Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 1**

**з дисципліни «Безпека програм та даних»**

**«Створення генератора псевдовипадкових чисел»**

**Лектор:**

доц. кафедри ПЗ

Сенів М. М.

**Виконав:**

студ. групи ПІ-41

Павленчик М. М.

**Прийняв:**

доц. кафедри ПЗ

Сенів М. М.

«\_\_\_» \_\_\_\_\_\_ 2018 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**ТЕМА РОБОТИ**: Створення генератора псевдовипадкових чисел.

**МЕТА РОБОТИ**: 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

Сучасна інформатика широко використовує випадкові числа в різних програмах – від методу Монте-Карло до криптографії. Ряд алгоритмів захисту мережі, заснованих на засобах криптографії, передбачає використання випадкових чисел.

Джерелами дійсно випадкових чисел потенційно можуть бути фізичні генератори шумів, такі як імпульсні детектори іонізуючого випромінювання, газорозрядні лампи, конденсатори з втратами струму тощо. Однак такі пристрої можуть знайти доволі обмежене застосування в додатках для захисту інформації.

Тому криптографічні додатки зазвичай використовують алгоритмічні методи генерування випадкових чисел. Відповідні алгоритми є детермінованими і тому породжують послідовності чисел, які не є статистично випадковими. Однак, якщо алгоритм є достатньо хорошим, породжувані ним послідовності чисел витримують багато тестів на випадковість. Такі числа часто називають псевдовипадковими.

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.

Модуль порівняння (*m*): *m* > 0

Множник (*а*): 0 ≤ *a* < *m*

Приріст (*с*): 0 ≤ *с* < *m*

Початкове число (): 0 ≤ < *m*

Послідовність псевдовипадкових чисел {X0} отримують за допомогою ітерацій наступного співвідношення:

**Завдання для лабораторної роботи**

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.

