

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи програмно-визначаємих
сховищ для NVMe на базі технології RDDA”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Мішура Р.М.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 123 "Комп'ютерна інженерія"
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мішурі Руслану Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту *23.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи *1 аркуш*

Функціональна схема системи *1 аркуш*

Діаграма процесів *1 аркуш*

Блок-схема алгоритму роботи додатку *2 аркуша*

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Мішура Р.М.
(прізвище та ініціали)

АНОТАЦІЯ

Мішура Р.М. Програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

Метою розробки є програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

Результат роботи – програмна реалізація системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерна інженерія, NVMe, RDDA

ABSTRACT

Mishura R.M. Software for a software-defined storage system for NVMe based on RDDA technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a software-defined storage system for NVMe based on RDDA technology.

The purpose of the development is software for a software-defined storage system for NVMe based on RDDA technology.

The result of the work is a software implementation of a software-defined storage system for NVMe based on RDDA technology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in Visual C#.

Keywords: computer engineering, NVMe, RDDA

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	4
1.1 Призначення системи.....	4
1.2 Область застосування.....	5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	13
2.3 Розгорнута постановка завдання	16
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	18
3.1 Опис функціонування системи	18
3.2 Розробка структурної схеми.....	22
3.3 Розробка функціональної схеми	34
3.4 Розробка діаграми процесів.....	45
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	47
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	47
4.2 Захист розробленого програмного забезпечення.....	59
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	65
6 ОСНОВНІ ВИСНОВКИ.....	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	74

					ВКРБ-123.26.0052.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата	<i>Програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA</i>	Літ.	Аркуш	Аркушів
Розроб.	Мішура Р.М.					Б	1	80
Перев.	Марченко К.М.							
Н.контр.	Коваленко А.С.					ЦНТУ КІ-22-2		
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АБГШ	—	аддитивний білий гаусів шум
БЧХ	—	код Боуза-Чоудхури-Хоквінгема
ЕОМ	—	електронно-обчислювальна машина
ІМС	—	інтегральна мікросхема
СЗД	—	системи зберігання даних
ARQ	—	автоматичний запит повторної передачі
CD-DA	—	Compact Disc Digital Audio
CIRC	—	Cross Interleaved Reed Solomon Code
EAB	—	Embedded Array Block, блок зосередженої пам'яті
ECC	—	error-correcting code, код корекції помилок
FEC	—	метод прямої корекції помилок
LDPC	—	коди Галлагера
NAK	—	негативне підтвердження

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

ВСТУП

Актуальність теми. SSD та NVMe-накопичувачі зараз домінують у українських ноутбуках, ігрових ПК, бізнес-серверах та робочих станціях зі штучним інтелектом. Поставки зросли більш ніж на 40% у 2025–2026 роках, проте кількість повідомлень про збої швидко зростає через вищу ємність, швидкість PCIe 5.0/6.0, високі робочі навантаження на запис та складні контролери. Коли ці накопичувачі виходять з ладу – чи то через логічне пошкодження, чи то через фізичне пошкодження NAND – відновлення часто є складнішим та дорожчим, ніж у традиційних жорстких дисках.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDMA.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- Огляд існуючих систем програмно-визначаємих сховищ для NVMe на базі технології RDMA.
- Дослідження системи програмно-визначаємих сховищ для NVMe на базі технології RDMA.
- Програмна реалізація системи програмно-визначаємих сховищ для NVMe на базі технології RDMA.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі програмно-визначаємих сховищ для NVMe на базі технології RDMA.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDMA, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сучасні SSD та NVMe-накопичувачі (на базі PCIe) швидші та надійніші, ніж жорсткі диски, за умов нормального використання, але кілька тенденцій, характерних для 2026 року, призводять до вищих показників відмов:

– Робочі навантаження штучного інтелекту та машинного навчання – українські компанії та дослідницькі лабораторії проводять інтенсивне навчання/виведення даних на високоємні NVMe-масиви. Постійний високий обсяг записів зношує NAND-комірки набагато швидше, ніж типове використання споживачами.

