90,
50 GOTO 10: REM "NEM MEGENGEDETT
   BILLENTYÜ"
60 Y=Y1-1: GOTO 100
70 Y=Y1+1: GOTO 100
80 X=X1-1: GOTO 100
90 X=X1+1
100 LOCATE(X1, Y1): PRINT " ";
110 LOCATE(X, Y): PRINT "*" ; X1=X: Y1=Y
00 GOTO 10 REM "HA A HIBÁT AZ OKOZTA, HOGY
   MEGKISÉRELTUK ELHAGYNI A KÉPERNYŐT,
   A PROGRAM INDULJON ELŐLRŐL!"
205 IF(ERL=110) AND (ERN=5) THEN RESUME 1
210 RESUME
```

b) ON *kifejezés* GOSUB *sorszám* 1, *sorszám* 2 . . .*sorszám n*
Működése mint az ON. . . .GOTO utasításé, a különbség az, hogy a vezérlést a tétellista sorszámaival megjelölt szubrutinokra adja át.

Ha a kifejezés aktuális értékének egész része nulla vagy nagyobb mint *n*, a vezérlés a következő sorra adódik. Ha ez az érték negatív, hibajelzést kap a felhasználó.

c) ON ERROR GOTO *sorszám*
Az utasítás lehetőséget nyújthat a program futtatásakor fellépő hibák megvizsgálására, felülbírálására és menet közbeni korrigálására is.

A hibakereső rutinban a programozó megvizsgálhatja a hiba okát (lásd ERN); megtudhatja a hiba előfordulási helyét (lásd ERL) és a lehetséges hibaokok megfontolása után, vagy nélkül dönthet a visszatérés helyét illetően (lásd RESUME).

R E S U M E
Az utasítás feladata a fenti ON ERROR GOTO hatására végrehajtott "hibacsapdából" való visszatérés biztosítása. A következő lehetőségek állnak rendelkezésre:

–RESUME	visszatérés a hiba utáni utasításokra
–RESUME *sorszám*	folytatás a megadott sorszámtól
–RESUME NEXT	folytatás a következő programsortól

E R R O R *kód*
Az utasítás segítségével szimulálni lehet a kóddal megadott hibát, és így lehetőség van az ON ERROR GOTO-val megvalósított „hibacsapda" kipróbálására.

S T O P
Az utasítás hatására a program futása megszakad.
Erről az alábbi üzenet tájékoztatja a felhasználót:

BREAK IN *sorszám*

Az ilyen módon megszakított program adatait (válaszait) a PRINT utasítás segítségével ellenőrízni lehet, azaz hibakeresés céljára jól felhasználható. A hibagyanus helyek elé és mögé, addig fel nem használt sorszámokra írjunk STOP utasításokat és futtassuk le a programunkat!
A program felfüggesztett állapotában a változók illetve a program vizsgálatával (LIST; PRINT) döntsük el, hogy található-e hiba az eddigiekben! Amennyiben igen, úgy azt javítani kell, ami már egy másik program feladata.

Ha a program futását a STOP utasításig rendben találtuk, akkor a CONT paranccsal folytathatjuk a megszakított programot.

Ezen utasítások megfelelő helyen való alkalmazásával a programhibák kiküszöbölése tapasztalt programozó számára gyors és egyszerű. Nyomkövetéskor célszerű még a TRON és TROFF utasítások felhasználása is.

R E M *megjegyzések*

Az utasítással lehetőségünk van a gép által figyelmen kívül hagyandó, a program áttekintését elősegítő megjegyzések tételére.

Ezekkel a megjegyzésekkel ellátott program listáján könnyű eligazodni, mert a közölt információ megadhatja az egyes változók szerepét, a szubrutinok belépési feltételeit, kilépési adatait stb.

A számítógép a REM alapjel felismerése után, az adott sorban lévő összes további karaktert a következő programsor elejéig figyelmen kívül hagyja. Tehát olyan programsorban amely több, kettősponttal elválasztott utasításcsoportot foglal magába, a REM utasítás csak az utolsó lehet!

A REM utasítás használatára is példát mutat az ON... GOTO utasítás tárgyalásánál közölt mintapélda.

Egy elkészült, „belőtt" programból a futási idő és a tár terhelésének csökkentése érdekében, célszerű törölni minden REM utasítást.

R E T U R N

A szubrutin befejező utasítása, melynek hatására a vezérlés automatikusan a szubrutinhívást követő utasításra adódik át.

A RETURN utasítás önmagában feltétel nélküli, de az IF THEN RISE utasítások variációval kombinálva feltételessé alakítható.

A feltétel nélküli használatára mutat példát a GOSUB utasítás tárgyalásánál közölt mintapélda, míg a feltételesre az alábbi:

```
10 IF A>B THEN RETURN
20 A=A + A: GOTO 10
```


E R L

Az utasítás megadja annak a programsornak a sorszámát, amelyben a hiba felbukkant.

E R N

Az utasítás megadja az előforduló hiba kódját (lásd a HIBA táblázatot)

4.10 Vegyes utasítások

MEM	POKE
	CALL
PEEK	VARPTR

MEM
Az utasítás megadja a felhasználó rendelkezésére álló szabad memória nagyságát byte-ban.

Karakterláncok számára tegyük szabaddá a memória harmadát!

```
CLEAR MEM/3    (CR)
```

vagy

```
PRINT (MEM + FRE (A$) ) *2
```

tetszőleges aritmetikai kifejezések tagjaként állhat!

P E E K *(kifejezés)*
Az utasítás a kifejezés aktuális értékét előjeles egész típusú számmá alakítja, a kapott 16 bites eredményt memóriacímnek tekinti, és az így elérhető memóriarekesz tartalmát szolgáltatja.

```
PRINT PEEK (256*48)
```

a képernyő bal felső sarkának adatát közli.

Megjegyzés: az előjeles számábrázolás miatt a 64Kb-os címmező felső 32Kb-ját csak negatív számokkal lehet megcímezni.

P O K E *kifejezés, kifejezés*
Az utasítás az első kifejezésből az elözőleg leírt módon címet képez, a másodikból pedig adatot. Az adatnak a (0, 255) tartományba kell esnie.

A kiszámított címre az utasítás megpróbálja kivinni az adatot, függetlenül attól, hogy ott ténylegesen RAM tár van-e, vagy sem.
Lásd a PEEK-nél lévő megjegyzést is!

C A L L *decimális memóriacím*
Az utasítás a paraméterként megadott decimális memóriacímtől kezdődően a Z80 μP gépi kódjában rendelkezésre álló programra adja át a vezérlést.

A gépi kódú programozást csak a haladó programozóknak javasoljuk. Nem szabad beleírni az IPT–002 ún. munkaterületére, mélységben elrontani a vermet (stacket), a program illetve változók adatait megváltoztatni stb.

A CALL célja az, hogy adott funkciók BASIC-nél hatékonyabb gépi kódú programozásával gyorsítsa a feldolgozást.

A gépi kódú programból a Z80-as RET utasításával lehet visszatérni. Paraméter- vagy értékátadást csak a PEEK, POKE utasításokkal lehet megvalósítani.

CALL 20480 (hívás az 5000 H címre!)

Megjegyzés: a CALL utasítással meg lehet hívni a BASIC ROM-tárának különböző rutinjait is.

3-féle hangot generálhatunk a

CALL 545
CALL 552
CALL 559

utasítások segítségével.
Ezek kombinációival meglepő hangeffektusok is képezhetők:

```
FOR A=1 TO 7: CALL 545: CALL 552:CALL 559
NEXT
```

V A R P T R *(változó)*
Az utasítás segítségével a gépben tárolt változók címének decimális értékét kaphatja meg a felhasználó.

A változók értékén kívül természetesen a változó neve és típusa is tárolva van.
A különböző típusú változók esetén kapott válaszok jelentését az alábbi ábrák szemléltetik:

a) egész:

típus	név	LSB	MSB

b) egyszeres:

típus	név	LSB	. . . . .	MSB	EXP

c) dupla:

típus	név	LSB. .	. . . . .	. . . . .	. . . . .	MSB	EXP

d) szöveges:

típus	név	hossz	LSB cím	MSB cím

5. SZERKESZTŐ ÜZEMMÓD

Szerkesztéskor a megváltoztatni kívánt programsort az

EDIT *sorszám* (CR) utasítással lehet előkészíteni.

Az

EDIT (CR) utasítással pedig az utolsóként tárolt vagy megváltoztatott programsort lehet előkészíteni.

Az előkészített programsor sorszáma megjelenik a képernyőn.

```
EDIT 15    (CR)
15 ←
```

5.1 A kurzor kívánt pozicióra állítása

A kurzort jobbra a szóköz, balra a CNTL–H billentyűkkel lehet mozgatni. Ha az említett billentyűk előtt egy számot is megadunk, úgy ezek együttes hatására a kijelölt irányba a számnak megfelelő nagyságú elmozdulás lesz.

Végezzük el a következőket!

```
10 PRINT  A * A+B  (CR)
EDIT 10            (CR)
10 PRINT  A * A+B
10
10 PRI
```

Az L billentyű segítségével kilistázhatjuk a szerkeszteni kívánt programsort anélkül, hogy kilépnénk a szerkesztő üzemmódból.

Nyomjuk meg a 3-as és szóközbillentyűket. A képernyőn ezek nem jelennek meg!

További mozgatási lehetőség a keresés, vagyis a kurzor pillanatnyi helyétől mozogjon egy megadott karakter megadott előfordulásáig. Az első előfordulást itt sem kell külön jelezni, elegendő csupán az S billentyűt megnyomni.
Előző példánkat folytatva:
Nyomjuk meg a 2,S,A billentyűket a fentiek bemutatására.

```
10 PRINT  A *←
```

5.2 Karakter cserélése

A legegyszerűbb változtatás a programban lévő karakter, illetve karakterek kicserélése más karakterre illetve karakterekre. Ezt a szerkesztő alapprogramot a C billentyű előzetes

megnyomásával lehet aktivizálni. Itt is egy számmal lehet közölni a folyamatosan megváltoztatandó karakterek számát; az egy kiírása felesleges!
Nyomjuk meg a 2, C, C, – gombokat, végül fejezzük be a szerkesztést a CR billentyűvel, listázzuk ki a szerkesztett 10. sort!

```
LIST 10    (CR)
10 PRINT  A * C – B
```

5.3 Karakterek törlése

A következő egyszerű szerkesztési funkció a törlés, azaz a pillanatnyi kurzorpozíció és a megadott karakterszám által kijelölt karakterek törlése. Ezt a szerkesztő funkciót a D billentyűvel lehet aktivizálni. A karakterszámot előre kell megadni, az egy megadása itt is elhagyható.

A törlésre kijelölt karakterek a kijelzésben felkiáltó jelek közé kerülnek, ezzel is figyelmeztetve a felhasználót az ellenőrzésre.
Szerkesszük újra az előzőekben kapott 10. sort!

```
EDIT 10    (CR)
10 PRINT  A * C–B        L
10 ←                     1, S, –
10 PRINT  A * C          2 D
10 PRINT  A * C!–B!          (CR)
```

Ez a műveletsor a 10 PRINT A*C programsorra vezet.

5.4 Kilépés a szerkesztő üzemmódból

Előfordulhat, hogy billentyűtévesztés vagy konkrét tévedés következtében a programsor tartalma a szerkesztéskor távolodik a kitűzött céltól. Ennek időben való felismerésekor a következő lehetőségek állnak rendelkezésre:

- Csere esetén visszalépés és újabb csere,
- Kilépni a szerkesztő alprogramból, ha abban vagyunk (CNTL–Z); törölni az összes addigi változtatást és automatikusan újrakezdeni a szóban forgó programsor szerkesztését. Ezt az A (ABORT) billentyű megnyomásával érhetjük el.
- Kilépni a szerkesztő alprogramból, ha abban vagyunk (CNTL–Z), törölni az összes eddigi változtatást, majd kilépni a szerkesztő üzemmódból. Ezt a Q (QUIT) megnyomásával érhetjük el.

Ha a létrehozott új programsor megfelelő, akkor a CR gomb megnyomásával menthetjük el a memóriába. Ugyanezt a funkciót látja el az E (EXIT) alprogram is!

Kilépés a szerkesztő üzemmód alprogramjaiból:
A továbbiakban tárgyalandó alprogramokból CNTL–Z karakter segítségével lehet kilépni. A kilépés után tetszőleges szerkesztő funkciók ismételt használatára van lehetőség.

5.5 Karakterek beszúrása (inzertálás)

Előfordulhat, hogy az előzetesen tárolt programsorból karakterek hiányoznak. Ilyenkor mozgassuk a kurzort az eddig leírt módok segítségével a hibahelyre, majd az I billentyű megnyomásával lépjünk be a módosító (inzertáló) alprogramba.

Az alprogram innen kezdve folyamatosan kiegészíti a billentyűzetről érkező karakterekkel a sort, míg csak a CNTL–Z karakterrel ki nem lépünk az alprogramból, vagy a CR billentyű segítségével, a változások elmentésével ki nem lépünk a szerkesztő üzemmódból.

