TVC BASIC

VIDEOTON

TV-Computer

BADICS BERNADETTE - BALOGH LÁSZLÓ

TVC BASIC

NOVOTRADE RT.

Lektorálta: Nagy Ákos
Szerkesztette: Siba László

A kiadásért felelős: RÉNYI GÁBOR, a Novotrade Rt. igazgatója
Budapest, 1988

ISBN 963 02 5544 8

Készült a Somogy Megyei Nyomdaipari Vállalat kaposvári üzemében

Felelős vezető: MIKE FERENC igazgató

TARTALOMJEGYZÉK

I. Rész
Alapok

I. RÉSZ
ALAPOK

Ülünk a számítógép előtt

A **TV-COMPUTER** kezelési útmutatója szerint bekapcsoltuk a számítógépet, áttekintettük a DEMO kazettán található programokat, játszottunk egy kicsit, és most már szeretnénk önállóan alkotni valamit.
Könyvünk – e cél szem előtt tartásával –, egyre bonyolultabb problémák felvetésével, és megbeszélésével vezeti el az Olvasót a **TVC BASIC** "magaslataira".

A programozási szabályok száraz felsorolása helyett kisebb-nagyobb feladatok megoldására irányítjuk figyelmünket.
A feladatok egymásra épülnek, mindegyik továbblépést jelent az előzőkhöz képest, és feltételezi az addig tanultak ismeretét. Aki most kezdi a tanulást, ne menjen tovább, amíg valami nem világos. Próbálkozzon saját ötlettel, gondolkozzon el a különbségeken!
Közös munkánk csak akkor lesz eredményes, ha aktív együttgondolkozás jellemzi.

Teljességre nem törekedhetünk, de igyekeztünk egy-egy utasítást minél több környezetben bemutatni.

Feladataink kezdőknek készültek, és a bonyolult matematikai formulákat szándékosan kerültük.
Lehetőségeinkhez képest igyekeztünk strukturált programozási elvekben gondolkodni, de az "igazi" programozástechnika begyakoroltatása nem célunk.

Könyvünket kedvcsinálónak szántuk és reméljük, hogy a továbblépést már ki-ki önállóan is meg tudja tenni a javasolt szakirodalmak segítségével.
Közös munkánk megkezdése előtt felhívjuk Kedves Olvasóink figyelmét, hogy egy feladat megfogalmazása és megoldása — jól működő számítógépes programmá alakítása — között sok teendő van, amelyet körültekintően és türelmesen kell elvégezni, hogy a várt sikerben mielőbb részünk lehessen.

Munkánk során egy alapkonfiguráció áll rendelkezésünkre (TVC, display, magnó).
Segédeszközeink a "Basic programozási segédlet", melyre (1)-gyel fogunk hivatkozni, és a "Kezelési útmutató", amelyet a továbbiakban (2)-vel jelölünk.

Írjunk programot!

A bekapcsolt számítógép várja az utasításainkat, parancsainkat.
Formailag csak annyi a különbség közöttük, hogy az utasításnak van sorszáma, egy program eleme, míg a parancsnak nincs.
Tartalmilag: a parancs, egy azonnal végrehajtandó feladatot jelöl ki a gépnek, míg az utasítást "megjegyzi" (eltárolja), és csak külön felszólításra hajtja végre. Lapozzuk fel az (1) "Parancsok és utasítások" c. fejezetet!

A beírt parancs, parancssor feldolgozása akkor kezdődik meg, ha visszaadjuk a vezérlést a számítógépnek a RETURN billentyű lenyomásával.
Egy utasítássor – akár a parancssor – a képernyőn több sornyi is lehet, hossza max. 250 karakter. Ez esetben a sor végét jelezzük a RETURN billentyű lenyomásával. A továbbiakban ezt a parancs-, illetve utasításlezáró karaktert nem jelöljük (hacsak nem kíván külön hangsúlyt), de ne felejtkezzünk meg róla! Így adjuk vissza a vezérlést a számítógépnek (a monitornak), jelezvén, hogy megkezdheti a beírt sorparancs feldolgozását.

A villogó négyzet – kurzor – jelzi, hogy a képernyő mely karakterpozíciójába tudunk éppen írni.
A beépített botkormánnyal bárhová elvihetjük, nem "tűnik el" a képernyőről.

Bekapcsolás után az első pár sorban üzenetet kapunk a BASIC interpreter (értelmező) verziójáról, a tár méretéről.
Az **ok** üzenet azt jelzi, hogy mi következünk, a gép kész parancsaink fogadására.

Próbálkozzunk! Írjuk például:

"képernyőtörlés"

Ezt nem értette meg, hibaüzenetet kaptunk:

*** Not understood
KÉPERNYŐTÖRLÉS

Az üzenetet angolul kaptuk, szószerint azt jelenti: nem értettem.

Mi lehet a baj? Ismételjük meg az előbbit csupa nagybetűvel, hátha csak ennyi. Láthatjuk ez sem segített.
Hiába, a gép csak a saját nyelvén ért meg bennünket, ami — mint a programnyelvek általában angol — kulcsszavakból áll. "Szókincse" azonban szűkös, bárki rövid idő alatt megtanulhatja, még ha nem is tud angolul.

A szándékunknak megfelelő kulcsszó:

CLS (A clear screen = képernyőtörlés rövidítése.)

A RETURN lenyomásáról se felejtkezzünk meg! Most már üres a képernyő, írhatjuk a programunkat!

```
10 PRINT  "Ez már program."
```

Itt a **PRINT** a kulcsszó, ezzel kérjük, hogy a gép az idézőjelben megadott szöveget írja ki a képernyőre. Kérhettük volna valamely matematikai kifejezés írását is.
Például:

```
10 PRINT "2+3-4="
```

vagy:

```
10 PRINT "2+3-4=" , 2+3-4
```

Ha az iménti programsorokat sorszám nélkül írtuk volna, látnánk rögtön az eredményt. Így sorszámozva tárolódik az utasítássor, és csak a végrehajtást kezdeményező parancs kiadása után kerül végrehajtásra.
Ez a **RUN** parancs, amit sorszám nélkül, RETURN-nel lezárva kell beírnunk. Ahányszor megismételjük ezt a parancsot, annyiszor írja ki a programunk a futás eredményét. Így aztán hamarosan a képernyő aljára jutunk.
Célszerű lenne minden kiíratás előtt a képernyőt törölni. Írjuk be a programba a megfelelő kulcsszót! Ezt az utasítást célszerű a már beírt utasítás elé beiktatnunk a programunkba, azaz a sorszámának kisebbnek kell lennie. Beíráskor nem kell ténylegesen elé írnunk.

```
 5 CLS
10 PRINT "2+3-4=" , 2+3-4
```

Indítsuk a végrehajtást a **RUN**, majd a **RUN 10** paranccsal. Gondolkozzunk el a különbségen!
Mi történik, ha a **RUN**-t utasításként is használjuk?

```
 5 CLS
10 PRINT "PROGRAM"
20 RUN
```

illetve

```
 5 CLS
10 PRINT "PROGRAM"
20 RUN 10
```

Mindkét esetben ki kell adnunk a **RUN** parancsot, hogy a program futása elinduljon. Most aztán újabb gondunk támadt. Mit tegyünk, hogy leálljon a kiírás? Kikapcsolhatnánk a gépet, de ez nem az igazi, hiszen akkor törlődik a programunk a tárból. A programfutás megszakítását a CTRL és az ESC gombok egyidejű lenyomásával kezdeményezhetjük (a továbbiakban ezt, és a többi billentyűkombinációt hasonló módon, így jelöljük: CTRL/ESC). Hatására megszakad a

végrehajtás. A **CONTINUE** (folytatás) kulcsszóval folytathatjuk, ha közben semmilyen javítást nem végeztünk.
Használhatjuk a **STOP** kulcsszót is a program megállítására, utasításként a megfelelő helyre beírva!

```
15 STOP
```

Tehát a leállás a kiírás után, az újraindítás előtt történik. A megállás okát a "STOP at line 15" üzenet jelzi (azaz: a 15-ös sorban van egy STOP). Továbbindíthatjuk a programot, az előbbiek szerint, **CONTINUE**-val. Ha javítunk, akkor már csak az újraindítás lehetséges.

Töröljük ki pl. az első sort. Egy teljes sor törlését úgy végezhetjük, hogy a sorszámot - RETURN-nel lezárva - újra beírjuk. Így egy üres sorral írjuk felül a régit.

```
5
```

Aki már lapozgatta az (1) segédletet, láthatta, létezik egy parancs is erre a célra, a **DELETE**. Használata akkor gazdaságos, ha több programsort vagy programrészletet - több egymás utáni sort - (más néven: programszegmenst) kívánunk törölni.

Ellenőrizzük a **LIST** paranccsal a sortörlést!

Nézzük meg, hogyan hajtódik végre utasításonként a programunk! Ezt a **TRACE ON** (nyomkövetés bekapcsolása) parancs beírásával kezdeményezhetjük. Indítsuk el a programot, láthatjuk a végrehajtott utasítások sorszámait.

```
<10> PROGRAM
<15>
STOP at line 15
ok
```

A **CONTINUE**-val indítsuk tovább!

```
<20> <10> PROGRAM
<15>
STOP at line 15
ok
```

Ebben a feladatban még nem tudjuk igazán értékelni ezt a parancsot, de később, amikor a programunk nem azt teszi, amit szeretnénk, e paranccsal informálódhatunk arról, merre is keressük a hibát.
Ha már vége a hibakeresésnek, a nyomkövetést a **TRACE OFF** paranccsal kapcsolhatjuk ki.

Általános iskolás vagy?

Döntse el a számítógép! Írjuk meg azt a programot, amely az életkor alapján eldönti a kérdést, majd az eredményt "intelligens" formában kiírja a képernyőre.

Pontosítsuk a feladatot! Ha az érdeklődő még általános iskolás korú, írja ki a program: Te még nagyon fiatal vagy!
Tanulj szorgalmasan!
Ha nem általános iskolás (még nem, vagy már nem), az üzenet legyen: Te nem vagy általános iskolás!
Tanulj szorgalmasan!
A döntéshez a programmal közölnünk kell az életkort. Használjuk ennek beolvasására az **INPUT** utasítás lehetőségeit.
A paraméterezés 2. és 3. változata alkalmazható feladatunkban. (Az adatot a billentyűzeten kívánjuk megadni, a periféria megnevezése nélkül.) Egy adatunk lesz, tehát egy változó nevet kell megadnunk, amely felveszi az adat értékét. A programban erre a változó névre hivatkozva férhetünk hozzá a — már korábban — beolvasott adathoz. A változó névben jelölnünk kell azt is, hogy szöveges vagy numerikus

információt kívánunk beolvasni (NS ill. N). Másképpen fogja tárolni a gép a begépelt karaktereket!

Egy **INPUT**-tal több adat is beolvasható, s ezek akár különböző típusúak is lehetnek, a sorrendjük helyességére azonban ügyelnünk kell. A beolvasás előtt szöveges üzenettel célszerű tájékoztatnunk erről a program felhasználóját.
Eddigi ismereteink szerint ezt egy **PRINT** és egy **INPUT** utasítással tudjuk megoldani. E kettő összevonása az **INPUT PROMPT ...**

```
10 CLS
20 PRINT "Hány éves vagy?"
30 INPUT N
```

illetve

```
10 CLS
20 INPUT PROMPT "Hány éves vagy?" " : N
```

Az üzenet az idézőjelek közötti karaktersorozat lesz, az N előtti kettőspont (:) pedig, elválasztó (termináló) karakter, a használata kötelező.

Nem ismerjük még azt az utasítást, amely az N vizsgálata alapján más-más utasítások végrehajtására ad lehetőséget, azaz a program elágazását teszi lehetővé.
Ez az **IF — THEN — ELSE** utasításhármas.
Ez az **IF** utáni feltételtől függ a folytatás. Ha a feltétel igaz, akkor a **THEN** utáni utasítás (-sorozat) hajtódik végre a kettőspontig, az **ELSE** utániak kimaradnak, és a következő sorszámon folytatódik a program. Ha a feltétel nem igaz, akkor a **THEN**-ág marad ki, és az **ELSE** utáni utasítások kerülnek végrehajtásra.
Ha az **ELSE**-ág hiányzik, és a feltétel nem teljesül, akkor a következő sorszámú utasításon folytatódik a program végrehajtása.

Próbálkozzunk!

```
30 IF 6 <= N >= 14 THEN "Te még nagyon fiatal vagy!"
   : ELSE PRINT "Te nem vagy általános iskolás!"
40 PRINT "Tanulj szorgalmasan!"
```

Valami baj van a 30-as sorral, így nem érti az interpreter.
A feltétel helyes megadása:

```
N <= 6 AND N >= 14
```

A javítást természetesen a hibaüzenetben megadott sorban kell elvégeznünk.
Pl. a következőképpen: állítsuk a kurzort a 6-osra, az INS gomb háromszori lenyomásával 3 üres helyet szúrjunk be elé, majd ide írjuk be az (N >=)-t. Mozgassuk a kurzort az eredeti N-re, az előtte lévő (<=)-t a DEL kétszeri lenyomásával töröljük. Hasonló módszerrel szúrhatjuk be az **AND** logikai operátort is. A javítás végét ne felejtsük el RETURN gomb lenyomásával jelezni. Ezt a sor bármely pontján megtehetjük. A **LIST** parancs kiadásával ellenőrizzük a javítást.

A programsoraink sorszámait úgy adtuk meg, hogy az újakkal felülírtuk a nem kellő régieket, s nem maradt az előző programból egy olyan sor sem, amely "összekeveredhetne" az újakkal. Erre azonban nem tudunk mindig figyelni. Egy új program írása előtt a régit törölni kell a tárból a **NEW** (új) paranccsal.

Eddigi csekély tapasztalataink alapján megállapíthatjuk, hogy a gép csak a saját szabályai szerint ért meg bennünket. Az utasítások formai (szintaktikai) és tartalmi (szemantikai) szabályait pontosan kell ismernünk és betartanunk.
A számítógép ugyan nem romlik el a hibás utasításoktól és parancsoktól, de nem azt csinálja, amit mi szeretnénk. És ez éppen elég bosszúság! Ezért javasoljuk, hogy amíg kellő jártasságra nem teszünk szert, minden új utasítás használatakor lapozzuk fel az (1)-et.

Fáradozásunk hamarosan megtérül. Sokkal kevesebb időt kell hibajavítással töltenünk!

Mi a neved?

Írassa ki mindenki a saját nevét 10 sorban, soronként kétszer. Az előző feladathoz hasonlóan, először, a név egy kívülről (billentyűzetről) jövő adat legyen, amelyet az **INPUT**-tal tudunk beolvasni. Ez az adat azonban szöveges (füzér azaz string) típusú.
A fogalom pontosításához lapozzuk fel az (1)-et a Konstansok, ill. a Változók címszavaknál.

```
20 INPUT PROMPT "NÉV:" : N$
```

Módosítsuk az utasítást úgy, hogy az adatot konstansként kezelve, eleve beépítjük a programba.

```
20 LET N$ = "KOVÁCS ANDREA"
```

vagy

```
20 N$ = "KOVÁCS ANDREA"
```

A **LET** kulcsszó azt jelenti, legyen egyenlő. Használata nem kötelező. Amíg az N$-nak más értéket nem adunk, ez a név lesz a tartalma.
Biztosan akad, aki furcsának találja, hogy egy feladaton belül az N$ tartalma más is lehet, de ez a számítástechnikában így van! A név és az érték összerendelése csupán átmeneti. A számítási folyamat pillanatnyi állapotát a változók pillanatnyi értéke jellemzi. Ettől nem kell megijedni, némi gyakorlat után természetessé válik!

Feladatunkban számolnunk kell, hányadik kiírásánál tartunk, hogy a 10. után befejezhessük. Erre a célra vezessünk be egy S segédváltozót.

Indulási értéke 0, amit minden kiírás után növeljünk 1-gyel (inkrementáljuk).
Eddigi ismereteinkkel megoldható a feladat:

```
NEW
10 CLS
20 N$ = "Kovács Andrea"
30 S = 0
40 S = S + 1 : PRINT N$ ; N$
50 IF S = 10 THEN STOP : ELSE 40
```

A 40-es sorban a kettősponttal 2 utasítássort vontunk össze egy sorszám alá. Ezzel a program áttekinthetőségét növeltük. Célszerű máskor is alkalmazni a szorosan összetartozó utasítások csoportosítására. Tudjuk, ennek az összevonásnak az szab határt, hogy egy programsor max. 250 karakterből állhat. A pontosvessző (;) a kiírás formátumára vonatkozik. Hatására a nevek közvetlenül egymás után íródnak a sorban. Lapozzuk fel a **PRINT** utasítást az (1)-ben, próbáljuk ki a kiíratás egyéb formában is.

A feladatban az S változó adott kezdőértékéből kiindulva adott növekménnyel, adott végértékig kell ugyanazt a feladatsort végrehajtani. Ezt másként ciklusnak nevezzük. A BASIC-ben erre van egy külön utasításkombináció: **FOR — TO — STEP — NEXT.**
A **FOR** után nevezzük meg a változót és adjuk meg a kiindulási, a **TO** után pedig, a végértékét. A **STEP** után határozhatjuk meg a változó ciklusonkénti növekményét.

```
30 FOR S = 1 TO 10 STEP 1
```

A ciklusonként végrehajtandó feladatsort ezek után kell megadni, majd a **NEXT** S-sel a ciklusváltozó következő értékét venni. Mindaddig ismétlődik a ciklus magja, amíg a változó új értéke a végértéket meg nem haladja.
Egészítsük ki programunkat!

```
10 CLS
20 N$ = "Kovács Andrea"
30 FOR S = 1 TO 10 STEP 1
40 PRINT N$ ; N$
50 NEXT S
60 STOP
```

Ha a **NEXT** S határása generált S = S + 1 érték nagyobb a kijelölt végértéknél, átlép a ciklusmagon, jelen esetben a kiíráson. A program végrehajtása a következő utasításon folytatódik (STOP).

A végeredményt tekintve a két megoldás között nincs különbség. Az S változó végértéke a korábbi változatnál S = 10 lesz, míg a FOR-ciklust használva S = 11.
Ezt le is tudjuk ellenőrizni a program futása utáni **PRINT** S paranccsal. (Természetesen ezt a programba utasításként is beírhatjuk a megfelelő helyre.)

Programunk szépséghibája, hogy a leállításra, ill. a befejezésre a **STOP**-ot használtuk, bár tudjuk, hogy ily módon a futás csak felfüggesztődik. Ismerkedjünk meg a programot ténylegesen lezáró utasítással: **END**.
Módosítsuk a megfelelő sorokat! Az első verzióban az 50-est kellett volna:

```
50 IF S = 10 THEN END : ELSE 40
```

a másodikban a 60-ast kell:

```
60 END
```

Aki figyelmesen elolvassa az (1)-ben a FOR — TO — NEXT ciklusutasításra vonatkozókat, az tudja, hogy: ha a lépésköz értéke (a növekmény) +1, nem kell megadni, mert az előre meghatározott (implicit).

```
30 FOR S = 1 TO 10
```

Gondolkozzon el azon, mi történik, ha az S nagyobb mint 22, vagy ha a név 18 karakternél hosszabb?
Próbálkozzunk! S a második kérdésre adjunk közösen választ. Módosítsuk a feladatot, hogy a paramétereket — a változók értékeit — "kívülről" kérje:

```
10 CLS
20 INPUT PROMPT "Mi a neved? :" N$
25 INPUT PROMPT "Hány sorba írjam?:" : N
30 FOR S = 1 TO N
40 PRINT N$ ; N$
50 NEXT S
60 END
```

Az N$, mint szöveges típusú változó, nem más, mint egy tömb, ahol az elemhossz a benne lévő karakterek számával egyenlő. Alapértelmezésben (impliciten) az ilyen szöveges típusú tömbök megengedett elemhossza 18 (ennyi hely áll a rendelkezésére a tárban). Ha a szövegünk ennél hosszabb, gondoskodnunk kell a nagyobb hely lefoglalásáról, különben

```
* * * Overflow (túlcsordulás)
```

hibaüzenetet kapunk. Ilyen esetben az adatbevitelt egy helyfoglaló utasításnak kell megelőznie, pl.:

```
15 DIM N$*20
```

Ekkor már 20 karakterből állhat az a szöveg, amit hiba nélkül elfogad a gépünk.
Ellenőrizzük az állításunkat a

```
PRINT LEN(N$)
```

paranccsal, ahol a LEN(N$) függvény a beírt név hosszát adja meg.
Nem elég a **DIM** utasítást használni, a felhasználót is tájékoztatni kell, hogy milyen hosszú név beírására adtunk lehetőséget, pl.:

```
15 DIM N$*24
20 INPUT PROMPT "Mi a neved? (max. 24. kar.):" : N$
```

Óvakodjunk a felesleges helyfoglalástól, a túlbiztosítástól. Egy nagyobb feladatnál már számottevő lehet ez a többletként lefoglalt tárterület. Törekedjünk arra, hogy a tömböt a ténylegesen kellő elemhosszal definiáljuk.

Az iménti problémát (a név hosszabb 18 karakternél) másként is megoldhattuk volna. Az adatbeíráskor a nevet két részre bontjuk, vezeték- és keresztnévre, s majd kiíráskor a két név "összegét" képezzük.
Gyakorlásképpen oldjuk meg a feladatot így is.

Technikai kitérő

Bizonyára akad, akinek nehézséget jelent egy hibás sort, egy félregépelt karaktert kijavítani. Ebben a fejezetben segítségképpen összegezzük a főbb tudnivalókat. A (2) elején részletes leírást olvashatunk a szerkesztési és programvezérlési funkciókról.

DELETE (törlés): a DEL gombbal kezdeményezhető. A kurzor előtt álló karakter törlődik és a kurzor balra lép, nem marad üres hely. Így végig lépegetve egy teljes sort is törölhetünk, de mint láttuk, van erre jobb megoldás is.

SHIFT/DEL: a 2 billentyűt együtt lenyomva azt a karaktert törölhetjük, amin a kurzor áll. (A sort ez is tömöríti.)

DELETE LINE (sortörlés): ilyen gomb nincs, ezt a funkciót a CTRL/Y valósítja meg.

A CTRL/K: a sormaradék törlésére ad lehetőséget, azaz a kurzor pillanatnyi helyétől jobbra eső karaktereket törli.

INSERT (beszúrás): az INS gombbal a kurzor elé iktathatunk be üres helyet (annyiszor kell lenyomni, ahány helyre szükségünk van). Ide írhatjuk aztán a beszúrni kívánt karaktereket. Használata előtt a kurzort ne feledjük arra a karakterre pozicionálni, amelyik elé beírni kívánunk.

INSERT LINE (sorbeszúrás): nincs ilyen gomb, a CTRL/N valósítja meg ezt a funkciót. Az előbbihez hasonlóan itt is előbb a kurzort arra a sorra kell vinni, amely elé új sort kívánunk beiktatni. (Tudjuk, új programsor beírása a sorszám megfelelő megválasztásával történik.)

Program javításakor használjuk bátran a **LIST** parancsot! Hatására a kijelölt programsor vagy programrészlet megjelenik a képernyőn, így nem kell a teljes sort újra írnunk, a régin elvégezhetjük a módosítást.
A **LIST** parancs paraméterezési lehetősége is nagyon kényelmes.

LIST 30 — csak a 30-as sort írja ki
LIST -30 — a program elejétől a 30-as sorig írja ki (a 30-ast is)
LIST 30-50,90 — a 30-as és az 50-es közé erő sorokat írja ki (a határokat beleértve), és a 90-est

Fontos, hogy a javítás után a **RUN** parancsot üres sorban adjuk ki, ill. valamely sort felülírva, ne felejtkezzünk meg a sormaradék törléséről (CTRL/K).

Szinusz függvény

Rajzoljunk egy szinusz görbét úgy, hogy az töltse ki a képernyőt! Rajzoljuk meg a koordinátatengelyeket is!

A feladat nem tűnik bonyolultnak, hiszen a szinusz egy beépített függvény. Azaz olyan alapfüggvény, amelyre a programunkban a nevével hivatkozhatunk. Tájékozódjunk az (1)-ben, mely függvényeket találjuk a beépített függvények között.
A görbe megrajzolásához sorra ki kell számítanunk az összetartozó értékpárokat, majd kirajzolni és összekötni őket. Pontosítsuk a feladatot!

Jelöljük ki az értelmezési tartományt! Legyen a (0,2*PI) intervallum. A szöget számoljuk radiánban, az egyszerűség kedvéért, a lépésközt válasszuk 0,02-nek.
A függvény pontjainak kiszámításához a már ismert **FOR — TO — NEXT** utasítást használjuk:

```
 60 FOR I = 0 TO 2*PI STEP 0.02
 65 Y = SIN (I) : Z = SIN (I + 0.02)
  .
  .
  .
100 NEXT I
```

Az ily módon kiszámított függvénypontok összekötését a **PLOT** utasítással végezzük. Tudnunk kell róla, hogy a számítógép - az üzemmódtól függetlenül - logikai koordinátákat vár (amelyeket majd "ő" számol át tényleges fizikai koordinátákra). A logikai koordináták vízszintesen 1024, függőlegesen 960 képpontra osztják a képernyőt. A (0,0) koordinátájú pont a bal alsó sarok. Ha tehát, mi a PLOT utasítást az imént kiszámított pontpárok összekötésére használjuk, csalódni fogunk.

```
70 PLOT I,Y ; (I * 0.02) , Z
```

A képernyő csücskében látszik valami, de nem ezt vártuk. Az ábránk akkor lesz megfelelő, ha a függvénypontokat átszámoljuk logikai koordinátákra, azaz normáljuk az ábrát a képernyőre. Kezdjük az okoskodást elölről!