– Впровадження PCIe 5.0 та 6.0 – Нові материнські плати та корпоративні сервери працюють з вищими швидкостями та енергоспоживанням, що призводить до теплового навантаження та збоїв контролера.

– Складні пропріетарні контролери – виробники використовують спеціальну прошивку, яку важко обійти під час відновлення – на відміну від старих SSD-накопичувачів, де інструменти могли легко зчитувати NAND безпосередньо.

– Більша ємність та щільніша NAND – споживчі накопичувачі ємністю 4–8 ТБ та корпоративні SSD-накопичувачі ємністю 30+ ТБ вміщують більше комірок у меншому просторі, що збільшує ризик виробничих дефектів та помилок читання/запису.

– Пошкодження живлення та прошивки – раптові вимкнення, неякісні блоки живлення або погані оновлення можуть призвести до виходу з ладу дисків NVMe, особливо в RAID-масивах або ноутбуках.

Все більше користувачів в Україні – від домашніх геймерів до компаній FTSE – стикаються з проблемою непрацюючих SSD/NVMe-накопичувачів, через які критично важливі дані (фотографії, документи, бази даних, моделі штучного інтелекту) раптово стають недоступними.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1.2 Область застосування

Відновлення даних із сучасних SSD/NVMe дуже відрізняється від відновлення HDD:

- TRIM та збирання сміття – після видалення даних диск активно стирає невикористані блоки → вікно відновлення набагато коротше, ніж у жорстких дисків.

- Шифрування та власні формати – багато накопичувачів NVMe використовують апаратне шифрування або власні таблиці зіставлення, які необхідно реконструювати.

- Необхідне зняття мікросхем у разі фізичних збоїв – коли контролери або мікросхеми NAND виходять з ладу, ми виймаємо мікросхеми та зчитуємо їх безпосередньо – це кваліфікований та трудомісткий процес.

- RAID NVMe-масиви – поширені на серверах в Україні; один несправний диск може вивести весь масив з ладу. Перебудова часто посилює пошкодження.

Профілактика набагато дешевша за одужання. Дотримуйтесь цих найкращих практик на 2026 рік:

1. Впроваджуйте резервне копіювання 3-2-1 – 3 копії, 2 різні типи носіїв, 1 віддалена/хмарна (наприклад, Backblaze, OneDrive, NAS з RAID 1/6).

2. Моніторинг стану диска – використовуйте CrystalDiskInfo (Windows) або smartctl (Linux/Mac) для відстеження атрибутів SMART (рівень зносу, температура, перерозподілені сектори).

3. Уникайте екстремальних температур і проблем з живленням – тримайте сервери/ноутбуки в прохолодному стані; використовуйте якісні блоки живлення/ДБЖ.

4. Обирайте перевірені бренди – Samsung, WD, Crucial, Kingston – уникайте дешевих безіменних накопичувачів для критично важливих даних.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

5. Окремі криптогаманці – використовуйте спеціалізоване обладнання (Ledger/Trezor) або машини з ізоляцією.

6. Регулярні оновлення прошивки (обережно) – лише з офіційних джерел; спочатку створіть резервну копію.

Якщо ж лихо все ж таки трапиться – не намагайтеся використовувати саморобні інструменти або блокувати/відкривати диск. Негайно зверніться до нас, щоб мати найкращі шанси на відновлення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи програмно-визначаємих сховищ для NVMe на базі технології RDDA, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ – 2026

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Технологія SDS початку розвиватися ще на початку 2000-х, але поки не змогла замінити класичні СЗД по цілому ряді причин – зараз ми їх обговорювати не будемо. Але виробники активно займаються розвитком своїх продуктів і інтерес до технологій SDS росте. За нашими оцінками, найближчим часом вони стануть тим інструментом, що дозволить скорочувати вартість ІТ-інфраструктури при росту потреби в збільшенні ємності СЗД.

