HT-1080Z
ISKOLASZÁMÍTÓGÉP

HT-2080Z
SZÁMÍTÓGÉP

Programozás kezdőknek

HÍRADÁSTECHNIKA SZÖVETKEZET

HT-1080Z
ISKOLASZÁMÍTÓGÉP

HT-2080Z
SZÁMÍTÓGÉP

programozás kezdőknek

HÍRADÁSTECHNIKA SZÖVETKEZET

TARTALOM

ELŐSZÓ

Ez a kézikönyv olyanok számára készült, akik semmit vagy csak nagyon keveset tudnak a számítógépekről. Részletesen tárgyaljuk a főbb számítógép-programozási eljárásokat és technikákat, az ezekhez tartozó magyarázatokat pedig a valós életből vettük.

Ne törődjön vele, hogy Ön mit gondol a számítógépekről ebben a pillanatban, a könyv olvasása után majd látni fogja, hogy a számítógép nem más, mint egy ember alkotta nagyon hatékony berendezés. Rájön arra is, hogy a programozás sem több, mint olyan, utasításokká szervezett logikai lépések sorozata, amelyeket a gép megért. A BASIC kézikönyv segítséget nyújt ahhoz, hogy ezt a számítógépet kívánsága szerint használhassa.

BEVEZETÉS

Mi a számítógép-programozás?

A programozás három alaplépésből áll:

1. Elemezzük a megoldandó feladatot
2. Fordítsuk le ezt a számítógép nyelvére
3. Oldassuk meg a feladatot a géppel

Egy probléma megoldhatóságát a következő tényezők korlátozzák:

1. A programozó problémaelemző képessége
2. A használt programnyelv adottságai
3. A számítógép teljesítőképessége és lehetőségei

1. A probléma és elemzése

Hogy pontosabban fogalmazzunk, például, ha Ön semmit sem tud az űrrakétákról, nagyon valószínűtlen, hogy olyan űrrakéta-szimuláló programot tud írni, amely értelmes egy szakértő szemével, hacsak egy szakértő nem elég türelmes ahhoz, hogy szabad idejében elmagyarázzon minden tényezőt és változót, ami az űrrakétánál előfordul!

2. A programnyelv

Egy programnyelv funkciói és jellemzői rávilágítanak a képességére. Van olyan nyelv, amit üzleti célokra szántak, olyan, amit a tudományos számításoknál használnak, s vannak általános programnyelvek. Például, ebben a számítógépben lévő bővített BASIC nyelv egyike a rendelkezésre álló legismertebb számítógépnyelveknek.

3. A számítógép

Végezetül nagyon fontos magának a számítógépnek a felépítése és sebessége. Ha az Ön számítógépe kétszer olyan gyors, mint a többieké, akkor ez azt jelenti, hogy a feladatát feleannyi idő alatt végzi el, mint a többiek. Továbbá, ha a számítógép memóriakapacitása viszonylag nagy, akkor hosszabb, hatékonyabb programot tud írni.

Tegyük fel, hogy a következő problémát akarja megoldani:

1. Példa

1. Dávidnak 50 Ft van a zsebében
2. Buszjegyet vesz, ára 1,50 Ft
3. Vesz egy fagylaltot 1,50 Ft-ért
4. Játszik öt játékot egy játékautomatán, játékonként 2 Ft-ért.
5. Vesz egy lottót 5 Ft-ért
6. Vesz egy újságot 8 Ft-ért

Az első probléma: mennyi pénze van Dávidnak, amikor hazaér?

Ha a fenti eseményeket megfogalmazzuk BASIC nyelven, akkor a következő programot írhatjuk:

1. program

```
10 D = 50       : REM  1. DAVIDNAK VAN 50 FT-JA
20 D = D – 1.5  : REM  2. MINUSZ A BUSZJEGY ARA 1,50 FT
30 D = D – 1.5  : REM  3. MINUSZ A FAGYLALT ARA 1,50 FT
40 D = D – 2*5  : REM  4. MINUSZ 5 JATEK AZ AUTOMATAN, EGYENKENT 2 FT
50 D = D – 5    : REM  5. MINUSZ A LOTTO ARA 5 FT
60 D = D – 8    : REM  6. MINUSZ AZ UJSAG ARA 8 FT
70 PRINT "DAVIDNAL VAN"; D; "FT": REM A DAVIDNAL LEVO OSSZEG KIIRASA
80 END          : REM  A PROGRAM VEGE
```

A programot bármikor futtatva, az eredmény a következő lesz:
DAVIDNAL VAN 24 FT

Valójában az 1. program az alábbira redukálható, ámbár ezt sokkal nehezebb követni:

2. program

```
10 PRINT "DAVIDNAK VAN"; 50–1.5–1.5–2*5–5–8; "FT"
20 END
```

A számítógép működése nem korlátozódik a fenti programban leírt probléma megoldására. Ez a számítás elvégeztethető egy kalkulátorral, vagy akár fejben is. A következő néhány fejezetben bemutatunk több programozási eljárást, nagyteljesítményű programokkal szemléltetve, amelyek képesek megoldani néhány valóban érdekes és sokkal nehezebb feladatot.

PARANCSOK ÉS SZÖVEGSZERKESZTÉS

A parancsok fő célja az, hogy a számítógéppel külső vagy belső működéseket hajtasson végre.

Amikor a READY üzenet megjelenik a képernyőn (ld. a Használati útmutatót a rendszer bekapcsolásáról), akkor a gép "parancs" üzemmódban várja az Ön parancsát. Egészen addig vár, amíg Ön be nem billentyűz egy utasítást!

