Gödöllői Agrártudományi Egyetem
Mezőgazdasági Főiskolai Kar
Gyöngyös

AIRCOMP-16.
személyi számítógép kezelése és programozása

Vajsz Tivadar
Gyöngyös

1985.

Gödöllői Agrártudományi Egyetem
Mezőgazdasági Főiskolai Kar
Gyöngyös

AIRCOMP-16.
személyi számítógép kezelése és programozása

Vajsz Tivadar
Gyöngyös

1985.

AIRCOMP 16 személyi számitógép kezelése és programozása

1. Általános ismeretek

1.1. A gép legfontosabb paraméterei

Mikroprocesszor: Z 80A
Memória: 1615 bájt RAM
8 KB ROM
Perifériák: UHF sáv vagy video vételére alkalmas TV
Kazettás magnetofon
Billentyüzet /egy[illegible]pitve a géppel/
Tápegység

1.2. A számitógép üzembe helyezése

A számitógép üzemeltetéséhez szükség van tápegységre és monitorra. A számitógép a müködéshez szükséges áramot megfelelő tápegységen keresztül a hálózatról kapja. Monitornak minden olyan TV megfelel, amely beállithatóa 6-os vagy a 21-es csatornára. Ettől jobb megoldás, ha olyan TV-t használunk, amelynek van videó-bemenete. Ilyenkor a TV-n minőségileg jobb képet kapunk. A számitógéphez - a megfelelő kábelekkel - a tápot - a TÁP feliratnál, a TV-t vagy a TV vagy a VIDEO feliratnál lehet csatlakoztatni A számitógépet és a TV-t összekötő kábel másik végét a TV antennabemenetére, illetve a videobemenetére kell csatlakoztatni

Az összekapcsolás után először a TV-t majd a tápegységen levő kapcsolóval a gépet kapcsoljuk be.
Ha a TV-hez az antennabemenetnél csatlakoztunk, akkor a TV-t a 6-os vagy 21-es csatornán rá kell hangolni a számitógépre. A TV-t addig hangoljuk, amig az alábbi felirat meg nem jelenik. /Videobemeneti csatlakozás esetén ez azonnal megjelenik/

-AIRCOMP-
16115 BYTES FREE COMPUTER
HOMELAB BASIC REV. 1.2
OK

Az OK felirat alatt a képernyő szélén egy négyzet alaku jel, a kurzor villog
Az OK felirat és a kurzor megjelenése jelzi, hogy a számitógép kész az operátori /gépkezelői/ utasitások végrehajtására.

1.3. Editálás /szövegszerkesztés/

1.3.1. Billentyüzet

Ahhoz, hogy a számitógépet használni tudjuk először a gép billentyüzetét kell megismerni /1 ábra/.

1. ábra

! 1	" 2	# 3	$ 4	% 5	& 6	' 7	(8	) 9	INS 0	* :	← →
SQR Q DEFFN	ATN W STEP	STR$ E END	RND R RETURN	TAN T THEN	EXP Y NEW	USR U LOAD	MID $ I IF	COS O ON	PEEK P POKE	- =	↓ ↑
NOT A AND	SIN S SAVE	INPUT D DIM	LFT$ F FOR	RGH$ G GOTO	CHR$ H GOSUB	INT J CONT	ASC$ K DATA	LEN$ L LIST	FRE ↑ TRC	+ ;	RUN BRK
LOG Z TO	POINT X POP	CALL C CLR	VAL$ V RESTR	PLOT B OR	SGN N NEXT	ABS M READ	, <	- >	/ ?	CR	
SH	SPACE									SH	

AIRCOMP - 16 személyi számítógép billentyűzete

Az egyes karakterek /jelek/ kiírási helyét a kurzor mutatja. Egy karakter beírása úgy történik, hogy a kívánt karaktert tartalmazó billentyűt lenyomjuk, ekkor a kurzor helyén megjelenik a karakter, s a kurzor egy pozícióval jobbra lép. Az AIRCOMP-16 számítógép minden billentyűje "ismétlő" billentyű. Ez azt jelenti, hogyha egy billentyűn levő funkciót többször végre akarunk hajtatni, elég a billentyűt egyszer lenyomni és nyomva tartani
Az első ami a billentyűzetről szembe tünik az, hogy majdnem minden billentyűhöz egynél több felirat tartozik, azaz a billentyük nagyrésze többfunkciós. Vegyük ezeket sorra!

Egyfunkciós billentyük

Az utolsó sorban 7 billentyűt találunk, ezek közül a két szélsőn SH felirat és egy kis téglalap látható. Ezek az irógépeken levő betűváltó billentyükre hasonlitanak. Akár az egyiket, akár a másikat lenyomva semmi nem történik, csak más billentyűvel való egyidejü lenyomásakor fejtik ki hatásukat. A két SH billentyü között bizonyos esetekben különbség van./lásd háromfunkciós billentyük/.
Az utolsó sorban levő többi öt billentyü azonos célra szolgál, ezek a space /szóköz/ billentyük

Kétfunkciós billentyük

Ezen billentyük azok, amelyekhez két felirat tartozik kivéve a CR feliratot tartalmazó billentyüt, amely kétfunkciós, de csak egy felirata van. Bizonyos értelemben a felette levő RUN/BRK billentyü is különbözik a többi kétfunkciós billentyütől, ezért erről a két billentyüről részletesebben később lesz szó.

A többi kétfeliratu billentyüre érvényes az, hogy lenyomásukkor mindig a billentyühöz tartozó alsó felirat határozza meg a nyomtatási képet. Ha a billentyün levő feliratot akarjuk aktivizálni, akkor a billentyü lenyomása előtt valamelyik [SH] billentyüt is le kell nyomnunk és nyomva kell tartanunk. Ebben az esetben a bal oldali és a jobb oldali [SH] billentyü között nincs különbség. A kétfeliratos billentyük között az utolsó oszlop első két billentyüje lenyomáskor nem ad nyomtatási képet a képernyőn. Ezek a kurzorvezérlő billentyük két-két nyilat tartalmaznak, a nyilak a kurzor elmozdulásának irányát mutatják. [SH] billentyü nélkül az alsó, [SH] billentyüvel lenyomva a felső nyil irányába mozdul el a kurzor.

Speciális kétfunkciós billentyük

1. [CR] billentyü lenyomásakor a kurzor egy uj sort kezd meg. Ha az [SH] billentyüvel együtt nyomjuk le, akkor CLEAR funkciót hajt végre /törli aképernyőt/.

2. [RUN BR] billentyü lenyomásakor a RUN felirat jelenik meg. Ha az [SH] billentyüvel együtt nyomjuk le, akkor PRINT felirat keletkezik a képernyőn. /A BRK funkció ismertetésére a programozási részben kerül sor/.

Háromfunkciós billentyük

Azok a billentyük, amelyekhez három felirat tartozik a háromfunkciós billentyük. A három feliratból az alsó és a felső angol szó vagy rövidités. A három funkciót az alábbiak szerint tudjuk aktivizálni.

1./ Alsó felirat: A bal oldali SH billentyűvel együtt nyomjuk le a billentyűt.

2./ Középső felirat: SH billentyű nélkül nyomjuk le a billentyűt.

3./ Felső felirat: A jobb oldali SH billentyűvel együtt nyomjuk le a billentyűt.

Megjegyzés
Az SH billentyű használatánál jegyezzük meg, hogy mindig az SH billentyűt nyomjuk le előbb, és ezt engedjük fel később!

1.3.2. Kurzor mozgatása a képernyőn

A képernyőre 25 sorban, soronként 40 karaktert lehet irni Ezeket a sorokat fizikai soroknak nevezzük. A későbbiek során használni fogjuk a logikai sor elnevezést. Egy logikai sort alkot közvetlenül egymás után gépelt karakterek sorozata. Egy logikai sor állhat 40 karakternél kevesebb, de több karakterből is. A logikai sor maximális hosszát a képernyő kapacitása korlátozza. A képernyőn 1000 karakter hely van. A kurzor-mozgató billentyükkel a képernyő tetszőleges sorában tetszőleges helyre irhatunk. A kurzor-mozgató használatakor a kurzor a képernyő tartalmának megváltoztatása nélkül tér át egyik pozicióról a másikra.

A kurzor automatikusan áttér egy uj fizikai sorra, ha valamelyik sor 1o pozíciójába irtunk. Amennyiben ez volt a 25. sor, akkor a képernyőn minden sor eggyel feljebb "lép" és igy az első sor eltünik a képernyőről. Ugyanez történik akkor is ha a kurzor a 25. sorban van és a kurzor-mozgató billentyüvel a kurzort lefelé mozgatjuk. Ha a kurzor az 1. sorban van a kurzort felfelé mozgató billentyü hatástalan. Ugyanugy hatástalan a kurzort balra mozgató billentyü is, ha a kurzor az 1 sor 1. pozíciójában van.

1.3.3. Szöveg módositása a képernyőn

Átirás

A képernyőn bármely karakter igen egyszerüen átirható. A kurzorral megfelelő pozicióra állunk és lenyomjuk a kivánt billentyüt. Függetlenül attól, hogy a kurzor helyén volt-e valami felirat, vagy sem, az uj feliratot az ujonnan lenyomott billentyü határozza meg.

Törlés

A szövegbe felesleges en bekerült karakterek törlése is igen egyszerüen megoldható a DEL funkció segitségével. /DEL az angol DELETE=töröl szó rövidítése /Ez az első sorban lévő Ø billentyü alsó felirata. Tehát a bal oldali SH és a Ø billentyüvel lehet aktivizálni. /Továbbiakban DEL billentyü / A karakter törlésnek két alapesete van:

1., A törlendő karakter után, ugyanabban a logikai sorban van még egy vagy több karakter. A kurzort rávisszük a törlendő karakterre és lenyomjuk a DEL billentyüt.

Hatására

- a kivánt karakter törlődik
- a kurzor változatlan pozicióban marad
- a törölt karaktertől jobbra lévő karakterek eggyel balra lépnek /a törölt karakter helye nem marad üresen/

2., A törlendő karakter a logikai sor utolsó karaktere

a., A kurzort rávisszük a törlendő karakterre, majd lenyomjuk a [DEL] billentyüt. Hatására
- a kivánt karakter törlődik
- a kurzor változatlan pozicióban marad,

b., A kurzort a törlendő karakter utáni pozicióra állitjuk, majd lenyomjuk a [DEL] billentyüt. Hatására
- a kivánt karakter törlődik
- a kurzor a törölt karakter helyére, eggyel balra lép.

Beszurás

A szövegből kifelejtett karakterek utólagos beirására szolgál az INS funkció /Az INS az angol INSERT=beszur szó rövidétése/. Ez a funkció ugyanahhoz a billentyühöz van rendelve, mint a DEL funkció, a [Ø] billentyü felirata. Tehát jobb oldali [SH] és a [Ø] billentyüvel lehet aktivizálni /továbbiakban [INS] billentyü/. Az [INS] billentyü segitségével helyet csinálhatunk a logikai sor belsejében az utólagosan beirandó karakterek számára.

A megfelelő számu hely kialakitása után beirjuk a kivánt karaktereket.

Az [INS] billentyü hatására

- a kurzortól balra lévő karakterek és a kurzor pozíciója változatlan marad
- az a karakter, amin a kurzor áll, és azok amelyek tőle jobbra vannak eggyel jobbra lépnek.

Ez mindaddig tart amig a jobbra lépkedő karakter /karakterek/ egy uj logikai sort el nem érnek. Ekkor az ezt megelőző logikai sorok egy fizikai sorral feljebb lépnek. Igy 40 karakterrel ujra bővithető az a logikai sor, amelyben helyet akarunk csinálni. Ha még ez is kevésnek bizonyul, akkor egy ujabb soremelés történik az előzőek szerint. Ez a folyamat addig tart, amig a kurzor a legfelső sorba nem ér.

2. Kalkulátor üzemmód

Az AIRCOMP-16 személyi számitógépek kalkulátor üzemmódját szokás parancs-üzemmódnak is nevezni. Ebben az üzemmódban a gép - a ROM memóriájában lévő BASIC INTERPRETER segitségével - BASIC nyelvü parancsok végrehajtására szolgál. Ebben a fejezetben még nem térünk ki részletesen a BASIC utasitásokra, velük kapcsolatban csupán a legszükségesebb információkat ismertetjük, amelyek a gép kezeléséhez feltétlenül szükségesek.

Egyelőre a billentyüzet megismerése és használatának begyakorlása a cél.
Az OK felirat megjelenése után lehetőségünk van utasitásokat kiadni. Ezek közül az első amivel megismerkedünk a PRINT /nyomtat/ utasitás.

2.1. Aritmetikai müveletek /operátorok/

A PRINT szó valamelyik SH és RUN BRK billentyük egyidejü lenyomásakor jelenik meg a képernyőn. A PRINT felirat után lehetőségünk van tetszőleges matematikai /aritmetikai/ kifejezést irni. A kifejezésben öt matematikai müveleti jel szerepelhet:

- hatványozás billentyü /a képernyőn csak jel látszik/
- szorzás SH 8 billentyük
- osztás SH 9 billentyük
- összeadás SH ; billentyük
- kivonás SH - billentyük

Ezeket a matematikai müveleteket szokás aritmetikai operátoroknak is nevezni.

A müveletek között prioritási sorrend van. A legerősebb müvelet a hatványozás, ezt követi azonos erősséggel a szorzás és az osztás, végül szintén azonos erősséggel az összeadás és kivonás.

Ez azt jelenti, hogy egy aritmetikai kifejezésben először a hatványozás, majd a szorzás és osztás, végül az összeadás és kivonás kerül végrehajtásra. Azonos erősségű müveletek esetén /pl.: két hatványozás vagy egy szorzás és egy osztás stb./ a művelet-végrehajtás balról jobbra történik /"balról-jobbra szabály"/.
Gépeljük be az alábbi utasitást
PRINT 2+8 * 5 ↑ 3
Ez egy karakterlánc tehát egy logikai sornak számit.
Az interpreternek a [CR] billentyü lenyomásával jelezzük, hogy a logikai sort befejeztük, nem akarjuk tovább irni.
A [CR] billentyü lenyomása után a PRINT alatt megjelenik a kifejezés értéke 1002. Tehát először a hatványozás majd a szorzás és legvégül az összeadás hajtódott végre:

$5^3 = 125$; $8 \cdot 125 = 1000$; $2+1000=1002$

Az interpreter mindig egy logikai sort értékel ki, mégpedig azt a logikai sort, amelyben a kurzor áll amikor a [CR] billentyüt lenyomjuk. Abban az esetben, ha a kurzor közvetlenül egy logikai sor utolsó karaktere után áll, akkor ezt a logikai sort értékeli ki Jelen esetben is ez fordult elő hiszen a kurzor a 3 után állt. A logikai sorban olyan jelsorozatnak kell állni, amit az interpreter értelmezni tud, ellenkező esetben ? SN ERROR hibaüzenetet ir ki.
/Hibakódokat lásd a függelékben; ERROR=hiba/

Addig amig a [CR] billentyüt le nem nyomjuk bármit irhatunk a képernyőre a számitógép nem reagál rá.
A [CR] billentyü lenyomásának hatására a kurzor uj fizikai sorra tér át, és kilép az előző logikai sorból és uj logikai sor megnyitására ad lehetőséget.

Megjegyzés:Két logikai sor összekapcsolható eggyé, ha a köztük lévő üres helyeket /ez nem azonos a [SPACE] billentyüvel beirt szóközzel/ valamilyen karakterrel feltöltjük /ez lehet SPACE is/.

Visszatérve az aritmetikai kifejezésre a müveletek közötti erősségi sorrendet zárójelek felhasználásával feloldhatjuk.
A zárójelek közül csak a kerek zárójelek használhatók.
A zárójel használatakor először mindig a zárójelben levő müveletek hajtódnak végre, ha a zárójelen belül több müvelete is van akkor itt is a prioritás szerint történik a müveletek végrehajtási sorrendje. A zárójelen belül lehetőség van ujabb zárójelek használatára. Ezt a zárójelek egymásba ágyazásának hivjuk. Ennél a géptipusnál a zárójelek egymásbaágyazási mélységének a logikai sor maximális hossza, az looo karakter szab határt.

