Randomus Anomalus

Генератор випадкових чисел з апаратним джерелом ентропії

2018 р.

ЗМІСТ

ВСТУП

За останні роки розвиток обчислювальної техніки досяг меж, які здавалися неможливими ще якесь десятиріччя тому. Повсякденне її використання стимулюється розширенням сфери можливих застосувань, а масовість реалізацій призводить до доступності з погляду цінового фактору.

Досить часто в нашому житті виникають ситуації, коли необхідно одержувати випадкові або псевдовипадкові числові послідовності. Найчастіше дана задача виникає при організації різного роду ігрових ситуацій. У таких випадках на допомогу людині приходить набір алгоритмів, результатом роботи яких є послідовність чисел.

Генерування випадкових послідовностей із заданим ймовірнісним законом та перевірка їх адекватності – одні з найважливіших проблем сучасної криптології. Наукова і практична значимість цієї проблеми настільки велика, що їй присвячені окремі монографії в області криптології, організовуються розділи в наукових журналах.

В даний час попит на генератори випадкових послідовностей із заданими ймовірнісними розподілами а також на самі випадкові послідовності настільки зріс, що за кордоном з'явилися науково-виробничі фірми, які займаються виробництвом і продажем великих масивів випадкових чисел. Наприклад, з 1996 р. у світі поширюється компакт-диск «The Marsaglia random number CDROM», який містить 4,8 млрд. «істинно випадкових» біт [15].

Таким чином, тема роботи видається актуальною. Робота присвячена створенню генератора випадкових чисел, що використовує як джерело ентропії шуми звукової карти. Генерація має відбуватися в межах деякого діапазону. Також генератор має створювати таку генерацію, що попередить злам даного набору чисел, тобто прогнозування наступного набору за значенням попереднього.

Об’єкт дослідження: послідовності випадкових чисел.

Предмет дослідження: генерація послідовностей випадкових чисел із використанням апаратного джерела ентропії.

Метою конкурсної роботи є створення бібліотеки для генерації випадкових чисел із використанням аудіоадаптера як джерела ентропії.

Для досягнення мети слід розв'язати такі задачі:

- проаналізувати актуальні підходи до генерації послідовностей випадкових чисел;
- порівняти наявні реалізації генераторів випадкових чисел;
- висунути функціональні вимоги до майбутнього програмного забезпечення;
- спроектувати алгоритми та структури даних;
- обрати інструменти розробки;
- створити програмну реалізацію бібліотеки для генерації випадкових чисел із апаратним джерелом ентропії.
- здійснити тестування та створити демонстраційну програму, що ілюструє можливості бібліотеки.

Практичне значення роботи полягає в тому, що її результати можуть бути використані в галузях криптографії, комп'ютерного моделювання та інших сферах, що потребують послідовності випадкових чисел високої якості.

Матеріали конкурсної роботи висвітлені в доповіді «Генератор випадкових чисел з апартним джерелом ентропії» на X Всеукраїнській науково-методичній конференції «Комп’ютерне моделювання та інформаційні технології в освіті» [18].

РОЗДІЛ 1.
ГЕНЕРАТОРИ ВИПАДКОВИХ ТА ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

1.1. Принцип роботи генераторів випадкових чисел

Генератор випадкових чисел – алгоритм, що генерує послідовність чисел, елементи якої майже незалежні один від одного і підпорядковуються заданого розподілу (зазвичай рівномірному).

Найбільш розповсюдженими на практиці є програмні генератори, які дають змогу отримувати послідовності випадкових чисел за рекурентними формулами. Якщо бути абсолютно точним, то числа, які виробляють програмні генератори, насправді є псевдовипадковими. Так їх називають тому, що алгоритми їх отримання завжди є детермінованими [7].

Загалом же програмні генератори повинні задовольняти таким вимогам:

- генерувати статистично незалежні випадкові числа, рівномірно розподілені в інтервалі [0, 1];
- мати можливість відтворювати задані послідовності випадкових чисел;
- витрати ресурсів процесора на роботу генератора повинні бути мінімальними;
- легко створювати незалежні послідовності випадкових чисел (потоки).

Якість роботи генераторів визначається статистичними властивостями послідовностей випадкових чисел, які він генерує, – незалежністю і випадковістю. Властивості послідовностей перевіряються за статистичними критеріями [14].

Здатність відтворювання послідовності випадкових чисел полягає в тому, що за однакових початкових умов і параметрів генератор повинен відтворювати одні й ті ж послідовності псевдовипадкових чисел [5]. Ідентичні послідовності випадкових чисел рекомендується використовувати у випадку, коли потрібно порівняти альтернативні варіанти систем, що моделюються, і налагодити програми.

Під час дослідження складних систем виникає необхідність у моделюванні послідовностей випадкових чисел великої довжини. Для їх створення потрібні високопродуктивні алгоритми генерування з мінімальними вимогами до ресурсів комп'ютера. Для моделювання різних випадкових факторів бажано мати окремі послідовності, які відтворювались би одним і тим же генератором, але за різних значень параметрів [8].

У більшості генераторів псевдовипадкових чисел $x_г$ використовується рекурентна процедура $x_{i+1} = f(x_i)$. Найпростішим та найдавнішим серед таких генераторів є генератор фон Неймана та Метрополіса, робота якого базувалась на методі середин квадратів [13].

1.2. Порівняльний аналіз генераторів випадкових чисел

Наведемо найбільш прості та відомі з методів генерації псевдовипадкових послідовностей:

1) Лінійний конгруентний метод

Запропонований Д. Х. Лемером [10]. Основна обчислювальна формула:

$$x_{n+1} = (ax_n + c)\ mod\ m$$

Алгоритм зациклюється з періодом, що не перевищує деякого m. Коефіцієнти a, m і x(0) можуть приймати довільні цілі значення, за винятком 0. Параметр c може бути також і 0, але в цьому випадку скорочується період.

2) Алгоритм «Mother-of-All»

Запропонований Джорджем Марсалією (Marsaglia), професором університету Флориди [11]. Обчислювальна формула:

$$\begin{cases} S = 21111111111x_{n-4} + 1429x_{n-3} + 1776x_{n-2} + 5115x_{n-1} + C \\ x_n = \dfrac{S}{2^{32}} \\ C = \left[\dfrac{S}{2^{32}}\right] \end{cases}$$

Цей алгоритм є узагальненням попереднього і позбавлений його головного недоліку – короткого періоду. Його період – 2^{250}.

3) Вихор Мерсена

Вихор Мерсенна [17] ґрунтується на властивості простих чисел Мерсенна і забезпечує швидку генерацію високоякісних псевдовипадкових чисел. Існують щонайменше два загальних варіанти алгоритму, що розрізняються тільки розміром використовуваного простого числа Мерсенна, найбільш поширеним з яких є MT19937 [19].

Алгоритм дуже швидкий через відсутність множень, але не має достатньої випадковості. Тому галузь застосування алгоритму дещо обмежена [16].

4) Генератори типу «Xorshift»

Одні з найновіших генераторів від Джорджа Марсалії [15]. Знову розглядається деяка початкова послідовність, до якої застосовуються операції xor та побітові зсуви. Ці операції полягають в наступному:

$$x_n = x_{n-1}\ xor\ (x_{n-1}\ shl\ a)$$

Замість shl можна використовувати також shr, та еквівалентне множення.

5) Лінійна змішана форма [9]

Основна обчислювальна формула:

$$x_i = \left(a_0 + \sum_{j=1}^{l} a_j x_{i-j} \right)$$

1.3. Атаки на генератори псевдовипадкових чисел

Атака на генератор псевдовипадкових чисел – атака, спрямована на розкриття параметрів генератора з метою подальшого передбачення псевдовипадкових чисел [21].

Успішна атака може розкрити багато криптографічних систем незалежно від того, наскільки ретельно вони були спроектовані. Проте деякі системи використовують недосконало спроектовані ГПВЧ або роблять це так, що зменшується складність атак. Більше того, потрібно лише одне єдине успішне проникнення, щоб скомпрометувати усю систему.

Якщо зловмисник здатний безпосередньо відстежувати вихідні дані генератора псевдовипадкових чисел і дослідити закономірність їх появи, то це пряма криптоаналітична атака. Цей вид атаки поширюється на більшість алгоритмів, що використовують ГПВЧ.

