Felhasználói Kézikönyv

128K

MOVITRADE ENTERPRISE®

ENTERPRISE

Felhasználói kézikönyv

NOVOTRADE RT.
1987

A könyv eredeti címe: ENTERPRISE Handbuch (1984)
Fordította: Lukács Sándor
Diószeghy Zoltán
Lektorálta: Pongor György

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT. igazgatója

Műszaki szerkesztő: Dévényi Erika

Szedte: ÖTLET Fényszedő Üzem
Készült Somogy Megyei Nyomdaipari Vállalat
kaposvári üzemében (11 A/5 ív)
Felelős vezető: Mike Ferenc igazgató

Hungarian translation © Lukács Sándor, Diószeghy Zoltán
Copyright © 1984 Enterprise Computers GmbH
– Sonnenstrasse 3., D–8000 München 2

Minden jog fenntartva.
Az ENTERPRISE COMPUTERS cég írásbeli hozzájárulása nélkül
tilos a könyvet vagy annak részeit
bármilyen eljárással (nyomtatás, fotókópia vagy egyéb technika),
elektronikus rendszerek felhasználásával
másolni, sokszorosítani, terjeszteni.

TARTALOM

1. ÖSSZEÁLLÍTÁSI UTASÍTÁS

Bevezetés 4
A számítógép összeállítása 7
A tv behangolása 9
A billentyűzet 9
A magnetofon csatlakoztatása 10
Gyári programok használata 11
Külső programtár használata 12
Egyéb problémák 12
A gép üzeneteinek nyelve 12

2. A PROGRAMOZÁS ALAPJAI

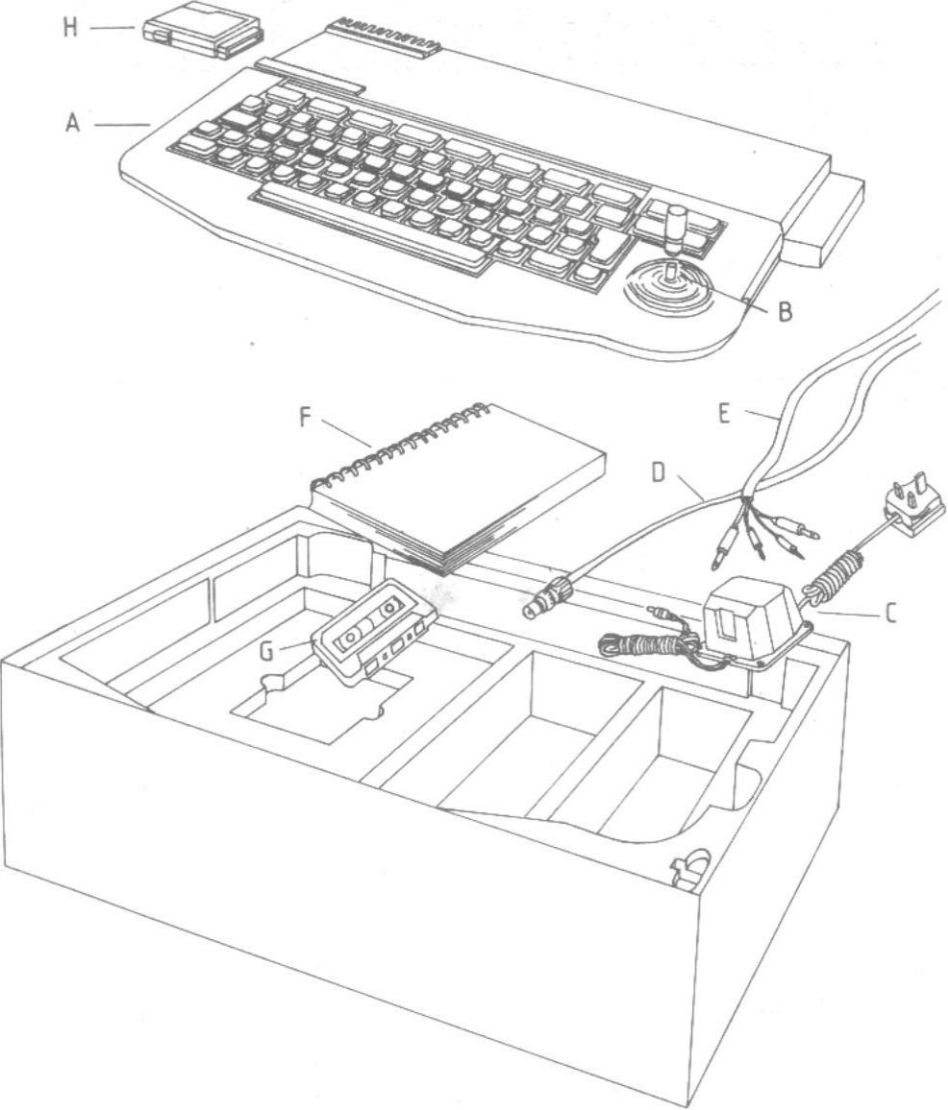
Bevezetés 15
Az első programok 15
Néhány szó a programozásról 22
A végrehajtási sorrend 25
A változók 26
A matematikai kifejezések 30
A kiírások formázása 34
A programszöveg szerkesztése 37
A szövegszerkesztő 41
A funkcióbillentyűk 44
A program tárolása kazettán 47

3. RÉSZLETES ISMERTETŐ

A karakterláncok 51
A ciklusok 57
A feltételek 62
Nagy mennyiségű adat tárolása 66
Függvények definiálása 72
A grafika 78
A karakterkészlet 91
A zenei kíséret 94
A feladattól a kész programig 101
A minimál BASIC 105
A csatornák 108
Kivételes esetek kezelése 109
A hálózat 111
A gépi kód használata 113

4. REFERENCIA KÉZIKÖNYV

A BASIC szabályai 117
Parancsok és utasítások 119
Gépi feltételek 144
Videofeltételek 148
Hangfeltételek 152
Belső függvények és változók 153
EXOS 160
Hibaüzenetek 164
A szövegszerkesztő üzenetei 174


H
A
B
F
E
D
C
G

1. ÖSSZEÁLLÍTÁSI UTASÍTÁS

BEVEZETÉS

Ez a fejezet a számítógép összeállításának és a magnetofon, ill. a külső programtár használatának az alapjait tartalmazza. A számítógép használatának egyéb részleteiről a későbbi fejezetekben lesz szó.

Aki még soha nem használt ilyen számítógépet, figyelmesen olvassa el ezt az útmutatót, mielőtt hozzáfog a gép összeállításához.

A szállítódobozban a következőket kell találnunk:

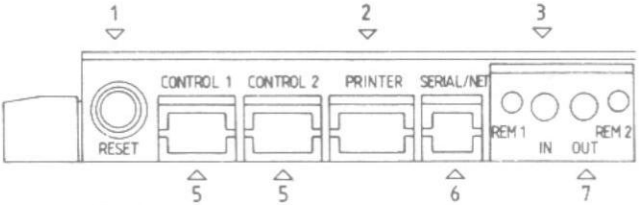
A - a számítógépet,
B - a botkormány karját,
C - a tápegységet,
D - a tv-csatlakozókábelt,
E - a magnetofon-csatlakozókábelt,
F - ezt a kézikönyvet,
G - egy magnetofonkazettát,
H - egy külső programtárat (cartridge).

A kazettán levő programok futtatásához szükségünk lesz a fentieken kívül egy kazettás magnetofonra is.

MŰSZAKI ADATOK:

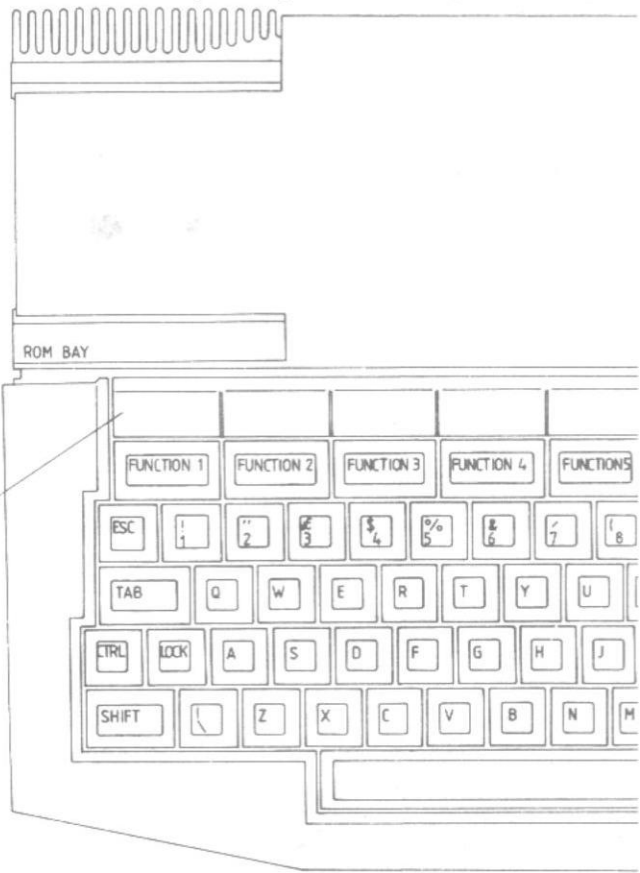
RAM:	128 Kbyte ebből 113 Kbyte Basic
ROM:	32 Kbyte
Feszültség:	220 V
Frekvencia:	50 Hz
Teljesítményfelvétel:	25 VA
Érintésvédelmi osztály:	II.

Az Enterprise különböző részeivel együtt a házi számítógépek új nemzedékéhez tartozik. A
végrehajtására is. Más számítógépekhez, illetve külső készülékek vezérléséhez megfelel

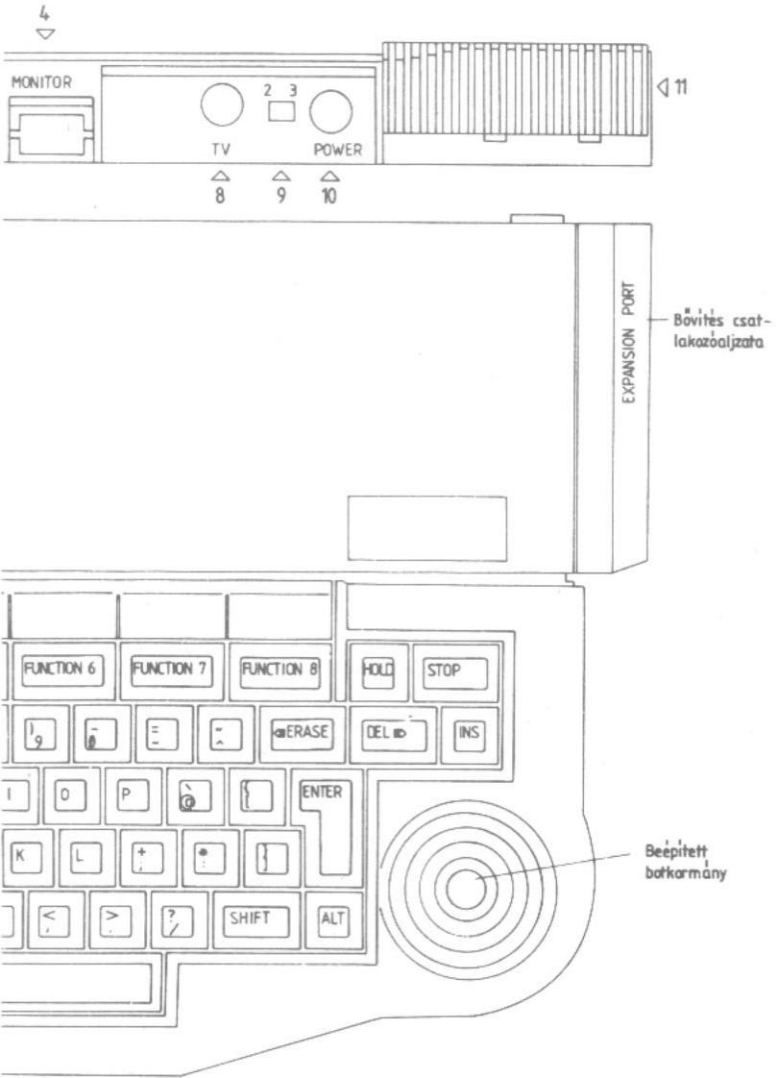


1 - újraindító nyomógomb
2 - nyomtató csatlakozó
3 - kazettás magnetofon csatlakozó - aljzat
4 - monitor csatlakozóaljzat
5 - csatlakozóaljzatok külső botkormányok, illetve más kiegészítések számára
6 - csatlakozóaljzat más számítógépek csatlakoztatására
7 - fejhallgató kimenet
8 - tv-antenna csatlakozóaljzat
9 - csak különleges tv-hez használt csatlakozóaljzat
10 - tápegység csatlakozóaljzat
11 - külső programtárak csatlakozó - aljzata

A funkcióbillentyűk címkéinek tartója



bővítő csatlakozó segítségével a gép alkalmassá tehető nagyobb számítási feladatok ben csatlakoztatható

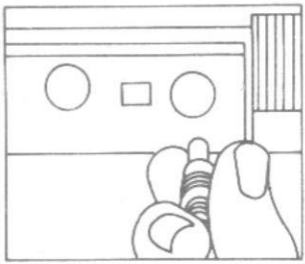


A SZÁMÍTÓGÉP ÖSSZEÁLLÍTÁSA

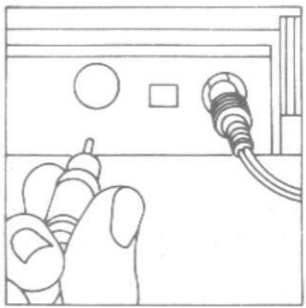
A számítógép összeállítására a következő sorrendet ajánljuk:

1. Húzzuk rá a beépített botkormány karját a gép jobb oldalán levő szárára!

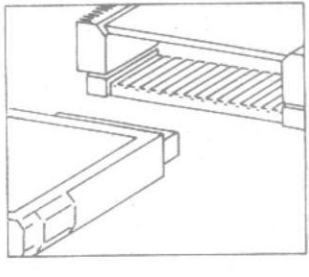
2. Csatlakoztassuk a tápegység csatlakozóját a gép hátoldalán levő, POWER feliratú aljzatba!



3. Helyezzük be a tv-csatlakozókábel hosszú pecekkel ellátott csatlakozóját a számítógép tv feliratú aljzatába, a vezeték másik végét pedig a tv antennacsatlakozójába!



4. Helyezzük a géphez tartozó külső programtárat a csatlakozóaljzatba, és nyomjuk be addig, amíg bepattan!



5. Kapcsoljuk be a tv-t és illesszük be a tápegység csatlakozóját a hálózatba. A készüléken ekkor egy piros lámpácskának kell kigyulladnia. A számítógép először néhány másodperc alatt ellenőrzi önmagát, majd – feltéve, hogy a tv-t előzőleg behangoltuk – a következő felirat jelenik meg a képernyőn:

ENTERPRISE
© 1985 Intelligent Software Ltd

A TV BEHANGOLÁSA

A számítógép a 36-os tv-csatornán sugározza a képet, tehát – a tv használati utasításának segítségével – erre a csatornára kell hangolnunk a készüléket. Mivel a számítógép hangjelet nem sugároz, a hangot kapcsoljuk ki a tv-n.

Ha a kép inog, a betűk szellemképesek, esetleg az egész kép féloldalra dől vagy ugrál, akkor óvatosan hangoljunk utána, amíg tiszta képet nem kapunk.

A programozható tv-ken célszerű külön csatornát fenntartani a számítógép számára, ezzel sok időt és fáradságot takaríthatunk meg.

A BILLENTYŰZET

A számítógépet leggyakrabban a billentyűzet segítségével irányítjuk. Az ehhez szükséges legfontosabb tudnivalók:

Ha egy billentyűt lenyomva tartunk, akkor kis idő múlva a gép ismételni kezdi a lenyomott betűt mindaddig, amíg el nem engedjük.

Nagybetűk használatához a kívánt betűvel együtt a SHIFT gombot is le kell nyomnunk.

Ha több nagybetűt akarunk egymás után beírni, akkor nyomjuk le együtt a LOCK és a CTRL billentyűket. Ettől kezdve a betűk leütésekor a megfelelő

nagybetű jelenik meg. Kisbetűket a SHIFT gomb lenyomásával írhatunk. Ez csak a betűbillentyűkre érvényes, a felső sorban levő számbillentyűkre nem.

Ahhoz, hogy ezeknél is felcseréljük a SHIFT-tel együtt, ill. nélküle való leütés hatását, ne a CTRL, hanem a SHIFT billentyűt nyomjuk le a LOCK-kal együtt a szolgáltatás bekapcsolásakor.

Az utóbbi két szolgáltatás kikapcsolásához a LOCK billentyűt kell még egyszer leütnünk.

A CTRL billentyű – hasonlóan a SHIFT-hez – a különböző egyéb billentyűk jelentését változtatja meg. Erről a későbbi fejezetekben lesz szó részletesen.

Az ALT billenttyűvel speciális jeleket tudunk – a többi billentyű segítségével - előállítani. A LOCK billentyűvel ez is állandósítható.

A funkcióbillentyűkkel egyes parancsok könnyebben begépelhetők, pl. a START parancs kiadható az F1 billentyűvel is. Az egyes funkcióbillentyűk hatása megváltoztatható, de a gép bekapcsolásakor már valamennyinek van valamilyen alapértelmezése. Erről a későbbi fejezetekben olvashatunk majd részletesebben.

Az ENTER billentyű a számítógépnek szóló üzeneteket zárja le. Ezek általában vagy programsorok, vagy közvetlenül végrehajtandó parancsok.

Az ERASE, ill. a DEL billentyűvel a kurzortól balra vagy jobbra levő betűt lehet törölni, ha véletlenül hibás szöveget írtunk. Ha ezekkel együtt a SHIFT vagy a LOCK billentyűt is leütjük, akkor egyszerre több betűt is törölhetünk.

Általában a begépelt betűk felülírják a már képernyőn levőket. Az INS billentyűvel szóközöket tudunk beszúrni a meglevő szövegbe. Ha a CTRL billentyűvel együtt ütjük le, akkor a számítógép nem írja felül a meglevőket, hanem beszúrja az új betűket. A szolgáltatás kikapcsolásához újra a CTRL és az INS billentyűket kell együtt lenyomni.

A számítógépen egyszerre több egymásba ágyazott program is futhat. Az ESC billentyűvel a soron következő programba térünk át.

A CTRL és F3 billentyűk együttes leütésével oszlopokat jelölhetünk ki a képernyőn. Ez után a TAB billentyű segítségével mindig a következő oszlop elejére tudunk ugrani.

A HOLD billentyűvel az éppen futó programot tudjuk felfüggeszteni, majd ismételt leütésével folytatni.

A STOP billentyűvel megszakíthatjuk a futó programot, különböző parancsokat adhatunk ki, majd - bizonyos esetekben - a CONTINUE paranccsal folytathatjuk azt.

A beépített botkormánnyal a villogó kurzort mozgatni tudjuk a képernyőn.

A MAGNETOFON CSATLAKOZTATÁSA

A kazettás magnetofon csatlakozókábelének mindkét végén négy-négy dugasz található. Ezek közül az egyik nagyobbat helyezzük a számítógép IN feliratú

aljzatába, míg a vezeték másik végén a neki megfelelő dugaszt a magnetofon EAR jelzésű csatlakozójába.

Ha a magnetofon távvezérelhető, akkor csatlakoztassuk az egyik kisebbik dugaszt a számítógép REM 1 jelű kimenetébe, a kábel másik végén levő csatlakozót pedig a magnetofon távvezérlő bemenetébe. Ezután kapcsoljuk be a távvezérlést az F4 billentyű leütésével, ettől kezdve – a gyorstekercselést kivéve – állandóan lejátszás állásban tarthatjuk a magnetofont, mert a számítógép automatikusan be-, ill. kikapcsolja, amikor szükséges.

GYÁRI PROGRAMOK HASZNÁLATA

A számítógéphez sokféle program illeszthető, a géppel együtt kapott kazetta is ilyen kész programokat tartalmaz. A kazettán van egy oktatóprogram a gép kezelésének elsajátítására, ezen kívül egy a gép jellegzetességeit bemutató-program és néhány játékprogram.

A programok betöltéséhez helyezzük a kazettát a magnetofonba és tekercseljünk a legelejére. Nyomjuk le az F1 (START) billentyűt és indítsuk el a magnetofont. A képernyő tetején ekkor a SEARCHING felirat látható, amely a program betöltődésének megkezdése után felcserélődik a LOADING..... szöveggel, ahol a pontok helyén a program neve áll. Minden kazettán tárolt programnak van ilyen neve, amelynek alapján könnyen megkülönböztethetjük őket akkor is, ha egy kazettán több program van.

A megfelelő hangerő beállításához segítségül a képernyő felső sávjában kis szines négyszög látható, amely vagy piros, vagy zöld, vagy színe a kettő között váltakozik. A beállítás akkor tökéletes, ha a négyzet színe váltakozik, ha piros, akkor lassan addig csökkentsük a hangerőt, amig el nem érjük az ideális szintet.

A programok betöltése kazettáról hosszabb ideig tart, az igazán nagy programoké akár öt percig is. Ha a korábban ismertetett jelzéseket látjuk a képernyőn, akkor nyugodtak lehetünk, a betöltés rendben folyik. Bármilyen hiba esetén a gép azonnal jelezni fog.

A mások által írt programok betöltése előtt figyelmesen olvassuk el a betöltési útmutatót, ez mindig tájékoztat a programok beolvasásáról.

Ha gyári programot használunk, gyakran megesik, hogy a billentyűk nem úgy működnek, ahogy azt megszoktuk. Ennek gyakori oka, hogy a program nem BASIC nyelven íródott, hanem a számítógép saját nyelvén, amelynek gépi kód a neve. Ilyenkor ne ijedjünk meg a szokatlan működéstől, hanem tanulmányozzuk át a program mellé kapott használati útmutatót.

A programokat tároló kazettákat mindig gondosan kezeljük:

Ne hagyjuk a kazettákat a tűző napon!

Vigyázzunk, hogy ne legyenek nedvesek vagy porosak! Használat után mindig tegyük el őket a tokjukba!

Soha ne érintsük meg a szalagot az ujjunkkal!

Ne hagyjuk a kazettákat olyan elektromos gépek közelében, amelyek mágneses teret gerjesztenek (televízió, rádió, hangszórók)!

Rendszeresen tisztítsuk a magnetofon lejátszófejét!

KÜLSŐ PROGRAMTÁR HASZNÁLATA

A külső programtárak használata sokkal egyszerűbb, mint a kazettáké. Az ilyen tár egy kis doboz, amelynek az egyik oldalán ún. élcsatlakozó vagy csatlakozósáv van, egy műanyag lemez fémcsíkokkal, amelyek a számítógépet kötik össze a doboz áramkőreivel.

A számítógéppel kapott programtár az IS-BASIC rendszert tartalmazza, amelynek segítségével BASIC nyelvű programokat tudunk írni, ill. futtatni. Természetesen más programok is kaphatók ilyen kivitelben, csatlakoztatás után ezek is azonnal használhatók.

A program futtatásához csak a tárat kell elhelyeznünk a ROM-BAY feliratú csatlakozóaljzatba, majd kétszer megnyomni a gép hátulján levő újraindító (RESET) gombot. Ha más programot akarunk használni, akkor kapcsoljuk ki a gépet, és a programtárat kicserélve újra kapcsoljuk be.

Ha semmilyen programtárat sem csatlakoztathatunk a géphez, akkor a beépített szövegszerkesztőt használhatjuk, amelyről a későbbi fejezetekben olvashatunk bővebben.

EGYÉB PROBLÉMÁK

Ha véletlenül hibásan gépelünk be valamit, a számítógép jelzi azt. Ezután kényelmesen kijavíthatjuk a hibát és az ENTER leütésével most már helyes parancsot adhatunk. Ha teljesen összezavarodnánk, nyomjuk meg az újraindító gombot és kezdjük elölről a munkát.

Ne féljünk attól, hogy hibás billentyű leütésével elronthatjuk a számítógépet – a gép nem megy tönkre attól, hogy nem ért bennünket. Sőt, nyugodtan próbáljunk játszadozni a billentyűzettel és figyeljük meg, mit tapasztalunk.

FIGYELEM!

A számítógép tetejének bal oldalán levő bordázattal egyes alkatrészek felesleges hőjét adja le a gép. Nem baj, ha ez a rész forró, ez normális működési jelenség.

A SZÁMÍTÓGÉP ÜZENETEINEK NYELVE

A forgalomba kerülő Enterprise számítógépek egyik része angol, másik része német nyelvű üzeneteket ír ki. Ennek az az oka, hogy a különböző gépek más-más forrásból érkeztek az országba. Ez az eltérés azonban sem a gép kezelését,

sem programozását nem módosítja. A programokat, parancsokat mindig a gép programozási nyelvén, azaz BASIC nyelven kell begépelnünk.

A különböző gépeken az egyes billentyűk elhelyezése is eltér, főleg az írásjeleké. A betűbillentyűk közül csak az Y és a Z cserél helyet: az angol nyelvű gépeken Y van az U mellett alulról a harmadik sorban, a német nyelvűeken pedig a Z. Ez azonban a szövegek begépelését nem nehezíti meg, mert a saját gépén mindenki a megfelelő helyen megtalálja az egyes jeleket.

Néhány számítógép mindkét nyelven tud üzeneteket küldeni, ezeken a „:UK" paranccsal tudunk angol, a „:BRD" paranccsal pedig német nyelvű üzemmódba átkapcsolni. Vigyáznunk kell azonban arra, hogy ezeken a gépeken a billentyűk feliratai a német üzemmódnak felelnek meg, tehát angol üzemmódba átkapcsolva az egyes billentyűk leütésekor nem mindig a rajtuk levő jelek jelennek meg a képernyőn. Éppen ezért csak azoknak ajánljuk az angol üzemmód használatát, akik tudnak angolul. Látni fogjuk azt is, hogy ezeknek az üzeneteknek a megértéséhez nincs szükség nyelvtudásra, ugyanakkor a gépelést igencsak megnehezíti, ha a billentyűkön nem a megfelelő feliratok vannak.

Az eltérő nyelvhasználat egyetlen ponton módosítja az utasításokat. Az angol nyelvű gépeken a csatornaszám előtt mindig font jel áll, míg a német nyelvűeken kettős kereszt (#). Erre nagyon gondosan oda kell figyelnünk, és a példaprogramok begépelése során is gépünk nyelvének megfelelően kell alkalmaznunk.

A számítógép általában két dolog kapcsán ír ki számunkra valamilyen üzenetet. Az egyik, amikor hibát észlel, a másik, amikor a szövegszerkesztő használata közben valamire fel akarja hívni a figyelmünket. A hibaüzenetek listáját mindhárom nyelven – angol, német, magyar – megtaláljuk a könyv végén, a velük foglalkozó fejezetben. Az egyes üzenetek jelentését a számuk alapján tudjuk a magyar nyelvű táblázatból kikeresni.

A szövegszerkesztő üzeneteit szintén külön fejezet tárgyalja.

2. A PROGRAMOZÁS ALAPJAI

BEVEZETÉS

Ebben a fejezetben a számítógép használatának alapjaival foglalkozunk. Tanulás közben sok-sok példaprogramot, gyakorlatot próbálunk ki, ez ugyanis a legjobb módszer a gép lehetőségeinek kitapasztalására. A szakkifejezéseket is csak fokozatosan kezdjük majd megismerni és használni.

A fejezet az egyes kérdéseket a tanulás számára szerintünk a legkedvezőbb sorrendben tárgyalja, de az egyes alfejezetek önállóan is megérthetők, így mindenki a számára legkényelmesebb módon sajátíthatja el a tudnivalókat. Az ez után következő fejezet a programozás minden részletét mégegyszer ismerteti majd, de ennél részletesebben és pontosabban. Végül az utolsó fejezetben az Enterprise használatának aprólékos ismertetését találhatjuk, ezt azonban csak azután érdemes kézbe venni, ha jól megismertük az alapokat.

AZ ELSŐ PROGRAMOK

Gépeljük be a következő programot. Vigyázzunk, mert a gép igen finnyás, a legapróbb gépelési hibákat sem fogadja el! Mielőtt az egyes sorokat az ENTER-rel lezárnánk, ellenőrizzük, hogy nincs-e benne hiba. Ne feledkezzünk meg a sorok elején álló számokról, ezek igen fontosak, ugyanakkor a számok után következő szóközöket nyugodtan elhagyhatjuk, ezeket a gép automatikusan beszúrja majd. Figyeljünk arra, hogy a számítógép külön jelet használ a nullára (0) és a nagy o betűre (O).

```
100 GRAPHICS
110 PLOT 640,360,
120 !
130 DO
140   FOR SUGAR=250 TO 1 STEP -16
150     SET INK RND(3)+1
160     PLOT ELLIPSE SUGAR,SUGAR,
170     PLOT PAINT
180   NEXT SUGAR
190   PING
200 LOOP
210 !
220 END
```

A GÉPELÉSI HIBÁK JAVÍTÁSA

A gépelési hibákat igen könnyen kijavíthatjuk. Ha még mindig abban a sorban áll a kurzor, ahol a hiba van, akkor az ERASE billentyűvel töröljük a sort addig, amíg a hibához nem érünk, majd innen kezdve újra írjuk a sort. Ha már más sorban vagyunk, akkor a beépített botkormánnyal a hibás sor végére visszük a kurzort, és az előzőek szerint járunk el. Javítás után ne felejtsük el ENTER-rel lezárni a sort és a kurzort visszavinni az első üres sor elejére. A PROGRAMSZÖVEG SZERKESZTÉSE c. alfejezetből még a javítások sok más módját is megismerjük majd.

A PROGRAMOK FUTTATÁSA

Miután begépeltük a programot, egy új sor elejére írjuk be a RUN kulcsszót, majd küldjük el az ENTER-rel. Ugyanezt érhetjük el, ha lenyomjuk az F1 funkcióbillentyűt, ami után már az ENTER-t sem kell használnunk. A funkcióbillentyűkről később még részletesen lesz szó. Ha a programban valamit elrontottunk, akkor kis idő múlva a gép visszajelez, leggyakrabban a „nicht verstanden" üzenettel. Ettől ne ijedjünk meg, üssük le az F5 majd az F2 billentyűket, mire újra feltűnik a képernyőn a programunk. Javítsuk ki a hibákat és próbáljuk meg újra futtatni.

A számítógépnek szánt parancsainkat mindig az ENTER billentyűvel kell zárnunk. Amíg ugyanis nem küldjük el vele a parancsot, addig a gép nem kezdi meg a végrehajtást, mert arra vár, hogy hátha meggondoljuk magunkat és módosítjuk a sort. A funkcióbillentyűk lenyomása után nem kell ENTER-t ütnünk, erről később még részletesen is olvashatunk.

A program megállítására üssük le a STOP billentyűt, mire a „STOP in Zeile..." üzenet jelenik meg, a pontok helyén egy számmal.

Ha mégis folytatni akarjuk a programot, akkor gépeljük be a CONTINUE parancsot, vagy – ami ezzel egyenértékű – üssük le egyszerre a SHIFT és az F1 billentyűt. Ezek hatására a program onnan folytatódik, ahol megszakítottuk.

A HOLD billentyű leütése felfüggeszti a program futását, a folytatáshoz a billentyűt újra le kell ütni. Ez lehetőséget nyújt arra, hogy megállítsuk a programot, ha mondjuk csöng a telefon, vagy kimerevítsük a képernyőn állandóan változó képet, ha jobban meg akarjuk vizsgálni.

Ha már unjuk a programot, akkor állítsuk meg, és gépeljük be a következő utasítást:

```
135 SET PALETTE RND(256), RND(256), RND(256), RND(256)
```

majd indítsuk el újra RUN-nal.

A KULCSSZAVAK GÉPELÉSE

Amikor a gép a programot listázza, akkor a BASIC kulcsszavakat – amilyen például a RUN, GRAPHICS, DO stb. – mindig nagybetűkkel írja, ezzel is hangsúlyozva őket. Mi azonban nyugodtan írhatjuk kisbetűvel, a gép tökéletesen megérti ezeket is.

A szakszavak magyarázatát a függelékben találhatjuk meg.

MI IS A PROGRAM?

A program a számítógépnek szóló utasítások sorozata, amely megmondja a gépnek, hogy mit kell tennie. Ebben nincs semmi különleges vagy titokzatos, a gép ugyanis magától semmire sem képes, mindenre nekünk kell utasítanunk. Éppen ezért a programoknak nagyon pontosaknak és részleteseknek kell lenniük. A programozás, mint minden más időtöltés, lehet komoly feladat, de lehet csupán szórakozás is.

A KÉPERNYŐ TÖRLÉSE

A programot megállítva, a képernyőn mindenféle ábrát láthatunk, ami igen zavaró, mert nem tudjuk tőle figyelemmel kisérni, mit is gépelünk. Töröljük tehát le a képernyőt a TEXT paranccsal vagy az azzal azonos hatású F5 billentyűvel. Ha a programban nem adtuk ki a GRAPHICS utasítást, akkor a CLEAR SCREEN paranccsal is törölhetjük a képet. A képernyőt törlő utasítások a programsorokat nem módosítják, azokat a gép nem felejti el, hanem bármikor listázhatók.

A PROGRAM LISTÁZÁSA

Valahányszor programot szeretnénk módosítani vagy olvasni, szükségünk van a program szövegének listájára. Ezt a LIST paranccsal vagy az F2 billentyűvel irathatjuk ki. Ilyenkor a program sorait a gép a sorszámok sorrendjében írja ki. Ha a program túl hosszú és nem fér ki a képernyőre, akkor a felső sorok lecsúsznak a képernyőről. Ahhoz, hogy ezeket a sorokat is elolvashassuk, a listázást bármikor felfüggeszthetjük a HOLD billentyűvel vagy megállíthatjuk a STOP-pal.

A PROGRAM TÖRLÉSE

Ha már meguntuk a programot és másikat szeretnénk kipróbálni, a NEW paranccsal törölhetjük. Az utasítás hatására a gép a programot végleg elfelejti, ha újra futtatni szeretnénk, akkor újra be kell gépelnünk.

PÉLDAPROGRAMOK

A következő programok a gép képességeit, lehetőségeit mutatják be. Próbáljuk ki őket!

```
10    PROGRAM "Tuzcsatorna"
20    !
30    ! Ez a program egy szines
40    ! csatornat rajzol a kepernyore,
50    ! felrobbano tuzlabdakkal
60    !
100   GRAPHICS HIRES 256
110   LET X=640:LET Y=360
120   FOR R=1 TO 255
130     SET INK R
140     LET A=X-R-220:LET A1=Y-R-50
150     LET C=X+R+220:LET C1=Y+R+50
160     PLOT A,A1;A,C1;C,C1;C,A1;A,A1
170     PRINT R
180   NEXT
190   FOR Labda=1 TO 100
200     CALL Tuzlabda(256,X,Y)
210   NEXT
220   !
230   END
240   !
250   !
1000  DEF Tuzlabda(Szin,A,B)
1010    SET LINE MODE 3
1020    SET INK RND(Szin)
1030    FOR Jel=1 TO 2
1040      FOR Sugar=1 TO 650 STEP 30
1050       PLOT A,B,ELLIPSE Sugar,Sugar
1060      NEXT
1070    NEXT
1080    SET LINE MODE 0
1090  END DEF
```

```
100 ! Ez a program negyszogeket rajzol
110 !
120 ! A 100-140 sor csak megjegyzes,
130 ! azaz nem kell feltetlenul beirni
140 !-----------------
150 CLEAR SCREEN
160 PRINT AT 5,13:"EZ A PROGRAM"
170 PRINT AT 6,10:"NEGYZETEKET RAJZOL."
180 PRINT AT 8,1:"A program meg fog kerni,"
190 PRINT AT 9,1:"hogy adj meg nehany szampart."
200 PRINT AT 10,1:"Az elso legyen"
210 PRINT AT 11,1:"kisebb mint 1279, a masodik pedig"
220 PRINT AT 12,1:"mint 719. Minden szam utan"
230 PRINT AT 13,1:"usd le az ENTER-t!"
240 FOR A=1 TO 5000
250 NEXT A
260 !-----------------
270 ! A 240-250 sor hatasara a gep
280 ! kb. 10 masodpercet var
290 !-----------------
300 DO
310   CLEAR SCREEN
320   INPUT AT 5,5,PROMT"Az egyik oldal:":X
330   INPUT AT 6,29,PROMT" ":Y
340   INPUT AT 8,5,PROMT"A masik oldal:":V
350   INPUT AT 9,34,PROMT" ":W
360   PRINT AT 11,5:"Mennyi ideig legyen"
370   PRINT AT 12,5:"lathato a negyszog?"
380   INPUT AT 14,5,PROMT"Masodperc:":Ido
390   !-----------------
400   ! A 450-480 sorok beolvassak
410   ! a negyszog parametereit
420   ! es hogy meddig
430   ! legyen lathato
440   !-----------------
450   GRAPHICS
460   PLOT X,Y;X,W;V,W;V,Y;X,Y
470   FOR B=1 TO 500*Ido
480   NEXT B
490   TEXT
500   PRINT AT 15,18:"Ujra?"
```

```
510   DO
520      INPUT AT 17,17,PROMT"i vagy n:":Valasz$
530   LOOP WHILE Valasz$<>"i" AND Valasz$<>"n"
540 LOOP WHILE Valasz$="i"
550 !
560 END !Itt van a program vege
```

```
100 ! Ez a program 10 szamot rendez
106 ! nagysag szerint sorba
108 !-----------------
110 NUMERIC Tomb(1 TO 10)
120 NUMERIC Valt,Szam,Max
130 CLEAR SCREEN
150 PRINT AT 10,10:"SZAMRENDEZES"
160 FOR N=1 TO 10
170   PRINT AT 14,10:"Az ";N;"-ik szam";
180   INPUT PROMT":":Tomb(N)
190   PRINT AT 14,25:" "
200 NEXT N
210 CLEAR SCREEN
220 PRINT AT 20,20:"Rendezek..."
240 LET Veg=10
250 FOR X=1 TO 10
255 LET Max=0
260   FOR Y=1 TO Veg
270     IF Tomb(Y)>Max THEN LET Max=Tomb(Y)
280     IF Tomb(Y)=Max THEN LET Szam=Y
290   NEXT Y
300   LET Valt=Tomb(Veg)
310   LET Tomb(Veg)=Max
320   LET Tomb(Szam)=Valt
340   LET Veg=Veg-1
350 NEXT X
360 FOR X=1 TO 10
370   PRINT Tomb(X)
380 NEXT X
390 END
```

```
100 ! Ez a program kiszamitja egy kor
110 ! keruletet, ill. teruletet
120 !-------------------
130 LET A$="A kor "
140 LET B$="Add meg a kor sugarat:"
150 NUMERIC Sugar,Terulet,Kerulet
160 DO
170   CLEAR SCREEN
180   PRINT AT 10,10:"1) TERULET"
190   PRINT AT 11,10:"2) KERULET"
200   PRINT AT 12,10:"3) BEFEJEZES"
210   DO
220     PRINT AT 15,10:"Add meg a valasztasod"
230     INPUT AT 16,10,PROMT"szamat:":Szam
240   LOOP WHILE Szam<1 OR Szam>3 OR Szam<>INT(Szam)
250   CLEAR SCREEN
260   IF Szam=1 THEN
270     INPUT AT 10,1,PROMT B$:Sugar
280     LET Terulet=PI*Sugar↑2
290     PRINT AT 15,5:A$;"terulete:";
300     PRINT AT 16,4:Terulet
310     FOR X=1 TO 5000
320     NEXT X
330   ELSE IF Szam=2 THEN
340     INPUT AT 10,1,PROMT B$:Sugar
350     LET Kerulet=2*PI*Sugar
360     PRINT AT 15,1:A$;"kerulete:";
370     PRINT AT 16,1:STR$(Kerulet)
380     FOR X=1 TO 5000
390     NEXT X
400   END IF
410 LOOP WHILE Szam<>3
420 END
```

A PROGRAMOK MÓDOSÍTÁSA

Próbáljuk meg módosítani ezeket a programokat. Ha a program kulcsszavait változtatjuk meg, akkor könnyen oda juthatunk, hogy a gép számára érthetetlenné, hibássá válik a program. Például a számokat sorba rendező programban a 250-es sort írhatnank úgy is, hogy:

```
25Ø FOR X=1 TOO 1Ø
```

Ekkor a program első részének lefutása után a gép megáll és hibaüzenetet küld, mivel nem érti ezt a sort:

```
***nicht verstanden
25Ø FOR X=1 TOO 1Ø
```

A különböző hibaüzenetek listáját a függelékben találhatjuk meg.

Az idézőjelek között álló szövegeket nyugodtan módosíthatjuk, ennek hatására csak a program üzenetei változnak meg. Megváltoztathatjuk a program szövegében levő számokat is, és a programokat lefuttatva megfigyelhetjük a változtatás hatását. Általában azt fogjuk tapasztalni, hogy vagy az változik meg, hogy valamit hányszor csinál a program, vagy az, hogy hova írja ki az üzeneteit a képernyőn. Egyes számok egészen más dolgokat jeleznek, például színeket. Az ilyen számokat úgy hívjuk, hogy kódszámok, róluk még bővebben is olvashatunk majd a továbbiakban.

ÚJRAINDÍTÁS

Előfordulhat, hogy annyira összezavarjuk a programot, hogy szeretnénk elölről kezdeni a munkát. Ilyenkor nyugodtan nyomjuk meg az újraindító (RESET) gombot, ezzel ugyanazt érjük el, mintha ki-, majd újra bekapcsolnánk a gépet azzal a különbséggel, hogy a korábban beírt programsorok nem vesznek el, bármikor kilistázhatók. Ez természetesen csak végszükség esetén célszerű, más esetekben elég a STOP gombbal megállítanunk a programot.

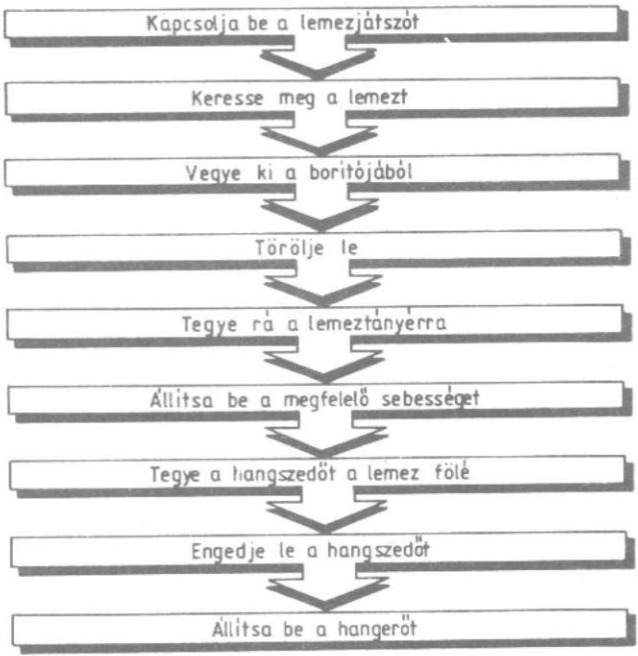
NÉHÁNY SZÓ A PROGRAMOZÁSRÓL

Az eddigiekből megismertünk néhány programot, de működésükkel még nem foglalkoztunk. Arra a kérdésre,hogy mi is a program, azt válaszoltuk, hogy a gép számára adott utasításaink sorozata. A program tehát egy problémą megoldásának a módját írja le a gép számára. A számítógéppel egyrészt növelhetjük és meggyorsíthatjuk számítási képességeinket, másrészt természetesen futtathatók játékprogramok és saját szórakoztató programjaink is.