Térjünk vissza példánkhoz és gépeljük be:

```
PRINT  (2+8) * 5  ↑ 3
```

A [CR] lenyomása után a PRINT alatt megjelenik az eredmény 1250. Most a müveletvégrehajtási sorrend a következő volt:

(2+8) = 10 /a zárójel miatt /: 5^3 = 125 /prioritás miatt/;
10 * 125 = 1250

Irjunk ujabb zárójelet a kifejezésbe
PRINT ((2+8) * 5) ↑ 3
Az eredmény 125000. Müveleti sorrend
(2+8)=10; 10 5 = 50 ; 50^3 = 125000
Két kiegészités a PRINT utasitáshoz
- a PRINT kulcsszó helyett használhatjuk a ? billentyüt
- a PRINT szó betünként is begépelhető /a betünkénti begépelés minden kulcszóra érvényes: a kulcszón belül nem szerepelhet szóköz/

2.2. Számok tárolása, kijelzése, beirása

Tárolás

A számokat a számitógép 7 értékes jeggyel tárolja, kijelzésénél azonban ezekből csak 6 jelenik meg, az utolsó jegy kerekitve. Az értéktelen nullák nem jelennek meg kijelzéskor A 7 értékes jegy tárolása azt jelenti például, hogy a számitógép számára az 1/3 = 0.3333333, ami természetesen matematikailag nem helyes. A számitógép véges memóriája ezen kivül még egyéb korlátokat is szab. Például a számitógépben tárolható legnagyobb szám kb. 10^{38}, a legkisebb -10^{38}, a nullától különböző legkisebb abszolutértékü szám pedig kb. $5 \cdot 10^{-39}$ értékü.

Kijelzés

Kijelzéskor a szám megjelenitése függ a szám értékétől. Ha a szám hat jegyre kerekitett értéke 0.1 vagy ettől nagyobb és 100000-től kisebb, akkor a kijelzés a szokásos módon történik. Ellenkező esetben a szám kijelzése normálalakban történik az alábbiak szerint, emEek, ahol:

e=előjel /mantissza előtt csak negativ számoknál, karakterisztika előtt mindig megjelenik/

m=mantissza értéke

E=állandó 10 hatványalapot jelöli

k=karakterisztika értéke /-39 ≤ k ≤ 38 /

Az emEek kifejezés matematikai alakja $em-10^{ek}$

Például gépeljük be:

? 1/80 a CR lenyomása után az eredmény

1.25E-2 alakban jelenik meg.

? 2300 * 4200 begépelése után pedig 9.66E+6 jelenik meg.

/A továbbiakban a CR billentyü lenyomását nem fogjuk külön jelölni/.

Az eddigi kifejezésekből is láthattuk, hogy a számitógépünk tizedes vessző helyett tizedes pontot használ. Erre gépeléskor is ügyeljünk. Csak a pont szolgál a tizedesek kijelölésére, a vesszőt különböző számok elválasztására használjuk. Ha egy szám értéke [0.1,0.1] zárt intervallumba esik akkor kijelzéskor a nulla elmarad. Pl: ? -5/6 begépelése után -.833333 kerül kijelzésre.

Beirás

Számok beirásánál nem kell figyelembe venni azt, hogy a gép csak az első hét értékes jegyet tárolja, egy szám beirása több jeggyel is történhet. Csak akkor kapunk hibajelzést, ha a beirt szám nagyobb mint 10^{38} vagy kisebb mint -10^{38}. Egytől kisebb abszolut értékü számok beirásakor a nulla egész beirása elhagyható. Lehetőségünk van normálalakban vagy a normálalakhoz hasonló formában is beirni számokat, ekkor a karakterisztika elé a + előjelet nem kötelező kitenni.

Pl. ? 1.2E4+52E4 begépelése után 532000 fog megjelenni a képernyőn.

2.3. Zsebszámológépekhez hasonlóan az AIRCOMP-16 személyi számitógépen is közvetlenül "hivhatók" a leggyakrabban használt aritmetikai függvények. Valamennyi függvény megtalálható valamelyik háromfunkciós billentyühöz rendelve. Természetesen itt is lehetőség van a függvény nevének karakterenkénti begépelésére. A függvények argumentumát mindig zárójelbe kell tenni. A függvények értelmezési tartományát szem előtt tartva a függvények argumentuma tetszőleges aritmetikai kifejezés lehet.
Az alábbiakban felsoroljuk az egyes függvények matematikai, illetve BASIC jelölését.

Matematikai jelölés	BASIC jelölés	Megjegyzés
/X/	ABS(X)	
arctgx	ATN(X)	Az eredményt radiánban kapjuk $\frac{\pi}{2}$ és $\frac{\pi}{2}$ között
cosx	COS(X)	Az argumentumot radiánban kell megadni
e^x	EXP(X)	
entierxvagy $[x]$	INT(X)	A legnagyobb egész szám, mely $\leq x$ /egész rész függvény/
lnx	LOG(X)	
-	RND(X)	Véletlen szám generáló *
signx	SGN(X)	SGN(X)=$\begin{cases} 1 & \text{ha } x > 0 \\ 0 & \text{ha } x = 0 \\ -1 & \text{ha } x < 0 \end{cases}$
sinx	SIN(X)	Az argumentumot radiánban kell megadni
$\sqrt{x}$	SQR(X)	
tgx	TAN(X)	Az argumentumot radiánban kell megadni.

* Ha $x < \phi$ akkor értéke /a véletlen szám/ $[\phi, x]$ intervallumból kerül ki.
Ha $x=\phi$ akkor azt a kiinduló értéket jelenti, amelyből $x < \phi$ esetén a véletlen számot generálja.
Ha $x > \phi$ akkor $|x|$ lesz az a kiinduló érték, amelyből $x > \phi$ esetén a véletlen számot generálja.

2.4. Memóriák

A számitógépben az adatok tárolására a memória szolgál. Egy számadat a memóriában 4 bájton kerül tárolásra. Továbbiakban ezt a memória-egységet rekesznek fogjuk nevezni. A rekeszeket egymástól nevük alapján lehet megkülönböztetni. Az AIRCOMP-16 BASIC nyelve a rekeszek megnevezésére vagy 1 betüt, vagy 2 betüt enged meg. Ha hosszabb nevet adunk, akkor a második betü utáni betüket figyelmen kivül hagyja. Az egyes rekeszeket feltölthetjük pl. értékadó utasitással. Az értékadó utasitás a rekesz nevéből, az = jelből és egy aritmetikai kifejezésből állhat. Pl. A=2 ↑3 * 4+6 utasitás begépelése után az A rekesz értéke 38 lesz. Az egyes rekeszek tartalma a PRINT utasitás segitségével jeleníthető meg a képernyőn. Folytatva az előző példát, a PRINT A utasitásra a képernyőn megjelenik a 38./Természetesen ? A ugyanezt eredményezi/.

A számitógépnek vannak olyan memória egységei amelyekben tetszőleges karakterek is tárolhatók, illetve olyanok amelyek több számadat tárolására használhatók. Ezek a szöveges, illetve indexes változók. Ezekről a későbbi fejezetekben lesz részletesebben szó. Most csak azt jegyezzük meg, hogy a memória egység elnevezése dönti el, hogy abban milyen adat tárolható. Ugyanaz a bájt szolgálhat számadat, tetszőleges karakter esetleg utasitás tárolására is. Tehát az egész RAM memória homogén felépitésü.

3. AircoMP-16 személyi számitógép programozása

Ebben a fejezetben BASIC nyelvü programok irásáról és azok futtatásáról lesz szó. Annyit már az előző fejezetekből is tudunk, hogy a számitógépünk - a ROM memóriában lévő interpreter segitségével - BASIC utasitások végrehajtására képes. A BASIC az angol "Beginner's All-purpose Symbolic Instruction Code" mondat szavainak kezdőbetüiből alkotott mozaikszó. A magyar megfelelője: kezdők általános célu szimbolikus utasitáskódja. A programozási nyelvek közül a BASIC az egyik legegyszerübb. A számitástechnikával ismerkedők is viszonylag gyorsan megtanulhatják a nyelv szabályait, és irhatnak BASIC nyelven programokat.

A BASIC interpreter lehetővé teszi a számitógép két üzemmódban való használatát. Az egyiket kalkulátor vagy parancs üzemmódnak - erről volt szó a 2. fejezetben - a másikat programfutási üzemmódnak nevezzük. Parancs üzemmódban lehetőségünk van BASIC parancsok kiadására, illetve programlépések beirására. A parancs is, a programlépés is - egy logikai sorban lévő - BASIC utasitásból ill. utasitásokból áll. Ha egy logikai sorba több utasitást is irunk, az egyes utasitásokat kettősponttal kell elválasztani. A parancs és a programlépés közti különbséget az jelenti, hogy a logikai sor előtt van-e sorszám vagy ninçs.

Azt a logikai sort, amely előtt nincs sorszám parancsnak, amely előtt van, programlépésnek nevezzük.

A legegyszerűbb esetben 1 logikai sorban 1 BASIC utasitás van. Az interpreter a parancsot azonnal végrehajtja és nem tárolja a memóriában. A programlépést az interpreter tárolja, és csak külön parancsra hajtja végre. Ennek a parancsnak a kiadása a gépet parancsüzemmódról program-futási üzemmódra állitja át.
A 2. fejezetben használt BASIC utasitások /kiiró és értékadó/ előtt nem volt sorszám, tehát azok parancsok voltak. Ebben a fejezetben a sorszámmal ellátott utasitásokról azaz a programlépésekről lesz szó. A program egy vagy több programlépésből /programsorból/ áll. Minden programlépést uj logikai sorba kell irni. Emlékeztetőül: a logikai sor véget ér, ha lenyomjuk a [CR] billentyüt. Az interpreter csak azokat a logikai sorokat értékeli ki, amelyeknél a kurzor a megfelelő pozicióban állt amikor lenyomtuk a [CR] billentyüt. /A megfelelő pozició a logikai sor bármely karakterét, illetve az utolsó karakter utáni pozicit jelenti/.

3.1. BASIC programok szerkezete

Azt már tudjuk, hogy egy program programlépésekből a programlépés, sorszámmal ellátott utasitásokat tartalmazó logikai sorból áll. A sorszámokról viszont még nem tudunk semmit. Ezért nézzük ezeket! Valamennyi BASIC nyelv a nem negativ egész számokat engedi meg sorszámoknak. Az egyes változatok a legnagyobb sorszám értékében különböznek, az AIRCOMP-nál ez 32759.

Tehát ha a sorszámot k-val jelöljük, akkor mindig teljesülnie kell a $0 \leq k \leq 32759$ egyenlőtlenségnek. A sorszámokra más megkötés nincs. A sorszámoknak a program végrehajtása során kettős szerepük van: egyrészt előirják az utasitások tárolásának és végrehajtásának sorrendjét, másrészt az utasitásokra való hivatkozást teszik lehetővé.

A programsorok végrehajtásának sorrendje általában az utasitások sorszámának növekvő sorrendjével egyezik meg. Ezt a természetes utasitás-végrehajtási sorrendet bizonyos utasitások megváltoztathatják. A programsorok begépelési sorrendje tetszőleges, az interpreter ezeket sorszám szerint növekvő sorrendbe rendezi és tárolja a memóriában.
A memóriában lévő programot a LIST paranccsal irhatjuk ki a képernyőre. A LIST parancs kiadása után a programsorok olyan sorrendben kerülnek a képernyőre mint ahogy a memóriában vannak
Az itt elmondottakat egy példával szemléltetjük.
Gépeljük be az alábbi fiktiv programsorokat

```
1[illegible]
38 U2
13 U3
20 U4
```

A LIST parancs után a képernyőn sorszám szerint rendezve jelenik meg a program.

```
13 U3
1[illegible] U1
20 U4
38 U2
```

3.2. BASIC programok javitása

a., Uj programsor irása

Uj programsor beépitése igen egyszerüen történhet. Miután eldöntöttük, hogy a program melyik két utasitása közé akarjuk ezt az uj utasitást beirni, választunk egy olyan sorszámot, amely a két utasitás sorszáma közé esik, és ezzel a sorszámmal begépeljük a szükséges programlépést. Természetesen megfelelő sorszám adásával az uj programlépés lehet a program első, illetve utolsó sora is.
Két programsor közé csak akkor nem tudunk uj sort beirni, ha a két sorszám egymást követő egész szám. Ilyenkor a program egy részét esetleg az egészet át kell sorszámozni, hogy az uj sort be tudjuk épiteni a programba. Ez az AIRCOMP-16 gépnél elég hosszadalmas. Ezért célszerü a sorszámokat tizesével növelni. Igy bármely két sor közé ujabb 9 sor irható.
Folytatva az előző példánkat, gépeljük be:
14 U5
A LIST parance után
13 U3
14 U5
15 U1
20 U3
38 U2
sorok jelennek meg a képernyőn. Tehát a 14 sorszám alapján az U5 utasitás beépült a programba. A program listájából az is megállapitható, hogy az U3, U5 és U1 sorok közé már nem irható be ujabb sor.

b./ Programsorban történő javitás

Ha valamely programban javitást akarunk végezni, akkor ehhez a szövegszerkezstési fejezetben megismert módok állnak rendelkezésre. Nevezetesen: átirás, törlés, beszurás. Hogy ezt végre tudjuk hajtani, szükség van arra, hogy a javitandó sor a képrnyőn látható legyen. A képrnyőn bárhol lehet ez a sor, az sem jelent problémát, ha a sor több helyen is szerepel. Ezek közül kiválasztunk egyet és abban végrehajtjuk a kivánt módositást. A módositás után a CR billentyüvel lépjünk ki a programsorból, ellenkező esetben a módositás csak a képernyőn történik meg, a memóriában nem. Ha a CR billentyüt nem nyomjuk le az interpreter nem "érzékeli" a módositást és igy a programsort sem javitja át.

Javitsuk át a 15 sorban szereplő U betüt T betüre!

Ha a 15 sor látható a képernyőn, akkor a kurzorral ráállunk a 15 sor U betüjére és lenyomjuk a T betüt majd a CR billentyüt /Ha a 15 sor nincs a képernyőn akkor először egy LIST parancsot kell kiadni!/

A javitás utáni LIST parancsot követően a programunk a következő lesz

```
13U3

14U5

15T1

20U4

38U2
```

c., Programsor átirás

Lehetőségünk van arra is, hogy egy egész programsort átírjunk. Erre a legegyszerűbb mód az, hogy a régi programlépés sorszámát újra felhasználjuk. Ekkor az utólag beirt programlépés kicserélődik a régivel. Tehát ha azonos sorszámokat használunk a program beirásakor, akkor mindig az utolsónak beirt programlépés marad érvényben.
Pl.: a 14A begépelése, valamint a LIST parancs kiadása után a következő jelenik meg a képernyőn.

13U3
14A
15T1
20U4
38U2

d., Programsor törlése

Programsort ugy törölhetünk, hogy leirjuk a törlendő programsor sorszámát, és utána lenyomjuk a CR billentyüt.
/Programsort a DEL ill SPACE billentyükkel csak a képernyőről lehet törölni, a memóriában nem/
Töröljük ki a 20 sorszámu programsort.
20 CR

Ellenőrizve a LIST parancsal!
13U3
14A
15T1
38U2

4. BASIC programnyelv utasitásai

Ebben a fejezetben soroljuk fel azokat a BASIC nyelvü utasitásokat, amelyeket a gép végre tud hajtani. A felsorolt utasitások sorrendjénél azt tartottuk szem előtt, hogy felhasználásukkal minél előbb működőképes programot lehessen irni és futtatni.

4.1, A LIST utasitás

A LIST utasitás parancsként és programlépésként egyaránt kiadható. Arra szolgál, hogy a programot vagy annak egy részét a képernyőn megjelenitsük.

A LIST parancs általános alakja:

LIST n-m

ahol LIST=az utasitás kulcsszava

n=nem negativ egész szám; n-nel egyenlő, illetve n-től nagyobb sorszámu utasitások képernyőre irását engedi meg.

-=kötőjel/negativ előjel/

m=nem negativ egész szám; m-mel egyenlő, illetve m-től kisebb sorszámu utasitások képernyőre irását engedi meg.

A LIST n-m utasitás azt jelenti, hogy a program azon lépései kivételével, amelyeknek sorszáma kisebb mint n, illetve nagyobb mint m a képernyőn megjelenjenek. Tehát azok a k sorszámu programlépések jelenjenek meg a képernyőn melyre az $n \leq k \leq m$ egyenlőtlenség teljesül.