A koordinátarendszerünk origóját helyezzük a (100,480) pontba. Ha ezen a ponton keresztül függőlegesen is és vízszintesen is húzunk egy-egy egyenest, már meg is rajzoltuk a koordinátarendszert. A tengelyeket hosszabbítsuk meg az origón túl is. Jelöljük ki ennek megfelelően az összekötendő pontokat. Az x tengelyhez: (50,480) és (900,480).
A koordináta geometriában járatlan olvasók számára hangsúlyozzuk, hogy a 2. koordináták azonossága biztosítja, hogy ez egy vízszintes egyenes legyen, és 480-as értékük pedig azt, hogy átmegy a kijelölt origón. Az y tengely megrajzolásához az origó alsó koordinátájával megegyező x koordinátájú pontokat kell választanunk. (Egyet a képernyő alján, egyet a tetején.) Legyen pl. (100,80) és (100,900).
A tengelyeket megrajzoló utasítások:

```
40 PLOT 900,480 ;  50,480
50 PLOT 100,900 ; 100,480
```

A pontosvessző hatására a "toll a papíron maradva" az egyik pontból a másikba megy. Az összekötővonal típusát az (1) függeléke szerint megváltoztathatjuk a **SET STYLE** N utasítással, ahol N a vonaltípus száma. Ha nem adunk meg semmit, a vonal folytonos lesz.

A függvény leképezését úgy végezzük el, hogy a (0,2*PI) intervallumot közelítően megfeleltetjük a (100,800) képpontoknak. Ebből adódik egy nagyítási arány:

$$\frac{80 - 100}{2*PI} = 111,4$$

Kerekítsünk, válasszuk 110-nek, és adjuk meg a vízszintes logikai koordinátákat:

$$x = 100 + (110*I)$$

Ahol a +100 az origó eltolásából adódik, a 110 pedig az imént kiszámolt nagyítási arány.

A függvény értékkészlete (azaz lehetséges értékei):

$$-1 \leq y \leq 1$$

Ezt a (80,880) képpontokra képezzük le. Az előbbiekhez hasonlóan kiszámítható a nagyítási arány: 400
A függőleges logikai koordináták:

$$y = 480 + 400*\sin(I)$$

Ahol a 480 a koordináta rendszer eltolása miatt kell, a 400 pedig... De hiszen ezt már tudjuk!
A függvényt megrajzoló programrészlet:

```
 60 FOR I = 0 TO 2*PI STEP 0.02
 80 X = 100 + (110*I) : Y = 480 + 400*SIN(I)
 90 PLOT X,Y
100 NEXT I
```

Ha igazán precízek akarunk lenni, a tengelyeken jelölnünk kell az egységet. Végezze el mindenki önállóan!

Mit tegyünk, hogy a programunk egy külső szemlélőnek is "beszédes" legyen? Magyarázatokat — megjegyzéseket (kommenteket) — írunk a szükséges helyekre. Erre a **REM** utasítás alkalmas, vele egyenértékű a felkiáltójel (!). Formailag utasításként írjuk a programba, de az interpreternek ezek "átlátszó" utasítások, azaz csak a programlistán jelennek meg, tárgykódot nem képeznek. A **REM** után írt magyarázatok a

listán pontosan megőrzik eredeti formájukat, annyi betűköz lesz, ahányat írtunk, a kis- és nagybetűk sem változnak meg, a szöveg része lehet az idézőjel (") is.

Bizonyára sokan észrevették, hogy az utasítások formai (szintaktikai) szabályai nem túl szigorúak. Utasításainkat akkor is megérti az interpreter, ha kisbetűvel írtuk, és ha nem tettünk a sorszám után szóközt (space-t)... A **LIST** paranccsal kért lista azonban már tagolt lesz, formailag nem biztos, hogy azonos a beírtakkal. Szabály helyett csak annyit: utasításainkban az összetartozó információkat ne tagoljuk feleslegesen, de az utasítássorok áttekinthetőségéhez kellő mértékben alkalmazzuk a szóközt, így biztos, hogy a gép számára is érthető lesz, amit leírtunk.

Írjunk magyarázósorokat az iménti programhoz!

```
 5 REM Szinusz egy periódusának megrajzolása a teljes képernyőre
35 ! x tengely
45 ! y tengely
70 ! x irányban 110-szeres nagyítás,
     y irányban 400-szoros nagyítás.
```

Próbáljuk ki a feladatot más színösszeállításban is. Az alap (4 színű) üzemmódnál maradva, az alapszínek legyenek:

sötétkék, szürke, sötétvörös, sárga

Ezek a színek kapják rendre (a definiálás sorrendjében) a 0, 1, 2, 3 színsorszámokat. (Lapozzuk fel a (2) idevonatkozó 4. fejezetét.) A keretszín nemcsak az imént felsorolt színek valamelyike lehet, hanem tetszőleges palettakódú is. Válasszuk sötétsárgának, azaz 20-as palettakódúnak. Az üzemmód 4 alapszíne a

```
30 SET PALETTE 1, 21, 4, 84
```

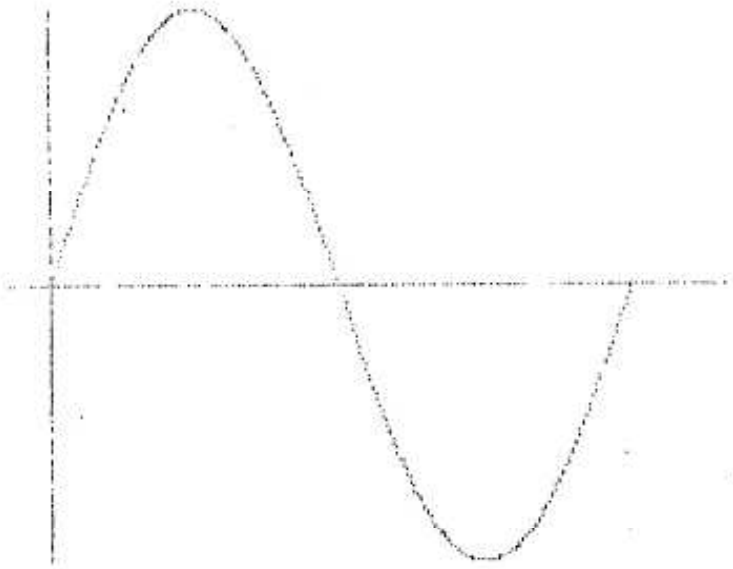
utasítással definiálható. (A táblázat alapján a kiválasztott színeknek megfelelő palettakódok: 1, 21, 4, 84.)
Ebből a sötétkék legyen a papír színe, a sárga tintáé. Ezeket a hozzájuk rendelt színszámokkal egy definiáló utasításban kell megadni.

```
30 SET PALETTE 1, 21, 4, 84 : SET PAPER 0 : SET INK 1 :
   SET BORDER 20
```

Nem definiáltuk még a 4 színű üzemódot! Ez ugyan az alapértelmezés (és most is ez érvényes), de az új színek érvényesítése miatt mégis használni kell. Az újonnan definiált színpalettát így tudjuk beállítani, és egyben elvégzi a képernyőtörlést is.
Tekintsük a teljes programlistát:

```
 10 !*****************************
    *  sinus rajzolása a tel-*
    *      jes képernyőre     *
    *****************************
 20 GRAPHICS 4:
    !*sötét kék,szürke,sötét
      vörös,sárga*
 30 SET PALETTE 1,21,4,84:
    SET  PAPER 0:SET BORDER 20:
    SET INK 1
 35 !** x tengely **
 40 PLOT 900,480;50,480
 45 !** y tengely **
 50 PLOT 100,900;100,80
 60 FOR I=0 TO 2*PI STEP 0.02
 70 ! x irányban 110-szeres,
      y irányban 400-szoros na-
      gyitás.
 80 X=100+110*I:
    Y=480+400*SIN(I)
 90 PLOT X,Y
100 NEXT I
```

És íme munkánk eredménye:



Jogos igényünk, hogy tároljuk ezt a programot, hiszen sok munkánk van benne. Tudjuk, hogy a tárból kitörlődik a program, ha a gépet kikapcsoljuk.
Más periféria híján írassuk ki programunkat magnókazettára.
A kazettás magnót a (2) szerint helyezzük működőképes állapotba, majd adjuk ki a

SAVE "SZINUSZ"

parancsot. Vigyázzunk! A parancs lezárása előtt indítsuk a magnót (a felvételt), különben a program fejrésze (a rávonatkozó — betöltéshez nélkülözhetetlen — információ) hiányos lesz. Célszerű a számlálót is beállítani, ha van ilyen a magnónkon. Programkiírás közben a képernyő keretszíne kétszer lesz csíkos. Először amikor a fejrész, majd amikor az utasítások kerülnek kivitelre. Ne kapcsoljuk ki a magnót az ok üzenet megjelenése előtt!

Ellenőrizzük, hogy a kiírt program azonos-e a tárban találhatóval. (Most még megismételhetjük a folyamatot, ha hibát észlelünk. Ha az ellenőrzést elmulasztjuk és a legközelebbi betöltéskor derül ki, hogy hibás a kimentett program, már nincs mit tenni, írhatjuk újra!) Az

ellenőrzést a **VERIFY** paranccsal végezzük. Tekerjük vissza a magnókazettát a program elé, és a parancs lezárása után indítsuk el. A parancsban megadhatjuk a program nevét is, ez akkor célszerű, ha nem tudunk pontosan az ellenőrizni kívánt program elejére állni. Ha megérkezett az ok üzenet, leállítjuk a magnót, és kezdhetjük a következő feladatot.

Nem esett még szó arról, hogyan tudjuk a szalagon lévő programot a tárba tölteni. Úgy gondoljuk, ez már senkinek sem rejtély, hiszen a DEMO kazettát bizonyára mindenki kipróbálta.
A teljesség kedvéért azért beszéljük meg!

```
LOAD "SZINUSZ"
```

vagy

```
LOAD #5: "SZINUSZ"
```

vagy

```
LOAD
```

A név nélkül kiadott parancs esetén természetesen a program elejére kell állítani a magnót. Névvel megadva meg fogja keresni (de csak előre haladva), s közben kiírja, mely programokat talált még keresés közben (FOUND). Amint rátalál a kért programra, beolvassa (READ).

```
FOUND : ALMA
FOUND : KÖRTE
READ  : SZINUSZ
ok
```

A periféria kijelölés (#5:) nem kell feltétlenül, mert a magnó az iménti parancsok alapértelmezett perifériája, azaz, ha nem jelölünk ki más perifériát, ezt tekinti cél, ill. forrás perifériának.

Esik a hó

Képzeljük el, hogy tél van, ülünk egy autóban és a szélvédőre hullanak a hópelyhek. Amikor már alig látunk ki, letöröljük őket.
Legyen most a szélvédő a képernyő és írjuk meg e jelenséget utánozó programot!

Gondolkozzunk eddigi ismereteink alapján! A beépített függvények között láttuk a **RND(X)**-et, amiről azt olvashattuk; véletlen (álvéletlen, de a mi szempontunkból ennek nincs jelentősége!) számokat generál. Az argumentumában megadott értéknél kisebb pozitív egész számokat ad eredményül.
A hópelyhek legyenek a csillagok (*), és a képernyőre véletlenszerűen kerüljenek. Az **RND(X)**-et a kiíráshoz megadandó sor és oszloppozíció előállítására alkalmazzuk.
Tudjuk, hogy az alapüzemmódban 24 sor és soronként 32 karakterhely van a képernyőn. Így a koordinátákat meghatározó és kirajzoló utasítássorok:

```
40 X= RND(24) + 1 : Y = RND(32) + 1
50 PRINT AT X,Y : "*"
```

Miért nem az RND(24)-öt, ill. RND(33)-at írtuk?
A törlést adott darabszám kiírása után kell elvégezni.
Töröljük 24-enként!

```
20 FOR I = 1 O M: REM M a hópelyhek száma
30 IF I = 24 *INT(I/24) THEN CLS
70 NEXT I
```

A látvány érdekében a kiírást lassítani kell, a 70-es sor elé valami időzítést kell beiktatni, pl. egy számlálóciklust.

```
60 FOR D = 1 TO 100 : NEXT D
```

Gondoljuk, többen szokatlannak találják a 30-as sort, ahogy azt figyeljük, hogy a 24 csillag kiírása megtörtént-e.
Az **INT(X)** függvényt is megtaláljuk a beépített függvények között. Ott megtudhatjuk, hogy pozitív számok esetén az argumentumának a tizedesek nélküli (egész) részét adja. A feltételünkben az egyenlőség csak akkor teljesül, ha az I maradék nélkül egyszer osztható 24-gyel. Ez a legegyszerűbb módja az oszthatóság vizsgálatának.
Természetesen akinek más ötlete van, próbálja meg aszerint és hasonlítsa össze a két módszert. Minden feladatnak létezik többféle megoldása. A tanulás fázisában az a célunk, hogy a futó program ténylegesen azt csinálja, amit a feladatban megfogalmaztunk. Amikor már az alapfeladatokon túljutottunk, a megoldás "szépségére" is figyelni kell!
Nagyon örülnénk, ha könyvünk olvasása közben ki-ki bátorságot kapna az önálló munkához és mielőtt a kész programot megnézné, próbálkozna egyedül is a feladat megoldásával.
Előbbi feladatunknál a program – indítás után – kérdezze meg, hogy "Essen a hó?". A hóesés csak akkor induljon, ha erre igenlő jelt adunk.

```
20 INPUT PROMPT "Essen a hó?" (I/N) :" V$
30 IF V$ = "N" THEN PRINT "Sajnálom": END
```

vagy

```
30 IF V$ = "I" THEN GOTO 40 : ELSE
   PRINT "Sajnálom!" : END
```

Egyértelműen eldönthető, hogy a 30-as utasítássor első változata a jobb. Azért mutatjuk be mindkettőt, hogy lássuk, a **GOTO** utasításra nincs is szükség, ha a feltételt jól választjuk meg. A **GOTO** kulcsszó használata a **THEN** után nem kötelező. Elhagyva is azt jelenti, hogy a feltétel teljesülésekor a 40-es utasítássoron folytatódjon a program.
Az adatok birtokában már csak egy jelre várunk, hogy indulhasson a hóesés, ezt a **GET** utasítással valósítsuk meg.
Használata és paraméterezése az **INPUT**-hoz hasonló, de itt csak egy

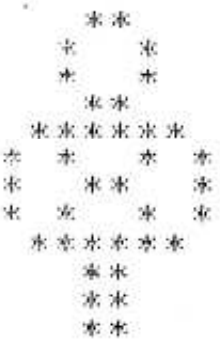
(bármelyik) karaktert olvashatunk be. Programunkban ez az utasítás paraméter nélküli, hiszen nem kívánjuk kiértékelni, hogy melyik karakterrel indították tovább a programot, csak az érdekel bennünket, elindították-e már. A perifériát sem kell megneveznünk, mert ezen utasítás eleve a billentyűzetről várja a karaktereket.

```
  5 !**************************
    *          hóesés         *
    **************************
 10 GRAPHICS 4
 20 INPUT PROMPT "Essen a hó?
    (i/n):":V$
 30 IF V$="i" THEN GOTO 40:ELSE
    PRINT "sajnálom":END
 40 INPUT PROMPT
 "Mennyi hópehely  essen?(db):"
                              :M
 50 CLS:PRINT
"bármely karakterrel indítható"
    :GET:CLS
 60 FOR I=1 TO M
 70 IF I=24*INT(I/24) THEN CLS:
    !*  24-enként letöröljük *
 80 A=RND(24)+1:B=RND(32)+1
 90 PRINT AT A,B:"*"
 95 !*** időzítés ***
100 FOR D=1 TO 100:NEXT D
110 NEXT I
```

Tervezzünk egy babát, amely alapállásban a képernyő közepén áll, és utasítani tudjuk (meghatározott lehetőségek közül kiválasztva), hogy különböző tornagyakorlatokat hajtson végre.
Pontosítsuk a feladatot! A baba megjelenése után szöveges üzenet formájában jelenjen meg a "menü", amit a baba kérdezne, ha beszélni tudna, a lehetséges válaszokkal.

```
mit csináljak?
                    1.alapállás
                    2.terpesz
                    3.magastartás
                    4.vége?
kérem a sorszámot!
```

A választól függően ténylegesen "hajtsa végre" a kijelölt gyakorlatot.
Tervezzük meg a babát minél egyszerűbben (pl. +-ból)!
Adjuk meg a kiírandó karakter helyeit mindhárom testhelyzetre! A baba törzse legyen azonos mindhárom gyakorlatnál, csak a kéz és a láb "mozgatására" korlátozzuk a feladatot!



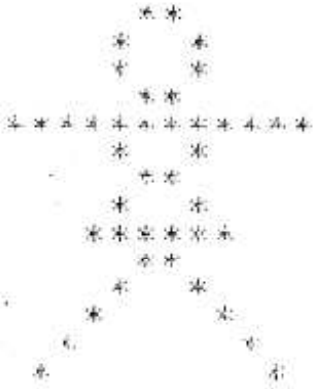
Az általunk megtervezett törzs koordinátapontjai (rendre sor- és oszlopkoordinátával megadva):

(3,16), (3,17), (4,15)...

Alapállásban a kéz és a láb:

(8,13), (8,20), (9,13)...

Terpeszállásban



a kéz és a láb:

(7,11), (7,12), (7,13)...

Magastartásban

a kéz és a láb:

(4,14), (4,19), (5,14)...

Úgy tervezzük meg a babánkat, hogy a 3-astól a 17-es sorig foglalja el a képernyőt, így a menü szövegét még alá tudjuk írni.

A feladat megoldásához szükséges utasításokat már javarészt ismerjük. Mit kezdjünk azonban ezzel a rengeteg adattal? Tárolni kell egy **DATA** utasításban! (Tárolhatnánk külön file-ban is, bár annak a kezelése egy kicsit nehézkesebb. Később majd megismerkedünk az adattárolás ezen módjával is.)
A **DATA**-ban lévő adatokat (numerikus vagy szöveges) a **READ** utasítással lehet beolvasni. Ezen utasítás paraméterei veszik fel rendre értéküket a **DATA**-ban soronkövetkező adatokat.

A beolvasás logikájának a megértéséhez tudnunk kell, hogy az adatterületen van egy mutató, amely jelzi, melyik adat beolvasása történt meg utoljára, s melyik következik. Azaz a beolvasást bármikor abbahagyhatjuk, majd legközelebb onnét folytathatjuk, ahol abbahagytuk.
"*** NO DATA" üzenetet kapunk, ha már az adatok végére értünk és mégis olvasni akarunk.
Az adatmutató átállítását a **RESTORE** utasítással végezhetjük. Paraméter nélkül az adatterület elejére; sorszámot megadva paraméterenként, az adott sorszámon kezdődő **DATA**-ra, azaz a benne lévő első adatra állítja a mutatót.

Az adatokat és a menüt helyezzük el a program elején!

```
 30 REM adattárolás (törzs)
 40 DATA 3, 16, 3, 17, 4, 15 ...
 48 REM alapállás
 50 DATA 8, 13, 8, 20 ...
 58 REM terpesz
 60 DATA 7, 11, 7, 12, 9, 13 ...
 68 REM magastartás
 70 DATA 4, 14, 4, 19, 5, 14 ...
110 REM menü
120 PRINT AT 19,4: "mit csináljak?"
130 PRINT AT 20,19: "1. alapállás":
    PRINT AT 21,19: "2. terpeszállás":
    PRINT AT 22,19: "3. magastartás":
```

```
    PRIN AT 23,19: "VÉGE?"
140 INPUT PROMPT "kérem a sorszámot!" :V
```

Szervezzük a programot úgy, hogy az egyes kívánságokat egy-egy szubrutin hajtsa végre.
Szubrutint akkor érdemes használnunk a feladatainkban, ha ugyanazt a műveletsort (programrészletet) többször kell végrehajtani. Ezt a programrészletet a programunk törzsétől elkülönítve, csak egyszer kell leírni, s a **RETURN** utasítással kell lezárni.
Végrehajtását a **GOSUB** utasítással kezdeményezhetjük. (Különleges ugrató utasítás.)
A szubrutint hívó utasítás hatására az interpreter "megjegyzi", hogy melyik utasításnál szakadt meg a főprogram végrehajtása, és a szubrutin végrehajtása után ott folytatódik, ahol abbamaradt. A főprogramba való visszatérést a **RETURN** szubrutinlezáró utasítás eredményezi. E két utasítás együttes, helyes alkalmazása biztosítja az utasítások végrehajtásának kívánt sorrendjét.
A főprogram és a szubrutin közötti adatátadás a változókon keresztül valósul meg. Assembler ill. már programnyelvű szubrutinok használatakor más lehetőség is van a paraméterátadásra, de arra most nem térünk ki.

A szubrutinok közös része a törzs kirajzolása lesz, kezdjük ezzel!
A törzs adatait 28 ponttal adtuk meg:

```
510 REM: törzs
520 CLS: RESTORE 40
530 FOR B = 1 TO 28
540 READ X,Y
550 PRINT AT X,Y: "*"
560 NEXT B
570 RETURN
```

Az alapállás:

```
210 REM alapállás
220 GOSUB 510
230 RESTORE 50
240 FOR A = 1 TO 14
250 PRINT AT X,Y: "*"
265 NEXT A
270 RETURN
```

A terpeszállás:

```
310 REM terpeszállás
320 GOSUB 510
330 RESTORE 60
340 FOR T = 1 TO 14
350 READ X,Y
360 PRINT AT X,Y: "*"
370 NEXT T
380 RETURN
```

Magastartás:

```
410 REM  magastartás
420 GOSUB 510
430 RESTORE 70
440 FOR M = 1 TO 14
450 READ X,Y
460 PRINT AT X,Y: "*"
470 NEXT M
480 RETURN
```

Készen vannak a szubrutinjaink! Miért célszerű ezt a feladatot szubrutinokból felépíteni?
Könnyen belátható, hogy programunk bizonyos mozdulatelemek sorozatos

ismétlésére épül, sőt az egyes mozdulatok megrajzolásának is van közös eleme. És az már tudjuk, hogy az ismétlődő programrészletek megírásának a legcélszerűbb módja — a BASIC-ben — a szubrutin. Az alapállás rajzolásánál pl. először az 510-es szubrutinnal a törzs közös részét rajzoljuk meg, majd visszatérve a kezeket és lábakat. A **GOSUB** ugyanolyan jellegű utasítás, mint a **GOTO**, direkt ugrást kezdeményez. A visszatéréshez azonban feltétlenül szükséges a RETURN, amit a végrehajtani kívánt feladatsor utolsó utasításaként kell megadni. Ez az utasításpár együtt eredményezi, hogy a feladat — jelen esetben —. a 230-as soron folytatódik. A **RESTORE 50**, az alapálláshoz tartozó adatterület elejére állítja a mutatót. (A RESTORE 40 esetünkben azonos értékű a RESTORE utasítással.)

Mi a teendő a rajzolás után? Újra tegyük fel a kérdést, írjuk ki a menüt!
Egy időzítést kell beiktatnunk — a már megismert elvek szerint —, hogy lássuk az ábrát, ne csak felvillanjon.

A szubrutinokat vezérlő főprogram:

```
150 IF V > 4 OR V < 1 THEN GOTO 600
160 ON V GOSUB 210, 310, 410, 610
170 FOR C = 1 TO 1000
180 NEXT C
190 GOTO 110
600 PRINT "hibás sorszám": GOTO 140
610 CLS
620 END
```

Szeretnénk felhívni a figyelmet a feladat megoldásának a menetére! A részletek kidolgozásához soha ne kezdjünk, amíg az egészet meg nem terveztük! Az összetettebb feladatokat már nem elég szavakban megfogalmazni, hanem folyamatábrát, blokkdiagramot, pszeudokódot ill. struktogramot is kell készíteni. Mindegyik leírásnak létezik egy saját jelölésrendszere, ami a feladatok könnyen áttekinthető és pontos megfogalmazására ad lehetőséget, és ezt használva más is megérti

elképzelésünket. A kezdő programozók dolgát nem akartuk ezzel is nehezíteni, de az összetettebb feladatoknál már könyvünkben is megtalálható a pszeudokódos leírási mód. Természetesen, ki-ki alkothat magának saját jelölésrendszert is, amíg meg nem ismerkedik a szabványossal, s akár ötvözheti is a kettőt, de a programlépések egyértelmű megfogalmazása nélkülözhetetlen. A könyvtárakban, könyvesboltokban rendelkezésre álló szakirodalomból az érdeklődőbbek mélyebb ismereteket szerezhetnek e témából.

```
 10 !**************************
    *         torna           *
    **************************
 38 !**    törzs       **
 20 CLS
 30 !** adattárolás **
 40 DATA  03,16,03,17,04,15,04,
    18,05,15,05,18,06,16,06,17,
    7,14,7,15,7,16,7,17,7,18,7,
    19,8,15,8,18,9,16,9,17,10,
    15,10,18,11,14,11,15,11,16,
    11,17,11,18,11,19,12,16,12,
    17,
 48 !** alapállás **
 50 DATA   8,13,8,20,9,13,9,20,
    10,13,10,20,13,16,13,17,14,
    16,14,17,15,16,15,17,16,16,
    16,17,
 58 !** terpesz **
 60 DATA  7,11,7,12,7,13,7,20,
     7,21, 7,22,13,15,13,18,14,
    14,14,19,15,13,15,20,16,12,
    16,21,
 68 !** magastartás **
 70 DATA 4,14,4,19,5,14,5,19,6,
     14,6,19,13,16,13,17,14,16,
    14,17,15,16,15,17,16,16,16,
    17
110 !** menü **
120 PRINT AT 19,4:
          "mit csinál jak?"
130 PRINT AT 20,19:
                "1.alapállás"
:PRINT AT 21,19:"2.terpesz"
:PRINT AT 22,19:"3.magastartás"
:PRINT AT 23,19:"4.vége?"
140 INPUT PROMPT
         "kérem a sorszámot!";V
150 IF V>4 OR V<1 THEN GOTO 600
160 ON V GOSUB 210,310,410,610
```

```
170 FOR C=1 TO 1000
180 NEXT C
190 GOTO 110
210 !** alapállás **
220 GOSUB 510
230 RESTORE 50
240 FOR A=1 TO 14
250 READ X,Y
260 PRINT AT X,Y:"*"
265 NEXT A
270 RETURN
310 !** terpeszállás **
320 GOSUB 510
330 RESTORE 60
340 FOR T=1TO 14
350 READ X,Y
360 PRINT AT X,Y:"*"
370 NEXT T
380 RETURN
410 !**  magastartás **
420 GOSUB 510
430 RESTORE 70
440 FOR M=1 TO 14
450 READ X,Y
460 PRINT AT X,Y:"*"
470 NEXT M
480 RETURN
510 !**     törzs       **
520 CLS:RESTORE
530 FOR B=1 TO 28
540 READ X,Y
550 PRINT AT X,Y:"*"
560 NEXT B
570 RETURN
600 PRINT "hibás sorszám":GOTO
    140
610 CLS
620 END
```

Utánozzuk a harangzúgást!

