

# Metody Monte Carlo

## Laboratorium 2

### Zadanie 2

#### Kod

```
#include <iostream>
#include <random>
#include <map>
#include <vector>
#include <gsl/gsl_rng.h>

std::map<unsigned long int, unsigned long int> generator(const gsl_rng_type *type,
int k, int n)
{
    gsl_rng_env_setup();

    const gsl_rng_type *T = type;
    gsl_rng *r = gsl_rng_alloc(T);

    std::map<unsigned long int, unsigned long int> samples;

    for (int i = 0; i < k; i++)
    {
        samples.insert(std::pair<unsigned long int, unsigned long int>(i, 0));
    }

    for (int i = 0; i < n; i++)
    {
        unsigned long int s = gsl_rng_uniform(r) * k;
        ++samples[int(s)];
    }

    gsl_rng_free(r);

    return samples;
}

int main()
{
    std::vector<int> keys;
    keys.push_back(11);
    keys.push_back(51);
    keys.push_back(101);

    std::vector<double> critical_values_0_05;
    critical_values_0_05.push_back(18.307); // for df = 10
    critical_values_0_05.push_back(67.505); // for df = 50
    critical_values_0_05.push_back(124.342); // for df = 100

    int n = 100000;

    // Good generator case
    for (int i = 0; i < 3; i++)
    {
        int k = keys[i];
        double average_value = n / k;
        std::map<unsigned long int, unsigned long int> samples =
generator(gsl_rng_ranlux, k, n);
```

```

    unsigned long int sum = 0;

    for (int i = 0; i < k; i++)
    {
        sum += samples[i] * samples[i];
    }

    double result = ((double)k * (double)sum) / (double)n - (double)n;

    std::cout << std::endl;
    std::cout << "Result from GOOD rng generator for " << n << " total probes and "
<< k << " intervals is " << result << std::endl;

    if (result > critical_values_0_05[i])
    {
        std::cout << "Result " << result << " is higher than critical value of " <<
critical_values_0_05[i] << std::endl;
    }
    else
    {
        std::cout << "Result " << result << " is lower than critical value of " <<
critical_values_0_05[i] << std::endl;
    }
}

// Bad generator case
for (int i = 0; i < 3; i++)
{
    int k = keys[i];
    double average_value = n / k;
    std::map<unsigned long int, unsigned long int> samples =
generator(gsl_rng_rand, k, n);
    unsigned long int sum = 0;

    for (int i = 0; i < k; i++)
    {
        sum += samples[i] * samples[i];
    }

    double result = ((double)k * (double)sum) / (double)n - (double)n;

    std::cout << std::endl;
    std::cout << "Result from BAD rng generator for " << n << " total probes and "
<< k << " intervals is " << result << std::endl;

    if (result > critical_values_0_05[i])
    {
        std::cout << "Result " << result << " is higher than critical value of " <<
critical_values_0_05[i] << std::endl;
    }
    else
    {
        std::cout << "Result " << result << " is lower than critical value of " <<
critical_values_0_05[i] << std::endl;
    }
}
return 0;
}

```

# Wyniki

---

## Wynik programu dla $n = 1000$ (tysiąc)

Result from GOOD rng generator for 1000 total probes and 11 intervals is 20.58

Result 20.58 is higher than critical value of 18.307

Result from GOOD rng generator for 1000 total probes and 51 intervals is 67.226

Result 67.226 is lower than critical value of 67.505

Result from GOOD rng generator for 1000 total probes and 101 intervals is 142.714

Result 142.714 is higher than critical value of 124.342

Result from BAD rng generator for 1000 total probes and 11 intervals is 8.942

Result 8.942 is lower than critical value of 18.307

Result from BAD rng generator for 1000 total probes and 51 intervals is 63.044

Result 63.044 is lower than critical value of 67.505

Result from BAD rng generator for 1000 total probes and 101 intervals is 114.232

Result 114.232 is lower than critical value of 124.342

---

## Wynik programu dla $n = 10000$ (dziesięć tysięcy)

Result from GOOD rng generator for 10000 total probes and 11 intervals is 11.6676

Result 11.6676 is lower than critical value of 18.307

Result from GOOD rng generator for 10000 total probes and 51 intervals is 58.5464

Result 58.5464 is lower than critical value of 67.505

Result from GOOD rng generator for 10000 total probes and 101 intervals is 120.624

Result 120.624 is lower than critical value of 124.342

Result from BAD rng generator for 10000 total probes and 11 intervals is 10.6798

Result 10.6798 is lower than critical value of 18.307

Result from BAD rng generator for 10000 total probes and 51 intervals is 56.588

Result 56.588 is lower than critical value of 67.505

Result from BAD rng generator for 10000 total probes and 101 intervals is 98.7274

Result 98.7274 is lower than critical value of 124.342

---

## Wynik programu dla $n = 100000$ (sto tysięcy)

Result from GOOD rng generator for 100000 total probes and 11 intervals is 3.7698

Result 3.7698 is lower than critical value of 18.307

Result from GOOD rng generator for 100000 total probes and 51 intervals is 39.5957

Result 39.5957 is lower than critical value of 67.505

Result from GOOD rng generator for 100000 total probes and 101 intervals is 89.7254

Result 89.7254 is lower than critical value of 124.342

Result from BAD rng generator for 100000 total probes and 11 intervals is 6.8201

Result 6.8201 is lower than critical value of 18.307

Result from BAD rng generator for 100000 total probes and 51 intervals is 53.4493

Result 53.4493 is lower than critical value of 67.505

Result from BAD rng generator for 100000 total probes and 101 intervals is 118.543

Result 118.543 is lower than critical value of 124.342

---

## Wynik programu dla $n = 1000000$ (milion)

Result from GOOD rng generator for 1000000 total probes and 11 intervals is 6.53801

Result 6.53801 is lower than critical value of 18.307

Result from GOOD rng generator for 1000000 total probes and 51 intervals is 44.1295

Result 44.1295 is lower than critical value of 67.505

Result from GOOD rng generator for 1000000 total probes and 101 intervals is 95.0461

Result 95.0461 is lower than critical value of 124.342

Result from BAD rng generator for 1000000 total probes and 11 intervals is 6.39226

Result 6.39226 is lower than critical value of 18.307

Result from BAD rng generator for 1000000 total probes and 51 intervals is 57.7413

Result 57.7413 is lower than critical value of 67.505

Result from BAD rng generator for 1000000 total probes and 101 intervals is 121.386

Result 121.386 is lower than critical value of 124.342

Jak możemy zauważyć, im więcej iteracji w badaniu (liczba  $n$ ) tym dystrybucja elementów jest bardziej zróżnicowana. Dla przypadku  $n = 1000$  (tysiąc) możemy zauważyć przekroczenie rozbieżności 5%, i co ciekawe jest to dla “dobrego” generatora (Ranlux), który został polecony w zadaniu:

*“wartości testu chi-kwadrat dla dwóch generatorów – „dobrego” (np. Mersenne Twister czy **Ranlux**) i „złego” (np. randu z biblioteki GSL)”*

Generator “zły”, czyli rand, wykazuje lepsze właściwości dla prób 1000 iteracji.

Sytuacja zmienia się dla iteracji większych od  $10^4$ . Tutaj możemy zaobserwować iż generator “dobry” (Ranlux) wykazuje wynik znacznie mniejszy od wartości krytycznej

## **Bibliografia**

- Wartości krytyczne dla funkcji chi-kwadrat, gdy znamy stopień swobody oraz 5% rozbieżności ze strony: <https://www.medcalc.org/manual/chi-square-table.php>