A LIST utasitásból az n és az m elmaradhat. Ha az n hiányzik, akkor a kiirás az első programsortól kezdődik. Ha az m hiányzik, akkor a kiirás az utolsó programsorig tart. Ha mindkét szám hiányzik akkor „-" jel is elmaradhat, ilyenkor a program valamennyi sora kiirásra kerül.
A LIST l utasitás az l.sor kiirását eredményezi.
Példák: Azok a k sorszámu utasitások kerülnek kiirásra amelyekre:

LIST n- n ≤ k
LIST -m k ≤ m
LIST n k = n relációk teljesülnek.

LIST- }
LIST } kilistázza az egész programot

4.2. NEW utasitás

A NEW /Uj/ utasitás parancsként és programlépésként is kiadható. Ezzel az utasitással lehet a memóriában lévő programot kitörölni. Uj program irása előtt célszerü mindig kiadni. Az utasitás általános alakja: NEW

4.3. RUN, BREAK, CONT utasitások

RUN /fut/ utasitás parancsként és programlépésként is használható. A RUN utasitás végrehajtása után a számitógép parancs-üzemmódból program-futási üzemmódba kerül. Ezután a gép müködését a program határozza meg. A RUN utasitás a

RUN
BRK billentyü lenyomásával, vagy a kulcsszó karakterenkénti begépelésével adható ki.

Mint azt már a billentyüzet ismertetésekor emlitettük, a RUN BRK speciális kétfunkciós billentyü. Ez abból is kedcrül, hogy lenyomásakor nem a BRK, hanem a RUN felirat jelenik meg. SH és RUN BRK lemyomásakor pedig nem a RUN, hanem a PRINT felirat, függetlenül attól melyik SH billentyüvel együtt nyomjuk le.

A RUN BRK billentyü SH nélküli lenyomásával a számitógép üzemmódjait tudjuk váltogatni.

1., Ha a gép parancs-üzemmódban van és lenyomjuk a RUN BRK billentyüt, akkor a RUN felirat jelenik meg és a CR lenyomása után a számitógép programfutási üzemmódba kerül.

2., Ha a gép programfutási-üzemmódban van és lenyomjuk a RUN BRK billentyüt, akkor általában a képernyőn megjelenik a BREAK /törni, megszakitani/ felirat és alatta az a programsor, ahol a megszakitás történt. Ezzel a számitógép parancs üzemmódba került. /Programfutási üzemmódból csak akkor nem tudunk a RUN BRK billentyüvel parancsüzemmódra váltani, ha INPUT utasitás végrehajtása miatt adatra vár a gép,
/lásd INPUT utasitás/.

Ha a RUN BRK billentyüvel megszakitottuk a programot, akkor a CONT/CONTINUE= folytat röviditése/ utasitással a program ott folytatható, ahol megszakadt. A program megszakitása után lehetőség van arra, hogy a számitógépet parancsüzemmódban használjuk.

/Pl.: Megnézhetjük a programban szereplő változók értékét; megváltoztathatjuk ezeket; listát kérhetünk a programról vagy annak egy részéről /A CONT parancs kiadása után a program úgy folytatódik mintha nem szakitottuk volna meg. Nem folytatható a programfutás ha parancsüzemmódban:

1. belejavitunk a programba
2. hibás parancsot adunk ki

Mindkét esetben a CONT parancsra? CN ERROR hibajelzést kapunk. Ilyenkor a program általában csak elölről RUN-nal indítható.
A CONT utasitás csak parancsként használható.

4.4. Változók, értékadó utasitások

Az értékadó utasitások segitségével adatokat tárolhatunk a számitógép memóriaegységeiben. A memóriaegységeket változóknak nevezzük. A változóknak különböző tipusai vannak, attól függően, hogy milyen jellegü adatok tárolására használják.

4.4.1. Változó-tipusok

Index nélküli, vagy skalár változók

A skalár változóknak két tipusa van az egyik a numerikus a másik karakteres /szöveges/ változó. A numerikus változóban csak numerikus adatot lehet tárolni. A karakteres változóban tetszőleges karaktereket /számjegyeket is/.

A numerikus változó minden számadatot - a megengedett korlátok között - 4 bájtban tárolja. A szöveges változó minden adatot karakterenként tárol, minden karaktert egy bájtban. Egy változó maximum 255 karaktert tárolhat. A karakteres változót a név után irt $ jellel különböztetjük meg a numerikus változótól. Tehát a D numerikus a D$ karakteres változó, és D ≠ D$ Mint már korábban emlitettük ennél a géptipusnál egy változó neve legfeljebb két betü lehet. Ha ettől hosszabb nevet adunk, akkor is csak az első két betüt használja fel a gép a változó azonositására, a többit figyelmen kivül hagyja. A változók jelölését az alábbiakkal egészitjük ki.

1., Nem lehetnek változónevek a speciális változók. /DL, CR, GL, HM/, valamint a kétbetüs kulcsszavak. /pl. ON, OR, stb...../

2., Ha egy változónévben szerepel a $ jel ezt az interpreter akkor is figyelembe veszi, ha az az első betü után bárhol van. Pl: ABCD változó megegyezik az AB változóval; ABCD$=AB$:, A$BCD=A$; ABCD=A$

3., A változó nevében kulcsszó akkor sem szerepelhet, ha az kettőnél több betüből áll. Pl. SZORZAT, TREMOLKARUNK stb.

A betükön és a $ jelen kivül más karaktert nem tartalmazhat a változónév.

Indexes változók /Részletes ismertetését lásd a DIM utasitánál!/
A számitástechnikában az indexes változókat tömböknek nevezzük. Az ARCOMP-16 BASIC változatában egy- és kétindexes változókat használhatunk. Az indexes változóknak is két tipusa van: karakteres és numerikus.
Az indexes változók nevére ugyanaz a megkötés érvényes, mint a skalár változók nevére.

4.4.2. Értékadó utasitások

Az értékadó utasitás parancsként és programlépésként egyaránt használható. Általános alakja V=A, ahol:

- V = megengedett változónév
- A = változó tipusának megfelelő kifejezés

Ha V numerikus változó, akkor A aritmerikai kifejezés.
Ha V karakteres változó, akkor A string kifejezés /Részletesen lásd 4.10. fejezetben/.
Az értékadó utasitás nem egyenlőség, ezért az egyenlőség jobb és bal oldala nem cserélhető fel. Az utasitás végrehajtása ugy történik, hogy először kiszámitódik a jobb oldali kifejezés értéke, majd ez az érték a bal oldali változóba kerül. Ez alapján az is megengedett, hogy ugyanaz a változó az értékadó utasitás jobb és bal oldalán egyaránt szerepeljen.

A BASIC interpreter minden numerikus változóhoz kezdeti értékként nullát rendel.
Egy változó mindaddig megtartja korábbi értékét, amig

1., értékadó utasitás bal oldalán nem szerepel
2., INPUT utasitás listáján nem szerepel /lásd INPUT utasitás/
3., egy uj programlépést be nem gépeltünk
4., NEW vagy CLR utasitást ki nem adtunk /CLR a CLEAR-tisztit röviditése. Lásd 4.8.2. pontban/
5., a gépet ki nem kapcsoljuk.

A 3.-4.-5. pontokban foglaltak minden numerikus változót nullázнak. Példaként irjuk be az alábbi értékadó utasitásokat:

```
A=32
BB=A ↑ 2-SQR (400)
A=A+1
```

Az első utasitás végrehajtása után az A változó értéke 32 lesz. A második utasitás végrehajtása után a BB értéke 1004 lesz. A harmadik utasitás végrehajtása után az A értéke eggyel megnő, igy 33 lesz.

4.5.1. INPUT utasitás

Az INPUT utasitás segitségével lehetőségünk van programfutási üzemmódban a billentyüzetről adatokat megadni a programban szereplő változóknak. /Ebben a részben az INPUT utasitásnak csak azt a változatát ismertetjük, amikor az adatmegadás billentyüzetről történik/. Ezzel elérhetjük azt, hogy a program javitása nélkül, más adatokra ugyanazt a feladatot többször végrehajtassuk a géppel. Az INPUT utasitás parancsként és programlépésként is kiadható.

Általános alakja:

INPUT lista

-INPUT az utasitás kulcsszava

- lista tetszőleges sorrendben tetszőleges számu változó nevet, idézőjelbe tett szöveget[X] és konstansokból álló aritmetikai kifejezést tartalmazhat.

X Megjegyzés: [SH] billentyüt [2] billentyüvel lenyomva idézőjelt generálunk. A [CR] billentyü kivételével bármely billentyün lévő karakter vagy karakterlánc idézőjelbe tehető. Az idézőjelbe tett szöveg három utasitásnál használható, ezek INPUT, PRINT és olyan értékadó utasitásmód, amelynek bal oldalán karakter tipusu változó áll.
Az INPUT utasitásban használt idézőjeles szöveget az interpreter az utasitás végrehajtásakor általában változtatás nélkül kinyomtatja a képernyőre. Kivételt képeznek azok az esetek, amelyeknél az idézőjelbe kurzorvezérlő billentyükkel irtunk. Idézőjel megnyitása után a kurzorvezérlő billentyük nem végzik el a kurzor vezérlését, hanem minden kurzorvezérlő billentyü helyett egy-egy karakter iródik az idézőjelbe. A kurzorvezérlést ezen karakterek alapján az INPUT illetve a PRINT utasitások végrehajtásakor végzik el. Tekintsük át ezeket a billentyüket és az idézőjelben megjelenő nekik megfelelő karaktereket:

Billentyü	Billentyü hatása idézőjelen kivül	Idézőjelben megjelenő karakter
[⇆]	kurzort jobbra mozgató	→
[SH] és [⇆]	kurzort balra mozgató	←
[↕]	kurzort felfelé mozgató	↑
[SH] és [↕]	kurzort lefelé mozgató	↓
[INS]	"helycsináló"	>
[DEL]	törlő	<
[SH] és [CR]	képernyőtörlés	□

A listában lévő elemek elválasztására a pontosvessző és a vessző szolgál. Ha a listában szereplő egymást követő elemek megkülönböztethetők, akkor a pontosvesszőt nem kötelező kitenni. /Például: az INPUT "A="; A és INPUT "A="A utasitás ugyanazt jelenti/.

Elválasztójel a lista első eleme előtt és az utolsó eleme után is állhat. Az utasitás végrehajtásakor az idézőjelbe tett szöveg, valamint az aritmetikai kifejezés értéke megjelenik a képernyőn. A változó név hatására a programfutás megszakad és csak megfelelő számu adat megadása után folytatódik. A begépelt adatok lesznek a listában szereplő változók értékei. Az adat numerikus változó esetén tetszőleges aritmetikai kifejezés lehet. Ez lényeges eltérés a többi géptipusnál alkalmazott BASIC váltzatoktól. A listában szereplő valamennyi változónak uj sorban kell megadni a szükséges adatot. Ez azt jelenti, hogy minden adatbeirása után le kell nyomni a [CR] billentyüt!

Az elválasztójelek szerepe függ a listaelemek sorrendjétől. Ezt foglaljuk össze az alábbiakban.

1., A változónév a lista első eleme, és nincs előtte elválasztójel, vagy előtte szintén változónév van, és az elválasztójel vessző. Ekkor az utasitás végrehajtásakor egy uj fizikai sor első poziciójában egy kérdőjel jelenik meg, a kurzor pedig ugyanennek a sornak a 3. poziciójára kerül. Ettől a poziciótól kezdve van lehetőség az adat megadására. A 2. pozició logikai sorelválasztó. Ezt tilos fölülirni. Ugyanis azt az adatot fogja átadni az interpreter a változónak, amely ugyanabban a logikai sorban van, ahol a kurzor, a [CR] billentyü lenyomásakor.

2., A lista első eleme változónév és van előtte elválasztó jel. Ebben az esetben a ? nem kerül kiirásra. A változó előtt több elválasztójel is lehet. Ha az elválasztójel pontosvessző, akkor az uj sor első poziciójába logikai sorelválasztó jel kerül, a második poziciójába pedig a kurzor. Ez a pontosvesszők számától független. Ha az elválasztójel a vessző, akkor egy vessző esetén a kurzor az uj sor tizedik poziciójába, minden további vessző esetén 8 pozicióval jobbra lép. Ha az adott sorban már nincs elég hely akkor áttér a következő sorra. Tehát a kurzor szóba jöhető poziciói:

1 sor: 10., 18., 26., 34.

2 sor: 2., 18., 26., 34., stb.

3., A lista első eleme nem változónév
Ekkor egy uj fizikai sorban kiiródik vagy a szöveg, vagy az aritmetikai kifejezés értéke. A kurzor poziciója pedig attól függ, hogy az elválasztójel vessző vagy pontosvessző. Pontosvessző esetén az utolsó kiirt karakter után a logikai sorelválasztó jelet követi a kurzor. A pontosvesszők számától függetlenül. Vessző esetén a 2. pontban foglaltak szerint változik a kurzor poziciója, figyelembe véve az előtte lévő kiirási képet. A kiirási kép és a kurzor között legalább egy logikai sorelválasztó karakternek kell lennie. A szóba jöhető kurzor poziciók a 2. pontban leirtak lehetnek.

4., A változónév után nem változónév következik.
Ekkor az első elválasztójel / vessző és a pontosvessző esetén egyaránt/ utáni kifejezés vagy szöveg a következő sor 1. poziciójától kezdődően jelenik meg a képernyőn. Ha több elválasztójel van a két listaelem között, akkor a nyomtatási képet csak a vesszők befolyásolják. Mégpedig a második vesszőtől ugy, ahogy a 2., 3. pontban ismertettük.
Ha az INPUT utasitást programlépésként használják, és nem adunk meg adatot ugy nyomjuk le a CR billentyüt, akkor a programfutási üzemmódból a gép parancsüzemmódba kerül. Ilyenkor általában a program nem folytatható, csak ujra indítható. /Lezárja a megnyitott FOR ciklusokat, GOSUB-okat,/

Példák INPUT utasitásokra

Utasitás	Megjegyzés a végrehajtáshoz
INPUT A,B,C,D	Az 1. 2., és 4. sor kezdődik kérdőjellel
INPUT A	A kurzor a 18. pozicióra áll
INPUT "A="; A, "B="B	Az A= és a B= két egymást követő sor elején jelenik meg. A kurzor és az egyenlőségjel között egy pozició marad "üresen" /logikai sorelválasztó/
INPUT "A=",A; "B="B	Az A=és a B= két egymást követő sor elején jelenik meg. Az elsősorban a kurzor a 10. a második sorban a 18. pozicióba kerül.

4.5.2. READ, DATA, RESTORE utasitás

Ez az utasitás-csoport csak programlépésként használható, parancsként nem. /READ=olvas; DATA=adat; RESTORE=visszaállit/ A három utasitás egymással szoros kapcsolatban van. Önállóan egyik sem használható. Az utasitások segitségével a programban szereplő változóknak értéket adhatunk.

READ utasitás általános alakja

k READ V

, ahol

-k az utasitás sorszáma
-READ a kulcsszó
-V változó lista

A változó listában különböző tipusu változók szerepelhetnek, egymástól vesszővel elválasztva. A listában szereplő változók az utasitás végrehajtásakor értéket kapnak.

DATA utasitás általános alakja

k DATA A

,ahol

-k utasitás sorszáma

-DATA kulcsszó

-A=adatlista

Az adat listában különböző aritmetikai és STRING kifejezések lehetnek. A STRING-ekről későbbi fejezetekben lesz szó, ezért most csak azokat az eseteket tárgyaljuk, amikor a változók numerikus tipusuak, az adatlistában pedig csak aritmetikai kifejezések vannak. Az adatlistában szereplő kifejezések értékét READ utasitásban szereplő változók kapják meg. A legegyszerűbb esetekben az aritmetikai kifejezések konstansok. A listaelemek elválasztására csak a vesszőt használhatjuk.

A READ utasitásban lévő változók az alábbiak szerint kapják meg a DATA utasitásban szereplő kifejezések értékét.

A DATA utasitásokban lévő kifejezések mindegyikéhez hozzá van rendelve egy pozitiv egész szám. A hozzárendelés először a DATA utasitások sorszáma másodszor a DATA utasitás listájában elfoglalt helyük szerint történik. A legkisebb számot - az egyest - a legkisebb sorszámu DATA utasitás első kifejezése kapja; a kettest ugyanebben a listában szereplő második kifejezés. Ha nincs második kifejezés, akkor nagyságszerint a következő sorszámu DATA utasitás listájában szereplő első kifejezés lesz a kettes számu. A legmagasabb számot a legnagyobb sorszámu DATA utasitás listájában szereplő utolsó kifejezés kapja.

