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 odczytywanego z komputera, więc konieczne będzie dołączenie biblioteki `time.h`

Poniższy program generuje losową liczbę całkowitą z przedziału 0 do RAND_MAX. Wartość RAND_MAX jest zdefiniowana w bibliotece `stdlib.h` i jest 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 losujaca liczbe
  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, aby losował ciąg 40 liczb pseudolosowych 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 wartość minimalną oraz długość takiego przedziału musimy dokonać kilku transformacji na wylosowanych liczbach. Jedną z metod jest wzięcie wyłącznie reszty z dzielenia przez zadaną długość przedziału i dodanie wartości minimalnej Przykładowo:

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

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

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

operator „%” umożliwia wyznaczenie reszty 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 rozkładzie równomiernym? Czy możesz podać lepsze rozwiązanie?

Ćwiczenia

- Zmodyfikuj powyższy program tak, aby generował liczby z przedziału od 1 do 8.
- Wykonaj 1000 oraz 10000 losowań. W obu przypadkach zliczaj, ile razy wylosowano każdą liczbę z ustalonego zakresu (tzn. od 1 do 8).
- Wynik (informację o tym, ile razy wylosowano każdą z wartości) wydrukuj na ekranie. Co możesz powiedzieć o rozkładzie tego losowania?
- Napisz funkcję, która losuje 5 liczb z zakresu od 1 do 8. Wylosowane wartości powinny być zapisane pod adresami podanymi jako argumenty funkcji.

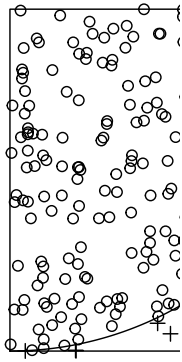
Liczby losowe typu rzeczywistego, rzutowanie

Do tej pory losowane liczby były liczbami całkowitymi. Często potrzebujemy liczb zmiennoprzecinkowych. Aby je otrzymać, 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 prze-konwertować 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));

// wyrazenie (double), rzutuje typ na double
x = (double) rand() / RAND_MAX;
```



Ćwiczenia

- Napisz funkcję `isInCircle()`, która będzie losowała punkt w kwadracie o wymiarach $[0, 1] \times [0, 1]$ i zwracała wartość 1 jeśli punkt znajduje się wewnątrz koła o promieniu $R = 1$. Jeśli punkt jest poza kołem, niech funkcja zwraca 0.
- W programie głównym sprawdź czy punkt wylosowany przez daną funkcję leży w obszarze koła, czy nie i wyświetl stosowny komunikat. Zauważ, że nie masz dostępu do współrzędnych wylosowanych przez funkcję. Funkcja zwraca tylko zero lub jedynkę.
- 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. Ponieważ wylosowane punkty znajdują się w obszarze kwadratu o wymiarach $[0, 1] \times [0, 1]$, wygodnie będzie przeskalować współrzędne 200-krotnie. Dodatkowo, narysuj na ekranie odpowiednio przeskalowaną ćwiartkę koła oraz kwadrat, tak aby było widać, że wylosowany punkt rzeczywiście jest wewnątrz koła.
- Wywołaj powyższą funkcję 100-krotnie, aby sprawdzić, jak działa[2]. Wynik działania powinien przypominać rysunek poniżej.

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. Wiedząc to, możemy wyznaczyć przybliżenie liczby π. Obliczamy iloraz liczby punktów, które znalazły się wewnątrz koła do całkowitej liczby punktów i mnożymy przez 4. Taka metoda zalicza się do metod Monte Carlo[3].

Ćwiczenia

- Przy użyciu poprzedniej funkcji policz przybliżenie liczby π. Wykonaj 10000 losowań.
- Sprawdź, jak zmienia się dokładność przybliżeń, gdy wykonujemy więcej losowań. Wykonaj 100, 1000 i 100000 losowań.

[2] Funkcję zwracającą 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 `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

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

- Wydrukuj na ekranie zależność względnego błędu przybliżenia liczby π od liczby losowań. Niech liczba losowań będzie równa potędze liczby 2 w zakresie 2^8 do 2^{32}.