```
>10 PNT A     (CR)
>EDIT 10      (CR)
10 P←
```

Mozgassuk a kurzort a hibahelyre, majd nyomjuk meg az I billentyűt.

```
10 PRI←
```

Egészítsük ki a helytelen kulcszót. A nyillal jelzett helyen adott CR karakter hatására a sor a dőlt betűs karakterekkel kiegészítődik és így tárolódik.

```
10 PRINT A←
           ↑
```


Ha a nyillal jelzett helyen a CNTL–Z karakterrel léptünk volna ki az inzertáló alprogramból (nem üzemmódból!) úgy megmaradtunk volna a

10 PRI ← pozíción és a fentiekben leírt szerkesztő funkciókkal folytathatnánk a programsor feldolgozását.

5.6 Beszúrás a programsor végén

Gyakran fordul elő, hogy a módosítás célja nem egy menetközbeni kimaradás pótlása, hanem a programsor végéhez való toldás, kiegészítés.

Ilyenkor a szám-szóköz kombináció illetve más kurzor mozgató funkciók felhasználása helyett elegendő csupán az X billentyű megnyomása. Ennek hatására kurzor az utolsó karakter utáni pozícióra áll, míg a vezérlés az inzertáló alprogramnak adódik át.

```
>EDIT 10      (CR)
10←
10 PRINT A←

10 PRINT A*A
READY
>←
```

Nyomjuk meg az X gombot! Egészítsük ki a *, A karakterekkel majd zárjuk le a CR billentyűvel

5.7 Kurzortól jobbra lévő összes karakter törlése és beszúrás

Mozgassuk a kurzort az eddig ismertetett módok valamelyikével a szükséges pozícióra. A H billentyű megnyomásával a kurzortól jobbra elhelyezkedő összes karakter törlődik és a vezérlés az inzertáló alprogramra adódik. Ha azonnal kilépünk csupán kényelmes törlés történt, egyébként pedig az eddig elmagyarázott beszúrás kezdődhet meg.

5.8 Összefoglaló táblázat

A	változások törlése, szerkesztés előlről;
C	csere;
D	törlés;
E	változások elmentése és kilépés a szerkesztő üzemmódból (CR);
H	kurzortól jobbra törlés és beszúrás;
I	beszúrás;
L	a kiválasztott programsor listázása;
Q	a változtatások törlése, kilépés a szerkesztő üzemmódból;
S	megadott karakter adott számú előfordulásáig mozgatja a kurzort;

6. NÉHÁNY TANÁCS–GONDOLAT!

1. Semmi sem kelt nagyobb ellenszenvet egy tapasztalt programozóban, mint az, ha egy kezdő – aki éppen most fejezett be egy hat utasításból álló, a másodfokú egyenlet gyökeit kiszámító BASIC programot – beszél neki a programozás elméletéről vagy gyakorlatáról.
2. Egy igazi „hivatásos programozó" úgy írja meg a programot, hogy módosítása ne jelentsen problémát.
3. A „hivatásos programozó" minden programját beírja a könyvtárába. Általában nem tudunk egyértelműen beszélni, ezért olyan kínzóan nehéz a dokumentálás.
4. A „hivatásos programozó" egyetlen problémát sem vesz túl komolyan.
5. Minden program esetében meg van a ráfordításnak és töprengésnek egy értelmes határa.
6. Alkalmazkodni kell a megoldandó problémához!
7. A programokat meghatározott időtartamra kell tervezni!
8. A feladat megoldása a rendelkezésre álló idő arányában megnyúlik.
9. Egy program megírásának helyes folyamata: probléma meghatározása; elemzése; folyamatábrák készítése; és végül a program dokumentálása.
10. A program tervezéséhez olyan szem kell, amelyik az egész erdőt látja, míg a hibakereséshez minden fát, ágat, levelet látni kell!
11. A programozók mindig azt mondják, hogy a gépben van a hiba. A rossz programozó egyik ismertető jele az, hogy tétlenül üldögél, ha a gép ledöglött.
12. Egy program kipróbálásakor a következő úton kell haladni
    - a hibák létezésének a megállapítása;
    - a létező hibák helyeinek meghatározása;
    - a megtalált hibák kijavítása.

    Az elsőhöz olyan észjárás kell amelyik automatikusan a legrosszabbat tételezi fel.
    A másodikhoz egy anyós konok kitartása és egy hörcsög gyűjtési ösztöne szükséges.
    Végül a hibák kijavításához egyfajta szintetizáló

észjárás szükséges, olyan programozóé aki hajlandó és képes az eredeti programíró stílusát elfogadva „foltozni".

13. Nem tudjuk mit kell csinálni addig, amíg nem a program megírásának gondolatával vizsgáljuk a problémát.
14. Programozói katasztrófák esetében általában a helyzetet senki nem tekinti annak.
15. Bármelyik működő program jobb a nem működőnél.
16. Valószínűleg még sosem írtak tökéletes programot!
17. Ne akarjunk mindent magunk kitalálni. Egy megfelelő szakkönyvtárban szinte biztosan megtalálhatjuk a kitűzött feladat rész-vagy akár teljes megoldásának útmutatását.

7. BASIC PROGRAMKÖNYVTÁR

A programkönyvtár összesen 50 db játék (J), matematikai (M), statisztikai (S), üzleti (U), elektrotechnikai (E), és számítástechnikai (C) témakörbe sorolható BASIC programot tartalmaz.

Az 1 db C–60-as kazettán a következő programok vannak, az alábbi sorrendben, minden programot **kétszer** felvéve:

TARTALOM

MONSZKOP	P – 1	– J
REKLAM–1	P – 2	– J
REKLAM–2	P – 3	– J
GRAFIKA	P – 4	– J
LABIRINT	P – 5	– J
FOGOCSKA	P – 6	– J
APOLLO	P – 7	– J
SZOPOKER	P – 8	– J
FOGD MEG	P – 9	– J
TENISZ	P – 10	– J
AUTO	P – 11	– J
LINALG E	P – 12	– M
2 FOKU E	P – 13	– M
3 FOKU E	P – 14	– M
4 FOKU E	P – 15	– M
LOAD MAT	P – 16	– M
INV MAT	P – 17	– M
MPY MAT	P – 18	– M
KVAD DET	P – 19	– M
SIMPSON	P – 20	– M
INT – INT	P – 21	– M
NEWTON	P – 22	– M
BISECT	P – 23	– M
R. KUTTA	P – 24	– M
POLIGON	P – 25	– M
KOORKONV	P – 26	– M
LNKO	P – 27	– M
PRIMEK	P – 28	– M
FIBONAC.	P – 29	– M
ROM. NUM.	P – 30	– M
STAT – 1	P – 31	– S
LIN. REG.	P – 32	– S

EXP. REG.	P – 33	– S
GAUSS	P – 34	– S
POISSON	P – 35	– S
BINOMIAL	P – 36	– S
NAPTAR	P – 37	– U
DATUMOK1	P – 38	– U
DATUMOK2	P – 39	– U
HITEL	P – 40	– U
AMORT	P – 41	– U
NEVSOR	P – 42	– U
HALOTERV	P – 43	– U
RLC – PARH	P – 44	– E
RLC – SOR	P – 45	– E
DELTA – CS	P – 46	– E
CS – DELTA	P – 47	– E
KAPAC.	P – 48	– E
HEX CONV	P – 49	– C
HEXA MON	P – 50	– C

P – 1 – J
MONSZKOP CLOAD "MONSZKOP"

A betöltött program a RUN paranccsal indítható.

A program az IPT–002-höz kapcsolt televíziós készülék beállítását könnyíti meg azáltal, hogy egy monoszkópábrát generál.

Az ábrát hol fehér alapon feketén, hol pedig ennek inverzeként jelzi ki. A TV-t úgy kell beállítani, hogy az ábrán a sorok sose csússzanak el!

A monoszkópábra vízszintes és függőleges párhuzamos vonalai segítenek a linearitási hibák felderítésében is.

A képernyő széle fekete-fehér kockákkal van szegélyezve, mellyel a képméret állítható be.

P – 2 – J
REKLAM–1 CLOAD "REKLAM–1"

Általános, látványos tájékoztató az IPT–002 személyi számítógéphez.
A program a RUN paranccsal indítható.

P – 3 – J
REKLAM–2 CLOAD "REKLAM–2"

A betöltött programot a RUN paranccsal lehet indítani.

A program csak látványos reklámfeliratot ad, de listája iskolapéldája lehet a DATA, READ, RESTORE utasításoknak, azért tanulmányozása javasolt.
A program, futásközben "*" karakterek segítségével az

IPT–002
HOME COMPUTER

feliratot készíti a képernyőre!

P – 4 – J
GRAFIKA CLOAD "GRAFIKA"

A betöltött programot a RUN paranccsal lehet indítani.
A program az IPT–002 kvázigrafikus karakterkészletét rajzolja fel a képernyőre.

Az	1. sor kódjai:	: 0	15
	2. sor kódjai	: 16	31
	3. sor kódjai	: 96	111
	4. sor kódjai	: 112	127
	5. sor kódjai	: 128	143
	6. sor kódjai	: 144	159
	7. sor kódjai	: 224	239
	8. sor kódjai	: 240	255 - ig terjednek.

P – 5 – J
LABIRINT CLOAD "LABIRINT"

A betöltött programot a RUN paranccsal kell elindítani. A program egy labirintust generál, amelyen egy 🚶 karaktert lehet mozgatni 4 irányban,

F fel
L le
J jobbra
B balra nyomógombokkal.

A játék célja az, hogy a labirintus bejáratától a kijáratig a játékos minél hamarább tegye meg az utat. Az a játékos, aki a mozgatás során nekimegy a labirintus falának, az hibapontot kap és az ideje is egy darabig nő anélkül, hogy mozogni tudna.

A játék végén a gép felteszi a kérdést „KERI UJRA EZT? " I (igen) válasszal ugyanezt a pályát kaphatjuk az óra újraindítása mellett! Viszont az N (nem) billentyű megnyomása után egy másik pálya kérhető a program újraindításával (**).

A program, a karakter vándorlása alatt már elfogadja a játékos következő iránykijelölését és előjegyzi azt. Ezért ne doboljunk türelmetlenül a billentyűkön!
Jó szórakozást!

P – 6 – J
FOGOCSKA CLOAD "FOGOCSKA"

A betöltött programot a RUN paranccsal lehet indítani.

Egy bekerített területen van a fogó és akit kergetnek . A fogó célja az, hogy minél kevesebb lépésben utolérje a másikat, azaz avval azonos kockára lépjen.
Erre a célra 4 billentyűvel

F fel
L le
J jobbra
B balra irányokba léphet.

A fogó előnye az, hogy ha kétszer egymás után ugyanabba az irányba lép, sebessége a duplájára nő. Akárcsak a futó emberé, akinek nem kell jobbra-balra kerülgetve kilépni. Ám ez az előny hátrány is lehet, mert túl lehet lépni a kergetett karakteren.

A gyakorlatlan játékosoknak azt ajánljuk, próbálják meg a karaktert valamelyik sarok felé terelni.

A program, a karakter villogással kezdődő vándorlása alatt már elfogadja a fogó következő lépését és előjegyzi. Ezért ne doboljunk türelmetlenül a billentyűkön!
Jó szórakozást!

P – 7 – J
APOLLO CLOAD "APOLLO"

A programot a RUN paranccsal lehet indítani.

Először tájékoztatást kap a felhasználó a körülményekről és a lehetőségekről:

Egy 16 t-ás űrhajó 8250 kg tömegű üzemanyaggal van ellátva. Az üzemanyag égetése 10 s-ként állítható. Üzemanyagégetés nélkül szabadesés következik be, míg a maximális fékezés 100 kg/s-al állítható be.

A cél az, hogy 193 km magasságból lehetőleg nulla sebességgel érjen holdat az űrhajó. Azonos leszállási sebességek esetén az a játékos nyer, aki kevesebb üzemanyagot használt fel.
A gép minden repült 10 s elteltével közli az

IDŐ, TÁVOLSÁG, SEBESSÉG, ÜZEMANYAG

adatokat és kéri a következő 10 s alatt elégetendő üzemanyag mennyiséget

SEC	KM	KM/H	UA	FELH.
0	193.00	5792.0	8250	? 100
10	176.99	5786.9	8150	? 10
20	160.93	5781.1	8050	? ...

Jó szórakozást!

P – 8 – J
SZOPOKER CLOAD "SZOPOKER"

A betöltött programot a RUN paranccsal lehet indítani.

A játékot ketten játszhatják. Először meg kell adni, hogy milyen hosszúságú szavakat kell kitalálni.

SZÓHOSSZ (5–15) = ?

majd közölni kell a géppel a két játékos szellemi összecsapásainak a számát:

FORDULOK SZAMA = ?

Ezek után a gép automatikusan felkéri a kitalálandó szó elrejtésére az éppen soron következő játékost a

1. JATEKOS REJT!
KITALALANDO: ?

üzenettel, visszautasítva az előbb megadott szóhossztól különböző számú karakterből álló szavak elrejtését!

A tárolás után a társnak ki kell találnia az elrejtett szót, mégpedig a sorszámozott kérdésekre adott próbálkozásokkal. A szóhossznál rövidebb karaktersorozat beadásakor a gép a hiányzó karaktereket szóközöknek tekinti, míg ellenkező esetben a CR gomb megnyomása után a plusz, felesleges karaktereket le fogja törölni.

A próbálkozás során beírt, és a kitalálandó szóban azonos pozíciókon szereplő azonos karakterek számát a gép ki fogja jelezni. Az így kapott információk alapján kell minél kevesebb lépésben kitalálni az elrejtett szót.

Az éppen játszó fél teljesítményét a gép a

KIVALO
JO
MEGFELELO jelzőkkel, illetve a
NA VEGRE felkiáltással minősíti.

Innen bármelyik billentyű megnyomásával tovább folytatódhat a játék, míg csak a kívánt mennyiségű forduló le nem zajlik.
Ekkor a végeredmény kiíródik:

	1. JATEKOS	2. JATEKOS
1	4	2
.	.	.
.	.	.
.	.	.
.	.	.
A fordulók száma	Az adott fordulóban az elrejtett szót hányadik próbálkozásra találta ki	

Jó szórakozást!

P – 9 – J
FOGD MEG

CLOAD "FOGD MEG"

A betöltött programot a RUN paranccsal lehet indítani.

Ekkor a képernyő X, Y pozícióin * karakterek jelennek meg, illetve tűnnek el véletlenszerűen.

A feladat az, hogy az utoljára megjelenő * helyét dacára a gyors és folytonos változásnak meg tudjuk jegyezni. Ha úgy képzeljük, hogy sikerült jól elkapni és megjegyezni a fent említett pozíciót, gyorsan nyomjuk meg az M billentyűt (most!)! Ekkor kezdődhet a játék, azaz a megfigyelőképességünk egyfajta tesztelése.

A bal alsó sarokból elindulva nyomógombok segítségével a következő irányokba léphetünk:

F fel
L le
J jobbra
B balra

Minden egyes, már használt pozíción meg fog jelenni az azon való áthaladások száma. Ha ez a szám meghaladja a 9-et, akkor

EZ MAR TOBB A SOKNAL!

üzenet, lesz kiírva, és az ügyetlen játékos ismét próbálkozhat. A képernyő mezőjéről lelépni a fenti nyomógombokkal

nem lehet. Ilyen irányú próbálkozásnál villogó * jelenik meg a képernyőn.
A célbaéréskor az igénybevett lépésszám és a képernyőt elhagyni szándékozó kitörések száma is ki lesz írva.
Jó szórakozást!

P – 10 – J
TENISZ CLOAD "TENISZ"

A betöltött program a RUN paranccsal futtatható.

A gép különböző Y irányú és sebességű szervákat generál amit a játékosnak a két ▌ jellel szimbólizált ütővel kell megpróbálnia visszaütni, erre az ütő F fel, illetve L le irányú mozgatásával kell törekedni.

Ha a *-al jelzett labda eléri a képernyő jobb oldalát, úgy az a gépnek jelent pontot, míg a bal oldalra visszaütött pedig a játékosnak.

A játék pillanatnyi állása a képernyő első sorában van mindig kijelezve.

Érdekesebbé teszi a játékot az, hogy a képernyő alja és teteje felé tartó labdák onnan visszapattannak. Ezek a vonalak nincsenek bejelölve, így a játékos a labda útjának megítélésekor nehezebb helyzetben van.
Jó szórakozást!

P – 11 – J
AUTO CLOAD "AUTO"

A betöltött program a RUN paranccsal indítható.

A program egy véletlenszerűen kanyargó autóversenypályát rajzol fel a képernyőre, melyen egy karakterrel szimbólizált versenyautót kell végig kormányozni.

Az A billentyűvel balra, a + ; billentyűvel pedig jobbra lehet fordulni.

Az a cél, hogy minél hosszabb utat tegyen meg a játékos karambol nélkül. Ha a játékos manőverezni kezd: jobbra-balra kanyarodik, sebessége csökken, míg egyenesen haladva gyorsul.

Ha több játékos van, az nyer, aki nagyobb átlagsebességgel vezette végig a 100 km-es pályát.

A játék teljes tartama alatt folyamatosan ki van jelezve a megtett út, a játék befejeztével pedig az átlagsebesség is. Karambol esetén a játék véget ér.
Jó szórakozást!

P – 12 – M
LINALG E CLOAD "LINALG E"

Lineáris algebrai egyenletrendszer megoldása Gauss--módszerrel.

A programot a RUN paranccsal lehet elindítani.*

