Gyártó: **MICROKEY Kutatási Fejlesztési Termelési Társulás**
Bp. XIV., Varga GY. A. park 12/A 1149
Menedzser: **COSY Műszaki Fejlesztő Vállalat**
Bp: XI. Kende u. 13/17.

PRIMO

felhasználói kézikönyv

MICROKEY

COSY

PRIMO

felhasználói kézikönyv

TARTALOMJEGYZÉK

Szeretettel köszöntjük Önt a PRIMO tulajdonosok táborában. Örömünkre szolgál döntése, hogy az általunk gyártott otthoni számítógépet vásárolta meg.

Felhasználói kézikönyvünk kiadásával célunk az, hogy könnyen olvasható, könnyen érthető és mindemellett használható útmutatót adjunk az Ön kezébe. A PRIMO felhasználói kézikönyv tartalmazza mindazokat az információkat, amelyek segítségével Ön készülékét szakszerűen használni tudja és bizalommal fordulhat a hazai számítástechnika — méreteiben egyik legkisebb és legújabb — terméke felé.

Azoknak a PRIMO tulajdonosoknak, akik már járatosak a számítógépes programozásban nem szükséges a könyvet az elejétől végigolvasniuk.

Amikor a következő oldalakon elkezdi olvasni a PRIMO felhasználói kézikönyvét, szeretnénk felhívni figyelmét, hogy nem egy BASIC tankönyvet tart a kezében. Tankönyvként a mellékletben felsorolt kiadványokat ajánljuk figyelmébe.

Reméljük, hogy Ön miután megismerte a PRIMO működését, megtisztel bennünket és partnerünk lesz ötleteivel, javaslataival műszaki fejlesztéseinkben, programjaink írásában, az oktatásban. Ennek szellemében létrehoztuk a PRIMO Klubot.

Most pedig jó szórakozást és tanulást a PRIMO számítógéppel!

Üzembehelyezés

A PRIMO számítógép gyári csomagolásban került Önhöz. Kicsomagolás után ellenőrizze, hogy a készülék nem sérült-e. Sérülés esetén három napon belül a vásárlás helyén a számítógépet kicserélik.

Ellenőrizze a csomagolás tartalmát! A felhasználói kézikönyvön kívül a csomagolásnak tartalmaznia kell:

1 db PRIMO-t
1 db bemutató kazettát
1 db garancialevelet

Hiány esetén azt a vásárlás helyén pótolják.

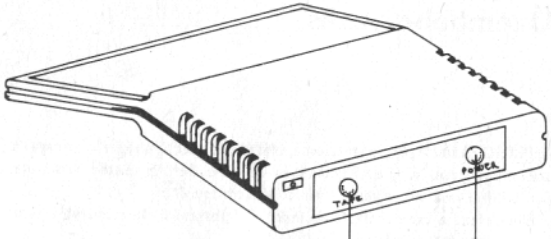
Most tekintsük át azokat a csatlakozási helyeket, ahol a PRIMO számítógéphez a tv és a magnetofon csatlakoztatható. Kapcsolja össze az alábbi utasításoknak megfelelően az egyes berendezéseket, *de még ne helyezze feszültség alá!*

CSATLAKOZÁSI HELYEK A PRIMO SZÁMÍTÓGÉPEN

A PRIMO számítógép hátoldalán a következő csatlakozási helyek találhatók (1. ábra):

1. **Tápfeszültség csatlakozás** (a POWER jelű tuchel)
   A PS A—01 vagy a PS A—02 típusú tápegység táp/video kábel csatlakoztatási helye.

2. **Magnetofon-csatlakozás** (a TAPE jelű tuchel)
   A magnetofon felvevő/lejátszó kábel csatlakoztatási helye.



1. ábra

TV KÉSZÜLÉK CSATLAKOZTATÁS

1. A TV-hez történő csatlakoztatást a 2. ábra mutatja.

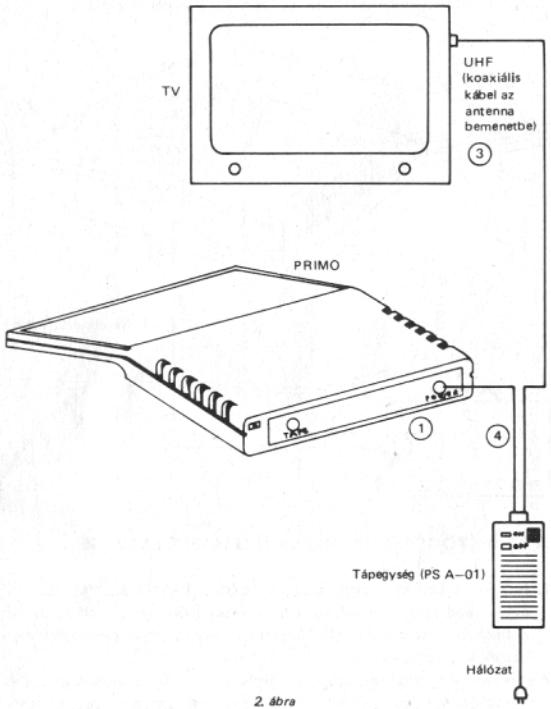
2. A tápegység TV koaxiális kábelét (3) csatlakoztassa a TV antenna (vagy video) bemenetére. A táp/video kábelt (4) pedig helyezze a PRIMO számítógép POWER jelű (1) csatlakozójába. Képernyőként bármilyen típusú normál TV készüléket használhat, mely alkalmas a TV 36—40-es műsorszóró csatornájának vételére, vagy video bemenettel rendelkezik. Ez utóbbihoz a PS A—02 típusú tápegység szükséges. A videobemenet használata jobb képminőséget eredményez.

3. A TV-készülék hangerejét csökkentse minimálisra.

MAGNETOFON-CSATLAKOZTATÁS

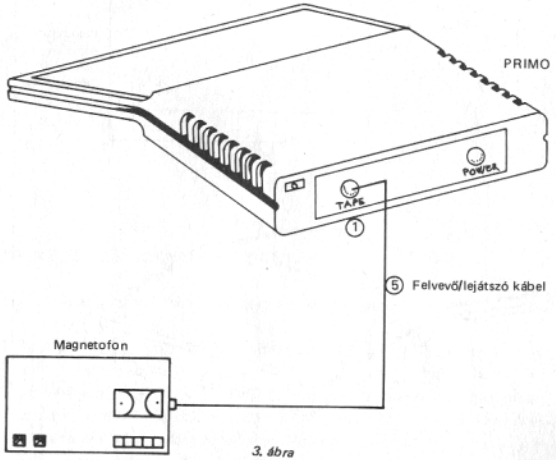
1. A PRIMO számítógép háttértárolóját a kiskereskedelmi forgalomban kapható bármilyen típusú magnetofon képezheti.

2. Ellenőrizze, hogy a magnetofon műszaki paraméterei kielégítik-e a PRIMO felhasználói kézikönyvben előírt követelményeket (lásd a magnetofonnal szembeni követelményeket: 1. Melléklet).



2. ábra

3. A magnetofon csatlakoztatását a 3. ábra szerint végezze el.



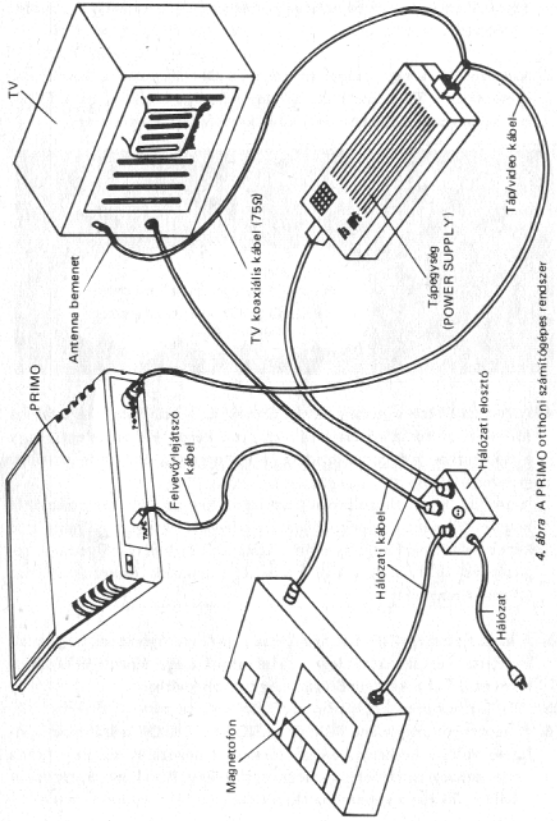
3. ábra

A SZÁMÍTÓGÉPES RENDSZER ÜZEMBEHELYEZÉSE

1. Ha már a fentiek szerint összekapcsolta a PRIMO számítógépet az egyes rendszerelemekkel, üzembehelyezés előtt kérjük, alaposan tanulmányozza át a **4.** ábrát! Ellenőrizze, hogy az egyes rendszerelemeket jól kapcsolta-e össze.

Figyelem! A PRIMO számítógépen nincs külön ki/be kapcsoló gomb. A számítógépet és a tápegységet kapcsolja össze, majd a tápegységet a hálózatba csatlakoztatva és *feszültség alá helyezve* kerül bekapcsolt állapotba a számítógép.

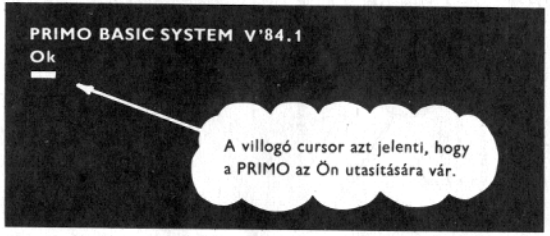
2. Csak azután helyezze feszültség alá a rendszert, miután meggyőző-



4. ábra A PRIMO otthoni számítógépes rendszer

dött az összekapcsolás helyességéről. Most csatlakoztathatja az elosztót a hálózatba.

3. Kapcsolja be a tápegységet (nyomja be ON állapotba a bekapcsoló gombot) és a TV készüléket. Néhány másodperc múlva, ha a TV jól van behangolva, a képernyőn a következő felirat jelenik meg:



4. A tv-készüléket kézihangolással Önnek kell beállítani, megtalálni a 36—40-es műsorszóró csatornák között a képet. (Ez azt jelenti, hogy a csatornahangoló gombját addig kell elforgatnia, amíg a fenti felirat meg nem jelenik.)
Javasoljuk, hogy hangoláskor pásztázza végig az UHF tartományt és miután túlhaladt a képen, többszöri oda-vissza hangolással lehet a jó képet beállítani. Hangolás után a kontraszt és a fényerő gombok segítségével állítható be a kívánt minőség (minimális kontraszt álláshoz állítsuk a fényerőt).

5. A képernyő megvilágítását úgy állítsa be a fényerőgombbal, hogy a felirat háttere egészen sötét legyen, a felirat pedig egészen világos. A kontraszt-gombbal a kép minőségét még tovább javíthatja.

6. A képernyőn megjelenő PRIMO BASIC SYSTEM Ok felirat alatt a világos, villogó keskeny vonalat *cursor*-nak nevezzük. Ez megmutatja azt a pozíciót, amelybe a klaviatúra egy billentyűjének megnyomásakor a billentyű képe a képernyőre kerül.

7. Amennyiben a PRIMO számítógép a leírtakat a képernyőn nem jeleníti meg, vizsgálja át ismét a rendszer felépítését a következő táblázat alapján.

Hibakeresés

Jelenség	Ok	Megszüntetés
A tápegység piros kapcsolólámpája nem világít	Nincs kapcsolat a hálózathoz	Vizsgálja meg, hogy a kapcsoló On állásban van-e
	Hálózatkimaradás	Kapcsolja ki az egész készüléket
	A biztosíték meghibásodott	Kérjen meg szakembert ennek cseréjére
Nincs kép a képernyőn	Hibás csatorna beállítás	Ellenőrizze a csatornaváltó beállítását
	Az antenna- ill. a táp/video kábel nincs jól csatlakoztatva, vagy rossz	A szóbanforgó kábeleket ellenőrizze
Erős zúgás a TV-ből	A hangerő túlságosan nagy	Csavarja a hangerőt minimumra

GYAKORLATI PÉLDA

1. Helyezze üzembe az Ön PRIMO számítógépes rendszerét a 10. oldalon leírtak szerint.

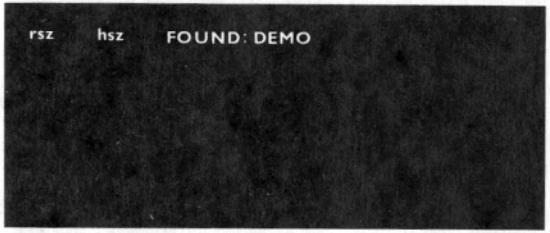
2. Helyezze a bemutató kazettát a magnetofonba. A színtszabályozót állítsa közepes szintnek megfelelő állásba.

3. Billentyűzze be a „LOAD" BASIC utasítást.

4. Nyomja meg a RETURN parancs billentyűt.

5. Indítsa el a magnetofont „lejátszás" állapotban.

6. Ezután automatikusan beolvasódik a kazetta első file-ja. A beolvasás alatt megjelenik:



Figyelem! Ha a beolvasásnál igen sok hiba fordul elő, vagy egyáltalán nem sikerül a beolvasás, javasoljuk a magnetofon alaposabb ellenőrzését fejkopás és fejtisztaság szempontjából. Szükség lehet a törlő- és kombinált fej alapos megtisztítására alkoholos, vagy denaturált szesszel átitatott vattával, esetleg szervízben történő beállítására, cseréjére.

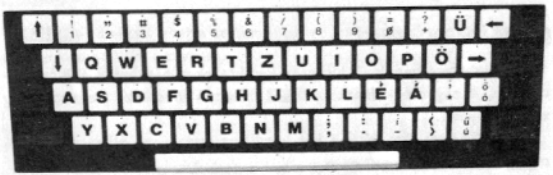
7. Ha a bemutató kazetta beolvasható, a berendezés működőképes, a BASIC programnyelv szabályai szerint használhatja a PRIMO otthoni számítógép-konfigurációt.

Funkcionális billentyűk

A PRIMO SZÁMÍTÓGÉP MŰKÖDTETÉSE

BILLENTYŰZET

Szánjon néhány percet arra, hogy megismerkedjen a billentyűzettel, mivel ez a PRIMO számítógép legfontosabb része a rendszerrel való kommunikációban.



1. A PRIMO számítógép klaviatúrája a hagyományostól eltérő, nyomtatott áramköri technológiával készített, érintésre érzékeny billentyűkből áll.

2. A billentyűzet elrendezése megegyezik az írógép betűelrendezésével. A legtöbb billentyűvel két különböző jelet lehet megjeleníteni.

3. A billentyűmegnyomás szubjektív érzésének elmaradásából adódó bizonytalanság kiküszöbölése érdekében a számítógép egy rövid hangjelzést ad, ha valamelyik billentyűt megérintjük és annak kódját beolvasta.

4. A billentyűzet összesen 60 db billentyűt tartalmaz, amelyek közül 53 az alfanumerikus karakterek, számok ill. különböző jelek bevitelére alkalmas (fehér alapszínű billentyűk).