Прогнозується, що ринок твердотільних накопичувачів NVMe на базі PCIe значно розшириться завдяки зростаючому попиту на високопродуктивні сховища в різних галузях промисловості. Ключовими каталізаторами зростання є зростаючі потреби центрів обробки даних у швидкому доступі до даних у хмарних обчисленнях та аналітиці великих даних, зростання впровадження автономних транспортних засобів та систем керування автопілотом (ADAS), що вимагають швидкої обробки даних, а також зростання використання високопродуктивних обчислень в іграх та професійних робочих станціях. Перехід від SATA до швидших інтерфейсів NVMe, поряд з досягненнями в технології флеш-пам'яті NAND, є ключовими рушійними силами, що збільшують ємність та знижують витрати. Незважаючи на потенційну волатильність ланцюга поставок та коливання цін на флеш-пам'ять NAND, очікується, що постійні інновації та високий попит подолають ці проблеми. Центри обробки даних лідирують за застосуваннями, за ними йдуть автомобільний та клієнтський сегменти, причому впровадження PCIe 4.0 випереджає попередні версії завдяки переважній пропускній здатності.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

Майбутнє ринку NVMe SSD на базі PCIe виглядає багатообіцяючим завдяки широкому впровадженню мереж 5G та експоненціальному зростанню обсягів даних з підключених пристроїв. Новітні технології, такі як CXL (Compute Express Link), підвищують швидкість та ефективність передачі даних, відкриваючи нові програми та сприяючи зростанню ринку. Очікується, що Північна Америка та Азіатсько-Тихоокеанський регіон залишатимуться домінуючими ринками завдяки технологічним інноваціям та високому впровадженню в центрах обробки даних та автомобільному секторі. Жорстка конкуренція між ключовими постачальниками, включаючи Samsung, Kingston та Western Digital, сприятиме інноваціям та знижуватиме ціни, роблячи ці передові рішення більш доступними. Це конкурентне середовище стимулюватиме розробку ще більш продуктивних та економічно ефективних рішень для зберігання даних.

NetApp ONTAP

NetApp ONTAP – це уніфікована платформа зберігання даних, розроблена для забезпечення управління даними корпоративного рівня для локальних середовищ. Вона забезпечує безперебійну масштабованість, високу продуктивність та надійний захист даних, що робить її ідеальною для різноманітних робочих навантажень, включаючи штучний інтелект, аналітику та гібридну хмарну інтеграцію. ONTAP спрощує операції зі зберіганням даних, забезпечуючи доступність та безпеку даних.

Основні характеристики включають:

- Уніфікована платформа зберігання: підтримує блочне, файлове та об'єктне зберігання в рамках єдиної архітектури, що забезпечує багатопроTOCOLний доступ (NFS, SMB, iSCSI та S3) для оптимізації операцій та зменшення складності інфраструктури.
- Масштабована продуктивність: Забезпечує стабільну високошвидкісну продуктивність завдяки конфігураціям NVMe over Fabrics (NVMe-oF) та повністю флеш-пам'яті, що гарантує низьку затримку та високу пропускну здатність для вимогливих робочих навантажень.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

– Інтегрований захист даних: пропонує вбудовані функції, такі як SnapMirror для реплікації, SnapVault для резервного копіювання та захист від програм-вимагачів, для захисту критично важливих даних та забезпечення безперервності бізнесу.

– Оптимізація штучного інтелекту та аналітики: пришвидшує робочі процеси зі штучним інтелектом та аналітикою даних завдяки таким функціям, як FlexCache для доступу до даних з низькою затримкою та FabricPool для автоматизованого перенесення на хмарне або об'єктне сховище.

– Готовність до гібридної хмари: безперешкодна інтеграція з публічними хмарами, що забезпечує мобільність даних та гібридні хмарні робочі процеси завдяки NetApp Cloud Volumes та вбудованим хмарним можливостям ONTAP.

– Розширений захист: забезпечує наскрізне шифрування, контроль доступу на основі ролей та функції забезпечення відповідності для захисту конфіденційних даних та дотримання нормативних вимог.