```
FOKSZÁM   = ? 3
A (1 1)   = ? 1
A (1 2)   = ? 1
A (1 3)   = ? 1
A (1 4)   = ? 6
```

Adjuk meg a fokszámot!
A gép kérdéseire, az egyes ismeretlenek együtthatóit kell megadni.
(fokszám+1) elemként az illető egyenlet eredményét írjuk!

```
X1   +  X2   +  X3   =  6
2X1  +  3X2  –  X3   =  5
–X1  –  X2   +  3X3  =  6
```

Az utolsó bevitelt követően automatikusan indul a feldolgozás.

```
X 1 = 1
X 2 = 2
X 3 = 3
```

A program válasza:
az egyenletrendszer megoldása

Az eredmények az A tömb A(I, N + 1) I = 1,.N indexű elemeiként vannak tárolva.

P – 13 – M
2 FOKU E CLOAD "2 FOKU E"

A betöltött program a RUN paranccsal indítható.

A program az $AX^2 + BX + C = 0$ alakú egyenlet X1 + J1, X2 + J2 alakú gyökeit határozza meg a beadott együtthatók ismeretében.

```
RUN
MASODFOKU EGYENLET
A B C?   1, 1, 1,     (CR)
X↑2 * (1) + X * (1) + 1 = 0
X 1 = –.5   J 1 =  .866026
X 2 = –.5   J 2 = –.866026
READY
```

Az eredmények a kijelzett változókban is rendelkezésre állnak!

*Az adatok, amelyeket a felhasználók adnak meg, dőlt betűvel vannak szedve.

P – 14 – M
3 FOKU E CLOAD "3 FOKU E"

A betöltött program a RUN paranccsal indítható.
A program az $AX^3 + BX^2 + CX + D = 0$ egyenlet
X1 + J1
X2 + J2
X3 + J3 alakú gyökeit határozza meg a beadott együtthatók ismeretében, a Cardan-módszerrel.

RUN
HARMADFOKU EGY. MEGOLDO PROGRAM
A, B, C, D ? *1, 2, 3, 4* (CR)

VALOS	KEPZETES
X1 = –1.65063	J1 = 0
X2 = –.174685	J2 = 1.54687
X3 = –.174685	J3 = – 1.54687

Az eredmények a kijelzett változókban is rendelkezésre állnak!

P – 15 – M
4 FOKU E CLOAD "4 FOKU E"

A betöltött programot a RUN paranccsal lehet indítani.
A program az $AX^4 + BX^3 + CX^2 + DX + E = 0$ egyenlet.
X1 + J1
X2 + J2
X3 + J3
X4 + J4 alakú gyökeit határozza meg a beadott együtthatók ismeretében a Ferrari-módszerrel.

RUN
NEGYEDFOKU EGYENLET MEGOLDO
A, B, C, D, E ? *1, –1, –3, 1, 2*

X1 = 2.00	J1 = 0.00
X2 = 1.00	J2 = 0.00
X3 = –1.00	J3 = 0.00
X4 = –1.00	J4 = 0.00

Az eredmények a kijelzett változókban is rendelkezésre állnak!

P – 16 – M
LOAD MAT CLOAD "LOAD MAT"

A betöltött program a RUN paranccsal futtatható.

A program az A mátrix dimenzionálására és az egyes mátrixelemek billentyűzetről való beadására készült. A soron következő mátrixelem indexeit a bevitel előtt minden egyes esetben kijelzik. Minden új mátrixsorra való rátérésnél hangjelzést ad a gép. Az utolsó mátrixelem bevitele után automatikusan kiíródik a
KESZ ! üzenet és a program futása véget ér.

```
> RUN
MATRIX  BEOLVASASA
XMAX = ?  2
YMAX = ?  2
MATRIX  ELEM (1, 1) = ?  1
KESZ
```

A képernyő a program futása alatt nem változik!
A beolvasott mátrixértékeket át lehet írni egy másik mátrixba! Ehhez két egymásba skatulyázott FOR ciklus szükséges:

```
FOR  J    =  1  TO  YM
FOR  I    =  1  TO  XM
B (J,I)   =  A (J, I)
NEXT I, J
```

P – 17 – M
INV MAT CLOAD "INV MAT"

A programot a RUN paranccsal lehet indítani.
Az A négyzetes mátrix inverzének kiszámítása Gauss-Jordan módszerrel.
Az eredmény A^{-1} mátrix az eredeti A mátrix helyén keletkezik.
A bemenő adatokat a 10. programsor DATA utasításával lehet közölni. A legelső adatot a mátrix dimenziójának értelmezi a program.

A program egy mintapéldát is magába foglal, amelynek alapján a következő számítást végzi el:

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

P – 18 – M
MPY MAT CLOAD "MPY MAT"

A beolvasott program a RUN paranccsal indítható.

A program az $n \times m$ dimenziójú **A** mátrixot megszorozza az $m \times r$ dimenziójú **E** mátrixal és az eredményt az $n \times r$ dimenziójú **C** mátrixban helyezi el.

Az **A** és **B** mátrixok beolvasását a 20. és 30. programsorok DATA utasításai után álló adatokból végezzük. Az eddig dimenzióként tárgyalt *n, r, m* adatokat a 10. sor DATA utasításaiból vesszük.

A beolvasás a program 60. sorában ér véget.

A tényleges mátrixszorzás a 100–160 sorokban zajlik. A kapott eredményt a program 400–410 sorai írják ki.

A kazettára felvett mintaprogram a következő mintapéldát foglalja magába:

$$\begin{bmatrix} 4 & 0 & -1 \\ -3 & 0 & 7 \\ -9 & 2 & 5 \\ 5 & -1 & 3 \end{bmatrix} * \begin{bmatrix} -1 & 5 \\ -6 & -6 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} -5 & 16 \\ -8 & -5 \\ 2 & -37 \\ 4 & 43 \end{bmatrix}$$

A program az eredmény nyomtatásakor az elemeket külön sorban adja meg, mögéjük írva indexeiket is.

P – 19 – M
KVAD DET CLOAD "KVAD DET"

Négyzetes mátrix determinánsának kiszámítása Gauss-eliminációs módszerrel!

A négyzetes mátrix a 10. programsor DATA utasításával adható meg. Az első adat a mátrix dimenzióját adja.

A beírt adatoknak az alábbi művelet felel meg:

$$\det \begin{vmatrix} 4 & 7 & 1 & 8 \\ 5 & -1 & 2 & -4 \\ 3 & 12 & -5 & 6 \\ 1 & 4 & 7 & 2 \end{vmatrix} = -3276$$

Az eredményt a program a képernyőn közli.

DET = –3276

Ez az érték a D jelű változóban is rendelkezésre áll.

P – 20 – M
SIMPSON CLOAD"SIMPSON"

A betöltött programot a RUN paranccsal lehet indítani.

A program a Riemann-integrál közelítő értékét határozza meg Simpson-módszerrel. Az integrálandó függvényt a 1000. programsortól kezdődően kell beírni szubrutinként.

A kazettára felvett program az alábbi mintapéldát is tartalmazza:

