Bencsikné Takács Márta

TANÁRI SEGÉDKÖNYV

a C 16-os számítógéphez készült
általános iskolai
feladatgyűjteményhez

TANÁRI SEGÉDKÖNYV

Commodore 16 számítógépre általános iskolásoknak

Budapest, 1986

Készítette:
BENCSIKNÉ TAKÁCS MÁRTA

Lektorok:
DR. KŐRÖSNÉ MIKIS MÁRTA
MAROSVÁRY ERIKA

Felelős szerkesztő:
TARR KÁLMÁNNÉ

Készült az OPI és a TII között létrejött állami szerződés keretében

ELŐSZÓ

Az általános iskolában számítástechnikát tanító tanárok számára nagy problémát jelentett, hogy nem volt forgalomban megfelelő segédanyag sem a tanárok, sem a diákok részére.

Örömmel javasolhatjuk az OPI és a TII együttműködésével a NOVOTRADE RT. gondozásában megjelent Hetedhét C 16, Felhasználói kézikönyv C 16 és az Általános iskolai feladatgyűjtemény C 16-os számítógépre című kiadványokat.

A feladatgyűjtemény példáinak nagy részét a pécsi Apáczai Csere János Nevelési Központban folyó kutatás során megoldottuk a gyerekekkel, néhányat maguk a gyerekek készítettek.

A témák kifejtésénél igyekeztünk a tanítás során felmerült problémákat is összegyűjteni, mert tapasztalataink szerint ezek gyakran előfordulnak.

A közölt tanmenetek, mint a nevükből is látszik, javaslatok. A kollégák által használt tanmenetnek természetesen alkalmazkodnia kell a helyi adottságokhoz.

Ha úgy találja, hogy csoportjának nem felel meg az általunk tervezett tempó, inkább szánjon több időt a gyakorlásra. Többet ér a kevesebb, de jól megértett anyag, mint a felszínes tudás!

Reméljük, hogy a feladatgyűjtemény és a segédkönyv együttes használatával valóban értékes segítséget tudunk nyújtani a témával foglalkozó kollégáknak.

BEVEZETÉS

A számítógépről

A számítástechnika fakultáció első foglalkozásán azt tapasztaljuk, hogy a gyerekek nagy része rendelkezik már bizonyos előismeretekkel. Ezek az ismeretek azonban általában nem alkotnak rendszert, hanem innen-onnan felcsipegetett információkból állnak.

Nagyon fontos, hogy már az első foglalkozáson felkeltsük a gyerekek érdeklődését a téma iránt. Ezt sok érdekes adalék közlésével érhetjük el.

Ismertessük meg a gyerekeket a számítógép kialakulásának történetével, a gép felépítésével, felhasználási területeivel.

Segítségül vázlatosan közöljük a főbb pontokat.

A számítógép kialakulása

1. Első leletek: i. e. I. századból Antikythera planetáriuma, amely csillagászati és hajózási számításokra szolgált. Abakusz, másnéven golyós tábla.

2. Mechanikus számológépek: 1623 Schickard, 1642 Pascal készít összeadó és kivonó gépet. Leibnitz a négy alapművelet elvégzésére alkalmas gépet készít. 1822 Babbage leírja egy programvezérlésű digitális automata tervét. 1890 Hollerith megoldja az USA népszámlálási adatainak feldolgozását, megalkotja a lyukkártyát stb. Megalapítja az IBM céget.

3. Elektromos számológépek jönnek létre a XX. században. Ezek fejlődnek tovább a mai asztali és zsebszámológépekké.

4. Elektronikus számítógépek: 1941 Zuse felépíti az első programvezérlésű külső programozáson alapuló számítógépet.

Neumann János elvei alapján készülnek az első mai értelemben vett számítógépek:

1. generáció: Elektroncsöves gépek (1950–57)
2. generáció: Tranzisztoros gépek (1958–64)
3. generáció: Nyomtatott áramkörös gépek (1965–77)
4. generáció: Mikroprocesszoros gépek (1977–)

(Természetesen a közölt évszámok nem kezelhetők mereven, csak a jobb tájékozódást szolgálják.)

Napjainkban folynak a kutatások az 5. generációs számítógépek létrehozása érdekében.

A számítógép felépítése

VEZÉRLŐEGYSÉG ARITMETIKAI-
LOGIKAI
EGYSÉG

BEMENETI
EGYSÉG

KIMENETI
EGYSÉG

OPERATÍV TÁR

HÁTTÉRTÁRAK

Az egyes egységek feladata

Bemeneti egység: A program és az adatok bevitelére és a géppel való kapcsolattartásra használjuk.

Kimeneti egység: A gép üzeneteinek és a feladatmegoldások eredményeinek kivitelére szolgál.

Operatív tár: A program és az adatok tárolására szolgál.

Aritmetikai-logikai egység: A programban szereplő műveletek, összehasonlítások zajlanak itt.

Vezérlőegység: Biztosítja a műveletek végrehajtásának helyes sorrendjét.

Háttértárak: Az adatok hosszabb ideig való megőrzésére szolgálnak.

A Commodore 16-os gépnél csak bemeneti egységként a billentyűzet, csak kimeneti egységként a képernyő (monitor) és a sornyomtató, dialóg eszközként (bemeneti és kimeneti is lehet) a kazettás egység, ill. a lemezmeghajtó egység szolgálhat.

A Commodore 16-os gépnél az operatív tár két részre osztható:

ROM (Read Only Memory): Csak olvasható memóriát jelent. A ROM-ba a gyártó cég olyan információkat éget bele, amelyekre a számítógépnek minden helyzetben szüksége lehet már a bekapcsolás pillanatában. Ilyen információ pl. a gép operációs rendszere és a BASIC nyelv.

RAM (Read Random Access Memory): A felhasználó ezt a memóriaterületet szabadon használhatja, program, ill. adatok tárolására. Kikapcsoláskor a RAM tartalma törlődik. Ha a beírt programra már nincs szükség, azt a NEW parancs begépelésével is kitörölhetjük.

Bit, byte, kbyte

A számítógép működésének alapja az áramimpulzus megléte, ill. hiánya az adott helyen.

Bit: Ha a megfelelő helyen van áramimpulzus, akkor a bit logikai értéke 1, különben 0.

Byte: A számítástechnikában használatos összes karakter száma nem haladja meg a 2↑8 értéket. Így egy karakter ábrázolására nyolc pozíciót használnak. Ennek a nyolc bitből álló egységnek a neve a byte.

kbyte: A tárak kapacitásának jellemzésére a kbyte-ot használjuk. 1 kbyte= 2 ↑ 10 byte=1024 byte.

A számítógép felhasználásának főbb területei:

A számítógép alapfeladata mindig adatok feldolgozása!

- Ügyvitel: számlázások, bérszámfejtés, nyilvántartások vezetése.
- Folyamatok modellezése, elemzése.
- Folyamatok ellenőrzése, vezérlése.
- Robottechnika.
- Műszaki rajzok készítése, elemzése.
- Oktatás.
- Egyéb: zene, játékok, „műalkotások" készítése, pl. grafika, versírás.

Pl. a következő verset egy számítógép készítette 1957-ben:

Az éj e macskánál is feketébb
a holdtájak olyan olvataggá váltak.
Nem ismert öröm tör fény felé,
s parthoz vetődik mint fáradt szárnyak.
Egy kínzott árva nomád ott bolyong
s a szakadék havas
mélye hívja, várja.

Feledve a félszt, panaszkodón
a szakadéknál háborgón bolyong.
A tűnt félelem felhatol a mennyig
ahogyan a sirály és a szél,
Eső szemerkél
és úgy borong kint,
s a gyertyák mindegyike csonkig ég,
És a lepkék ott a tűznél egyre
körtáncot lejtenek Buster emlékére.

A számítógép használatára mutathatunk néhány demó programot is az általunk használtak közül.

1. A BILLENTYŰZET HASZNÁLATA

A kollégák jó része úgy érzi, hogy fölösleges időpocsékolás ezzel a témával foglalkozni, hiszen a programozói munkától viszi el az amúgy is szűkre szabott időt.

Be kell azonban látni, hogy a későbbi gördülékeny munkához szükséges bizonyos szintű gépismeret. Az itt tanult, gyakorolt kezelési fogásokkal eltöltött idő a későbbiekben megtérül.

A legfontosabb kezelési ismeretek:

- képernyőtörlés (SHIFT+CLR HOME)
- kurzor a bal felső sarokba képernyőtörlés nélkül (CLR HOME)
- karakterek használata, a gyors leütés gyakorlása
- bal oldali grafikus jelek (C= + a billentyű)
- jobb oldali grafikus jelek (SHIFT+ a billentyű)
- kurzor mozgatása (megfelelő irányba mutató nyilakkal)
- betűnagyság váltása (C= + SHIFT)
- betűk inverze (CTRL+RVS/ON)
- visszatérés alapállapotba (CTRL+RVS/OFF)
- törlés jobbról balra (INST DEL)
- törlés balról jobbra (SPACE)
- beszúrás két betű közé (SHIFT+INST DEL).

Fontos, hogy a gyerekek kipróbálhassák a bemutatott lehetőségeket és önállóan, játszva próbálkozhassanak a gép működtetésével!

A témának igen nagy a motivációs hatása, mert gyakorlatilag előismeretek nélkül is sikerélményt biztosít. A feladatgyűjtemény 1. fejezetének példáit vizsgálva, a fontosabb kezelési tudnivalókat gyakorolhatják a gyerekek. Az 1. fejezet első programját célszerű a foglalkozás előtt rögzíteni és a foglalkozáson beolvasni a gépbe. Bár a gyerekek igen lelkesen vállalják a program begépelését, ez azonban nagyon hosszú ideig tart, és általában sok hibát is tartalmaz. A program futtatása jó gyakorlási lehetőség a tanulók számára.

Ha van idő, akkor a téma végén közölt játékokat is ki lehet próbálni, akár a jó munka jutalmaként is! Igaz, hogy mindhárom játékot csak ketten játszhatják, de a gyerekek szívesen szurkolnak is egymásnak!

2. SZÁMOK ÉS MŰVELETEK

A kalkulátor üzemmód parancs üzemmódot jelent, azaz a begépelt utasítások mint parancsok azonnal végrehajtódnak.

A parancs üzemmódban a géppel általában matematikai műveleteket hajtunk végre, ezért nagyon fontos a számítógépes műveletvégzés szabályainak tisztázása.

Ez a műveletek prioritásának vizsgálatát jelenti. Mivel a gyerekek a hetedik osztályos matematikában találkoznak ezzel a problémával, ezért itt csak az ismeretek felfrissítéséről van szó.

A műveletvégzés sorrendje:

1. zárójelben levő kifejezés
2. hatványozás, gyökvonás
3. szorzás, osztás
4. összeadás, kivonás.

Fel kell hívni a figyelmet arra, hogy a törtvonal zárójelet pótol, és a zárójellel törtvonalat is pótolhatunk. Azonos erősségű kifejezések esetén a balról jobbra szabályt érvényesíti a számítógép.

Itt jó lehetőség kínálkozik arra, hogy egy egyszerű példán keresztül rámutassunk: bár a számítógép igen gyorsan képes műveleteket elvégezni, gondolkodni azonban nem tud!

Pl. 5−7−5+7 =

A gyerekek rögtön látják, hogy a számokat átcsoportosítva, szinte számolás nélkül megkapják az eredményt:

(5−5)+(7−7)=0.

Mutassuk meg, hogy a számítógép erre nem képes. A gyerekek többsége használt már számítógépet matematikai feladatok elvégzésére, így nem idegen tőlük a géppel való műveletvégzés. Hívjuk fel a figyelmüket a számológép és a számítógép üzemeltetése közti különbségekre, mutassuk be és gyakoroltassuk be a számítógépnél használatos műveleti jeleket: +; −; *; /.

Itt célszerű bevezetni a RETURN billentyű és a PRINT utasítás használatát. Nagyon fontos, hogy a gyerekek megértsék, hogy a képernyőre való gépelés nem azonos a memóriába való bevitellel.

Sok példa, könnyen megoldható gyakorlati feladat alapján érdemes a gépkezelési ismereteket fejleszteni, mert ez a tudás a programozás során felgyorsítja a munkánkat. A tapasztalat szerint a témának igen erős a motivációs hatása, mivel a gyerekek ezt a munkát, mint „számítógépezést" élik meg és minimális tudásanyaggal is sikerélményeket érhetnek el.

A feladatok megoldása során előkerül a törtek, ill. a tizedes törtek használatának problémája. Mutassuk be, hogy a gép a tizedesvessző helyett tizedespontot alkalmaz.

A bevezetett utasítás birtokában nemcsak matematikai feladatok oldhatók meg, hanem mód nyílik a szövegek kiíratására.

Tegyük lehetővé, hogy a gyerekek maguk is készíthessenek feladatokat, és azokat önállóan meg is oldhassák!

2.1 Példák

Mint azt a feladatgyűjteményben is közöltük, a megadott megoldások egy lehetséges megoldást kínálnak.

2.2 A változók

A számítógép memóriáját szemléletesen egy sokfiókos szekrényhez szokták hasonlítani. A fiókban a gép adatokat tud tárolni, egy fiókba egy adat fér. A fiókokat a későbbi felismerhetőség kedvéért meg szokták jelölni. Ezt a jelet azonosítónak is nevezik. A fiókok tartalma a változó, mert tetszés szerint módosítható.

A változók jelölésére több lehetőség van:

- egy betű használata: A, B, C . . .
- egy betű és egy szám: A1, B7 . . .
- két betű használata: AB, BC . . .

A harmadik módszer használata megengedett, de nem ajánlott, mivel a számítógép utasításkészletében több kétbetűs is található, pl. IF, OR, TO . . .

Ha a gép ilyen nevű változót talál, azt utasításként értelmezi és jobb esetben csak szintaktikai hibát jelez.

Számítógépünk három változótípust különböztet meg:

– Valós típusú: Az ilyen típusú változókba bármilyen valós szám elhelyezhető. Jele: A, B1, AB . . .

