

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ МОДЕЛЕЙ
МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ДАНИХ
ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИМУЛЯЦІЇ
RESEARCH OF MATHEMATICAL MODELS OF MECHANICAL DATA
ENCRYPTION DEVICES AND SOFTWARE IMPLEMENTATION
OF A SIMULATION

Виконав(ла): студент(ка) 2 курсу, групи КНТ-124м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КУЧЕР А.О.

(ПРІЗВИЩЕ та ініціали)

Керівник ЗАЙКО Т.А.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КУЧЕРА Андрія Олеговича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження математичних моделей механічних пристроїв шифрування даних та програмна реалізація симуляції. Research of Mathematical Models of Mechanical Data Encryption Devices and Software Implementation of a Simulation.

керівник проєкту (роботи) к.т.н., доцент, ЗАЙКО Тетяна Анатоліївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Математичне моделювання механічних пристроїв шифрування. 3. Матеріали і методи. 4. Опис програми. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЗАЙКО Т.А., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 30 ” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	9-10 тижні	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Андрій КУЧЕР
(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ЗАЙКО
(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
82 с., 2 табл., 20 рис., 3 дод., 29 джерел.

PYTHON, VISUAL STUDIO CODE, ШИФРУВАННЯ ДАНИХ, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СИМУЛЯЦІЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних.

Предмет дослідження – програмні засоби для симуляції роботи механічних пристроїв шифрування даних.

Мета роботи – дослідження та розробка програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки Visual Studio Code.

Результати. Розроблено програмне забезпечення для симуляції роботи механічних пристроїв шифрування даних за допомогою мови програмування Python, та середовища розробки Visual Studio Code.

Практична цінність роботи полягає у розробці програмного забезпечення симулятора механічної шифрувальної машини.

Висновки. Виконано проєктування програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних. Розроблено програмне забезпечення для симуляції роботи механічних пристроїв шифрування даних. Здійснено тестування розробленого програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних.

Галузь використання – навчальні, дослідницькі та демонстраційні цілі.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 82 pages, 2 tables, 20 figures, 3 appendixes, 29 sources.

PYTHON, VISUAL STUDIO CODE, DATA ENCRYPTION, PROGRAMMING LANGUAGE, SOFTWARE, SIMULATION.

Object of research – the process of software development for simulating the operation of mechanical data encryption devices.

Subject of research – software tools for simulating the operation of mechanical data encryption devices.

Purpose of the work – research and development of software for simulating the operation of mechanical data encryption devices.

Materials, methods, and technical means: the Python programming language, the Visual Studio Code development environment.

Results. Software for simulating the operation of mechanical data encryption devices was developed using the Python programming language and the Visual Studio Code development environment.

Practical value of the work lies in the development of software for a mechanical cipher machine simulator.

Conclusions. The software design for simulating the operation of mechanical data encryption devices was carried out. Software for simulating the operation of mechanical data encryption devices was developed. The developed software for simulating the operation of mechanical data encryption devices was tested.

Field of application – educational, research, and demonstrational purposes.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Історичні передумови виникнення механічних пристроїв шифрування.	11
1.2 Класифікація механічних пристроїв шифрування.....	12
1.3 Принципи функціонування механічних криптографічних систем	13
1.4 Математичні основи аналізу механічних пристроїв шифрування.....	16
1.5 Проблеми та обмеження механічних криптографічних пристроїв.....	18
1.6 Висновки за розділом 1.....	20
2 Математичне моделювання механічних пристроїв шифрування	22
2.1 Формалізація процесу шифрування механічних криптосистем.....	22
2.2 Моделі підстановочних і перестановочних механізмів шифрування.....	23
2.3 Модель роторних механічних пристроїв	25
2.4 Моделювання динаміки зміни станів механічних криптографічних систем.....	27
2.5 Аналіз властивостей математичних моделей механічних криптосистем	29
2.6 Висновки до розділу 2	30
3 Матеріали і методи.....	32
3.1 Вибір мови програмування	32
3.2 Вибір середовища розробки для створення програмного забезпечення симуляції механічних пристроїв шифрування даних	34
3.3 Висновки за розділом 3.....	37
4 Опис програми.....	39
4.1 Структура програмного забезпечення симуляції механічних пристроїв шифрування даних.....	39
4.2 Функціонування програмного забезпечення симулятора.....	43

4.3	Проектування інтерфейсу програмного забезпечення для симулятора механічної шифрувальної машини	45
4.4	Висновки за розділом 4.....	48
5	Експлуатація, тестування та експериментальне дослідження програми	49
5.1	Призначення й умови застосування програми	49
5.2	Характеристики програми симулятора механічної шифрувальної машини	51
5.3	Інструкція по експлуатації програми	52
5.3.1	Звернення до програми	52
5.3.2	Вхідні й вихідні дані	52
5.3.3	Повідомлення.....	53
5.4	Виконання програмного забезпечення симулятором механічної шифрувальної машини.....	54
5.5	Тестування програмного забезпечення симулятора механічної шифрувальної машини.....	55
5.6	Висновки за розділом 5.....	56
	Висновки	57
	Перелік джерел посилань	60
	Додаток А Технічне завдання	63
	Додаток Б Фрагмент тексту програми	69
	Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GUI – графічний інтерфейс користувача;

ЗІ – захист інформації;

ІБ – інформаційна безпека;

ІТ – інформаційні технології;

ПЗ – програмне забезпечення.

ВСТУП

Стрімкий розвиток інформаційних технологій (ІТ), глобальна цифровізація суспільства та зростання обсягів передавання і зберігання даних зумовлюють підвищені вимоги до забезпечення інформаційної безпеки (ІБ). У сучасному інформаційному просторі захист даних є критично важливим елементом функціонування державних, комерційних і приватних інформаційних систем [1], [2]. Криптографічні методи захисту інформації посідають центральне місце серед засобів забезпечення конфіденційності, цілісності та автентичності даних [1].

Історично, розвиток криптографії розпочинався з використання механічних пристроїв шифрування, які реалізовували криптографічні перетворення за допомогою фізичних механізмів. До таких пристроїв належать шифрувальні диски, роторні машини, комбінаторні механізми та інші апаратні реалізації класичних криптосистем [2], [24], [28]. Незважаючи на те, що більшість механічних шифраторів сьогодні не застосовуються для реального захисту інформації, вони відіграли ключову роль у формуванні фундаментальних принципів сучасної криптографії.

Механічні криптографічні пристрої стали прототипами багатьох сучасних алгоритмічних конструкцій. Саме в них вперше були реалізовані такі базові криптографічні ідеї, як підстановки, перестановки, композиція перетворень, використання ключів, керування станом системи та циклічність операцій [3], [24], [28]. Принципи роботи механічних шифраторів згодом лягли в основу програмних та апаратних криптосистем, зокрема симетричних і асиметричних алгоритмів шифрування.

Важливим аспектом вивчення механічних пристроїв шифрування є можливість їх формалізації у вигляді математичних моделей. Робота таких пристроїв може бути описана засобами теорії множин, теорії перестановок, алгебраїчних структур, матричних перетворень та теорії автоматів [4]. Формалізація дозволяє не лише описати принципи функціонування

механічних криптосистем, а й здійснювати їх криптографічний аналіз, оцінювати стійкість та виявляти вразливості.

У сучасній криптології математичне моделювання є одним із ключових інструментів дослідження криптосистем. Побудова математичних моделей дає змогу перейти від інтуїтивного розуміння процесів шифрування до формального аналізу, що є необхідною умовою для розроблення надійних засобів захисту інформації [5]. Водночас дослідження механічних шифрувальних пристроїв з позицій сучасної математики дозволяє глибше усвідомити еволюцію криптографічних методів та закономірності їх розвитку.

Тому актуальною є розробка програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних.

У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання дослідження формальних математичних моделей механічних пристроїв шифрування даних та програмних моделей таких систем без використання фізичних механізмів.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- проаналізувати розвиток та еволюцію механічних пристроїв шифрування інформації;
- дослідити принципи функціонування класичних механічних криптографічних систем;
- провести класифікацію механічних пристроїв шифрування за типами криптографічних перетворень;
- формалізувати процеси шифрування та дешифрування механічних криптосистем у вигляді математичних моделей;
- розробити програмне забезпечення симуляції роботи механічного криптомеханізму шифрування даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історичні передумови виникнення механічних пристроїв шифрування

Потреба у захисті інформації (ЗІ) виникла одночасно з появою організованих форм суспільної діяльності, таких як державне управління, військова справа та дипломатія [2], [3], [6]. Передавання конфіденційних повідомлень завжди супроводжувалося ризиком їх перехоплення, що зумовило пошук методів приховування змісту інформації від сторонніх осіб. Уже в античні часи використовувалися примітивні методи шифрування, засновані на підстановці або перестановці символів, зокрема шифр Цезаря та його варіації [7], [24], [28].

Із розвитком писемності, ускладненням мовних структур і зростанням обсягів передаваних повідомлень ручні методи шифрування поступово втрачали ефективність. Людський фактор, обмежена швидкість виконання операцій та висока ймовірність помилок стали серйозними перешкодами для використання ручних криптографічних методів у масштабних інформаційних системах [8], [24], [28]. Це зумовило необхідність створення механізмів, які могли б автоматизувати процес шифрування та забезпечити більшу надійність криптографічних перетворень [2].

Перші механічні пристрої шифрування з'явилися як спроба поєднати криптографічні принципи з інженерними рішеннями. До таких пристроїв належать шифрувальні диски Леонардо да Вінчі, криптодиски Альберті, шифрувальні циліндри Джефферсона, які дозволяли реалізувати складніші схеми підстановки та перестановки символів [3], [9], [24], [28]. Використання рухомих елементів дало змогу змінювати ключі шифрування без повного переписування алгоритму, що стало важливим кроком у розвитку криптографічних систем.

Подальший розвиток механічних криптографічних пристроїв був тісно пов'язаний із потребами військової та дипломатичної сфери. Особливого

значення механічні шифратори набули у період Першої та Другої світових воєн, коли обсяги переданої інформації зросли в рази, а вимоги до швидкості та секретності шифрування стали критичними [3], [10], [24], [28]. Саме в цей період були створені роторні шифрувальні машини, які забезпечували складну композицію підстановок і перестановок та динамічну зміну ключа в процесі роботи.

Роторні механізми дозволили реалізувати принцип багаторазового перетворення символів, при якому одна й та сама літера відкритого тексту могла шифруватися по-різному залежно від її позиції в повідомленні. Це значно ускладнювало криптоаналіз і стало важливим кроком на шляху до створення поліалфавітних криптосистем. Фактично роторні машини заклали основу для сучасних концепцій потокового та блочного шифрування [1], [6], [11].

Важливо зазначити, що механічні пристрої шифрування не лише виконували практичну функцію захисту інформації, а й сприяли формуванню теоретичних основ криптографії. Аналіз принципів їх роботи дозволив сформулювати ключові криптографічні поняття, зокрема роль ключа, оборотність перетворень, композицію операцій та залежність стійкості системи від складності математичної структури [4], [5].

Таким чином, історичний розвиток механічних криптографічних пристроїв став важливим етапом еволюції криптографії. Саме ці системи створили фундамент для переходу від інтуїтивних методів шифрування до формалізованих математичних моделей, що використовуються у сучасних криптографічних алгоритмах [1], [7], [12], [24], [28].

1.2 Класифікація механічних пристроїв шифрування

Механічні пристрої шифрування можна класифікувати за кількома ознаками, залежно від принципу побудови та реалізованих криптографічних операцій [1], [6].

За типом криптографічного перетворення виділяють:

- підстановочні механічні шифри, у яких кожен символ відкритого тексту замінюється іншим символом;
- перестановочні механічні шифри, що змінюють порядок символів у повідомленні;
- комбіновані системи, які поєднують підстановку та перестановку;
- поліалфавітні механічні шифри, у яких використовується кілька алфавітів;
- поліграмні механічні шифри, що працюють з групами символів [2], [5], [13].

За конструктивними особливостями механічні криптопристрої поділяють на:

- дискові системи;
- роторні системи;
- барабанні механізми;
- комбінаторні пристрої з декількома рухомими елементами.

Окрему групу становлять механічні пристрої з керованою динамікою ключа, у яких параметри шифрування змінюються під час роботи пристрою. Саме такі системи стали попередниками сучасних поточкових і блочних криптографічних алгоритмів [1].

1.3 Принципи функціонування механічних криптографічних систем

Потреба у захисті інформації виникла одночасно з появою організованих форм суспільної діяльності, таких як державне управління, військова справа та дипломатія [2], [3], [14]. Передавання конфіденційних повідомлень у будь-яку історичну епоху супроводжувалося ризиком їх перехоплення, спотворення або підміни, що зумовлювало необхідність розроблення спеціальних методів приховування змісту інформації від

сторонніх осіб. Умови ведення воєнних дій, дипломатичних переговорів і управління територіями вимагали забезпечення таємності наказів, донесень і стратегічних планів, що стало одним із ключових стимулів розвитку криптографії.

Уже в античні часи використовувалися примітивні методи шифрування, засновані на підстановці або перестановці символів, зокрема шифр Цезаря та його численні варіації [15], [24], [28]. Ці методи дозволяли відносно просто приховати зміст повідомлення, однак їх криптографічна стійкість була обмеженою через збереження статистичних властивостей мови. Попри це, саме класичні ручні шифри заклали основи для подальшого розвитку криптографічних підходів та формування поняття алгоритму шифрування.

Із розвитком писемності, ускладненням мовних структур і зростанням обсягів передаваних повідомлень ручні методи шифрування поступово втрачали ефективність [15], [16], [24], [28]. Людський фактор, обмежена швидкість виконання операцій, втома операторів та висока ймовірність помилок стали серйозними перешкодами для використання ручних криптографічних методів у масштабних інформаційних системах. У великих організаційних структурах та арміях зростала потреба у швидкому, повторюваному та стандартизованому процесі шифрування, що зумовило необхідність створення механізмів, здатних автоматизувати криптографічні перетворення та підвищити їх надійність [2], [17].

Перші механічні пристрої шифрування з'явилися як спроба поєднати криптографічні принципи з інженерними рішеннями. До таких пристроїв належать шифрувальні диски Леонардо да Вінчі, криптодиски Альберті, шифрувальні циліндри Джефферсона, які дозволяли реалізувати складніші схеми підстановки та перестановки символів порівняно з ручними методами [3], [18], [24], [28]. Використання рухомих механічних елементів дало змогу змінювати ключі шифрування шляхом фізичного

переналаштовування пристрою без повного переписування алгоритму, що суттєво підвищувало гнучкість криптографічних систем.

Подальший розвиток механічних криптографічних пристроїв був тісно пов'язаний із потребами військової та дипломатичної сфери. Особливого значення механічні шифратори набули у період Першої та Другої світових воєн, коли обсяги переданої інформації зросли в рази, а вимоги до швидкості, безперервності та секретності шифрування стали критичними [3], [19], [24], [28]. Саме в цей період були створені роторні шифрувальні машини [14], [20], які реалізовували багатоетапні криптографічні перетворення та забезпечували динамічну зміну ключа під час обробки кожного символу повідомлення.

