Véletlenszámok

Tartalom

1 A véletlen (számok) természete

Mennyire véletlenek az alábbi 0-1 sorozatok? A sorozatok a „valamilyen” tulajdonságú sorozat első néhány, jellegzetes tagját mutatja (azaz a folytatás is hasonló lenne).

A.

`1000010000 0100011000 0000100000 0000001000`

Nem tűnik annak, mert túl sok a 0, túl kevés az 1: nagyon eltérő számúak a kijöhető értékek.

B.

`1010101010 1010101010 1010101010 1010101010`

Bár azonos számúak a kijöhető értékek, még sem tűnik annak, mert túl szabályos: csak ’10’ és ’01’ kételemű sorozatot tartalmaz, egyetlen egyet sem a ’00’-ból, s az ’11’-ből.

C.

`0011001100 1100110011 0011001100 1100110011`

Ez sem jó, mert bár 0-k és 1-ek száma azonos, sőt a 2-hosszú részsorozatok is kiegyensúlyozottak (Db(’00’): 10, Db(’01’): 10, Db(’10’): **9**, Db(’11’): 10) , a szabályosság megmaradt. A kiegyensúlyozatlanság a 3-asokra, és a többesekre változatlanul fenn áll.

D.

`0110111001 0111011110 0010011010 1011110011 0111101111`

Ránézésre elfogadható, mert nem fedezhető föl benne szabályosság. Előfordulnak hosszabb 0- és 1-sorozatok. Persze ez a szakasz statisztikailag nem teljesen kiegyensúlyozott, de a rövidségre foghatjuk. (Db(’0’): 18, Db(’1’): 32; Db(’00’): 5, Db(’01’): 11, Db(’10’): 12, Db(’11’): 19…

Valójában –algoritmikusan– teljesen „szabályos”. A szabály: a 0, 1, 2 … 15 számok bináris alakban!

`0110111001 0111011110 0010011010 1011110011 0111101111`

E.

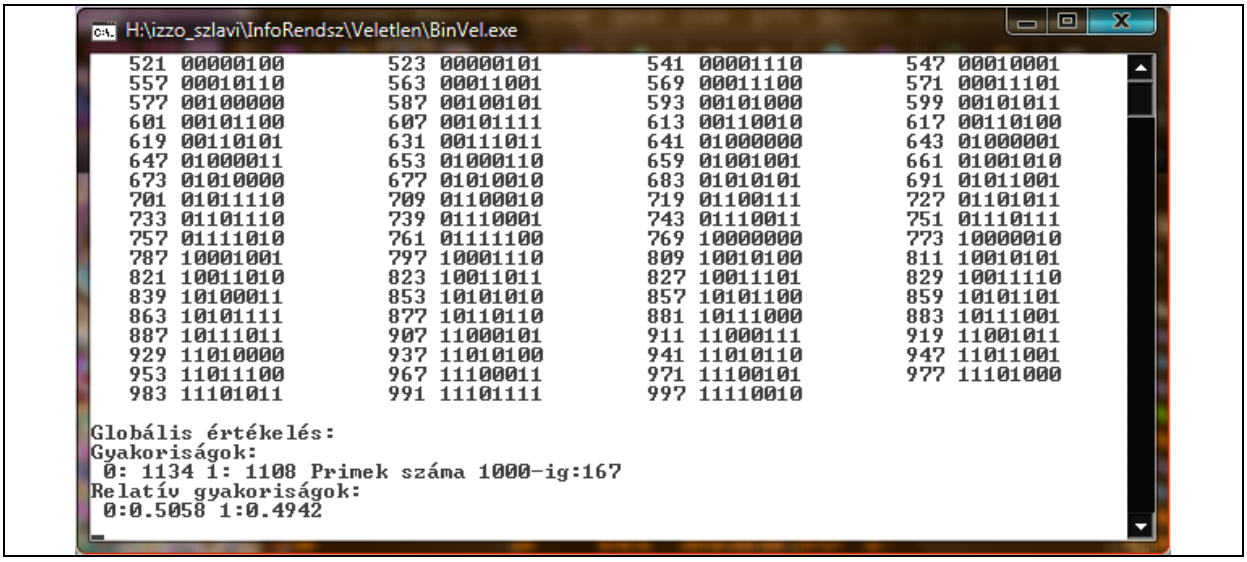
`0101100000 0101111011 1001001000 1010111101 0110111100`
`0001000110 0100001110 1001011001 0000`

Ránézésre ez is elfogadható, mert ebben sem fedezhető föl szabályosság. Előfordulnak hosszabb 0- és 1-sorozatok. Persze ez a szakasz sem teljesen kiegyensúlyozott statisztikailag, de a rövidségre foghatjuk. (Db(’0’): 92, Db(’1’): 76; Db(’00’): 24, Db(’01’): 21, Db(’10’): 21 , Db(’11’): 17…

Valójában ez is teljesen „szabályos”. A szabály: 5-től a prímek bináris alakban első és utolsó jegyük nélkül. A bonyolítás magyarázata: mivel a prímek (a 2 kivételével) páratlanok, így utolsó jegyük 1-es, első értékes jegyük szintén 1-es, ezért az 1-esekből eleve többlet lenne. A fenti sorozat „levezetése”:

5: ’0’, 7: ’1’, 11: ’01’, 13: ’10’, 17: ’000’, 19: ’001’, 23: ’011’, 29: ’110’, 31: ’111’,
37: ’0010’, 41: ’0100’, 43: ’0101’, 47 ’0111’, 53: ’1010’, 59: ’1101’, 61: ’1110’,
67: ’00001’, 71: ’00011’, 73: ’00100’, 79: ’00111’, 83: ’01001’, 89: ’01100’, 97: ’10000’

Nézzük meg kicsit hosszabb sorozatra az alábbi ábrán (prímek 1000-ig), és a BinVel.exe program[1] futtatásával még hosszabbra (pl. a prímek 10000-ig)!