– Egész típusú: A változóban csak egész számok szerepelhetnek. Ha nem egész számot helyeznek a változóba, azt a gép a tizedespont után csonkolva jegyzi meg. Pl. 3.5→3. Jele: A%, B1%, AB% . . .

– Szöveges típusú: A változóba helyezett karakter szövegként értelmeződik, még akkor is, ha történetesen számot gépeltünk be. Jele: A$, B1$, AB$ . . .

Az értékadás:

Az azonosítót egyenlővé tesszük a megjegyzendő értékkel. Pl. A=1, B1=8.9, A%=22, A$="szöveg".

Fontos megjegyezni, hogy a szöveget mindig idézőjel közé kell zárni az értékadáskor.

Hangsúlyozni kell, hogy a számítástechnikában használt egyenlőségjel tulajdonképpen a definiáló egyenlőségnek felel meg, vagyis az értelme csak balról jobbra haladva igaz. Tehát A=B és B=A kifejezések tartalmilag különbözők.

Ha már ennél a témánál sikerült ezt az ismeretet átadnunk, akkor a programozás tanítása során sok fölösleges problémától megkíméljük a gyerekeket.

Nagyon fontos, hogy a gyerekek sok egyszerű példán keresztül tanulják meg a feladatnak megfelelő változótípus kiválasztását.

Mutassuk be, és nyomatékosan hívjuk fel a gyerekek figyelmét arra, hogy matematikai műveletek csak valós ill. egész típusú változókkal végezhetők, még akkor is, ha a szöveges változó számot tartalmaz.

A szöveges változóval végezhető egyetlen művelet az összefűzés, más néven konkatenáció. Összefűzéskor az összefűzött változók tartalmát szöveges változóként kezeli a gép. Pl.:

1. A$="A", B$="B", D$=A$+B$
   D$ változó tartalmát kiíratva:
   AB jelenik meg.
2. A$="1", B$="2", D$=A$+B$
   D$ változó értékét kiíratva:
   12 jelenik meg.

Tapasztalat szerint a gyerekek kezdetben keverik a különböző változótípusokat. Mivel azonban ezek biztos tudására a későbbiekben nagy szükség van, ezért érdemes időt szánni a gyakorlásra. Az ilyen típusú feladatok megoldása a PRINT utasítás, ill. a vessző és a pontosvessző gyakorlására is jó lehetőség.

3. A PROGRAMOZÁS

A program sorszámmal ellátott utasítások egymásutánja.

A jó programozás alapja az átgondolt tervezés. A programozó feladata, hogy kikeresse a feladat megoldására számba jöhető lehetőségek közül a legmegfelelőbbet.

A program megírása előtt megoldási tervet célszerű készíteni, amely gyakran a folyamatábra.

A program írásakor figyelni kell a nyelv nyelvtani szabályainak betartására, az ún. szintaktikára.

A programok számozása:

A sorszám nélküli sor mint parancs értelmeződik, és ha lehetséges, a gép azonnal végrehajtja. Ha nem végrehajtható, a gép hibát jelez.

A számozást célszerű tízesével végezni, mert így a program bővítésére bármely két programsor között lehetőség van. Ha mégsincs elegendő hely, vagy a kész program sorszámozásán változtatni szeretnénk, úgy célszerű a RENUMBER utasítást használni. A programot kilistázva, a sorok 10-től tízesével számozottak. Ha így nem tetszik a sorszámozás, a RENUMBER A,B begépelésével tetszőleges kezdőérték és lépésköz állítható be, ahol A a kezdőérték, B a lépésköz. A RENUMBER utasítás hatására nemcsak a sorszámok, hanem az ugró utasításokban szereplő címek is átszámozódnak, így a program javítás nélkül futtatható. A sorszámozást a gépre is bízhatjuk.

Az AUTO utasítás után begépelve az első programsort, a gép ehhez igazodva tízesével növekvő sorszámokat ad automatikusan. Az AUTO üzemmódból egy üres sor begépelésével léphetünk ki.

A gép a programsorokon mindig a kisebb sorszámútól a nagyobb felé halad, kivéve, ha külön utasítással egy másik programsorra küldjük.

A Commodore 16 a programsor begépelésekor nem jelzi az esetleges szintaktikai hibákat. Azok csak a program futtatásakor jelentkeznek.

A SINTAX ERROR IN... megjelenésekor a HELP billentyűt megnyomva a gép automatikusan kiírja a képernyőre a hibás sort. A hibás sor a képernyőn villogva jelenik meg. A javított részen álló betűk íródnak ki. Így a javítás is könnyen ellenőrizhető.

A kész programot tesztelni kell. A tesztelést a feladat szempontjából kritikus értékekre érdemes elvégezni.

A kész, hibátlan programokat érdemes kazettán, ill. floppyn (hajlékony mágneses lemezen) rögzíteni. Ha van lehetőség sornyomtató használatára, célszerű a programot kiíratni, mert így az archiválható is, és a nagyobb terjedelmű program kinyomtatva könnyebben elemezhető.

3.1 A PRINT utasítás

„A billentyűzet használata" és a „Számok és műveletek" témákra fordított idő és energia itt térül majd meg. A már ismert és begyakorlott funkciókat könnyebb beépíteni a programokba, mintha itt találkoznának velük először a gyerekek.

A PRINT *utasítás néhány tulajdonsága*

```
10 PRINT A
20 PRINT A$
30 PRINT A%
```

A PRINT után írt változó értéke megjelenik a képernyőn.

```
40 PRINT"A"
50 PRINT"A$"
60 PRINT"A%"
```

A PRINT után idézőjelbe írt változó alakja megjelenik a képernyőn.

```
130 PRINT
```

Ha a PRINT utasítás után nem írunk semmit, a gép üres sort húz.

```
140 PRINT A;B
150 PRINT A,B
```

A PRINT utasítás után írt változók közt a pontosvessző tömörít, a vessző zónáz.

Ha a kurzormozgató műveleteket a programba építjük, azok jele megjelenik a program listájában is:

```
70 PRINT"■"
```

Képernyőtörlés

```
80 PRINT"■"
```

Kurzor egy hellyel jobbra mozdul

```
90 PRINT"■"
```

Kurzor egy hellyel balra mozdul

```
100 PRINT"■"
```

Kurzor egy hellyel felfelé mozdul

```
110 PRINT"■"
```

Kurzor egy hellyel lefelé mozdul

```
120 PRINT"■"
```

Kurzor a bal felső sarokba ugrik képernyőtörlés nélkül

Ez utóbbi jelekre hívjuk fel a gyerekek figyelmét, mert idegen programban olvasva gyakran tanácstalanul szemlélik ezeket.

Sok gondot okoz a gyerekeknek a PRINT "K=";K programsor értelmezése, ezért fontos hangsúlyozni, hogy az idézőjelen belül szereplő karakter, még ha az történetesen egy változó azonosítója is, nem értékével, hanem alakjával jelenik meg. Tehát ha a képernyőn az adott változó értékét is meg akarjuk jeleníteni, akkor azt az idézőjelen kívül is el kell helyezni, méghozzá a megfelelő írásjellel elválasztva.

A képernyőn megjelenő feladatok összképét nemcsak a megfelelő kurzormozgató billentyűk segítségével lehet kialakítani. Célszerű bevezetni a TAB formulát.

A PRINT TAB(X);A utasítás hatására a gép az utolsó kiírt karaktertől számított X. helyre kiírja A értékét. Ha a sorban még nem volt karakter, a gép az X. oszlopban kezdi meg a kiírást. A TAB formulával szép táblázatok készíthetők, feltéve, hogy azonos hosszúságú karaktersorokat tartalmazó változókat íratunk ki. A probléma akkor jelentkezik, ha az azonos oszlopba kiírt értékek hossza nem egyezik meg. Ezt a hiányosságot a karakterfüzér (sztring) függvények használata kiküszöböli.

3.2 *A változók cseréje*

„A változók" című témánál már kitértünk a definiáló egyenlőség fogalmára. Beláttattuk, hogy az A=2 és a 2=A egyenlőség nem azonos, sőt, a második a számítástechnikában értelmetlen is.

Meg kell mutatnunk, és be kell láttatnunk azt is, hogy az A=B és a B=A egyenlőségek értelme sem egyezik meg.

Ezt egy egyszerű példa alapján is megtehetjük:

Legyen A=3 és B=4. Tegyük A-t egyenlővé B-vel (A=B)! Ekkor A=4 és B=4. Ellenben ha B-t tesszük egyenlővé A-val, (B=A), akkor B=3 és A=3.

Tehát mindkét esetben csak a bal oldalon álló változó értéke módosult. A memóriában levő változók értéke tehát módosítható, a változók értékei egymás között felcserélhetők.

Ha két változó értékét fel akarjuk cserélni, azt csak egy ún. munkaváltozó bevezetésével lehet megtenni. Ha nem használnánk munkaváltozót, az egyik változó értéke elveszne, mint az az előbbi példából is látszik. A megoldás könnyen belátható:

X=A A munkaváltozó felveszi A értékét

A=B Az A változó felveszi B értékét

B=X A B változó felveszi a munkaváltozó értékét, ami megegyezik az A változó kiindulási értékével.

A folyamat eredményeként a két változó értéke felcserélődött a memóriában. A csere lehetősége minden változótípusra fennáll.

A tanulásban a változók cseréjére vonatkozó feladatok megoldásán túl, nagy szerepe van a kész programok elemzésének is. A programok elemzését először célszerű papíron elvégezni. A megoldások helyessége a programok futtatásával ellenőrizhető.

3.3 *Az INPUT utasítás*

A gyerekek eddig csak olyan programokat készítettek, amelyek csak egy előre megadott adategyüttessel dolgoztak. Meglehetősen hamar felmerül a probléma, hogyan lehet tetszőleges, akár minden futtatáskor más-más adatokkal dolgozni.

Ennek megoldására lehetővé kell tenni a kívülről történő adatbevitelt. Ezt a célt szolgálja az INPUT utasítás bevezetése. Az INPUT utasítás néhány tulajdonsága:

```
10 INPUT A
20 INPUT A$ Az utasítás minden változótípus bevitelére alkalmas.
30 INPUT A
40 INPUT "SZÖVEG";A Az idézőjelben levő szöveg megjelenik a képernyőn,
                    és a változó értékét közvetlenül ez után kéri be a gép
```

Az idézőjelen belül kurzormozgató utasítások is elhelyezhetők. Ugyanazzal az INPUT utasítással több érték is bekérhető. Ilyenkor a változókat vesszővel választjuk el egymástól.

```
50 INPUT A,B,C
```

Tapasztalat szerint a gyerekek kezdetben még keverik az INPUT és a PRINT utasításokat. Ezért mindig hangsúlyoznunk kell a két utasítás közti különbséget. Pl. úgy is megtehetjük, hogy kezdetben nem használjuk a 40. utasítássorban megadott formát, hanem helyette a:

```
50 PRINT "SZÖVEG";
55 INPUT A
```

használatával még jobban hangsúlyozzuk, fizikailag is elkülönítjük egymástól a két utasítás feladatát. Természetesen kellő gyakorlás után az INPUT "SZÖVEG";A is jól használható.

Előnye: egyszerűbb használni, egy utasítás megtakarítható.

Hátránya: túl hosszú szöveg esetén a gép hibát jelez.

Az INPUT gyakorlása jó alkalom a PRINT gyakorlására is!

A feladatokat célszerű már most úgy megíratni a gyerekekkel, hogy azok kívülálló alkalmazó számára is érthetők legyenek.

A feladatgyűjteményben ennél a témánál szereplő feladatok jó része a későbbiek során újra előkerül, azokat bővítjük, javítjuk. Ezért érdemes az alapfeladatokat megoldani. Az ilyen bővítgetések, változtatások rászoktathatják a gyerekeket arra, hogy a már elkészült programjaikkal a későbbiekben is foglalkozzanak, ne elégedjenek meg az olcsó sikerekkel.

3.4 *Néhány gyakori függvény*

Az elágazás nélküli programok is igen változatossá tehetők néhány függvény bevezetésével.

A gép „BASIC"-je lehetővé teszi, hogy ezekre a megfelelő utasítással lehessen hivatkozni.

Az egészrész függvény INT(X)

A szám egészrésze a számnál nem nagyobb egészek közül a legnagyobb. Az X szám egészrészét az INT(X) segítségével kaphatjuk meg.

A=INT(X)

Pl. 1. X=0, A=0 2. X=1, A=1 3. X=−3.5, A=−4

Az egészrész függvény jól használható oszthatósági problémák vizsgálatánál, vegyes számok egészrészének számításánál, kerekítéseknél.

A négyzetgyök függvény SQR(X)

A négyzetgyök fogalmával a gyerekek már matematikai tanulmányaik alatt is találkoztak. Hetedik osztályban megtanulják Pitagorasz tételét is. Ezen előismeretek alapján a négyzetgyök függvényt sokrétűen lehet felhasználni.

Egy X szám, ahol X nem negatív, négyzetgyökén azt a számot értjük, amelynek négyzete X. Ezt az SQR(X) függvény segítségével kapjuk meg.

A négyzetgyök függvényt használhatjuk például táblázatok készítésénél, geometriai számításoknál, egyenletek megoldásánál.

A véletlenszámot előállító (generáló) függvény RND(X)

Sokszor és egyszerűen felhasználható függvény. Véletlenszerű jelenségek szimulálásánál, játékprogramoknál, gyakorlóprogramoknál egyaránt használható.

A véletlenszámok az RND(X) segítségével állíthatók elő. Az A=RND(1) értéke mindig a 0–1 balról zárt, jobbról nyitott intervallumon belül van. Ha a feladat szempontjából más értékekre van szükség, azok egyszerűen generálhatók.

Az A=(RND(1)*B)+C értéke a C–(C+B) balról zárt, jobbról nyitott intervallumba esik. Ha B nagyobb, mint 1, a kapott értékek között egész számok is szerepelhetnek.

Ha egész típusú véletlenszámokat kell előállítani, akkor a generált véletlenszámok egészrészét kell venni.