Dell EMC PowerScale

Dell EMC PowerScale – це масштабована NAS-платформа, розроблена для задоволення потреб у продуктивності, масштабованості та безпеці середовищ зі штучним інтелектом, високопродуктивними обчисленнями та багатохмарних середовищ. Вона забезпечує високошвидкісний доступ до величезних наборів даних, одночасно спрощуючи управління інфраструктурою. Як частина платформи даних Dell AI, вона дозволяє ефективно та безпечно виконувати робочі навантаження, що інтенсивно використовують графічні процесори.

Основні характеристики включають:

– Масштабований NAS для неструктурованих даних: Створений для масштабування, PowerScale забезпечує безперешкодне розширення та підтримує набори даних петабайтного масштабу з інтелектуальним балансуванням навантаження та високою доступністю між вузлами.

– Пропускна здатність, оптимізована для ШІ: забезпечує високу пропускну здатність на одиницю стійки та швидке отримання даних, гарантуючи

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

повне використання серверів GPU та скорочуючи час навчання великих моделей ІІІ.

- Уніфіковане озеро даних: забезпечує універсальний доступ завдяки підтримці кількох протоколів, включаючи NFS, SMB, S3 та HDFS, через єдиний глобальний простір імен, що дозволяє усунути надлишкові копії даних та пришвидшити співпрацю.

- Підтримка GPUDirect та NFSoRDMA: забезпечує пряму високошвидкісну передачу даних між сховищем та пам'яттю графічного процесора для зменшення затримки та оптимізації робочих процесів навчання моделей штучного інтелекту.

- Комплексна архітектура безпеки: захищає конфіденційні дані штучного інтелекту за допомогою вбудованого виявлення програм-вимагачів, принципу нульової довіри.

Qumulo

Qumulo – це програмно-визначена, хмарна платформа для зберігання файлів, призначена для керування неструктурованими даними в локальних, хмарних та гібридних середовищах. Її архітектура підтримує високу масштабованість, аналітику в режимі реального часу та багатопроTOCOLний доступ, що робить її придатною для різноманітних робочих навантажень – від високопродуктивних обчислень до розподіленої співпраці.

Основні характеристики включають:

- Працюйте будь-де, масштабуйте будь-де: підтримує розгортання локально, у хмарі або на периферії, з єдиним простором імен у різних середовищах. Забезпечує стабільну продуктивність та взаємодію з користувачем незалежно від місцезнаходження.

- Хмарна, програмно-визначена архітектура: розділяє обчислення та сховище, забезпечуючи незалежне масштабування ємності та продуктивності.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

– Файлова система екзабайтного масштабу: керує трильйонами файлів в одному просторі імен. Масштабується до сотень петабайт на 264 вузлах без зниження продуктивності чи обмежень inode.

– Прогнозоване кешування за допомогою NeuralCache: відстежує шаблони доступу до даних за допомогою теплових карт і автоматично переміщує часто використовувані дані на швидші рівні сховища (флеш-пам'ять або оперативну пам'ять), оптимізуючи продуктивність читання.

– Підтримка кількох протоколів: Забезпечує вбудований доступ через NFS, SMB та S3 з підтримкою дозволів для кількох протоколів.

– Аналітика в режимі реального часу: інтегрована аналітика забезпечує миттєве відображення використання, продуктивності та місткості без необхідності ручного сканування. Оновлення метаданих відстежуються та агрегуються миттєво.

Cloudian HyperStore

Cloudian HyperStore – це локальна платформа для зберігання об'єктів для підприємств, які керують великими обсягами неструктурованих даних. Вона підтримує ємність у екзабайтному масштабі та оптимізована для робочих процесів з інтенсивною обробкою даних, включаючи штучний інтелект, аналітику та довгострокове архівування. HyperStore є програмно-визначеною, тобто працює на стандартному обладнанні.