1. ábra. A BinVel.exe futási képe. (1000-ig)

2 A véletlenszám-generátor

Lássunk egy kis bemutatót a véletlenszámokról! A program feladata, hogy bemutassa a véletlenszám-generálás egy-két „titkát”. (Gyakorlat_letolt.zip [2])
A továbbiakban –a rövidség kedvéért– hadd hivatkozzunk a véletlenszámot generáló függvényre RND-vel.

A program lépései:

1. Az RND függvény segítségével generálunk 80 darab 0 és 9 közötti egyenletesen véletlenszámot.
2. A négyzet-közép módszer kipróbálása.
3. A négyzet-közép módszerrel előállítjuk az összes kezdőértékkel a sorozatokat, az első ismétlésig. Kíváncsiak vagyunk a leghosszabb sorozatra.
4. A két nevezetes transzcendens szám statisztikai elemzését végezzük el. Meghatározzuk a ’ π’ és az ’e’ számjegy-gyakoriságát az első 1000 szám-jegyük alapján.
5. Az előbbi két nevezetes transzcendens szám statisztikai elemzését végezzük el ismét: a számjegy-párok gyakoriságára vonatkozólag.
6. Véletlen ábrákat generálunk a véletlenszám-generátorral. S így jópofa Vasarely-szerű képekhez juthatunk.
7. A Pascal Random függvényével előállított számsorozat véletlenszerűségét vizsgáljuk az ún. Póker-teszttel.
8. A π „színei”.

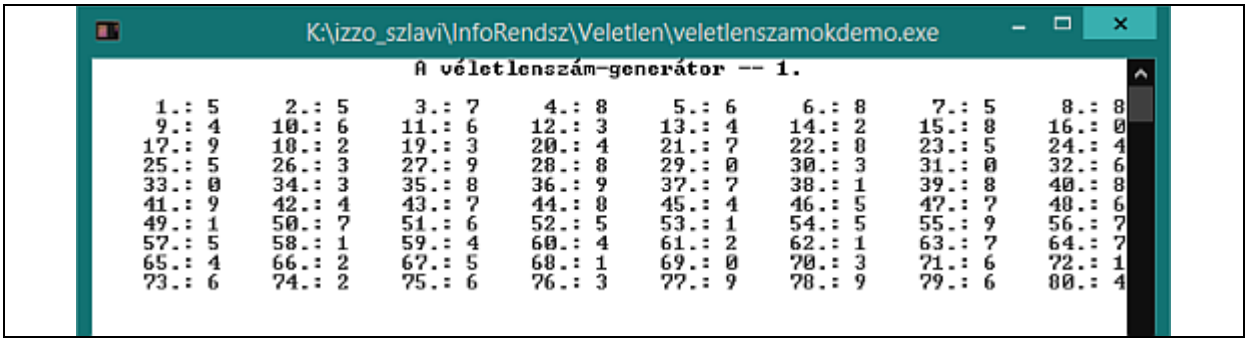
2.1 A véletlenszám-generátor függvénye

Az RND függvény segítségével generálunk 80 darab 0 és 9 közötti egyenletesen véletlenszámot.

[1] http://people.inf.elte.hu/szlavi/InfoRendsz/Veletlen/BinVel.exe
[2] http://people.inf.elte.hu/szlavi/InfoRendsz/Veletlen/Gyakorlat_letolt.zip

Pascal tudnivalók:	VBA tudnivalók:
Function Random(**Const** x:Word):Word {Random:→N, 0≤Random(x)<x} **Function** Random:Real {Random:→R, 0≤Random<1}	**Function** Rnd:Single 'Rnd:→N, 0≤Rnd<1

Megfigyelhető, hogy minden lehetséges számjegy előfordul, eléggé „váratlan” sorrendben.



2. ábra. A demonstrációs program „1. eredménye”:
mennyire tűnik véletlennek a sorozat?