A megengedett parancsok közül néhány:

AUTO	a sorokat automatikusan számozza
DELETE	programsor(oka)t töröl
EDIT	programsor szerkesztését teszi lehetővé
LIST	kilistázza a beírt programot
RUN	végrehajtja a programot
NEW	töröl egy programot

Most próbáljuk meg bebillentyűzni a következő sorokat:

Példánkban az aláhúzás nélküli karaktereket a számítógép automatikusan jeleníti meg, az aláhúzott karaktereket nekünk kell begépelni, míg a [ ] *-el jelölt karaktercsoportok mindegyike egy nyomógombot jelöl.*

3. program

```
READY
> AUTO        [NEW LINE]
10 PRINT "ELSO PROGRAMOM" [NEW LINE]
20 FOR A = 1 TO 10 [NEW LINE]
30 PRINT A, A+A [NEW LINE]
40 NEXT A [NEW LINE]
50 END [NEW LINE]
60 _ [BREAK]
>
```

Ennél a pontnál ellenőrizzük, hogy nem követtünk-e el gépelési hibát. Például, ha a 30-as sorban hibáztunk, akkor újra be kell írni az aláhúzott részt:

```
READY
> 30 PRINT A, A * A [NEW LINE]
```

Ekkor ez a javított sor kerül az előző 30-as sor helyett a programba.
Ha nem kívánunk automatikus sorszámozást használni, parancs üzemmódban a > jel után közvetlenül írhatjuk az általunk sorszámozott programsorokat, tetszőleges sorrendben (végrehajtva a sorszámok növekvő sorrendjében lesznek). Ha azonos sorszámmal írunk több sort, mindig az utolsónak beírt az érvényes.
A következő fejezetben tárgyaljuk az ennél hatékonyabb szövegszerkesztési eljárást.

Ha mindennel elkészültünk, indíthatjuk a programot. Nézzük, mi történik!

```
READY
> RUN      [NEW LINE]
ELSO PROGRAMOM
1             1
2             4
3             9
4             16
5             25
6             36
7             49
8             64
9             81
10            100
READY
>
```

Az egész program során két parancsot használtunk, az AUTO-t és a RUN-t.

Most írjuk be a LIST parancsot és nyomjuk le a [NEW LINE] gombot.

```
READY
> LIST          NEW LINE
10 PRINT "ELSO PROGRAMOM"
20 FOR A=1 TO 10
30 PRINT A, A * A
40 NEXT A
50 END
READY
>
```

A bevitt program a LIST parancs hatására megjelenik a képernyőn.

A további parancsok részletes leírását a BASIC kézikönyv tartalmazza.

SZÖVEGSZERKESZTÉS

Hibákat mindig elkövethetünk a program írása közben, de az mindaddig nem baj, amíg ezeket fel tudjuk deríteni és gyorsan ki tudjuk javítani.

A BASIC szövegszerkesztője lehetőséget ad Önnek arra, hogy az előzőleg begépelt programszakaszon nagy hatékonysággal tudjon módosításokat, javításokat elvégezni.

Néhány a szerkesztési lehetőségek közül:

- L (listázás)
- I (beszúrás)
- C (csere)
- D (törlés)

A szövegszerkesztés teljes leírását a BASIC kézikönyv 2. fejezetében találja meg.

Most térjünk vissza a *3. programra*!

```
READY
> LIST        [NEW LINE]
10 PRINT "ELSO PROGRAMOM"
20 FOR A=1 TO 10
30 PRINT A, A * A
40 NEXT A
50 END
READY
>
```

Tegyük fel, hogy a szöveg 30-as sorában az A * A-t A * 2-re kívánja megváltoztatni.

Mindössze a következőket kell tennie:

```
> READY
> EDIT 30 [NEW LINE]
30 _
```

Most nyomja meg az L billentyűt, mire a bevitt sor megjelenik a képernyőn:

```
30 PRINT A, A * A
30 _
```

Egyszer megnyomva a [SPACE] billentyűt, a kurzor egy pozícióval jobbra lép. Most vigyük a kurzort a 12. pozícióba.

```
30 PRINT A, A * _
```

Nyomjuk le a [C] majd a 2 billentyűt:

```
30 PRINT A, A * [C] 2
```

Ha egyszer megnyomja a NEW LINE billentyűt, a gép minden eddig végrehajtott változtatást rögzít és visszatér a "parancs" üzemmódba.

```
30 PRINT A, A * 2  NEW LINE
> _
```

Ha most elindítjuk a programot, a végeredmény A * 2 lesz A^2 helyett (A értéke 1-től 10-ig változik).

Listázzuk ki még egyszer a programot!

```
> LIST
10 PRINT "ELSO PROGRAMOM"
20 FOR A=1 TO 10
30 PRINT A, A * 2
40 NEXT A
50 END
READY
>
```

Változtassuk meg egy kicsit a 10-es sort!

```
> EDIT 10  NEW LINE
10  L
```

A 10-es sor kiíródik a képernyőre:

```
10 PRINT "ELSO PROGRAMOM"
10 _
```

Most nyomjuk meg a 7-es, majd a SPACE billentyűt!

```
10 PRINT "ELSO PROGRAMOM"
10  7  SPACE
```

A képernyőn ez látszik:

```
10 PRINT "ELSO PROGRAMOM"
10 PRINT "_
```

[I]-t, majd két *-ot billentyűzzünk be!

```
10 PRINT "ELSO PROGRAMOM"
10 PRINT "I ** _
```

Ha most megnyomjuk a [NEW LINE] billentyűt, a képernyőn ez jelenik meg:

```
10 PRINT "** ELSO PROGRAMOM"
```

Most már Önnek is van némi elképzelése a szövegszerkesztésről. Tegyük fel, hogy nem akarja másokkal tudatni, hogy ez az Ön első programja, tehát ismét változtatni akar a programon!

```
> EDIT 10 [NEW LINE]
10 [L]
```

A képernyőn ez látszik:

```
10 PRINT "** ELSO PROGRAMOM"
10 _
```

Vigyük a kurzort a 11. pozícióba!

```
10 PRINT "**
```

Az 5-ös, majd a D billentyűt nyomjuk meg!

```
10 PRINT "** !ELSO !
```

A két felkiáltójel között lévő karakterek törlődni fognak.

Nyomjuk meg a NEW LINE billentyűt!

```
10 PRINT "** !ELSO ! PROGRAMOM"
> _
```

Ismét "parancs" üzemmódban vagyunk, így ki lehet listáztatni a 10-es sort:

```
> LIST 10 NEW LINE
10 PRINT "** PROGRAMOM"
```

Ez azt jelenti, hogy a program futtatásakor a fejléc "** ELSO PROGRAMOM" helyett "** PROGRAMOM" lesz.

