Преглед НЦД 23 (2013), 44–50

Djordje Djordjević
University Niš
Srbislav Nešić
Construction Cluster Dundjer, Niš

DAPHNIA BIO-SENSOR DATA TRACKING, RECORDING, AND RANDOM GENERATION

Abstract. One of the most used bioassays for toxicity screening of chemicals and for toxicity monitoring of effluents and contaminated waters is the acute toxicity test performed with Daphnia Magna. Standard methods have been developed and gradually improved by national and international organizations dealing with toxicity testing procedure, in view of its application within a regulatory framework. As for all toxicity tests, the organisms used for the acute Daphnia magna assay have to be obtained from live stocks which are cultured in the laboratory on live food (micro-algae). The technical and biological problems inherent in year-round culturing and the culturing/maintenance costs of live stocks restricts its application to a limited number of highly specialized laboratories. This bottleneck in toxicity testing triggered investigations forward the concept of "microbiotests" or "small-scale" toxicity tests.
This paper deals with the measurement of changes in the behavior of Daphniae using device BIOTOXINOMER, awarded by Ministry for Science and Technological Development of Republic Serbia with Diploma for best Innovation idea in year 2010 in category Medicine, Health, and Ecology. Actually, the main topic is digitalization of bio-monitoring results and random number generating on the base of Daphniae movement (distance). The on-line monitoring is available on site www.dundjer.co.rs/Daphniae and open-source software support is available on the same site.
Keywords. Bio-sensoring, Water quality, Random number generator.

1 Basic working principle of device

BIOTOKSINOMER is the device for water quality control using small water crustacean – (shrimp) Daphnia, named water flea (Fig. 1.), which is very sensitive to the presence of toxins in the water and therefore informs and warns in real time on presence of toxins in water.

Fig.1. Daphnia

The innovation of the device is in the function of real time identification and early warning system for bad quality of drinking and river water, what is realised by implementation of device on the entrance of raw water into water purification plant or entry of waste water into the river, or using bypass for taking river water for Daphnia test. In that way this unique device provides on-line 24-hour monitoring of water quality or identification the presence of dangerous substances in water. The device achieves it in a quick, inexpensive and ecologically acceptable way. If Daphnia can survive in test water, then the water is safe for humans and water animals. If Daphnia dies (or only reduce movement), it means that water is in some way contaminated. This water test is quality water test which should be followed by quantity test (chemical analysis) after any indications of contamination, i.e. presence of toxins.

"Biotoksinomer" works through a simple and cheap hardware and complex and sophisticated software. The real core of the device is an unique software that has the ability to perceive the death of Daphniae in real time (by counting them) or indicate slower activity (movement) of Daphniae and immediately informs the competent.

The device consists of bioassay with 2 chambers, one with a control group Daphnia, which is located in stable water conditions (conditions that are imitating Daphnias natural environment) and the second chamber through which flows constantly raw water, with desired dynamics, before treatment into the drinking water, that contains Daphnia testing group (Fig. 2).

Fig. 2. Biotoksinomer

Using simple web camera and own designed software, this system is at least 10 times cheaper than similar devices (using infra-red rays camera). This camera in a short period of time (every 300-500 msec. or even 1 sec.), round a clock (24 hours a day) shoots and records Daphnia position, number, size, and movement in chambers. Actually, software performs the analysis of successive photographs captured by webcam, and monitors the number, size and mobility of Daphnia. When the numbers of live Daphnia in the test chamber falls below 50% or more (limit determined by European Daphnia test of toxicity, EC50), the alarm is activated in order to inform that raw water is not correct for the standard treatment into the drinking

water (because of the presence of toxic substances), or that waste water is polluted, or that river water is dangerous for river life, i.e. river ecosystem.

The data is automatically sent to the server, and by SMS or the Internet to the authorities that are in charge to switch off the flow of water. Then it is necessary immediately to switch off the supply of raw water to treatment plant because the water contains some toxin(s) that caused the death (or immobility) of Daphnia. The software automatically sends data to the server and graphically displays the state of Daphnia in the last 24 hours, and throughout the year, what can be monitored online via Internet.