Атака з частковим попереднім обчисленням може застосовуватись до будь-якого генератора, який використовує лічильник. Припустимо, що вхідні дані не оброблені, і зловмисник може спостерігати послідовні вихідні дані. У наступному кроці зловмисник повинен обчислити вихідні дані кожного t-го значення лічильника і запам'ятати їх у списку. Одне з t спостережених значень повинно увійти до списку. Після запису вхідних даних у списку, можна знайти внутрішній стан генератора. Усі подальші вихідні дані залишаються відомими до того часу, поки не будуть оброблені нові вхідні дані [4].

Часова атака використовує той факт, що інкрементація лічильника вимагає різної кількості часу, залежно від передбаченої кількості байтових додавань. Якщо зловмисник може виміряти час, необхідний для інкрементації лічильника, він може зробити висновки по числу нулів в поточному стані лічильника [2].

Клас атак на основі вхідних даних можливий у разі, коли зловмисник може використовувати інформацію про вхідні сигнали ГПВЧ або контролювати її. Атаки, що ґрунтуються на вхідних даних, можуть бути розділені на атаки з відомими вхідними даними, атаки з відтворюваними вхідними даними і атаки з вибраними вхідними даними [20].

Найпоширенішими атаками є кореляційні атаки [23] через специфіку побудови потокових шифрів. Ці атаки використовують кореляцію вихідних послідовностей схеми шифрування з вихідною послідовністю регістрів для відновлення початкового заповнення останніх.

Існує багато теоретичних матеріалів щодо генераторів випадкових чисел та їх злому, але існують приклади і реальних зломів. Серія зломів відбувалася у казино Центральної та Східної Європи, про що була написана стаття

«Russians Engineer a Brilliant Slot Machine Cheat – And Casinos Have No Fix» [12].

На початку червня 2014 року бухгалтер з казино Lumiere Place в Сент-Луїсі зауважив, що деякі гральні автомати за декілька днів немов зійшли з розуму. Другого і третього червня деякі автомати в Lumiere Place почали видавати набагато більше грошей, ніж отримували, хоча ніяких великих джекпотів не було.

Служба безпеки казино переглянула відеозаписи і через деякий час вирахувала зловмисника. Він просто натискав кнопки, граючи в «Зоряного бродягу» і «Пелікана Піта», а сам в цей час крадькома прикладав свій айфон до екрану автомата. Погравши кілька хвилин, він йшов, а потім повертався, щоб спробувати ще раз. На 20-60 витрачених доларів він отримував 1 300, після чого закінчував гру і переходив до іншого автомату, де все починав усе спочатку. Привертало увагу те, що він довго тримав палець над кнопкою «Старт», а потім натискав її різко і поспішно.

9 червня казино поділилося своїми спостереженнями з комісією з азартних ігор штату Міссурі, яка оголосила тривогу по всьому штату. Деякі казино незабаром виявили, що вони також стали жертвами подібних махінацій. Алгоритм дій був той самий: зловмисник прикладав стільниковий телефон до автомата моделі Aristocrat Mark VI, а потім натискав кнопку.

Афера в Lumiere Place показала, як хакери вирішили цю проблему. Почувши про те, що сталося в Міссурі, експерт з безпеки казино Даррін Хоук (Darrin Hoke), який працював на той час директором служби спостереження в казино L'Auberge du Lac в Луїзіані, вирішив самостійно розслідувати розмах цієї хакерської операції. Опитавши колег, які повідомили йому про підозрілі моменти в роботі ігрових автоматів, і вивчивши записи відеоспостереження, він зумів виявити 25 шахраїв, які працювали в казино від Каліфорнії до Румунії і Макао.

Вилучені в казино телефони, а також дані розслідувань в Міссурі і Європі дозволили з'ясувати кілька важливих деталей. Зловмисники за допомогою

телефонів записували близько двох десятків оборотів в грі. Передавали ці дані технічного персоналу в Санкт-Петербург, де відеозапис аналізувався, а фахівці вираховували закономірності в роботі автомата на підставі того, що їм було відомо про ГПВЧ даної моделі. В результаті команда з Санкт-Петербурга передавала на додаток в телефоні гравця тимчасові маркери. Ці маркери змушували телефон вібрувати протягом 0,25 секунди, після чого гравець повинен був різко натиснути на кнопку.

«Звичайний час реакції людини близько чверті секунди, через що вони надходили саме так», – говорить Еллісон, який є засновником щорічної міжнародної конференції щодо захисту ігор. Натиснути кнопку вчасно виходить не завжди, але в результаті зловмисник все одно отримує набагато більше звичайного. Еллісон зазначає, що ці гравці намагаються обмежити свій виграш на кожному автоматі сумою менше тисячі доларів, щоб не викликати підозр.

Технічно зробити автомати невразливими для злому дуже важко. Як зазначає Хоук, Aristocrat, Novomatic і інші виробники ігрових автоматів зі зламаними ГПВЧ «будуть змушені вивести з експлуатації всі свої машини, поставивши замість них щось інше». Але вони явно не мають наміру це робити. У своїй заяві для WIRED Aristocrat підкреслила, що вона не в змозі «виявити дефекти у зламаних іграх», і що її автомати «виготовлені відповідно до найсуворіших технічних стандартів».

Так що все навантаження ляже на співробітників служби безпеки казино, які будуть змушені уважно стежити за тим, що відбувається, виявляючи найменші ознаки шахрайства.

1.4. Інтерфейс звукового адаптеру. Аудіовхід

Аудіоадаптер (Sound Blaster або звукова плата) – це спеціальна електронна плата, яка надає можливість записувати звук, відтворювати його з застосуванням апаратних та програмних засобів ПК. Аудіоадаптер містить у собі два або декілька перетворювачів інформації:

– аналого-цифровий, який перетворює безперервні звукові сигнали у цифровий двійковий код і записує його на магнітний носій;
– цифро-аналоговий, що виконує зворотне перетворення збереженого в цифровому вигляді звуку в аналоговий сигнал, який потім відтворюється за допомогою акустичної системи, синтезатора звуку або навушників.

Професійні звукові плати дозволяють виконувати складну обробку звуку, забезпечують стереозвук, мають власне ПЗП, де зберігаються масиви тембрів звучань різних музичних інструментів. Звукові файли зазвичай мають дуже великі розміри. Так, трихвилинний звуковий файл зі стереозвуком займає приблизно 30 Мбайт пам'яті. Область застосування звукових плат – комп'ютерні ігри, навчальні програмні системи, рекламні презентації, «голосова пошта» між комп'ютерами, озвучування різних процесів, що відбуваються в комп'ютерному обладнанні. Але головна, і часто використовувана можливість сучасної звукової карти – це здатність відтворювати аудіо-файли, що зберігаються на комп'ютері.

Незалежно від місця розташування, звукові пристрої мають роз'єми для підключення мікрофону й акустичних систем, а також можуть оснащуватися і роз'ємами для підключення MIDI-пристроїв (старі моделі також були обладнані ігровим портом). З програмного погляду звукові адаптери вимагають підтримки драйверів, що або містяться в конкретних програмах, або вбудовуються в операційну систему [1].

Типова звукова карта включає звукову мікросхему, що містить цифро-аналоговий перетворювач, який конвертує записаний або згенерований цифровий звук в аналоговий формат. Вихідний сигнал подається на підсилювач, навушники або зовнішній пристрій, використовуючи стандартні роз'єми. Більш сучасні звукові карти містять декілька мікросхем для досягнення вищої якості або поліпшення виконання різних операцій одночасно, наприклад для запису музики в реальному часі важливо, щоб синтез звуків відбувався з мінімальною затримкою процесора.

Відтворення звуку звичайно здійснюється за допомогою багатоканальних ЦАП, що підтримують одночасне відтворення звуків різної висоти й гучності, а також звукові ефекти в реальному часі. Багатоканальне відтворення звуку також використовується для синтезу звуку за допомогою цифрових банків інструментів, що займає невелику кількість постійної або флеш-пам'яті і містить звукові семпли MIDI-інструментів. Інший шлях синтезу звуків полягає у використанні «аудіо-кодеків», цей шлях вимагає відповідного програмного забезпечення, сумісності з MIDI, та багатоканальної емуляції.

Більшість звукових карт мають роз'єми для вхідних та вихідних сигналів. Нерідко звукові карти оснащуються двома вхідними роз'ємами. Один з них, line-in, призначений для підключення пристроїв високого рівня сигналу, таких як, наприклад, магнітофон. Цифрова карта оцифровує цей сигнал і зберігає на жорсткому диску комп'ютера (пізніше збережений сигнал можна обробляти). Інший вхідний роз'єм, microphone, призначений для підключення мікрофону або подібного пристрою низького рівня сигналу. Професійні звукові плати оснащуються кількома вхідними роз'ємами, що дозволяє здійснювати багатоканальний запис звуку.