Основні характеристики включають:

– Архітектура масштабу екзабайта: Модульна конструкція забезпечує безмежне зростання з розширенням без переривання роботи та масштабованою продуктивністю в одному або кількох місцях розташування.

– Продуктивність, готова до використання зі штучним інтелектом: підтримує високопродуктивні робочі навантаження з низькою затримкою, паралельний доступ через S3 API, інтеграцію з NVIDIA GPUDirect та конфігурацію виключно флеш-пам'яті для сценаріїв використання штучного інтелекту, орієнтованих на дані.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

- Сумісність з API S3: Сумісність з AWS S3, що спрощує інтеграцію, міграцію та гібридну або мультихмарну сумісність.
- Уніфіковане сховище файлів та об'єктів: керує даними файлів та об'єктів на єдиній платформі для спрощення операцій.
- Безпека військового рівня: пропонує наскрізне шифрування, безпечну багатокористувацьку оренду, доступ на основі ролей, багатофакторну автентифікацію та захист від програм-вимагачів за допомогою функцій блокування об'єктів та відповідності вимогам.

DataCore Swarm

DataCore Swarm – це програмно-визначена платформа для зберігання об'єктів для локальних, периферійних та гібридних хмарних розгортань. Вона спрощує керування та захист великомасштабних неструктурованих даних, забезпечуючи високу доступність, лінійну масштабованість та миттєвий доступ через протоколи S3 та HTTP.

Основні характеристики включають:

- Програмно-визначене об'єктне сховище: розгортається на стандартному обладнанні x86 з жорсткими дисками та твердотільними накопичувачами, забезпечуючи до 95% корисної ємності сховища та дозволяючи безперешкодне оновлення обладнання без необхідності оновлення навантажувачами.
- Уніфікована веб-консоль та API: пропонує централізоване керування для моніторингу продуктивності, контролю орендарів, планування потужностей та відстеження контенту для мільярдів об'єктів та тисяч користувачів.
- Самокерована, самовідновлювана архітектура: автоматично обробляє збої, перебалансування та розширення потужності, щоб зменшити операційні витрати та підтримувати масштабованість без перебоїв у роботі.
- Масштабована паралельна проєктування: усі вузли виконують усі функції (доступ до даних, захист та обробку метаданих), що забезпечує лінійне зростання продуктивності в міру додавання вузлів сховища.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

– Потокове передавання контенту з рівня сховища: потокове передавання медіа безпосередньо внутрішнім користувачам або зовнішнім глядачам, що підходить для робочих процесів OTT, VOD та медіа-виробництва без проміжних рівнів обробки.

Локальне сховище: довгострокова цінність та безпека

Локальне сховище залишається стратегічним вибором для організацій із суворими вимогами до контролю, продуктивності або відповідності. Хоча воно вимагає більших інвестицій та внутрішньої експертизи, воно забезпечує неперевершену прозорість, налаштування та безпеку для конфіденційних або високопродуктивних робочих навантажень. У галузях, де локальність даних, управління або затримка є критично важливими, локальні рішення продовжують пропонувати довгострокову цінність та стійкість.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об’єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу “сервер-клієнт”, додатки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувацьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю визнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожную із цих

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – крапку входу додатка – інкапсулюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово `override`, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримує спадкування.

На додаток до основних описаних об'єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

- Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.
- Властивості, що виступають у ролі методів доступу для закритих змінних-членів.
- Атрибути з декларативними метаданими про типи під час виконання.
- Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску системи програмно-визначаємих сховищ для NVMe на базі технології RDDA у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

Крім служб часу виконання, в .NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керування рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному додатку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

забезпечення, яке призначено для системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Концентрація та характеристики твердотільних накопичувачів NVMe на базі PCIe

Ринок NVMe SSD на базі PCIe характеризується високою концентрацією між кількома ключовими гравцями, де Samsung, Western Digital та SK Hynix займають значну частку ринку. Ці компанії вииграють від економії за рахунок масштабу у виробництві та сильної впізнаваності бренду. Інновації зосереджені в таких сферах, як флеш-пам'ять NAND високої щільності, передова технологія контролерів, що забезпечує вищу швидкість та підвищену енергоефективність, а також розробка інтерфейсів PCIe наступного покоління (таких як PCIe 5.0 і новіших):