A számítógépek az utasításokat vagy szavak, vagy számsorozatok formájában tudják felhasználni. Mi szavakat fogunk használni az Enterprise irányításához. Minden egyes ilyen szó egy-egy kis utasítás, belőlük a programot ugyanúgy rakhatjuk össze, ahogy a kirakós játékot elemeiből, vagy egy történetet szavakból, azaz egyre nagyobb részeket, utasításcsomagokat, önálló

részfeladatokat képezünk belőlük. A programozás lényege: parancsokat adni a gépnek arra, hogy mit és milyen sorrendben csináljon.

Az következő ábra is azt mutatja, hogyan lehet egy nagyobb feladatot – a lemezhallgatást – kisebb végrehajtandó munkákra felbontani.



PROGRAMOZÁSI NYELVEK

Ugyanúgy, ahogy egy emberrel is sokféleképpen lehet beszélgetni, a számítógépnek is sokféle módon adhatunk utasításokat. Sőt, hasonlóan az emberi nyelvekhez, sokféle számítógépes nyelv is létezik. Ezek a nyelvek tartalmaznak bizonyos korlátozásokat, ugyanakkor más feladatok elvégzésére igen jól alkalmazhatóak. Egyes nyelvekkel jól tudunk kezelni sok-sok elemből álló adathalmazokat, listákat, névsorokat, másokkal grafikus ábrákat, képeket tudunk kényemesen keszíteni, megint mások pedig arra jók, hogy könnyen lehet velük megtanulni a programozás alapjait.

MI A BASIC?

Az a nyelv, amit ebben a könyvben ismertetünk, a BASIC. Mivel a BASIC az angol nyelv egyes szavait használja, méghozzá az eredetivel megegyező formában és nagyon hasonló tartalommal, ezért az angolul tudók számára a programok még akkor is könnyen érthetőek, ha a programozásban egyelőre nincs gyakorlatuk. Azoknak sem kell azonban megijedniük, akik nem tudnak angolul, mert nagyon kevés és igen könnyen megjegyezhető szót kell megtanulniuk.

Ebben a könyvben a programok mind BASIC nyelven íródtak és amíg a géphez az IC-BASIC külső programtár csatlakozik, addig a gép is ezen a nyelven ért. Ne felejtsük el, hogy a könyvben levő információk a BASIC nyelvre, annak is az IS-BASIC nevű változatára vonatkoznak, más nyelvek ettől formájukban és gondolkodásmódjukban is teljesen eltérőek.

KULCSSZAVAK

A BASIC nyelv által használt szavakat kulcsszavaknak nevezzük, ezek mindegyike egy vagy több feladat végrehajtására utasítja a számítógépet. Például a PRINT kulcsszó az utána álló számokat, ill. karakterláncokat iratja ki a képernyőre. Sok más kulcsszó is van, ezek egy részével már megismerkedtünk (pl. RUN, LIST, CONTINUE), másokról majd csak később lesz szó.

KARAKTERLÁNCOK

Karakterláncokkal a leggyakrabban a PRINT utasításokban találkozunk, pl. a PRINT „HELLO" utasításban az idézőjelek között álló szöveg egy karakterlánc. Ezeket a számítógép is úgy tekinti, mint betűről betűre átmásolandó idézeteket, tehát ha valamit idézőjelek közé írunk, akkor az a képernyőn változtatás nélkül jelenhet meg a program futása során. Ha magát az idézőjelet szeretnénk megadni, akkor azt kétszer kell egymás után leírnunk az idézetben, például a PRINT „elotte"„utana" utasítás hatására a képernyőn a következő szöveg jelenik meg:

elotte"utana

KARAKTEREK

A karakter szóval még sokszor találkozunk, ezért tudnunk kell, mik is azok a karakterek. A számítógép által használt valamennyi betű, számjegy, jel, írásjel, mindazok az elemek, amelyek a képernyőn egy betűnyi helyet foglalnak el, karakterek. A gép által használható összes karakter együttesét a gép karakterkészletének nevezzük.

A VÉGREHAJTÁSI SORREND

A BASIC programsorok elején álló szám igen fontos a végrehajtás szempontjából.

Gépeljük be azt az utasítást, hogy:

PRINT „HELLO"

Miután elküldtük a parancsot az ENTER-rel, a gép rögtön végrehajtja azt és kiírja a következő sorba az idézőjelek között álló szöveget, azaz azt hogy HELLO. Korábban viszont tapasztaltuk már azt, hogy a programjainkban levő utasításokat csak akkor hajtja végre a gép, ha lefuttatjuk a programot. A különbséget a sor elején álló sorszám okozza!

A BASIC programok sorokból állnak, minden sornak van saját sorszáma.

Ha olyan sorszámmal írunk be egy sort, amilyen már van a programban, akkor a korábbi sort felülírja az új. Egy sort tehát úgy is törölhetünk, hogy begépeljük a számát és elküldjük az ENTER-rel, ezáltal egy üres sor keletkezik, amit a gép nem is tárol, így ezt a sort kitöröltük. Egyszerre több sort a DELETE paranccsal törölhetünk, például a DELETE 10 TO 100 parancs a 10-től 100-ig terjedő sorokat törli.

Az, hogy milyen sorrendben írjuk be a sorokat, soha nem lényeges. A gép mindig számsorrendbe rendezi őket, és csak ez a sorrend számít.

A program futása során a számítógép a legkisebb sorszámú sornál kezdi a végrehajtást és végighalad a legnagyobbig, feltéve, hogy bizonyos utasításokkal nem módosítjuk ezt. Ezek a módosítási lehetőségek – amint azt még látni fogjuk – igen fontosak a programozás szempontjából. Az egyik ilyen parancs az END, aminek hatására a gép befejezi a program végrehajtását, úgy tekinti, hogy a programnak itt a vége. Sok más végrehajtási sorrendet módosító utasítás is létezik, ezeket majd később ismerjük meg.

PARANCSMÓD

Ha egy utasítást sorszám nélkül adunk meg, azaz azonnal végrehajtatjuk a géppel, akkor a gépet parancsmódban használjuk. Ilyen parancsmódú utasítás a RUN vagy a LIST, de gyakran használjuk a számítógépet parancsmódban számológépként is.

A billentyűzeten könnyen megtalálhatjuk a négy alapművelet jelét, bár meg kell szoknunk, hogy itt az osztás jele a „/" és nem a „:", a szorzásé pedig a „*" és nem a kereszt vagy a pont. A PRINT kulcsszó után írva egy képletet a gép kiszámítja az értékét, és az eredményt azonnal kiírja. Például a PRINT 2+2 hatására válaszul a következő sorban megjelenik a 4. Ezekben a képletekben mind a négy alapműveletet használhatjuk, sőt lehetőségünk van számításokra is.

```
PRINT SQR(1ØØ)
```

A fenti sor hatására a számítógép a 100 négyzetgyökét írja ki (angolul a négyzetgyök square root). A gép sok ehhez hasonló, ún. beépített függvényt ismer, ezeknek a listáját a függelék tartalmazza.

A parancsmódban elvégzett számítások a programsorokat nem módosítják, tehát azokat nyugodtan használhatjuk akár a program egyes számításainak ellenőrzésére is.

A legtöbb BASIC utasítás parancsmódban ugyanolyan hatású, mint a program futása közben. Nyugodtan próbáljuk ki a még ismeretlen szavakat parancsmódban; ha valamit nem értünk, keressük ki a kulcsszavak táblázatából a szó ismertetését. Ez azt is tartalmazza, hogy mely szavak nem használhatók parancsmódban.

A VÁLTOZÓK

A változókat kis példákból fogjuk megismerni. Képzeljünk el két dobozt, legyen a jelük X és Y. Mindkettőbe egy-egy számot tehetünk be, azt később onnan kivehetjük vagy egy másikkal felcserélhetjük. Mindkét dobozban egyszerre csak egy szám lehet. A változók is ilyen kis dobozokként képzelhetők el. A következő program beolvas két számot, beteszi őket egy-egy dobozba, más néven változóba, majd kiszámítja a két doboz, azaz a két változó tartalmának összegét és azt beteszi egy harmadik változóba. Ez a program anélkül is érthető, hogy lefuttatnánk.

```
1Ø  !-----------------
2Ø  ! A 6Ø es a 7Ø sor a ket szamot
25  ! olvassa be az X es az Y nevu
3Ø  ! valtozokba. A szamok beirasa utan
4Ø  ! ne felejtse el leutni az ENTER-t
5Ø  !-----------------
6Ø  INPUT PROMT"X erteke= ":X
7Ø  INPUT PROMT"Y erteke= ":Y
8Ø  !-----------------
9Ø  ! A 12Ø sor osszeadja az X es az Y
95  ! tartalmat es az eredmenyt a
1ØØ ! Z valtozoba teszi
11Ø !-----------------
12Ø LET Z=X+Y
13Ø !-----------------
14Ø ! A 16Ø sor az eredmenyt irja ki
```

```
150 !-------------------
160 PRINT X;"+ ";Y;"= ";Z
170 END
180 !-------------------
190 ! A 170 sor a szamitogep es a mi
200 ! szamunkra is azt jeloli, hogy
210 ! a programnak itt a vege
220 !-------------------
```

MEGJEGYZÉSEK A PROGRAM SZÖVEGÉBEN

Azok a sorok, amelyek felkiáltójellel kezdődnek, csak a számunkra fontos emlékeztető szövegeket tartalmazzák, amik alapján később is könnyen meg tudjuk majd állapítani, hogy a program egyes részei mit csinálnak. A számítógép ezeket a sorokat nem törli a program szövegéből, de nem hajtja őket végre, mert a felkiáltójelből észreveszi, hogy ezek a sorok nem neki szólnak. Ha a programsorban levő utasítások után egy felkiáltójelet teszünk, a sor további részét a gép megjegyzésként kezeli. A közvetlenül a sorszám után álló felkiáltójelet a REM kulcsszóval is helyettesíthetjük (angolul megjegyzés = remark). A megjegyzések segítségével könnyen megjelölhetjük azokat a sorokat, amelyeket később esetleg ki akarunk keresni.

Soha ne felejtsük el, hogy a megjegyzések sokkal olvashatóbbá teszik a program szövegét. Bár a végrehajtás szempontjából nincs jelentőségük, a használó számára mégis fontosak!

Az INPUT PROMT UTASÍTÁS

Ennek a parancsnak a hatására a gép egy kérdést ír ki a képernyőre, majd a választ egy változóba olvassa be.

Ha csak az INPUT utasítást használjuk önmagában, akkor a gép csak egy kérdőjelet ír ki, és egy piros kurzor jelenik meg, amikor ezt az utasitást hajtja végre és a válaszra vár. A PROMT és az utána következő karaktersorozat a program futása közben a képernyőre másolódva emlékeztet bennünket, hogy minek az értékét akarja a program beolvasni. Ilyen emlékeztetőt más módon is kiírathatunk:

```
10 PRINT "X erteke= ";
20 INPUT X
30 PRINT "Y erteke= ";
40 INPUT Y
```

Ez egy kicsit hosszadalmasabb megoldás, de a hatása is más, igaz csak abban az apró részletben, hogy a kérdőjel, aminek a kiírását a gép az INPUT PROMT utasításnál elhagyta, most megjelenik.

A VÁLTOZÓK

A változók olyan jelképes nevek, amelyekkel különböző számokat jelölhetünk, ill. az általuk jelölt számokat meg is változtathatjuk. Ha az X változóval most éppen a 4000 számot jelöljük, akkor azt szoktuk mondani, hogy az X változó értéke 4000.

A változókat gyakran használjuk olyankor, ha olyan számmal szeretnénk számolni, aminek az értékét nem tudjuk előre. A könyv elején levő példaprogramok is használnak változókat, vagy mert egyes értékeket maga a program számít ki, vagy mert más számokat nekünk kell beírni akkor, amikor a program már fut. Változót használhatunk akkor is, ha egy hosszú számot akarunk többször használni egy programban, és nem akarjuk minden alkalommal kiírni, inkább csak hivatkozunk rá a nevével. A most következő program ismét új felhasználási lehetőséget mutat be.

```
10 LET A=0          ! Az A kezdoerteke 0
20 DO ! ciklus eleje
30    LET A=A+1 ! noveld eggyel az A-t
40    PRINT A, ! ird ki az A erteket
50 LOOP UNTIL A=20 ! ugorj a ciklus elejere
60          ! ha az A nem 20
70 END
```

Most az A a ciklus számlálója. A ciklus belsejében a gép mindig eggyel növeli az A értékét és kiírja azt. A ciklus végén megvizsgálja, hogy az A nem 20-e. Ha nem, akkor újra a ciklus elejére ugrik, ha igen akkor véget ér a program. A 40. sorban levő vessző a kiírt képet teszi szebbé, hatását könnyen érzékelhetjük, ha nélküle is lefuttatjuk a programot.

A VÁLTOZÓNEVEK

A változóknak igen sokféle nevet adhatuk, az X, Y, Z csak egy-egy példa. A nevek legfeljebb 31 karakter hosszúak lehetnek, a gép ilyen hosszú neveket tud megkülönböztetni. Valójában ilyen hosszú nevek használata már számunkra is kényelmetlen, általában nyolc-tíz karakter elegendő ahhoz, hogy a változó feladatát jól jelölő nevet alkossunk. A változóneveknek igen fontos feladata az is, hogy ha később javítani vagy módosítani akarjuk a programot, akkor könnyen felismerjük a nevekből az egyes változók feladatát. Két szám összegét tároló változót tehát célszerű mondjuk "Osszeg"-nek hívni, valaki nevét tároló változót pedig „Nev$"-nek. (A karaktersorozatokat tároló változókról majd később lesz szó.)

A gép számára mindegy, hogy egy változónevet kis- vagy nagybetűvel

írunk, tehát vigyázzunk, mert a "VÁLTOZÓ, változó, Változó" szavak mind ugyanazt a változót jelölik. A változónevekben nem minden karakter használható. A névnek mindig betűvel kell kezdődnie, és utána a névben is csak a betűk, a számok, az aláhúzás és a pont használható – eltekintve a karaktersorozatokat tároló változókat jelölő „$" jeltől. A

```
"Szam, A2$, OSSZEG$, Hello1, Resz.ossz.3, Nev, A1234"
```

nevek tehát helyesek, az

```
"En valtozom, 2A, !!!A$, <>3MOST, 3*szam"
```

nevek viszont nem.

A VÁLTOZÓK KEZDŐÉRTÉKE

A számlálóprogram első sorában az A kezdőértékére 0-t adtunk meg. Ha ezt a sort kihagytuk volna, akkor a program nem működne, mert amikor a 30-as sorban az A értékét kellene eggyel növelnie, nem tudná, hogy mihez képest növelje. Az olyan programokban azonban, ahol egy változó értékét INPUT utasítással olvassuk be, nem szükséges kezdőértéket megadni, csak arra kell vigyáznunk, hogy mielőtt beolvasnánk az értékét, ne tároljunk olyan értéket a változóban, amire később szükségünk lehet, mert ez az érték elvész.

A VÁLTOZÓK HASZNÁLATA

A változók működésének jobb megismeréséhez változtassuk meg a számlálóprogram első sorát „LET A = 1"-ről „LET A = 3"-ra. Futtassuk le így is a programot!

A változók alapvetően különböznek az első hasonlatként használt dobozoktól abban, hogy ha értéküket áttöltjük egy másik változóba, akkor a forrásváltozó sem veszíti el az értékét. Tegyük fel, hogy S = 0 és G = 10. Ha most kiadjuk a LET S = G parancsot, akkor a G értékét áttöltjük az S-be, azaz az S értéke 10 lesz, de a G sem veszíti el az értékét, az is megmarad 10 értékűnek. Ha a G-t nullázni akarjuk, akkor a LET G = 0 parancsot kell kiadnunk.

A LET kulcsszó után a számítógép mindig egy változónevet vár, amelynek az értéke vagy 0, vagy ha korábban már adtunk neki értéket, akkor valami más. Az értékadó utasítások elejére nem mindig kötelező kitenni a LET kulcsszót, de a gép ezt automatikusan beírja mindenhova, mivel ez megkönnyíti a programban levő ilyen utasítások megtalálását. Néhány kivételtől eltekintve a BASIC kulcsszavak is használhatók változónévként, de ilyenkor mindig ki kell tenni a LET szócskát, hogy a gép fel tudja ismerni az értékadást.

A következő program a változók használatának most ismertetett alapjait mutatja be néhány példán keresztül.

```
10  LET A$="A ket szam osszege="
20  !-----------------
30  ! A 10 sor az A$ kezdoerteket adja meg
40  !-----------------
50  LET Osszeg=0
60  !-----------------
70  ! Az Osszeg a lenti osszeadas
75  ! eredmenyet fogja tartalmazni
80  ! Kezdoerteke 0
90  !-----------------
100 INPUT PROMT"Gepeld be az elso szamot:":Szam1
110 INPUT PROMT"Gepeld be a masik szamot:":Szam2
120 !-----------------
130 ! A 100-as és a 110-es sor a ket osszeadando
135 ! szamot olvassa be
140 ! A LET szocskat nem kell hasznalnunk,
150 ! mert INPUT utasitassal adunk kezdoerteket
160 !-----------------
170 LET Osszeg=Szam1+Szam2
180 !-----------------
190 ! Az Osszeg nevu valtozo (ami eddig
195 ! nulla volt) felveszi a Szam1 es a
200 ! szam2 osszegenek erteket
210 !-----------------
220 PRINT A$;Osszeg
230 !-----------------
240 ! A 220-as sor hatasara a gep a 10-es sorban
245 ! levo szoveget es az Osszeg valtozo
250 ! erteket fogja kiirni egymas utan
260 ! ugyanabba a sorba
290 !-----------------
300 END
```

A MATEMATIKAI KIFEJEZÉSEK

A számítógép használata során igen gyakran kell különböző matematikai kifejezéseket használnunk. Ezekben mind számokat, mind változóneveket használhatunk az értékek jelölésére, az elvégzendő műveletek meghatározásá-

ra pedig úgynevezett operátorokat. Az operátoroknak két nagy csoportja van, a műveleti jelek és a relációs jelek. A műveleti jelek közűl négyet (+, -, *, /) már ismerünk, az ötödik a hatványozás jele, a ↑. A 2↑3 kifejezés jelentése tehát „kettő a harmadikon", más néven „kettő köbe", értéke ennek megfelelően 8.

A relációs jelek a következők:

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

Van ezen kívül még egy különleges operátor, a „=". Ezt összehasonlításban használva relációs jelnek számít, jelentése: egyenlő. A LET kulcsszó után azonban nem operátor, hanem az értékadás jele.

Lássunk példát a relációs operátorok használatára.

```
10   INPUT PROMT"Adj meg egy szamot:":A
20   !-------------------
30   ! A 10-es sor beolvassa a megvizsga-
35   ! lando szamot
40   !-------------------
50   IF A<0 THEN PRINT"Ez a szam negativ."
55   IF A=0 THEN PRINT"Ez nulla."
60   !-------------------
70   ! Az 50 sor egy egyszeru dontes, ha
75   ! az A negativ, akkor ezt kiirja.
80   ! Az IF (magyarul: ha) szocska utal
90   ! a vizsgalatra. Ha a vizsgalatot
100  ! elvegezve a feltetel teljesul, ak-
110  ! kor a sor hatralevo reszet vegre-
120  ! hajtja a gep, egyebkent nem. Az
130  ! "A<50" kiolvasva: A kisebb mint 50
140  !-------------------
150  IF A>0 AND A<50 THEN PRINT"Ez 0 es 50 koze esik."
160  !-------------------
165  ! Az IF/THEN szerkezet igen gyakori,
170  ! magyarul HA/AKKOR-nak olvashato.
180  ! HA a feltetel teljesul, AKKOR a sor
```

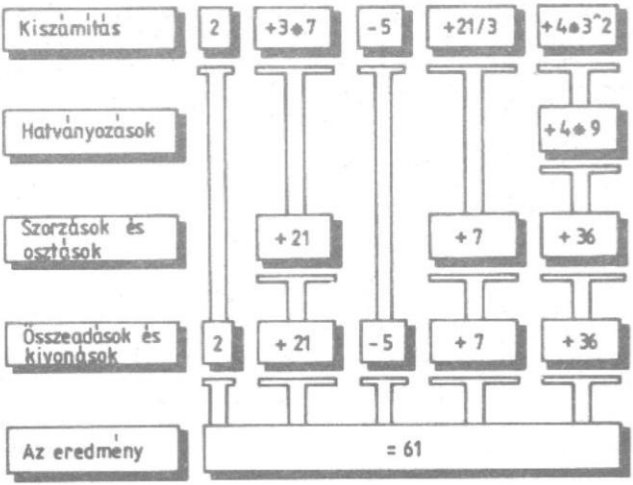
```
190 ! tobbi reszet is hajtsd vegre. HA az
195 ! A nagyobb mint Ø ES (angolul: AND)
200 ! az A kisebb mint 50 AKKOR...
210 !-------------------
220 IFA>50 ANDA<100 THENPRINT"Ez50 es100 kozottvan."
230 !-------------------
240 ! Az utolso osszehasonlitasnal a 300-as
245 ! sorban a gep megvizsgalja, hogy az
250 ! A nagyobb-e mint 100
290 !-------------------
300 IF A>100 THEN PRINT"Ez a szam nagyobb mint 100."
310 END
```

Arról, hogy a számítógép hogyan tud döntéseket hozni, később még részletesen is szó lesz, most csak az az érdekes, ahogy a relációs jeleket használja a számok és a változók összehasonlítására. Az ilyen relációs kifejezéseknek nincs szám jellegű értékük, csak „igen vagy nem" típusú választ adnak.

A MŰVELETI JELEK VÉGREHAJTÁSI SORRENDJE

Legyen példánk a 4 + 2*3-5 kifejezés. Mennyi ennek az értéke? Ha balról jobbra sorban elvégezzük a műveleteket, tizenhármat kapunk. A számítógép mégis azt válaszolná, hogy 5.

A gép a különböző műveleteket ugyanis nem azonos súllyal kezeli, hanem ún. prioritási sorrendben dolgozza fel. Először a hatványozásokat végzi el, mivel ezeknek a legmagasabb a prioritása, ezután a szorzásokat és az osztásokat, végül az összeadásokat és a kivonásokat. Az azonos prioritású műveletek között viszont már a balról- jobbra szabály szerint halad.



Nézzünk erre egy példát:

Legyen a kifejezés:2 + 3*7-5 + 21/3 + 4*3↑2

Először a hatványozás kerül sorra: 3↑2 = 9.

Most a kifejezés így néz ki:

2 + 3*7-5 + 21/3 + 4*9

Ezután a szorzásokat és az osztásokat végzi el, sorban haladva balról jobbra:

3*7 = 21 21/3 = 7 4*9 = 36

A kifejezés újabb alakja: 2 + 21-5 + 7 + 36

Végül ezeket a műveleteket is elvégzi:

2 + 21 = 23 23-5 = 18 18 + 7 = 25 25 + 36 = 61

A kifejezés értéke tehát 61

Amint láttuk, a műveletek prioritási sorrendje jelentősen befolyásolja a gép számításainak az eredményét. Ha meg akarjuk változtatni a műveletvégzési sorrendet, akkor hasonlóan a megszokotthoz, zárójeleket kell használnunk. Figyeljünk azonban arra, hogy a zárójeleken belül már újra érvényes a prioritási sorrend. Rövid példaként nézzük meg a következő kifejezés kiszámítását:

```
(2+3)*7/(5*2↑3-5)
        5*7/(5* 8 -5)
        5*7/( 40   -5)
        5*7/ 35
        35 / 35
           1
```

azaz ennek a kifejezésnek az értéke 1.

A zárójeleken belül természetesen további zárójel-párokat is használhatunk, melyekben megint továbbiakat stb. Próbáljunk ki más kifejezéseket is, ezeket legkönnyebben parancsmódban a PRINT kulcsszó után írva számíttathatjuk ki. A kifejezésekben törtszámok is előfordulhatnak, de ilyenkor vessző helyett ponttal kell elválasztanunk az egész és a törtrészt. Ha előjeles számot használunk, akkor azt leggyakrabban zárójelbe kell tennünk azért, hogy a gép meg tudja különböztetni az előjelet a műveleti jelektől, pl.:

```
-1*(0.5*(-6.5)) = 3.25
```

A KIÍRÁSOK FORMÁZÁSA

A PRINT utasítások lezárására többféle jelet használhatunk. Ha semmit sem írunk a végére, akkor az utasítás szerinti szám(ok), illetve karaktersorozat(ok) kiírása után a gép a következő sor elejére áll és a következő PRINT utasítás tartalmát ott kezdi kiírni. Ha egy PRINT utasításba semmit sem írunk, azaz „üres" PRINT-et adunk ki, akkor ezzel egy üres sort hagyhatunk ki, feltéve, hogy éppen egy új sor elején állunk. Az utasítások végére pontosvesszőt téve azonban egymás után ugyanabba a sorba irathatunk ki.

```
10 PRINT"Szervusz! "
20 PRINT"En "
30 PRINT"vagyok "
40 PRINT"a "
50 PRINT"szamitogep "
```

Ennek a programnak a hatására a következő jelenik meg a képernyőn:

```
Szervusz!
En
vagyok
a
szamitogep
```

Ha viszont a sorok végére egy-egy pontosvesszőt teszünk, akkor ez:

```
Szervusz! En vagyok a szamitogep
```

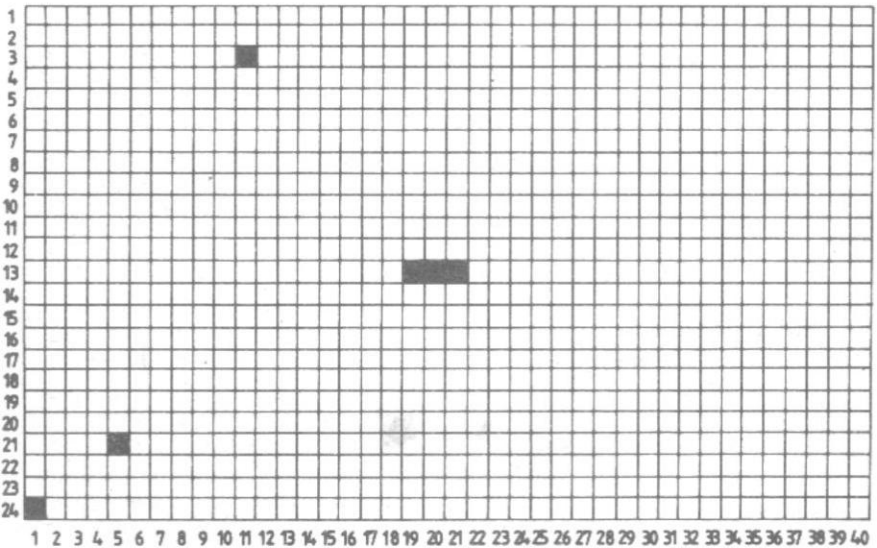
Egészítsük ki a programot a következő sorral:

```
15 PRINT
```

Ez egy üres PRINT, mégse kapunk üres sort, mert a 10-es sorban levő PRINT utasítás végén álló pontosvessző hatására a gép még a „Szervusz" szó sorában állt, amikor kiadtuk az üres PRINT-et a 15-ös sorban. Ennek hatására tehát csak a következő sor elejére állunk. Az üres PRINT teljesen úgy működik, mintha egy elektromos írógépen leütnénk a „kocsivissza – soremelés" gombot. Ha a 17-es sorba is írunk egy üres PRINT-et, akkor már tényleg kapunk egy üres sort.

```
10  LET A=0
20  DO
30    LET A=A+1
40    PRINT A,
50    !-----------------
60    ! A PRINT vegen allo vesszo hata-
65    ! sara a gep a kepernyo kovetkezo
70    ! oszlopanak elejere all
80    !-----------------
90  LOOP UNTIL A=20 ! ismetles, amig A nem lesz 20
100 END
```

A 40 karakter széles képernyőt a gép nyolc karakter széles oszlopokra osztja fel, és a vessző hatására mindig a legközelebbi következő oszlop elejére áll. Ezzel a módszerrel könnyen elválaszthatjuk egymástól a számokat (pontosvesszőt használva ugyanis olvashatatlanul összefolynának – próbáljuk ki), ugyanakkor sokkal kevesebb sort kell elhasználnunk, mintha minden számot új sorba íratnánk.



A „PRINT AT" UTASÍTÁS

A képernyőre a gép az ábrának megfelelően 24 sorban és 40 oszlopban tud karaktereket kiírni. Minden ilyen ún. karakterpozícióra az ábrán látható sor- és oszlopszámának a megadásával tudunk hivatkozni. A PRINT utasításban az AT kulcsszót követően egymástól vesszővel elválasztva adhatjuk meg annak a karakterpozíciónak a sor-, majd az oszlopszámát, ahonnan kezdve a képernyőre akarjuk írni a karaktersorozatot. A kiírandó szöveget kettősponttal kell elválasztanunk a pozíciószámoktól. Az ilyen PRINT utasítás végén ugyanolyan jelek állhatnak és ezek hatása is ugyanaz, mint a korábbiak. Próbaképpen futtassuk le a következő programot:

```
10 CLEAR SCREEN
20 PRINT AT 1,10:"Ez a teteje"
30 PRINT AT 12,10:"Ez a kozepe"
40 PRINT AT 24,10:"Ez az alja"
```

```
50 PRINT AT 2,1:;
60 END
```

Próbáljuk ki az 50. sor nélkül is!

A 80 KARAKTERES KÉPERNYŐ

A „TEXT 80" parancs hatására a képernyőn nem 40, hanem 80 oszlop jelenik meg, az egyes karakterek azonban csak fele olyan szélesek lesznek, éppen ezért csak igen rosszul olvashatók. Ez a parancs egyidejűleg törli is a képernyőt. A 40 karakter széles képernyőre a „TEXT 40" paranccsal térhetünk vissza.

A PROGRAMSZÖVEG SZERKESZTÉSE

Az egyszerű gépelési hibák kijavítására már megismertünk egy módszert, de a program szövegének jelentősebb módosítása ezzel nagyon bonyolult lenne. A gép több szövegszerkesztési módszert is kínál számunkra, ezzel is megkönnyítve a programok megváltoztatását.

AZ ÚJRASZÁMOZÁS

A korábbi példaprogramokban általában tizesével számoztuk a sorokat és nem egyesével. Ez nem véletlen, ugyanis csak így van módunk arra, hogy a korábbi sorok közé újakat illesszünk be. Ha azonban a 10-es és a 20-as sor közé már mind a kilenc lehetséges új sort (11, 12,... 18, 19) beszúrtuk, akkor más módszerrel kell a további soroknak helyet csinálni. Ilyenkor kell használnunk a RENUMBER (magyarul újraszámozás) parancsot. Ennek a parancsnak a hatására a gép átsorszámozza a sorokat, mégpedig száztól kezdődően tizesével. A szolgáltatás a SHIFT + F3 billentyűkkel is elvégezhető. Az átsorszámozás kezdőértékét az AT kulcsszó után adhatjuk meg, még a parancson belül. Ehhez hasonlóan megadhatjuk a sorszámozás lépésközét is, a STEP kulcsszó után írva, de a RENUMBER parancson belül. A programot tehát például a RENUMBER AT 150 STEP 3 utasítással úgy sorszámoztatjuk újra, hogy az első sor sorszáma a 150 lesz, a lépésköz pedig 3.

A későbbiek során találkozni fogunk olyan utasításokkal is, amelyek nem a programszövegre vonatkozó parancsok, hanem a program szövegében vannak, és mégis valamely sor sorszámára hivatkoznak. A RENUMBER parancs az ilyen utasításokban átírja a hivatkozott sorszámot az új megfelelőre, így nekünk ezzel már nem kell törődnünk.

A PARANCSOK PARAMÉTEREI

A programok szövegére vonatkozó parancsok legtöbbjének vannak paraméterei, azaz bizonyos számok, amelyek a parancs működésére jellemzők. A

RENUMBER parancsnál ilyen a kezdősorszám és a lépésköz. Ezeket minden olyan parancsban, ahol egyáltalán értelme van és lehetséges, az AT és a STEP kulcsszavak után adhatjuk meg.

Egyes parancsok hatását korlátozni lehet a programnak egy részére. Ilyenkor általában a program sorszámaira kell hivatkoznunk és a korlátozást közvetlenül a parancsszó után kell írnunk. Ezt a lehetőséget a már korábban megismert DELETE parancs példáján ismerjük meg.

A parancs hatását korlátozhatjuk egyetlen sorra, ha a sor számát írjuk a kulcsszó után. Tehát a DELETE 100 parancs csak a 100-as sort törli. Ha a program egy szakaszát akarjuk törölni, akkor a legelső és a legutolsó törlendő sor számát kell megadnunk, először a kisebbiket, majd a TO kulcsszóval tőle elválasztva a nagyobbikat. DELETE 100 TO 200 törli a 100. és a 200. sor közötti programsorokat.

Az ilyen tartománymegadásnál a proram legelső sorára hivatkozhatunk a FIRST szóval is, ez kényelmesebb és egyszerűbb, mintha előtte meg kellene néznünk a sorszámát. „DELETE FIRST TO 100" a 100-as sorig törli a programot. Az előzőhöz hasonlóan a program legutolsó sorára is hivatkozhatunk a LAST kulcsszóval. DELETE FIRST TO LAST az egész programot törli, mintha csak a NEW parancsot adtuk volna ki.

Az általunk ismert parancsok közül a LIST és a RENUMBER parancsban adható meg még a hatókör. Természetesen valamennyi ilyen parancs paraméterének van alapértelmezése is, ha tehát paraméterek nélkül adunk ki egy parancsot, akkor azt a gép alapértelmezési értékekkel hajtja végre. Például a LIST parancs alapértelmezése: LIST FIRST TO LAST.

Az AUTO PARANCS

A programsorok begépelését meggyorsítja az AUTO parancs, hatására a gép a kezdősorszámtól kezdve a lépésköznek megfelelő értékkel növekvő sorszámokat ír ki, amelyek után nekünk csak a megfelelő utasításokat kell begépelnünk. A következő sorszámot mindig az ENTER hatására írja ki a gép. A szolgáltatást vagy a STOP billentyűvel, vagy egy üres sor elküldésével tudjuk kikapcsolni.

A JAVÍTÁSOK VÉGLEGESÍTÉSE

Ha a program képernyőre listázott szövegét módosítjuk, akkor a gép először csak a képernyőn végzi el a változtatásokat. A valódi módosítás csak akkor következik be, amikor a megváltozott sort elküldjük az ENTER billentyűvel. Ehhez nem kell a sor végére mennünk, azt a programsoron – amely esetenként több képernyősorra is kiterjedhet – belül bárhol megtehetjük. Javítások után azonban soha ne felejtsük el ENTER-rel elküldeni a sort, mert különben a gép nem veszi tudomásul a módosításunkat.

A BESZÚRÁS ÜZEMMÓD

Az eddigiek során mindig azt tapasztaltuk, hogy a begépelt karakterek a kurzor helyén jelentek meg és a korábban ott levőket eltüntették, mintegy felülírták. A gép lehetőséget nyújt arra is, hogy a meglevő betűket ne veszítsük el, hanem a sor kurzortól jobbra eső része mindig úgy csússzon el jobbra, hogy az új karakterek is a régiek közé beszúrva jelenhessenek meg. Ezt a szolgáltatást a CTRL és az INS billentyűk egyidejű lenyomásával tudjuk be-, ill. kikapcsolni.

Természetesen előfordulhat az is, hogy a jobbra mozduló karakterek nem férnek el a képernyőn és eltünnek annak jobb oldali szélén. Ilyenkor ezek nem vesznek el, a program újralistázásakor rendesen megjelennek, addig pedig a sor szélén „>" jel figyelmeztet bennünket arra, hogy nem látható az egész sor. Ha mégis látni akarnánk, akkor a CTRL és az F1 billentyűket együtt lenyomva, újrarendeztethetjük a szöveget a képernyőn.

A KURZOR MOZGATÁSA

Azt már megszokhattuk, hogy a kurzort a beépített botkormánnyal irányíthatjuk. Most csupán arra a szolgáltatásra szeretnénk felhívni a figyelmet, hogy a képernyő felső vagy alsó szélét elérve és a kurzort tovább mozgatva a kép soronként lépni kezd, és a korábban eltűnt sorok újra láthatóvá válnak. Egy hosszú program listáját tehát mintegy mozgatható ablakon keresztül tudjuk megnézni.

Ha fel-le lépegetve a botkormánnyal egy időben lenyomjuk a SHIFT gombot is, akkor nem egy sort, hanem egy teljes képernyőnyit tudunk lépni a kívánt irányba. A SHIFT helyett a CTRL billentyűt használva egy-egy bekezdésnyi (programok listáiban: utasításnyi) szöveget ugorhatunk át egyetlen lépéssel. Az oldalirányú mozgásnál a CTRL billentyűvel együtt használva a botkormányt szavanként, a SHIFT-tel együtt pedig a sor elejére, ill. végére tudunk lépni.

A következő két táblázat a szövegtörlés, ill. beszúrás lehetőségeit foglalja össze.

FELADAT	BILLENTYŰ	HATÁS
Bal oldali karakter törlése: A kurzortól balra levő karaktert törli és a sort összevonja	ERASE	A kurzort egy pozícióval balra mozdítja, ha a sor elején áll, akkor a sort az előző sorhoz fűzi
Bal oldali sorrész törlése: A sor elejétől a kurzorig törli a sort, a maradékot a sor elejére viszi	SHIFT + ERASE	A kurzor a sor bal szélére kerül
A jobb oldali karakter törlése: A kurzor alatti karaktert törli és a sor jobb oldali részét balra lépteti	DEL	A kurzor nem mozdul. Ha a sor végén áll, akkor a következő sort ehhez csatolja
A jobb oldali sorrész törlése: A kurzortól a sor végéig mindent töröl	SHIFT + DEL	A kurzor nem mozdul
A bal oldali szó törlése: A kurzortól balra levő szó törlése	CTRL + ERASE	A kurzortól balra levő pozíciónál kezd törölni. Csak a betűket és a számokat tekinti a szavak részének, az ezektől eltérő karaktereket nem veszi figyelembe
A jobb oldali szó törlése: A kurzortól jobbra levő szót törli	CTRL + DEL	A kurzor alatti pozíciótól indul, egyebekben az előzővel megegyezik
A karakter beszúrása: A kurzortól jobbra levő karaktereket eggyel jobbra tolva üres helyet hagy a kurzornak	INS	Beszúrás üzemmódban a szóköz billentyűvel azonos hatású, csak a kurzort nem mozgatja
Sor beszúrása: A kurzortól jobbra levő karaktereket egy sorral lejjebb lépteti	SHIFT + INS	Ha a kurzor sor elején áll, akkor egy üres sort állít elő
Beszúrás-üzemmód ki/be: A beszúrás- és felülírás-üzemmód között kapcsol át	CTRL + INS	Másfajta kurzor jelenik meg az üzemmódtól függően

A SZÖVEGSZERKESZTŐ

A gép egyik leghasznosabb szolgáltatása, amely lehetőséget nyújt különböző szövegek – levelek, tanulmányok – legépelésére és ha sornyomtató is rendelkezésünkre áll, akkor a kinyomtatására is. A számítógép előnye az írógéppel szemben, hogy egyszerű javítási és sokféle speciális szövegformázási lehetőséget kínál.

Az Enterprise beépített szövegszerkesztője akkor is használható, ha semmilyen programtárat sem csatlakoztatunk a géphez, ilyenkor bekapcsoláskor rögtön a szövegszerkesztő jelentkezik. A BASIC rendszerből a TYPE paranccsal vagy a SHIFT + F8 billentyűvel léphetünk a szövegszerkesztőbe, ilyenkor a gépben tárolt BASIC programok törlődnek, a képernyő alsó és felső részén pedig a funkcióbillentyűk hatását ismertető címkék tűnnek fel.

Figyelem! Ha korábban megváltoztattuk volna a gép karakterkészletét, akkor mielőtt a szövegszerkesztőbe lépnénk, állítsuk vissza az eredetit!

A szövegszerkesztő nem értelmezi a begépelt szöveget, tehát sem hibaüzeneteket nem kapunk, sem a megszokott szövegrendezési parancsokat nem használhatjuk. Az ENTER csak az írógépeken szokásos „kocsivissza – soremelés" műveletét végzi, de erre a sorok végén nincs szükség. A szövegszerkesztőbe beépített szolgáltatás ugyanis azokat a szavakat, amelyek a sor végén túlnyúlnának, automatikusan átmásolja a következő sor elejére (angol jelölése: word wrap). Az ENTER nem a sorok, hanem a bekezdések végét jelzi. Egyes szövegformázási szolgáltatások ugyanis egy-egy teljes bekezdésre vonatkoznak.

A szöveg módosítására minden, a BASIC rendszerben megszokott szolgáltatás használható. A következőkben ismertetendő szolgáltatások a szövegszerkesztőn kívül is működnek, de funkciójuk alapján elsősorban ide tartoznak:

SPECIÁLIS SZOLGÁLTATÁSOK

Ezeket a szolgáltatásokat a funkcióbillentyűkkel és a CTRL, ill. az ALT billentyűkkel működtethetjük.

CTRL + F1 – újraformázás

A bekezdés újra eligazítása, elsősorban szavak törlése, ill. beszúrása után, ha a sorhosszak nagyon különbözők.

ALT + F1 – kiegyenlítés és újraformázás

A kiegyenlítés a szavak olyan elrendezése a bekezdés soraiban, hogy a sorok eleje és vége függőleges oszlopot alkosson az újságok hasábjaihoz hasonlóan.

CTRL + F2 – középre állítás

A szövegsor középre állítása, ami elsősorban címsorokban, fejlécekben hasznos.

ALT + F2 – a tabulátorok törlése

Az írógéphez hasonlóan beállítható tabulátorpozíciók törlése, az újak beállítása előtt.

CTRL + F3 – tabulátor beállítás/törlés

A kurzor oszlopába tabulátorpozíció beállítása, vagy az ott levő törlése.

ALT + F3 – vonalzósor ki/be

A vonalzósor megjelenítésének ki- és bekapcsolása a képernyő tetején, amely a bal, ill. a jobb margó helyét és a tabulátorpozíciókat jelölő függőleges vonalkákat tartalmazza.