Tehát minden adathoz hozzárendeltünk egy sorszámot. Ezután már könnyü lesz meghatározni az egyes változókhoz rendelt értéket. Ehhez felhasználunk egy adatszámlálót /segédváltozót/, melynek értéke mindig a soron következő adat sorszáma. Ennek a változónak a kezdeti értéke 1, és minden READ utasitás végrehajtása után eggyel nagyobb lesz. Pontosabban; értéke akkor nő meg eggyel, ha a READ utasitásban szereplő változó megkapta azt az értéket, amelynek sorszáma megegyezett a számláló értékével.

RESTORE utasitás

A segédváltozó értékét a RESTORE utasitással lehet módositani. Az AIRCOMP 16 gép BASIC változata csak igen kis változtatást tesz lehetővé. A RESTORE utasitás végrehajtásakor a segédváltozó értéke 1 lesz függetlenül attól, hogy mi volt korábban az értéke.

Az utasitás általános alakja:

k RESTORE

A RESTORE utasitás tetszőleges sokszor szerepelhet egy programban.

Ezen utasitások használatánál két hibalehetőség van:

1., A változó és az adat különböző tipus.

2., A segédváltozó értéke meghaladja az adatok számát

Példa a fenti utasitások használatára

```
10 READ A
20 DATA 35, SQR (43+21)
30 DATA A/7
40 READ X, B
50 DATA X+B, 18
60 READ L
```

A program alapján az egyes változók értéke
A=35; X=8; B=7; L=15
Javitsunk bele a programba! Irjuk be az alábbi két programsort:
15 RESTORE
55 RESTORE
A változók értéke:
A=35; X=35; B=8; L=35

4.5.3. PRINT utasitás

A PRINT utasitással a 2. fejezetben már találkoztunk. Az ott beirtakból megtudhattuk, hogy különböző értékek kiirására használjuk, és a kérdőjellel helyettesithetjük. Ebben a részben kiegészitjük, illetve pontositjuk eddigi ismereteinket.
A PRINT utasitás parancsként és programlépésként is használható. Általános alakja:
PRINT lista vagy ? lista
, ahol
-PRINT=az utasitás kulcsszava
- ? = az utasitás röviditése
- lista tetszőleges sorrendben, tetszőleges számu, idézőjelbe tett szöveget, aritmetikai és string kifejezést tartalmazhat, de el is maradhat.

A listában szereplő változók elválasztási módjai megegyezhek az INPUT utasitásnál megismert elválasztási modokkal. /Vessző vagy pontosvessző lehet; a pontosvessző bizonyos esetekben elhagyható./

Két lista elem között több elválasztó jel is lehet, de ez csak vessző esetén befolyásolja a nyomtatási képet. Az idézőjelbe tett kifejezések általában változatlanul jelennek meg a képernyőn. Ezalól azok a kifejezések kivételek, melyekben kurzorvezérlő billentyü is szerepel. Ebben az esetben a nyomtatási kép az INPUT utasitásnál ismertetett módon változik.

Aritmetikai kifejezések esetén a PRINT utasitás az aritmetikai kifejezés értékét irja a képernyőre. Számok kiirásánál a következőt kell ismerni:

1., a számok előjelét csak negativ számok esetén irja ki, pozitiv számok esetén az előjel helye "üresen" marad (SPACE karakter)

2., minden szám után egy számelválasztó jelet ir ki (logikai sorvégjel)

Tehát a 314 kinyomtatásához 5 pozicióra van szükség (1 előjel + 1 számelválasztójel+3 számjegy = 5 karakter)

A képernyő minden fizikai sora 5 szektorra van osztva. Minden szektor 8 karakter hosszuságu. Az egyes szektorok rendre a : 1., 9., 17., 25., 33. pozicióknál kezdődnek.

A szektoronkénti kiirás a listaelemeket elválasztó vesszővel szabályozható. Minden egyes vessző érzékelésekor az interpreter a kurzort a legközelebbi üres szektor első pozíciójára viszi. Ha egy szektor utolsó pozíciójába nem logikai sorelválasztójel kerül kinyomtatásra, akkor a következő szektort nem tekinti szabadnak.

Ez azt jelenti, hogy két karakter tipusu listaelemet csak pontosvesszővel lehet közvetlenül egymásután kinyomtatni. Ha a kiirási kép meghaladja az 5 szektor hosszuságot, akkor a nyomtatás automatikusan a következő sor megfelelő szektoraiban folytatódik. A lista nélküli PRINT utasitás egy üres sor kinyomtatását eredményezi.
Ha egy programban vagy egy parancsban több PRINT utasitás is szerepel, akkor a lista utolsó elemét követő elválasztójelet a soron következő PRINT utasitás figyelembe veszi. Pontosvesszó esetén a nyomtatást közvetlenül a korábbi kiirás után folytatja. Vessző esetén áttér a soron következő üres szektor elejére. Ha nincs elválasztójel az utolsó elem után, akkor a következő PRINT utasitás uj sor megkezdését jelenti. Gyakorlásként gépeljük be az alábbi parancsokat és elemezzük a történteket!

```
? "A" ; "B"
? "A" "B"
? "A", "B"
? 4; 3
? 4, 3
? , 4,,,,3
? , 4;,,,;3
? 1 2 3 4 5.6,7
? "12345.6";7
? 4:?3
?4,:?3
? 4; : ?3
? 4 :?:?3
? "x1="13, "x2="; 39
```

Példa: Az eddig megismert utasitások felhasználásával irjunk programot, amely két szám összegét meghatározza! A beirás után futtassuk a programot!

Megoldás:

```
10 INPUT "ELOE SZAAM=?"; A, "MAASODIK SZAAM=?"B
20 D=A+B
30 ? "A KEET SZAAM ÖSSZEGE="D
```

A program begépelése után a LIST parancssal ellenőrizzük, hogy azt gépeltük-e be, amit akartunk. A parancs végrehajtása után tapasztalhatjuk, hogy a 30-as sorban a ? rövidités helyett a PRINT kulcsszó szerepel. A LIST után a számitógép a ?-et mindig PRINT-re javitja.
Miután meggyőződtünk, hogy a programunk hibátlan, inditsuk el a RUN paranccsal. A RUN hatására gépünk programfutási üzemmódba került és végrehajtja a 10-es programlépést. Kiirja az INPUT utasitásban szereplő első szöveget az A változó nevéig és a kurzor megjelenésével jelzi, hogy adatra vár. Nyomjuk le az [5] billentyüt, majd a [CR] - t. Ennek hatására az INPUT utasitásban szereplő második idézőjelbe tett szöveget irja ki, és ujra adatra vár. Második adatnak 47-et adjunk meg! A [CR] billentyü lenyomása után hajtódik végre a 20-as, majd ez után a 30-as utasitás. A 30-as utasitás hatására megjelenik a PRINT listában szereplő szöveg és az eredmény. Az OK felirat jelzi, hogy a számitógép ujra parancs üzemmódba került. Természetesen ujabb RUN parancs után más adatokra is müködik a program. Ha kiiráskor az eredményt el akarjuk választani az INPUT soroktól, akkor javitsunk bele a programba!

```
25 PRINT
```

4.6. GOTO utasitás, kiegészités RUN utasitáshoz

4.6.1. GOTO utasitás

A GOTO /menj/ utasitás segitségével elérhetjük, hogy egy programban eltérjünk a programsorok végrehajtásának természetes sorrendjétől. Segitségével kijelölhetjük a soron következő végrehajtandó utasitást. A GOTO parancsként és programlépésként egyaránt használható. Általános alakja:

GOTO N

, ahol

-GOTO az utasitás kulcsszava

-N tetszőleges matematikai kifejezés

A matematikai kifejezés értéke, határozza meg a soron következő programképés sorszámát. Tehát a GOTO után ennek a programlépésnek az utasitásait hajtja végre a gép.

Az N matematikai kifejezés értékének meghatározása eltér a hagyományos kiszámitási módtól.

1., Ha a matematikai kifejezés számmal kezdődik, akkor értéke ezen szám egész része lesz.

2., Ha a matematikai kifejezés nem számmal kezdődik, akkor a kifejezés értéke a tényleges érték egész része lesz.

Ebben az esetben gyakran lép fel hibajelzés tárolási és számolási pontatlanság miatt. Ez ellen ugy védekezhetünk, hogy a kifejezéshez hozzáadunk egy kis pozitiv számot, pl.: 0.1-et,

Leggyakrabban az N matematikai kifejezés egy számkonstans.

Ha az N aritmetikai kifejezés értékével nincs azonos sorszámu programlépés akkor a GOTO N ? US ERROR hibajelzést, ha értéke negativ, akkor a GOTON ? IQ ERROR hibajelzést eredményez.

Példák GOTO utasitásokra:

GOTO 300	A program a 300 programsornál folytatódik.
GOTO 250+50	A program a 250 programsornál folytatódik.
GOTO (250+50)	A program a 300 programsornál folytatódik.
GOTO A+250+50	A program a 300 programsornál folytatódik (A=Ø)
GOTO SQR (100)+20	A program a 30 programsornál folytatódik.
GOTO SQR (400)+12	A program a 31 programsornál folytatódik (számolási pontatlanság miatt)
GOTO SQR (400)+12+0.1	A program a 32 programsornál folytatódik.
GOTO 45.5 * 2+9	A program a 45. programsornál folytatódik.

4.6.2. RUN utasitás

4.3. pontban ismertetett RUN utasitást az alábbiakkal egészitjük ki.

Az utasitás általános alakja:

RUN N

, ahol N a GOTO utasitásnál ismertetett matematikai kifejezés /ez el is maradhat/

A kifejezés kiszámitási módja megegyezik a 4.6.1. pontban ismertetett kiszámitási móddal. A kifejezés helytelen értékei is ugyanazokat a hibajelzéseket eredményezik.

A GOTO N és a RUN N utasitások csak egy dologban térnek el egymástól. A RUN N utasitás végrehajtásakor valamennyi változó értéke nulla lesz. A GOTO N utasitás a változók értékét nem módositja. Ha a RUN után nem szerepel matematikai kifejezés, akkor ez a legkisebb sorszámu programlépés kijelölését jelenti. A GOTO után mindig kell lenni matematikai kifejezésnek.

4.7. Feltételes utasitás

Feltételes utasitásoknak azokat az utasitásokat nevezzük, amelyek végrehajtása bizonyos feltételtől függ.

4.7.1. Logikai kifejezések

Relációjelek

Ha két aritmetikai kifejezést összekapcsolnak relációjellel, akkor logikai kifejezéshez jutnak. A BASIC nyelv az alábbi relációjeleket használja:

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

Ez a BASIC változat a logikai kifejezések értékéhez számot rendel. Ha a kifejezés értéke igaz, akkor 1-et, ha nem, akkor Ø-t. Igy bármely logikai kifejezés tárolható numerikus változóban.

Például a
B= 15:A=3 < 7:B=3 > 7 parancs után A értéke 1, B értéke Ø lesz.

Logikai müveletek

A BASIC nyelvben is megtaláljuk a szokásos logikai müveleteket:

a konjunkciónak az AND /és/
a diszjunkciónak az OR /vagy/
a negációnak a NOT /nem/

kulcsszó felel meg.

Az AIRCOMP-16 számitógépre vonatkoztatott BASIC nyelv megengedi, hogy logikai kifejezéseken kivül, aritmetikai kifejezéseket is összakapcsolhassunk logikai müveletekkel. A logikai kifejezések összekapcsolásakor a keletkezett logikai kifejezés értéke megegyezik a szokásos értékkel. Numerikus kifejezéseket csak akkor kapcsolhatunk össze, ha értékük $[-65535, 65535]$ zárt intervallumba esik /17 biten tárolható kettes számszendszerbeli számok/. A számok között a logikai müveletek bitenként hajtódnak végre. Ismeretes, hogy a számitógép a számokat kettes számrendszerben tárolja. A logikai müveletek ezeket a számjegyeket ugy adják össze, hogy az 1-nek az igaz a Ø-nak a hamis felel meg.

Tehát például a

6 OR 1Ø értékét a következőképpen

$6=11Ø_2$ $\quad$ $1Ø=1Ø1Ø_2$

ØllØ OR 1Ø1Ø hatására keletkező uj szám azon helyiértékein lesz egyes, ahol legalább az egyik egyes és azokon lesz Ø, ahol mindkét számnál nulla volt. Igy

ØllØ OR 1Ø1Ø = 111Ø

vagyis

6 OR 1Ø = 14

A 6 AND 1Ø hasonlóan számitható ki.

ØllØ AND 1Ø1Ø = ØØ1Ø

tehát

6AND 1Ø = 2

A NOT müveletnél pedig a szám kettes számrendszerbeli alakjánál a nullák helyére egy, az egyesek helyére nulla kerül.
pl.: NOT 1=-2
Ahhoz, hogy a NOT művelet utáni eredményt megértsük, tudni kell azt, hogy a gépünk a negativ számokat kettes komplemens kódban tárolja. A kettes komplemens lényegét 4 jegyü /4 bibes/ kettes számrendszerbeli számokon mutatjuk be. Ismeretes, hogy 4 bitnek összesen 16 különböző állapota lehet. Ezekhez az állapotokhoz hozzárendelhetjük nullától tizenötig a számokat, ilyenkor minden jelsorozat egy négyjegyü kettes számrendszerbeli számnak felel meg. Ilyen megoldás esetén viszont nincs lehetőség negativ számok tárolására. Éppen ezért az is megtehetjük, hogy a legmagasabb helyiértéket előjelbitnek tekintjük, és igy -7-től +7-ig tudunk számokat tárolni. Általában, ha 1 van a legmagasabb helyiértéken /jelen esetben a 4.-en/, akkor a szám negativ, ha nulla akkor a szám pozitiv.
/Pl.: 6=Ø11Ø ; -6=111Ø /

Igy viszont csak 15 számot rendelünk a 16 különböző állapothoz. Mivel -Ø=+Ø /1ooo=oooo/ tehát a nullához két különböző állapot tartozik.
Ezért a számitástechnikában nem ezt a megoldást választják a negativ számok tárolására, hanem az ugynevezett kettes komplemenst. Ennél a tárolási módnál igaz marad az, hogy ha a legmagasabb helyiértékü bit egyes, akkor a szám negativ, ha nulla, akkor pozitiv. Az eltérés abban van, hogy a pozitiv számokból nem ugy képezzük a negativ számokat, hogy a legmagasabb helyiértékü bitet egyre változtatjuk, hanem ugy, hogy balról haladva az első egyesig a biteket nem módositjuk, utána viszont minden bitet az ellenkezőjére változtatjuk.

Például a minusz 6-ot igy állitjuk elő:

6 = Ø11Ø

-6 = 1Ø1Ø

Ennek a tárolási módnak az az előnye is megvan, hogy ha bitenként összeadunk két azonos abszolut értékü, de különböző előjelü számot az eredmény nulla lesz. A müvelet elvégzése után az eredménynél csak az első 4 bitet szabad figyelembe venni, hiszen 4 jegyü számokról volt szó, és a komplemenst is 4 jegyre képeztük. Pl: $Ø11Ø_2$

$+1Ø1Ø$

$1ØØØØ$

Az eredményből az 5. bitet figyelmen kivül hagyva valóban nullát kapunk.

Természetesen hasonló eredményre jutunk tetszőleges számu bit esetén is, csak előtte azt kell tisztázni, hogy hány jegyre nézve alakitjuk ki a komplemenst.

Ezzel a tárolási módszerrel az alapmüveletek technikai megvalósitása is leegyszerüsödik. Pl. a kivonás egy komplemens képzésből és egy összeadásból elvégezhető. Nem szükséges külön áramköri kivonómü.

Az AIRCOMP-16 számitógépre vonatkoztatott BASIC a logikai müveletekkel, olyan aritmetikai kifejezéseket tud összekapcsolni, amelyek értéke 17 biten tárolható, és a negativ számokat 16 bites kettes komplemensben tárolja. A logikai müveletek elvégzése után a 17. bitet figyelmen kivül hagyja.

Visszatérve a NOT müveletre nézzük meg, hogy
a NOT 1 értéke miért lesz -2.
Az egy 16 biten a következő lesz:

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1

A NOT minden bitet ellenkezőjére változtat

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Ø

Ez pedig a kettőnek a kettes komplemense, tehát -2.

Ellenőrzés: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 Ø
1 Ø
1 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø

A 17. bitet figyelmen kivül hagyva az eredmény nulla.

