Liczby losowe

Generacja liczb losowych jest bardzo przydatna w wielu obszarach począwszy od komputerowych gier w kości a na skomplikowanych symulacjach mechanicznych i kryptografii skończywszy. Poniższe zadania przedstawiają mechanizm generacji liczb pseudolosowych w języku C/C++[1].

Generatory liczb pseudolosowych wymagają zastosowania tzw. ziarna czyli liczby, która posłuży do inicjalizacji procesu losowania kolejnych liczb. Zazwyczaj do inicjalizacji generatora używa się czasu sczytywanego z procesora komputera, więc konieczne będzie dołączenie biblioteki `time.h` Poniższy program generuje losową liczbę całkowitą z przedziału 0 do RAND_MAX. RAND_MAX jest zdefiniowane w bibliotece `stdlib.h`, jest ono postaci 2^{n-1} np. 32767.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

void main()
{
     int i;

     srand ( time(NULL) );        //inicjalizacja generatora
     i = rand();                  //funkcja losująca liczbę
     printf(" Wylosowana liczba to %d\n", i);
}
```

Ćwiczenia

- Wykonaj powyższy program kilka razy z rzędu. Co możesz powiedzieć o wyniku?
- Zastąp ziarno generatora `time(NULL)` stałą liczbą. Co się wówczas stanie?
- Zmodyfikuj program tak, by losował ciąg 40 liczb psedudolosowych i wypisywał je na ekran.
- Wypisz na ekran wartość RAND_MAX.

[1] Mówimy, że generowane liczby są pseudolosowe, gdyż niemożliwe jest wygenerowanie prawdziwie losowego ciągu liczb. W praktyce świetnym źródłem ciągów liczb losowych są dane uzyskiwane z natury np. dane meteorologiczne (www.random.org).

Zazwyczaj interesuje nas konkretny przedział liczb. Aby określić kres dolny oraz długość takiego przedziału musimy dokonać kilku transformacji na wylosowanych liczbach: Po prostu bierzemy wyłącznie resztę z dzielenia przez zadaną długość przedziału i dodajemy kres dolny jak w poniższym fragmencie kodu:

```
//  kod losuje liczby z przedzialu [ 20 do 50 ]

    int min = 20;
    int max = 50;
    int L = max - min + 1;

    i = rand()%L + min;
```

operator % liczy resztę z dzielenia jednej liczby przez drugą. Tutaj, licząc resztę z dzielenia wyniku losowania przez 31, otrzymujemy liczbę z przedziału 0 do 30.

Uwaga

Czy taki wzór będzie generował liczby losowe o rokładzie równomiernym? W domu zastanów się nad lepszym rozwiązaniem, a poniżej wykorzystaj to najprostsze.

Ćwiczenia

- Zmodyfikuj powyższy program tak, aby generował liczby z przedziału od 0 do 100.
- Wybierz krótki przedział (rzędu kilku możliwych do wylosowania liczb) i wykonaj 1000 oraz 10000 losowań. W obu przypadkach zliczaj, ile razy wylosowano każdą liczbę.
- Wynik (informację o tym, ile razy wylosowano każdą z wartości) wydrukuj na ekranie. Co możesz powiedzieć o rozkładzie tego losowania?

Liczby losowe typu rzeczywistego, rzutowanie

Do tej pory losowane liczby były liczbami całkowitymi. Często potrzebujemy liczb zmiennoprzecinkowych. W tym celu po prostu przeskalujemy wylosowaną liczbę tak, aby zawsze należała do przedziału 0-1. Wystarczy podzielić wylosowaną

liczbę przez RAND_MAX. Dodatkowo należy pamiętać, że wylosowana liczba jest typu całkowitego `int`, więc dzielenie przez RAND_MAX da nam zazwyczaj 0. Aby tego uniknąć, musimy przekonwertować typ `int` na typ zmiennoprzecinkowy `double`. Taką operację nazywamy rzutowaniem. Rzutowanie ma bardzo szerokie zastosowanie i odnosi się nie tylko do operacji na liczbach. Poniższy kod losuje liczbę z przedziału 0-1:

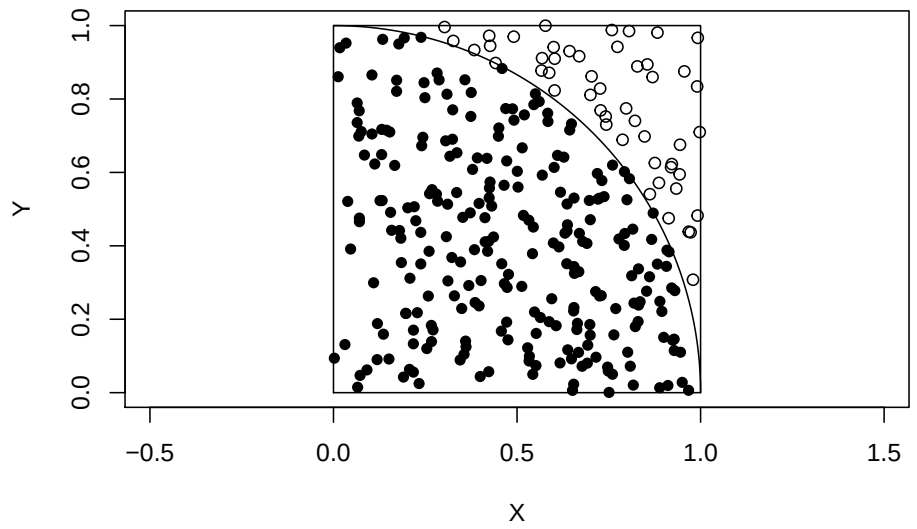
```
double x;
srand(  time  (NULL)  );