The program could be given the thresholds so that when the number of “fleas” is not into the given interval, the alarm is activated. Quiet alarm is red light on the desk. Sound alarm is emitted by small PC loud speakers. In addition, there is a Skype alarm where PC sends the messages to enlisted users and e-mail alarm with sending e-mails to pre-defined addresses. With installed GSM Gateway program, PC could send additional SMS alarm or necessary information. The data collected from all measure stations are stored on server and presented on its WEB page. There is also a possibility to send current (real-time) and periodical reports to distinct addresses in a different way (e-mail, Skype, SMS).

In addition to WEB report, using developed software - client application, it is possible to reach all the data from any measure point and get miscellaneous reports, control the measure stations on distance, or simple get measure data for own analysis.

2 Data Tracking and Recording

Web camera is shutting the aquarium round o clock, in distinct period of time (300 ms, 600 ms, or 1sec), depending on camera and Daphnia state. After first shutting, we have gray scale picture given in upper part of Fig. 3.

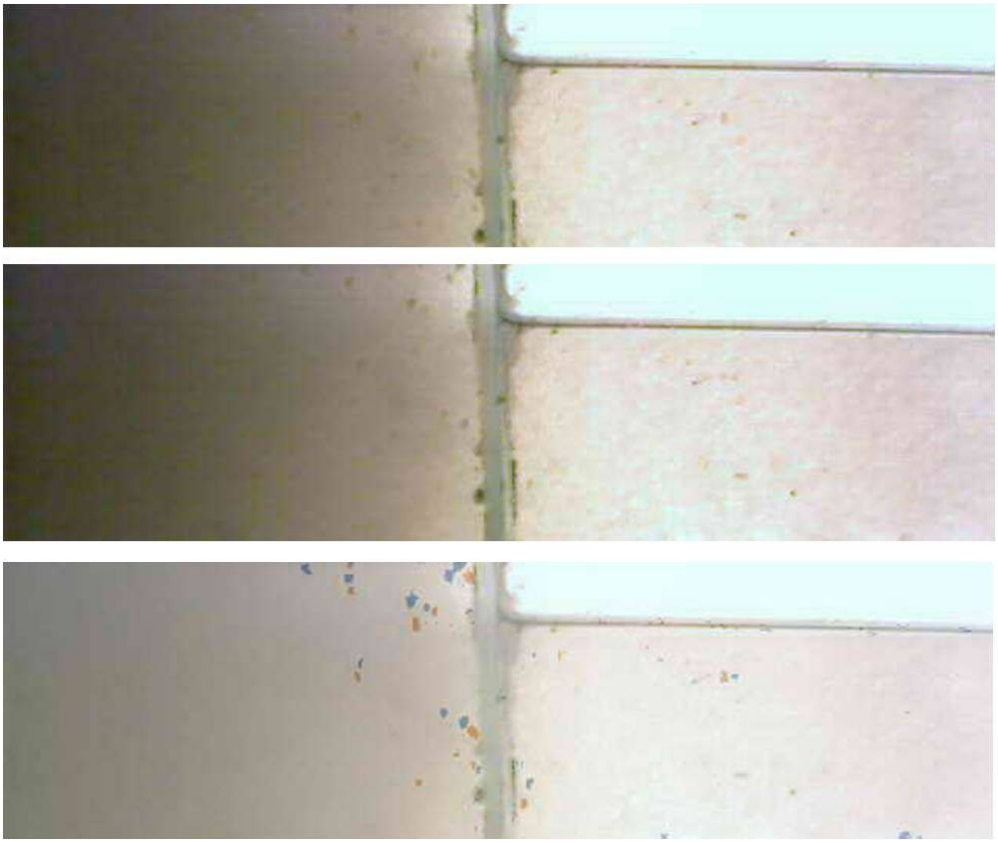

Fig.3. First taken picture (upper part), next (middle part), superimpose (lower part)