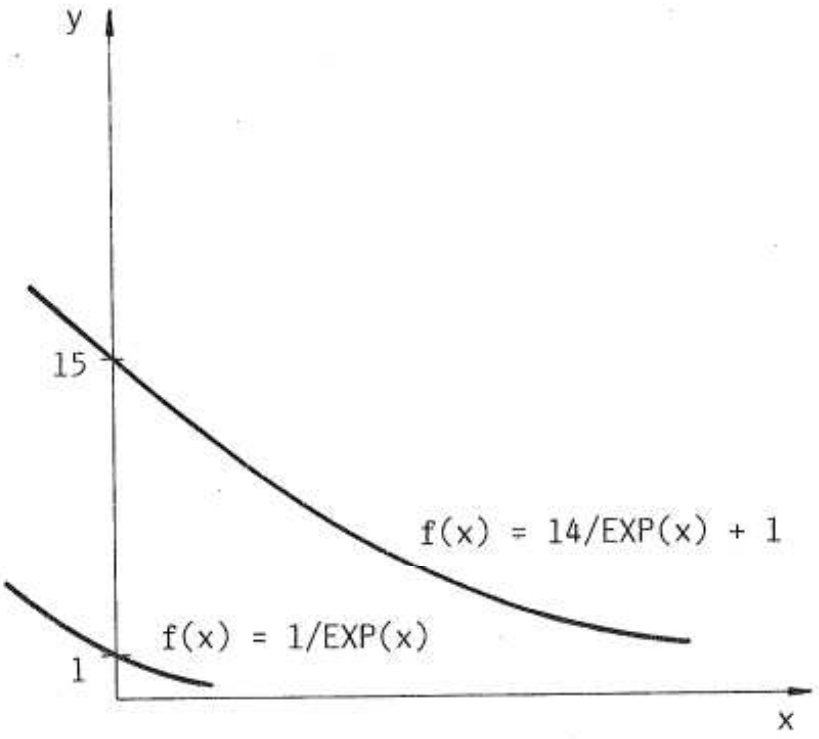
Először ismerkedjünk meg a hangkeltés utasításaival. Lapozzuk fel az (1)-ben a **SOUND**-ot! Ez az utasítás paraméter nélkül egy közép C hangot ad, közepes hangerővel, közepes hosszúságban (2 sec). Figyeljünk a (;) szerepére is! Az utasítás paramétere:

- a **PITCH**, amellyel a hangmagasságot adjuk meg;
- a **VOLUME**, amellyel a hangerőt állíthatjuk be;
- és a **DURATION**, amellyel a hang hosszát változtathatjuk.

A hangmagasság (frekvencia) és a **PITCH** közötti összefüggést egy táblázatból kiolvashatjuk, ezt a (2) végén találjuk meg. A hang hosszúság 0-5 sec között változhat, 20 msec lépésekben (0-255).
A hangerő 15 fokozatban adható meg, a **VOLUME 0** a leghalkabb és a **VOLUME 15** a leghangosabb.

Válasszuk a harang hangját a 2500-as **PITCH** értékűnek.

Milyen legyen a hangerő?
Egy harang fokozatosan halkul el. Ezt úgy tudjuk megoldani, ha a hangerőhöz meghatározunk egy burkológörbét, amelyet lecsengő függvénnyel írunk le. A beépített függvények között találjuk az **EXP(X)**-et. Ha az 1/EXP(X) függvényt megfelelően eltoljuk, és a (0,3) intervallumban vizsgáljuk, a feladatunkhoz éppen megfelel.
Bizonyára a legtöbb Olvasó tudja, hogy az 1/EXP(X) egy olyan függvény, amely az y tengelyt a +1 pontban metszi és az x pozitív értékeinél szigorúan monoton csökkenő, az x tengelyhez közelít, azt a $+\infty$-ben éri el. A függvényt el kell tolnunk, hogy a hangerő a (1,15) tartományba essen. Legyen tehát f(x) = 14/EXP(X) + 1.



A függvény folytonos, a hangerő csak egész szám lehet, így **VOLUME** **INT**(f(x)) adható csak meg. Programunk "beszédesebb" lesz, ha az iménti függvényt V(X)-nek nevezzük; és hogy ezen formális változóval hivatkozhassunk rá, definiálni kell. A formális változó és a függvény összerendelése egy értékadással, a DEF segítségével történik.

```
20 DEF FNV(X) = INT(14/EXP(X) + 1
```

A harangzúgást utánozó programunk:

```
  5 !*************************
    *          harang        *
    *************************
 10 CLS
 20 DEF FNV(X)=INT(14/EXP(X)+1)
    :       !** hangerö **
 30 P=2500:D=8: !**p-magasság**
                  **d-hossz    **
 35 FOR H=1 TO 10
 40 FOR E=0 TO 3 STEP 0.2
 50 SOUND;PITCH P,VOLUME FNV(E)
    ,DURATION D
 60 NEXT E,H
 70 END
```

Szebben szól a harang, ha kéthangú. Hallgassuk meg így is!

```
 5 !*****************************
   *          harang-2           *
   *****************************
10 CLS
20 DEF FNV(X)=INT(14/EXP(X)+1)
   :      !** hangerő **
30 RESTORE:FOR S=0 TO 1
40 READ P:!** hang váltás **
50 FOR E=0 TO 3 STEP 0.2
60 SOUND;PITCH P,VOLUME FNV(E)
   ,DURATION 8
70 NEXT E,S
75 GOTO 30
80 DATA 3155,2977
```

Ismerkedjünk tovább a hangokkal! Szólaltassunk meg egy közkedvelt gyerekdalt, a "BOCI BOCI TARKÁT"!
A dal:

Nem lesz nehéz dolgunk, mindössze 2 hanghosszúság van benne.
A tá legyen 40-es **DURATION** értékű, a ti pedig 20-as. Lapozzuk fel a hangjegy táblázatot és a 3349-3723-as **PITCH** közé eső oktávban azonosítsuk a hangokat. A dalban a hangerőt nem változtatjuk, csak a hangmagasságot és a hanghosszúságot. Tekintsük összetartozó értékpároknak a 2 változó adatot, így tároljuk egy DATA utasításban. Adjunk lehetőséget a dal többszöri lejátszására.

```
  1 !**************************
    * "boci,boci tarka..."   *
    **************************
  5 CLS
 10 INPUT PROMPT
  "hányszor szóljon a dal?:":K
 15 INPUT PROMPT
  "milyen hangerövel?(0-15):":V
 20 H=24:!** hagok száma **
 30 FOR S=1 TO K
 40 RESTORE
 50 FOR I=1 TO H:READ P,D
 60 SOUND;VOLUME V,DURATION D,
    PITCH P
 70 NEXT I
 75 !** szünet következik **
 80 SOUND;VOLUME 0
 90 NEXT S:END
 95 !** dallam **
100 DATA  3349,20,3503,20,3349,
    20,3503,20,3598,40,3598,40,
110 DATA  3349,20,3503,20,3349,
    20,3503,20,3598,40,3598,40,
120 DATA  3723,20,3701,20,3652,
    20,3598,20,3537,40,3652,40,
130 DATA  3598,20,3537,20,3503,
    20,3431,20,3349,40,3349,40,
```

A dal ismételt megszólaltatása előtt szünet van. Ez úgy is megoldható, hogy az utolsó hang a 4095-ös **PITCH** értékű lesz, de akkor ezt a H értékének megadásakor figyelembe kell venni! Ügyeljünk a pontosvesszőre is! Ezzel biztosítjuk, hogy a hangok olyan hosszan szóljanak, ahogyan definiáltuk őket. Próbáljuk ki a feladatot pontosvessző nélkül is!

Általánosítsuk a programunkat, más dal eljátszására is tegyük alkalmassá! Ennek egy lehetséges megoldását beszéljük meg a következő fejezetben.

Adatfile létrehozása

Szervezzük az előbbi feladatunkat úgy, hogy a dallamot egy adatfile-ból vegye.

Az adatfile-ok kezelése egyszerű dolog, de az alapszabályokat be kell tartani. A legfontosabb, hogy csak akkor tudunk hozzáférni az adatfile-hoz, ha először az **OPEN**-nel megnyitjuk, és a művelet végeztével a **CLOSE**-zal lezárjuk azt.

A file-ba írás **OUTPUT** irányt jelent, míg az olvasás **INPUT** irányt. A file adatai (rekordok), ciklus utasítással kerülnek kivitelre, ill. beolvasásra. (A TVC BASIC a rekordok elválasztásáról maga gondoskodik.) A két művelet — a kiírás és a beolvasás — sebességének azonosnak kell lenni, mert különben adatokat veszítünk.

Hozzuk létre az előbbi dal adatfile-ját feldolgozó programokat. Először írjuk meg az adatfile-t generáló programot, ami a billentyűzeten begépelt **PITCH** és **DURATION** értékeket egy általunk meghatározott nevű adatfile-ban tárolja, kazettán.

Módszerünk az legyen, hogy egy-egy vektorba összegyűjtjük a billentyűzeten begépelt adatokat, s a szalagot csak akkor indítjuk, ha már az adataink együtt vannak.

Mit tegyünk, ha valamelyik adatot hibásan gépeltük be?
Futtassuk le a feladatot a magnó bekapcsolása nélkül, és a vektor megfelelő elemének értékét módosítsuk közvetlenül. Pl. ha az 5. hangjegy helyett 3098-as írtunk, akkor parancsként írjuk be: P(5) = 3598, és ezzel el is végeztük a javítást. Javítás után a program futtatását onnan kezdjük, ahol már befejeződött az adatgyűjtés. Az esetünkben ez: RUN 70

A feladat kérje a magnó bekapcsolását, és egy visszajelzőkarakter megadásáig várakozzon. Mi a "*"-t választottuk jelzőkarakternek. Az ada-

tok kiküldése csak ennek beírása után indul.
"Dallam" néven megnyitjuk a file-t **OUTPUT**-ra, kivisszük az adatokat (a vektor elemeit) **PRINT**-tel, majd lezárjuk a file-t. A **PRINT** alapértelmezésben a képernyőre ír, tehát a periféria megjelölése most nem hagyható el! (PRINT#5:)

```
  5 !*************************
    *   adatfile létrehozás  *
    *************************
              "adgen"
 10 CLS
 25 H=24:!** hangjegyszám **
 30 DIM P(H),D(H)
 40 FOR I=1 TO H
 50 INPUT PROMPT
    "pitch,duration ":P(I),D(I)
 60 NEXT I
 70 OPEN OUTPUT "dallam"
 75 PRINT   "Inditsd a magnót,
   ha kész nyomd meg a '+'-t!"
 80 GET A$:IF A$<> "+" THEN
           GOTO 80
 90 FOR I=1 TO H
100 PRINT #5: P(I)
105 NEXT I
110 FOR I=1 TO H
115 PRINT #5: D(I)
120 NEXT I
125 CLOSE OUTPUT
130 END
```

Vigyázzunk, az adatfile létrehozásának 3 fázisa van, ne kapcsoljuk ki a magnót az ok megjelenése előtt!

Következik az iménti adatfile feldolgozása. Tudjuk, a "Dallam" nevű file-ban rekordonként találjuk a hangjegyeket és a hanghosszakat. (A feldolgozóprogramban adjuk meg a kívánt hangerőt, hacsak nem elégszünk meg az alapértékkel.)

```
  1 !**************************
    *  adatfile feldolgozás  *
    **************************
              "adin"
  5 CLS
 10 INPUT PROMPT
    "hányszor ...? ":K
 20 INPUT PROMPT
    "hangerö      ? ":V
 25 H=24:!** hangok száma **
 30 DIM P(H),D(H)
 40 OPEN INPUT "dallam":
    !** adatfile **
 45 FOR I=1 TO H:!** hangok **
 50 INPUT#5: P(I)
 55 NEXT I
 60 FOR I=1 TO H:!** dallam **
 65 INPUT#5: D(I)
 70 NEXT I
 80 CLOSE
 90 FOR S=1 TO K:!** hányszor**
 95 FOR I=1 TO H
100 SOUND;VOLUME V,
    DURATION D(I),PITCH P(I)
110 NEXT I
120 SOUND;VOLUME 0
130 NEXT S
140 END
```

Feladatunk egyik szépséghibája, hogy H értékét nem az adatfile-ból vettük. Módosítsuk tehát úgy, hogy a P(0) tartalmazza a hangjegyek számát.

Kell módosítani a DIM utasítást? Miért (nem)?

Az **INPUT** utasításnak a billentyűzet az alapértelmezett perifériája, tehát a (#5:) megadása kötelező. Az **INPUT**-ra megnyitott "Dallam" nevű adatfile-ból beolvassuk rendre az összes adatot a P(I), D(I) vektorokba. Az összes adat beolvasása után — a korábban megismert módon — feldolgozzuk azokat, azaz megszólaltatjuk a dalt.

Rajzoljunk kört. Hol is kezdjük? Nézzük meg a beépített függvényeket! Nem találunk olyat, amely kört rajzol. Mely adatok határoznak meg egy kört? Egy kör jellemzője a középpontja és a sugara.

Próbálkozzunk először úgy, hogy egy szabályos sokszöget rajzolunk. Ugye az nyilvánvaló, hogy minél nagyobb a szögek száma, annál inkább hasonlít az ábránk a körhöz.
Azt már láttuk, hogy 2 adott pontot hogyan kell összekötni (l. szinusz).
Határozzuk meg az iménti körbe berajzolható sokszög csúcsait és kössük őket össze. Próbálkozzunk 15, 20, 25 szöggel!

```
 1 !*****************************
    *  kör1 "sokszögesitve"  *
    *****************************
 5 CLS
10 !**  Meg kell adni a közép
   pont koordinátáit és a suga
   rat. Nincs adat ellenörzés!
20 INPUT PROMPT "x1,y1,r:":X1,
   Y1,R
30 CLS:A=X1+R:B=Y1:
   !** kiindulási pont **
40 FOR T=0 TO 2*PI+PI/10 STEP
   PI/10
50 C=X1+R*COS(T):D=Y1+R*SIN(T)
60 PLOT A,B;C,D
70 A=C:B=D
80 NEXT T
```

A témához kapcsolódó matematikai alapismeretek az Obádovics: Matematika koordinátageometriával foglalkozó fejezeteiben megtalálhatók. Ábránkat finomíthatjuk, ha a 60-as utasítássort egy egyenesrajzoló szubrutin hívására cseréljük ki.

Az egyenesrajzoló szubrutin elve az, hogy az (A,B) és (C,D) pontokat összekötő szakaszt további kis szakaszokból rajzoljuk meg. A pontokat összekötő szakasz nagyobb abszolút értékű koordinátájából (E) meg-

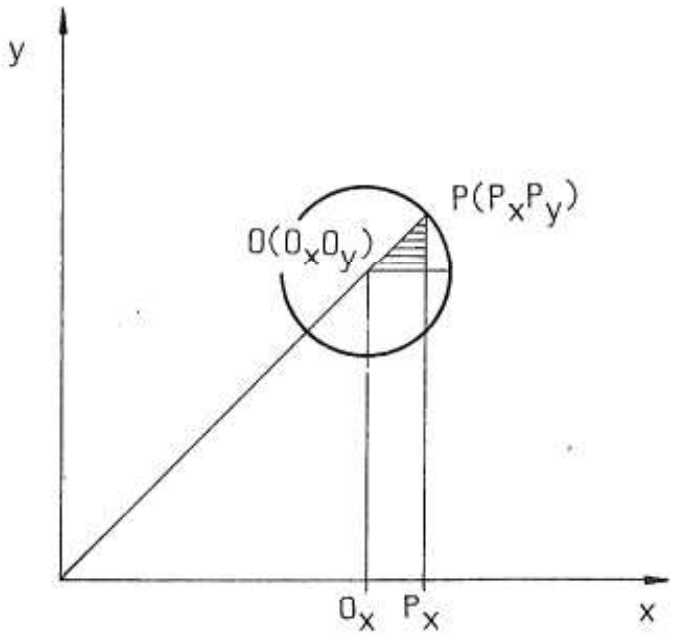
határozunk egy nagyítási arányt, s ennek alapján közelítünk az új ponthoz. Így ugyanis legalább az egyik koordináta növekménye egységnyi lesz, és az egyenesünk nem lesz olyan "tördelt". Egyébként, a megrajzolt vonal helyenként túl vastag lesz; egymás mellett 2-3 négyzetet is befeketít.
Gondoljunk vissza a szinuszgörbénkre! Új ismereteink birtokában hogyan oldanánk meg a feladatot?

Egészítsük ki programunkat az egyenesrajzoló szubrutinnal:

```
 60 GOSUB 200
200 REM egyenesrajzoló szubrutin
210 F = C : G = D-B
220 E = ABS(F)
230 IF ABS(C) >= ABS(F) THEN
    E = ABS(G) + (G*0)
240 FIR X = 0 TO 1 STEP 1/E
250 PLOT A + F*X, B + G*X
260 NEXT X
270 RETURN
```

Próbáljuk ki a feladatot mindkét változatban. A szubrutin használata lassítja a rajzolást. Használatát érdemes mérlegelni.

Próbáljuk ki a feladatot az **END** elhagyásával is. Így megérthetjük, hogy az **END** a program logikai végét jelzi, leállítja a továbbfutását.
Próbáljuk meg a körrajzolást úgy, hogy definiáljunk egy függvényt! A kört leíró egyenlet legyen a kör paraméteres egyenlete. Az adataink: R = 200, O = O(480,512)



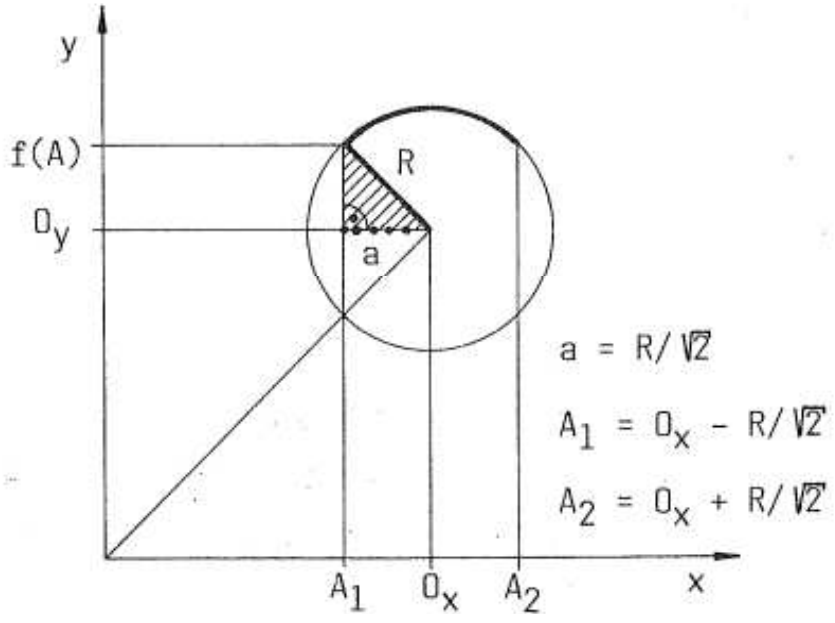
Aki ismeri a kör paraméteres egyenletét, annak könnyű a dolga; aki nem, az próbálja meg az ábrán bevonalkázott derékszögű háromszögre a Pitagorasz-tételt felírni! (Avagy, egyenletünket összevetni az adatainkkal és ennek mintájára más adatokkal dolgozni.)

```
DEF FNK(x) = 512 + SQR (200^2 - (x-480)^2)
DEF FNK(x) = 512 - SQR (200^2 - (x-480)^2)
```

Programunkban tehát, rendre kiszámítjuk a függvény összetartozó értékpárjait, és azokat összekötjük.

```
 1 !*****************************
    *   kör2   "Függvényként"   *
    *****************************
10 CLS
15 !** felsö félkör **
20 DEF   FNKF(X)=512+SQR(200^2-
   (X-480)^2)
25 !** alsó félkör   **
30 DEF   FNKA(X)=512-SQR(200^2-
   (X-480)^2)
40 FOR A=280 TO 680
50 Y=FNKF(A):Z=FNKA(A)
60 PLOT A,Y:PLOT A,Z
70 NEXT A
80 !** A kiirást gyorsithatjuk
   (step 4) beirásával.**
90 END
```

Láthatjuk, hogy a pontok sűrűsége csak középen megfelelő. Módosítsuk a programot úgy, hogy a görbe egy ilyen szakaszát rajzoljuk csak meg, majd azt elforgatjuk a "ritka pontú helyekre".
Válasszuk ki a felső félkörből a középső negyedkört és azt forgassuk el a középpont körül 90 fokkal egymás után háromszor.



Az ív végpontjainak x koordinátái:

480 - 200/SQR(2) ill. 480 + 200/SQR(2)

Hogyan kell egy pontot elforgatni?
A forgatást végezhetjük pozitív és negatív irányba (az óramutató járásával ellentétesen és megegyezően).
Szükségünk van egy forgatási középpontra (esetünkben ez a kör középpontja), és a továbbviendő pont koordinátájára: (A,B). Derékszögű elforgatás esetében az új pont koordinátái.

$$A' = 480 + (B - 512)$$
$$B' = 512 - (A - 480)$$

ill. a másik irányba forgatva:

$$A' = 480 - (B - 512)$$
$$B' = 512 + (A - 480)$$

Az összefüggés természetesen bonyolultabb, ha az elforgatás nem 90 fokkal történik. E témához Olvasóink figyelmébe ajánljuk az "Etüdök személyi számítógépre" című könyvet (Votisky, Gondolat 1984.)

Próbáljuk meg a körrajzolást mindkét irányú elforgatással.

```
  5 !*****************************
    *     körrajzoló program    *
    *****************************
  6 ! o(480,512):r=200
  7 !Elv:negyedkört forgat el a
       középpont körül
  8 !A stop után töröljük a kép
    ernyöt és"continue"-val in
    ditsuk tovább!
  9 !Látható, ugyanazt az ered
    ményt kapjuk akkor is,ha
    az alsó félkörböl indulunk
    ki.
 10 CLS
 20 DEF    FNKF(X)=SQR(200^2-
    (X-480)^2)+512:!*felsökör*
 30 DEF    FNKA(X)=-(SQR(200^2-
    (X-480)^2))+512:!*alsókör*
 32 GOSUB 40
 33 STOP
 34 B=FNKA(A)
 36 GOSUB 40
 38 END
 40 GOSUB 90
 50 STOP
 52 C=480+(B-512):D=512-(A-480)
 54 E=480-(B-512):F=512+(A-480)
 56 G=480+(D-512):H=512-(C-480)
 60 B=FNKA(A)
 70 GOSUB 90
 80 END
 90 FOR  A=480-200/SQR(2)  TO
    480+200/SQR(2)
100 B=FNKF(A)
110 C=480+(B-512):D=512-(A-480)
120 E=480-(B-512):F=512+(A-480)
```

```
130 G=480+(D-512):H=512-(C-480)
140 PLOT A,B:PLOT C,D:PLOT E,F:
    PLOT G,H
150 NEXT A
160 RETURN
```

Programunkban a függvény definiálása után megrajzoljuk a negyedkörök pontjait az éppen aktuális pont elforgatásával. (A segédváltozókat az áttekinthetőség kedvéért vezettük be.) Majd megismételjük a feladatot a másik irányú elforgatással. A mintaprogram megértéséhez pontosan kell ismernünk a **GOSUB — RETURN** utasításpár végrehajtási mechanizmusát. Kövessük ezt nyomon még egyszer a feladat kapcsán!

Természetesen a feladat általánosítása a következő teendőnk, hogy bármely középpont és sugár megadása esetén működőképes legyen a programunk. Végezzük ezt el egyénileg! Ne felejtkezzünk meg az adatok ellenőrzéséről se. (Ráfér-e az adott kör a képernyőre?)
Oldjuk meg a feladatot úgy is, hogy a képernyőn kívülre eső pontoknál ne kapjunk hibajelzést, hanem maradjunk a szélső pontban, és csak a képernyőre eső ívet rajzoljuk meg.

Rajzoljunk spirált az első körrajzoló program felhasználásával. A rajzolás elve azonos a körrel, csak a sugár növekszik a szögelfordulás arányában, a körülfordulás szöge is a 2*PI-nek lesz a többszöröse.
A körrajzoló programba írandó módosítások:

```
40 FOR T = PI/10 TO 20*PI STEP PI/10
50 C = 480 + (3*T)*COS(T) : D = 512 + (3*T)*SIN(T)
```

A teljes program:

```
 1 !*****************************
   *         spirál              *
   *****************************
 5 GRAPHICS 4
10 !** 10-szer tekeredik; **
   t:szögelfordulás paramétere
20 INPUT PROMPT
"x1,y1,r(nincs ellenörzés):  "
                    :X1,Y1,R
30 A=X1:B=Y1
40 FOR T=PI/10 TO 20*PI  STEP
   PI/10
50 C=X1+(3*T)*COS(T):
   D=Y1+(3*T)*SIN(T)
60 PLOT A,B;C,D
70 A=C:B=D
80 NEXT T
```

Készítsünk feliratot ehhez a tekeredő spirálhoz úgy, hogy az ábra alá írjuk képújságszerűen:

"Tekeredik a kígyó, rétes akar lenni."