Окрім того, звукові карти містять так званий «ігровий порт», початково призначений для підключення джойстиків, проте пізніше він знайшов своє призначення для підключення MIDI-клавіатур, цифровий вихід S/PDIF та інші роз'єми. При цьому кожне з гнізд роз'ємів маркують певним кольором згідно додатку А.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ГЕНЕРАТОРА ВИПАДКОВИХ ЧИСЕЛ

2.1. Постановка задачі

Розробити генератор випадкових чисел, що приймає за джерело ентропії шуми звукової карти. Спроектувати алгоритми та структури даних, що використовуватимуться у програмній реалізації бібліотеки. Обрати інструменти розробки, що є зручними для роботи зі звуковою картою та покажуть можливості бібліотеки із її подальшим застосуванням. Передбачити можливість генерації чисел у заданому користувачем діапазоні. Перевірити гіпотезу щодо розподілу отриманої послідовності випадкових чисел.

2.2. Алгоритм роботи бібліотеки

Прикладна програма викликає функцію з підключеної бібліотеки та передає до функції необхідний перелік аргументів, які задає користувач в залежності від його потреб. В якості аргументів передається кількість елементів, мінімальний та максимальний елементи (бажаний діапазон, який використовуватиме користувач).

Після отримання аргументів програма потрапляє у бібліотеку, в якій відбувається запис звуку, що використовуватиметься для генерації послідовності випадкових чисел. Запис звуку відбувається у допоміжному віконному додатку, з якого бібліотека отримує масив із записаним звуком. Наступним кроком відбувається перезапис масиву з буферу пам’яті в створений динамічний масив. При перезаписі відбувається перевірка на повторюваність елементів буферу звуку. Дана перевірка пов’язана з особливостями wave-файлів та, власне, записаного звуку.

Після виконання операції з отримання звуку у бібліотеці відкривається файл із записаним звуком та створюється масив відповідного розміру виходячи із бажань користувача. Для того, щоб у кінцевому результаті програма видала масив чисел із указаного діапазону, необхідно виконати

пошук мінімального та максимального елементів масиву. За допомогою масиву звуку та знайдених елементів за розробленою формою в масиві елементи нормуються до вказаного діапазону.

Нормований масив передається користувачу до основної програми, де користувач користується отриманою послідовністю випадкових чисел відповідно до власних потреб.

Алгоритм роботи бібліотеки передано у загальній блок-схемі бібліотеки (Додаток Б.1) та, власне, блок-схемі алгоритму (Додаток Б.2), що нормує шум, записаний користувачем, до необхідного діапазону.

2.3. Обґрунтування вибору інструментів розробки

Для написання бібліотеки використовувалась середовище розробки Visual Studio 2017. Загалом, Visual Studio є набором продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в машинному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight [24].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудоване налагодження програми може працювати як налагодження рівня вихідного коду, так і налагодження машинного рівня. Решта вбудованих інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Так як мовою написання бібліотеки є C++, то вибір даного середовища розробки є очевидним. Microsoft Visual C++ (MSVC) [22] – інтегроване

середовище розробки додатків на мові C++, розроблена корпорацією Microsoft і поставляється або як частина комплекту Microsoft Visual Studio, або окремо у вигляді безкоштовного функціонально обмеженого комплекту Visual C++ Express Edition.

Однією із найважливіших складових бібліотеки є запис звуку, що реалізовується засобами Windows API [3], під якими розуміють цілий набір базових функцій інтерфейсів програмування додатків операційних систем сімейств Microsoft Windows корпорації «Майкрософт». Windows API є найкращим способом взаємодії додатків з Windows. Для створення програм, що використовують Windows API, «Майкрософт» випускає комплект розробника програмного забезпечення, який називається Platform SDK, і містить документацію, набір бібліотек, утиліт та інших інструментальних засобів для розробки.

Windows API спроектований для використання в мові Сі для написання прикладних програм, призначених для роботи під управлінням операційної системи MS Windows. Робота через Windows API – це найбільш близький до операційної системи спосіб взаємодії з нею через прикладні програми. Більш низький рівень доступу, необхідний тільки для драйверів пристроїв, в поточних версіях Windows надається через Windows Driver Model. Так як бібліотека викликатиме додаток, що звертається до драйвера звукової карти, то використання Windows API є зрозумілим. Варто також зауважити, що при перенесенні бібліотеки на різні комп’ютери її не потрібно переналаштовувати відповідно до різновиду звукової карти, оскільки все це передбачено в базових функціях Windows API. Якщо було б обрано більш високорівневі функції та не застосовувалось обрані, постало б питання у написанні бібліотеки відповідно для кожного типу звукових карт, оскільки вони містять у собі різні налаштування та виклики, а це покликало за собою дуже великий об’єм роботи, що вплинуло би на загальний час створення бібліотеки та її якість.

2.4. Етапи створення бібліотеки

Для реалізації та демонстрування бібліотеки генерації випадкових чисел на основі шумів звукової карти засобами Visual Studio 2017 необхідно виконати такі етапи:

1) Створення статичного проекту бібліотеки.
2) Додавання класу в статичну бібліотеку.
3) Створення консольного додатку, який посилається на статичну бібліотеку.
4) Використання функції статичної бібліотеки в додатку.

Розглянемо створення статичного проекту бібліотеки. Для цього необхідно:

1) У рядку меню обрати «Файл», «Создать», «Проект».
2) У лівій області діалогового вікна «Создать проект» розгорнути «Установленные, Шаблоны, Visual C++» і потім обрати «Win32».
3) У центральній області обрати «Консольное приложение Win32».
4) Задати ім’я створеного проекту (створювана бібліотека матиме ім’я «Library»)та ім'я рішення («Library_sr»).
5) На сторінці «Параметры приложения» у розділі «Тип приложения» необхідно обрати статичну бібліотеку.

Наступним кроком необхідно додати клас до створеної бібліотеки. Для цього:

1) Створюємо файл заголовка для нового класу. Відкриваємо контекстне меню проекту в «обозревателе решений», а потім обираємо «Добавить, Новый элемент». У діалоговому вікні «Добавление нового элемента» обираємо «Заголовочный файл (.h)». Вказуємо ім’ям файлу заголовку «srgenerate.h».
2) Додаємо клас MyGenerate, в якому відбувається оголошення функції для генерації випадкових чисел, яка в майбутньому буде викликатись із додатку, що використовує бібліотеку.
3) Оголошуємо функцію, задаючи необхідний тип та оголошуючи параметри, які передаватимуться їй при виклику.

4) Створюємо вихідний файл для нового класу.

5) Використовуємо створений файл для реалізації функції генерації випадкових чисел.

6) Компілюємо статичну бібліотеку.

Для того, щоб продемонструвати роботу бібліотеки, необхідно створити консольний додаток, який посилається на статичну бібліотеку. Для цього необхідно:

1) У рядку меню обрати «Файл», «Создать», «Проект».

2) У лівій області діалогового вікна «Создать проект» розгорнути «Установленные, Шаблоны, Visual C++» і потім обрати «Win32».

3) У центральній області обрати «Консольное приложение Win32».

4) Вказуємо ім’я додатку «Example». У списку поруч з полем «Решение» обрати «Добавить в решение». В результаті новий проект буде додано до рішення, що містить статичну бібліотеку.

5) На сторінці «Параметры приложения» у розділі «Тип приложения» необхідно обрати консольний додаток.

Використання функціональності зі статичної бібліотеки в додатку:

1) Для використання процедури генерації випадкових чисел із статичної бібліотеки необхідно послатися на цю бібліотеку. Для цього необхідно відкрити контекстне меню проекту в «обозревателе решений», а потім обрати пункт «Ссылки». У діалоговому вікні «Страницы свойств» розгорнути вузол «Общие свойства», обрати «.NET Framework и ссылки» та натиснути кнопку «Добавить новую ссылку»

2) У діалоговому вікні «Добавление ссылки» перераховані бібліотеки, на які можна вставити посилання. Втановлюємо прапорець на «Library_sr».

3) Для створення посилання на файл заголовок «srgenerate.h» необхідно змінити шлях до каталогів підключених файлів. =

4) Тепер клас MyGenerate можна використовувати у додатку відповідно до встановленого синтаксису та заданих параметрів.