CTRL + F4 – a bal margó beállítása

A kurzor oszlopába állítja a bal margót.

ALT + F4 – a jobb margó beállítása

A kurzor oszlopába állítja a jobb margót.

CTRL + F5 – a margó felfüggesztése

A margók ideiglenes törlése, ezáltal írni lehet a margókra is. Bekapcsolását csillag jelzi a vonalzósorban, ismételt használatával a margók visszaállíthatók.

ALT + F5 – a margók törlése

A margókat a lap bal, ill. jobb széléhez illeszti, de úgy, hogy később újra beljebb állíthatók legyenek. A tabulátorokat is törli.

CTRL + F6 – felfelé léptetés

A kurzort tartalmazó bekezdés egy sorral előbbre helyezése úgy, hogy az ott lévő sor mögéje kerül. A billentyű lenyomva tartásával a szolgaltatás addig ismételtethető, amig a bekezdés a megfelelő helyre nem kerül.

ALT + F6 – lefelé léptetés

Az előzőhöz hasonló, csak lefelé irányul.

CTRL + F7 – a sor színének változtatása

A kurzor sorában a papír és a tinta színének megváltoztatása, 40 karakteres soroknál két, 80 karakteresnél négy variáció szerint. A billentyű lenyomva tartásával a sor színe oda-vissza változik.

ALT + F7 – a bekezdés színének változtatása

Az előzőhöz hasonló, csak az egész bekezdésre vonatkozik.
A szövegszerkesztőben a funkcióbillentyűk kiegészítő billentyű nélküli hatása is megváltozik. Egyes szolgáltatások – mint például a 40, ill. 80 karakteres képernyő váltása, ami a szöveget is törli – védettek a véletlen parancskiadás elkerülésére. Ilyenkor a számitógép visszakérdez, és csak az ENTER leütésére végzi el a módosítást. A helytelen parancsot az ESC billentyűvel törölhetjük.

A SZÖVEG KINYOMTATÁSA

A nyomtatáshoz csatlakoztatni kell a nyomtatót a gép hátulján levő, PRINTER feliratú aljzatba. Bekapcsolás és az ON LINE állapot elérése után üssük le az F3 billentyűt. A gép még megkérdezi a nyomtató ún. címét, erre elegendő ENTER-rel válaszolnunk.

A SZÖVEG KAZETTÁRA MENTÉSE

A szöveg kazettára mentéséhez a kazettás magnetofon kezelését tárgyaló fejezetben leírtak szerint kell csatlakoztatni a magnetofont. Ha az F2 billentyűvel jelezzük a gépnek, hogy menteni akarjuk a szöveget, a számítógép megkérdezi, hogy milyen néven mentse. A megszokottól eltérően most nem kell idézőjelek közé tenni a karaktersorozatot, csupán a később, a programok mentésénél ismertetendő szabályokat kell betartanunk. A magnetofon indítása után küldjük el a nevet, mire a mentés rögtön megkezdődik.

Az így mentett szövegeket természetesen újra be is olvashatjuk a szövegszerkesztőbe. Ilyenkor a programok betöltéséhez hasonlóan kell csatlakoztatnunk a magnetofont. Az F1 billentyű lenyomása után a számítógép a betöltendő szöveg nevét kérdezi meg. Ha erre csak ENTER-t ütünk, akkor a szalagon talált első szöveget tölti be.

KILÉPÉS A SZÖVEGSZERKESZTŐBŐL

A munka befejezése után az F8 billentyű lenyomásával térhetünk vissza a BASIC rendszerbe. A gép ilyenkor még megkérdezi, hogy biztosan befejeztük-e a feladatot. Ha erre ENTER-rel vagy a BASIC szóval válaszolunk, akkor kilépünk a szövegszerkesztőből és a benne levő szöveg törlődik. Ha a WP szócskát adjuk meg, akkor újra a szövegszerkesztőbe kerülünk, de a korábbi szöveg törlődik. Ha pedig az ESC billentyűt ütjük le, akkor a korábbi szöveghez térünk vissza.

A FUNKCIÓBILLENTYŰK

A funkcióbillentyűk a gyakran használt karaktersorozatok begépelését gyorsítják meg. Ezek a szövegek lehetnek parancsok, de lehetnek akár értelmetlen betűsorozatok is. A gép bekapcsolásakor minden funkcióbillentyűnek négy jelentése van aszerint, hogy önmagában vagy a SHIFT, a CTRL, ill. az ALT billentyűvel együtt ütjük le. Ezeket a funkciókat azonban meg lehet változtatni.

A következő parancs az F1 funkcióbillentyű jelentését változtatja meg, a parancs kiadása után a billentyű leütésekor a számítógép az idézőjelek közötti karatersorozatot értelmezi.

```
SET FKEY 1"Mostantol ezt jelenti az F1 billentyu"
```

A különböző parancsok kiadását is megkönnyítheti, ha ez egyetlen billentyű lenyomásával elvégezhető. Ilyenkor nagyon kényelmes, ha a parancs elküldését is automatikusan elvégzi a gép a funkcióbillentyű leütésekor.

Az ENTER-t helyettesítő ún. „kocsivissza" (carriage return) karakter a következőképpen írható a karaktersorozat végére: & CHR$(13)

A PRINT parancs a F3 funkióbillentyű leütésével a következő lesz:

```
SET FKEY 3"PRINT" & CHR$(13)
```

A következő két táblázat összefoglalja a különböző funkcióbillentyűk alapértelmezéseit. Ügyeljünk arra, hogy az önmagukban leütött billentyűk hatása a szövegszerkesztőben megváltozik, ezeket a második táblázatban találhatjuk meg.

Funkciók csak BASIC-ben — Funkciók BASIC-ben és a szövegszerkesztőben is

Billentyű	NORMAL	+ SHIFT	+ CTRL	+ ALT
1	START Lefuttatja a programot. Ha nincs program a számítógépben, akkor előbb betölti a lemezről, vagy ha ez nincs, kazettáról	CONTINUE Folytatja a STOP-pal megszakított programot	RE-FORM A magnóknak megfelelően elrendezi, szerkesztés után pedig újrarendezi a bekezdés sorait	JUSTIFY AND REFORM Egyenesre igazítja a sorok széleit
2	LIST Listázza az egész programot a képernyőre	LLIST Listázza az egész programot a sor-nyomtatóra	CENTRE A sor közepére igazítja a sorban levő szöveget	CLEAR ALL TABS Törli a tabulátor-pozíciókat
3	AUTO Automatikusan sorszámoz tízesével, a STOP-pal állítható le	RENUMBER Újra sorszámozza az egész programot tízesével	TAB SET/CLEAR Beállít vagy töröl egy tabulátorpozíciót a kurzor oszlopában	RULER LINE Be/kikapcsolja a vonalzósor kirajzolását
4	REMOTE 1 Be/kikapcsolja a „rem 1" vezérlő kimenetet	REMOTE 2 Be/kikapcsolja a „rem 2" vezérlő kimenetet	LEFT MARGIN A bal oldali magnót a kurzorhoz állítja	RIGHT MARGIN A jobb oldali magnót kurzorhoz állítja
5	TEXT A teljes képernyőt szöveges üzemmódba állítja, és törli a korábbi lapokat	DISPLAY TEXT Szöveges üzemmódra kapcsol anélkül, hogy törölne	RELEASE MARGINS Felfüggeszti a margók használatát	RESET MARGINS A margókat szét-húzza a képernyő széleihez
6	GRAPHICS Az első húsz sort grafikus, az utolsó négyet szöveges üzemmódba kapcsolja és törli a korábbi lapokat	DISPLAY GRAPHICS Átkapcsol grafikus üzemmódba anélkül, hogy törölne	MOVE UP Egy sorral feljebb lépteti a bekezdést	MOVE DOWN Egy sorral lejjebb lépteti a bekezdést
7	CLICK Ki/bekapcsolja a billentyűk leütését jelző hangot	SPEAKER Ki/bekapcsolja az összes hangjelzést	CHANGE LINE COLOUR A sor színét változtatja	CHANGE PARAGRAPH COLOUR A bekezdés színét változtatja
8	INFO A gépben levő programokról nyújt információt	TYPE Átlép a szövegszerkesztőbe		

Normál funkciók a szövegszerkesztőben

1. LOAD Betölti a kazettán vagy lemezen levő szöveget	2. SAVE Kimenti a szöveget a kazettára vagy a lemezre	3. PRINT Kiírja a szöveget a sornyomtatón	4. HELP Német vagy angol nyelvű listát ír ki a szövegszerkesztő speciális szolgáltatásairól, az ENTER-rel kapcsolható vissza
5. TEXT 80 Törli a szöveget és átkapcsol 80 oszlopos üzemmódba	6. TEXT 40 Törli a szöveget és átkapcsol 40 oszlopos üzemmódba	7. CLICK Ki/bekapcsolja a billentyűk leütését jelző hangot	8. EXIT Visszatér a BASIC rendszerbe

A PROGRAMOK TÁROLÁSA KAZETTÁN

Gyakran előfordul, hogy egy programunkat szeretnénk máskor is használni vagy másoknak megmutatni. Ha azonban kikapcsoljuk a gépet, akkor elfelejti az összes benne levő programot, és az egészet legközelebb újra be kell gépelnünk. Hogy ezt elkerüljük, programjainkat, a gyáriakhoz hasonlóan, kazettára menthetjük.

Ehhez először a kazettás magnetofont kell megfelelően csatlakoztatnunk. A még szabad csatlakozók közül a nagyobbikkal kössük össze a számítógép OUT jelű kimenetét a magnetofon MIC jelű bemenetével. Ha a magnó távvezérelhető, akkor a kisebbik csatlakozóval a gép REM2 kimenetét csatlakoztassuk a magnetofon vezérlő (REM) bemenetéhez, és a SHIFT + F4 billentyűkkel kapcsoljuk be a távvezérlést. Természetesen a magnetofon csatlakoztatásához nem kell kikapcsolni a számítógépet.

Ezután meg kell adnunk, hogy milyen néven mentse a gép a programot. A program neve legfeljebb 28 karakterből állhat, de nem tartalmazhat szóközt, csak betűket, számokat és egyes írásjeleket, mint pl. a pont, a kötőjel, az aláhúzás és a per jel. A program nevét a SAVE paranccsal adjuk meg:

```
SAVE "Program-1"
```

Mielőtt a parancsot elküldenénk, indítsuk el a magnetofonon a felvételt, az ENTER leütése után a számítógép ugyanis azonnal megkezdi a program mentését. Erről a

```
SAVING Program-1
```

üzenettel értesít, amelyben mindig az éppen mentett program nevét írja ki. A mentés befejeztével az OK üzenetet írja ki, ezután leállíthatjuk a magnetofont. A távvezérelhető magnetofonokat a számítógép maga indítja el és állítja le.

A mentés során a szalagra kerülő jelek halkan a számítógép hangszórójából is hallhatók, ami igen hasznos a felvétel figyelemmel kíséréséhez.

A FELVÉTEL ELLENŐRZÉSE

Az elkészült felvétel hibátlanságát célszerű ellenőrizni. Ehhez tekercseljük vissza a szalagot oda, ahol a felvétel kezdődik, és csatlakoztassuk a magnetofont úgy, ahogy a beolvasáshoz kell. Ezúttal azonban nem a LOAD, hanem a VERIFY parancsot kell használni.

```
VERIFY "Program-1"
```

Indítsuk el a magnót és várjuk meg, amig a gép ellenőrzi a felvételt. Ha nincs hiba, ezt az OK üzenettel jelzi, ha azonban hibát talál, akkor erről megfelelő üzenetet küld. Ilyenkor ellenőrizzük, hogy helyesen csatlakoztattuk-e a gépet a felvételhez, és megfelelő felvételszintet állítottunk-e be a magnetofonon, majd próbáljuk meg újra felvenni a programot.

PROGRAMOK ÖSSZEMÁSOLÁSA

Ha egy kazettán levő programot szeretnénk egy, a tárban levővel összemásolni, akkor ezt a MERGE paranccsal tehetjük. A MERGE parancs hasonló a LOAD-hoz, a magnetofon csatlakoztatása is ugyanolyan.

A különbség, hogy a LOAD betöltés előtt törli a már a tárban levő programot, a MERGE pedig nem. Vigyázzunk azonban, mert ha az új programban egy sor sorszáma megegyezik a régi program valamelyik soráéval, akkor az új sor felülírja a régit, hasonlóan ahhoz, mintha a billentyűzetről írnánk be az újat. Az összemásolás előtt tehát sorszámoztassuk át a régi program sorait úgy, hogy az újonnan beérkező sorokkal ne keveredjenek össze és ne írják egymást felül.

A betöltés során a magnetofont ugyanúgy kell kezelnünk, ahogy a LOAD esetében. A MERGE parancs alakja a következő:

```
MERGE "Ujprogram"
```

ahol „Ujprogram" az újonnan betöltendő program neve. Próbáljuk ki a programok tárolására és visszaolvasására vonatkozó parancsokat, és ne veszítsük el kedvünket az esetleges hibák esetén. Próbálkozzunk újra, amíg rá nem jövünk, hogy mit hibáztunk el.

A FILE-OK

A file angol szó, a számítástechnikában a szalagra vagy lemezre mentett információk egy-egy csomagját jelöli. A file, aszerint, hogy mit tartalmaz, lehet program-file vagy adat-file.

A LEMEZEGYSÉG HASZNÁLATA

A lemezegységet a csatlakoztatás után ugyanúgy használhatjuk, mint a magnetofont, ugyanazokat a parancsokat adhatjuk ki, csak ezek végrehajtása sokkal gyorsabb. Ha a számítógéphez lemezegységet is csatlakoztattunk, akkor automatikusan azt használja a magnetofon helyett.

3. RÉSZLETES ISMERTETŐ

A KARAKTERLÁNCOK

Az előző fejezetekben a karakterláncokkal csak igen nagyvonalúan foglalkoztunk. Most, hogy már rendelkezünk alapvető ismeretekkel, ideje, hogy részletesen megismerkedjünk a karakterek kezelésével.

A karakterlánc a számon kívül a másik fő adattipus, amelyet a számítógép kezelni tud. Abból a tényből, hogy a számítógép nem tud gondolkodni, az következik, hogy pontosan azt fogja tenni, amit használója mond neki az adott parancsban, ill. adattal. Tehát a programozás alapvetően adatok rendezgetése, átcsoportosítása, módosítása.

Az előzőkben már megfigyelhettük, hogy ha azt akarjuk, hogy gépünk ismételje meg azt, amit mondtunk neki, akkor a szöveget idézőjelek közé kellett tennünk. A gép ezt a szöveget módosítás nélkül, betűről betűre írja vissza, így nem szükséges, hogy bármilyen értelme legyen. A gép az idézőjelek között lévő szöveget karakterláncként, jelentés nélküli szimbólumok sorozataként kezeli.

Lássunk néhány karakterláncot :

```
"Hány éves vagy ?"
"Most múltam 24."
```

Figyeljük meg, hogy a karakterláncban nemcsak betük, hanem különféle írásjelek, sőt számok is lehetnek.

Próbáljuk ki a következőket!

```
PRINT 2*2
```

Ez a sor arra utasítja számítógépünket, hogy végezzen el egy számtani műveletet: 2 értékét szorozza meg 2 értékével, és a kapott eredményt írja ki a képernyőre - másképp fogalmazva, végezze el a szorzást, és írja ki a végeredményt.

A következő példa első látásra szinte teljesen megegyezik az előzővel:

```
PRINT "2*2"
```

Itt azonban a 2*2 idézőjelek közé került, és így karakterlánc vált belőle. Most tehát arra utasítottuk a számítógépet, hogy írja ki a képernyőre azt, hogy 2, majd, hogy *, végül pedig ismét azt, hogy 2.

Számítógépünk képes arra, hogy megadja egy karakterlánc hosszát, ha felkérjük erre. Próbáljuk ki a következő programot:

```
100   LET A$ = "COMPUTER"
110   LET B$ = "MICROCOMPUTER"
120   !-----------------
130   ! A fenti ket sor ket karak-
140   ! terlanc-valtozot deklaral.
150   ! Ezek hasonloak a numerikus
160   ! valtozokhoz azzal a kulonb-
170   ! seggel, hogy ok karakterek egy
180   ! csoportjanak, nem pedig
190   ! szamertekeknek nevei. A karak-
200   ! terlanc-valtozok nevenek
210   ! vegen mindig egy $ jelnek
220   ! kell allnia
230   !-----------------
240   LET A = LEN( A$ )
250   LET B = LEN( B$ )
260   !-----------------
270   ! A és B numerikus valtozok,
280   ! melyek ket karakterlanc
290   ! hosszat ( az altala tartal-
300   ! mazott karakterek szamat )
310   ! jelzik. Ezt az erteket a
320   ! LEN fuggveny adta meg
330   !-----------------
340   PRINT "Az A$ nevu karakterlanc";
          A;" karakter hosszusagu."
350   PRINT "A B$ nevu karakterlanc";
          B;" karakter hosszusagu."
360   END
```

Próbálkozzon meg A$ és B$ tartalmának megváltoztatásával – a számítógép mindig a helyes hosszúságot adja meg.

SZÓKÖZÖK

Érthető lenne, ha azt gondolnánk, hogy egy szóköz semmit sem jelent, nem számít, hogy van valahol, vagy tetszőlegesen elhagytuk. De próbáljuk csak az előbbi programban megváltoztatni A$ - t és B$ - t

" " - re, ill. " " - re.

Megfigyelhetjük, hogy a számítógép pontosan kiírja, hogy melyik karakterlánc hány szóközt tartalmaz, noha mindkettő üresnek tűnik.

Egy valóban üres karakterlánc a következő:

" "

Amint látható, ez még szóközöket sem tartalmaz, ezért null-karakterláncnak, (null-stringnek) is nevezik.

KARAKTERLÁNCOK KARAKTERLÁNCOK BELSEJÉBEN

A számítógép több müveletet végezhet a karakterláncokon. Így például készíthet karakterláncot egy másik felhasználásával. A következő program erre mutat példát:

```
100  LET NAGY$ = "par szobol allo szoveg"
110  ! ----------------
120  ! a fenti sor egy karakterlanc-
130  ! valtozot deklaralt
140  ! ----------------
150  LET KICSI$ = NAGY$( 4 : 9 )
160  ! ----------------
170  ! a 150. sorban egy masik ka-
180  ! rakterlanc-valtozonak adtunk
190  ! erteket, mely NAGY$ 4. karak-
200  ! terevel kezdodik, és tart
210  ! egeszen a 9. karakterig
220  ! ----------------
230  PRINT NAGY$
240  PRINT KICSI$
250  END
```

A számítógép egyenként átmásolt néhány karaktert az egyik változóból egy ettől teljesen független másikba. Ezt úgy értük el, hogy adtunk egy nevet a készítendő, kisebb karakterláncnak (KICSI$), majd egy másik karakterláncnak (NAGY$) a nevét követő zárójelek között megadtuk, hogy hol kívánjuk az új karakterláncot kezdeni és befejezni.

Így például a

```
LET MONOGRAM$ = NEV$( 1 : 1 )
```

utasítás a következőt jelenti: hozzunk létre egy MONOGRAM$ nevű karakterláncot, és értéke legyen NEV$ első betűje.

Természetesen egy karakterláncból nem csak úgy emelhetünk ki egy részt, hogy az egy másik karakterlánc-változó értéke legyen. Előző példánkban a 150. sor kicserélhető pl. a

```
150  PRINT NAGY( 4 : 9 )
```

utasítással, mely a következőt jelenti:
írd ki NAGY$ karakterlánc betűit a negyediktől a kilencedikig.

A programozó saját belátására van bízva, hogy mikor melyik megoldást alkalmazza. Általában, ha egy karakterlánc bizonyos részletére többször van szükség, érdemes azt új változóba menteni, de ha csak egyszer kívánjuk használni, akkor a második megoldás megfelelőbb.

INKEY$

Nagyon fontos és igen sokkoldalúan használható karakterfüggvény az INKEY$. (A BASIC-nek a karakterláncokat kezelő funkcióit, karakterlánc-függvénynek vagy karakterfüggvénynek nevezzük. Neveik legtöbbje $ jellel végződik, jelezve, hogy értékük szintén karakterlánc.)

Az INKEY$ függvény lehetővé teszi, hogy a program futása közben nyomjunk le egy billentyűt úgy, hogy közben a program látszólag egy pillanatra sem áll meg.

Bizonyos mértékig hasonlít az INPUT funkcióra – a különbséget a következő példán keresztül szemléltetjük:

```
100  PRINT "Erted ezt a fejezetet ?"
110  PRINT
120  PRINT "Valaszolj I-vel vagy N-nel"
130  DO
140     LET A$ = INKEY$
150  LOOP UNTIL A$ <> ""
160  !-----------------
170  ! a 140. sorban a billentyuzet-
180  ! figyeles eredmenyet egy valto-
190  ! zoban helyeztuk el. Rovidesen
200  ! kiderul, miert volt ez fontos
210  !-----------------
220  IF A$ = "I" THEN
230   PRINT "Remelem, nem tevedsz."
240  ELSE IF A$ = "N" THEN
250   PRINT "Akkor olvasd el ujra !"
260  ELSE
```

```
270  PRINT "Hibas billentyu !"
280 END IF
290 ! ----------------
300 ! a korabbiakban mar talalkoztunk
310 ! az IF / THEN szerkezetekkel.
320 ! Az ELSE szo jelentese: kulon-
330 ! ben. ( A fejezet vegen reszle-
340 ! tesen megtargyaljuk az ilyen
350 ! felteteles kifejezeseket.)
360 ! ----------------
370 END
```

Miután a számítógép felteszi kérdését, az ezt követő első billentyűlenyomás a válasz, továbbengedve a program végrehajtását a 160. sorra és tovább.

Az INPUT kifejezéstől eltérően az INKEY$ függvény csak egy billentyű leütését várja, nincs szükség az ENTER használatára. (Így roppant hasznos játékprogramok írásához, egybetűs válaszok leolvasásához vagy a játék folytatásához, amely egy ciklus közepén várakozik...) Megjegyezzük, hogy a SHIFT billentyű használata nem jelent külön billentyűleütést.

Másik lényeges eltérés az INPUT utasítástól, hogy INKEY$ használata esetén a számítógép nem vár a válaszra, ezért kell példaprogramunkban a 130. és a 150. sor. Ha ezeket töröljük, programunk a másodperc tört része alatt lefut, nem hagyván időt a válaszadásra. Amikor ugyanis a számítógép a 140. sorhoz ér, megvizsgálja, hogy éppen abban a pillanatban le van-e nyomva egy billentyű, de mivel nincs, ezért A$ értéke „" (üres karakterlánc) lesz.

A 130. és a 150. sornak éppen az a feladata, hogy a számítógépet várakoztassa mindaddig, míg A$ valami más lesz, mint „" – ezután lehet folytatni a programot.

Figyeljük meg tehát, hogy az INKEY$ úgy viselkedik, mint egy változó, amely nagyon gyorsan váltogatja az értékét. Valójában a számítógép körülbelül minden másodpercben ötvenszer „megnézi", hogy le van-e nyomva billentyű. Ha pl. ekkor lenyomjuk az „a" gombot, INKEY$ értéke „a" lesz – de csak nagyon rövid időre. Általában ötvened másodperc múlva INKEY$ értéke ismét „" lesz, és ez marad a következő billentyű lenyomásáig.

Ez a magyarázata a program 140. sorának. Itt INKEY$ értékét karakterlánc-változóba mentjük, hogy ne változzon meg, míg programunk megvizsgálja értékét (220 - 280. sor).

Ha például programunkat a következőképpen változtatjuk meg :

```
150 LOOP UNTIL INKEY$ <> ""
220 IF INKEY$ = "I" THEN
240 ELSE IF INKEY$ = "N" THEN
```

az eredmény hibás lesz, ugyanis INKEY$ értéke megváltozik (visszaáll " " - ra), míg a program a 220., 240. sorra ér.

Akkor, ha INKEY$ értékére a program további részében nem lesz szükség, nem kell egy változóban tárolnunk. Így például, ha programunkat várakoztatni akarjuk addig, míg a használó le nem nyom egy billentyűt, a megoldás a következő lehet:

```
100  PRINT "Nyomj meg egy billentyut"
110  DO
120  LOOP UNTIL INKEY$ <> ""
```

UCASE$ és LCASE$

Ezekkel az újabb, hasznos karakterfüggvényekkel juthatunk karakterláncaink nagy-, ill. kisbetűs változatához. A következő mintapélda ezt mutatja:

```
100  !————————————
110  ! ez a program a begepelt szove-
120  ! get elobb kis, majd nagybetus-
130  ! se konvertalja
140  !————————————
150  INPUT PROMPT "Irj be nehany betut" : A$
160  PRINT A$
170  PRINT LCASE$( A$ )
180  PRINT UCASE$( A$ )
190  END
```

VAL

Az olyan karakterláncok, amelyek csak számjegyeket tartalmaznak, ezzel a függvénnyel átalakíthatók valódi számmá, mintha nem is karakterláncok volnának. Nézzük meg a következő példát:

```
100 INPUT PROMPT „Irj be nehany
    karaktert : ": A$"
110 PRINT A$
120 PRINT VAL( A$ )
130 END
```

A karakterlánc számértékének kiszámításákor a számítógép csak azokat a számjegyeket fogja figyelembe venni, amelyek megelőzik az első nemszámjegy karaktert. Így VAL(„123ADFG45") 123 lesz, míg VAL(„AB123") 0 értékű.

Ennek a műveletnek az ellenkezője, ha az STR$ függvény számot alakít karakterlánccá.

KARAKTERLÁNCOK ÖSSZEFŰZÉSE

Az előzőkben azt láttuk, hogyan készíthetünk új karakterláncot meglevőből úgy, hogy kivágunk belőle egy részt. Most ennek ellenkezőjét tesszük: karakterláncokat fűzünk egymáshoz. Ezt a műveletet konkatenációnak nevezzük, és végrehajtásához mindössze egy & jelet kell a karakterláncok közé írni. A következő példában felhasználjuk a már ismert karakterlánc műveletek egy részét is.

```
100 INPUT PROMPT "Irj be egy szot: ":
    KAR$
110 CLEAR SCREEN
120 LET ROV$ = KAR$( 1:1 ) & KAR$( LEN(
    KAR$ ) : LEN( KAR$ ) ) & „."
130 ! ----------------
140 ! a 120. sorban ROV$ erteke KAR$
150 ! elso es utolso karaktere lett,
160 ! melyeket egy pont kovet – je–
170 ! lezve, hogy roviditesrol van
180 ! szo
190 ! ----------------
200 PRINT KAR$; " roviditve : "; ROV$
210 END
```

A CIKLUSOK

Már eddig is többször találkoztunk olyan programrészletekkel, amelyekben a számítógép önmagát ismételte. Nyilvánvaló, hogy ilyen esetekben lényegesen egyszerűbb az ismétlendő részletet egyszer megírni és többször végrehajtani, mint sokszor egymás után megismételni a programban. Az így kialakított hurkokat ciklusoknak nevezzük.

Amint azt rövidesen látni fogjuk, egy ciklus szervezése alkalmas lehet egy teljes program vezérlésére is.

Minden ciklus kezdetét és vegét speciális utasítás jelzi - ez alól egyetlen kivétel van: ha a ciklust a GOTO utasítással szervezzük meg.

DO/LOOP CIKLUSOK

Ciklusok szervezésének legegyszerűbb módja a DO/LOOP utasításpár használata. Ezzel már korábban is találkoztunk, az INKEY$ függvényt bemutató példaprogramban. Íme egy újabb példa:

```
80  INPUT PROMPT „Irj be egy szamot,
    es en kiirom a szorzotablajat ! „:
    A
90  LET B = A
100 DO ! A ciklus kezdete
110   LET B = B + A
120   PRINT B
130   !----------------
140   ! a 90. sorban B valtozonak A er-
150   ! teket adjuk, majd a 110. sor-
160   ! ban A valtozatlan ertekevel
170   ! megnoveljuk, es a 120. sorban
180   ! ezen uj erteket kiirjuk.
190   ! Ezt hajtjuk vegre ciklikusan
200   !----------------
210 LOOP UNTIL B > 150   ! ciklus vege
220 !-----------------
230 ! a 210. sor megvedi programunkat
240 ! attol, hogy a vegtelensegig fus-
250 ! son - illetve mig a felhasznalo
260 ! le nem allitja
270 ! A LOOP UNTIL B > 150 jelentese:
280 ! „addig hajtsd vegre a ciklust,
290 ! MIGNEM B erteke 150-nel nagyobb
300 ! lesz"
310 !-----------------
320 END
```

Amint példánkból láthatjuk, a DO/LOOP utasításpár addig járatja a számítógépet a ciklusban, míg egy feltétel bekövetkezése azt meg nem állítja. Ez a feltétel két módon adható meg: az UNTIL vagy a WHILE szóval.

A WHILE és az UNTIL a feltételek két ellentétes tipusa. Például WHILE B = 150 és UNTIL B = 150 teljesen különböző dolgot jelentenek; WHILE használata esetén addig maradunk a ciklusban, amíg a feltétel igaz (B = 150), UNTIL esetén pedig akkor lépünk ki a ciklusból, ha a feltétel igazzá válik.

Mind az UNTIL, mind a WHILE tipusú feltétel alkalmazható a ciklus elején (DO UNTIL/DO WHILE) vagy a ciklus végén (LOOP UNTIL/LOOP WHILE). E két megoldás között az alapvető különbség az, hogy ha a DO UNTIL, ill. a DO WHILE struktúrát alkalmazzuk, akkor a feltételt a számítógép a ciklusba való belépéskor vizsgálja meg, míg a másik esetben csak a ciklus magját alkotó utasítások végrehajtása után értékeli ki. Így az első megoldás alkalmazása esetén előfordulhat, hogy a ciklus utasításaira egyetlenegyszer sem lesz szükség, míg a második esetben ez legalább egyszer mindenképpen bekövetkezik.

Javasoljuk, hogy a fenti példaprogramunk változtatgatásával próbálja ki e lehetséges ciklusszervezési módokat.

KILÉPÉS A CIKLUSOKBÓL

Az EXIT paranccsal bármikor kiléphetünk egy ciklusból. Az EXIT parancs nemcsak a DO/LOOP ciklusokhoz, hanem a következőkben ismertetendő FOR/NEXT ciklusokhoz is használható. Ezért alkalmazásakor meg kell adnunk, hogy adott esetben melyik ciklusból kívánunk kilépni: EXIT FOR, ill. EXIT DO.

E kilépést általában egy feltétel vizsgálatával kötjük össze, pl.:

```
1000  IF X > 25 THEN EXIT DO
```

A feltételekről szóló alfejezetben majd látni fogjuk, hogy hogyan végezhetjük el ezt a SELECT CASE utasítással.

Felhívjuk a figyelmet arra, hogy az EXIT utasítás az egyetlen helyes mód arra, hogy kilépjünk egy ciklusból, mielőtt az befejeződne. A GOTO utasítás e célra nem használható, megtévesztő lenne a programozó és a számítógép számára egyaránt. Az EXIT parancs egyszerűen utasítja a számítógépet arra, hogy a program végrehajtását a ciklus vége után következő első utasítással folytassa.

FOR/NEXT CIKLUSOK

A FOR/NEXT ciklusok talán kevésbé általánosak, mint az előbb látott DO/LOOP ciklusok, de bizonyos esetekben igen célszerűnek bizonyulhatnak.

Íme egy rövid példa használatukra:

```
100  INPUT PROMPT „Hanyszor menjek
     korbe ?": X
110  FOR I = 1 TO X
120    PRINT „Korbe, korbe ..."
130  NEXT I
140  END
```

Látható, hogy most a begépelt szám határozza meg, hogy hányszor hajtsuk végre a ciklust. Ez FOR/NEXT ciklus fő célja – egyszerűen és félreérthetetlenül megadhatjuk, hogy hányszor kell ismételni a ciklus magját. Így itt nincs szükség külön feltételek alkalmazására.

A FOR/NEXT ciklusban a növekmény lehet 1, 2, 100 vagy éppen 435678, vagy akár visszafelé is számolhatunk. A következő program például kettesével számol visszafelé:

```
10  FOR I = 40  TO 0  STEP -2
20     PRINT I,
30  NEXT I
40  END
```

A STEP parancs megadása nélkül nem tudjuk gépünket rávenni arra, hogy 40-től számoljon 0-ig, a STEP megadása nélkül ugyanis a FOR/NEXT ciklus mindig egyesével számol felfelé.

Érdemes példánkban megfigyelni a PRINT I utasítást. A 10. sorban használtunk ugyanis egy I nevű változót, amelynek értéke minden ciklus után megváltozik: megnövekszik azzal az értékkel, amit a STEP után megadtunk, ill. 1-gyel, ha hiányzott a STEP parancs.

A DO/LOOP ciklusutasítással készített szorzótábla-programunkat most rövidebben megírhatjuk a FOR/NEXT utasításpár felhasználásával:

```
100 INPUT PROMPT „Irj be egy szamot,
    es en kiirom a szorzotablajat.":A
110 FOR P = 0  TO 150  STEP A
120    PRINT P,
130 NEXT P
140 END
```

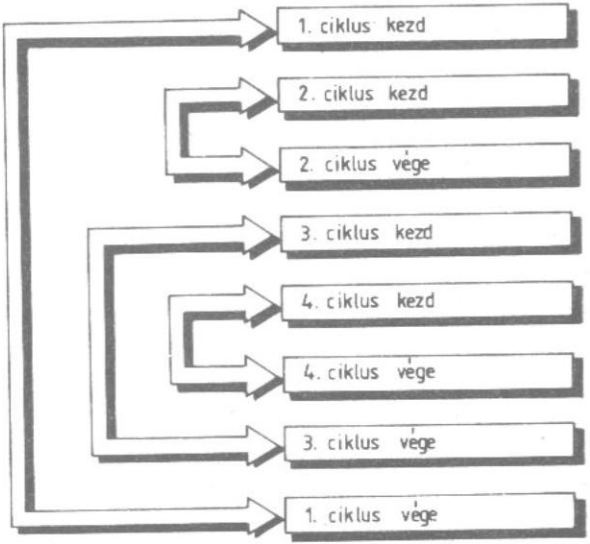
Fontos megfigyelnünk, hogy a FOR/NEXT ciklusok esetén a számítógép a feltételt mindig a ciklusmag végrehajtása előtt vizsgálja meg, tehát előfordulhat, hogy a ciklusra egyszer sem lesz szükség. Ezt ki is próbálhatjuk, ha a szorzótábla programunkba 150-nél nagyobb számot írunk.

EGYMÁSBA ÁGYAZOTT CIKLUSOK

Ez a furcsa kifejezés arra vonatkozik, ha egy ciklust egy másik belsejébe helyezünk. A legfontosabb, ezzel kapcsolatos szabály, hogy a belső ciklus sohasem végződhet a külsőn kívül. Íme egy példa:

```
100 FOR I = 1 TO 20 ! kulso ciklus kezdodik
110.   PRINT I,
120    FOR J = 1 TO 4 ! belso ciklus kezdodik
140      PRINT I + 10; I - 10;
150    NEXT J ! belso ciklus vege
160    PRINT
170 NEXT I ! kulso ciklus vege
180 END
```

A ciklusok tetszés szerinti mélységig egymásba ágyazhatók. Végignézve egymásba ágyazott ciklusok sorozatát, a ciklusok kezdetét emelkedő, a végét csökkenő sorrendben kell megtalálnunk, ahogy az a következő ábrán látszik:



Látható, hogy a ciklusokkal igen sokrétűen gazdálkodhatunk. Sok ciklus egymásba ágyazása azonban nagy figyelmet kíván. Az Enterprise az előző mintapéldában látott módon megkönnyíti a ciklusok azonosítását, emellett segítségünkre lehet megjegyzések elhelyezésében is. Ezek ellenére, ha sokszorosan egymásba ágyazott ciklusokra van szükségünk, célszerű a belső szinteket függvényekkel kifejezni. Ez nemcsak a program elkészítését egyszerűsíti, hanem áttekinthetőségét is javítja.

A FELTÉTELEK

Gyakran találkozhatunk irodalmi művekben olyan számítógépekkel, amelyek ember módjára gondolkodnak. Néha mi magunk is hajlamosak vagyunk úgy gondolni rájuk, mint felettébb okos masinákra. Valójában a számítógép nem képes gondolkodásra, csak nagyon egyszerű döntésekre, és összehasonlításokra.

IF/THEN UTASÍTÁS

Tárgyalásunkat az összehasonlításokkal kezdjük. Az 1. fejezetben már említettünk bizonyos relációs operátorokat – nos, ezek a számítógép által végzett összehasonlítások és az ezekre épülő döntések alapjai.

A következő példaprogramunk bemutatja, hogyan hasonlíthatjuk össze ezekkel a karakterláncokat:

```
100 LET EGYIK$ = "mosomedve"
110 INPUT PROMPT "Irj be egy szot !":
    MASIK$
120 ! ----------------
130 ! a fenti ket sor mar ismeros
140 ! szamunkra. A kovetkezo sorokban
150 ! a ket karakterlancot a <>
160 ! operatorral hasolitjuk ossze,
170 ! mely "nem egyenlo"-t jelent.
180 ! Ha megegyeznek, mindkettot, ha
190 ! nem, csak a "mosomedve"-t irjuk
200 ! ki
210 ! ----------------
220 PRINT
230 PRINT
240 IF EGYIK$ = MASIK$ THEN PRINT
    EGYIK$;" = ";MASIK$
250 IF EGYIK$ <> MASIK$ THEN PRINT
    EGYIK$
260 END
```

Az IF/THEN utasítás az Enterprise olyan döntési lehetőségeinek egyike, amely közel áll a valós angol nyelvhez. Segítségével változókat, ill. állandókat vizsgálhatunk meg, hogy teljesítenek-e bizonyos feltételt, vagy sem. Az IF/THEN utasítással nem végezhető számítás, de segítségével a program végrehajtását úgy módosíthatjuk, ahogy azt bizonyos számok vagy karakterláncok viszonya szükségessé teszi.

AZ IF UTASÍTÁSCSOPORT

Az IF/THEN szerkezet nem csak egy sorból állhat, használható ún. IF utasításcsoport is. Ez lehetővé teszi, hogy programunk több soron át több összehasonlítást és döntést tegyen, valamint hogy egy feltételre több utasítás vonatkozzon.

Előző példaprogramunkat más módon is megírhattuk volna, pl. ha kicseréljük a 240 – 250. sorokat a következőkre:

```
240  IF EGYIK$ = MASIK$ THEN
244      PRINT EGYIK$;" = ";MASIK$
250  ELSE
255      PRINT EGYIK$
257  END IF
```

Az eredmény ugyanaz lesz. Ennek az az oka, hogy egy feltételnek két eredménye lehet: vagy igaz (a két karakterlánc megegyezik), vagy hamis (különböznek). Így tehát módosított programunk a következőre utasítja a számítógépet: ha (IF) EGYIK$ és MASIK$ megegyezik, akkor (THEN) nyomtasd ki mindkettőt, különben (ELSE), (ha bármi más a helyzet), nyomtasd ki EGYIK$ -t.

Ez az IF utasításcsoport egy IF, egy THEN és egy ELSE szót tartalmazott, és az END IF parancs zárta le.

Az ELSE utasítás tehát „valami más"-t jelent. Ilyenkor az IF/THEN-t követő utasítást a gép akkor hajtja végre, ha a feltétel igaz, ha pedig hamis, akkor végrehajtja az ELSE-t követő utasítást, ha létezik.

Az ELSE utasításnak mindig egy új sor elején kell állnia, nem lehet az IF/THEN-nel azonos sorban.

A következő programban hosszabb IF utasításcsoportot mutatunk be:

```
100 RANDOMIZE
110 LET A = RND( 5 )
120 ! ——————————
130 ! A fenti ket sor veletlenszeru
140 ! szamokat allit elo. Ezt a lehe-
```

```
150 ! toseget nagyon sokoldaluan
160 ! hasznosithatjuk – peldaul ide–
170 ! alis egy Roulette jatek keszitese–
180 ! hez! Az RND fuggveny "kesziti"
190 ! el a veletlenszamot, a 100. sor
200 ! utasitasa pedig biztositja,
210 ! hogy ha a programot tobbszor
220 ! futtatjuk, mindig mas es mas
230 ! szamot kapjunk
240 ! ————————————
250 PRINT "Gondoltam egy szamot :";
260 IF A = 1 THEN
270   PRINT "Egyes ! "
280   PRINT
290 ELSE IF A = 2 THEN
300   PRINT "Kettes! "
310   PRINT
320 ELSE IF A = 3 THEN
330   PRINT "Harmas! "
340   PRINT
350 ELSE
360   PRINT A;" Ez bizony tul
      bonyolult nekem."
370 END IF
380 END
```

Ez a program tehát véletlenszám előállításával kezdődik. Erre ne is legyen gondunk a továbbiakban – de ha ezt részletesebben is meg akarjuk vizsgálni, egy ciklus segítségével kiirathatunk néhány tucatot belőlük, és kereshetünk valamilyen törvényszerűséget köztük!

Ezután, a szám értékétől függően mást és mást ír ki a számítógép a képernyőre: ha szám van az IF utasítások valamelyikében, akkor a szám értékét betűvel, ha nincs ilyen, akkor a program a 350. sorban található magányos ELSE utasítással folytatódik, és kiírja a 360. sorban látható üzenetet.

Példát láttunk tehát egy összetett IF utasításcsoport használatára, amelyben a lehetséges esetekre vonatkozó utasítások az egyes IF, ill. ELSE IF sorok után következnek, az utasításcsoportot pedig minden esetben az END IF sor zárja le.

Összefoglalásul azt mondhatjuk, hogy az IF/THEN utasítást akkor használjuk, ha csak egy egyszerű teendőnk van egy feltétel teljesülése esetén. Az IF utasításcsoport viszont teljes programot vagy programrészt vezérelhet több feltételtől függően.

SELECT CASE UTASÍTÁS

Ezzel az utasítással sokkal egyszerűbb vizsgálatokat végezhetünk, mint az IF/THEN-nel. A SELECT CASE utasítás tulajdonképpen egy változó lehetséges értékeitől függő többirányú választást tesz lehetővé.
Írjuk át előző példaprogramukat:

```
100 RANDOMIZE
110 LET A = RND( 5 )
120 ! ----------------
130 ! A kovetkezokben A lehetseges
140 ! erteket vizsgaljuk meg a SELECT
150 ! CASE utasitassal.
160 ! Peldaul a 300. sor jelentese a
170 ! kovetkezo lesz: ha A erteke 2,
180 ! akkor hajtsd vegre a kovetkezo
190 ! sorban talalhato utasitast, ha
200 ! nem, menj a kovetkezo CASE sor-
210 ! hoz
240 ! ----------------
250 PRINT "Gondoltam egy szamot :";
260 SELECT CASE A
270 CASE 1
280   PRINT "Egyes ! "
290   PRINT
300 CASE 2
310   PRINT "Kettes! "
320   PRINT
330 CASE 3
340   PRINT "Harmas! "
350 ELSE
360   PRINT A;" Ez bizony tul
      bonyolult nekem."
370 END SELECT
380 END
```

A példából látható, hogy a SELECT CASE utasításcsoportot – az IF-hez hasonlóan – END SELECT sorral kell lezárni.