Az A=INT(RND(1)*(B+1))+C értéke a C–(C+B) zárt intervallumon belülre esik, és minden A érték egész szám.

3.5 Az *IF THEN* és a *GOTO* kapcsolata

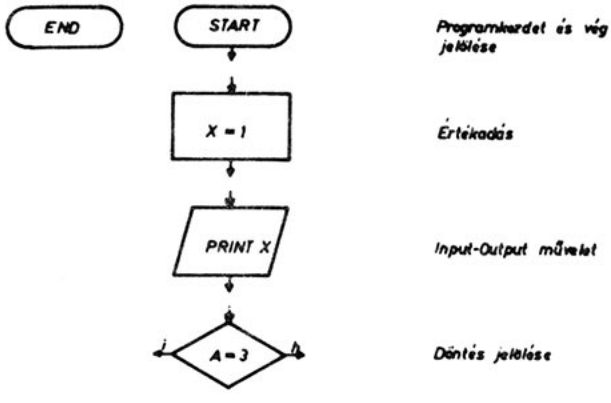
Az elágazásos algoritmusok vizsgálata és azok lényegének megértése nagyon fontos része a számítástechnika tanításának.

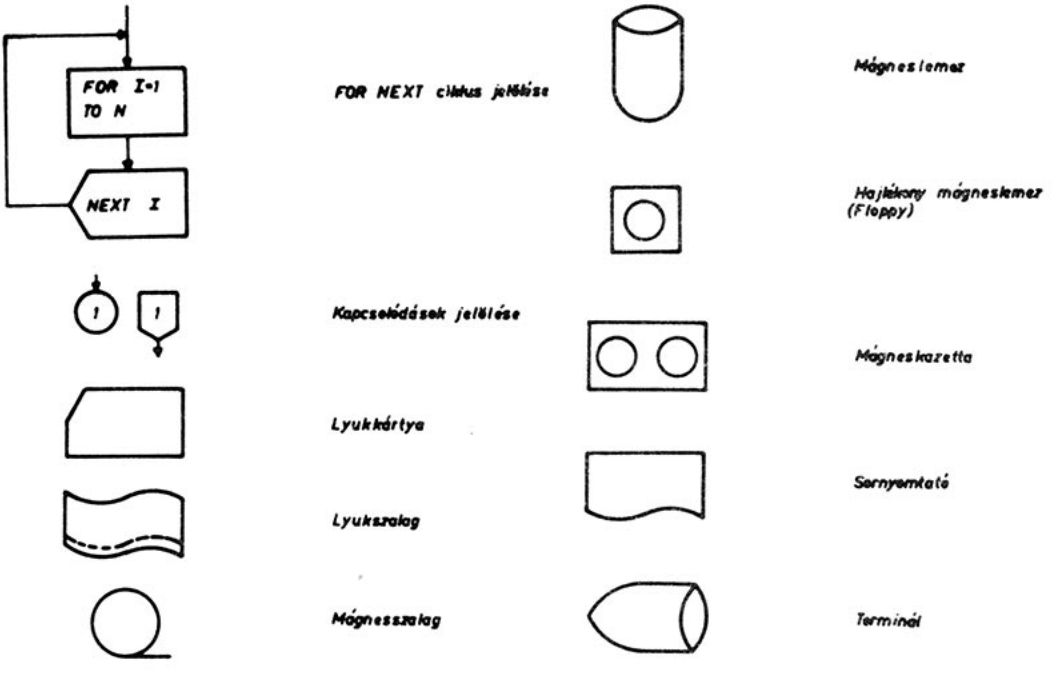
A gyerekek nemcsak tanulmányaik során, de a mindennapi életben is gyakran találkoznak olyan problémákkal, amelyek megoldása során az állítások igazságtartalmának megállapítása a cél.

A feladatok megoldása során rendszerint az egyenlő; kisebb; nagyobb; nem egyenlő; nem kisebb; nem nagyobb relációkat használjuk.

Mivel az elágazásos programok ilyen logikai kapcsolatokra vezethetők vissza, ne sajnáljuk a probléma tisztázására és gyakorlására fordított időt! Az elágazásos algoritmuson alapuló programok megírása még alaposabb tervezést igényel, mint az eddigiek.

A megoldások elkészítéséhez nagy segítséget jelent a folyamatábra készítése. *Előforduló fontosabb jelölések:*





Ha a programot a folyamatábra segítségével tervezzük meg, akkor a folyamatábrának nem kell túl részletezőnek lennie, elegendő a program fő lépéseit tartalmaznia.

Az elágazás jelölésére a programban az IF THEN kapcsolat ad lehetőséget.

Pl. 1ØØ IF A=B THEN GOTO 8Ø

Az összehasonlításnak két eredménye lehet. Vagy igaz az állítás, vagy hamis. Mivel a Commodore 16 BASIC-je nem ismeri az ELSE* ágat, ezért ha az állítás igaz, akkor a THEN után álló utasítás hajtódik végre, ellenkező esetben a gép a következő utasítássorra lép. A THEN után általában a GOTO utasítást szokták használni, de bármely más utasítás is írható ide, sőt több utasítást is írhatunk, egymástól kettősponttal elválasztva. Nézzünk egy egyszerű példát az IF THEN használatára:

Készítsünk programot, amely meghatározza egy szám abszolút értékét!
A feladat egy lehetséges megoldása:

```
10 PRINT"□"
20 PRINT"ABSZOLUTERTEK SZAMOLO"
30 PRINT
40 INPUT"A SZAM=";X
```

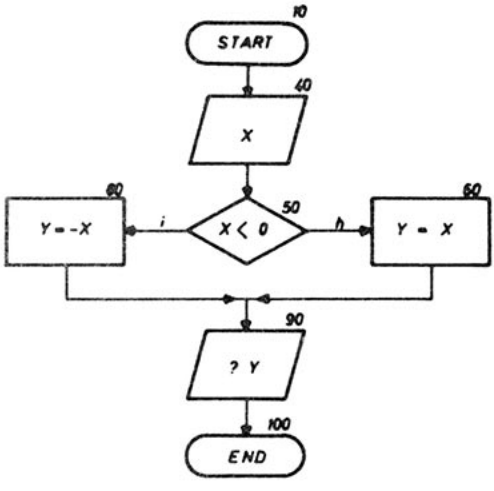
* Az újabb típusú Commodore 16-os gépek megengedik az ELSE ág használatát. A szükséges ismereteket a C–16 Felhasználói kézikönyvben megtalálja. *(A szerző megjegyzése.)*

```
50 IF X<0 THEN GOTO 80
60 Y=X
70 GOTO 90
80 Y=-X
90 PRINT X;"ABSZOLUTERTEKE ";Y
100 END
110 PRINT"[illegible]"
120 PRINT"[illegible]"
```

Az ELSE ág nélküli szervezés miatt az elágazások után a programsorok számozása a hamis (nem) ág felé folytatódik.

Vigyázni kell arra is, hogy az egyik ágban megoldott feladat után a gép ne folyjon bele a másik ágon történő feladatmegoldásba. Ennek elkerülésére feltétel nélküli ugrásokat kell a programba építeni. Ilyen pl. a 90. utasítássor is.

A program készítésénél jó szolgálatot tesz a beszámozott folyamatábra is, melyen jól látható, hogy az ágak számozása mindig a hamis ág felé halad.



Az elágazásos algoritmus ismerete lehetőséget nyújt összehasonlítások elvégzésére, így pl. nagyságrendi viszonyok megállapítására, rendezésekre is. Vigyázzunk azonban arra, hogy a gyerekeknek ezen a szinten már három elem összehasonlítása és rendezése igen komoly problémát jelent. Több elem rendezésével csak a későbbiekben foglalkozhatunk.

A gép nemcsak számok sorrendbe állítását teszi lehetővé, hanem szavak abcsorrendbe rendezését is. Lehetőség van ugyanis szöveges változók közötti logikai kapcsolatok vizsgálatára, pl. A$=B$, A$<B$. Ez azért lehetséges, mert a gép a karaktert mint kettes számrendszerbeli jelsorozatot tárolja, és az összehasonlításokat is ennek alapján végzi.

Az elágazásos algoritmus lehetőséget nyújt az adathibák kivédésére. A gyerekekkel mindig bővíttessük úgy a programokat, hogy azok ne fogadjanak el hibás adatokat.

Nagyon fontos, hogy a gyerekek maguk keressék ki az adott program hibalehetőségeit, mert így megszokják, hogy a feladatokat sok szempont szerint meg kell vizsgálniuk ahhoz, hogy valóban jól működő programot készíthessenek.

Egy program akár több elágazást is tartalmazhat, így lehetőség van egy adott programon belül, több különálló feladat megoldására is. Ilyen esetben célszerű az úgynevezett menü módszer alkalmazása. A módszer lényege, hogy a képernyőn sorszámmal ellátva megjelenik az összes lehetséges használati mód. A használó a megfelelő sorszám begépelésével kérheti az általa kiválasztott módot. A begépelt szám alapján a gép a megfelelő utasítássoron folytatja a munkát.

Egy példa a menü módszer alkalmazására:

```
140 PRINT"■"
150 PRINT"■■■■■MELYIK ADAT ALAPJAN DOLGOZUNK?■■"
160 PRINT"1. NEV"
170 PRINT"2. LAKCIM"
180 PRINT"3. TELEFONSZAM"
190 INPUT X
200 IF X=1 THEN GOTO 250
210 IF X=2 THEN GOTO 330
220 IF X=3 THEN GOTO 410
230 PRINT"ILYEN LEHETOSEG NINCS"
240 GOTO 140
```

Az elágazásos algoritmus jó lehetőséget nyújt érdekes játékok írására, pl. számkitaláló, szókitaláló stb. gyakorlóprogramok.

A feltétel nélküli ugrás bemutatására is időt kell szánnunk, mert a gyerekek egy része hajlamos arra, hogy a GOTO utasítást csak az IF–THEN részeként használja.

Az elágazásos algoritmus begyakorlására, ilyen programok írására időt kell biztosítani, mert ez a számítástechnika-oktatás egyik sarkalatos pontja! Míg a gyerekek előtt nem egészen érthető a téma, addig nem szabad tovább haladni, mert a sietség a későbbiekben megbosszulja magát!

3.6 A ciklus (IF . . . THEN alkalmazásával)

Eddig a gép által végzett műveletsor minden lépését külön-külön le kellett írni, még akkor is, ha a program futása közben ugyanaz a programrészlet egymás után többször is ismétlődött. Ettől a munkától megszabadulhatunk a ciklus bevezetésével.

A ciklus olyan programrészlet, amely a program futása közben egymás után többször végrehajtódik.

A ciklus lehet feltétel nélküli:

```
1Ø PRINT "ALMA"
2Ø GOTO 10
```

Ez a kis program az ALMA szót végtelen sokszor kiírja a képernyőre, ezért végtelen ciklusnak is nevezzük.

Gyakran felmerül a kérdés, hogy lehetne egy szépen megszerkesztett képernyőt hosszabb ideig megőrizni a READY kiírása nélkül. Ennek a problémának a megoldására is használható a végtelen ciklus.

Mivel a READY akkor jelenik meg a képernyőn, ha a program futása befejeződött, vagyis megtalálta az END vagy STOP utasítást, ill. nem talál újabb utasítássort, ezeket a lehetőségeket kell kivédeni.

Az egyértelmű, hogy ilyen esetben a STOP, ill. az END használata felesleges. Ha az utolsó programsorba egy feltétel nélküli ugró utasítást teszünk, amely a saját sorszámára ugrat, a program végén végtelen ciklus alakul ki, így nem jelenik meg a READY.

Pl. 10Ø GOTO 10Ø

Vigyázat! Ilyenkor a program csak a RUN/STOP gomb lenyomásával állítható meg.

A ciklus lehet feltételes is:

Ilyenkor a ciklusmagban levő utasítás újra és újra végigfut, amíg a ciklusváltozó értéke eleget nem tesz egy adott feltételnek. Pl.:

```
1Ø  I=0
2Ø  I=I+1
3Ø  PRINT I
4Ø  IF I<1ØØ  THEN GOTO 2Ø
5Ø  END
```

Példánkban I a ciklusváltozó

a 3Ø-as utasítás a ciklusmag

a 4Ø-es utasítás a ciklusvezérlő

Az I értékének kiírása 100-szor történik meg, majd a program megáll.

A ciklusváltozókat általában az I, J, K, L betűkkel szokták jelölni. Ez természetesen nem kötelező, de célszerű megszokni ezt a jelölésmódot, mert így nem fordul elő, hogy a ciklusváltozót mint azonosítót másra is használjuk.

A ciklusváltozó értékének változtatását mindig a feladat határozza meg. A ciklusváltozó értékét nemcsak növelni lehet, hanem csökkenteni is, sőt, a változtatást tört számokkal is végezhetjük.

Nagyon fontos a ciklusvezérlő feltételének jó meghatározása. A feltételben nem célszerű az egyenlőségre kérdezni, mert nagyobb a hibalehetőség, mintha más relációt vizsgálnánk.

```
1Ø  I=Ø
2Ø  I=I+2
3Ø  PRINT I
4Ø  IF I=99 THEN END
5Ø  GOTO 2Ø
```

Láthatjuk, hogy a kezdőérték és a ciklusváltozó értékének változtatása miatt az I értéke mindig páros, így az I=99 feltétel soha nem teljesül, végtelen ciklus jön létre. A ciklus lehetőséget ad arra, hogy ugyanaz a program megállás nélkül többször futtatható legyen.

A program végtelen sokszor fut:

Ha a program utolsó sorába az END helyett egy olyan feltétel nélküli ugró-utasítást teszünk, amely visszaviszi a programot az elejére, a program végtelen sokszor fut. Ekkor a program futtatása csak a RUN/STOP gomb lenyomásával fejezhető be.