РОЗДІЛ 3.
ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕРАТОРА ВИПАДКОВИХ ЧИСЕЛ ІЗ АПАРАТНИМ ДЖЕРЕЛОМ ЕНТРОПІЇ

3.1. Опис основних функцій та структур даних

Розроблений додаток, що відповідає за запис звуку, що використовуватиметься для майбутньої генерованої послідовності розроблений відповідно засобами Windows API, з причин, описаних у другому розділі конкурсної роботи.

В даному додатку використовується структура даних:

```
    BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM
wParam, LPARAM lParam)
    {
       static BOOL  bRecording, bEnding;
       static DWORD dwDataLength, dwRepetitions = 1;
       static HWAVEIN hWaveIn;
       static HWAVEOUT hWaveOut;
       static PBYTE  pBuffer1, pBuffer2, pSaveBuffer, pNewBuffer;
       static PWAVEHDR pWaveHdr1, pWaveHdr2;
       static TCHAR szOpenError[] = TEXT ("Помилка при відкритті
звукового файлу");
         static TCHAR szMemError [] = TEXT ("Помилка при
виділенні пам’яті");
         static WAVEFORMATEX waveform;
```

Через те, що необхідності у показі вікна у користувача немає (оскільки уся програма побудована на системі повідомлень, то повністю відмовитись від створення вікна не є можливим), його необхідно сховати за допомогою функції: `ShowWindow (hwnd,  SW_MINIMIZE);`, де `SW_MINIMIZE` відповідає за видимість вікна.

Наступним кроком необхідно відкрити пристрій введення функцією waveInOpen, в параметрах якій вказуємо формат звукового потоку і спосіб повідомлення про виконання викликаних операцій.

Запис потоку запускається функцією waveInStart. Після цього драйвер запустить аналого-цифровий перетворювач і почне заповнення переданих буферів.

Якщо буфер заповнений, програма може обробити дані. Для того, щоб відстежити момент заповнення буфера, оброблюється повідомлення MM_WIM_DATA, що посилається драйвером в момент заповнення буфера вікна, при цьому відбувається накопичування звукової доріжки:

```
pNewBuffer = (PBYTE) realloc (pSaveBuffer, dwDataLength + ((PWAVEHDR)lParam) -> dwBytesRecorded);
```

Далі необхідно знову передати буфер функцією waveInAddBuffer драйверу для запису. Рух потоку припиняється одразу після того, як буфер заповнюється відповідно до вказаної максимальної кількості елементів.

По закінченню запису звуку та, відповідно, перед закриттям додатку, необхідно вивести отриманий масив в файл, попередньо обробити його в міру того, що додаток працює з wave-файлом.

Створюється допоміжний масив довжини, що відповідає довжині буферу із записаним звуком. В нього записуються тільки ті елементи, що не повторюються, тобто, елемент, що встає в чергу для запису, не дорівнює елементу, що йде перед ним. В цьому випадку буде виключена ситуація занадто великої повторюваності елементів у вихідному масиві. За це відповідає цикл:

```
while (j < ARRAY_LEN)
{
     if (pSaveBuffer[i] != pSaveBuffer[i - 1])
     { mas[j] = pSaveBuffer[i]; j++; }
     i++;
}
```

В якому pSaveBuffer – буфер із записаними елементами звуку, а mas – новий масив, в який потрапляють тільки відібрані елементи. Цикл закінчується тоді, коли весь буфер переглянуто та елементів не залишилось.

Отримані елементи записуються в файл для його подальшого використання бібліотекою генерації випадкових чисел:

```
fwrite(mas, sizeof(char), ARRAY_LEN, f);
```

По завершенню перенесення масиву до файлу, звільнюємо масив та повертаємося до основної програми.

Оскільки більше ніякої інформації з додатку отримано не буде та залишати його ввімкненим не потрібно, то завершуємо його роботу, посилаючи повідомлення:

```
SendMessage(hwnd, WM_SYSCOMMAND, SC_CLOSE, 0L);
```

при виклику якого відбувається звільнення буферів та заголовків, викликається функція, що коректно завершує роботу додатку.

Далі розглянемо, власне, бібліотеку та її складові. Відповідно до етапів створення бібліотеки, що описані у пункті 2.4, розглянемо створення заголовку для класу, що використовуватиме бібліотека:

```
namespace Generate
{
  class MyGenerate
  {
    public:
     static unsigned char* generateNumbers(int len, int a, int b);
  };
}
```

де відбувається оголошення майбутньої функції для її майбутнього наповнення та використання. Функція є статичною та має тип unsigned char* позаяк повертатиме до основної програми масив із генерованою послідовністю елементів. До функції передаватимуться такі аргументи, як:

– `len` – кількість елементів (розмір масиву);

– `a` – початок інтервалу (мінімальний елемент);

– `b` – кінець інтервалу (максимальний елемент).

Далі було створено вихідний файл для нового класу. Першою та основною функцією в ньому є запуск додатку, що записує звук, адже тільки

після цього створиться файл, який буде використовуватись для нормування елементів. Для цього викликаємо функцію:

```
WinExec(«Record.exe», SW_SHOW);
```

Після закінчення роботи програми запису звуку у бібліотеці розпочинається робота з нормування елементів, тобто приведення зчитаних з файлу елементів до необхідного діапазону та повернення вже нормованих елементів користувачу.

Перш за все, необхідно виконати пошук максимального та мінімального елементів із зчитаної послідовності, адже вони необхідні для формули, що приводитиме послідовність до бажаної:

```
for (int j = 0; j <len; j++)
{
     if (mas[j] > max)
     { max = mas[j]; }
}
for (int j = 0; j < len; j++)
{
     if (mas[j] < min)
     { min = mas[j]; }
}
```

Далі відбувається нормування елементів за формулою та перезапис нормованих елементів до нового масиву:

```
for (int j = 0; j < len; j++)
 { mas2[j] = ((mas[j] - min)*(b - a) / (max - min)) + a; }
```

Після закінчення перезапису закриваємо файл з якого зчитувались елементи та передаємо отриманий масив до програми, що зверталась до бібліотеки генерації випадкових чисел.

3.2. Дослідження характеру розподілу випадкових чисел

В ході виконання конкурсної роботи досліджувався розподіл випадкових чисел, що приймав за джерело ентропії шуми звукової карти. Використовувалось 10 000 випадкових чисел, взятих з діапазону від 1 до 100.

По осі ОУ відкладається кількість елементів, а по осі ОХ – елементи, тобто генерована послідовність, що використовувала за джерело ентропії шуми звукової карти. Відповідно, послідовність є нормованою до вказаного діапазону. Для створення графіків необхідно підрахувати кількість входжень кожного значення до послідовності. Підрахунок реалізовується формулою:

=СЧЁТЕСЛИ($A$1:$NTP$1; B3),

де $A$1:$NTP$1 – перелік нормованих елементів звукового масиву,

B3 (B4, B5… B102) – відповідно значення, за яким відбувається перевірка наявності елементу 1 (2, 3…100) у переліку.

Для демонстрації результатів роботи бібліотеки, створено послідовності чисел, запис звуку для яких відбувався у різних умовах. Наприклад, запис звуку у шумній аудиторії демонструє таку послідовність чисел (додаток Д.1)

При записі звуку в тихій аудиторії, графік розподілу відповідної послідовності відповідає графіку з додатку Д.2.

Відмінності мало помітні, тому необхідно провести ще декілька тестів, результати яких можна побачити у додатках Д.3 та Д.4.

В силу того, що звук, який отримує бібліотека для генерації випадкової послідовності чисел є випадковим, важко прослідкувати за особливостями отримуваного звуку. Тому, постає питання в перевірці того, до якого типу розподілу належать отримані послідовності.

Припустимо, що отримані послідовності відповідають рівномірному розподілу виходячи із означення рівномірного розподілу: безперервна величина X розподілена рівномірно на інтервалі (a, b), якщо всі її можливі значення знаходяться на цьому інтервалі і щільність розподілу ймовірностей постійна. Таблиця обрахування показників приведена у додатку В.

Для оцінки ряду розподілу знайдемо такі показники:

1) Середня зважена (вибіркова середня):

$$\bar{x} = \frac{\sum x_i * f_i}{\sum f_i} = \frac{5063.425}{107.41} = 47.14$$

2) Розмах варіації − різниця між максимальним і мінімальним значеннями ознаки первинного ряду:

$$R = x_{max} - x_{min} = 100 - 1 = 99$$

3) Дисперсія − характеризує міру розкиду близько її середнього значення:

$$D = \frac{\sum(x_i - \bar{x})^2 f_i}{\sum f_i} = \frac{92332.69}{107.41} = 859.63$$

4) Незміщена оцінка дисперсії:

$$S^2 = \frac{\sum(x_i - \bar{x})^2 f_i}{\sum f_i - 1} = \frac{92332.69}{106.41} = 867.71$$

5) Середнє квадратичне відхилення:

$$\sigma = \sqrt{D} = \sqrt{859.628} = 29.32$$

Кожне значення ряду відрізняється від середнього значення 47.14 в середньому на 29.32.