Amint tudjuk, az IF utasítást nem feltétlenül kell ELSE-nek követnie, de követheti. Hasonló a helyzet a SELECT CASE utasításnál is: lehetőség van egy „egyéb eset" ág feltüntetésére, ennek formája itt CASE ELSE, ahogy azt a példaprogramban láthattuk.

Lehetőség van továbbá egy CASE utasítás után több lehetséges érték feltüntetésére is, pl.:

```
CASE 1, 3, 5
 PRINT "Ezek páratlan számok."
```

Javasoljuk, hogy ezután a most megismert feltételvizsgáló utasítások segítségével próbáljon meg dobókocka-programot írni, vagy felhasználva a másutt ismertetendő ORD függvényt, írjon olyan programot, amely a karakterláncokat ABC sorrendbe rendezi. A tényleges programozás közben szerzett tapasztalatokat ugyanis semmiféle könyv vagy leírás nem pótolhatja.

NAGY MENNYISÉGŰ ADAT TÁROLÁSA

Már felfedeztük, hogy a programozás valójában az információk kezelésének egy módja. Mindeddig azonban csak kevés adattal dolgoztunk - szavakkal vagy számokkal, melyeket az INPUT utasítással feltett kérdésekre válaszként írtunk be, vagy a számítógép írta ki nekünk valamilyen üzenet formájában. Nem beszéltünk azonban még arról, hogy mi módon kezelhetünk olyan rengeteg adatot, ami egy játékprogramhoz, vagy bankszámlánk nyilvántartásához szükséges. Az Enterprise rendkívül hatékonyan használható ilyen problémák megoldására.

Névsorokat a programban ún. tömbökben tárolunk. A tömb olyan változó, amely több más változót tartalmaz önmagában. Felfoghatjuk a tömböket olyan nagy dobozokként is, amelyek további kis dobozokat rejtenek.

A tömbök nemcsak karakterláncok, hanem számok, numerikus változók tárolására is alkalmasak, amint azt a most következő fejezetben látni fogjuk.

NUMERIKUS TÖMBÖK

Ahhoz, hogy programunkban egy tömböt használhassunk, azt először deklarálnunk kell. Írjuk be számítógépünkbe a következő programsort:

```
100 NUMERIC TOMB( 1 TO 10 )
```

Ezzel utasítjuk a számítógépet, hogy hozzon létre valahol a memóriájában egy TOMB nevű tárolót, amely elég nagy ahhoz, hogy 10 változó elférjen benne. Ezek a változók a tömb elemei, és bármilyen szám tárolható bennük, pontosan úgy, ahogy azt az egyszerű változók esetében már megszoktuk. E változók

valamelyikére a TOMB névvel és az utána zárójelben álló számmal hivatkozhatunk; így pl. TOMB(7) a TOMB nevű tömb hetedik elemét jelöli.

Folytassuk tovább programunkat:

```
110 FOR S = 1 TO 10
120    PRINT PROMPT „Irj be egy szamot"
 : TOMB( S )
130 NEXT S
```

Ezzel a programrészlettel számokat írtunk a tömbbe. Futtassuk le, majd adjuk ki a PRINT TOMB(9) , PRINT TOMB(6) utasítást, vagy irassuk ki a tömb bármelyik elemét. Láthatjuk, hogy számsorozatot írtunk be a számítógépbe.

Hasonlóképpen akkor, ha pl. fel akarjuk jegyezni december hó minden egyes napján a hőmérsékletet, egy DECEMBER nevű 1-től 31-ig számozott elemű tömb szükséges.

Egy tömb elemeinek a sorszáma tetszőleges tartományba eshet; lehet 34-től 79-ig, 123-tól 456-ig, de 3675324-tól 3675399-ig is bármennyi, sőt akár negatív is, pl. NUMERIC TABLA(-20 TO -10).

KÉTDIMENZIÓS TÖMBÖK

A tömbök nem mindig olyan egyszerűek, ahogy azt az előző pontban láttuk. A következőkben bemutatjuk az egy- és a kétdimenziós tömbök közti különbséget.

Egydimenziós tömböt úgy ábrázolhatunk, mint egy listát; sorszámként, amelyet a hozzá tartozó érték követ:

Sorszám	Érték
1	12.343
2	234.78767
3	3
4	5.9
6	34.321

A kétdimenziós tömböket mátrixoknak nevezik, ezek leginkább egy nagy polcos szekrényhez hasonlíthatók, ahol minden polcnak és minden oszlopnak van egy sorszáma, és egy rekesz a két szám együttes használatával azonosítható, valahogy így:

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4
4.1	4.2	4.3	4.4

A számunkra fontos adatokat a tömb elemei, az egyes rekeszek tartalmazzák.

KARAKTERLÁNCOKAT TARTALMAZÓ TÖMBÖK

A karakterláncokat tartalmazó tömbök hasonlóak a numerikusahoz, azzal a különbséggel, hogy a NUMERIC szó helyett a STRING-et használjuk. Itt azonban – a numerikus tömbökével ellentétben – az egyes elemek hossza változtatható.

Egy tömbelem hosszának változtatása valójában a legnagyobb lehetséges hosszat változtatja meg. Mint ahogy egy vödröt sem kötelező minden alkalommal szinültig megtölteni, senki sem kötelez minket arra, hogy egy tömb elemeit minden alkalommal teljes mértékben felöltsük.

Egy tömb deklarálásakor a számítógép lefoglal számára egy bizonyos területet a memóriájában. Ez – ha mást nem adunk meg – karaktertömbök esetén 132 * 132 karakter nagyságú lesz. Ettől eltérő méretet a következő módon adhatunk meg:

```
STRING TOMB$( 3Ø TO 5Ø ) * 1Ø
```

Ezzel az utasítással egy olyan 21 elemű tömböt foglaltunk le, amelyben mindegyik elem 10 karakter hosszúságú. Természetesen, minél több helyet foglal el egy tömb, annál több adatot tudunk benne tárolni. De számítógépünk memóriája nem végtelen nagy, ezért javasoljuk, hogy csak akkora tömböket foglaljunk le, amekkorákra valóban szükségünk lesz.

Változók deklarálása

Egyszerű, nem tömb változók is deklarálhatók a NUMERIC, ill. STRING kulcsszó segítségével. Az egyetlen eltérés a tömbök deklarációjától, hogy nem adunk meg elemhatárokat. A következőkben változók deklarálására mutatunk példákat:

```
NUMERIC A
NUMERIC C,D,H,X
STRING A$, B$, HELLO$*8
```

READ/DATA UTASÍTÁSOK

Már több olyan példát láttunk, ahol két (esetleg több) BASIC kulcsszó egymással szoros összefüggésben van, külön-külön nincs is értelmük. A READ és a DATA is ilyen szavak, sőt hozzájuk tartozik a RESTORE parancs is.

Ezek újabb lehetőséget nyújtanak nagyobb mennyiségű adat kezelésére. Egyúttal alkalmasak arra is, hogy velük számokat vagy karakterláncokat töltsünk tömbök elemeibe. A munka oroszlánrésze a READ utasítással végezhető – amint ezt a következő példában látjuk:

```
 8Ø LET P = 1
 9Ø DO UNTIL P = Ø
1ØØ       READ P
11Ø       PRINT P
12Ø LOOP
13Ø DATA 1,2,3,4,5,6,7,8,9,1Ø
14Ø DATA 11,12,13,14
15Ø DATA 15,16,17,18,19,Ø
16Ø END
```

A DATA sorokban csak adatok tárolhatók. Az adatokat vesszőkkel választjuk el egymástól. A READ parancs utasítja a számítógépet arra, hogy vegyen elő egy DATA adatelemet és tegye be egy változóba, amit ezután ugyanúgy lehet használni, mint bármely más adatot vagy változót. Példánkban ez az adat a P változóba kerül, amit azután kinyomtatunk a képernyőre.

Figyeljük meg, hogy minden egyes DATA adatelemhez ugyanazt a P változót használjuk miden alkalommal. Tehát a program beolvas (READ) egy adatot a P változóba, kiírja, beolvassa a következőt, és így tovább mindaddig, míg az utolsó, a 0 elemet be nem olvastuk.

A számítógép minden egyes DATA adatelemet csak egyszer olvas be. Miután beolvasta, megjegyzi a pozícióját, és a következő READ utasítás a következő adatelemet olvassa, így haladva balról jobbra és fentről lefelé, ahogy egy könyvben olvasunk. Ha valamennyit beolvastuk, a gép megállapítja, hogy nincs több adat – és nagy gondban lenne, ha újabb READ utasítást kapna. Ha mégis erre kerül sor, akkor a gép hibát jelez.

Az következő program bemutatja a READ/DATA utasítás használatát karakterláncok esetén.

```
150 CLEAR SCREEN
160 PRINT "Most pedig mondok egy
    meset ..."
170 PRINT
180 PRINT "Ime :"
190 PRINT
200 PRINT
210 FOR X = 1 TO 5000
220 NEXT X
230 ! ----------------
240 ! a kovetkezo resz a program fo
250 ! resze: egy DO / LOOP cikluson
260 ! belul beovassuk A$-t a prog-
270 ! ramban barhol levo DATA utasi-
280 ! tas soron kovetkezo adateleme-
290 ! bol. Ezutan megvizsgaljuk, hogy
300 ! nem „Vege"-e – ez jelzi sza-
310 ! munkra az adatok veget. Ha ve-
320 ! ge, kilep a ciklusbol, ha nem,
330 ! kiirjuk a felolvasott szot,
340 ! es indul az egesz elolrol...
350 ! ----------------
360 DO
370     READ A$
```

```
380     IF A$ = „Vege" THEN EXIT DO
390     PRINT A$;" ";
400      FOR Y = 1 TO 500
410     NEXT Y
420 LOOP
430 !
440 PRINT
450 PRINT
460 PRINT „ Itt a vege ..."
470 PRINT
480 PRINT
490 END
500 ! ————————————————
510 ! a program hatralevo resze a
520 ! DATA adatokat tartalmazza
530 ! ————————————————
540 DATA Egyszer,volt,hol,nem,volt,
    volt,egyszer,egy,kicsi
550 DATA ENTERPRISE,nevu,szamitogep.,ez
    egy, roppant,vidam,szamitogep
560 DATA volt.,Egy,nagy,Anglia,nevu,
    orszag,legjobb,programozoi
570 DATA hónapokon,at,ejjel,nappal,
    dolgoztak,azon",",hogy,az
580 DATA ENTERPRISE-t,a vilag,legjobb,
    szamitogepeve,tegyek.
590 DATA Es,most,te,az ENTERPRISE,
    BASIC,nyelvet,tanulod.
600 DATA Hat,nem,vagy,egy,szerencses,
    ficko ?!!!,Vege
```

Ha most programunkba betesszük a következő sorokat:

```
140 DO
485 LOOP
```

láthatjuk, hogy számítógépünk egyszer kiírja a történetet hibátlanul, utána azonban egy hibaüzenetet ad arról, hogy nincs több DATA.

Adjuk tehát programunkhoz a

```
488 RESTORE
```

sort. Ez utasítja a gépet arra, hogy álljon vissza a DATA sorozat legelejére. Ha a RESTORE utasítás után egy sorszámot írunk, akkor ez után az adott számú sor DATA adatait fogja beolvasni a következő READ utasítás.

Figyeljük meg, hogy példánkban a DATA adatelemei nem állnak idézőjelek között. Ha ugyanis,a READ utasítás után karakterláncváltozó áll (példánkban A$), a számítógép tudja, hogy karakterláncnak kell tekintenie a soron következő DATA elemet. Idézőjelet csak akkor kell használnunk, ha vesszőt akarunk tenni az adatok közé (mint ahogy azt példánk 570. sorában láthatjuk) – egyébként a gép azt gondolná, hogy a vessző az adatelemeket választja el egymástól.

FÜGGVÉNYEK DEFINIÁLÁSA

A függvények, amelyekről e fejezetben szó lesz, programok a programban, amelyeket egy-egy speciális feladat végrehajtására hozunk létre, hogy aztán újra és újra bármikor felhasználhassuk őket.

Egyszerű példaként tegyük fel, hogy adott üzenetet a program különböző részein ki akarunk íratni a képernyőre. Ezt a műveletet példánkban függvényként definiáljuk.

```
100 DEF FIGYELMEZTETES
110   CLEAR SCREEN
120   PRINT AT 10,7:"Na most figyelj!"
130   ! ————————
140   SOUND ! ez az utasitas egy
150   ! hangjelzest ad.
160   ! ————————
170 END DEF
```

Minden esetben nevet kell adni a függvénynek – példánkban ez FIGYELMEZTETES – és ezt a DEF kulcsszónak kell megelőznie. Az ezt követően definiált függvényt – lehet egy vagy akár tízezer sor hosszú – mindenképpen az END DEF sorral kell befejezni.

FÜGGVÉNYEK HÍVÁSA

Egy függvény önmagában nem működőképes, ha megpróbáljuk lefuttatni mintaprogramunkat, semmi sem történik. A definiált függvényt a CALL utasitással aktivizálni kell – ezt a műveletet nevezzük a függvény hívásának. A függvény definiálható a program bármelyik részén, akár a hívás, akár az END utasítás után is.

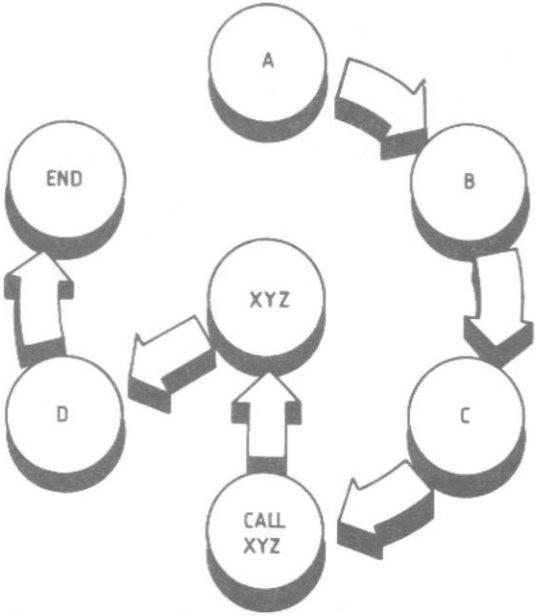
Példánkat egészítsük ki a következő sorral:

```
8Ø CALL FIGYELMEZTETES
9Ø END
```

– és a függvény már működni fog.

Ha a számítógép CALL utasítást talál, minden alkalommal megkeresi, hogy a programban hol található az adott függvény definíciója, és azon a helyen végigfuttatja a függvényt, ezután visszatér a CALL-t követő utasításra, és ott folytatja a program végrehajtását.

A következő ábra segítségével ez könyen érthetővé válik:



Ne feledjük el, hogy egy függvény mindaddig semmit sem csinál, míg egy CALL utasítással a nevére nem hivatkozunk. Ha a program folyamatos végrehajtása közben a gép függvénydefiníciót talál, átugorja, és az azt követő utasítással folytatja a végrehajtást.

LOKÁLIS ÉS GLOBÁLIS VÁLTOZÓK

Ahhoz, hogy a függvényekben a változókat hibátlanul használhassuk, néhány fontos szabályra kell tekintettel lennünk. Próbáljuk ki a következő programot:

```
100 DEF KOB
110    INPUT PROMPT "A kobre emelendo
       szam :":Z
120    PRINT Z;" a kobon = ";Z*Z*Z
130 END DEF
140 CALL KOB
```

Futtassuk le, majd írjuk be a következő sort:

```
150 PRINT Z
```

és most futtassuk le ismét. Azt tapasztaljuk, hogy annak ellenére, hogy a KOB függvény helyesen működik, számítógépünk hibaüzenetet küld, mikor a 150. sorhoz ér. Vajon mi ennek az oka ?

A válasz e kérdésre az, hogy jelen esetben Z a KOB függvény belső, ún. lokális változója, és mint ilyent, csak a KOB függvény ismeri, a program többi része semmit sem tud róla. A számítógép a függvényt önálló kis programként kezeli, a saját változóival együtt, amelyek a függvény saját céljait szolgálják. E változókat a program többi része nem ismeri, és ha a függvény végrehajtása befejeződik, ezek a lokális változók elvesznek. Így példánk 150. sorában a program nem ismeri a Z változót, ezért küld hibaüzenetet.

Ha azonban a 110. sort átszámozzuk 90-re – azaz a függvényen kívülre helyezzük –, programunk fő része is ismerni fogja a Z változót, így a 150. sor nem okoz hibát többé.

Azzal, hogy a Z változót a függvényen kívülre helyeztük, külső, ún. globális változó lett belőle. Egy olyan globális változó, amely a program teljes egészében ismert, felhasználható.

Ha most beszúrjuk programunkba a

```
125 LET Z = 20
```

sort, megfigyelhetjük, hogy a függvény megváltoztatja a globális változó értékét. A függvény előbb felhasználja Z főprogramban beállított értékét, elvégez segítségével egy számítást, majd megváltoztatja értékét. A főprogram – visszatérve a függvényből – ezt a megváltoztatott értéket találja a Z változóban.

Összefoglalva: ha egy függvény olyan változót használ, amely a függvény meghívása előtt nem fordult elő a program fő részében, akkor a függvény ezt mint lokális változót kezeli. Ha azonban ez a változó már valahol deklarálva volt, a függvény globális változónak tekinti; így ha a függvény megváltoztatja a változó értékét, a program többi része ezzel a megváltozott értékkel fut tovább.

Korábbi fejezetekben már találkoztunk változók deklarálásának lehetőségével. Akkor azt a megjegyzést tettük, hogy bár a deklaráció nem mindig szükséges, mégis ajánlott. Most láthatjuk, hogy sok függvény használata esetén a deklaráció gyakran elengedhetetlen.

Egy változó egyrészt a NUMERIC, ill. STRING kulcsszavak segítségével, másrészt pedig a LET utasítással deklarálható (pl. LET A = 20). Előző példáinkban nem volt jelentősége annak, hogy melyik módot használjuk, de függvények használata esetén ez már egyértelműen tisztázódik.

Módosítsunk ismét kissé az eredeti példaprogramunkon:

```
110 NUMERIC Z
115 LET Z = 3
```

Most azt tapasztaljuk, hogy programunk két különböző Z változóval dolgozik – egyik lokális változó a függvény belsejében, a másik globális változó kívül. Ha viszont töröljük a 110. sort, ismét csak egy globális változónk lesz – az, amelyet a 90. sorban deklaráltunk –, mivel a 115. sor LET utasítása önmagában nem deklarál lokális változót, ellentétben a NUMERIC kulcsszóval.

Mindeddig olyan függvényekkel találkoztunk, amelyek CALL utasítással történő meghívásuk után különféle bonyolult funkciókra képesek, így pl. üzenetek kiírására, sípolásra stb. Most megvizsgálunk néhány olyan függvényt, amelyek egyszerűen csak egy számértéket adnak a főprogram számára.

Egész csomó ilyen függvény áll eredetileg is rendelkezésünkre a BASIC nyelvben: ilyen pl. az SQR (négyzetgyök) függvény. Egy program egyik sora lehet például a következő:

```
PRINT SQR( 121 )
vagy
LET M = 2 * SQR( N ) + 1
```

Az SQR függvény kiszámolja a zárójelek között álló szám vagy kifejezés négyzetgyökét. Az eredményül kapott szám pedig éppen úgy használható a továbbiakban, mint bármelyik változó vagy állandó.

Már ismerjük az INT függvényt is; a BASIC ezeken kívül még sok olyan függvényt ismer, amelyek sok, gyakran szükséges számítástól kímélnek meg minket.

Tegyük fel, hogy most olyan programot kívánunk írni, amely számok faktoriálisát számítja ki (pl. 4 faktoriálisa = 4 * 3 * 2 * 1; 6 faktoriálisa = 6 * 5 * 4 * 3 * 2 * 1 stb.). Faktoriális számítására nincs előre elkészített BASIC funkciónk, ezért megkíséreljük magunk elkészíteni a szükséges függvényt.

Próbálkozzunk pl. a következő változattal:

```
100 DEF FACT
110 ! ————————————
120 ! e fuggveny a globalis F valtozo
130 ! erteket hasznalja fel, valtoz-
140 ! tatja meg, es adja vissza a
150 ! foprogramnak.
```

```
160 ! ————————————————
170   FOR Y = F - 1 TO 1 STEP -1
180     LET F = F * Y
190   NEXT Y
200   IF F = 0 THEN LET F = 1
210 END DEF
```

A következő program arra mutat példát, hogy hogyan írathatjuk ki pl. a 13 faktoriálisát vagy hogyan használhatjuk a 7 faktoriálisát egy kifejezésben.

```
220 LET F = 13
230 CALL FACT
240 PRINT F
250 LET F = 7
260 CALL FACT
270 LET SZAM = F * 1.5 + 3
280 PRINT SZAM
```

Amint látjuk és érezzük, ez a megoldás sokkal körülményesebb, mint pl. az SQR függvény hívása volt, hiszen itt a FACT minden egyes meghívása előtt egy globális változóba (F) kellett töltenünk azt a számot, amivel a műveletet végre akartuk hajtani. Ám számítógépünk lehetőséget ad a kényelmetlenség megszüntetésére. Töröljük most az előbbi példaprogramunkat, és írjuk be a következőt:

```
100 DEF FACT( X )
110   FOR Y = X - 1 TO 1 STEP -1
120     LET X = X * Y
130   NEXT Y
135   IF X=0 THEN LET X=1
140   LET FACT = X
150 END DEF
160 PRINT FACT( 13 )
170 LET SZAM = FACT * 1.5 + 3
180 PRINT SZAM
```

Ezzel a FACT függvényt a BASIC saját függvényeihez hasonlóvá tettük és pontosan úgy használhatjuk, mint ahogy pl. az SQR vagy az INT függvényt.

Programunk az eddigitől két sorban különbözik: a 100. sorban zárójelek között álló X jelzi a számítógépnek, hogy a főprogramban a FACT szó után keressen egy zárójelek között álló számot, és annak értékét adja az X belső változónak. A 140. sor értéket ad egy változónak, amelynek neve megyezik a

függvény saját nevével – ilymódon lehetővé teszi, hogy ez az érték közvetlenül abban a sorban térjen vissza a főprogramhoz, ahol a neve előfordult. Így a függvény hívásakor nincs szükségünk a CALL szóra.

FÜGGVÉNYEK BEMENŐ VÁLTOZÓI

A programunk 100. sorában lévő zárójelek között álló X nevű változót a függvény bemeneti változójának nevezzük. Ez jelzi a függvénynek, hogy egy számot várjon adatként, amellyel majd tovább dolgozik. (Alfejezetünk utolsó példájában olyan függvényt láthatunk, amelynek két bemeneti változója van.) Ez a szám származhat a főprogram valamelyik globális változójától is.

Helyettesítsük most programunk két sorát a következőkkel:

```
160 LET A = 11
170 PRINT FACT( A )
```

A számítógép a függvény meghívásakor belenyúz az A dobozba, és annak tartalmáról másolatot helyez a függvény saját, X nevű dobozába – ezt fogja a továbbiakban használni a számításokhoz. A függvény végrehajtása közben X értéke módosul, A azonban változatlan marad.

Ha most ismét megváltoztatjuk programunkat:

```
160 LET X = 11
170 PRINT FACT( X )
```

láthatjuk, hogy definiáltunk egy globális változót, a függvény bemenő változójával azonos néven. A program azonban továbbra is helyesen működik – a számítógép most is úgy kezeli ezt a két változót, mint két különböző dobozt, azonos neveiktől függetlenül. Erről meg is győződhetünk, ha két további sort írunk a programhoz:

```
145 LET X = 100
```

és a főprogramba:

```
180 PRINT X
```

Lefuttatva a programot láthatjuk, hogy a 180. sorban X értéke 11 és nem 100. A 145. sor tehát csak a lokális X változót változtatta meg.

HIVATKOZÁSI (REFERENCIA) PARAMÉTEREK

Az előző részben olyan függvényekkel ismerkedhettünk meg, amelyeknek van bemeneti változójuk (paraméterük), ezekkel valamilyen műveletet végeznek, és

az a változó, amelynek értékét a bemeneti változóba átmásoltuk, változatlan marad.

Más a helyzet akkor, ha a bemeneti változó elé a REF szót írjuk. Ilyenkor függvényünk megváltoztatja a híváskor feltüntetett változó(k) értékét.

A következő mintaprogram két számot, A-t és B-t kér tőlünk, majd megváltoztatja őket oly módon, hogy A-t B-edik, majd B-t A-adik hatványára emeli.

```
100 INPUT PROMPT „Elso szam :":A
110 INPUT PROMPT „Masodik szam :":B
120 CALL HATVANY( A, B )
130 PRINT A,B
140 END
150 DEF HATVANY( REF X, REF Y )
160   LET Z = X
170   LET X = X ↑ Y
180   LET Y = Y ↑ Z
190 END DEF
```

A 150. sor két bemenő változót jelez. A függvény meghívásakor a zárójelek közti első változó X, a második Y helyére kerül. Eddig minden megegyezik a korábban látottakkal – annyi csak a különbség, hogy most nem csak egy, hanem két bemeneti változónk van, azon kívül mivel e függvény nem egy, hanem két értéket ad vissza, hívásakor szükséges a CALL szó.

A REF szavak példánk 150. sorában egyszerűen azt jelentik, hogy miután függvényünk befejezte tevékenységét, X új értékét helyezze vissza A, Y-ét pedig B globális vátozóba.

A változók, ill. értékeik adásvételét a függvények és a program más részei között paraméterátadásnak nevezzük. Ha egy függvény megváltoztatja annak a globális változónak az értékét, ahonnan a működéséhez szükséges adatokat vette, hivatkozási paraméter átadásáról beszélünk. Így az utolsó példában A és B ún. hivatkozási (referencia) paraméterek.

A GRAFIKA

A bemutatókazetta használatakor nyilván észrevette, hogy milyen érdekes képek, szép rajzok készíthetők az Enterprise hatékony grafikai lehetőségeinek felhasználásával. A könyv első részében már megismerkedtünk a PRINT AT utasítás használatával, amely felosztja a képernyőt képzeletbeli négyzetekre. Ezekre koordináták megadásával hivatkozhatunk, amikor adott helyre kívánunk valamit kiírni. Így pl. a PRINT AT 1,1 utasítással szöveget irathatunk a képernyő bal felső sarkába.

A grafikus parancsok hasonló módszerrel pontokat, vonalakat írnak a képernyőre, amelyekkel diagramokat vagy képeket állíthatunk össze.

A következő rövid programrészlet egy vonalat rajzol.

```
100 GRAPHICS
110 PLOT 640,360; 1000,700
120 END
```

Az első sorban levő GRAPHICS utasítás gyors és egyszerű módszer egy olyan üres „lap" kiválasztására, amire rajzolni fogunk.

A PLOT parancs segítségével pontokat vagy vonalakat rajzolhatunk. Az ezt követő négy szám adja meg, hová kerüljön a rajzolandó vonal – ebben a példában a képernyő közepe (640,360) és a jobb felső sarok (1000,700) közé.

Figyeljük meg, hogy az itt használt számok nagyobbak, mint a PRINT AT utasításnál látottak. Ez nem azt jelenti, hogy így sokkal nagyobb területet jelöltünk ki, hanem azt, hogy a grafikus lapot sokkal több pontra kell felosztani, mint a szöveges lapot. Ezzel lényegesen pontosabban helyezhetők el a képrészletek a képernyőn, azaz grafikus módban a képernyő felbontása nagyobb.

Másik eltérés a szöveges módtól, hogy grafikus üzemmódban a bal alsó kiindulási pontot a (0,0) koordináták jelölik, és a koordináták első száma a vízszintes eltérést adja meg – hasonlóan a hagyományos (x,y) koordinátafelíráshoz.

Ha lefuttatjuk a programot, megfigyelhetjük, hogy a képernyő alján négy sor megmaradt az eredeti szöveges lapból. Így lehetőség van arra, hogy a rajz elrontása nélkül írjunk be valamit a számítógép számára.

A GRAPHICS utasítás hatására képernyőnk tehát két részre oszlik: egy felső, grafikus területre és az imént említett alsó négy szöveges sorra. A grafikus képernyő 1280 vízszintes és 720 függőleges pontra van felosztva – így a jobbfelső pont koordinátái (1279,719) lesznek.

A képernyő két része külön-külön letörölhető a CLEAR GRAPHICS, ill. a CLEAR TEXT utasítással, vagy egyszerre a CLEAR SCREEN paranccsal.

A DISPLAY TEXT parancs beírásával a teljes képernyőt visszaállíthatjuk szöveges módba; hasonlóképpen a DISPLAY GRAPHICS utasítás visszaállítja a grafikus képernyőt anélkül, hogy tartalmát megváltoztatná. Ezzel szemben a TEXT, ill. GRAPHICS parancsok úgy váltanak üzemmódot, hogy közben a megfelelő képernyőrészt letörlik.

Később, ha megismerkedünk a csatornákkal és a grafika kezelésének még bonyolultabb módozataival, lehetőségünk lesz arra, hogy a lapok méretét meghatározzuk és elhelyezzük a képernyő bármelyik részén.

PONTOK VAGY VONALAK

Most változtassuk meg programunk 110. sorát a következőképpen:

```
110 PLOT 100,100
```

Ha lefuttatjuk a programot, egy pontot látunk megjelenni a képernyőn. Tegyünk pontosvesszőt a 110. sor végére, és írjuk be a következő sort:

```
115 PLOT 1000,700
```

A program ezúttal egy vonalat rajzol.

Vegyük el a pontosvesszőt a 110. sor végéről, és két pont tűnik fel a képernyőn. Vajon miért?

A válasz a kérdésre a következő: a kettőspont arra utasítja a számítógépet, hogy hagyja a videosugarat ott, ahol éppen áll. Ha ez a sugár a képernyőn van, akkor látható vonalat húz két pont között. Ahhoz, hogy a sugarat ott tartsuk, pontosvesszőt kell tennünk a PLOT utasítás után.

Ezt a sugarat úgy képzelhetjük el, mint egy ceruzát. Egy PLOT utasítás elhelyezi a ceruzát a papíron, és legalább egy pontot rajzol is vele. Ahhoz, hogy vonalat rajzoljunk vele, meg kell adnunk két koordinátapárt, amelyeket pontosvesszővel választunk el. Így ez az utasítás a következőt jelentheti: „rajzolj egy pontot a (100,100) pozícióba, és ne emeld fel a ceruzát, hanem húzz egy egyenes vonalat az (1000,700) pontba". Ha kihagyjuk a pontosvesszőt, képzeletbeli ceruzánk mozog, de nem érinti a papírt.

A sugár ki- és bekapcsolására (a ceruza felemelésére, ill. letételére) használható a SET BEAM OFF, ill. a SET BEAM ON parancs.

Íme egy cikcakk program:

```
100 RANDOMIZE
110 INPUT POMPT „Hany cikcakkot
    parancsolsz ?":B
120 GRAPHICS
130 LET Z = 0
140 DO
150   LET X = RND( 1279 )
160   LET Y = RND( 719 )
170   PLOT X,Y
180   LET Z = Z + 1
190 LOOP UNTIL Z = B
200 END
```

Ez a program pontokat rajzol a képernyő véletlenszerűen választott pozícióiba. Ha most a 170. sort megváltoztatjuk:

```
170    PLOT X,Y;
```

akkor valóban cikcakkokat kapunk.

Természetesen a grafikus parancsok bármelyike előfordulhat függvény belsejében is. A következő példában a függvény pontokat helyez el a képernyőn, és össze is köti őket vonalakkal.

```
100 DEF DIAGRAM
110     GRAPHICS
120      PLOT 504,544; 564,464; 516,448;
        504,544; 460,464; 516,448
130 END DEF
```

Futtassuk ezt le, majd írjuk be parancs-módban, hogy CALL DIAGRAM.

Ha a koordinátapár után vesszőt teszünk, nem kerül pont erre a pozícióra; a számítógép odaállítja a sugarat és kikapcsolja. Később látni fogjuk, hogy ez a PLOT PAINT utasításhoz szükséges.

TEKNŐC PARANCSOK

A következőkben másik parancskészletet láthatunk vonalak rajzolására. Ezeket teknőc-parancsoknak nevezzük, mert először egy lassan mozgó állatrobot irányítására használták őket.

Ezúttal nincs szükségünk minden egyes képpont koordinátájának kiszámítására.

```
100 OPTION ANGLE DEGREES
110 GRRAPHICS
120 PLOT 300,150;
130 PLOT ANGLE 80;
140 PLOT FORWARD 500;
150 PLOT BACK 320;
160 PLOT RIGHT 35;
170 PLOT FORWARD 420;
180 PLOT BACK 285;
190 PLOT RIGHT 100;
200 PLOT FORWARD 340
210 END
```

Amint látható, ez csakugyan olyasmi, mintha egy angolul értő állatot irányítgatnánk a képernyőn.

A 100. programsor közli a számítógéppel, hogy a szögeket a továbbiakban fokban és nem radiánban kívánjuk számolni. (A radián a szögek mérésére használt mértékegység, 180 fok = PI radián, azaz körülbelül 3.1415 radián.) A 120. sor beállítja a sugár (ill. az állat) kezdő pozícióját – mivel a GRAPHICS utasítás a (0,0) pontra állította és kikapcsolta azt.

A következő PLOT ANGLE utasítás a következőt mondja a számítógépnek: „vedd úgy, hogy a sugár a képernyő jobb oldala felé irányul; majd fordítsd el az óra járásával ellenkező irányba annyi fokkal, amennyit megadtunk". Így pl. a PLOT ANGLE 90 hatására a sugár vagy az állatkánk egyenesen felfelé tartana.

A PLOT FORWARD, ill. a PLOT BACK parancsok hatására a sugár annyi pozíciót halad előre, ill. hátra, amennyit utánuk megadtunk, példánk 140. sorában tehát 500 pozíciót előre.

A PLOT RIGHT és a PLOT LEFT utasítások a megadott szöggel jobbra, ill. balra forgatják a sugár haladási irányát az előző irányhoz képest, programunk 160. sorában pl. jobbra 35 fokkal.

Figyeljük meg, hogy e teknőc-parancsok használata esetén is szükség van a pontosvesszőre ahhoz, hogy a ceruzát felemeljük vagy letegyük a papírra. Érdemes kipróbálni a programot a pontosvesszők nélkül is.

ELLIPSZISEK ÉS KÖRÖK

A következő program ellipszist rajzol:

```
100 GRAPHICS
110 PLOT 640, 250,
120 PLOT ELLIPSE 100,200,
130 END
```

A 110. sor az ellipszis középpontját adja meg. A PLOT ELLIPSE utasítást követő első szám a vízszintes, a második a függőleges irányban vett távolságot adja meg (képernyő-pozíció egységben) az ellipszis középpontja és körvonala között. Ha e két szám egyenlő, az utasítás kört rajzol.

Figyeljünk a 110. és a 120. sor végén található vesszőkre. Ha ezek bármelyikét elhagyjuk, az ellipszis középpontját pont fogja jelölni. Ebből következik, hogy az ellipszis rajzolása után a sugár a középpontban marad.

SZÍNEK

Már eddig is több lehetőségünk volt arra, hogy az Enterprise 256 színének legalább egy részét megcsodáljuk képernyőnkön. A következő példa valamennyi színt egyszerre tárja elénk:

```
100 ! ————————
110 GRAPHICS 256 ! Figyeljük a 256-os
120 ! szamot !
130 ! ————————
140 LET Z = 0
150 FOR Y = 0 TO 560 STEP 80
160   FOR X = 32 TO 1052 STEP 32
170     SET INK Z
180     PLOT X,Y; X,Y+70
190     LET Z = Z + 1
200   NEXT X
210 NEXT Y
220 END
```

Az Enterprise minden egyes színt azonosítószámmal (0-255) azonosít. Az RGB nevű, később ismertetendő függvény azonban megkönnyíti a színkódokkal való bánást.

SZÍNES ÜZEMMÓDOK

Az előző programban a GRAPHICS kulcsszót követően a 256-os számmal jeleznünk kellett, hogy a számítógép valamennyi rendelkezésre álló színét használni kívánjuk. Ezt úgy mondhatjuk, hogy kiválasztottuk a megfelelő színes üzemmódot.

Megfigyelhettük, hogy az ezzel a programmal húzott vonalak vastagabbak a korábban látottaknál. Ennek az az oka, hogy a számítógép korlátozott memóriakapacitása miatt több szín használatakor csökken a grafika felbontása.

Korábban azt írtuk, hogy egy grafikus lap nagyobb felbontású, mint egy szöveges. Most azonban a felbontás meghatározásához figyelembe kell vennünk, hogy melyik szín-üzemmódot alkalmazzuk. Négy ún. nagyfelbontású (HIRES: HIgh RESolution = nagyfelbontású) grafikus üzemmódot különböztetünk meg. Ezek a következők:

GRAPHICS HIRES 2 - e parancs kiadása esetén egyszerre csak két szín használható, de a képernyő szélességében 640 pontra van felosztva;

GRAPHICS HIRES 4 - négy színnel rajzolhatunk a rendelkezésünkre álló 320 pont szélességű képernyőn;

GRAPHICS HIRES 16 - ezúttal 16 szín és 160 pont/sor használható;

GRAPHICS HIRES 256 - a 256 szín megjelentetésének lehetőségéért azzal fizetünk, hogy képernyőnk vízszintes felbontása 80 pontra csökken.

Ha csak a GRAPHICS parancsot adjuk ki, ennek hatása megegyezik az

utoljára kiadott GRAPHICS parancs hatásával. A gép bekapcsolása vagy a reset kapcsoló használata után a GRAPHICS parancs hatása a GRAPHICS HIRES 4 parancséval egyezik meg.

A különböző színes üzemmódok nem hatnak a képernyő függőleges felbontására, ez a normál grafikus lapon, amelynek alján négy szövegsor van, mindig 180 pont.

A felbontásbeli különbségek ellenére a színes üzemmódok ugyanazt a koordináta-rendszert használják - így (0,0) mindig a bal alsó sarkot, (640,360) pedig a képernyő közepét jelöli. E koordináta-rendszer valójában akkora képernyőhöz is használható lenne, mint az Enterprise legnagyobb felbontásának kétszerese.

GRAFIKUS ÜZEMMÓDOK

Az eddig megismert nagyfelbontású (HIRES) üzemmódokon kívül ván két további grafikus üzemmód.

A kisfelbontású, LORES (LOw RESolution = kisfelbontású) üzemmód mindenben megegyezik a HIRES módokkal, de azzal a különbséggel, hogy a grafikus lap vízszintes felbontása fele akkora lesz. Így a GRAPHICS LORES 16 utasítás kis felbontású képernyőt jelent, amelyen 16 színt használhatunk.

Az attributum- (ATTRIBUTE) mód a képernyő felhasználásának különleges formája, valahol a grafikus és a szöveges mód között. Egyaránt használható karakterek kiírására és rajzolásra is 16 színnel, de elővigyázatos bánásmódot igényel.

A GRAPHICS ATTRIBUTE paranccsal állítható be, itt nincs lehetőség színes mód megadására.

Részletesen a későbbiekben ismertetjük.

SZÍNEK KIVÁLASZTÁSA

Térjünk most vissza ismét a 256 színes üzemmódhoz, ahol az ábrákat mindenféle színben megrajzolhattuk úgy, hogy a rajzoló utasítások előtt végrehajtottuk a SET INK parancsot, amelyet a kívánt szín kódszáma követett. Hasonlóképpen, a CLEAR GRAPHICS utasítás előtt kiadhatjuk a SET PAPER ... parancsot, ily módon állítva be a háttér (a papír) szinét.

Némi gyakorlatozás után megtanulhatjuk, hogy melyik szám milyen színt jelöl – pl. a 18 világoszöld, 91 a fényes sárga stb. De ha konkrét színt szeretnénk használni, és történetesen nem tudjuk a hozzá tartozó kódszámot, nem kell elcsüggednünk, van más mód is a megadására.

Minden létező szín előállítható a piros, a zöld és a kék keverékeként. A fehér pl. mindhárom összükeveréséből adódik, sárgát kapunk, ha pirosat és zöldet keverünk össze, és így tovább.

Ez az alapja annak a módszernek, ahogy az Enterprise kezeli a színeket. Ennek megfelelően tetszőleges szín megadható az RGB kulcsszóval és az azt követő, zárójelek között álló, három számmal. Ezek a 0 és 1 közé eső számok azt az arányt adják meg, hogy mennyi pirosat, zöldet, ill. kéket kell összekeverni.

Így SET INK RGB(1, .5, .5) rózsaszínűre állítja a rajz színét; RGB(.4, .4, 0) halvány sárgát, RGB(.6, .6, .4) pedig a szürke egy árnyalatát eredményezi.

A következő nyolc színre egyszerűen nevük leírásával is hivatkozhatunk (pl. SET INK GREEN). A következő táblázatban megadjuk a hozzájuk tartozó keverékeket:

BLACK = RGB(0,0,0) (fekete)
RED = RGB(1,0,0) (piros)
GREEN = RGB(0,1,0) (zöld)
YELLOW = RGB(1,1,0) (sárga)
BLUE = RGB(0,0,1) (kék)
MAGENTA = RGB(1,0,1) (lila)
CYAN = RGB(0,1,1) (világoskék)
WHITE = RGB(1,1,1) (fehér)

A PALETTA

A GRAPHICS HIRES 16 utasítás végrehajtása után 16 szín áll rendelkezésünkre, ez még mindíg elég nagy választék. Választani közülük paletta megadásával lehet; ez a paletta tartalmazza a rajzoláshoz használható valamennyi színt.

Mindenekelőtt be kell írnunk a SET PALETTE parancsot, ezt követően pedig legalább nyolcat az általunk használni kívánt színekből. Ezek a 256 lehetséges szín közül bármelyikek lehetnek. Megadhatjuk őket kódszámukkal, nevükkel (az előbbi lista szerint) vagy az RGB függvény segítségével. Írhatjuk pl. a következőt:

```
100 SET PALETTE 67, 31, WHITE, 4,
    RGB( 0,.3,.8 ), RGB( .7,.7,.1 ),
    187, 190
```

ez esetben a fenti színek palettánkon a 0–7 sorszámot kapják.

A hátralevő nyolc szín megválasztása már kevesbbé tetszőleges. A paletta 8–15 számú szineinek összefüggő színcsoportba kell tartozniuk. Ezeket a SET BIAS paranccsal választhatjuk ki, amelyet egy számnak kell követnie. E szám tetszőleges színcsoportot jelöl ki. Ha pl. a SET BIAS 67 (vagy bármely más szám a 64–71 tartományban) parancsot adjuk ki, akkor palettánk 8. színe a 64-es lesz, a 9. a 65-ös, és így tovább egészen 71-ig. A színcsoport tehát 8 egymást követő színt tartalmaz, amely nyolccal osztható színkódon kezdődik.