A kiírandó szöveget tároljuk el egy **DATA**-ban úgy, hogy a végét egy jelzőkarakter mutassa. (Gondoljunk vissza a dalunkra! Milyen kényelmetlen volt, hogy a hangjegyek számát tudnunk kellett.)
A jelzőkaraktert úgy kell megválasztani, hogy ne forduljon elő a szövegben. Válasszuk pl. a számkeresztet (#), a heshmarkot. A szöveg

karaktereit vesszővel elválasztva kell beírnunk, figyelve arra, hogy a szóköz és a vessző is a szöveg része. Az idézőjelet csak a vesszőnél és szóköznél kötelező használni. Nézzünk utána pontosan a szabálynak!

Az adatfeldolgozást a korábban megismert módszerrel végezzük. A kiíráshoz a szöveg egy tömbben legyen, így a karakterek elérését a tömb indexelésével valósítsuk meg.
Minden adat beolvasásakor meg kell vizsgálnunk, hogy végkarakter érkezett-e. (Összehasonlítást kell végezni.) Erre több lehetőségünk is kínálkozik. Nézzük meg, mi a számkereszt ASCII kódja. Nem baj, ha nincs kódtábla a kezünk ügyében, legalább kipróbáljuk az **ORD(X$)** beépített függvényt. Írjuk be:

```
PRINT ORD("#")
```

az eredmény 35 lesz.
Ennek ismeretében az összehasonlítást a **CHR$(X)** függvény felhasználásával is elvégezhetjük, ahol az X egy karakter ASCII kódjának megfelelő numerikus érték.

Az eddigiekben csak e két függvény felhasználási lehetőségére szerettünk volna rámutatni, az összehasonlítást elvégezhetjük közvetlenül is.

```
60 IF X$(I) = "#" THEN GOTO ...
```

vagy

```
60 IF X$(I) = CHR$(35) THEN GOTO ...
```

Határozzuk meg a szöveg kiírásának paramétereit:

N – az a sorba írható karakterek száma
H – a szöveg hossza
G – a kiírások megengedett száma (ennyiszer fut végig a szöveg)
K – az aktuális kiírás sorszáma

I – az egy kiíráson belüli karaktersorszám

Z – soron belüli karakterhelyzet

B – a szöveg melyik karakterénél tartunk

A paramétereink közötti kapcsolatot a következő táblázattal állapítjuk meg.

Legyen: H = 3; N = 32 ($H \leqq N$)

K	J	B	Z
1		1	32 = 33-1
2	1. 2.	2 1	31 = 33-2 32 = 33-1
3	1. 2. 3.	3 2 1	30 = 33-3 31 = 33-2 32 = 33-1
4	1. 2. 3. 4.	4 3 2 1	29 = 33-4 30 = 33-3 31 = 33-2 32 = 33-1
.	.	.	.
.	.	.	.
.	.	.	.
33	1. 2. 3. . . .	33 32 31 . . .	0 = 33-33 1 = 33-32 2 = 33-31 . . .
34	1. 2. 3. . . .	34 33 32 . . .	-1 = 33-34 0 = 33-33 1 = 33-32 . . .

A táblázat alapján:

$$Z = (N + 1) - B$$
$$\underline{B = (K + 1) - I}$$
$$Z = N - K + I$$

Hányszor írjuk ki a szöveget? (N + H)-szor.
Ügyelnünk kell az I és Z végértékeire. Az I sohasem lehet nagyobb H-nál, és a Z csak pozitív lehet. Mi történjen a Z negatív értékeinél? A szöveg bal oldalon "kilépő" karakterei jelenjenek meg újra a jobb oldalon.
Ekkor Z = N - ABS(Z) lesz.

A programlista:

```
105 !*****************************
    *          képújság         *
    *****************************
110 GRAPHICS 2
115 M=64:!** szöveg megengedett
        hossza(max.egy sor!)**
120 DATA    "T","e","k","e","r",
    "e",d,i,k," ",a," ",k,i,g,
     y,ó,",",","  ",r,é,t,e,s," ",
     a,k,a,r,"  ",l,e,n,n,i,.,
     "  ",#,
130 DIM X$(M)*1
135 RESTORE
140 FOR I=1 TO M
150 READ X$(I)
160 IF X$(I)=CHR$(35) THEN GOTO
    180
170 NEXT I
180 H=I-1:
    !** h=a szöveg tényleges
                  hossza **
190 PRINT " Inditsd!":
    GET
200 J=22:N=64:G=N+H :
    !** j=sorszám
    n=egy sorba irható karakter
    g=kiirások száma  **
```

```
205 CLS
210 FOR K=1 TO G
220 FOR I=1 TO K
225 Z=N-K+I:!**oszloppozicio**
230 IF I>H THEN 260
240 IF Z<=0  THEN Z=N-ABS(Z)
245 PRINT AT J,Z:X$(I)
250 NEXT I
260 NEXT K
270 PRINT  "ismétlés ?(i\n):"
275 GET A$:IF A$="I" THEN 135
280 GRAPHICS 4:END
```

E feladat bonyolultabb változatával még a könyv második részében foglalkozunk.

Szerkesszük össze a két programot!
Törölnünk kell a 90-es, 205-ös, 270-es és a 275-ös programsorokat.
A megjelenítést "szépítsük" azzal, hogy az "INDÍTSD!" feliratot töröljük a képernyőről, s így az ábra változatlan marad.

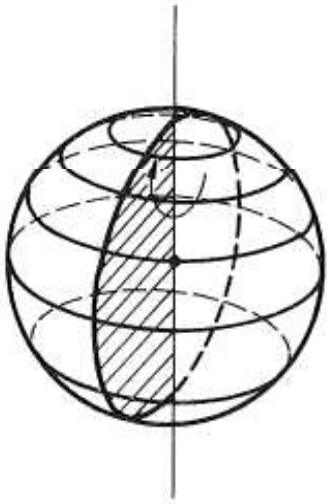
Végezzük el az általánosítást (a paramétereket kérjük a billentyűzetről). Gondolkozzunk el a H>N eseten is! Aki egyedül nem boldogul, lapozza fel könyvünk második részében az összetettebb megoldást.

Mindazok, akik eddig sorról-sorra végiggondolták az előző fejezetek feladatait, büszkén kihúzhatják magukat és megelégedéssel nyugtázhatják, a javán túl vagyunk. Sok rejtelmét megismertük a **TVC BASIC**-nek, most már — szinte — csak a begyakorlás van hátra.
Ne feledjük, sok feladatot meg kell még oldanunk, hogy azt mondhassuk, megtanultuk.

Kikapcsolódásként most rajzoljunk valami szépet!
Rajzoljuk meg pl. egy pitypang gyönyörű bóbitáját! Ehhez persze némi térszemlélet is kell, hiszen most egy gömböt kell képeznünk a képernyő síkjába.
Ezt tekintjük (x,y) síknak.
A megoldás nyitja, a már megtanult pontelforgatás.

Képzeljük csak el, milyen ez a bóbita! Egy gömbalakzat közepéből indul ki a "szár", ami sugárirányba szétágazik, s mindegyik végén van egy másik kis gömb (nevezzük ezt "virágnak"), aminek a felépítése hasonló, csak a sugarak hossza lényegesen rövidebb.

Programunkban először egy szárat, majd a hozzá tartozó teljes "virágot" rajzoljuk meg.
A képernyő síkjával zárjon be E szöget az a sík (II), amelyben elhelyezkedő szárakat (gömbmetszetet) le kell képeznünk a képernyőre.



Az így adódó koordináták:

$$IX = M*\cos(E*PR)*\cos(N*PR)$$
$$IY = M*\cos(e*PR)+\sin(N*PR)$$

M: a normálási paraméter
PR = PI/180, így a szög fokban lesz,
H = M*cos(E*PR), a szár képernyősíkba eső vetülete.

A "virág" x és y koordinátája két komponensből áll (a virágnak nem képezzük a képernyősíkra eső vetületét).

A szirom koordinátái:

$$M*\cos(I*PR) = MX_S$$
$$M+\sin(I+PR) = MY_S$$
$$MX = MX_S + M*\cos(N*PR)*\cos(E*PR)$$
$$MY = MY_S + M*\sin(N*PR)*\cos(E*PR)$$

Minél nagyobb az E szög, annál rövidebb lesz a vetület. Vezessünk be egy R eltolási értéket, hogy a szár rajzolását ne mindig ugyanabból a pozícióból kezdjük.

Az érdeklődőknek figyelmébe ajánljuk Obadovics: Matematika, A gömbi trigonometria alapfogalmai c. fejezetét.

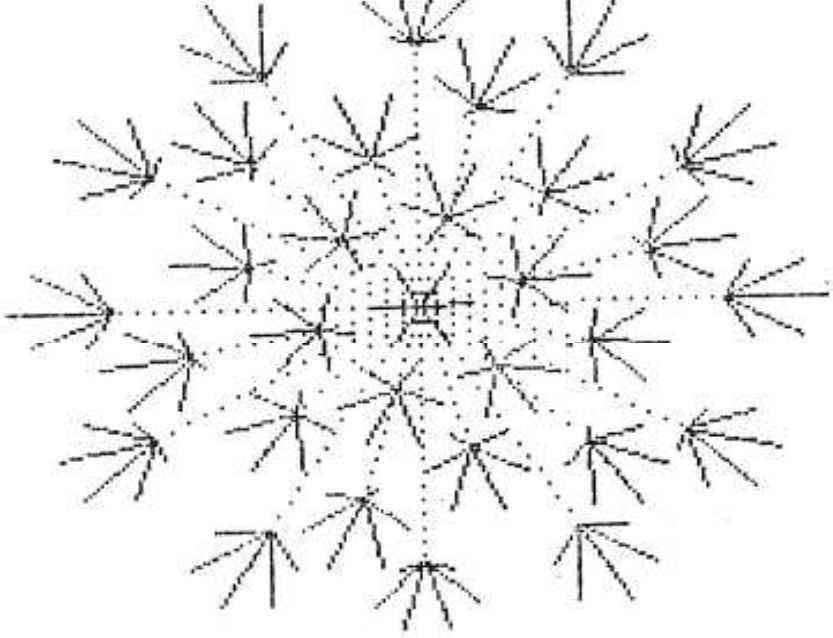
S most, az összefüggések és eddigi ismereteink alapján állítsuk össze a programunkat:

```
 1 !**************************
   *        pitypang        *
   **************************
 5 PR=PI/180
10 GRAPHICS 4
20 SET PALETTE 0,80,68,65
30 SET PAPER 3
35 SET STYLE 5
40 CLS
45 PX=520:PY=500
```

```
 50 L=30:R=0:E=5:GOSUB 460
 70 L=60:R=15:E=40:GOSUB 460
 80 L=60:R=45:E=55:GOSUB 460
 82 L=60:R=15:E=70:GOSUB 460
 85 L=395:R=0:E=90:GOSUB 460
 90 END
459 !*****************************
    *        rajzolás           *
    *****************************

460 FOR N=0 TO 359 STEP L
465 N=N+R:M=300
470 IX=M*COS(E*PR)*COS(N*PR)
480 IY=M*COS(E*PR)*SIN(N*PR)
490 PLOT PX,PY;PX+IX,PY+IY
491 SET STYLE 1
492 GOSUB 810
494 SET STYLE 5
495 N=N-R
500 NEXT N
510 RETURN
809 !*****************************
    *         "virág"           *
    *****************************
810 FOR I=1 TO 359 STEP 60
825 M=50
830 MXS=M*COS(I*PR)
840 MYS=M*SIN(I*PR)
842 MX=MXS+M*COS(N*PR)*
                      COS(E*PR)
845 MY=MYS+M*SIN(N*PR)*
                      COS(E*PR)
850 PLOT PX+IX,PY+IY;PX+MX+IX,
                      PY+MY+IY
860 NEXT I
880 RETURN
```

Próbálgassuk, milyen ábrát rajzol a program különböző E, R paraméterértékekkel. A mi tetszésünket a programban rögzített paraméterekkel készült ábra nyerte meg.



Játékos kitérőnket tovább folytatjuk. Próbálkozzunk ismét a zenével.

Rajzoljunk egy "zongorát", amely 8 billentyűből áll. Adjunk lehetőséget a teljes skála megszólaltatására, azaz definiáljunk oktávváltó karaktereket is. Ezeknek jól megjegyezhetőeknek és könnyen használhatóaknak kell lenniük. Válasszuk sorban a billentyűzetről az Y, C, V, B karaktereket. A kiválasztott oktávon belül rendeljük a hangokhoz — sorrendjüknek megfelelően — számbillentyűket. (A félhangokat megrajzoljuk ugyan, de a megszólaltatásukra most ne adjunk lehetőséget.) Tegyük érdekesebbé az ábránkat úgy, hogy definiáljunk egy "figurát" (karaktert), amely mindig a megszólaló hang billentyűjére ugrik.
A karakterdefiniálás technikáját a (2) 4.4.1 szakaszában részletesen megtalálhatjuk; lapozzuk fel!
A képernyőn, egy karakter 10 képpontsorból és 8 oszlopból áll, tehát egy 8x10-es téglalapban kell megterveznünk. Ne feledjük, hogy az új hang megszólaltatásakor a régi helyéről törölni kell a "figurát". Ezért a kettőt együtt kell megtervezni. Legyen a törlőkarakter egy jobb oldali függőleges vonal, a "figura" pedig ez a függőleges vonal és egy "zászlós" hangjegy, azaz egy nyolcad hang.
A törlőkaraktert definiáló utasítás:

```
SET CHARACTER 172,1,1,1,1,1,1,1,1 (= ALT/<)
```

A "figurá"-t definiáló utasítás:

```
SET CHARACTER 180,17,17,17,17,17,17,113,241,241,97 (= ALT/D)
```

Tervezzük meg a képernyőt is. A billentyűzetet piros háttérrel rajzoljuk meg a képernyő közepére. Dolgozzunk 16 színű üzemmódban, ahol az egy karakterre jutó vízszintes logikai koordináta: 1024/16 = 64 képpont, a függőlegesen egy sorra jutó: 960/24 = 40 képpont.
Az ábrát a 8. sortól a 16-ig és a 4. karakterhelyzettől a 12-ig

helyezzük el. Így a billentyűzetet befoglaló téglalap csúcspontjának koordinátái (az eddigi ismereteink alapján):

1. 320, 256
2. 640, 256
3. 640, 768
4. 320, 768

Ha ebbe a téglalapba berajzoljuk az imént definiált karaktert, azt tapasztaljuk, nem esnek egybe a határaik. A szükséges korrekció: a képernyő középvonalához viszonyított karakterpozíciót ki kell vonni az előbbi határokból.
Így adódik:

1. 320, 252
2. 640, 252
3. 640, 764
4. 320, 764

Írjuk meg először azt a szubrutint, amely megrajzolja az imént megtervezett billentyűzetet.

```
4000 GRAPHICS 16:
     !** zongora rajzolás
4003 SET MODE 0
4005 SET PAPER 10:SET BORDER 10
4008 CLS
4010 SET INK 15
4020 PLOT  ,256,320;256,640;
     764,640;764,320;256,320
4025 PLOT,290,330,PAINT
4028 SET INK 0
4030 FOR I=5 TO 12
4040 L=I*64-4
4050 PLOT,L,399;L,320
4060 NEXT I
4065 SET INK 0
4070 FOR I=5 TO 12
4075 IF I=7 OR I=11 THEN
       L=I*64-4:PLOT,L,640;L,399:
       NEXT I
4080 L=I*64-20
```

```
4085 IF I=12 THEN PLOT,L,400;
     L,640;L+20,640;L+20,400;
     L,400:PLOT,L+10,420,PAINT:
     GOTO 4110
4090 PLOT,L,400;L,640;L+40,640;
     L+40,400;L,400
4095 NEXT I
4110 FOR I=5 TO 11
4120 L=I*64-4
4130 PLOT,L,420,PAINT
4140 NEXT I
4900 SET PAPER 15:SET MODE 0
4910 SET INK 0
4920 SET CHARACTER 180,17,25,21,
     17,17,17,113,241,241,97
4930 SET CHARACTER 172,1,1,1,1,
     1,1,1,1,1,1
4935 T1=1
4940 RETURN
5200 !**hangjegy törlés,kiiratás
5205 PRINT AT 16,T1+4:CHR$(172)
5210 PRINT AT 16,I+4:CHR$(180)
5220 T1=I
5230 RETURN
```

Ezek után még tisztáznunk kell, hogyan azonosítjuk a leütött billentyűt és a hangmagasságot.

A korábban megbeszéltek szerint, a leütött számjegyet mint indexet használjuk fel; amelyet az Y, X, C, V, B leütésétől függően módosítunk. A hangjegy táblázat (PITCH) értékeit eltároljuk egy **DATA**-ban, ahonnét a program első fázisában egy P(I) vektorba beolvassuk. A vektor indexe:

$$I = VAL(X\$) + IS$$

ahol a **VAL(X$)** a lenyomott számjegy ASCII kódjából állítja elő a numerikus értékét:

IS = 0, ha X$ = Y;
IS = 8, ha X$ = X;
.
.
.
IS = 32, ha X$ = B;

A karakter beolvasását végezzük az **INKEY$**-ral. Gondolkozzunk el, mi a különbség az **INKEY$** és a **GET** között! (Az (1)-ben megtalálhatjuk.)

```
  1 !*****************************
    *        zongora              *
    *****************************
 10 DIM P(40):
    !**          skála             **
 20 GOSUB 2000:!** Pitch töltés
 25 GOSUB 4000:!**   Taszt irás
 30 X$=INKEY$:IF X$="" THEN 30:
    !**      karakterre vár       **
 35 !**     oktávváltás           **
 40 IF X$="y" THEN IS= 0:GOTO30
 50 IF X$="x" THEN IS= 8:GOTO30
 60 IF X$="c" THEN IS=16:GOTO30
 70 IF X$="v" THEN IS=24:GOTO30
 80 IF X$="b" THEN IS=32:GOTO30
 90 I=VAL(X$)
 95 IF I<1 OR I>8 THEN  30
100 GOSUB 5200
110 I=I+IS:Q=P(I)
120 SOUND; PITCH Q ,DURATION 7,
    VOLUME 8
130 GOTO 30
2000 RESTORE:FOR I=1 TO 40:
    !** pitch töltés
2010 READ R:P(I)=R:NEXT I
2020 RETURN
3010 DATA 1110,1436,1726,1859,
    2103,2320,2514,2603
3020 DATA 2603,2766,2911,2977,
    3099,3208,3305,3349
3030 DATA 3349,3431,3503,3537,
    3598,3652,3701,3723
3040 DATA 3723,3763,3800,3816,
    3847,3874,3898,3909
3050 DATA 3909,3930,3948,3956,
    3971,3985,3997,4003
```

```
4000 GRAPHICS 16:
     !** zongora rajzolás
4003 SET MODE 0
4005 SET PAPER 10:SET BORDER 10
4008 CLS
4010 SET INK 15
4020 PLOT  ,256,320;256,640;
     764,640;764,320;256,320
4025 PLOT,290,330,PAINT
4028 SET INK 0
4030 FOR I=5 TO 12
4040 L=I*64-4
4050 PLOT,L,399;L,320
4060 NEXT I
4065 SET INK 0
4070 FOR I=5 TO 12
4075 IF I=7 OR I=11 THEN
      L=I*64-4:PLOT,L,640;L,399:
      NEXT I
4080 L=I*64-20
4085 IF I=12  THEN PLOT,L,400;
     L,640;L+20,640;L+20,400;
     L,400:PLOT,L+10,420,PAINT:
     GOTO 4110
4090 PLOT,L,400;L,640;L+40,640;
     L+40,400;L,400
4095 NEXT I
4110 FOR I=5 TO 11
4120 L=I*64-4
4130 PLOT,L,420,PAINT
4140 NEXT I
4900 SET PAPER 15:SET MODE 0
4910 SET INK 0
4920 SET CHARACTER 180,17,25,21,
     17,17,17,113,241,241,97
4930 SET CHARACTER  172,1,1,1,1,
     1,1,1,1,1,1
4935 T1=1
4940 RETURN
5200 !**hangjegy törlés,kiiratás
5205 PRINT AT 16,T1+4:CHR$(172)
5210 PRINT AT 16,I+4:CHR$(180)
5220 T1=I
5230 RETURN
```

Készen is vagyunk!

Amikor már eleget gyönyörködtünk munkánk eredményében, újabb igényeink támadnak. Milyen jó lenne, ha a dallamot el tudnánk tárolni és kérésre visszajátszaná a gép! Gondolkozzunk el ezen a lehetőségen.

A hangjegyek tárolásához meg kell oldanunk, hogy hibásan leütött hangjegy ne kerülhessen a pufferbe. A feladat előbbi logikáját megtartva, különböző betűkaraktéreket rendelünk az egyes kívánságokhoz. Az eltárolható hangjegyek számát pedig pl. 400-ra korlátozzuk. A hangjegyeket sorszámaikkal tároljuk el.

A kívánságokhoz rendelt betűkarakterek:

Á — ne tároljuk a hangot

Ő — tároljuk a hangot

Ü — lejátszást kérünk

É — az utolsó hangot törölni szeretnénk

L — a dallam elejére állunk

A programbővítés segédváltozói:

TM = 0, ha nem kell tárolni a hangot (X$ = "Á")

TM = 1, ha kell tárolni a hangot (X$ ="Ő")

IT = 1, a dallam eleje (IT mutatja, hol tartunk a tárolásban)

T(IT) = I az eltárolt hangjegyek sorszámai

P(T(IM)), a megszólaltatandó hang pitch értéke

(IM, a futó paraméter)

```
 1  !*****************************
    *          zenegép          *
    *****************************
10  DIM P(40):DIMT(400):
    IT=1:TM=0:
    !**max.400 hangot tud tárol
       ni,külön kérésre.
        tm=0,nem tárol;
        it=1,dal eleje;        **
20  GOSUB 2000:!** Pitch töltés
25  GOSUB 4000:!**   Taszt irás
30  X$=INKEY$:IF X$="" THEN 30:
    !**      karakterre vár    **
35  !**     oktávváltás        **
```

```
  40 IF X$="y" THEN IS= 0:GOTO30
  50 IF X$="x" THEN IS= 8:GOTO30
  60 IF X$="c" THEN IS=16:GOTO30
  70 IF X$="v" THEN IS=24:GOTO30
  80 IF X$="b" THEN IS=32:GOTO30
  82 IF X$="ö" THEN TM= 1:GOTO30
     :!**  tároljuk a hangot  **
  83 IF X$="ü"THEN GOSUB 8000:
                              GOTO  30
     :!**  lejátszást kérünk  **
  84 IF X$="á" THEN TM= 0:GOTO30
     :!**  nem tároljuk       **
  85 IF X$="é" THEN IT=IT-1:GOTO
                                 30
     :!**utolsó hang törlése  **
  86 IF X$="l" THEN IT= 1:GOTO30
     :!**  dallam elejére áll **
  90 I=VAL(X$)
  95 IF I<1 OR I>8 THEN  30
 100 GOSUB 5200
 110 I=I+IS:Q=P(I)
 115 IF TM=1 THEN GOSUB 7000
 120 SOUND; PITCH Q ,DURATION 7,
     VOLUME 8
 130 GOTO 30
2000 RESTORE:FOR I=1 TO 40:
     !** pitch töltés
2010 READ R:P(I)=R:NEXT I
2020 RETURN
3000 RESTORE
3010 DATA 1110,1436,1726,1859,
     2103,2320,2514,2603
3020 DATA 2603,2766,2911,2977,
     3099,3208,3305,3349
3030 DATA 3349,3431,3503,3537,
     3598,3652,3701,3723
3040 DATA 3723,3763,3800,3816,
     3847,3874,3898,3909
3050 DATA 3909,3930,3948,3956,
     3971,3985,3997,4003
4000 GRAPHICS 16:
     !** zongora rajzolás
4003 SET MODE 0
4005 SET PAPER 10:SET BORDER 10
4008 CLS
4010 SET INK 15
4020 PLOT  ,256,320;256,640;
     764,640;764,320;256,320
4025 PLOT,290,330,PAINT
4028 SET INK 0
4030 FOR I=5 TO 12
4040 L=I*64-4
```

```
4050 PLOT,L,399;L,320
4060 NEXT I
4065 SET INK 0
4070 FOR I=5 TO 12
4075 IF I=7 OR I=11 THEN
      L=I*64-4:PLOT,L,640;L,399:
      NEXT I
4080 L=I*64-20
4085 IF I=12  THEN PLOT,L,400;
     L,640;L+20,640;L+20,400;
     L,400:PLOT,L+10,420,PAINT:
     GOTO 4110
4090 PLOT,L,400;L,640;L+40,640;
     L+40,400;L,400
4095 NEXT I
4110 FOR I=5 TO 11
4120 L=I*64-4
4130 PLOT,L,420,PAINT
4140 NEXT I
4900 SET PAPER 15:SET MODE 0
4910 SET INK 0
4920 SET CHARACTER 180,17,25,21,
17,17,17,113,241,241,97
4930 SET CHARACTER 172,1,1,1,1,1
,1,1,1,1,1
4935 T1=1
4940 RETURN
5200 !**hangjegy törlés,kiiratás
5205 PRINT AT 16,T1+4:CHR$(172)
5210 PRINT AT 16,I+4:CHR$(180)
5220 T1=I
5230 RETURN
7000 IT=IT+1:T(IT)=I:
     !** hang tárolása
7010 RETURN
8000 FOR IM=2 TO IT:
     !**lejátszás
8002 IF IT<2 THEN RETURN
8003 R=P(T(IM))
8004 I=T(IM)
8005 IF I>32 THEN I=I-32
8006 IF I>24 THEN I=I-24
8007 IF I>16 THEN I=I-16
8008 IF I>8 THEN I=I-8
8009 GOSUB 5200:
     !**berajzolja a hangjegyet
8010 SOUND,PITCH R,DURATION 7,
     VOLUME 8
8019 FOR IR=1 TO 100:NEXT IR:
     !**ennyit vár egy hanggal
```

```
8020 NEXT IM
8510 RETURN
8800 GOSUB 5200
```

Ezzel közös gondolkozásunk első lépcsősorának a tetejére értünk, pihenjünk egy kicsit.
Mielőtt tovább lépnénk, gondoljuk át az eddigi feladatokat s megoldásaikat, a megismert utasításokat.