4.7.2. IF utasitás

Az IF /ha/ utasitás bizonyos feltételek teljesülésétől, illetve nem teljesülésétől teszi függővé egyéb utasitások végrehajtását. Az utasitásnak két alakja van. Az egyik

IF A THEN U1 :U2: .....: UN

Ahol.- IF = az utasitás egyik kulcsszava
- A = logikai kifejezés, vagy matematikai kifejezés, melyeknek értéke [Ø, 255] intervallumba kell, hogy essen.
- THEN /akkor/ = az utasitás másik kulcsszava.
- U1; U2,....,UN=tetszőleges utasitás.

Megjegyzés:

Az U1, U2,....,UN utasitások helyére nem javasolt ujabb IF illetve a FOR utasitás.
Az IF utasitás hatására az A értékétől függően hajtódnak végre az U1,U2,...,UN utasitások

1., A logikai kifejezés és értéke:
   a., 1/igaz/, akkor a THEN utáni utasitások végrehajtódnak
   b., Ø /hamis/, akkor a THEN utáni utasitások nem hajtódnak végre, és programsor esetén áttér a következő programlépésre.
2., A matemetikai kifejezés és értéke:
   a., [1,255] intervallumba esik, akkor a THEN utáni utasitások végrehajtódnak
   b., Ø, akkor a THEN utáni utasitások nem hajtódnak végre, és programsor esetén a soronkövetkező programlépésnél folytatódik a program.
   c., negativ vagy nagyobb mint 255 akkor? ØIQ ERROR hibajelzéssel a gép leáll.

Megjegyzés:
Az A kifejezésben nem szerepelhet a NOT művelet, mivel a 16 bites kettes komplemens képzés miatt az A kifejezés értéke nem esik a [Ø, 255] intervallumba.
Az IF utasitás másik alakja:

IF A THEN U1:U2:.....:UN ! V1;V2:.....V

Ahol
- IF = az utasitás egyik kulcsszava
- A = logikai vagy matematikai kifejezés, értékük [Ø,255] intervallumba kell, hogy essék.

-THEN= az utasitás második kulcsszava
-U1, U2,...,UN= tetszőleges utasitások
-' / aposztrof; jelentése=egyébként/ = az utasitás harmadik kulcsszava
-V1, V2,.....VM= tetszőleges utasitások

Megjegyzés:

Az U1, U2,.. ,UN utasitások helyére nem javasolt ujabb IF illetve a FOR utasitás. A V1, V2,...,VM helyére nem javasolt a FOR utasitás.

Az utasitás végrehajtásakor A értékétől függően a következő történik:

1., Ha az A logikai kifejezés értéke:
   a., 1 (igaz), akkor az U1,U2,...,UN utasitások végrehajtódnak, de a V1,V2,.....,VM utasitások nem
   b., Ø (hamis), akkor a V1,V2,....VM utasitások végrehajtódnak, de az U1,U2,...,UN utasitások nem
2., Ha az A matematikai kifejezés értéke:
   a., [1,255] intervallumba esik, akkor az U1,U2, ...,UN utasitások végrehajtódnak, de a V1,V2,.....VM utasitások nem.
   b., Ø akkor V1,V2....,VM utasitások végrehajtódnak, de az U1,U2,...,UN utasitások nem
   c., negativ vagy nagyobb mint 255, akkor? IQ ERROR• hibajelzéssel a gép leáll.

Megjegyzés:
A NOT művelet itt sem szerepelhet az A kifejezésben.

Példa:

Irjunk programot amely 3 szám közül kiválasztja a legkisebbet és kiirja a képernyőre.

```
1Ø INPUT A,B,C
2Ø IF A < B    THEN GOTO 5Ø
3Ø IF B < C    THEN  M=B ' M=C
4Ø GOTO 6Ø
5Ø IF A < C  THEN  M= A ' M = C
6Ø ?  "A LEGKISEBB ELEM=" M
```

Tanulságképpen nézzük meg az alábbi programot, amelyben egy IF utasitás után ujabb IF-et használunk.

```
1Ø INPUT  A,B,C
2Ø IF A < B  THEN  IF   A < C THEN   M=A:GOTO 4Ø ' M=C:GOTO 4Ø
3Ø IF B < C  THEN M=B ' M=C
4Ø ?  " A LEGKISEBB ELEM=" M
```

A programban szereplő 30 utasitásra soha nem kerül sor. A 20 utasitásban nem határozódott, hogy az ' /apasztróf/ melyik THEN-hez tartozik. Igy akár A < B, akár A < C nem teljesülése esetén az aposztróf utáni utasitás következik. Az apasztróf után viszont már állhat ujabb IF utasitás!

```
1Ø INPUT A,B,C
2Ø IF A<B THEN GOTO 3Ø' IF B<C THEN M=B:GOTO 4Ø' M=C:GOTO 4Ø
3Ø IF A<C THEN M=A' M=C
4Ø ? "A LEGKISEBB ELEM="M
```

Helyes megoldást ad az alábbi program is:

```
1Ø INPUT A,B,C
2Ø IF A<B THEN IF A<C THEN M=A' IF B<C THEN M=B' M=C
3Ø ? "A LEGKISEBB ELEM="M
```

Megjegyzés:
Ha a THEN előtti kifejezés értéke Ø, akkor a legelső aposztrófnál folytatódik az utasítások végrehajtása.

4.8. Indexes változók, a DIM és a CLR utasítás

Az indexes változókat már a 4.4.1. pontban megemlítettük. Itt most ezekről részletesen lesz szó. Ez a BASIC változat egy- és kétindexes változók használatát engedélyezi mind numerikus, mind karakteres tipusú változó esetén.
Az indexes változókat szokás tömböknek is nevezni.
Az egyindexes változók vektornak, a kétindexes változók mátrixnak tekinthetők. Az egyindexes váltczó egy elemére a váltczó nevével és az elem sorszámával hivatkozhatunk.
A kétindexes váltczó egy elemét szintén a változó nevével valamint az elem sor és oszlopszámával érhetjük el. Mint azt már a 4.4.1. pontban ismertettük az indexes változók nevénél ugyanazok az előírások érvényesek, mint az index nélküli változóknál.

4.8.1. DIM utasítás

Az indexes változók első használata előtt a változót általában deklarálni kell. Ezt a DIM utasítással tehetjük meg. A DIM utasítás általános alakja

DIM $A_1(N_1,M_1),A_2(N_2,M_2),\ldots\ldots A_k(N_k,M_k)$,

ahol

- DIM = (DIMENSION = dimenzió rövidítése) az utasítás kulcsszava
- A_i = az i. indexes változó neve
- (= nyitózárójel
- N_i = az i. változó első indexének maximális értéke
- , = vessző két indexet elválasztó jel /vektor esetén elmaradhat /
- M_i = az i. változó második indexének maximális értéke /vektor esetén elmarad vagy nulla/

/i=1,2,....,k/

Az N_i és M_i tetszőleges matematikai kifejezés lehet $\emptyset \leq N_i,M_i < 256$ megkötéssel. Ha az N_i vagy az M_i nem egész, akkor az index maximális értéke a kifejezés egész értéke lesz. A DIM utasításban $A_i \neq A_j$ ha $i \neq j$, azaz nem lehet két azonos nevű változót deklarálni még akkor sem, ha az egyik vektor a másik mátrix.

Ha a deklarálást elhagyjuk, akkor a vektor vagy mátrix első előfordulásakor egy automatikus deklarálás történik. Ez vektor esetén a DIM V(10), mátrix esetén DIM M(10,10) deklarálásnak felel meg. /V a vektor, M pedig a mátrix neve./

Minden indexes változó csak egyszer deklarálható! Kivéve akkor, ha a deklarálás után NEW, RUN vagy CLR utasítást hajtott végre a gép. A NEW utasítást a 4.2. fejezetben, a RUN utasítást a 4.3. fejezetben ismertettük. A CLR utasításról a következő fejezetben lesz szó.

Az indexes változók indexének legkisebb értéke mindig nulla /ezért ezt a deklarálásban nem kell külön jelezni/, a legnagyobb értéke pedig 255 lehet.
Az indexes változó egy elemére a változó nevével és az index /indexek/ értékével lehet hivatkozni. Az index értéke tetszőleges matematikai kifejezés lehet. Ha a kifejezés értéke nem egész, akkor az index aktuális értéke a kifejezés egész része lesz.
Például az

A(32/5-1,SQR(25)-7/2)=283

értékadó utasitás után a 283 az A(5,1) változóba kerül.
Megjegyzés: A mátrix nullás indexű oszlopát az interpreter a mátrix nevével megegyező nevű vektornak tekinti. Azaz a vektorok olyan speciális mátrixok, amelyeknek második indexe mindig nulla, de ezt nem kötelező kiírni. Tehát

$$M(I,\emptyset) \equiv M(I)$$

4.8.2. CLR utasítás

A CLR /a CLEAR = tisztít rövidítése/ utasítás parancsként és programlépésként is használható. Hatására valamennyi változó értéke nulla lesz. Megsemmisíti a DIM utasításokkal deklarált indexes változókat. A DATA pointer /számláló/ értékét ugyanúgy visszaállítja egyre, mint a RESTORE utasítás.
Példa:
Irjunk programot, amely meghatározza n adat átlagát és szórását!

```
1Ø INPUT "ADATOK SZAAMA=";N
2Ø DIM A(N)
3Ø S=Ø
4Ø I=1
5Ø ? I ".ADAT=";  : INPUT A(I)
6Ø S=S+A(I)
7Ø IF I=N THEN GOTO 8Ø: I=I+1:GOTO 5Ø
8Ø AT=S/N
9Ø W=Ø
1ØØ I=1
11Ø W=W+ABS (A(I)-AT)↑ 2
12Ø IF I=N THEN GOTO 13Ø: I=I+1 : GOTO 11Ø
13Ø T=SQR (W/N)
14Ø ?:?
15Ø ? "AATLAG=" AT
16Ø ? "SZOORAAS=" T
```

4.9. Ciklusszervező utasítások FOR, TO, STEP, NEXT

A BASIC nyelvű programoknál a ciklusok szervezését ciklusszervező utasítások könnyítik meg. A ciklusszervező utasítások kulcsszavai a következők: FOR /-ra, -re/, TO /-ig/, STEP /lépés/, NEXT /következő/. A kulcsszavakkal szervezett ciklust elsősorban programlépésként használjuk, de lehetőség van parancskénti felhasználására is. Ilyenkor egy parancsként kell kiadni a ciklus valamennyi utasítását. A ciklus szerkezetét programlépésenként mutatjuk be.

k_1 FOR V=A_1 To A_2 STEP A_3

k_2 U_1

⋮

k_n U_{n-1}

k_{n+1} NEXT V

ahol

- k_i = a programlépések sorszáma (i = 1, 2,.....,n+1)
- U_i = a ciklusmag utasításai (i = 1, 2,.....,n-1)
- V = a ciklusváltozó. V-t célszerű index nélküli /skalár/ változónak választani
- A_i = tetszőleges matematikai kifejezések (i = 1, 2, 3)

A_1 a ciklusváltozó kezdőértéke, A_2 a végértéke, A_3 a ciklusváltozó növekménye (ez lehet negatív is). Amennyiben az A_1, A_2, A_3 matematikai kifejezések változókat is tartalmaznak és ezen változók értékét a ciklusmagban megváltoztatjuk, ez semmilyen hatással nincs a ciklusra. A matematikai kifejezések értéke ugyanis csak egyszer, a ciklus megkezdésekor számítódik ki. A ciklusparaméter értéke a ciklusmagban megváltoztatható.

A NEXT utáni ciklusváltozó elmaradhat, tehát a k_{n+1} sor így is helyes

k_{n+1} NEXT

A ciklusmag utasításai között DATA utasítás nem szerepelhet. Amennyiben a növekmény /A_3 matematikai kifejezés/ értéke egy, a STEP A_3 elhagyható. Tehát a

FOR V=A_1 TO A_2 STEP A_3 és a FOR V=A_1 TO A_2

utasítások megegyeznek.

Mivel a ciklusparaméterek tetszőleges matematikai kifejezések lehetnek azt, hogy a ciklusváltozó mely értékeire hajtódnak végre a ciklusmag utasításai, a következő programrészlettel szemléltetjük:

k_1 $V=A_1$

k_2 U_1

⋮

k_n U_{n-1}

k_{n+1} $V=V+A_3$

k_{n+2} IF $(V-A_2) * L>\emptyset$ THEN GOTO k_{n+3}' GOTO k_2

k_{n+3} U

A bemutatott programrészletből láthatjuk, hogy a ciklusváltozó kezdőértékére a ciklusmag utasításai egyszer mindig végrehajtódnak, függetlenül a növekmény értékétől és a végértéktől. A ciklusváltozó értékét a ciklusmag végén vizsgálja meg az interpreter. A ciklusból való kilépés után a ciklusváltozó értéke függ attól, hogy ez milyen módon történt. A ciklusból GOTO utasítással is kiléphetünk, ekkor a ciklusváltozó értéke megegyezik azzal az értékkel, amivel a kilépéskor rendelkezett. Ha a ciklusból a NEXT utasítással léptünk ki, akkor a ciklusváltozó értéke megegyezik azzal az értékkel, amely már meghaladta a végértéket. A ciklusmag utasításaira csak akkor szabad GOTO-val „ráugrani", ha a NEXT utasítás előtt GOTO-val ki is léptünk a ciklusból.

Ciklusok egymásba ágyazása

Ha egy FOR ciklus megnyitása után egy újabb ciklust nyitunk mielőtt az előzőt NEXT-tel lezártuk volna, akkor ciklusok egymásba ágyazásához jutunk. Ciklusok egymásba ágyazásánál fontos szabály, hogy mindig azt a ciklust kell előbb NEXT-tel lezárni, amelyiket később nyitottuk meg. Minden ciklus megnyitásakor a stack pointerben /veremszámlálóban/ a megnyitott ciklus ciklusmagja első utasításának címe kerül. A ciklus lezárásakor ez a cím kikerül a „veremből". A „veremből" min-

dig az utoljára betett címet lehet először kivenni, majd az ezt megelőzőt, stb... Ez a szervezés teszi kötelezővé, hogy az utoljára megnyitott ciklust kell először lezárni. Minden NEXT V után az interpreter ellenőrzi, hogy a V nevű ciklust nyitottuk-e meg utoljára, ha nem ?PP ERROR hibaüzenetet ír ki. Ha a NEXT után nincs változónév, akkor a ciklusból való kilépéskor automatikusan az utoljára megnyitott ciklust fogja lezárni.
Összefoglalva két ciklusnak csak akkor lehet közös része, ha az egyik ciklus teljes egészében benne van a másik ciklusban. Ha két vagy több ciklus is olyan, hogy ugyanaz a ciklusmagjának az utolsó utasítása, akkor a NEXT utasításokat úgy lehet tömöríteni, hogy a NEXT-et csak egyszer írjuk le és utána vesszővel elválasztva olyan sorrendben írjuk a ciklusváltozókat, amilyen sorrendben a ciklusokat le kell zárni. Tehát először az utolsónak megnyitott ciklus ciklusváltozóját, utoljára pedig a legelőször megnyitott ciklus ciklusváltozóját. Itt is lehetőség van a NEXT utáni változónevek elhagyására. Ilyenkor csak a vesszőket kell kitenni, amiből az interpreter a lezárandó ciklusok számát kapja meg. A NEXT és minden további vessző egy-egy ciklus lezárását jelenti.
Általában minden BASIC-változat meghatározza a ciklusok egymásba ágyazásának maximális mélységét. Azaz azt, hogy maximum hány ciklus ágyazható egymásba. (Ez a legtöbb esetben 10.)
Az AIRCOMP-16 BASIC változatnál ez a mélység korlátlan.
Ez azt jelenti, hogy csak a RAM memória kapacitása szab határt az egymásba ágyazásnak.

Példák:

1./ Irjunk programot, amely n elem közül kiválasztja a legkisebb elemet!

Megoldás:

```
1Ø INPUT "ELEMEK SZÁMA="; N
2Ø DIM A(N)
3Ø FOR I=1 TO N
4Ø ? I ".ELEM="; : INPUT A(I)
5Ø NEXT I
6Ø LK = A(1)
7Ø FOR I = 2 TO N
8Ø IF LK>A(I) THEN LK = A(I)
9Ø NEXT
1ØØ ?"A LEGKISEBB ELEM=", LK
```

2./ Irjunk programot kót N x M tipusú mátrix összegének kiszámítására!