### Результати

Протокол роботи програми

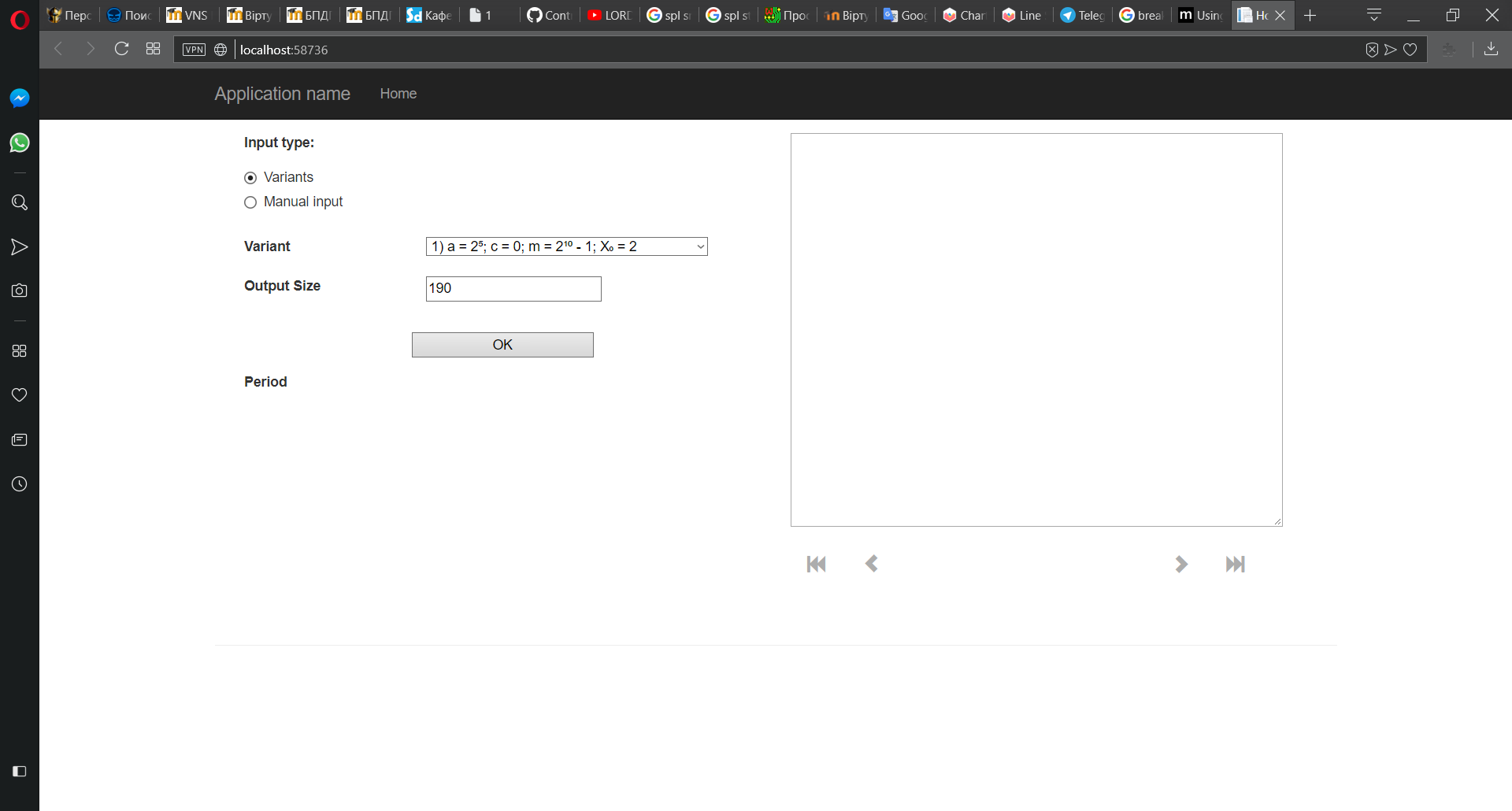


Рис. 1. Початок роботи застосунку

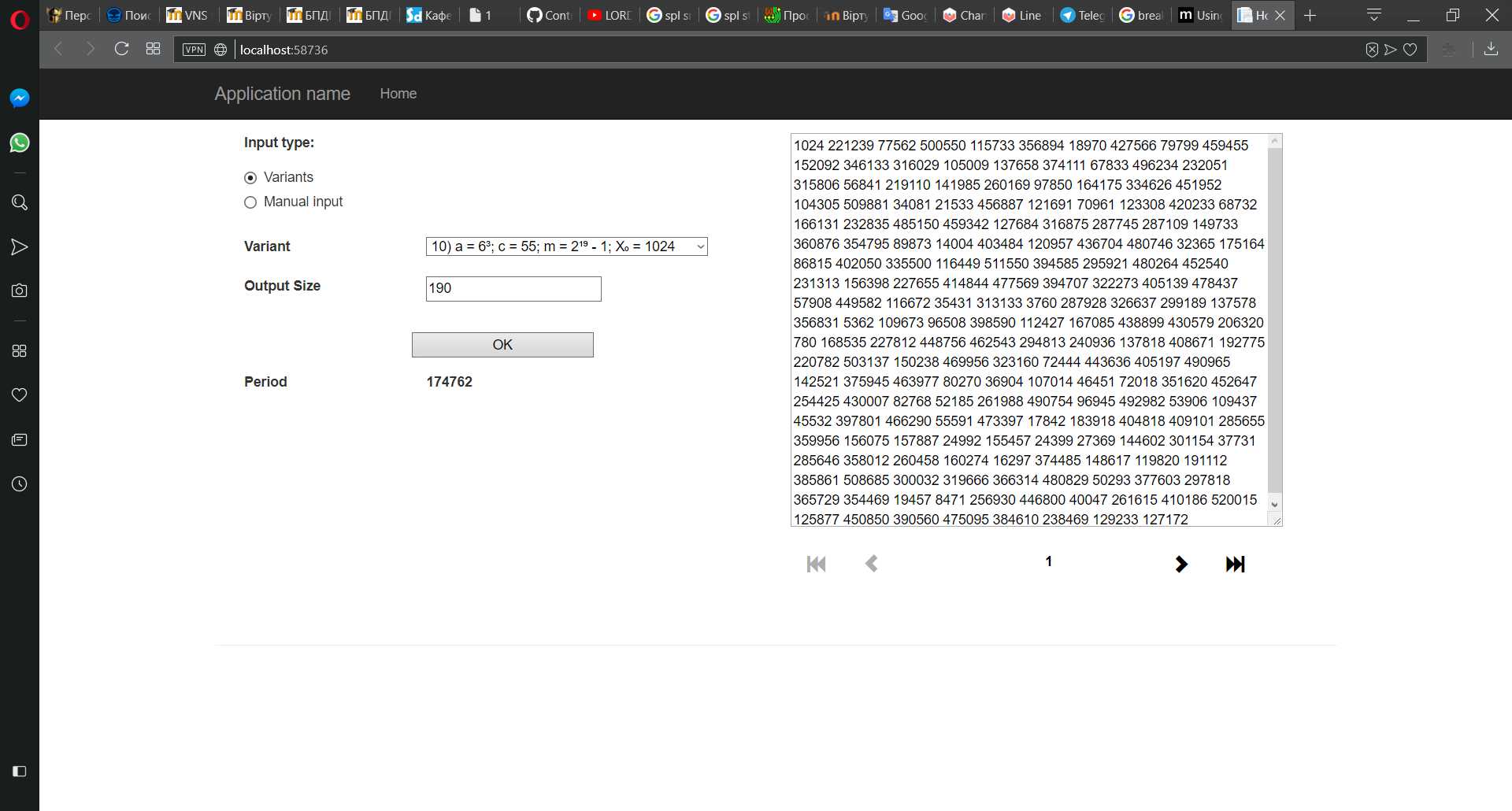


Рис. 2. Обраховано вхідні дані за варіантом №10

Вихідний код програми

internal class RandomNumberGenerator : IRandomNumberGenerator

{

public RandomNumberGenerator(long a, long c, long m, long x0)

{

this.a = a;

this.c = c;

this.m = m;

this.x = x0;

}

private long a { get; set; }

private long c { get; set; }

private long m { get; set; }

private long x { get; set; }

public long GetNextNumber()

{

var result = x;

x = (a \* x + c) % m;

return result;

}

}

public class RandomSequenceGenerator : IRandomSequenceGenerator

{

public const int PartSize = 40000;

public RandomSequenceGenerator(long a, long c, long m, long x0)

{

this.Generator = new RandomNumberGenerator(a, c, m, x0);

this.x0 = x0;

this.Period = 0;

this.IsEnded = false;

this.IsStart = true;

this.FirstElements = new List<long>();

}

private RandomNumberGenerator Generator { get; set; }

private bool IsStart { get; set; }

private List<long> FirstElements { get; set; }

public long x0 { get; protected set; }

public long Period { get; protected set; }

public bool IsEnded { get; protected set; }

public List<long> GetNextSequencePart()

{

if (IsEnded)

{

return null;

}

if (IsStart)

{

FirstElements = new List<long> { x0 };

}

var result = new List<long>();

for (int i = 0; i < PartSize; i++)

{

var nextNumber = Generator.GetNextNumber();

result.Add(nextNumber);

if (IsStart && i < 2)

{

FirstElements.Add(nextNumber);

}

