УДК 004.942

В.В. Химич
І.А. Клєопа

ГЕНЕРАЦІЯ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ РЕКУРЕНТНИХ СПІВВІДНОШЕНЬ І ЇХ СТАТИСТИЧНИЙ АНАЛІЗ

Вінницький національний технічний університет

Анотація

У роботі розглядається задача програмної реалізації та верифікації генераторів псевдовипадкових чисел на базі рекурентних співвідношень. Об'єктом дослідження є лінійний конгруентний метод реалізований засобами мови програмування С++. У ході роботи розроблено програмне забезпечення, що виконує генерацію числових послідовностей із використанням часової мітки для забезпечення унікальності вибірки. Проведено комплексний статистичний аналіз отриманих даних, що включає розрахунок математичного сподівання, дисперсії та середньоквадратичного відхилення. Для перевірки гіпотези про рівномірний закон розподілу застосовано метод частотного аналізу з побудовою гістограми. Результати експерименту підтверджують ефективність застосованого алгоритму та відповідність згенерованих чисел статистичним критеріям якості.

Ключові слова: псевдовипадкові числа, лінійний конгруентний метод, рекурентні співвідношення, рівномірний розподіл, дисперсія, гістограма, алгоритмізація.

Abstract

This paper addresses the problem of software implementation and verification of pseudo-random number generators based on recurrence relations. The object of the study is the Linear Congruential Generator (LCG), implemented using the C++ programming language. During the work, software was developed to generate numerical sequences utilizing a timestamp to ensure sample uniqueness. A comprehensive statistical analysis of the obtained data was conducted, including the calculation of the expected value, variance, and standard deviation. To test the hypothesis of a uniform distribution, a frequency analysis method with histogram construction was applied. The experimental results confirm the effectiveness of the applied algorithm and the compliance of the generated numbers with statistical quality criteria.

Keywords: pseudo-random numbers, Linear Congruential Generator (LCG), recurrence relations, uniform distribution, variance, histogram, algorithmization.

Вступ

Генерація псевдовипадкових чисел є фундаментальним елементом сучасного комп'ютерного моделювання, криптографії та імітаційних експериментів, де вимагається відтворення стохастичних процесів у детермінованому середовищі обчислювальних машин. Дана робота присвячена дослідженню та програмній реалізації алгоритмів генерації на основі лінійних рекурентних співвідношень, які, попри свою математичну простоту, дозволяють отримувати числові послідовності з високими показниками якості. Основна увага в роботі приділяється не лише алгоритмічній реалізації лінійного конгруентного методу мовою С++, але й комплексній верифікації результатів шляхом розрахунку ключових статистичних характеристик та аналізу рівномірності розподілу, що є необхідною умовою для підтвердження надійності та придатності генератора для практичного застосування.

Результати дослідження

1. Підключення бібліотек

```
#include <vector>
#include <numeric>
#include <cmath>
#include <map>
#include <string>
#include <cstdio>
#include <chrono>
#include <Windows.h>
using namespace std;
```

Освновні застсування кожної бібліотеки:

<vector> - Підключає контейнер std::vector. Ти використовуєш його для створення масиву numbers, щоб зберігати 50,000 згенерованих чисел.

<numeric> - Ця бібліотека містить алгоритми для роботи з числовими послідовностями. Тут вона необхідна для функції std::accumulate, яка підраховує суму всіх елементів вектора для обчислення середнього значення (Mean).

<cmath> - Надає стандартні математичні функції мови С. У твоєму коді вона потрібна для функції sqrt() (квадратний корінь), яка використовується при розрахунку *середньоквадратичного відхилення* (Standard Deviation).

<map> - Підключає std::map. Ти використовуєш його для створення гістограми (histogram). Це словник, де ключ — це номер інтервала ("кошика"), а значення — кількість чисел, що в нього потрапили.

<string> - Потрібна для використання класу std::string. У твоєму коді це використовується в кінці для ASCII-графіки: string hashes(num_hashes, '#'); створює рядок із символів # для візуалізації стовпчиків.

<cstdio> - Це бібліотека з мови С (аналог stdio.h). Вона підключається для функції printf. Ти обрав її замість iostream (cout) для зручного форматування виводу (наприклад, %.6f для округлення дробів або %10lu для вирівнювання).

<chrono> - Сучасна бібліотека С++ для точного часу. Ти використовуєш її, щоб отримати поточний час у наносекундах (chrono::high_resolution_clock або system_clock). Це критично важливо для створення унікального seed (зерна), щоб при кожному запуску програма генерувала різні числа.