Így tehát kijelöltük azt a 16 színt, amelyet a továbbiakban használhatunk. Fontos tudnunk, hogy minden esetben, ha nem a 256 színes üzemmódot használjuk, a SET INK, ill. a SET PAPER utasításokban levő számok a választott paletta színeire vonatkoznak. E sorszámok 0-val kezdődnek, nem 1-gyel, mint a színek kódjai.

A négyszínű üzemmódban palettánk négy színe a 0–3 sorszámot viseli, így a SET PALETTE utasításban nincs értelme ennél több színt felsorolni. Hasonlóan a kétszínű üzemmódban a paletta megadásakor csak két színre van szükség.

Szükség lehet arra, hogy palettánknak csak egy színét változtassuk meg úgy, hogy közben a többi változatlan maradjon. Ez a SET COLOUR utasítással érhető el; pl. a paletta 3. színét a

```
SET COLOUR 3,11Ø
```

utasítással változtathatjuk meg.

Természetesen bármelyik üzemmódot használhatjuk anélkül, hogy palettát definiálgatnánk. Ilyenkor a számítógép bizonyos előre definiált színkészletet kínál számunkra.

A PALETTA HASZNÁLATA

Amennyiben palettánk színkészletét megváltoztatjuk, ez nem csak az ezt követő rajzok, hanem a már képernyőn lévő ábrák színében is változást okoz. Ez lehetőséget ad arra, hogy egyetlen utasítással megváltoztassuk a teljes képernyőt, ami felettébb látványos megoldásokat eredményezhet.

A következő program ellipsziseket rajzol véletlenszerűen választott sugarakkal és színekben, de mivel kezdetben palettánk minden színét egyformán kékre választjuk, a rajzolás nem fog megjelenni a képernyőn. Ha lenyomunk egy billentyűt, a rajzolás befejeződik, és a paletta színeinek váltogtásával programunk váltogatja a tinta és a háttér színét.

A program végtelen ciklusban fut, ezért megállításához nyomjuk meg a STOP billentyűt, ha meguntuk a látványosságot.

```
1ØØ RANDOMIZE
11Ø GRAPHICS HIRES 4
12Ø SET PALETTE BLUE,BLUE,BLUE,BLUE
13Ø ! ————————————
14Ø ! Lathatatlanna tettuk az
15Ø ! abrat !
16Ø ! ————————————
```

```
170 DO
180    SET INK RND * 3 + 1
190    PLOT 625,330,
200    PLOT ELLIPSE RND*500,RND*300,
300 LOOP WHILE INKEY$ = „"
310 ! ————————————
320 ! Egy billentyu lenyomasa utan
330 ! megmutatjuk a kepernyot.
340 ! ————————————
350 DO
360    SET PALETTE BLUE,BLUE,RED,GREEN
370    ! ———————
380    FOR X = 1 TO 500 ! kb. 1 mp-es
390    NEXT X ! varakozas
400    ! ———————
410    SET PALETTE BLUE,RED,GREEN,BLUE
420    FOR X = 1 TO 500
430    NEXT X
440    SET PALETTE BLUE,GREEN,BLUE,RED
450    FOR X = 1 TO 500
460    NEXT X
470 LOOP
480 ! ————————————
490 ! A program a „STOP" billentyuvel
500 ! allithato le.
510 ! ————————————
```

Megjegyezzük, hogy a PALETTE, INK és PAPER parancsok használhatók szöveges módban is. Ezt a későbbiekben fogjuk tárgyalni.

A PLOT PAINT PARANCS

Ezzel az utasítással az aktuális INK szinnel kiszínezhetünk egy zárt idomot. A következő program két kört rajzol, egyiket a másik belsejében, majd különböző színekkel kiszinezi őket.

```
100 GRAPHICS HIRES 4
110 SET PALETTE WHITE,YELLOW,BLUE
120 PLOT 400, 400,
130 PLOT ELLIPSE 200,200,
140 PLOT ELLIPSE 80,80,
```

```
150 SET INK 3
160 PLOT PAINT
170 PLOT 400,250,
180 SET INK 2
190 PLOT PAINT
200 END
```

A PLOT PAINT utasítás mindig azt a területet szinezi be, ahol a sugár éppen áll. Ezt a területet folytonos vonalnak kell körülvennie, amelynek színe nem azonos a sugár aktuális színével. Ha e vonalon bárhol rés van, a számítógép az egész képernyőt be fogja színezni. A 120., 130., 140. és 170. sorok vesszővel végződek, erre azért van szükség, hogy a sugár ne rajzoljon pontot az adott helyre. Ebben az esetben ugyanis a PLOT PAINT parancs ezt a pici pontot színezné be.

LINE STYLE ÉS LINE MODE UTASÍTÁSOK

A LINE STYLE utasítás lehetővé teszi különböző szaggatott vonalak használatát. Pl. a SET LINE STYLE 10 utasítás végrahajtása után minden vonal hosszú, kis közű vonalakból fog állni, egészen a LINE STYLE visszaállításáig. 14 féle vonaltipus lehetséges, ezeket 1-től 14-ig számozzuk.

A SET LINE MODE utasítás arra vonatkozik, hogy az egymást metsző vonalak hogyan viselkedjenek egymással szemben. 0-s módban (amely a gép bekapcsolásakor érvényes) a másodszor rajzolt vonal egyszerűen felülírja a már meglévőt, a további módokban (1–3) a tinták színei különböző módokon keverednek.

Részletes leírás a könyv utolsó fejezetében található.

LAPOK ÉS CSATORNÁK

Mindeddig abból a célból, hogy üres laphoz jussunk, amire rajzolni, ill. írni lehet, a GRAPHICS, ill. a TEXT parancsot használtuk. Az esetek nagy részében ez a lehetőség meg is felel igényeinknek, ám később igényelni fogjuk azokat a széles körű lehetőségeket, amiket a csatornák nyújtanak a lapok meghatározására. A csatornákat általában 1 és 100 közé eső számmal azonosítjuk.

A most következőkben ajánlatos figyelembe vennünk a Csatornák alfejezet, valamint a Referencia Kézikönyv vonatkozó részeit (pl. Video feltételek, Az OPEN kulcsszó).

Annak, hogy egy lapot egy csatornával azonosítsunk, két fő előnye lehet: (1) Több, különböző ábrát vagy szöveget tartalmazó lap tárolható segítségével a számítógép memóriájában, és ezek bármelyike megjeleníthető egyetlen paranccsal.

(2) Megadhatjuk a lap méretét. Így készíthető olyan rajz, amely (majdnem) az egész képernyőt kitölti (a legfelső, állapotkijelző sorba nem írhatunk); avagy takarékoskodhatunk a memóriával azáltal, hogy kisebb képméretet adunk meg. Megadható továbbá a kép vízszintes helyzete is, előfordulhat, hogy csak egy részén jelenik meg.

A következő példa kicsi szöveges lapot definiál és a képernyő közepére helyezi:

```
 9Ø SET BORDER CYAN
1ØØ SET VIDEO MODE Ø
11Ø SET VIDEO COLOUR Ø
12Ø SET VIDEO X 2Ø
13Ø SET VIDEO Y 1Ø
14Ø OPEN #1:"VIDEO:"
15Ø DISPLAY #1: AT 7 FROM 1 TO 1Ø
16Ø PRINT #1:"Ez egy kis lap..."
17Ø END
```

A 140. sorban az 1. számú csatornát hozzárendeltük az új képlapunkhoz. De ezelőtt ki kellett adnunk a videomód és a színmód beállító parancsokat.

A VIDEO MODE 0 egy 40 oszlopos szöveges; VIDEO MODE 1 grafikus, a VIDEO MODE 2 pedig 80 oszlopos szöveges képernyőt definiál.

A SET VIDEO COLOUR utasítás a szín-üzemmódot állítja be a következők szerint:

```
VIDEO COLOUR Ø – 2 szín mód;
VIDEO COLOUR 1 – 4 szín mód;
VIDEO COLOUR 2 – 16 szín mód;
VIDEO COLOUR 3 – 256 szín mód;
```

A szöveges képernyőnek mindig a VIDEO COLOUR 0 módban kell lennie, ez is lehetővé teszi néhány szín használatát.

Programunk 120. és 130. sora a lap méretét adja meg karakterpozíció egységekben. A 150. sor a lap felső részét (10 karaktersort) a képernyőre helyezi, a 7. sorban kezdődően. Ez esetben, mivel lapunk összesen 10 sor hosszúságú, a teljes lap látható lesz.

Figyeljük meg, hogy a 160. sorban levő PRINT utasításban fel kell tüntetni a csatorna számát (#1).

Definiálható olyan lap is, amelynek függőleges mérete nagyobb a képernyő méreténél. Egy lap max. 255 karakter-sor magas lehet.

A következő példaprogram 3 lapot definiál, és ezeket egyszerre a képernyő különböző részein jelenteti meg:

```
100 SET VIDEO MODE 0
110 SET VIDEO COLOUR 0
120 SET VIDEO X 42
130 SET VIDEO Y 8
140 ! -------
150 OPEN #1:"VIDEO:" ! szoveges lap
160 ! -------
170 SET VIDEO MODE 1
180 SET VIDEO COLOUR 3
190 ! -------
200 OPEN #2:"VIDEO:" ! 256 szin
210 ! -------
220 SET VIDEO COLOUR 1
230 ! -------
240 OPEN #3:"VIDEO:" ! 4 szinu
250 ! -------
260 DISPLAY #1: AT 9 FROM 1 TO 8
270 DISPLAY #2: AT 1 FROM 1 TO 8
280 DISPLAY #3: AT 17 FROM 1 TO 8
290 PRINT #1:"Ez szoveg."
300 SET #2:BEAM ON
310 PLOT #2: 100,100;
320 SET #3:BEAM ON
330 PLOT #3: 100,100;
340 END
```

Az előbbi program lefuttatása után vissza kell térnünk normál, szöveges üzemmódba ahhoz, hogy lássuk, amit begépelünk. Ehhez írjuk be a TEXT parancsot, vagy nyomjuk le az 5-ös funkcióbillentyűt.

Minden egyes laphoz önálló paletta tartozik, de a SET BIAS parancs az összes képernyőre vonatkozik.

A csatornák használata lehetővé teszi, hogy akár akkor is írjunk egy lapra, ha az éppen nem látszik a képernyőn. Szintén lehet karaktereket írni grafikus lapokra az aktuális sugárpozíciótól kezdve. Ha pl. a 3. csatorna grafikus lapként van megnyitva, a következőképpen írhatunk rá betűket:

```
PLOT #3: 640, 50
PRINT #3: „Udv !"
```

A KERET SZÍNÉNEK BEÁLLÍTÁSA

Számítógépünk lehetőséget ad arra, hogy a látható képernyőt körülvevő keret színét beállítsuk. Ez a SET BORDER paranccsal végezheyő, amely független a palettaszínektől, így hozzá önálló színkódszámot, színnevet vagy keverési arányt (RGB függvény) kell megadnunk. Így pl. a SET BORDER 255, a SET BORDER WHITE és a SET BORDER RGB(1,1,1) parancsok egyformán fehérre állítják a képernyő keretét.

Amennyiben a SET BORDER parancs kiadásakor a 101. csatorna – amely a normál grafikus képernyőt jelöli – nincs megnyitva, a parancsot megfelelő csatornaszámmal kell kiadnunk (erre a célra általában a 102. csatornát használjuk, amely a normál szöveges képernyőt jelöli). Pl.:

```
SET #102: BORDER 116
```

A KARAKTERKÉSZLET

Az Enterprise-nak meghatározott karakterkészlete van, amely az általánosan használt ún. ASCII készlet kibővített változata.

Ebben a készletben valamennyi karakternek 32 és 159 közötti kódszáma van. Egy karakterre hivatkozhatunk karakteres formájában (pl. PRINT"N") vagy kódja segítségével (PRINTCHR$(78)) egyaránt.

A számítógép ezekkel a kódokkal kezeli a karaktereket,és mindaddig, míg valami különleges kívánságunk nincsen, ennek megoldásával nem kell törődnünk. Rövid magyarázatként azt mondhatjuk, hogy ha pl. lenyomunk egy billentyűt – mondjuk az „a"-t – a gép jelzést kap, hogy egy billentyű le van nyomva. Ezután a lenyomott billentyű kódját – és nem az alakját – továbbítja egy másik részbe, amely értelmezi a leolvasott karaktert, és olyan ábrát állít elő, amely már kiírható a képernyőre.

Mindeddig azt tapasztalhattuk, hogy ha lenyomunk egy billentyűt, az ábra abban a pillanatban megjelenik a képernyőn. Elképzelhetjük tehát, hogy milyen gyorsnak kell lennie számítógépünknek ahhoz, hogy ezeket a bonyolult műveleteket egy pillanatnak tűnő idő alatt végrehajtsa.

A következő program kiírja számunkra a teljes ASCII karakterkészletet és egy beírt karakternek kiírja a kódját.

```
100 FOR N = 32 TO 159
110 !-----------------
120 ! Ez a ciklus kiirja a 128 elore
130 ! definialt karaktert - kihagyva
140 ! a 32 - nel kisebb kodu un.
150 ! kontrol-karaktereket
160 !-----------------
170    PRINT CHR$( N )
180 NEXT N
190 DO
195   DO
200      INPUT PROMPT "Irj be egy
         karaktert, és kiirom a kodjat!"
         :C$
210      IF LEN( C$ ) > 1 THEN
220        PRINT "Csak egy karaktert
           kernek !!!"
230      END IF
240   LOOP WHILE LEN( C$ ) > 1
250   PRINT C$;" ASCII kodja : ";
260   PRINT ORD( C$ )
270   !-----------------
280   ! Az ORD fuggveny egy karakter
290   ! kodjat adja meg
300   !-----------------
310   DO
320      INPUT PROMPT "Tovabbi
         karakterek ( i / n ) ?":A$
330   LOOP WHILE A$ <> "i" AND A$ <>
      "n"
340 LOOP WHILE A$ = "i"
350 END
```

Amint látható, az ORD a CHR$ függvény ellentéte. Az ORD a karakter ASCII kódját adja meg, míg a CHR$ az adott kódhoz tartozó karaktert.

SAJÁT KARAKTERKÉSZLET DEFINIÁLÁSA

Miután megismerkedtünk az ASCII kódokkal, legfőbb ideje arról is szólnunk, hogyan lehet olyan karakterkészletet adni a számítógépnek, amilyenre éppen szükségünk van. Minden egyes ASCII karakterhez tartozik ugyan egy előre meghatározott forma, mi azonban ezt a formát megváltoztathatjuk.

Képzeljük el, hogy minden egyes karakterábra kilenc lámpasorból áll, ahol minden egyes sorban nyolc lámpácska van. Ahhoz, hogy ezekből ábrát formáljunk, egyes lámpácskáknak világítani kell, másoknak nem. Számítógépünkben ezeknek a lámpácskáknak apró képpontok felelnek meg, amelyek olyan közel állnak egymáshoz, hogy ha két, egymás melletti pont be van kapcsolva (világít a lámpa), azok egybefüggőnek tűnnek. Egy karakter definiálása ezeknek a potoknak az elhelyezkedését adja meg, egy 8 * 9 -es mezőben.

Ahhoz, hogy ezt az információt számítógépünk számára érthetővé tegyük, a bekapcsolt pontokat 1-es, az üreseket 0-s számnak kell elképzelnünk. Így soronként nyolc számjegyből összeállítható egy karakter ábrája, ahogy azt a következő példában látni fogjuk.

```
100 NUMERIC N( 1 TO 9 )
110 ! ----------------
120 ! Egy normal karakter 8 pont sze-
130 ! les, es 9 pont magas. Igy min-
140 ! den sort egy 8 jegyu szam ir
150 ! le. Kilenc ilyen szam pedig
160 ! definialja a karaktert.
170 ! A kovetkezo sorok ezeket a sza-
180 ! mokat tartalmzzak.
190 ! ----------------
200 DATA 00111110
210 DATA 01000001
220 DATA 01010101
230 DATA 01000001
240 DATA 00100010
250 DATA 00010100
260 DATA 00001000
270 DATA 00000000
280 DATA 00000000
290 FOR SOR = 1 TO 9
300    READ N( SOR )
310 NEXT SOR
320 SET CHARACTER 63, BIN( N(1) ),
    BIN(N(2)), BIN(N(3)), BIN(N(4)),
    BIN(N(5)), BIN(N(6)), BIN(N(7)),
    BIN(N(8)), BIN(N(9))
330 ! ----------------
340 ! A BIN utasitja a szamitogepet,
```

```
350 ! hogy a kovetkezo szamokat bi-
360 ! naris szamkent kezelje.
370 ! ----------------
380 PRINT "?"
390 PRINT CHR$( 63 )
400 END
```

A 320. sorban van az a parancs, amely az információt tárolja az új karakter számára. Ezzel átdefiniáltuk a kérdőjelet! A 63 szám ennek ASCII kódja, és az egyesek és nullák jelentik a pontokat és üres helyeket, amelyek az új karaktert formálják. A 380. és a 390. sorban ezt az új kérdőjelet íratjuk ki. Érdemes megpróbálni, gépeljünk be közvetlenül egy kérdőjelet.

Ha egyszer egy program módosította a karaktereket, akkor mindaddig, amíg a CLEAR FONT parancsot ki nem adjuk, vagy a „reset" gombot kétszer gyors egymásutánban meg nem nyomjuk vagy a gépet ki nem kapcsoljuk, az új karaktereket iratháthatjuk ki a képernyőre.

A karakterek módosításával szinte új távlatok nyilnak meg előttünk, hiszen teljes karakterkészletünket gót vagy cirill betűssé alakíthatjuk, vagy a különböző játékokhoz édekes új alakzatokat teremthetünk és így tovább.

A ZENEI KÍSÉRET

A legtöbb programnak értékes kiegészítői lehetnek különféle hangeffektusok. Komoly, nagy programok is használhatnak hangjelzéseket (pl. figyelmeztetésre) a játékprogramok vonzerejét pedig a jó zenei aláfestés nagymértékben növelheti.

Az Enterprise lehetővé teszi, hogy a hangokat – akár sztereóban is – fejhallgatón vagy erősítőn keresztül hallgathassuk. Ha így használjuk, a beépített hangszóró a SET SPEAKER OFF pranccsal vagy a 7. funkcióbillentyűvel kikapcsolható.

A hangokat a SOUND és az ENVELOPE BASIC kulcsszavakkal lehet szabályozni. A SOUND parancs beállítja a hang kezdeti magasságát, max. erősségét, időtartamát stb. Az ENVELOPE (= boríték) kifejezés olyan utasítások halmaza, amelyek a hang kibocsátási idejére részletesen megadják a hangmagasság- és hangerőváltozásokat.

A SOUND UTASÍTÁS

Lássunk egy példát a SOUND utasítás használatára:

```
100 SOUND PITCH 40, LEFT 127, RIGHT
      191, DURATION 200, ENVELOPE 10
```

Példánkban a PITCH után álló szám megadja, hogy a hang milyen magasságon szólaljon meg. Ez a szám elvileg 0 és 127 közötti lehet, de mivel a 127-es hang már olyan magas, hogy szinte nem hallható, általában nem használunk a 83-asnál magasabb hangokat. E tartományon belül az egységnyi növekedés a hang magasságát fél hanggal emeli. A 37 PITCH-érték felel meg a normál C hangnak; ha nem adunk meg külön PITCH-értéket, a számítógép ezt használja alapértelmezésként.

A sor második eleme megadja, hogy a bal oldali hangszóró max. milyen hangerővel működhet, értéke 0 és 255 közötti lehet. A 0 megadása kikapcsolja a hangszórót, míg a 255 lehetővé teszi, hogy a számítógép az elérhető leghangosabban szóljon a bal hangszóróból. Ez utóbbi érték a hangerő alapértelmezése is.

Példánkban a LEFT 127 azt jelenti, hogy a bal oldali hangszórónk soha nem szól a max. hangerő felénél hangosabban.

Ehhez hasonlóan a RIGHT szóval adhatjuk meg a jobb hangszóró max. hangerejét.

Ez után a DURATION kulcsszóval beállíthatjuk a hang hosszát. Itt az egység a másodperc ötvened része – így példánkban a DURATION 200 kb. 4 másodperces hangeffektust jelent. Az időtartam alapértelmezése 50 egység, azaz kb. 1 másodperc.

Az ENVELOPE szó után álló szám egy ilyen számú ENVELOPE utasításra utal, amelyet a SOUND paranccsal összefüggésben használhatunk. Számítógépünk előre beépített ENVELOPE kifejezése a 255. számot viseli, ez egyben alapértelmezés. Az általunk készített ENVELOPE kifejezéseket a 0 és 254 közötti számok valamelyikével azonosíthatjuk.

Megjegyezzük, hogy a SOUND kulcsszót követő paraméterek tetszés szerinti sorrendben állhatnak. Rövidesen sor kerül annak a néhány további szónak a tárgyalására, amelyeket felvehetünk erre a listára.

Mivel valamennyi paraméternek van alapértelmezése, bármelyiket kihagyhatjuk, sőt az egyedülálló SOUND parancs is meg tud szólaltatni egy bizonyos hangot.

AZ ENVELOPE

Lássunk egy példát az ENVELOPE kifejezésre:

```
90 ENVELOPE NUMBER 10; 2,8,63,50;
   0,24,-16,100; -5,47,-39,50
```

A kulcsszót ismét meglehetősen hosszú paramétersorozat követi.

Ezek közül a NUMBER 10 az ENVELOPE azonosítója, amint azt előző példánkban láttuk.

Ezt pontosvessző követi, majd több, vesszővel elválasztott szám. Ezek a számok határozzák meg a hang jellemzőit az első fázisban, azaz teljes idejének az első részében (példánkban ez az első másodperc). A számcsoport első értéke azt adja meg, hogy ez alatt az idő alatt mennyit változik a hang magassága. A példában a hang az első másodpercben 2 félhangot emelkedik az eredeti hangmagassághoz képest (ha pl. -1.5 értéket adtunk volna meg, ez három negyedhanggal csökkentette, 0 pedig állandó értéken tartotta volna a hangmagasságot). A következő szám – példánkban a 8 – a bal hangszóró hangerőváltozását adja meg. Az ENVELOPE utasításban a hangerő egysége a SOUND utasításban megadott max. hangerő 64-ed része, a kiidulás pedig a SOUND utasításban megadott hangerő. Így tehát a 63-as szám mindig maximumra növeli, míg a -63 minimumra csökkenti a hangerőt.

Az időtartam kezdetén a hangnak természetesen semmiféle hangereje nincsen. Így példánkban az ENVELOPE első fázisában a hangerő nulláról indulva emelkedik addig, míg a maximális hangerő nyolcadát el nem éri.

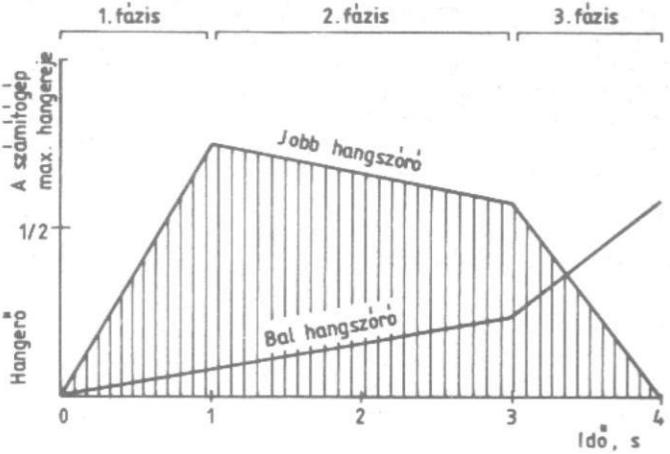
A következő szám hasonlóan adja meg a jobb hangszóró hangerejének változását.

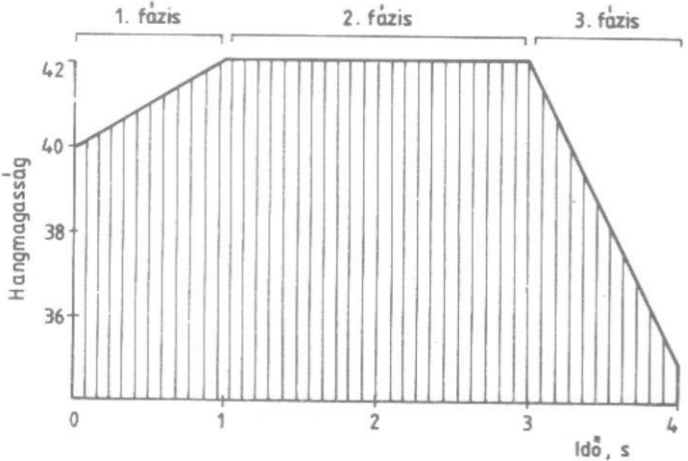
A SOUND és az ENVELOPE kifejezések együtt adják meg a hangerő értékét a jobb és a bal oldali hangszóróra külön-külön. E két érték összege adja azt a hangerőt, amelyet akkor hallunk, ha nem sztereo berendezést használunk.

A negyedik szám a sorban az első fázis időtartamát adja meg. Az idő egysége itt ismét a másodperc ötvened része.

Ezek után pontosvesszővel elválasztva következik a második fázis leírása pontosan ugyanúgy, ahogy ezt eddig láttuk. Majd újabb pontosvessző után a harmadik szakaszt leíró számcsoport fejezi be az ENVELOPE utsítást.

Ha lefuttatjuk példaprogramunkat, a következő diagramokon ábrázolható hangot hallhatjuk:


1. fázis
2. fázis
3. fázis
A számítógép max. hangereje
1/2
Hangerő
Jobb hangszóró
Bal hangszóró
0
1
2
3
4
Idő, s


1. fázis
2. fázis
3. fázis
42
40
38
36
Hangmagasság
0
1
2
3
4
Idő, s

Természetesen lehetőség van ennél sokkal bonyolultabb hangok megszólaltatására is. Rövidesen megvizsgáljuk, hogyan írhatunk le egy ENVELOPE-ban max. 255 fázist.

Emlékeztetünk arra, hogy az ENVELOPE utasítás valamennyi értéke a SOUND parancsban megadott értékekhez csatlakozik. Így egy ENVELOPE használható több SOUND utasításhoz is. Ügyelni kell azonban arra, hogy az ENVELOPE kifejezés a program olyan helyén legyen, hogy a legelső rá hivatkozó SOUND utasítás előtt végrehajtsuk.

A HANGOK SORBAN ÁLLÁSA

Egy hang kiadása nem tart fel semmiféle más tevékenységet a számítógépben. A következő programban grafikus parancsokat hajtunk végre, miközben angot bocsát ki a számítógép.

```
100 ENVELOPE NUMBER 20; -2,63,63,100;
     -3,0,0,100; 3,-36,-36,100; 2,-12,
     -12,100; 0,0,0,100
110 SOUND PITCH 61, LEFT 255, RIGHT 0,
     DURATION 500, ENVELOPE 20
120 CLEAR SCREEN
130 PRINT AT 10,5:"Most egy abrat
     rajzolunk,"
140 PRINT AT 11,5:"mely bemutatja a
     felhasznalt hangokat."
150 PRINT AT 12,5:"A hangerot kek, a
     hangmagassagot „
160 PRINT AT 13,5:"sarga szinnel
     jelezzuk."
170 SOUND PITCH 43,LEFT 127,RIGHT 127,
     DURATION 200, ENVELOPE 20
180 ! ————————————
190 ! A 170. sorban megszolatatott
200 ! hang az ENVELOPE elso ket
210 ! fazisaban szol csak.
220 ! ————————————
230 FOR X = 1 TO 2000
240 NEXT X
250 SOUND PITCH 25, LEFT 0, RIGHT 255,
     DURATION 500, ENVELOPE 20
260 GRAPHICS
270 PLOT 0,0; 250,540;500,540;750,270; 1000,
     180; 1250,180; 1250,0; 0,0
```

```
280 PLOT 100,20,
290 PLOT PAINT
300 SET INK 3
310 PLOT 0,450; 250,350; 500,100;1000,
      450; 1250,450
320 END
```

A 110. sorban a számítógép játszani kezdi a zenét, de nem várja meg, míg befejeződik, hanem továbblép a következő programsorokra. Ezeket végrehajtva elérkezik a 170. sorhoz, ahol újabb hang megszólaltatására adtunk utasítást. De mivel az előző hang még szól, a gép megjegyzi, hogy újabb zenei kérés érkezett, és ezt csak az első befejezése után kezdi játszani. Ehhez hasonlóan a 250. sorban kért hangeffektus is csak akkor szólal meg, ha a 170. sorban kért már befejeződött.

A SOUND utasítások tehát sorban állva kezdésre várnak, mialatt a gép tovább folytatja programját, azaz példánkban tovább rajzol.

A RELEASE KULCSSZÓ HASZNÁLATA

Az Enterprise lehetőséget ad arra, hogy egy ENVELOPE végén egy vagy több fázist a többitől kissé eltérő módon kezeljünk. Ezeket a fázisokat a RELEASE (= kioltás) szóval kell bevezetni. Általában egy ENVELOPE kioltási részének azt a szakaszt nevezzük, ahol a hang valójában már megszűnt (lecseng), már csak utóhangjai hallhatók.

Nézzük meg a következő példát:

```
100 ENVELOPE NUMBER 4; 1,63,40,5; -.3,
      -32,-20,20; 2,0,0,25; RELEASE; 0,
      -16,-10,10; 0,-15,-10,15;
```

Itt az ENVELOPE fő része 1, míg a RELEASE része fél másodpercig tart.

A RELEASE és a fő rész közötti különbség, hogy ha a SOUND utasításban megadott időtartam (DURATION) letelik, mielőtt a fő rész befejeződne, akkor megrövidül; ezzel szemben a RELEASE rész még akkor is elhangzik, ha a számára fenntartott idő letelik, feltéve, hogy nem áll sorban egy új SOUND utasítás.

Tehát ha a fenti ENVELOPE-ra hivatkozó SOUND utasításban DURATION 25-öt adtunk meg, akkor a harmadik fázis nem hangzik el, helyette a számítógép a RELEASE részre ugrik, vagy ha van várakozó hangkérés, akkor arra.

Ha viszont pl. DURATION 60-at adunk meg, a számítógép végrehajtja mindhárom fázist, majd az első RELEASE fázist, és ha ekkor van várakozó SOUND, akkor azzal folytatja a zenélést.

DURATION 100 esetén pedig valamennyi fázis elhangzik, és ezután legalább fél másodpercnyi szünet van a következő hang felharsanásáig.

MEGSZAKÍTÁS

Bár már sok érdekes dolgot megismertünk a hangadás vezérléséről, még mindig van néhány felfedezésre váró megoldás: lehetőség van pl. arra, hogy az egyik hang félbeszakítsa a másikat. Ha visszatérünk ahhoz a példához, amelyben a gép ábrákat rajzolt hangadás közben, kipróbálhatjuk ezt a lehetőséget is, ha a következő változtatásokat tesszük:

Töröljük ki a 130–160, 230–240 és a 260–310. sort, és írjuk be a következőket:

```
260 ENVELOPE NUMBER 30; 0,63,63,1; 0,
    0,0,24
270 PRINT „Nyomd meg az I billentyut,
    ha azt akarod, hogy megszakitsam
    ezt a hangot !"
280 DO
290 LOOP UNTIL INKEY$ = „I"
300 SOUND PITCH 37,LEFT 255,RIGHT 255,
    DURATION 25, ENVELOPE 30,INTERRUPT
```

A 300. sorban az INTERRUPT szó jelzi, hogy amikor a program ehhez a sorhoz ér, az éppen szóló zenébe belevágva meg kell szólaltatni ezt a hangot – figyelmen kívül hagyva minden sorban álló utasítást.

HANGFORRÁSOK

Az Enterprise lehetővé teszi, hogy egyszerre több hangot megszólaltassunk. Rendelkezésünkre áll a 0–3 számokkal azonosított négy hangforrás és mindegyik egymástól függetlenül sorbanállhat.

Alapértelmezésként mindeddig a 0-s hangforrást használtuk. Ahhoz, hogy meghatározott hanggenerátornak adhassunk parancsot, a SOUND parancsba be kell illesztenünk a SOURCE szót, amelyet a kívánt hangforrás száma követ.

A SYNC kulcsszó teszi lehetővé a különböző hangforrásokhoz rendelt zenék egyszerre indítását. A SYNC szót – amelyet szintén a SOUND parancsban adhatunk meg – követő szám azt jelzi, hogy hány másik hangforrásra kell várakoznunk.

Próbáljunk meg több SOUND parancsot beírni egy programba, megosztva őket a 0-s, 1-es és a 2-es forrás között. Elindítva a programot, a három hanggenerátorra több zene kérés fog várakozni.

Ha mindegyik hangforrásra vonatkozó legelső parancshoz hozzáírjuk a SYNC 2 parancsot, pl.

```
120 SOUND PITCH 40,LEFT 255,RIGHT 127,
    DURATION 200,ENVELOPE 6,SOURCE 1,
    SYNC 2
```

akkor elérjük azt, hogy a három hang azonos pillanatban szólaljon meg.

A „CLEAR QUEUE n" paranccsal megállíthatjuk az n-edik hangforrás működését, és kitörölhetjük valamennyi e forrásra várakozó SOUND kérést.

Megjegyezzük, hogy az INTERRUPT alkalmazása nem szakít félbe más hangforrásokból származó zenéket.

MÉG BONYOLULTABB HANGKELTÉS

Ha nem utasítjuk másképp, számítógépünk feltételezi, hogy egy ENVELOPE max. 20 fázist tartalmaz. Ha valami miatt ennél többre lenne szüségünk, pl. 25-re, azt a következő módon tehetjük lehetővé:

```
100 CLOSE #103
110 SET SOUND BUFFER 25
120 OPEN #103:"SOUND"
```

A 100. és a 120. sorban egyszer lezárunk, majd megnyitunk egy csatornát. Erre azért van szükség, mert a SET SOUND BUFFER utasítás csak az utána megnyitandó csatornákra vonatkozik, a már nyitva levőkre nem. A 103-as az általánosan használt hangcsatorna.

Végül a SOUND kifejezés tartalmazhatja a STYLE szót, amelyet követő 0 és 255 közé eső szám különféle hanghatások kiváltását teszi lehetővé. Ezzel kapcsolatban részletesebb információ a Referencia kézikönyv című fejezetben található.

A FELADATTÓL A KÉSZ PROGRAMIG

Az eddigiekben nem foglalkoztunk azzal, hogyan lehet megfogalmazni egy-egy feladatot úgy, hogy azt a számítógép megértse. Pedig nagyon fontos, hogy olyan programot készítsünk, hogy annak működését később is nehézségek nélkül megértsük. Ebben a fejezetben éppen ezt a kérdést tárgyaljuk.

A FELADAT

Foglalkozzunk most azzal a feladattal, hogy mennyi pénzt takaríthatunk meg azzal, ha leértékelésen vásárolunk bizonyos árukat.

E probléma megoldásához csak két dolgot kell tudnunk: az eredeti árat és azt a százalékot, amennyivel a bolt olcsóbban kínálja portékáját. E két számból könnyedén kiszámíthatjuk az árcsökkenés mértékét és azt, hogy valójában mennyit kell fizetnünk az áruért.

Ha programot akarunk írni, az első és legfontosabb feladat annak az eldöntése, hogy nagyon pontosan mit is akarunk csinálni. Esetünkben ez a program

1. beolvas két számot, amelyek az eredeti árat és a százalékot jelentik;
2. kiszámítja, hogy milyen összeg felel meg az ár ilyen százalékának;
3. kivonja a csökkenést az eredeti árból, hogy megkapjuk az új vételárat, és végül
4. közli velünk számításai eredményét, és megkérdezi, hogy akarunk-e más árakkal is számolni.

Ez eléggé egyszerű volt. A következő lépés annak eldöntése, hogy a program mindezt hogyan fogja végrehajtani.

Egy program elkészítésének mindig nagyon sok módja van. Mégis van néhány olyan általános szabály, amelyeket érdemes figyelembe vennünk.

MODULÁRIS PROGRAMOK

Egy igazán jól, szépen megírt program moduláris felépítésű – tehát jól elkülöníthető, egymással összekapcsolódó részekből áll. Nincsen értelme csak írni egymás után a BASIC sorokat, az ilyen, nem átgondolt módszer eredménye áttekinthetetlen, zavaros program lesz, amelyben rendkívül nehéz lehet megtalálni a hibákat.

Igyekezzünk olyan programot írni, amelyik a legkisebb erőfeszítéssel hajtja végre a feladatát, és működési módja bárki számára jól áttekinthető és világos.

A moduláris felépítésű program a következő részekből áll:

1. a globális változók deklarációja;
2. a fő- (vezérlő-) program;
3. az END utasítás, amely jelzi a számítógépnek, hogy a program futása befejeződik (nem kell, hogy ez egyben a program utolsó sora is legyen);
4. a különböző függvények, amelyek a feladat egyes részeit hajtják végre;
5. a DATA adathalmazok – ha egyáltalán használjuk őket.

Ha ilyen általános programszerkezetet követünk, a programozás kevésbbé időtrabló, kevésbbé fáradságos és lényegesen egyszerűbb.

A PROGRAM RÉSZEI

Elsőként a globális változókat kell deklarálni, azért, mert ha a számítógép egy olyan sort talál, amelyben egy még korábban értéket nem kapott változó van, előfordulhat, hogy hibát jelezve leáll. Ha viszont valamennyit a program legelején összegyűjtve deklaráljuk, akkor ez nem következhet be. A változók másik tipusát, a lokális változókat majd csak a függvényeikben deklaráljuk.

Másodiknak a program fő része következik, amelyet azért kell ide helyezni, hogy programunk legelején azonnal tisztázódjék, hogy a program mire jó és milyen az általános működése.

A főprogram nagyrészt alprogramok, függvények hívásait tartalmazza. A függvényt fekete dobozként képzelhetjük el, amelybe a főprogram adatokat ad be, és az új adatokat ad vissza. Minden egyes függvény kezelhető önálló programként is, tartalmazhatják további, alacsonyabb szintű függvények hívásait, igy programunk nagyon világos rendszerben építhető fel.

A DATA utasításoknak nem kell feltétlenül a program végén állniuk, de ajánlatos valamilyen ehhez hasonlóan kiemelt helyre tenni őket. Ez ugyanis egyszerűbbé teszi a hibás adatok keresését. DATA utasítások használhatók függvények belsejében lokális változóként is.

Végül ne felejtsünk el sok megjegyzéssort elhelyezni a programban, amelyek elmagyarázzák az egyes utasítások, változók feladatát, és segítenek a homály eloszlatásában, ha egy régebben írt programunkat vesszük ismét elő.

A FELADAT FOLYTATÁSA

Térjünk most vissza a korábban felvetett problémánkhoz.

Először is határozzuk meg a főprogram feladatát. Jelen esetben ez elég egyszerű: kérje meg a használót, hogy írja be az ár- és százalékadatot, ezeket az értékeket adja át a számítást végző függvénynek, majd közölje velünk a végeredményt. Végül kérdezze meg, hogy akarunk-e tovább dolgozni.

Ezután el kell döntenünk, hogy hány függvényre lesz szükségünk. Esetünkben, mikor mindkét szükséges számítás roppant egyzerű, egy függvény is elegendő lesz. Nevezzük e függvényt SZAMOLO-nak.

A következő lépés annak meghatározása, hogy a számítógép mi módon hajtsa végre a szükséges számításokat.

Példánkban az, hogy mennyi egy ár szorozva a százalékával, a következő képlettel számítható ki:

engedmény = (ár/100) * százalékérték

A fizetendő árat pedig úgy kaphatjuk meg, hogy a régi árból levonjuk az engedményt.

Ezek után már kigondolhatjuk, hogy milyen változókra lehet szükségünk; ezek a következők:

– az eredeti ár –> AR;
– az engedmény százalékban –> ESZ;
– az engedmény pénzben –> EP;
– az engedményes ár –> UJAR.

Mivel az első két változót INPUT utasítás kérdésére fogjuk beírni, ezért ezeket nem szükséges külön deklarálni. Azt tehát már tudjuk, hogy a program elején két változót kell deklarálnunk.

Az eddigiek alapján már megírhatjuk a függvényt, amely a következő lesz:

```
DEF SZAMOLO
   LET EP = ( AR / 1ØØ ) * ESZ
   LET UJAR = AR - EP
END DEF
```

Most, miután már tudjuk, hogy a programnak melyek a részei, és milyen sorrendben követik egymást, hozzáláthatunk a teljes BASIC program megírásához.

Ennél lényegesen bonyolultabb feladat esetén természetesen nem kezdhetnénk meg máris a konkrét programozást, nagy valószínűséggel előtte még rengeteg jegyzetpapírt teleírnánk az előző godolatmenet végigjárása közben.

Programunk tehát a következőképpen kezdődik:

```
1ØØ NUMERIC EP, UJAR
11Ø ! a foprogram kovetkezik
12Ø DO
13Ø  CLEAR SCREEN
14Ø  INPUT PROMPT "Ird be az arat :"
     :AR
15Ø  INPUT PROMPT "Ird be a szazalekos
     arengedmenyt :":ESZ
16Ø  CALL SZAMOLO
17Ø  CLEAR SCREEN
18Ø  PRINT "A ";AR;" Ft. arbol a ";
     ESZ;" szazalekos engedmeny "
19Ø  PRINT EP;" Ft. megtakaritast
     jelent."
2ØØ  PRINT
21Ø  PRINT "Az uj ar ";UJAR;" Ft.
     lesz."
22Ø  PRINT
```

```
230  PRINT
240  INPUT PROMPT „Akarsz uj arakkal
     szamolni ?":VALASZ$
250 LOOP WHILE UCASE$( VALASZ$( 1:1 ))
     = „I"
260 END
```

Ez volt a főprogram. Ezek után már csak a függvényt kell megírni:

```
270 DEF SZAMOLO
280     LET EP = ( AR / 100 ) * ESZ
290     LET UJAR = AR - EP
300 END DEF
```

Ezzel megoldottuk a fejezet elején felvetett problémát. Programunk működik, bár még találhatunk benne finomítandó részleteket.

Tehetünk bele pl. még néhány megjegyzést, amelyek elmagyarázzák a változók jelentéseit vagy az egyes utasítások feladatát.

Programunkat ezen kívül nem védi semmi hibás beírások ellen sem, amelyek pedig hibás működéshez, esetleg a program megszakadásához vezethetnek. Javasoljuk, hogy gyakorlásképpen egészítsük majd ki a programot olyan sorokkal, amelyek megakadályozzák, hogy a program hibás adat (pl. negatív százalékérték vagy a szám helyett betű) megadása esetén hibás műveleteket végezzen és megszakadjon a futása.

A MINIMÁL BASIC

Amint tudjuk, a BASIC nyelvnek – a beszélt nyelvekhez hasolóan – több változata, dialektusa van. Az Enterprise ezeknek a tájszólásoknak az egyikét beszéli. Ez a BASIC változat az eredeti BASIC nyelv jelentős kibővítésével jött létre.