II. RÉSZ

MESTERFOGÁSOK

Néhány gondolat a feladatok előtt

A következő fejezetek tartalma és tárgyalási módja némiképp eltér az előzőekétől. Ha figyelmesen elolvasta és megértette az eddig leírtakat, akkor joggal merülhet fel az Olvasóban a gondolat, hogy most már eljött az ideje egy "nagyobb" feladat megoldásának. A nagyobb természetesen nem elsősorban a program méretére vonatkozik — az csak következmény — hanem az összetettségre.
Könyvünk következő fejezetei kidolgozott mintafeladatokat és azok részletes magyarázatait tartalmazzák. A feladatok kiválasztásánál szem előtt tartottuk, hogy azok tükrözzék a TV-computer — továbbiakban TVC — felhasználási lehetőségeinek jellegzetes irányait, természetesen az áttekinthetőség és könnyebb tanulhatóság miatt az alapesetekre, alapproblémákra helyezve a hangsúlyt. Célunk bemutatni a feladat megfogalmazásától a program tervezésén át a program kódolásáig vezető utat esetenként be-be fordulva olyan irányokba is, amelyek utóbb zsákutcának bizonyulhatnak.

A tárgyalt feladatokban sok olyan, tipikusnak mondható problémával találkozunk majd, amelyekre a bemutatott megoldás is tipikusnak nevezhető. Szeretnénk hasznosítható ötletekkel, programozástechnikai fogások bemutatásával elősegíteni a későbbi önálló programozási munkát.

Itt rögtön felmerülhet egy kérdés: egy adott feladatnak hány féle megoldása lehetséges?
A feladatban a programot tervező ember számára előre meghatározottak azok az adatok, tevékenységek, amelyek a feladat bemenő adatai,

végrehajtási paraméterei. Számítástechnikai nyelven szólva ezek a program inputjai. A feladat kitűzője meghatározza a produkálandó eredményeket is. Ezt nevezzük outputnak.
A felhasználó szempontjából a program (rendszer) egy fekete doboz, amely az input adatokból előállítja az output adatokat, eredményeket. A fekete doboz belseje, azaz a feladat megoldásának miként je azonban nagyon sokféle lehet, az élet egyéb területeihez hasonlóan ugyanazt az eredményt több különböző úton is el lehet érni. A programozással rendszeresen foglalkozó előbb-utóbb kialakít magának egy rá jellemző programozási stílust, egyéni hangvételt. Az általunk bemutatott programok is tükröznek egyfajta stílust, ez elkerülhetetlen. Tehát nem egyedül üdvözítő megoldásokat fogunk bemutatni — ilyenek nem is léteznek — csak egy lehetségest a sok közül.

Fényújság

Az első részben már szerepelt egy hasonló program, itt annak egy bővített, újabb funkciókkal ellátott változatát mutatjuk be.

Egy valódi képújság — amilyen például egyes áruházak homlokzatán látható — úgy működik, hogy egy szöveg, ill. annak részlete jobbról balra vonul a megjelenítő eszközön, kényelmes olvashatóságot nyújtó sebességgel. Közben valahol valaki már készíti elő, gépeli be az új szöveget, amely egy parancs hatására felváltja a régit. Próbáljuk ezt a folyamatot a TVC-n modellezni!

A megjelenítő eszköz a színes képernyő — használhatunk fekete-fehéret is, de akkor nem lesz olyan látványos —, az adatbeviteli eszköz a billentyűzet. Célunk a szöveg megjelenítése és léptetése, valamint ezzel a látszólag egyidőben új szöveg bevitelét lehetővé tenni. Legyen mód az írás színének és a betűk típusainak megváltoztatására menet közben bármikor. A betűk típusát a TVC három grafikai üzemmódja határozza meg.

Az üzemmódok váltása a következő billentyűkombinációkkal történjen:

billentyű	üzemmód
ALT/1	graphics 2
ALT/2	graphics 4
ALT/3	graphics 16

Négy színből lehessen kiválasztani az aktuális írás színét a következő billentyűkkel:

billentyű	szín
1	sárga
2	cián
3	fehér
0	zöld

A feladat meghatározása után a program megtervezése a következő lépés.

Bonyolultabb programok (rendszerek) esetén érdemes azok struktúráját, algoritmusát először szimbolikus formában, programnyelvtől függetlenül leírni. Az áttekinthetőség, a kódolási munka, a tesztelés és a javíthatóság lényegesen javul, könnyebbé válik ezen a módon.
Ilyen szimbolikus forma lehet a folyamatábra. A folyamatábrában grafikai jelek szerepelnek, amelyek a program egyes funkcióit szimbolizálják.
Napjainkban egyre jobban terjed egy másik rendszer, az ún. pszeudokód. A pszeudokód egy olyan nem létező programnyelv, amelyben az utasításcsoportok, részfunkciók szöveges módon vannak leírva, a program felépítését, algoritmusát leíró szerkezet pedig olyan programutasításokkal leírt, amelyek általában minden magas szintű programnyelvben szerepelnek.

Példaként tekintsük meg mindjárt a fényújság program pszeudokódját.

```
        az aktuális sorhossz és a színek beállítása
        a szöveg paramétereinek beállítása
1:      a grafikus üzemmód beállítása
2:      a szöveg írásának kezdete
3:      a szöveg léptetése 1 karakterrel balra
        IF szöveg vége
          THEN IF grafikus mód váltása
                 THEN paraméterek beállítása
                      GOTO 1:
               ENDIF
               a szöveg újrakezdése
               GOTO 3:
          ELSE GOSUB billentyűzet olvasása
               IF új szöveg
                  THEN régi szöveg törlése
                       új szöveg jellemzőinek beállítása
                       GOTO 2:
               ENDIF
        ENDIF
        GOTO 3:
```

Billentyűzetolvasó eljárás (szubrutin)

```
    1 karakter beolvasása
    IF háttérszöveg zárása
       THEN új szöveg jelző beállítása
       ELSE IF színkód
               THEN tintaszín váltás
               ELSE IF üzemmód váltás
                       THEN üzemmódváltás-jelző beállítása
                       ELSE háttérszöveg módosítása a
                            beolvasott karakterrel
                    ENDIF
            ENDIF
    ENDIF
    RETURN
```

Sajnos a BASIC nem strukturált programnyelv, ezért a pszeudokód szerkezete és a konkrét program szerkezete el fog térni egymástól, azonban az algoritmus jobban nyomon követhető a pszeudokódból.

```
    1 !*****************************
      *       Képüjság            *
      *****************************

    4 ! kezdöértékek beállitása
    5 M=4:G=31:TI=1
   10 DIM K$*254,X$*254
   15 K$="Irható ide másik szöveg
 is "
   20 GRAPHICS M
   21 SET PALETTE 80,84,81,85
   22 SET BORDER 85:SET PAPER 0
   23 CLS:SET INK TI
   24 PRINT AT 10,1:STRING$(G+1,"
*"):PRINT AT 14,1:STRING$(G+1,"*
")
   25 P1=1:P2=1:H=LEN(K$)
   29 !**** szöveg léptetése ****

   30 FOR D=G TO 1 STEP -1
   35 PRINT AT 12,D:K$(P1:P2);
   40 GOSUB 200
   45 IF F1=1 THEN 95 ! új szöveg
et gépeltek be közben
   50 P2=P2+1
   55 IF P2>H THEN P2=H
   60 NEXT D:D=D+1
   65 P1=P1+1:IF P1<=H THEN 75
   70 IF F2=1 THEN F2=0:GOTO 20 !
      grafikus módot kell váltani

   72 GOTO 25
   75 P2=P2+1:IF P2>H THEN P2=H
   80 PRINT AT 12,D:K$(P1:P2);
   85 GOSUB 200:IF F1=1 THEN 95
   90 GOTO 65
   95 PRINT AT 12,1:STRING$(G,32)

   96 K$=X$&" ":X$="":F1=0
  100 GOTO 25
  196 !*****************************
      * klaviatura olvasó         *
      *     szubrutin             *
      *****************************

  200 FOR H=1 TO 10 !léptetési se
besség
  205 A$=INKEY$:IF A$="" THEN 240

  210 A=ORD(A$)
  215 IF A=13 THEN F1=1:RETURN
  220 IF A>48 AND A<52 THEN TI=A-
      48:GOTO 245 !szinváltás
  225 IF A>160 AND A<164 THEN
      F2=1:GOTO 260 !graf.mód vál
      tás
  230 X$=X$&A$
```

```
235 IF LEN(X$)=254 THEN F1=1
    :RETURN
240 NEXT N
245 SET INK TI
250 RETURN
260 ON A-160 GOTO 265,270,275
265 M=2:G=63:F2=1:RETURN
270 M=4:G=31:F2=1:RETURN
275 M=16:G=15:F2=1:RETURN
```

A 4-25-ös sorokban állítjuk be a kezdőértékeket. A K$ változó tartalmazza az aktuális, írás alatt álló szöveget, az X$ azt a "háttérszöveget", melyet írás közben gépelhetünk be. A szöveg a képernyő 11. sorában fog megjelenni, ill. lépni karakterként jobbról balra.

A léptetés persze látszólagos, hiszen valójában a kiírás kezdőhelyzetét változtatjuk meg folyamatosan és ezzel természetesen a kiírt szövegrész is lépésről lépésre változik. Ez a folyamat több részre bontható.

Az első részben — 30-60-as sorok — a szöveget a sor végétől a bal szélig léptetjük, azaz egyre növekvő számú karaktert kell kiírnunk. A P1 változó tartalmazza a kiírandó szövegrész kezdő karakterének helyzetét, a P2 pedig az utolsó kiírandó karakter helyét a szövegben. Ügyelnünk kell arra, hogy a P2 ne lehessen nagyobb, mint a szöveg hossza.
A második részben történik a szöveg "fogyasztása", vagyis mutatók (pointerek) különbsége határozza meg a szöveg kiírandó részét. Ha a P1 nagyobb, mint a szöveg karakterekben mért hossza (H), akkor vége a léptetési ciklusnak.
A további folytatást az határozza meg, hogy érkezett-e igény a grafikus mód váltására a léptetés során. Ebben az esetben a 200-as soron kezdődő billentyűzetolvasó szubrutin az F2 jelzőt 1-es értékre állítja, de ez csak akkor jut érvényre, ha már a teljes szöveg kilépett a képernyőről.

A billentyűzetolvasó szubrutin lehetővé teszi új szöveg karakterenkénti begépelését az aktuális léptetése közben. Az X$ változóba gyűjtjük a háttérszöveget. Ennek gyűjtése akkor ér véget, ha bármikor megnyomjuk a RETURN billentyűt vagy ha a szöveg hossza eléri a 254

karaktert (ekkor nem kell RETURN). A régi szöveg léptetését meg kell állítani, ha elkészült az új. Ezt az F1 jelző 1-es állapota jelzi. Az új szöveg belép a régi helyére és ettől kezdve azt léptetjük.

A billentyűzetolvasó szubrutin olvassa be színváltás kérése esetén a színkód helyzeteként a sorszámát is, amely a SET PALETTE utasítás egyik paraméterére mutat. A szubrutin 260-275-ös sorai a grafikus mód váltásához kellő paramétereket állítják be:

- az üzemmód sorszámát (M);
- az egy sorba írható karakterek számát (G);
- és a jelzőt (F2).

A fényújság program működésének megértése természetesen úgy a legkönnyebb, ha a magyarázat elolvasása előtt kipróbáltuk és pontosan megfigyeltük mit csinál.

Ennyit bemelegítésül, ezután következzék egy kicsit összetettebb feladat.

Számok szöveggé alakítása

Képzeljünk magunk elé egy csekket. Minden ilyen nyomtatványon a befizetendő összeget kétféleképp kell feltüntetnünk, számmal és betűvel. Adatfeldolgozó számítógépes rendszereknél — pl. számlázásnál — is előfordul, hogy egy számot, biztonsági okokból, betűvel is le kell írnunk.

A második feladatban készítsünk egy olyan programot, amely egy 0 és 999999 közötti egész számot beolvas a billentyűzetről és azt betűkkel, szöveges formában kiírja a képernyőre. Természetesen ügyelnünk kell a nyelvhelyességi szabályok betartására is: ha a szám nagyobb mint 2000, akkor az ezresek után kötőjelet kell írni, persze csak akkor, ha van értékes százas, tízes vagy egyes helyi érték.
A megoldásban törekedjünk arra, hogy minél kevesebb szöveges konstanst — állandó szövegelemet — használjunk fel.

Első lépésként elemeznünk kell az átalakítandó számot, benne van-e a megadatott tartományban, majd számjegyekre bontással meg kell vizsgálni az egyes helyi értékek tartalmát.
Érdemes megfontolnunk, hogy a felbontást a legnagyobb vagy a legkisebb helyi értéknél kezdjük-e. Mivel nem tudjuk melyik a legnagyobb helyi érték az adott számban, viszont egyes helyi érték biztosan van, ezért célszerű a felbontást innen elkezdeni.
A számjegyek azonosítása a következő lépés, ez azonban nem választható el a számjegy-szöveg hozzárendelés módjának megválasztásától. Kézenfekvőnek tűnő megoldás, IF utasításokkal meghatározni a számjegy tényleges értékét és ennek ismeretében hozzárendelni a megfelelő szöveges konstanst, pl.:

```
IF számjegy=2 THEN betű$="kettő"
```

Persze ezt a műveletet nem csak egy, hanem valamennyi helyi érték kiértékelésénél el kell végeznünk. A program ennek a "favágó" módszernek köszönhetően nagyon hosszú lesz, igaz, megírása meglehetősen mechanikus, különösebb erőfeszítést nem igénylő.

Lényegesen rövidítheti a programot egy másik megoldás.
Helyezzük el egy tömbben a számjegyek neveit, így:

tömb$(1)="egy", tömb$(2)="kettő" stb.

Tekintsük ezek után a számjegyet a tömb indexének, ekkor nincs szükség a konkrét szám meghatározására a megfelelő szöveges konstans kiválasztásához, az teljesen automatikus lesz, pl.:

betű$=tömb$(számjegy)

A program pszeudokódos felépítése:

```
   az üzemmód és a színek beállítása
   a szövegtömbök létrehozása, feltöltésük
1: INPUT szám
   FOR helyi érték=1 to legnagyobb helyiérték
   IF számjegy nem nulla
      THEN szövegtömb kiválasztása
           index=számjegy értéke
           IF szám  2000 E$ a helyi érték  százas
              THEN IF szöveg nem üres
                      THEN szöveg = szöveg + "-"
                   ENDIF
           ENDIF
           szöveg = szöveg + megcímzett tömbelem
      ELSE szöveg = "nulla"
           GOTO 2
   ENDIF
2: NEXT
   a szöveg kiírása
   IF újabb átalakítás
      THEN GOTO 1
      ELSE END
   ENDIF
```

És maga a program:

```
    1 !*************************
      *                       *
      *   szám-betü átalakitó *
      *                       *
      *************************

   10 GRAPHICS 4
   20 SET PALETTE 0,85,80,68
   30 SET BORDER 21:SET PAPER 1:
      CLS
   40 DIM EGY$(9),TIZ$(9),BETU$*
      100
   49 !** számnevek beolvasása **

   50 RESTORE
   55 FOR N=1 TO 9
   56 READ EGY$(N)
   60 NEXT N
   65 FOR N=1 TO 9
   70 READ TIZ$(N)
   80 NEXT N
   90 SET INK 0:PRINT AT 3,1:"Kér
em a számot:                    ";:
      PRINT AT 3,16:"";
   95 SET INK 2:INPUT PROMPT "":
      SZAM
  100 IF SZAM>999999 OR SZAM<0
      THEN PRINT AT 10,1:"Ezt nem
  tudom":FOR X=1 TO 2000:
      NEXT X:PRINT AT 10,1:
      STRING$(26," "):GOTO 90
  110 SET INK 3
  130 N=1:BETU$="":M=0:EM=1:EF=0
  131 !*************************
       * N - aktuáli helyiértek *
       * M - maradék           *
       * EM- elözö maradék     *
       * EF- ezres helyiértek  *
       *     jelzö             *
       *************************

  140 SZAM=SZAM/10:SZ=INT(SZAM)
  150 M=(SZAM-SZ)*10
  160 ON N GOSUB 500,600,700,
      800,900,1000
  170 IF SZ=0 THEN 190
  180 SZAM=SZ:N=N+1:GOTO 140
  189 !
  191 !
  200 PRINT AT 23,1:"Akar új szám
ot megadni ? (i/n):"
  210 A$=INKEY$:IF A$="" THEN 210

  220 IF A$="i" OR A$="I" THEN
      CLS:GOTO 90
  230 IF A$="n" OR A$="N" THEN
      END
  240 GOTO 210
  500 !*************************
       *       egyesek         *
       *************************

  501 IF M<>0 THEN 520
  505 EM=M
```

```
510 IF SZ=0 THEN BETU$="nulla":
    RETURN
520 BETU$=EGY$(M):EM=M
530 RETURN
600 !**************************
    *       tizesek          *
    **************************

601 IF M=0 THEN 650
610 IF EM=0 THEN 640
620 IF M=1 THEN BETU$="tizen"&
    BETU$:GOTO 650
630 IF M=2 THEN BETU$="huszon"&
    BETU$:GOTO 650
640 BETU$=TIZ$(M)&BETU$
650 EM=M
660 RETURN
700 !**************************
    *        százasok        *
    **************************

701 IF M=0 THEN 720
710 BETU$=EGY$(M)&"száz"&BETU$
720 EM=M
730 RETURN
800 !**************************
    *        ezresek         *
    **************************

801 IF SZAM*10<2 THEN 810
805 IF BETU$="" THEN 810
806 BETU$="-"&BETU$
810 IF M=0 THEN 830
820 BETU$=EGY$(M)&"ezer"&BETU$:
    EF=1
830 EM=M
840 RETURN
900 !**************************
    *       tizezresek       *
    **************************

901 IF M=0 THEN EM=M:RETURN
910 IF EF=0 THEN BETU$="ezer"&
    BETU$:EF=1
920 GOTO 610
1000 !*************************
     *       százezresek     *
     *************************

1001 IF M=0 THEN EM=M :RETURN
1010 IF EF=0 THEN BETU$="ezer"&
     BETU$:EF=1
1020 GOTO 710
1099 !*************************
     *       számnevek       *
     *************************

1100 DATA egy,kettö,három,négy,
     öt,hat,hét,nyolc,kilenc
1110 DATA tiz,húsz,harminc,
     negyven,ötven,hatvan,
     hetven,nyolcvan,kilencven
```

A 40-80-as sorok végzik el a számnevek tömbökbe töltését, ezek eredetileg az 1100-1110-es **DATA** sorokban voltak. Következő lépésként a 95-ös sorban beolvassuk a számot a billentyűzetről. Ezt rögtön egy tartalmi ellenőrzés követi és, ha kell hibaüzenetet írunk a képernyőre, majd új számot várunk.

A 140-150-es sorokban történik meg a szám helyi értékekre bontása a legkisebb helyi értéktől kezdve. A H változó tartalmazza a helyi érték kódját, az M változó pedig annak értékét. Kicsit körülményesnek tűnhet ez a megoldás, de mivel a TVC BASIC-ben nincs egész osztás és maradék osztás művelet, így jobb lehetőség nem nagyon kínálkozik.

Az **ON** H **GOSUB** utasításban megadott sorszámokon kezdődnek azok az alfunkciók (szubrutinok), amelyek a számjegy-betű hozzárendeléseket elvégzik. Helyi értékenként egy-egy ilyen kell.

A betűS változóban gyűjtjük össze elemenként azt a szöveget, amely végül megfelel a megadott számnak. Az "előző maradék"-ot azért tároljuk, hogy el tudjuk dönteni adott helyen, hogy a "tíz-tizen", ill. "húsz-huszon" számnevek közül melyik kell.

A 900-as és az 1000-es soron kezdődő szubrutinok nem önálló alfunkciók, hiszen belőlük egy ugrás történik a 600-as ill. a 700-as szubrutinokra. Ennek a megvalósításnak hátránya, hogy felborítja a program moduláris szerkezetét, előnye viszont egyrészt az, hogy rövidebb lesz a program, másrészt bemutatható a szubrutinkezelés nagymérvű rugalmassága.

Az SZ változó 0 értékénél a szám elemzése befejeződött, már csak az eredmény kiírása van hátra — 190-es sor — és a program befejezhető vagy újraindítható a megfelelő válasz begépelésével.

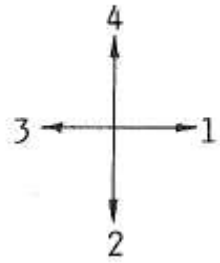
Bekerítőjáték

A személyi számítógépek otthoni felhasználásának egyik fő területe a játék. Erre lehetőséget nyújt a nagy felbontású grafikus megjelenítés a TV képernyőjén, az egyszerű kezelés és nem utolsó sorban a nagy teljesítmény.
A játékprogramok többsége gépi kódú program, de BASIC nyelven is lehet egyszerűbb, ám szórakoztató játékot készíteni.

A harmadik feladat egy akciójáték, amelyet két játékos játszhat egymás ellen.
A játékmező 24*64 karakter méretű terület, minden oldalon egy karakter szélességű kerítés határolja. Ezen belül épít jobb és bal oldalról egyidőben indulva, mindkét játékos folyamatosan egy-egy kerítést. Építés közben jobbra és balra 90 fokkal bármikor el lehet fordulni. A cél az, hogy a kerítés minél hosszabb legyen — az alakja természetesen tetszőleges —, de nem szabad nekiütközni sem az ellenfél kerítésének, sem a pálya szélének, sem a saját előzőleg felépített kerítésnek. Ütközéskor az ellenfél egy pontot kap. A játék 15 nyert pontig tart. Együttes ütközés esetén a gép dönti el, ki kapja a pontot.

A BASIC, mint tudjuk, az interpreteres megvalósítás miatt meglehetősen lassú a gépi kódú programhoz képest. Egy olyan játékprogrammal, amely folyamatos beavatkozást igényel a játékostól, nagyon fontos, hogy a reakcióidő ne legyen nagy, vagyis a játékost ne "tartsa fel" a program. A feladatot tehát akkor oldjuk meg jól, ha ezt követelményt helyezzük mindenek elé.

A pályát tekintsük egy olyan koordináta rendszernek, amelyben y irányban 24, x irányban 64 osztás van, és a játékos 4 irányba haladhat benne. Jelöljük sorszámmal ezeket az irányokat:



Az irányváltoztatás mindig az aktuális irányhoz viszonyított jobbra ill. balra fordulást jelenti. A programnak adminisztrálnia kell tehát az aktuális irány kódját ahhoz, hogy forduláskor ki tudja számítani a fordulás irányának koordinátáit. Ezt természetesen mindkét játékosnál el kell végeznie.
A balról induló játékos a Q billentyűvel fordulhat balra, a W-vel jobbra, a jobbról induló O-val fordulhat balra és a P-vel jobbra.

Érezhető már, hogy egy új kerítéselem felrajzolását rengeteg vizsgálatnak kell megelőznie:

— akar-e fordulni valamelyik játékos és ha igen, melyik;
— milyen irányba halad jelenleg a játékos;
— melyik irányba akar fordulni.

Ezek után még ki kell számítani mindkét játékos lépésének koordinátáit — az a játékos is lép, aki nem változtat irányt. Ha csupán **IF** utasítások sorozatával szeretnénk megoldani a problémát, akkor a program olyan lassú lenne, hogy gyakorlatilag nem lehetne vele játszani. Ezért hatékonyabb, bár bonyolultabb megoldásra van szükség, olyanra, amely csökkenti a vizsgálatok számát. Az indexelés igen alkalmas erre a célra, ezzel a módszerrel kb. negyedére lehet csökkenteni a vizsgálatok számát.

A program másik fő része az ütközésvizsgálat és a pontozás.
A TVC BASIC-ben nincs a képernyő adott pontjainak tartalmát lekérdező utasítás (pl. SCREEN (y,x)). Ezt helyettesítendő egy térképet — 24*64-es mátrixot — kell készítenünk, amely a képernyőn megjelenő ábra tükörképét tartalmazza.
Például írjuk a képernyő 2. sorának 3. oszlopába az "a" betűt:

```
PRINT AT 2,3:"a"
```

A térképen ezt így jelöljük:

```
térképS(2,3)="a" vagy térkép(2,3)=65
```

ahol a 65 az "a" betű ASCII kódja.
A programban a kerítéselemet felhasználói karakter definiálásával állítjuk elő — SET CHARACTER 161, ... —, tehát az ASCII kódja 161 lesz, a térképre is ezt írhatjuk be.

A játékosok minden lépésénél megvizsgáljuk történt-e ütközés. Ha igen, akkor először azonosítanunk kell, ki ütközött, majd a másik játékos pontszámát meg kell növelni.