<Windows.h> - Це специфічна для Windows бібліотека. Вона використовується для функцій SetConsoleCP(1251) та SetConsoleOutputCP(1251), щоб консоль коректно відображала українські літери (кирилицю). Важливо: Цей код не скомпілюється на Linux або macOS без змін.

2. Підготовка інструментів для генерації псевдо випадкового числа

```
class LCG {
public:
    const unsigned long m;
    const unsigned long a;
    const unsigned long c;
private:
    unsigned long Xn;

public:
    LCG(unsigned long seed, unsigned long modulus, unsigned long multiplier, unsigned long increment)
        : m(modulus), a(multiplier), c(increment), Xn(seed) {
        if (Xn == 0) {
            Xn = 1;
        }
    }
    unsigned long next_int() {
        Xn = (static_cast<unsigned long long>(a) * Xn + c) % m;
        return Xn;
    }
    double next_double() {
        return static_cast<double>(next_int()) / m;
    }
};
```

Перш ніж програма почне щось робити, описується "креслення" генератора. Це клас LCG.

Зберігання стану (Xn): У класі є приватна змінна Xn. Це найважливіша частина. Це "пам'ять" генератора. Він повинен пам'ятати останнє згенероване число, щоб на його основі зробити наступне.

Конструктор: Коли створюється об'єкт LCG, йому передається seed (початкове зерно). Важливий нюанс: Код перевіряє if (Xn == 0) Xn = 1;. Це захист. Якщо формула a * X + 0 помножить на 0, результатом буде 0. Генератор буде неправильно працювати і буде видавати тільки нулі.

Функція next_int() : Тут знахадиться формула: (a * Xn + c) % m.

Множення: Беремо старе число, множимо на величезне a. Число стає гігантським. Static_cast дого, щоб це гігантське число помістилося в пам'ять і не переповнилося перед діленням.

Модуль % m: Це операція "залишку від ділення". Вона гарантує, що результат ніколи не перевищить число m.

Метод next_double() (Нормалізація): Генератор видає цілі числа (наприклад, 10543). Але для статистики потрібні числа від 0.0 до 1.0. Тому отримане число ділиться на максимально можливе (m).

3. Функція main

3.1 Ініціалізація параментрів

```
const unsigned long m = 2147483647; // Модуль
const unsigned long a = 16807;      // Множник
const unsigned long c = 0;          // Приріст
```

Тут задаються константи для формули генератора.

Логіка: Це "фундамент" алгоритму. Використання const гарантує, що ці параметри випадково не зміняться під час роботи програми, що критично для коректності математики.

2. Генерація ункального зерна (seed)

```
auto now = chrono::system_clock::now();
auto duration = now.time_since_epoch();
unsigned long seed = chrono::duration_cast<chrono::nanoseconds>(duration).count();
```

Ми беремо поточний час у наносекундах. Оскільки час постійно змінюється, змінна seed буде унікальною при кожному запуску. Це число стає "точкою старту" (X_0) для генератора.

3. Підготовка пам’яті (оптимізація)

```
LCG generator(seed, m, a, c);
vector<double> numbers;
numbers.reserve(n_samples);
```

Створення об'єкта: Ініціалізуємо клас LCG з нашим унікальним зерном.

numbers.reserve(n_samples) - це критична оптимізація. Ми одразу просимо систему виділити цільний блок пам'яті під 50,000 чисел. Без цього вектор би постійно розширювався і копіював дані, що сповільнило б роботу.

4. Цикл генерації даних

```
for (int i = 0; i < n_samples; ++i) {
    unsigned long next_Xn = generator.next_int();           //  Отримання
    double normalized_val = static_cast<double>(next_Xn) / m; //  Нормалізація
    numbers.push_back(normalized_val);                      //  Збереження
 }
```

Тут відбувається наповнення даними:

Оновлення стану: Метод класу обчислює нове ціле число за формулою.

Нормалізація: Ділення величезного цілого числа на модуль m перетворює його на дробове число в діапазоні [0, 1) (наприклад, з 1073741823 робить 0.5).

Запис: Число додається у підготовлений вектор.

5. Аналіз

```
double sum = accumulate(numbers.begin(), numbers.end(), 0.0);
double mean = sum / n_samples;
```

std::accumulate: Використовуємо алгоритм з бібліотеки <numeric>, щоб просумувати всі елементи вектора одним рядком коду (замість написання циклу вручну). Результат ділимо на кількість, отримуючи середнє значення.