```
1000  F = X*X : RETURN
```

futtatás

```
RUN
KEZDŐÉRTÉK  =  ?  0
VÉGÉRTÉK    =  ?  1
PONTSZÁM    =  2
INTEGRÁL    =  .333333
1000  F = X :  RETURN

READY
>←
```

Az X irányú felbontás finomsága, amit meredekebb vagy gyorsan változó függvényeknél növelni kell!

Az integrál értéke az S jelű változóban is rendelkezésre áll.

P – 21 – M
INT–INT CLOAD "INT–INT"

A betöltött programot a RUN paranccsal lehet indítani.

A program az 1000. programsortól kezdődően szubrutinként beírt f = (X,Y) kétváltozós függvény Riemann-integrálját számolja ki a kétváltozós Simpson-formulával.

$$S = \begin{bmatrix} Y1 & X1 \\ YK & XK \end{bmatrix} f(X, Y)\, d\,X\, dY$$

A kazettára felvett program a következő mintapéldát is tartalmazza:

```
1000  F = X*X+4*Y :  RETURN
```

```
RUN
XK=?  -2        integrálás X határai
XV=?   2
YK=?   1        integrálás Y határai
YV=?   3
FX=?   2        X szerinti felbontás finomsága
FY=?   2        Y szerinti felbontás finomsága
KETTOS  INTEGRAL = 74.66667
```

```
1000  F = X*X+4*Y :  RETURN
READY
>←
```

A kettős integrál értéke az S jelű változóban is rendelkezésre áll!

P – 22 – M
NEWTON CLOAD"NEWTON"

A betöltött program a RUN paranccsal indítható.
Az f(X)=0 alakú egyenlet gyökhelyének meghatározása Newton-Raphson módszerrel. Meg kell adni az egyenlet egy közelítő megoldását és egy lépésközt, ami a megoldandó egyenlet meredekségének növekedésével finomodik.

A megoldandó egyenletet szubrutinként az 1000. programsortól kezdődően kell beírni.

A kazettára felvett program a következő mintapéldát is tartalmazza

```
1000  F = X ↑ X – 10 :  RETURN
```

Vegyük észre, hogy ezt az egyenletet elemi módszerekkel nem lehet megoldani! Szavakban: melyik az a szám, melyet önmagára hatványozva 10-t ad?

```
RUN
KÖZELITÉS  = ?     5
LÉPÉSKÖZ   = ?   .01
GYÖK       =   2.50 618
1000       =  F = X ↑ X – 10 :  RETURN
READY
```


A gyök értéke a X0 jelű változóban is rendelkezésre áll, és el lehet végezni a megoldás próbáját:

```
>PRINT  X0 ↑X0
 10
 READY
 >←
```

P – 23 – M
BISECT CLOAD "BISECT"

A betöltött program a RUN paranccsal indítható.

Függvény zérushelyének megállapítása előre megadott közelítő tartományban a felező módszer segítségével. A

kérdéses függvényt az 1000. programsortól kezdődően kell beírni szubrutinként, zérusra redukált alakban (f(X) = 0).
A kazettára felvett mintaprogram az $Y = X^2 - 2$ függvényt tartalmazza.
Indítsuk el a programot a RUN paranccsal. Meg kell adni az intervallumot, hogy megkapjuk a zérushelyet.

```
A = ? 1
B = ? 2
X = 1.41421
1000 F = X*X - 2: RETURN
```

Az f(A)=0 f(B)=0 feltételeknek ki kell elégülni, egyébként a program új A;B értékeket kér!

Az A, B, FA, FB, X értékek az azonos jelű változókban is rendelkezésre állnak.

P – 24 – M
R. KUTTA

CLOAD "R. KUTTA"

A betöltött programot a RUN paranccsal lehet indítani.
A program az 1000. programsortól kezdődően beírt derivált függvény integrálását végzi el a megadott X0, Y0 peremfeltételek ismeretében, megadandó lépésközzel és megadandó számú lépésben a Runge-Kutta módszerrel.
A kazettára felvett program a következő mintapéldát is tartalmazza:

```
1000  F = 2*X :  RETURN
```

ami a jól ismert $Y = X^2$ függvény deriváltja, tehát könnyen ellenőrizhető.

```
RUN
XK = ?   0
YK = ?   0
LK = ?   1
DB =    10

0     0.0000E + 00    0.0000E + 00
1     0.1000E + 01    0.1000E + 01
2     0.2000E + 01    0.4000E + 01
3     0.3000E + 01    0.90000 + 01
4     0.4000E + 01    0.16000 + 02
5     0.5000E + 01    0.25000 + 02
6     0.6000E + 01    0.36000 + 02
7     0.7000E + 01    0.49000 + 02
```

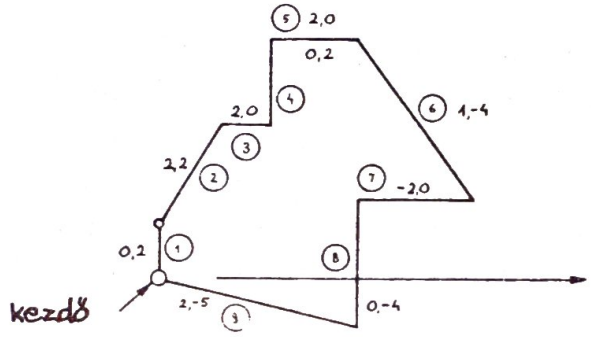
8	0.8000E + 01	0.64000 + 02
9	0.9000E + 01	0.81000 + 02
10	0.1000E + 02	0.10000 + 03

```
1000  F = 2*X :  RETURN
READY
>←
```

P – 25 – M

POLIGON CLOAD"POLIGON"

A program segítségével egy a koordináta irányokba való elmozdulásokkal megadott és így körüljárt poligon területe, súlypontja és súlya, az előre megadott súly/terület mérőszám alapján kalkulálható!
Amint a körüljárás során újra a kezdőponba érkezünk, az eredmény automatikusan kiíródik.



3. ábra.

A programot a RUN paranccsal kell elindítani.

```
KEZDO KOORDINATAK
X 0 = ?  0
Y 0 = ?  0
SULY/TER. (KP/M↑2)?  2

000  000  DX = ?  0
          DY = ?  2
000  200  DX = ?  2
          DY = ?  2
200  400  DX = ?  2
          DY = ?  0
400  400  DX = ?  0
          DY = ?  2
400  600  DX = ?  2
          DY = ?  0
600  600  DX = ?  1
          DY = ?  -4
700  200  DX = ?  -2
          DY = ?  0
```

Adjuk meg a kezdő koordinátát és a mérőszámot!

```
500  200  DX = ?  0
          DY = ? -4
500 -200  DX = ? -5
          DY = ?  2
TERULET =   31        M ↑2
SULY    =   62        KP
SULYPONT KOORDINÁTÁI:     XG = 3.51615 YG = 2.02151

READY
```

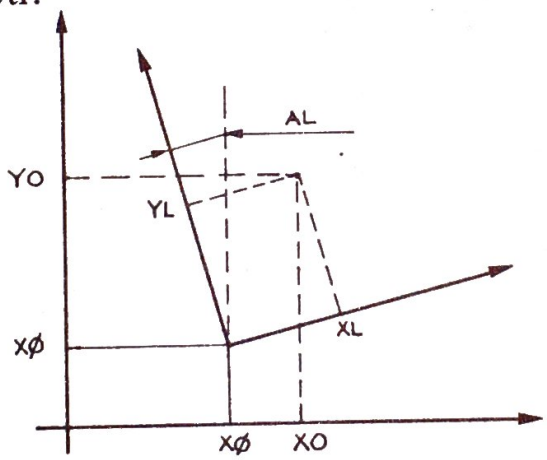
Az eredmények a T, S, XG, YG jelű változókban is rendelkezésre állnak.

P – 26 – M
KOORKONV CLOAD "KOORKONV"

A program a helyi koordinátákat az eredeti koordinátákra képes konvertálni ha a RUN vagy a RUN 100 paranccsal indítják. Ilyenkor az

$$XO = X0 + XL*COS(AL) - YL*SIN(AL)$$
$$YO = Y0 + XL*SIN(AL) + YL*COS(AL)$$

egyenleteket használja a program. A változók jelentését az ábra szemlélteti:



4. ábra.

Az inverz irányú, eredetiből helyi koordinátákra való átszámításnál az alábbi

$$XL = (XO-X0)*COS(AL) + (YO-Y0)*SIN(AL)$$
$$YL = (YO-Y0)*COS(AL) - (XO-X0)*SIN(AL)$$

képleteket alkalmazzuk, ha a a RUN 200 paranccsal elindítjuk a programot.
A program bemenő adatai az ábrán jelölt:

X0
Y0
AL adatok, valamint a konvertálandó koordinátapár.

```
RUN
X0 = ?   12
Y0 = ?   15
AL = ?   19
                LOKALBOL ORIGINALBA;
XL = ?   2
YL = ?   3
XO = 12.9143     YO = 18.4877
READY
>←
```

P – 27 – M
LNKO CLOAD "LNKO"

A betöltött program a RUN paranccsal indítható.

A program két szám legnagyobb közös osztóját határozza meg az euklideszi algoritmus alapján.
A két számot M,N ? alakban kéri a gép. A két számot bevitelkor vesszővel kell elválasztani.

Az eredmény LNKO = *X* alakban a képernyőre kerül. Ezt az eredményt egyébként még az M nevű változóban is megtalálhatja a programozó.

Ha a felhasználó valamelyik számot nulla értékkel szerepeltetné, akkor az eredmény végtelen lenne. Ezt elkerülendő ilyenkor hibaüzenetet kap a felhasználó és a program a két szám ismételt bevitelével folytatódik.

```
>RUN
M,N  ?     10, 15     (CR)
LNKO = 5
READY
>←
```


P – 28 – M
PRIMEK CLOAD "PRIMEK"

A betöltött program a RUN paranccsal indítható.

A program előállítja az 1 és 1000 közötti 167 prímszámot és kiírja a képernyőre. A prímszámok generálása a szita módszerrel történik.

P – 29 – M
FIBONAC. CLOAD "FIBONAC."

A betöltött programot a RUN paranccsal lehet indítani.

A program az ábrázolható tartományon belül egyébként tetszőleges *n*. Fibonnacci szám megadására képes.