2.2 Négyzet-közép módszer 1.

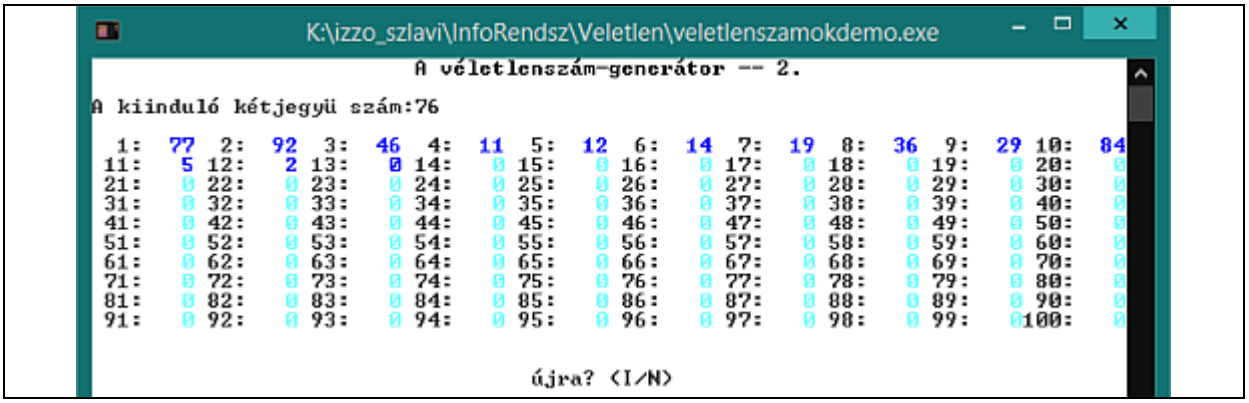
A négyzet-közép módszer kipróbálása néhány kiinduló értékre. Egy 2-jegyű számot megadva a program kiszámolja a belőle kapható 100 db „véletlenszámot”.
Érdekes kezdőérték „jóság” szerinti csoportosításban:

- 69, 76, 77;
- 15, 99;
- 24, 57, 79;
- 10, 50.

A módszer képlete (2-jegyű véletlenszámok előállítására):

R(i+1):=((R(i)*R(i)) Div 10) Mod 100,
ahol R(x) az x. véletlenszám (x=0, 1, 2 …).

Újra meg újra megadható a kezdő érték, így ameddig tetszik folytatható a kísérletezgetés.



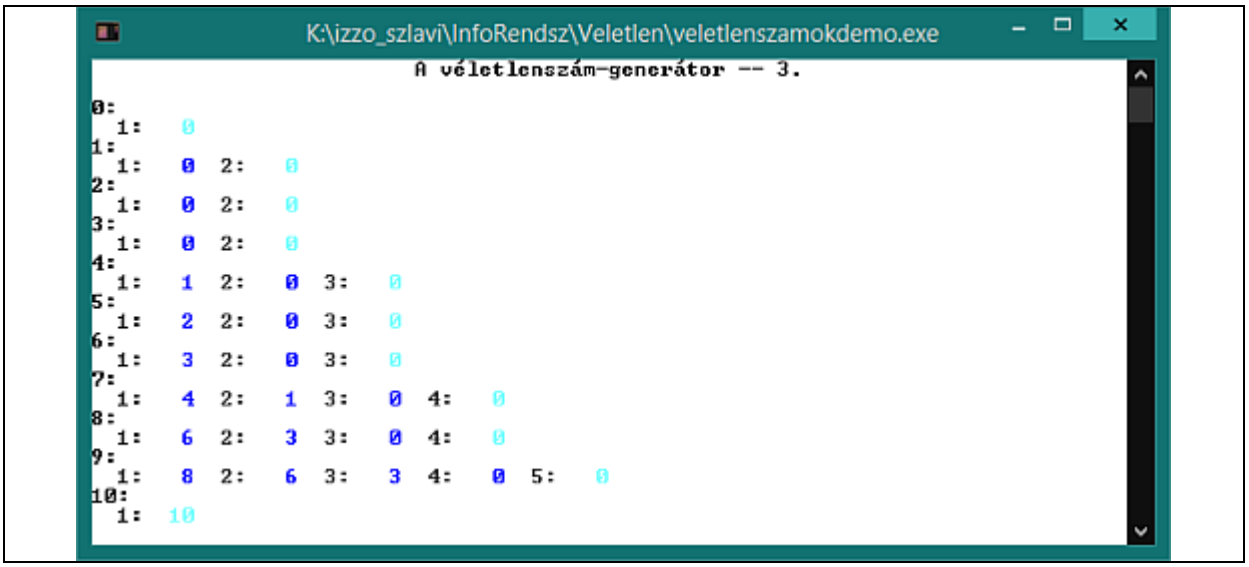
3. ábra. A demonstrációs program „2. eredménye”.
A 76-ből eredő sorozat véletlenszerű „fejsora” 14 elemű.

1. feladat: Adja meg a formulát, ha nem 2, hanem 2*k jegyű véletlen számokat akarunk a fenti módszerrel generálni!

2. feladat: Írja meg azt a függvényt, amely előállítja a következő véletlen értéket! Világos, hogy az előzőleg generált értéket és a k-t a függvényen kívül elhelyezett globális változók tartalmazzák. A korrekt használathoz kell egy inicializáló eljárás is, amely a 0. értéket állítja be.

