**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 1

**З дисципліни:** *“Безпека програм та даних”*

**На тему:** *“Створення генератора псведовипадкових чисел”*

**Лектор:**

проф. каф. ПЗ

Сенів М.М.

**Виконав:**

ст. гр. ПЗ-45

Хруставчук М.Л.

**Прийняла:**

асист. каф. ПЗ

РабійчукІ.О.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2025 р.

∑= \_\_\_\_\_ .

Львів – 2025

**Тема роботи:** Створення генератора псевдовипадкових чисел.

**Мета роботи:** Ознайомитись з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел для використання в системах захисту інформації.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

В сучасній інформатиці випадкові числа використовуються в моделюванні (метод Монте-Карло), криптографії, статистиці тощо. Ключовими вимогами до них є випадковість і непередбачуваність. Хоча фізичні джерела випадковості (радіоактивне випромінювання, шум тощо) можуть забезпечити справжню випадковість, на практиці переважає використання псевдовипадкових чисел, які генеруються алгоритмічно.

Найбільш популярний метод – лінійне порівняння (Linear Congruential Generator, LCG):

Параметри:

m – модуль, a – множник, c – приріст, X₀ – початкове значення (seed).

Послідовність має значення в межах: 0 ≤ < m

Критерії якості генератора:

1. Повний період (усі значення в межах m перед повторенням).
2. Поведінка як у випадкової послідовності (проходження статистичних тестів).
3. Ефективність реалізації в 32-бітній арифметиці.

Попри високу якість, LCG є детермінованим. Якщо відомо кілька елементів послідовності, можна обчислити параметри a, c, m, а отже – відтворити всю послідовність. Це є загрозою безпеці в криптографії.

Для підвищення захищеності можна:

* Використовувати системний час для ініціалізації.
* Регулярно змінювати seed.
* Додавати часові зсуви до вихідних значень.

Таким чином, псевдовипадкові генератори є потужним інструментом, але для безпечного застосування важливо правильно їх конфігурувати і ускладнювати передбачуваність.

**ЗАВДАННЯ**

1. Згідно до варіанту, наведеного в таблиці, створити програмну реалізацію генератора псевдовипадкових чисел за алгоритмом лінійного порівняння. Програма повинна генерувати послідовність із заданої при вводі кількості псевдовипадкових чисел, результати повинні як виводитись на екран, так і зберігатись у файл. Перевірити період функції генерації, зробити висновок про адекватність вибору параметрів алгоритму. У звіті навести протокол роботи програми, значення періоду функції генерації та зробити висновок про придатність цього генератора для задач криптографії.
2. Індивідуальний варіант №13: .

**ХІД ВИКОНАННЯ**

Файл RandomNumbers.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_01

{

public class RandomNumber

{

public int a {get; set;}

public int c {get; set;}

public int m {get; set;}

public RandomNumber(int a, int c, int m)

{

if (m <= 0)

{

throw new ArgumentException("m should be > 0");

}

if (a < 0 || a >= m)

{

throw new ArgumentOutOfRangeException(nameof(a), "a should be 0 ≤ a < m ");

}

if (c < 0 || c >= m)

{

throw new ArgumentOutOfRangeException(nameof(c), "c should be 0 ≤ c < m ");

}

this.a = a;

this.c = c;

this.m = m;

}

public List<long> GenerateNumbers(long x0, long N)

{

if (x0 < 0 || x0 >= m)

{

throw new ArgumentOutOfRangeException(nameof(x0), "x0 should be 0 ≤ x0 < m ");

}

if (N <= 0)

{

throw new ArgumentOutOfRangeException(nameof(N), "N should be > 0");

}

long xn = x0;

var res = new List<long>();

for (long i = 0; i < N; i++)

{

xn = LCG(xn);

res.Add(xn);

}

return res;

}

public int GetPeriod(long x0)

{

int period = 0;

var numbers = new List<int>();

long xn = x0;

do

{

//numbers.Add(xn);

xn = LCG(xn);

period++;

} while (x0 != xn);

//Console.WriteLine("\nPeriod numbers:");

//Printer.Print(numbers);

return period;

}

private long LCG(long x0)

{

return (this.a \* x0 + c) % m;

}

}

}

Файл Printer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_01

{

public static class Printer

{

public static void Print(List<long> nums)

{

string text = "{" + String.Join(", ", nums) + "}";

Console.WriteLine(text);

}

}

}

Файл DualWriter.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_01

{

public class DualWriter : TextWriter

{

private readonly TextWriter console;

private readonly StreamWriter file;

public DualWriter(string filePath)

{

console = Console.Out;

file = new StreamWriter(filePath);

file.AutoFlush = false;

}

public override Encoding Encoding => console.Encoding;

public override void Write(char value)

{

console.Write(value);

file.Write(value);

}

public override void WriteLine(string? value)

{

console.WriteLine(value);

file.WriteLine(value);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

file.Dispose();

}

base.Dispose(disposing);

}

}

}

Файл Program.cs

namespace Lab\_01;

public static class Program

{

static void Main(string[] args)

{

using var writer = new DualWriter("output.txt");

Console.SetOut(writer);

int a = (int)Math.Pow(9, 3), c = 233, m = (1 << 22) - 1, x0 = 5;

Console.Write("Enter N: ");

if (!int.TryParse(Console.ReadLine(), out int N))

{

Console.WriteLine("N should be a number");

return;

}

try

{

var random = new RandomNumber(a, c, m);

var nums = random.GenerateNumbers(x0, N);

Printer.Print(nums);

int period = random.GetPeriod(x0);

Console.WriteLine($"Period: {period}");

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

}

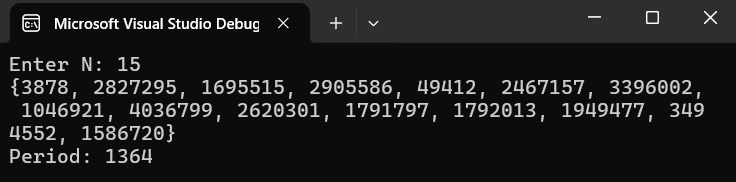


Рис. 1. Результат генерації 15 псевдовипадкових чисел та період функції генерації

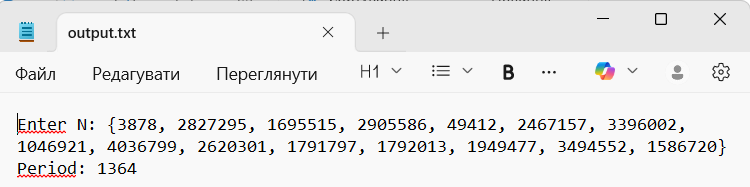


Рис. 2. Вивід згенерованих чисел у файл

Як видно, параметри функції підібрано погано, оскільки її період є надзвичайно малим відносно розміру m. Реалізований алгоритм генерації мало підходить для задач криптографії, оскільки в самому алгоритмі немає нічого випадкового взагалі, крім вибору початкового значення, тому відкривши всього одне значення з послідовності, можна отримати всі наступні. Перевагою алгоритму лінійного порівняння є те, що якщо вибрати адекватні множник та модуль порівняння, то створювана послідовність чисел виявляється статистично невідрізнимою від послідовності чисел, що вибираються випадково.

**ВИСНОВКИ**

На цій лабораторній роботі я ознайомився з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел для використання в системах захисту інформації.