Azért, hogy a különböző BASIC nyelvjárásokat értő gépek között mégis tudjanak programokat cserélni, kidolgozták a BASIC nyelv minimális változatát, amelyet a legtöbb mikroszámítógép megért. Ezek közé tartozik az Enterprise is.

Ebben az alfejezetben ismertetjük a BASIC minimális változatának azokat az elemeit, amelyekkel eddig még nem találkoztunk.

ELÁGAZÁSOK

Már jónéhány BASIC programmal találkoztunk e könyvben és még egyszer sem volt arra szükségünk, hogy egy programot ne sorban, az elejétől a vége felé haladva hajtsunk végre, hanem valami más, mesterkélt sorrend szerint. Ezt a lehetőséget, amely nemcsak egyszerüsíti a programírást, de a program áttekinthetőségét is óriási mértékben növeli, csak az Enterprise BASIC dialektusának köszönhetjük. A Minimál BASIC nyelvben óhatatlan, hogy a program végrahajtásának sorrendjét időnként elágazások szervezésével áttekinthetetlenebbé tegyük.

Ennek megvalósítására két utasítás használható: a GOTO és a GOSUB. Mindketten arra utasítják a számítógépet, hogy ugorjon a program egy másik részére.

A GOTO UTASÍTÁS

A GOTO utasítás után sorszám jelzi, hogy melyik sorban kell a program végrehajtását folytatni.

```
100 CLEAR SCREEN
110 INPUT PROMPT "Szereted a
      szamitogepeket ? ":V$
120 IF V$ = "I" THEN
130     GOTO 220
134 ! a program a 220. sorban
136 ! folytatodik
140 ELSE IF V$ = "N" THEN
150     GOTO 240
152 ! "N" valasz eseten ugras
156 ! a 240. sorra
160 ELSE
170     GOTO 110
180     ! ha ertelmetlen a valasz,
190     ! ujra beolvassuk.
200 END IF
220 PRINT "Akkor jo !!!"
230 END
240 PRINT "De engem ugye majd
      megszeretsz..."
250 END
```

Programunkból látható, hogy a GOTO utasítással egy programsorra hivatkozhatunk. Érdemes megfigyelni, hogy a 170. sor utasításával tulajdonképpen egy ciklust szerveztünk.

A GOSUB UTASÍTÁS

A GOSUB utasítás hasonló a GOTO-hoz annyiban, hogy ez esetben is egy sorszámmal adjuk meg, hogy programunkat melyik sorban akarjuk folytatni. A lényeges különbség az, hogy itt elvárjuk, hogy előbb-utóbb vissza is térjen.

Ahhoz, hogy a GOSUB utasítást használjuk, programunknak egy részét el kell különítenünk a többitől. Ez a rész – az ún. szubrutin – végrehajt egy adott feladatot, majd a végén levő RETURN utasítás hatására visszatér arra az utasításra, amely az őt hívó GOSUB-ot követi. (Tulajdonképpen a függvény is egy szubruin, amelyet speciálisan a CALL szóval hívunk.)

A GOSUB/RETURN utasításokkal kapcsolatban fontos szabályok a következők:

Először is, a GOSUB utasítást soha nem használhatjuk RETURN nélkül, mert különben a számítógép hibaüzenetet küld.

Másodszor vigyáznunk kell arra, hogy a szubrutin valóban különálló része legyen a programnak, és véletlenül, GOSUB utasítás nélkül, ne kerülhessen rá a vezérlés, ez ugyanis szintén hibához vezetne. Ezt úgy kerüljük el, hogy a szubrutin első sora elé mindig END vagy STOP utasítást helyezünk, amely megállítja a program futását.

A DIM KIFEJEZÉS

A Nagyobb mennyiségű adat kezelése c. alfejezetben láttuk, hogyan lehet tömböket definiálni, kihasználva az Enterprise BASIC nyelve által nyújtott sokoldalú lehetőségeket. A tömbök deklarálásának kevésbbé hatékony lehetősége a Minimál BASIC DIM utasítása.

A DIM A(10) kifejezéssel pl. egydimenziós tömböt deklaráltunk, amelynek elemeit a 0–10 index értékekkel azonosítjuk. A DIM B(4,4) kétdimenziós tömböt definiál, amelynek az elemeit az előbbi módszer szerint nullától számozzuk.

A (X TO Y) használatával azt a tartományt is megadhatjuk, amelyben az elemek sorszámai mozoghatnak, de nincs mód az elemek hosszának megadására, ami pedig rendkívül hasznos lehetne, különösen nagy programok esetében, ahol takarékoskodnunk kell a memóriával.

A következő programban példát láthatunk a DIM utasítás használatára:

```
1ØØ DIM TOMB$( 9 ) ! 1Ø elemu tomb
11Ø FOR N = Ø TO 9
12Ø    READ TOMB$( N )
```

```
130 NEXT N
140 FOR N = 0 TO 9
150   PRINT TOMB$( N );" ";
160 NEXT N
170 DATA Ez,egy,tomb,melyet,a DIM
180 DATA kifejezessel,deklaraltunk.
190 DATA Hat,nem,egyszeru ?
```

A CSATORNÁK

Azért, hogy a számítógépet és a hozzá csatlakoztatott további egységeket minél hatékonyabban kihasználhassuk, az Enterprise ún. csatornákat alkalmaz.

A csatorna speciális, a számítógép két része között nyitott út. Ha ilyen utat nyitottunk, a két részegység között egyszerűen a csatornaszám megadásával kommunikálhatunk. A csatornaszámot utasításokban a # vagy a £ karakter után kell megadni, aszerint, hogy német vagy angol nyelvű a számítógép.

Az Enterprise bekapcsolásakor minden olyan esetben, ahol erre lehetőség van, alapértelmezés szerint nyit bizonyos csatornákat. Ezért, ha begépeljük a PRINT „hello" parancsot, az üzenet megjelenik a képernyőn. Ha azonban azt akarjuk, hogy az üzenet a nyomtatóra kerüljön, használhatjuk a PRINT #104:"hello" parancsot. A számítógép bekapcsolásakor ugyanis a 104-es csatorna a nyomtatóhoz kapcsolódik.

Ezek mellett azonban lehetőségünk van arra, hogy igényeinknek megfelelően különféle csatornákat definiáljunk.

A következő programban változót használunk csatornaszámként, így a program maga döntheti el, hogy kimenete hova kerüljön.

```
120 INPUT PROMPT "Irj be egy szoveget"
      :A$
130 PRINT
140 PRINT "Hova irjam vissza ?"
150 DO
160   INPUT PROMPT "0 - kepernyo; 104
      - nyomtato ":CSAT
170 LOOP UNTIL CSAT = 0 OR
      CSAT = 104
180 PRINT #CSAT: ! ures sor a kep-
      ernyore vagy a nyomtatora
190 PRINT #CSAT:A$
200 END
```

Ez a program csak akkor működik helyesen, ha a számítógéphez bekapcsolt nyomtató csatlakozik. Érdemes azonban kipróbálni 104-es helyett a 101-üs csatornával is, amely a normál grafikus képernyőre vonatkozik.

Jegyezzük meg, hogy a számítógépben minden ki- és bemenet ilyen csatornák útján működik, beleértve a hanggal és a magnetofonnal kapcsolatos tevékenységeket is. Ha tehát meggondolatlanul bánunk a csatornákkal, előfordulhat, hogy meghökkentő dolgokat tapasztalunk. Ezek nem okozhatják a számítógépünk meghibásodását, de az megtörténhet, hogy csak a gép ki- és újra bekapcsolásával állíthatjuk vissza a helyes működést.

A csatornákkal kapcsolatban további részletek a Referencia kézikönyv c. fejezetben találhatók.

KIVÉTELES ESETEK, HIBÁK KEZELÉSE

Hibák kezelésére, a hálózattal vagy néhány más eszközzel végzett munkára és kivételes eset kezelésére az Enterprise-nak bizonyos speciális módszerei vannak.

Kivételes esetnek nevezzük az olyan folyamatokat, amelyek a programtól teljesen függetlenül mennek végbe, de valami módon arra befolyással lehetnek. Így a STOP billentyű lenyomása kivételes eset, mert a programnak nem kötelessége a STOP billentyűt figyelni, de lenyomása hatással lehet rá.

Ha futó programban olyan hiba van, amely a program megállását okozza, akkor a számítógép egy üzenetet és egy számot ír ki. Ez esetben a szám a kivételes eset tipusát adja meg.

A BASIC lehetővé teszi, hogy saját magunknak definiáljunk kivételes eseményeket, és megadjuk, hogy bekövetkeztük esetén mi a teendő.

A kivételes eseteket kezelő műveletek olyanok, mint a függvények, azzal a különbséggel, hogy általában nem speciálisan erre szolgáló utasítással hívjuk meg őket, mint amilyen a CALL.

A következő program kivételes eset kezelésére mutat példát.

```
50 WHEN EXCEPTION USE HIBA
60 ! ----------------
65 ! A fenti sor utasitja a gepet,
70 ! hogy kiveteles esetben a HIBA
75 ! nevu kezelot hasznalja.
80 ! Ez az utasitas a 145. sorban
85 ! levo END WHEN elereseig
90 ! ervenyes.
95 ! ----------------
100 INPUT PROMPT "Irj be egy szot:"
    :SZO$
```

```
110 IF VAL( SZO$ ) <> 0 THEN
115 CAUSE EXCEPTION 10
120 ELSE
130   PRINT „A szot elfogadom."
140 END IF
145 END WHEN
150 END
160 ! ————————————
170 ! A program kovetkezo resze a ki-
180 ! vetelés eset kezelese. Ese-
190 ! tunkben a 10 szamu kivetel allt
200 ! elo, ha szo helyett egy szamot
210 ! irtunk be.
220 ! ————————————
230 HANDLER HIBA
240   IF EXTYPE = 10 THEN
250      PRINT „Ez nem szo !!!"
260   ELSE IF EXTYPE <> 10 THEN
270      EXIT HANDLER
280   END IF
290 END HANDLER
```

Ebben a példában nyilván egyszerűbb megoldást is alkalmazhattunk volna.

A CAUSE EXCEPTION utasítás aktivizálja a kivételes eset kezelőt (a handlert) olyan hiba esetén, amelyet a gép magától nem venne észre, mint pl. esetünkben akkor, ha számmal kezdődő szót írunk be.

Az EXIT HANDLER hasonlít az EXIT DO, ill. az EXIT FOR utasításhoz, de természetesen a kezelőből való kilépést eredményezi.

Az EXTYPE változót követő érték a kivételes eset tipusát jelzi CAUSE EXCEPTION használata esetén. Az EXTYPE értékeit különböző hibaüzenetek esetére a Referencia kézikönyvben c. fejezetben láthatjuk. Ebben a példában a CAUSE EXCEPTION utasításban adtuk meg a hiba típusát.

Egy másik változó, az EXLINE megadja, hogy melyik sorban történt a kivételes esemény.

Fenti programunkban a kivételes eset kezelése nagymértékben hasolított egy függvény hívásához. Fontos tény ezzel kapcsolatban, hogy a WHEN EXCEPTION USE szavakat használtuk (kapcsolódva egy END WHEN kifejezéshez), amely a következőt jelenti: „bármilyen tipusú hiba esetén ezt és ezt a kezelőt használd!".

A HÁLÓZAT

Az Enterprise egyszerű kábelhálózat felhasználásával képes arra, hogy más számítógépekkel társalogjon.

Természetesen, ha a többi számítógép is fel akarja használni a számára küldött információt, akkor azoknak is rendelkezniük kell ezzel a szolgáltatással, amelyet Intelligens Hálózatnak (Intelligent Net) nevezünk.

Ha több számítógép van a hálózatba kapcsolva és az egyik társalogni kíván egy másikkal, akkor eközben – a telefonhálózathoz hasonlóan – a többi elnémul.

Számítógépünk nem rendelkezik előre megadott hálózati számmal, mint egy telefonkészülék; mikor összeállítjuk a hálózatot, nekünk kell e számot beállítanunk pl. a következő módon:

```
SET NET NUMBER 5
```

A hálózati szám 1 és 32 között bármi lehet. Az ASK NET NUMBER paranccsal lekérdezhetjük a saját aktuális sorszámunkat.

PROGRAMOK ÁTVITELE

Ha számítógépünknek már van hálózati sorszáma, akkor a programok átvitele rendkívül egyszerű.

A LOAD „NET-0:" parancs utasítja gépünket, hogy a hálózat bármely tagjától vegyen át programot. Ha viszont egy adott géptől várunk egy programot, a LOAD „NET-n:" parancsban levő „n" értékkel választható ki a hálózat egy bizonyos tagja, ahol n természetesen a kiválasztandó gép hálózati száma.

Teljesen hasonlóan adunk át egy programot, ha pl. a 17. számú számítógépnek a SAVE „NET-5:" parancsot adjuk, akkor az az 5. számú gép részére elküldi a memóriájában lévő programot. Az 5. számú gépnek ilyenkor a LOAD „NET-17:" paranccsal kell várakoznia a program megérkezésére.

ÁLTALÁNOS ADÁS

A 0-s hálózati sorszám speciális eset, ez nem adható egyetlen gépnek sem, hanem általános hálózati műveletekre van fenntartva.

Így 0-s címet megadva a teljes hálózatra küldhető általános adás, amelyet minden hálózatra kapcsolt gép figyelhet.

Ez a lehetőség nagyon hasznos lehet rövid, mindenkinek szóló üzenetek küldésére. Hátránya azonban, hogy sokkal kevésbé megbízható, mint a

meghatározott címre szóló átvitel. Ez utóbbi esetben ugyanis az adó gép addig küldi újra és újra az üzenetet, míg a címzett nem nyugtázza, de az általános adásnál ilyesmire természetesen nincs lehetőség.

Hasonlóan nyithatunk egy csatornát „NET-0:" módon, ekkor ez a csatorna minden általános adást vesz a hálózaton, valamint veszi a neki szóló adásokat is. Ez utóbbi üzenetet gépünk akkor is venni tudja, ha erre előzetesen nem is készítettük fel egy csatorna megnyitásával. Ha adott gép felé külön csatornát nyitottunk, e gép adásait akkor is ezen a csatornán keresztül fogjuk venni, ha azok általános adások voltak, és volt nyitott általános csatornánk.

KOMMUNIKÁCIÓS CSATORNÁK

A hálózat magasabb fokú kihasználása végett nyithatunk egy csatornát kifejezetten egy adott géppel való kapcsolattartás céljára. A csatorna megnyitása a szokásos, pl.:

```
OPEN #11Ø:"NET-17:"
```

Ezzel megnyitottuk a 110. csatornát a 17. számú számítógéppel való kommunikáció céljára.

Ezek után e csatorna a normál ki/beviteli utasításokon keresztül használható, pl.:

```
PRINT #11Ø:"Uzenet a 17.
    szamitogepnek !"
    LINE INPUT #11Ø:A$ ! A$-t a 17.
    geptol fogjuk venni.
```

A hálózatban küldött adatokat a számítógépek pufferelik, azaz adás előtt, ill. után a memóriában tartják. Ezért gyakran van szükség néhány speciális utasítás használatára.

Általában egy adatcsomagot a számítógép mindaddig nem küld el, míg az el nem éri a 256 karakter hosszúságot. Ezen változtatni a FLUSH #csat paranccsal lehet; ennek hatására a „csat" csatornához tartozó puffer tartalmát haladéktalanul elküldjük, függetlenül attól, hogy esetleg csak néhány karakter hosszú.

A CLEAR #csat:NET paranccsal a „csat" csatornához tartozó puffert törölhetjük. Erre azért lehet szükség, mert a számítógép azért, hogy az adatok ne kavarodjanak össze, mindaddig nem fogad el semmilyen üzenetet, míg az adott csatornához tartozó puffer nem teljesen üres.

A hálózati műveletekhez a 100 feletti sorszámú csatornákat használjuk (de az alapértelmezésbelieket nem – l. az OPEN utasítás ismertetését a Referencia kézikönyvben). Amikor ugyanis a BASIC rendszer törli a program változóinak értékét, lezárja az 1–99 számú csatornákat, és ez a művelet parancs módban igen gyakran előfordul (pl. a RUN utasítás mellékhatása).

HÁLÓZATKEZELÉS A HÁTTÉRBEN

Az Enterprise operációs rendszere a hálózat kezelésére az ún. megszakításos rendszert alkalmazza, amely azt jelenti, hogy a számítógépek a használó számára nem látható időben társalognak egymással.

A kivételes esetek kezelésével a BASIC is nyújt némi lehetőséget arra, hogy ilyen műveletekkel foglalkozhassunk.

Ha ugyanis ezt a futó BASIC programunkban a SET INTERRUPT NET ON utasítással engedélyeztük, akkor a hálózatról érkező üzenet programunkban kivételes esetet okoz, melyet a megfelelően elkészített kezelővel feldolgozhatunk.

A kezelőben végig kell kérdeznünk a csatornákat az ASK NET CHANNEL utasítással, hogy tartalmaznak-e adatokat. Mivel egyszerre több csatorna is tartalmazhat adatokat, az ASK NET CHANNEL utasítást mindaddig meg kell ismételni az adatok továbbítását követően, míg az a 255-ös értékkel nem tér vissza, jelezve, hogy nincs több begyűjtésre váró csatorna. A SET INTERRUPT NET ON utasításnak a kezelőből való kilépés előtti utolsó utasításnak kell lennie. A kivételes esetek kezelését programvezérelt megszakítások vezérlik, így ezek csak futó program esetén hatásosak, közvetlen parancsok végrehajtásakor nem.

A GÉPI KÓD HASZNÁLATA

Az Enterprise agya egy Z8Ø-as tipusú mikroprocesszor, amely mintegy 5ØØ különböző utasítás végerhajtására képes. Ezen utasítások mindegyikét egy (esetleg több) kódszám jelöli, az utasítás gépi kódja. Ha egy processzort saját gépi kódján programozunk, akkor közvetlenül, és nem a BASIC értelmezőn keresztül használjuk.

Két fő oka lehet annak, hogy BASIC programunkba gépi kódú rutinokat építünk. Vagy nagy sebességet szeretnénk elérni, vagy az Enterprise olyan hardver adottságát akarjuk kihaszálni, ami a BASIC-en keresztül nem volna lehetséges.

A gépi kódú programozás önmagában is igen nagy téma, e könyvben nincs lehetőségünk ezzel foglalkozni. A téma iránt érdeklődők számos ilyen témájú kiadvány közül válogathatnak.

Az Enterprise BASIC nyelvének több olyan funkciója van, amely segíti gépi kódú programrészek használatát.

AZ ALLOCATE PARANCS

Mindenekelőtt helyet kell foglalnunk a gépi kódú program számára a memóriában. Ezt az ALLOCATE parancs teszi lehetővé, amelynek formája a következő:

```
ALLOCATE lefoglalandó-byte-ok-száma
```

Mivel e parancs végrehajtása az összes BASIC változó értékét elrontja, az ALLOCATE utasítást csak programok elején szabad használni.

CODE ÉS HEX$

A gépi kódú rutin a CODE utasítás segítségével tárolható a memóriában úgy, hogy nevet adunk neki. E név formailag ugyanolyan lehet, mint egy változó neve.

A CODE utasítás csak karakterláncot tud a memóriába küldeni, ezért a gépi utasítások hexadecimális értékeit a HEX$ függvény segítségével konvertálhatjuk a megfelelő formába.

Íme egy rutin, amely megkettőz egy számot:

```
KETT: 29 ADD HL,HL
      C9 RET
```

Ezt a BASIC programba így illeszthetjük be:

```
100 ALLOCATE 2
110 CODE KETT=HEX$( „29,C9" )
```

Figyeljük meg, hogy a HEX$ függvényben a hexadecimális értékek idézőjelek között állnak és vesszőkkel választjuk el őket egymástól.

A fenti programrészlet végrehajtása után rutinunk a memóriában található.

A GÉPI KÓDÚ RUTIN VÉGREHAJTÁSA

A gépi kódú rutinok – hasonlóképpen a BASIC-ben definiált függvényekhez – kétfélék lehetnek, és meghívásuk módja is a típusuktól függ.

Az egyik típus az, amelyiknek visszatérő értéke, és bemenő paramétere van. Ilyenkor a

```
USR(név,paraméter)
```

utasítást használjuk a meghívására.

Az átadott paraméter a HL regiszterpárba kerül, és visszatéréskor e regiszterpár tartalma adja a függvény értékét.

Pl. a

```
PRINT USR(KETT,4)
```

utasítás 8-ast ír a képernyőre, a

```
LET A = 3 * USR(KETT,3)
```

hatására A változó értéke 18 lesz.

Ha rutinunk nem ad vissza értéket, a

```
CALL USR(nev,Ø)
```

formában hívhatjuk meg. Ez esetben is kell paramétert írnunk a rutin neve után, de ennek semmilyen szerepe nincs.

A WORD$ függvény

A WORD$ függvény az argumentumát két byte-os karakterlánccá alkítja a Z8Ø-as mikroprocesszor által értelmezett formában (LSB, MSB: azaz elöl áll a kisebb helyiértékű byte). Pl. a

```
WORD$( KETT )
```

függvény értéke a KETT rutin kezdőcíme lesz olyan formában, ahogyan azt számítógépünk gépi kódú ugrásaiban értelmezni tudja.

4. REFERENCIA KÉZIKöNYV

A BASIC SZABÁLYAI

Ez a fejezet a BASIC nyelv, pontosabban az Enterprise által használt változatának utasításait tárgyalja részletesen. Természetesen lesznek köztük már ismert parancsok is. Azok számára, akik korábban már használták a BASIC nyelvet, könnyen elég lehet, hogy csak ezt a fejezetet kell átnézni a gép használatához. Érdemes az új utasításokat parancsmódban kipróbálni.

A könyv korábbi fejezeteiben gondosan ügyeltünk arra, hogy az általános szakkifejezéseket csak fokozatosan kezdjük használni. Ebben a fejezetben már sokkal bátrabban alkalmazzuk őket, hogy elkerüljük a túl terjengős fogalmazást. Az esetleg ismeretlen szavak magyarázatát a fejezet végén levő szójegyzékben találhatjuk meg.

ÁLTALÁNOS SZABÁLYOK

A kis- és nagybetűk tetszés szerint használhatók a kulcsszavakban és az azonosítókban, tehát pl. a FOR, for, For szavak hatása megegyezik. Az azonosítók legfeljebb 31 karakter hosszúak lehetnek, de a gép minden betűjüket figyelembe veszi. Tartalmazhatnak betűket, számjegyeket, pontot és aláhúzásjelet, az első karakterüknek azonban betűnek kell lenni. Egy programsor max. 250 karakterből állhat és a sorszámának 1 és 9999 közé kell esnie. A sor több, egymástól kettősponttal elválasztott utasítást is tartalmazhat.

A sorokban levő „!" jelek azt jelzik, hogy a sor további része megjegyzésnek számít. Ha parancsmódban egy sort kettősponttal kezdünk, akkor a gép a sor további tartalmát anélkül, hogy megpróbálná végrehajtani, átadja az operációs rendszernek.

A gép a sorszámokat megelőző és követő szóközöket törli, majd a sorszám után automatikusan annyit helyez el, hogy a blokkoknak megfelelő tördelés alakuljon ki. A FOR, DEF, DO, HANDLER, SELECT és WHEN blokkok utasításai, ill. az ELSE és a CASE ágak 2-2 karakterrel beljebb kezdődnek, mint az őket megelőző sorok. A blokkok végét a LOOP, END, NEXT szavakról ismeri fel a gép. A sorszámokat mindig oszlopba rendezve kapjuk.

```
  1 LET A=Ø
 1Ø   DO WHILE A<1Ø
1ØØ     LET A=A+1
11Ø     SELECT CASE A
12Ø     CASE 1
```

```
  130      PRINT „eloszor"
  140    CASE ELSE
  150      PRINT „nem eloszor"
  160    END SELECT
  170    PRINT A
  180  LOOP
  190  GOTO 1
 1000  END
```

Az Enterprise legtöbbször az ANSI X3J2/82-17 „Draft Proposal for Standard BASIC"-nak megfelelően jár el.

TÖBB PROGRAM KEZELÉSE

Az Enterprise egyszerre több programot is tárolhat, amelyeknek saját sorszámaik és önálló változóik lehetnek. Mindegyik programra egy számmal, ill. a PROGRAM utasításban megadott névvel hivatkozhatunk (l. még: CHAIN, EDIT, PROGRAM parancsok). A legtöbb parancs mindig az aktuális programra fejti ki hatását (ez alapértelmezésben a nullás), amelynek a számát az állapotjelző sorban is olvashatjuk. A nullás program max. 42, a többi legfeljebb 32 kbyte helyet foglalhat el a tárban.

KITERJESZTÉSEK

Léteznek olyan programok, amelyeket betöltve korábban nem ismert utasításokat is végre tud hajtani a számítógép. Ezeknek az utasításoknak a leírását a programokhoz mellékelt tájékoztató mindig részletesen ismerteti, ezért azt gondosan tanulmányozzuk át használat előtt.

ADATTÍPUSOK

Két adattípus van, a szám és a karaktersorozat. A változók neveire ugyanazok a szabályok érvényesek, mint bármely más azonosítóra, de a karakterváltozók neve végén egy dollárjelnek kell állnia. A számváltozók tartalmát binárisan kódolt decimális alakban tárolja a gép tíz számjeggyel. A legkisebb, még kezelhető szám az 1e-64, a legnagyobb a 9.999999999e62. A karaktersorozatok hossza max. 254 karakter lehet, de a karakterváltozókat kisebbre is definiálhatjuk a STRING utasítással. A részsorozatokra változónév$(x:y) alakban hivatkozhatunk, ahol x az első, y az utolsó karakter sorszáma a sorozatban. Ha x vagy y elmarad, akkor a sorozat elejét, ill. végét használja helyette a gép.

Az ismert műveleti jelek:
* - szorzás;
/ - osztás;
↑ - hatványozás;
+ - összeadás;
– - kivonás.
Karaktersorozatokra vonatkozó műveleti jel:
& - konkateráció: összekapcsolás.
Relációs (összehasonlító) műveleti jelek:
> - nagyobb;
< - kisebb;
= - egyenlő;
>= - nagyobb vagy egyenlő;
<= - kisebb vagy egyenlő;
<> - nem egyenlő;
AND - logikai ÉS;
OR - logikai VAGY;
BAND - bináris ÉS;
BOR - bináris VAGY.

RÖVIDÍTÉSEK

A kulcsszavak leírása közben a következő rövidítéseket használjuk:
csat - csatornaszám;
azon - azonosító;
str - fűzér, azaz karaktersorozat;
vált - változó;
kif - kifejezés;
relop - relációs műveleti jel;
para - paraméter;

PARANCSOK ÉS UTASÍTÁSOK

Sorszám

sorszám szöveg
sorszám szóköz
sorszám

Egy új sor ad hozzá a programhoz vagy töröl a programból. Ha a sorban csak egy szóköz van, akkor automatikusan „!" jel kerül az elejére. A magában álló sorszám a megfelelő sort törli. Csak parancsmódban adható ki. Törli a változókat.

Jegyezzük meg, hogy azok a parancsok, amelyek törlik a változókat, automatikusan lezárják az 1–99 között levő nyitott csatornákat.

ALLOCATE

ALLOCATE kif

A gépi kódú rutinokkal kapcsolatban használatos. A kifejezésnek megfelelő helyet hagy ki a programban a rutin számára és az üres hely elejére állítja az elhelyezésszámlálót. Törli a változókat, ezért csak a program elején használható.

ASK

ASK gépi feltétel vált

A változóba teszi a megadott gépi feltétel értékét. (Ezekről bővebben A gépi, A video- és A hangfeltételek c. alfejezetben olvashatunk majd.) L. még: SET, TOGGLE utasítás.

AUTO

AUTO
AUTO AT 100 STEP 10
AUTO STEP 10

Speciális, csak parancsmódban alkalmazható utasítás, amely bekapcsolja az automatikus sorszámozást. Alapértékek: kezdő sorszám = 100, lépesköz = 10. Az új sorszámú sorok felülírhatják a régieket. A szolgáltatást a STOP billentyűvel kapcsolhatjuk ki.

CALL

CALL függvény
CALL függvény(paraméterlista)

A beépített vagy felhasználó által definiált (DEF) függvények hívásának egyik módja. E megoldás esetén a gép a függvény értékét kiszámítja, de nem használja fel. CALL USR(A,B) pl. egy gépi kódú rutin meghívása. Parancsmódban is alkalmazható.

CAPTURE

CAPTURE FROM #csat TO #csat

Az első csatornáról érkező inputtal helyettesíti a másik csatornáról érkezőt. A második csatorna olvasása mindaddig áll, mig le nem ütjük a STOP gombot, az első csatornán end-of-file jelzés érkezik, vagy hiba keletkezik. A parancs hatását később úgy tudjuk kikapcsolni, hogy ugyanezt az első csatornát adjuk

meg ugyanilyen parancsban, de második csatornának a 255-öst írjuk. A 255 csatorna más használata hibát okoz.

CASE

L. SELECT blokk

CAUSE EXCEPTION

CAUSE EXCEPTION kif

A megadott számú hiba bekövetkezését okozza, a hibaszám 1–99 között lehet. Az EXCEPTION szó elhagyható.

CHAIN

CHAIN programszám

CHAIN „név"(paraméterlista)

Az aktuális programból egy másik programot hív meg, a paramétereket az új program kapja meg. L. még: PROGRAM.

CLEAR

CLEAR #csat:
CLEAR ENVELOPE
CLEAR FKEYS
CLEAR FONT
CLEAR GRAPHICS
CLEAR #csat:NET
CLEAR QUEUE hangforrás-szám
CLEAR SCREEN
CLEAR SOUND
CLEAR TEXT

Törli a megadott feltételeket. Parancsmódban is végrehajtható.

CLOSE

CLOSE #csat

Kiírja a pufferben levő adatokat, lezárja a csatornát, és felszabadítja a puffert.

CODE

CODE=str
CODE vált-név=str

A gépi kódú rutinokkal kapcsolatban haználatos. A karaktereket sorra beírja az elhelyezésszámlálóhoz és annak értékét betölti a változóba (ha az adott). Az utasítás végrehajtása után az elhelyezésszámláló az első szabad címre mutat. A változó később a rutin hívásához vagy ugrási címként használható fel.

CONTINUE

Parancsmódban a STOP billentyűvel megállított programot indítja tovább. Program közben a különleges eseteket kezelő rutinokból, ún. handlerekből való visszatérésre használható, ilyenkor a gép a mögé az utasítás mögé tér vissza, ahol a végrehajtás megszakadt.

COPY

COPY FROM #csat TO #csat

Az első csatornáról másol a másodikra. Mindkét csatornának nyitva kell lennie. A művelet a STOP gomb, az end-of-file jel vagy valamilyen hiba hatására fejeződik be. Ha az egyik csatornát nem adjuk meg, akkor az alapértelmezés a beolvasásra a 0., a kiírásra a 104. csatorna használható, azaz a rendszercsatornáról olvas és a sornyomtatóra ír ki.

DATA

DATA adatlista

A READ utasítással olvasható adatok itt helyezhetők el. L. még: READ.

DATE

DATE „eeeehhnn"

A gép által tárolt dátumot állítja be, amely automatikusan változik, ha a tárolt idő eléri az éjfélt. Parancsmódban is használható. L. még: DATE$ függvény. A megadás formája a nemzetközi szabványnak felel meg, tehát pl. 1987.jan.31. a DATE 19870131 paranccsal állítható be.

DEF

DEF szám-vált = kif
DEF szám-vált(paraméterek) = kif
DEF str-vált = str-kif
DEF str-vált(paraméterek) = str-kif
Két felhasználása létezik, az első a közvetlen függvény megadása.
DEF atlag(x,y) = (x + y)/2

A második a DEF blokk, ami olyan programrészlet, amit a nevével lehet meghívni és a lefutása után egy értéket ad vissza a saját neve helyett. L. még: CALL, EXIT DEF, NUMERIC és STRING.

A struktúrája a következő:

```
def-sor
    néhány utasítás vagy belső blokk
end-def-sor
def-sor:
DEF szám-vált
DEF szám-vált(paraméterek)
DEF str-vált
DEF str-vált(paraméterek)
end-def-sor:
END DEF
```

A visszaadandó értékre a blokkon belül a blokk nevével hivatkozhatunk.

```
DEF valasz(a$)
    IF UCASE$(a$(1:1))="Y" THEN
        valasz=1
    ELSE IF UCASE$(a$(1:1))="N" THEN
        valasz=0
    ELSE
        valasz=-1
    END IF
END DEF
```

A változók hatásköre dinamikusan változik, tehát függ a korábban lefuttatott soroktól és nem függ az utasítások programlistabeli helyétől.

```
100 NUMERIC FRED
110 LET FRED=1
120 CALL Q
130 PRINT FRED
140 END
200 DEF P
210   LET FRED=123 !ez a kulso FRED
220 END DEF
300 DEF Q
310   NUMERIC FRED !ez a helyi FRED
320   LET FRED=0
325 CALL P
330   PRINT FRED
350 END DEF
```

A 210. sorban nem a 100. sorban megadott, hanem a 310. sor FRED nevű változója kap új értéket. A program a 123 és az 1 számot írja ki, pedig a listabeli

sorrend alapján 0-t és 123-t várnánk. Minden a blokkon belül definiált változó lokális (helyi) és csak a blokk futása alatt létezik és kaphat értéket. Ha egy korábban már használt változónevet újra deklarálunk, akkor a blokkon belül az új, azon kívül a régi változót érhetjük el. A külön nem definiált változók lokális vagy globális (nem helyi) volta csak a programban korábban végrehajtott utasításoktól függ, ezért ajánlatos a blokkban használt helyi változókat a blokk elején definiálni.

```
100 CALL P !ez a hivas a P lokalis I-jet valtoztatja,
110 LET I=9
120 CALL P !ez meg a globalis I-t
130 END
200 DEF P
210    LET I=6
220 END DEF
```

Ha azt akarjuk, hogy mindkét hívás ugyanarra az I-re vonatkozzon, akkor szúrjuk be a következő sort:

```
90 NUMERIC I
```

Mivel a lokális változók csak a blokk futása közben foglalnak el helyet a tárból, ezért a tártakarékosság miatt is célszerű lehet használatuk.

A hétköznapi paraméterek csak az értékűket adják át a blokkon belüli változóknak, de az ezekre vonatkozó blokkon belüli utasítások nem tudják az eredetí változókat megváltoztatni. Az ún. referenciaparaméterek értéke is megváltozhat a blokkban. A következő program pl. először a 23-t és csak aztán a 99-t írja ki.

```
100 DEF CSERE(REF A,REF B)
110    NUMERIC T
120    LET T=A
130    LET A=B
140    LET B=T
150 END DEF
200 LET X=99
210 LET Y=23
220 CALL CSERE(X,Y)
230 PRINT X,Y
```

A tömböket és a függvényeket mindig REF paraméterként kell átadni. A következő program 66-ot és 0.5-et ír ki.

```
100 NUMERIC A(10)
110 OPTION ANGLE DEGREES
120 DEF P(REF FN,X)
130    PRINT FN(X)
140 END DEF
150 LET A(2)=66
```

```
160 CALL P(A,2)
170 CALL P(SIN,30)
```

A függvények rekurzívan is hívhatják egymást vagy magukat.

DELETE

DELETE sormeghat TO sormeghat
DELETE sormeghat-sormeghat
DELETE blokknév

Törli a megadott programsorokat. Csak parancsmódban használható. Törli a változókat.

DELETE LAST
DELETE FIRST TO 100
DELETE 1 TO 99,300,500 TO 9999

A TO szócska helyettesíthető a „–"jellel is. Ha valamelyik paramétert elhagyjuk, akkor a program első, ill. utolsó sorát használja helyette a gép. Egy definiált függvényblokk sorait a függvény nevére kiadott DELETE-vel tudunk törölni. Az üres DELETE parancs az egész programot törli.

DIM

DIM tömblista

Szám- vagy karaktertömböket definiál. Az alsó indexhatár alapértelmezése 0. Max. kétdimenziós tömbök megengedettek. A karakterláncok hossza itt nem adható meg, hanem az alapértelmezés szerinti 132-t használja. (L. STRING.) A következő tömbök mind tízelemesek:

DIM a(1 TO 10),FRED$(9),B(-7899 TO -7890)

DISPLAY

DISPLAY #csat: AT a FROM b TO c

Egy grafikus vagy szöveges lap egy részét megjelenítő ablakot definiál. Az ablak első sora a képernyő a-adik sorában lesz, és az ablakban a lap b-edik sorától c-edik soráig terjedő része látható. A sorokat karaktersorban mérjük. (L. még: PRINT.)

DISPLAY GRAPHICS

A felső 20 sort nyitja meg a legutóbb használt grafikus lapnak, ha van már ilyen megnyitva (pl #101). A szöveglapot nem törli.

DISPLAY TEXT

A teljes képernyőt nyitja meg szöveges lapnak (#102). A grafikus lapot nem törli. Ha csak kicsi szöveges lap volt eddig nyitva, akkor megnyit egy teljes képernyős lapot.

DO struktúra

```
do-sor
    néhány utasítás vagy blokk
loop-sor
do-sor:
DO
DO WHILE relációs kifejezés
DO UNTIL relációs kifejezés
loop-sor:
LOOP
LOOP WHILE relációs kifejezés
LOOP UNTIL relációs kifejezés
```

A DO és a LOOP sor közti rész a ciklustörzs. A DO és a LOOP nem lehet feltételes sorban. A ciklus belsejébe nem ugorhatunk be kívülről. (L. még:EXIT DO.)

```
DO WHILE A>3 AND A<10
    LET A=A+1
    PRINT A
LOOP
```

EDIT

```
EDIT programszám
EDIT "név"
```

Csak parancsmódban használható. A megadott prgramot teszi az aktuális programmá, tehát ezután erre vonatkoznak az olyan parancsok, mint pl. a LIST, RENUMBER. (L. még:CHAIN, INFO, PROGRAM.)

ELSE

```
L.:IF
```

END

A program logikai végét jelzi, a program futása ennél az utasításnál ér véget. De END DEF, END HANDLER, END IF, END SELECT és END WHEN a nekik megfelelő blokk végét jelzik.

ENVELOPE

```
ENVELOPE #csat:NUMBER a;b,c,d,e;...
...;RELEASE;j,k,l,m;...
```

Egy hangburkolót határoz meg, amelyet majd a SOUND utasítás használ. A hangburkolót azonosító „a" számnak 0-254 között kell lennie. A b,c,d,e

paraméterek egy hangfázist határoznak meg, ahol „b" a hangmagasság félhangokban,"c" és „d" a bal és a jobb oldali hangszóró hangerejét adja a max. hangerőhöz viszonyíva, végül „e" a fázis időtartama 1/50-ed másodpercekben. A „b" és a „c" 0–63-ig terjedhet. A -63 kikapcsolja a hangot, az értelmes tartományon kívül levő értékek pedig nem változtatják meg. A hangburkoló elején a hang ki van kapcsolva. Mono hallgatáskor a két csatorna összege szól.

Egymás után több fázis adható meg, amíg a hangpuffer be nem telik. (L.: SOUND BUFFER a hang opcióknál.)

A RELEASE szó és az azt követő fázisok elhagyhatók. Ezek nem csak akkor szólalnak meg, ha az előttük levők véget értek, hanem ha a SOUND utasításban megadott időtartam lejárt, és nincs ugyaezen a csatornán más következő hang.

EXIT DO
EXIT FOR
EXIT DEF

A megfelelő blokkból ugrik ki. A blokkon kívül kiadva hibát okoz.

EXIT HANDLER

Egy rendkívüli esetet kezelő „handler" működését szakítja meg és ezzel újabb rendkívüli esetet okoz, amit vagy a felhasználó által megírt, vagy a beépített handler fog kezelni.

EXT

EXT paraméter-str

A parancsban levő karaktersorozatot adja át a külső programnak, vagy ha ilyen nincs, akkor az operációs rendszernek. Az első szó ezért általában valamilyen parancs vagy egy új program neve szokott lenni. Pl. a következő utasítás a szövegszerkesztőbe ugrik a programból.

EXT "WP"

A HELP szó hatása itt különleges, ez ugyanis az összes külső programmal kiíratja a nevét az alapértelmezésnek megfelelő rendszerű csatornára. (L.:DEFAULT CHANNEL.)

Ha a HELP szó után külső program nevét is megadjuk, akkor a program a paramétereit is listázza: EXT"HELPNÉV". Parancsmódban nem az EXT szót kell használnunk, hanem kettőspontot kell a sor elejere írnunk és a karaktersorozat idézőjeleit elhagynunk a parancs használatahoz:

:HELP
:HELP név
pl.
:HELP BASIC
:HELP WP
:WP

FLUSH

FLUSH #csat

A pufferben levő adatokat küldi ki a csatornára – kiüriti a puffert – anélkül, hogy lezárná a csatornát vagy end-of-file jelet küldene. Csak a lokális hálózat bizonyos eszközeinél használatos. Parancsmódban is kiadható.

FOR

A struktúra:

for sor

néhány utasítás vagy blokk

next sor

for sor:

FOR egyszerű változó = kif TO kif STEP kif

Ha nem adunk meg STEP részt, a lépésköz alapértelmazésben 1.

next sor:

NEXT

NEXT valt

A FOR és a NEXT nem lehet feltételes sorban.

FOR Y = 0 TO 10 STEP 2

PRINT Y

NEXT Y

A ciklusváltozó értéke a ciklus lefutása után a (végérték + lépésköz) lesz, az előbbi példában 12.

Az egymásba ágyazott ciklusok nem használhatják ugyanazt a ciklusváltozót. A végérték és a lépésköz a ciklus elején tárolódik, így azt a ciklusban már nem lehet megváltoztatni. Tilos a ciklusba kívülről beugrani. (L. még:EXIT FOR.)

GET

GET #csat:str-valt

Egyetlen karaktert olvas be a megadott csatornáról. Ha nincs a csatornában karakter, akkor üres karaktersorozatot ad vissza. A csatorna alapértelmezése a 105 (KEYBOARD:, billentyűzet), így hatása hasonló a „LET str-változó = INKEY$"-hoz.

GOSUB

GOSUB sorszám

Meghívja a megadott sorral kezdődő programrészletet (szubrutint).

GOTO

GOTO sorszám
A program végrehajtása a megadott sornál folytatódik.

GRAPHICS

GRAPHICS
GRAPHICS HIRES/LORES színállapot szám
GRAPHICS ATTRIBUTE

A GRAPHICS parancs lezárja és újra megnyitja az alapértelmezés szerinti grafikus és szöveges lapokat (#101, #102) úgy, hogy a képernyő felső húsz sorát rendeli a grafikus laphoz.

A lehetséges színállapot számok a 2,4,16 és a 256. Ha sem ezt, sem a HIRES/LORES opciók egyikét nem adjuk meg, akkor az előző GRAPHICS parancs paramétereit használja újra a gép. A paraméterek jelentését a VIDEO MODE pontnál találhatjuk a Videofeltételek alfejezetben. Alapértelmezésben a GRAPHICS nagyfelbontású (HIRES) képernyőt nyit 4 színnel.