Következzék a program pszeudokódja és maga a programlista.

```
   az üzemmód és a színek beállítása
   a kezelés kiírása a képernyőre
1: sebességválasztás
2: a pálya felrajzolása a képernyőre
   a két kerítés kezdőpontjának meghatározása
3: a játékosok lépéseinek beolvasása
   IF irányváltoztatás lesz
      THEN a játékos azonosítása
           a fordulás irányának meghatározása
           az új irány meghatározása
           iránykód tárolás
           a másik játékos lépésének meghatározása
      ELSE mindkét játékos következő lépésének meghatá-
           rozása
   ENDIF
   IF ütközés
      THEN az ütköző játékos azonosítása
           a másik játékos pontszámának növelése
```

```
            IF 15 pont
               THEN IF újabb játék legyen
                       THEN eredmény törlése
                            GOTO 1:
                       ELSE program vége
                    ENDIF
                 ELSE GOTO 2
              ENDIF
        ELSE a lépések végrehajtása
              GOTO 3
     ENDIF
```

```
   1  !*************************
      *    Bekerítö játék      *
      *************************

   5 GRAPHICS 2:SET PALETTE 0,85

   7 POKE 2917,10 :!* DELAYKEY
  10 SET BORDER 21:SET PAPER 1:
     SET INK 0:CLS
  15 PRINT AT 3,17:
 "A játék szabályai és kezelése"

  16 PRINT AT 5,1:"A játékot két
 játékos játszhatja. Mindketten
egy-egy keritést   épitenek úgy,
 hogy ne ütközzenek neki semmine
k. Aki továbbtudja épiteni, pont
ot kap. Az győz, aki elöbb gyüjt
 15 pontot."
  17 PRINT AT 9,5:"Bal oldali já
tékos:";TAB(45);"Jobb oldali ját
ékos"
  18 PRINT AT 12,5:"q - balra";
     TAB(45);"o - balra"
  19 PRINT AT 15,5:"w - jobbra";
     TAB(45);"p - jobbra"
  20 PRINTAT 18,5:"";:INPUT PROM
PT   "sebesség(1-5):":Q
  22 IF Q<1 OR Q>5 THEN 20
  24 PRINT AT 22,5:"Ha elolvasta
, nyomjon meg egy gombot"
  26 GET
  27 !*************************

  28 DIM X(2):DIM Y(2):DIM IR(2)
     :DIM Z$(24,64)*1
  29 FOR N=1 TO 24:
     FOR M=1 TO 64
  30 Z$(N,M)=" ":
     NEXT M:NEXT N
  35 SET CHARACTER 161,255,255,
     195,195,195,195,195,195,255
,    255
```

```
 40 RANDOMIZE:TJ=0:TB=0
 45 SET PALETTE 20,0:
    SET PAPER 0:SET INK 1:CLS
 50 FOR N=1 TO 64:PRINT #0,
    AT 1,N:CHR$(161);
 52 Z$(1,N)="*":NEXT N
 54 FOR N=1 TO 64:PRINT #0,
    AT 24,N:CHR$(161);
 56 Z$(24,N)="*":NEXT N
 58 FOR N=1 TO 24:PRINT #0,
    AT N,1:CHR$(161);
 60 Z$(N,1)="*" :   NEXT N
 62 FOR N=1 TO 24:PRINT #0,
    AT N,64:CHR$(161);
 64 Z$(N,64)="*": NEXT N
 66 PRINT AT 1,30:TB;":";TJ
 69 !**************************
    * a játék kezdete       *
    **************************

 70 X(1)=1:X(2)=64
 75 Y(1)=2+INT(RND*21):Y(2)=2+
    INT(RND*21)
 82 IR(1)=1:IR(2)=3
 85 F1=0:F2=0
 90 FOR N=1 TO 11-(2*Q)
100 A$=INKEY$:IF A$="" THEN 120

110 GOTO 145
120 NEXT N
140 IF A$="" THEN 165
145 IF A$="q" THEN IND=1:
    GOTO 170
150 IF A$="ö" THEN IND=1:
    GOTO 190
155 IF A$="p" THEN IND=2:
    GOTO 190
160 IF A$="o" THEN IND=2:
    GOTO 170
165 IND=2:GOSUB 210:IND=1:GOSUB
    210:GOTO300
166 !**************************

170 ON IR(IND) GOSUB 180,182,
    184,186
172 IF IND=1 THEN IND=2:GOSUB
    210:GOTO 300
175 IND=1:GOSUB 210:GOTO 300
179 !*** irányváltoztatás *****

180 Y(IND)=Y(IND)-1:IR(IND)=4:
    RETURN
182 X(IND)=X(IND)+1:IR(IND)=1:
    RETURN
184 Y(IND)=Y(IND)+1:IR(IND)=2:
    RETURN
186 X(IND)=X(IND)-1:IR(IND)=3:
    RETURN
189 !**************************

190 ON IR(IND) GOSUB 200,202,
    204,206
192 IF IND=1 THEN IND=2:GOSUB
```

```
    210:GOTO 300
195 IND=1:GOSUB 210:GOTO 300
199 !*** irányváltoztatás******

200 Y(IND)=Y(IND)+1:IR(IND)=2:
    RETURN
202 X(IND)=X(IND)-1:IR(IND)=3:
    RETURN
204 Y(IND)=Y(IND)-1:IR(IND)=4:
    RETURN
206 X(IND)=X(IND)+1:IR(IND)=1:
    RETURN
207 !**********************
    *   változatlan irány  *
    **********************

210 ON IR(IND) GOSUB 212,214,
    216,218:RETURN
212 X(IND)=X(IND)+1:RETURN
214 Y(IND)=Y(IND)+1:RETURN
216 X(IND)=X(IND)-1:RETURN
218 Y(IND)=Y(IND)-1:RETURN
300 GOSUB 370
301 !**************************

302 T=TB+TJ:T1=(INT(T/2))*2
305 IF T=T1 THEN 320:ELSE 330
320 IF F3=1 THEN TJ=TJ+1:PRINT
    #0,AT Y(1),X(1):"*":
    GOTO 400
325 GOSUB 350:IF F2=1 THEN 85
330 IF F4=1 THEN TB=TB+1:PRINT
    #0,AT Y(2),X(2):"*":
    GOTO 400
335 GOSUB 360:IF F1=1 THEN 85
340 GOTO 320
350 SET INK 1:PRINT #0,AT Y(1),
    X(1):CHR$(161)
352 Z$(Y(1),X(1))="*"
354 F1=1:RETURN
360 SET INK 1:PRINT #0,AT Y(2),
    X(2):CHR$(161)
362 Z$(Y(2),X(2))="*"
364 F2=1:RETURN
365 !**************************

370 F3=0:F4=0
380 IF Z$(Y(1),X(1))="*"
    THEN F3=1
385 IF Z$(Y(2),X(2))="*"
    THEN F4=1
390 RETURN
399 !**************************

400 SOUND PITCH 2420,DURATION
    40
403 PRINT AT 1,30:TB;":";TJ
405 IF TB=15 OR TJ=15 THEN 415
410 B$=INKEY$:IFB$="" THEN 410
411 IF B$=" " THEN 45:ELSE 410
415 PRINT AT 12,10:
    "Játszik újra? (i/n):":
    GET V$
```

```
425 IF V$="i" OR V$="I" THEN 5
430 IF V$="n" OR V$="N" THEN
    POKE 2917,30:END
435 GOTO 415
```

Mivel a pálya mérete 24*64 karakter, a GRAPHICS 2 üzemmódot állítjuk be, ilyenkor egyidőben csak két szín alkalmazható.

A 7-es sorban lecsökkentjük, a billentyűzeten, egy billentyű megnyomásától a kódjának a beolvasásáig eltelő időt. A DELAYKEY rendszerváltozó a 2917d ill. 0B65h címen helyezkedik el a tárban, eredeti értéke 30. Ezt meg kell szorozni 1/50 ms-al és megkapjuk a billentyűzet késleltetési idejét ms-ban. Ha 10-re csökkentjük a DELAYKEY tartalmát, akkor a játék folyamatosabb és nem utolsó sorban igazságosabb lesz. Azért lesz így, mert egyik játékos sem tudja blokkolni a billentyűzetet, megakadályozva a másikat a fordulásban.

A 15-19-es sorok a játékszabályok és a kezelés képernyőre írását végzik el. Öt különböző sebességgel folyhat a játék; a 90-es sorban egy késleltető ciklus kezdődik, amelynek magja a billentyűzet olvasása. Ezen a ponton avatkozhatnak be a játékosok a játék menetébe. A Q változó a várakozás idejét határozza meg.

A 28-as sorban hozzuk létre a térképet — ezt a Z$ tömb tárolja — és a játékosok azonosítására valamint a lépések nyilvántartására való vektorokat. Ezek értelmezését tartalmazza a következő felsorolás:

X(1) — bal oldali játékos helyzetének x koordinátája
X(2) — jobb oldali játékos helyzetének x koordinátája
Y(1) — bal oldali játékos helyzetének y koordinátája
Y(2) — jobb oldali játékos helyzetének y koordinátája
IR(1) — bal oldali játékos irányának kódja
IR(2) — jobb oldali játékos irányának kódja

A játékosokat — a könnyebb indexelhetőség kedvéért — kódoljuk: a balról induló játékos lesz az egyes, a jobbról induló a kettes sorszámú.

A 140-160-as sorokban vizsgáljuk meg, hogy kért-e fordulást valamelyik játékos, azaz megnyomott-e valaki vezérlő billentyűt. Ha igen, akkor két eset lehetséges: a "Q", "W", "P", "O" karakterek esetén, az 1-es vagy a 2-es játékos szeretne irányt változtatni, az ezektől eltérő karaktereket figyelmen kívül hagyjuk, s a 165-ös soron folytatódik a program.
Ha az addigi haladási irány mindenkinek megfelelő, akkor nincs beolvasott karakter, az A$ változó értéke az üres (" ") lesz.

A program további működését egy példán keresztül lehet a legegyszerűbben nyomon követni. Tételezzük fel, hogy a bal oldali játékos az 1-es kódú irányba halad és jobbra akar fordulni (W), a jobb oldali játékos iránya 2-es. Kövessük végig a program működését!
Az IND változó értéke 1, ez a játékos azonosító kódja. A 190-es sorban az **ON** ... **GOSUB** utasítás arra a sorra adja át a vezérlést, amelynek listabeli sorszáma egyenlő az aktuális irány kódjával, azaz jelen esetben a 200-as szubrutinra.
Az Y(1) az 1-es játékos helyzetének y koordinátáját tartalmazza, ezt 1-gyel növelni kell, és be kell állítani az új irányt, vagyis az IR(1) = 1. Az X(1) koordináta nem változik. Visszatérés következik a 192-es sorra.
Ezután a 2-es játékos lépését — amely változatlan irányú — is végre kell hajtani. Mivel az IND = 1, ezért a **THEN**-ágon megyünk tovább.
A 210-es sorban az IR(2) értéke 2, a vezérlést tehát a 214-es szubrutin kapja meg. Ezt végrehajtva először visszatérünk a 210-es sorra, majd tovább a 192-esre. Vegyük észre, hogy ha a példánkban felcseréljük a játékosokat, akkor is ugyanezeket az utasításokat kell végrehajtani!

A következő lépés az ütközések vizsgálata és azok adminisztrálása. Az F3 változó az 1-es számú játékos ütközésjelzője, az F4 változó a kettesé. Értékük akkor 1, ha a térkép aktuális helyzetén — vagyis az új lépés helyén — csillag karakter áll.
A TB és TJ változók tartalmazzák a pontszámokat. A 302-es sorban vizsgáljuk meg ezeket: ha összegük páros, akkor először az 1-es játékos lépését vizsgáljuk, ütközött-e (320-as sor). Ütközés esetén

egy csillagot írunk ki a képernyőre az ütközés pontjába és figyelmeztető hangjelzést is generálunk (400-as sor). Amennyiben a TB és TJ változók összege páratlan, úgy a 2-es játékos lépését vizsgáljuk először (330-as sor). A program további menete megegyezik az előbb leírtakkal.
Ezzel a módszerrel kiküszöböltük a vizsgálatok igazságtalanságát egyidejű ütközések esetén.' Általánosan is elmondható, hogy csak az igazságosan játszó program nyújt szórakozást, mert másképpen hamar kedvét szegi a játékosnak.

Ha senki sem ütközött, akkor a képernyőre is kiírjuk a lépések eredményéül az új kerítéselemeket és a 85-ös soron folytatódik a program. Bárki eléri a 15 pontot, véget ér a játék és választható a kilépés vagy a program újraindítása.

A fenti program alapján talán érzékelhetővé vált, hogy egy egyszerű játék megírása is elég nehéz feladat. A bonyolult, "igazi" játékprogramok megírására csak profi programozók vállalkozhatnak a siker reményében.

Nyilvántartó program

Most egy olyan, az eddigiektől eltérő típusú feladat következzék; amelyben eddigi ismereteinket mintegy összegezzük egy nagyobb terjedelmű feladat megoldásában. Témánk az adatfeldolgozás lesz.

Az adatfeldolgozás a számítógépek alkalmazásának egyik fő területe. Egyre több az olyan vállalat, intézmény, vállalkozás, ahol az adminisztrációs feladatok kisebb-nagyobb részét a számítógéppel végeztetik el.
Ezt a folyamatot felgyorsította, szinte forradalmasította a mikroszámítógépek elterjedése, hiszen ez tette lehetővé, hogy széles körben alkalmazzák, mindennapos munkaeszközként, a számítógépet.

A jelenlegi tipikus alkalmazások az adatfeldolgozás körében:

- bérszámfejtés;
- anyag- és költségnyilvántartás;
- könyvelés;
- számlázás;
- szövegszerkesztés;
- szövegfeldolgozás.

Természetesen a számítógép alkalmazási területét alapvetően meghatározza a számítógép teljesítménye. Nagyon fontos ennek az összhangnak a megteremtése, mert e nélkül a számítógép használatának egy folyamatban nagyobb lehet a kára, mint a haszna.

A személyi számítógépek alapvetően házi használatra készültek, nem teszik lehetővé nagy tömegű adat hatékony feldolgozását. Ennek fő okai:

- hatékony adatkezelő rendszer (file-kezelő) hiánya;
- nagy kapacitású, közvetlen adatelérést nyújtó háttértároló hiánya;

— viszonylag lassú adatelérés;
— nagy teljesítményű és megbízhatóságú perifériák hiánya.

Ez persze nem azt jelenti, hogy pl. a TVC alkalmatlan lenne adatfeldolgozásra, csak a feladatot körültekintően kell megválasztani, figyelembe véve a gépkategória korlátait.
Erre szeretnék a továbbiakban egy példát bemutatni.

Készítsünk nyilvántartást, amelyben egy gépjármű üzemeltetési költségei szerepelnek egy 12 hónapos időtartományra: üzemanyag, alkatrész, javítás, esetleges bírság stb. Lehessen új adatokat felvenni, különféle statisztikákat készíteni, egyes adatokat visszakeresni.
Fontos követelmény, hogy az adatok a háttértárolón tárolhatók legyenek.

Első lépésként meghatározzuk a nyilvántartást felépítő elemi adatokat. Az adatfeldolgozásban az adatok elemi egységei a mezők. Mező lehet pl. egy vezetéknév, utónév, személyi szám, költség megnevezés stb.
Egy mező — tehát egy adat — típusa szerint lehet numerikus, alfanumerikus, alfabetikus és logikai. A numerikus mező csak számok leírására szolgáló karaktereket, az alfabetikus csak betűket és írásjeleket, az alfanumerikus bármilyen karaktert tartalmazhat. A logikai mező logikai "igaz" vagy "hamis" értéket vehet fel.

A logikailag összetartozó mezők alkotják a rekordot. A rekordban a mezők előre meghatározott, rögzített sorrendben helyezkednek el.

A példánkban szereplő mezők:

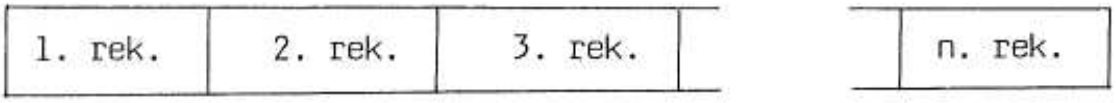
Sorszám	Mezőnév	Hossz (byte)	Típus
1.	Dátum	8	alfanumerikus
2.	Költség megnevezése	15	alfanumerikus
3.	Km-óra állása	6	alfanumerikus
4.	Anyag-üzemanyag-költség	5	alfanumerikus
5.	Munkadíj	5	alfanumerikus
6.	Egyéb költség	5	alfanumerikus
Összesen:		44	

Mint az a táblázatból kiolvasható, a rekord az egy alkalommal felmerülhető költség — kiadás — jellemzőit tartalmazza.

A rekordokat a háttértárolón — mágnesszalagon vagy hajlékony lemezen — file-okban tároljuk. A file a logikailag összetartozó rekordok halmazát jelenti.
Az adattároláskor, a háttértárolón, a rekordokat keletkezésük sorrendjében szervezzük file-okba; s később sorrendi olvasással érhetjük el őket.

Nyilvántartásunk 12 file-ból fog állni, azaz minden hónap adatait egy-egy file tartalmazza, ez az adatok könnyebb módosíthatósága miatt célszerű. Ahhoz ugyanis, hogy a file egyetlen rekordját módosíthassuk, a file teljes tartalmát újra kell írni.
Ha mágneslemezen van a file, akkor egy másik file-ba kell kiírni a módosítás után, mert a file tartalmát nem írhatjuk felül.
Szalagon lévő file esetén az eredeti file-t felülírhatjuk az új file tartalmával — persze csak akkor, ha előzőleg visszacsévéltük a szalagot a file elejére.

Nézzünk erre egy példát:



amely egy file felépítését mutatja.
A 3. rekord kiolvasásához először az 1., majd a 2. rekordot kell kiolvasni. Programunkkal módosíthatjuk a végül beolvasott 3. rekord tartalmát és vissza szeretnénk azt írni a szalagra. Sajnos, közvetlenül nem tehetjük meg, mert nem ismerjük a rekord fizikai helyét a szalagon.
A problémát csak úgy oldhatjuk meg, hogy a file minden rekordját beolvassuk a tárba, módosítjuk a kívánt rekord tartalmát és ezek után a teljes file-t felírjuk újra a szalagra.
Mágneslemezen elhelyezkedő file-ok esetén az eredeti file-t nem lehet felülírni, mert ezt a file-kezelő nem engedi meg. Új file-nevet kell tehát megadnunk a módosítás elvégzése után, és ebbe kell kiíratni a tárból a rekordokat.

Mivel a rekordok kezelését a tárban kell megoldanunk, és a tár "befogadóképessége" korlátozott, maximálnunk kell az egy file-ba írható — egyidőben a tárban lévő — rekordok számát. Ebben a programban ezt 5-ben állapítjuk meg, de ez az érték nagyon egyszerűen módosítható, csak a megfelelő ciklusok végértékét kell majd megváltoztatni hozzá.

A tervezés következő lépéseként határozzuk meg a program fő funkcióit!

1. Az adatállomány területének létrehozása.
   A mágnesszalagon helyet kell foglalni a 12 file-nak a maximális rekordszám figyelembevételével.
2. Adatfelvétel.
   Új rekordok bevitele egy file-ba, azaz a file módosítása.
3. Költségszámítások, adatkeresés.
   Ez olyan funkció, amelyet érdemes további alfunkciókra bontani:

   a) átlagfogyasztás számítása egy kiválasztott időtartományra;

b) 1 km-re eső üzemeltetési költség számítása;

c) tetszőleges rekord visszakeresése és listázása.

A program tervezésénél és kódolásánál törekednünk kell arra, hogy ezek a funkciók és alfunkciók a BASIC programban is különálló részeket, ún. modulokat alkossanak.

Ennek több előnye is van: a nagyméretű program is könnyen áttekinthető, javítható. Újabb funkciók beépítésével a program bővíthető anélkül, hogy a meglévő részt nagymértékben módosítani kellene.

A további előnye, hogy a nyilvántartás, mint keret más tartalommal is megtölthetővé válik. A rekordkép természetesen más lesz ebben az esetben, és ez jelentős módosulást fog eredményezni az adatok beolvasásánál, listázásánál, módosításánál, de a főfunkciók szerkezete megtartható.

A feladat — talán kissé hosszú, ám a könnyebb érthetőség miatt elengedhetetlen — leírása után következzék a program pszeudokódja és ezt követően a programlista.

Főmenü

```
   a tömbök létrehozása
   a menü kiírása a képernyőre
1: funkcióválasztás
   ON funkciósorszám GOSUB
      1. file-ok inicializálása
      2. adatfelvétel, módosítás
      3. költségszámítások
      4. program vége
   ELSE GOTO 1:
```

```
File-ok inicializálása

   FOR január TO december
       file megnyitása
       a rekordok feltöltése töltőkarakterrel
       a rekordok felírása
   NEXT

Adatfelvétel, módosítás

   a hónapfile kiválasztása
   IF a file-ban vannak rekordok
      THEN a rekordok beolvasása és formátumozott kiírásuk a
           képernyőre
   ENDIF
1: a rekordok beolvasása a billentyűzetről
   IF adatbevitel vége
      THEN az összes rekord beírása a file-ba
           a file lezárása
           RETURN a főmenübe
      ELSE a rekord összeállítása a beolvasott mezőkből
           a rekord tárolása
           GOTO 1:
   ENDIF

Költségszámítási menü

   a menü kiírása a képernyőre
1: funkcióválasztás
   ON funkciósorszám GOSUB
      1. átlagfogyasztás számítása
      2. 1 km-re eső költség
      3. rekordok keresése, listázásuk
   ELSE GOTO 1:
```

Átlagfogyasztás számítása

IF a rekordok nincsenek a tárban
THEN a file-ok beolvasása
ENDIF
Az időtartomány kijelölése a képernyőre laponként, a listázott rekordok alapján.
A megtett út kiszámítása a rekordok KM mezőiből, az átlagfogyasztás számítása.
Az eredmény listázása.