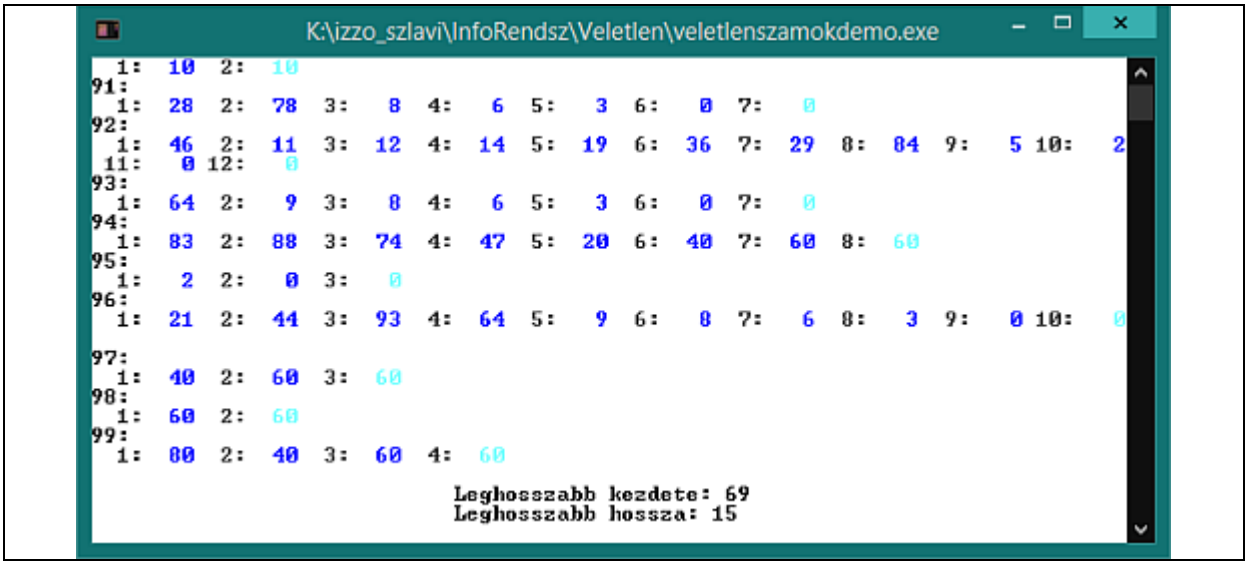
2.3 Négyzet-közép módszer 2.

Ismét 2-jegyű véletlenszámokat generálunk a négyzet-közép módszerrel: 00-99. Előállítjuk az összes kezdőértékkel a sorozatokat, az első ismétlésig. Kíváncsiak vagyunk, melyik a leghosszabb sorozat.
A kiírást tetszőleges (mondjuk a SPACE) billentyű lenyomásával időlegesen felfüggeszthetjük. Újabb billentyű lenyomása után folytatódik a listázás.



4. ábra. A demonstrációs program „3. eredményének” egy részlete.
Az eddigi leghosszabb (5 hosszú) véletlen sorozat a 9-ből készült.

… és végül:



5. ábra. A demonstrációs program „3. eredményének” a befejező részlete.

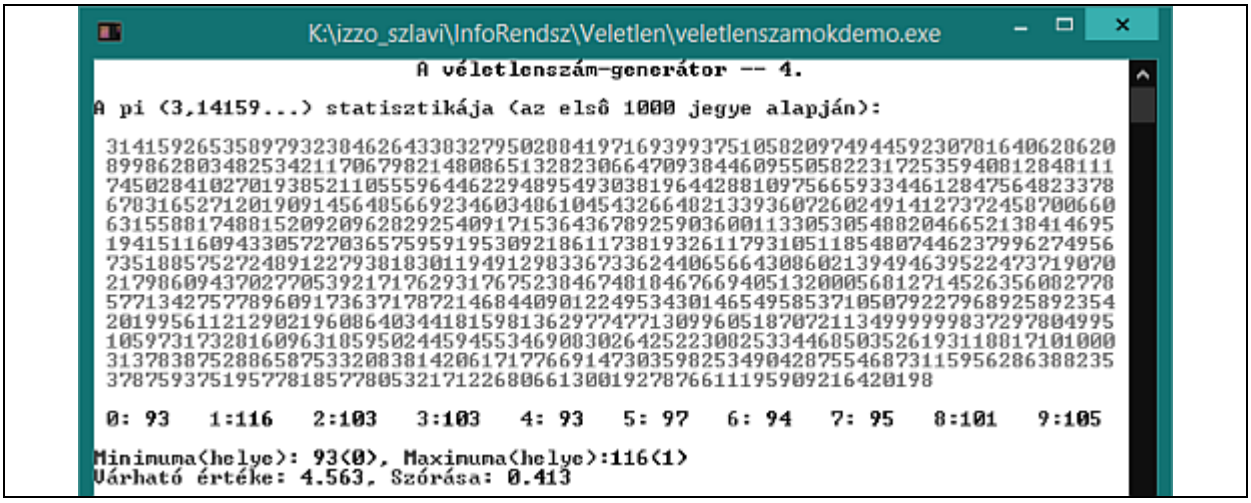
3. feladat: Készítsen paraméteres függvényt, amely a korábbi véletlenszám függvényt addig hívja újra meg újra, amíg ismétlés nem lép föl. Paramétere a kezdőérték, visszaadott értéke az **aperiodikus szakasz** hossza (az első ismétlődésig generált elemek száma). Milyen módszert lát arra, hogy az ismétlődést a leghatékonyabban tudja megállapítani? Először felteheti, hogy a véletlenszámok 0..99 közöttiek.