6) Оцінка середньоквадратичного відхилення:

$$s = \sqrt{S^2} = \sqrt{867.71} = 29.46$$

Таблиця обрахування теоретичних частот наведена у додатку Г.

Для того, щоб перевірити гіпотезу про рівномірний розподіл X, тобто, за законом: $f\ (x)\ =\ \frac{1}{(b-a)}$ на інтервалі (a, b), необхідно:

1) Оцінити параметри a і b − кінці інтервалу, в якому спостерігалися можливі значення X, за формулами (через знак * позначені оцінки параметрів):

$a^* = \bar{x} - \sqrt{3}\sigma,\ b^* = \bar{x} + \sqrt{3}\sigma$

$a^* = 47.14 - \sqrt{3} * 29.32 = -3.64,\ b^* = 47.14 - \sqrt{3} * 29.32 = 07.92;$

2) Визначити щільність передбачуваного рівномірного розподілу:

$f(x) = \frac{1}{b^* - a^*} = \frac{1}{(97.92 - (-3.64))} = 0.00985;$

3) Знайти теоретичні частоти:

$$n_1 = n * f(x)(x_1 - a^*) = 107.41 * 0.00985\big(10 - (-3.64)\big) = 14.43$$

$$n_{10} = n * f(x)(b^* - x_9) = 107.41 * 0.00985(97.92 - 90) = 8.38$$

Інші n_s будуть дорівнювати: $n_s = n * f(x)(x_i - x_{i-1})$

4) Визначимо межу критичної області. Статистика Пірсона вимірює різницю між емпіричним і теоретичним розподілами, тому чим більше її бачимо значення $K_{спос}$, тим сильніше аргумент проти основної гіпотези. Тому критична область для цієї статистики завжди правобічна: $[K_{kp}; +\infty]$.

Її межу $K_{kp} = \chi^2(k - r - 1; \alpha)$ знаходимо за таблицями розподілу χ^2 і заданим значенням s, k (число інтервалів), r = 2 (параметри a і b).

$K_{kp} = 14.06714;$ $K_{спос} = 9.26$

Спостережуване значення статистики Пірсона не влучає у критичну область: $K_{спос} < K_{kp}$, тому немає підстав відкидати основну гіпотезу. Справедливо припущення про те, що дані вибірки мають рівномірний закон.

3.3. Тестування

Варто зауважити, що генератори випадкових чисел мають свої недоліки та переваги, тому кожен із користувачів обирає генератор випадкових чисел відповідно до власних бажань та поставлених перед ним задач. Виходячи з цього, постає необхідність протестувати та виявити, за яких умов виникає необхідність використання розроблюваної бібліотеки.

До переваг апаратних генераторів, як і створюваного, можна віднести високу стійкість отриманої послідовності, адже відтворити повністю початкові умови не є можливим, адже користувач є джерелом ентропії.

Основна проблема апаратних генераторів випадкових чисел – це їх відносно повільна робота в порівнянні з генераторами псевдовипадкових генераторів випадкових чисел. Через що і створювана бібліотека генерації випадкових послідовностей на основі шумів звукової карти працює повільніше за рахунок того, що в процесі своєї роботи має записувати звук, який є основою для створюваної послідовності.

Тестування показало, що при стандартному нормуванні вхідної послідовності, вихідна включає в себе велику кількість однакових елементів, а саме – повторюється мінімальний елемент, тобто значення, якому відповідає початок діапазону вихідної послідовності.

При формуванні послідовності із 10 тисяч чисел із діапазону від 1 до 100 отримується результат, що включає в себе велику кількість мінімальних елементів (відповідно число 1), що проілюстровано у додатку Е.1.

Аби переконатися, що мінімальних елементів дійсно занадто багато, був побудований графік, що підтвердив дане припущення (додаток Е.2).

Дана ситуація виходить виключно із особливостей записування звуку, внаслідок чого постає необхідність змінити функцію нормування елементів задля виключення даної ситуації.

В результаті досліджень було розроблено алгоритм, що дозволяє виключити ситуацію перекосу математичного очікування в послідовності. Її суть доволі проста: виключити усі мінімальні елементи з послідовності, що отримує користувач.

Для включення даного алгоритму в бібліотеку необхідно змінити цикл перезапису масиву, в який записується нормовані елементи, які розраховуються за формулою. Міняти формулу практично не потрібно, тільки відняти від неї 1, так як на початку необхідно збільшити діапазон на 1 (тобто, b = b + 1), оскільки буде проведена процедура вилучення усіх мінімальних елементів. Сам процес вилучення елементів являє собою перевірку того, чи є записуваний елемент рівний мінімальному, в цьому випадку запис не відбувається і процес запису переходить до наступного елементу. Описується даний алгоритм таким циклом:

```
for (int i = 0; i < len; i++)
{
     if ( (((mas[j] - min)*(b - a)/(max - min)) + a) != a)
     { mas2[i] = ((mas[j] - min)*(b - a)/(max - min)) + a - 1; }
     else
     { i--; }
```

```
j++;
}
```

В результаті, отримана послідовність так само знаходиться у вказаному діапазоні, проте не має у собі недопустимої кількості повторень. Результат даних змін у роботі бібліотеки продемонстровано у додатку Е.3.

В результаті підрахунку елементів, виявлено, що тепер кількість мінімальних елементів дорівнює 439, що є припустимим для результатів роботи бібліотеки (повторний експеримент продемонстрував подібні результати, додаток Е.4). Стрибки різкої зміни кількості елементів між сусідніми значеннями вказують виключно на результат нормування записаного звуку.

3.4. Інструкція з використання бібліотеки

При створенні програми, що використовує бібліотеку генерації випадкових чисел на основі шумів звукової карти, необхідно виконати ініціалізацію даної бібліотеки, відповідно, написавши:

```
#include "srgenerate.h"
```

Далі виконується написання коду відповідно до власних бажань.

Коли настає момент використання функції для отримання випадкової послідовності, необхідно створити динамічний масив, в який, як результат роботи бібліотеки, запишеться послідовність. Для цього необхідно написати:

```
unsigned char *mas = (unsigned char*)malloc(len);
```

де mas – масив, len – кількість елементів.

Наступним кроком необхідно викликати функцію з параметрами:

```
mas = Generate::MyGenerate::generateNumbers(len, a, b);
```

де a – початок інтервалу, b – кінець інтервалу.

Фінальна демонстрація зображена у додатку Ж, де сгенеровано дві випадкові послідовності – числову та символьну (діапазон встановлювався за допомогою ASCII-кодування).

ВИСНОВКИ

Потреба генерації випадкових чисел з’являється в багатьох прикладних задачах. Найбільш вимогливими до якості випадкових чисел є задачі, пов’язані з комп’ютерним моделюванням та криптографією. В цих випадках недоречно використовувати стандартні детерміновані генератори псевдовипадкових чисел, тому постає необхідність у створенні генераторів випадкових чисел із джерелом ентропії.

В ході виконання конкурсної роботи було розроблено бібліотеку генерації випадкових чисел на основі шумів звукової карти.

Бібліотека написана з використанням стандартних засобів мови програмування C++, додаток для запису звуку застосовує функції Windows API, що надають всю необхідну функціональність для роботи зі звуком і забезпечують достатню ефективність і швидкодію.

Алгоритм роботи генератора передбачає запис звуку у побічному додатку та його передачу до бібліотеки, в якій реалізовується нормування елементів, які записались, до діапазону, який користувач ввів при виклику функції.

При тестуванні бібліотеки було визначено переваги та недоліки обраного алгоритму та ентропії. Основним недоліком бібліотеки є час її виконання, який є більшим, ніж у бібліотеках, що генерують псевдовипадкові послідовності. Проте, цей недолік компенсується тим, що вихідна послідовність має істинно випадковий характер, що залежить від записуваного звуку.