The Daphnia coordinates and magnitude (in pixels) are recording in SQL data base. The next picture is taken after some delay, what is in this case given in middle part of Fig. 3.
The position and magnitude are recorded in the same way as from previous picture. The output file with recorded data before and after move, is given in Fig. 4.

```
UTC 2010.09.13 16:19:00.000 --- before move:    UTC 2010.09.13 16:19:01.000 --- after move:

  1.     9 (600,348)-(602,351) [  3 x   4]        1.     9 (312,327)-(316,329) [  5 x   3]
  2.     9 (316,386)-(319,389) [  4 x   4]        2.     9 (300,257)-(303,259) [  4 x   3]
  3.     9 (210,390)-(213,393) [  4 x   4]        3.     9 (255,245)-(258,248) [  4 x   4]
  4.    10 (335,137)-(338,139) [  4 x   3]        4.    10 (296,226)-(298,229) [  3 x   4]
  5.    10 (314,364)-(317,367) [  4 x   4]        5.    11 (374,378)-(378,380) [  5 x   3]
  6.    10 (245,330)-(247,334) [  3 x   5]        6.    11 (381, 56)-(383, 60) [  3 x   5]
  7.    11 (312,308)-(314,311) [  3 x   4]        7.    11 (276,417)-(279,420) [  4 x   4]
  8.    12 (296,  7)-(300,  9) [  5 x   3]        8.    11 (303, 83)-(305, 87) [  3 x   5]
  9.    12 (317,411)-(321,415) [  5 x   5]        9.    12 (288, 44)-(292, 48) [  5 x   5]
 10.    14 (225,383)-(228,388) [  4 x   6]       10.    12 (259,234)-(261,238) [  3 x   5]
 11.    15 (256,382)-(260,386) [  5 x   5]       11.    12 (226,316)-(229,320) [  4 x   5]
 12.    16 (284, 44)-(287, 47) [  4 x   4]       12.    14 (309,223)-(313,227) [  5 x   5]
 13.    17 (308,232)-(311,238) [  4 x   7]       13.    14 (214,411)-(217,415) [  4 x   5]
 14.    18 ( 22,416)-( 27,420) [  6 x   5]       14.    16 (283,381)-(285,387) [  3 x   7]
 15.    18 (299, 96)-(302,101) [  4 x   6]       15.    17 (306,155)-(311,158) [  6 x   4]
 16.    18 (282,414)-(287,418) [  6 x   5]       16.    18 (234,392)-(238,396) [  5 x   5]
 17.    22 (288,394)-(292,403) [  5 x  10]       17.    18 (289,401)-(293,405) [  5 x   5]
 18.    22 (232,356)-(237,361) [  6 x   6]       18.    19 (205,282)-(208,288) [  4 x   7]
 19.    22 (285,304)-(288,310) [  4 x   7]       19.    20 (299, 44)-(302, 49) [  4 x   6]
 20.    24 (216,399)-(220,404) [  5 x   6]       20.    20 (310,281)-(314,287) [  5 x   7]
 21.    24 (445,366)-(450,370) [  6 x   5]       21.    21 (294,291)-(300,296) [  7 x   6]
 22.    25 (311,288)-(315,294) [  5 x   7]       22.    22 (472,369)-(477,374) [  6 x   6]
 23.    26 (345,364)-(350,370) [  6 x   7]       23.    23 (298, 20)-(302, 26) [  5 x   7]
 24.    26 (210,293)-(214,299) [  5 x   7]       24.    24 (340,380)-(346,385) [  7 x   6]
 25.    27 (284,339)-(289,346) [  6 x   8]       25.    24 (286,344)-(289,351) [  4 x   8]
```

```
26.    27 (278,278)-(282,285) [  5 x   8]
27.    27 (272,390)-(277,396) [  6 x   7]
28.    29 (634,129)-(639,135) [  6 x   7]
29.    31 (260,244)-(263,252) [  4 x   9]
30.    31 (298,343)-(304,349) [  7 x   7]
31.    35 (204,336)-(209,344) [  6 x   9]
32.    36 (290,369)-(297,377) [  8 x   9]
33.    38 (265,  4)-(272, 10) [  8 x   7]
34.    40 (240,235)-(245,243) [  6 x   9]
35.    41 (480,366)-(488,373) [  9 x   8]
36.    45 (311,343)-(317,351) [  7 x   9]
37.    46 (251,232)-(257,241) [  7 x  10]
38.    46 (505,207)-(511,215) [  7 x   9]
39.    49 (285,317)-(290,327) [  6 x  11]
40.    52 (361,356)-(368,364) [  8 x   9]
41.    75 (275,  6)-(282, 18) [  8 x  13]
42.    95 (300,326)-(315,337) [ 16 x  12]
```

```
26.    25 (288,  0)-(296,  3) [  9 x   4]
27.    25 (610,132)-(615,138) [  6 x   7]
28.    26 (262,416)-(266,422) [  5 x   7]
29.    26 (215,325)-(220,331) [  6 x   7]
30.    29 (273,395)-(279,401) [  7 x   7]
31.    30 (246,362)-(250,369) [  5 x   8]
32.    31 (308,410)-(313,420) [  6 x  11]
33.    34 (285, 27)-(290, 34) [  6 x   8]
34.    35 (300,365)-(308,370) [  9 x   6]
35.    36 (305,164)-(310,173) [  6 x  10]
36.    37 (273,284)-(277,292) [  5 x   9]
37.    37 (237,218)-(242,226) [  6 x   9]
38.    38 (512,206)-(517,215) [  6 x  10]
39.    38 (389,364)-(394,371) [  6 x   8]
40.    50 (349,366)-(356,372) [  8 x   7]
41.    51 (279,312)-(284,322) [  6 x  11]
42.    52 (257,204)-(262,214) [  6 x  11]
43.    53 (303,320)-(311,329) [  9 x  10]
44.    73 (307,334)-(317,347) [ 11 x  14]
45.    86 (261,382)-(273,394) [ 13 x  13]
46.    86 (272,403)-(282,415) [ 11 x  13]
47.   126 (281,  8)-(299, 21) [ 19 x  14]
```