Következő lépésként olvassa el a BASIC kézikönyvet és ismerkedjen meg az egyéb parancsokkal és szerkesztési eljárásokkal.

BASIC PROGRAMOZÁS

Ebben a fejezetben arról lesz szó, hogy miként elemezzünk egy problémát és hogyan oldjuk meg a BASIC segítségével.

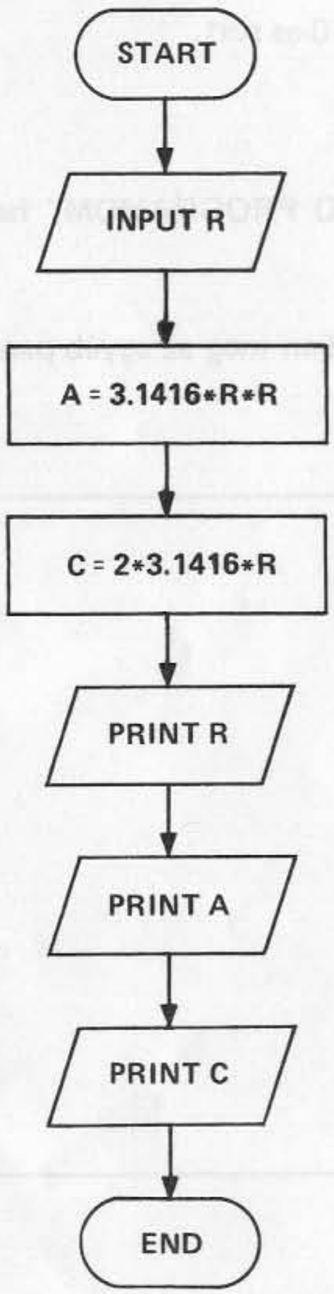
Mielőtt tovább mennénk, kérjük tanulmányozza át az 1. példát a bevezetésben.

Most próbáljuk meg analizálni azt az eljárást, amelynek segítségével ki tudjuk számolni egy R sugarú kör területét és kerületét.

A kör területét az $A = \pi \cdot R^2$ képlet, kerületét pedig a $C = 2\pi \cdot R$ összefüggés segítségével számíthatjuk ki.

Az ábrán egy ún. folyamatábra látható. A logikai sorrendet a nyilak mutatják. A START és az END a program kezdő és befejező pontját mutatja. A többi művelet a billentyűzetről történő bevitelt, a képernyőre való kiírást és értékadást jelöl.

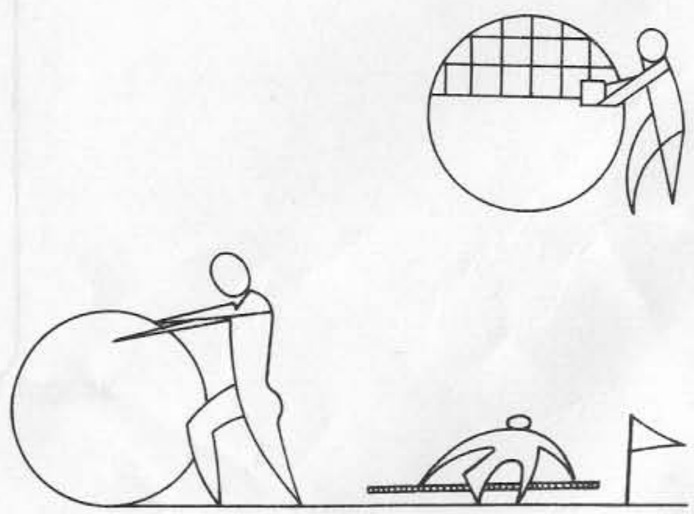
A folyamatábrán látható, hogy bemenő adatként az R szerepel. Az R-rel mint sugárral a program kiszámítja a területet ($\pi \cdot R^2$) és a kerületet ($2\pi \cdot R$). Nyomtatásra kerül a sugár, a terület és a kerület értéke.