2.4 Transzcendens számok 1.

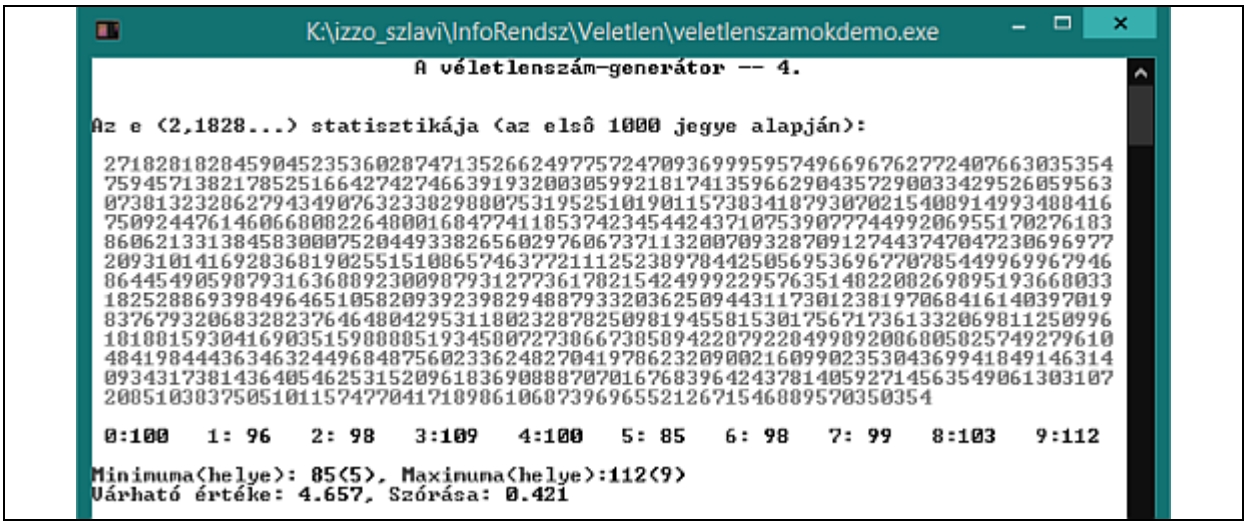
„A véletlen(számok) természete” fejezetben már próbálkoztunk speciális számokból kiindulni a véletlen számsorozat előállításánál. Kézenfekvő gondolat valamely transzcendens számból kiindulni, mondván: legyen az i. véletlenszám az ő i. véletlen számjegye. A transzcendens volta garantálja, hogy periodikus szakaszokkal nem fogunk találkozni.

A két nevezetes transzcendens szám statisztikai elemzését végezzük el. Meghatározzuk a ’π’ és az ’e’ számok első 1000 szám-jegyük alapján a számjegy-gyakoriságot, ezek várhatóértékét és szórását, továbbá a minimum- és maximum-helyet/-értéket.

A két transzcendens számot egy-egy fájlból olvassuk be, amelyeket az internetről töltöttük le (hogy ne kelljen hosszú, bonyolult számítással magunknak létrehozni).[3]



6. ábra. A demonstrációs program „4. eredményének” az első fele.



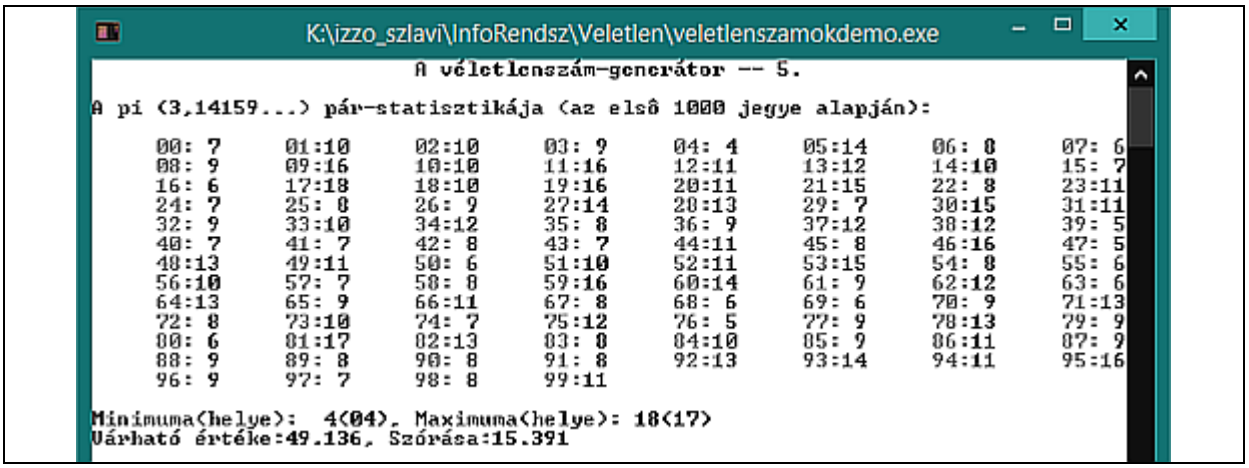
7. ábra. A demonstrációs program „4. eredményének” folytatása.

4. feladat: Készítsen paraméteres eljárást, amely egy szövegfájlból számkaraktereket olvas, közben számolja ezek gyakoriságát, majd megadja a legkevesebbszer előforduló karaktert, gyakoriságával együtt, a legtöbbször előfordulót… a gyakoriságok átlagát és szórását! A program paramétere a fájl neve legyen! Az eredményt a képernyőre írja.

