МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

**ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

до лабораторної роботи №1

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

**На тему:** *“СТВОРЕННЯ ГЕНЕРАТОРА ПСЕВДОВИПАДКОВИХ ЧИСЕЛ*”

**Лектор:**

доц. каф. ПЗ

Сенів М.М.

**Виконала:**

ст. гр. ПЗ-42

Крук Д. С.

**Перевірив:**

доц. каф. ПЗ

Сенів М.М.

«\_\_\_» \_\_\_\_\_\_ 2024 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Львів – 2024

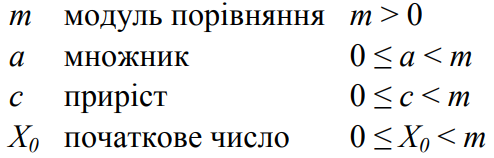
**Тема:** створення генератора псевдовипадкових чисел.

**Мета:** ознайомитись з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел та навчитись створювати програмні генератори псевдовипадкових чисел для використання в системах захисту інформації.

**Теоретичні відомості**

Генератор псевдовипадкових чисел – алгоритм, що генерує послідовність чисел, елементи якої незалежні один від одного і підлягають заданому розподілу.

Найбільш популярним алгоритмом для генерування псевдовипадкових чисел є алгоритм, запропонований Лемером, який називається методом лінійного порівняння. Цей алгоритм має чотири наступних параметри.



Послідовність псевдовипадкових чисел { } отримують за допомогою

ітерацій наступного співвідношення:



Бажано, щоб m було дуже великим, щоб потенційно могли генеруватись

дуже довгі серії різних псевдовипадкових чисел. Загальним правилом тут є вибір значення m, близького до максимально допустимого для даного комп'ютера додатного цілого числа. Тому доволі часто значення m вибирається рівним або майже рівним значенню .

Пропонується три критерії, за якими можна оцінити якість будь-якого

генератора псевдовипадкових чисел.

1. Функція генерації повинна бути функцією повного періоду, тобто функція повинна породити усі числа від 0 до m перед тим, як числа почнуть повторюватись.
2. Створена послідовність повинна вести себе як випадкова. Насправді ця послідовність не буде випадковою, оскільки генерується детермінованим алгоритмом, але існує багато статистичних тестів, які можна використовувати для того, щоб оцінити ступінь випадковості поведінки послідовності.
3. Функція генерації повинна ефективно реалізовуватись в рамках 32-бітної арифметики.

**Завдання до виконання роботи**

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.

**Індивідуальний варіант**

****

**Хід роботи**

1. Ознайомилась із завданням.
2. Створила реалізацію генератора псевдовипадкових чисел за алгоритмом лінійного порівняння.

calculateRandomNumbers(enteredM: string, enteredA: string, c: number, x0: number, amount: number) {

        this.calculateValues(enteredM, enteredA);

        this.randomNumbers = [];

        this.randomNumbers.push(x0);

        this.randomNumbersDisplay = this.randomNumbers.join(', ');

        let Xn = x0;

        let Xn1 = 0;

        let i = 0;

        console.log(`Xn+1 = ${Xn}`);

        for (i = 0; i < amount - 1; i++) {

            Xn1 = (this.a \* Xn + c) % this.m;

            this.randomNumbers.push(Xn1);

            this.randomNumbersDisplay = this.randomNumbers.join(', ');

            Xn = Xn1;

        }

    }

1. Створила функцію запису у файл.

@app.post('/calculate')  
async def calculate\_and\_save(data: CalculateData):  
 m = data.m  
 a = data.a  
 c = data.c  
 x0 = data.x0  
  
 Xn = x0  
 Xn2 = 0  
 iteration = 0  
 file\_path = 'result.txt'  
 buffer\_size = 20000   
 buffer = []   
  
 if os.path.exists(file\_path):  
 os.remove(file\_path)  
  
 with open(file\_path, 'a') as f:  
 buffer.append(str(Xn))   
  
 while True:  
 iteration+=1  
 if(iteration==2): Xn2=Xn1  
 Xn1 = (a \* Xn + c) % m  
 if Xn == Xn1 or Xn1 == x0 or Xn2 == Xn1:  
 break  
  
 buffer.append(str(Xn1))   
  
 if len(buffer) >= buffer\_size:  
 f.write('\n'.join(buffer) + '\n')  
 buffer.clear()   
  
 Xn = Xn1  
  
 if buffer:  
 f.write('\n'.join(buffer) + '\n')  
  
 return {  
 'message': 'Data saved successfully',  
 'status': 'OK'  
 }

1. Створила функцію для знаходження періоду.

@app.post('/get-period')  
async def get\_period(data: CalculateData):  
 m = data.m  
 a = data.a  
 c = data.c  
 x0 = data.x0  
  
 Xn = x0  
 Xn2 = 0  
 iteration = 0  
 period = 1  
  
 while True:  
 iteration += 1  
 if (iteration == 2): Xn2 = Xn1  
 Xn1 = (a \* Xn + c) % m  
  
 if Xn == Xn1 or Xn1 == x0 or Xn2 == Xn1:  
 break  
  
 period += 1  
 Xn = Xn1  
  
 return {  
 'period': period  
 }

1. Перевірила період функції генерації. Він становить 24. Отже, вибір параметрів не є ефективним.