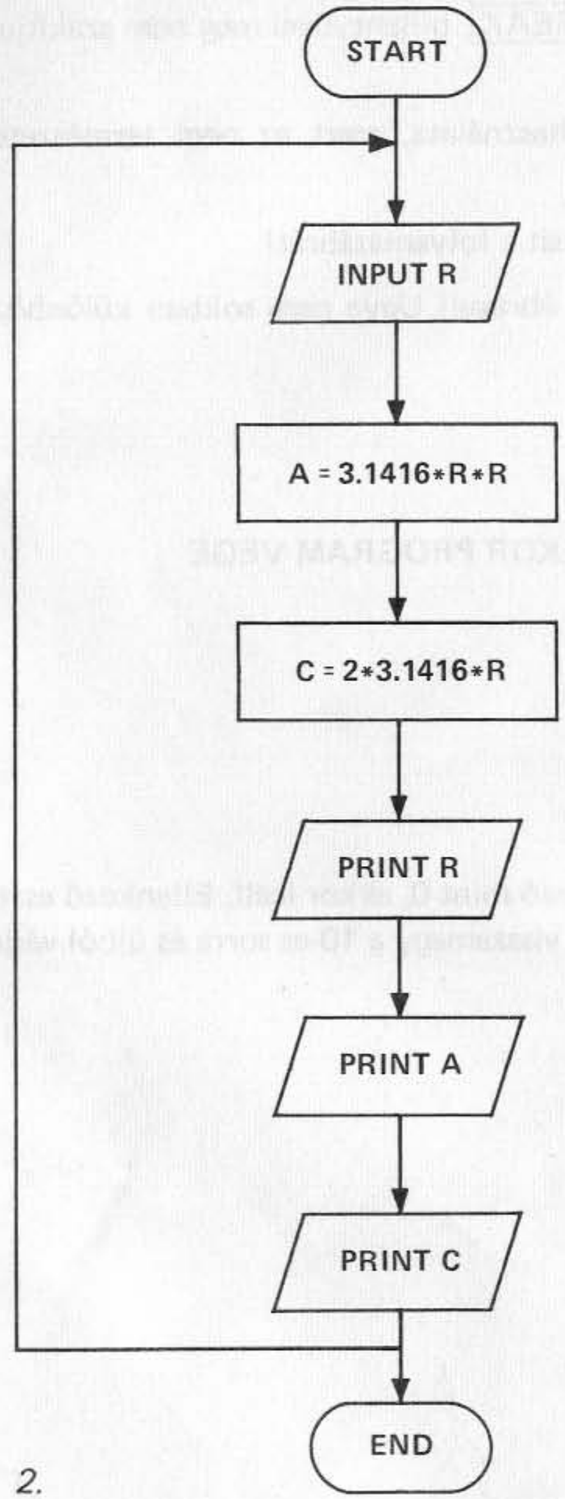


1.

Most fogalmazzuk meg a problémát BASIC nyelven és a programot gépeljük be a számítógépbe.

```
10 INPUT "A SUGAR ERTEKE"; R
20 A = 3.1416 * R * R
30 C = 2 * 3.1416 * R
40 PRINT "A SUGAR:"; R
50 PRINT "A TERULET:"; A
60 PRINT "A KERULET:"; C
70 END
READY
> RUN
```





2.

Példánkban adjunk meg 5-öt sugárnak.

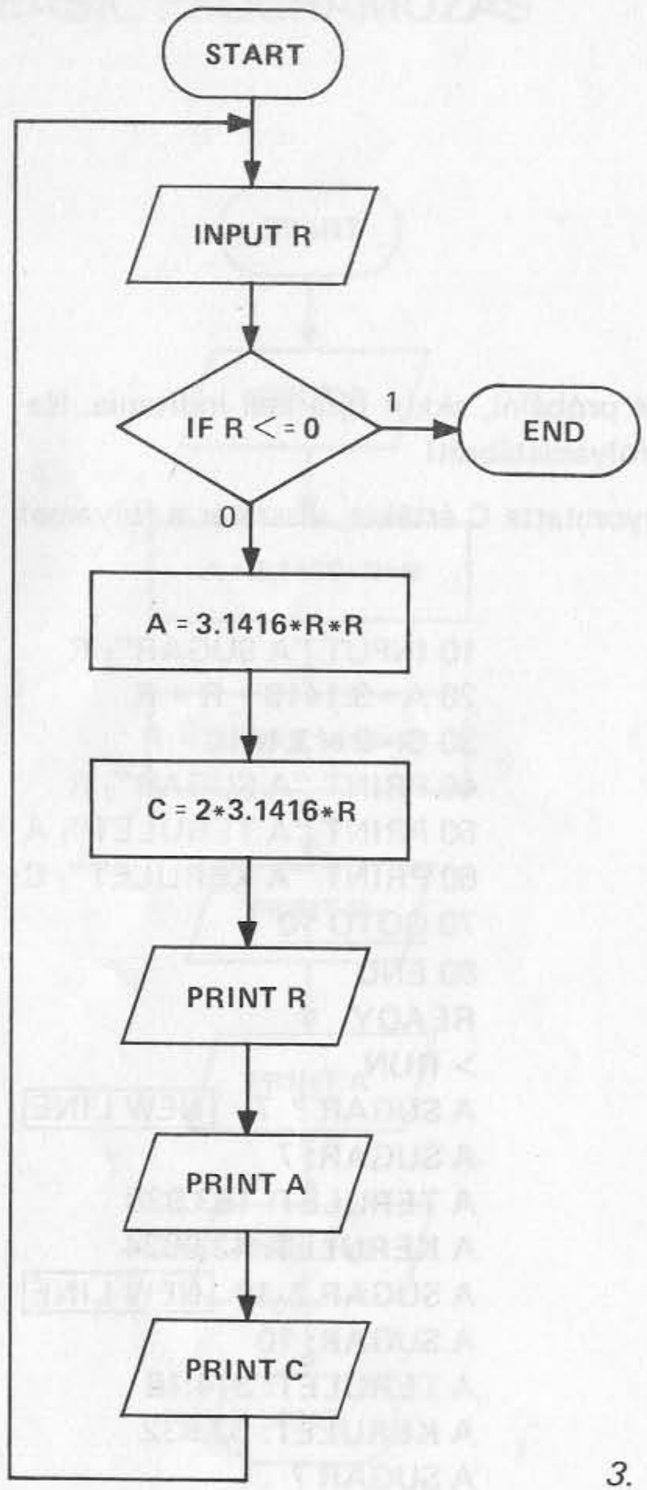
```
A SUGAR ERTEKE ? 5 [NEW LINE]
A SUGAR: 5
A TERULET: 78.54
A KERULET: 31.416
READY
>
```

Ha a programot egy másik sugár értékével is ki szeretné próbálni, akkor újra kell indítania. Nagyon kényelmetlen ugye? Változtassuk meg egy kissé a folyamatábrát!

Ahogyan a folyamatábra mutatja, ha a számítógép kinyomtatta C értékét, visszatér a folyamat elejére és újra bekéri R értékét és így tovább.

Ismét fordítsuk le a folyamatábrát BASIC nyelvre.

```
10 INPUT "A SUGAR"; R
20 A = 3.1416 * R * R
30 C = 2 * 3.1416 * R
40 PRINT "A SUGAR"; R
50 PRINT "A TERULET"; A
60 PRINT "A KERULET"; C
70 GOTO 10
80 END
READY
> RUN
A SUGAR ? 7 [NEW LINE]
A SUGAR: 7
A TERULET: 153.938
A KERULET: 43.9824
A SUGAR ? 10 [NEW LINE]
A SUGAR: 10
A TERULET: 314.16
A KERULET: 62.832
A SUGAR ?
```



A számítógép addig ismétli a teljes folyamatot, amíg a BREAK billentyűvel meg nem szakítjuk a program további végrehajtását.

(Nem jó programozási gyakorlat a BREAK billentyű használata, mert ez nem természetes program-megállítást jelent.)