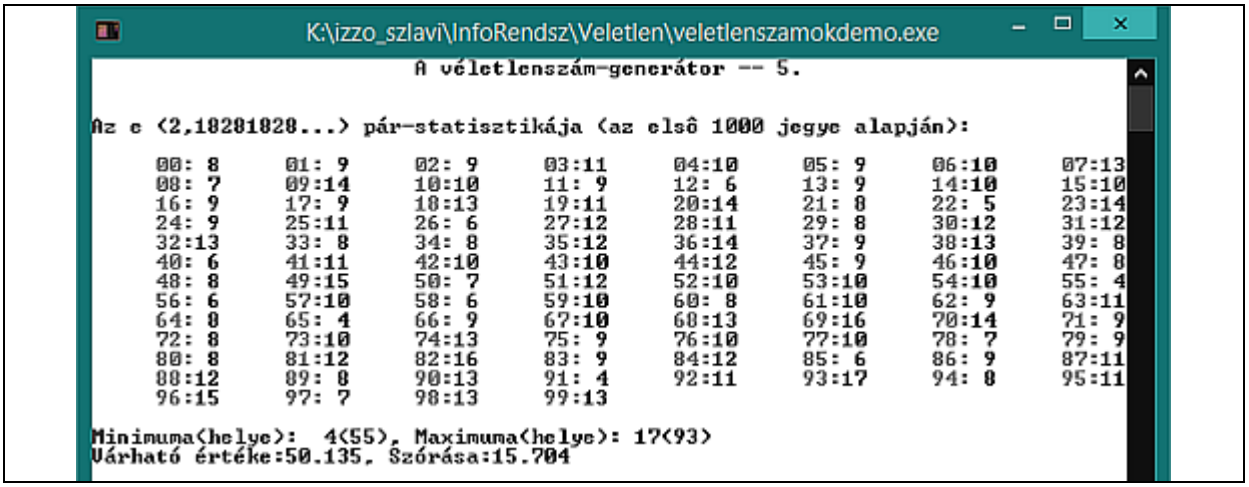
[3] http://www.gutenberg.org/ebooks/50, http://www.gutenberg.org/ebooks/127

2.5 Transzcendens számok 2.

A két nevezetes transzcendens szám statisztikai elemzését végezzük el ismét: most a számjegy-párjainak a gyakoriságát határozzuk meg az első 1000 számjegyük alapján. Megadjuk a számjegy-párok gyakorisága mellett a várhatóértékeket, szórásokat és a minimum- és maximum-helyeket, valamint értékeket.



8. ábra. A demonstrációs program „5. eredményének” az első fele.
A ’π’ elemzése.

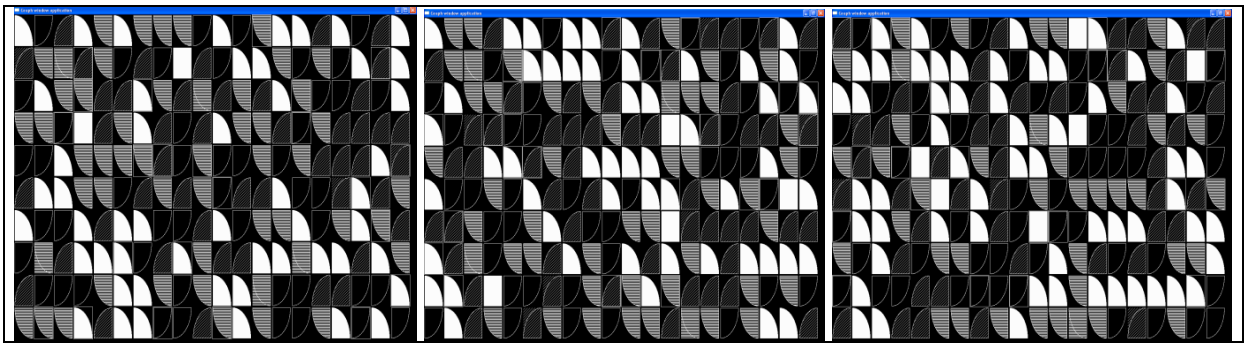


9. ábra. A demonstrációs program „5. eredményének” a folytatása.
Az ’e’ elemzése.

Konklúzió: az első ezer számjegyük alapján nem igazán tűnnek megfelelő véletlenszám-forrásnak; ui. nem kellően egyenletesek a vizsgált két (számjegy és számpár gyakoriság) szempontból.

2.6 Egy művészi alkalmazás

Egy kis pihenésként nézzünk egy véletlenszám alkalmazást: képeket „festünk” Vasarely stílusában. Alapgondolat: véletlen ábrákat generálunk a véletlenszám-generátorral. Összesen 4-féle alapábrából fogjuk kirakni mozaikszerűen a képet. Hogy mely alapábrát válasszuk éppen, azt a véletlenre bízzuk.