**Протокол роботи програми**

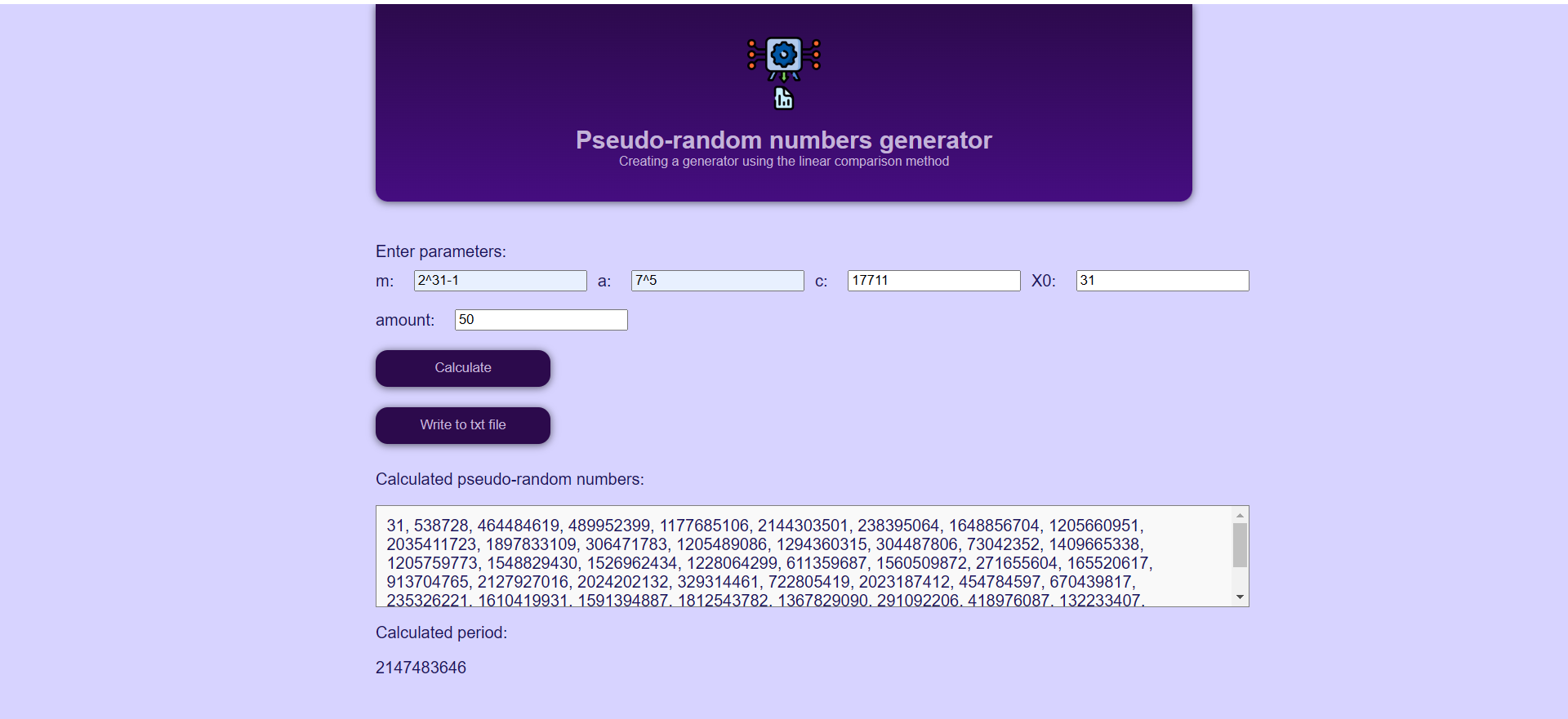
****

Рис.1. Вигляд програми.

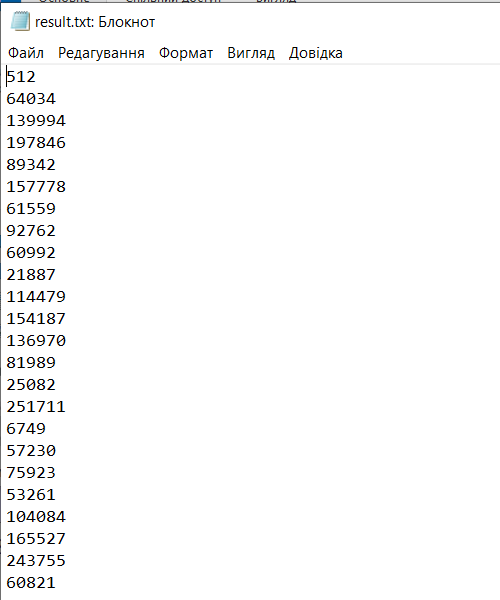
****

Рис.2. Запис у текстовий файл.

**Висновки**

Виконуючи дану лабораторну роботу, було створено генератор псевдовипадкових чисел за допомогою методу лінійного порівняння. Було перевірено період функції генерації та записано отримані дані у файл. Можна зробити висновок, що генератори псевдовипадкових чисел методом лінійного порівняння не є придатними для серйозних криптографічних задач через їхню передбачуваність і низьку випадковість.