

(Véletlen számok, néhány feladat)

Vannak olyan jelenségek, folyamatok, amelyek kimenetelét nem tudjuk (vagy nem akarjuk) előre megjósolni. Ilyen például az, hogy egy szabályos kockát feldobva, az melyik oldalára esik; vagy hogy egy csomag kártyát alaposan megkeverve mi lesz a kártyák sorrendje.

A számítógépektől megkívánjuk, hogy ilyen véletlen jelenségek is programozhatók legyenek. A BASIC-nyelvben „véletlen” számokat a *RND* (random = véletlen) függvény meghívásával kaphatunk; a függvényérték 0 és 1 közé esik, és ebben az intervallumban nagyjából egyenletesen oszlik meg. Ez például azt is jelenti, hogy a (0; 1/6), az (1/6; 2/6) stb. intervallumokba a függvényérték nagyjából egyforma gyakran esik. Így az $INT(6 * RND) + 1$ kifejezést használhatjuk a kocka feldobása helyett, hiszen értéke 1, 2, 3, 4, 5 illetve 6 lehet, és mindegyiket nagyjából egyforma gyakorisággal veszi fel. Az alábbi program azt számítja ki, hogy két különböző színű lapot egy dobozból n -szer kihúzva, hányszor kapunk más-más színűt. Az első sorban a RANDOMIZE utasításra azért van szükség, mert a gép a véletlen számokat nem „véletlenül” állítja elő, hanem egy jól meghatározott algoritmussal, így azok értéke futásról futásra nem változna. (Ezért is hívják az ilyen számokat pszeudovéletlen, álvéletlen számoknak, hiszen valójában nem is véletlenek.) A RANDOMIZE utasítás viszont ennek az algoritmusnak a kezdő értékét valamilyen valóban „véletlen” értékkel állítja be (például az aktuális idő értéke ezredmásodpercekben), s így a *RND* függvény futásonként is más-más értéket ad.

```
10 RANDOMIZE          90 E=E+1
20 PRINT „FEKETE-FEHÉR” 100 GOTO 120
30 PRINT „HÁNYSZOR HÚZZAK”; 110 R=R+1
40 INPUT N            120 NEXT K
50 E = 0; R = 0        130 PRINT ?FEKETE?; E; „ESETBEN”
60 FOR K=1 TO N        140 PRINT „FEHÉR”; R; „ESETBEN”
70 L=INT(2* RND)+1     150 END
80 IF L=1 THEN 110
```

Másik programunk egy feladat „játékos megoldására” mutat példát. A feladat, mely eredetileg az American Mathematical Monthly-ban jelent meg, a következő : Van kilenc különböző méretű palacsintánk. Sütés után egy tányéron, egymás felett helyezkednek el. Az a feladat, hogy a palacsintákat nagyság szerint rendezzük : a legkisebb legyen felül, a következő alatta stb. és a legnagyobb pedig legalul. Ehhez a következőket tehetjük: felülről valahány (akár az összes) palacsintát megfogjuk, majd megfordítva, „fejfel lefelé” visszatesszük. Például ha a palacsinták eredetileg a következőképpen helyezkedtek el:

2 3 4 5 1 6 7 8 9 [tányér],

akkor a felső négy palacsinta megfordítása után a; helyzet :

5 4 3 2 1 6 7 8 9 [tányér].

Most már csak a felső öt palacsintát kell megfordítanunk, hogy a kívánt sorrendet megkapjuk

1 2 3 4 5 6 7 8 9 [tányér],

```
100 PRINT „PALACSINTA PROGRAM”      240 FOR K=1 TO 9 : PRINT A(K) : NEXT K
110 RANDOMIZE                        250 T=T+1
120 DIM A(10)                        260 INPUT „HÁNYAT FORDITSÁK MEG”,F
130 REM ÖSSZEKEVERJÜK A PALACSINTÁT 270 FOR K=1 TO INT(F-1)/2)
140 A(1)=INT(8* RND)+2               280 Z=A(K):A(K)=A(F-K+1):A(F-K+1)=
150 FOR K=2 TO 9                     =Z
160 A(K)=INT(9* RND)+1               290 NEXT K
170 FOR J=1 TO K-1                   300 REM RENDBEN VAN-E?
180 IF A(J)=A(K) THEN 160            310 FOR K=1 TO 9
190 NEXT J                           320 IF A(K)I <>K THEN 230
200 NEXT K                           330 NEXT K
210 PRINT „MEGVOLNA...A PALACSINTÁK” 340 PRINT „MEGVAN!!!”; T;
220 T=0                              „FORGATÁSSAL”
230 PRINT                           350 END
```

Könnyen belátható, hogy n palacsinta esetén legfeljebb $2n - 3$ forgatással célhoz érhetünk, de a szükséges forgatások minimális száma még 9 palacsinta esetén sem ismert.

Gyakran előfordul, hogy egy adott program vagy programrészlet matematikai alapjait is meg kell vizsgálni. A most kitűzésre kerülő két feladat is ezt a célt szolgálja.

1. Adott a sík egy $P(A; B)$ pontja. Határozzuk meg az egységnyi sugarú, origó középpontú körre vonatkozó $T(X; Y)$ inverzét. Elemezzük, hogyan hajtja végre ezt a feladatot az alábbi program.

```
10 INPUT A, B
20 Z = A * A + B * B
30 IF Z = 0 THEN 70
40 X = A/Z : Y = B/Z
50 PRINT, , X ="; X, , Y ="; Y
60 GOTO 10
70 PRINT „IDEÁLIS PONT”
80 GOTO 10
```

2. Adott egy $n > 1$, egész szám. Ha n páros, osszuk el kettővel. Ha páratlan, akkor szorozzuk meg 3-mal, és adjunk hozzá 1-et. Ezut folytassuk addig, míg 1-et nem kapunk. Hogyan működik az alábbi program? Hogyan értelmezhetjük a program futásának eredményét?

```
10 INPUT L, C          90 GOTO 60
20 FOR J=3 TO C STEP 2 100 N=N/2:K=K+1
30 FOR I=2 TO L         110 IF K>10 000 THEN 160
40 N=I                 120 IF N > THEN 60
50 K=0                130 PRINT,J, I, K
60 IF N= INT(N/2)*2 THEN 100 140 NEXT J
70 N=J*N+1 : K=K+1     150 NEXT J
80 IF K> 10 000 THEN 160 160 END
```