Megoldás:

```
1Ø INPUT "N="; N, "M="; M
2Ø DIM A(N, M), B(N,M), D(N,M)
3Ø FOR I=1 TO N
4Ø FOR J=1 TO M
5Ø ? "A("I", "J")=";: INPUT A(I,J)
6Ø NEXT J
7Ø NEXT I
8Ø FOR I=1 TO N
9Ø FOR J=1 TO M
1ØØ ? "B("I", "J")=";: INPUT B(I, J)
11Ø NEXT
12Ø NEXT
13Ø FOR I=1 TO N
14Ø FOR J=1 TO M
150 D (I, J) = A(I,J) + B(I,J)
16Ø NEXT J, I
17Ø FOR I=1 TO N
18Ø FOR J=1 TO M
19Ø ? D(I, J);
2ØØ NEXT J:?: NEXT I
```

4. 10. Stringek kezelése

A BASIC programozási nyelv lehetőséget biztosit arra, hogy tetszőleges szöveget tároljunk és ezt feldolgozzuk a számítógép segítségével. A szövegkezelést a string /karakterlánc/ változók, kifejezések és függvények könnyítik meg.

4.10.1. Karakter tipusú /string/ változók

Mint azt már a 4.4.1. pontból tudjuk a karakteres változók megkülönböztetésére a $ jelet használjuk. A karakteres változó lehet index nélküli, egy- és kétindexes.
A karakteres változóknak - hasonlóan a numerikus változókhoz - értékadó, INPUT és READ utasítással adhatunk értéket.
Egy változóban maximum 255 karakter hosszúságú szöveget /stringet, karakterláncot/ tárolhatunk.
Az indexes string változókra a 4.8. fejezetben ismertetett szabályok értelemszerüen érvényesek.

Értékadó utasítás

Ha az értékadó utasítás bal oldalán karakteres változó van, akkor a jobb oldalon string kifejezésnek kell állni.
A string kifejezés legegyszerűbb esetben idézőjelbe tett karakterlánc. (A string kifejezés általános alakját a 4.10. fejezet végén adjuk meg.)
Idézőjelet speciális értékadó utasítással lehet tárolni a változóban /lásd stringfüggvények!/.

INPUT utasítás

Az INPUT listában szereplő karakteres változóban - néhány speciális karaktertől eltekintve - tetszőleges karakterlánc tárolható. Ilyen tipusú változóba adatmegadáskor lenyomott

billentyűk által meghatározott karakterlánc kerül. A karakterláncban szereplő idézőjel nem kerül tárolásra.
Ennek a jelnek speciális jelentése van az utasítás végrehajtásakor. A karakterlánc lezárására szolgál. A karakteres változóba csak azok a karakterek kerülnek, amelyek az idézőjel előtt vannak. Ebből adódik, hogy INPUT listában szereplő karakteres változóba nem kerülhet kurzorvezérlő billentyű.

READ utasítás

A READ utasítás listájában szereplő karakteres változókhoz tartozó adatoknak string kifejezéseknek kell lenni. Tehát megfelelő DATA lista elemek, vagy idézőjelbe tett karakterláncok, vagy stringváltozók vagy string kifejezések lehetnek.

Nézzünk egy mintaprogramot a fenti utasításokra!

```
1Ø A$ = "ALMA"
2Ø E$ = "SZILVA"
3Ø INPUT B$
4Ø READ D$, F$
5Ø ? A$, B$, D$, F$
6Ø DATA "MEGGY", E$
```

A RUN után megjelenő kérdőjel után gépeljük be a BARACK szöveget! Az 5Ø sor végrehajtásakor a képernyőn az alábbi szöveg jelenik meg:

ALMA BARACK MEGGY SZILVA

Az egyes karakteres változókba kerülő szöveget a szöveget alkotó karakterek kódja segítségével tárolja a számítógép.

Minden karakterhez hozzá van rendelve egy pozitív egész szám. Ezt nevezzük a karakter kódjának. Valamennyi kód kisebb mint 255, tehát egy bájton tárolható. A BASIC programozási nyelv ASCII szabvány kódrendszert használja. Az egyes karakterek kódját a függelékben megtaláljuk. Az ASCII kódrendszer nagy előnye, hogy a betükhöz rendelt számok a betűk ABC sorrendjét követik. Igy az ABC szerinti rendezés számok nagyság szerinti rendezésére redukálódik.

4.10.2. String függvények

A különböző szövegek feldolgozásához az alábbi string függvényeket használhatjuk fel:

a./ ASC (X$) X$ = tetszőleges string kifejezés
Az ASC függvény az argumentumban szereplő string kifejezés első karakterének ASCII kódját állitja elő.

Például: ? ASC ("ALMA") utasítás végrehajtásakor 65-öt ir ki. Ez az A betű kódja.

b./ CHR$ (K_1, K_2, ..., K_{255})
K_i = matematikai kifejezés (i = 1,2,...., 255)
és $\emptyset \leq K_i \leq 255$ (i = 1,2,...., 255)

A CHR$ függvény az argumentumban lévő matematikai kifejezések értékének megfelelő karaktereket generálja.

Például a ? CHR$ (65, 76, 77, 65) utasításra a képernyőn az ALMA felirat jelenik meg.

c./ LEN (X$) X$ = tetszőleges string kifejezés
A LEN függvény az argumentumban szereplő string kifejezés hosszát generálja. (A karakterlánccot alkotó karakterek számát.)

Például a ? LEN ("ALMA") utasítás a 4 megjelenését eredményezi

d./ LFT$ (X$, n) X$ = tetszőleges string kifejezés
n = olyan tetszőleges matematikai kifejezés, melyre $\emptyset \leq n < 256$ teljesül.

Az LFT$ függvény balról k/k = INT(n)/db karaktert generál az X$ string kifejezésből. Ha n értéke nagyobb mint az X$ string kifejezésben levő karakterek száma, akkor az egész stringet generálja. (n>LEN (X$) megegyezik n = LEN (X$) esettel.)

Például a ? LFT$ ("ALMA", SQR (12)) utasítás után a képernyőn az ALM felirat jelenik meg.

e./ RGH$ (X$,n) X$ = tetszőleges string kifejezés
n = olyan tetszőleges matematikai kifejezés, melyre $\emptyset \leq n < 256$ teljesül

A RGH$ függvény jobbról k/ k = INT (n)/ db karaktert generál az X$ stringkifejezésből. Ha n értéke nagyobb, mint az X$ string kifejezésben levő karakterek száma, akkor az egész stringet generálja. /n> LEN (X$) megegyezik n = LEN (X$) esettel./

Például a ? RGH$ ("ALMA", 3.9) utasítás után a képernyőn a LMA felirat jelenik meg.

f./ MID$ (X$, n, m)

X$ = tetszőleges string kifejezés
n, m = tetszőleges matematikai kifejezések, melyekre $0 \leq n, m < 256$ teljesül.

A MID$ függvény a string kifejezés k./k=INT (n)/ elemétől kezdve j /j=INT (M)/ db karaktert generál. k = 0 esetén a MID$ függvény az üres stringet generálja függetlenül m értékétől. Ha m nagyobb mint a (k-1). elem utáni karakterek száma, akkor a k. elemtől a string végéig generálja a karaktereket. Ha n nagyobb mint a stringben lévő elemek száma, akkor az üres stringet generálja.

Például: ? MID$ ("FEKETE", 2,3) utasítás az EKE kiírását eredményezi.

g./ STR$ (X)

X = tetszőleges matematikai kifejezés

Az STR$ függvény az X matematikai kifejezés nyomtatási képét string kifejezéssé alakítja át.

Például: A$ = STR$ (-SQR (3↑2 + 4↑2)) utasítás végrehajtása után az A$ értéke -5 lesz.

h./ VAL (X$)

X$ = tetszőleges string kifejezés

Ha X$ egy matematikai kifejezés string képe, akkor a VAL függvény ezen kifejezés numerikus értékét generálja.

Példa:

```
1Ø A$ = A↑2 + B↑2
2Ø A = 3 : B = 2
3Ø ? VAL (A$)
```

A RUN parancs után a képernyőn 25 jelenik meg.
($3^2 + 4^2 = 25$)

Megjegyzés: Az idézőjel kódja 34, a [CR] billentyű kódja 13, ezeket a karaktereket a CHR$ függvény segitségével lehet csak karakteres változóban tárolni.
A$ = CHR$ (34) : B$ = CHR$ (13) utasítások hatására az A$ az idézőjelet, a B$ a [CR] billentyűt tárolja.

4.10.3. String kifejezések és összehasonlításuk

Stringek összekapcsolása

A stringek között egy művelet van értelmezve, a stringek egymás után fűzése. A müveleti jel megegyezik a numerikus összeadással: +. Ha két string kifejezést ezzel összekapcsolunk, akkor az eredmény egy olyan string lesz, amely a két összeadandó karakterlánc összekapcsolásából adódik. Az összeadásnál a sorrend fontos, tehát a művelet nem kommutatív. A$ + B$ ≠ B$ + A$

```
1Ø  A$ = "MACSKA"
2Ø B$ = "FEKETE"
3Ø D$ = B$ + A$
4Ø ? D$
```

A RUN után a képernyőn a FEKETEMACSKA felirat jelenik meg.

String kifejezések

Ha idézőjelbe tett karakterláncokat, karakteres változókat, olyan függvényeket, melyek stringeket generálnak összekapcsolunk a + jellel, akkor string kifejezéshez jutunk. Az önmagában álló idézőjeles karakterláncot, karakteres változót, stringet generáló függvényt is string kifejezésnek nevezzük.

Stringek összehasonlítása, kiegészítés a logikai kifejezéshez

A 4.7.1. pontban a logikai kifejezést úgy vezettük be, mint aritmetikai kifejezések relációjellel történő összekapcsolása. Ezt most azzal egészítjük ki, hogy logikai kifejezéshez jutunk akkor is, ha string kifejezéseket kapcsolunk össze relációjellel. Az egyes relációk jelentése stringek esetén a következő:

Reláció	Értelmezés
$S_1 = S_2$	S_1 és S_2 strint ugyanaz a karakterlánc. /S_1 egyenlő S_2-vel/
$S_1 < S_2$	S_1 string ABC sorrendben (karakterek kódjainak összehasonlítása) megelőzi S_2 stringet.
$S_1 <= S_2$ vagy $S_1 =< S_2$	S_1 string egyenlő S_2 stringgel vagy S_1 ABC sorrendben megelőzi S_2 stringet.
$S_1 > S_2$	S_2 string ABC sorrendben megelőzi S_1 stringet.
$S_1 >= S_2$ vagy $S_1 => S_2$	S_2 egyenlő S_1 stringgel, vagy S_2 ABC sorrendben megelőzi az S_1 stringet.

$S_1 <> S_2$ vagy $S_1 >< S_2$ S_1 és S_2 stringek különbözőek.

A karakterek ABC sorrendjét az ACCII kódjuk számértéke határozza meg. A kisebb kódú karakter ABC sorrendben mindig megelőzi a nagyobb kódú karaktert. Ezek alapján lehetőség van a szokásos ABC sorrend kiterjesztésére tetszőleges karakterek összehasonlítása esetén.

Nézzünk néhány egyszerű példát stringek közötti relációk teljesülésére!

Reláció	Magyarázat
"ABB" < "ADAT"	Számjegyek megelőzik a betűket.
"2 KORTE" < "ALMA"	A betűköz megelőzi a számjegyeket.
"␣30" < "10"	Az első string rövidebb.
"AAA" < "AAAA"	
"15" = "15"	

4.11. DEFFN utasítás

A DEFFN (DEFINE FUNCTION = függvénydefiniálás rövidítése) utasítás segítségével tetszőleges egyváltozós függvények definiálhatók. Csak programlépésként használható, parancsként nem.
Általános alakja:

k DEFFN V(X) = F(X)

ahol

- k = a programlépés sorszáma
- DEFFN = az utasítás kulcsszava
- V = a függvény neve /csak egy betűből állhat/
- X = a függvény változója /formális paraméter/
- F(X) = X-től függő tetszőleges matematikai kifejezés

Az utasítás végrehajtása után az FNV (T)-hez /T = aktuális paraméter/ hozzárendeli azt az értéket, amit az F(X) matematikai kifejezésbe T-nek X helyébe történő behelyettesítése után kapunk. T tetszőleges olyan kifejezés lehet, amire az F(T) matematikai kifejezés értelmezhető. Tehát a program futása közben vagy a program futása után az FN, a függvény neve és az aktuális paraméter megadásával hivatkozhatunk a függvényre /FNV(T)/.

Mivel a függvény neve csak egy betű lehet, ezért egy programban maximum 26 függvény definiálható. Ha ugyanazzal a névvel több függvényt is definiálunk, akkor a programban az első definíció marad érvényben, a többi figyelmen kívül marad. A DEFFNV(X) = F(X) utasítás a programban tetszőleges helyen szerepelhet, ugyanis ez az utasítás nem végrehajtható, hanem a DATA utasítással megegyezően deklarációs.

Példa:

Irjunk egy olyan programot a DEFFN utasítás felhasználásával, amely kiszámítja a fokokban megadott szög szinusz értékét!

Megoldás:

```
1Ø INPUT "FOK =" ;A
2Ø ? "SIN ("A")=" FNS (A)
3Ø GOTO 1Ø
4Ø DEFFN S(X) = SIN (X/18Ø * 3.141592)
```

4.12. GOSUB és END utasítás

4.12.1. GOSUB utasítás

A GOSUB (GOTO SUBROUTINE = menj az alprogramhoz /eljáráshoz/ rövidítése) utasítás parancsként és programlépésként egyaránt kiadható, de csak programlépésként van értelme használni.

Az utasítás általános alakja:

GOSUB N

ahol

- GOSUB = az utasítás kulcsszava
- N = matematikai kifejezés

Az N matematikai kifejezésre vonatkozó ismeretek teljes egészében megegyeznek a 4.6.1. pontban a GOTO utasításban szereplő N kifejezésnél tárgyaltakkal.
Mint láthatjuk a GOTO N - GOSUB N között szoros kapcsolat van. A GOSUB N utasítás után ugyanott folytatódik a program, mint a GOTO N után. Lényeges különbség a két utasítás között csak akkor van, ha a GOSUB N után egy RETURN (=vissza) utasításhoz ér a program. A RETURN utasítás hatására szintén megváltozik a programlépések természetes sorrend szerinti végrehajtása. A GOSUB N hatására a stack pointerben (lsd. FOR ciklust) tárolódik a GOSUB N után - természetes sorrend szerint - következő utasítás címe. Erre a címre tér vissza a program a RETURN utasítás végrehajtásakor. A GOSUB utasítások egymásba ágyazásának elve megegyezik a FOR ciklusnál ismertetettekkel. Mindig az utoljára kiadott GOSUB utáni utasítást követő utasításra adódik át a vezérlés RETURN utasítás végrehajtásakor. A GOSUB utasítást általában bonyolultabb, hosszabb programoknál célszerű használni akkor, ha a programban többször, több helyen kell ugyanazokat az uta-

sításokat végrehajtani. Ilyenkor ezeket célszerű a program végén összegyűjteni és GOSUB utasítással hivatkozni rá, mint alprogramra. Az utasítás-csoport első utasítása elé egy END utasítást kell beirni, hogy megakadályozzuk ezen utasítások GOSUB nélküli végrehajtását. (Az END utasítást a 4.12.2. pontban ismertetjük.) Az END utasítás elhagyása esetén ?PP ERROR hibaüzenetet kapunk. Ez azt is jelenti, hogy az alprogramban lévő utasításokat utoljára feleslegesen hajtotta végre a gép. Alprogramok alkalmazásával a bonyolultabb, hosszabb programok áttekinthetőbbekké, rövidebbekké tehetők.

Példa:
Adjunk össze két-két számot úgy, hogy az összeadás és a kiírás alprogrammal történjen!

```
1Ø INPUT A, B, C, D
2Ø X=A : Y = B
3Ø GOSUB 1ØØ
4Ø X=C : Y = D
5Ø GOSUB 1ØØ
6Ø END
1ØØ K = X+Y
11Ø ? K
12Ø RETURN
```

Megjegyzés:
Az itt bemutatott példa csak a GOSUB és RETURN utasítások szemléltetésére jó. Az alprogramok tényleges hasznát ebből nem láthatjuk /rövidség, áttekinthetőség/, hiszen a fenti két utasítás nélkül a probléma jóval egyszerűbben megoldható lett volna. Próbáljuk ki a programot a 6Ø sor elhagyásával is!