Csináljuk valahogy logikusabban! Ismét módosítsuk egy kicsit a folyamatábrát!

Ne essen kétségbe! Hasonlítsa össze a 3. ábrát az 1. és 2. ábrával! Ugye nem sokban különböznek?

A BASIC program a következő:

```
10 INPUT "A SUGAR"; R
15 IF R < = O THEN END  :REM HA R < Ø, AKKOR PROGRAM VEGE
20 A = 3.1416 * R * R
30 C = 2 * 3.1416 * R
40 PRINT "A SUGAR"; R
50 PRINT "A TERULET"; A
60 PRINT "A KERULET"; C
70 GOTO 10
```

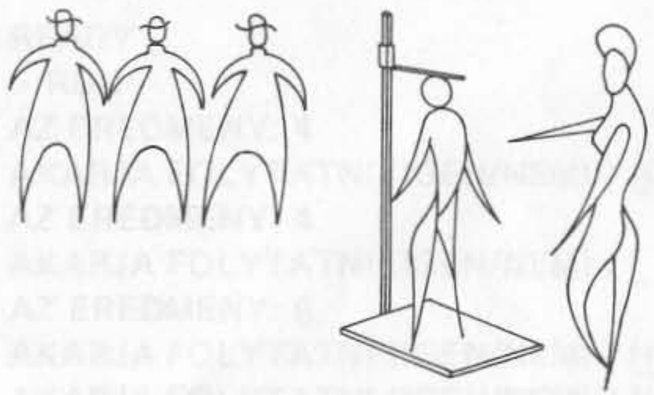
Ez a program beolvassa R értékét, de ha R kisebb vagy egyenlő mint 0, akkor leáll. Ellenkező esetben kiszámítja és kinyomtatja a kör területét és kerületét, visszamegy a 10-es sorra és újból várja R értékét.

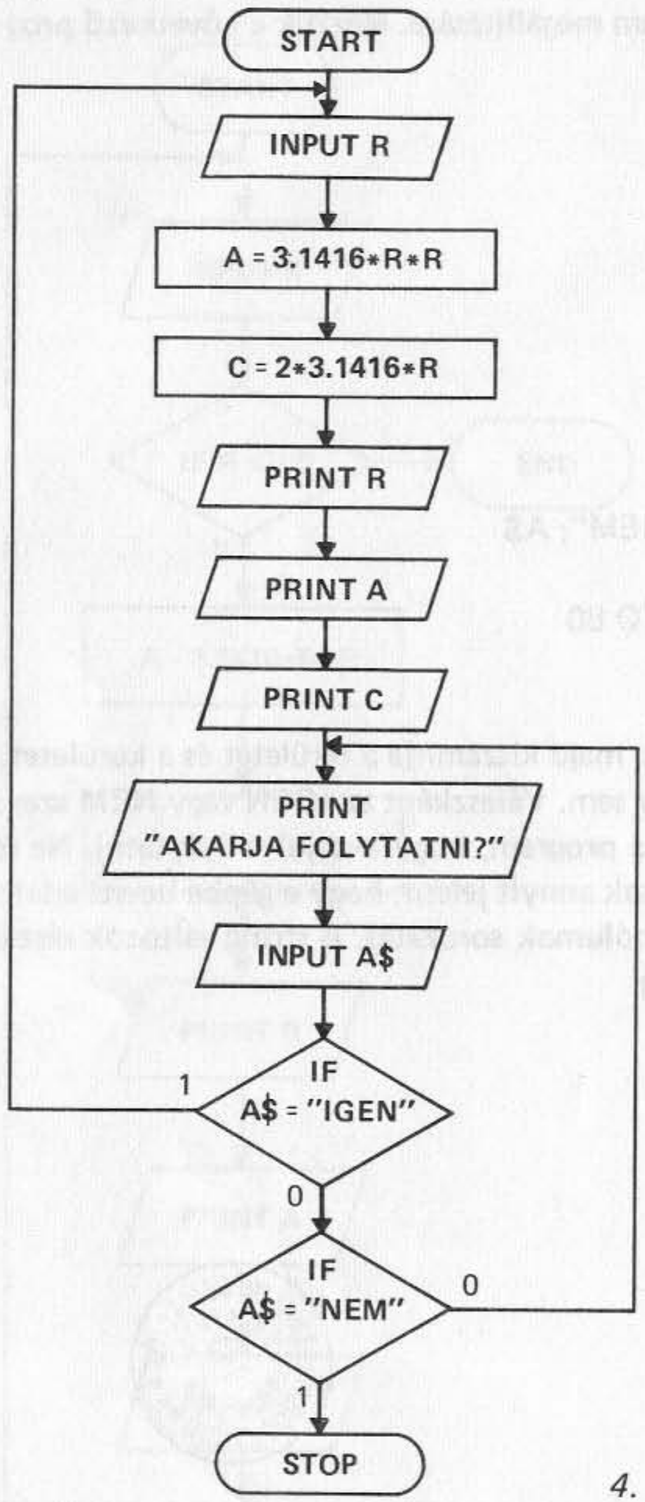
3.

Számos egyéb emberközeli módszer létezik egy program megállítására. Nézzük a következő programot!

```
10 INPUT "A SUGAR"; R
20 A = 3.1416 * R * R
30 C = 2 * 3.1416 * R
40 PRINT "A SUGAR:"; R
50 PRINT "A TERULET:"; A
60 PRINT "A KERULET:"; C
70 PRINT
80 PRINT "AKARJA FOLYTATNI?"
90 INPUT "IRJA BE HOGY IGEN VAGY NEM"; A$
100 IF AS = "IGEN" THEN 10 ELSE 110
110 IF AS = "NEM" THEN END ELSE GOTO 80
```

Hasonlóan az előbbihez, a program kéri a sugár értékét, majd kiszámítja a területet és a kerületet. Utána megkérdezi, hogy Ön akarja-e folytatni azt vagy sem. Válaszként az IGEN vagy NEM szavakat kell beírni. Ha mást válaszolunk, újra megkérdezi a program, hogy akarjuk-e folytatni. Ne rémüljünk meg az $ jeltől az A$-nél a 90-es sorban! Ez csak annyit jelent, hogy a gépbe bevitt adatot string-ként kell kezelni (a string karakterek vagy szimbólumok sorozata). A string változók részletesebb magyarázatát a BASIC kézikönyvben találja meg.