A program előre meghatározott számú futást hajt végre:

Ilyen esetben a program elején be kell kérni, hányszor kívánja a használó futtatni a programot. A feladatot megoldó program köré egy olyan ciklust kell szervezni, amely annyiszor hajtódik végre, amennyi a begépelt érték.

```
10  INPUT "HÁNY FUTÁS LESZ";X
20  I=0
30  I=I+1
40        }
50        }
.         }          Ez az alapprogram helye
.         }
.         }
100  IF I<X THEN GOTO 30
```

A program tetszőleges, előre nem meghatározott számú futást hajt végre:

1. Pl.:

```
10  INPUT "AZ ADAT";X
20  IF X=-9999 THEN END
30
40
.
.
.
100  GOTO 10
```

A programozó ilyen megoldás esetén megvizsgálja a feladat szempontjából szóba jöhető adatokat, majd választ egy olyan értéket, amely a feladat megoldása során elképzelhetetlen.

Pl. ha az adatok érdemjegyek voltak, bármely szám, amely kisebb egynél vagy nagyobb ötnél, a feladat szempontjából már nem használható. Ezek közül célszerű nagyon extrém esetet választani, mert így elkerülhető, hogy esetleges gépelési hiba miatt leálljon a program. Ez a választott érték mintegy kapcsolóként működik a továbbiakban, mert adatként begépelve a program leállítható.

Ezt a lehetőséget leginkább a rendszert jól ismerő programozó tudja kihasználni. A program leírásában mindig emeljük ki a kapcsoló értékét!

2. Pl.:

```
10  Az alapprogram
.
.
.
100  INPUT "DOLGOZUNK MÉG (I-N)";V$
110  IF V$="I" THEN GOTO 10
120  END
```

A program csak az I gomb lenyomásával fut újra. A 110-es utasítás kérdésfeltevése miatt tehát a program leállításához bármely más billentyű használata megfelelő. Így történhet meg a figyelmetlen használóval, hogy a kérdésre begépelt IGEN hatására a program futása befejeződik, mert a V$="I" és a C$="IGEN" változók értéke nem egyezik meg. Az ilyen típusú szervezés olyan esetekben célszerű, ha a programot nem kell túl sokszor újra futtatni, mert a válaszok begépelése nagyon sok futás esetén nehézkessé teszi a felhasználást.

Az ilyen szervezés előnye, hogy a kérdésfeltevés miatt mindenki számára érthető, a programot nem ismerő használó is könnyen tudja futtatni.

Egy programon belül több ciklus is szervezhető, sőt ezek a ciklusok szükség esetén egymáson belül is megszervezhetők. Ha két vagy több ciklus egymáson belül fut, akkor egymásba skatulyázott ciklusokról beszélünk. Pl.:

```
1Ø  I=0
2Ø  I=I+1
3Ø  J=0
4Ø  J=J+1
5Ø  PRINT "*";
6Ø  IF J<4Ø THEN GOTO 4Ø
7Ø  IF I<25 THEN GOTO 2Ø
8Ø  END
```

A program teleírja a képernyőt csillagokkal. A külső ciklus a 2Ø és 7Ø utasítássor között, a belső pedig a 4Ø és 6Ø utasítássor között működik.

A belső ciklus feladata, hogy megoldja a 40 db csillag egymás mellé írását, amivel teleír egy sort.

A külső ciklus gondoskodik arról, hogy ez a művelet 25-ször megismétlődjék, vagyis az egész képernyő megteljen csillagokkal. A külső ciklus tehát 25-ször, a belső pedig 40*25=1000-szer fut.

Vigyázni kell arra, hogy a külső ciklusnak teljes egészében magába kell foglalnia a belső ciklust. A ciklus lehetőséget nyújt érdekes feladatok megoldására: modellezésekre, játékok írására.

Az egymásba skatulyázott ciklus bemutatására csak akkor térjünk át, ha a gyerekek már biztonsággal kezelik a ciklusokat. Nagyon fontos begyakorolni a ciklusváltozók módosításának lehetséges eseteit és ezek hatását. Amikor a feladat megengedi, kérjük a program többszöri futtathatóságának megoldását. A gyerekek maguk válasszák ki a feladat szempontjából legmegfelelőbb eljárást!

3.7 *A tömb*

Ha a programban sok adat, ill. változó használatára volt szükség, akkor a programok hosszúak, nehézkesen szervezhetők.

A problémát a tömbök használatának bevezetésével oldhatjuk meg. A tömbökbe elhelyezett adatokat a gép meghatározott sorrendben tárolja, így nemcsak az elem azonosítója, hanem a sorszáma is fontos.

A tömbök lehetnek egy dimenziósak, két dimenziósak, három dimenziósak . . ., n dimenziósak. Általában az első három eset a leggyakrabban használt.

Egy dimenziós tömb (vektor):
Az ilyen tömböt úgy modellezhetnénk, mintha a gép a tömb elemeit egy egyenes mentén helyezné el. A tömbök első eleme mindig a 0. indexű. Így egy elem megadásához elegendő megadni az azonosítóját és a sorszámát (indexét). Pl. az A nevű valós típusú elemeket tartalmazó tömb elemei:

A(0), A(1), A(2), A(3), . . . A(n) n elemű a tömb.

Az X$ nevű szöveges változókat tartalmazó tömb elemei:

X$(0), X$(1), X$(2), . . . X$(k) k elemű a tömb.

A tömbindexet i, j, k betűkkel szokás jelölni, mert a tömb elemeinek kezelése általában cikluson belül történik, és a ciklusváltozó értéke felhasználható a tömbelemek kikeresésére.

Kétdimenziós tömb (mátrix):
A gép az elemeket úgy tárolja, mintha azok egy derékszögű koordinátarendszer első síknegyedében helyezkednének el. Így egy elem megadásához az oszlop és

a sor indexét is meg kell adni. Pl. Az A nevű valós típusú elemeket tartalmazó tömb elemei:

A(0,0), A(0,1), A(0,2), ... A(1,0), A(1,1), ... A(i,j) i*j elemű tömb

Háromdimenziós tömb (mátrix):
Az elemek tárolása úgy képzelhető el, mintha azok egy térbeli derékszögű koordinátarendszerben helyezkednének el. Az elemek megoldásához itt már három index szükséges.

n dimenziós tömb (mátrix):
Az n dimenziós mátrix értékelésére már nincs lehetőség. Használatuk is meglehetősen nehézkes és bonyolult, így a gyakorlatban inkább az első három típus terjedt el.

A gép a tömb elemeit egymás után tárolja, ezért szüksége van megfelelő helyre. A program elején meg kell adnunk, hogy előreláthatólag milyen és hány elemű tömböt kívánunk használni. Ezt a DIM utasítással tehetjük meg.

Pl. DIM A(100), DIM X$ (4,91), ...

Az utasítás hatására a gép lefoglal a memóriában annyi helyet, amennyit így megadtunk. Ha kevesebb adatot tárolunk a tömbben, mint a lefoglalt helyek száma, akkor ezek a helyek a gép számára már nem használhatóak. Előfordulhat, hogy a túldimenzionálás miatt annyira leszűkül a használható memóriaterület, hogy a program futtatása már nem lehetséges, viszont sok felhasználatlan, de lefoglalt hely marad üresen. Ezért célszerű a dimenzionálással lefoglalt helyek számát pontosan meghatározni a programozás előtt.

Vigyázni kell arra, hogy adott DIM utasításon a program csak egyszer haladhat át. Ez főleg akkor jelent problémát, ha a programot úgy szerveztük, hogy az tetszés szerinti futást hajtson végre. Ügyelni kell ilyenkor arra, hogy az újrafutáskor a program kerülje ki a DIM utasításokat tartalmazó sort.

Egy tömbben levő adatot tetszőlegesen sokszor ki lehet olvasni, a tömbelem tartalmát tetszés szerint lehet változtatni.

A tömbelemeket sorban ki lehet olvasni, akár visszafelé is. A tömb elemeinek kezelésére gyakran használnak ciklusokat, akár a tömb feltöltéséről, akár kiolvasásáról vagy módosításáról van szó. Ez a lehetőség a programozó számára egyszerű, kényelmes munkát biztosít.

A tömbök alkalmazásakor gyakran használják azt a módszert, hogy a különböző összetartozó adatokból több tömböt hoznak létre. Pl.:

A$ tömb a tanulók nevét tartalmazza.

B tömb a tanulók tanulmányi átlagát tartalmazza.

A tömbök feltöltésekor ügyelni kell arra, hogy az összetartozó adatok azonos indexszel szerepeljenek. Így az adatok kezelésénél az azonos indexű adatok összepárosíthatók, ill. az egyik adat ismeretében a másik könnyen kikereshető lesz. Ha azonos típusú adatokat kell tárolnunk, létrehozhatunk többdimenziós tömböket is.

Pl. A kétdimenziós valós típusú adatokat tartalmazó tömb tartalmazza 34 tanuló névsorban elfoglalt helyét és tanulmányi eredményét tárgyanként.

Így egy tömb felhasználásával megállapítható bármelyik tanuló összes jegye:

```
.
.
.
100  J=0
110  J=J+1
120  PRINT A(5,J)
130  IF J<13 THEN GOTO 110
```

A tömb felhasználásával megállapítható egy adott tárgyból szerzett összes jegy:

```
.
.
.
100  I=0
110  I=I+1
120  PRINT A(I,13)
130  IF I<34 THEN GOTO 110
```

A tömbök használatát jól be kell gyakoroltatni a gyerekekkel, mert e nélkül a későbbi komoly programozás nem képzelhető el. Ezért érdemes a feladatgyűjtemény erre vonatkozó feladatait következetesen végigvenni.

3.8 *FOR . . . TO . . . STEP és NEXT ciklusutasítások*

A BASIC nyelv igen egyszerűen és jól használható ciklusképző utasítással rendelkezik. A megfelelő feltételek megadásával könnyen szervezhetők olyan ciklusok, amelyek előre meghatározott számú futást hajtanak végre.

Hangsúlyozni kell, hogy nem minden IF . . . THEN-nel szervezett ciklus oldható meg korrektül a FOR . . . NEXT használatával. A FOR I=A TO B STEP C és NEXT I utasítássorok közé zárt ciklusmag a feltételek által meghatározott számú futást hajt végre.

Az A értéke a ciklusváltozó kezdőértéke, amelytől C lépésenként kell eljutni a B végértékhez.

A NEXT I utasításhoz érve az I értékét növeli C-vel és utána megvizsgálja, hogy I nagyobb-e mint B értéke. Így az sem idéz elő végtelen ciklust, ha A-tól B-ig C lépésközökkel nem lehet pontosan eljutni. A ciklus befejezésekor tehát I értéke mindig nagyobb mint B értéke. Ezt azért jó tudni, mert az I értékét kontroll nélkül használva esetleg rossz eredményt kaphatunk. Ha a programban C értéke egy, akkor a STEP utasítás elhagyható.

Vigyázni kell arra is, hogy a ciklusváltozó azonosítóját a cikluson belül más változóként ne használjuk, mert az a változó értékének felírásához vezet.

Figyelni kell arra is, hogy a ciklusból lehetőleg ne ugorjunk ki, ill. a ciklusba a vezérlő megkerülésével ne ugorjunk be, mert ez gyakran hibához vezet.

A FOR . . . NEXT típusú ciklusokat is egymásba lehet skatulyázni. Pl.:

```
10  FOR I=1 TO 25
20  FOR J=1 TO 40
30  PRINT "*";
40  NEXT J
50  NEXT I
60  END
```

Ez a program az IF . . . THEN utasítások tárgyalásánál már előkerült. Láthatjuk, hogy ugyanaz a feladat mennyivel egyszerűbben megoldható FOR . . . NEXT segítségével.

Ügyelni kell arra, hogy a külső ciklusnak teljes egészében tartalmaznia kell a belső ciklust, valamint a NEXT utasítások sorrendjére.

A tömbök és a ciklus ismerete lehetőséget ad a komolyabb adatkezelési feladatokra. Bár a nagyon jól dolgozó gyerekek meg tudnak birkózni a feladattal, mégis inkább csak bemutatni érdemes a következő két módszert:

A buborék, ill. a szomszédos elemek cseréjének módszere.

A módszerek elnevezése jól jellemzi a rendezés végrehajtásának módját. Mindkét lehetőség két rendezendő elemet feltételez!

A módszerek részletes leírását a feladatgyűjtemény tartalmazza, mintapéldákkal együtt. A rendező programok általában mint egyéb programok programrészletei is használhatók.

3.9 READ, DATA, RESTORE

Bár a tömbök segítségével sok adat tárolható a memóriában, a program leállításával, ill. a gép kikapcsolásával azok elvesznek, a program újra indításakor újra kell őket rögzíteni.

Ennek a problémának a megoldására vezették be a READ, DATA, RESTORE utasítások használatát. Pl.:

```
1Ø  FOR I=1 TO 3
2Ø  READ A$
3Ø  PRINT A$
4Ø  NEXT I
5Ø  DATA, A, B, C
```

A READ utasítás hatására a gép megkeresi a DATA-val jelölt programsort és annyiadik adatot olvassa ki, ahányadik READ utasítással találkozott a program futása közben.

A DATA utasítás a programban bárhol elhelyezhető, általában vagy a program elején, vagy még gyakrabban a program végén található. Így ugyanis az adathalmaz nem zavarja a program szerkezeti áttekinthetőségét.

Ha ugyanazon adatok kiolvasására többször is szükség van, úgy a RESTORE utasítással az olvasás újra az első adattól indul. Pl.:

```
1Ø  FOR I=1 TO 3
2Ø  READ A$
3Ø  PRINT A$
4Ø  NEXT I
5Ø  RESTORE
6Ø  GOTO 1Ø
7Ø  DATA A, B, C
```

Ügyelni kell arra, hogy a program számára mindig megfelelő számú adatot biztosítsunk.

A READ–DATA használata természetesen nem jelenti azt, hogy megoldódott a nagy mennyiségű adat tárolásának problémája. A DATA-ban olyan adatokat érdemes csak tárolni, amelyek az adott program futtatásánál mindig előfordulnak, amelyekre a program futtatásakor rendszeresen szükség van.