Роторні механізми дозволили реалізувати принцип багаторазового перетворення символів, при якому одна й та сама літера відкритого тексту могла шифруватися по-різному залежно від її позиції в повідомленні та поточного стану механізму [14], [20], [21]. Такий підхід суттєво ускладнював застосування класичних методів криптоаналізу, зокрема частотного аналізу, і став важливим кроком на шляху до створення поліалфавітних криптосистем. Фактично роторні машини заклали концептуальну основу для сучасних поточкових і блочних методів шифрування, у яких використовується послідовна зміна внутрішнього стану алгоритму [1], [6], [14], [20], [22].

Важливо зазначити, що механічні пристрої шифрування не лише виконували практичну функцію захисту інформації, а й сприяли формуванню теоретичних основ криптографії. Аналіз принципів їх роботи дозволив сформулювати ключові криптографічні поняття, зокрема роль ключа як параметра безпеки, необхідність оборотності перетворень, значення композиції операцій та залежність криптографічної стійкості системи від складності її математичної структури [4], [5]. Саме досвід використання механічних шифраторів показав, що безпека системи не може ґрунтуватися виключно на секретності її конструкції.

Таким чином, історичний розвиток механічних криптографічних пристроїв став важливим етапом еволюції криптографії [14], [20], [23]. Ці

системи створили фундамент для переходу від інтуїтивних і емпіричних методів шифрування до формалізованих математичних моделей, які використовуються у сучасних криптографічних алгоритмах та протоколах захисту інформації [1], [7], [24].

1.4 Математичні основи аналізу механічних пристроїв шифрування

З формальної точки зору механічні пристрої шифрування можуть бути розглянуті як системи, що реалізують відображення символів відкритого тексту у символи шифротексту відповідно до заданих правил. Такі правила визначаються конфігурацією механічних елементів пристрою та обраним ключем, який задає початковий стан шифрувальної системи. У більшості випадків ці перетворення мають бути оборотними, що забезпечує можливість відновлення початкового повідомлення при використанні правильних параметрів дешифрування [4], [5], [25].

Базовим елементом математичного опису механічних криптосистем є поняття перестановки символів. Кожен механічний компонент – диск, ротор або барабан – може розглядатися як окреме перетворення, яке змінює відповідність між символами відкритого та зашифрованого тексту [14], [20]. Сукупність таких компонентів утворює складне перетворення, властивості якого визначають криптографічну стійкість системи. Аналіз структури цих композицій дозволяє оцінити складність шифрування та можливі шляхи його розкриття [5], [7].

Особливістю роторних механічних пристроїв є наявність динамічного стану, який змінюється в процесі обробки повідомлення. Це означає, що результат шифрування окремого символу залежить не лише від ключа, а й від позиції символу в повідомленні та поточної конфігурації механізму [14], [20]. Такий підхід суттєво ускладнює криптоаналіз порівняно зі статичними підстановочними шифрами та дозволяє реалізувати поліалфавітний принцип шифрування [3], [6].

З формальної точки зору роботу механічних криптосистем можна розглядати як послідовність станів, кожен з яких відповідає певному положенню рухомих елементів пристрою. Перехід між станами відбувається за фіксованими правилами, що визначаються конструкцією механізму. Такий підхід дозволяє аналізувати не лише окремі перетворення, а й загальну динаміку системи, включаючи повторюваність конфігурацій і періодичність шифрування [4], [26].

Важливим аспектом аналізу механічних криптографічних систем є дослідження статистичних властивостей отриманого шифротексту. Багато механічних пристроїв, незважаючи на складність конструкції, не повністю усувають статистичні закономірності природної мови. Збереження частот символів, повторюваних фрагментів або структурних особливостей тексту створює передумови для застосування методів криптоаналізу, зокрема частотного та структурного аналізу [7].

Окрему групу становлять механічні криптосистеми, що працюють з групами символів, а не з окремими літерами. Такі системи дозволяють частково зменшити вплив статистичних властивостей мови, проте водночас характеризуються більш складною внутрішньою структурою. Аналіз подібних механізмів потребує формалізованого підходу та врахування взаємозв'язків між елементами перетворення [5], [27].

Таким чином, математичні основи аналізу механічних пристроїв шифрування на концептуальному рівні полягають у формалізації принципів підстановки, перестановки, композиції перетворень і динамічної зміни станів системи. Такий підхід дозволяє систематизувати знання про механічні криптосистеми, виявити їхні сильні та слабкі сторони та створює теоретичне підґрунтя для побудови строгих математичних моделей і програмної симуляції [1], [4], [6].

1.5 Проблеми та обмеження механічних криптографічних пристроїв

Незважаючи на значну історичну та наукову цінність, механічні криптографічні пристрої мають низку суттєвих обмежень [14], [20], які зумовили їх поступове витіснення електронними та програмними криптосистемами. Аналіз цих обмежень є важливим з точки зору розуміння еволюції криптографії та причин переходу до сучасних алгоритмічних методів захисту інформації [1], [2], [28].

Однією з основних проблем механічних криптосистем є обмежений простір ключів, що безпосередньо пов'язано з фізичними можливостями механізму. Кількість можливих конфігурацій дисків, роторів або барабанів є скінченною та, як правило, значно меншою порівняно з простором ключів сучасних криптографічних алгоритмів. Це створює передумови для реалізації атак повного перебору, особливо за наявності достатніх обчислювальних ресурсів або тривалого часу для криптоаналізу [4], [5].

Суттєвим обмеженням механічних пристроїв є також висока залежність від точності виготовлення та фізичного стану елементів. Зношування механічних деталей, люфти, деформації та похибки складання можуть призводити до відхилень у роботі пристрою, що, з одного боку, знижує надійність шифрування, а з іншого – створює побічні канали витоку інформації. Подібні ефекти неодноразово використовувалися у практиці криптоаналізу механічних шифрувальних машин [3], [6].

Ще однією проблемою є обмежена масштабованість механічних криптосистем. Зі зростанням обсягів переданої інформації та необхідності підвищення рівня криптографічної стійкості механічні пристрої вимагали ускладнення конструкції, що призводило до збільшення габаритів, маси та складності експлуатації. Це робило їх малопридатними для використання в умовах, де необхідна висока мобільність або швидка адаптація до змінних вимог безпеки [2], [7], [29].

З точки зору криптографічного аналізу, значним недоліком багатьох механічних шифрів є збереження статистичних закономірностей мови. Навіть у складних роторних системах повне руйнування частотних характеристик відкритого тексту досягалося не завжди [14], [20]. Наявність повторюваних структур, циклічності або обмеженого періоду роботи механізму створювала умови для застосування статистичних методів криптоаналізу, що підтверджено як історичними прикладами, так і теоретичними дослідженнями [5], [7], [17].

Окрему увагу слід приділити проблемі керування ключами у механічних криптосистемах. Процес зміни ключів часто вимагав фізичного переналаштовування пристрою, що було трудомістким, повільним і потенційно небезпечним з точки зору витоку інформації. У великих організаціях це ускладнювало централізоване управління криптографічними засобами та підвищувало ризик компрометації ключів [1], [4], [19].

Крім того, більшість механічних криптографічних систем не відповідали сучасному формулюванню принципу Керкгоффа [13], [16], згідно з яким, безпека криптосистеми має ґрунтуватися виключно на секретності ключа, а не на приховуванні алгоритму. У багатьох випадках стійкість механічних шифраторів значною мірою залежала від секретності їх конструкції, що знижувало рівень криптографічної надійності таких систем у довгостроковій перспективі [4], [6].

Попри зазначені обмеження, механічні криптографічні пристрої відіграли ключову роль у становленні науки шифрування даних. Їх аналіз дозволив сформулювати фундаментальні вимоги до криптографічних систем, зокрема необхідність великого простору ключів, динамічної зміни стану, мінімізації статистичних закономірностей та формалізації алгоритмів. Саме усвідомлення недоліків механічних систем стало рушійною силою переходу до електронних і програмних методів шифрування [1], [2], [7].

За результатами проведеного аналізу можна зробити висновок, що дослідження проблем і обмежень механічних криптографічних пристроїв має

не лише історичне значення. Воно створює підґрунтя для подальшого математичного моделювання, формалізації та програмної симуляції процесів шифрування.

Тому актуальною є розробка програмного забезпечення для симуляції механічних пристроїв шифрування.

При розробці програмного забезпечення симуляції механічних пристроїв шифрування необхідно забезпечити такі функціональні вимоги:

- забезпечення вибору типу механічного пристрою шифрування для симуляції;
- надання можливості налаштування параметрів конфігурації (ключа) обраного пристрою;
- підтримка введення, редагування та збереження відкритого тексту для обробки;
- реалізація процесів шифрування та дешифрування інформації відповідно до обраної моделі;
- відображення результатів шифрування та дешифрування у зрозумілій для користувача формі;
- забезпечення коректності роботи алгоритмів шляхом перевірки оборотності перетворень;
- надання можливості візуального відображення принципу роботи механічного пристрою;
- забезпечення збереження та завантаження налаштувань симуляції;
- формування звітів про результати проведених експериментів;
- забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу.

1.6 Висновки за розділом 1

Виконано аналіз предметної області механічних пристроїв шифрування даних. Розглянуто історичні передумови їх виникнення, основні класи та

принципи функціонування. Показано, що механічні криптосистеми реалізують базові криптографічні ідеї, які зберегли свою актуальність у сучасній криптографії [1], [6]. Отримані результати створюють теоретичне підґрунтя для побудови математичних моделей та програмної симуляції.

Сформульовано функціональні вимоги до програмного забезпечення симуляції моделей механічних пристроїв шифрування даних.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ

2.1 Формалізація процесу шифрування механічних криптосистем

Формалізація процесу шифрування механічних криптосистем створює основу для побудови математичних моделей, які можуть бути використані як для теоретичного аналізу, так і для подальшої програмної симуляції процесів шифрування даних [1], [8].

У загальному випадку процес шифрування розглядається як перетворення деякого повідомлення з відкритого вигляду у зашифрований відповідно до певних правил, що визначаються ключем [22], [26], [27]. Для механічних криптосистем такими правилами є конфігурація рухомих елементів пристрою, їх початкові положення та алгоритм взаємодії між ними. На відміну від сучасних електронних алгоритмів, ці правила реалізуються фізично, проте з формальної точки зору вони можуть бути описані як абстрактний алгоритм [4], [9].

Формалізація шифрування передбачає введення низки базових понять, зокрема алфавіту, повідомлення, ключа та операції шифрування. Алфавіт визначає множину символів, з якими працює криптосистема, а повідомлення розглядається як послідовність елементів цього алфавіту. Ключ у механічних системах не є числовим параметром у класичному розумінні, а виступає у вигляді сукупності налаштувань пристрою, що визначають його поведінку під час шифрування [5], [10].

Важливою властивістю будь-якої криптографічної системи є оборотність перетворень. Це означає, що для кожного зашифрованого повідомлення за умови використання правильного ключа має існувати однозначний спосіб відновлення початкового тексту. [25], [27], [29] У механічних криптосистемах оборотність досягається завдяки симетричності конструкції або можливості виконання зворотних операцій, що безпосередньо впливає на побудову формальної моделі [4], [11].

Процес шифрування в механічних пристроях можна розглядати як послідовність елементарних перетворень, кожне з яких відповідає окремому механічному елементу або їх групі. Такі перетворення можуть виконуватися послідовно або комбіновано, утворюючи складну структуру, яка визначає криптографічні властивості системи. Формалізований опис цієї структури дозволяє аналізувати її незалежно від конкретної конструкції пристрою [8], [12].

Особливістю механічних криптосистем є те, що результат шифрування залежить не лише від ключа, а й від поточного стану пристрою, який змінюється в процесі роботи. Це означає, що формальна модель має враховувати часову або позиційну залежність перетворень. Такий підхід зближує механічні шифратори з динамічними криптографічними системами та створює передумови для використання моделей зі змінним внутрішнім станом [6], [9].

Формалізація процесу шифрування також є необхідною умовою для подальшого аналізу криптографічної стійкості механічних систем [12], [21], [23]. Саме абстрактний опис дозволяє виявити закономірності, повторюваність або структурні обмеження, які не завжди очевидні при розгляді фізичної реалізації.

Таким чином, формалізація процесу шифрування механічних криптосистем створює основу для побудови математичних моделей, які дозволяють описати принципи їх роботи в узагальненій формі. Це, у свою чергу, є необхідним кроком для розроблення програмної симуляції механічних пристроїв шифрування.

2.2 Моделі підстановочних і перестановочних механізмів шифрування

Одними з найперших і водночас найпростіших механічних криптографічних систем є пристрої, що реалізують підстановочні та

перестановочні перетворення. Незважаючи на відносну простоту, саме ці механізми заклали фундамент для подальшого розвитку складніших криптографічних моделей і стали базою для формування формалізованих підходів до опису процесу шифрування [1], [8].

Підстановочні механічні шифри ґрунтуються на принципі заміни символів відкритого тексту іншими символами відповідно до наперед визначеного правила. У механічній реалізації таке правило задається фізичною конфігурацією пристрою – наприклад, взаємним розташуванням символів на дисках або контактами між елементами. Класичним прикладом є криптодиск Альберті, у якому два диски з алфавітами дозволяють здійснювати заміну символів шляхом їх взаємного повороту [3], [9].

Особливістю механічної підстановки є її наочність та детермінованість: за фіксованого положення елементів кожному символу відкритого тексту відповідає єдиний символ шифротексту. З точки зору моделювання це дозволяє розглядати підстановочний механізм як статичне перетворення, яке не змінюється в процесі шифрування. Саме ця властивість робить такі системи вразливими до криптоаналізу, зокрема до частотного аналізу [5], [10].

Перестановочні механізми шифрування реалізують інший принцип – зміну порядку символів у повідомленні без зміни самих символів. У механічних пристроях це досягається за рахунок маршрутів проходження тексту, використання решіток або спеціальних шаблонів. Прикладом є решітка Кардано, де шифрування відбувається шляхом накладання шаблону на текст і зчитування символів у визначеному порядку [8], [11].

На відміну від підстановочних шифрів, перестановочні механізми не порушують частотного розподілу символів, однак значно ускладнюють сприйняття структури повідомлення. У механічній реалізації такі системи часто комбінуються з підстановочними, оскільки самотійно вони не забезпечують достатнього рівня криптографічної стійкості. Це призвело до появи комбінованих механічних пристроїв, які реалізують послідовність підстановок і перестановок [1], [6].

Важливою характеристикою як підстановочних, так і перестановочних механізмів є залежність їх властивостей від фізичної реалізації. Обмежена кількість можливих положень дисків або шаблонів безпосередньо впливає на кількість допустимих ключів і, відповідно, на рівень криптографічної стійкості системи. Саме ця особливість зумовила необхідність переходу від статичних механізмів до динамічних систем із змінним станом [7], [12].

Підстановочні та перестановочні механізми є зручними моделями для вивчення базових криптографічних властивостей. Вони дозволяють наочно простежити вплив конфігурації ключа на результат шифрування, дослідити оборотність перетворень і виявити закономірності, які можуть бути використані для криптоаналізу. Саме на прикладі таких механізмів уперше було усвідомлено, що складність конструкції не гарантує криптографічної стійкості без належного математичного обґрунтування [4], [10].

2.3 Модель роторних механічних пристроїв

Роторні механічні пристрої шифрування становлять найбільш складний клас механічних криптографічних систем.

Основною особливістю роторних систем є наявність рухомих елементів, кожен з яких реалізує окреме криптографічне перетворення. Після обробки кожного символу положення принаймні одного ротора змінюється, що призводить до зміни відповідності між символами відкритого тексту та шифротексту.

Процес шифрування у роторному механізмі можна подати як послідовне застосування кількох підстановочних перетворень.