4.

Most próbálja megérteni a következő folyamatábrát.

Próbálja meg lefuttatni a programot:

```
READY
> RUN
A SUGAR ? 8
A SUGAR: 8
A TERULET: 201.062
A KERULET: 50.2636
AKARJA FOLYTATNI?
IRJA BE HOGY IGEN VAGY NEM? IGEN
A SUGAR ? 12
A SUGAR: 12
A TERULET: 452.39
A KERULET: 75.3984
AKARJA FOLYTATNI?
IRJA BE HOGY IGEN VAGY NEM? N
AKARJA FOLYTATNI?
IRJA BE HOGY IGEN VAGY NEM? NEM
READY
>
```

Most már látja, hogy egy egyszerű utasítás mennyire megváltoztatja egy program képességeit? Az Ön BASIC kézikönyvében sok ilyen hatékony utasítás található.

TOVÁBBI TUDNIVALÓK A PROGRAMOZÁSRÓL

Az előző fejezetben már bemutattuk az alapvető programozási elveket, ebben a fejezetben továbblépünk, és megismerkedünk más hasznos eljárásokkal is.

Játszott-e már számítógépes játékot valaha? Ha igen, akkor tudja, hogy hogyan keveri a kártyákat a gép huszonegyezésnél, vagy hogyan rázza a kockát, hogy meghatározza az Ön következő lépését.

Az Ön számítógépében van egy beépített függvény — RND(n) — amely véletlen számokat generál. *(Megjegyzés: n a generálandó számok felső határa. A részleteket nézze meg a beépített függvénynél a BASIC kézikönyvben!)*

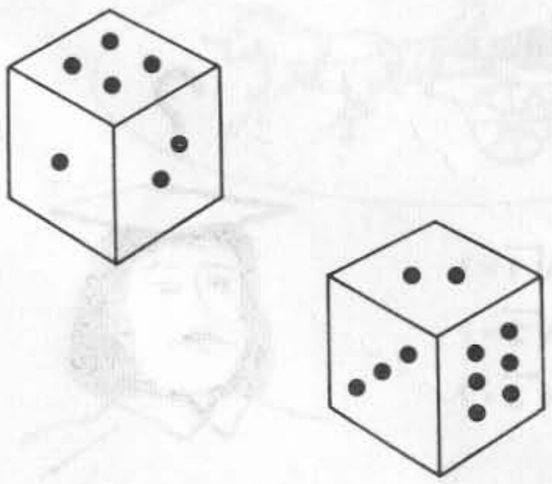
Az RND(n) függvény segítségével a számítógép automatikusan és véletlenszerűen keveri a kártyákat, dobja a kockát, stb.

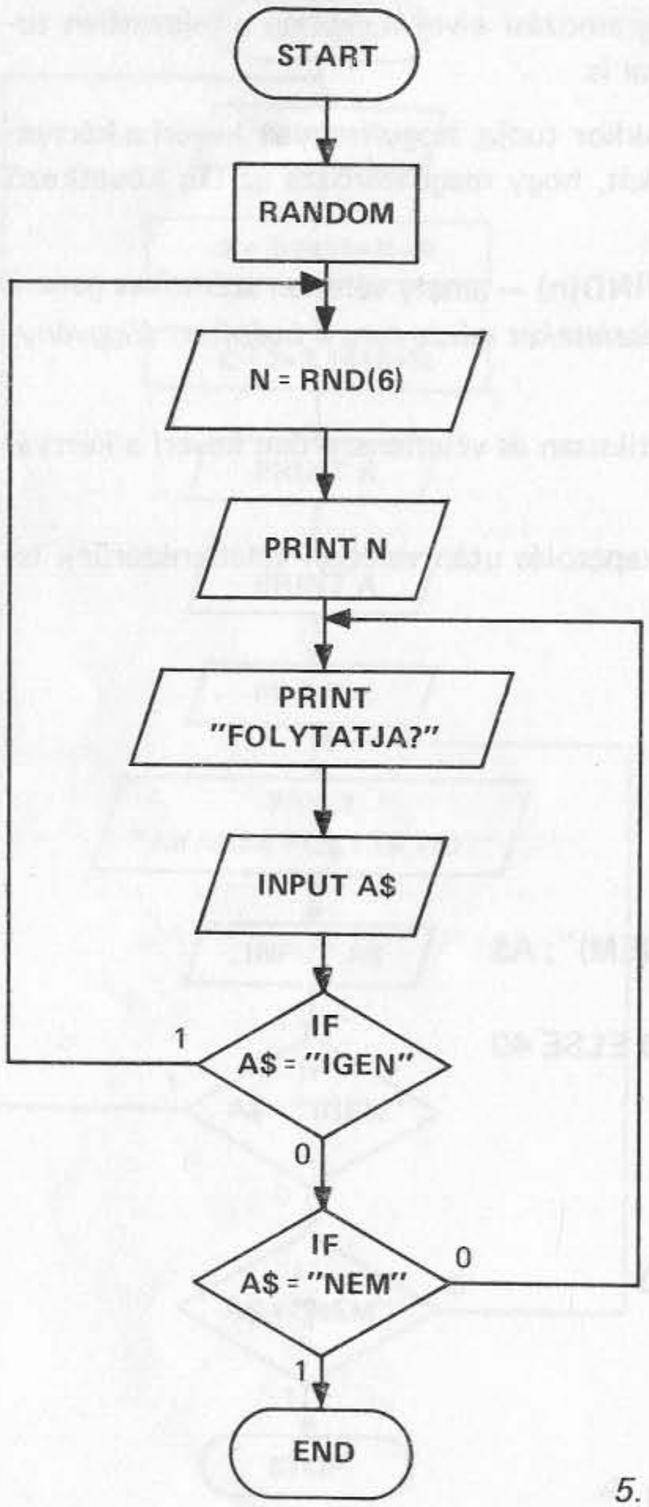
Annak érdekében, hogy a generált számok minden bekapcsolás után valóban véletlenszerűek legyenek, a RANDOM függvényt kell használni.