5. A 7 db zöld alapszínű billentyű különböző vezérlő, ill. kód-módosító funkciókra használható.

A funkcióbillentyűk

CTR

A CTR-billentyű (a control rövidítése) mindig egy másik, fehér alapszínű billentyűvel együtt használható. A vele együtt lenyomott billentyűhöz tartozó kód értékét (lásd karakterkód táblázat, 3. Melléklet) módosítja.

UPPER

Az alaphelyzetben levő kisbetűs karaktereket nagybetűs állásra kapcsolja át. A nagybetű-állapot az UPPER billentyű következő megnyomásáig van érvényben.

Az UPPER nem módosítja a két karaktert tartalmazó billentyűk értelmezését!

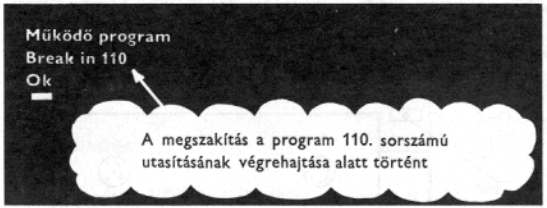
SHIFT

A SHIFT-billentyű megfelel az írógépeknél használatos kisbetű/nagybetű váltónak.

A két karaktert vagy jelet tartalmazó billentyűk SHIFT -billentyűvel egyidőben történő megnyomása a billentyűn felül lévő karakter vagy jel bevitelét eredményezi.

BRK

A BRK-billentyű megnyomásakor megszakad a gépen futó BASIC program. A képernyőn a következő üzenet jelenik meg:



Az automatikus sorszámozás üzemmódban alkalmazott megszakítás csak az Ok üzenetet eredményezi. A megszakítások után a cursor a következő sor első karakterpozíciójára áll.

RETURN

A RETURN-billentyű lezárja az aktuális sort a következő sor első karakterpozíciójára állítva a cursort.

Parancs üzemmódban megnyomása a parancs végrehajtását is indítja.

CLS

A CLS-billentyű lenyomásakor a képernyő törlődik és a cursor alaphelyzetbe (a bal felső sarokba, az ún. HOME pozícióba) áll.

Repeat — üzemmód

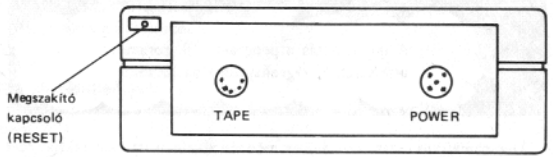
1. A billentyűzet lényeges funkciója a repeat-üzemmód, amely azt jelenti, hogy a billentyűt folyamatosan lenyomva (érintve) tartva a kiválasztott karakter kb. 0,1 másodpercenként ismétlődik.

2. Az UPPER-billentyű lenyomásának hatása az újbóli lenyomásig érvényes.

Gyakorlati tanács! A billentyűzet az érintésre kellően érzékeny, de célszerű a kiválasztott billentyűt minél nagyobb felülettel érinteni.

MEGSZAKÍTÓ KAPCSOLÓ (RESET)

A PRIMO számítógép hátlapján található a megszakító kapcsoló:



5. ábra

A megszakító kapcsoló megnyomásakor a processzor alapállapotba kerül, de a programot nem törli. A képernyőn ezután az Ok felirat és a villogó cursor jelenik meg.

A PRIMO számítógépes rendszer működése

PROGRAMÍRÁS

1. Kapcsolja össze a PRIMO számítógépes rendszert az 1.4 pont szerint. A rendszer bekapcsolásakor a képernyőn a következő üzenet jelenik meg:

```
PRIMO BASIC SYSTEM V '84.1
Ok
```

2. Az üzenet kiírása után a cursor a következő sor elején áll.

3. Ezután Ön máris elkezdheti a programírást BASIC nyelven.

Figyelmébe ajánljuk a 4. Mellékletben található DEMO programlistát. Írja be a programlistát és próbálja ki a rendszert.

4. Írja be a RUN parancsot.

5. Zárja le a sort RETURN -nel.

KÉPERNYŐKEZELÉS

A képernyő 16 sornyi szöveg kiírására alkalmas. Az utolsó sor betelte után a képernyőn megjelenített szöveg automatikusan egy sorral feljebb csúszik. A következő sor beírása a képernyő utolsó sorába történik.

A képernyő mindig az utoljára beírt 16 szövegsort tartalmazza.

A képernyő a CLS gomb megnyomásával, vagy a programban kiadott utasítás hatására teljesen törlődik. A törlés a gépben tárolt programot nem befolyásolja, de a RETURN karakterrel le nem zárt sorok törlődnek.

A képernyő törlése után a cursor alaphelyzetbe (bal felső sarok) áll.

MAGNETOFONKEZELÉS

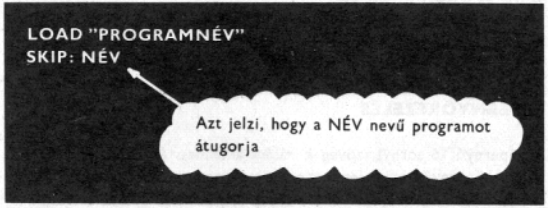
Felvétel és lejátszás esetén célszerű a magnetofon szintszabályozóját közepes állásba (ha van szintjelző, akkor a 0 dB szintre) állítani.

Beolvasás

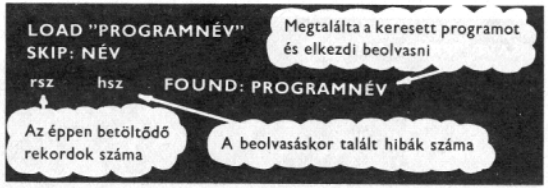
1. A kazettát helyezze a magnetofonba.
2. A billentyűzeten írja be a következő parancsot: LOAD "PROGRAMNÉV".
3. Nyomja meg a RETURN gombot.
4. Indítsa el a magnetofont (play).

Ezután a PRIMO számítógép automatikusan keresni kezdi az adott nevű programot.

Ha a szalag aktuális helyzete és a keresett program között más programok is vannak, akkor a képernyőn a következő sor elején az alábbi üzenet jelenik meg:



5. A keresett program megtalálásakor a következő felirat jelenik még meg a képernyőn:



A betöltés csak akkor hibátlan, ha hsz = 0 0.

6. A hibátlanul betöltött program a RUN parancs beírásával indítható. Utána a RETURN billentyűt meg kell nyomni.

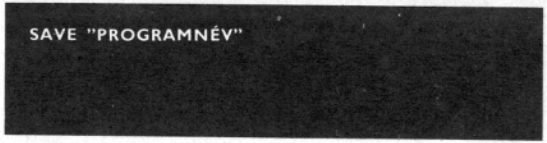
7. A beolvasás a RESET gomb megnyomásával szakítható meg.

Ha a rendszer a kívánt programot a nem megfelelő lejátszási szint miatt nem találná, akkor a szint beállítása és a szalag visszatekercselése után a beolvasás LOAD parancs megismétlése nélkül automatikusan folytatódik.

Tárolás

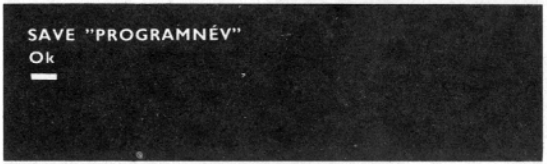
1. Billentyűzze be az Ön által írt programot a számítógépbe.

2. Billentyűzze be a következő parancsot:



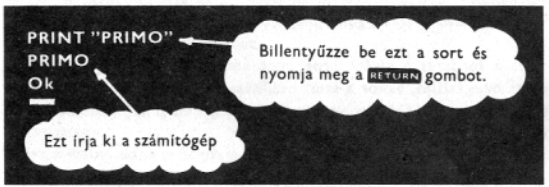
3. Indítsa el a magnetofon-felvételt.

4. A program kimásolása után a képernyő állapota:

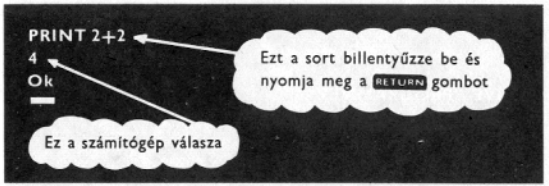


A PRIMO, MINT SZÁMOLÓGÉP

Mivel Ön már ismeri a PRIMO számítógép kezelését, most billentyűzze a következőket és vegye figyelembe a kommentárt:



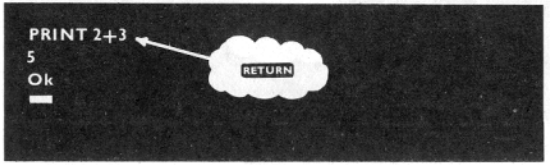
Billentyűzze be az alábbiakat:



Hasonlítsa össze, mi történik akkor, ha a 2+2-t idézőjelbe teszi!

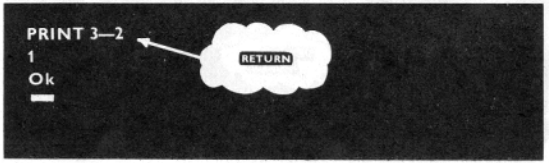
Összeadás

Az összeadás jele: +.



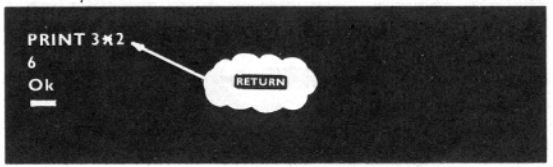
Kivonás

A kivonás jele: —



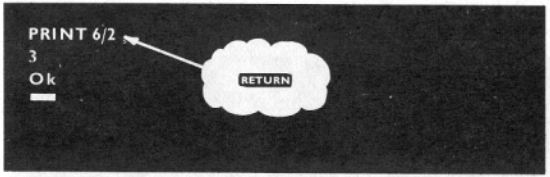
Szorzás

A szorzás jele: *



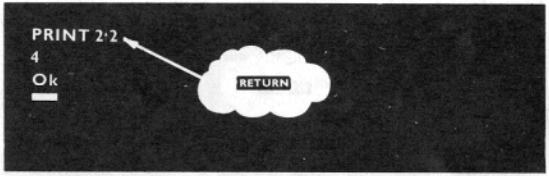
Osztás

Az osztás jele: /



Hatványozás

A hatványozás jele: ↑



	A beírható legkisebb	legnagyobb
Egész típusú adat	— 32768	+32767
Valós típusú adat	$-9{,}9\times10^{-38}$	$+1{,}70\times10^{38}$

ARITMETIKAI KIFEJEZÉSEK KISZÁMÍTÁSA

Az aritmetikai kifejezések kiértékelését a prioritási szabály és a balról jobbra szabály határozza meg.

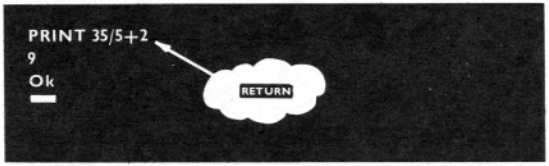
A kiértékelés sorrendje:

1. Először mindig a magasabb prioritású műveletek hajtódnak végre. (A prioritási sorrend kerek zárójelek alkalmazásával módosítható.) A műveletek prioritása, csökkenő sorrendben:
   — hatványozás (↑)
   — előjelváltás [negálás] (—)
   — szorzás (*), osztás (/)
   — összeadás (+), kivonás (—)

2. Az azonos prioritású műveleteket a BASIC balról jobbra haladva értékeli ki.

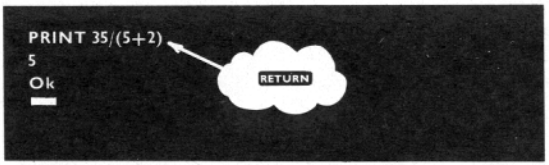
Egy aritmetikai kifejezésen belül egymás mellett két műveleti jel nem állhat, ez alól csak a negálás kivétel.

Írja be a következő példát:



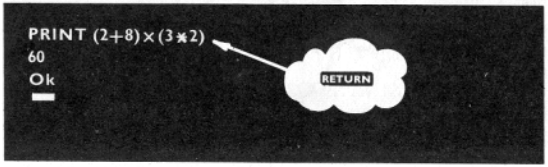
Ez azt jelenti, hogy a számítógép a 35-öt elosztja 5-tel (=7) és az eredményhez hozzáadja a 2-t (= 9).

Tegye a 5+2 összeget kerek zárójelbe.



Most a PRIMO számítógép először a zárójelben levő műveletet végzi el (= 7) és a 35-öt az eredménnyel osztja (= 5).

A következő példában a számítógép először mindkét zárójeles műveletet elvégzi és azután a szorzást:



Karbantartás

A PRIMO otthoni számítógép üzemeltetése során karbantartást nem igényel.

Az esetleges hibák előfordulása esetén számítógépét az alábbi márkaszervizben javítják:

PRIMO Márkaszervíz
Budapest, XIV.,
Varga Gyula András park 12/A
1149

Mellékletek

Az Ön által megvásárolt PRIMO otthoni számítógépet a MICROKEY Kutatási Fejlesztési Termelési Társulás fejlesztette ki és gyártotta. A Társulásnak a következő cégek a tagjai:

Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézet
Elektromodul
„Új Élet" MgTsz Sárisáp

A MICROKEY KFTT 1983 tavaszán alakult meg. S egy év alatt a teljes innovációs folyamatot megvalósította a PRIMO forgalombahozatalával.

Az innovációs folyamat menedzselését a COSY Műszaki Fejlesztő Vállalat végezte.

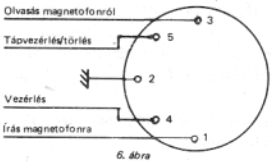
Az Ön gépe a PRIMO otthoni számítógép. Egykártyás kivitelű, klaviatúrája érintős, kapacitív elven működik. A számítógép alkalmas BASIC nyelven és gépi kódban írt programok futtatására. A számítógépes rendszer megjelenítő egységét bármely típusú TV-készülék képezheti, mely csatlakoztatható TV műsorszóró csatornán, vagy közvetlen video bemeneten. A rendszer háttértárolójaként kiskereskedelmi forgalomban kapható, bármilyen típusú magnetofonkészülék felhasználható. A központi egység energiatáplálására önálló tápegység szolgál.

Az otthoni számítógép sikeresen alkalmazható:
— a háztartásokban
— az oktatásban
— az iparban
— az államigazgatásban
műszaki, gazdasági, ügyviteltechnikai stb. feladatok ellátására.