6. Аналіз: Дисперсія та Відхилення

```
    double sq_sum = 0.0;
    for (const double& val : numbers) {
        sq_sum += (val - mean) * (val - mean);
    }
    double variance = sq_sum / n_samples;
    double std_dev = sqrt(variance);
```

Тут реалізована формула розрахунку "розкиду" даних:

Проходимо по кожному числу, знаходимо його різницю з середнім (val - mean). Підносимо різницю до квадрату і сумуємо. Корінь квадратний з отриманої дисперсії дає нам стандартне відхилення — зрозумілу величину похибки.

7. Сортування по “кошиках”

```
map<int, int> histogram;
 for (const double& val : numbers) {
    int bin = static_cast<int>(val * n_bins);
```

```
    histogram[bin]++;
 }
```

Це підготовка даних для гістограми:

Математика індексу: Множимо число (напр., 0.45) на кількість кошиків (10), отримуємо 4.5. Відкидаємо дробну частину -> індекс 4. histogram[bin]++: Використовуємо асоціативний масив (map), щоб підрахувати частоту попадання чисел у кожен діапазон.

Висновки

У результаті виконання роботи було досліджено та програмно реалізовано процес генерації псевдовипадкових чисел методом рекурентних співвідношень. Аналіз роботи програми дозволяє зробити наступні узагальнені висновки:

Ефективність рекурентних алгоритмів: Підтверджено, що використання рекурентних формул (зокрема, лінійного конгруентного методу) є ефективним способом отримання числових послідовностей, які імітують випадковість. Доведено, що якість генерації критично залежить від коректного вибору початкових параметрів (модуля, множника) та динамічного "зерна" (seed).

Статистична відповідність розподілу: Розрахунок основних числових характеристик (математичного сподівання, дисперсії та середньоквадратичного відхилення) показав, що згенерована вибірка відповідає теоретичним показникам рівномірного закону розподілу. Це свідчить про відсутність систематичних похибок в алгоритмі.

Рівномірність заповнення інтервалу: Частотний аналіз та побудова гістограми продемонстрували, що згенеровані числа рівномірно заповнюють увесь заданий діапазон значень. Відхилення частот влучення чисел у різні піддіапазони знаходиться в межах допустимої статистичної похибки.

Практична цінність програмної реалізації: Створений програмний продукт на мові C++ продемонстрував високу швидкодію та ефективне використання пам'яті при обробці великих масивів даних. Розроблений інструментарій дозволяє не лише генерувати послідовності, але й оперативно оцінювати їх якість, що є необхідним етапом у задачах комп'ютерного моделювання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Фаур Е.В., Щерба А.І., Рудницький В.М. Метод та критерій оцінювання якості послідовностей випадкових чисел Кібернетика та системний аналіз. 2020 р., Т. 52 №2.
2. Будько М.Б., Будько М.Ю., Гірік А.В., Грозів В.А.Методи генерації та тестування випадкових послідовностей. Університет ІТМО. 2019 р
3. Маліновська, О. О. Вимоги до криптографічної системи захисту інформації / О. О. Маліновська, О. І. Зінченко ; наук. кер. Я. Ю. Усов // Новітні технології у науковій діяльності і навчальному процесі : матеріали тез доп. Всеукр. наук.- практ. конф. студентів, аспірантів та молодих учених (м. Чернігів, 10 -11 квітня 2019 р.). - Чернігів : ЧНТУ, 2019. – С. 113-116.
4. Modern usage of “old” one-time pad/Mariusz Borowski and Marek Lesniewicz
5. Jorgen Veisdal, Shannon Ciphers and Perfect Security/Cantor’s Paradise/2020

Химич Вячеслав Вадимович – студент групи 2 БС – 25б, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: 04-25-002.stud@vntu.edu.ua

Науковий керівник: ***Клєопа Ірина Анатоліївна*** – pHd, доцент, кафедра вищої математики, Вінницький національний технічний університет, м. Вінниця, Хмельницьке шосе, 95, e-mail: paceka08@vntu.edu.ua

Chymych Viacheslav Vadumovuch – student of group 2 SS – 25b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: 04-25-002.stud@vntu.edu.ua

Scientific supervisor: ***Klieopa Iryna Anatoliivna*** – PhD, Associate Professor, Department of Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, Khmelnytske Shosse, 95, e-mail: paceka08@vntu.edu.ua