- Галузі спеціалізації: виробництво флеш-пам'яті NAND, розробка контролерів та рішення для центрів обробки даних.
- Характеристики інновації: Зосередженість на збільшенні ємності сховища (досягнення багатотерабайтних ємностей), покращенні продуктивності (швидкість послідовного читання/запису понад 10 Гбіт/с) та підвищенні надійності (покращення витривалості та виправлення помилок).
- Вплив нормативних актів: Нормативи про конфіденційність даних та галузеві стандарти щодо безпеки та надійності даних стимулюють інновації в можливостях безпечного стирання та функціях забезпечення цілісності даних.
- Замінники продуктів: Традиційні SATA SSD та HDD все ще використовуються, хоча NVMe SSD поступово замінюють їх у чутливих до продуктивності застосуваннях. Новітні технології, такі як SCM (Storage Class Memory), представляють собою потенційну довгострокову заміну.
- Концентрація на кінцевих користувачах: Центри обробки даних є

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

основним сегментом кінцевих користувачів, за ними йдуть клієнтський ринок (настільні комп'ютери, ноутбуки) та автомобільна промисловість.

– Рівень злиттів та поглинань: На ринку спостерігається помірний рівень злиттів та поглинань, головним чином зумовлений компаніями, які прагнуть розширити свій портфель продуктів та отримати доступ до нових технологій.

Тенденції PCIe-базованих NVMe SSD

Ринок твердотільних накопичувачів NVMe на базі PCIe переживає вибухове зростання, і, за прогнозами, до 2033 року щорічне відвантаження досягне десятків мільйонів одиниць. Це зростання зумовлене кількома факторами. Зростаючий попит на швидші рішення для зберігання даних у центрах обробки даних, зумовлений хмарними обчисленнями та аналітикою великих даних, є основним рушієм. Крім того, впровадження в автомобільній промисловості передових систем допомоги водієві (ADAS) та технологій автономного водіння вимагає високопродуктивного та надійного сховища, що створює значний попит на твердотільні накопичувачі NVMe автомобільного класу. Клієнтський ринок також робить значний внесок, зі зростаючим впровадженням твердотільних накопичувачів NVMe у високопродуктивних ноутбуках та настільних комп'ютерах. Перехід до стандартів інтерфейсу PCIe 4.0 та майбутніх стандартів інтерфейсу PCIe 5.0 стимулює подальше зростання, оскільки ці нові стандарти пропонують значно покращену пропускну здатність та продуктивність порівняно з їхніми попередниками. Ключові ринкові дані включають послідовний перехід до накопичувачів більшої ємності та зростаючий інтерес до функцій корпоративного рівня, таких як комплексний захист даних та інструменти управління, оптимізовані для центрів обробки даних. Ринок також спостерігає зростання попиту на спеціалізовані твердотільні накопичувачі NVMe, розроблені для конкретних застосувань, таких як робочі навантаження штучного інтелекту/машинного навчання, високопродуктивні обчислення (HPC) та периферійні обчислення. Компанії також зосереджуються на розробці більш енергоефективних твердотільних накопичувачів NVMe для досягнення цілей

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

сталого розвитку. Зрештою, ціни на твердотільні накопичувачі NVMe продовжують знижуватися, роблячи їх все більш доступними для ширшого кола споживачів та підприємств. Прогнозований період (2025-2033) передбачає сукупний річний темп зростання (CAGR) для твердотільних накопичувачів NVMe на базі PCIe, що перевищує 20%. Протягом історичного періоду (2019-2024) ринок демонстрував значне зростання, перевищивши 10 мільйонів поставок, що створило передумови для очікуваного буму в прогнозований період. До розрахункового року (2026) ринок, за прогнозами, досягне 25 мільйонів одиниць, зі стабільним зростанням, що перевищить 50 мільйонів одиниць до 2033 року.