1 km-re eső költség

IF a rekordok nincsenek a tárban
THEN a file-ok beolvasása
ENDIF
FOR első rekord TO utolsó rekord
anyag, munkadíj, egyéb mezők tartalmának összegzése
NEXT
Az összegek listázása
A megtett út kiszámítása
1 km-re eső költség meghatározása

```
   IF a rekordok nincsenek a tárban
      THEN a file-ok beolvasása
   ENDIF
   a megnevezés mező beolvasása
   rekordkeresés a file-ban a megnevezés mező alapján
1: IF file vége
      THEN RETURN a főmenübe
      ELSE IF megvan
              THEN a rekord listázása
           ENDIF
           a következő egyező rekord keresése
    ENDIF
    GOTO 1:
```

```
    1  !*************************
       * Gépkocsi üzemeltetési   *
       * költségeinek nyilvántar-*
       * tása 1 évre             *
       *************************

   20  GRAPHICS 2:SET PALETTE 85,1

   21  DIM T$(60)*44,Y$*44,Y1$*44,
       Y2$*44,H(2)
   22  SET BORDER 85:SET PAPER 0:
       SET INK 1:CLS
       !**** Menü kiirása ********

   23  PRINTAT 1,1:STRING$(64,"*")
   24  PRINTAT 3,1:"Jármü üzemelte
tési költségeinek nyilvántartása
 1 évre"
   26  PRINTAT 5,1:STRING$(64,"*")

   28  PRINTAT 7,28:"FUNKCIÓK"
   30  PRINTTAB(28);STRING$(8,"-")

   32  PRINTAT 10,2:"1. adatállomá
ny területének létrehozása"
   34  PRINTAT 12,2:"2. adatfelvét
el/módositás"
   36  PRINTAT 14,2:"3. költségszá
mitások"
   38  PRINTAT 16,2:"4. program vé
ge"
   40  A$=INKEY$:IF A$="" THEN 40
       !*** Funkció választás ****

   42  IF VAL(A$)<1 OR VAL(A$)>4
       THEN 40
   44  ON VAL(A$) GOSUB 50,70,240
   46  GOTO 22
```

```
 50 !**************************
    *   File-ok létrehozása   *
    *     (inicializálás)     *
    **************************

 51 CLS:FOR N=1 TO 12
 52 H$=STR$(N)
 54 GOSUB 1200:
    OPEN OUTPUT H$
 55 FOR M=1 TO 5
 56 PRINT#5:STRING$(44,"*") !
   ** 5 db rekord egy file-ban
      *-al feltöltve
 57 NEXT M
 58 CLOSE OUTPUT
 62 NEXT N:
    RETURN
 70 !**************************
    *  Rekordok felvétele és  *
    *   file-ok módosítása    *
    **************************

 71 CLS:INPUTPROMPT"kérem adja
meg a hónap sorszámát (1-12)":H$

 72 IF VAL(H$)<1 OR VAL(H$)>12
    THEN 70
 76 OPEN INPUT H$
 78 I=1:GOSUB1100:
    GOSUB 1250
 80 INPUT#5:Y$
 85 IF Y$(1:1)="*" THEN CLOSE
    INPUT:GOTO 120 ! a "*" a
    file-vég jele
100 T$(I)=Y$
102 D$=Y$(1:8):ME$=Y$(9:23):KM$
    =Y$(24:29):AH$=Y$(30:34):
    MN$=Y$(35:39):EG$=Y$(40:44)
   !** a rekord mezökre bontása
    ,listázás a képernyöre

105 PRINTTAB(2);D$;TAB(13);ME$;

110 PRINTTAB(30);KM$;TAB(38);
    AH$;TAB(45);MN$;TAB(52);EG$

115 I=I+1:GOTO 80
120 !* mezök beolvasása  ******
    PRINTTAB(2);:M=8:GOSUB 200:
    D$=X$
122 IF D$(1:1)="*" THEN 165
125 PRINTTAB(13);:M=15:
    GOSUB 200
130 ME$=X$:PRINTTAB(30);
135 M=6:GOSUB 200:KM$=X$:PRINT
    TAB(38);
140 M=5:GOSUB 200:AH$=X$:PRINT
    TAB(45);
145 M=5:GOSUB 200:MN$=X$:PRINT
    TAB(52);
150 M=5:GOSUB 200:EG$=X$:
    PRINT""
```

```
155 T$(I)=D$&ME$&KM$&AH$&MH$&
    EG$   !
    ** a rekord összeállitása
    a mezökböl
160 I=I+1:GOTO 120
165 T$(I)="*"
170 I=1:GOSUB 1200:
    OPEN OUTPUT H$
175 Y$=T$(I):IF Y$(1:1)="*"
    THEN 190

180 PRINT#5:Y$   !** a rekordok
    felirása a file-ba

185 I=I+1:GOTO 175
190 PRINT#5:STRING$(44,"*"):
    CLOSE OUTPUT:
    RETURN

200 !**************************
    * mezö beolvasó szubrutin *
    **************************

201 X$="":
    FOR N=1 TO M
205 A$=INKEY$:
    IF A$="" THEN 205
210 IF ORD(A$)=13 THEN 225
215 IF ORD(A$)=64 AND LEN(X$)>0
    THEN X$=X$(1:LEN(X$)-1):
    PRINT CHR$(8);:N=N-1:
    GOTO205   !** karakter visz-
    szalépés ***
220 X$=X$&A$:PRINTA$;:
    NEXT N
225 IF LEN(X$)<M THEN
    X$=X$&STRING$(M-LEN(X$),32)
   !** mezö feltöltése space-el

230 RETURN
240 !**************************
    *                        *
    *   Költségszámitás,adat-  *
    *   keresés,listázás       *
    **************************

241 CLS:PRINTAT 7,2:"1. Átlagfo
gyasztás adott idöszakban"
242 PRINTAT 9,2:"2. 1 km-re esö
üzemeltetési költség"
244 PRINTAT 11,2:"3. adatok ker
esése"
246 A$=INKEY$:IF A$="" THEN 246

248 IF VAL(A$)<1 OR VAL(A$)>3 T
HEN 246
250 ON VAL(A$) GOTO 255,420,600

254 !**************************
    * Átlagfogyasztás számit. *
    **************************

255 IF F1=1 THEN 260
257 GOSUB 1000   !** File-ok be-
    olvasása a memóriába ******
```

```
260 K=1:I=1
262 GOSUB 1100:S=0
265 Y$=T$(I)
267 IF Y$(1:1)="*" THEN V=I:
    GOTO 276
270 PRINT" ";Y$(1:8);TAB(13);
    Y$(9:23);TAB(30);Y$(24:29);
    TAB(38);Y$(30:34);TAB(45);
    Y$(35:39);TAB(52);Y$(40:44)
    :S=S+1:I=I+1
275 IF S<15 THEN 265
276 I=1  !**************************
         * intervallum kijelö- *
         * lése kurzorpozicio- *
         * nálással            *
         **************************

278 C=1
280 PRINTAT C+3,60:"<"
285 B$=INKEY$:IF B$="" THEN 285

290 IF ORD(B$)=102 AND C=1
    THEN 285
295 IF ORD(B$)=102 THEN PRINTAT
    C+3,60:" ":
    C=C-1:I=I-1:
    GOTO 280
300 IF ORD(B$)=108 AND C=S THEN
    I=I+1:GOTO 262
305 IF ORD(B$)=108 THEN PRINTAT
    C+3,60:" ":C=C+1:I=I+1:
    GOTO 280
310 IF ORD(B$)=13 THEN H(K)=I:
    K=K+1:GOTO 320
312 IF I=V THEN 340
315 GOTO 285
320 IF K<3 THEN 285
339 !**************************
    *    Számitás            *
    **************************

340 I=H(1)+1:UZ=0
345 Y$=T$(I):AH$=Y$(30:34):
    ME$=Y$(9:23)
350 IF ORD(AH$)=32 THEN 360
351 IF ME$(1:1)="ü" OR
    ME$(1:1)="Ü" THEN 355:
    ELSE 360
355 UZ=UZ+(VAL(AH$)/20)
360 I=I+1:
    IF I>H(2) THEN 370
365 GOTO 345
370 Y2$=T$(H(2)):Y1$=T$(H(1))
375 K1$=Y1$(24:29):
    K2$=Y2$(24:29)
380 UT=VAL(K2$)-VAL(K1$)
385 CLS:PRINT AT 3,2:"Megtett k
ilométerek száma:";TAB(30);UT
390 PRINT AT 5,2:"Felhasznált ö
sszes üzemanyag:";TAB(30);UZ
395 PRINT AT 10,2:"Átlagfogyasz
tás:";TAB(30);100*(UZ/UT)
400 GET:GOTO 22
```

```
 419  !**************************
      * 1 km-re esö üzemeltetési*
      * költség számitása 1 évre*
      **************************

 420  IF F1=1 THEN 430
 425  GOSUB 1000
 430  CLS:PRINT AT 12,20:"Kérem v
ärjon,most számolok"
 435  I=1:UZ=0:AS=0:JS=0:ES=0
 440  Y$=T$(I)
 445  IF ORD(Y$)=42 THEN 500
 450  AH$=Y$(30:34):MH$=Y$(35:39)
      :EG$=Y$(40:44)
 455  ME$=Y$(9:23)
 460  IF ORD(AH$)=32 THEN 475
 465  IF ME$(1:1)="ü" OR
      ME$(1:1)="Ü" THEN 471
 470  AS=AS+VAL(AH$):GOTO 475
 471  IF I>1 THEN UZ=UZ+VAL(AH$)
 475  IF ORD(MH$)=32 THEN 485
 480  JS=JS+VAL(MH$)
 485  IF ORD(EG$)=32 THEN 495
 490  ES=ES+VAL(MH$)
 495  I=I+1:GOTO 440
 500  Y2$=T$(I-1):
      U2=VAL(Y2$(24:29))
 505  Y1$=T$(1):
      U1=VAL(Y1$(24:29))
 510  CLS:PRINT AT 3,2:"km-ek szá
ma";TAB(25);":";U2-U1
 515  PRINT AT 5,2:"üzemanyag kts
g (Ft)";TAB(25);":";UZ
 525  PRINT AT 7,2:"anyag ktsg (F
t)";TAB(25);":";AS
 526  PRINT AT 9,2:"javitási ktsg
 (Ft)";TAB(25);":";JS
 527  PRINT AT 11,2:"egyéb ktsg (
Ft)";TAB(25);":";ES
 528  PRINT STRING$(64,"-")
 529  PRINT AT 13,2:"összesen ";T
AB(25);":";AS+UZ+JS+ES
 530  PRINT AT 15,2:"1 km-re esö
ktsg";TAB(25);":";(UZ+AS+JS+ES)/
(U2-U1)
 535  GET:GOTO 22
 600  !**************************
      * Rekordok keresése a     *
      * file-ban                *
      **************************

 601  CLS:IF F1=1 THEN 610
 605  PRINT AT 14,1:"Várjon,beolv
asom az adatokat":GOSUB 1000:CLS

 610  INPUT PROMPT"Kérem a kerese
ndö adatot:":K$:
      H1=LEN(K$)
 615  IF H1>15 THEN 610
 618  I=1
 620  Y$=T$(I):IF ORD(Y$)=42 THEN
      GET:
      RETURN
 625  ME$=Y$(9:23)
```

```
  630 P=1:P1=1  !********************
                *  string keresése*
                *  stringben      *
                ********************
  635 Q=1
  640 IF ME$(P1:P1)<>K$(Q:Q) THEN
      P=P+1:P1=P:GOTO 655
  645 IF Q=H1 THEN 660
  650 Q=Q+1:P1=P1+1:GOTO 640
  655 IF P>15-(H1-1) THEN 665 :
      ELSE GOTO 635
  660 PRINT Y$(1:8);"   ";
      Y$(9:23);"   ";Y$(24:29);
      "   ";Y$(30:34);"   ";
      Y$(35:39);"   ";Y$(40:44)
  665 I=I+1:GOTO 620
 1000 !**************************
      *    File-ok beolvasása a   *
      *    memóriába              *
      *    (szubrutin)            *
      **************************

 1001 F1=0
 1005 N=1:I=1
 1010 H$=STR$(N)
 1015 OPEN INPUT H$
 1020 INPUT #5:Y$:
      IF ORD(Y$)=42 THEN 1040
 1030 T$(I)=Y$
 1035 I=I+1:GOTO 1020
 1040 IF N=12 THEN T$(I)="*":
      F1=1:
      CLOSE INPUT:
      RETURN
 1045 CLOSE INPUT:
      N=N+1:GOTO 1010
 1100 !**************************
      *      Fejléc nyomtató       *
      *       szubrutin            *
      **************************

 1101 CLS
 1105 PRINT AT 2,2:"Dátum";
      TAB(13);"Megnevezés";
      TAB(30);
 1110 PRINT "Km";TAB(38);"Anyag";
      TAB(44);
 1115 PRINT"Munka";TAB(52);
      "Egyéb"
 1120 PRINT AT 3,2:STRING$(5,"-")
      ;TAB(13);STRING$(10,"-");
 1125 PRINTTAB(30);STRING$(2,"-")
      ;TAB(38);
 1130 PRINT STRING$(5,"-");
      TAB(44);STRING$(7,"-");
      TAB(52);STRING$(5,"-")
 1135 RETURN
 1200 CLS:PRINT AT 12,2:"Kérem in
ditsa el a magnót és nyomjon meg
 egy gombot"
 1205 A$=INKEY$:
      IF A$="" THEN 1205
 1210 RETURN
```

```
1250 FOR Q=1 TO 5
1260 PRINT"[                    ] [
     ][          ][          ][          ][
   ]"
1270 NEXT Q
1275 PRINT AT 4,2:"";:
     RETURN
```

Többfunkciós programok esetén tipikus megoldás az egyes funkciók ún. menüből való kiválasztása. Nyilvántartásunk főmenüje a 23-42-es sorokban található. A funkciók tömör, egy-két szavas leírását jelenítik meg a képernyőn, közülük a sorszámmal választható ki a kívánt. Például a 3. funkció kiválasztásakor a 240-es soron kezdődő szubrutint – modult – hívjuk. Itt újabb 3 funkciót tartalmazó almenüt találunk, amelyből szintén a sorszámmal választhatjuk ki a megfelelőt. A menük tehát, fő- és almenükből álló hierarchiát alkotnak egy rendszerben. A menüvezérlés megkönnyíti a program kezelését, áttekinthetőségét, amely egy adatfeldolgozó program esetén is nagyon fontos, hiszen azokat általában nem számítástechnikai szakemberek, hanem laikusok kezelik.

A továbbiakban elemezzük az egyes funkciókon végighaladva azok működését.

Az 1. funkció (50-62-es sor) tizenkét darab file-t ír fel a háttértárolóra. A file-ok nevei rendre az év hónapjainak sorszámai, karakteres típusra alakítva. Minden egyes file öt rekordot tartalmaz, amelyek a csillag karakterrel vannak feltöltve a maximális hosszra, 44 byte-ra. Ezzel egy előzetes helyfoglalást végeztünk. Az 1200-as sortól kezdődő szubrutin várakozik, míg a magnót elindítjuk és csak valamelyik gomb megnyomása után indul el a file kiírása.
Ne feledjük el ugyanis, hogy a BASIC akkor is elküldi a háttértárra az adatokat, ha az nem üzemkész. A file-ok között célszerű 5–10 fordulat helyet hagyni a szalagon, hogy a későbbi felírások alkalmával nehogy véletlenül ráírjunk egy másik file-ra is.

A második funkció lényege új rekordok felvitele az állományba: 70-230-as sorok, valamint az 1100-as, 1250-es, 1200-as szubrutinok. Egyszerre csak egy hónap rekordjait lehet felvenni, ill. a meglévő rekordokhoz hozzáírni.

Nagyon fontos egy adatfeldolgozó programnál, hogy az adatok a képernyőn rendezett formában, bizonylatszerűen jelenjenek meg. Az 1100-as szubrutin a képernyőre írja a fejlécet, amely egy rekord mezőinek a nevét tartalmazza. Ezalatt oszlopba rendezve kerülnek kiírásra a rekordok mezői.
A 80-as sorban addig olvassuk a file-t, amíg az első "üres", csillag karakterrel feltöltött rekordot el nem érjük. A rekordokat először mezőkre kell bontani — 102-es sor — és csak ezután lehet azokat a képernyőre listázni, ill. feldolgozni. A felbontásnál bevezetett változónevek az egész programban, azonos tartalmúak, elősegítendő a programlista könnyebb olvashatóságát:

D$ — dátum
ME$ — költség megnevezése
KM$ — km-óra állása
AN$ — anyagköltség (alkatrész ára)
MN$ — munkabérköltség
EG$ — egyéb költség
T$ — az összes rekordot tartalmazó tömb
Y$ — az aktuális rekord

A következő lépésben új rekordokat lehet a file-ba felvenni. A mezők beolvasását a 200-as szubrutin végzi el, amely általános célú beolvasó rutin. Bemenő paramétere a beolvasandó mező hossza. A mezőt karakterenként beolvassa a billentyűzetről és átadja az X$ segédváltozóba. A beolvasás vagy a maximális hosszig vagy a RETURN-ig tart. Gépelés közben a mező — karakter-visszalépéssel — javítható. A rekordok bevitelét a dátummező elejére begépelt csillag zárja.
A file felírását a 170-190-es sorok végzik. Kazettára írásnál ne felejtsük el a szalagot visszacsévélni a file elejére.

A harmadik funkció első alfunkciója végzi el az átlagfogyasztás számítását tetszőleges, általunk kijelölt tartományra. Az F1 jelző azt jelzi, hogy az összes rekord a tárban van, avagy sem. Ha nem, akkor az 1000-es szubrutin beolvassa az összes file minden rekordját a T$ tömbbe. Ezután következhet annak a két rekordnak a kiválasztása, amely

határolja a tartományt. A rekordokat kilistázzuk laponként a képernyőre — 1 lap 15 sor — és az általunk készített mutatóval (< jel) pozicionáltuk először az első, majd a második rekordra. A kiválasztást a RETURN billentyű leütésével végezzük.
A H tömb tartalmazza a kiválasztott rekordok T tömbön belüli sorszámát.
A számítást a 340-395-ös sorok utasításai végzik. Feltételezzük, hogy azoknak a rekordoknak, amelyek üzemanyagköltséget tartalmaznak, a megnevezés mezőjében "ü" vagy "üzemanyag" szöveg található.

A második alfunkció a 3. funkción belül az 1 km-re eső üzemeltetési költség számítását végzi. Ez a rész a 419-535-ös sorokban helyezkedik el. Itt is be kell olvasnunk az összes rekordot tárba. A teljes adatállományt végigolvassuk és eközben a rekordokból kiemelt költségeket költségnemenként halmozzuk az UZ, AS, JS, ES változókba, majd mindezeket külön-külön és együttesen is elosztva a megtett kilométerek számával megkapjuk a teljes költség 1 km-re eső részét, és költségnemenkénti bontásban is megkapjuk annak 1 km-re eső hányadát. A képernyőn csak a végeredmény jelenik meg.

Az utolsó funkcióban kikereshetjük az állományból az azonos költségmegnevezéseket tartalmazó rekordokat és kilistázhatjuk a képernyőre. Például kigyűjthetjük az összes olyan rekordot, amelynek megnevezés mezője az "olajcsere" szöveget tartalmazza. A program "lelke" a 630-655-ös sorokban elhelyezkedő utasítások. Egy tetszőleges karakterfüzérben (sztring) lévő másik füzért kereshetünk meg ezzel a programrésszel.
A rekordokat a megnevezés mező alapján keressük. Ez a programrész lehetővé teszi, hogy a rekordokat megtaláljuk akkor is, ha csak egy tetszőleges hosszúságú töredéket adunk meg a teljes megnevezésből. Persze a töredékeknek is egyértelműen kell azonosítania a rekordokat.

Ez a feladat kicsiben ugyan, de bemutatta az adatfeldolgozó programok sok jellegzetességét, s problémáinak egyfajta megoldásait is.

Faltörőjáték

Az utolsó feladat valószínűleg nem ismeretlen azoknak, akik ZX Spectrum számítógépen már játszottak, dolgoztak. A Faltörő eredetileg ezen a gépen futó játékprogram, vagyis az ötlet nem új, de a program TVC-re történő adaptációja könyvünk céljaihoz illeszkedik.

A játékot egy játékos játszhatja. A képernyő felső harmadában öt sor tégla van — 1 tégla két karakter méretű — ezeket kell lebontani egy labdával, amelyet ütővel irányíthatunk.
A labda a 8 főirányba mozoghat. A pálya oldalfalairól és a téglák fölötti oldalról is visszapattan, de az ütő alatti "földre" nem eshet le.
A labdával eltalált tégla "kiesik" a falból, törlődik a képernyőről és a játékos ezért pontokat kap. A pontszám attól függ, hogy a tégla melyik sorban helyezkedett el.
Ha sikerül lebontani a falat úgy, hogy még van a játékosnak "élete", azaz próbálkozási lehetősége, akkor új falat rajzol a program és lehet folytatni a bontást.

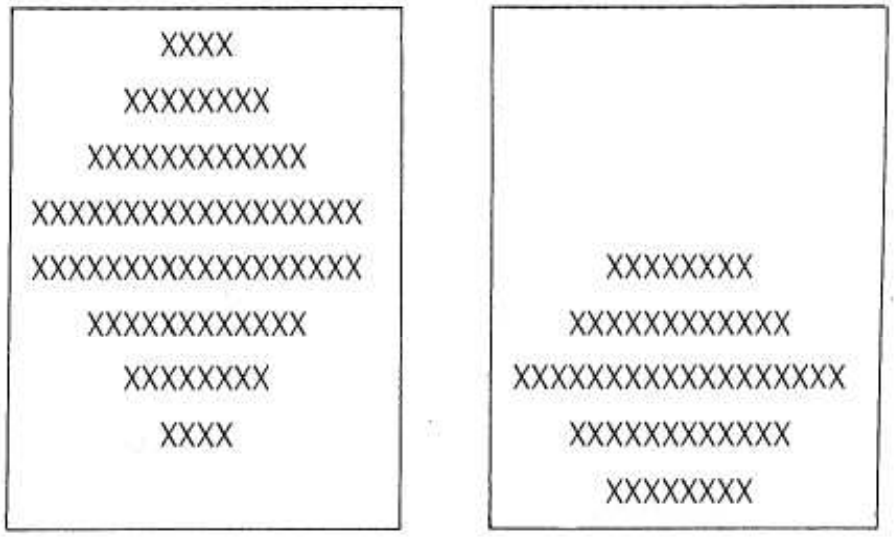
Az ütőt irányító billentyűk:

O — ütő jobbra
P — ütő balra
SHIFT — kétszeres ütősebesség

A feladat megoldásához első lépésként a fő funkciókat határozzuk meg. Megfigyelhető, hogy azonos típusú feladatok általában hasonló részfunkciókból épülnek fel. Visszatekintve a Bekerítőjátékra, párhuzam vonható közte és Faltörő között. Tehát felhasználói karaktereket kell definiálnunk, amelyekből a képet majd felépítjük. Vizsgálnunk kell a billentyűzetet — program vezérléséhez —, a találatokat és adminisztrálnunk kell azokat. A legfontosabb funkció természetesen a játékos lépéseinek végrehajtása és az azokra adott válasz.

Felhasználói karakterként definiáljuk a labdát, az ütőt és a téglákat. Az ábrákon az X jel egy-egy karaktermátrix-pontnak felel meg.

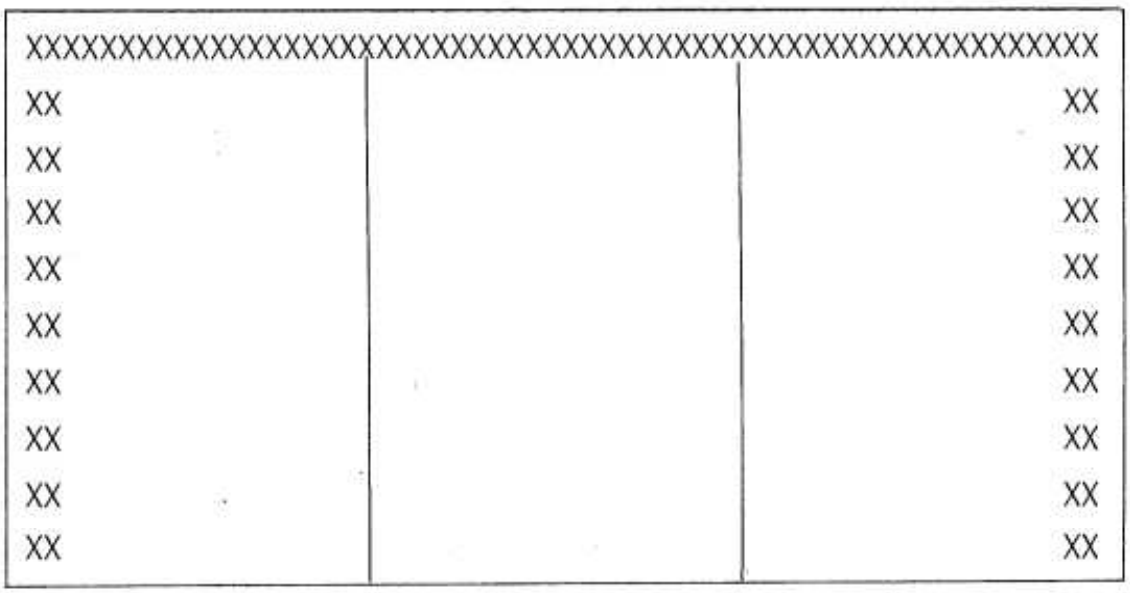
A labda:



normál labda CHR$(165)

labda ütközéskor CHR$(166)

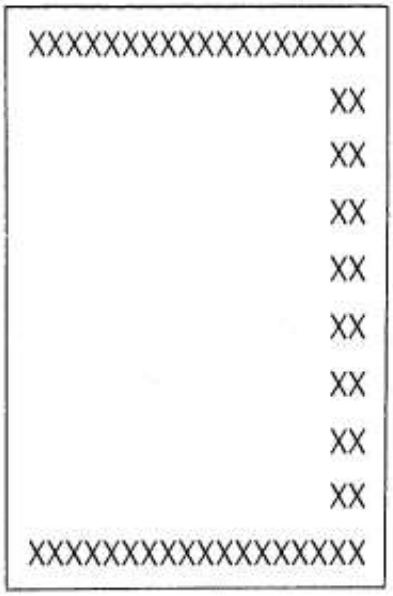
Az ütő:



CHR$(162) CHR$(163) CHR$(164)

A téglák:

CHR$(160)



CHR$(161)

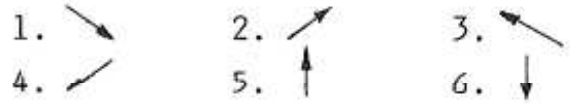
A BASIC programok talán legnagyobb hátránya a lassúsága. Már a Bekerítőnél is tapasztaltuk, hogy az ilyen típusú játékprogram esetén a legfőbb szempont a megfelelő sebesség elérése, mert a játszhatóság ettől függ. Ennek kell alárendelni minden mást: a strukturáltságot, a könnyű javíthatóságot stb.

Nem mindegy például, hogy a Faltörőben mekkora az ütő és a téglafal távolsága. Ha túl nagy, akkor a labda pályája az ütő és a fal vagy a játéktér oldala között túlságosan hosszúvá válik, a játékos csak viszonylag ritkán avatkozhat közbe, unalmas lesz a játék. Ha kicsi ez a távolság, bármilyen ügyes is a játékos, csak ritkán érheti el a visszapattanó labdát az ütővel, tehát nem "igazságos" a játékossal a program.

A megfelelő távolság kialakítására tehát ügyelnünk kell. Gyakorlati próbák alapján, a megfelelőnek tűnő elrendezés:

téglák sorai – 5,6,7,8,9
ütő sora – 20
"talaj" – 21,22,23,24

A labda a pályán hat irányban haladhat, ezeket az irányokat célszerű sorszámmal ellátnunk:



A labda mindaddig megtartja eredeti irányát, amíg téglának, a pálya szélének vagy az ütőnek nem ütközik. Ütközés után a labda új iránya függ az előző iránytól és ütővel való találkozás esetén attól is, hogy annak közepére, jobb vagy bal szélére esik rá.

Amennyiben a labda a téglafal és a pálya közé kerül, ott "pattogni" fog, magától is bontja a falat egészen addig, amíg egy résen ki nem esik onnan.

Az ütközések vizsgálata a Bekerítő programban már megismert térkép alapján történik.