Створена бібліотека генерації випадкових чисел може бути використана в проектах, які мають потребу в високоякісних послідовностях випадкових чисел. В майбутньому генератор випадкових чисел може бути вдосконалений за рахунок використання більш чистого звуку (для кращої випадковості чисел), можливість вибору користувачем розподілу, за яким генеруватимуться випадкові числа, перевірка правильності та випадковості розподілу за статистичним критерієм Стьюдента або Пірсона.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аудиоустройства [Электронный ресурс] – Режим доступа: http://perscom.ru/index.php/2012-03-17-20-47-11
2. Бараш Л. Ю. Алгоритм AKS проверки чисел на простоту и поиск констант генераторов псевдослучайных чисел // Безопасность информационных технологий. – 2005. – № 2. – 27-38 с.
3. Гэри Неббет. Справочник по базовым функциям API Windows NT/2000. – М.: «Вильямс», 2002. – 528 с.
4. Діффі У. New directions in cryptography / У. Діффі, Хелман М. /IEEE Trans. Inform. Theory, 22 – 1976 – 644-654 с.
5. Жельников В. Псевдослучайные последовательности чисел // Криптография от папируса до компьютера. – М.: ABF, 1996. – 335 с.
6. Зеннер Е. On Cryptographic Properties of LFSR-based Pseudorandom Generators. – Мангейм, 2004. – 102 с.
7. Зубинский А.Т. В поисках случайности // Компьютерное обозрение. – 2003. – № 29.
8. Иванов М. А. Глава 4. Методика оценки качества генераторов ПСП. // Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М. А. Иванов, И. В. Чугунков. – М.: КУДИЦ-ОБРАЗ. 2003 – 240 с.
9. Каханер Д. Численные методы и математическое обеспечение: Пер. с англ. / Д. Каханер, К. Моулер, С. Нэш. М.: Мир, 1998. − 575 с.
10. Кнут Д. Э. Глава 3. Случайные числа // Искусство программирования. – 3-е изд. – М.: Вильямс, 2000. – Т. 2. Получисленные алгоритмы. – 832 с.
11. Кнут Д. Э. Глава 3.3. Спектральный критерий // Искусство программирования. – 1999 – 129-130 с.
12. Корнер Б. Russian engineer a brilliant slot machine cheat – and casinos have no fix [Електронний ресурс] – Режим доступу:

https://www.wired.com/2017/02/russians-engineer-brilliant-slot-machine-cheat-casinos-no-fix/?mbid=social_fb.

13. Коробейников А. Г. Математические основы криптологии: учеб. пособие / А. Г. Коробейников, Ю.А. Гатчин. − СПб: СПб ГУ ИТМО, 2004. – 106 с.

14. Лифшиц Ю.О. Курс Современные задачи криптографии // Лекция 9: Псевдослучайные генераторы. – 2005.

15. Марсалия Г. Random numbers fall mainly in the planes – 1968 – 25-28.

16. Мацумото М. CryptMT Stream Cipher Ver. 3.The eSTREAM Project / М. Мацумото, М. Сайто, Т. Нішімура, М. Хагіте. – 2007.

17. Мацумото М. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator / М. Мацумото, Т. Нішімура. – ACM Trans. on Modeling and Computer Simulations 8. – 1998 – 3-30 с.

18. Новітні комп’ютерні технології. – Кривий Ріг : Видавничий центр ДВНЗ «Криворізький національний університет», 2017. – Том XV. – 85-87 с.

19. Нішімура Т. Tables of 64-bit Mersenne twisters // ACM Trans. on Modeling and Computer Simulations 10. – 2000 – 248-357 с.

20. Рок А. Генератори псевдовипадкових чисел для криптографічних додатків. − Зальцбург, 2005. – 57-65 с.

21. Форсайт Дж. Машинные методы математических вычислений / Дж. Форсайт, М. Малькольм, К. Моулер. − М.: Мир, 1980. − 279 с.

22. Хортон А. Microsoft Visual C++ 2005: базовый курс – М.: «Диалектика», 2007. – 1152 с.

23. Шапочка Н.В. Аналіз атак на генератори випадкових бітів // Інформаційна безпека – ХНУРЭ – 2010.

24. Щупак Ю. Win3 2 API. Эффективная разработка приложений – Спб.: Питер – 2007 – 15 с.

Додаток А

Таблиця маркування гнізд звукової карти

Колір	Функція
Рожевий	Аналоговий вхід для мікрофону
Блакитний	Аналоговий вхід line-in
Світло-зелений	Аналоговий вихід для динаміків або навушників, у системах об'ємного звуку – вихід для передніх динаміків
Чорний	Аналоговий вихід для тильних динаміків.
Помаранчевий	Цифровий вихід (S/PDIF), іноді аналоговий вихід для центральних або низькочастотних динаміків.

Додаток Б

Блок-схеми розроблюваного додатку

Рис.Б.1. Блок-схема загального алгоритму роботи

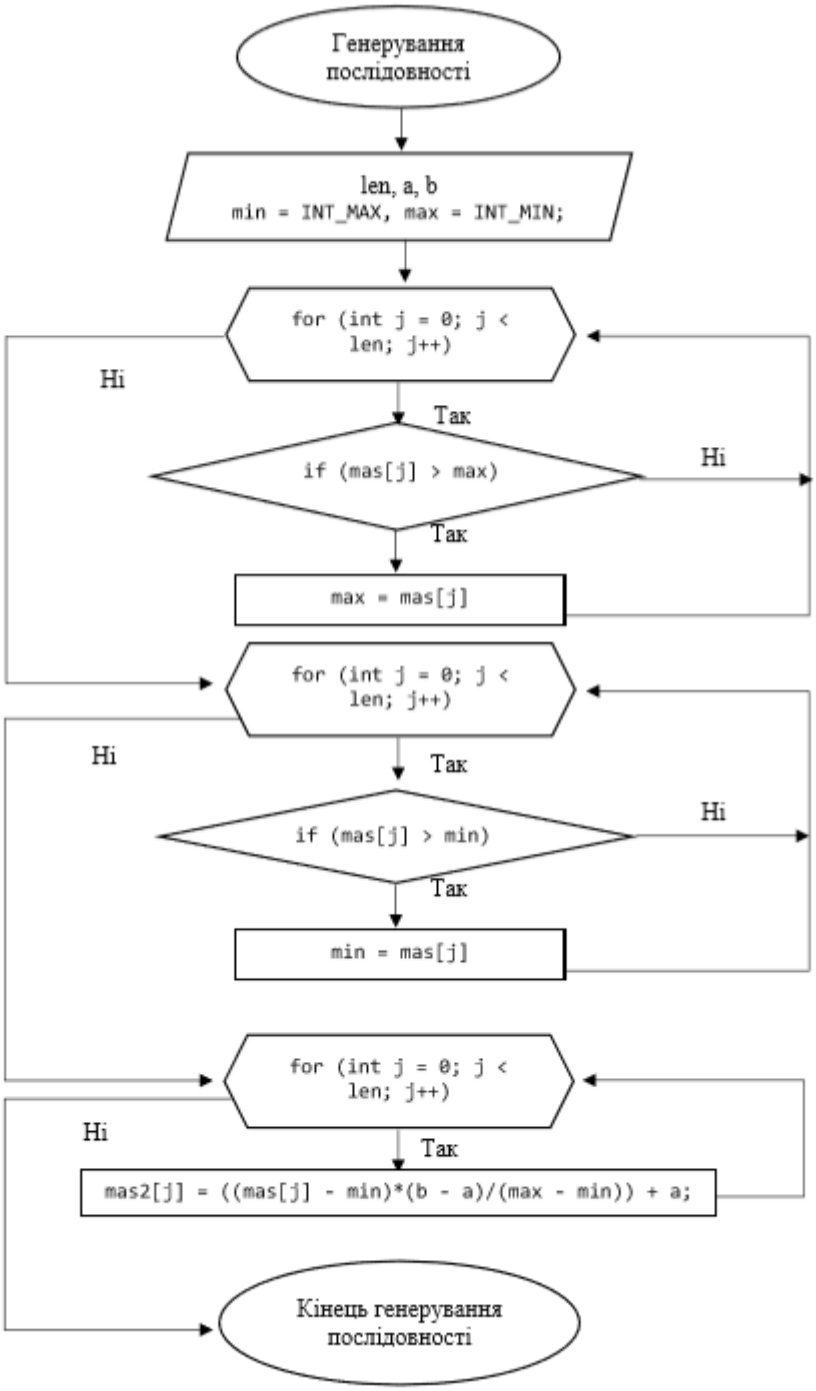

Рис.Б.2. Блок-схема генерування послідовності випадкових чисел

Додаток В

Таблиця обрахування показників

Групи	Сер. інт.	К-сть, f_i	$x_i * f_i$	Нак. част, S	$\lvert x - x_{cp} \rvert * f_i$	$(x - x_{cp})^2 * f_i$	Частота, f_i/f
1 - 10	5.5	12.35	67.925	12.35	514.268	21414.663	0.115
10 - 20	15	13.98	209.7	26.33	449.333	14442.039	0.13
20 - 30	25	10.53	263.25	36.86	233.146	5162.102	0.098
30 - 40	35	12.52	438.2	49.38	152.007	1845.525	0.117
40 - 50	45	9.28	417.6	58.66	19.869	42.542	0.0864
50 - 60	55	9.53	524.15	68.19	74.895	588.596	0.0887
60 - 70	65	13.27	862.55	81.46	236.988	4232.34	0.124
70 - 80	75	5.21	390.75	86.67	145.145	4043.578	0.0485
80 - 90	85	8.1	688.5	94.77	306.657	11609.703	0.0754
90 - 100	95	12.64	1200.8	107.41	604.937	28951.601	0.118
Всього		107.41	5063.425		2737.243	92332.689	1