GRAPHICS ATTRIBUTE az ún. attributum-üzemmódba juttatja a grafikus lapot, ahol minden színcellának (8 pont széles és 1 pont magas) egy-egy tinta-, ill. papírszín felel meg. Ez az üzemmód 16 színű palettával még ugyanazt a felbontást adja, mint a négyszínes HIRES grafika. (Ilyenkor a felbontást és a színállapotszámot nem mi adjuk meg.) Az ilyen lapra írni és rajzolni is tudunk. (L. még: ATTRIBUTES video opció és a DIPLAY GRAHICS parancs.)

HANDLER

HANDLER handler-név
handler utasításook
END HANDLER

A handlerblokkok – továbbiakban megszakításkezelők – az olyan különleges problémákat kezelik a program futása során, mint a hibák, a CAUSE EXCEPTION utasítások által, ill. a gép által generált megszakítások. Az ilyen eseményekkor használandó megzakítaskezelő nevét a WHEN blokk fejlécében adhatjuk meg. (L. még: CONTINUE, RETRY, EXIT HANDLER utasítások és az EXLINE, ill. EXTYPE függvények.) A megszakításkezelők nem hívhatók meg GOSUB vagy GOTO utasítással, csak a CAUSE EXCEPTION-nal.

Ha a blokk végrehajtása közben újabb megszakítás következik be, akkor a következő külső megszakításkezelőé a vezérlés – hasonlóan az EXIT HANDLER utasítás hatasához –, de az EXLINE és az EXTYPE értéke természetesen megváltozik. Külső megszakításkezelő az, amelyiket a jelenlegi WHEN blokkot tartalmazó WHEN blokkban jelöltünk ki.

IF

IF relációs-kifejezés THEN sorszám
IF relációs-kifejezés THEN egyszerű utasítás
Az IF utasítás sorában nem lehetnek a következő utasítások: DATA, DEF, END, DIM, NUMERIC, STRING, egy másik IF vagy bármely olyan utasítás, amihez blokk tartozik.

```
IF A>=3 AND A<=9 THEN 100
IF A>=3 THEN GOTO 100
```

A másik if-struktúra az if-then-else-end elágazási redszer:

```
if sor
utasítások vagy blokkok
else if sor
utasítások vagy blokkok
else sor - (nem kötelező)
utasítások vagy blokkok
end if sor
```

if sor:
IF relációs kifejezés THEN
else if sor:
ELSE IF relációs kifejezés THEN
Az else if ágból több is lehet egymás után.
else sor:
ELSE
end if sor:
END IF
Az ilyen if blokkok bármilyen, programban végrehajtható utasítást tartalmazhatnak. A blokk belsejébe nem ugorhatunk be kívülről, csak az if soron keresztül.

```
IF A<10 THEN
  PRINT A
ELSE IF A>30 AND A<=40 OR A>50 THEN
  PRINT A+100
ELSE
  PRINT B
END IF
```

IMAGE

IMAGE: formátummegadás
A PRINT utasításokkal kapcsolatban a kiírási formát irányítja. A formátummegadás a következő vezérlőkarakterekből állhat:

, - írj vesszőt a számba
$ - írj vezető dollárjelet az előjel elé
– - írj vezető szóközt vagy - előjelet
+ - írj + vagy – előjelet
% - írj egy számjegyet, vezető nullákat is
# ill. £ - írj egy számjegyet, vezető nullákat elhagyva
* - írj egy számjegyet vagy egy vezető * jelet
. - írj decimális pontot
↑ - írd ki az exponenst, minimum négy karakteren
Ha a szám nem felel meg a formátumnak, akkor ez hibaüzenetet okoz.
Karaktersorozat formátum karakterek:
< - A # ill. jelekkel megadott mezőben levő karaktersorozat balra igazítása
> - Ugyanennek jobbra illesztése
ill. £ - írj ki egy karaktert
Az illesztést meghatározó karakternek a mező előtt kell állnia, ha nem adjuk meg, akkor a mező tartalma középre állítódik. Az IMAGE utasításban a formátummeghatározás a : jeltől kezdve a sor utolsó karakteréig tart.

INFO

Információt ír ki a képernyőre a tár foglaltságáról és a tárban levő programokról. Törli a változókat. Csak parancsmódban használható. Az információ formája:

programszám programméret a program első sora
(byte-okban) (sorszám)

INPUT

INPUT #csat, IF MISSING sorszám, AT sorszám-kif, oszlopszám-kif, PROMT str: valt-lista

Adatokat olvas a megadott csatornáról a változókba. Alapértelmezés a 0 csatorna (EDITOR:). A változólista elemei között vesszőknek kell állniuk. Az INPUT kulcsszón és a változólistán kívül minden más elhagyható.

A PROMT után megadott karaktersorozat az alapértelmezésként használt „?" jelet váltja fel. Az AT után megadott sor-, ill. oszlopszám független a PROMT résztől. Az IF MISSING-ben meghatározott tevékenységre akkor jut a vezérlés, ha a csatonán end-of-file (adatvég) jel vagy hibás típusú adat érkezik. (L. még: LINE INPUT.)

LET

LET változólista = kifejezés

Értékadó utasítás. Ha a változó neve nem kulcsszó, akkor a LET elhagyható, de a program listázásakor a számítógép automatikusan kiírja. Parancsmódban is használható. Egyszerre több változó is kaphat értéket:

```
LET a,b(4),c,=0
A.VAR=A.VAR+1
A$,FRED$="Azt mondja,"&"FRED$(i:j)
LET INPUT=3
```

LINE INPUT

Hasonló az INPUT-hoz, de ezzel csak karaktersorozatokba lehet beolvasni, és egy teljes sort egyetlen karakterláncba olvas be.

LIST

```
LIST #csat:sor-meghat TO sor-meghat
LIST #csat:sor-meghat - sor-meghat
LIST blokknév
```

A programot vagy annak egy részét listázza ki alapértelmezésben a 0 csatornára. A STOP billentyűvel megszakítható, a PAUSE-vel pedig felfüggeszthető. A TO kulcsszó helyettesíthető a – jellel. Ha valamelyik sorszámot elhagyjuk, hatása hasonló a DELETE utasításnál leírtakhoz.

```
LIST 300
LIST 300 TO 400
LIST FIRST TO 900,1000,2000 TO LAST
LIST TO 500,700 TO
LIST Eljaras
LIST LAST
LIST FIRST–100,500–LAST
LIST FIRST TO 100,500 TO LAST
```

LLIST

Hasonló a LIST-hez, de az alapértelmezés ennél a 104 (PRINTER:) csatorna.

LOAD

```
LOAD #csat:file-név
LOAD eszköznév
```

Az adott csatornáról olvas be egy file-t, alapértelmezés itt a 106 csatorna (lemez vagy magnetofon). Ha nem adunk meg file-nevet, akkor az első file-t olvassa be. Ha a file BASIC programot tartalmaz, akkor az aktuális program helyére tölti be, ha egyszerre több programot tartalmaz, akkor az összes korábbi

programot törli, és helyettük beírja az újakat, majd a nullás programot választja aktuálisnak. A file más adatokat, ill. különleges (pl. gépi kódú) programokat is tartalmazhat, ezeket az operációs rendszer kezeli. Törli a változókat. Csak parancsmódban adható ki, programutasításként azonos hatású a RUN-nal.

Az eszköznevet és a file-nevet az OPEN parancsnál ismertetjük részletesen.

LOOK

LOOK #csat AT x,y:v

A v változóba tölti az x,y pont színének palettaszámát. A csatorna alapértelmezése a standard grafikus csatorna. Ha nem adunk meg pozíciót, akkor a rajzolósugár aktuális helyét – továbbiakban rajzolópont – használja helyette. Ha megadunk egy pozíciót, akkor a rajzolósugár kikapcsolódik, és a rajzolópont új helye a megadott lesz.

LOOP

L.: DO

LPRINT

Hasonló a PRINT-hez, csak az alapértelmezés itt a 104 csatorna (PRINTER:).

MERGE

Hasonló a LOAD-hoz, de nem törli a korábbi programokat. Az új program(-ok) sorai felülírhatják a régieket. Csak parancsmódban alkalmazható. Törli a változókat.

NEW

Törli az aktuális programot. Csak parancsmódban használható. Törli a változókat.

NEW ALL

Az összes programot törli a tárból és a 0-ás szintet teszi aktuálissá.

NEXT

L.: FOR

NUMERIC

NUMERIC vált/tömb-lista

Szám típusú változók és tömbök deklarálására alkalmazható, amelyek egy DEF blokkon belül lokálisak lehetnek. Az alsó indexhatár alapértéke 0. L.: DIM

NUMERIC i,a(10),b(-10 TO 20,2 TO 4)

ON

ON kif GOTO sorszám-lista

ON kif GOSUB sorszám-lista

Kiszámítja a kifejezés értékét és egésszé alakítja át. Ha a kifejezés értéke N, akkor a sorszámlistából az N-edik sorszámot véve ott folytatja a programot (a számlálás 1-től kezdődik). Ha a listában nincs N-edik elem, akkor az utasításnak nincs hatása. Ennél célszerűbb a SELECT vagy az IF utasítás használata.

ON A+2 GOTO 100,200,300,400,99,700

OPEN

OPEN #csat:NAME eszköz/file-név ACCESS mód

OPEN #csat:eszköz/file-név

Az ACCESS után megadott mód (elérési mód) lehet INPUT vagy OUTPUT. OUTPUT hatására új file-t nyit meg, INPUT hatására egy már meglevőt próbál használni, az alapértelmezés itt INPUT.

A parancs egy eszközt vagy file-t és egy csatornát rendel egymáshoz, úgy, hogy ettől kezdve a különböző parancsok a csatornára hivatkozva tudnak a file-ba írni vagy onnan olvasni. Egy csatornához egyszerre csak egy eszköz, ill. file rendelhető. Az egymáshoz rendelési viszonyt a CLOSE paranccsal szüntethetjük meg. A csatornák számozása 0–254 közötti, a 255 csatorna nem használható csak egyes utasítások különleges alkalmazásaira. A BASIC rendszer a következő csatornákat használja alapertelmezésben:

0:

A parancsok beolvasására és a normál szövegek kiírására szolgál (LIST és PRINT), a gép bekapcsolásakor automatikusan az „EDITOR:" eszközhöz kapcsolódik. Az „EDITOR:" a „KEYBOARD:" és a „VIDEO:" eszközöket használja, videomód 0-t és 24 sor 40 oszlopot beállítva. Ez a csatorna a REDIRECT FROM és a COPY FROM alapértelmezése.

A számítógép (újra)indításakor megnyitása automatikus. Az ANSI előírásoknak megfelelően ez a csatorna az alapértelmezés szerinti parancs-csatorna, az összes többi csatorna száma 100 felett van, az ez alattiakat a felhasználó kezelheti tetszőlegesen.

101:

A grafikus írás, ill. olvasás alapértelmezés szerinti csatornája, a GRAPHICS parancs hatására a „VIDEO:" eszközhöz rendelődik, videomód 1, video colour 1, 20,40 lapméretet beállítva. A csatorna megnyitása automatikus, és csak egyes parancsok zárják le, mint pl. a TEXT.

102:

A szabványos szöveglap. Megnyitása automatikus 24,40-es lapmérettel.

103:

Hangkimenet. Alapértelmezésben a „SOUND:" eszközhöz csatlakozik és megnyitása automatikus.

104:

Sornyomtató kimenet. Alapértelmezésben a „PRINTER:" eszközhöz csatlakozik és megnyitása automatikus. A COPY TO és a REDIRECT TO alapértelmezése.

105:

Billentyűzet bemenet, alapértelmezése a „KEYBOARD:" eszköz. Megnyitása automatikus.

106:

A file jellegű adatmozgatásra szolgál. Ha lehetséges, akkor a DISK-1:" eszközhöz, ha nem, akkor a „TAPE:"-hez csatlakozik. Alapértelmezésben ezt használja a LOAD, MERGE, VERIFY utasítás. Ha nem külön OPEN utasítással nyitjuk meg, akkor a rendszer minden rá vonatkozó művelet előtt automatikusan megnyitja, majd a művelet befejezése után le is zárja.

107:

A lokális hálózat kapcsolódási pontja. Hasonlóan az előzőhöz, minden adatmozgatás előtt automatikusan megnyílik és utána lezáródik.

A 100–254 csatornák csak külön utasítással zárhatók le, az 1–99 csatornákat minden olyan parancs lezárja, amelyik törli a változókat. Ha a BASIC rendszer valamelyik alapértelmezésben használt csatornája zárva van, akkor az összes csatornát lezárva (0–254) az alapértelmezés szerintieket újra megnyitja. Amennyiben ez nem sikerül, a gép villogtatni kezdi a bordert, és mindaddig áll, míg újra nem indítjuk (RESET).

Az eszközök nevét a végükön álló kettőspontról ismeri fel a rendszer. Ha több azonos nevű eszköz is van, akkor ezeket egymástól számokkal lehet megkülönböztetni, pl. „DISK-1:" és „DISK-2:". A lehetséges eszköznevek:

DISK-n:

Lemezmeghajtók.

EDITOR:

Ez a képernyős szerkesztő egység, amely maga is a „VIDEO:" és a „KEYBOARD:" eszközöket használja.

KEYBOARD:

A billentyűzet, amely a beépített botkormányt is tartalmazza.

NET-n:

A lokális hálózat, „n" a hálózatbeli gép címe, amelyikkel kommunikálni akarunk (1–32). „n" = 0 esetén valamennyi egység üzeneteit be akarjuk olvasni, és valamennyi egységnek üzenni kívánunk.

PRINTER:
CENTRONICS sornyomtató illesztés.
SERIAL:
RS423 soros I/O csatorna.
SOUND:
Hanggenerátor.
TAPE-n:
Magnetofonegység.
VIDEO:
A képernyő.

A géphez csatolt többi eszköznek ezektől eltérő nevűnek kell lennie.

A legtöbb esetben elég egy eszköz nevét megadni az OPEN utasításban, csak a file szervezésű adatforgalomnál kell file-nevet is kijelölni. A teljes file-név szerkezete a következő:

eszköz-n:név

– eszköz: elhagyható, ilyenkor a rendszer automatikusan jelöl ki, ha nincs lemezegység csatolva, akkor a „TAPE:"-t.

– n: az egység száma, ha van. Ha elhagyjuk, alapertelmezése 1.

– név: a file azonosítója az egységen belül. A BASIC azonosítókra vonatkozó szabályok erre is érvényesek, azzal a különbséggel, hogy a rendszer csak az első 28 karaktert veszi figyelembe.

Ha a teljes file-névből kifelejtjük a kettőspontot, akkor a gép azt hiszi, hogy nem adtunk meg eszközt, így pl. a „SOUND" egy file a „TAPE-1:" egységen, a „SOUND:" pedig a hanggenerátor.

A „TAPE:" és a „DISK:" egységek kivételével a többi eszköz esetén a rendszer nem veszi figyelembe a „név" mezőt, annak tartalma hatástalan. Egyes utasításokban mind a csatornát, mind a file-nevet megadhatjuk, ilyen pl. a LOAD, SAVE.

OPTION

OPTION ANGLE DEGREES/RADIANS

A szögek megadásának módját választja ki: DEGREES fokban, RADIANS radiánban. Alapértelmezés: radián.

OUT

OUT n,a

Az „a" byte-ot írja ki az „n"-edik portra.

PING

Billentyűhangot ad ki.

PLOT

PLOT #csat:pontlista
PLOT #csat:ANGLE kif
PLOT #csat:FORWARD/BACK kif
PLOT #csat:LEFT/RIGHT kif
PLOT #csat:ELLIPSE kif,kif
PLOT #csat:PAINT

A pontlistát tartalmazó PLOT utasítás pontokat és/vagy vonalakat rajzol a képernyőre. Ha az utasítás végén pontosvessző áll, akkor a parancs végrehajtása után a rajzolósugár bekapcsolva marad, ha valami más, akkor nem.

Ha tehát a rajzolósugár be van kapcsolva és mi a sugarat az aktuális helyéről a „PLOT x,y" utasítással egy másik pontba mozdítjuk el, akkor a sugár nyomot hagy a képernyőn és vonalat rajzol a két pont közé. Az utasítás végére pontosvésszőt téve ("PLOT x,y;") a sugár bekapcsolva marad, így további vonalakat tudunk folytatólagosan húzni. A „PLOT x1,y1; x2,y2;..." utasítás az x1,y1 és az x2,y2 pont közé húz vonalat, amelyet folytatni is lehet, mivel a sugár bekapcsolva marad.

Rajzolni mindig az aktuális tintaszínnel (INK) lehet a „line style", ill. a „line mode" feltételeknek megfelelően. L.: videofeltételek.

A PLOT utasítás a grafikus koordinátákat használja, tehát a (0,0) pont a képernyő bal alsó sarkában van és az első szám a vízszintes, a második pedig a függőleges távolságát jelenti e (0,0) ponttól.

Az ELLIPSE segíségével ellipszist rajzolhatunk, amelynek középpontja az aktuális rajzolópontban lesz, a kontúr távolságát vízszintes irányban az első, függőlegesen pedig a második paraméter adja meg. Ha nem akarjuk, hogy az ellipszis közepén levő pont is megjelenjen a képernyőn, akkor rajzolás előtt kapcsoljuk ki a rajzolósugarat.

A PAINT feltétellel egy zárt alakzatot tudunk kiszínezni az aktuális tintaszínnel. Az alakzat határát onnan ismeri fel a számítógép, hogy az ott levő pontok színe különbözik a rajzolópont kiindulási helyének eredeti színétől. Ha tehát a rajzolópont helyén a képernyő szine éppen a tintaszín, akkor nem történik semmi, mert a gép azt hiszi, hogy máris elérte az alakzat határát.

A PLOT parancs alapértelmezésben a 101-es csatornát használja.

POKE

POKE cím,érték

A megadott értéket helyezi el a kijelölt tárcímre.

PRINT

PRINT #csat, AT sorszám-kif,oszlopszám-kif:adatlista
PRINT #csat,USING sorszám:adatlista
PRINT #csat,USING formátumleírás:adatlista

Az adatlista eleme kifejezés vagy a TAB kulcsszó és az utána zárójelek között álló oszlopszám lehet. Az elemeket vesszők vagy pontosvesszők választják el. A kettő között az a különbség, hogy a vessző hatására a következő elem kiírását mindig a következő tabulátorpozíciónál kezdi, a pontosvessző esetén pedig közvetlenül az előző mező után (l. korábban: vonalzósor). A TAB hatására a kurzor a megadott oszlopra ugrik és ott folytatódik a kiírás. Az adatlista végén álló vessző vagy pontosvessző hatására nem generálódik sorvégejel az utasítás végén, így a kiírás ugyanabban a sorban folytatható.

Parancsmódban is használható. Az AT feltétellel a kurzor a kívánt helyre állítható, ilyenkor a szöveges koordináta-rendszert kell használni, amelynek a kiindulópontja a képernyő bal felső sarkában van és első paramétere a sorszámot, a második az oszlopszámot jelöli. A „PRINT AT 2,15: a$" tehát a második sor tizenötödik oszlopától kezdve írja ki az a$ tartalmát.

A USING feltétellel a kiírás formáját határozhatjuk meg vagy egy IMAGE utasításra mutató sorszám, vagy egy az ottanihoz hasonló formátumleíró lista megadásával.

Az utasítás alapértelmezésben a 102-es csatornára ír.

PROGRAM

PROGRAM név (változólista)

Az aktuális program nevét adja meg, amit később a CHAIN vagy a SAVE utasításban majd használhatunk. A változólistán keresztül az egyes programok paramétereket adhatnak át egymásnak. (L. még: CHAIN, EDIT.)

RANDOMIZE

A véletlenszám-generátor kiindulóértékét változtatja meg, enélkül ugyanis az egyes lefuttatások alkalmával mindig ugyanazokat a számokat állítaná elő.

READ

READ változólista
READ IF MISSING sorszám:változólista
READ IF MISSING EXIT DO:változólista

A DATA utasítással tárolt adatokat olvassuk be a változókba. Ha az adatok elfogyása utan is olvasni akarunk, akkor az IF MISSING után meghatározott tevékenységet hajtja végre a gép.

REDIRECT

REDIRECT FROM #csat TO #csat

Az első csatornáról beolvasott adatokat írja ki a második csatornára, addig, amíg adatvége jel nem érkezik, hiba nem következik be, vagy le nem ütjük a STOP billentyűt. A tevékenységet úgy tudjuk kikapcsolni, hogy ugyanezt a parancsot adjuk ki, de a második csatorna helyett a 255 csatornára hivatkozunk.

REM

A megjegyzéssorokat jelöli, utána legalább egy szóköznek kell állnia. Helyettesíthető a „!" jellel.

RENUMBER

RENUMBER sormehatározás TO sormeghatározás AT kif STEP kif
RENUMBER blokknév AT kif STEP kif

A programnak, ill. egyes részeinek az újrasorszámozására alkalmas, csak parancsmódban alkalmazható. A STEP vagy az AT paramétert elhagyva, a kezdő sorszám 100, a lépésköz 10 lesz. Sortartomány megadása nélkül az egész programot újrasorszámozza. Az átsorszámozandó blokkot DEF vagy HANDLER blokk esetén a nevével is megadhatjuk. Egyéb tekintetben a sorszámtartomány megadására ugyanazok a szabályok vonatkoznak, mint a DELETE utasításra.

A programban levő valamennyi sorszámhivatkozást módosítja az új sorszámoknak megfelelően. A parancs hatására azonban a program sorainak sorrendje nem módosulhat. Az átsorszámozott sorok nem írhatják fölül és nem vehetik körbe a régieket, nem kerülhetnek a programon belül új helyre és természetesen sorszámuk nem haladhatja meg a lehetséges legnagyobb értéket (9999). Ilyen esetekben a gép nem hajtja végre a parancsot, a program változatlan marad.

RESTORE

RESTORE
RESTORE sorszám

A következő READ utasítás által olvasandó DATA adatsor sorszámát adja meg. Ha a paraméter hiányzik, akkor a program első DATA utasítására vonatkozik.

RETRY

A megszakításkezelő (HANDLER) rutinból visszatér arra az utasításra,

aminek végrehajtása közben a megszakítás bekövetkezett. (L.: CONTINUE.) A STOP billentyűt elfogó megszakításkezelő ezzel tér vissza a megfelelő utasításra.

RETURN

A GOSUB-bal hívott programrészletből való visszatérésre szolgál.

RUN

RUN (paraméterlista)
RUN sorszám
RUN #csat:file-név(paraméterlista)
RUN eszköz(paraméterlista)

Az önmagában álló RUN a programot a legelső sorától kezdi lefuttatni. Más sorszámmal a belépési pontot tudjuk módosítani. Valamilyen file-nevet megadva az ilyen nevű file-ban levő programot tudjuk betölteni és lefuttatni. A paramétereknek a PROGRAM utasításban megadottakkal illeszkedőeknek kell lenniük. (l.: PROGRAM.) Törli a változókat.

SAVE

SAVE #csat file-név
SAVE eszköznév
SAVE ALL #csat:file-név

Az aktuális, ill. az ALL kiterjesztést megadva az összes programot a megadható eszközre, ill. csatornára lehet menteni (alapértelmezés: #106) a szintén nem kötelezően megadandó file-névvel. File-név nélkül a korábban a PROGRAM utasításban kijelölt nevet használja.

A programokat kódolt formában menti ki, ha ASCII karakterekkel akarjuk kiíratni, akkor használjuk a LIST #csat:file-név utasítást.

Csak parancsmódban használható.

SELECT

A struktúra:
select sor
case sor
  utasítások vagy blokkok
end select sor
A case sor és a hozzá tartozó utasítások többször is ismétlődhetnek.
select sor:
SELECT CASE kif
case sor:

```
CASE kif
CASE kif TO kif
CASE IS relop kif
CASE ELSE
end select sor:
END SELECT
```

A select struktúrában megvizsgáljuk egy változó, ill. kifejezés értékét, és aszerint, hogy a vizsgálat eredménye igaz vagy hamis, haladunk tovább egyes programágakon. A megvizsgálandó változót, ill. kifejezést a select sorban adjuk meg, az itt levő CASE kulcsszó elhagyható, kivéve, ha a kifejezés éppen a CASE azonosítóval kezdődik:

```
SELECT CASE CASE+23
```

Az egyes ágakat a sorszámok sorrendjében vizsgáljuk meg, a CASE ELSE ág után nincs értelme továbbiakat írni, mert ezeket úgysem hajtja végre a program. Több CASE feltétel egyetlen sorba is összevonható, egymástól vesszővel elválasztva.

```
SELECT CASE N
CASE 1
  PRINT „első"
CASE 2 TO 9,11,21
  PRINT „valami más"
CASE IS<=A+20
  PRINT „ez is más"
CASE ELSE
  PRINT „maradék"
END SELECT
```

A CASE ELSE sor csak egyszer fordulhat elő, és utolsónak kell lennie. A SELECT blokkokba tilos kívülről ugrani.

SET

A különböző gépi feltételek állíthatók be vele. (L.: gépi, video-, hangfeltételek, ASK, TOGGLE.

SOUND

SOUND #csat:PITCH kif, DURATION kif, LEFT kif, RIGHT kif, SOURCE kif, STYLE kif, ENVELOPE kif, SYNC kif, INTERRUPT

A hanggenerátor irányítására szolgál. A paraméterek sorrendje nem lényeges.

A PITCH után álló szám 0–127 közötti lehet, de csak a 0–83 tartomány használatos. Itt a hang magasságát adja meg félhangokban. A középső C-nek a 37 felel meg, ez egyben az alapérték is.

A DURATION a hang időtartamát adja meg, 1/50 másodpercben, az alapértéke 50.

A LEFT és a RIGHT a bal és a jobb oldali hangszóró hangerejét jelöli ki a lehetséges maximumhoz viszonyítva (0 = csend, 255 = teljes hanerő). Mono eszköznél a két csatorna hangereje összeadódik.

A SOURCE a felhasználandó hanggenerátort adja meg (0–3). A 3 a zajgenerátor, ennek alkalmazásakor a hangmagasság paraméterét figyelmen kívül kell hagyni. Alapértelmezés: 0.

A STYLE (0–255, alapérték: 0) jelentését a hang feltételeknél ismertetjük majd.

Az ENVELOPE a használandó hangtár számát határozza meg, alapérték a 255, a beépített hangtár. (L.: ENVELOPE utasítás.)

A SYNC segítségével különböző hanggenerátorokat tudunk szinkronizálni. Ha pl. három hangot akarunk egyszerre indítani, akkor mindegyik hangfeltételei között meg kell adjuk a SYNC 2-t. Alapérték: 0.

Az INTERRUPT hatására valamennyi más hangforrásból származó hang megszakad, és az új hang szólal meg.

SPOKE

SPOKE szegmens, cím, érték

Hasonló a POKE-hoz, csak ez szegmens- és ofszetcímet ad meg.

START

Beolvassa és elindítja az első programot a 106 csatornáról. Ha már van program a tárban, akkor hatása azonos a RUN-éval.

STOP

Megszakítja a program futását, folytatni a CONTINUE paranccsal lehet.

STRING

STRING*n vált/tömb-lista*n

A karatersorozat típusú változókat deklarálja, alapértelmezésben 132 karakter max. hosszúságot megadva. A kulcsszót követő csillag utáni számmal az összes itt deklarált változóra más hosszúságadatot adhatunk meg, a nevet követő csillag után csak arra az egy változóra vonatkozót. Mindkét megadás elhagyható. A tömbindex alsó határértékének alapértéke 0.

STRING*8 nev$*20, cim$*40, jel

STRING tomb$(4 TO 99)*10

A „nev" hossza 20, a „cim"-é 40, a „jel"-é pedig 8 karakter lehet. A „tomb"-nek 96 darab 10 karakteres eleme van.

TEXT

TEXT
TEXT 40
TEXT 80

Az állapotjelző sor kivételével a teljes képernyőre nyit meg egy szöveges lapot és lezárja a korábban megnyitott grafikus lapokat. A paraméter az egy sorban levő karakterek számát adja meg, hiánya esetén az előző értéket használja újra.

THEN

L. IF

TIME

TIME „oo:pp:mm"

A gép belső óráját állítja be. Parancsmódban is használható. (L. még: TIME$ függvény, WAIT DELAY parancs, TIMER gépi feltétel.) Pl. a délután fél ötöt a következő módon állítom be:

TIME „16:30:00"

TOGGLE

A kétértékű gépi feltételek átkapcsolására szolgál. (L. még: gépi, video, hang feltételek, SET, ASK.)

TRACE

TRACE ON TO #csat
TRACE OFF

Az éppen végrehajtott utasítás sorszámát írja ki a megadott csatornára, annak hiányában a 0 csatornára. A TRACE OFF kikapcsolja ezt a nyomkövetést.

TYPE

A szövegszerkesztőt hívja meg a BASIC-ből, de előtte még egy ENTER-rel meg kell erősítenünk a parancsot, nehogy véletlenül töröljük a programunkat. Az utasítás ugyanis törli az összes programot és változót a tárból.

VERIFY

VERIFY #csat file-név

A korábban mentett programot ellenőrzi, beolvassa a programot és

összehasonlítja az aktuális programmal. Ha eltérést talál, hibát jelez. Alapértelmezés a 106 csatorna. Csak parancsmódban használható.

WAIT DELAY

WAIT DELAY kif

A kifejezésben megadott számú másodpercig felfüggeszti a program futását. Max. érték: 32767. (L. még: TIMER gépi feltétel.) A DELAY szó elhagyható. A következő utasítás egy percre állítja meg a programot:

WAIT DELAY 60

WHEN

A struktúra:

WHEN EXCEPTION USE megszakításkezelő
utasítások
END WHEN

A blokkon belül bekövetkező megszakításokat, hibákat, különleges eseteket kezelő rutint határozza meg. A WHEN blokkok egymásba ágyazhatók. (L.: HANDLER.)

GÉPI FELTÉTELEK

A legtöbb rendszerváltozót és gépi eljárást közvetlenül irányíthatjuk BASIC nyelven. Egy-egy ilyen feltétel beállítását a SET utasítással, értékének lekérdezését az ASK utasítással, átkapcsolását pedig a TOGGLE utasítással érhetjük el.

DEFAULT CHANNEL

SET DEFAULT CHANNEL kif

Az alapértelmezés szerinti rendszercsatorna számát lehet vele beállítani. Ezt a csatornát használják a renszer eljárásai a felhasználóval való kommunikációra. Ezen a csatornán keresztül válaszolnak az egyes programok a HELP hívásra. Induláskor a rendszer a 0 csatornát jelöli ki.

EDITOR BUFFER

SET EDITOR BUFFER kif

Az editor pufferjének a méretét adja meg 256 byte-os lapokban. Használható az ASK.

EDITOR KEY

SET EDITOR KEY csat

Az ezzel kijelölt csatornán keresztül olvassa az editor a billentyűzetet. (KEYBOARD:) Használható az ASK.

EDITOR VIDEO

SET EDITOR VIDEO csat

Ezen a csatornán keresztül ír az editor a képernyőre. (VIDEO:) Használható az ASK.

FAST SAVE

SET FAST SAVE ON/OFF

A szalagra vétel sebességét állítja be. A gyors felvétel kb. 2400 baud, a lassú ennek a fele. Betöltéskor a megfelelő sebesség automatikusan beáll. Használható a TOGGLE.

FKEY

SET #csat:FKEY bill.szám str

A funkcióbillentyűk hatására megjelenő karaktersorozatokat állítja be. A csatorna alapértelmezése: #105.

A funkcióbillentyűk 1–16-ig számozódnak, 1–8 önmagukban, 9–16 a SHIFT-tel együtt lenyomva. A rendszer által eredetileg beállított jelentéseket a CLEAR FKEYS paranccsal állíthatjuk vissza. Az utasításvégi automatikus ENTER-t a CHR$(13)-mal érhetjük el.

INTERRUPT

ASK INTERRUPT CODE vált

A legutolsó szoftver megszakítás kódját kérdezi le.

SET INTERRUPT KEY ON/OFF

Bekapcsoláskor minden billentyű leütése szoftver megszakítást okoz. Használható a TOGGLE.

SET INTERRUPT NET ON/OFF

Bekapcsoláskor a lokális hálózatról érkező adatok szoftver megszakítást okoznak.

SET INTERRUPT STOP ON/OFF

A STOP billentyű hatására bekövetkező szoftver megszakítást kapcsolja be, ill. ki. Használható a TOGGLE.

KEY CLICK

SET KEY CLICK ON/OFF

Bekapcsolt állapotban a billentyűk lenyomását rövid hanggal jelzi a gép. Használható a TOGGLE.

KEY DELAY

SET KEY DELAY kif

Azt az időt állítja be – 1/50 másodpercekben mérve –, ameddig egy billentyűt lenyomva kell tartanunk ahhoz, hogy a gép elkezdje automatikusan ismételni. Használható az ASK.

KEY RATE

SET KEY RATE kif

Beállítja, hogy az automatikus ismétlés során az egyes billentyűk mekkora időközökkel ismétlődjenek. Mértékegysége 1/50 másodperc. Használható az ASK.

NET CHANNEL

ASK NET CHANNEL vált

Annak a csatornának a számát adja vissza, amelyiken a pufferben levő olvasásra váró adat érkezett. Ha az első byte-ot beolvassuk, akkor a NET CHANNEL értéke automatikusan átáll a következő olyan csatorna számára, amelyik adatot szeretne nekünk átadni. Ha nincs több ilyen csatorna, akkor a 255 értéket veszi fel.

NET MACHINE

ASK NET MACHINE vált

A NET CHANNEL-lel együtt kap értéket és annak a gépnek a hálózatbeli számát adja vissza, amelyik az adatot küldte. Ez különösen fontos, ha az adat az általános (NET-0:) csatornán érkezett.

NET NUMBER

SET NET NUMBER kif

A saját gépünk hálózatbeli számát állítja be. Eredeti értéke 0, ami hibát okoz, ezért a hálózat használata előtt be kell állítanunk egy 1–32 közötti, még szabad értékre.

REM1

SET REM1 ON/OFF
Az 1. távvezérlő kimenetet állítja át. (A magnetofont kezelő utasítások automatikusan állítják.)

REM2

SET REM2 ON/OFF
Ugyanaz, csak a REM2 kimenetre.

SERIAL BAUD

SET SERIAL BAUD kif
A kifejezés (0–15) a soros kimenet (RS232) és a lokális hálózat sebességét állítja be. Használható az ASK. A kódok:

0 => 50 (baud)	8 => 1200
1 => 75	9 => 1800
2 => 110	10 => 2400
3 => 134.5	11 => 3600
4 => 150	12 => 4800
5 => 200	13 => 7200
6 => 300	14 => 9600
7 => 600	15 => 9600

Alapérték:15 => 9600

SERIAL FORMAT

SET SERIAL FORMAT kif
A soros meghajtó által használt formátumot állítja be az egyes biteken keresztül. A 3-nál magasabb helyiértékű biteknek 0-nak kell lenniük. Alapértelmezés: 8 bit, paritás nélkül, 2 stop-bit. Ügyeljünk arra, hogy a lokális hálózat használata mindig az alapértelmezést állítja vissza. A bitek jelentése:

BIT	ÉRTÉK	HATÁS
0	0	8 bit
	1	7 bit
1	0	nincs paritás
	1	van paritás
2	0	páros paritás
	1	páratlan paritás
3	0	két stop bit
	1	egy stop bit

STATUS

SET STATUS ON/OFF

A képernyő tetején levő állapotjelző sor kiírását kapcsolja be/ki. Használható a TOGGLE.

TAPE LEVEL

SET TAPE LEVEL kif

A magnetofonkimenet jelszintjét állítja be. A kifejezés az 1–6 tartományban használatos, itt minden szint az előzőnek a kétszeresét jelenti, és az 1 szint kb. 40mV-nak felel meg. Alapértelmezés: 2.

TAPE SOUND

SET TAPE SOUND ON/OFF

A kazettás bemenetről jövő jeleknek a beépített hangszóróra és a HI-FI kimenetre való vezérlését kapcsolja be/ki. Használható a TOGGLE.

TIMER

SET TIMER kif

Olyan órát indít el visszafelé, amely a nullát elérve szoftver megszakítást generál. A kifejezés a beállítandó időt adja meg másodpercben, maximuma 255. A 0 érték beállításakor az óra kikapcsol. A nullát elérve az óra mindig automatikusan megáll és újra el kell indítanunk.

Az ilyen megszakításkor az EXTYPE értéke 9229, az INTERRUPT CODE 64 lesz.

VARIABLE

SET vált-szám,kif

ASK vált-szám vált

TOGGLE vált-szám

Az operációs rendszer megfelelő sorszámú változójára végzi el a műveletet. Ezekről részletes leírást ez a könyv nem tartalmaz, csupán az operációs rendszernél ismertetjük röviden.

VIDEOFELTÉTELEK

Ezekkel a feltetelekkel a „VIDEO:" működését tudjuk irányítani. Ez az eszköz egyszerre több videolapot kezelhet, de leggyakrabban a standard szöveges (102), ill. a grafikus (101) lapot használjuk, amit általában nem is kell külön jelölnünk az egyes parancsoknál. A feltételek használata során a COLOUR szó helyett bármikor használhatjuk a COLOR szót.

ATTRIBUTES

SET ATTRIBUTES kif

Speciális jelzőkkel irányítja az „attributum video mód" működését (l. VIDEO MODE 15). Az egyes jelzőértékek a következőek:

1 - rajzolás papírszínnel (0);
2 - rajzolás a képpontok megváltoztatása nélkül;
4 - rajzolás a tintaszínek megváltoztatása nélkül;
8 - rajzolás a papírszínek megváltoztatása nélkül;
16 - írás papírszínnel
32 - írás a képpontok megváltoztatása nélkül;
64 - írás a tintaszínek megváltoztatása nélkül;
128 - írás a papírszínek megváltoztatása nélkül.

A kívánt szolgáltatások jelzőértékeit össze kell adni, és az összeget be kell írni a feltételbe. Alapértelmezés: 0.

BEAM

SET #csat:BEAM ON/OFF

A rajzolósugarat lehet vele be/ki kapcsolni. Kikapcsolt állapotban a sugár mozgatása nem hagy nyomot a képernyőn. A csatornaszám alapértelmezése: 101.

BIAS

SET #csat:BIAS színkód

A paletta felső nyolc színét (8–15) állíthatjuk be vele. A lehetséges 256 szín 32 db nyolcas csoportba van osztva, egy csoport bármelyik színének a kódját megadva, a csoport színei kerülnek a paletta felső részére. A színt az RGB függvénnyel is megadhatjuk. A csatorna alapértelmezése 101, azonban a BIAS valamennyi lapra kifejti hatását.

BORDER

SET #csat:BORDER színkód

A képernyő peremének színét állítja be. A csatorna alapértelmezése: 101.

CHARACTER

SET #csat:CHARACTER n,a,b,c,d,e,f,g,h,i

Az „n" kódú (ASCII) karakter alakját határozza meg. Minden további paraméter az alak egy-egy sorát adja, felülről lefelé haladva. Az sorokat a BIN függvénnyel adhatjuk meg kényelmesen, ahol a nullák a papír, az egyesek a tinta színét jelentik. Bár megadhatunk csatornaszámot, a parancs az összes

videolapra vonatkozik. A csatornaszám alapértelmezése 102. Az eredeti karakteralakokat a CLEAR FONT paranccsal állíthatjuk vissza.

COLOUR

SET #csat:COLOUR palettaszám:színkód
A paletta egyik alsó (0–7) színét állítja be. A szín kódját az RGB függvénnyel vagy színnévvel is megadhatjuk.

CURSOR

SET #csat:CURSOR CHARACTER kód
SET #csat:CURSOR COLOUR palettaszám
A kurzor alakját és színét adhatjuk meg vele, az alakot a CHARACTER utasításban használt ASCII kóddal, a színt a palettáról választva. Csatorna alapérték: 102.

INK

SET #csat:INK színszám
A tinta színét állítja be, a „COLOUR MODE 3"-nál a színkóddal, a másik három üzemmódnál a palettaszámmal. Csatorna-alapérték: 101.

LINE MODE

SET #csat:LINE MODE kif
A régi és az új képernyősorok színeinek egymásra hatását adja meg. Nullás módban (alapértelmezés) az új sorok színei felülírják a régieket, a többi módban viszont a kettő kombinációja alakul ki a következők szerint:
mód 1 - vagy (OR)
mód 2 - és (AND)
mód 3 - kizáró vagy (XOR)
Csatorna alapértelmezés: 101.

LINE STYLE

SET #csat:LINE STYLE kif
Szaggatott vonalak rajzolását teszi lehetővé a paraméter (1–14) értékétől függően. Csatorna-alapértelmezés: 101.

PALETTE

SET #csat:PALETTE a,b,c,d,e,f,g,h
A paletta első nyolc (0–7) színét állítja be. Csatorna-alapérték: 101.

Egyes színmódban (COLOUR MODE) csak az első négy szín használható, grafikus lapokon nullás színmódban csak az első kettő. A meg nem adott színek alapértelmezése: 0.

A színeket a standard kódjukkal (0–255), az RGB függvénnyel vagy egyes esetekben a nevükkel adhatjuk meg. A paletta felső nyolc színét a BIAS opcióval tudjuk beállítani.

PAPER

SET #csat:PAPER szinszám

Hasonló az INK-hez, csak itt a papír színét adhatjuk meg. Csatorna-alapértelmezés: 101. Grafikus lapokon (VIDEO MODE 1, ill. 5) a parancs csak a lap letörlése után hatásos.

A 80 karakteres sorok használata esetén (VIDEO MODE 2) a papír csak a 0,2,4,6, a tinta csak az 1,3,5,7 színű lehet, de az egyik megadása a másikat is megfelelően beállítja. A 40 karakteres sorok esetén (VIDEO MODE 0) csak az első két papír–tinta páros használható.

SCROLL

SET #csat: SCROLL ON/OFF

Az automatikus képernyőléptetést kapcsolja be/ki. Csatorna-alapérték: 102.

SET #csat:SCROLL UP/DOWN n,m

A képernyőt lépteti az (n-32)-ik sortól az (m-32)-ig. Csatorna-alapértelmezés: 102.

VIDEO COLOUR

SET VIDEO COLOUR kif

A színüzemmódot (színmód, VIDEO COLOUR) állítja be az ezután megnyitandó videolapokra. A lapok megnyitásakor a nullás módot választja ki alapértelmezesként a gép. Nagyfelbontású grafikus lapoknál (HIRES) a vízszintes irányban egymás mellet levő képpontok és a használható színek száma a következő:

mód 0 – 2 szín, 640 képpont;
mód 1 – 4 szín, 320 képpont;
mód 2 – 16 szín, 160 képpont;
mód 3 – 256 szín, 80 képpont.

Kisfelbontású lapokon (LORES) a színek száma nem változik, csak a képpontok száma fele akkora.