A program megtervezése után tekintsük meg a pszeudokódját és a programlistát.

```
   üzemmód-, billentyűzet-, időbeállítás
   felhasználói karakterek definiálása
   a kezelés leírása a képernyőre
   a pálya felrajzolása
   FOR életek száma=6 TO 1 STEP -1
1: labda- és ütőkezdőhelyzet beállítás
2: labdamozgatás 1 helyzettel az aktuális irányba
   IF ütőmozgatás kérése
      THEN ütőmozgatás az új helyzetbe
   ENDIF
   IF labda helyzete = ütő helyzete
      THEN IF ütő bal széle
              THEN 1. irány
              ELSE IF ütő közepe
```

```
                        THEN 2. irány
                        ELSE 3. irány
                    ENDIF
                ENDIF
    ENDIF
IF pálya széle
    THEN oldal azonosítása
         új irány beállítása
    ELSE IF téglával ütközés
            THEN tégla törlése
                 pontszám növelése
                 új irány beállítása
                 IF nincs több tégla
                    THEN új pálya felrajzolása
                         GOTO 1
                 ENDIF
             ELSE IF leesett a labda
                     THEN GOTO 3
                     ELSE GOTO 2
                  ENDIF
          ENDIF
    ENDIF
3: NEXT életek száma
    END
```

```
 5  !*************************
    *      faltörő játék      *
    *************************

 6  DIM Z$(24,32)*1
10  GOSUB 500:T=0:TT=480
20  P=1:CLS:SET PAPER 2:
    SET INK 3
21  GOSUB 22:GOTO 29
22  M=5:GOSUB 700:M=6:GOSUB 750

23  M=7:GOSUB 700:M=8:
    GOSUB 750:M=9:GOSUB 700:!
    ** téglák rajzolása ***
24  FOR Y=5 TO 9:
    FOR X=1 TO 32 :!* térkép
25  Z$(Y,X)="*":
    NEXT X:NEXT Y
26  RETURN
28  !*************************
```

```
 29 SET PAPER 1:SET INK 3
 30 FOR Y=22 TO 24!**föld
 31 FOR X=1 TO 32
 32 PRINT #0,AT Y,X:CHR$(167);
 33 NEXT X:NEXT Y
 34 FOR X=1 TO 5 !**életek
 35 PRINT AT 24,X:CHR$(169);
 36 NEXT X
 46 U=1:V=1:SET INK 0:
    SET PAPER 1:I=5
 48 W=0:RANDOMIZE
 50 FOR R=1 TO 6
 52 M=9:N=3+INT(RND*27):A=15
 54 G=6:P=0
 56 PRINT#0,AT 21,1:
    STRING$(14,32);CHR$(162);
    CHR$(163);CHR$(164);
    STRING$(15,32);
 65 ON G GOSUB 100,120,140,160,
    180,200 !** pályák
 67 PRINT#0,ATU,V:" ":
    Z$(U,V)=" "
 71 PRINT#0,ATM,N:CHR$(165):
    U=M:V=N
 74 IF M=20 THEN PRINT#0,ATM,N:
    CHR$(166)
 80 A$=INKEY$:IF A$="o" THEN
    GOSUB 220
 81 IF A$="O" THEN GOSUB 224
 85 IF A$="p" THEN GOSUB 230
 86 IF A$="P" THEN GOSUB 234
 90 GOTO 65
 99 !**************************
    *  1. jobbra le          *
    **************************

100 IF M>20 THEN GOTO 240
101 IF M<20 THEN GOTO 106
102 IF T>=TT THEN TT=TT+480:
    GOSUB 22:W=0:GOTO 52
103 P=0:W=0:IF N=A+1 OR N=A+2
    THEN G=2:GOTO 120
104 IF N=A THEN G=5:GOTO 190
105 IF N=A-1 THEN G=3:GOTO 140
106 IF N>31 THEN G=4:GOTO 160
110 M=M+1:N=N+1
112 IF Z$(M,N)="*" THEN
    GOSUB 250:IF P=0 OR W=1
    THEN G=2
114 RETURN
119 !**************************
    * 2.  jobbra fel        *
    **************************

120 IF N>31 THEN G=3:GOTO 140
125 IF M<2 THEN W=1:G=1:
    GOTO 100
130 M=M-1:N=N+1
132 IF Z$(M,N)="*" THEN
    GOSUB 250:P=1:G=1+(2*W)
134 RETURN
```

```
139 !**************************
    *   3. balra fel         *
    **************************

140 IF M<2 THEN W=1:G=4:
    GOTO 160
145 IF N<2 THEN G=2:GOTO 120
150 M=M-1:N=N-1
152 IF Z$(M,N)="*" THEN
    GOSUB 250:P=1:G=4-(2*W)
154 RETURN
159 !**************************
    *   4. balra le          *
    **************************

160 IF M>20 THEN GOTO 240
161 IF M<20 THEN GOTO 166
162 IF T>=TT THEN TT=TT+480:
    GOSUB 22:W=0:GOTO 52
163 P=0:W=0:
    IF N=A+3 THEN G=2:GOTO 120
164 IF N=A+2 THEN G=5:GOTO 180
165 IF N=A OR N=A+1 THEN G=3:
    GOTO 140
166 IF N<2 THEN G=1:GOTO 100
170 M=M+1:N=N-1
172 IF Z$(M,N)="*" THEN
    GOSUB 250:IF P=0 OR W=1
    THEN G=3
174 RETURN
179 !**************************
    *   5.   fel             *
    **************************

180 IF M<2 THEN G=6:GOTO 212
190 M=M-1:IF Z$(M,N)="*" THEN
    GOSUB 250:G=6
195 RETURN
199 !**************************
    *    6.   le             *
    **************************

200 IF M>20 THEN 240
202 IF M<20 THEN 212
203 IF T>=TT THEN TT=TT+480:
    GOSUB 22:W=0:GOTO 52
204 P=0:W=0:
    IF N=A+2 THEN G=2:GOTO 120
206 IF N=A+1 THEN G=2:GOTO 120
210 IF N=A THEN G=3:GOTO 140
212 M=M+1:RETURN
219 !**************************

220 IF A<2 THEN RETURN
223 A=A-1:PRINT#0,AT21,A:
    U$&" ":RETURN
224 IF A<3 THEN 220
227 A=A-2:PRINT#0,AT21,A:
    U$&"  ":RETURN
230 IF A>29 THEN RETURN
232 A=A+1:PRINT#0,AT21,A-1:
    " "&U$:RETURN
234 IF A>28 THEN 230
236 A=A+2:PRINT#0,AT21,A-2:
    "  "&U$:RETURN
```

```
239 !**************************
240 RESTORE 241:
    FOR X=1 TO 6:READ K:
    SOUND; PITCH K,DURATION 8,
    VOLUME 3+2*X:
    NEXT X
241 DATA 3985,3874,3652,3208,
    2320,545
242 SET INK 3:PRINT AT 24,I:
    CHR$(167);:SET INK 0:
    I=I-1:NEXT R
243 GRAPHICS 16:
    PRINT AT 5,5:"Eredmény:":
    PRINT AT 10,4:T;"  pont":
    FOR X=0 TO 3000:NEXT X:
    GRAPHICS 4
244 PRINT AT 22,1:
    "Akar újra játszani? (i/n)"

245 A$=INKEY$:IF A$="" THEN 245

246 IF A$="i" OR A$="I" THEN 10

247 IF A$="n" OR A$="N" THEN EN
D
248 GOTO 245
249 !**************************

250 T=T+(10-M): !** pontszám
252 B=ABS(M-N)
254 Y=(INT(B/2))*2
256 IF B=Y AND N<32 THEN
    PRINT #0,AT M,N+1:" ";:
    Z$(M,N+1)=" ":T=T+(10-M):
    GOTO 260
258 IF B<>Y AND N>1 THEN
    PRINT #0,AT M,N-1:" ";:
    Z$(M,N-1)=" ":T=T+(10-M)
260 RETURN
499 !**************************

500 REM inicializalas
502 GRAPHICS 4
510 SET CHARACTER 160,255,128,
    128,128,128,128,128,128,128
    ,255
520 SET CHARACTER 161,255,1,1,1
    ,1,1,1,1,1,255
530 SET CHARACTER 162,255,255,
    255,255,128,128,128,128,128
    ,128
540 SET CHARACTER 163,255,255,
    255,255,0,0,0,0,0,0
550 SET CHARACTER 164,255,255,
    255,255,1,1,1,1,1,1
560 SET CHARACTER 165,0,24,60,
    126,255,255,126,60,24,0
570 SET CHARACTER 166,0,0,0,0,0
    ,60,126,255,126,60
575 SET CHARACTER 167,255,0,0,
    255,0,0,255,0,0,255
580 SET PALETTE 65,85,68,0:SET
    PAPER 1:SET BORDER 21
```

```
590 SET INK 2:CLS
595 U$=CHR$(162)&CHR$(163)&
    CHR$(164)
600 PRINT AT 2,4:
    "Tv-computer faltörö játék"
    :PRINT AT 5,5:"Kezelése:"
620 PRINT AT 7,5:
    "o - ütö balra"
630 PRINT AT 9,5:
    "p - ütö jobbra"
640 PRINT AT 11,5:
    "SHIFT - ütő gyorsitása"
650 PRINT AT 20,2:
  "Ha elolvasta,nyomjon meg egy
   gombot"
660 GET:RETURN
699 !************************

700 FOR N=1 TO 31 STEP 2
710 PRINT AT M,N:CHR$(160);
    CHR$(161);
720 NEXT N
730 RETURN
750 FOR N=1 TO 31 STEP 2
760 PRINT AT M,N:CHR$(161);
    CHR$(160);
770 NEXT N
780 RETURN
```

Az első mozzanat a felhasználói karakterek definiálása és a játékszabályok felírása a képernyőre (500-660-as sorok).
A pálya elkészítése több részből áll: a 22-23-as sorokban a sorszám paraméterrel hívjuk a 700-as és a 750-es szubrutinokat, amelyek a téglafalat rajzolják fel. Az egyik féltéglával kezdi a sort, a másik egész téglával. A 24-26-os sorokban a térképre is bejelöljük a téglák helyét a pályán.

Könnyebben nyomon követhető a program, ha ismerjük az egyes változók funkcióit. A fontosabbak:

A – az ütő bal szélének az aktuális pozíciója
G – a labda aktuális irányának a kódja
M – a labda y koordinátája
N – a labda x koordinátája
P – téglával ütközés jelzője
R – életek száma (ciklusváltozó)
T – az aktuális pontszám

U – a labda előző pozíciójának y koordinátája
V – a labda előző pozíciójának x koordinátája
W – a pálya szélének ütközés jelzője

A 65-212-es sorokban található utasítások alkotják a program magját. Az **ON** G **GOSUB** ... utasítás (65-ös sor) választja ki a labda irányának megfelelő szubrutint. A szubrutinok nem függetlenek egymástól, belső felépítésük sem azonos.

Kövessük végig a játék menetét egy darabon, ebből érthetjük meg a leggyorsabban a működést. Induljunk el attól a ponttól, amikor a játék kezdődik.

A labda a képernyő 9. sorából indul, lefelé esik, az x koordinátáját a véletlenszám-generátor állítja elő az 52-es sorban. Az ütő x koordinátája 15.
Mivel a G változó értéke 6, a 200-as szubrutinra kerül a vezérlés a 65-ös sorból.

Ha M nagyobb mint 20, akkor a labda leesett a földre, erre egy rövid dallam figyelmezteti a játékost (240-241-es sor) majd az "életek" száma eggyel csökken. A képernyő 24. sorában a hátralévő próbálkozások számának megfelelő számú labda van kirajzolva. Most ezek közül is törölnünk kell egyet (242-es sor). Az I változó az utolsó labda helyzetét mutatja a képernyőn.
Ha az M változó értéke 20, akkor meg kell vizsgálni, van-e még lebontandó tégla. A térképet vizsgálni e célból hosszadalmas lenne, tehát más módot kell találni. Tudjuk azt, hogy a téglák együttes pontértéke 480, vagyis ha az aktuális pontszám ezt eléri, akkor már nem lehet tégla a falból. Ha még van próbálkozási lehetősége, élete a játékosnak, akkor a program újra indul a pálya felrajzolásától és tovább lehet gyűjteni a pontokat.
Az M 20 feltétel teljesülése esetén a labda eshet tovább, a 212-es sorban 1-gyel nő M értéke majd viszatérünk a 67-es sorra. Ne feledjük, a G változó értéke maradt 6. Itt kiírjuk a labda karakterét az új helyzetébe és töröljük az előző helyéről.

Ezen a ponton van lehetősége a játékosnak az ütőmozgatásra. Négy típusa van: jobbra lépés egy vagy két karakterrel, és balra lépés egy vagy két karakterrel.
A 220-236-os sorokban helyezkednek el a mozgatást végző szubrutinok. Ezután a labda folytatja útját az előző lépésben meghatározott irányba.
A leírt ciklus a labda és az ütő találkozásáig működik.

A 204-210-es sorokban vizsgáljuk meg, hogy az ütő melyik részére esett a labda, ugyanis ettől függ az új iránya. Például, ha a bal szélére esik, akkor N = A, azaz balra 45 fokos szögben pattan tovább és a G változó értéke 3 lesz.

A téglával való ütközést minden mozgatást végző szubrutinban vizsgálnunk kell a térkép (Z$ tömb) m,n elemének tartalma alapján. Mivel egy tégla általában 2 karakter méretű — leszámítva a széleken elhelyezkedő féltéglákat —, ütközéskor valójában csak a tégla egyik felével találkozhat a labda. Ki kell tehát még számítanunk a tégla másik felének elhelyezkedését, ez az ütközés pontjától balra vagy jobbra egyaránt lehet. Ezt a műveletet és a pontozást a 250-es soron kezdődő szubrutin végzi.
Minden téglasorban más-más a téglák pontértéke, alulról felfelé haladva 2-től növekszik a pontszám kettesével.

Természetesen ez a program is, mint az eddigiek közül bármelyik, továbbfejleszthető. Érdekes megvizsgálni, hogyan változna meg a játék abban az esetben, ha a téglafal alakját megváltoztatnánk. Például, ha a fal és a pálya oldala között mindkét oldalon hézagot készítünk 2 karakter szélességben. Várhatóan ettől könnyebb lesz a játék, hiszen a labda könnyebben felmehet a fal mögé és automatikus bontásba kezdhet.

A további módosításokat, kiegészítéseket azonban már a Tisztelt Olvasóra bízzuk, remélve, hogy könyvünk elolvasása hozzájárult ahhoz, hogy a **TVC** számítógépét sokféle célra, hatékonyan használhassa.

AJÁNLOTT IRODALOM

BASIC programozási segédlet TV-COMPUTER, NOVOTRADE RT., Bp., 1987.

Kezelési útmutató TV-COMPUTER, NOVOTRADE RT., Bp., 1987.

Varga József: Személyi számítógépek kezelése és programozása, Terra, 1986.

Lőcs Gyula: BASIC és a kíváncsi, Tankönyvkiadó, Bp., 1985.

Lőcs Gyula: BASIC feladatgyűjtemény, Tankönyvkiadó, Bp., 1986.

Dr. Kocsis András: TV BASIC, SZÁMALK, Bp., 1984.

Lőcs - Sarkadi - Nagy - Sztankó: BASIC programozási nyelv, Műszaki Kiadó, Bp., 1976.

Ász-Basic programozás kártyával, NOVOTRADE RT., 1984.

Aszalós János - Erki Irén: Bevezetés a strukturált programozásba, KSH Oktatási Központ, 1980.

Major Zoltán - Valovics István: A BASIC feladatok tükrében, Tankönyvkiadó

Szlávi Péter - Zsakó László: Módszeres programozás, MK, 1986.

A BASIC programozás magasiskolája, NOVOTRADE - DATABAC, 1985.

Donald D. Spencer: Játékok BASIC nyelven, SZÁMALK, 1983.

Etüdök személyi számítógépre, Gondolat, 1984. (szerk. Vatisky Zsuzsa)

VIDEOTON TV-KAZETTÁK

Játékprogramok

(NOVOTRADE - GESTOR STÚDIÓ)

Kincskeresők

Akciójáték

Kincskeresés föld alatt, föld felett.

A bányásznak a föld mélyén az ezer veszéllyel fenyegető természettel, a bánya szellemével kell megküzdenie a kincsért. A szerencselovagnak a várban a szellemet kell kijátszania, hogy a létrákon keresztül a gyémántokat megszerezze. Ha ügyesen irányítod őket a botkormánnyal, Tiéd lesz a sok kincs. A játékot azoknak ajánljuk, akik szeretik a kalandot.

Ára: 331,– Ft

Hamburger

Ügyességi játék

Próbáld ki, milyen hamburgersütő lenne belőled. A pavilonod előtt sorban állnak a vevők. A grillsütő izzik, sülnek a húspogácsák. A joystick-kel irányíthatod a hamburger alkotórészeit – vízszintes mozgatásával jobbra-balra, előre tolásával megfordíthatod a zsemlét. A szóköz billentyűvel teheted a helyére. Ha elkészítettél egyet, már viszik is, és szaporodik a pénz. Ha nem igyekszel eléggé, az alapanyag szemétbe kerülhet, és ilyenkor veszteség ér.

Ne lepődj meg, ha a játék közben kedved támad néhány finom falatra, ezért legjobb, ha az elején magad mellé készíted.

Fogyókúrázóknak nem ajánljuk.

Ára: 392,– Ft

Törpe

Grafikus kalandjáték

A kíváncsi törpe nem tudott veszteg maradni a kertben. El is tévedt egy útvesztőben, ahol a levegő és az elemózsia egyre kevesebb lesz.

Segíts a törpének kijutni!

Kaphatsz szekercét, fogót, fáklyát, sőt számítógépet, kódkártyát és még más harci eszközt is.

Ha ügyesen bánsz velük, túljuthatsz az akadályokon.

Ügyes, kreatív játékosok kedvence ez a program.

Ára: 392,– Ft

TV-póker

Grafikus logikai játék

Az ismert kártyajáték számítógépes változata.

Fapofa helyett fa "képernyő". Nem vedeli a whiskyt, nem dohányzik. Egyébként igazi pókerjátékos. "Képernyőrebbenés" nélkül blöfföl. Csak a botkormány "szavát" érti, azt is csak akkor, ha a (return) billentyűvel megerősítettük szándékunkat.

Ára: 392,– Ft

Áttörés

Ügyességi játék

Izgalmas és veszélyes küldetésed repülővel behatolni és megsemmisíteni az ellenséges bolygó energiaellátó központját. Változatos terepen, barlangokban ezer veszély közepette tudod a végső célt elérni.

Az ellenség rakétákat és démonokat bevetve küzd; repülőd bombákkal és lézerágyúval van felszerelve.

A játék többféle nehézségi fokon is játszható.

Jó reflex és kitartás szükséges a játékhoz.

Ára: 429,– Ft

TV-STACK

Logikai játék

A TV-STACK típsú játékok méltán arattak nagy sikert a küldöldi játékszoftver-piacon az utóbbi időben. Most Ön is megismerkedhet ezzel az egyszerű, ugyanakkor mégis ötletes, ügyességet és helyzetfelismerést kívánó játékkal.

Egy verembe különféle formájú tárgyak esnek, melyeket jobbra-balra irányítani és saját tengelyük körül forgatni lehet.

Cél az, hogy úgy helyezzük el őket, hogy a verem alján hézagmentesen töltsék ki a helyet. Amint egy sor teljesen megtelt, el is tűnik. A játékos minden leérkezett tárgyért pontot kap. A magasról leejtett tárgyak több pontot érnek. További jutalompontok járnak a megtelt sorokért.

Az elért legmagasabb pontsztámokat a program tárolja, így érdekes versenyeket lehet rendezni a TV-STACK-kel.

Ára: 356,– Ft

Digipók

Akciójáték

A bokrok közül csapatostul törnek elő a gyilkos hangyák. Digipók nem veszíti el lélekjelenlétét, fegyveréhez kap és tüzel.

Hogy ki ez a hidegvérű fegyverbajnok? Ön, kedves játékosunk. Minél több ízeltlábút kell lelőni. Nem tehet mást. Jönnek a hangyák!

Ára: 429,– Ft

Windsurfer

Akciójáték

Veszélyes terroristák bombákat helyeztek el a szigetvilágon, 10 különböző szigeten. A bombák hamarosan felrobbannak és megsemmisítik az egész szigetvilágot. Te kaptad a kormány terroristaellenes bizottságától a feladatot, hogy hatástalanítsad a bombákat.

Az óceán ezen a vidéken annyira zátonyos, hogy csak windsurffel tudod a szigeteket megközelíteni. Szörfödet a beépített joystick-kel tudod irányítani.

Vigyázz, a szigeteket is korallzátony veszi körül!

A bomba hatástalanodik, ha eléred a szigetet – egy veszéllyel kevesebb.

Feladatod teljesítésére három életed van.

A színvonalas ügyességi játékok kedvelőinek ajánljuk.

Ára: 356,– Ft

Oktatóprogramok

(NOVOTRADE - OCTASOFT-STÚDIÓ)

Matematikai sorozatok

A program az általános iskolában tanított főbb sorozattípusok gyakoroltatására készült. A feladat: a képernyőn látható sorozat folytatása, amelynek feltétele az elemek közötti összefüggés felismerése. Ha ez túl nehéznek bizonyul, segítséget kapunk a géptől. A feladatok egyre nehezedő sorrendben következnek, sikeres megoldás esetén juthatunk el a következő nehézségi fokozathoz. Az eredményeket a gép pontozással és osztályzattal értékeli.

A program alkalmazása általános iskolában és a középiskola 1. osztályában ajánlott.

Ára: 356,– Ft

Boszorkánykonyha

Kémiai reakciók 7–8. osztály

A program az általános iskola 7. és 8. osztályában tanított összes kémiai reakciót gyakoroltatja a számítógéppel folytatott kalandjáték keretében.

A válaszadások során alkalom adódik kémiai alapfogalmak (elem, vegyület, bomlás, kettős kötés stb.) felelevenítésére, az anyagok tulajdonságainak (halmazállapot, szag, lángfestés, éghetőség stb.) felidézésére, a tanult folyamatok rendszerezésére.

A program kezelése és felépítése önmagát magyarázza, egyéni és csoportos, iskolai és otthoni tanulásra egyaránt alkalmas.

Ára: 454,– Ft

Nagy függvényábrázoló

A Nagy Függvényábrázoló program hasznos segítséget nyújt a függvénytani alapismeretek elsajátításához. Alkalmas függvények ábrázolására, egyismeretlenes egyenletek megoldásának bemutatására, valamint az alapvető függvénytranszformációk – zsugorítás, nyújtás, eltolás, tükrözés – szemléltetésére. A feladatok során időnként kicsit várakoznunk kell, amíg a program a képernyőn látottakat alposabban kiértékeli. A program nyolc alprogramot tartalmaz.

Ára: 392,– Ft

Optika

A program a fizika egy igen érdekes és látványos területét dolgozza fel. Elsősorban a 8. osztályos tananyag jobb megértését segíti, de felhasználható középiskolában, valamint szakköri munkában is.

A nyitókép után megjelenő menü tartalmazza a választható témaköröket.

Az első öt menüpont (1–5) a kijelölt témakörhöz tartozó ismeretközlő, magyarázó anyagrészeket tartalmazza.

A 6-os témakör a fénytani eszközök gyakorlati alkalmazását mutatja be példákon keresztül.

A 7-es témakörben ellenőrizhetjük tudásunkat. Három feladattípus közül választhatunk:

1. IGAZ–HAMIS TESZT: a gép állításokat közöl, amelyekről az I, illetve az N billentyű lenyomásával el kell döntenünk, hogy mgfelelnek-e a valóságnak vagy nem.

2. TOTO: feleletválasztásos teszt. Az 1, 2 vagy X billentyűk valamelyikével jelölhetjük ki, hogy a feltett kérdésre megadott három válasz közül melyiket tartjuk helyesnek.

3. KÉPSZERKESZTÉS: ez a legérdekesebb feladat, amelynek megoldása során a gyűjtőlencse és a homorú gömbtükör képalkotási szabályait lehet elsajátítani.

Ára: 356,– Ft

Bűvös négyzetek

A program a jól ismert matematikai problémát dolgozza fel sajátos módszerrel. A program használójának feladata egy bűvös négyzet kitöltése. Ha a négyzetet sikerült kitölteni, gratulációk után a játék újra indul. Ha a játékos végleg elakadt, feladhatja a játékot. Ekkor megjelenik a helyes megoldás, és a paraméterek beállítására nyílik ismét lehetőség.

A program remek lehetőséget nyújt bárkinek logikai képességei felmérésére, illetve fejlesztésére. Alkalmas az általános iskola felső tagozatában az összetettebb matematikai készségek fejlesztésére.

Ára: 276,– Ft

Keresd a térképen! Európa

A felhasználó feladata: Európa legfontosabb városait a képernyőn látható vaktérképre bejelölni. A program nemcsak név szerint kérdezi az egyes városokat, hanem úgy is, hogy a város történelméből, kulturális és művészeti életéből, valamint iparából vett adatok alapján kéri azonosításukat.

Ajánlott az általános iskolák földrajz- és történelemóráihoz.

Ára: 282,– Ft

A nyuszi olvasni tanít

Az egymásra épülő, de önállóan is felhasználható programokból álló sorozat a legkisebbeknek szeretne segíteni abban, hogy a betűkkel való ismerkedésben örömüket leljék, az olvasás élménnyé váljon. A programok lelkét a derűs, színes képek, vidám grafikák adják.

A sorozat részei:

1. Ismerkedés a betűkkel
2. Betűjáték
3. Szótagolás
4. Szavak és képek I.
5. Szavak és képek II.
6. A nyuszi mesél

A programok kezelése a lehető legegyszerűbb. Leírás tulajdonképpen alig szükséges: a grafikus képernyőről minden tudnivaló egyértelműen leolvasható.

A programok készítői a feladatok összeállításánál szigorúan figyelembe vették az általános iskola első osztályos követelményrendszerét.

Ára:	Programcsomag	1165,– Ft
	1–5 rész	307,– Ft
	6. rész	490,– Ft

Kalandozás a fizikában

A program teljesen új megközelítésben tárgyalja az általános iskolai fizika tananyag egészét.

A HELP billentyűvel előhívható összefoglalóban felsorolt fizikai fogalmak, törvények megértését lépésenkénti döntések sorozatán át, a kalandjáték módszerével éri el. A program használata végtelenül egyszerű, a döntéseknél az "i" vagy az "n" billentyű lenyomásával válaszolunk. A HELP billentyűvel rövid összefoglalót kérhetünk a választott jelenség vagy törvény logikai rendszeréről.

Ára: 571,– Ft

Geometriai transzformációk

Ez az első olyan TV-Computerre készült program, amely az általános iskola 4. osztályától a középiskola végéig egyaránt használható. Lehetőséget nyújt az összes geometriai transzformáció tulajdonságainak tanulmányozására. A mozgatandó alakzatot a program felhasználója rajzolhatja meg, majd kiválaszthatja a megfelelő transzformációt.

A geometriai hozzárendelések paraméterei a képernyő méretén belül grafikus módszerrel tetszőlegesen megadhatók. Ugyanarra az alakzatra egymás után akár több transzformációt is alkalmazhatunk. Ezáltal egészen egyszerű alakzatoktól bonyolult szerkesztésekig juthatunk el.

Ára: 380,– Ft

99,– Ft

VIDEOTON

ELEKTRONIKAI VÁLLALAT
SZÁMÍTÁSTECHNIKAI GYÁRA