MŰSZAKI ADATOK

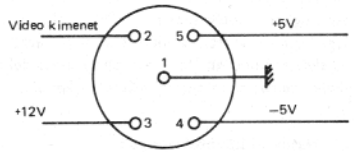
Megnevezés: PRIMO
Típus: A—32, A—48, A—64
Mikroprocesszor: 8 bites (U 808, NDK gyártmány)
Órajel frekvenciája: 2,5 MHz
Operatív tár kapacitása:
A—32 16 Kbyte RAM
16 Kbyte ROM (BASIC interpreter tárolására)
A—48 32 Kbyte RAM
16 Kbyte ROM (BASIC interpreter tárolására)
A—64 48 Kbyte RAM
16 Kbyte ROM (BASIC interpreter tárolására)
Programnyelv: BASIC
Billentyűzet:
— működési módja: érintős (kapacitív)
— billentyűk száma: 60
— jelkészlet: számok (0—9)
magyar ABC: nagybetűk (kivéve Í, Ő, Ú, Ű) kisbetűk
írásjelek, matematikai jelek (, ; : . — ! ? " ' * / ( ) #
$ % & < > = ↑
— vezérlő billentyűk: CTR, UPPER, SHIFT, BRK, RETURN, CLS, ↑ ↓ → ←
— érintés visszajelzése: hanggal
— kis- és nagybetűs váltás: rögzíthető
— kontroll-karakter előállítás: lehetséges A—Z-ig
Csatlakozási pontok:
— video kimenet: min. 1 $V_{cs—cs}$, 75Ω
— magnetofon felé kimenet: 80 mV
impedancia: 200 Ω
— magnetofon felől bemenet: min. 1 mV
impedancia: 4,3 kΩ
— energiafelvétel: +5 V, 750 mA
+12 V, 180 mA
—5 V, 8 mA

Csatlakozási pontok bekötése:
— TAPE jelű tuchel csatlakozó



6. ábra

— POWER jelű tuchel csatlakozó



7. ábra

Működési hőmérséklet: +5 C°— +40 C°
Méretek: 45×260×310 mm
Tömeg: 800 g.
Tartozékok: Felhasználói kézikönyv, bemutató kazetta, garancialevél, csomagolás
opcionális: tápegység (PS A—01 vagy PS A—02).

Képernyővel szembeni követelmények

Csatlakoztatható készülék: bármilyen típusú TV-készülék.
Csatlakoztatási lehetőség: TV műsorszóró csatornán (36-os 40-es csatorna között, 1 $mV_{cs—cs}$, 75Ω).

Megjelenítés a képernyőn

Megjelenítés módja: grafikus 192 rasztersor, soronként 256 raszterpont
Karakter megjelenítés: 16 sor, soronként 42 karakter
Karakter mátrix mérete: 6×12
Megjelenített karakterek: 98 — a billentyűzeten megjelölt, különleges karakterek (ROM-ban tárolva): π, [,], →, ←, ↓, ↑, |, ⌐, §, ≠, @, {, }, ʃ, ~, μ, Ω, ², ∞, ∑, Π, Ft
Definiálható karakterkészlet: 128 db (RAM-ban tárolva).
Indexelés: n-ed fokon alsó, felső
Karakterszélesség: normál (6 raszterpont), nyújtott (12 raszterpont).
Grafikus ábrák és alfanumerikus jelek keverhetősége: tetszőleges.
Aláhúzás: programból vezérelhető.
Karakter-alap megvilágítása: sötét (normál), világos (inverz).
Képernyő-alap megvilágítása: sötét (normál), világos (inverz).
Karakterpozíció kijelölése: kötetlen (sor, oszlophely nincs definiálva).
Megjeleníthető ábrák száma: programból vezérelve két darab

Magnetofonnal szembeni követelmények

Bemeneti ellenállás: 20 kΩ—2 MΩ.
Bemeneti jelszint: min. 80 mV.
Kimeneti ellenállás: 5 kΩ—25 kΩ
Kimeneti jelszint: min. 1 mV
Torzítás: (2 kHz és 4 kHz között) max. 15%.
Szalagsebesség ingadozás: ±5%.

Magnetofon kezelés

Rögzítési mód: háromszintű impulzus moduláció
Rögzítési sebesség: 100 byte/sec
Magnetofon típusa: kiskereskedelmi forgalomban kapható bármely típus (mely kielégíti a megjelölt követelményeket).
Rögzíthető file: BASIC program file
adat file

Beolvasható file-ok: BASIC program file
adat file
gépi szintű program file
Beolvasás jelzése: rekord számlálás, hibaszám kijelzés
File-név hossza: 1—16 karakter (bármilyen).

PRIMO—BASIC GÉPI REPREZENTÁCIÓ

1. Üzemmódok, parancsok

A BASIC gépi reprezentáció *parancs-* és *programmódot különböztet meg.*

A *parancsmód jellegzetessége* a beírt parancs, vagy bizonyos BASIC utasítás a sor lezárása RETURN után azonnal végrehajtódik.

A programmódban a korábban beírt program végrehajtása folyik. Parancsmódból programmódba például RUN parancs kiadásával térhetünk át.

Mindkét üzemmódban a parancsokat vagy utasításokat a billentyűzetről visszük be. A bevitel alapegysége a *parancssor* vagy programsor, amely a képernyőn megjelenített fizikai sortól eltérően *max. 210 karaktert tartalmazhat* (öt fizikai sor). Ennél több karaktert a rendszer nem fogad el, a sor beteltét hangjelzéssel jelzi. A sorhosszba a felesleges betűközök beszámítanak, viszont a sort záró RETURN *nem*. A parancs-vagy programsor a rendszer által egy lépésben kezelhető max. információ hosszát jelenti. A BASIC kulcsszavait a tipikus felhasználásuk alapján parancsokra és utasításokra szokás osztani, de ez a felosztás nem egyértelmű, mivel a parancsok egy része a programba beépítve is használható, amíg az utasítások egy része parancsként (azaz a sor lezárását követően azonnal végrehajtódó funkcióként) is kiadható. Az alábbiakban ismertetjük a BASIC gépi reprezentációban értelmezett parancsokat:

Parancsok:

RUN — Elindítja a programtárolóban lévő BASIC nyelvű programot. Az indítás a program elejéről, a legkisebb sorszámú sortól (RUN) vagy a megadott (nn) számú sortól (RUN nn) kezdődik. A RUN nn forma használatakor nemlétező sorszámra hivatkozás az *UL Error* hibaüzenetet eredményezi. A programtároló törlése után kiadott RUN parancs a következő sor elejére kiírt *Ok* üzenetet eredményezi és a rendszer parancs üzemmódban marad.

AUTO — A programírás segítése érdekében automatikusan generálja a sorszámot. Az AUTO parancs hatására a sorszámozás a 10-zel kezdődik, 10-es lépésközzel növekszik. Az AUTO nn parancs a sorszámozást az nn számú sorral kezdi. Az AUTO nn, mm formájú parancs a sorszámozást az nn számú sorral kezdi és a sorszám növekvény lépésköze mm (az előző változatok ennek speciális eseteként is felfoghatók, amelyeknél a rendszer az elhagyott paraméterek helyébe 10 értéket helyettesít). Az nn értéke csak pozitív, egész szám lehet és értékét a kiadható max. sorszám (65529) korlátozza. Értelemszerűen hasonló szabályok vonatkoznak mm értékére is. Az automatikus sorszámozás üzemmódból csak a BRK vagy RESET gombok megnyomásával lehet kilépni. Egy már meglevő program automatikus sorszámozású bővítésekor az esetleges felülírások elkerülése érdekében a már létező programsorokat az automatikusan kiírt sorszám után elhelyezett * (csillag) karakter jelzi.

LIST — A programtároló tartalmának listázása. A LIST parancs a teljes programot listázza a képernyőn.

A listázás folyamatosan történik, és a ↓ karakterrel szakítható meg. Újraindítása bármelyik gomb megnyomásával történik. A LIST nn formájú parancs az nn számú sort írja ki, ha van ilyen (ha nincs, akkor nem keletkezik hibajelzés, hanem az *Ok* üzenet kiírásával jelzi a rendszer a parancs formális végrehajtását). A LIST nn—mm változat az nn és mm számú sorok közti tartomány (az nn- és mm-edik sort is beleértve) kilistázását kéri. Amennyiben nn, ill. mm nem létező sorszámok, akkor a listázás az nn-t követő első létező sortól kezdődik és az mm-et megelőző, nálánál kisebb számú létező sorral fejeződik be. A parancsból nn vagy mm elhagyható (a kötőjelet viszont ki kell írni), és ilyenkor a program az első (legkisebb számú) sortól az mm-edik sorig, ill. az nn-edik sortól az utolsó (legnagyobb számú) sorig történik.

NEW — Törli a programtárolót. A programírást célszerű ezzel az utasítással kezdeni, hogy a korábbi programrészleteknek az új programba való beépülését elkerüljük.

CONT — A BRK gomb lenyomásával megszakított vagy a STOP ill. END utasításokkal leállított program újraindítása a megszakítást vagy leállítást követő utasítástól. A program futása úgy folytatódik, mintha nem történt volna megszakítás vagy leállítás.

DELETE$_{nn}$ — A megadott sorszámú sor törlése. A parancs az nn elhagyásával nem használható. Két sorszám megadása esetén törli a két sorszám közti sorokat (az nn- és mm-edik sort is beleértve). A sorszámoknak létező sorszámoknak kell lenni. A DELETE-$_{mm}$ változat az mm-edik számú sorig (mm-ediket is beleértve) törli a programot.

CLS — A CLS billentyűhöz hasonlóan törli a képernyőt és a cursort a HOME pozícióba (balfelső sarok) állítja.

TRON — Bekapcsolja a nyomkövetést. Nyomkövetési üzemmódban a rendszer a képernyőn [nn] formában kiírja a végrehajtódó utasítássorok sorszámát (a REM sorokat is beleértve). A kiírással egyidejűleg minden más, a programban előírt funkciót is végrehajt, így az esetleges eredmény kiíratás a sorszámok közt fog elhelyezkedni. A parancs a BASIC nyelvű programok belövését segíti.

TROFF — Kikapcsolja a nyomkövetési funkciót. A TRON parancs inverze.

CLEAR$_n$ — Karakteres változókhoz rendelt tárolóterület méretének definiálása. Bekapcsoláskor a karakteres változók számára a rendszer 50 byte-ot tart fenn, ami a CLEAR n parancs hatására n byte-ra módosítható. Az n szám elméleti max. értéke a rendelkezésre álló tároló méretétől függ, és alapkiépítésű számítógép esetén 9450 lehet (de ilyenkor a program számára már nem marad hely). A beállított érték csak egy újabb CLEAR n utasítással módosítható.

EDIT$_{nn}$ — A program nn-edik sorának javítás és módosítás céljából történő elővétele. A parancs használatával a 11. pontban fogunk részletesen foglalkozni.

SAVE "NÉV" — A programtároló tartalmának "NÉV" programnévvel kazettára történő felvétele.

LOAD "NÉV" — A kazettán "NÉV" néven tárolt program betöltése a tárolóba. Ha a "NÉV" programazonosítót elhagyjuk, akkor az első megtalált programot tölti be.

TEST "NÉV" — A kazettán "NÉV" programnévvel rögzített program ellenőrzése a rögzített file formátumának ellenőrzése alapján.

Mint azt a bevezetőben említettük, a parancsok általában a program utasításai közé beépítve is használhatók (a funkciója miatt értelemszerűen kivétel a CONT), de a parancsok többsége a program futása közben végrehajtódva a vezérlést nem a programnak adja vissza (parancs üzemmódra tér át a rendszer).

2. Programozott üzemmód

Programozott üzemmódban a rendszer a programtárolóban elhelyezett BASIC nyelvű programot hajtja végre. A program sorszámozott, egyenként max. 210 karakteres sorokba írt BASIC utasításokból áll. A sorszámozás tetszőleges lehet, de vezérlésátadó utasítások hiánya esetén a program a sorszámok növekvő sorrendjében hajtódik végre. A max. sorszám 65529 lehet. A tárolóban lévő program *a RUN paranccsal indítható*, és elindulása után a felhasználó és a számítógép közti kapcsolat a billentyűzet és a display felhasználásával jön létre.

3. Azonosítók és változók

A programon belül a programozó az egyes adatait konstans formájában vagy szimbolikus névvel ellátva használja. A szimbolikus névvel ellátott adatokat azonosítóknak nevezzük, és az azonosító legfőbb jellemzője az általa reprezentált adat típusa. A programban az egyes azonosítók közt formális műveletek jelölhetők ki, amelyek csak a program futása során fognak végrehajtódni. A végrehajtás feltétele, hogy az egyes azonosítók-

hoz a típusuknak megfelelő értéket rendeljünk. Az értékkel rendelkező azonosítót változónak nevezzük. A BASIC gépi reprezentáció a változók értékét a program lefutása után is megőrzi mindaddig, amíg nem módosítjuk a programot. A rendszer az értéket nem kapott numerikus változókhoz nulla kezdőértéket rendel.

A BASIC gépi reprezentációban használt azonosítók egy névből és egy típusjelző karakterből állnak. A név tetszőleges hosszúságú lehet (de csak az első két karakterét különbözteti meg a rendszer), első karaktere csak az angol ABC karaktereiből állhat, és a további karakterei számjegyek vagy az angol ABC karakterei lehetnek. A jelkészletben szereplő ékezetes karakterek használata nem megengedett, csakúgy, mint a különböző jeleké. A név (azonosító) nem egyezhet a BASIC kulcsszavaival sem (ill. beágyazva nem tartalmazhat kulcsszavakat).

Az azonosítókat, ill. változókat az általuk reprezentált adat típusa alapján az

— egész,
— valós (egyszeres pontosságú),
— kétszeres pontosságú és
— karakteres

azonosítók, ill. változók csoportjába sorolhatjuk.

Egy másfajta csoportosítás szerint megkülönböztethetünk

— elemi (egyedi) azonosítókat és változókat, amelyek jellegzetessége, hogy egy azonosítóhoz egyetlen érték rendelhető, ill. egy változó csak egyetlen értékkel rendelkezik, és
— indexes (tömb) azonosítókat és változókat, amelyeknél az azonosító egy rendezett (egy vagy több szempont szerinti rendezést feltételezve) adathalmazt jelöl ki, és az adathalmazon belüli egy érték kijelölése az azonosítóhoz csatolt indexszel történik.

A következőkben az egyes adatfajták jellegzetességeit és felhasználási módjait foglaljuk össze.

Elemi adatok

Mint az előző fejezetben már kifejtettük, az elemi adatok jellegzetessége, hogy egy azonosító egyetlen adatra való hivatkozást jelöl, ill. egy elemi változóhoz egyetlen érték tartozik. Az azonosító, ill. változó típusa numerikus vagy karakteres lehet.

Numerikus adatok

Az egész típusú adatokat a BASIC gépi reprezentáció két byte-on (16 biten) tárolja, így értéke —32768 és +32767 közé eshet. Az egész típusú adatok jellegzetessége, hogy nem tartalmaznak tizedespontot (azaz a program futása során soha nem kaphatnak olyan értéket, amelynek van törtrésze). Ilyen értelemben a 35, — 78, 12784 stb. egész típusú konstansok, viszont a 35.0, — 78.0, 12784.00 olyan valós konstansok, amelyeknek a törtrésze éppen nulla (de a program futása során ez változhat, szemben az egész típusú adatokkal). Az egész típust az azonosító vagy konstans után írt % jel jelzi. Ennek hiányában a rendszer az azonosítót vagy konstansot (még egész típusú érték esetén is) valós típusúnak tekinti. Egész típusú azonosító pl. az A%, BX%, I% stb. és egész típusú konstans a —65%, 983%, —4% stb.