```
RUN
FIBONNACCI SZAMOK!
HANYADIKAT KERI ? 10
A KIVANT SZAM   = 55
```

A program folyamatosan kínálja szolgáltatását, míg a futását a CNTL–A-val vagy a BREAK gombbal meg nem szakítják.

P – 30 – M
ROM.NUM. CLOAD "ROM. NUM."

A beolvasott programot a RUN paranccsal lehet indítani.

A program a billentyűzetről beadott római számot decimális számmá konvertálja.

A program felismeri a hibás alakú római számokat és a bevitelt megismételteti. Ahol fel lehet ismerni azt, hogy az illető szám biztosan befejeződik, ott automatikusan – a CR billentyű megnyomása nélkül végrehajtja a konvertálást.
A legnagyobb beadható római szám: MMMDCDXCIX
aminek a decimális megfelelője:3999

P – 31 – S
STAT – I CLOAD "STAT–I"

A betöltött program a RUN paranccsal indítható.

A program számolja a beadott adatok számát, kiszámolja azok totális összegét, átlagát és szórását is az alábbi definíciónak megfelelően:

$$\sigma = \frac{\sqrt{\Sigma(X_i - \overline{X})^2}}{n-1}$$


```
RUN
ADAT     = ?  1,1
```

Adat bevitele

```
DARAB    = 1.00
ÖSSZEG   = 1.10
ÁTLAG    = 1.10
SZÓRÁS   =
```

Kijelzett eredmények

A szórás csak a második beadott számtól kezdődően jelződik ki.

A kijelzett eredmények az N, T, A, S változókban is rendelkezésre állnak!

P – 32 – S
LIN. REG. CLOAD "LIN. REG."

A betöltött program a RUN paranccsal indítható!

A program a beadott X, Y párok alapján meghatározza a ponthalmazhoz legjobban illeszkedő Y = A*X + B alakú egyenes A és B paramétereit a legkisebb négyzetes eltérések módszerével.

Kijelzi az $\overline{X}$, $\overline{Y}$ átlagokat, a VX; VY varianciákat, és a pillanatnyi pontpárok N számát is.

Az Y tengellyel párhuzamosan adódó közelítő egyenesnél a program a "FUGGOLEGES" üzenetet írja ki:

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{k} X_i \qquad \overline{Y} = \frac{1}{N} \sum_{i=1}^{k} Y_i$$

$$VX = \frac{1}{N-1} \left(\sum_{i=1}^{k} X_i^2 - N*\left(\sum_{i=1}^{k} X_i \right)^2 \right)$$

$$VY = \frac{1}{N-1} \left(\sum_{i=1}^{k} Y_i^2 - N*\left(\sum_{i=1}^{k} Y_i \right)^2 \right)$$

$$A = \frac{\sum_{i=1}^{k} X_i^2 - N*\left(\sum_{i=1}^{k} X_i\right)^2}{\sum_{i=1}^{k} X_i Y_i - N*\sum_{i=1}^{k} X_i * \sum_{i=1}^{k} Y_i}$$

$$B = \frac{1}{N} \sum_{i=1}^{k} Y_i - \frac{A}{N} * \sum_{i=1}^{k} X_i$$

Figyelem: helyes regressziós együtthatók kialakításához legalább 3 adatpárra van szükség!

LINEARIS REGRESSZIO

X, Y ?
Ernyőkép az *(1,1)*.*(2,2)* *(3,3)* *(4,4)* *(5,5)* *(6,6)* *(7,7)* *(8,8)* pontpárok beadása után

ATLAG	X	=	4.50	Y = 4.50
VARIANCIA	VX	=	6.00	VY = 6.00
PONTOK SZÁMA		=	8.00	

Y = A*X + B
A = 1 B = 0

A kijelzett adatok az X, Y, VX, VY, N, A, B változókban is megtalálhatók.

A közelítő egyenes Y tengellyel párhuzamos voltát a K1 változó nulla értéke jelzi.

P – 33 – S
EXP. REG. CLOAD "EXP. REG"

A betöltött program a RUN paranccsal indítható.

A program a beadott X; Y pontpárok alapján meghatározza a ponthalmazhoz legjobban illeszkedő Y = A*B↑X alakú exponenciális függvény A és B paramétereit a legkisebb négyzetes eltérések módszerével.

Kijelzi az $\bar{X}$, $\bar{Y}$, átlagokat a VX, VY varianciákat, és a pillanatnyi pontpárok N számát is.

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{k} X_i \qquad \bar{Y} = \frac{1}{N}\ln(Y_i)$$

$$VX = \frac{1}{N-1}\left(\sum_{i=1}^{k} X_i^2 - N*\left(\sum_{i=1}^{k} X_i\right)^2\right.$$

$$VY = \frac{1}{N-1}\left(\sum_{i=1}^{k}(\ln(Y_i))^2 - N*\left(\sum_{i=1}^{k}\ln(Y_i)\right)^2\right)$$

$$B = \exp\frac{\sum_{i=1}^{k}(X_i\ln(Y_i)) - \sum_{i=1}^{k} X_i * \sum_{i=1}^{k}\ln(Y_i)}{\sum_{i=1}^{k} X_i^2 - \left(\sum_{i=1}^{k} X_i\right)^2} = \exp(B^1)$$

$$A = \exp\left(\sum_{i=1}^{k}\ln(Y_i) - B^1 * \frac{1}{N}\sum_{i=1}^{k} X_i\right)$$

X,Y?

Az ernyőkép a *(0,1)* *(1,2)* *(2,4)* *(3,8)* *(4,16)* *(5,32)* pontpárok beadása után:

ATLAG	X =	2.50	Y =	1.73
VARIANCIA	VX =	3.50	VY =	1.68
PONTOK SZAMA	=	6.00		
Y = A*B ↑ X				
A = 1	B =	2		

A kijelzett adatok az X, Y, VX, VY, N, A, B változókban is megtalálhatók.

Ha a ponthalmaz közelítése az Y tengellyel párhuzamos egyenes lenne, akkor az A, B paraméterek kijelzése helyett a FUGGOLEGES üzenet jelenik meg, amire a K1 változó nulla értéke utal.

P – 34 – S
GAUSS

CLOAD "GAUSS"

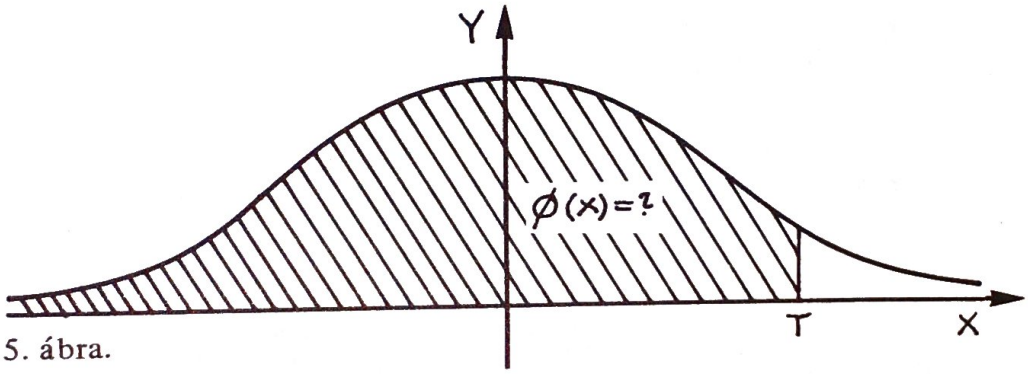
A betöltött programot az alábbi két funkciónak megfelelően a RUN 100, illetve a RUN 200 paranccsal lehet indítani.
A program első része a normál eloszlás

$$\phi(X) = \int_{-\infty}^{T} \phi(T)\, dx$$ értéket számolja ki.

ahol

$$\phi(T) = \frac{1}{\sqrt{2\Pi}}\, e^{-\frac{x^2}{2}} \qquad T = \frac{1}{1 + PX}$$

a Hasting-féle közelítéssel.



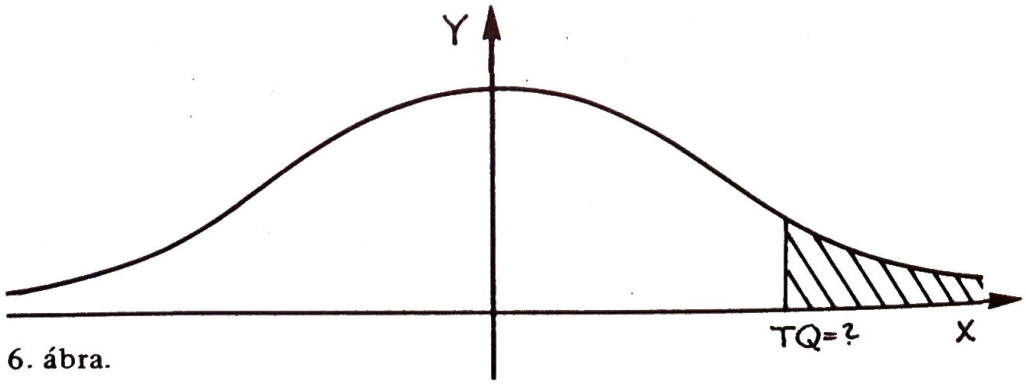
5. ábra.

```
RUN 100
GAUSS ELOSZLÁS
T = ?     0
F(X) = . 499999
```


A program második része az ismert

$$Q = \int_{TQ}^{\infty} \phi(T)\, dx$$ értékből a TQ értékét

határozza meg a Hasting-féle közelítéssel.



6. ábra.

```
RUN .200
INVERZ  GAUSS
Q    = ? .158655
TQ   = 1.00007
READY
>←
```

P – 35 – S

POISSON CLOAD "POISSON"

A betöltött program a RUN paranccsal indítható.

A program a $P(X) = \frac{e^{-M} * M * X}{X!}$ $X = 0, 1, 2 \ldots$

összefüggés alapján számolja az M paraméterű Poisson eloszlású X valószínűségű változó különböző pozitív egész számokra vonatkozó P(X) valószínűségeit, valamint az

$$L(P) = \sum_{i=0}^{k} P(X_i); \qquad V = M; \qquad S = SQR(M)$$

sűrűség, várható és szórás értékeket.

X számára negatív számot megadni nem lehet, ilyenkor a program megismételteti a bevitelt. A program a beadott szám egész részével számol tovább.

```
RUN
POISSON ELOSZLAS
M = ?  2,5
X = ?  5

VARHATO ERTEK =  2.5
SZORAS        =  1.58114

P(0) = .082      L(P) = .082
P(1) = .205      L(P) = .287
P(2) = .257      L(P) = .544
P(3) = .214      L(P) = .758
P(4) = .134      L(P) = .891
P(5) = .067      L(P) = .958
```


Az utolsóként kijelzett valószínűségű adatok a P és L változókban is rendelkezésre állnak.

P – 36 – S

BINOMIAL CLOAD "BINOMIAL"

A betöltött program a RUN paranccsal indítható.

A program a $P(X) = \binom{N}{X} P^X (1 - P)^{N-X}$ $X = 0, 1, 2 \ldots$

összefüggés alapján számolja az N, P paraméterű binomiális eloszlású X valószínűségi változó különböző pozitív egész számokra vonatkozó P(X) valószínűségeit, valamint az

$$L(P) = \sum_{i=0}^{x} P(X_i); \quad V = N*P; \quad S = SQR(N*P(1-P))$$

sűrűség várható és szórás értékeket.

X számára negatív számot megadni nem lehet, ilyenkor a program megismételteti a bevitelt. A program a beadott szám egész részével számol tovább.

```
RUN
BINOMIALIS ELOSZLAS

    N = ?   4
    P = ?   .33333
    X = ?   4

VARHATO ERTEK = 1.33333
SZORAS        = .942809

    P(0) = .197531    L(P) = . 197531
    P(1) = .395062    L(P) = .592594
    P(2) = .296296    L(P) = .88889
    P(3) = .098765    L(P) = .987655
    P(4) = .012347    L(P) = 1
```

Az N, P, X, V, S jelű változókban rendelkezésre állnak a paraméterek és a számított értékek.

A B jelű változóban P(X) míg az L jelűben L(P) utolsó értéke található.

P – 37 – U

NAPTAR CLOAD "NAPTAR"

Öröknaptár a XX. századra. Indítsuk el a programot a RUN paranccsal.

Válaszoljunk a számítógép alábbi kérdéseire:

```
EV      = ? 1983
HONAP   = ? 8
NAP     = ? 6
SZOMBAT
READY
```

A XX. század éveinek túl nem lépése, illetve létező hónapok és napok megadása a felhasználó feladatát képezik!

P – 38 – U
DATUMOK1 CLOAD "DATUMOK1"

A betöltött program a RUN paranccsal indítható.
A program egy a XX. századra érvényes öröknaptár segítségével meghatározza egy START és STOP dátum közti napok számát. A program negatív eredményt ad, ha a START dátum későbbi mint a STOP dátum. A program mindaddig kínálja szolgáltatását, míg futását meg nem szakítják.
A két dátum között eltelt napok száma az X jelű változóban is rendelkezésre áll.

Figyelem:
a) A XX. században belüli létező hónapok és napok megadása a felhasználó feladatát képezik!
b) Azonos dátumok között eltelt napok száma nulla!

RUN

```
START   EV     HO     NAP ? 1983. 1, 1
STOP    EV     HO     NAP ? 1984.1, 1
KOZTI  NAPOK  SZAMA       = 365

START   EV     HO     NAP ? 1984,1, 1
STOP    EV     HO     NAP ? 1985,1, 1
KOZTI  NAPOK  SZAMA       = 366
```

P – 39 – U
DATUMOK2 CLOAD "DATUMOK2"

A betöltött program a RUN paranccsal indítható.
A program egy a XX. századra érvényes öröknaptár segítségével,egy referenciaként megadott dátumhoz képest, kívánt számú nappal a jövő felé (pozitív), illetve a múlt felé (negatív) számolva megadja az illető nap dátumát.
A program mindaddig kínálja szolgáltatásait, mígcsak futását meg nem szakítják.
A válasz év, hó, nap adatai rendre az A, B, C, jelű változókban is rendelkezésre állnak.
Figyelem:
a) A XX. században belüli, létező hónapok és napok megadása a felhasználó feladatát képezik!
b) Napok számaként nullát megadva az eredeti dátumot kapja vissza a felhasználó.