Példa:

```
10 RANDOM
20 N = RND (6) :PRINT
30 PRINT "AZ EREDMENY:"; N
40 INPUT "AKARJA FOLYTATNI (IGEN/NEM)"; A$
50 IF A$ = "IGEN" OR A$ = "I" GOTO 20
60 IF AS = "NEM" OR A$ = "N" THEN END ELSE 40

READY
> RUN
AZ EREDMENY: 4
AKARJA FOLYTATNI (IGEN/NEM) ? IGEN
AZ EREDMENY: 4
AKARJA FOLYTATNI (IGEN/NEM) ? I
AZ EREDMENY: 6
AKARJA FOLYTATNI (IGEN/NEM) ? NN
AKARJA FOLYTATNI (IGEN/NEM) ? NEM
```





5.

A program folyamatábrája a következő:

A program véletlenszerűen generál egy számot 1-től 6-ig (mintha egy kockát dobálnánk), kinyomtatja a számot, majd megkérdezi a játékost, hogy akarja-e folytatni, vagy sem. Ha a válasz IGEN, vagy I, akkor a számítógép megismétli a folyamatot, ha NEM vagy N, akkor a program befejeződik. Ha a válasz a fentiek közül egyik sem, akkor a gép ismételten megkérdezi a játékost.

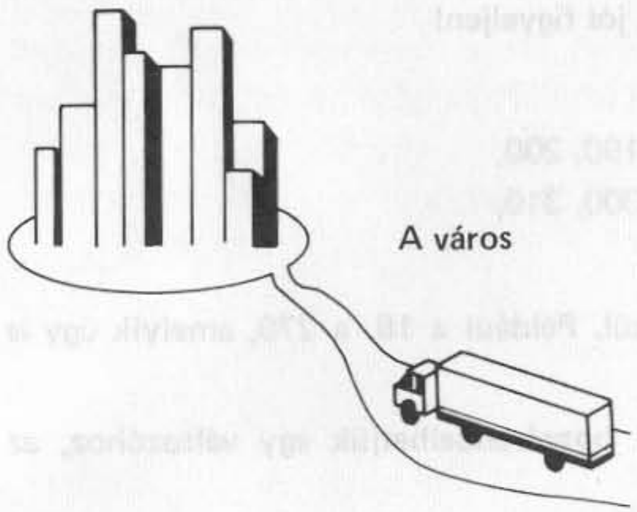
A véletlen-szám generáláson kívül nagyon sok beépített függvény van, amelyek lehetővé teszik, hogy a problémák széles körét egyszerűen egy függvényhívással oldjuk meg.

Egy további példa:

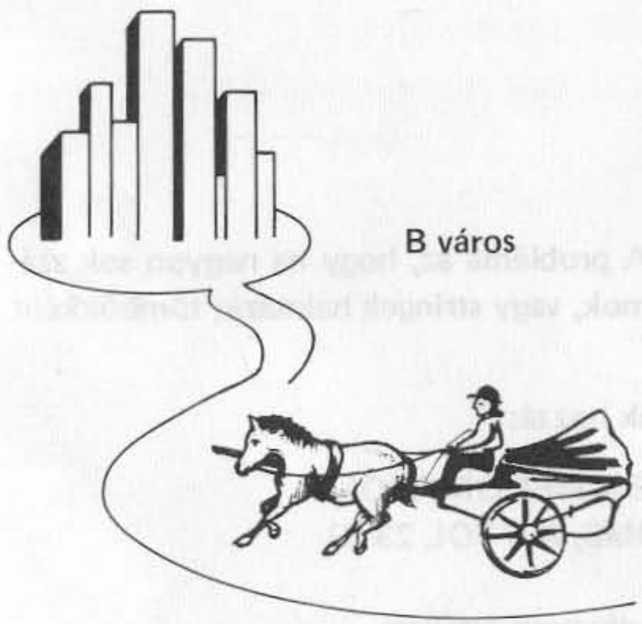
```
10 FOR A = 1 TO 10 STEP 1
20 PRINT A,SQR (A)
30 NEXT A
40 END
```

Ez a program kinyomtatja a számok négyzetgyökét 1-től 10-ig. A kiinduló érték A=1, a 20-as programsorban az 1 és az 1 négyzetgyöke kinyomtatásra kerül. A 30-as programsorban A értéke 1-gyel növekszik, és a gép ellenőrzi, hogy A nagyobb-e 10-nél, ha nem, visszamegy a 20-as sorra.

$\sqrt{1} = ?$
$\sqrt{2} = ?$
$\sqrt{3} = ?$



A város



B város

Előfordul, hogy stringekkel — számok, betűk és egyéb írásjelek sorozatával — kell műveletet végeznünk számok helyett. Ebből következően ismerni kell néhány string műveletet is.

Tegyük fel, hogy Ön egy olyan cég kereskedelmi munkatársa, amely rendeléseiben számokból és betűkből álló kódokat használ.

A kód formátuma: AB1234S001

cikkszám raktári szám

Az első hat karakter a cikkszám kód, az utolsó négy karakter a raktári szám kód, ebből az utolsó három a raktár sorszáma.

Tegyük fel, hogy 20 raktár van. A raktárak 1-től 7-ig az A városban vannak, 8-tól 20-ig a B városban. Hasznos, ha az 1–7 és 8–20 sorszámú raktárakra vonatkozó rendeléseket külön csoportba rendezzük, hogy a szállítási költségek alacsonyak legyenek. Programunk ellenőrizze, hogy a raktárszám kisebb-e, mint 8, ha igen akkor a rendelést tegye hozzá az A városra, ellenkező esetben a B városra vonatkozókhoz.