Додаток Г

Таблиця обрахування теоретичних частот

i	n_i	n_i^*	$n_i - n_i^*$	$(n_i - n_i^*)^2$	$(n_i - n_i^*)^2/n_i^*$
1	12,35	14,4266	-2,08	4,31	0,29891087
2	13,98	9,5179	4,46	19,91	2,091883337
3	10,53	9,5179	1,01	1,02	0,107623153
4	12,52	9,5179	3,00	9,01	0,946911021
5	9,28	9,5179	-0,24	0,06	0,005946313
6	9,53	9,5179	0,01	0,00	1,53826E-05
7	13,27	9,5179	3,75	14,08	1,479134516
8	5,21	9,5179	-4,31	18,56	1,949800104
9	8,10	9,5179	-1,42	2,01	0,21122731
10	12,64	8,3798	4,26	18,15	2,165839762
Всього	107,41				9,257291769

Додаток Д

Дослідження характеру розподілу випадкових чисел

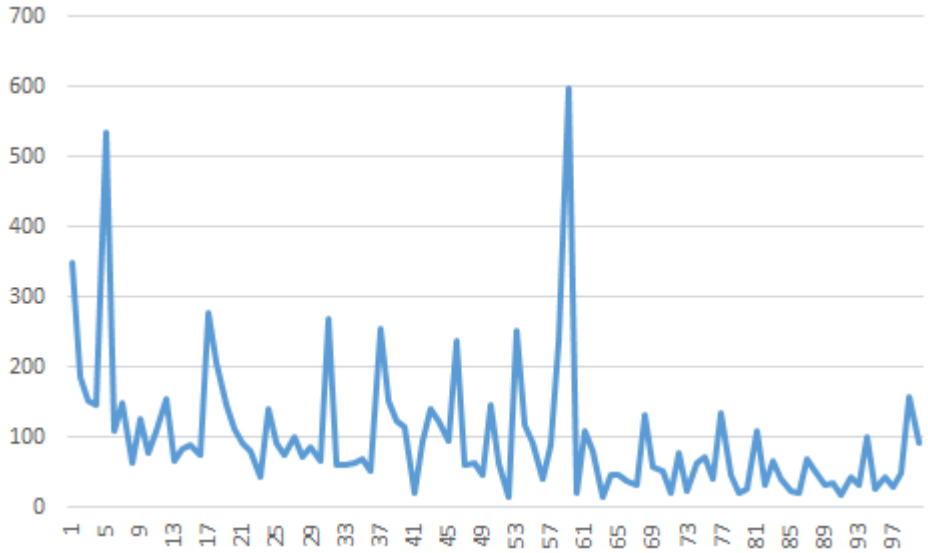

Рис.Д.1 Розподіл звуку у шумній аудиторії

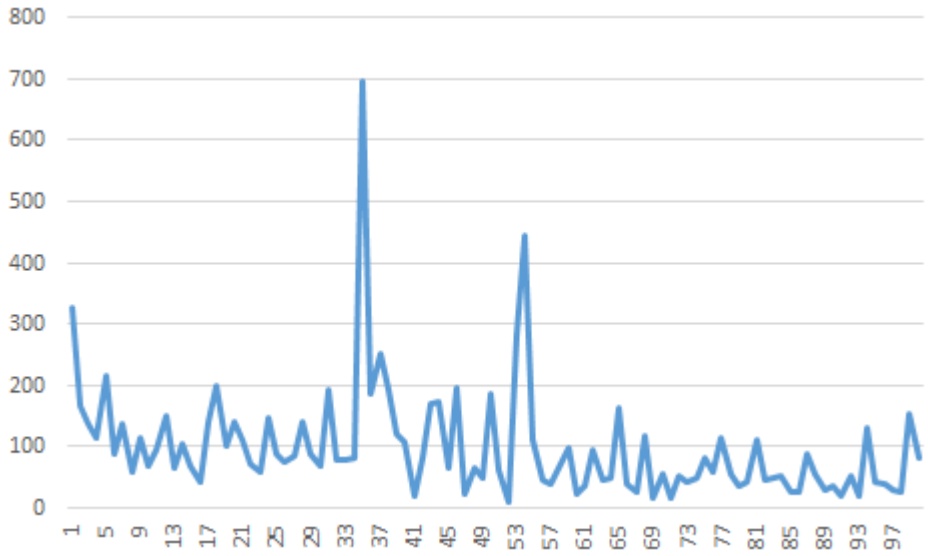

Рис.Д.2 Розподіл звуку в тихій аудиторії

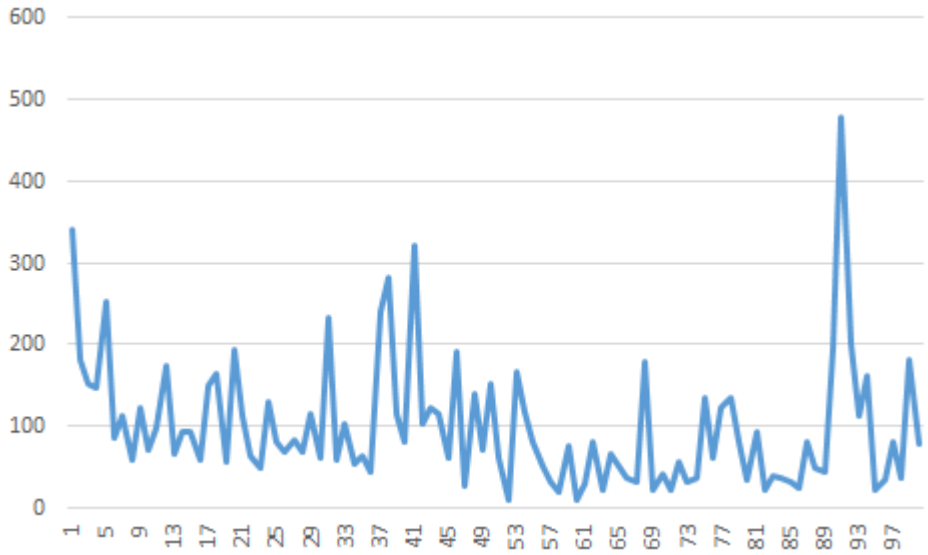