Fig. 4. Data recording before and after Daphniae move

Comparison of successive photos given in upper and middle part in Fig. 3 gives resulting photo with Daphnias before move (blue) and after move (red), and their superimpose gives picture presented in the lower part of Fig. 3.

That algorithm enables immediately counting of Daphniae. In the left down corner of superimposed picture are given time and number of Daphniae. In the right down corner of picture is given random number generated by position of Daphniae. The resulting graphic, with numbers of Daphnia in last 24 hours and trend with standard deviation in last hour is given in Fig. 5.

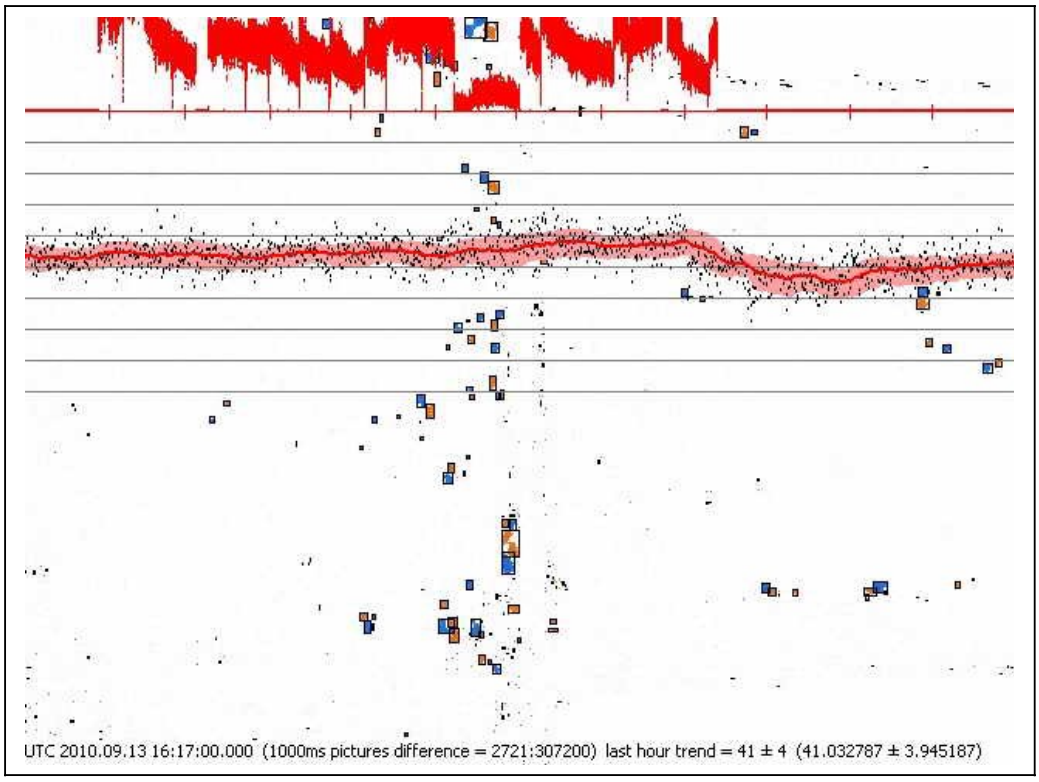

Fig. 5. Round-o-clock number of Daphniae

The data recording round-a-clock for all period the laboratory is working (more than one year) are available on server. SQL data base with cloud technology will enable accessibility to the data globally. The graphic presentation of number of Daphniae in bioassay for longer period of time (in this case, one year), with trends which enable stronger analysis of different influence factors, is also available.

3 Bio-Random Number Generator

It is often to use linear congruent method that for one random number computes next number as $x_{i+1}=(x_i a+c) \bmod m$. Because it always holds $0 \le x < m$, then $0 \le x/m < 1$, so that random number is in interval $(0,1)$. The value x_0 is called seed of random number. Random number generator of hypothetical processor MMIX, which works with 64 bits, has a constants $a=6364136223846793005$, $c=1442695040888963407$, and $m=2^{64}$, what is given in the following code.

```
type octa=int64;
var seed:octa=0;
const
 mmixa=6364136223846793005;   //$5851F42D4C957F2D
 mmixc=1442695040888963407;   //$14057B7EF767814F

function nextseed:octa;
begin
 seed:=seed*mmixa+mmixc;
 result:=seed;
end;
```

In a similar way it is possible to calculate previous random number $x_i=(x_{i+1}b+d) \bmod m$. Herewith we have $((x_i a+c)b+d) \bmod m=(x_i ab+cb+d) \bmod m$. In order to hold for every x, it must be $ab=1 \bmod m$ and $cb=-d \bmod m$. Of course, it holds $x_i=((x_i b+d)a+c) \bmod m$, wherefrom follows $da=-c \bmod m$. Next procedure will compute the corresponding values for a and c, $b=-4\,568\,919\,932\,995\,229\,531$ and $d=-7\,379\,792\,620\,528\,906\,219$.

```
procedure inversion(a,c:octa;var b,d:octa);
var m:octa;
begin
 b:=0;d:=1;m:=0;
 while d<>0 do begin
  m:=m or d;
  if(a*b and m)<>1 then b:=b or d;
  d:=d shl 1;
 end;
 d:=-c*b;
end;
```

This procedure will compute the previous value of seed:

```
const
 mmixb=-4568919932995229531;  //$C097EF87329E28A5
 mmixd=-7379792620528906219;  //$9995B5B621535015

function prevseed:octa
begin
```

```
 seed:=seed*mmixb+mmixd;
 result:=seed;
end;
```

Regardless of using functions **nextseed** or **prevseed**, it is the same generator. The same series of numbers will be generated, but direction depends on function choice. For this purpose, the direction will be defined by logical variable **direction**.

```
var direction:boolean=true;

function calcseed:octa;
begin
 if direction
  then result:=nextseed
  else result:=prevseed;
end;
```

The whole code of random number generator using as a seed Daphnia movement is given in Appendix.