Сегмент центрів обробки даних готовий домінувати на ринку NVMe SSD на базі PCIe протягом прогнозованого періоду (2025-2033).

– Домінування центрів обробки даних: Масове зростання цього сегмента зумовлене постійно зростаючим попитом на хмарне сховище, аналітику великих даних та високопродуктивні обчислення. Центрам обробки даних потрібні високошвидкісні рішення для зберігання даних високої ємності для обробки величезних обсягів даних, що генеруються та обробляються щодня. Потреба в швидшому отриманні та обробці даних у хмарних середовищах підживлює попит на високопродуктивні твердотільні накопичувачі NVMe на базі PCIe. Масштабне розгортання в гіпермасштабних центрах обробки даних значною мірою сприяє домінуванню сегмента. Прогноз передбачає, що сотні мільйонів одиниць будуть поставлені спеціально в сегмент центрів обробки даних, що значно випереджає інші прикладні сегменти.

– Географічний розподіл: Очікується, що Північна Америка та Азія (особливо Китай та інші країни регіону, що швидко розвиваються) залишатимуться провідними регіонами з впровадження PCIe NVMe SSD завдяки сильній присутності як технологічних гігантів, так і швидкозростаючих центрів обробки даних. Однак значне зростання прогнозується також у Європі та інших регіонах світу. Зростання світового попиту на хмарні послуги та збільшення

інвестицій у цифрову інфраструктуру є ключовими факторами, що сприяють цьому світовому зростанню.

Огляд продуктів: SSD-накопичувачі NVMe на базі PCIe

Ринок пропонує широкий асортимент твердотільних накопичувачів NVMe, класифікованих за ємністю (від сотень гігабайт до кількох терабайт), інтерфейсом (PCIe 2.0, 3.0 та 4.0, з появою PCIe 5.0) та форм-фактором (плати розширення, M.2, U.2). Диференціація продуктів зосереджена на характеристиках продуктивності (швидкість послідовного та випадкового читання/запису), довговічності (записаних терабайтів – TBW), енергоефективності та таких функціях, як шифрування та безпека даних. На ринку спостерігається зростаюча тенденція до накопичувачів більшої ємності та продуктивності з розширеними функціями, що задовольняють зростаючі потреби програм з інтенсивною обробкою даних.

Цей розділ містить комплексний аналіз ринку в різних сегментах:

– Застосування: центри обробки даних, автомобільна промисловість, клієнтська техніка (настільні комп'ютери та ноутбуки) та інші (промислові, мережеві тощо). Сегмент центрів обробки даних займає найбільшу частку ринку завдяки високому попиту на ємність та швидкість зберігання даних, тоді як автомобільний сектор демонструє значне зростання завдяки зростаючому впровадженню систем допомоги водієві (ADAS) та автономного водіння. Клієнтський сегмент продовжує зростати, оскільки споживачі вимагають швидшого сховища для персональних комп'ютерів. Сегмент «інші» включає різноманітні програми з різними вимогами, але всі вони отримують вигоду від підвищення продуктивності завдяки NVMe SSD.

– Типи: PCIe 2.0, PCIe 3.0 та PCIe 4.0. Кожне покоління пропонує значні покращення продуктивності порівняно з попередником. Очікується, що PCIe 4.0, хоча наразі є провідною технологією, зрештою буде замінена PCIe 5.0 і наступними версіями, причому ці новіші покоління пропонуватимуть ще вищі швидкості та збільшену пропускну здатність.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

Рушійні сили: Що рухає SSD-накопичувачі NVMe на базі PCIe

Основними рушійними силами зростання ринку є зростаючий попит на швидше зберігання даних у центрах обробки даних, розширення автомобільного ринку (ADAS та автономне водіння), потреба в покращенні продуктивності клієнтських пристроїв та технологічний прогрес, що призводить до більшої ємності та нижчих витрат.

