

Nasumica (engleski Random) "Ni po babu, ni po stričevima..."

Često se koristi linearna kongruentna metoda kod koje se od nekog nasumičnog broja računa naredan kao $x_{i+1} = (x_i \cdot a + c) \bmod m$. Kako uvek važi $0 \leq x < m$, onda je $0 \leq x/m < 1$, nasumičan broj u opsegu $[0, 1)$. Vrednost x_0 zove se seme (**seed**).

Generator nasumičnih brojeva hipotetičkog procesora MMIX, koji radi sa 64 bita tačnosti, ima konstante $a = 6\,364\,136\,223\,846\,793\,005$, $c = 1\,442\,695\,040\,888\,963\,407$ i $m = 2^{64}$.

```
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;
```

Na sličan način, moguće je izračunati prethodan nasumičan broj, $x_i = (x_{i+1} \cdot b + d) \bmod m$. Odavde je $x_i = ((x_i \cdot a + c) \cdot b + d) \bmod m = (x_i \cdot a \cdot b + c \cdot b + d) \bmod m$. Da bi ovo važilo sa svaku vrednost x , mora biti $a \cdot b = 1 \bmod m$ i $c \cdot b = -d \bmod m$. Naravno, važi i $x_i = ((x_i \cdot b + d) \cdot a + c) \bmod m$, odakle je $d \cdot a = -c \bmod m$. Sledeća procedura će izračunati da navedenim vrednostima a i c odgovaraju vrednosti $b = -4\,568\,919\,932\,995\,229\,531$ i $d = -7\,379\,792\,620\,528\,906\,219$.

```
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;
```

Ova procedura će izračunati prethodnu vrednost semena generatora.

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

function prevseed:octa;
begin
  seed:=seed*mmixb+mmixd;
  result:=seed;
end;
```

Bez obzira da li koristili funkciju `nextseed` ili `prevseed`, radi se o istom generatoru. Isti niz brojeva će biti generisan, samo će od izbora funkcije zavistiti smer generisanja. U tu svrhu, smer generisanja određuje logička promenljiva `direction`.

```

var direction:boolean=true;
function calcseed:octa;
begin
  if direction
    then result:=nextseed
    else result:=prevseed;
  end;
end;

```

Promenljive tipa octa imaju znak i nalaze se u opsegu $[-2^{63}, 2^{63})$, pa bi deljenjem sa 2^{64} dobili broj u opsegu $[-0.5, 0.5)$. Ovaj problem može biti prevaziđen na dva jednostavna načina: jedan je da rezultatu dodamo 0.5, a drugi da samo negativnom rezultatu dodamo 1. Kada se koristi drugi način, ustvari, izbacuje se znak broja. Definisaćemo funkciju `uniform` koja će navedenom metodom vraćati broj u opsegu $[0, 1)$ po uniformnoj raspodeli.

```

const
  eps16=1/65536;
  eps32=eps16*eps16;
  eps64=eps32*eps32;
function uniform:extended; overload;
begin
  result:=calcseed*eps64;
  if result<0 then result:=result+1;
end;

```

Nasumičan broj u opsegu $[0, p)$

```

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

```

Nasumičan broj u opsegu $[p, q)$

```

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

```

Nasumičan ugao

```

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

```

Nasumičan bajt

```

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

```

Kako u početnu vrednost semena ubaciti neku nasumičnu vrednost? Često se koristi, takozvani, Time Stamp Counter procesora i čija vrednost može biti saznata sledećom funkcijom:

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

Procedura `deadseed` će u seme ubaciti vrednost procesorovog brojača i uobičajeno je da se u programu pozove samo jednom, najčešće negde na početku.

```
procedure deadseed;
begin
  seed:=rdtsc;
end;
```

Početna vrednost semena može biti definisana i preko interneta, a evo kako. Prvo definišemo funkciju `gethttp` koja će za navedeni argument `url`, koji pokazuje na neku datoteku na internetu, kao rezultat vratiti `string` u kome se nalazi kompletan sadržaj te datoteke. U klauzuli `uses` na početku programa, mora biti uključen unit `idhttp`.

```
function gethttp(url:string):string;
var w:tidhttp;
begin
  w:=tidhttp.create(nil);
  try result:=w.get(url) except end;
  w.free;
end;
```

Funkcija `getocta` će uzeti datoteku sa interneta i prvih 8 bajtova (64 bita) te datoteke prebaciti u rezultat funkcije.

```
function getocta(url:string):octa;
begin
  move(gethttp(url)[1],result,sizeof(result));
end;
```

Na adresi <http://www.dundjer.co.rs/daphniae/liveseed.dat> svakog minuta se generiše 64-bitno seme koje predstavlja razliku u bitovima dve slike akvarijuma sa dafnijama (vodene buve), slikane u razmaku od 1 sekunde. Ukoliko nema živih dafnija rezultat će biti 0.

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

I sada je sve jednostavno. Pravo, živo (`live`), non-pseudo, seme generatora nasumičnih brojeva.

```
procedure liveseed;
begin
  seed:=daphniae;
end;
```

Nasumični brojevi po nekim raspodelama

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

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

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

function normal(mu,sigma:extended):extended; overload;
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; overload;
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; overload;
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;

procedure shuffle(var s: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;
end;

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

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

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

function bingo drum:string;
begin
  result:=permutation(75);
end;

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

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

```