4 Conclusion

The most important advantage of BIOTOKSINOMER is its prevention function, i.e. preventing the negative consequences of delay of results of standard methods for water quality examination. The standard methods usually give late results because taking water samples in discontinuity, in periods of time. Water sampling for BIOTOKSINOMER is continuous and testing proceeds 24 hours a day. Besides, standard methods demand expensive equipment and reagents, high qualified staff, and they are time consuming. Until results of standard analyses reach the responsible persons, water is already consumed (humans, fish, biosphere) and toxins are spreading through organisms. BIOTOKSINOMER solves this problem in quick, cheap, and ecologically efficient way.

Acknowledgement

The device described here has been awarded from Ministry of Science and Technology of Republic Serbia by Diploma for the Best Innovation Idea in the field of Medicine, Health, and Environment in the year 2010. This work is in part supported by the Serbian Ministry of Education and Science (research projects III44006 and TR37003).

References

Knuth, D.E.: The Art of Computer Programming, Addison-Wesley, 1969, vol. 2, pp. 1-160.

Djordjević, Dj., Ignjatović, L.,Nešić, S.: "Biotoksinomer" – A Bio-Device for Water Quality Control, International Conference "Innovation as a Function of Engineering Development", Faculty of Civil Engineering and Architecture, Niš, November 25-26, 2011, Proceedings, pp. 91-96.