Ilyen pl. a feladatgyűjtemény 139. feladata, ahol minden tanulónál ki kell írni a tantárgyak nevét. Mivel a gép a DATA-ból maga olvassa be a megfelelő

adatokat, a kézi adatbevitelt alkalmazó megoldásokhoz képest lényegesen gyorsabb. Természetesen a feladatokban változóként szereplő értékeket ez esetben is a billentyűzetről kell bevinni.

Adatként a DATA-ban szerepelhet betű, szám, ill. betű és szám vegyesen. Ügyelni kell arra, hogy a DATA-ból való kiolvasáskor minden változó a megfelelő típusú azonosítóba kerüljön. Pl.:

```
1Ø  FOR I=1 TO 5
2Ø  READ A, A$
3Ø  PRINT A;A$
4Ø  NEXT I
5Ø  DATA 1,A,2,B,3,C,4,D,5,E
```

Egy READ utasítással több adat is beolvasható, de ilyenkor az azonosítókat vesszővel kell elválasztani egymástól.

3.10 A GET A$

Eddig a billentyűzetről történő adatbevitelt az INPUT utasítás segítségével oldottuk meg. A kérdőjel után begépelt adatot a gép csak a RETURN billenyű lenyomása után értelmezi.

Előfordul, hogy egyszerre csak egy karaktert szabad bevinni a gépbe, ill. az is fontos, hogy a megfelelő billentyűt melyik pillanatban nyomtuk meg. Pl. Kapcsolók alkalmazása, játékprogramok . . .

Az ilyen jellegű adatbevitelre vezették be a GET utasítást.

A GET utasításhoz érve a gép megvizsgálja, hogy lenyomtunk-e valamilyen billentyűt. Ha az adott pillanatban talál ilyet, akkor az adott karaktert a GET után írt füzérváltozóban helyezi el. Ha nem talál lenyomott billentyűt, a program futása a következő utasításon azonnal folytatódik. Pl.:

```
1Ø  GET A$
2Ø  PRINT A$
3Ø  END
```

A GET utasítás használható arra is, hogy vele a program futását szabályozzuk.

A GET mint kapcsoló használata:

Pl. 1.

```
1Ø  GET A$
2Ø  IF A$="" THEN GOTO 1Ø
3Ø
4Ø  }
 .  }
 .  }          Itt folytatódik az alapfeladat
 .  }
```

A program futása tetszőleges billentyű lenyomására indul.

Pl. 2.

```
1Ø  GET A$
2Ø  IF A$<>"*" THEN GOTO 10
3Ø
4Ø  Az alapfeladat itt folytatódik
 .
 .
 .
```

A program egy adott billentyű lenyomásával indítható. A GET utasítás lehetőséget ad arra is, hogy adott szöveget folyamatosan begépelve, mégis karakterenként helyezzük el a memóriában.

```
10 DIM A$(100)
20 I=0
30 I=I+1
40 GET X$
50 IF X$="*" THEN END
60 A$(I)=X$
70 IF I<100 THEN GOTO 30
```

A program maximum 100 karakternyi adat begépelését teszi lehetővé. Ha a begépelt szöveg rövidebb 100 karakternél, akkor a program az utolsó karakter után begépelt csillag jellel leállítható.

A GET utasítást használva, a gyerekek már önállóan is ügyes játékprogramokat készíthetnek. A tapasztalat alapján szívesen és jól alkalmazzák ezt az utasítást.

3.11 Időmérés számítógéppel (TI és TI$)

Játékok, ill. gyakorlóprogramok írásakor gyakran jó lenne mérni a megoldásra fordított időt.

Erre a célra használhatók a TI és TI$ változók.

A gép bekapcsolásának pillanatában TI=0 és TI$="000000". A TI változó értéke alaphelyzetben a gép működésének idejét hatvanad másodpercben adja meg.

A TI$ változó ugyanezt az időt hat karakterben ábrázolja úgy, hogy az első két karakter az órákat, a harmadik és a negyedik a perceket, az ötödik és a hatodik pedig a másodperceket mutatja.

A TI és a TI$ értékét bármikor kiírathatjuk.

A feladatokon belül azonban nem a gépidő meghatározása a cél, hanem egy adott feladat elvégzésére fordított idő mérése. Erre azért van lehetőség, mert a programban a TI és a TI$ értéke bármikor beállítható, így az óra tetszés szerint bármikor elindítható.

Ha a géppel valamilyen időtartamot akarunk mérni, akkor a TI$ értékét az indítás pillanatában nullázni kell. TI$="000000", a nullázás csak ilyen alakban végezhető el. Ekkor TI értéke is nullázódik. A nullázást követő pillanatban az óra már működik, ezért figyelni kell arra, hogy ezt a műveletet a megfelelő időben végezzük. Ellenkező esetben előfordulhat, hogy néhány programutasításra fordított időt is a megoldási időbe számítunk.

Mivel TI értéke a másodpercekben mért idő hatvanad része, ezért lehetőség van rövid időtartamok mérésére is.

Pl. A=TI/6 esetén A értéke az eltelt időt tizedmásodpercekben adja meg.

Ha az időt a képernyőn óra formájában folyamatosan ki akarjuk íratni, úgy ügyelni kell arra, hogy a TI$ aktuális értékét a képernyőnek mindig ugyanazon helyére írassuk ki, mert így kelti egy igazi óra benyomását.

Ha a programban megoldási időket, ill. reflexidőket mérünk, úgy a programot bővíthetjük azzal is, hogy az időt a kijelzésén kívül értékeljük is. Meg kell határozni, hogy az adott feladat szempontjából milyen időtartamok milyen teljesítménynek felelnek meg, és ennek alapján értékelhetjük a játékos teljesítményét.

A változók használatának főleg játék- és gyakorlóprogramok írásánál van jelentősége. Tapasztalat szerint a gyerekek szívesen alkalmazzák programjaikban, bár igazán jó megoldásokat főleg a haladóknál látni.

3.12 Képernyőkezelés, grafika

A Commodore 16-os gép talán az egyik legjobb, viszonylag egyszerű képernyőkezelést teszi lehetővé. Azért tekinthető a legjobbnak, mert a gép segédprogramok nélkül is magas szinten tesz eleget feladatának.

A számítógép alaphelyzetben a képernyőt 40 oszlopra és 25 sorra bontja. Rajzolni a billentyűzeten található kvázigrafikus jelekkel lehet.

Alapállapotban is színezhető a képernyő három része a COLOR utasítás segítségével. Ez a színezhetőség természetesen a grafikus üzemmódban is megmarad.

A háttér színezése:	COLOR 4, színkód, árnyalatkód
A papír színezése:	COLOR 0, színkód, árnyalatkód
A kurzor színezése:	COLOR 1, színkód, árnyalatkód

Az árnyalatkódok 0–7 közötti számmal adhatók meg. A feladatgyűjtemény példáiban azért a fekete színt használom rendszeresen, mert fekete-fehér televízión a többi szín nehezen olvasható, különíthető el. Ahol rendelkezésre áll színes televízió, ott a színezés különböző lehetőségét célszerű kipróbálni.

Színkódok:

1 fekete	9 narancs
2 fehér	10 barna
3 piros	11 sárgászöld
4 türkiz	12 rózsaszín
5 bíbor	13 kékeszöld
6 zöld	14 világoskék
7 kék	15 sötétkék
8 sárga	16 világoszöld

A finomgrafikára való áttérést a GRAPHIC utasítás segítségével oldhatjuk meg.

Lehetséges üzemmódok:

Szöveges:	GRAPHIC 0,1
Nagy felbontóképességű grafika:	GRAPHIC 1,1
Nagy felbontóképességű grafika+szöveg:	GRAPHIC 2,1
Többszínű grafika:	GRAPHIC 3,1
Többszínű grafika+szöveg:	GRAPHIC 4,1

Nagy felbontóképességű grafika+szöveg, ill. többszínű grafika+szöveg esetén csak 160 sorral dolgozik. Ilyenkor a képernyő alján – és csak itt – lehetőség van a normál üzemmódban való írásra. (Az ismert PRINT és INPUT utasításhoz tartozó karakterek másutt nem jeleníthetők meg.)

A szöveges üzemmód a normál üzemmódot jelenti. A hozzá tartozó utasítás segítségével lehet visszatérni a normál üzemmódba.

A grafikus üzemmódban a képernyőn tetszőleges színű pontot megjeleníthetünk, vagyis rajzolhatunk, ill. adott pontot kitörölhetünk.

A Commodore 16 rendelkezik olyan utasításokkal is, amelyek lehetővé teszik különböző alakzatok egyszerű megoldását.

A grafikus üzemmód néhány speciális utasítása:

DRAW

Az utasítás alkalmas pont, egyenes, megtört vonal, egyenesek által határolt sokszög rajzolására. Az utasításban szereplő színkód értéke lehet 1 és ekkor a gép rajzol, ill. lehet 0 és ekkor a gép töröl. A rajzolandó alakzat tényleges színét az utasítás előtt a COLOR utasítással be kell állítani.

Pont rajzolása:	DRAW színkód, oszlop, sor
Egyenes rajzolása:	DRAW színkód, oszlop, sor TO oszlop, sor (Itt meg kell adnunk az egyenes kezdő- és végpontjának koordinátáit.)
Egyenes a legutóbbi pontból:	DRAW színkód TO oszlop, sor.

CIRCLE

Az utasítás alkalmas kör, ellipszis, ív és ezekbe rajzolható síkidomok megrajzolására. A színkód itt is a rajzolás, ill. a törlés funkciót állítja be. Az alakzat tényleges színét szintén a COLOR utasítással állíthatjuk be.

Kör:	CIRCLE színkód, középpont oszlop, középpont sor, sugár
Ellipszis:	CIRCLE színkód, középpont oszlop, középpont sor, szélesség, magasság
Ív:	CIRCLE színkód, középpont oszlop, középpont sor, szélesség, magasság, kezdet, vég
Elforgatott ellipszis:	CIRCLE színkód, középpont oszlop, középpont sugár, szélesség, magasság, szög
Sokszög:	CIRCLE színkód, középpont oszlop, középpont sugár, szélesség, magasság, 360 szög

Az elforgatott ellipszisnél a szög az elforgatás szögét jelenti, míg a sokszögnél a két szomszédos csúcsba húzott középponti szög nagyságat jelenti.

A programozás során ügyelni kell arra, hogy a megfelelő helyeken a megfelelő számú vesszőt használjuk!

Az adatok megválasztásánál figyelni kell, hogy az ábra elférjen a képernyőn. Adott programon belül a képernyőre több alakzat is rajzolható.

BOX

Az utasítás négyzetek és egyéb derékszögű idomok rajzolására alkalmas.

Kontúr:	BOX színkód, oszlop1, sor1, oszlop2, sor2
Elforgatás:	BOX színkód, oszlop1, sor1, oszlop2, sor2, szög
Kitöltött alakzat:	BOX színkód, oszlop1, sor1, oszlop2, sor2, kitöltés
Kitöltött alakzat elforgatva:	BOX színkód, oszlop1, sor1, oszlop2, sor2, szög, kitöltés

A BOX utasítás használatakor a két átellenes csúcs koordinátáit kell megadni, ill. ha a síkidomot el akarjuk forgatni, akkor az elforgatás szögét is meg kell adnunk.

Ha nemcsak a síkidom körvonalára van szükségünk, hanem az egész körülhatárolt területre, azt át is színezhetjük úgy, hogy a kitöltés helyére 1-et írunk.

PAINT

Az utasítás összefüggő vonallal körülhatárolt területek színezésére alkalmas.

Színezés: PAINT színkód, oszlop, sor.

A színezéskor mindig egy belső pont koordinátáját kell megadni. A színkód itt is az írás, ill. a törlés üzemmódjának megadására szolgál. A tényleges szín beállítása a COLOR utasítással történik.

A PAINT utasítással megtehetjük azt is, hogy a képernyőt úgy színezzük be, hogy az összefüggő vonallal körülhatárolt terület maradjon meg az eredeti színben. Ezt úgy oldhatjuk meg, hogy a gépnek egy külső pont koordinátáját adjuk meg.

CHAR

A grafikus üzemmódban a képernyőre PRINT utasítással csak a kombinált üzemmód esetén írathatunk, akkor is csak a képernyő néhány alsó sorába.

A grafikus üzemmódban elkészített ábrát azonban gyakran el kell látni magyarázó szöveggel, jelekkel. Ezt a célt szolgálja a CHAR utasítás.

Kiíratás: CHAR színkód, oszlop, sor, "szöveg"

Ügyelni kell arra, hogy az így kiíratott karakter normál karakterméretben jelenik meg. A kiíratás kezdőoszlopának és sorának megadása a kis felbontású normál írásmódnak megfelelően történik. Ez azt jelenti, hogy a CHAR utasítást használva a képernyőt 40-szer 25-ös osztásúnak kell tekinteni.

A grafikus üzemmódban nem használható a normál üzemmódnál alkalmazott képernyőtörlési módszer sem. Itt az SCNCLR utasítás begépelése szükséges a képernyő törléséhez.

A grafikus üzemmód egy érdekessége, hogy szintaktikai hiba esetén visszaáll normál üzemmódba. Ezt ki is használhatjuk. Ha a programunk valamilyen oknál fogva úgy áll le, hogy a gép grafikus üzemmódban maradt, szándékos szintaktikai hiba, pl. egy grafikus jel RETURN-nal való begépelésével a gép azonnal visszatér normál üzemmódba.

Mivel a gép grafikája igen fejlett, ezzel érdemes a gyerekeket mélyebben megismertetni. Ez nem nehéz feladat, mert a gyerekek igen nagy érdeklődést tanúsítanak minden mozgó, változó ábra és annak megalkotása iránt. A grafikus üzemmód ügyes alkalmazása alapja lehet érdekes játék-, ill. oktatóprogramoknak.

Grafikus programok írása és futtatása, ízléses ábrák készíttetése a gyerekek esztétikai nevelésében is fontos szerepet kaphat.

A gyerekek a témával foglalkozva nem csak programozói, hanem bizonyos szempontból művészi alkotótevékenységet is folytathatnak. A téma iránt érdeklődő tanulók egyszerűbb mozdulattanulmányokat, rajzfilmrészleteket stb. is létrehozhatnak a géppel.

