Национальный исследовательский университет
«Высшая Школа Экономики»
Московский институт электроники и математики им. А. Н. Тихонова

# Лабораторная работа №3

Выполнил:

Дёма Иван Романович, СКБ212

Проверил:

Драчёв Григорий Александрович

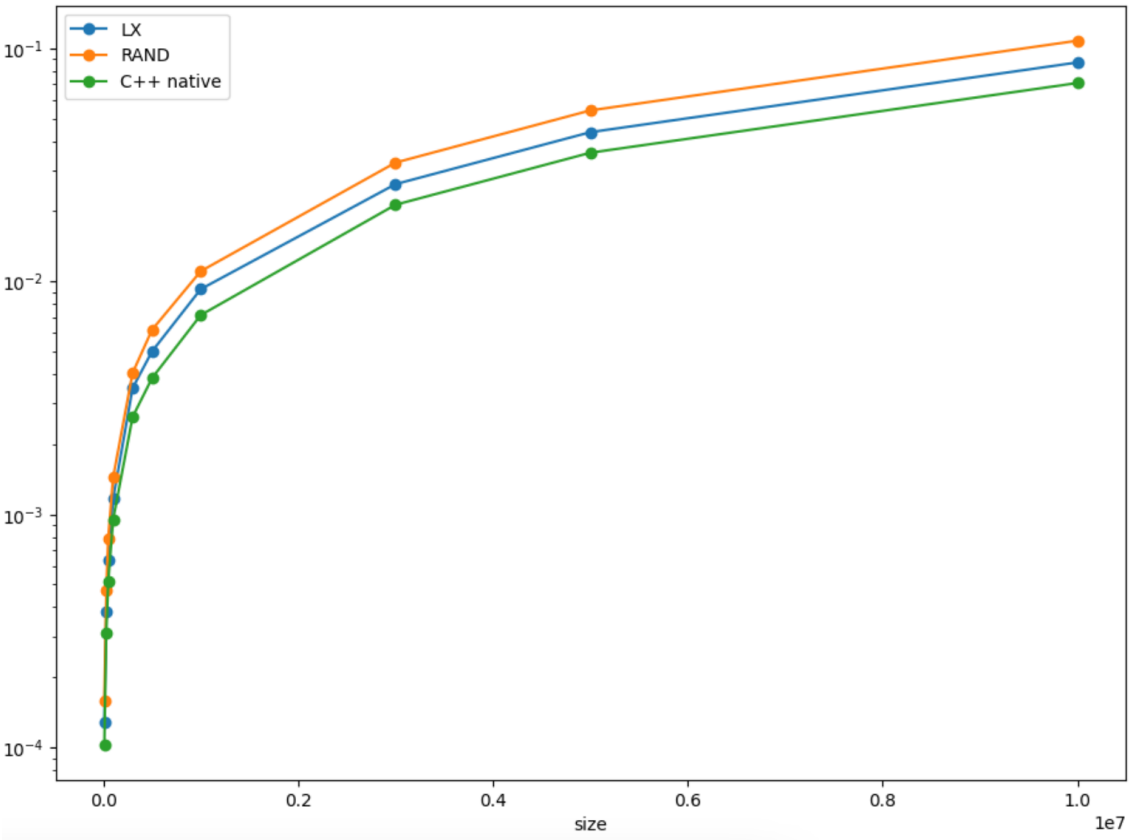
Москва,
2024

## 1. Задание

Модифицировать (предложить собственные) два метода генерации псевдослучайных чисел.

## 2. Задание

Сравнение скорости работы двух алгоритмов со встроенным




## 3. Задание

Репозиторий с исходным кодом:

Ссылка

Документаци

# Алфавитный указатель классов

## Классы

Классы с их кратким описанием.

# Список файлов

## Файлы

Полный список файлов.

# Классы

## Класс LXRandomGenerator

Второй класс генератора случайных чисел
`#include <mygenerators.h>`

### Открытые члены

- **LXRandomGenerator** (const unsigned int seed)
  *Стандартный конструктор класса*

- unsigned int **generate** ()
  *Открытая функция, позволяющая пользователю получить новое число из генератора*

---

### Подробное описание

Второй класс генератора случайных чисел

Класс основан на конкатенации Xor-шифта и лиейно-конгурентного метода

Класс имеет приватное поле:

#### Параметры шаблона

| | |
|---|---|
| *state* | - состояние генератора. Из этого числа расчитывается каждое следующее число |

---

### Конструктор(ы)

#### LXRandomGenerator::LXRandomGenerator (const unsigned int *seed*) `[inline], [explicit]`

Стандартный конструктор класса

Инициализирует поле state, как seed, подаваемый при первоначальной инициализации пользователем

---

### Методы

#### unsigned int LXRandomGenerator::generate () `[inline]`

Открытая функция, позволяющая пользователю получить новое число из генератора

**Возвращает**

число, полученное после конкатенации результатов приватных функций xorshift() и lcg()