DJOKA@NI.AC.RS
SrbaNesic@gmail.com

Appendix

```
unit Nasumica64;

interface

uses classes,idhttp;

type octa=int64;

function rdtsc:octa;
procedure gethttp(url:string;var buf;len:octa);
function getocta(url:string):octa;
function daphniae:octa;

procedure inversion(a,c:octa;var b,d:octa);

var
  seed:octa=0;
  direction:boolean=true;

procedure deadseed;
procedure liveseed;

function nextseed:octa;
function prevseed:octa;
function newseed:octa;
function oldseed:octa;

function uniform:extended; overload;
function uniform(p:extended):extended; overload;
function uniform(p,q:extended):extended; overload;
function anyangle:extended;
function anybyte:byte;
procedure enigma(var s:string);

function exponential:extended; overload;
function exponential(lambda:extended):extended; overload;
function geometric(p:extended):integer;
function normal:extended; overload;
function normal(mu,sigma:extended):extended; overload;
function poisson(lambda:extended):integer;
function chisquare(k:integer):extended;
function studentt(mu:extended;nu:integer):extended; overload;
function studentt(nu:integer):extended; overload;
function snedecorf(d1,d2:integer):extended;

function discrete(n:integer):integer;
```

```
function shuffle(s:string):string;
function ordered(n:integer):string;
function permutation(n:integer):string;
function carddeck:string;
function bingodrum:string;
function tomboladrum:string;
function kenodrum:string;

implementation

function rdtsc:octa; assembler;
asm db 0fh,31h end;

procedure gethttp(url:string;var buf;len:octa);
var
  w:tidhttp;
  s:tmemorystream;
begin
  s:=tmemorystream.create;
  w:=tidhttp.create(nil);
  if len>0 then w.request.contentlength:=len;
  try
    w.get(url,s);
  finally
    s.seek(0,0);
    s.read(buf,len);
  end;
  w.free;
  s.free;
end;

function getocta(url:string):octa;
begin
  gethttp(url,result,sizeof(result));
end;

function daphniae:octa;
begin
  result:=getocta('http://www.dundjer.co.rs/daphniae/liveseed.dat');
end;

procedure inversion(a,c:octa;var b,d:octa);
var m:octa;
begin
  b:=0; d:=1; m:=0;
  while d<>0 do begin
    m:=m or d;
    if (a*b and m)<>1 then b:=b or d;
```

```
    d:=d shl 1;
  end;
  d:=-(c*b);
end;

procedure deadseed;
begin
  seed:=rdtsc;
end;

procedure liveseed;
begin
  seed:=daphniae;
end;

function nextseed:octa;
const
  a=6364136223846793005; //$5851F42D4C957F2D
  c=1442695040888963407; //$14057B7EF767814F
begin
  seed:=seed*a+c;
  result:=seed;
end;

function prevseed:octa;
const
  b=-4568919932995229531; //$C097EF87329E28A5
  d=-7379792620528906219; //$9995B5B621535015
begin
  seed:=seed*b+d;
  result:=seed;
end;

function newseed:octa;
begin
  if direction
    then result:=nextseed
    else result:=prevseed;
end;