Нехай відкритий текст представлений у вигляді послідовності символів за формулою (2.1):

$$P = (p_1, p_2, \dots, p_n). \quad (2.1)$$

Тоді результат шифрування одного символу визначається композицією перетворень, які відповідають поточному стану роторів за формулою (2.2):

$$C_i = E(p_i, S_i), \quad (2.2)$$

де p_i – i -й символ відкритого тексту;

C_i – відповідний символ шифротексту;

S_i – стан роторного механізму під час обробки i -го символу.

Стан роторної системи змінюється після кожного кроку шифрування відповідно до правил обертання роторів. Таким чином, для двох однакових символів відкритого тексту, що знаходяться в різних позиціях повідомлення, результат шифрування, як правило, буде різним (2.3):

$$E(p_i, S_i) \neq E(p_j, S_j), \text{ якщо } i \neq j. \quad (2.3)$$

Ця властивість є принциповою відмінністю роторних систем від моноалфавітних шифрів і забезпечує реалізацію поліалфавітного принципу шифрування [5], [6].

Кожен окремий ротор може бути описаний як фіксоване відображення алфавіту, однак у процесі роботи пристрою це відображення циклічно зміщується. Якщо позначити початкове налаштування ротора як K , а поточний зсув як Δ_i , то перетворення, що реалізується ротором, залежить від обох параметрів (2,4):

$$R_i = R(K, \Delta_i). \quad (2.4)$$

Композиція кількох роторів утворює загальне шифрувальне перетворення (2.5):

$$E = R_1 \circ R_2 \circ \dots \circ R_k, \quad (2.5)$$

де k – кількість роторів у системі.

Процес дешифрування у роторних механічних системах можливий за умови точної відповідності початкових налаштувань пристрою. Дешифрування реалізується шляхом застосування обернених перетворень у зворотному порядку (2.6):

$$p_i = D(C_i, S_i). \quad (2.6)$$

Таким чином, коректність роботи роторного механізму безпосередньо залежить від синхронізації станів пристрою під час шифрування та дешифрування.

Роторні криптографічні пристрої можна розглядати як системи з великим, але скінченним простором станів. Кожна унікальна конфігурація положень роторів відповідає окремому стану системи, а загальна кількість таких станів визначається числом роторів і кількістю можливих позицій кожного з них [8], [9].

Саме динамічна зміна стану та велика кількість можливих конфігурацій забезпечили практичну стійкість роторних шифрів протягом певного історичного періоду. Водночас обмеженість простору станів і наявність регулярностей у механізмі обертання стали причинами їх подальшого успішного криптоаналізу [3], [6], [10].

2.4 Моделювання динаміки зміни станів механічних криптографічних систем

Однією з ключових особливостей механічних криптографічних пристроїв є наявність динамічного внутрішнього стану, який змінюється у процесі шифрування кожного символу повідомлення. На відміну від статичних підстановочних систем, де криптографічне перетворення залишається незмінним протягом усього процесу шифрування, механічні

роторні пристрої реалізують принцип змінної конфігурації, що суттєво ускладнює аналіз отриманого шифротексту [6], [8].

Під станом механічної криптосистеми розуміють сукупність параметрів, які визначають поточне положення всіх рухомих елементів пристрою. Для роторних механізмів такими параметрами є позиції роторів, їхній порядок, а також взаємодія між ними. Зміна стану відбувається за фіксованими правилами, закладеними в конструкцію пристрою, і, як правило, залежить від кількості вже оброблених символів повідомлення [3], [9].

Динаміка зміни станів у механічних криптосистемах має дискретний характер. Після обробки кожного символу відбувається переміщення одного або кількох механічних елементів, що призводить до переходу системи в нову конфігурацію. У простіших моделях змінюється лише положення одного ротора, тоді як у складніших системах реалізується каскадний механізм, за якого обертання одного елемента викликає зміну положення інших. Такий підхід забезпечує значну варіативність перетворень навіть за фіксованого ключа [6], [10].

Важливою характеристикою динаміки механічних систем є періодичність зміни станів. Оскільки кількість можливих конфігурацій механічного пристрою є скінченною, рано чи пізно система повертається до початкового стану, утворюючи цикл. Довжина такого циклу залежить від конструктивних особливостей пристрою та кількості рухомих елементів. Короткі цикли є небажаними, оскільки вони призводять до повторюваності криптографічних перетворень і можуть бути використані для криптоаналізу [7], [11].

З точки зору моделювання динаміки, механічні криптосистеми можна розглядати як системи з пам'яттю, у яких результат обробки поточного символу залежить не лише від вхідних даних і ключа, а й від усієї попередньої історії роботи пристрою. Такий підхід дозволяє формалізувати поведінку механічних шифраторів і підкреслює їх концептуальну спорідненість із сучасними потоковими алгоритмами шифрування [1], [8].

При моделюванні динаміки зміни станів важливо також враховувати обмеження механічної реалізації, зокрема неможливість реалізації довільних або випадкових переходів між станами. Усі зміни визначаються жорстко заданими правилами, що, з одного боку, забезпечує відтворюваність процесу шифрування, а з іншого – обмежує криптографічну стійкість системи. Саме детермінований характер зміни станів у поєднанні зі скінченністю простору конфігурацій став однією з причин успішного криптоаналізу багатьох роторних машин [9], [11].

Для цілей програмної симуляції механічних криптографічних пристроїв моделювання динаміки зміни станів має принципове значення. Коректне відтворення послідовності переходів між конфігураціями дозволяє не лише імітувати роботу реальних механізмів, а й проводити експериментальні дослідження, спрямовані на аналіз повторюваності, періодичності та чутливості системи до початкових параметрів. Це створює основу для подальшого аналізу криптографічних властивостей і розробки навчальних та дослідницьких програмних засобів [1], [10], [12].

Таким чином, моделювання динаміки зміни станів механічних криптографічних систем є необхідним етапом формалізації їх роботи. Воно дозволяє узагальнити принципи функціонування механічних шифраторів, виявити їхні обмеження та підготувати ґрунт для реалізації відповідних алгоритмів у програмному середовищі.

2.5 Аналіз властивостей математичних моделей механічних криптосистем

Аналіз властивостей математичних моделей механічних криптосистем дозволяє оцінити їх придатність для забезпечення криптографічної безпеки та виявити причини їх обмеженої стійкості. Основними характеристиками таких моделей є оборотність, періодичність та наявність слабких місць, що можуть бути використані для криптоаналізу [4], [7].

Оборотність є базовою вимогою до будь-якої криптографічної системи, оскільки без неї неможливе коректне відновлення відкритого тексту. У механічних криптосистемах оборотність досягається шляхом суворо визначених правил зміни станів. Порушення цих умов призводить до втрати інформації або неоднозначності дешифрування [5], [9].

Періодичність механічних систем зумовлена скінченністю простору станів. Навіть у складних роторних пристроях кількість можливих конфігурацій обмежена, що неминуче призводить до повторення станів. За умови наявності достатньо довгих повідомлень це створює передумови для статистичного аналізу та відновлення структури шифрування [7], [11].

До основних слабких місць механічних криптосистем належать обмежений простір ключів, регулярність зміни станів і часткове збереження статистичних характеристик мови. Крім того, фізична реалізація механізмів часто призводила до виникнення побічних ефектів, таких як нерівномірність обертання або залежність роботи пристрою від зовнішніх умов, що додатково знижувало рівень безпеки [3], [6], [8].

Таким чином, аналіз властивостей математичних моделей механічних криптосистем демонструє, що їх криптографічна стійкість є обмеженою з позицій сучасної теорії. Водночас саме ці обмеження роблять механічні моделі цінним об'єктом дослідження, оскільки дозволяють наочно простежити взаємозв'язок між структурою шифру, динамікою його станів та можливостями криптоаналізу. Отримані результати є підґрунтям для побудови програмної симуляції та проведення експериментальних досліджень.

2.6 Висновки до розділу 2

Виконано формалізований аналіз механічних пристроїв шифрування даних.

Розглянуто підходи до формалізації механічних криптосистем, зокрема моделі підстановочних і перестановочних механізмів, а також особливості

роторних пристроїв, що характеризуються динамічною зміною внутрішнього стану. Наявність змінного стану дозволяє реалізувати поліалфавітний принцип шифрування та ускладнює застосування простих криптоаналітичних методів.

Окрему увагу приділено моделюванню динаміки зміни станів механічних систем. Доведено, що розмежування понять ключа та поточного стану є критично важливим для аналізу криптографічної стійкості. Встановлено, що обмежений простір станів та їх періодичність є одним з основних чинників зниження криптографічної надійності механічних криптосистем.

Проведений аналіз дозволив виявити типові слабкі місця механічних пристроїв шифрування, зокрема лінійність перетворень, статистичні закономірності та структурну регулярність. Водночас аналіз показав, що саме дослідження цих недоліків сприяло формуванню принципів сучасної криптографії, таких як вимога великого простору ключів, нелінійність перетворень і мінімізація статистичних залежностей.

Отримані результати створюють основу для подальшої програмної реалізації симуляції механічних пристроїв шифрування.

3 МАТЕРІАЛИ І МЕТОДИ

3.1 Вибір мови програмування

Одним із ключових етапів розробки програмного забезпечення для симуляції механічних пристроїв шифрування є обґрунтований вибір мови програмування. Від цього вибору залежить не лише коректність реалізації математичних і логічних моделей, а й зручність розробки, можливості візуалізації, розширюваність програмного продукту та його придатність до використання в навчальному процесі.

Програмне забезпечення, що розробляється в межах даної кваліфікаційної роботи, має реалізовувати симуляцію роботи механічних криптографічних пристроїв, забезпечувати наочне відображення процесів шифрування та дешифрування, а також дозволяти проводити експериментальні дослідження з різними параметрами конфігурації. Це накладає низку вимог до мови програмування, зокрема підтримку модульної архітектури, засобів роботи з графічним інтерфейсом, достатню продуктивність та доступність бібліотек для реалізації алгоритмів обробки даних [8], [9].

Серед можливих варіантів було розглянуто такі мови програмування, як C++, Java, Python та C#.

Мова C++ забезпечує високу продуктивність і широкі можливості контролю ресурсів, однак потребує значних зусиль для реалізації графічного інтерфейсу та супроводжується підвищеною складністю розроблення.

Java є платформонезалежною та має розвинені бібліотеки, проте її використання у навчальних симуляторах часто ускладнює інтеграцію з сучасними засобами візуалізації та швидке прототипування [9].

Мова програмування Python вирізняється простотою синтаксису, наявністю великої кількості бібліотек для моделювання, візуалізації та аналізу даних, а також широким використанням у наукових і навчальних проєктах. Завдяки цьому Python часто застосовується для реалізації криптографічних

прототипів, навчальних симуляторів і дослідницьких програмних засобів [10]. Крім того, підтримка об'єктно-орієнтованої парадигми програмування дозволяє відобразити структуру механічних пристроїв шифрування у вигляді взаємодіючих програмних компонентів.

Важливим аргументом на користь Python є наявність бібліотек для створення графічних інтерфейсів та візуалізації процесів, що є критично важливим для наочного відображення принципів роботи механічних криптографічних систем [8], [11].

З урахуванням зазначених факторів для реалізації програмного забезпечення симуляції механічних пристроїв шифрування обрано мову програмування Python. Вона забезпечує оптимальний баланс між простотою реалізації, гнучкістю, наочністю та функціональними можливостями, необхідними для побудови програмної симуляції та проведення експериментальних досліджень.

Обґрунтування вибору мови програмування для реалізації програмного забезпечення симуляції механічних пристроїв шифрування даних наведено у таблиці 3.1.

Таблиця 3.1 – Обґрунтування вибору мови програмування для реалізації програмного забезпечення симуляції механічних пристроїв шифрування даних

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Ruby
1	2	3	4
Ефективне використання пам'яті	+	+—	+—
Компіляція програмного коду	+	—	—
Висока швидкість функціонування розроблюваного програмного забезпечення	+	+—	+—

Продовження таблиці 3.1

1	2	3	4
Зручність реалізації алгоритмічних і модельних обчислень	+—	+	+—
Наявність бібліотек для математичного моделювання	+—	+	—
Зручність створення навчального та дослідницького програмного забезпечення	+—	+	+—
Простота реалізації симуляційних моделей	+—	+	+—
Кросплатформеність	+	+	+—

3.2 Вибір середовища розробки для створення програмного забезпечення симуляції механічних пристроїв шифрування даних

Розробка програмного забезпечення симуляції механічних пристроїв шифрування потребує використання сучасного, гнучкого та зручного середовища розробки, яке забезпечує ефективну роботу з програмним кодом, підтримку обраної мови програмування, а також можливість розширення функціональності за рахунок додаткових інструментів.

Вибір середовища розробки є важливим етапом проєктування програмної системи.

Для реалізації програмного забезпечення симуляції механічних пристроїв шифрування у даній кваліфікаційній роботі було обрано Visual Studio Code – сучасне кросплатформенне середовище розробки, яке активно використовується як у навчальних, так і в промислових програмних проєктах [8]. Visual Studio Code поєднує в собі функціональність інтегрованого середовища розробки та легкість текстового редактора, що робить його зручним для створення програм різного рівня складності.

Однією з ключових переваг Visual Studio Code є повноцінна підтримка мови програмування Python, яка була обрана для реалізації симуляції механічних пристроїв шифрування. Середовище забезпечує інтеграцію з офіційним розширенням Python, що надає засоби підсвічування синтаксису, автодоповнення коду, статичного аналізу, налагодження та запуску програм безпосередньо з інтерфейсу редактора [9]. Це значно спрощує процес розробки та зменшує ймовірність синтаксичних і логічних помилок.

Visual Studio Code є безкоштовним середовищем розробки з відкритим вихідним кодом, що робить його доступним для широкого кола користувачів. Важливою характеристикою Visual Studio Code є підтримка модульної структури програмного коду. Це є особливо актуальним для розроблюваної симуляції, оскільки програмна система складається з окремих логічних компонентів: реалізації моделей механічних шифраторів, обробки ключів, керування станами, введення та виведення даних. Засоби навігації по проєкту та швидкого переходу між модулями сприяють підвищенню продуктивності розробки та спрощують супровід коду [10].

Ще однією суттєвою перевагою Visual Studio Code є кросплатформенність. Середовище підтримується операційними системами Windows, Linux та macOS, що дозволяє розробляти та тестувати програмне забезпечення незалежно від апаратної платформи. Це відповідає вимогам до переносимості програмного забезпечення та забезпечує можливість подальшого використання симуляції в різних навчальних і дослідницьких середовищах [8].

Visual Studio Code також надає широкі можливості для налагодження програмного забезпечення. Вбудований дебагер дозволяє виконувати покрокове виконання коду, відстежувати значення змінних і аналізувати поведінку програми під час симуляції процесу шифрування. Це є важливим інструментом при перевірці коректності реалізації моделей механічних криптосистем та дослідженні динаміки зміни їх внутрішніх станів [9].

Окремої уваги заслуговує підтримка систем керування версіями, зокрема Git, яка інтегрована у Visual Studio Code. Це дозволяє вести контроль змін програмного коду, експериментувати з різними реалізаціями алгоритмів симуляції та забезпечує відтворюваність результатів досліджень [10].