---

**Объявления и описания членов класса находятся в файле:**

mygenerators.h

# Класс RandomGenerator

Класс генератора случайных чисел
`#include <mygenerators.h>`

## Открытые члены

- **RandomGenerator** (unsigned int seed)
  *Стандартный конструктор класса*

- unsigned int **generate** ()
  *Открытая функция, позволяющая пользователю получить новое число из генератора*

---

## Подробное описание

Класс генератора случайных чисел

Класс основан на использовании Xor-шифта с модификациями

Класс имеет приватное поле:

### Параметры шаблона

| | |
|---|---|
| *state* | - состояние генератора. Из этого числа расчитывается каждое следующее число |

---

## Конструктор(ы)

### RandomGenerator::RandomGenerator (unsigned int *seed*) `[inline], [explicit]`

Стандартный конструктор класса

Инициализирует поле state, как seed, подаваемый при первоначальной инициализации пользователем

---

## Методы

### unsigned int RandomGenerator::generate () `[inline]`

Открытая функция, позволяющая пользователю получить новое число из генератора

#### Возвращает

число, полученное после выполнения приватной функции xorshift()

---

**Объявления и описания членов класса находятся в файле:**
**mygenerators.h**

# Файлы

## Файл main.cpp

```
#include <ctime>
#include <fstream>
#include <iostream>
#include <vector>
#include "mygenerators.h"
```

### Макросы

- #define **SIZE** 20
  *Устанавливает количество выборок, которые мы будем генерировать*

### Функции

- template<typename S > std::ostream & **operator<<** (std::ostream &os, const std::vector< S > &vector)
  *Перегрузка оператора << для последовательного вывода вектора*

- int **main** ()
  *Основная функция в программе*

---

### Макросы

#### #define SIZE 20

Устанавливает количество выборок, которые мы будем генерировать

---

### Функции

#### int main ()

Основная функция в программе

**Возвращает**

ноль, если программа завершилась успешно

В начале программы создаётся массив sample_sizez, в котором описаны все размеры, которые будут генерироваться.

```
int sample_sizes[SIZE];
```

Проходим два цикла (так как два метода генерации) по всем размерам и генерируем данные по заданным размерам Создание объекта первого генератора

```
LXRandomGenerator generator(n)
```

Создание объекта второго генератора

```
RandomGenerator generator(n)
```

Проходим цикл по всем размерам от 1e3 до 1e7 и замеряем время генерации каждого типа (два собственной реализации и один встроенный) Создание объекта первого генератора

```
LXRandomGenerator generator(n)
```

Создание объекта второго генератора

```
RandomGenerator generator(n)
```

Использование третьего (встроенного) генератора

```
std::rand()
```

**template<typename S > std::ostream & operator<< (std::ostream & *os*, const std::vector< S > & *vector*)**

Перегрузка оператора << для последовательного вывода вектора

**Аргументы**

| | |
|---|---|
| *os* | - адрес буфера, в которой необходимо записывать данные |
| *vector* | - вектор, который необходимо вывести |

**Возвращает**

адрес буфера

# Файл mygenerators.h

## Классы

class **RandomGenerator***Класс генератора случайных чисел*

class **LXRandomGenerator***Второй класс генератора случайных чисел*

## Макросы

- #define **A** 214013
  *Устанавливает параметр A для линейно-конгруентного метода*

- #define **C** 2531011
  *Устанавливает параметр C для линейно-конгруентного метода*

---

## Макросы

### #define A 214013

Устанавливает параметр A для линейно-конгруентного метода

### #define C 2531011

Устанавливает параметр C для линейно-конгруентного метода

## mygenerators.h

См. документацию.

```
1 #ifndef MYGENERATORS_H
2 #define MYGENERATORS_H
3
6 #define A 214013
9 #define C 2531011
10
11
21 class RandomGenerator {
22 private:
23     unsigned int state;
24
29     unsigned int xorshift() {
30         state += C ^ state;
31         state ^= (state << 13);
32         state ^= (state >> 17);
33         state ^= (state << 5);
34         state *= 0x2545F4914F6CDD1D;
35         return state;
36     }
37
38 public:
43     explicit RandomGenerator(unsigned int seed) : state(seed) {}
44
49     unsigned int generate() { return xorshift(); }
50 };
51
52
60 class LXRandomGenerator {
61 private:
62     unsigned int state;
63
68     unsigned int lcg() {
69         state = A * state + C;
70         return state;
71     }
72
77     unsigned int xorshift() {
78         state ^= (state << 13);
79         state ^= (state >> 17);
80         state ^= (state << 5);
81         return state;
82     }
83
84 public:
89     explicit LXRandomGenerator(const unsigned int seed) : state(seed) {}
90
95     unsigned int generate() { return lcg() ^ xorshift(); }
96 };
97
98 #endif // MYGENERATORS_H
```