4.12.2. END utasítás

Az END (=vége) utasítást csak programlépésként van értelme használni. Az END utasítás hatására a gép programfutási üzemmódból parancsüzemmódba kerül. Hatására a gép olyan állapotba kerül, mintha a legmagasabb sorszámú utasítást (amely nem vezérlésátadó) hajtotta volna végre. A programban több helyen is előfordulhat. Akkor használjuk, amikor a program logikai és fizikai vége nem esik egybe. Lásd az előző példát.

4.13. ON utasítás

Az ON (-ra, -re előljáró szó) utasítás segítségével egy matematikai kifejezés értékétől függően más-más utasítások végrehajtását írhatjuk elő. Az utasítást programlépésként és parancsként egyaránt használhatjuk.

Az utasítás általános alakja:

ON M '1. utasításcsoport' 2. utasításcsoport'...

Ahol

ON = az utasítás kulcsszava

M = olyan matematikai kifejezés, melyre $0 \leq M < 256$ teljesül

' = aposztrof (Az utasításcsoportok elválasztására szolgál)

i. utasításcsoport = egy vagy több BASIC utasítás

Az ON utasítást a gép úgy hajtja végre, hogy kiszámítja M matematikai kifejezés értékét, s ha van annyi utasításcsoport, amennyi a kifejezés értékének egész része, akkor ezen utasításcsoport utasításait hajtja végre, különben a futás

a soron következő programlépéssel folytatódik. A soron következő programlépéssel folytatódik akkor is a program, ha M kifejezés egész része nulla. Ha az M értéke valamelyik utasításcsoport végrehajtását eredményezi, akkor az utasítások végrehajtása után az ON programsort követő program-lépésre adódik a vezérlés.
Abban a speciális esetben, ha közvetlenül az apostrófok után álló valamennyi utasítás ugyanazzal a kulcsszóval kezdődik, akkor ezt elég egyszer az első apostróf elé írni.

Példa:
Irjunk programot, amely szöveggel kiírja a vizsgán szerezhető érdemjegyet!

Megoldás:

```
1Ø INPUT "JEGY =" A
2Ø ON A '?"ELEEGTELEN": END' ?"ELEEGSEEGES": END'
   ?"KOEZEPES": END' ?"JOO": END' ?"JELES": END
3Ø ?"TEVES JEGY": GOTO 1Ø
```

Mivel minden apostróf után közvetlenül PRINT utasítás van, ezért ezt elég lett volna egyszer kiírni. Tehát a 2Ø sor így is helyes:

```
2Ø ON A ?'"ELEEGTELEN": END'"ELEEGSEEGES": END' "KOEZEPES":
   END' "JOO": END' "JELES": END
```

A program szépséghibája az, hogy nem egész számokra is működik, ilyenkor az A kifejezés egész része alapján választja ki a végrehajtandó utasítást, tehát nem kerekít.

4.14. Speciális változók

A változó nevek tárgyalásakor utaltunk arra, hogy néhány név nem használható tetszőleges változók elnevezésére. Ezek a nevek speciális változókat jelentenek, amelyek értéke bizonyos hatással van az interpreter működésére, illetve bizonyos információkat szolgáltat a számára. Ebben a részben ezeket a változókat ismertetjük. Minden speciális változó a gép bekapcsolása után rendelkezik kezdeti értékkel. Ezt az értéket csak értékadó utasítással lehet megváltoztatni, READ és INPUT utasítással nem!

4.14.1. DL változó

A DL változó a képernyő szerkesztett /felfrissitett/ sorainak számát tárolja. Ezek a sorok nem karakter-sort, hanem pontsort jelentenek. Tudni kell azt, hogy egy karakter kiírásához egy 8x8-as pontmátrix áll rendelkezésre és minden karakter ebből a 64 pontból épül fel. A képernyő tartalmát a gépnek pillanatonként fel kell frissítenie, ami természetesen időt vesz igénybe. Igaz ugyan, hogy ez az idő olyan kicsi, hogy az emberi szem nem veszi észre ezt az állandó felfrissítést (újra írást). De ez az állandó újra írás a programok futási idejét megnöveli. Ha letiltjuk a képernyő felfrissítését, akkor a programok futási ideje több mint felére csökken, tehát a gép gyorsasága 2-3-szorosára nő. A felfrissített sorok számát a DL változó aktuális értéke határozza meg. A változó értékére az alábbi relációknak kell teljesülni:

$$0 \leq DL \leq 255 \quad \text{és} \quad 0 \leq DL - GL \leq 200$$

A GL speciális változót a 4.14.2. pontban ismertetjük.

A gép működési sebessége arányos a DL értékével. Ha a DL=Ø, akkor a legnagyobb, ha a DL=255 akkor a legkisebb.
A DL változó értékéből az INT (DL/8) képlettel számíthatjuk ki a képernyőn megjelenő karakter sorokat.
/Egy karakter = 8 pontsor./
Tehát például karakteres kiírás esetén a DL=40 és a DL=47 egyaránt 5 sor megjelenését /felfrissítését/ irja elő. Ez az 5 sor a normál kiírás utolsó 5 sora, vagyis a 21., 22., 23., 24. és 25. sor.

A DL változó alapértelmezés szerinti értéke: 200.
Ha INPUT utasítás listáján szerepel a DL változó, akkor nem adatot vár, hanem kiírja DL aktuális értékét. Ha READ listaelemként szerepel, akkor ? SN ERROR hibaüzenet jelenik meg.

A DL=Ø csak programlépésként eredményezi a képszerkesztés letiltását és csak addig, amíg a programfutás be nem fejeződik, illetve a [BRK] billentyűvel meg nem szakítottuk.
A DL=Ø parancsként használata a felfrissítés szempontjából megegyezik a DL=2ØØ paranccsal.

4.14.2. A GL változó

A GL változó segítségével lehet kijelölni a képernyőn a grafikus terület pontsorainak számát. Értéke nem lehet nagyobb a képszerkesztésben résztvevő pontsorok számától és értéke nulla és 2ØØ között kell, hogy legyen. Tehát:

$$Ø \leq GL \leq 2ØØ$$ és $$GL \leq DL$$ kell, hogy teljesüljön.

Alapértelmezés szerinti értéke: Ø
Mivel a grafikus terület nyilvántartásához nincs külön memóriája a gépnek, az ehhez szükséges memóriát a RAM memóriából nyeri. Tehát a GL változó értékének növekedése csökkenti a BASIC területet. Ha a GL értékét eggyel növeljük, akkor a BASIC terület 4Ø bájttal csökken. Minden képpont nyilván-

tartása 1 bit memóriát igényel. Egy sorban 4Ø * 8 pont van, ami pontosan 4Ø bájtnyi memóriának felel meg. Ha GL=2ØØ, akkor a BASIC terület /16115 bájt/ 8ØØØ bájttal közel a felére csökken. Az INPUT és READ listában szerepeltetése ugyanazt eredményezi, mint a DL változó.

4.14.3. A CR változó

A grafikus terület pontjainak színét szabályozza. Ha a grafikus területre akarunk rajzolni, akkor a CR értékétől függően a kirajzolt pontok vagy sötétek vagy világosak lesznek.
Alapértelmezés szerinti értéke: 1.
A CR változó értékét csak értékadó utasítással lehet megváltoztatni. A utasítás általános alakja:

CR = M

ahol M matematikai kifejezés, melyre $Ø \leq M < 256$ kell, hogy teljesüljön.

A CR változó értéke az értékadó utasítás után vagy nulla vagy egy lesz. Ha az M kifejezés egész része páros, akkor nulla, ha páratlan, akkor egy.
A képpontok színe pedig

- világos, ha CR = 1
- sötét, ha CR = Ø.

A CR változó minden parancs és minden program végrehajtása után felveszi az alapértelmezés szerinti értékét. READ és INPUT listában ugyanazt eredményezi, mint a DL változó.

4.14.4. A HM változó

A HM változó a szabad memória felső határát tartalmazza. A BASIC terület alsó határa a 16544. bájtnál van. A HM értékét a GL változó értéke befolyásolja. Ha GL értékét eggyel meg-

növeljük, akkor a HM értéke 40-nel csökken. A GL csökkentése a HM értéket nem módosítja. Tehát ha a grafikus területre nincs szükség, akkor a HM értékét állítsuk vissza a kezdeti értékére.

A HM változó alapfeltételezés szerinti értéke: 32768

A HM értéket értékadó utasítással lehet megváltoztatni.

$$HM = M$$

ahol M olyan matematikai kifejezés, melynek értékére

$16660 \leq M < 32773$ egyenlőtlenség teljesül.

Az utasítás végrehajtása után HM felveszi az M kifejezés értékének egész részét. HM alsó határa függ a gépben lévő program hosszától. Tehát a 16660 akkor érvényes, ha nincs program a gépben, ilyenkor parancsokat sem tud végrehajtani a gép.

Ha a HM értéke nagyobb mint 32767, akkor ezeket az értékeket 16 bites komplemensnek megfelelően tárolja. (Lásd 4.7.1. pont.) Tehát kijelzéskor ezek negatívok. Ilyenkor a valódi értéket a $HM + 2^{16}$ kifejezés adja. READ és INPUT listában ugyanazt eredményezi, mint a DL változó.

5. Programok és adatok tárolása kazettán, magnóhasználat

A kazettás magnóról, mint a géphez csatlakoztatható perifériális egységről az 1. fejezetben már volt szó. Az AIRCOMP 16 személyi számítógéphez bármilyen kazettás magnó használható, ha megfelelően össze van hangolva a géppel. A jegyzetben az MK 29 tipusú kazettás magnó használatát tárgyaljuk. A magnón levő billentyűzet és a magnó kezelését ismertnek tételezzük fel. Általános alapszabályként fogadjuk el azt, hogy mielőtt a magnót használnánk mindig ellenőrizzük le, hogy megvan-e a

megfelelő összeköttetése a számítógéppel, illetve, hogy a hangerő szabályozó maximális, a hangszín szabályozó minimális állásban van-e!

5.1. Programok tárolása kazettán

5.1.1. Program mentés kazettára
A SAVE utasítás

A SAVE /ment/ utasítás segítségével programokat tárolhatunk kazettán. Az utasítás programlépésként és parancsként egyaránt kiadható. Az utasítás általános alakja:

SAVE "programnév"

ahol

SAVE = az utasítás kulcsszava
" = idézőjel
programnév = tetszőleges karakterlánc

A SAVE utasítás hatására a gép kiküldi a magnóra a memóriában lévő programot. Természetesen a program a kimentés után a memóriában is megmarad. A SAVE utasítás nem ellenőrzi a magnó állapotát. Akkor sem jelez hibát, ha a gép össze sincs kötve a magnóval.
Az utasítás kiadása után a képernyőről eltünik a felirat (megszűnik a képszerkesztés) és egy sípoló hang jelzi, hogy elindult a program mentése. A program mentés befejezését a sípoló hang megszűnése és a korábbi képernyőtartalom megjelenése jelzi. A SAVE utasítás kiadásakor ügyelni kell arra, hogy a magnó felvételre legyen állítva.
Pontosabban az alábbi lépéseket tartsuk be:

1. Helyezzük el azt a kazettát a magnóba, amelyre a programot fel kívánjuk venni.
2. Előre vagy hátralépéssel állítsuk be a szalagot arra a pontra, ahol a program rögzítését el akarjuk kezdeni. Ehhez használjuk a magnó számlálóját.
3. Adjuk ki a SAVE parancsot, de a [CR] billentyűt ne nyomjuk le!
4. Indítsuk el a magnót felvételre!
5. Nyomjuk le a [CR] billentyűt!

A programok kivitelénél ügyeljünk arra, hogy a korábban felvitt programokra ne írjunk rá új programot. Kivitelnél nem ellenőrzi a gép a kazetta tartalmát, tehát ha a kiírófej elér egy régi programot, akkor arra ráír, s így az a program már nem tölthető be a gépbe. A magnón levő számlálót használjuk fel a programok elkülönítésére. Minden program elejénél és végénél jegyezzük fel a számláló állását és ennek segítségével tartsuk nyilván a kazettánkon lévő programokat.

5.1.2. Programok betöltése
LOAD utasítás

A LOAD (tölt) utasítás segítségével a kazettán lévő programokat betölthetjük a számítógépbe. Az utasítás programlépésként és parancsként is használható. Az utasítás általános alakja:

LOAD "programnév"

ahol:

LOAD = az utasítás kulcsszava
" = idézőjel
programnév = annak a programnak a neve, amelyet a kazettáról a gépbe akarunk tölteni.

Az utasítás hatására a képernyőről eltűnik a korábbi tartalom és különböző jelek jelennek meg, a magnó elindítása után a képernyőn levő jelek mozgásba jönnek. Ha az iró/olvasó fej elér egy program elejére, akkor összehasonlítja a LOAD utasításban megadott nevet a program elején lévő névvel, amennyiben a két név azonos, akkor elkezdi betölteni a programot a gépbe. A program betöltésének a végét az eredeti képernyőtartalom megjelenése jelzi. Ha a két név nem azonos, akkor a kazettán lévő program nevét attól a karaktertől kezdődően írja ki a képernyőre, ahol az első eltérést észlelte. Ebben az esetben vagy a programunk nevét tévesztettük el a LOAD utasításban, vagy rossz helyre álltunk a kazettán. Mindkét esetben újra ki kell adni a LOAD utasítást. Természetesen a hibánkat korrigálni kell. Sikeres LOAD utasítás végrehajtja a NEW funkciót is, tehát LOAD előtt nem szükséges törölni a memóriában lévő programot.

A LOAD utasítás kiadható az idézőjelbe tett programnév nélkül is. A programnév nélküli LOAD parancs a kazettán lévő első olyan programot fogja betölteni a memóriába, amely a LOAD utasítás után elhalad az olvasófej előtt. A LOAD parancsnál is ügyelni kell a lépések helyes sorrendjére.

1. Helyezzük el azt a kazettát a magnóba, amelyről a programot be kívánjuk tölteni a gépbe!
2. Előre vagy hátracsévéléssel állitsuk be a szalagot arra a pontra, amely a kazettán tárolt program kezdete előtt van!
3. Adjuk ki a LOAD "programnév" vagy a LOAD parancsot!
4. Nyomjuk meg a [CR] billentyűt!
5. Indítsuk el a magnót lejátszásra!

Ha a beállításnál túlhaladtunk a betöltendő program elején, akkor a gép a következő programot tekinti első programnak, ilyenkor újra ki kell adni a LOAD parancsot. Ezért inkább 2-3 fordulattal előbbre álljunk mint egy fél fordulattal hátrább. Célszerű két program között 4-5 fordulatot kihagyni a programok felvitelénél.

5.2. Adatok tárolása kazettán

Kazettára történő adatkiírás és beolvasás a 4.5.3. pontban megismert PRINT illetve a 4.5.1. pontban megismert INPUT utasítás segítségével történik.

5.2.1. Adatkivitel kazettára

Az adatok kivitelére a PRINT#1; utasítás szolgál. Az utasítás parancsként és programlépésként egyaránt használható. Általános alakja:

PRINT #1; lista

ahol

PRINT #1; = az utasítás kulcsszava

lista = numerikus és string kifejezések lehetnek, a kifejezéseket vesszővel kell egymástól elválasztani.

Az utasítás végrehajtásakor a gép a kifejezések értékét kiküldi a magnóra. A gép itt sem ellenőrzi a magnó állapotát, tehát ha a magnó nincs felvételre állítva, akkor az adatok nem kerülnek ki a kazettára. Az utasítás végrehajtásakor a képszerkesztés szünetel. Ez azt jelenti, hogy amikor az adatátvitel a magnóra elkezdődik, a képernyő elsötétül és úgy is marad a program végéig. Ha a képszerkesztést újra akarjuk indítani, akkor használjuk a DL speciális változót.

A PRINT #1; utasítás a DL értékét Ø-ra állítja. Ezt az értéket kell módosítani értékadó utasítással /pl. DL = 2ØØ/. A PRINT #1; utasítás végrehajtásakor a magnónak felvétel állapotban kell lennie, hogy az adatok a kazettára kerüljenek. Ezt a programfutás esetén úgy érhetjük el, hogy közvetlenül a PRINT #1; utasítás előtt egy olyan INPUT utasítást helyezünk el, ami figyelmeztet arra, hogy a magnót felvételre kell állítani és el kell indítani. Az INPUT utasítás listájában egy fiktív változót is el kell helyezni, ami a programfutás felfüggesztését eredményezi. Így marad időnk arra, hogy a magnót megfelelő állapotba hozzuk. A magnó elindítása után megadunk egy adatot és a [CR] billentyű lenyomása után a PRINT #1; utasítás a kívánt adatokat kiviszi a kazettára.