```
RUN
REF. EV        HO        NAP    ?    1984, 2, 27
NAPOK SZAMA                     ?    3
A DÁTUM                         :    1984. 3. 1
```

P – 40 – U
HITEL CLOAD "HITEL"

A betöltött program a RUN paranccsal indítható.
A program az OTP áruvásárlási hiteleinek pénzügyi kamatos kamat számításait végzi el.

Pl. Vásárolni kívánunk egy INFRA–COLOR színes TV-t.
Ára: 28.800 Ft az előleg 50 %-a az árnak, tehát 14.400 Ft.

```
RUN
                OTP HITEL
AR          =          =  28.800  A
ELOLEG      =          =  14.400  E
KAMAT       =          =       1
HONAP       =          =      11  H
ELSO  RESZLET          =   1.379  D
HAVI  RESZLET          =   1.389  Y
KIFIZETENDO            =  15.269  F
KEZELESI DIJ           =     869  K
READY
```

A kijelzett értékek a feltüntetett változókban is rendelkezésre állnak.

A számítás alapját a következő képlet adja:

$$Y = (A - E) * \frac{K}{1 - (1 + K)^{-H}}$$

P – 41 – U
AMORT CLOAD "AMORT"

A betöltött programot a RUN paranccsal lehet indítani.
A program az értékcsökkenés értékét számolja ki a beszerzési ár, leírási százalék és a leírási évek adataiból a

csökkent érték = beszerzési ár $*[1-(\frac{\text{leírási százalék}}{\text{beszerzési ár}})^{-\text{évek}}]$

képlet alapján

```
RUN
BESZERZESI AR          = ? 700
LEIRASI SZAZALEK       = ? 10
LEIRASI EVEK           = ? 14
```

EV	*ERTEK*	*LEIRVA*
1	593.84	106.16
2	503.78	90.06
3	427.37	76.40
4	362.56	64.82
5	307.57	54.99
6	260.93	46.65
7	221.35	39.57
8	187.78	33.57
9	159.30	28.48
10	135.14	24.16
11	114.65	20.50
12	97.26	17.39
13	82.51	14.75
14	70.00	12.51

```
TOTAL 630.00
READY
>←
```

P – 42 – U

NEVSOR CLOAD "NEVSOR"

A betöltött program a RUN paranccsal indítható.

A program a tetszőleges sorrendben beadott neveket ábécé rendbe rendezi és kiírja a képernyőre.

Maximum 60 nevet tud kezelni és egy névre 30 karakternyi hely jut átlagosan. Ennél kevesebb név beadásakor a bevitelt VEGE szóval kell lezárni.


```
RUN
      KEREM A NEVEKET
1     ?    KISS        OTTO      (CR)
2     ?    KOCSIS      JOZSEF    (CR)
3     ?    NAGY        ANDRAS    (CR)
4     ?    ANDAI       KATALIN   (CR)
5     ?    DARABOS     ELEK      (CR)
6     ?    TOTH        RUDOLF    (CR)
7     ?    PETHES      ROBERT    (CR)
8     ?    KISS        GYULA     (CR)
9     ?    VEGE
```

```
NEVSOR

ANDAI        KATALIN
DARABOS      ELEK
KISS         GYULA
KISS         OTTO
KOCSIS       JOZSEF
NAGY         ANDRAS
PETHES       ROBERT
TOTH         RUDOLF

READY
```

A kijelzett nevek ebben a sorrendben megtalálhatók az N $(I) tömb megfelelő indexű elemeként is!

```
P – 43 – U
HALOTERV                    CLOAD "HALOTERV"
```

A betöltött program a RUN paranccsal indítható.

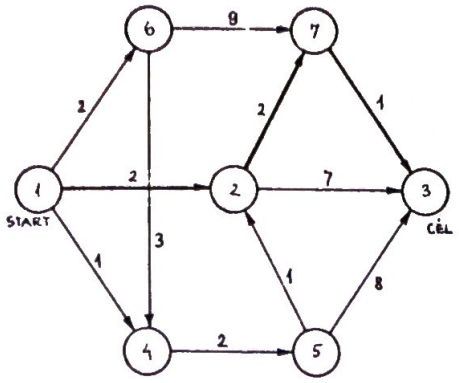
A program egy megadható csomópontú PERT háló ún. kritikus útját képes meghatározni.

```
RUN
CSOMOPONT    = ?  7
UTVONAL      = ?  11
KEZDOPONT    =    1
CELPONT      =    3

PONTBOL, PONTBA, IDO  ?  1,  2,  2
PONTBOL, PONTBA, IDO  ?  1,  4,  1
PONTBOL, PONTBA, IDO  ?  1,  6,  2
PONTBOL, PONTBA, IDO  ?  6,  4,  3
PONTBOL, PONTBA, IDO  ?  6,  7,  9
PONTBOL, PONTBA, IDO  ?  4,  5,  2
PONTBOL, PONTBA, IDO  ?  5,  2,  1
PONTBOL, PONTBA, IDO  ?  5,  3,  8
PONTBOL, PONTBA, IDO  ?  2,  7,  2
PONTBOL, PONTBA, IDO  ?  7,  3,  1
PONTBOL, PONTBA, IDO  ?  2,  3,  7
```


A program az előre közölt útvonalszám megadása után a bevitelt automatikusan befejezi.

```
KRITIKUS UT HOSSZA = 5
MENETE
3    7    2    1
READY
>←
```



7. ábra.

P – 44 – E
RLC – PARH CLOAD "RLC–PARH"

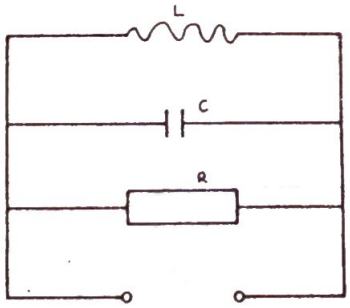
A betöltött program a RUN paranccsal indítható.

A program egy R, L, C párhuzamos elemeivel jellemzett rezgőkör elektromos impedanciáját számolja ki megadott frekvencián az alábbi képletek segítségével:

$$\bar{Z} = \frac{1}{\frac{1}{R} - j(\frac{1}{\omega L} - \omega C)}$$

$$|\bar{Z}| = \frac{1}{\sqrt{\frac{1}{R^2} + (\frac{1}{\omega L} - \omega C)^2}}$$

$$\phi = ATN[R(\frac{1}{\omega L} - \omega C)]$$



8. ábra.

RUN

PARHUZAMOS REZGOKOR

F(HZ) = ? *50*
L(H) = ? *50 E–3*
R(OHM) = ? *10*
C(F) = ? *1E–6*

```
Z  =  8.44764
FI  = 32.3534
RE  =  7.13625
IM  =  4.52067
```

A kijelzett eredmények a Z, FI, RE, IM jelű változókban is rendelkezésre állnak.

P – 45 – E
RLC – SOR CLOAD "RLC–SOR"

A betöltött program a RUN paranccsal indítható.
A program egy R, L, C soros elemeivel jellemzett rezgőkör elektromos impedanciáját számolja ki megadott frekvencián az alábbi képletek segítségével:

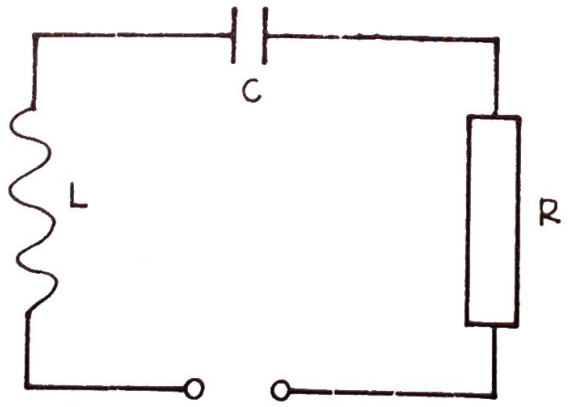
$$\bar{Z} = R + j(\omega L - \frac{1}{\omega C})$$

$$|Z| = \sqrt{R^2 + (\omega L - \frac{1}{\omega C})^2}$$

$$\phi = ATN\left(\frac{\omega L - \frac{1}{\omega C}}{R}\right)$$

```
RUN
                SOROS REZGOKOR
F(HZ)  =  ?   50
L( H)  =  ?   30 E-3
R(OHM) =  ?   5
C( F)  =  ?   10 E-6

Z   =  308.926
FI  = -89.0727
RE  =  5
IM  = -308.885
```

9. ábra.

A kijelzett eredmények a Z, FI, RE, IM jelű változókban is rendelkezésre állnak.

P – 46 – E
DELTA – CS

CLOAD "DELTA–CS"

A betöltött program a RUN paranccsal indítható.

A program az elektronikában jól ismert delta – csillag (△→ Y) átalakítást végzi el komplex impedanciákkal.

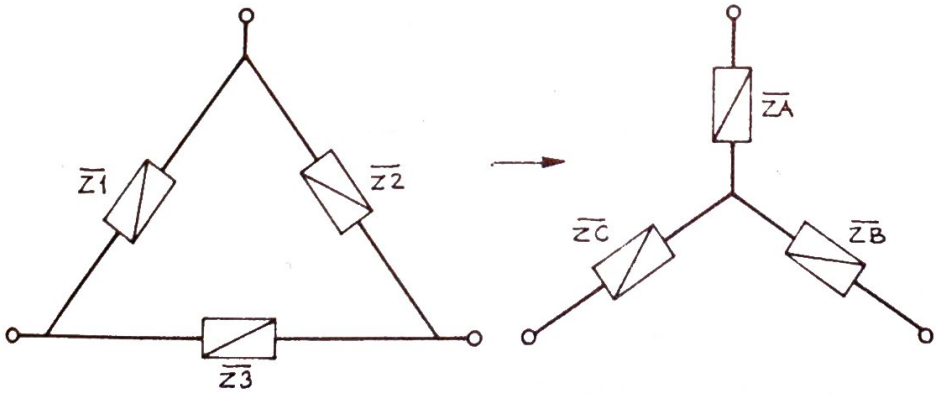
$$\overline{ZA} = \frac{\overline{Z1} * \overline{Z2}}{\overline{Z1} + \overline{Z2} + \overline{Z3}} \qquad \overline{ZB} = \frac{\overline{Z2} * \overline{Z3}}{\overline{Z1} + \overline{Z2} + \overline{Z3}}$$

$$\overline{ZC} = \frac{\overline{Z1} * \overline{Z3}}{\overline{Z1} + \overline{Z2} + \overline{Z3}}$$