Завдяки широкому набору доступних розширень Visual Studio Code може бути легко адаптований під потреби конкретного проєкту. Для розроблення програмного забезпечення симуляції механічних пристроїв шифрування це дозволяє додатково використовувати інструменти для роботи з графікою, побудови діаграм, аналізу даних та формування звітів. Таким чином, середовище розробки не обмежує функціональні можливості програмної реалізації та сприяє подальшому розвитку проєкту.

Отже, вибір Visual Studio Code як середовища розробки є обґрунтованим з огляду на його функціональні можливості, зручність використання, підтримку мови Python та відповідність вимогам до програмного забезпечення симуляції механічних пристроїв шифрування даних.

Обґрунтування вибору середовища розробки для створення програмного забезпечення для симуляції механічних пристроїв шифрування даних наведено у таблиці 3.2.

Таблиця 3.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення симуляції механічних пристроїв шифрування даних

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio Code	PyCharm	Spyder
1	2	3	4
Можливості розширення та плагіни	+	+	+—
Повна підтримка мови Python	+	+	+

Продовження таблиці 3.2

1	2	3	4
Підтримка віртуальних середовищ	+	+	+
Зручність розробки наукових та навчальних проєктів	+—	+	+
Підтримка відлагодження	+	+	+—
Інтеграція з системами контролю версій (Git)	+	+	—
Кросплатформеність	+	+	+
Активна спільнота та документація	+	+	+—
Невисокі апаратні вимоги	+	+—	+
Відкритість та безкоштовність	+	+—	+

Таким чином, для створення програмного забезпечення симуляції механічних пристроїв шифрування даних обрано середовище розробки Visual Studio Code, яке забезпечує кросплатформенність, підтримку мови програмування Python, розвинену систему розширень, зручні засоби налагодження та інтеграцію з сучасними інструментами розробки, що робить його доцільним для реалізації поставлених завдань дослідження.

3.3 Висновки за розділом 3

Обґрунтовано вибір інструментальних засобів для розробки програмного забезпечення симуляції механічних пристроїв шифрування даних.

За результатами порівняльного аналізу сучасних мов програмування для реалізації програмної симуляції обрано мову Python. Такий вибір

зумовлений її універсальністю, простотою синтаксису, наявністю широкого спектра бібліотек для моделювання, обробки даних і візуалізації, а також активною спільнотою розробників. Використання Python дозволяє ефективно реалізувати алгоритми симуляції, забезпечити наочність роботи програмного забезпечення та спростити подальше розширення функціональних можливостей системи.

Для реалізації програмного забезпечення, як середовище розробки обрано Visual Studio Code, що обґрунтовано його кросплатформенністю, гнучкістю налаштувань та широкими можливостями розширення. Підтримка мови Python, інтеграція з системами контролю версій, налагоджувачами та зовнішніми інструментами аналізу коду роблять Visual Studio Code зручним і ефективним середовищем для розробки прикладного програмного забезпечення дослідницького та навчального призначення.

4 ОПИС ПРОГРАМИ

4.1 Структура програмного забезпечення симуляції механічних пристроїв шифрування даних

Структуру програмного забезпечення симуляції роботи механічного шифрувального пристрою у вигляді діаграми класів наведено на рис. 4.1.

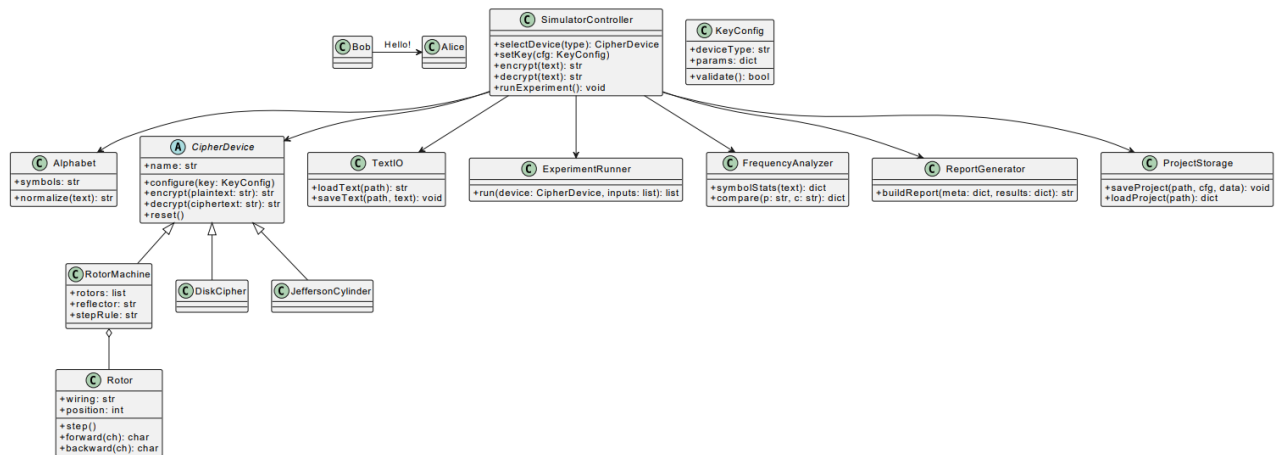


Рисунок 4.1 – Структура програмного забезпечення симуляції механічних пристроїв шифрування даних

Як видно з рис. 4.1, програмне забезпечення симуляції механічних пристроїв шифрування побудоване за модульним принципом із використанням об'єктно-орієнтованого підходу. Така архітектура забезпечує гнучкість системи, можливість розширення функціональності та повторного використання окремих компонентів.

Центральним елементом архітектури є клас `SimulatorController`, який виконує роль керуючого компонента системи. Він відповідає за координацію роботи інших модулів, вибір типу механічного пристрою шифрування, встановлення параметрів ключа, запуск процесів шифрування, дешифрування та експериментального аналізу. Саме через цей клас реалізується основна логіка взаємодії користувача з програмним забезпеченням.

Для зберігання та перевірки параметрів ключа використовується клас `KeyConfig`. Даний клас інкапсулює інформацію про тип пристрою та його

параметри, а також забезпечує перевірку коректності конфігурації перед запуском симуляції. Це дозволяє уникнути помилок, пов'язаних із некоректним налаштуванням механічного шифру.

Абстрактний клас `CipherDevice` є базовим для всіх механічних пристроїв шифрування, що реалізуються у програмі. Він визначає спільний інтерфейс для конфігурації пристрою, виконання операцій шифрування та дешифрування, а також скидання внутрішнього стану. Використання абстрактного класу дозволяє уніфікувати роботу з різними типами механічних криптосистем.

На основі класу `CipherDevice` реалізовано конкретні моделі механічних пристроїв. Клас `DiskCipher` моделює дискові шифрувальні механізми, у яких криптографічне перетворення здійснюється за рахунок обертання шифрувальних дисків. Клас `JeffersonCylinder` реалізує принцип роботи циліндра Джефферсона, що є прикладом поліалфавітного механічного шифру. Клас `RotorMachine` представляє роторні шифрувальні машини та містить параметри, які визначають набір роторів, відбивач та правила крокування.

Клас `Rotor` є допоміжним компонентом для моделювання роторних механізмів. Він описує окремий ротор, включаючи його внутрішню схему з'єднань, поточну позицію та методи прямого і зворотного проходження символів. Наявність окремого класу для ротора забезпечує деталізовану та наочну реалізацію роторних систем.

Для роботи з текстовими даними використовується клас `TextIO`, який відповідає за завантаження та збереження текстів із файлів. Це дозволяє працювати з великими обсягами даних та використовувати зовнішні текстові файли у межах експериментів.

Клас `Alphabet` забезпечує керування алфавітом, що використовується під час шифрування. Він виконує нормалізацію тексту відповідно до заданого набору символів та усуває некоректні або невідомі символи, що є важливим для коректної роботи механічних криптосистем.

Для проведення серійних експериментів передбачений клас ExperimentRunner, який автоматизує запуск симуляцій для заданого набору вхідних даних. Це дозволяє досліджувати властивості механічних шифрів у різних конфігураціях без ручного втручання.

Аналіз статистичних властивостей шифротексту здійснюється за допомогою класу FrequencyAnalyzer, який підраховує частоти символів та порівнює розподіли відкритого тексту і результатів шифрування. Такий аналіз є важливим інструментом для оцінки криптографічної стійкості механічних моделей.

Клас ReportGenerator відповідає за формування звітів про результати проведених експериментів. Він збирає метадані, параметри конфігурації та результати аналізу, після чого формує структурований текстовий звіт для подальшого використання у навчальних або наукових цілях.

Збереження та завантаження стану проєкту реалізується за допомогою класу ProjectStorage, який дозволяє відновлювати конфігурації, результати симуляцій та експериментальні дані. Це забезпечує повторюваність досліджень і зручність роботи з програмним забезпеченням.

Таким чином, запропонована структура програмного забезпечення є логічно впорядкованою, модульною та орієнтованою на розширення.

На рис. 4.2 зображено діаграму компонентів програмного забезпечення симуляції механічних пристроїв шифрування даних.

Діаграма компонентів відображає архітектуру програмного забезпечення симуляції механічних пристроїв шифрування та демонструє взаємодію між основними функціональними підсистемами. Запропонована архітектура реалізує модульний підхід, що забезпечує розділення відповідальностей між компонентами, підвищує гнучкість системи та спрощує її розширення й супровід.

Центральним елементом діаграми є компонент UI/Controller, який виконує роль координатора взаємодії між користувачем та внутрішніми підсистемами програми. Даний компонент відповідає за обробку дій

користувача, ініціацію експериментів, формування запитів до інших компонентів та відображення результатів роботи системи.

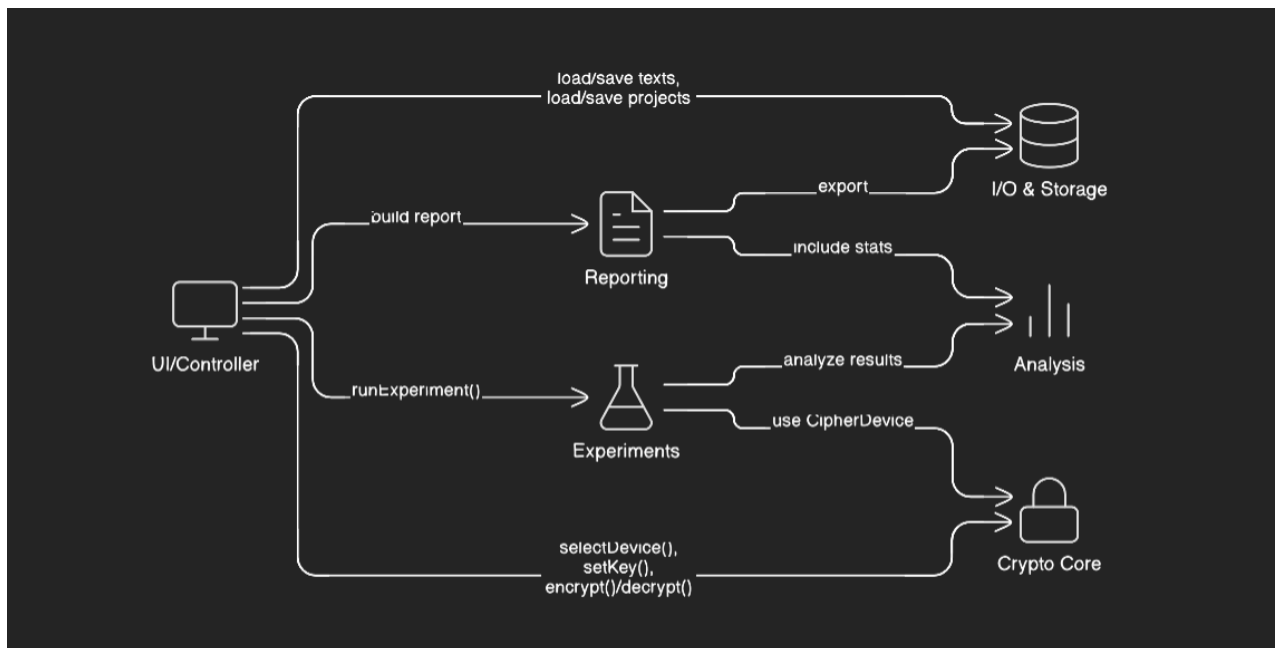


Рисунок 4.2 – Діаграма компонентів вебзастосунку Інтернет аукціону

Компонент Experiments реалізує функціональність проведення криптографічних експериментів. Він отримує керування від компонента UI/Controller через виклик операції runExperiment(). У межах своєї відповідальності цей компонент використовує конкретні реалізації механічних пристроїв шифрування (CipherDevice) для виконання операцій шифрування та дешифрування. Результати експериментів передаються далі для аналізу та збереження.

Компонент Analysis призначений для обробки та аналізу результатів експериментів. Він отримує дані від компонента Experiments і виконує статистичні та структурні дослідження зашифрованих текстів, зокрема аналіз частот символів, порівняння відкритого тексту та шифротексту, а також виявлення характерних закономірностей. Отримані результати аналізу використовуються як для формування звітів, так і для подальших дослідницьких висновків.

Компонент Reporting відповідає за формування підсумкових звітів за результатами проведених експериментів. За запитом від UI/Controller він здійснює побудову звіту (build report), використовуючи як результати експериментів, так і статистичні дані, отримані від компонента Analysis (include stats). Сформовані звіти можуть бути експортовані у зовнішні формати для подальшого використання.

Компонент I/O & Storage забезпечує взаємодію системи з файловою підсистемою. Він відповідає за завантаження та збереження текстів, конфігурацій і проєктів (load/save texts, load/save projects), а також за експорт сформованих звітів (export). Даний компонент ізольований від логіки криптографічних обчислень, що відповідає принципам чистої архітектури та спрощує зміну способів збереження даних.

Взаємодія між компонентами організована через чітко визначені інтерфейси, що мінімізує залежності між ними. UI/Controller виступає єдиною точкою входу для користувача та не містить реалізації криптографічних алгоритмів або аналічних процедур, а лише керує потоком виконання між компонентами.

Таким чином, представлена діаграма компонентів відображає логічно структуровану, масштабовану та розширювану архітектуру програмного забезпечення. Поділ системи на компоненти UI/Controller, Experiments, Analysis, Reporting та I/O & Storage дозволяє реалізувати симуляцію механічних пристроїв шифрування придатний для подальшого розвитку та інтеграції нових функціональних можливостей.

4.2 Функціонування програмного забезпечення симулятора

Функціонування програмного забезпечення симулятора подано за допомогою діаграми послідовностей, зображеної на рис. 4.3.