VIDEO MODE

SET VIDEO MODE kif

Az ezután megnyitandó lapokon állítja be a video-üzemmódot. A lehetséges értékek a következők:

mód 0 – 40 oszlopos szöveglap (2 színpár);
mód 1 – nagyfelbontású grafikus lap;
mód 2 – 80 oszlopos szöveglap (4 színpár);
mód 5 – kisfelbontású grafikus lap;
mód 15 – attributum típusú grafikus lap.

VIDEO X

SET VIDEO X kif

Az ezután megnyitandó lapok szélességét adja meg. A méretet (2–42) karakterekben kell számolni.

VIDEO Y

SET VIDEO Y kif

Az előzőhöz hasonló, csak a függőleges méretet határozza meg karaktersorokban mérve (1–255).

HANGFELTÉTELEK

SOUND BUFFER

SET SOUND BUFFER kif

A legközelebb a „SOUND:" eszközre megnyitandó csatorna hangtárának a méretét adja meg hangfázisokban mérve (1–255). Az alapérték 20. Használható az ASK.

SOUND STYLE

A paraméter hatása a következő:

A 0 hangcsatornára:
16 – alacsony torzítás;
32 – közepes torzítás;
48 – magas torzítás;
64 – felüláteresztő szűrő használata. Órajel az 1 csatorna;
128 – gyűrűmoduláció a 2 csatornával.

Az 1 hangcsatornára:

Ugyanaz, de a felüláteresztő szűrő órajele a 2 csatorna és a gyűrűmoduláció a 3 csatornát használja.

A 2 csatornára:

Ugyanaz, de a felüláteresztő szűrő órajele a 3 csatorna és a gyűrűmoduláció a 0 csatornát használja.

A 3 csatornára (zajcsatorna):

1,2,3 - A 0, 1, ill. 2 csatorna az órajel a standard 31.25kHz helyett.

4,8,12,16 - Az eredeti 17 bites polinom számláló helyett 15, 11, 9 ill. 7 bitest használ.

32 - Alulateresztő szűrő használata, órajele a 2 csatorna.

64 - Felüláteresztő szűrő használata, órajele a 0 csatorna.

128 - Gyűrűmoduláció az 1 csatornával.

A kívánt paramétereket össze kell adni, és az utasításba az összeget kell beírni.

SPEAKER

SET SPEAKER ON/OFF

A beépített hangszórót kapcsolja be/ki.

BEÉPÍTETT FÜGGVÉNYEK ÉS VÁLTOZÓK

A trigonometrikus függvények a megfelelő feltétellel beállított szög mértékegységben dolgoznak. A Minimál BASIC az ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAB és TAN függvényeket ismeri.

ABS(x)

Az x abszolút értéke, a függvény egyszerűen csak az előjelet törli.

ACOS(x)

A koszinuszfüggvény ellentettje, azaz azt a szöget adja meg, amelynek a koszinusza x, ACOS(COS(X)) = X.

ANGLE(x,y)

A pozitív x tengely és a (0,0) pontból induló, (x,y) ponton átmenő egyenes szöge.

ASIN(x)

A szinuszfüggvény ellentettje.

ATN(x)

A tangensfüggvény ellentettje.

BIN(x)

Az x bináris szám értéke, pl. BIN(11001) = 25.

BLACK

A fekete szín, azaz RGB(0,0,0).

BLUE

A kék szín, azaz RGB(0,0,1).

CEIL(x)

Az x-nél nem kisebb, legkisebb egész szám, azaz a legközelebbi egészre felkerekített érték. Pl. CEIL(3.45) = 4, CEIL(-3.45) = -3.

CHR$(x)

Az x ASCII kódú karakter.

COS(x)

A koszinuszfüggvény.

COSH(x)

A koszinusz hiperbolikusz függvény.

COT(x)

A kotangensfüggvény.

CSC(x)

A koszekánsfüggvény.

CYAN

A lila szín, azaz RGB(0,1,1).

DATE$

A dátum, szabványos formában: eeeehhnn. (L. DATE parancs.)

DEG(x)

Az x radián szög értéke fokban, azaz x*180/PI.

EPS(x)

A legkisebb mennyiség, amit x-hez adva vagy belőle levonva az x tárolt értéke megváltozik.

EXLINE

A legutóbbi megszakítást okozó sor sorszáma.

EXP(x)

Az „e" az x-edik hatványon függvény. Az „e" (=2.71828...) a természetes logaritmus bázisa.

EXSTRING$(n)

Az n kódú megszakításra jellemző üzenet (hibaüzenet) szövege, az elején egy szóközzel.

EXTYPE

A legutóbbi megszakítás kódszáma.

FP(x)

Törtrész függvény, az IP ellentéte. FP(1.23)=0.23, FP(-1.23)=-0.23.

FREE

Az aktuális program számára rendelkezésre álló szabad byte-ok száma a tárban, nem azonos a BASIC rendszer számára szabad tárral.

GREEN

A zöld szín, azaz RGB(0,1,0).

HEX$(x$)

Az x$-ban egymástól vesszővel megadott hexadecimális byte-okból képzett karaktersorozat. HEX$("31,32,33")="123"

IN(n)

Az n input portról olvas be.

INF

A gép által használható legnagyobb pozitív szám= 9.999999999*10↑62.

INKEY$

A lenyomott billentyű karaktere, ill. ha nincs egyetlen billentyű sem benyomva, akkor egy üres karaktersorozat ("").

INT(x)

A legnagyobb, x-nél nem nagyobb egész szám. INT(3.4) = 3, INT(-3.4) = -4.

IP(x)

Egész rész, azaz a tizedespont utáni részt törli. IP(9.9) = 9, IP(-9.9) = -9.

JOY(n)

Az n botkormány helyzetétől függő értéket ad ki:
1 – jobbra dől;
2 – balra dől;
4 – lefelé;
8 – felfelé;
16 – a tűzgomb benyomva.
Ha a botkormány átlósan áll vagy a tűzgomb is be van nyomva, akkor a függvény az egyes értékek összegét adja. A beépített botkormány száma 0, ezenkívül az 1 és a 2 használható. A beépített botkormánynál a szóközbillentyű a tűzgomb.

LBOUND(a)

Az „a" egydimenziós tömb alsó indexhatára.

LBOUND(a,n)

Az „a" többdimenziós tömb n-edik dimenziójának alsó indexhatára.

LCASE$(a$)

A nagybetűket kisbetűvé alakítja.

LEN(a$)

A a$-ban levő karakterek számát adja.

LOG(x)

Természetes, azaz „e" alapú logaritmus.

LOG10(x)

Tizes alapú logaritmus.

LOG2(x)

Kettes alapú logaritmus.

LTRIM$(a$)

Törli a karaktersorozat elején álló szóközöket. LTRIM$("hh") = "hh".

MAGENTA

A bíborvörös, azaz RGB(1,0,1).

MAX(x,y)

A két szám közül a nagyobbik, pl. MAX(6,99) = 99.

MAXLEN(a$)

A karakterváltozó lehetséges max. hossza.

MIN(x,y)

A két szám közül a kisebbik.

MOD(x,y)

X modulo y, azaz az x és y osztásának egész maradéka. MOD(-1,3) = 2. L.: REM(x,y).

ORD(a$)

Az a$ karakter vagy karaktersorozat esetén az első karakter ASCII kódja.

PEEK(n)

A Z80 n memóriacímén levő érték.

PI

A pi nevű szám, a gép kerekítve 3.141592654-gyel számol.

POS(x$,y$)

Az x$ szövegrészlet első előfordulásának a helyét adja meg az y$ szövegben. Ha x$ nincs benne az y$-ban, akkor az értéke nulla.

POS(x$,y$,n)

Hasonló az előzőhöz, de a keresést az y$ szöveg n-edik karakterénél kezdi. Tehát POS("ON","LONDON") = 2, de POS("ON","LONDON",4) = 5, mert a második esetben a keresést a „D" betűnél kezdte.

RAD(x)

Az x fok értéke radiánban, azaz x*PI/180.

RED

A piros szín, azaz RGB(1,0,0).

REM(x,y)

Az x és y osztásának maradéka. REM(-1,3) = -1. L.: MOD(x,y).

RGB(r,g,b)

A függvényt egy szín megadására használhatjuk a szinkódja helyett. A piros (RED), a zöld (GREEN) és a kék (BLUE) színekből keverhetünk ki egy újat, az egyes összetevők arányát megadva a maximumhoz képest, pl.:

```
SET INK RGB(0.5,1/3,0.25)
```

RND

Egy 0 és 1 közötti véletlenszámot állít elő, de az 1-et nem éri el soha. Az értéktartományt szorzással tudjuk növelni, pl. az INT(RND*100) egy 0 és 99 közti egész számot ad.

RND(x)

Egy 0 és x közötti egész véletlenszámot ad, „n" maximuma 32767.

ROUND(x,n)

Az x számot kerekíti n jegyre, pl. ROUND(1.7668,2) = 1.77, de ROUND(-1.7668,2) = -1.76.

RTRIM$(a$)

A karaktersorozat végén levő szóközöket levágja.

SEC(x)

A szekánsfüggvény.

SGN(x)

Az x előjelétől függően, ha x negatív, akkor -1, ha x pozitív, akkor +1, ha pedig x nulla, akkor a függvény értéke is az.

SIN(x)

A szinuszfüggvény.

SINH(x)

A szinusz hiperbolikusz függvény.

SIZE(a)

Az „a" tömb elemeinek a száma.

SIZE(a,n)

Az „a" tömb n-edik dimenziójában levő elemek száma.

SPEEK(s,n)

Hasonló a PEEK-hez, de az s szegmens n címéről olvas.

SQR(x)

Az x négyzetgyöke.

STR$(x)

Az x számot karaktersorozattá alakítja, és ha szükséges a „–" előjelet is elé teszi.

TAB(x)

Csak a print utasításon belül alkalmazható, az x-edik oszlopba állítja a kurzort.

TAN(x)

A tangensfüggvény.

TANH(x)

A tangens hiperbolikusz függvény.

TIME$

Az aktuális idő értéke szabványos formában:"oo:pp:mm". (L.: TIME parancs.)

TRUNCATE(x,n)

Az x-ből n db decimális pozíciót vág le (csonkít).

UBOUND(a)

Az „a" tömb felső indexhatára.

UBOUND(a,n)

Az „a" tömb n-edik dimenziójának felső indexhatára.

UCASE$(a$)

A kisbetűket alakítja át nagybetűkké a karaktersorozatban.

USR(n,x)

Az n című (ami általában a CODE utasításban betöltött cím) gépi kódú rutint hívja meg, és az x értékét átadja neki a HL-ben. A függvény értéke a rutinból való visszatéréskor a HL-ben levő szám lesz.

VAL(a$)

Egy karaktersorozatot alakít át számmá. A STR$ függvény ellentettje. Az átalakítást a karaktersorozat első nem szám karakteréig végzi.

WHITE

A fehér szín, azaz RGB(1,1,1).

WORD$(n)

Az n szám értékének megfelelő két byte-ot adja vissza karaktersorozatként, elöl a kisebb helyiértékű byte-tal. Általában egy CODE utasításban beállított „n" értékét használjuk fel ugrási cím számításhoz.

YELLOW

A sárga szín, azaz RGB(1,1,0).

EXOS

Az EXOS az Enterprise eXpandable Operating System, azaz az Enterprise-t bővíthető operációs rendszer rövidítése. Ez olyan program, amely lehetővé teszi, hogy a gép szolgáltatásait és lehetőségeit valamely magas szintű nyelvből (pl. a

(BASIC-ből) közvetlenül vegyük igénybe. A számítógép legfőbb szolgáltatásai a különböző eszközök és perifériák, mint pl. a képernyő, a magnetofon, a sornyomtató. Az operációs rendszer egyik része ezeket az eszközöket segít kezelni, a másik pedig a tárral való gazdálkodást végzi.

AZ INPUT/OUTPUT RENDSZER

Az Enterprise számítógép rendkívül összetett, így az olyan egyszerűnek tűnő feladatok, mint egy karakter vagy sor kiírása a képernyőre vagy a sornyomtatóra is több száz vagy több ezer gépi kódú utasítást igényel. Az operációs rendszer lehetővé teszi, hogy az egyes eszközöket bizonyos formai szabályoknak eleget tevő azonosítókra hivatkozva kezeljük. (Pl. VIDEO:, DISK-n:) Minden írás, ill. olvasás az ún. csatornákon keresztül történik, ezek a csatornák az egyes eszközökkel kötik össze a programot. A rendszer a következő funkciókat képes ellátni:

KÓDSZÁM	FUNKCIÓ
0	Rendszer újraindítás
1	Csatorna megnyitása
2	Csatorna létrehozása és megnyitása
3	Csatorna lezárása
4	Csatorna lezárása és törlése
5	Karakter olvasása a csatornáról
6	Blokk olvasása
7	Karakter írása a csatornára
8	Blokk írása
9	Csatornaállapot olvasása
10	Csatornaállapot állítása és beolvasása
11	Speciális feladat végrehajtása
16	Egy rendszerváltozó írása, olvasása vagy átkapcsolása
17	Az egyik csatorna helyettesítése a másikkal
18	Az egyik csatorna átirányítása a másikra
19	Az alapértelmezésként használt eszköz nevének beállítása
20	A rendszerállapot beolvasása
21	Eszköz csatlakoztatása
22	A rendszerhatárok beolvasása
23	A felhasználó hatókörének beállítása
24	Egy szegmens lefoglalása
25	Egy szegmens felszabadítása
26	A bővítések vizsgálata
27	Csatornapuffer lefoglalása

KÓDSZÁM	FUNKCIÓ
28	A hibaüzenet küldése
29	Modul betöltése
30	Áthelyezhető modul betöltése
31	A idő beállítása
32	A idő kiolvasása
33	A dátum beállítása
34	A dátum kiolvasása

Ezeket a szolgáltatásokat bármely programnyelvből – így gépi kódból is – használhatjuk. Ilyenkor az RST 30H utasítással hívhatjuk meg a rendszert és az RST utasítás utáni byte-ba kell letennünk a kívánt funkció kódját. A funkciók teljes listája és a hívásuk pontos leírása az Enterprise Technikai Ismertetőben található.

TÁRKEZELÉS

Az operációs rendszer ún. ROM tárban van, de a rendszerváltozókat RAM-ban tárolja. Az Enterprise rendkívül nagy tár kezelésére is alkalmas. A gép a tárat 16 kbyte-os lapokra (nem azonos a videolapokkal!) osztja, s ezekből 256-ot tud használni, tehát összesen mintegy 4 Mbyte memóriát. A mi számítógépünk természetesen nem tartalmaz ennyit, csak 128 kbyte-nyit. A gép lelkét jelentő Z80-as processzor egyszerre csak négy ilyen lapot kezel, ahogy mi is egyszerre csak két lapot látunk egy könyv olvasása közben.

RENDSZERVÁLTOZÓK

Mivel lehetőségünk van a rendszerváltozók közvetlen beolvasására, módosítására vagy átkapcsolására, ezért most röviden leírjuk a fontosabbakat (részletes információt ezekről is a Technikai Ismertető ad):

SORSZÁM	RÖVIDÍTÉS	FUNKCIÓ
0	IRQ ENABLE STATE	b0- hang b2- 1 Hz b4- video b6- külső b1,b3,b5,b7 nulla kell legyen
1	FLAG SOFT IRQ	szoftver megszakítás, engedélyezés
2	CODE SOFT IRQ	szoftver megszakítás kódja
3	DEF TYPE	0 => magnetofon 1 => lemez

SORSZÁM	RÖVIDÍTÉS	FUNKCIÓ
4	DEF CHAN	aktuális csatornaszám
5	TIMER	1 Hz-cel számol visszafelé 0-nál megszakítást generál
6	LOCK KEY	nagybetű/kisbetű
7	CLICK KEY	0=> nincs billentyűhang
8	STOP IRQ	0=> STOP billentyűre megszakítás
9	KEY IRQ	0=> minden billentyűre megszakítás
10	RATE KEY	billentyű ismétlés 1/50 másodpercben
11	DELAY KEY	az ismétlés megkezdése előtti idő 1/50 másodpercben
12	TAPE SND	0=> magnetofonhang kikapcsolva
13	WAIT SND	
14	MUTE SND	0=> belső hangszóró bekapcsolva
15	BUF SND	takarógörbe
16	BAUD SER	soros vonal sebesség
17	FORM SER	soros vonal formátum
18	ADR NET	hálózati cím
19	NET IRQ	0=>hálózatihíváshatásáramegszakítás generálódik
20	CHAN NET	a blokkot küldő csatorna száma
21	MACH NET	feladó gép NET címe
22	MODE VID	video üzemmód
23	COLR VID	szín üzemmód
24	X SIZ VID	videolap x dimenziója,
25	Y SIZ VID	videolap y dimenziója
26	ST FLAG	0=> állapotkijelző sor van
27	BORD VID	keret színe
28	BIAS VID	paletta felső színei
29	VID EDIT	a csatorna, amire az EDITOR ír
30	KEY EDIT	a csatorna, amiről az EDITOR olvas
31	BUF EDIT	puffméret 256 byte-okban
32	FLG EDIT	elérhetőségjelzők
33	SP TAPE	0=> gyorstárolás kazettán
34	PROTECT	0=> nincs kazettavédelem
35	LV TAPE	kazetta felvétel erősség
36	REM1	0=> távirányító kimenet kikapcsolva
37	REM2	ugyanaz, csak a másik kimenetre
38	SPRITE	külső színprioritás
39	RANDOM IRQ	minden megszakításnál eggyel nő

HIBAÜZENETEK

A programban levő hibák felismerése és megtalálása gyakran igen nehéz és bonyolult feladat. A számítógép segít nekünk ebben is azzal, hogy ha hibát észlel a programban, akkor rövid üzenetet küld a hiba feltételezett okáról és kijelzi a hiba bekövetkezésének a helyét. Ne felejtsük el azonban, hogy a gép csak bizonyos hibákat képes felismerni! Ha például nem vesszük figyelembe a műveletek módosult végrehajtási sorrendjét egy kifejezés felírásakor, akkor könnyen azt tapasztalhatjuk, hogy a gép által kiírt eredmény nem egyezik az általunk várttal. A gép azonban nem jelez hibát, mert csak a formai (szintaktikai) vagy a programszerkezetbeli hibákat jelzi, a program számítási menetének (szemantika) a hibáját nem. Ha a program futása közben észlel hibát, akkor a hibaüzeneten kívül azt a sort is kiírja, ahol a feltételezett hiba van, ezzel is segítséget nyújtva a hibás sor kijavításához.

```
*** ungültiges Argument für SQR
300 PRINT SQR(Y)
```

Ezután a szokásos módon módosíthatjuk a sort. Az EXLINE, EXTYPE, EXSTRIN$ függvények az ilyen hibák programbeli kezelését könnyítik meg. Minden ilyen hibából vagy más rendkívüli eseményből adódó megszakításnak van egy kódja és a legtöbbhöz valamilyen hibaüzenet is tartozik. Ha nincs ilyen hibaüzenet, akkor a hiba típusa és azonosító kódszáma jelenik meg, pl.:

```
*Überlauf 1234
```

Az ilyen megszakítások kezelésére előre felkészülhetünk és ezeket a WHEN blokkok és a megszakításkezelő „HANDLER"-ek segítségével elintézhetjük. A bekövetkező megszakítás jellegétől függetlenül mindig a legbelső WHEN blokkhoz tartozó HANDLER-t hívjuk meg, ezekkel óvatosan kell bánnunk, hiszen nem tudhatjuk, hogy milyen megszakítás miatt került a vezérlés a kezelőre. Egyes hibák (pl. a szintaktikai hibák vagy a memória túlcsordulás) esetén ugyanis a RETRY típusú visszatérés stabilizálja a hibás állapotot, és a programot végtelen ciklusba juttatja. (L.: CONTINUE hatása a programon belül.)

Más hibák elfogása, mint pl. a nullával osztás vagy a negatív számból való négyzetgyökvonás nem okozhatja a program megsérülését.

ANGOL NYELVŰ HIBAÜZENETEK

0 – 999	User
1000 – 1999	Overflow
2000 – 2999	Subscript
3000 – 3999	Mathematical
4000 – 4999	Parameter
5000 – 5999	Storage exhausted
6000 – 6999	Matrix

7000 – 7999 File use
8000 – 8999 Input–output
9000 – 9999 Exos
10000 – 10999 Control
11000 – 11999 Graphical
12000 – 12999 Real–time
20000 – 20999 Syntax
30000 – System

1000 – Unexpected value given
1001 – Overflow in numeric constant
1002 – Overflow in numeric expression
1051 – Overflow in string expression
1106 – Overflow in string assignment (ie. string too long)

2001 – Array subscript out of bounds
3001 – Division by zero
3004 – Invalig argument to LOG
3005 – Invalig argument to SQR
3007 – Invalid argument to ASIN or ACOS

4000 – Error in DEF parameters
4002 – Argument to CHR$ out of range
4003 – Invalid argument to ORG
4004 – Index to SIZE out of range
4005 – Argument to TAB out of range
4008 – Index to LBOUND out of range
4009 – Index to UBOUND out of range
4301 – Error in CHAIN parameters

5000 – Insufficient memory
5100 – Insufficient stack space
5110 – Insufficient extension space
5120 – Insufficient ALLOCATE space

7000 – Invalid channel no.
7003 – Channel already open
7004 – Channel not open
7401 – TRACE channel not open

8001 – out of data in READ/INPUT
8101 – Numeric data expected

8201 – Invalid USING strong
8202 – No format item in USING string
8203 – USING format item too short

9208 – Casette CRC error
9209 – Editor – load file too big
9210 – Editor – keyboard channel error
9211 – Editor – keyboard channel error
9212 – Editor – video channel error
9213 – Network link already exists
9214 – Network address not set
9215 – Cannot use both serial and network
9216 – Invalid beam position
9217 – Invalid cursor coordinates
9218 – Invalid row number to scroll
9219 – Invalid video page file
9220 – Invalid display parameters
9221 – Invalid video mode
9222 – Invalid video page size
9223 – Sound queue full
9224 – Envelope storage full
9225 – Envelope too big
9226 – Function key string too long
9227 – Protection violation
9228 – Unexpected end of file
9229 – STOP key pressed
9230 – Invalid escape sequence
9231 – Call not supported by this device
9232 – Invalid unit number
9233 – Device already in use
9234 – Invalid special function call
9235 – invalid date or time value
9236 – End of file module
9237 – Invalid relocatable module
9238 – Unknown module type
9239 – Invalid Enterprise file header
9240 – Unrecognised command string
9241 – Invalid device description
9242 – Unknown EXOS variable number
9243 – Invalid user boundary
9244 – Cannot free segment
9245 – No free segment

9246 – Insufficient video memory
9247 – Insufficient memory
9248 – Channel open error
9249 – Channel already exists
9250 – Device does not exist
9251 – Channel does not exist
9252 – EXOS stack overflow
9253 – Invalid EXOS string
9254 – EXOS function call not allowed
9255 – Invalid EXOS function code

10002 – Return without GOSUB
10004 – No CASE selected
10005 – Program does not exist

20000 – Not understood
20001 – Invalid line number
20002 – Invalid line number range
20004 – Line number does not exist
20010 – Cannot do specified RENUMBER
20020 – Continue not possible
20030 – Indentifier expected
20031 – String identifier expected
20032 – Array identifier expected
20034 – Type mismatch
20040 – Variable not initialised
20041 – Identifier declared twice
20042 – Identifier too long
20043 – Missing closing quotes
20050 – Missing end of block
20051 – Invalid end of block
20052 – Too many nested blocks
20060 – Invalid machine option use
20071 – Statement in immediate mode
20072 – Command in program
20073 – Statement not allowed after THEN
20074 – Invalid multi–statement line
20075 – Line too long
20080 – Invalid file format
20081 – Programs do not VERIFY

30000 – BASIC data has been corrupted

NÉMET NYELVŰ HIBAÜZENETEK

1000 – unerwarterer Wert eingegeben
1001 – Überlauf numerischer Konstante
1002 – Überlauf numerischen Ausdrucks
1051 – Überlauf des String–Ausdrucks
1106 – Überlauf der String–Zuweisung

2000 – falscher index
2001 – Feld auserhalb des erlaubten Bereichs
2002 – falscher index

3000 – mathematisch
3001 – Division durch Null
3004 – ungültiges Argument für LOG
3005 – ungültiges Argument für SQR
3007 – ungültiges Argument für ASIN oder ACOS

4000 – ungültige DEF–Parameter
4001 – Parameter
4002 – CHR$–Argument auserhalb des erlaubten Bereichs
4003 – ungültiges Argument für ORD
4004 – SIZE–Index auserhalb des erlaubten Bereichs
4005 – TAB–Argument auserhalb des erlaubten Bereichs
4008 – LBOUND–Index auserhalb des erlaubten Bereichs
4009 – UBOUND–Index auserhalb des erlaubten Bereichs
4301 – CHAIN–Parameter–Fehler

5000 – Speicherplatz zu klein
5100 – Stapel–Speicher zu klein
5110 – Speichererweiterung zu klein
5120 – ALLOCATE–Speicher zu klein

6000 – Matrix
7000 – Datei–Benutzung
7001 – ungültige Kanal–Nummer
7003 – Kanal schon offen
7004 – Kanal nicht offen
7401 – TRACE–Kanal nicht offen

8000 – Ein–/ Ausgabe
8001 – Keine Daten mehr für READ/INPUT
8101 – numerische Daten erwartet

8201 – ungültiger USING–String
8202 – keine Format–Angabe im USING–String
8203 – USING–Format–Angabe zu kurz

9000 – EXOS (Betriebssystem)
9110 – Datei nicht gefunden
9111 – Diskette voll
9207 – Copyright INTELLIGENT SOFTWARE LTD
9208 – Kasetten–Lese–Fehler
9209 – Editor:Überlange der zu ladenden Datei
9210 – Fehler in der zu ladenden Datei
9211 – Editor: Tastatur–Kanal–Fehler
9212 – Editor: Video–Kanal–Fehler
9213 – Netzwerk–Verbindung schon vorhanden
9214 – Netzwerk–Adreese nicht gesetzt
9215 – Schnittstelle schon belegt
9216 – Graphik–Cursor–Koordinaten auserhalb des erlaubten Bereichs
9217 – Cursor–Koordinaten auserhalb des erlaubten Bereichs
9218 – zu scrollender Zeilenbereich auserhalb des erlaubten Bereichs
9219 – ungültige Bildschirmseiten–Datei
9220 – Display–Parameter auserhalb des erlaubten Bereichs
9221 – Bildschirm–Modus auserhalb des erlaubten Bereichs
9222 – Bildschirm–Format auserhalb des erlaubten Bereichs
9223 – Ton–Stapel voll
9224 – Hüllkurvenstapel voll
9225 – Hüllkurve auserhalb des erlaubten Bereichs
9226 – Funkstionstasten–String auserhalb des erlaubten Bereichs
9227 – Schutz–Verletzung
9228 – Datei–Ende
9229 – STOP–Taste gedrückt
9230 – ungültiges Steuerzeichen
9231 – Aufruf dieses Gerates nicht möglich
9232 – Gerate–Nummer auserhalb des erlaubten Bereichs
9233 – Gerat Wird schon benutzt
9234 – ungültiger Aufruf einer Spezialfunktion
9235 – Datum oder Uhrzeit auserhalb des erlaubten Bereichs
9236 – Ende der Modul–Datei
9237 – Modul nicht verlegbar
9238 – unbekannte Modulart
9239 – ungültiger Datei–Anfang
9240 – unbekannter Befehls–String
9241 – ungültige Gerat–Bezeichnung

9242 – unbekannte EXOS–Variablen–Nummer
9243 – EXOS (Betriebssystem)
9246 – Video–Speicher–Platz zu klein
9247 – Speicherplatz zu klein
9248 – EXOS (Betriebssystem)
9249 – Kanal schon vorhanden
9250 – Gerat nicht vorhanden
9251 – KAnal nicht vorhanden
9252 – EXOS–Stapel–Speicher Überlauf
9253 – ungültiger EXOS–String
9254 – verbotener EXOS–Funktionsaufruf
9255 – ungültiger EXOS Funktions–Code

10000 – Programmablauffehler
10002 – RETURN ohne GOSUB
10004 – kein CASE angegeben
10005 – Program nicht vorhanden
11000 – Graphisch
12000 – Echt–Zeit
13000 – System

20000 – nicht verstanden
20001 – Zeilennummer ausrhalb des erlaubten Bereichs
20002 – ungültiger Zeilennummer–Bereich
20004 – Zeilennummer ist nicht vorhanden
20010 – RENUMBER kann nicht ausgeführt werden
20020 – CONTINUE nicht möglich
20030 – Identifikator wird erwartet
20031 – String–Identifikator wird erwartet
20032 – Feld–Identifikator wird erwartet
20034 – Typen–Verwechslung
20040 – Variable nicht initialisiert
20041 – Identifikator doppelt initialisiert
20042 – Identifikator zu lang
20043 – Anführungszeichen nicht geschlossen
20050 – Blockende fehlt
20051 – ungültiges Blockende
20052 – zu viele Blocks ineinander verschachtelt
20060 – ungültige Anweisung an das Betriebssystem
20071 – Programm–Befehl im Direktmodus unzulassig
20072 – Direktmodus–Befehl im Programm unzulassig
20073 – Befehl nach THEN ungültig

20074 – ungültige Mehr–Befehls–Zeile
20075 – Zeile zu lang
20080 – ungültiges Datei–Format
20081 – Programme stimmen nicht überein

A HIBAÜZENETEK MAGYAR MEGFELELŐI

Általános hibatípusok:

0 –	999	Felhasználói hiba
1000 –	1999	Túlcsordulás
2000 –	2999	Indexhiba tömbnél
3000 –	3999	Matematikai hiba
4000 –	4999	Paraméterhiba
5000 –	5999	Nem elegendő tár
6000 –	6999	Mátrixhiba
7000 –	7999	File–kezelési hiba
8000 –	8999	Beolvasás/kiírás hiba
9000 –	9999	EXOS hiba
10000 –	10999	Vezérlési hiba
11000 –	11999	Grafikai hiba
12000 –	12999	Időzítési hiba
20000 –	20999	Szintaxis hiba
30000 –		Rendszerhiba

Egyedi hibaüzenetek:

1000 – A megadott érték nem megfelelő
1001 – Túlcsordulás a számkonstansban
1002 – Túlcsordulás a számkifejezésben
1051 – Túlcsordulás a szövegkifejezésben
1106 – Túlcsordulás a értékadásban (pl. a karakterlánc túl hosszú)

2001 – Tartományon kívüli tömbindex
3001 – Nullával osztás
3004 – Hibás LOG argumentum
3005 – Hibás SQR argumentum
3007 – Hibás ASIN vagy ACOS argumentum

4000 – Hiba a DEF paramétereiben
4002 – A CHR$ argumentuma nem megfelelő
4003 – Hibás ORD argumentum

4004 – SIZE–beli index a megengedett tartományon kívül
4005 – A TAB argumentuma nem megfelelő
4008 – Az LBOUND–beli index hibás
4009 – Az UBOUND–beli index hibás
4301 – Hiba a CHAIN paramétereiben

5000 – A tár betelt
5100 – A veremtár betelt
5110 – A bővítéstár betelt
5120 – Az ALLOCATE tár betelt

7001 – Hibás csatornaszám
7003 – A csatorna már nyitva van
7004 – A csatorna nincs nyitva
7401 – A TRACE csatorna nincs nyitva

8001 – Nincs elég adat a READ/INPUT–hoz
8101 – Szám kellene
8201 – Hibás USING karakterlánc
8202 – Nem formátumelem a USING–ban
8203 – A USING formátumelem túl hosszú

9208 – Kazetta ellenőrzőkód hiba (CRC)
9209 – Editor – a betöltendő file túl nagy
9210 – Editor – billentyűzet-csatorna hiba
9211 – Editor – billentyűzet-csatorna hiba
9212 – Editor – videohiba
9213 – A hálózat összekapcsolás már létezik
9214 – A hálózatcím nincs beállítva
9215 – Egyszerre nem lehet a soros vonalat és a hálózatot is használni
9216 – Hibás rajzolópont pozíció
9217 – Hibás kurzorpozíció
9218 – Hibás sorszám a képernyőléptetéshez
9219 – Hibás videolap–file
9220 – Hibás képernyő paraméter
9221 – Hibás video–üzemmód
9222 – Hibás videolap–méret
9223 – Hangtár megtelt
9224 – Az ENVELOPE tár megtelt
9225 – ENVELOPE túl nagy
9226 – A funkcióbillentyű szövege túl hosszú

9227 – Tárvédelem megsértése
9228 – Hibás adatvége jel
9229 – A STOP billentyű lenyomva
9230 – Hibás kilépési sorozat
9231 – A hívás nem támogatott ezen az eszközön
9232 – Hibás egységszám
9233 – Az eszközt már használják
9234 – Hibás speciális funkció hívás
9235 – Hibás dátum vagy idő megadás
9236 – Adatvége modul
9237 – Hibás áthelyezhető modul
9238 – Ismeretlen modultípus
9239 – Hibás file–fejléc
9240 – Ismeretlen parancs
9241 – Hibás eszközleíró
9242 – Ismeretlen EXOS rendszerváltozó
9243 – Hibás felhasználói határ
9244 – A szegmens nem szabadítható fel
9245 – Nincs szabad szegmens
9246 – A videotár elfogyott
9247 – A tár megtelt
9248 – Csatornamegnyitási hiba
9249 – A csatorna már létezik
9250 – Az eszköz nem létezik
9251 – A csatorna nem létezik
9252 – A rendszerverem megtelt
9253 – Hibás EXOS szöveg
9254 – Az EXOS funkció hívás nem engedélyezett
9255 – Hibás EXOS funkciókód

10002 – RETURN GOSUB nélkül
10004 – Nincs CASE ág
10005 – A program nem létezik

20000 – Nem érthető
20001 – Hibás sorszám
20002 – Hibás sortartomány
20004 – A sorszám nem létezik
20010 – Végrehajthatatlan RENUMBER
20020 – Nem folytatható
20030 – Azonosító hiányzik
20031 – Karaktersorozat azonosító hiányzik

20032 – Tömbazonosító hiányzik
20034 – Tipuseltérés
20040 – A változónak nincs értéke
20041 – Kétszer deklarált változó
20042 – Az azonosító túl hosszú
20043 – Hiányzik a lezáró macskaköröm
20050 – Hiányzik az blokkvége jel
20051 – Hibás blokkvége jel
20052 – Túl sok egymásba ágyazott blokk
20060 – Hibás gépi feltétel használat
20071 – Ez az utasítás nem lehet parancsmódban
20072 – Ez a parancs nem lehet programban
20073 – THEN után nem állhat ilyen utasitás
20074 – Hibás többutasításos sor
20075 – A sor túl hosszú
20080 – Hibás file–formátum
20081 – A program nem azonos

30000 – A rendszer változói megsérültek

A SZÖVEGSZERKESZTŐ ÜZENETEI

A szövegszerkesztő üzenetei a gép nyelvének megfelelően vagy német, vagy angol nyelvűek. A továbbiakban az egyes funkcióknál mindkét nyelven megadjuk a kiírt üzenetet, valamint a jelentését magyarul.

A szövegszerkesztőbe való belépéskor a gép mindig a „Press ENTER to exit BASIC" felirattal kéri a parancs megerősítését, azaz azt, hogy nyomjuk le az ENTER billenyűt, ha valóban ki akarunk lépni a BASIC rendszerből. Ennél és valamennyi más szövegszerkesztő funkciónál az ESC billentyűvel tudjuk a parancsot törölni, azaz jelen esetben a BASIC rendszerbe visszatérni.

A funkcióbillentyűk használatakor a következő üzenetekkel találkozhatunk: (Először mindig a német, aztán az angol, majd a magyar nyelvű szöveget közöljük.)

F1:

Zum LADEN des Dokuments: Titel
eintippen, dann ENTER drücken
LOAD the document.
Type filename and then press ENTER
Szöveg betöltése következik.
Gépelje be a szöveg nevét, aztán nyomja le az ENTER-t!

F2:

Zum ABSPEICHERN des Dokuments: Titel
eintippen, dann ENTER drücken
SAVE the document.
Type filename and then press ENTER
Szöveg kiírása következik.
Gépelje be a szöveg nevét, aztán nyomja le az ENTER-t!

F3:

Falls Parallel-Drucker, so ENTER.
Sonst Vorrichtungsnamen eingeben:
Press ENTER for parallel printer,
or type device-name:
A párhuzamos csatolású sornyomtató használata esetén nyomja le az ENTER-t, egyébként pedig adja meg az eszköz nevét!

F4:

A billentyű hatására a szövegszerkesztő funkcióit röviden ismertető ún. HELP-lapot ír ki a gép, majd az F4 újabb leütésére egy másikat és ettől kezdve ezt a kettőt váltogatja, amíg az ENTER leütésével vissza nem térünk a szövegszerkesztéshez. A HELP-lapok kiírása nem módosítja a tárolt szöveget, tehát az nem sérül meg!

A HELP-lapok szövege a következő (csak magyar nyelven közöljük):

Az első lap:

ctrl F1 – Bekezdés újraformázása
ctrl F2 – Sor középre állítása
ctrl F3 – Tabulátor beállítása/törlése
ctrl F4 – Bal margó beállítása
ctrl F5 – Margók felfüggesztése
ctrl F6 – Bekezdés mozgatása felfelé
ctrl F7 – Sor színének megváltoztatása
alt F1 – Bekezdés újraformázása igazítással
alt F2 – Tabulátorok törlése
alt F3 – Vonalzósor be/ki kapcsolása
alt F4 – Jobb margó beállítása
alt F5 – Margók és tabulátorok törlése
alt F6 – Bekezdés mozgatása lefelé
alt F7 – Bekezdés színének változtatása

Nyomja le az ENTER-t, hogy visszatérjen a szerkesztéshez, vagy az F4-et további információért.

A második lap:

↑ ↓	Egy sorral fel/le
schift ↑ ↓	Egy lappal fel/le
ctrl ↑ ↓	Egy bekezdéssel fel/le
← →	Betűnként balra/jobbra
schift ← →	Sor elejére/végére
ctrl ← →	Szavanként balra/jobbra
ERASE	Betű törlése balra
schift ERASE	Sor törlése balra
ctrl ERASE	Szó törlése balra
DEL	Betű törlése jobbra
schift DEL	Sor törlése jobbra
ctrl DEL	Szó törlése jobbra
INS	Szóköz beszúrása
schift INS	Sor beszúrása
ctrl DEL	Beszúrás üzemmód be/ki

F5:

ENTER stellt 80-Zeichenmodus bei
Löschung des Textes ein.
ESC für zurück zum Editor.
Press ENTER for 80 column mode, this
will delete your text.
Press ESC to return to editing.

Nyomja le az ENTER-t, ha át akar térni 80 karakteres üzemmódra, de ekkor törlődik a korábbi szöveg. Ha nem akar áttérni, akkor az ESC-et nyomja le.

F6:

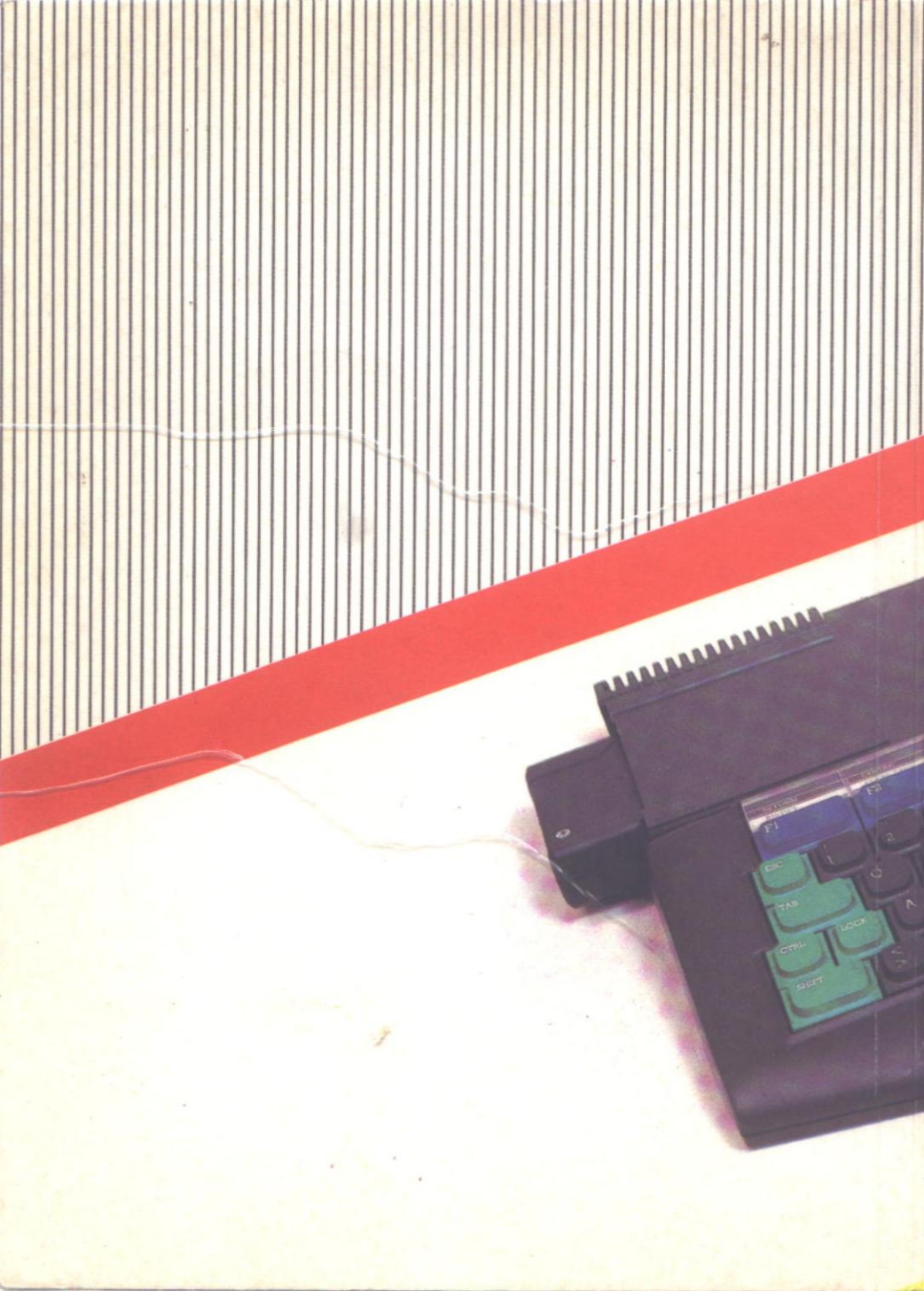
Ugyanaz, mint az F5, csak 40 karakteres üzemmóddal.

F7:

Nincs üzenet.

F8:

Um BASIC einzuschalten: ENTER drücken.
Press ENTER to start BASIC.
Üsse le az ENTER-t, ha vissza akar térni a BASIC-be!


F1
F2
ESC
TAB
CTRL
LOCK
SHIFT