3.13 Zene

A legtöbb modern személyi számítógép képes különböző magasságú és hangszínű hangokat kiadni. Ezeknek a hangoknak a segítségével zeneprogramok is készíthetők.

A Commodore 16 két hanggenerátorral rendelkezik. Ezek külön-külön, de egyszerre is megszólaltathatók.

A hang generálásához két utasítást kell ismerni:

VOL X: Az utasítás után szereplő változó értéke 0–7 között változhat. Ha az X értéke 0, a hanggenerátor kikapcsol. 1–7 a hangerő beállítására szolgál.

SOUND A, B, C: Az A értéke a hanggenerátort, a B értéke a hang magasságát, a C értéke a megszólalás időtartamát határozza meg.

Ha A=1 a gép az első hanggenerátort szólaltatja meg.

Ha A=2 a gép a második hanggenerátort szólaltatja meg.

Ha A=3 a gép újra a második hanggenerátort szólaltatja meg, de most nem zenei hangot, hanem zörejt ad.

A zenei hangoknak kódok felelnek meg. Ezeket táblázatba foglalva közöljük. A B értéke innen választható.

A C értéke 0–65535 közötti szám lehet. Ez az érték adja meg a hanghatás hosszát. C=1 esetén ez az időtartam a másodperc ötvened része, tehát egy másodpercet úgy kapunk, ha C értékét 50-re állítjuk.

Vigyázzunk arra, hogy ha a program futása közben egy SOUND utasításhoz ér, akkor a hangot megszólaltatja, de nem várakozik ezen az utasításon addig, amíg a hang szól. Ezért, ha pl. a hanggenerátort és a képernyőt szinkronban kívánjuk működtetni, a programba megfelelő várakoztató ciklusokat kell építeni.

A géppel véletlenszerűen is létrehozhatunk hanghatásokat. Ekkor az A, B, C változó értékét a megfelelő határok között RND segítségével hozhatjuk létre. Ezek a zenei hatások jól használhatók olyan játékprogramoknál, ahol valamilyen, a gép által végrehajtott eredmény kiíratását szeretnénk késleltetni a szórakoztatóbb játék kedvéért. A gép által véletlenszerűen létrehozott hanghatások azt a benyomást keltik, mintha a gép még valóban feladatot oldana meg.

Az egyes billentyűket megfeleltetve az egyes zenei hangoknak, a gépet elektromos orgonaként használhatjuk. A hanghatások a billentyű lenyomásával jönnek létre. A karakterértékeket ilyenkor GET utasítással célszerű fogadni.

Előre megadott zenedarab is lejátszható a géppel. Ekkor a zenei hangok kódját és a hangok hosszúságát is célszerű DATA-ban tárolni. A programot elindítva a gép maga játssza el a programban szereplő zeneművet. A hanggenerátorral létrehozható zene jó kiegészítője lehet akár egy játék, akár egy grafikus programnak is.

Tapasztalat szerint a gyerekek szívesen készítenek olyan programokat, amelyek valamilyen hangeffektust is tartalmaznak. A teremben egyszerre működő számítógépek esetén ez néha meglehetősen próbára teszi a felnőttek türelmét. Érdekes, hogy a gyerekek annyira elmerülnek a munkában, hogy őket az így kialakuló zaj szinte nem is zavarja.

A zenei hangok kódtáblázata

A zenei hang	Kódja	Frekvenciája
A	7	110
H	118	123.5
C	169	130.8
D	262	146.8
E	345	164.7
F	383	174.5
G	453	195.9
A	516	220.2
H	571	246.9
C	596	261.4
D	643	293.6
E	685	330
F	704	349.6
G	739	392.5
A	770	440.4
H	798	494.9
C	810	522.7
D	834	588.7
E	854	658
F	864	699
G	881	782.2
A	879	880.7
H	911	989.9
C	917	1045
D	929	1177
E	939	1316
F	944	1398
G	953	1575

3.14 Karakterfüzér függvények

A karakterfüzéreket jól használhatjuk képernyőn, ill. sornyomtatón kialakított képek szerkesztésénél, adott szövegek elemzésénél, sztringváltozók vizsgálatánál, összehasonlításoknál.

Néhány gyakran használt karakterfüzér függvény:
CHR$ függvény

A számítógép az általa használt karaktereknek, ill. néhány utasításnak számokat feleltet meg. Ezeket a kódtáblázatból kereshetjük meg. A táblázatból kitűnik, hogy 32 és 90 között a mindennap használt betűk, számok és jelek kódjai találhatók. Ha a CHR$ függvény argumentumául választott függvény érték az adott karakter lesz, akkor a PRINT CHR$(X) utasítással a képernyőn megjeleníthető.

Ennek az utasításnak a segítségével már nem csak véletlenszerű számok, hanem ezek alapján véletlenszerű karakterek is megjeleníthetők.

Ha a CHR kódok segítségével szeretnénk az összes karaktert kilistáztatni, azt tapasztaljuk, hogy az utasításokat tartalmazó kódok némelyike utasításként értelmeződik és végre is hajtódik, és ez a listázás során problémát okoz.

Ezért a teljes karakterkészlet listázását a 32–127 és a 161–191 között végezhetjük teljes biztonsággal.

LEFT$ függvény

Gyakran szükség van adott szövegek elemzésére. A LEFT$ függvény lehetővé teszi, hogy egy füzér elejétől tetszés szerinti pozícióig vizsgáljuk az adott szót.

Az A$=LEFT$(X$,X) hatására az A$ szövegváltozóba kerül az X$ változó első X darab karaktere. Pl.:

```
X$="KATALIN": X=4
A$=LEFT$(X$,X)
PRINT A$
KATA
```

RIGHT$ függvény

A LEFT$ függvény párja. Míg a LEFT$ függvény a füzér bal oldalát vizsgálja, addig a RIGHT$ függvény ugyanezt a jobb oldallal teszi meg.

Az A$=RIGHT$(X$,X) hatására az A$ szövegváltozóba kerül, az X$ változó jobb oldalról vett X darab karaktere. Pl.:

```
X$="KATALIN": X=4
A$=RIGHT$(X$,X)
PRINT A$
ALIN
```

MID$ függvény

A MID függvény lehetővé teszi, hogy egy adott sztring tetszés szerinti darabját vizsgáljuk. Az A$=MID$(X$,A,B) hatására az A sztringváltozóba kerül az X$ változóban levő sztring A. és B. karaktere között levő szövegrész. Pl.:

```
X$="KATALIN": A=2; B=5
A$=MID$(X$, A,B)
PRINT A$
ATAL
```

Ezzel lehetőség nyílik arra, hogy egy adott szöveges változóból kigyűjtsük a kívánt karaktereket, és megvizsgáljuk, hogy egy sztring tartalmaz-e egy meghatározott karaktersort . . .

LEN függvény

Az eddig tárgyalt sztringfüggvények használatakor tudnunk kell, hogy a vizsgált sztring milyen hosszú. Mivel a sztring hosszának billentyűzetről való bekérése nem igazán jó módszer, bevezették a LEN függvény használatát.

Az A=LEN(X$) hatására A értéke annyi lesz, ahány karaktert tartalmaz az X$ változó. Pl.:

```
X$="KATALIN"
A=LEN(X$)
PRINT A
7
```

Gyakran előfordul, hogy adott számokat sztringként szeretnénk tárolni. Ez könnyű akkor, ha az adatokat billentyűzetről írjuk be, de a gép által generált számok esetén már más a helyzet.

Az A$=STR$(X) hatására az A$ sztringváltozó sztringként vesz fel az X változóban szereplő számot. Így már csak a szám alakja tárolódik, vele művelet nem végezhető. Pl.:

```
X=122
A$=STR$(X)
PRINT A$
 122
```

Az STR függvény pl. alkalmas arra is, hogy a grafikus üzemmódban a képernyőn műveletek eredményeit is megjelenítsük. Erre azért van szükség, mert a CHAR 1,X,Y,A$ utasításban a koordináták után csak sztring szerepelhet.

A sztringfüggvények kiválóan alkalmasak arra, hogy a képernyőszerkesztéseknél, táblázatkészítéseknél a felmerülő proglémákat kiküszöböljük.

A szépen szerkesztett táblázat lényege, hogy az azonos típusú adatok egymás alá kerüljenek. Ezt eddig a TAB formulával oldottuk meg. A táblázatok egészen addig szépek voltak, amíg az első oszlopban kiírt adatok hossza megegyezett.

Tízes szorzótábla

Szorzó	Szorzat
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100

Az elcsúszás abból következik, hogy a TAB mindig az utolsó kiírt karakterhez illeszti a következőket. Így ha az egyik szám hosszabb a többinél, a hozzá tartozó értéket a többihez képest jobbra írja. Megoldás lenne, ha meg tudnánk oldani, hogy az első oszlopban levő karakterek ugyanolyan hosszon ábrázolódjanak.

Használjuk ki a sztringfüggvények adta lehetőségeket! Töltsünk fel egy sztringváltozót 20 darab üres karakterrel! Legyen a változó neve B$. B$="
". Helyezzük el a szorzó aktuális értékét egy A$ nevű sztringváltozóban! A$=STR$(X).

Fűzzük össze a két sztringváltozónkat!

C$=A$+B$.

Mivel a szorzó biztosan nem nagyobb 10 ↑ 20-nál, ezért 20 karakternyi helyen elfér.

Vegyük ezért a C$ változó első húsz karakterét!

C$=LEFT$(C$,20)

Kiíratva C$ tartalmát, láthatjuk, hogy a gép így minden szorzót pontosan 20

karakternyi helyen ábrázol. Ez magát a szorzót és az őt 20 karakternyivé kiegészítő üres karaktersort jelenti.

A TAB utasítást alkalmazva így már biztosan megfelelő táblázatokat kapunk.

A módszer sztringek illesztésére is használható.

A sztringfüggvények alapos elemzése és gyakorlása sok segítéget jelent a sornyomtató használatának megfelelő megtanításához.

Felhasználásával érdekes és komoly feladatok is megoldhatók.

Karakterkódok táblázata

CHR$ kód	Hatás
0	
1	
2	
3	
4	
5	fehér
6	
7	
8	SHIFT+C=kikapcs.
9	SHIFT+C=bekapcs.
10	
11	
12	
13	RETURN
14	karakterkészlet csere kis/nagy betű
15	
16	
17	kurzor fel
18	RVS bekapcs.
19	HOME
20	DEL
21	
22	
23	
24	
25	
26	
27	ESC
28	piros
29	kurzor jobbra
30	zöld
31	kék
32	szóköz
33	!
34	"
35	#
36	$

CHR$ kód	Hatás
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	Ø
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T

CHR$ kód	Hatás
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	£
93	]
94	↑
95	←
96	
97	♠
98	
99	
100	
101	
102	
103	
104	
105	
106	
107	
108	
109	
110	
111	
112	
113	●
114	
115	♥
116	
117	
118	
119	
120	♣
121	
122	◆
123	+
124	
125	
126	
127	
128	
129	narancs
130	villogó be
131	villogó ki
132	

CHR$ kód	Hatás
133	f1
134	f3
135	f5
136	f7
137	f2
138	f4
139	f6
140	HELP
141	RETURN
142	karakterkészlet csere kis/nagy betű
143	
144	fekete
145	kurzor le
146	RVS kikapcs.
147	CLEAR
148	INST
149	barna
150	sárgászöld
151	rózsaszín
152	kékeszöld
153	világoskék
154	sötétkék
155	világoszöld
156	bíbor
157	kurzor balra
158	sárga
159	enciánkék
160	szóköz
161	▮
162	▬
163	▁
164	▔
165	▏
166	▒
167	▕
168	▒
169	◤
170	▌
171	▐
172	▪
173	▖
174	▗
175	▀
176	▝
177	▘
178	▚
179	▞
180	▙
181	▟

CHR$ kód	Hatás
182	▌
183	▖
184	▗
185	▘
186	▙
187	▚
188	▝
189	▞
190	▟
191	■

3.15 Logikai műveletek (AND, OR, NOT) és ON . . . GOTO

Az olyan program írása, amelyeknek futása közben egyszerre több feltételnek is teljesülnie kell, meglehetősen körülményes, mert sok elágazást tartalmaz.

A probléma megoldására vezették be a logikai műveleteket.

Segítségével egy IF . . . THEN utasításon belül egyszerre több logikai kapcsolat is vizsgálható. A logikai műveletek alkalmazása a matematikai logika szabályai alapján történik.

Mivel matematika oktatásunk nagy figyelmet fordít a gyerekek ilyen irányú képzésére (lásd a minden témánál előforduló „Döntsd el, hogy igaz-e az állítás" típusú feladatokat), ezért a gyerekek számára világosak ezek a kapcsolatok.

A számítástechnikában használt logikai műveletek:

AND

Magyarul ÉS.
Az X AND Y logikai értéke akkor igaz, ha mind az X, mind az Y logikai értéke igaz. Ellenkező esetben hamis.

A logikai műveletek igazságtartalmát táblázatokkal is meg szokták adni:

X	Y	AND
i	i	i
i	h	h
h	i	h
h	h	h

Pl. IF A>7 AND A<9 THEN . . .
Az IF . . . THEN kapcsolat logikai értéke csak 7<9 A esetén igaz.

OR

Magyarul VAGY
Az X OR Y logikai értéke csak akkor hamis, ha mind az X, mind az Y logikai értéke hamis.

Táblázattal:

X	Y	OR
i	i	i
i	h	i
h	i	i
h	h	h

Ezt úgy is mondhatjuk, hogy az X OR Y logikai értéke igaz, ha legalább az egyik állítás igaz. Pl. IF A>9 OR A<2 THEN . . .
Az IF . . . THEN kapcsolat logikai értéke csak a 2<A<9 értékekre hamis.