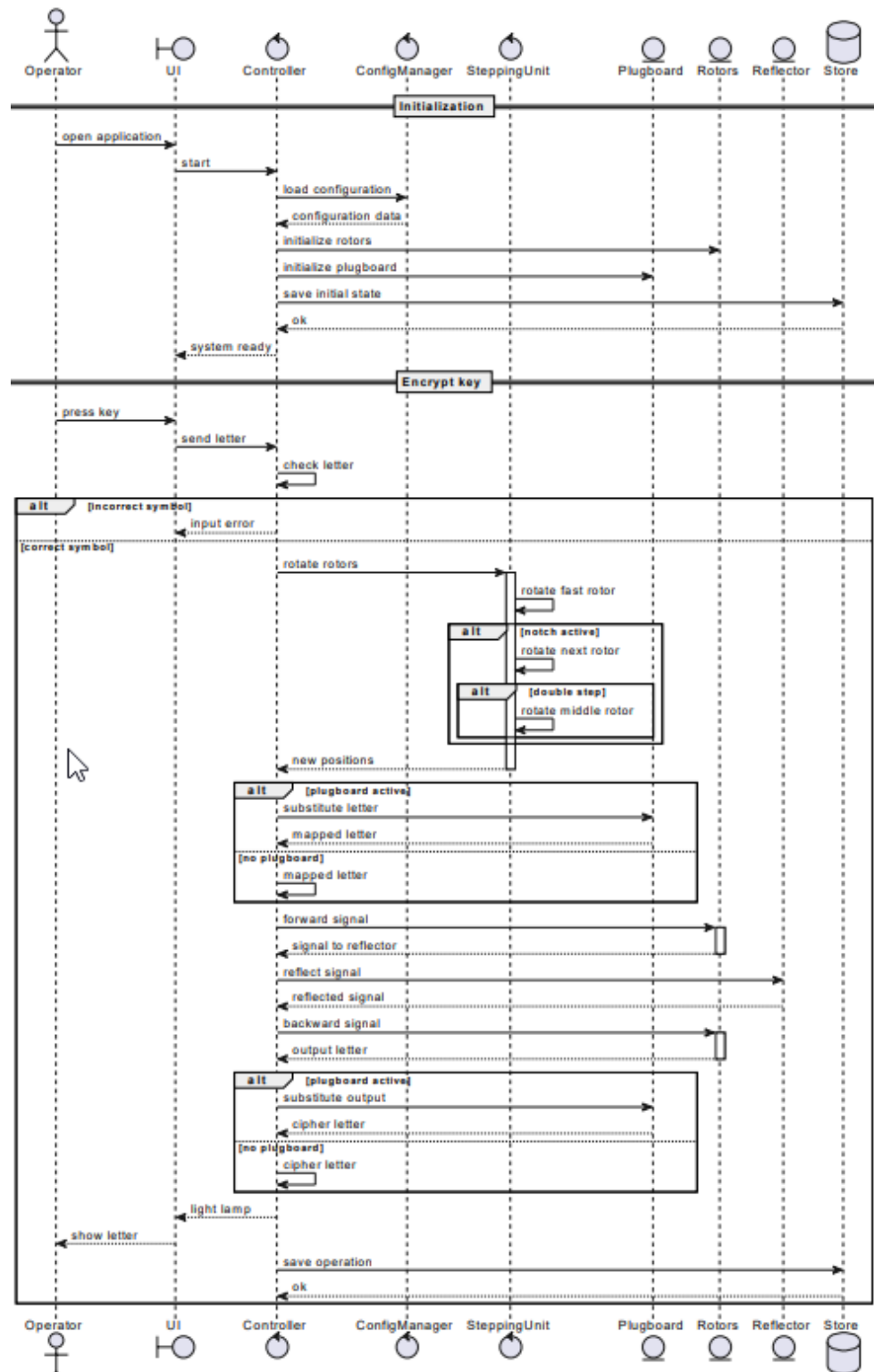


Рисунок 4.3 – Схема функціонування програмного забезпечення симулятора шифрування даних у вигляді діаграми послідовностей

На діаграмі послідовностей наведено процес взаємодії між оператором (користувачем) та основними компонентами програмного забезпечення симулятора роторної шифрувальної машини: інтерфейсом користувача

(клавіатура/лампова панель), контролером (керуючим модулем), модулями крокування ротора(ів), плагбордом (за наявності), пакетом роторів, відбивачем, а також сховищем стану/журналом.

У межах сценарію показано, як після натискання клавіші:

- виконується перевірка введення;
- запускається механізм крокування (з урахуванням notch і можливого double-step);
- сигнал проходить через плагборт (опційно), ротори у прямому напрямі, відбивач і ротори у зворотному напрямі;
- формується вихідний символ, який відображається на “лампах”;
- результат і поточні позиції роторів фіксуються в журналі.

4.3 Проектування інтерфейсу програмного забезпечення для симулятора механічної шифрувальної машини

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для симулятора механічної шифрувальної машини виконано з урахуванням вимог технічного завдання. При розробці програмного забезпечення для механічної шифрувальної машини враховані функціональні вимоги до програми:

- забезпечення вибору типу механічного пристрою шифрування для симуляції;
- надання можливості налаштування параметрів конфігурації (ключа) обраного пристрою;
- підтримка введення, редагування та збереження відкритого тексту для обробки;
- реалізація процесів шифрування та дешифрування інформації відповідно до обраної моделі;
- відображення результатів шифрування та дешифрування у зрозумілій для користувача формі;

- забезпечення коректності роботи алгоритмів шляхом перевірки оборотності перетворень;
- надання можливості візуального відображення принципу роботи механічного пристрою;
- забезпечення збереження та завантаження налаштувань симуляції;
- формування звітів про результати проведених експериментів;
- забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу.

Інтерфейс головної форми програмного забезпечення «Симулятор механічної шифрувальної машини» наведено на рис. 4.4.

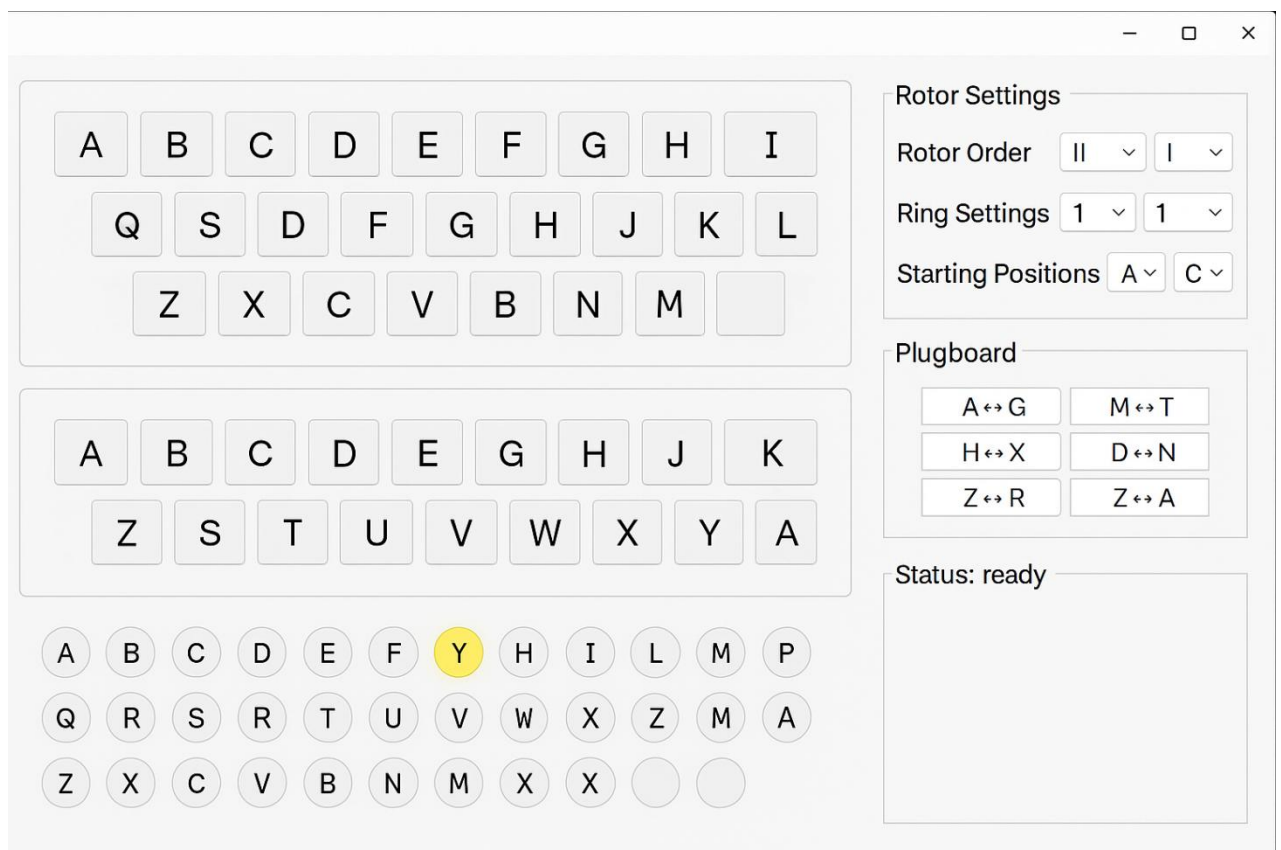
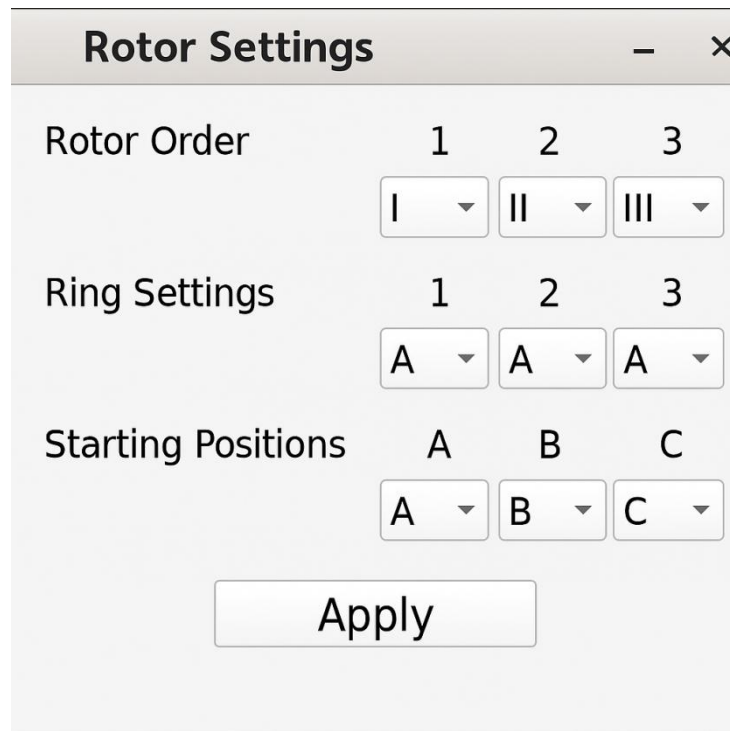


Рисунок 4.4 – Інтерфейс головної форми програмного забезпечення
Симулятор механічної шифрувальної машини

Форму налаштування роторів наведено на рис. 4.5.



Rotor Settings [minimize] [close]

Rotor Order	1	2	3
	I ▾	II ▾	III ▾

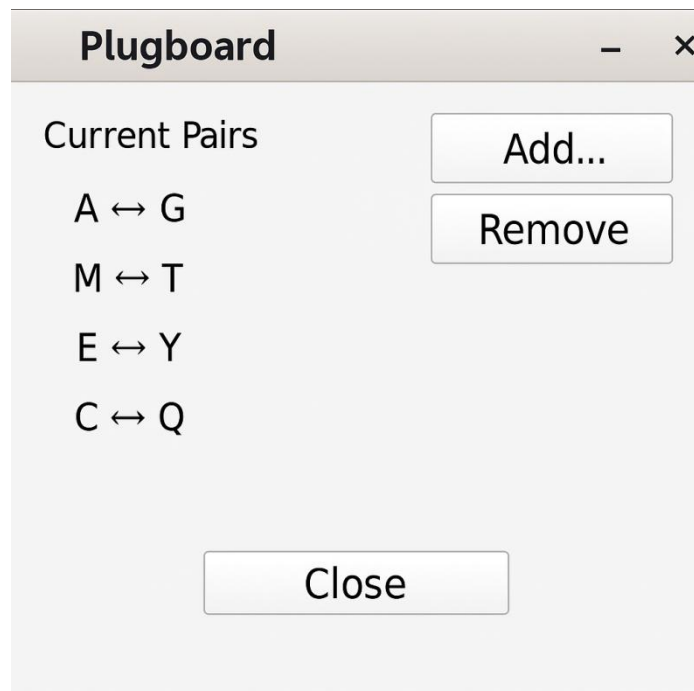
Ring Settings	1	2	3
	A ▾	A ▾	A ▾

Starting Positions	A	B	C
	A ▾	B ▾	C ▾

[Apply]

Рисунок 4.5 – Форма налаштування роторів

Форму плагборду симулятора механічної шифрувальної машини наведено на рис. 4.6.



Plugboard [minimize] [close]

Current Pairs

- A ↔ G
- M ↔ T
- E ↔ Y
- C ↔ Q

[Add...]
[Remove]

[Close]

Рисунок 4.6 – Форма плагборду симулятора механічної шифрувальної машини

4.4 Висновки за розділом 4

Розроблено програмне забезпечення симуляції механічного пристрою шифрування даних.

Запропоновано структуру програмного забезпечення симулятора механічної шифрувальної машини.

Визначено, що основними класами симулятора механічної шифрувальної машини є: SimulatorController, CipherDevice, JeffersonCylinder, RotorMachine, Клас Rotor, TextIO, Alphabet, FrequencyAnalyzer, ReportGenerator, ProjectStorage та інші.

Описано функціонування програмного забезпечення симулятора механічної шифрувальної машини.

Описано особливості реалізації програмного забезпечення симуляції механічного пристрою шифрування даних.

Виконано проєктування інтерфейсу симулятора механічної шифрувальної машини.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення й умови застосування програми

Програма симулятора механічної шифрувальної машини призначена для моделювання процесів шифрування та дешифрування текстової інформації з використанням принципів роботи класичних роторних пристроїв. Програмне забезпечення відтворює логіку функціонування основних компонентів механічної шифрувальної машини, зокрема клавіатури, роторного механізму, відбивача, комутаційної панелі (плагборду) та лампової панелі індикації результатів шифрування.

Програма написана на мові Python. Такий вибір зумовлений її універсальністю, простотою синтаксису, наявністю широкого спектра бібліотек для моделювання, обробки даних і візуалізації, а також активною спільнотою розробників. Використання Python дозволяє ефективно реалізувати алгоритми симуляції, забезпечити наочність роботи програмного забезпечення та спростити подальше розширення функціональних можливостей системи.

Для реалізації програмного забезпечення у якості середовища розробки обрано Visual Studio Code, що обґрунтовано його кросплатформенністю, гнучкістю налаштувань та широкими можливостями розширення. Підтримка мови Python, інтеграція з системами контролю версій, налагоджувачами та зовнішніми інструментами аналізу коду роблять Visual Studio Code зручним і ефективним середовищем для розробки прикладного програмного забезпечення дослідницького та навчального призначення.

Програмне забезпечення симулятора механічної шифрувальної машини забезпечує виконання таких функцій:

- забезпечення вибору типу механічного пристрою шифрування для симуляції;

- надання можливості налаштування параметрів конфігурації (ключа) обраного пристрою;
- підтримка введення, редагування та збереження відкритого тексту для обробки;
- реалізація процесів шифрування та дешифрування інформації відповідно до обраної моделі;
- відображення результатів шифрування та дешифрування у зрозумілій для користувача формі;
- забезпечення коректності роботи алгоритмів шляхом перевірки оборотності перетворень;
- надання можливості візуального відображення принципу роботи механічного пристрою;
- забезпечення збереження та завантаження налаштувань симуляції;
- формування звітів про результати проведених експериментів;
- забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу.

Для експлуатації програмного забезпечення симулятора механічної шифрувальної машини необхідні такі умови застосування:

- персональний комп'ютер з операційною системою Windows або сумісною з середовищем розробки платформною;
- наявність пристроїв введення (клавіатура, миша);
- задані або вручну налаштовані параметри шифрувальної машини (порядок роторів, початкові позиції, кільцеві налаштування, конфігурація плагборду).