Рис.Д.3 Графік повторного запису № 1

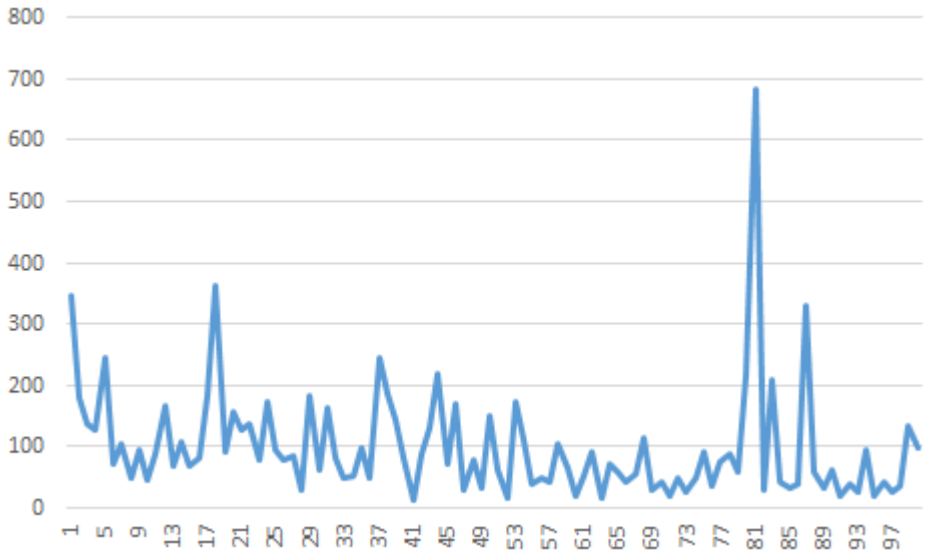

Рис.Д.4 Графік повторного запису № 2

Тестування створюваного додатку

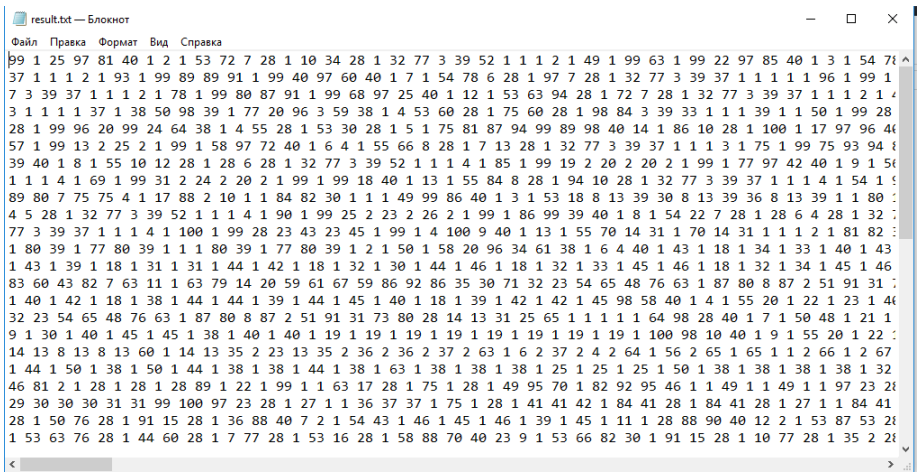

Рис.Е.1 Результат роботи бібліотеки при діапазоні (1; 100)

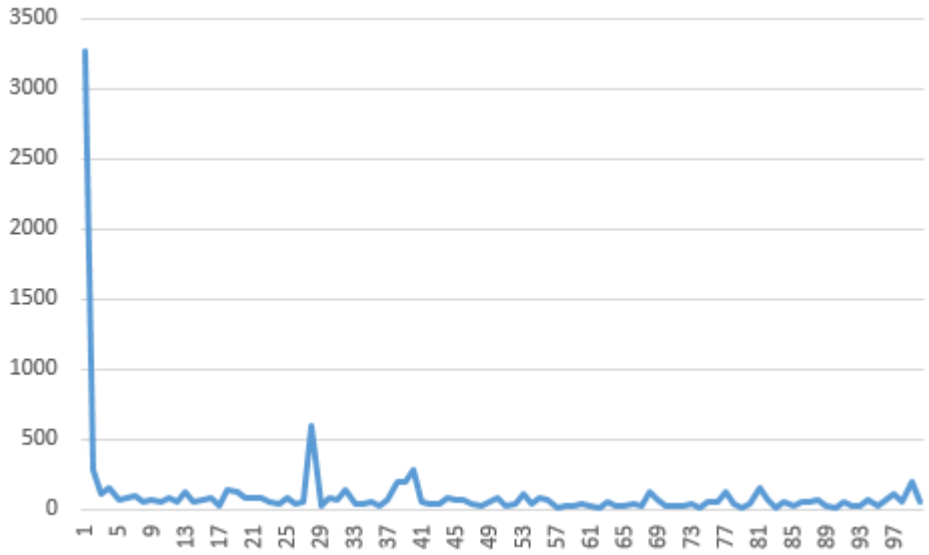

Рис.Е.2 Графік розподілу елементів

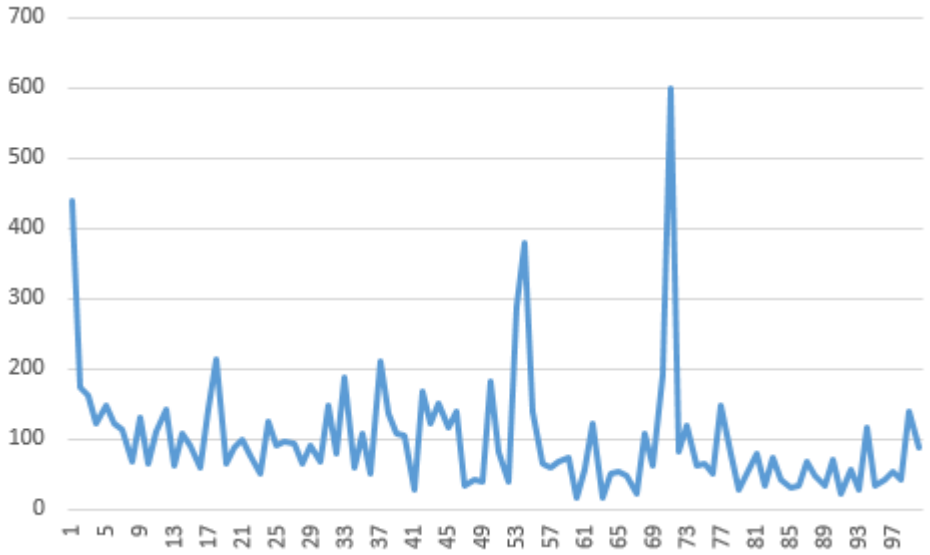

Рис.Е.3 Результат алгоритму виключення ситуації зміщення математичного очікування послідовності вихідних бітів

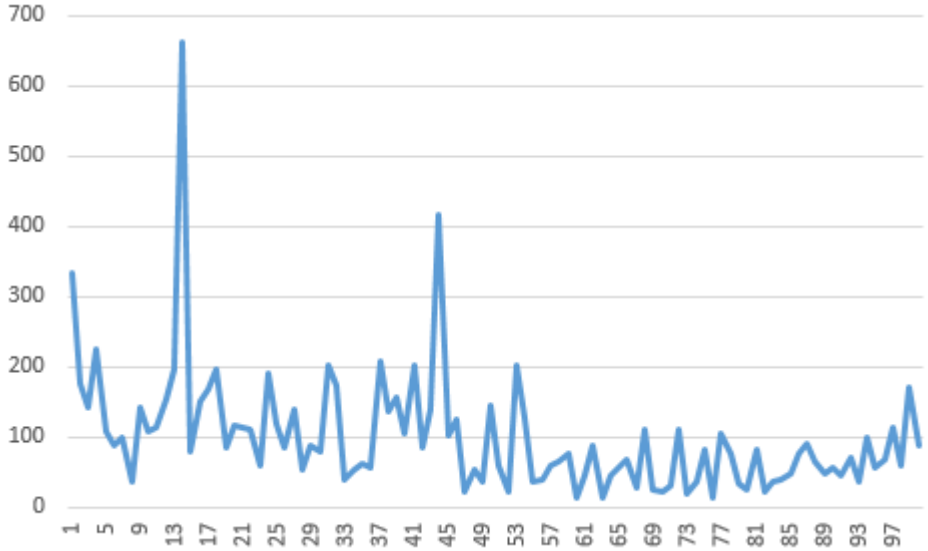

Рис.Е.4 Результат алгоритму виключення ситуації зміщення математичного очікування послідовності вихідних бітів (повторна побудова)

Додаток Ж

Інструкція з використання бібліотеки

Рис.Ж.1 Генерована послідовність чисел

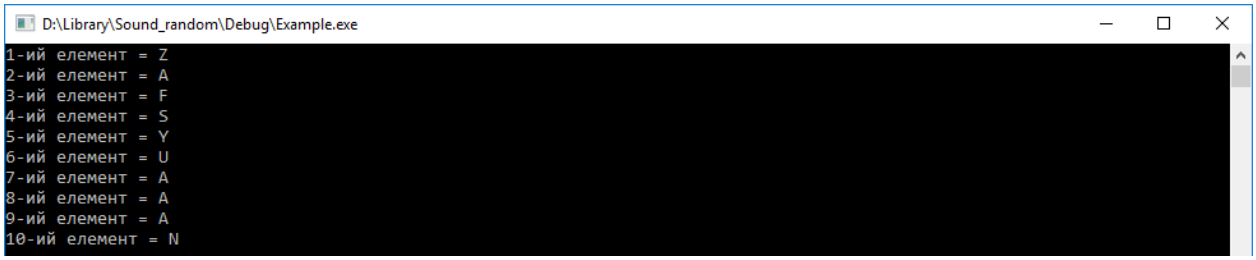

Рис.Ж.2 Генерована послідовність літер