A típusjelző %-jel hatására a rendszer (ha ez lehetséges) elvégzi a típus-konverziót, azaz konstans esetén levágja a törtrészt és értékadás esetén az egész típusú azonosítóhoz az érték egészrészét rendeli. A konverzió nem végezhető el, ha az érték nagysága meghaladja az egész típusú adatokra megengedett értékhatárokat (ilyenkor hibajelzés generálódik).

Valós típusú adatok. A BASIC gépi reprezentáció alapfeltételezés szerinti változó-típusa (azaz minden azonosítót vagy konstansot valósnak tekint a rendszer, ha másképpen nem definiáljuk). A BASIC a valós adatokat négybyte-os, lebegőpontos alakban tárolja. A tárolási forma következtében a valós adatok abszolútértékének a $9.9 \cdot 10^{-38}$... $1.70 \cdot 10^{38}$ tartományba kell esni, és az így tárolt szám hat decimális jegyre pontos. A nagyságrend túllépése esetén a rendszer hibajelzést generál, hat jegynél több számjegyet tartalmazó adatok esetén pedig hat jegyre kerekít. Minden, más módon nem jelzett azonosítót vagy konstansot a rendszer valós típusúnak tekint, de megengedett a valós adatok megkülönböztetése az azonosító vagy konstans után írt ! jellel. Ennek alapján valós típusú konstans a —13.78, .035, (az értéktelen vezető nullák és a pozitív

előjel kiírása nem kötelező), 6E+23 (ahol az E az exponens jelzésére szolgál és az exponens — 38... +38 közé eshet), 12345.6789 stb., ill. —57.8!, 623.15! stb. Valós típusú azonosítót jelöl az AX, DD, R!, L7! stb. A típuskonverzió valós adatok esetén is automatikusan létrejön, és mindig elvégezhető.

*A kétszeres pontosságú adatok*at a BASIC nyolc byte-os formában (a valós változóknál használt hossz kétszeresén) tárolja. A kétszeres pontosságú adatok lehetséges nagyságrendje megegyezik a valós típusú adatok nagyságrendjével, de az így tárolt adatok 16 decimális jegyre pontosak.

A kétszeres pontosságú azonosítók és konstansok megkülönböztetése a # jellel történik. A kétszeres pontosságú konstansokra példa: —1#, .8323456789#, 3.141592#, 137.00015D—6#. Az alábbi azonosítók kétszeres-pontosságú adatot reprezentálnak: XX#, KL#, PX#, U8# stb. A kétszeres pontosságú adatoknál végrehajtódik az automatikus típuskonverzió. Itt hívnánk fel a figyelmet arra, hogy a kétszeres pontosságú változókkal elérhető nagy számolási pontosságot nagymértékben lerontható, ha a kétszeres pontosságú változókat a valós változókkal keverten használjuk.

Karakteres adatok

A karakteres adatok vagy karakterfüzérek tetszőleges alfanumerikus információk tárolására alkalmasak. A BASIC gépi reprezentáció a karakterfüzéreket a numerikus adatoktól elkülönítve tárolja, és alapállapotban a karakterfüzérek tárolására fenntartott hely 50 karakter (byte) hosszú. Ettől a CLEAR n parancs alkalmazásával térhetünk el, ahol n értékére vonatkozó megfontolásokat lásd a CLEAR parancs leírásánál. Itt kívánjuk megjegyezni, hogy a rendszer a karakterfüzérekkel végzett műveletekhez szükséges átmeneti tárolóterületet is a szövegtárolóból veszi el, így a programban használt karakterfüzérek együttes hosszánál nagyobb tárolóterületet kell kijelölni. A karakteres adatok max. 255 karakter hosszúságúak lehetnek. A karakteres konstansokat (szövegkonstansokat) idézőjel (") határolja, így azokban az idézőjel kivételével bármilyen más karakter szerepelhet. A karakterfüzérek azonosítóit az azonosító után írt $ karakter jelzi. Így szövegkonstansot reprezentál a "ABCdef GH56",

"c123", "ADAT:", "Eredmény" stb. (Szemben az azonosítókkal a BASIC a szövegkonstansokban megkülönbözteti a nagy- és kisbetűket.) Az alábbi azonosítók karakterfüzéreket jelölnek: Y3$, Z$, PA$ stb.

A BASIC gépi reprezentáció az azonos névrészű, de különböző típusjelzésű azonosítókat és változókat különbözőeknek tekinti.

Szükségesnek tartjuk megjegyezni, hogy implicit módon a BASIC gépi reprezentáció logikai típusú adatokat is használ. A logikai adatok „igen" vagy „nem" értékeket vehetnek fel, amelyeket a BASIC egész típusú numerikus adatokkal reprezentál. A belső hozzárendelés szerint a 0 a logikai „nem" értéknek a —1 pedig a logikai „igen" értéknek felel meg. Programon belüli értékadást használva a rendszer a 0 értéket logikai „nem" értékként, minden más értéket pedig logikai „igen" értékként kezel.

A különböző adattípusok definiálása a típusdeklaráló utasítások felhasználásával egyszerűsíthető. Ezeket az utasításokat a programban a definiálni kívánt változó első felhasználása előtt kell elhelyezni. A BASIC a következő típusdeklaráló utasítások használatát teszi lehetővé:

DEFINT
DEFSNG
DEFDBL
DEFSTR

Mindegyik deklaráló utasítást az angol ABC betűiből álló, egykarakteres elemeket tartalmazó, tetszőleges hosszúságú lista, vagy egy K1—K2 határérték-páros követ (ill. ezek tetszőleges kombinációja). A deklaráló utasítások a mögöttük lévő listában felsorolt karakterekkel kezdődő azonosítókat egész, valós, kétszeres pontosságú vagy karakterfüzér típusúnak definiálja. A definíció minden, adott karakterrel kezdődő azonosítóra érvényes, így adott kezdőbetű csak egy deklaráló utasításban szerepelhet. (Valójában megengedett azonos kezdőbetű több deklarációban való szerepeltetése is, de ez áttekinthetetlenné teszi a programot. Ilyen esetben az azonosító típusát a hozzárendelt érték típusa határozza meg, de ez nem mindig egyértelmű, mivel a szöveges változók számjegyeket is tartalmazhatnak, de velük művelet nem végezhető.) A program során az azonosítók típusa részlegesen vagy teljesen, tetszőleges sokszor átdeklarálható.

A DEF... K1—K2 típusú deklarálás azt jelenti, hogy az angol ABC

sorrendjében K1 és K2 karakterek közé eső kezdőbetűjű azonosítók típusa a deklarált típus lesz. A kétféle deklarálási módszer egy listán belül kombinálható. A deklaráló utasítások közül a DEFINT egész típusú, a DEFSNG valós típusú, a DEFDBL kétszeres pontosságú és a DEFSTR karakterfüzér típusú adatokat deklarál. A DEF... típus-deklarálással megadott típusú azonosítók után a típusjelző karakter kiírása nem kötelező (de megengedett), és a típusjelző karakter kiírása felülbírálja a globális DEF... típusú deklarálás hatását.

Tömbök

A tömbazonosítók vagy tömbváltozók egyetlen névvel egy többelemű, többszörösen rendezett adathalmazt jelölnek. Az adathalmaz egy szempont szerinti rendezettségét egy indexhalmaz írja le. A BASIC gépi reprezentáció az indexek számát (a rendezési szempontok számát) nem korlátozza, az egyedüli korlátot a rendelkezésre álló tárolóhely mérete jelenti. A tömb kívánt eleme a tömbazonosító után zárójelben elhelyezett egy vagy több indexszel választható ki (több index esetén az egyes indexeket vessző választja el). Az indexelés a 0 indextől indul és az index értéke csak egész szám lehet (ha nem az, akkor a BASIC a törtrész levágásával állít elő egész értéket). Negatív indexek nem használhatók. Az indexes változókat az első felhasználásuk előtt dimenzionálni kell a DIM deklaráló utasítással. A dimenzionálás a DIM AZ (10, 15, ...) formában történik, ahol AZ a tömbazonosítót jelzi (amely tetszőleges típusjelző karakterrel is kiegészíthető, és így egész, valós, kétszeres pontosságú vagy karakteres tömbökről beszélünk) és a zárójelben lévő lista az egyes indexek lehetséges max. értékeit jelenti (ami eggyel kisebb, mint az adott indexhez tartozó adatok száma, mivel az indexelés a 0-val indul). Az így megadott max. értékek a programban nem léphetők túl.

A programon belül egy tömb csak egyszer deklarálható, és a DIM utasításban az indexhatárok változóval is megadhatók (de végrehajtás előtt a változónak értéket kell kapni). Egy DIM utasításban tetszőleges sok tömb deklarálható, a deklarált tömböket vesszővel kell elválasztani. A DEF... utasítással történő típus-deklarálás a tömbökre is vonatkozik.

Itt kívánjuk megjegyezni, hogy bár az indexek számára és max. értékére semmiféle korlátozás nincs, nem célszerű a szükségesnél nagyobb méretű tömböket deklarálni, mivel ez nagymértékben csökkenti a felhasználói program rendelkezésére álló helyet.

4. Értékadó utasítás

A BASIC értékadó utasítását a LET kulcsszó jelzi, de ez elhagyható. Az általános értékadó utasítás alakja:

LET azonosító = kifejezés
vagy
azonosító = kifejezés.

Az értékadás baloldalán szereplő azonosító tetszőleges típusú, elemi vagy tömb azonosító lehet. Az értékadás során a BASIC kiszámítja a jobboldali kifejezés értékét és (az esetleg szükséges típus-konverziót elvégezve) hozzárendeli a baloldali azonosítóhoz. Mivel a művelet során egy-egy típusú hozzárendelés jön létre, így indexes azonosítóknak csak egy konkrét, meghatározott indexű eleme kaphat értéket. A jobboldali kifejezés kiértékelhető kell hogy legyen, és numerikus baloldali azonosító esetén numerikus, karakteres baloldali azonosító esetén karakteres értéket kell szolgáltatnia.

A numerikus kifejezések

— numerikus konstansokból
— numerikus változókból és
— numerikus értéket szolgáltató függvényekből

állnak. A numerikus konstansok a szokásos számírási szabályok szerint (vezető nullák betűközzel helyettesíthetők, a pozitív előjel kiírása nem kötelező) vagy lebegőpontos alakban írhatók. A lebegőpontos alak egy tetszőleges előjeles mantisszából, az azt követő E betűből és az utána írt 38-nál nem nagyobb abszolútértékű előjeles karakterisztikából áll. A lebegőpontos numerikus konstansnak mindig kell hogy legyen mantisszája (azaz az E vagy D betű előtt legalább egy számjegynek kötelező állni). A numerikus konstans tetszőleges típusú lehet, a BASIC a kifejezés kiértékelésekor az automatikus típus-konverziót végrehajtja. A numerikus

változó tetszőleges típusú (de numerikus), egyedi vagy indexes (adott indexszel) változó (azaz értékkel rendelkező mennyiség) lehet. Mivel a BASIC gépi reprezentáció az értéket nem kapott változókhoz alapfeltételezés szerint nulla kezdőértéket rendel, így a kifejezésben szereplő azonosítók (értéket nem kapott változók) a kifejezés kiértékelése közben nem okoznak hibajelzést, de a nulla értékük miatt a kifejezés értéke általában hibás lesz. A kifejezés tartalmazhatja a műszaki-matematikai gyakorlatban megszokott egyértékű, numerikus értéket adó függvényeket is. Ezek a függvények a BASIC beépített függvényei, és a rájuk való hivatkozás a nevük megadásával és a nevet zárójelben elhelyezve követő argumentumból álló szintaktikai egységgel történik. A BASIC gépi reprezentációban megengedett numerikus függvények:

ABS(X) Kiszámítja X abszolút értékét. X tetszőleges típusú numerikus kifejezés lehet.

FIX(X) Előállítja az X argumentum egészrészét (levágja a tizedespontól jobbra eső részt). Az X tetszőleges típusú numerikus kifejezés lehet.

INT(X) Előállítja az X argumentumnál kisebb egész számot. X tetszőleges típusú numerikus kifejezés lehet.

SGN(Q) Előjel-függvény. Értéke $X < 0$ esetén -1; $X > 0$ esetén $+1$ és $X = 0$ esetén 0. X tetszőleges típusú numerikus kifejezés lehet.

CINT(X) A valós vagy kétszeres pontosságú X argumentumot egész típusúra konvertálja. Az X argumentum értékének —32768 és +32767 közé kell esni, és az előállított egész típusú függvényérték az argumentumnál kisebb egész szám.

CDBL(X) A tetszőleges típusú numerikus argumentumot kétszeres pontosságú értékké konvertálja. A kapott érték a kiinduló értékben nem szereplő, tizedesponttól jobbra eső pozíciókon véletlenszerű számjegyeket tartalmazhat.

CSNG(X) A tetszőleges típusú numerikus argumentumot valós típusúra konvertálja. Az átalakítás a kétszeres pontosságú adatokból csak hat értékes jegyet hagy meg, az egész típusú argumentum értéke és pontossága nem változik.

VAL(X$) A számjegykaraktereket, ill. az exponenst jelölő E vagy D betűt, tizedespontot és előjeleket (a mantissza és exponens előtt) tartalmazó X$ karakterfüzért numerikus értékké alakítja. Ha a karakterfüzér a számokban nem megengedett karaktert is tartalmaz (vigyázat, ilyen az E vagy D helyett írt e is, mivel ezek karakterfüzérben különböző karaktereknek számítanak!), akkor csak az illegális karakterig található számjegyeket dolgozza fel.

ASC(X$) Előállítja az X$ karakterfüzér első karakterének karakterkódját decimális egész szám formájában.

LEN(X$) Megadja az X$ karakterfüzérben lévő karakterek számát (a karakterfüzér hosszát).

RND(X) Egy kvázivéletlen számot állít elő. Ha az X argumentum egynél nagyobb, akkor a véletlen szám a 0 és az X egész része közti intervallumba, ha $X < 1$, akkor a 0...0.999999 intervallumba esik. Az X argumentum csak pozitív szám lehet.
Az RND (X) függvény a bekapcsolás hatására mindig azonos (de egymáshoz képest véletlenszerű) számsorozatot generál, mivel azonos kezdőértékről indul. A kezdőérték véletlenné tehető a RANDOM utasítás segítségével, így minden számsorozat egymástól független, véletlenszerű lesz.

COS(X) A radiánban megadott X argumentum koszinusza. X tetszőleges típusú numerikus kifejezés lehet.

SIN(X) A radiánban megadott X argumentum színusza. X tetszőleges típusú numerikus kifejezés lehet.

TAN(X) A radiánban megadott X argumentum tangense. Az X argumentum tetszőleges típusú numerikus kifejezés lehet. Ha X nagyon megközelíti a $\pi/2$ értéket, akkor a keletkező túl nagy érték túlcsordulást eredményezhet.

ATN(X) Radiánban mérve megadja az X argumentum arcus tangensének főértékét. X tetszőleges típusú numerikus kifejezés lehet.

SQR(X) Kiszámítja az X argumentum négyzetgyökét. X tetszőleges típusú numerikus kifejezés lehet, de teljesülni kell az $X > 0$ feltételnek.