Функціональні характеристики розробленого програмного забезпечення симулятора механічної шифрувальної машини наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення Інтернет аукціону наведено у додатку Б.

5.2 Характеристики програми симулятора механічної шифрувальної машини

Програмне забезпечення симулятора механічної шифрувальної машини складається з сукупності взаємопов'язаних програмних модулів, кожен з яких реалізує окрему функціональну частину класичного механічного шифрувального пристрою. Архітектура програми побудована за модульним принципом, що забезпечує наочність, розширюваність і зручність супроводу.

До складу програмного забезпечення входять:

- модуль інтерфейсу користувача, який відповідає за відображення головної форми, віртуальної клавіатури, лампової панелі, форм налаштування роторів, плагборду та журналу шифрування;
- керуючий модуль (контролер), що координує взаємодію між інтерфейсом та логічними компонентами системи;
- модуль роторного механізму, який реалізує алгоритм прямого і зворотного проходження сигналу через ротори з урахуванням їхніх позицій, кільцевих налаштувань та правил крокування;
- модуль відбивача, що забезпечує симетричність процесу шифрування;
- модуль плагборду, який виконує комутацію символів на вході та виході шифрувального механізму;
- модуль журналювання, призначений для збереження результатів шифрування та поточного стану роторів.

Користувач програмного забезпечення симулятора механічної шифрувальної машини взаємодіє з ним через графічний інтерфейс користувача (GUI). Взаємодія здійснюється шляхом натискання клавіш віртуальної клавіатури, зміни параметрів роторів, налаштування комутаційних пар плагборду та перегляду результатів шифрування на ламповій панелі й у журналі операцій. Інтерфейс побудований таким чином, щоб максимально

наблизити процес роботи до принципів функціонування реального механічного шифрувального пристрою.

Програма симулятора механічної шифрувальної машини є функціонально завершеним програмним продуктом, що забезпечує коректне моделювання процесу шифрування та зручну взаємодію користувача з усіма основними компонентами шифрувального механізму.

5.3 Інструкція по експлуатації програми

5.3.1 Звернення до програми

Для запуску програмного забезпечення симулятора механічної шифрувальної машини необхідно запустити відповідний файл.

5.3.2 Вхідні й вихідні дані

Вхідними даними до програмного симулятора механічної шифрувальної машини є сукупність параметрів і дій користувача, необхідних для ініціалізації та виконання процесу шифрування. До вхідних даних належать:

- символи латинського алфавіту (A–Z), що вводяться користувачем за допомогою віртуальної клавіатури;
- конфігурація роторів (кількість, порядок встановлення);
- початкові позиції роторів;
- кільцеві налаштування (ring settings);
- налаштування плагборду (пари взаємозамінних літер) або ознака його відсутності;
- параметри відбивача (тип або фіксована конфігурація).

Усі вхідні дані вводяться користувачем через графічний інтерфейс програми та проходять перевірку на коректність перед використанням у процесі шифрування.

Вихідними даними програмного симулятора механічної шифрувальної машини є результати роботи алгоритму шифрування, що формуються після обробки кожного введенного символу. До вихідних даних належать:

- зашифрований символ, який відображається на ламповій панелі;
- оновлені позиції роторів після виконання крокування;
- записи в журналі шифрування, що містять вхідний символ, відповідний вихідний символ, стан роторів та час виконання операції;
- службові повідомлення про стан роботи програми або помилки введення (за наявності).

Таким чином, програмний симулятор забезпечує чітку відповідність між вхідними та вихідними даними, що дозволяє відтворювати процес механічного шифрування з високою точністю та забезпечує можливість аналізу результатів у навчальних і дослідницьких цілях.

5.3.3 Повідомлення

У процесі роботи з програмним симулятором механічної шифрувальної машини користувач може отримувати такі повідомлення:

- повідомлення про готовність програми до роботи;
- інформаційні повідомлення про успішне застосування налаштувань;
- повідомлення про результат шифрування (відображення зашифрованого символу);
- попереджувальні повідомлення про некоректне введення або помилки налаштувань;
- службові повідомлення про збереження стану та завершення роботи програми.

Наведені повідомлення забезпечують зворотний зв'язок між користувачем і програмою та сприяють коректній і зручній експлуатації симулятора.

5.4 Виконання програмного забезпечення симулятором механічної шифрувальної машини

Після запуску програмного забезпечення симулятора механічної шифрувальної машини користувачу відображається головна форма програми, яка містить віртуальну клавіатуру, лампову панель, елементи керування та інформаційний блок зі станом системи. Тут користувачі можуть бачити поточні налаштування роторів, індикацію зашифрованих символів, а також доступні елементи для переходу до додаткових форм налаштування (рис. 5.1).

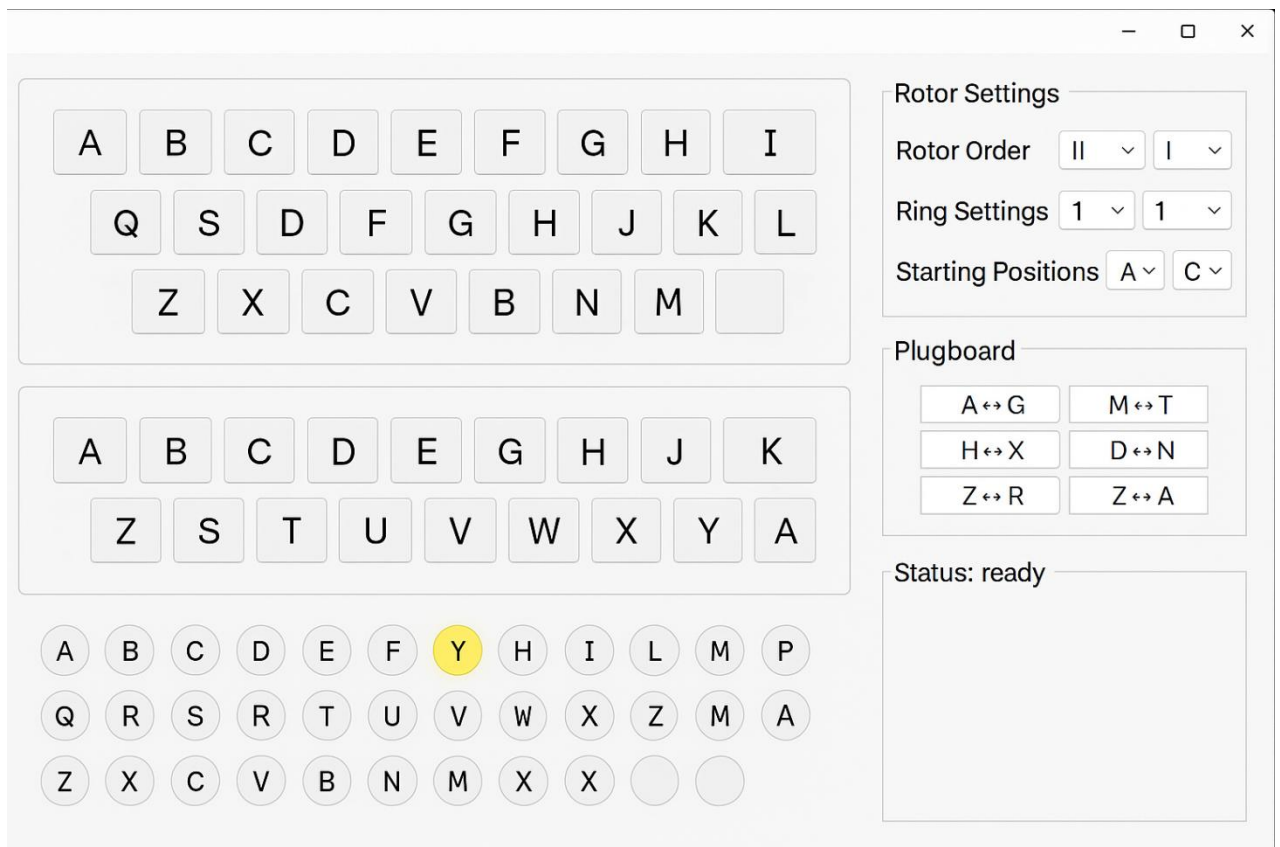


Рисунок 5.1 – Інтерфейс головної форми програмного забезпечення симулятора механічної шифрувальної машини

Для отримання можливості коректного виконання процесу шифрування необхідно задати або перевірити параметри роторів, їх початкові позиції та, за потреби, налаштування плагборду, після чого користувач може здійснювати введення символів за допомогою віртуальної клавіатури (рис.5.2).

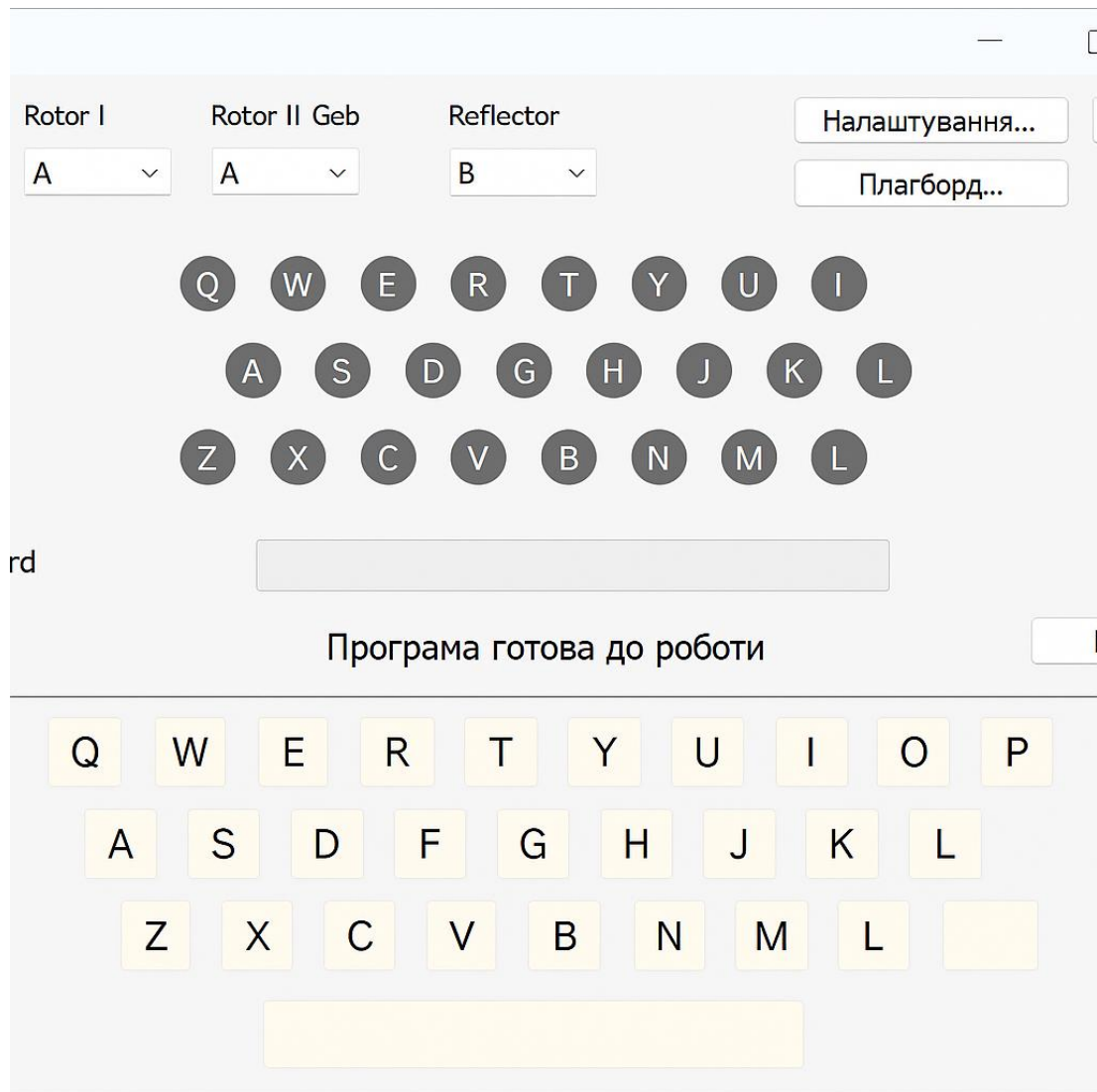


Рисунок 5.2 – Коректне виконання процесу шифрування

5.5 Тестування програмного забезпечення симулятора механічної шифрувальної машини

Виконано тестування програмного забезпечення симулятора механічної шифрувальної машини з метою перевірки коректності реалізації функціональних можливостей, стабільності роботи та відповідності програмного продукту поставленим вимогам.

У процесі тестування перевірялися:

- запуск та ініціалізація програми;
- коректність налаштування роторів і плагборду;
- правильність крокування роторного механізму;

- коректність шифрування введених символів;
- відображення результатів шифрування на ламповій панелі;
- формування та ведення журналу шифрування;
- обробка помилкових або некоректних дій користувача.

Виявлено, що програма функціонує правильно та злагоджено. Вона реалізує всі передбачені функціональні вимоги та успішно виконує свою основну задачу – моделювання процесу шифрування даних за допомогою механічної роторної шифрувальної машини.

Розроблене програмне забезпечення дозволяє наочно відтворювати роботу класичних механічних шифрувальних пристроїв, змінювати конфігурацію шифрувального механізму, аналізувати результати шифрування.

За результатами тестування критичних помилок у роботі програми не виявлено, що підтверджує її готовність до експлуатації.

5.6 Висновки за розділом 5

Описано програмне забезпечення симулятора механічної шифрувальної машини. Виконано тестування розробленого програмного забезпечення симуляції механічної роторної шифрувальної машини. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє наочно відтворювати роботу класичних механічних шифрувальних пристроїв, змінювати конфігурацію шифрувального механізму, аналізувати результати шифрування.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено механічні пристрої шифрування даних. Розглянуто історичні передумови їх виникнення, основні класи та принципи функціонування. Показано, що механічні криптосистеми реалізують базові криптографічні ідеї, які зберегли свою актуальність у сучасній криптографії [1], [6]. Сформульовано функціональні вимоги до програмного забезпечення симуляції моделей механічних пристроїв шифрування даних.

Було розроблено та реалізовано програмний симулятор механічної роторної шифрувальної машини, який відтворює принципи роботи класичних електромеханічних криптографічних пристроїв. У процесі виконання роботи проаналізовано теоретичні основи механічного шифрування, спроектовано архітектуру програмного забезпечення, розроблено графічний інтерфейс користувача та реалізовано основні функціональні модулі симулятора.

Розроблене програмне забезпечення надає можливість налаштування роторів, плагборду та відбивача, виконує коректне крокування роторного механізму, здійснює шифрування введених символів у режимі реального часу та веде журнал результатів роботи. Проведене тестування підтвердило правильність і стабільність функціонування програми, а також її відповідність поставленим вимогам.

Визначено, що механічні пристрої шифрування не лише виконували практичну функцію захисту інформації, а й сприяли формуванню теоретичних основ криптографії. Аналіз принципів їх роботи дозволив сформулювати ключові криптографічні поняття, зокрема роль ключа як параметра безпеки, необхідність оборотності перетворень, значення композиції операцій та залежність криптографічної стійкості системи від складності її математичної структури [4], [5]. Саме досвід використання механічних шифраторів показав, що безпека системи не може ґрунтуватися виключно на секретності її конструкції.