function oldseed:octa;
begin
  if direction
    then result:=prevseed
    else result:=nextseed;
end;

function uniform:extended;
```

```
const eps=1/4294967296/4294967296;
begin
  result:=newseed*eps;
  if result<0 then result:=result+1;
end;

function uniform(p:extended):extended;
begin
  result:=p*uniform;
end;

function uniform(p,q:extended):extended;
begin
  result:=p+uniform(q-p);
end;

function anyangle:extended;
begin
  result:=uniform(2*pi);
end;

function anybyte:byte;
begin
  result:=newseed shr 56;
end;

procedure enigma(var s:string);
var
  o:octa;
  i:integer;
begin
  o:=seed;
  for i:=1 to length(s) do
    s[i]:=chr(ord(s[i]) xor anybyte);
  seed:=o;
end;

function exponential:extended;
begin
  result:=-ln(1-uniform);
end;

function exponential(lambda:extended):extended;
begin
  if lambda>0
    then result:=exponential/lambda
    else result:=0;
end;
```

```
function geometric(p:extended):integer;
begin
  if (0<p) and (p<1)
    then result:=trunc(exponential(-ln(1-p)))
    else result:=0;
end;

function normal:extended;
begin
  result:=sqrt(2*exponential)*cos(anyangle);
end;

function normal(mu,sigma:extended):extended;
begin
  result:=mu+sigma*normal;
end;

function poisson(lambda:extended):integer;
begin
  result:=-1;
  repeat
    result:=result+1;
    lambda:=lambda-exponential;
  until lambda<=0;
end;

function chisquare(k:integer):extended;
var i:integer;
begin
  result:=0;
  for i:=1 to k do result:=result+sqr(normal);
end;

function studentt(mu:extended;nu:integer):extended;
var v:extended;
begin
  if nu>0 then begin
    repeat v:=chisquare(nu) until v<>0;
    result:=(normal+mu)*sqrt(nu/v);
  end else result:=0;
end;

function studentt(nu:integer):extended;
begin
  result:=studentt(0,nu);
end;
```

```
function snedecorf(d1,d2:integer):extended;
begin
  if (d1>0) and (d2>0)
    then result:=(chisquare(d1)*d2)/(chisquare(d2)*d1)
    else result:=0;
end;

function discrete(n:integer):integer;
begin
  if n>0
    then result:=trunc(uniform*n)+1
    else result:=trunc(uniform*(1-n));
end;

function shuffle(s:string):string;
var
  i,j:integer;
  c:char;
begin
  for i:=length(s) downto 2 do begin
    j:=discrete(i);
    c:=s[i]; s[i]:=s[j]; s[j]:=c;
  end;
  result:=s;
end;

function ordered(n:integer):string;
var i:integer;
begin
  result:='';
  for i:=1 to n do result:=result+chr(i);
end;

function permutation(n:integer):string;
begin
  result:=shuffle(ordered(n));
end;

function carddeck:string;
begin
  result:=permutation(52);
end;

function bingodrum:string;
begin
  result:=permutation(75);
end;
```

```
function tomboladrum:string;
begin
  result:=permutation(90);
end;

function kenodrum:string;
begin
  result:=permutation(80);
end;

initialization

deadseed;

end.
```