5.2.2. Adatok beolvasása kazettáról

Az adatok beolvasására az INPUT #1; utasítás szolgál. Az utasítás parancsként és programlépésként is használható. Az utasítás általános alakja:

INPUT #1; lista

ahol

INPUT #1; = az utasítás kulcsszava,
lista = numerikus és string változók, egymástól vesszővel elválasztva.

Az utasítás végrehajtásakor a gép az INPUT #1; lista elemszámának megfelelő számú adatot beolvas a kazettáról. Az utasítás végrehajtásakor a képernyő a LOAD parancsnál ismertetett állapotba kerül. Ha kevesebb adat van a kazettán mint amennyit a program vár, akkor a gépet csak a RESET gomb megnyomásával lehet újra parancsüzemmódra váltani. Az eredeti

képernyőtartalom hasonlóan a PRINT #1; utasításhoz, a program végén jelenik meg, addig a képszerkesztés szünetel. A DL változó programból történő beállításával ezt módosíthatjuk. Az INPUT #1; listában szereplő változók típusa meg kell, hogy egyezzen az adatok típusával.

6. Grafikus ábrázolási lehetőségek

A GL, CR speciális változók ismertetésekor már szó volt arról, hogy lehetőség van a képernyőn grafikus terület kijelölésére. A grafikus terület maximális nagysága 2ØØ x 32Ø pontmátrix. 2ØØ a sorok, 32Ø az oszlopok száma. A grafikus terület kijelölésekor csak a sorok számát határozhatjuk meg a GL változó értékével. Az oszlopok száma mindig 32Ø lesz. Lehetőség van arra is, hogy a képernyő egy részét grafikus, másik részét karakteres kiírásra használjuk. A grafikus sorok számát a GL, a karakteres sorok számát a DL - GL értéke adja (mindkét esetben pontsorról van szó). A GL értéke és a DL - GL értéke egyaránt maximum 2ØØ lehet, ez utóbbi azzal a megszorítással, hogy DL maximum 255 lehet.

Ha képernyőn grafikus és karakteres terület is van, akkor a grafikus terület van alul, a karakteres terület pedig fölül. A karakteres területről azt kell tudni, hogy az eredeti képernyő (csak karakteres terület) utolsó sorait tartalmazza. Tehát ha a kurzor az eredeti képernyő 1. sorában van és kiadjuk a GL = 1ØØ parancsot, akkor a képernyő alsó fele grafikus, a felső fele karakteres lesz és a kurzor eltűnik. Ez azért van, mert a karakteres terület az előző képernyő utolsó 12 sorát tartalmazza, a kurzor pedig a parancs kiadása után a 4. sorba került. A kurzort tehát lefelé kell mozgatni a 14. sorig, hogy megjelenjen a képernyő tetején.

A grafikus terület pontjaira azok koordinátáival lehet hivatkozni. A koordináta rendszer egy kicsit eltér a szokásos koordináta rendszerektől. Az origó a grafikus terület bal felső sarkában van. Az X tengely pozitív fele innen vízszintesen jobbra, míg az Y tengely pozitív fele innen függőlegesen lefelé halad. Tehát a grafikus terület egy koordinátarendszer 1. síknegyedének X tengelyre vonatkoztatott tükrözéseként kezelhető.

6.1. PLOT utasítás

Az egyes pontokra a PLOT (rajzol) utasítással hivatkozhatunk. Az utasítás mind parancsként mind programlépésként használható.
Az utasítás általános alakja

PLOT X,Y

ahol

PLOT = az utasítás kulcsszava,
X = tetszőleges matematikai kifejezés,
, = vessző
Y = tetszőleges matematikai kifejezés.

Az X matematikai kifejezés értékére teljesülnie kell a $\emptyset \leq X < 32\emptyset$ egyenlőtlenségnek.
Az Y matematikai kifejezés értékére pedig a $\emptyset \leq Y < GL$ egyenlőtlenségnek kell teljesülnie.
A fentiekből kitűnik, hogy a GL a grafikus sorok számát határozza meg, de ez nem egyenlő az utolsó sorban lévő pontok Y koordinátájával, mivel az első sorban lévő pontok Y koordinátája nulla. Hasonlóan a 32Ø oszlop koordinátái is eggyel kisebbek, mint az oszlopok sorszámai.

Az utasítás végrehajtásának hatására az X, Y koordinátájú pont színe az aktuális CR értéknek megfelelő lesz. (Világos, ha CR = 1, sötét, ha CR = Ø.)

A PLOT utasítás segítségével (pontonként) tetszőleges ábrát rajzolhatunk a grafikus területre. Mielőtt azonban ezt megtennénk, szükség van arra, hogy a grafikus terület minden pontját az ábra színének ellenkezőjére állítsuk! A gép bekapcsolása után kijelölt grafikus terület pontjai véletlenszerűen lesznek világosak és sötétek. Tehát rajzolás előtt minden pontot azonos színre kell állítani. Létre kell hozni a rajzoláshoz szükséges háttérszínt. Ezt megtehetjük a következő rövid programmal:

```
1Ø CR = 1:GL = 2ØØ : DL = 255
2Ø FOR X = Ø TO 319
3Ø FOR Y = Ø TO 199
4Ø PLOT X, Y
5Ø NEXT Y, X
```

A program a maximális grafikus területet világos színűre változtatja. A 1Ø. sorban a GL = 2ØØ : DL = 255 utasítások sorrendje nem változtatható meg. A DL = 255 biztosítja a maximális terület mellett a maximális karakteres területet (6 sor). Az itt bemutatott programnak van egy szépséghibája, mégpedig az, hogy pontonként festi át a grafikus területet és ez igen időigényes, kb. 15 percig tart. A 4Ø sorszámú utasítást a gép 64ØØØ-szer hajtja végre.

A speciális változók tárolásánál említettük, hogy a grafikus terület nyilvántartása a RAM memória bizonyos részén történik. A memóriában lévő bitek értéke határozza meg az egyes képpontok színét. Ezen memóriaterület feltöltésére a gép memóriájában rendelkezésre áll egy gépi kódú program. A gépi kódú

program megnézi a CR változó értékét és ennek megfelelően feltölti a RAM memóriának azt a részét, amelyet a grafikus terület nyilvántartására a GL változóval kijelöltünk.
A gépi kódú program a CALL 7535 utasítással indítható el (CALL = hívni).
Tehát az előző program így is írható:

```
1Ø  CR = 1:GL = 2ØØ:DL = 255:CALL 7535
```

A program futási ideje alig 1 másodperc.
Tehát a háttér kialakításához érdemes használni a CALL 7535 utasítást.
Ha a 1Ø. utasításban a CR = 1-et CR = Ø-ra módosítjuk, akkor a háttér nem világos, hanem sötét lesz.

6.2. POINT utasítás

A POINT utasítás segítségével lekérdezhetjük a grafikus terület pontjainak színét. Az utasítás parancsként és programlépésként is használható. Általános alakja:

POINT (X, Y)

ahol

- POINT = az utasítás kulcsszava,
  (= nyitó zárójel,
  X = tetszőleges matematikai kifejezés, melyre Ø ≤ X < 32Ø teljesül,
  , = vessző,
  Y = tetszőleges matematikai kifejezés, melyre Ø ≤ Y < GL teljesül,
) = bezáró zárójel.

Az utasítás végrehajtásakor a POINT (X, Y) függvény értéke 1 lesz, ha az (X, Y) koordinátájú pont világos és Ø lesz, ha az (X, Y) koordinátájú pont sötét a grafikus területen.

Ha az X, Y kifejezések értéke nem egész, akkor a koordinátákat a kifejezések egész része adja. Ez érvényes a PLOT utasításnál is.

Példa:
Irjunk programot, amely egy koordináta rendszerben kirajzolja a szinusz függvény egy teljes periódusát! Használjuk fel a maximális grafikus területet. Az origó a grafikus terület középpontjában legyen. A vízszintes tengely 2π, a függőleges tengely 2 egység hosszú legyen és rajzoljuk is meg a tengelyeket!

Megoldás:

```
 10 GR = 1:GL = 200:DL = 255:CALL 7535
 20 FOR X = 0 TO 319
 30 PLOT X, 100
 40 NEXT
 50 FOR Y = 0 TO 199
 60 PLOT 160, Y
 70 NEXT
 80 FOR X = 0 TO 319
 90 A = X/319 * 2 * 3.141592
100 Y = SIN (A - 3.141592)
110 PLOT X, - (99 * Y) + 100
120 NEXT X
```

7. Hanggenerálási lehetőségek

A hanggeneráláshoz egy újabb utasítást kell megismerni, ez a POKE (bök) utasítás. A POKE utasítás segítségével a RAM memória bármely bájtjába tetszőleges (0 és 255 közötti) értéket elhelyezhetünk. Az utasítás parancsként és programlépésként egyaránt használható. Az utasítás általános alakja:

POKE m, n_1, n_2, n_3, ..., n_k

ahol

POKE = az utasítás kulcsszava,
m = tetszőleges matematikai kifejezés, melyre $\emptyset \leq m < 65536$ egyenlőtlenség teljesül,
n_i = tetszőleges matematikai kifejezés, melyre $\emptyset \leq n_i < 256$ egyenlőtlenség teljesül ($i = 1, 2, ..., k$),
, = vessző.

Az utasítás hatására az m. bájtba n_1, az (m +1)-be n_2, ..., az (m + k - 1)-be n_k matematikai kifejezés értékének egész része kerül.

A hangkeltéshez a memória 16384. bájtjába 132-t, a 16385. bájtjába pedig 29-et kell elhelyezni.
Ezt az alábbi utasítással érhetjük el:

POKE 16384, 132, 29

Miután ezt a két bájtot feltöltöttük a megfelelő értékekkel az A = USR (256 * B + C) értékadó utasítással hanggenerálást érünk el. Az utasítás parancsként és programlépésként is használható. A keletkező hang magasságát a B, a hang hosszát a C kifejezés határozza meg. Az utasításban szereplő jelölések az alábbiakat jelentik:

A = tetszőleges numerikus változónév,
USR = az utasítás kulcsszava (függvény),
B = tetszőleges matematikai kifejezés,
C = tetszőleges matematikai kifejezés.

A B értéke Ø és 255 között változhat. A Ø értékhez tartozik a legmagasabb, a 255 értékhez a legalacsonyabb hang. A C értéke Ø és 127 között változhat. A Ø-hoz tartozik a legrövidebb,

a 127 értékhez a leghosszabb hang. (kb. C = 5Ø az a legkisebb érték, melytől felfelé a hang már jól hallható.)
A keletkező hang recsegő, dudaszerű, mert a gépnek hangkeltés közben is fel kell frissítenie a képernyő tartalmát, és emiatt a hangkeltést állandóan megszakítja. Ezért hanggenerálás esetén DL = Ø utasítással tiltsuk le a képernyőszerkesztést, ekkor tiszta, elektromos síp-szerű hangot hallunk.

Megjegyzés:

Az A = USR (256 * B + C) utasítás kiadható A = USR (K) alakban is. Ilyenkor a Ø K 65535 egyenlőtlenségnek kell teljesülnie. A hang magassága befolyásolja a hang hosszúságát és a hang hosszúsága is módosítja a hang magasságát.

Példa: Irjunk olyan programot, amelyben a gép az RND függvény segítségével "zenét szerez"!

Megoldás:

```
1Ø DL = Ø : POKE 16384, 132, 29
2Ø FOR I = 1 TO 5ØØ
3Ø A = INT (RND (6Ø)) : B = INT (RND (1ØØ))
4Ø A = A + 4Ø : B = B + 8Ø
5Ø X = USR (256 * B + A )
6Ø NEXT
```

8. Függelék

Karakterkód-táblázat

A 32 - 95 ASCII kódok a szokványos /billentyűzhető/ karaktereket jelölik a blanktól az alsó vonalig. Ugyanezek generálódnak a 160-224 kódok hátára is csak fehér alapon fekete rajzolattal.

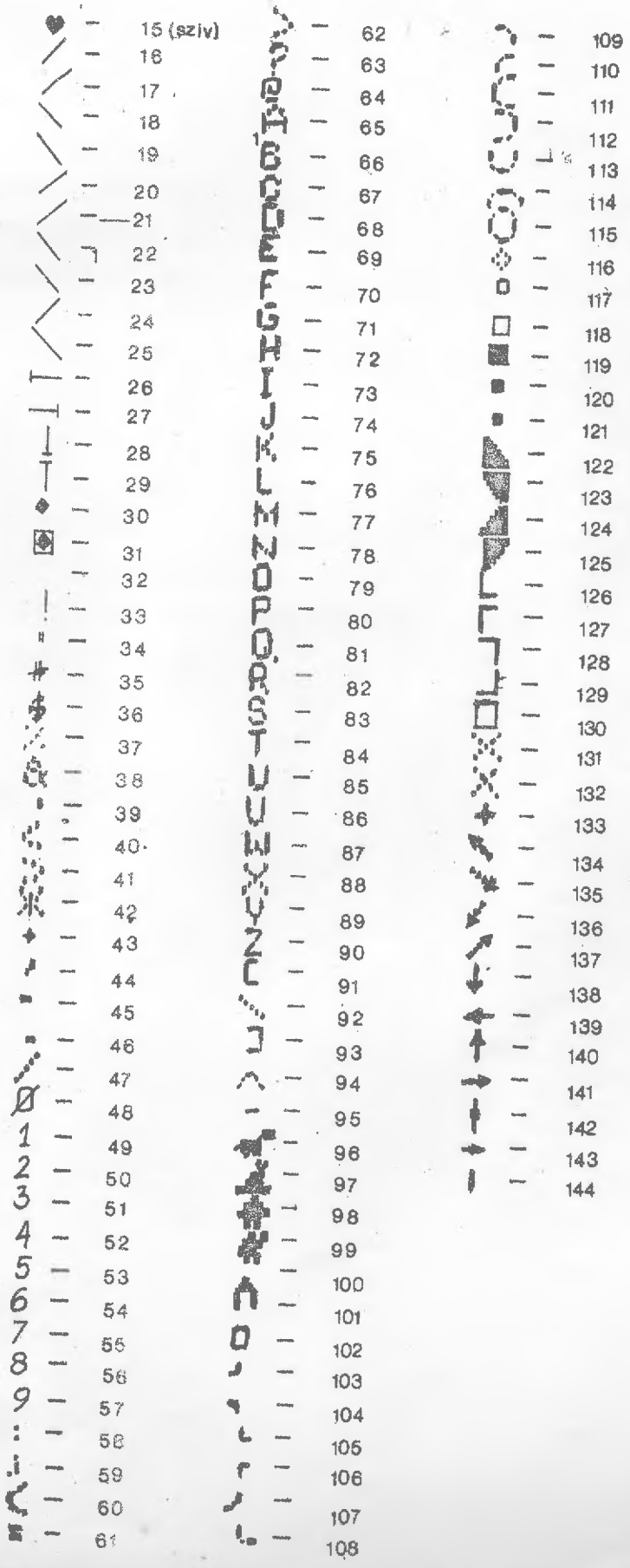
Cursor mozgató karakterek:

↓ - 8
↑ - 9
← - 10
→ - 11

Képernyő manipuláló karakterek:

Shift CR	- 12	képernyőtörlés
INS	- 6	beszúrás
DEL	- 7	törlés
CR	- 13	

Nem billentyűzhető karakterek



Hibakód-táblázat:

BS - méreten kívül eső tömbindex
CH - nem folytatható futás
IQ - az argumentum kívül esik az értelmezési tartományon
OD - több a READ utasítás, mint a DATA
OM - elfogyott az engedélyezett memóriaterület
OV - túlcsordulás
PP - NEXT FOR nélkül, RETURN GOSUB nélkül vagy POP FOR ill. GOSUB nélkül
SL - 255-nél hosszabb string
SN - szintaktikus hiba
TM - az előfordult kifejezés tipusa nem megfelelő
UF - definiálatlan függvény
US - definiálatlan sorra történt hivatkozás
/O - Ø-val való osztás

Tartalomjegyzék