За результатами проведеного аналізу зроблено висновок, що побудова математичних моделей дає змогу перейти від інтуїтивного розуміння процесів шифрування до формального аналізу, що є необхідною умовою для розроблення надійних програмних засобів захисту інформації [5]. Водночас дослідження механічних шифрувальних пристроїв з позицій сучасної математики дозволяє глибше усвідомити еволюцію криптографічних методів та закономірності їх розвитку.

Тому актуальною є розробка програмного забезпечення для симуляції роботи механічних пристроїв шифрування даних.

Сформульовано функціональні вимоги до програмного забезпечення симулятора роботи пристрою шифрування даних.

Для реалізації програмного забезпечення симуляції роботи пристрою шифрування даних обрано мову програмування Python. Такий вибір зумовлений її універсальністю, простотою синтаксису, наявністю широкого спектра бібліотек для моделювання, обробки даних і візуалізації, а також активною спільнотою розробників. Використання Python дозволяє ефективно реалізувати алгоритми симуляції, забезпечити наочність роботи програмного забезпечення та спростити подальше розширення функціональних можливостей системи.

Для реалізації програмного забезпечення обрано середовище розробки Visual Studio Code, що обґрунтовано його кросплатформенністю, гнучкістю налаштувань та широкими можливостями розширення. Підтримка мови Python, інтеграція з системами контролю версій, налагоджувачами та зовнішніми інструментами аналізу коду роблять Visual Studio Code зручним і ефективним середовищем для розробки прикладного програмного забезпечення дослідницького та навчального призначення.

Новизна роботи полягає у тому, що створено програмний симулятор механічної шифрувальної машини з розширеним навчальним функціоналом, який поєднує точне відтворення логіки роботи роторного механізму з наочним графічним інтерфейсом і можливістю детального аналізу процесу

шифрування. На відміну від існуючих реалізацій, розроблений симулятор орієнтований не лише на отримання результату шифрування, а й на демонстрацію внутрішніх етапів роботи шифрувального пристрою.

Практичне значення роботи полягає у можливості використання розробленого програмного забезпечення в освітньому процесі для вивчення основ криптології, історичних методів шифрування та принципів побудови симетричних криптографічних систем.

Визначено, що основними класами симулятора механічної шифрувальної машини є: SimulatorController, CipherDevice, JeffersonCylinder, RotorMachine, Клас Rotor, TextIO, Alphabet, FrequencyAnalyzer, ReportGenerator, ProjectStorage та інші.

Описано функціонування програмного забезпечення симулятора механічної шифрувальної машини.

Описано особливості реалізації програмного забезпечення симуляції механічного пристрою шифрування даних.

Виконано проєктування інтерфейсу симулятора механічної шифрувальної машини.

Виконано тестування розробленого програмного забезпечення симуляції механічної роторної шифрувальної машини. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє наочно відтворювати роботу класичних механічних шифрувальних пристроїв, змінювати конфігурацію шифрувального механізму, аналізувати результати шифрування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Schneier B. Applied cryptography: protocols, algorithms, and source code in C. – 2nd ed. – New York : John Wiley & Sons, 1996. – 784 p.
2. Singh S. The code book: the science of secrecy from ancient Egypt to quantum cryptography. – London : Fourth Estate, 2000. – 412 p.
3. Kahn D. The codebreakers: the comprehensive history of secret communication. – New York : Scribner, 1996. – 1181 p.
4. Katz J., Lindell Y. Introduction to modern cryptography. – 2nd ed. – Boca Raton : CRC Press, 2014. – 623 p.
5. Menezes A., van Oorschot P., Vanstone S. Handbook of applied cryptography. – Boca Raton : CRC Press, 1996. – 816 p.
6. Stallings W. Cryptography and network security: principles and practice. – 7th ed. – Boston : Pearson, 2017. – 768 p.
7. Shannon C. E. Communication theory of secrecy systems // Bell System Technical Journal. – 1949. – Vol. 28. – P. 656–715.
8. Smith M. Station X: the codebreakers of Bletchley Park. – London : Pan Books, 2011. – 432 p.
9. Deavours C. A., Kruh L. Machine cryptography and modern cryptanalysis. – Boston : Artech House, 1985. – 422 p.
10. Bauer F. L. Decrypted secrets: methods and maxims of cryptology. – Berlin : Springer, 2007. – 502 p.
11. Корченко О. Г. Основи криптографічного захисту інформації : навч. посіб. – Київ : НАУ, 2004. – 240 с.
12. Diffie W., Hellman M. New directions in cryptography // IEEE Transactions on Information Theory. – 1976. – Vol. 22, № 6. – P. 644–654.
13. Хорошко В. М., Корченко О. Г. Методи та засоби захисту інформації : навч. посіб. – Київ : Юніор, 2003. – 504 с.
14. Reeds J. A. Cryptanalysis of the Enigma // Cryptologia. – 1997. – Vol. 21, № 4. – P. 291–309.

15. Бабак В. П., Козловський В. В. Захист інформації в комп'ютерних системах : навч. посіб. – Київ : КНЕУ, 2008. – 352 с.
16. Kerckhoffs A. La cryptographie militaire // Journal des sciences militaires. – 1883. – Vol. 9. – P. 5–38.
17. Пархоменко О. Ю. Криптографічні методи захисту інформації : навч. посіб. – Київ : Кондор, 2011. – 296 с.
18. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Communications of the ACM. – 1978. – Vol. 21, № 2. – P. 120–126.
19. Жуков І. А., Корченко О. Г. Інформаційна безпека та захист інформації : навч. посіб. – Київ : НАУ, 2012. – 432 с.
20. Hodges A. Alan Turing: the enigma. – London : Vintage Books, 2014. – 768 p.
21. Конахович Г. Ф. Захист інформації в інформаційно-телекомунікаційних системах : підручник. – Київ : МК-Прес, 2010. – 640 с.
22. Landwehr C. E. A survey of cryptographic techniques // IEEE Computer. – 1981. – Vol. 14, № 9. – P. 29–41.
23. Поліщук Є. А. Криптографія та стеганографія : навч. посіб. – Львів : Новий Світ–2000, 2014. – 280 с.
24. Copeland B. J. (ed.). The essential Turing: seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life. – Oxford : Oxford University Press, 2004. – 600 p.
25. Основи захисту інформації : навч. посіб. / В. М. Хорошко, О. Г. Корченко, Ю. Є. Яремчук [та ін.]. – Київ : ДУІКТ, 2009. – 370 с.
26. Корченко О. Г. Сучасні криптографічні методи захисту інформації // Захист інформації. – 2010. – № 3. – С. 5–12.
27. Яремчук Ю. Є. Симетричні криптографічні алгоритми та їх застосування // Вісник НТУУ «КПІ». – 2012. – № 45. – С. 98–104.

28. Гавриленко С. В. Аналіз історичних криптосистем та їх вплив на сучасну криптографію // Наукові праці ОНАХТ. – 2015. – Т. 48, № 2. – С. 112–118.

29. Романенко В. А. Моделювання криптографічних алгоритмів у навчальному процесі // Інформаційні технології в освіті. – 2018. – № 3. – С. 41–47.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для програмної симуляції механічного шифрувального пристрою.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження математичних моделей механічних пристроїв шифрування даних та програмна реалізація симуляції», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для симуляції роботи механічного шифрувального пристрою.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення симулятора механічних шифрувальних пристроїв забезпечує виконання таких функцій:

- забезпечення вибору типу механічного пристрою шифрування для симуляції;
- надання можливості налаштування параметрів конфігурації (ключа) обраного пристрою;
- підтримка введення, редагування та збереження відкритого тексту для обробки;

- реалізація процесів шифрування та дешифрування інформації відповідно до обраної моделі;
- відображення результатів шифрування та дешифрування у зрозумілій для користувача формі;
- забезпечення коректності роботи алгоритмів шляхом перевірки оборотності перетворень;
- надання можливості візуального відображення принципу роботи механічного пристрою;
- забезпечення збереження та завантаження налаштувань симуляції;
- формування звітів про результати проведених експериментів;
- забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для симуляції роботи механічного шифрувального пристрою повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення симуляції роботи механічного шифрувального пристрою повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для симуляції роботи механічного шифрувального пристрою необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного забезпечення для симуляції роботи механічного шифрувального пристрою необхідно таке апаратне та програмне забезпечення:

Апаратні вимоги:

- персональний комп'ютер або ноутбук;
- процесор з тактовою частотою не нижче 1,5 ГГц;
- оперативна пам'ять обсягом не менше 2 ГБ;
- вільне місце на жорсткому диску не менше 200 МБ;
- монітор з роздільною здатністю не нижче 1280×720;
- пристрої введення (клавіатура, миша або сенсорна панель).

Програмні вимоги:

- операційна система сімейства Windows (Windows 10 або новіша версія) або сумісна з використанням середовищем розробки платформа;
- встановлене середовище виконання програм (у разі використання інтерпретованої або віртуальної платформи);
- графічна підсистема з підтримкою стандартних віконних інтерфейсів;
- доступ до файлової системи для збереження налаштувань і журналу роботи програми.

Зазначені вимоги є мінімальними та забезпечують стабільну роботу програмного симулятора. Програмне забезпечення не потребує спеціалізованих апаратних засобів або додаткових периферійних пристроїв, що дозволяє використовувати його на більшості сучасних персональних комп'ютерів.

A.3.6 Вимоги до маркування і пакування

Програма для симуляції роботи механічного шифрувального пристрою може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення «Симулятор роботи механічного шифрувального пристрою».

A.4 Вхідні дані до роботи

Вхідними даними до програмного симулятора механічної шифрувальної машини є сукупність параметрів і дій користувача, необхідних для ініціалізації та виконання процесу шифрування. До вхідних даних належать:

- символи латинського алфавіту (A–Z), що вводяться користувачем за допомогою віртуальної клавіатури;
- конфігурація роторів (кількість, порядок встановлення);
- початкові позиції роторів;
- кільцеві налаштування (ring settings);
- налаштування плагборду (пари взаємозамінних літер) або ознака його відсутності;
- параметри відбивача (тип або фіксована конфігурація).

Усі вхідні дані вводяться користувачем через графічний інтерфейс програми та проходять перевірку на коректність перед використанням у процесі шифрування.

Вихідними даними програмного симулятора механічної шифрувальної машини є результати роботи алгоритму шифрування, що формуються після обробки кожного введенного символу. До вихідних даних належать:

- зашифрований символ, який відображається на ламповій панелі;
- оновлені позиції роторів після виконання крокування;

- записи в журналі шифрування, що містять вхідний символ, відповідний вихідний символ, стан роторів та час виконання операції;
- службові повідомлення про стан роботи програми або помилки введення (за наявності).

ДОДАТОК Б
Фрагмент тексту програми

```

from __future__ import annotations

import json
import math
import os
import time
from dataclasses import dataclass, asdict
from typing import Dict, List, Optional, Tuple, Protocol

import tkinter as tk
from tkinter import ttk, filedialog, messagebox

# =====
# Core utilities
# =====

class Alphabet:
    """Alphabet utilities: normalization, indexing, filtering."""
    def __init__(self, letters: str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"):
        uniq = []
        for ch in letters:
            ch = ch.upper()
            if ch not in uniq:
                uniq.append(ch)
        self.letters = "".join(uniq)

        if len(self.letters) < 2:
            raise ValueError("Alphabet must contain at least 2 unique characters.")

        self._index = {c: i for i, c in enumerate(self.letters)}

    def has(self, ch: str) -> bool:
        return ch.upper() in self._index

    def normalize(self, text: str, keep_unknown: bool = False) -> str:
        out = []
        for ch in text:
            up = ch.upper()
            if up in self._index:
                out.append(up)
            else:
                if keep_unknown:
                    out.append(ch)
        return "".join(out)

    def idx(self, ch: str) -> int:
        return self._index[ch.upper()]

    def ch(self, idx: int) -> str:
        return self.letters[idx % len(self.letters)]

```

```

def size(self) -> int:
    return len(self.letters)

class TextIO:
    """Text load/save helper."""
    @staticmethod
    def load_text(path: str) -> str:
        with open(path, "r", encoding="utf-8") as f:
            return f.read()

    @staticmethod
    def save_text(path: str, text: str) -> None:
        with open(path, "w", encoding="utf-8") as f:
            f.write(text)

class FrequencyAnalyzer:
    """Simple frequency analysis + Shannon entropy (educational)."""
    def __init__(self, alphabet: Alphabet):
        self.alphabet = alphabet

    def frequencies(self, text: str) -> Dict[str, float]:
        text = self.alphabet.normalize(text)
        total = len(text)
        if total == 0:
            return {c: 0.0 for c in self.alphabet.letters}

        counts = {c: 0 for c in self.alphabet.letters}
        for ch in text:
            counts[ch] += 1
        return {c: counts[c] / total for c in self.alphabet.letters}

    def entropy(self, text: str) -> float:
        freqs = self.frequencies(text)
        h = 0.0
        for p in freqs.values():
            if p > 0:
                h -= p * math.log2(p)
        return h

class ProjectStorage:
    """Save/load simulation settings, text, results."""
    @staticmethod
    def save_project(path: str, data: dict) -> None:
        with open(path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=2)

    @staticmethod
    def load_project(path: str) -> dict:
        with open(path, "r", encoding="utf-8") as f:
            return json.load(f)

```

```

class ReportGenerator:
    """Generate a simple Markdown report about experiment."""
    @staticmethod
    def generate_markdown(
        title: str,
        device_name: str,
        config: dict,
        plaintext: str,
        ciphertext: str,
        decrypted: str,
        analysis: dict,
        log_lines: List[str],
    ) -> str:
        ts = time.strftime("%Y-%m-%d %H:%M:%S")
        ok = (plaintext == decrypted)
        cfg_pretty = json.dumps(config, ensure_ascii=False, indent=2)

        md = []
        md.append(f"# {title}\n")
        md.append(f"- **Дата/час:** {ts}\n")
        md.append(f"- **Пристрій:** {device_name}\n")
        md.append(f"- **Оборотність (перевірка):** {'OK' if ok else 'FAIL'}\n")
        md.append("## Конфігурація\n")
        md.append("`json\n" + cfg_pretty + "\n`\n")
        md.append("## Вхідні та вихідні дані\n")
        md.append("### Відкритий текст\n")
        md.append("`text\n" + plaintext + "\n`\n")
        md.append("### Шифртекст\n")
        md.append("`text\n" + ciphertext + "\n`\n")
        md.append("### Результат дешифрування\n")
        md.append("`text\n" + decrypted + "\n`\n")
        md.append("## Статистичний аналіз\n")
        for k, v in analysis.items():
            md.append(f"- **{k}:** {v}\n")
        md.append("\n## Журнал подій\n")
        md.append("`text")
        md.extend(log_lines[-200:]) # limit
        md.append("`")
        md.append("")
        return "\n".join(md)

# =====
# Cipher interfaces
# =====

class CipherDevice(Protocol):
    """Interface for cipher devices."""
    name: str