// instrukcja (double) rzutuje typ int na double
x = (double)rand()/RAND_MAX;
```

Ćwiczenia

- Napisz funkcję `isInCircle` typu `bool`, która będzie losowała punkt w kwadracie $[0,1] \times [0,1]$ i zwracała wartość 'true', jeśli punkt znajduje się wewnątrz koła o promieniu 1. Jeśli punkt jest poza kołem, niech funkcja zwraca 'false'.
- W programie głównym sprawdź, czy punkt wylosowany przez daną funkcję jest w obszarze koła, czy nie. Zauważ, że nie masz dostępu do współrzędnych wylosowanych przez funkcję. Funkcja zwraca tylko wartość logiczną.
- Niech powyższa funkcja dodatkowo zaznacza dany punkt na ekranie. Jeśli jest on wewnątrz koła, niech oznacza go kółkiem. Jeśli jest na zewnątrz, niech oznacza go krzyżykiem. Najwygodniej będzie przeskalować dane zaznaczenie 200-krotnie. Dodatkowo, narysuj na ekranie ćwiartkę koła o promieniu 200 oraz kwadrat $[0,200]\times[0,200]$, aby było widać, że wylosowany punkt rzeczysiście jest wewnatrz koła.
- Wywołaj powyższą funkcję 100-krotnie, aby sprawdzić, jak działa[2]. Wynik działania powinien przypominać rysunek poniżej.

[2]Funkcję zwracajacą typ możemy wywołać zarówno przypisując wartość, która jest przez nią zwracana do jakiejś zmiennej np. `a = isInCircle()` jak i po prostu wywołać ją w programie bez przypisania, pisząc po prostu `isInCircle()`. Wówczas wartość zwracana przez funkcję przepadnie, ale wszystkie inne rzeczy, które się dzieją w funkcji (u nas jest to rysowanie), zostaną i tak wykonane



Liczenie przybliżeń liczby π metodą Monte Carlo

Koło wpisane w jednostkowy kwadrat ma powierzchnię równą $\pi/4$. Toteż losując punkty w kwadracie z prawdopodobieństwem $\pi/4$ trafiamy w obszar koła. Dzięki temu, zliczając odsetek wylosowanych punktów, które trafiły do wnętrza koła, możemy określić przybliżenie liczby π. Taka metoda zalicza się do metod Monte Carlo[3].

Ćwiczenia

- Przy użyciu poprzedniej funkcji policz przybliżenie liczby π. Wykonaj 10000 losowań.

[3]w Monte Carlo mieści się bodaj najsłynniejsze w Europie kasyno, stąd taka nazwa dla metod losowych

- Sprawdź, jak zmienia się dokładność przybliżeń, gdy wykonujemy coraz więcej losowań. Wykonaj 100, 1000 i 100000 losowań.
- – Wydrukuj na ekran zależność względnego błędu przybliżenia π od potęgi liczby 2 w zakresie 2^8 do 2^{32}.