if (nextNumber == x0 && (!IsStart || IsStart && result.Count > 1) || FirstElements.Contains(nextNumber) && (!IsStart || IsStart && i > 1))

{

IsEnded = true;

Period += i;

return result;

}

}

Period += PartSize;

if (IsStart)

{

IsStart = false;

}

return result;

}

}

public class GeneratorService

{

public const string path = @"D:\RandomSequence.txt";

public static GeneratingResultModel GenerateSequence(InputModel input)

{

var generator = new RandomSequenceGenerator(input.A, input.C, input.M, input.X0);

var result = new List<long>();

File.WriteAllText(path, string.Empty);

using (var fileStream = File.OpenWrite(path))

{

using (var writer = new StreamWriter(fileStream))

{

writer.AutoFlush = true;

while (!generator.IsEnded)

{

var sequencePart = generator.GetNextSequencePart();

result.Add(sequencePart[0]);

var fileLine = new StringBuilder();

sequencePart.ForEach(num => fileLine.Append(num).Append(" "));

writer.WriteLine(fileLine.ToString());

}

}

}

return new GeneratingResultModel { TuchPoints = result, Period = generator.Period };

}

}

**Висновки**

На даній лабораторній роботі я ознайомився з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел та навчився створювати програмні генератори псевдовипадкових чисел для використання в системах захисту інформації.

Як показали результати роботи із всіх опрацьованих вхідних даних для задач криптографії найкраще підходять вхідні дані №22. Адже послідовність, що формується за допомогою зазначених вхідних даних має період наближений до повного, а саме 2147483646, що дорівнює , а враховуючи, що модуль порівняння дорівнює , можна зробити висновок, що дана послідовність найкраще підходить для задач криптографії серед всіх представлених.