```

```

def set_config(self, config: dict) -> None: ...
def get_config(self) -> dict: ...
def encrypt(self, plaintext: str) -> str: ...
def decrypt(self, ciphertext: str) -> str: ...
def visualize_state(self) -> str: ...

# =====
# Rotor machine implementation
# =====

@dataclass
class Rotor:
    """A single rotor with wiring permutation and ring setting."""
    alphabet: str
    wiring: str          # permutation of alphabet
    notch: str           # notch character(s) that trigger next
rotor step
    position: int = 0     # 0..N-1
    ring_setting: int = 0 # 0..N-1 (ring offset)

    def __post_init__(self):
        if sorted(self.wiring) != sorted(self.alphabet):
            raise ValueError("Rotor wiring must be a permutation of
alphabet.")
        self._fwd = {self.alphabet[i]: self.wiring[i] for i in
range(len(self.alphabet))}
        self._rev = {self.wiring[i]: self.alphabet[i] for i in
range(len(self.alphabet))}
        self._idx = {c: i for i, c in enumerate(self.alphabet)}

    def at_notch(self) -> bool:
        ch = self.alphabet[self.position]
        return ch in self.notch

    def step(self) -> None:
        self.position = (self.position + 1) % len(self.alphabet)

    def _shift_in(self, ch: str) -> str:
        n = len(self.alphabet)
        i = self._idx[ch]
        # position and ring setting influence
        i = (i + self.position - self.ring_setting) % n
        return self.alphabet[i]

    def _shift_out(self, ch: str) -> str:
        n = len(self.alphabet)
        i = self._idx[ch]
        i = (i - self.position + self.ring_setting) % n
        return self.alphabet[i]

    def forward(self, ch: str) -> str:
        x = self._shift_in(ch)

```

```

        y = self._fwd[x]
        return self._shift_out(y)

    def backward(self, ch: str) -> str:
        x = self._shift_in(ch)
        y = self._rev[x]
        return self._shift_out(y)

class RotorMachine:
    """Educational Enigma-like rotor machine (3 rotors + reflector +
    optional plugboard)."""
    name = "RotorMachine"

    def __init__(self, alphabet: Alphabet):
        self.alphabet = alphabet
        self.plugboard: Dict[str, str] = {}
        self.rotors: List[Rotor] = []
        self.reflector: Dict[str, str] = {}

        # default config
        self.set_config(self.default_config())

    @staticmethod
    def default_rotor_sets(alpha: str) -> Dict[str, Tuple[str, str]]:
        # Classic-like rotors (educational). Not intended for real
        security.
        # Each: wiring, notch
        return {
            "I": ("EKMFLGDQVZNTOWYHXUSPAIBRCJ", "Q"),
            "II": ("AJDKSIRUXBLHWTMCQGZNPYFVOE", "E"),
            "III": ("BDFHJLCPRTXVZNYEIWGAKMUSQO", "V"),
            "IV": ("ESOV郑JAYQUIRHXNLFTGKDCMWB", "J"),
            "V": ("VZBRGITYUPSDNHLXAWMJQOFECK", "Z"),
        }

    @staticmethod
    def default_reflectors(alpha: str) -> Dict[str, str]:
        # Reflector "B" style mapping (educational)
        # Must be involution: R(R(x)) = x
        return {
            "B": "YRUHQSLDPXNGOKMIEBFZCWVJAT",
            "C": "FVPJIAOYEDRZXWGCTKUQSBNMHL",
        }

    def default_config(self) -> dict:
        return {
            "rotor_order": ["I", "II", "III"],
            "positions": ["A", "A", "A"],
            "ring_settings": [0, 0, 0],
            "reflector": "B",
            "plugboard_pairs": [], # e.g. ["AB", "CD"]
        }

```

```
def set_config(self, config: dict) -> None:
    alpha = self.alphabet.letters
    rotor_sets = self.default_rotor_sets(alpha)
    refl_sets = self.default_reflectors(alpha)

    rotor_order = config.get("rotor_order", ["I", "II", "III"])
    positions = config.get("positions", ["A", "A", "A"])
```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЗАПОРІЗЬКА ПОЛІТЕХНІКА"
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ МОДЕЛЕЙ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ДАНИХ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИМУЛЯЦІЇ

ВИКОНАВ:
СТУДЕНТ ГР.КНТ-124м

Андрій КУЧЕР

КЕРІВНИК:
К.Т.Н., ДОЦЕНТ

Тетяна ЗАЙКО

Рисунок В.1 – Слайд 1

2

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- ▶ **ОБ'ЄКТ ДОСЛІДЖЕННЯ** – ПРОЦЕС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ РОБОТИ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ДАНИХ.
- ▶ **ПРЕДМЕТ ДОСЛІДЖЕННЯ** – ПРОГРАМНІ ЗАСОБИ ДЛЯ СИМУЛЯЦІЇ РОБОТИ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ДАНИХ.
- ▶ **МЕТА РОБОТИ** – ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ РОБОТИ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ДАНИХ.

Рисунок В.2 – Слайд 2

ЗАВДАННЯ ДО РОБОТИ

- ▶ 1.ПРОАНАЛІЗУВАТИ РОЗВИТОК ТА ЕВОЛЮЦІЮ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ІНФОРМАЦІЇ.
- ▶ 2.ДОСЛІДИТИ ПРИНЦИПИ ФУНКЦІОНУВАННЯ КЛАСИЧНИХ МЕХАНІЧНИХ КРИПТОГРАФІЧНИХ СИСТЕМ.
- ▶ 3.ПРОВЕСТИ КЛАСИФІКАЦІЮ МЕХАНІЧНИХ ПРИСТРОЇВ ШИФРУВАННЯ ЗА ТИПАМИ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ.
- ▶ 4.ФОРМАЛІЗУВАТИ ПРОЦЕСИ ШИФРУВАННЯ ТА ДЕШИФРУВАННЯ МЕХАНІЧНИХ КРИПТОСИСТЕМ У ВИГЛЯДІ МАТЕМАТИЧНИХ МОДЕЛЕЙ.
- ▶ 5.РОЗРОБИТИ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИМУЛЯЦІЇ РОБОТИ МЕХАНІЧНОГО КРИПТОМЕХАНІЗМУ ШИФРУВАННЯ ДАНИХ.

Рисунок В.3 – Слайд 3

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Ruby
Ефективне використання пам'яті	+	+-	+-
Компіляція програмного коду	+	-	-
Висока швидкість функціонування розроблюваного програмного забезпечення	+	+-	+-
Зручність реалізації алгоритмічних і модельних обчислень	+-	+	+-
Наявність бібліотек для математичного моделювання	+-	+	-
Зручність створення навчального та дослідницького програмного забезпечення	+-	+	+-
Простота реалізації <u>симуляційних</u> моделей	+- ^I	+	+-
<u>Кросплатформеність</u>	+	+	+-

Рисунок В.4 – Слайд 4

ПОРІВНЯЛЬНИЙ АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio Code	PyCharm	Spyder
Можливості розширення та плагіни	+	+	+—
Повна підтримка мови Python	+	+	+
Підтримка віртуальних середовищ	+	+	+
Зручність розробки наукових та навчальних проєктів	+—	+	+
Підтримка відлагодження	+	+	+—
Інтеграція з системами контролю версій (Git)	+	+	—
Кросплатформеність	+	I +	+
Активна спільнота та документація	+	+	+—
Невисокі апаратні вимоги	+	+—	+
Відкритість та безкоштовність	+	+—	+

Рисунок В.5 – Слайд 5

ДІАГРАМА КЛАСІВ ПРОГРАМИ

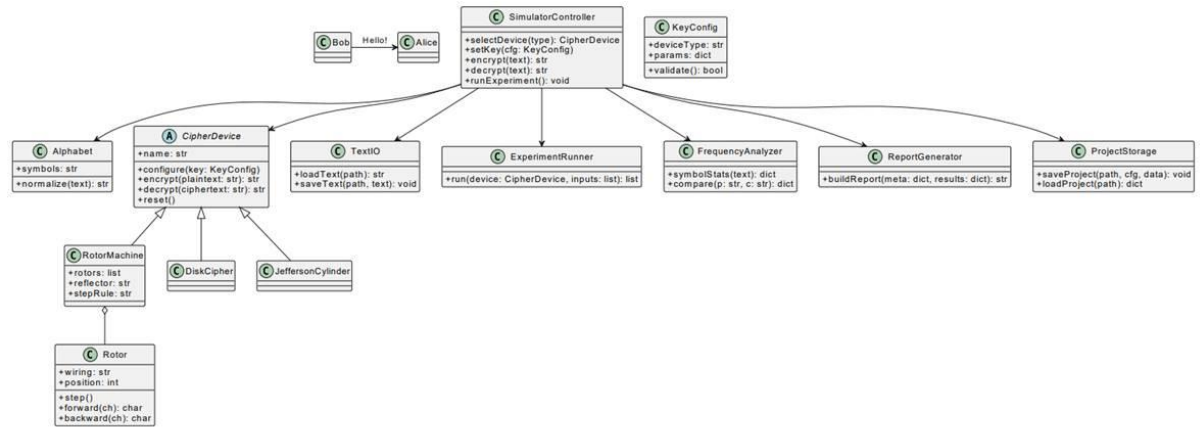
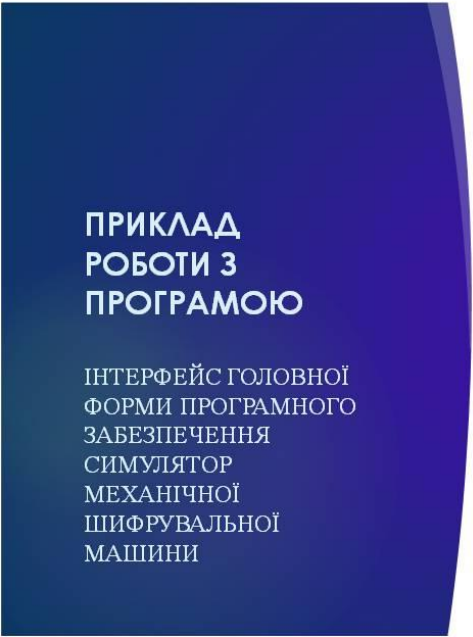


Рисунок В.6 – Слайд 6



9

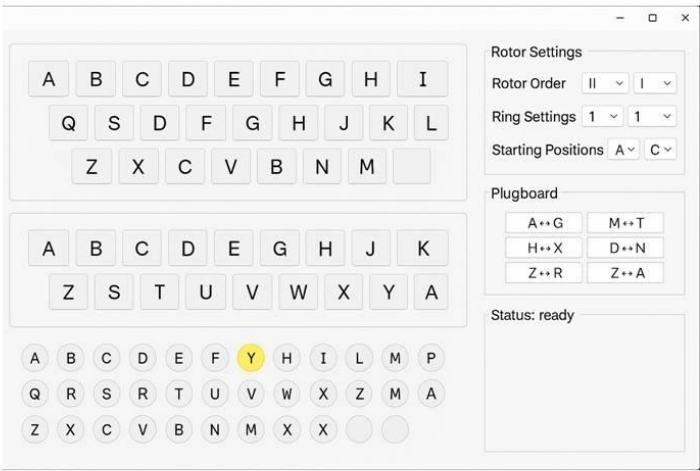
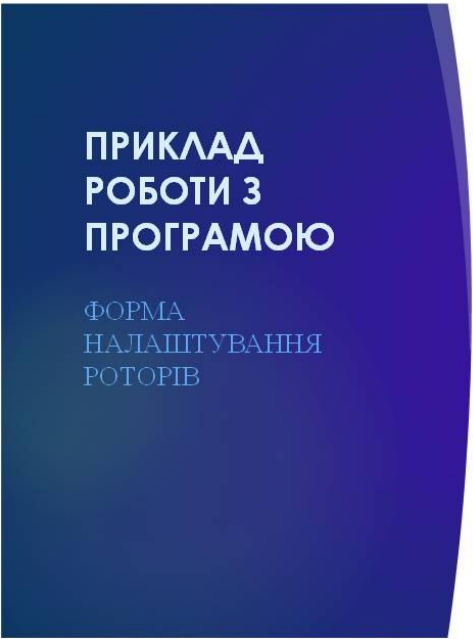


Рисунок В.9 – Слайд 9



10

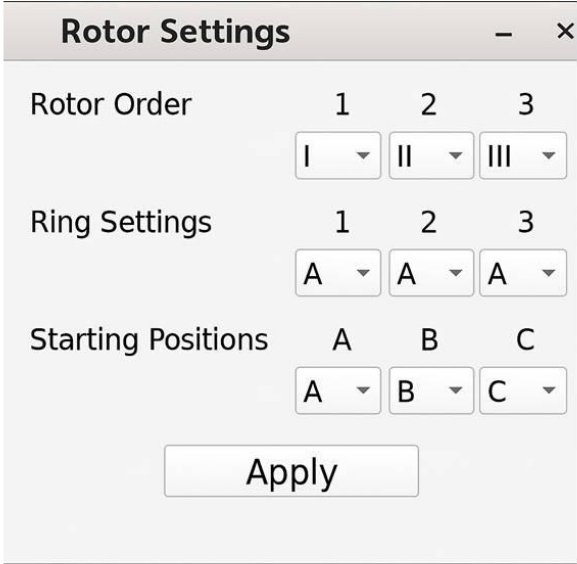


Рисунок В.10 – Слайд 10

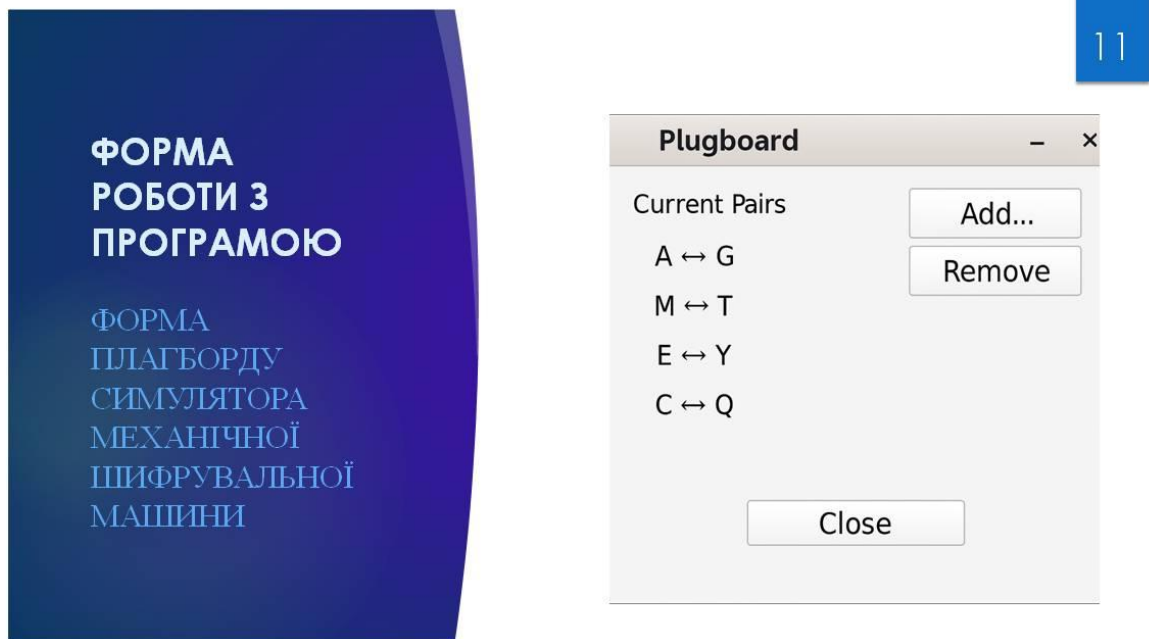


Рисунок В.11 – Слайд 11



Рисунок В.12 – Слайд 12