**--in file randint.ads**

**package randint is** -- Generates a uniformly distributed random integer

**function next\_int return integer;** -- between 0 and 100.

**end randint;**

**--in file randint.adb**

**Package body randint is**

**x: integer := 737;** -- seed, prime numbers best

**function next\_int return integer is**

**n : integer;**

**begin**

**x := x\*29+37;**

**n := x;**

**x := x mod 1001;** -- prevent overflow

**return n mod 101;** -- number between 0 and 100

**end next\_int;**

**end randint;**

**--in file randflo.ads**

**package randflo is** --Generates a uniformly distributed random real

**function next\_float return float;** -- between 0.0 <= next <= 1.0.

**end randflo;**

For Ada’s built in random number generation facility use package Ada.Numerics.Discrete\_Random

for integers and

Package Ada.Numerics.Float\_Ranom

for floating point values. Both packages are generics.

**--in file randflo.adb**

**Package body randflo is**

**x: integer := 737;**

**function next\_float return float is**

**n : integer;**

**begin**

**x := x\*29+37;**

**n := x;**

**x := x mod 1001;**

**return float(n mod 101) / 100.0;**

**end next\_float;**

**end randflo;**

**--in file expdist.ads**

**package expdist is** --Approximate the exponential distribution.

**function next\_exp return float;**

**end expdist;**

**--in file expdist.adb**

**with randflo; use randflo;**

**package body expdist is**

**function next\_exp return float is**

**x: float;**

**begin**

**x:= next\_float;** -- Return next exponential arrival interval.

**if x = 0.0 then** -- Mean arrival time = 1.

**return 0.0;**

**elsif x <= 0.1 then**

**return ((x - 0.0) \* 1.04 + 0.0);**

**elsif x <= 0.2 then**

**return ((x - 0.1) \* 1.18 + 0.104);**

**elsif x <= 0.3 then**

**return ((x - 0.2) \* 1.33 + 0.222);**

**elsif x <= 0.4 then**

**return ((x - 0.3) \* 1.54 + 0.355);**

**elsif x <= 0.5 then**

**return ((x - 0.4) \* 1.81 + 0.509);**

**elsif x <= 0.6 then**

**return ((x - 0.5) \* 2.25 + 0.690);**

**elsif x <= 0.7 then**

**return ((x - 0.6) \* 2.85 + 0.915);**

**elsif x <= 0.75 then**

**return ((x - 0.70) \* 3.60 + 1.2);**

**elsif x <= 0.8 then**

**return ((x - 0.75) \* 4.40 + 1.38);**

**elsif x <= 0.84 then**

**return ((x - 0.8) \* 5.75 + 1.60);**

**elsif x <= 0.88 then**

**return ((x - 0.84) \* 7.25 + 1.83);**

**elsif x <= 0.9 then**

**return ((x - 0.88) \* 9.00 + 2.12);**

**elsif x <= 0.92 then**

**return ((x - 0.90) \* 11.0 + 2.30);**

**elsif x <= 0.94 then**

**return ((x - 0.92) \* 14.5 + 2.52);**

**elsif x<= 0.95 then**

**return ((x - 0.94) \* 18.0 + 2.81);**

**elsif x <= 0.96 then**

**return ((x - 0.95) \* 21.0 + 2.99);**

**elsif x <= 0.97 then**

**return ((x - 0.96) \* 30.0 + 3.20);**

**elsif x <= 0.98 then**

**return ((x - 0.97) \* 40.0 + 3.50);**

**elsif x <= 0.99 then**

**return ((x - 0.98) \* 70.0 + 3.90);**

**elsif x <= 0.995 then**

**return ((x - 0.99) \* 140.0 + 4.60);**

**elsif x <= 0.998 then**

**return ((x - 0.995) \* 300.0 + 5.30);**

**elsif x <= 0.999 then**

**return ((x - 0.998) \* 800.0 + 6.20);**

**else**

**return ((x - 0.9997) \* 1000.0 + 8.0);**

**end if;**

**end next\_exp;**

**end expdist;**

--in file randuse.adb

**with expdist; use expdist;**

**with randflo, randint; use randflo, randint;**

**with Ada.Text\_io; use Ada.Text\_io;**

**with Ada.Integer\_Text\_IO, Ada.Float\_Text\_IO;**

**use Ada.Integer\_Text\_IO, Ada.Float\_Text\_IO;**

**procedure RandUse is**

**a,b,c: integer;**

**x,y,z: float;**

**begin**

**for i in 1..20 loop**

**for j in 1..10 loop**

**a := next\_int; a := (a mod 10)+1;**  --Random variant from 1 to 10.

**put(a); put(" ");**

**end loop;**

**new\_line;**

**end loop;**

**loop**

**for i in 1..20 loop**

**x := next\_exp; put("x = "); put(x,6,3,0); new\_line; x := x \* 10.0;**

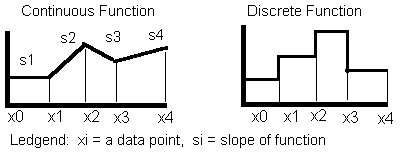
--Random variant exponentially distributed with a mean of 10 time units.

**put("exp with mean of 10 "); put(x,6,3,0); new\_line;**

**end loop;**

**end loop;**

**end RandUse;**



**To approximate a continuous distribution use extrapolation:**

**a) generate a uniformly distributed random variant 0.0 <= ri <=1.0**

**b) the value of the function, yi, at point ri is**

**yi = (ri - xi) \* si + xi for i such that xi <= ri < xi+1.**

**[i.e., yi = distance past base point \* slope + base point]**

**To approximate a discrete function:**

**a) generate a uniformly distributed random variant 0.0 <= ri <=1.0**

**b) select yi from the graph region where xi <= ri < xi+1.**

**Assume we wish to simulate arrivals at a grocery store exponentially distributed with a mean of 20 starting at time 0. Each time we wish to schedule a customer, get their inter-arrival time and add it to the current clock time. Keep the events in order by time.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer** | **Next\_Exp** | **Next\_Exp \* 20** | **Arrival Time (minutes)** |
| 1 | 0.104 | 2.08 | 2.08 |
| 2 | 0.470 | 9.40 | 11.48 |
| 3 | 0.087 | 1.74 | 13.22 |
| 4 | 4.600 | 92.0 | 105.22 |
| 5 | 0.137 | 2.74 | 107.96 |
| 6 | 0.000 | 0.0 | 107.96 |