Kísérje figyelemmel az alábbi programot!

```
10 INPUT "IRJA BE A KODOT"; CN$
20 IN$  = LEFT$ (CN$,6)   :REM  ELSO 6 BETU = CIKKSZ.
30 SN$  = RIGHT$ (CN$,3)  :REM  UTOLSO 3 BETU = RAKT, SZ,
40 SN   = VAL (SN$)       :REM  STRING ERTEK VALTOZTATASA
                                ARITMETIKAI ERTEKRE
50 INPUT " IRJA BE A RENDELT MENNYISEGET"; Q
60 IF SN > 8 THEN GOSUB 1000 ELSE GOSUB 2000
```

Ez még nem a teljes program, az 1000-es és a 2000-es szubrutinokba utasításokat kell írni.

Megjegyzés: Javasoljuk, nézzen utána a LEFT$, RIGHT$ és a VAL függvényeknek a BASIC kézikönyvben.

Ezek után ismerkedjen meg a tömbökkel. Ez fontos rész, jól figyeljen!

Tegyük fel, hogy van 24 számunk.

100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200,
210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310,
320, 330,

Többféle módon is kiragadhatunk egyet a számok közül. Például a 18. a 270, amelyik úgy is meghatározható, hogy a 7. a 2. sorban.

Számítógépes programban a fenti számok mindegyikét hozzárendelhetjük egy változóhoz, az alábbi módon.

```
10 A1 = 100
20 A2 = 110
30 A3 = 120
40 A4 = 130
50 A5 = 140
 :
```

Így már könnyen hivatkozhatunk ezek bármelyikére. A probléma az, hogy ha nagyon sok számunk van, elfogyhatnak a változók. Ezért jobb, ha a számok, vagy stringek halmazát tömbökként kezeljük.

Ehhez a 24 számhoz egy egydimenziós tömböt rendeltünk hozzá:

```
10 CLEAR 200   :REM  200 BYTE TORLESE A MEMORIABOL
20 DIM A (23)  :REM  EGY 24 ELEMES TOMB, A Ø-TOL 23-IG
```

Egy programban a tömb elemeinek a következő módon adhatunk értéket:

```
10 A (0) = 100
20 A (1) = 110
30 A (2) = 120
 :
```

A tömb 270-es értékére a következő egyszerű módon hivatkozhatunk:

```
50 PRINT A (17)  :REM  A TOMB 18.ELEMENEK KINYOMTATASA
100 A (17) = A (17) + 100  :REM HOZZAAD 100-T A 18.ELEMHEZ
```

Hát nem egyszerű?

Ha az eredeti 24 számot három 8-as csoportba osztjuk, egy kétdimenziós tömböt használhatunk.

```
20 DIM  B(2,7)  :REM  EGY 3x8 ELEMU KETDIMENZIOS TOMB
```

Ily módon a számok így rendezhetők el:

SOR	0	100	110	120	130	140	150	160	170
SOR	1	180	190	200	210	220	230	240	250
SOR	2	260	270	280	290	300	310	320	330
OSZLOP		0	1	2	3	4	5	6	7

Most ha a 270 -es számmal akarunk foglalkozni, azt a következőképpen tehetjük meg.

```
100 PRINT B (2,1)  : REM A 2.SOR 1.ELEMENEK KINYOMTATASA
230 F = B (2,1) + 10 : REM 10-ET HOZZAAD
```

Hasonlóan ehhez, ha a 160-as számot akarjuk kezelni, akkor:

```
210 PRINT B (0,6): REM A 0.SOR 6. ELEMENEK KINYOMTATASA
```

Lépjünk eggyel tovább, ezt a huszonnégy számot még kisebb csoportokra lehet osztani, mondjuk 4 lapra, 2x3 elemű tömbökre. Másszóval, felosztásuk végeredménye egy háromdimenziós tömb lesz.

```
20 DIM E (3,1,2)  :REM  EGY 24 ELEMU TOMB E (4x2x3)
```

	0.LAP			1.LAP		
SOR 0	100	110	120	160	170	180
SOR 1	130	140	150	190	200	210
OSZLOP	0	1	2			

	2.LAP			3.LAP		
	220	230	240	280	290	300
	250	260	270	310	320	330

Ekkor, ha a 270-es számot akarjuk kezelni, azt következőképpen tehetjük:

```
70 PRINT E (2,1,2)  :REM  A 2.LAP 1.SOR 2. ELEMENEK KINYOMTATASA
```

Ha a 200-as számot akarjuk kezelni, E (1,1,1) formában hivatkozhatunk rá.

Tekintsük a következő programot:

```
10 DIM E (3,1,2)     :REM  E EGY 4x2x3 DIMENZIOS TOMB
20 FOR P = 0 TO 3  :REM  HUROK A KOVETKEZO P-IG, 4-SZER
30 FOR R = 0 TO 1  :REM  HUROK A KOVETKEZO R-IG, 2-SZER
40 FOR C = 0 TO 2  :REM  HUROK A KOVETKEZO C-IG, 3-SZOR
50 INPUT "IRJON BE EGY SZAMOT"; E (P,R,C): REM EGY SZAM BEVITELE
60 NEXT C
70 NEXT R
80 NEXT P
90 FOR P = 0 TO 3
100 FOR R = 0 TO 1
110 FOR C = 0 TO 2
120 PRINT E (P,R,C), :REM AZ OSSZES ELEM KINYOMTATASA
130 NEXT C
140 PRINT
150 NEXT R
160 PRINT
170 NEXT P
180 END
```

Ez a program kinyomtatja a bevitt számokból álló 4 lapot, ha a bevitel helyes sorrendben történt.

Most már önállóan programozhat, az egyedüli korlátozó tényező az Ön képzelőereje, és nem a számítógép.

Gyártó:
HÍRADÁSTECHNIKA SZÖVETKEZET

H–1519 Budapest
Pf. 268
Telex: 22–6151 htsz h

II/83 nyomat
(Az I/83 nyomat átdolgozott kiadása.)

Felelős kiadó:
Somlyay Endre

Készült a MEDIA GM gondozásában

Aranykalász , Dunaföldvár

75~