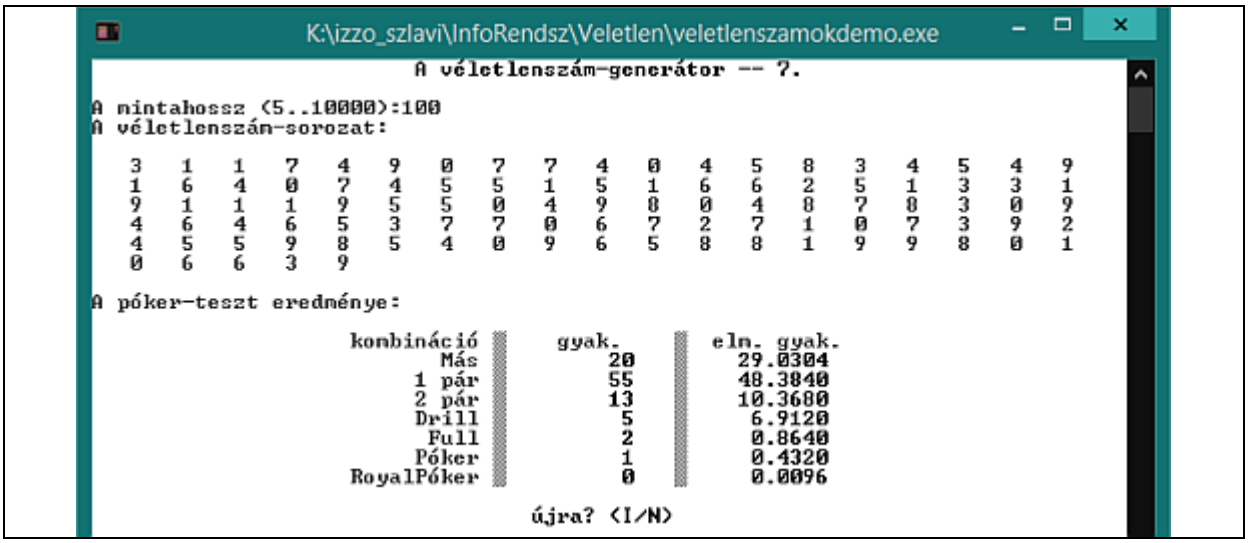


10. ábra. A demonstrációs program 6. „stációjának” 3 kompozíciója.

2.7 A Póker-teszt

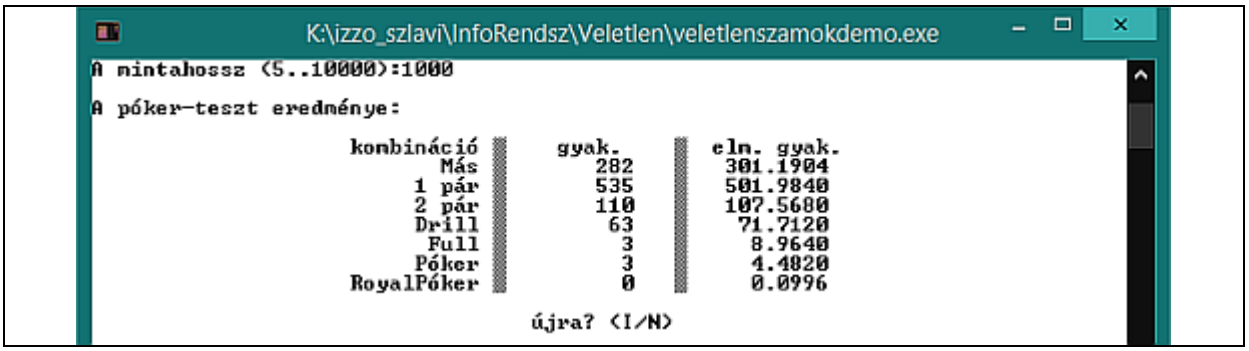
A Pascal `Random` függvénnyel előállított számsorozat véletlenszerűségét vizsgáljuk az ún. **Póker-teszt**tel. A lényege, hogy a pókerben nevezetes (4-es helyett 5-ös) kombinációkat megszámoljuk, majd összevetjük az egyenletesség esetén „elméleti” eloszlásból kapható értékkel. A program véletlenszerűen 10-féle lapból (0..9) húzogat a paraméter meghatározta számszor (4 helyett) 5 értéket. Így egy adott „figura” kijövetelének 1/10 a valószínűsége. Ennek figyelembe vételével határoztuk meg az egyes pókerbeli kombinációk valószínűségét.

Póker-leosztás	Kombináció	Valószínűség
royal póker	aaaaa	0,0001
póker	aaaab	0,0045
full	aaabb	0,0090
drill	aaabc	0,0720
2 pár	aabbc	0,1080
1 pár	aabcd	0,5040
más	abcde	0,3024



11. ábra. A demonstrációs program „7. eredményének” részlete (N=100).

Ha a képernyőn már „követhetetlenül” sok a generálandó szám, akkor azokat már nem jeleníti meg a program, csak az összegzést.