LOG(X) Az X argumentum természetes alapú logaritmusa. X tetszőleges típusú numerikus kifejezés lehet, de teljesülni kell az X > 0 feltételnek.

EXP(X) Az e^x függvény értéke tetszőleges típusú numerikus argumentum esetén.

FRE(X) Numerikus argumentum esetén megadja a még rendelkezésre
FRE(X$) álló programtároló terület hosszát byte-ban. Szöveges argumens esetén a karakteres változók tárolására használt tárolóterületből még rendelkezésre álló helyet adja (szintén byte-ban). A kapott értékek függetlenek az argumentum értékétől és csak annak típusától függenek.

VARPTR Decimális formában visszaadja az argumentumában szereplő nu-
(X) merikus vagy karakteres változó tárolóbeli címét (ill. karakte-
VARPTR res változó esetén a címleíró blokk címét). Indexes változót ar-
(X$) gumentumként használva az adott indexű tömbelem címét adja. Az argumentum nem lehet konstans.

PEEK(X) Decimális formában megadja az egész típusú argumentumnak megfelelő tárolócímen lévő byte értékét. Az argumentum kifejezés lehet, és ha nem egész értékű, akkor a rendszer az argumentum egész részét veszi.

INP(X) Az X című portról beolvas egy egybyte-os adatot.

Az itt felsorolt numerikus függvények egy részével a tipikus alkalmazásuknál is foglalkozni fogunk.

A numerikus kifejezésekben szereplő szintaktikai egységeket (konstans, változó és függvény) műveleti jelek kapcsolják össze. A BASIC gépi reprezentációban használható műveleti jelek:

+ összeadás
— negálás és kivonás
* szorzás
/ osztás
↑ hatványozás.

Az aritmetikai kifejezések kiértékelését a prioritási szabály és a balról-jobbra szabály határozza meg. A kiértékelés sorrendje:
— először mindig a magasabb prioritási műveletek hajtódnak végre (a prioritási sorrend kerek zárójelek alkalmazásával módosítható). A műveletek prioritása (csökkenő sorrendben):

1. hatványozás
2. előjelváltás (negálás)
3. szorzás, osztás
4. összeadás, kivonás

— az azonos prioritású műveleteket a BASIC balról-jobbra haladva értékeli ki.

Egy aritmetikai kifejezésen belül egymás mellett két műveleti jel nem állhat, ez alól csak a negálás (unáris mínusz jel) a kivétel.

Az aritmetikai kifejezések egy speciális esetét alkotják a relációk, amelyek formailag egy relációjel két oldalán elhelyezett két (nem szükségképpen aritmetikai) kifejezésből állnak. A relációk mindig logikai típusú (igen vagy nem) értéket szolgáltatnak, amelyeket a BASIC — 1 (igen), ill. 0 (nem) értékekkel reprezentál. A BASIC nyelvben használható relációk:

= egyenlő
< kisebb
> nagyobb
< = kisebb vagy egyenlő
> = nagyobb vagy egyenlő
< > nem egyenlő (a > < írásmód is megengedett).

Mivel a reláció értékeként kapott logikai értéket a BASIC numerikus értékként reprezentálja, így aritmetikai értékadás jobboldalán egy reláció is állhat, és ilyenkor a baloldali változó a logikai igen és nem értéknek megfelelő —1 és 0 értékek valamelyikét kapja.

Szintén numerikus értéket szolgáltatnak a logikai kifejezések, amelyekre az

AND logikai-ÉS művelet,
OR logikai-VAGY művelet és
NOT logikai negálás

műveletek használhatók. A BASIC gépi reprezentáció a logikai műveleteket csak numerikus változókkal vagy konstansokkal végzi el, és ezek tetszőleges típusúak lehetnek. A rendszer a műveletvégzés előtt az operandusokat kétbyte-os egész típusú értékké alakítja, majd ezekre, mint

bináris számokra végzi el a kívánt műveletet (bitenkénti ÉS, ill. VAGY kapcsolat és egyes-komplemens képzés).

Mivel végsősoron egy kifejezésen belül aritmetikai, ill. logikai műveletek és relációk egyaránt előfordulhatnak, így teljes általánosságban a prioritási sor a következő:

1. aritmetikai műveletek
2. relációk és
3. logikai műveletek.

A logikai műveletek egymás közti prioritása (csökkenő sorrendben): NOT, AND és OR.

A relációk mindegyike azonos prioritású (így itt a balról jobbra szabály határozza meg a végrehajtási sorrendet).

A karakteres kifejezések

— karakteres konstansból (szövegkonstansból)
— karakteres változóból és
— karakteres (string-) függvényből

állnak. A szövegkonstansok tetszőleges, idézőjelek közt elhelyezett, max. 255 karakter hosszúságú szövegből állnak. A szövegben minden, a billentyűzeten szereplő karakter használható, és a BASIC a szövegkonstansokban megkülönbözteti a kis- és nagybetűket, valamint lényeges a betűközök száma (amely a programban közömbös).

A karakteres változók olyan változók, amelyek szöveg tárolására alkalmasak. A karakteres változókat az azonosító után írt $ jel vagy a DEFSTR utasítás felhasználásával definiálhatjuk. Mégegyszer felhívjuk a figyelmet arra, hogy a BASIC a karakteres változók számára alapállapotban csak 50 byte-ot (50 karakternyi helyet) tart szabadon, és ha ennél több helyre van szükség, akkor azt a CLEAR n utasítással vagy paranccsal a felhasználás előtt le kell foglalni.

A karakteres függvények közös jellemzője, hogy szöveges függvényértéket állítanak elő (tehát eredményük csak szöveges változóban helyezhető el).

A BASIC gépi reprezentációban használt karakteres függvények a következők:

CHR$(X) A függvény argumentuma tetszőleges numerikus érték vagy kifejezés lehet. A függvény kiértékeli a kifejezést, ill. veszi a numerikus argumentumot, egész számmá alakítja és függvényértékül előállítja a kapott számértéknek megfelelő kódú karaktert. A kódtábla a mellékletben található. A függvény argumentumának a 0...255 intervallumba kell esni, 255-nél nagyobb, ill. negatív argumentum nem megengedett. A CHR$ függvény az ASC numerikus függvény inverze. A CHR$ függvénnyel előállított kódok egy része vezérlőkaraktert eredményez, ill. nincs nyomtatható képe.

LEFT$(X$, Y) Kiválasztja az X$ karakteres konstans, változó vagy kifejezés első Y karakterét. (A szöveg balszélső, Y db karakterét). Y csak 0 és 255 közti egész szám lehet (ha nem egész, akkor a BASIC az egész részével számol), és ha értéke nagyobb X$ hosszánál, akkor a teljes X$ tartalmat kiírja. Y numerikus változóval vagy kifejezéssel is megadható.

RIGHT$(X$, Y) Kiválasztja az X$ karakteres konstans, változó vagy kifejezés utolsó (jobbszélső) Y karakterét. Y numerikus konstans, változó vagy kifejezés lehet, és értékére a LEFT$ függvénynél elmondottak érvényesek.

MID$(X$, I, J) Kiválasztja az X$ karakteres konstans, változó vagy kifejezés I-edik pozíciót követő J darab karakterét. I és J tetszőleges numerikus konstans, változó vagy kifejezés lehet és értékére a LEFT$ függvénynél elmondottak érvényesek.

STR$(X) Szövegváltozóvá alakítja az X tetszőleges típusú numerikus konstans, változó vagy kifejezés értékét. (A VAL numerikus függvény inverze). Az előállított szöveges érték hosszát a numerikus érték hossza (típusa) határozza meg. A karakteres értékké alakítás esetén a numerikus értéket a rendszer, ha szükséges normálalakra hozza.

STRING$(N, X) Egy N karakterből álló, az X számértéknek megfelelő kódú karakterekkel feltöltött szöveges értéket állít elő.

A szöveges értékű elemekre a BASIC csak az egyesítés műveletét en-

gedi meg, amit a + jel jelöl. Az egyesítés hatására a szöveges értékek tartalmuk megváltoztatása nélkül egymás után másolódnak. A karakterfüzérekre is érvényesek a relációk, de a relációjel mindkét oldalán karakteres adatnak kell állni. A relációk kiértékelése a karakterfüzér balszélétől kezdődően, az egyes karakterkódok, mint numerikus értékek összehasonlításával történik. (A kódtábla a mellékletben található, de itt kívánjuk megjegyezni, hogy a betűk kódjai az angol ABC szerint növekszenek és az ékezetes betűkhöz extra kódok tartoznak.)

Az egyes műveletek prioritása itt is a numerikus kifejezéseknél elmondott, azaz először az egyesítés, majd a relációk kiértékelése történik meg.

5. Vezérlésátadó utasítások

A BASIC legegyszerűbb vezérlésátadó utasítása a feltétel nélküli GOTO utasítás, amelynek formája:
GOTO sorszám
Az utasítás hatására a program a megadott sorszámú utasítássor első utasításával folytatódik. A sorszám csak pozitív, egész numerikus értékkel adható meg és ilyen számú sornak a programban lennie kell.
A feltételes vezérlésátadás az
IF log.kif. THEN (BASIC utasítások) ELSE (BASIC utasítások) formájú utasítással valósítható meg. Az utasítás a THEN után elhelyezett utasításcsoportot hatja végre, ha a logikai kifejezés értéke: igaz (THEN-ág) és az ELSE utáni utasításcsoportot, ha a logikai kifejezés értéke: hamis (ELSE-ág). A logikai kifejezés leggyakrabban egy reláció, de tetszőleges bonyolultságú logikai kifejezés lehet, és helyette numerikus érték (konstans, változó, aritmetikai kifejezés) is alkalmazható, amelyet a BASIC egész számmá alakít és logikai változóként kezel (a nulla a logikai nemnek, minden nullától különböző érték pedig logikai igennek minősül). A THEN- és ELSE-ágak tetszőlegesen sok BASIC utasítást tartalmazhatnak, de a teljes IF utasítás legfeljebb egy utasítássor hosszúságú (címkével együtt 210 karakter) lehet. A THEN- és ELSE- ágakban tetszőleges BASIC utasítások (vagy programba beépíthető parancsok) lehetnek, így azokba további IF utasítások is beépíthetők. Az IF utasítások egymásba ágyazásának mélységét a BASIC nem korlátozza.

Az IF utasításból az ELSE-ág elhagyható, ilyenkor a logikai kifejezés hamis értéke esetén a program következő sorában lévő vagy az IF utasítást követő utasítás hajtódik végre. Gyakori eset, hogy az ELSE- és THEN-ágak feltétel nélküli vezérlésátadó utasítást (GOTO) tartalmaznak. Ilyen esetekben a GOTO szócska elhagyható, és megengedett az
IF log. kifejezés THEN sorsz. 1 ELSE sorsz. 2
forma alkalmazása. A THEN sorsz. 1 alak helyett az
IF log. kifejezés GOTO sorsz. 1 ELSE sorsz. 2
alak is írható (ilyenkor az ELSE kiírása természetesen csak akkor kötelező, ha az ELSE-ág tartalmaz BASIC utasításokat).

A vezérlésátadások egy speciális változata a szubrutinhívás, amely a
GOSUB sorszám
alakú utasítással történik. Az utasítás hatására a program végrehajtása a megadott sorszámú sor első utasításával folytatódik (mint a GOTO esetén), de az első előforduló RETURN utasítás hatására (amely a szubrutin logikai vége) a vezérlés visszakerül a GOSUB utasítást követő utasításra. A GOSUB—RETURN utasításpáros lehetővé teszi, hogy a programban többször előforduló, azonos funkciókat megvalósító programrészeket egyetlen szubrutin formájában realizáljuk és tetszőlegesen sokszor, a program bármely helyéről aktivizáljuk. A szubrutin a hívó programmal közös adatokat használ, ill. a szubrutinban bevezetett új változók a hívó programban is értelmezve lesznek, így a hívó és hívott programrészek közti paraméter átadás-átvétel leegyszerűsödik. A szubrutin további szubrutinokat hívhat, a szubrutinok egymásba ágyazásának mélysége nincs korlátozva. A szubrutin logikai végét jelző RETURN utasítás bárhol előfordulhat és egy szubrutinban tetszőleges sok RETURN utasítás lehet, de ezek közül csak egy hajtódhat végre. Ha a programban GOSUB utasítás végrehajtása előtt egy RETURN utasítás kerülne sorra, akkor a rendszer hibajelzést ad. A szubrutinból nem tilos GOTO vagy IF utasítással visszatérni a hívó programba, de ilyenkor a szubrutinból való visszatérés feltételei tárolódnak, és a program a bárhol előforduló RETURN utasítás hatására mindig a legutolsónak végrehajtott GOSUB utáni utasítást fogja végrehajtani. Ezért a szubrutinok ilyen abnormális befejezése számos programfutási hiba oka lehet, és ennek elkerülése érdekében mindig használjuk a normális (RETURN utasításon keresztüli) visszatérést. A BASIC program szempontjából a szubrutin egyetlen BASIC utasításnak minősül, és

az IF utasítás THEN vagy ELSE-ágába beépítve minden szempontból a GOTO utasítással azonos módon használható. A BASIC gépi reprezentáció a rekurzív szubrutinhívást is megengedi (azaz egy szubrutin önmagát is meghívhatja szubrutinként), de a RETURN utasítások helyének és számának bonyolult meghatározása miatt ezt a lehetőséget csak gyakorlott programozók használják!

A feltételes és feltétel nélküli vezérlésátadások közti átmenetet alkotja az ún. kiszámított (vagy ON GOTO, ill. ON GOSUB) vezérlésátadás, amelynek formája:

ON aritm.kif. GOTO S1, S2, —— Sn

vagy

ON aritm. kif. GOSUB S1, S2, —— Sn

Az utasítások kiértékelik az aritmetikai kifejezést (amely tetszőleges típusú eredményt adhat), az eredményt egész számmá alakítják és a GOTO vagy GOSUB utasításnál leírt módon a vezérlést a sorszámlista i-edik helyén álló sorszámra adják, ha a kifejezés egészértéke i. Ebből következik, hogy n db sorszámból álló lista esetén a kifejezés egészértékének 1 és n közé kell esnie. Ha a kifejezés egészértéke 0 vagy n értékénél nagyobb, akkor az ON GOTO vagy ON GOSUB utasítást követő utasítás hajtódik végre, (ugyancsak ez történik az ON GOSUB utasítás esetén, ha a szubrutinból a RETURN utasítással térünk vissza). A kifejezés értéke negatív szám nem lehet.