```
RUN
Z1  R1  =  ?  2
Z1  X1  =  ?-1
Z2  R2  =  ?  3
Z2  X2  =  ?  1
Z3  R3  =  ?  1
Z3  X3  =  ?-1
```

		R		X
ZA	=	1.16216	J *	.0270271
ZB	=	.702703	J *	–.216217
ZC	=	–.243243	J *	–.45946

$\overline{ZA}$, $\overline{ZB}$, $\overline{ZC}$ -ről közölt eredmények reális és imaginárius részei rendre az R1, R2, R3 és X1, X2, X3 változókban is rendelkezésre állnak.



10. ábra.

P – 47 – E
CS–DELTA CLOAD "CS–DELTA"

A betöltött program a RUN paranccsal indítható.
A program az elektronikában jól ismert csillag – delta (Y → Δ) átalakítást végzi el komplex impedanciákkal.

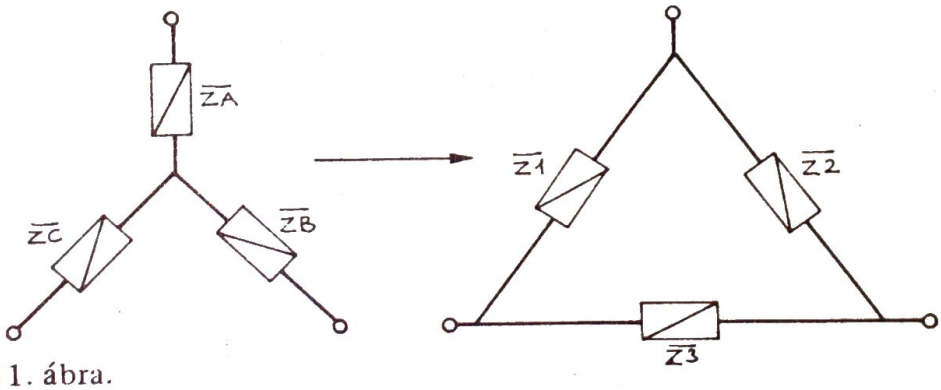
$\overline{Z1} = \overline{S}/\overline{ZB}$ $\overline{Z2} = \overline{S}/\overline{ZC}$ $\overline{Z3} = \overline{S}/\overline{ZA}$
ahol

$\overline{S} = \overline{ZA}$ $\overline{ZB} + \overline{ZB}$ $\overline{ZB} + \overline{ZA}$ $\overline{ZC}$

```
RUN
ZA  RA  =  ?   8
ZA  XA  =  ?   4
ZB  RB  =  ?   2
ZB  XB  =  ?  -1
ZC  RC  =  ?   3
ZC  XC  =  ?   6
```

	R	X
Z1	= – 1.00009	J * 34
Z2	= 11.3333	J * .333347
Z3	= 6.64999	J * 5.30001

$\overline{Z1}$, $\overline{Z2}$, $\overline{Z3}$-ról közölt eredmények reális és imaginárius részei az R1, R2, R3 és X1, X2, X3 változókban is rendelkezésre állnak.



11. ábra.

P – 48
KAPAC. CLOAD "KAPAC"

A betöltött program a RUN paranccsal indítható.
A program két párhuzamos felületű elektróda közötti kapacitás értékét számolja ki a következő képleteknek megfelelően:

$$C(PF) = .0885419 * \frac{Er*L*\omega}{D} (1 + P)$$

ahol

$$P = \begin{cases} 0 & \text{ha} \quad 100\omega > L \\ \frac{D}{\Pi*\omega} (1 + \ln \frac{2*\Pi*\omega}{D}) & \text{egyébként.} \end{cases}$$

```
RUN
KAPACITAS KET PARH. ELEKTRODA
                KOZOTT

L(CM)  =  ?   20
ω(CM)  =  ?   1
D(CM)  =  ?   .01
ER     =  ?   1
C(PF)  =  177.084
```

A kijelzett kapacitásérték a C jelű változóban is rendelkezésre áll!

P – 49 – C

HEX CONV CLOAD "HEX CONV"

A betöltött programot a RUN paranccsal lehet indítani.

A program indítása után kéri a decimális számot, és ha az a 0 – 65535 tartományba esik, akkor kiírja hexadecimális alakban, egyébként ismételten kéri a decimális számot.

A program ezt folytatólagosan ismétli, míg csak a program futását meg nem szakítjuk.

```
DECIMALIS  =  ?  1E6
DECIMALIS  =  ?  12
HEXA       =  ?  000C

DECIMALIS  =  ?  10000
HEXA       =  ?  2710
```


P – 50 – C

HEXA MON CLOAD "HEXA MON"

A betöltött programot a RUN paranccsal lehet indítani.

A program a DUMP
FILL
MODIFY funkciókat valósítja meg, hexadecimális adatokkal.

A programot betöltés után RUN-nal kell indítani. Ennek hatására a képernyő törlése után a bal felső sarokban egy > karakter jelenik meg. Ezután lehetséges az előbb felsorolt 3 funkció aktivizálása a megfelelő kezdőbetű segítségével.

A funkciókhoz hexadecimális formában meg kell adni a szükséges címeket és adatokat, ezeket akár vesszővel, akár szóközzel lehet elválasztani egymástól.

A címként használt paramétereknél az utolsó négy karakter az érvényes, adatoknál pedig az utolsó kettő. Ha tehát rontunk, akkor nem kell visszalépni, hanem csupán újrakezdeni a bevitelt.

A hexadecimálisan nem értelmezhető bevitelek hatására a gép kilép a kiválasztott alprogramból.

A program az alprogramok felismerésére egy kötőjellel reagál.

a) DUMP cím 1., cím 2.,
Pl. > D–5000␣50 FF␣

Ettől eddig dumpold ki
␣ szóköz

b) FILL cím 1., cím 2., adat
Pl. >F–3000, 30FF, 2A

Ettől eddig, ezzel töltsd fel a memóriát

c) MODIFY cím 1.,

Pl. > M–5000,
5000␣3A–␣2C–

Tedd lehetővé a memória módosítását az adott címtől kezdve
A rutin egy kötőjellel ajánlja fel a változtatási lehetőséget. Ha evvel nem kívánunk élni, úgy szóközt vagy vesszőt kell beütni.

5000␣␣3A–3E␣2C–2A␣

Módosításkor be kell írni a szükséges adatot majd lezárni szóközzel vagy vesszővel.

5000 3A←FF 4FFF

Visszalépés a ↑ billentyűvel lehetséges.

Kilépés a CR billentyűvel lehetséges

Figyelmeztetés: Bizonyos memóriacímeken a program látszólag nem működik. Ez akkor fordul elő, ha magát a programot, illetve annak saját belső változóit próbáljuk átírni.

8. ÁLTALÁNOS ÜZEMELTETÉSI ÉS BIZTONSÁGTECHNIKAI ELŐÍRÁSOK, ÓVÓ RENDSZABÁLYOK

Kérjük, hogy a biztonságos és hibamentes működés érdekében tartsa szem előtt a következőket.

- A számítógépet ne tegye ki nagy hőmérsékletváltozásoknak, pornak, vagy egyéb szennyeződésnek! Tartsa száraz levegőjű helyiségben!
- A számítógép tisztítását végezze puha, száraz rongy segítségével! Ne használjon semmiféle tisztító folyadékot, vagy vizet!
- A számítógép magno-, antenna- és bővítőcsatlakozóinak érintkező felületeit érinteni még kézzel sem szabad!
- A csatlakozókábeleket úgy kell elhelyezni, hogy azok nehogy útba kerüljenek!
- Ne takarja el a gép szellőzőnyílásait! A programkazettákat, papírlapokat és könyveket a gép mellett helyezze el!
- Hiba esetén forduljon bizalommal a gyártóhoz!

Kérjük, hogy észrevételeit, kritikáját és javaslatait juttassa el hozzánk levélben.

9. KARBANTARTÁSI ÉS SZERVÍZELŐÍRÁS

Az IPT–002 személyi számítógép különleges karbantartást nem igényel. A 8. fejezetben leírtak betartása ajánlott.

A személyi számítógép a gyártás folyamán többszörös „burn in" technológiai lépcsőn esett át, így várható élettartama igen magas.

Meghibásodás esetén a felhasználó forduljon a gyártóhoz.

Illetéktelen behatolás, a nem megengedett körülmények közötti működtetés következményeiért a vevő felel.

10. A MŰSZAKI ÁLLAPOT ELLENŐRZÉSE

A készülék a tápfeszültség bekapcsolása után azonnal üzemképes, és a TV képernyőjén a már ismertetett üzenet jelzi a működőképes állapotát:

