
Generowanie wartości losowych

W C++ mamy dwa sposoby generowania liczb pseudolosowych:

1. opierajac się na wywodzącej się z jezyka C funkcji `int rand()` z biblioteki <cstdlib>
2. korzystajac ze standardowej biblioteki `<random>`

Korzystanie z `rand()`

```
#include <cstdlib>
#include <iostream>
#include <ctime>

int main()
{
    std::srand(std::time(nullptr)); // use current time as seed for random generator
                                    // without this rand is called with srand(1)
    int random_variable = std::rand();
    std::cout << "Random value on [0 " << RAND_MAX << "]: "
              << random_variable << '\n';

    // roll a 6-sided die 20 times
    for (int n=0; n != 20; ++n) {
        int x = 7;
        while(x > 6)
            x = 1 + std::rand()/((RAND_MAX + 1u)/6);  // Note: 1+rand()%6 is biased
        std::cout << x << ' ';
    }
}
```

Korzystanie z `<random>`

Biblioteka oferuje rozmaite generatory liczb pseudolosowych w tym generujące liczby o rozkładzie jednostajnym, normalnym, Bernouliego, `std::discrete_distribution` itd.

- Najprostszy, bardzo podobny do `rand()`, wersja 1

```
#include <random>
#include <iostream>

int main(){

    std::random_device rd1;     // inicjalizacja obiektu reprezentujacego
                                // generator liczb pseudolosowych

    std::cout << "rd.min(): " << rd1.min()
              << "\nrd.max(): " << rd1.max()
              << '\n';

    for(int i = 0; i<20 ; i++){
        std::cout << rd1() << " ";  // kolejna liczba pseudolosowa otrzyywana
                                    // jest przez wywołanie rd1()
                                    // zwracany typ: unsigned int
    }
```

```
    std::cout << '\n';
}
```

- Najprostszy, bardzo podobny do rand(), wersja 1

```
#include <random>
#include <iostream>

int main()
{
    std::random_device rd;  //jako seed dla generatora

    // generator zainicjalizowany losową wartością
    // przy każdym uruchomieniu programu zwróci inne wartości
    std::default_random_engine generator1(rd());

    // generator zainicjalizowany ustaloną wartością
    // przy kazdym uruchomieniu programu zwróci takie same wartości
    std::default_random_engine generator2(2);

/* domyślnie zwracanym typem jest unsigned int
 *
 * niektóre inne dostępne generatory
 * std::minstd_rand -- linear congruential generator
 * std::mt19937 -- Mersene twister, wysoka jakość
 */

    for (int n=0; n<30; ++n)
        std::cout << generator1() << ' ';
    std::cout << '\n';

    for (int n=0; n<30; ++n)
        std::cout << generator2() << ' ';
    std::cout << '\n';

}
```

- liczby o rozkładzie jednostajnym

```
#include <random>
#include <iostream>

int main()
{
    std::random_device rd;  //jako seed dla generatora

    // generator zainicjalizowany losową wartością
    // przy każdym uruchomieniu programu zwróci inne wartości
    std::default_random_engine generator1(rd());

    // generator zainicjalizowany ustaloną wartością
    // przy kazdym uruchomieniu programu zwróci takie same wartości
    std::default_random_engine generator2(2);

    std::cout << "\n--------------------------------\n";
    std::cout << "Z rozkładem jednostajnym na [a,b]:\n";

    // przekształca wartość zwracana przez generator
```

```
    // na liczbę zgodną z rozkładem  P(x|a,b) = 1/(b-a+1)
    // domyślnie zwracanym typem jest int
    std::uniform_int_distribution<> distribution1(-10, 10);

    std::cout << "Parametry rozkładu:"
              << "\ndistribution1.a: " << distribution1.a()
              << "\ndistribution1.b: " << distribution1.b()
              << '\n';

    std::cout << "\nZ użyciem generator1 - losowe wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution1(generator1) << ' ';
    std::cout << '\n';

    std::cout << "\nZ użyciem generator2 - zawsze te same wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution1(generator2) << ' ';
    std::cout << '\n';

}
```

- liczby o rozkładzie normalnym

```
#include <random>
#include <iostream>

int main()
{
    std::random_device rd;  //jako seed dla generatora

    // generator zainicjalizowany losową wartością
    // przy każdym uruchomieniu programu zwróci inne wartości
    std::default_random_engine generator1(rd());

    // generator zainicjalizowany ustaloną wartością
    // przy kazdym uruchomieniu programu zwróci takie same wartości
    std::default_random_engine generator2(2);

    std::cout << "\n--------------------------------\n";
    std::cout << "Z rozkładem normalnym o średniej mean i odchyleniu stddev:\n";

    // przekształca wartość zwracana przez generator na liczbę zgodną z rozkładem
    // domyślnie zwracanym typem jest double
    std::normal_distribution<> distribution2 (4, 20);

    std::cout << "Parametry rozkładu:"
              << "\ndistribution2.mean: " << distribution2.mean()
              << "\ndistribution2.stddev: " << distribution2.stddev()
              << '\n';

    std::cout << "\nZ użyciem generator1 - losowe wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution2(generator1) << ' ';
    std::cout << '\n';

    std::cout << "\nZ użyciem generator2 - zawsze te same wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution2(generator2) << ' ';
    std::cout << '\n';
```

```
}
```

- liczby o rozkładzie dyskretnym

```cpp
#include <random>
#include <iostream>

int main()
{
    std::random_device rd;  //jako seed dla generatora

    // generator zainicjalizowany losową wartością
    // przy każdym uruchomieniu programu zwróci inne wartości
    std::default_random_engine generator1(rd());

    // generator zainicjalizowany ustaloną wartością
    // przy kazdym uruchomieniu programu zwróci takie same wartości
    std::default_random_engine generator2(2);

    std::cout << "\n--------------------------------\n";
    std::cout << "Z rozkładem dyskretnym na [0,n):\n";

    std::discrete_distribution<> distribution3 ({5,1,4,4,1,5});
    // jako argument podaje się wagi w_i, wtedy P(i) = w_i/Suma wag

    std::vector<double> p = distribution3.probabilities();
    std::cout << "Parametry rozkładu: ";
    for(auto n : p)
        std::cout << n << ' ';
    std::cout << '\n';

    std::cout << "\nZ użyciem generator1 - losowe wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution3(generator1) << ' ';
    std::cout << '\n';

    std::cout << "\nZ użyciem generator2 - zawsze te same wartości\n";
    for (int n=0; n<30; ++n)
        std::cout << distribution3(generator2) << ' ';
    std::cout << '\n';
}
```