Hibakezelés:

A programban előforduló végrehajtási hibák (tehát nem szintaktikai hibák) detektálására és programozott javítására az ON GOTO utasítás egy speciális változata szolgál, amely az

ON ERROR GOTO sorszám

alakban írható. Az utasításban szereplő sorszám a felhasználói program hibakezelő szubrutinjának kezdő sorszáma. amelyet a rendszer az ON ERROR GOTO utasítás első végrehajtásakor eltárol, és programhiba esetén erre a sorra adja a vezérlést. Az ON ERROR GOTO utasítás első végrehajtásáig a hibakezelés hatástalan (azaz a lehetséges hibaforrás előtt kell végrehajtani). A sorszám nulla vagy tetszőleges, a programban előforduló sorszám lehet. A 0 sorszám lekapcsolja a hibakezelést (azaz egy ON ERROR GOTO 0 után a programhibák leállítják a program futását és a képernyőre egy hibaüzenetet generálnak, mivel a hibakezelő programrész nem működik). A hibakezelő szubrutin speciális utasításokkal zárul. A hibakezelés logikai végét a RESUME utasítás jelzi (hasonlóan a szokásos szubrutinok RETURN utasításához). A RESUME utasítás három formában használható:

RESUME — Befejezi a hibakezelő szubrutint és visszatér a hibás utasításra, majd azt újból végrehajtja (természetesen a hibaokot a hibakezelő szubrutinban el kell hárítani, különben végtelen ciklus alakul ki). Az utasítás a javítható hibák kezelésénél használható.

RESUME NEXT — Befejezi a hibakezelő szubrutint és visszatér a hibás utasítást követő utasításra. A RESUME NEXT típusú befejezés olyan esetekben alkalmazható, amikor a hibás utasítás egyszerűen elhagyható.

RESUME sorsz. — A hibakezelő szubrutin befejezése után a vezérlést a megadott sorszámú sorra adja. (Pl. fatális hiba esetén hibajelzés kiírása, vagy adathiba esetén új adatok bekérése.)

A hibakezelő szubrutinban a hiba oka és helye az ERR és ERL rendszerparaméterek értékének lekérdezésével határozható meg. Az ERR és ERL változók a felhasználó által definiált változókkal azonos módon használhatók, értékük pozitív egész szám. A ERR változó értéke az utoljára kialakult programhiba kódját (a kódok a mellékletben találhatók), az ERL változó a hibás utasítást tartalmazó programsor sorszámát tartalmazza. A BASIC rendszer az ERR hibakódot a 2, 4, 6, ... 44 értékek valamelyikére állítja a hiba jellegétől függően.

Az ERR értéke programból az ERROR nn utasítással állítható be (és természetesen ilyenkor az ERL az ERROR utasítást tartalmazó sor sorszáma lesz) úgy, hogy a generált hibakód a 2 * nn — 2 értékű lesz. Ez lehetőséget nyújt a programozónak, hogy a programjából hibajelzéseket generáljon, ill. a generált hibát az ON ERROR GOTO utasítással kezelje.

6. Ciklusszervezés

A hagyományos, értékadáson és IF utasításon alapuló ciklusok egyszerűsítésére a BASIC a FOR — NEXT utasításpárból álló ciklusszervező utasítást használja. A FOR utasítás a ciklusmag előtt helyezkedik el, amíg a NEXT utasítás lezárja a ciklusmagot. A FOR — NEXT utasításpár formája:

FOR cikl.vált.=kezdőért. TO végért. STEP lépésköz

.

. (ciklusmag)

.

NEXT cikl. vált.

Az utasítás hatására — az egész vagy valós típusú — ciklusváltozó felveszi a megadott kezdőértéket, végrehajtódik a ciklusmag, majd ezt követően az előírt lépésközzel módosul a ciklusváltozó értéke és ha a módosított érték még nem haladta meg a végértéket, akkor visszakerül a vezérlés a ciklusmag első utasítására. Ha a ciklusváltozó értéke meghaladta (növekvő irányú ciklusnál nagyobb, csökkenő irányúnál kisebb) a végértéket, akkor a program végrehajtása a NEXT utasítást követő utasítással folytatódik (normális ciklusbefejezés). A fenti végrehajtási szabályból következik, hogy a ciklusmag egyszer mindenképpen végrehajtódik, még akkor is, ha a kezdőérték eleve meghaladta a végértéket. A ciklus paraméterei (kezdő- és végérték, valamint lépésköz) tetszőleges típusú numerikus konstansok, változók vagy kifejezések lehetnek, de megválasztásuknál ügyelni kell arra, hogy a végértéknél kisebb kezdőérték esetén a negatív, a végértéknél nagyobb kezdőérték esetén pedig a pozitív lépésköz a ciklusmagnak csak az egyszeri végrehajtását eredményezi, és nulla lépésköz esetén végtelen ciklus jön létre. Törtrészt is tartalmazó ciklusparaméterek esetén — a számolási hibák következtében — a ciklusmag végrehajtásának száma eltérhet a várt értéktől.

A BASIC megengedi a ciklusok tetszőleges mélységben egymásba ágyazását, de átfedő (átlapolódó) ciklusok kialakítása tilos. A FOR — NEXT utasításpár a ciklus paramétereit (kezdő és végérték, lépésköz) a ciklus kezdetekor a belső rendszerváltozókba letárolja, így ezek értékének ciklusmagban történő bármilyen módosítása a ciklus működésére nézve hatástalan. A ciklusváltozó értéke a ciklusmagban tetszőleges célra használható, és a ciklusváltozó értékének cikluson belüli módosítása (értékadás) megengedett, de nehezen követhető hatásokkal jár. Hasonló módon megengedett a cikluson kívülről a a ciklusmag bármelyik utasítására adni a vezérlést, vagy a ciklusmagból a ciklust definiáló FOR utasításra adni a vezérlést, de a mellékhatások miatt ezeket is célszerű kerülni. A lezáratlan (NEXT utasítást nem tartalmazó) ciklus FOR utasítása egyszerű értékadó utasításként működik (a ciklusváltozó felveszi a kezdőértéket), viszont a FOR utasítás végrehajtása nélkül NEXT utasításra kerülő vezérlés hibajelzést generál.

A ciklusmagból feltételes vagy feltétel nélküli vezérlésátadó utasítással szabad kilépni, és a kilépés után a ciklusmagba visszatérve a ciklus folytatható (természetesen a folytatást befolyásolhatja a ciklusváltozó értékének módosítása). A cikluszáró NEXT utasításból a ciklusváltozó elhagyható, a BASIC az először előforduló NEXT utasítást automatikusan az utoljára megnyitott ciklus végének tekinti.

7. Adatátviteli utasítások

A BASIC belső és külső adatátviteli utasításokat különböztet meg. A belső adatbevitel a programban DATA utasítással (vagy utasításokkal) deklarált adatlista változókhoz történő hozzárendeléséből áll. A hozzárendelést a READ utasítás végzi el. Az ilyen típusú összerendelés általános alakja:

DATA a1, a2, a3, a4, . . . an

.

.

.

.

READ V1, V2, V3, V4, . . . Vn

A BASIC minden READ utasítás hatására sorban egymás után előveszi a DATA adatmezejében szereplő értékeket és hozzárendeli a READ után álló egyszerű vagy indexes változókhoz. A kiolvasott adatok száma alaphelyzetben nem haladhatja meg az adatmező (amely több DATA utasításból állhat, és úgy fogható fel, mintha egyetlen DATA utasítás lenne, ami az előfordulásuk sorrendjében tartalmazza az adatokat) elemeinek szá-

mát. Ha minden adatot kiolvastunk a DATA utasításokból, akkor a következő READ utasítás végrehajtásakor hibajelzést kapunk. A DATA utasítások a programban bárhol elhelyezkedhetnek és a teljes adatmezőben lévő adatok sorrendjét a DATA utasítások sorszám szerinti sorrendje, nem pedig a program végrehajtási sorrendje határozza meg. (A DATA és READ utasítások elemzésénél célszerű egy DATA és egy READ utasításban gondolkodni, amelyek az összes adatot, ill. változót tartalmazzák.) A kiolvasás sorrendjét a READ utasítások végrehajtási sorrendje dönti el. A DATA utasításban szereplő adatok típusa és a READ utasításban szereplő változók típusa közt bizonyos kompatibilitási feltételeknek kell fennállni. Alapszabály, hogy numerikus változóhoz csak numerikus adat rendelhető hozzá (a különböző numerikus adattípusok összerendelésénél a BASIC elvégzi a szükséges típuskonverziót), viszont karakteres változóhoz numerikus adat is hozzárendelhető (de természetesen a kapott karakteres változóval numerikus műveletek nem végezhetők). A DATA utasításokkal definiált adatmező sorrendi kiolvasása sokszor hátrányos. Ez a hátrány a RESTORE utasítás alkalmazásával küszöbölhető ki. Az utasítás a

RESTORE vagy

RESTORE sorszám

alakban használható. Az utasítás sorszám nélküli alakja az adatmező mutatóját (amely mindig a következő kiolvasásra kerülő adatra mutat) visszaállítja a legkisebb sorszámú sorban lévő DATA utasítás első adatára (azaz az adatmező újra kiolvasható, vagyis így az adatelemek számánál több adat is kiolvasható az adatmezőből).

A RESTORE sorszám forma működése hasonló, de ez a mutatót a megadott számú sorban lévő DATA utasítás első adatára állítja, és így az adatmező egyes részei újraolvashatók.

A külső adatátvitel a számítógép és a hozzá kapcsolódó perifériák közti adatátvitelt jelenti, és így megkülönböztethetünk adatbevitelt és adatkivitelt. Ebben és a következő fejezetben csak az egyszerű perifériákkal történő kapcsolatot megvalósító utasításokat tárgyaljuk, a háttértároló (kazetta) kezeléséről a file-kezeléssel foglalkozó fejezetben foglalkozunk.

Egyszerű adatbevitel:

Adatbevitelre általában a billentyűzetet használjuk. Adatok billentyűzetről való bekérésére az

INPUT A1, A2, A3, ...

utasítás használható. Az INPUT szócska után tetszőleges sok és tetszőleges típusú azonosító vagy változó helyezkedik el, és az elsőnek beírt adat az első változóhoz, a másodiknak beírt a másodikhoz stb. rendelődik hozzá. A beírt adatok típusának illeszkedni kell a felsorolt azonosítók vagy változók típusához. Numerikus azonosítóhoz vagy változóhoz tetszőleges típusú numerikus érték hozzárendelhető (a BASIC elvégzi az automatikus típus-konverziót), viszont a karakteres azonosítókhoz vagy változókhoz egyaránt numerikus vagy karakteres érték rendelhető (természetesen az így bevitt numerikus érték csak a szövegkezelő műveletekkel együtt szerepelhet a későbbiekben). Ha az azonosítólista megfelelő eleme és a beírt adat közt inkompatibilitás van, akkor a rendszer a REDO hibaüzenettel jelentkezik és újra bekéri az adatot. A REDO hibaüzenet a szokványos (ON ERROR GOTO) hibakezelési eljárással nem védhető ki. Az adatbekéréskor az INPUT utasítás hatására a képernyőn, az aktuális cursorpozícióban egy ? jelenik meg és a számítógép várja az adatot. A beírt adatot a rendszer visszaírja a képernyőre. Az adat a RETURN vagy vesszők karakterekkel zárható. A RETURN karakteres lezárás esetén a következő sor elején (és minden további RETURN utáni sorban) a rendszer ?? karaktereket ír ki és várja a következő adatot. A vesszővel történő lezárás után folyamatosan írható a következő adat (nem jelenik meg új ?). A vesszővel elválasztott adatok esetén előfordulhat, hogy a kívánttnál több adatot írunk. Ilyenkor a rendszer a felesleges adatot elhagyja. Ha az adatlistán egymás mellett két vessző szerepel, akkor az 0 közbenső értékű adatot jelöl. Az adatbevitel RETURN leütésével történő megszakítása (amikor a ? után érvényes adat helyett RETURN karaktert ütünk) lezárja az adatbevitelt, és a még értéket nem kapott azonosítók vagy változók megtartják korábbi értéküket (azonosítóknál ez nulla). Az adatbeviteli hibák elkerülésére és a bevitt adat egyértelmű azonosítására a bekért adatok elé szöveg íratható ki az

INPUT "SZÖVEG"; A1, A2, ...

forma alkalmazásával. A szöveg csak az INPUT utáni lista első eleme lehet és pontosvesszővel kell zárni.

Az adatbevitel egy másik, speciális módját az INKEY$ utasítás valósítja meg. Ez az utasítás mintát vesz a billentyűzetről, és a mintavétel értékét adja eredményül. A mintavételezés akkor is megtörténik, ha nem ütöttünk le billentyűt, ilyenkor az INKEY$ a " " értéket adja (vigyázat, ez az üres karakter és nem pedig a " " formájú betűköz). Az értéktelen minták alkalmas programszerkezettel hagyhatók el, pl. az alábbi módon:

XX A$ = INKEY$: IF A$ = " " THEN XX ELSE YY

Az utasítássor addig folytatja a mintavételezést, amíg valamilyen karaktert le nem ütöttünk, és akkor az YY számú sorra adja a vezérlést (fenti utasítássor egyben példa más BASIC gépi reprezentációkban megengedett GET$ utasítás PRIMO BASIC-ben történő megvalósítására). A fenti utasítássort gyakran használják a futó program átmeneti leállítására, amikor is a feldolgozás tetszőleges karakter leütésével folytatható.

Egyszerű adatkivitel:

Az egyszerű adatkivitel a PRINT utasítással valósítható meg, amely szövegek, ill. numerikus értékek képernyőre való kiírását végzi. A PRINT utasításszó a ? karakter leírásával helyettesíthető (de a BASIC a ?-et PRINT szóvá alakítja és a program listáján már ez jelenik meg). Az utasítás formája:

PRINT lista

vagy

? lista

Az utasítás után lévő lista tetszőlegesen sok szöveges vagy numerikus elemet tartalmazhat. Az elemek konstansok, változók vagy kifejezések lehetnek. Kifejezések esetén a BASIC kiértékeli a kifejezést és az eredményt írja ki. A lista egyes elemei vesszővel vagy pontosvesszővel választhatók el, és az elválasztó karakter egyben meghatározza a nyomtatási formátumot is. A listaelemek vesszővel történő elválasztása ún. tabulált kiíratási formát eredményez, amíg a pontosvesszővel történő elválasztás tömör formátumú kiíratást hoz létre. A kiírt szöveges adatok teljes egészében megegyeznek a szöveges változó vagy konstans tartalmával, viszont a numerikus adatok kiíratásánál automatikus formátumbeállítás történik. Ez a következő módon megy végbe.

Egész típusú érték a szokásos formában íródik ki.

Valós típusú érték kiírása nagyságrendtől függő, a 10^7 értéknél kisebb számok kiírása a szokásos fixpontos formában, hat decimális jegy pontossággal történik. A 10^7 vagy nagyobb nagyságrendű számokat a gép mindig lebegőpontos formában, 1 és 10 közé eső normált mantisszával és kétjegyű kitevővel, összesen hat decimális jegy pontossággal írja ki.

Kétszeres pontosságú értékeknél hasonlóan működik, mint a valósnál, de itt a fixpontos és lebegőpontos forma közti átváltás határa 10^{16}, és a kiírás 16 decimális jegy pontosságú.

A tabulált kiírási forma (ami az elválasztókarakterként vesszőt használva alakul ki) a logikai sort 16 karakteres mezőkre osztja, és minden adatot a következő mező elejére ír. Mivel a képernyőn egy sor csak 42 karaktert tartalmazhat, így az első két mező minden további nélkül kiíródik, a harmadik csak akkor, ha az ide kerülő adat a sorban elfér (kiíráskor az adatot a sorvégén nem töri meg a következő sorra való átmenettel). Egy logikai sor három mezőből áll, és ha a kiíratáskor az egyik mező tartalma átnyúlik a szomszédos mezőbe, akkor annak a mezőnek a fennmaradó része üresen marad és a következő adat csak a rákövetkező mezőbe íródik. A PRINT utáni outputlistában elhelyezett extra vesszők karakterek a következő mező kihagyását jelzik.