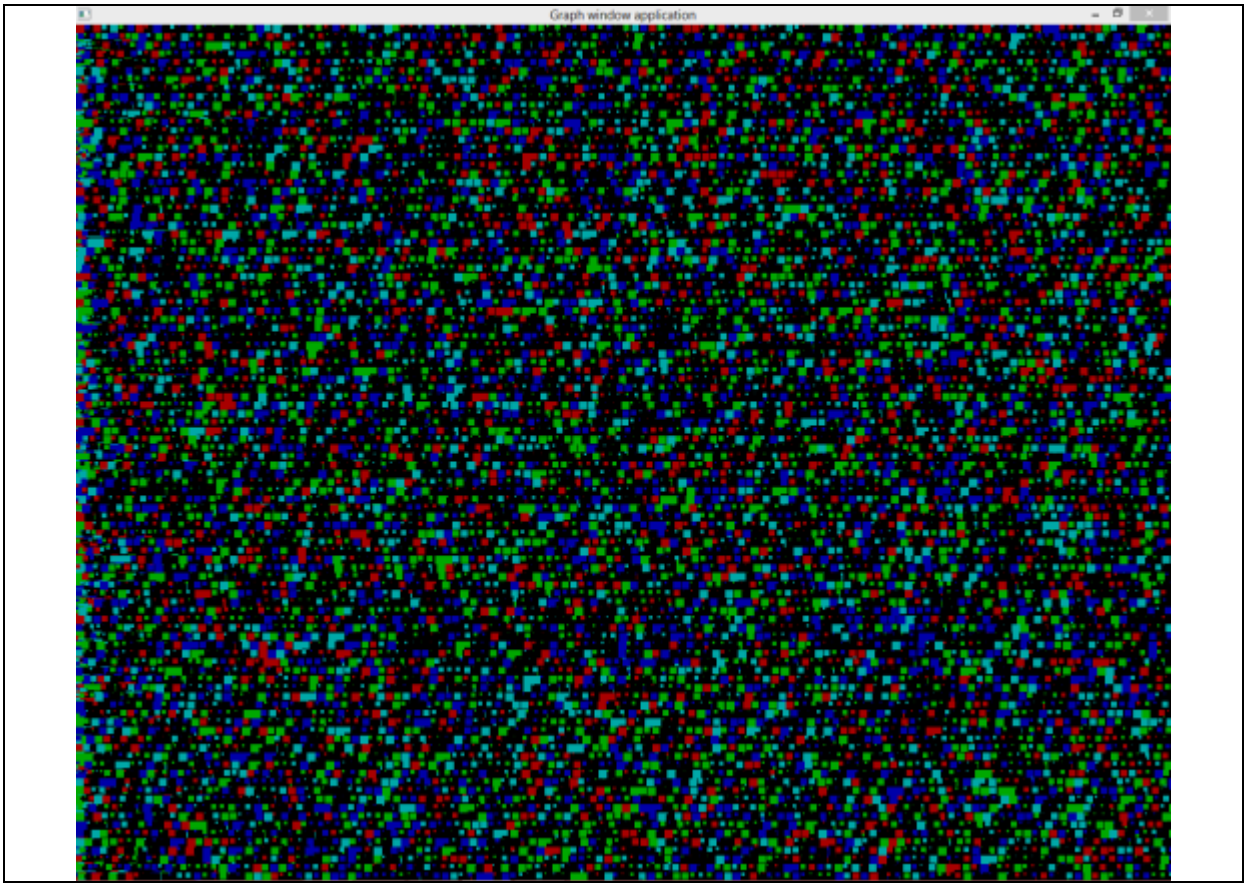
12. ábra. A demonstrációs program „7. eredményének” egy további részlete (N=1000).

A további kísérletek megnyugtatóak: jól konvergál az elvárt értékekhez.

5. feladat: Írjon függvényt, amely paraméterül kap 5 értéket, és eldönti, hogy a póker mely nevezetes leosztása! Értéke 1, ha **1 pár**; 2, ha **2 pár**; 3, ha **drill**; 4, ha **full**; 5, ha **póker**; 6, ha **royal póker**, és 0, ha **egyéb**. Erre a függvényre építve szervezze meg egy számsorozat Póker-teszt szerinti kiértékelését.

2.8. A π „színei”

A π számjegyeit színkódnak tekintve generálunk egy színes „dinamikus” ábrát az első 1 000 000 jegyéből. (A „dinamikus” jelző azt jelenti, hogy időben változó képként jön létre a π lenyomata.)



10. ábra. A demonstrációs program 8. „stációjának” végső állapota.

3 Kongruenciális véletlenszám-generátor

Végezetül és közvetlen tapasztalatszerzésül írjunk programot, amely az alábbi érdekes állítás igazságtartalmát „empirikusan” megvizsgálja!

Állítás: [4]

Legyen

$R(i+1) \equiv b*R(i)+c \pmod m$, $c \neq 0$, $b \neq 1$, $m=p^e$ (p prímszám, $e>1$ egész)

rekurzívan definiált számsorozat akkor és csak akkor m (azaz maximális) periódushosszú, ha

(1) c és m relatív prímek,

(2) $b \equiv 1 \pmod p$,

(3) $b \equiv 1 \pmod 4$, ha $p=2$.

■

A program lépései legyenek a következők:

1. A program bekéri a véletlenszám-generátor paramétereit: b, c, m és R(0)-at. Ezek mind nem negatív egészek.
2. Ellenőrzi az állítás feltételeinek a teljesülését, és „megjósolja”, hogy az állítás szerint van-e esély a maximális hosszúságú (azaz m-hosszú) véletlenszám-sorozatra.
3. A vizsgáló részben addig generál a fenti rekurzív formulával számokat, amíg az első ismétlődő számig el nem jut. A generált számok száma legjobb esetben m lesz, azaz ekkora tömb elegendő lesz a számok gyűjtésére (ez egyben az aperiodikus szakasz hossza).

A két azonos érték sorszámának a különbsége lesz a periódushossz.

A program addig gyártja az újabb és újabb véletlenszám-sorozatot, amíg ESC-pel ki nem lépünk.

Valami ilyesmit kellene kapni: KongruenciaModszer.exe [5]. (Példaparaméterek: c=2, b=28, $m=3^3$, R(0)=tetszőleges.)

---

[4] Deák István: „*Véletlenszámok-generátorok és alkalmazásuk*” c. (Akadémiai Kiadó, 1986) könyvben a 42. oldalon szereplő állítás.

[5] http://people.inf.elte.hu/szlavi/InfoRendsz/Veletlen/KongruenciaModszer.exe,
és a megoldás: http://people.inf.elte.hu/szlavi/InfoRendsz/Veletlen/KongruenciaModszer.htm