NOT

Magyarul: NEM
A NOT X logikai értéke akkor igaz, ha az X logikai értéke hamis.
Táblázattal:

X	NOT
i	h
h	i

Pl. IF NOT(A=8) THEN . . .
Az IF . . . THEN kapcsolat logikai értéke csak akkor igaz, ha A<>8.
Egy IF . . . THEN kapcsolaton belül több logikai művelet is elvégezhető. Ekkor a műveletek összessége adja meg az IF . . . THEN kapcsolat logikai értékét.
Pl. IF(A=3 OR B<8) AND C>6 THEN . . .

Több logikai műveletből álló műveletsor egyszerűsítésére néhány azonosságot is alkalmazhatunk.

Pl. De Morgan azonosságok:

NOT(A AND B)=NOT A OR NOT B

NOT (A OR B)=NOT A AND NOT B

A téma tanításához és gyakorlásához felhasználhatók a matematika tankönyvek ilyen típusú feladatai. A logikai kapcsolatok vizsgálatát érdekesebbé tehetjük, ha foglalkozunk a logikai áramkörök problémájával. Mivel a gyerekeknek megvannak az alapvető villamosságtani ismereteik, ezért az ilyen felaoatokat könnyen meg tudják oldani. Buzdítsuk őket ilyen feladatok önálló konstruálására, a bonyolult áramköröknek a matematikai logika eszközeivel való egyszerűsítésére. A témának széles szakirodalma van, vizsgálatával sokan foglalkoznak.

ON . . GOTO

Ha egy program több kisebb feladat végrehajtására íródott, akkor célszerű ezeket a funkciókat menüben felsorolni. Ezzel a témával már az előzőekben foglalkoztunk.

A menütechnika használhatóságát ugrásszerűen megnöveli az ON X GOTO típusú utasítás használata.

Az ON X GOTO A,B,C hatására a gép a program futtatását a GOTO után írt sorszámok közül annyiadikon folytatja, amennyi az X értéke. Ha nincs annyi sorszám feltüntetve, mint amit begépeltünk, a gép automatikusan a következő sorszámú utasítást veszi.

Pl.

```
40 PRINT"■■■■■■■■AZ ORS TAGJAINAK JELOLESE VETELKEDOKRE"
50 PRINT"1. SPORTVETELKEDO■"
60 PRINT"2. CSAPAT KRESZ VERSENY■"
70 PRINT"3. NEPDALENEKLESI VERSENY■"
80 PRINT"4. EGYENI MUVESZETI VETELKEDO■"
90 PRINT"5. TERMESZETTUDOMANYI (CSAPAT) VETELKEDO■"
100 PRINT"6. MATEMATIKA VERSENY"
110 INPUT"■MELYIK MODON DOLGOZUNK";X
120 ON X GOTO 350,380,410,440,480,510
```

Menüvel megoldható feladatoknál szorgalmazzuk a GOTO használatát, mert segítségével a program jelentősen lerövidíthető.

A gyerekek általában hamar belátják az utasítás használatának előnyeit. A GOTO-val való szervezés hatására a program áttekinthetőbb és kevesebb a hibalehetőség is.

3.16 A sornyomtató használata

A számítástechnika terjedésével remélhetőleg minden iskolába eljut a sornyomtató is. Ahol még nincs ilyen periféria, ott is érdemes szóban ismertetni a működését, gyakorlati jelentőségét.

A sornyomtató funkciói nagyon hasonlóak a képernyőéhez. Megjeleníthető rajta tetszőleges program listája, a gép üzenetei, a program futtatásakor kapott eredmények, tablók.

A Commodore 16-os számítógéphez általában grafikus nyomtatókat szoktak csatlakoztatni. Ezekbe úgynevezett nyolcvan oszlopos papírt (leporellót) fűzhetünk. Egy sorban tehát normál üzemmódban nyolcvan karakter helyezhető el.

A sornyomtató használatához speciális utasítások ismerete szükséges:

OPEN le, fe, m

A sornyomtatót használat előtt meg kell nyitni.
Az le, vagyis a *l*ogikai egységszám egy tetszőleges 1–255 közé eső szám lehet. Az egyszerűség kedvéért általában akkorának szokták választani, amekkora az fe, vagyis a fizikai egységszám. Ez a gép beállításától függően 4 vagy 5 (általában 4).

Az m értéke azt jelöli, hogy a kurzor felső (0) vagy alsó (1) állásban van. Normál üzemmód esetén m értékét nem kell beírni, mert önműködően beáll 0-ra.

A megnyitás után a sornyomtatóval vagy programot listázunk ki, vagy mint kimeneti egységet használjuk a program futása közben.

Listázás:

Listázáskor át kell adnunk a vezérlést a sornyomtatónak, hogy a lista megjelenhessen a papíron. Ez a CMD utasítással történik.

CMD

A CMD le hatására a sornyomtató átveszi a vezérlést a számítógéptől. Ekkor a gép üzenetei a sornyomtatón jelennek meg.

A logikai egységszámnak mindig meg kell egyezni azzal a logikai egységszámmal, amelyre a sornyomtatót megnyitottuk.

Pl. OPEN4,4
CMD4

Ha listázni kívánunk, akkor az OPEN és a CMD is mint parancs szerepelhet, nem kell programba építeni azokat.

A sornyomtatón való listázást a gép a LIST parancs begépelésével kezdi meg.

Ha vissza kívánjuk adni a vezérlést a számítógépnek, akkor a sornyomtatót le kell zárni. Ezt a PRINT # le begépelésével végezzük.

Pl. OPEN4,4
CMD4
LIST
PRINT# 4

Ha CMD üzemmódban szintaktikai hibát vétünk, a gép lezárja a sornyomtatót, és a vezérlést visszaadja a számítógépnek.

Ha nagyon sok programot kell kilistáztatnunk, kínos a PRINT # le sokszori begépelése. Ilyenkor a listázás után vétett szándékos szintaktikai hibával ezt az eljárást kiküszöbölhetjük.

Vigyázat! A PRINT # típusú utasítás esetében a PRINT nem helyettesíthető a kérdőjel begépelésével.

Ha a sornyomtató lezárását elmulasztjuk, a begépelt utasításokat a gép hibásan hajtja végre.

Táblázatok, szövegek, karakterek nyomtatása

A sornyomtató megnyitása után a géppel kiíratásokat végezhetünk. A kiíratáshoz minden esetben a PRINT # le utasítást kell használni.

PRINT # le

Az utasítás segítségével betűk, számok egyaránt kiírathatók a PRINT utasításban megismert szabályok alapján.

Pl. PRINT#4,A,B,$;"SZÖVEG"

Egy sorban nyolcvan karakter fér el, a vessző és a pontosvessző jelentése megegyezik a PRINT utasításnál tanultakkal.

PRINT # le, TAB(X)

A tabulátor formula használatára a sornyomtatón is lehetőség van. A sztringfüggvényeknél megismert módszer segítségével nagyon könnyen szerkeszthetünk táblázatokat.

PRINT#le, USING "###.##";X

A gép a USING után idézőjelek között szereplő maszk alapján helyezi el az X változóban szereplő számértékeket. Míg a TAB és a sztringfüggvények együttes használata a füzérek megfelelő elrendezését, addig a USING a számok rendezett kinyomtatását végzi.

A gép a számokat a maszkban talált tizedespontra illesztve írja ki.

Ha a tizedesponttól balra a kiírandó szám több helyiértéken ábrázolódik, mint a maszkban megadott, a gép a számok helyére csillagokat ír, ezzel jelezve a hibát.

A USING alkalmas a képernyő szerkesztésére is.

Vezérlőkódok

Néhány speciális művelet elvégeztetéséhez vezérlőkódok használata szükséges. A vezérlőkódok a PRINT# le után, attól vesszővel elválasztva fejtik ki hatásukat.

A következőkben közölt vezérlőkódok az MPS–801 típusú grafikus nyomtatóra vonatkoznak.

Ha más típusú nyomtatóval rendelkezik, akkor a sornyomtató gépkönyvét hasonlítsa össze az itt közöltekkel, mert lehetséges, hogy eltérések vannak.

Vezérlőkódok

Kód	Szerepe
CHR(10)	Soremelés nyomtatás után
CHR(13)	Soremelés
CHR(8)	Áttérés grafikus üzemmódba
CHR(14)	Áttérés dupla szélességű karaktermódba
CHR(15)	Áttérés standard karaktermódba
CHR(16)	A nyomtatás kezdőoszlopának megadása
CHR(18)	Áttérés reverz módba
CHR(146)	Reverz mód megszüntetése

A sornyomtatót a munka befejeztével le kell zárni.

CLOSE le

Az utasítás hatására a számítógép lezárja a sornyomtatót. A számítógép számára a sornyomtató csak új megnyitás után válik újra hozzáférhetővé.

Ha rendelkezésre áll a sornyomtató, célszerű ezzel a témával komolyabban foglalkozni, ill. a sornyomtató használatát a későbbiekben a feladatmegoldások során szorgalmazni.

Hívjuk fel a tanulók figyelmét arra, hogy a programokat nemcsak magnón érdemes rögzíteni, hanem sornyomtatón is. A hosszabb programok elemzése meglehetősen nehézkes úgy, hogy a képernyőn annak mindig csak egy darabja látszik. A programokat kinyomtatva ez a probléma megoldódik.

3.17 *A kazettás egység*

A kazetta programok és adatok tartós tárolására alkalmas.

Programok tárolása:

A gép memóriájában levő, lehetőleg kipróbált, jó programot kimenthetjük a kazettára. Ez alól csak a védett programok kivételek, de speciális másolóprogramokkal általában ez is megoldható.

A kazettán a programok egymás után, sorban helyezkednek el.

Mivel egy program általában kis helyet foglal el, ezért egy kazettán sok program tárolható. A programokat címmel szokták ellátni, így azok könnyen visszakereshetők. Célszerű megjegyezni, hogy melyik program hányadik fordulatnál kezdődik, így a visszakereséskor jelentős időt takaríthatunk meg.

A program kimentése kazettára:

A SAVE "PROGRAMNÉV" parancs begépelése után a képernyőn megjelenik a PRESS RECORD AND PLAY ON TAPE felirat.

A felhívásnak engedelmeskedve, a RECORD és a PLAY billentyűk egyidejű megnyomásának hatására a kimentés megkezdődik. Ekkor a képernyőn nem látunk semmit. A kimentés befejeztével a képernyőn újra megjelenik a kép.

Hogy a kimentés jól sikerült-e, arról a VERIFY parancs segítségével győződhetünk meg.

Visszatekerve a magnót a program elejére, és begépelve a VERIFY parancsot, megjelenik a képernyőn a PRESS PLAY ON TAPE felirat. Megnyomva a PLAY gombot, a gép összehasonlítja a szalagon levő programot a memóriában levővel. Ha a program kimentése sikeres volt, a gép kiírja: OK.

Ellenkező esetben a kimentést meg kell ismételni.

Program beolvasása kazettáról:

A kazettán levő programokat legegyszerűbben a programnév alapján kereshetjük vissza. Mivel a magnó meglehetősen lassan dolgozik, célszerű a programkezelésnek megfelelő fordulatra állítani. Ha ezt az értéket nem jegyeztük meg, a gép a programot a programnév alapján maga is megkeresi. Ha nem adjuk meg a keresendő program nevét, úgy a gép az első programot betölti.

A LOAD "PROGRAMNÉV" parancs begépelése után a képernyőn megjelenik a PRESS PLAY ON TAPE felirat. Ekkor a PLAY gombot lenyomva a program megkeresése megindul, a képernyőn nem látunk semmit. Amint a gép megtalál egy programot, leállítja a magnót és kiírja ennek a nevét. Várakozik néhány másodpercig, majd folytatja a keresést, ill. a betöltést attól függően, hogy milyen programot talált.

Ha nem akarjuk kivárni azt, hogy a gép magától folytassa a munkát, akkor a C= gomb lenyomásával azt meggyorsíthatjuk. Ekkor a gép rögtön végzi tovább a dolgát.

A betöltött program azonnal futtatható. Vigyázni kell arra, hogy ha a tárban volt már valamilyen program, arra új programot töltve, az eredeti elveszik.

Mivel a számítógépünk nem tudja egyszerre kezelni a képernyőt és a magnót, ezért a programozás során azokat szakaszosan érdemes működtetni.

A programokat a kazettára való kimentés mellett érdemes sornyomtatóval is rögzíteni, legalább két példányban tárolni, számítva az esetleges meghibásodásokra, felülírásokra.

Adattárolás mágnesszalagon

A szalag tudajdonságaiból adódik, hogy csak sorosan lehet rajta adatokat tárolni.

Ez azt jelenti, hogy ha ki akarjuk olvasni az adatállomány n. elemét, akkor előbb ki kell olvasnunk az azt megelőző (n–1) darab elemet. Ez viszonylag lassúvá teszi az adatállomány (angolul: file) kezelését.

A file rekordokból áll. Egy rekord a file egy egysége, ami akár több egyedi adatból is állhat.

Mivel a kazettás egység is periféria, ezért az adatforgalom számára meg kell nyitni.

Adatkivitel kazettára

A kazettás egység megnyitása írás esetén OPEN le,fe,m,"file-név" utasítással lehetséges.

A logikai egység száma: 1–255.
Fizikai egység száma: 1.
m (íráskor): 1.

A file-névre azért van szükség, hogy a visszakereséskor a gép megtalálja a keresett adathalmazt.

Az adatok felírása a PRINT#le utasítással történik.

Pl. PRINT#1,X

Ha egy PRINT # utasítással egyszerre több adatot is szeretnénk kiíratni, azokat egymástól pontosvesszővel és CHR$(13) vezérlőkóddal célszerű elválasztani.

Pl. PRINT # 1, A;CHR$(13);B;CHR$(13);C

Az adatok kimentése után a file-t le kell zárni.

A file lezárása a CLOSE le utasítással történik.

A logikai egységszámnak az OPEN, a PRINT és a CLOSE utasítás esetén meg kell egyeznie!

Adatok visszaolvasása

A kazettás egység megnyitása olvasásra az OPEN le,fe,m,"file-név" utasítással történik.