A tömör kiírási formában az adatok közvetlenül egymás mellé íródnak (a pozitív számoknál megjelenő elválasztó betűköz a pozitív előjel helye). A numerikus értékek kiírását itt sem zavarja a sorváltás, ha a szám az adott sorban nem fér el akkor automatikusan a következő sorba íródik ki. A szöveges adatok kiírása áthúzódhat a következő sorra.

Az output listában a vessző és pontosvessző vegyesen is előfordulhat, ilyenkor minden vessző hatására a kiíratás a következő mező elején indul.

A PRINT utasítás mindig egy logikai sornyi információt ír ki, és a képernyőn a sorváltást a kiírt adatok hossza határozza meg. Ez szinte lehetetlenné teszi az áttekinthető formátumú kiírást a képernyőre. A probléma a kiírandó adatok közé elhelyezett vezérlőkarakterekkel hárítható el csakúgy, mint a soron belüli pozicionálás, ill. a display üzemmód beállítása. A vezérlőkarakterek a CHR$ függvény értékeként írhatók ki. A megengedett vezérlőkódok és funkciójuk:

1. Normális üzemmódú kiírás,
2. Nyújtott (elongált) karakteres kiírás,

3. Negatív kép beállítása (az egész képernyőre).
4. Negatív alap (az aktuális karakterpozíciótól kezdve, az egyes karakterpozíciókon).
5. Aláhúzott karakterek kiírása.
6. Az egyes karakterek egymásra írásának tiltása (előtörlés).
7. Bell (kb. 0.25 másodperces 800 Hz-es hangimpulzus generálása).
8. Egy karakterpozícióval visszalép (nincs tekintettel az előző karakterpozíció aktuális állapotára, tehát egy nyújtott karakter után normális karaktert beállítva, majd visszalépve a nyújtott karakternek csak a második felét írja felül).
9. Horizontális tabulátor (a sor következő nyolccal osztható karakterpozíciójára áll).
12. CLS (képernyőtörlés).
13. Kocsivissza-soremelés (RETURN).
14. Kocsivissza (visszaáll a sor elejére, és felülírja a sort).
16. Alsóindex pozíció (a kiíratás a 192 soros képmezőben négy sorral, azaz egy normális méretű karakter magasságának felével lefelé eltolva folytatódik).
17. Felsőindex pozíció (négy sorral feljebb tolva folytatódik a kiírás). Az alsó- és felsőindex pozíció mindig az előző karakter helyzetéhez viszonyítva értendő.
18. A nyújtott karakteres (elongált) kiírás kikapcsolása.
19. Negatív kép visszaállítása (a 3. kód inverze).
20. Inverz alap visszaállítása (a 4. kód inverze).
21. Aláhúzás kikapcsolása.
22. Az egyes karakterek egymásra írásának engedélyezése (a 6. kód inverze).

A kiíratás soron belüli kezdőpontja a TAB(X) függvény outputlistába történő beépítésével választható meg. A függvény argumentuma tetszőleges numerikus kifejezés lehet, amelynek értéke pozitív kell, hogy legyen és egészrészét véve nem lehet 41-nél nagyobb. A TAB hatására a nyomtatás a sor X-edik pozícióján folytatódik (ha még nem haladta túl a nyomtatás az adott pozíciót). Ha a sorban már a kiírt információ túlhaladta a TAB függvénnyel megadott pozíciót, akkor a kiírásnál a függvényt nem veszi figyelembe a rendszer.

Összetett, áttekinthető formátumú adatkiíratást tesz lehetővé a PRINT USING utasítás használata, amely a következő formában használható:

PRINT USING Y$; output lista

Az utasításban szereplő Y$ karakteres konstans vagy változó a kiíratási formátum vezérlését végzi, és az output lista a PRINT utasításnál használt listával megegyező tulajdonságú (de az elválasztó karakter akár vesszső, akár pontosvessző lehet). A kiíratási formátum az Y$ karakteres változóban vagy konstansban elhelyezett speciális mezőleíró karakterekkel szabályozható. A megengedett formátumvezérlő karakterek és funkciójuk:

\# Számjegypozíciót jelöl. Ahol a # karakter előfordul, ott számjegykarakter íródik ki.

. Számokon belül a tizedespont helyét jelzi. Egy fixpontos kiíratási formátum pl. a ###.## karaktersorozattal adható meg, és ez egy háromjegyű egészrészből és két tizedesjegyből álló számot ír le.

↑↑↑↑ Lebegőpontos formátumban az exponens helyének kijelölése. A mantissza ettől függetlenül, a # és . karakterekkel írható le.

, Fixpontos számleírásban alkalmazva a tizedespont előtt az egészrészt vesszővel elválasztott hármas csoportokra tagolja (a tizedesponttól balra indulva).

** Az egészrész értéktelen nullái helyett nem betűközt, hanem * karaktert ír.

$$ A szám legnagyobb helyiértékű számjegye elé egy $ karaktert ír (negatív szám esetén az előjel után).

**$$ A szám előtti üres helyekre * karaktert ír és a legnagyobb helyiértékű jegy elé egy $ karaktert.

\+ A formátummező elején vagy végén elhelyezve a szám előtt vagy után kiírja a + és — előjelet.

— A formátummező elején vagy végén elhelyezve a szám előtt vagy után kiírja a — előjelet és a + előjel helyére betűközt ír.

%% Az output listában lévő karaktermező helyének definiálása. A szöveg hossza a % jelek közti betűközök számánál kettővel nagyobb lehet. Ha a szöveg ennél rövidebb, akkor betűközökkel pótolja ki, ha hosszabb, akkor a végét levágja.

! Az output listában lévő karakteres információ első karakterének kiírása.

A PRINT USING utasítás Y$ formátumleíró információja tetszőlegesen sok numerikus és karakteres információt specifikálhat. Ha a specifikált elemek száma nagyobb, mint az output lista elemeinek száma, akkor a rendszer a felesleges formátumleírásokat nem veszi figyelembe. Ha a formátumleírások száma kisebb, mint a output listában lévő elemek száma, akkor a rendszer a formátumleírást elölről kezdi feldolgozni. A formátumleírásban specifikált mezők és az adatok kompatibilitásáról gondoskodni kell (amikor numerikus formátum van, akkor numerikus adatnak kell következni és amikor szöveges specifikáció, akkor karakteres információnak), ellenkező esetben a rendszer hibajelzést generál. A formátumleírásban szereplő minden, a fentiektől eltérő karaktert az előfordulási helyén szövegkonstansként ír ki a rendszer. Az adott formátum szerint kiírt szám egész- és törtrésze együttesen max. 24 karaktert tartalmazhat, és a specifikációban szereplő ** vagy $$ karakterek úgy számítanak, mintha # karakterek lennének. Ha a kiíratás a kívánt formában nem fér el, akkor a rendszer eltér az adott formátumtól és a szám elé egy % jelet ír.

8. Grafikus üzemmód

A PRIMO számítógép BASIC gépi reprezentációjának egyik legnagyobb előnye a jó felbontású, gyors grafikus üzemmód. A grafika a 192 sorra, soronként 256 képpontra osztott teljes képmezőt használja, és a megjelenített ábra tetszőleges szöveges információval kombinálható. A grafikus funkciók a képernyő (x, y) koordinátájú pontjának címzésével valósíthatók meg, ahol x értéke 0 és 255, y értéke 0 és 191 közé kell hogy essen. A grafikát kezelő utasítások:

SET (X, Y) A képernyő (x, y) koordinátájú pontját bekapcsolja (azaz normális kép esetén világosba, negált kép esetén sötétbe kapcsolja).

RESET(X, Y) Kikapcsolja a képernyő (x, y) koordinátájú pontját (a kép normális vagy negált formájától függően).

POINT(X, Y) Megvizsgálja a képernyő (x, y) koordinátájú pontját, és ha az bekapcsolt, akkor logikai igen (—1) értéket, ha nem akkor logikai nem (0) értéket ad. A be- és kikapcsolt állapot értelmezése szintén a kép normális vagy negált jellegétől függ.

Itt jegyezzük meg, hogy az ábrákkal együtt kiírt szöveg különböző speciális karaktereket is tartalmazhat, amelyek a CHR$ függvény felhasználásával írathatók ki, és kódjukat a mellékletben található kódtábla tartalmazza.

9. Programvezérlő utasítások

A BASIC programok futás közbeni leállítására a STOP és az END utasítások használhatók. Funkcióját tekintve mindkét utasítás azonos, a velük megállított program a CONT paranccsal folytatható. Az END okozta megállás a képernyőn az *Ok* üzenettel, a STOP okozta megállás a képernyőn a *Break in nn* (ahol nn a STOP utasítást tartalmazó sor sorszáma) és a következő sorba írt *Ok* üzenettel jelentkezik. A megállás után a számítógép parancsmódban van.

10. File-kezelés

A PRIMO BASIC a program- és adatfile-okat jelenleg kazettás magnetofonon tárolja. A BASIC file-kezelő funkciói lehetővé teszik az egységes, eszközfüggetlen (logikai) file-kezelést. A kezelés szempontjából meg kell különböztetni program- és adatkezelést. A programfile-ok a jelenlegi változatban programból kiadott utasításokkal nem kezelhetők, csak parancsokkal. A programkezelést három parancs végzi:

LOAD
SAVE
TEST

A parancsok használatát és funkcióját a parancsok közt, az 1. pontban már ismertettük.

Az adatfile-ok kezelése a szakaszos hozzáférési igény és a pufferelt adatátvitel miatt bonyolultabb, és input, ill. output file-ok esetén kismértékben különbözik. A file-kezelés logikailag a file megnyitására, a file használatára és a lezárásra osztható.

Input file megnyitása:

Az input file az

OPEN "file-név"

utasítással nyitható meg. A file-név egy tetszőleges, 16 karakteres név (ami természetesen megegyezik a file generálásakor használt névvel) lehet. Az utasítás megkeresi az adott eszközön a kívánt nevű file-t, ellenőrzi a felíráskor készített blokkfejet, majd ráál az első adatra. A PRIMO BASIC a file-kezelés (LOAD, SAVE, TEST, OPEN, CREATE, INPUT#, PRINT#, CLOSE utasítások) során elvégzi a kazettás magnetofon távvezérlését. A megfelelő kapcsolójel a TAPE jelű tuchelen érhető el (lásd 31. oldal). Amennyiben az Ön magnetofonja alkalmas a távvezérlésre, úgy mentesülhet az egyébként szükséges manuális indítás és megállítás végrehajtásától. (Kapcsoló-jel: =0 V magnó állj; = 5 V magnó indulj. Felhívjuk a figyelmet arra, hogy a PRIMO meghibásodásának elkerülése érdekében — szakemberrel vizsgáltassa meg magnetofonját, hogy a fenti kapcsoló-jel értékek mellett a távvezérlés megvalósítható-e.)

Output file megnyitása:

Az output file a

CREATE "file-név"

utasítással történik. Az utasításban szereplő paraméterek megegyeznek az OPEN utasításban szereplőkkel. Az utasítás a kijelölt eszközön az adott névvel egy blokkfejet generál, majd az első adat helyére áll. A kazettás magnetofon kezelésénél itt is hasonló problémák merülnek fel, mint az OPEN utasításnál, így az ott elmondottak szintén érvényesek.

Írás és olvasás:

A megnyitott file-ba való írás, ill. az abból való olvasás a már ismertetett INPUT és PRINT utasítások speciális változatával lehetséges. Az utasítások formája a következő:

INPUT # input-lista

PRINT # output-lista

Az utasításokban a # jel után szereplő n a megnyitáskor a file-hoz rendelt file-szám. Egy INPUT vagy PRINT utasítással csak egy file kezelhető. Az utasításokban lévő input- és output-lista megegyezik a korábban ismertetett listákkal. Az INPUT utasítás hatására a file-ból annyi adat olvasódik be, ahány elemből áll az input lista, és a beolvasott adatok sorban hozzárendelődnek a listában szereplő azonosítókhoz vagy változókhoz. A numerikus azonosítókhoz vagy változókhoz csak numerikus adat, a karakteresekhez csak karakteres adat rendelhető, ellenkező esetben hibajelzést kapunk.

Az adatkezelés pufferelve történik, a rendszer adatbeolvasáskor egy adatblokkot olvas be, majd az egyes INPUT utasítások azonosítóihoz, ill. változóihoz a pufferből rendeli az értékeket, amíg csak a puffer ki nem ürül. Ekkor egy új blokk olvasódik a pufferba. A pufferelt kezelés az output file-ok esetén hasonlóan zajlik le, csak ott a kiírt adatokat tárolja a pufferban a rendszer addig, amíg az be nem telik, és csak akkor írja az eszközre. A pufferelés miatt lehetséges, hogy az egyes INPUT vagy PRINT utasítások végrehajtása során tényleges (fizikai) eszköz-hozzáférés nem alakul ki. Itt jegyezzük meg, hogy a kazettás magnetofonon a rendszer 256 byte-os blokkokat használ. Ha a magnetofont bekapcsoltuk, akkor csak a 256 byte-os puffer betelte esetén íródik ki az információ, és így a blokkok közt nagyon nagy távolság alakulhat ki. Mivel a blokkok elején egy szinkronjel helyezkedik el, ez a nagy távolság az adatfile későbbi beolvasásakor nem okoz problémát, de nagyon rossz szalagkihasználást eredményez. Ezért célszerű a programban egyszerre megvalósítani az adatkivitelt (és így természetesen a beolvasást is), vagy a kiírás előtt figyelmeztető jelzést generálni és a programot a magnetofon elindítása után folytatni.

File-ok lezárása:

Az input és output file-ok azonos módon

CLOSE

utasítással zárhatók le. Az utasítás input file esetén lényegében semmilyen funkciót nem tölt be, output file esetén viszont kiírja a puffer tartalmát (ami a lezárás elmulasztása esetén elvész).

Ezért az input file-ok lezárása elhagyható, az output file-ok lezárásának elmulasztása a későbbeikben hibát okoz. Input file-ok esetén a lezáráskor a pufferben lévő, még nem feldolgozott adatok törlődnek (természetesen a file-ban minden adat megmarad).

A grafikus funkciók alkalmazását és szabványos szövegek megjelenítésének egyszerűsítését segíti a

SAVE SCREEN "file-név"

utasítás, amelynek hatására a teljes képernyő tartalma kiíródik a megadott (max. 16 karakteres) nevű file-ba. Az így létrehozott file a LOAD utasítással tölthető be.

Itt kívánjuk megjegyezni, hogy a LOAD utasítás gépi szintű programok betöltésére is alkalmas.

11. A programozást segítő lehetőségek

A programozást segítő lehetőségek a program írását és módosítását egyszerűsítő, ill. a kész program belövését segítő tevékenységekből állnak.