```
IPT – 002     BASIC
TOT. MEM.     (BYTES) 16384
READY
> ←
```

A készülék működésre kész.

11. AZ M 4001 TÍPUSU TÁPEGYSÉG CSATLAKOZTATÁSA

Az IPT– 002 nem rendelkezik saját tápellátással.
Az IPT– 002-ben csak *törpefeszültség* található (max. 12 V).
Az IPT– 002 speciális 6 pólusú tuchel csatlakozón át kívülről kapja a DC tápfeszültségeket:

+ 5 V	stabilizált
+ 12 V	stabilizált
– 12 V	stabilizált
– 5 V	stabilizált

Kb + 10 V stabilizálatlan, szűrt.

A tápfeszültség a bementi pontnál zener diódás túlfeszültségvédelemmel és olvadó biztosító betétes túláramvédelemmel van ellátva.
Az M 4001 típusú tápegység a fenti DC feszültségeket, a megfelelő terhelő áramokkal szolgáltatja. Az M 4001 típusú tápegység részletes leírását külön gépkönyv tartalmazza.

12. MELLÉKLETEK

AZ IPT–002 TÍPUSÚ SZEMÉLYI SZÁMÍTÓGÉP KULCSSZAVAINAK LISTÁJA

ABS
AND
ASC
ATN
AUTO
CALL
CDBL
CHR$
CINT
CLEAR
CLOAD
CLOAD?
CONT
COS
CSAVE
CSNG
DATA
DEFDBL
DEFINT
DEFSNG
DEFSTR
DEG
DELETE
DIM
EDIT
ELSE
END
ERL
ERN
ERROR
EXP
FIX
FOR
FRE
GOSUB
GOTO
IF
INKEY$
INP
INPUT
INPUT#
INT
LEFT$
LEN
LET
LIST
LLIST
LNT
LOCATE
LOG
LPRINT
MEM
MID$
NEW
NEXT
NOT
ON
OR
OUT
PEEK
PI
POKE
PRINT
PRINT#
RAD
RANDOM
READ
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN
SGN
SIN
SQR
STEP
STOP
STRING$
STR$
TAB
TAN
THEN
TO
TROFF
TRON
USING
VAL
VARPTR

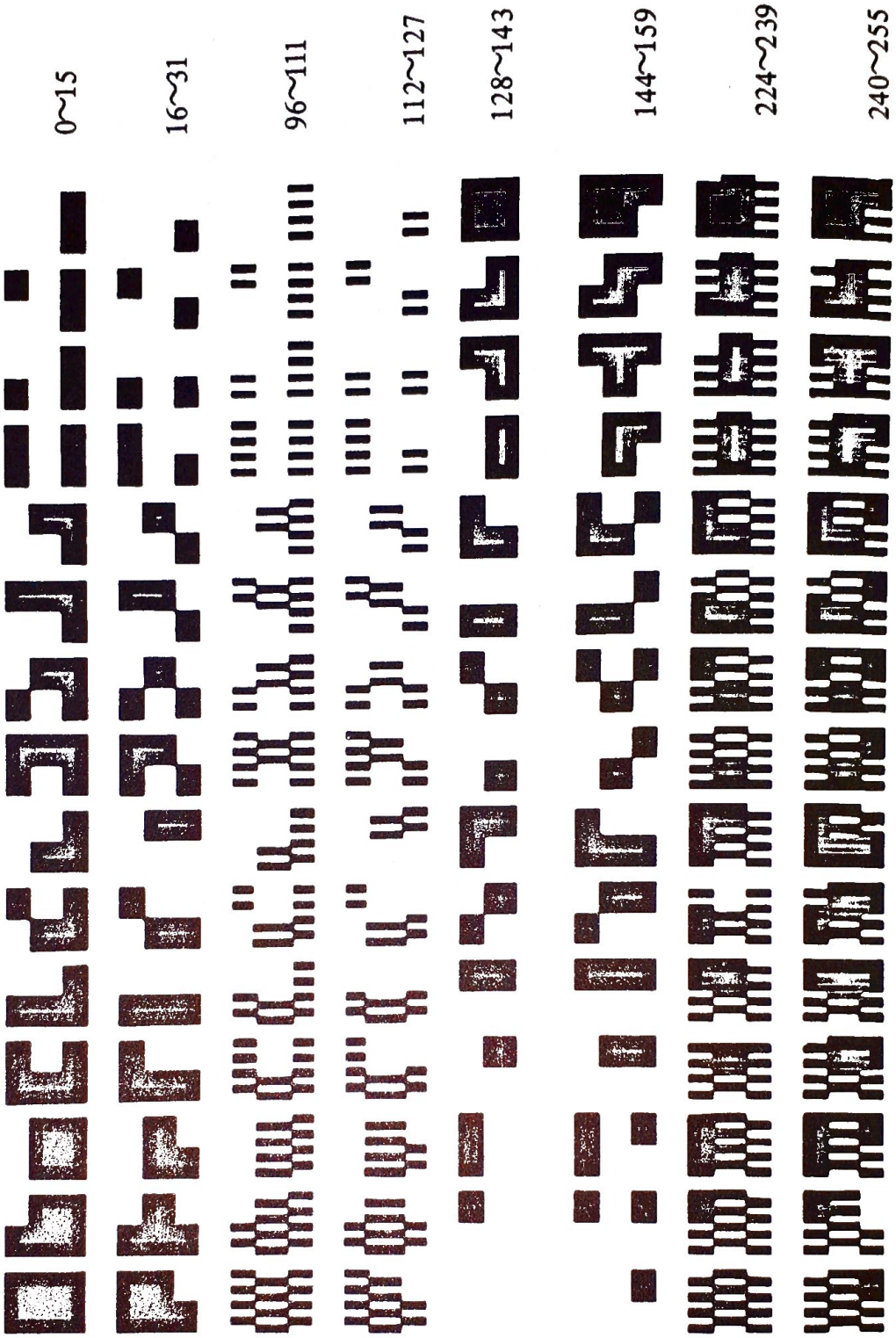
HIBAÜZENETEK LISTÁJA

Hiba-kód	Hiba-üzenet	Jelentése
1	NF	NEXT utasítás megelőző FOR nélkül;
2	SN	Szintaktikus (nyelvtani) hiba.
3	RG	RETURN utasítás megelőző GOSUB nélkül;
4	OD	Túl sok a READ a DATA listához;
5	FC	Funkcionális hiba az elemi függvények alkalmazásakor;
6	OV	Túlcsordulás. A keletkezett eredmény nem ábrázolható az előírt típusú változóban!
7	OM	Nagyméretű tömb vagy hosszú program, a sok változó miatt a szabad tárterület elfogyott!
8	UL	Vezérlésátadás megkísérlése nem létező sorra
9	BS	Tömb nem létező indexű elemére való hivatkozás
10	DD	Kísérlet egy tömb újradimenzionálására
11	/0	Nullával osztás
12	ID	Input utasítás nem programból
13	TM	Numerikus és szöveges változók használatának összekeverése
14	OS	A szöveg számára kijelölt terület kevés, elfogyott
15	LS	Túl hosszú karakterlánc
16	ST	Túlságosan bonyolult karakterlánc- formula
17	CN	A megszakított programot nem lehet folytatni.
18	NR	A hibacsapdából való kilépésnél nem volt RESUME utasítás
19	RW	RESUME utasítás megelőző hiba nélkül
20	UE	Az ERROR utasítással szimulálandó hiba kódja nincs ebben a táblázatban
21	MO	Hiányzó operandus
22	FD	Hibás felvétel, rossz program a kazettán
23	DB	Csak a lemezzel (diskkel) bővített számítógép számára fenntartott utasítás

ASCII, ALFANUMERIKUS ÉS KVÁZIGRAFIKUS KARAKTEREK LISTÁJA

␣	32	0	48		64	P	80
!	33	1	49	A	65	Q	81
"	34	2	50	B	66	R	82
#	35	3	51	C	67	S	83
$	36	4	52	D	68	T	84
%	37	5	53	E	69	U	85
&	38	6	54	F	70	V	86
'	39	7	55	G	71	W	87
(	40	8	56	H	72	X	88
)	41	9	57	I	73	Y	89
*	42	:	58	J	74	Z	90
+	43	;	59	K	75	[	91
,	44	<	60	L	76	\	92
–	45	=	61	M	77	]	93
.	46	>	62	N	78	↑	94
/	47	?	63	O	79	←	95

IPT-002 KVÁZIGRAFIKUS KARAKTEREI



MI 4001 típusú tápegység gépkönyve

TARTALOM

1. A készülék rendeltetése 103

2. Műszaki adatok 103

3. A kezelőszervek, működtetésük 104

4. Biztonsági előírások, óvó rendszabályok 104

5. Karbantartási- és szervizutasítások 105

6. Tartozékok 105

1. A készülék rendeltetése

A készülék az IPT–002 típusú személyi számítógép tápellátására szolgál. Elhelyezési lehetőséget nyújt a számítógép bővítő kártyái és csatlakozói számára.

2. Műszaki adatok

Hálózati feszültség:	220 V ± 10 %
Hálózati frekvencia:	50/60 Hz
Fogyasztás:	max. 60 VA a teljes konfigurációs terhelésnél
Kimenő stabilizált DC feszültségek és max. terhelő áram:	+ 5 V – 2 A + 12 V – 1,5 A – 12 V – 100 nA – 5 V – 700 nA
Kimenő stabilizálatlan feszültség:	kb. 10 V
Túláramvédelem:	a stabilizált DC feszültségek túláramvédelemmel vannak ellátva, ez a megengedett maximális terhelő-áramérték felett, nem specifikált áramértéknél működőképes.
Életvédelem:	a tápegység életvédelmi földeléssel van ellátva. *Kizárólag védőföldeléssel* ellátott csatlakozóba csatlakoztatható.
Vagyonbiztonság:	a tápegység primer oldalán és szekunder oldalán túláramvédelemmel van ellátva, a beépített 1 db primeroldali és 6 db szekunderoldali olvadó biztosító betét látja el a védelmet. Olvadó biztosíték értéke: primer: 315 mA (220 V) szekunder: 315 mA (25 V) 315 mA (25 V) 1,6 A (14 V)

1,6 A	(14 V)
3,15 A	(9 V)
3,15 A	(9 V)

A tápegység kizárólag az IPT–002 tápellátására szolgál.

Működési hőmérséklettartomány: + 5 C° – + 35 C°

A készülék megfelel az MSZ 94 élet- és vagyonbiztonsági előírásainak.

3. Kezelőszervek, működtetésük

Az M 4001 tápegység bekapcsolására a POWER ON feliratú kapcsoló szolgál.

A FUSE feliratú primer olvadó biztosító szolgáltatja a túláram elleni védelmet.

A hálózati kábel háromerű, védőföldeléssel ellátott hálózatba csatlakoztatható.

A hálózatra kapcsolt állapotot a AC POWER ON feliratú LED világítása jelzi.

Az IPT–002 mikrogép-tápegység kábeljának csatlakoztatására speciális csatlakozó szolgál.

Az M 4001 tápegység a hálózatra való kapcsolás után azonnal üzemképes.

4. Biztonsági előírások, óvó rendszabályok

Az M 4001 tápegység kizárólag védőföldeléssel ellátott hálózatba csatlakoztatható. A névleges hálózati feszültség 220 V.

A készülék bekapcsolt, illetve hálózatra kapcsolt állapotban nem szedhető szét, mert a magas feszültség miatt *életveszélyes* a nem hozzáértők számára.

Az esetleges olvadó biztosító csere csak feszültségmentes állapotban hajtható végre.

5. Karbantartási és szervizutasítások

M 4001 tápegység karbantartást nem igényel.

Ha a gép üzemképtelen, célszerű az olvadó biztosító betéteket megvizsgálni és ha kell cserélni. További működésképtelenség, üzemzavar esetén a gyártó vállalathoz kell fordulni.

6. Tartozékok

A készülék az alábbi tartozékokkal kerül forgalomba:
1 db gépkönyv
7 db olvadó biztosító betét

INDEX

A

ABS 34
adatlista 30
adatmegadás 29
adattípusok megadása 32
alfanumerikus karakterek 97
AMORT 80
AND 39
APOLLO 59
aritmetikai függvények 33
aritmetikai műveletek 18
ASC 36
ATN 34
AUTO (sorszámképzés) 20
AUTO 62

B

beviteli utasítások 23
beolvasás kazettáról 27
billentyűzet 77
BINOMIAL 77
BISECT 68

C

CALL 48
CDBL 32
CLEAR 29
CHRS 36
ciklusutasítás 41
CINT 32
CLOAD 28
CLOAD? 28
CONT 20
COS 34
CSAVE 28
CSNG 32
CS-DELTA 86

D

DATA 30
DATUMOK1 79
DATUMOK2 79
DEFDBL 32
DEFINIT 33
DEFSNG 33
DEFSTR 33
DEG 34
DELETE 21
DELTA-CS 85
DIM 30
dupla pontosságú számok 15

E

EDIT 21,50
egész (integer) számok 15
egyszeres pontosságú számok 15
END 40
ERL 47
ERN 47
ERROR 46
EXP 34
EXP.REG. 75

F

FOGD MEG 61
FOGOCSKA 59
felfüggesztés (programé) 46
feltétel nélküli vezérlés-átadás 42
feltételes utasítás 44
FIBONAC. 72
FIX 34
FOR TO STEP NEXT 41
FRE 37

G

GAUSS 76
gépi kódu programozás 48
GOSUB 43
GOTO 42
GRAFIKA 58

H

HALOTERV 82
HEX.CONV. 87
hibaüzenetek 96

I

IF THEN ELSE 44
INKEY$ 37
INP(port) 27
INPUT 26
INPUT# 27
INT 34

INT–INT 67
INV MAT 65
inzertálás 52

K

kalkulátor üzemmód 14
KAPAC 86
karakterek beszúrása 52
karakterek cserélése 50
karakterek leválasztása 26
karakterek törlése 50
karakterlánc 15
karakter függvények 36
karbantartás 90
kazettára rögzítés 27,28
kazettáról beolvasás 27
kiírás sornyomtatóval 26
kiírás képernyőre 21,23
kiviteli utasítások 23
kocsivissza 13
KOOR KONV 71
kulcsszavak 95
kurzor állítása 24
kurzor mozgatása 50
KVAD DET 66
kvázigrafikus karakterek 97

L

LABIRINT 58
LEFT$ 37
LEN 37
LET 30
LINALGE 63
LIN. REG. 74
LIST 21
LNKO 72
LNT 30
LOAD MAT 65
LOCATE 24
LOG 34
logikai függvények 39
logikai műveletek 19
LPRINT 26

M

magnetofon kiválasztása 10
magnetofon csatlakoztatása 11
megjegyzések 47
MEM 48
MID$ 38
MONOSZKOP 57
MPY MAT 66

N

NAPTAR 78
NEVSOR 81
NEW 21
NEWTON 68
NOT 40
numerikus változók 15

NY

nyomkövetés 22
nyomtató csatlakoztatása 11

O

olvasás 31
ON 45
ON ERROR GOTO 46
ON GOSUB 45
ON GOTO 45
OR 39
OUT port 28
összehasonlító műveletek 19

P

PEEK 48
PI 35
POISSON 77
POKE 48
POLIGON 70
PRINT 23
PRINT TAB 24
PRINT USING 24
PRINT# 27
PRIMEK 72
programfuttatás 22
program felfüggesztése 13
programkönyvtár 56
programmegszakítás 13
programsor törlése 21
program tabulálás 13
programutasítás 40
program üzemmód 14,20
program visszalépés 13

R

RAD 35
RANDOM 35
READ 31
REKLAM–1 57
REKLAM–2 57
REM 47
RESTORE 31

RESUME 46
RETURN 47
RIGHT$ 38
R. KUTTA 69
RLC–PARH 38
RLC–SOR 84
RND 35
ROM. NUM. 73
RUN 22

S

SGN 35
SIN 35
SIMPSON 67
SQR 36
sorszámképzés (automatikus) 20
STAT–I 73
STOP 46
STRING$ 39
STR$ 38

SZ

szerkesztő üzemmód 21,50,51
SZOPOKER 60
szubrutinhívás 43

T

TAN 36
tápegység csatlakoztatása 92
tápegység gépkönyve 99
tárba töltés (kazettáról) 28
tárfoglalás 29
televízió csatlakoztatása 11
TENISZ 62
tömb 15
törlés 21
TROFF 22
TRON 22

Ü

üzemeltetési előírások 89

V

változók azonosítása 16
változók formátuma 24,25
változók megadása 16,26
VARPTR 49
végrehajtó üzemmód 14,22
vezérlő funkciók 13