A logikai egységszám: 1–255.
Fizikai egységszám: 1.
m (olvasáskor): 0.

A file-névnek meg kell egyezni a rögzítéskor adott névvel. Az adatok visszaolvasása az INPUT le utasítással történik.

Pl. INPUT#1,X

Egy INPUT # utasítással több adat is visszaolvasható egyszerre. Ekkor a változókat vesszővel kell elválasztani egymástól.

Pl. INPUT#1,X,A$,B(I)

Az adatok beolvasását addig kell végezni, amíg egy ST nevű változó értéke nem egyenlő 64-gyel.

Ezért az ST változó értékét folyamatosan kell vizsgálni.

Az olvasás befejeztével a file-t le kell zárni.

A kazettás egység használata a programokba jól beépíthető.

Mutassuk meg a gyerekeknek, hogy ez a módszer alkalmas nagy mennyiségű adat hosszú távon történő tárolására.

Ügyviteli jellegű programok írásakor nélkülözhetetlen a mágneses adatrögzítés. Ügyelni kell arra, hogy hosszú távon az adattárolás csak akkor biztonságos, ha az adatállományt folyamatosan gondozzuk, vagyis néha egyik kazettáról a másikra átmásoljuk. Ezzel a szalag állás közben bekövetkező átmágneseződéséből eredő hibákat küszöbölhetjük ki.

Néhány ajánlott könyv felnőttek részére

DONALD ALCOOK: Ismerd meg a BASIC nyelvet! (Bp. Műszaki Könyvkiadó, 1983)
Kezdők számára igen jól használható, mert a nyelvezete könnyen érthető.

DR. KOCSIS ANDRÁS: TV BASIC (Bp. SZÁMALK, 1984)
A könyv forgatásakor vigyázni kell arra, hogy ha a könyvben megadott programokat futtatni szeretnénk, azokat előbb át kell írni Commodore 16-os gépre.

DR. KOCSIS ANDRÁS: Programozás BASIC nyelven (Bp. SZÁMALK, 1983)
Az ügyvitel programozása iránt érdeklődők haszonnal forgathatják ezt a könyvet.

DONALD D. SPENCER: Játékok BASIC nyelven (Bp. SZÁMALK, 1983)
A játékprogramok, ill. a játék matematikája iránt érdeklődőknek ajánlott. A felvetett problémák jó részének megoldásához előismeretekre van szükség. Ha a közölt programokat futtatni kívánjuk, azokat át kell írni Commodore 16-ra.

BENESCH–BUSSE–TWIEHAUS–WEITZEL: Kulcs a számítógéphez
Ez a négykötetes mű a számítástechnika négy fő területébe enged bepillantást.

SAIN MÁRTON: Matematikatörténeti ABC (Bp. Tankönyvkiadó, 1978)
A számítástechnika kialakulására vonatkozó részletek jól felhasználhatók az oktatásban. Gyerekeknek önálló feldolgozásra is ajánlott.

PÁL–RÉVBÍRÓ: Hetedhét Commodore 16 (Bp. NOVOTRADE, 1985)
A könyv kezdők számára érthetően, sok példával mutatja be a Commodore 16 működését. Gyerekeknek készült, de kezdő felnőttek számára is ajánlott ez a könyv.

Commodore 16 Felhasználói kézikönyv (Bp. NOVOTRADE, 1985)

Bevezetés a BASIC nyelvbe (Commodore 16 és Commodore plus/4) (Bp. NOVOTRADE, 1986)

MELLÉKLET

Megjegyzés a tanmenet-javaslatokhoz

A pécsi Apáczai Csere János Nevelési Központban folyó kutatások alapján úgy látjuk, hogy a két tanmenetben összeállított ismeretanyag átlagos képességű csoportok számára elsajátítható.

Az oktatás intézményünkben ciklusonként háromórás tömbökben folyik. Ez az időbeosztás jól bevált, ezért javaslom a kollégáknak is. A tanmenetek is ilyen bontásban készültek.

A csoportok összeállításánál figyelembe kell venni azt, hogy a gyerekekkel egyénileg is kell foglalkozni, ez pedig nagy létszám esetén nem oldható meg. Úgy tapasztaltuk, hogy a még igazán jól irányítható csoport maximális létszáma 14 fő, az ideális 8–10 fő.

A tanmenetben jelölt feladatok a feladatgyűjteményben találhatók. A feladatgyűjteményt végiglapozva látható, hogy jóval több feladatot tartalmaz, mint ahányat a tanmenetben javasoltunk. Természetesen ezek a feladatok is felhasználhatók önálló feladatok, önképzés formájában is.

Az egyes foglalkozásokra több feladatot javasoltunk, mint ahányat egy átlagos csoporttal fel lehet dolgozni. Az esetleg kimaradó feladatok a gyakorló órák anyagát képezhetik, ill. önálló feladatként felhasználhatók.

A differenciált foglalkozásra javasolt feladatok általában nehezebbek. Megoldását a jobban dolgozóknak javasoljuk, a kész feladatok elemzését már a csoport együtt is végezheti.

Az általunk adott feladatokon kívül jó tréning, ha a gyerekeket késztetjük problémák felkutatására, és azok önálló kidolgozására. Ezt azonban főleg a már programozni tudó gyerekektől kérhetjük.

TANMENETJAVASLAT A 7. OSZTÁLYOS FAKULTÁCIÓHOZ

Fogl.	*Téma*
1.	A gépterem rendje, balesetvédelem. Bevezető: A számítógépek kialakulása, a számítógép felépítése. A számítástechnika alkalmazási területei. A billentyűzet használata. (Ajánlott példák a feladatgyűjteményből, továbbiakban: F. Gy. 1–12) Számok és műveletek. Műveletek sorrendje, műveleti jelek, tizedespont. A PRINT, RETURN, vessző és pontosvessző használata (F. Gy. 16–18). Differenciált munkára javasolt feladatok: Billentyűzet játékai: F. Gy. 13–15. Számok és műveletek: F. Gy. 19–22.
2.	A változók fogalma, típusai (F. Gy. 23–29). A program fogalma. Egyszerű program írása alapján a programozás főbb lépéseinek megismerése (F. Gy. 32–33). Differenciált munkára javasolt feladatok: A változók: F. Gy. 30. A PRINT utasítás: F. Gy. 34.

3. Elágazás nélküli programok készítése PRINT utasítással. Képernyőkezelés a kurzormozgató utasítások programba építésével. TAB formula bevezetése. F. Gy. 31; 35; 36; 37; 40.)
A változók cseréje (F. Gy. 41–45).
Differenciált munkára javasolt feladatok:
Mit csinál a gép? (F. Gy. 211.)
A PRINT utasítás (F. Gy. 38; 39).

4. Elágazás nélküli programok készítése, INPUT utasítás bevezetése (F. Gy. 46; 47; 48; 49; 52; 53).
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 57–60; 64.)

5. Néhány gyakori függvény bevezetése.
Programok írása az INT(X), SQR(X), RND(X) függvények segítségével. (F. Gy. 66; 67; 68; 72; 74; 75.)
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 69; 70; 71; 73; 76; 77.)

6. Elágazás nélküli programok írása, gyakorlása. Válogatás a javasolt, de még meg nem oldott feladatokból.

7. Önálló programozás.
Az elágazás nélküli algoritmusok témájának lezárása, a tanulók munkájának értékelése. A feladatokat a javasolt, de még meg nem oldott feladatokból állítsuk össze.

8. Elágazásos algoritmus.
A folyamatábra szerepe a programozásban.
Egyszerű feladat megoldásán keresztül az elágazásos algoritmus vizsgálata. (F. Gy. 78.)
Az IF . . .THEN kapcsolat bevezetése. (F. Gy. 79–81.)
Differenciált munkára javasolt feladatok:
INPUT (F. Gy. 61; 62; 63; 65).

9. Az elágazásos algoritmus gyakorlása.
Egy programban több elágazás (F. Gy. 82 . . . 86).
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 94; 95; 97).

10. Az elágazásos algoritmus gyakorlása.
Rendezések (F. Gy. 89 . . . 92).
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 96; 98; 99).

11. A ciklus.
A ciklus részei, működése. Véges és végtelen ciklusok. (F. Gy. 100. . .103.)
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 106; 111; 112).

12. A tömb.
A tömb fogalma, egy-, két-, három- és többdimenziós tömbök értelmezése, modellezése. Tömbelemek megadásának módjai, tömbök kezelése.
(F. Gy. 113; 114; 117.)
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 115; 116; 119).

13. Gyakorlás.
Programozás az eddigi ismeretek alapján.
Ciklusok. (F. Gy. 104; 105; 107.)
Tömbök. (F. Gy. 120.)
Differenciált foglalkozásokra javasolt feladatok:
(F. Gy. 121).

14. Gyakorlás.
Hányszor fusson a program?
Ciklus. (F. Gy. 108; 109; 110.)
Válogatás a javasolt, de még meg nem oldott feladatok közül.

15. Ciklus sképző utasítás bevezetése.
FOR . . . TO . . . STEP utasítás bevezetése. (F. Gy. 122; 123; 124.)
Differenciált foglalkozásra javasolt feladatok. (F. Gy. 126; 127.)

16. Gyakorlás a FOR . . . TO . . . STEP és NEXT alapján.
Mit csinál a gép? (F. Gy. 212; 214; 216; 217; 218.)
FOR . . . TO . . . STEP és NEXT. (F. Gy. 130; 131; 132.)

17. Gyakorlás.
Rendezések n elemre.
F. Gy. 133 elemzése.
Válogatás a meg nem oldott feladatok közül.
Differenciált foglalkozásra javasolt feladatok. (F. Gy. 134; 135; 136.)
Mit csinál a gép? (F. Gy. 213; 215.)

18. Éves munka értékelése, programbörze.

TANMENETJAVASLAT A 8. OSZTÁLYOS FAKULTÁCIÓHOZ

Fogl. *Téma*

1. Ismétlés.
Elágazás nélküli algoritmusok.
INPUT, PRINT, változók.
Válogatás a javasolt, de meg nem oldott feladatok közül.

2. Ismétlés.
Elágazásos algoritmus.
IF THEN és a GOTO kapcsolata, ciklus.
Válogatás a feladatgyűjtemény példáiból.

3. Ismétlés.
Ciklusképző utasítás FOR . . . TO . . . STEP és NEXT.
Tömbök.
Válogatás a feladatgyűjtemény feladataiból.

4. READ, DATA, RESTORE utasítások bevezetése.
Adattárolás a programban.
A felhasználás ésszerű határai. (F. Gy. 137; 138.)
Differenciált munkára javasolt feladatok. (F. Gy. 141; 142; 143.)

5. READ, DATA, RESTORE gyakorlása. (F. Gy. 139; 140.)
GET A$, TI és TI$ bevezetése. (F. Gy. 144; 145 és 147; 148.)
Mit csinál a gép? (F. Gy. 230; 231.)
Differenciált foglalkozásra ajánlott feladatok. (F. Gy. 149 ; 150.)

6. Képernyőkezelés, grafika.
Színezések.
Grafikus üzemmód: pontok, vonalak a képernyőn. (F. Gy. 151 . . . 160.)
Differenciált foglalkozásra javasolt: F. Gy. 165.

7. Képernyőkezelés, grafika.
Görbék, sokszögek, feliratok a képernyőn. (F. Gy. 161 . . . 164.)
Differenciált foglalkozásra javasolt: F. Gy. 166.

8. Zene.
A hanggenerátorok használata.
Hanggenerátor és képernyő együttes használata. (F. Gy. 167 . . . 170.)
Differenciált foglalkozásra javasolt: F. Gy. 171; 172; 173.

9. Karakterfüzér függvények használata. (F. Gy. 174 . . . 185; 187.)
Differenciált foglalkozásra javasolt: F. Gy. 186; 188.

10. Logikai műveletek.
AND, OR, NOT bevezetése.
ON . . . GOTO.
Logikai áramkörök.
Mit csinál a gép? (F. Gy. 219 . . . 223.)
Logikai műveletek. (F. Gy. 189; 190.)
Differenciált foglalkozásokra javasolt: A néhány vegyes feladatból. F. Gy. 252.
Sornyomtató használata.
Vezérlőkódok.
Táblaképek tervezése, nyomtatása. (F. Gy. 193 . . . 199.)
Differenciált foglalkozásra javasolt: F. Gy. 200 ; 201.

11. Ha nem áll rendelkezésre sornyomtató, úgy tervezzünk gyakorlást az eddigi anyagból.

12–13. Válogatás az ajánlott, de meg nem oldott feladatokból. Differenciált foglalkozásra javasolt: F. Gy. 240; 249.

14. Kazettás egység.
Programok kimentése és beolvasása SAVE, LOAD, VERIFY használata.
Adattárolás kazettán.
Rekord, file (adatállomány), soros file.
Adatok rögzítése, visszaolvasása.
Javasolt feladatok: F. Gy. 202 . . . 207.
Differenciált munkára javasolt feladatok. (F. Gy. 208.)

15. Önálló programozás. (F. Gy. 238; 239; 241.)
Differenciált foglalkozásra javasolt: F. Gy. 243; 245; 248.

16. Önálló programozás. (F. Gy. 250; 254.)
Differenciált foglalkozásra javasolt: F. Gy. 253; 255.

17. Éves munka értékelése.
Saját programok futtatása.
Programbörze.

18. Üzemlátogatás.
Ezt a foglalkozást az év folyamán a lehetőségeknek megfelelően bármikor beépíthetjük az anyagba.

JEGYZET

TARTALOMJEGYZÉK

Kiadta: a NOVOTRADE RT.
A kiadásért felel: RÉNYI GÁBOR igazgató
Kiadványmenedzser: BÉKÉS TAMÁS
Műszaki szerkesztő: DÉVÉNYI ERIKA
A borító R. KOVÁCS ANNA munkája

Készült a Somogy Megyei Nyomdaipari Vállalatnál, Kaposváron
85-6735 – 17 000 példányban (5,6 A/5 ív terjedelemben)
Felelős vezető: Mike Ferenc igazgató

Ára: 52,—Ft