A program írását segítő lehetőségek:

A program írását a billentyűzet funkciói, az *AUTO* parancs és az *EDIT* parancs használata egyszerűsíti. A program írásakor elkövetett hibák azonnal javíthatók a ← gomb megnyomásával, amely minden megnyomásra eggyel visszalép a sorban és törli az ott lévő karaktert. A hiba törlése után a helyes szöveg írható. Ha ennek ellenére maradt hiba a programban, vagy a programot módosítani akarjuk, akkor a korábban már vázlatosan ismertetett EDIT parancsot használjuk. A javítani vagy módosítani kívánt sor az EDIT sorszám vagy az EDIT. (pont) paranccsal hívható elő. Az EDIT . forma főleg a futás közben jelentkező hibák javítására szolgál, mivel a hibajelzésben megadott sort hívja elő. Az előhívott sor kiíródik a képernyőn, majd a rendszer várja a szerkesztést vezérlő karakterek valamelyikét. A megengedett vezérlőkarakterek és funkcióik:

→ A cursort eggyel jobbra lépteti és kiírja a sor következő karakterét. A gomb folyamatos nyomása a szöveg folyamatos kiíratását eredményezi. Ezzel a funkcióval keressük meg a sorban a javítani vagy módosítani kívánt részt (célszerűen a módosítani kívánt rész utolsó karakterére pozicionálunk).

← A cursort eggyel balra lépteti és törli az ott lévő karaktert. A törölt karakterek helyére (de nem csak oda) tetszőleges új karakterek írhatók (egyedüli korlátot az jelenti, hogy a beírásokkal együtt sem lehet 210 karakternél több a sorban).

↓ Az előhívott sor eredeti formában maradva az editálás újraindul (ezzel pl. a hibás módosítások hatása megszüntethető).

SHIFT/← A már elvégzett módosításokat meghagyva visszaáll az editálandó sor elejére, és a sor újra módosítható.

SHIFT/→ Befejezi a sor feldolgozását és lezárja, majd a módosításokkal együtt a programtárolóba írja a sort.

RETURN Sorközben bárhol lenyomva befejezi a sor feldolgozását és törli a sor hátralevő részét.

Az EDIT lehetővé teszi a sorszám módosítását is, és a vezérlőkarakterek megfelelő kombinálásával lehetséges egy programsor több sorra tördelése. A szerkesztéssel módosított sorszámú sor bekerül a programtárolóba, de az eredeti sor is megmarad.

A program belövését segítő lehetőségek:

Ezekről a lehetőségekről a korábbiakban már szóltunk, így itt csak hivatkozunk rájuk. A belövést részben a TRON—TROFF parancspáros, részben az ON ERROR GOTO—RESUME utasításpáros segíti. Szintak-

tikai és futási hibák esetén a hibát tartalmazó sor a hiba jellegétől függően vagy automatikusan kiíródik, és ilyenkor közvetlenül használható az EDIT összes funkciója, vagy az EDIT . paranccsal előhívható. Jó szolgálatot tesz hiba esetén az adatok aktuális értékének ellenőrzése is, ami parancsként kiadott PRINT utasítással lehetséges (az adatmező csak új utasítássor vagy a RUN parancs hatására törlődik). Szintén a hiba helyére, ill. jellegére vonatkozó információkat kaphatunk az ERR és ERL rendszerváltozók értékének kiíratásával. A program dokumentálását és későbbi módosítását nagymértékben segítheti a kritikus részekhez beépített megfelelő magyarázószöveg. Ezt a REM utasítás alkalmazásával érhetjük el. A REM után tetszőleges szöveg írható, ami listázáskor megjelenik, de nem része a programnak. A REM hatása a sor teljes hátralévő részére kiterjed.

12. Gépi szintű programozási lehetőségek

A BASIC gépi reprezentáció lehetővé teszi, hogy a felhasználói program hozzáférjen a számítógép tárolójához, I/O portjaihoz, vagy gépi szintű programrészletet építsen a BASIC nyelvű programba. Ezek a funkciók a következő utasításokkal és függvényekkel valósíthatók meg:

PEEK (X) — Függvény, amely a decimálisan megadott, egész típusú X argumentumnak megfelelő tárolócím tartalmát decimális formában adja vissza. Felhasználásával a tároló tetszőleges része kiolvasható.

VARPTR (X)
VARPTR (X$) — Függvény, amely megadja az argumentumában szereplő változó tárolóbeli címét (ill. karakteres változó esetén a hárombyte-os címleíró blokk címét, ahol az első byte a karakteres változó karakterekben mért hosszát, a következő két byte pedig az első karakter címét tartalmazza). Felhasználásával vizsgálható az egyes változók tartalma, ill. kapcsolat teremthető a gépi szintű és BASIC programok közt.

POKE C, A1, A2... — Utasítás, amely a megadott, 0 és 255 közé eső A1, A2... értékeket eltárolja a tároló C címmel kezdődő részére (növekvő címek irányába haladva).

INP (X) — A 0 és 255 közé eső port-címek valamelyikéről beolvas egy egybyte-os adatot.

OUT X, A1, A2... — A 0 és 255 közé eső X port-címre kiadja az A1, A2.... egybyte-os adatokat (decimális formában megadva).

CALL (X) — Függvény, amely átadja a vezérlést az X címen lévő gépi programnak, és a gépi programmal a HL regiszterpárba töltött egész adatot adja eredményül.

CALL (X, Y) — Az előzőhöz hasonló függvényszerű eljárás, amely a vezérlésátadáson kívül a DE regiszterbe teszi a hívás előtt az egész típusú Y paraméter értékét.

Az egyes utasításokban használható tárolócímek a számítógép aktuális kiépítettségétől függenek, és a felhasználható értékeket a hardware-leírás tartalmazza. A tárolóba való beírást, ill. a gépi szintű és BASIC programok együttes alkalmazását csak gyakorlott felhasználóknak ajánljuk, mert a tároló egyes részeinek felülírása kiszámíthatatlan hatásokkal járhat.

CHR$ KÓDOK

CHR$ kód	karakter	CHR$ kód	karakter	CHR$ kód	karakter
1	Normál üzemmód	36	$	72	H
2	Nyújtott karakter	37	%	73	I
3	Negatív kép	38	&	74	J
4	Negatív alap	39	'	75	K
5	Aláhúzott karakter	40	(	76	L
6	Előtörlés	41	)	77	M
7	Hangjelzés	42	*	78	N
8	Egy karakter vissza	43	+	79	O
9	Horizontális tabulátor	44	,	80	P
		45	—	81	Q
10	—	46	.	82	R
11	—	47	/	83	S
12	CLS	48	0	84	T
13	RETURN	49	1	85	U
14	Csak kocsivissza	50	2	86	V
15	—	51	3	87	W
16	Alsóindex	52	4	88	X
17	Felsőindex	53	5	89	Y
18	2. kód inverze	54	6	90	Z
19	3. kód inverze	55	7	91	ó
20	4. kód inverze	56	8	92	Ö
21	Aláhúzás kikapcsoló	57	9	93	Á
22	6. kód inverze	58	:	94	Ü
23	—	59	;	95	ú
24	—	60	<	96	é
25	—	61	=	97	a
26	—	62	>	98	b
27	—	63	?	99	c
28	—	64	É	100	d
29	—	65	A	101	e
30	↑	66	B	102	f
31	←	67	C	103	g
32	Space	68	D	104	h
33	!	69	E	105	i
34	"	70	F	106	j
35	#	71	G	107	k

CHR$ kód	karakter	CHR$ kód	karakter	CHR$ kód	karakter
108	l	123	ő	138	⌊
109	m	124	ö	139	⌐
110	n	125	á	140	§
111	o	126	ü	141	≠
112	p	127	ű	142	@
113	q	128	π	143	{
114	r	129	[	144	}
115	s	130	]	145	~
116	t	131	←	146	Ω
117	u	132	→	147	±
118	v	133	↓	148	²
119	w	134	√	149	∎
120	x	135	⌙	150	Σ
121	y	136	∫	151	Π
122	z	137	Ft.		

DEMO PROGRAMOK

1. RAJZOLÓ DEMO-program

Rövid ismertetés: A program a TV képernyőjére egy téglalapot rajzol ki. Ön a téglalapba különböző ábrákat rajzolhat be.

Funkciondlis billentyűk:

S = STOP, program vége
T = Törlés
Z = ↑ irányban rajzol
N = ↓ irányban rajzol
I = → irányban rajzol
H = ← irányban rajzol
V = Váltó: törlés/írás

PROGRAMLISTA:

```
 10 CLS:A = 31:B = 51:S = 1
 20 FOR I = 30 TO 160
 40 SET (50,I):SET (220,I)
 50 NEXT I
 60 FOR I = 50 TO 220
 70 SET (I,30):SET (I,160)
 80 NEXT I
 90 A$ = INKEY$
100 IF A$ = "S" THEN 190
110 IF A$ = "T" THEN 10
120 IF A$ = "Z" THEN A = A+1
130 IF A$ = "N" THEN A = A—1
140 IF A$ = "H" THEN B = B—1
150 IF A$ = "I" THEN B = B+1
155 IF A$ = "V" AND S =1 THEN S =2: GOTO 160
156 IF A$ = "V" AND S=2 THEN S=1
160 IF S = 1 THEN SET (B, A)
170 IF S = 2 THEN RESET (B, A)
180 GOTO 90
190 END
```


2. TORPEDO DEMO-program

Rövid ismertetés: A számítógép egy 66×66-os sakktáblát generál (ezt Önnek nem rajzolja ki) és abban elrejt egy hajót, amit Önnek ki kell lőnie.
A „lövés" az X, Y koordináták megadásával történik.

```
  1 S = 1
 10 A = RND (66)
 20 B = RND (66)
 35 PRINT S "Lövés"
 40 INPUT X, Y
 50 S = S+1
 55 H = SQR ((X—A) * (X—A)+(Y—B) * (Y—B))
 60 PRINT "LÖVÉS-HAJÓ TÁVOLSÁG: "H" km"
 70 IF H = 0 THEN 90
 80 GOTO 35
 90 PRINT "ELSÜLLYEDT" S—1 "Lövésből": PRINT CHR$ (7)
100 END
```

NEMDEFINIÁLT FÜGGVÉNYEK KISZÁMÍTÁSA

FÜGGVÉNY	BASIC MEGFELELŐJE
Szekáns	SEX(X) = 1/COS(X)
Koszekáns	CSC(X) = 1/SIN(X)
Kotangens	COT(X) = 1/TAN(X)
Arkuszszinus	ARSIN(X) = ATN(X/SQR(1—X↑2))
Arkuszkoszinus	ARCOS(X) = —ATN(X/SQR(1—X↑2))+π/2
Arkuszkotangens	ARCOT(X) = ATN(X)+π/2
Arkuszszekáns	ARSEC(X) = ATN(X/SQR(X↑2—1))
Arkuszkoszekáns	ARCSC(X) = ATN(X/SQR(X↑2—1))+(SGN(X)—1)* * π/2
Szinusz hiperholikusz	SINH(X) = (EXP(X)—EXP(—X))/2
Koszinusz hiperbolikusz	COSH(X) = (EXP(X)+EXP(—X))/2
Tangens hiperbolikusz	TANH(X) = EXP(—X)/(EXP(X)+EXP (—X)) * * 2+1
Kotangens hiperbolikusz	COTH(X) = EXP(—X)/(EXP(X)—EXP (—X)) 1 * * 2+1
Szekáns hiperbolikusz	SECH(X) = 2/(EXP(X)+EXP(—X))
Koszekáns hiperbolikusz	COSH(X) = 2/(EXP(X)—EXP(—X))
Areaszinus hiperbolikusz	ARSINH(X) = LOG(X+SQR(X↑2+1))
Areakoszinusz hiperbolikusz	ARCOS(X) = LOG (X+SQR(X↑2—1))
Areatangens hiperbolikusz	ARTANH(X) = LOG((1+X)/(1—X))/2
Areakotangens hiperbolikusz	ARCOTH(X) = LOG((X+1)/(X—1))/2
Areaszekáns hiperbolikusz	ARSECH(X) = LOG((SQR(1—X↑2)+1)/X)
Areakoszekáns hiperbolikusz	ARCSCH(X) = LOG((SQR(1+X↑2)+1)/X) * SGN(X)

HIBAJELZÉSEK

A BASIC a program működése során előforduló hibák hatására AA ERROR IN NN típusú hibajelzést ad és a program működése félbeszakad. Az AA egy kétkarakteres hibakód, amelyhez a rendszer egy belső (az ERR rendszerváltozóba helyezett) numerikus kódot rendel. A következő táblázat az egyes AA üzeneteket, az ERR változó hiba esetén felvett értékét és az ERROR XX típusú utasításban a megfelelő hibajelzés generálásához szükséges XX értékeket tartalmazza.

ERR	XX	AA	*A hiba oka:*
0	1	NF	NEXT használata FOR nélkül
2	2	SN	Színtaktikai hiba a programban (végrehajtás során jelentkezik)
4	3	RG	RETURN utasítás GOSUB nélkül
6	4	OD	A szükségesnél kevesebb adat READ vagy INPUT utasításban
8	5	FC	Nem megengedett függvény-argumentum
10	6	OV	Fix- vagy lebegőpontos túlcsordulás (túl nagy szám)
12	7	OM	A programtároló betelt
14	8	UL	Hivatkozás nem létező sorszámra
16	9	BS	A DIM-ben specifikáltnál nagyobb index használata
18	10	DD	Indexes változó újradimenzionálása
20	11	/0	Nullával való osztás
22	12	ID	Az utasítás parancsként nem használható
24	13	TM	Típusilleszkedési hiba (numerikus és szöveges adatok összekeveredése)
26	14	OS	A karakteres változók tárolóhelye betelt
28	15	LS	A megengedettnél hosszabb karakterfüzér
30	16	ST	Túl bonyolult karakteres műveletsor
32	17	CN	A CONT parancs nem hajtható végre
34	18	NR	A program RESUME nélkül fejeződött be a hibakezeléskor
36	19	RW	RESUME használata ON ERROR GOTO nélkül
38	20	UE	USER ERROR (a rendszer minden nem pontosan azonosítható oku hibára ezt generálja)
40	21	MO	Hiányzó operandus egy kifejezésben
42	22	FE	File kezelési hiba

A többi hibakód jelenleg nincs kihasználva, így a felhasználó a saját programjából ERROR XX utasítással tetszőleges hibákat generálhat, ha XX értéke 23-nál nagyobb. Ilyen hibagenerálás esetén mindig az UE hibaüzenet jelenik meg és az ERR rendszerváltozó értéke 38.

IRODALOMJEGYZÉK

Donald Alcock: Ismerd meg a BASIC nyelvet!
Műszaki Könyvkiadó, Bp. 1983.

Bodor T., Gerő P.: A BASIC programozás technikája.
SZÁMALK, Bp., 1983.

dr. Kocsis A: BASIC I—II—III.
SZÁMALK, Bp., 1983.

ISBN 963 422 7031

A COSY megbízásából kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat
Felelős kiadó: dr. Petrus György
Megjelent 4,75 (A/5) ív terjedelemben

85-2437 Pécsi Szikra Nyomda — Felelős vezető: Farkas Gábor