Проблеми та обмеження в SSD-накопичувачах NVMe на базі PCIe

Серед труднощів – висока початкова вартість NVMe SSD порівняно з традиційними жорсткими дисками та SATA SSD, складність інтеграції та проблеми сумісності між різними системами, а також потенційні обмеження ланцюга поставок під час отримання критично важливих компонентів.

Нові тенденції в NVMe SSD на базі PCIe

Нові тенденції включають впровадження PCIe 5.0 і вище, зростаюче використання технології флеш-пам'яті 3D NAND та розробку більш енергоефективних та екологічних рішень NVMe SSD.

Каталізатори зростання в індустрії NVMe SSD на базі PCIe

Зростання галузі стимулюється розширенням хмарних обчислень, аналітики великих даних, зростаючим використанням штучного інтелекту та машинного навчання, а також розвитком високопродуктивних обчислювальних програм, які потребують швидших рішень для зберігання даних. Постійна мініатюризація компонентів та постійний розвиток більш просунутих технологій контролерів ще більше сприяють зростанню галузі.

3.2 Розробка структурної схеми

Прямий доступ до пам'яті (RDMA) – це прямий доступ до пам'яті з пам'яті одного комп'ютера в пам'ять іншого без залучення операційної системи будь-якого з комп'ютерів. Це дозволяє отримати високопродуктивний доступ до пам'яті з низькою затримкою через мережу, що особливо корисно в масово паралельних комп'ютерних кластерах.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

NVMe-over-Fabrics та сучасний ландшафт SDS

NVMe поверх TCP (NVMe/TCP) тепер впроваджується в основні програмно-визначені сховища, що дозволяє створювати високопродуктивні, масштабовані розгортання для хмарних та корпоративних робочих навантажень. Оскільки середовища обробки даних продовжують розвиватися, багато платформ зберігання даних виходять за рамки однопротокольних конструкцій, зосереджуючись на гнучкості, економічній ефективності та архітектурній адаптивності.

У даній роботі пропонується платформа програмно-визначеного сховища, розроблену для високопродуктивних робочих навантажень. Її модульна, адаптивна, уніфікована архітектура MAUS (shared-everything) підтримує NVMe over Fabrics (NVMe-oF), розгортання на основі Kubernetes та безперешкодну масштабованість у різних середовищах. Така конструкція забезпечує стабільну продуктивність, водночас створюючи відкриту, розподілену основу, яка бездоганно інтегрується із сучасними корпоративними екосистемами.

Основні переваги використання:

- Лінійне масштабування продуктивності: Розширте ємність сховища без шкоди для продуктивності.
- Розширене кодування стирання: Покращений захист даних з ефективним використанням сховища.
- Висока щільність IOPS: оптимізовано для вимогливих робочих навантажень, що потребують швидкого доступу до даних.
- Підтримка ARM64 та x64: Гнучкість для розгортання на різних апаратних платформах.
- Оптимізація DPU: Перенесення завдань зберігання на блоки обробки даних для підвищення ефективності.
- Комплексна підтримка NVMe-oF: забезпечення низької затримки та високої пропускної здатності мережевого сховища.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

- Прозоре для додатків багаторівневе сховище: автоматичне переміщення даних між рівнями сховища без відома додатків.

Що відрізняє даний проєкт від конкурентів у сфері програмно-визначених блочних сховищ

- Гнучкість розгортання: зосереджений на NVMe/TCP та NVMe/RoCE (або NVMe/RDMA) , simplyblock пропонує ширшу підтримку NVMe-oF , що дозволяє адаптуватися до різноманітних мережевих середовищ.

- Економічна ефективність: Зменшення капітальних витрат шляхом використання існуючого обладнання та уникнення прив'язки до певного постачальника.

- Спрощені операції: Оптимізуйте управління завдяки інтеграції з Kubernetes та автоматизованому розподілу на рівні.

- Архітектура, орієнтована на майбутнє: Залишайтеся на крок попереду завдяки підтримці нових технологій, таких як DPU та ARM64.

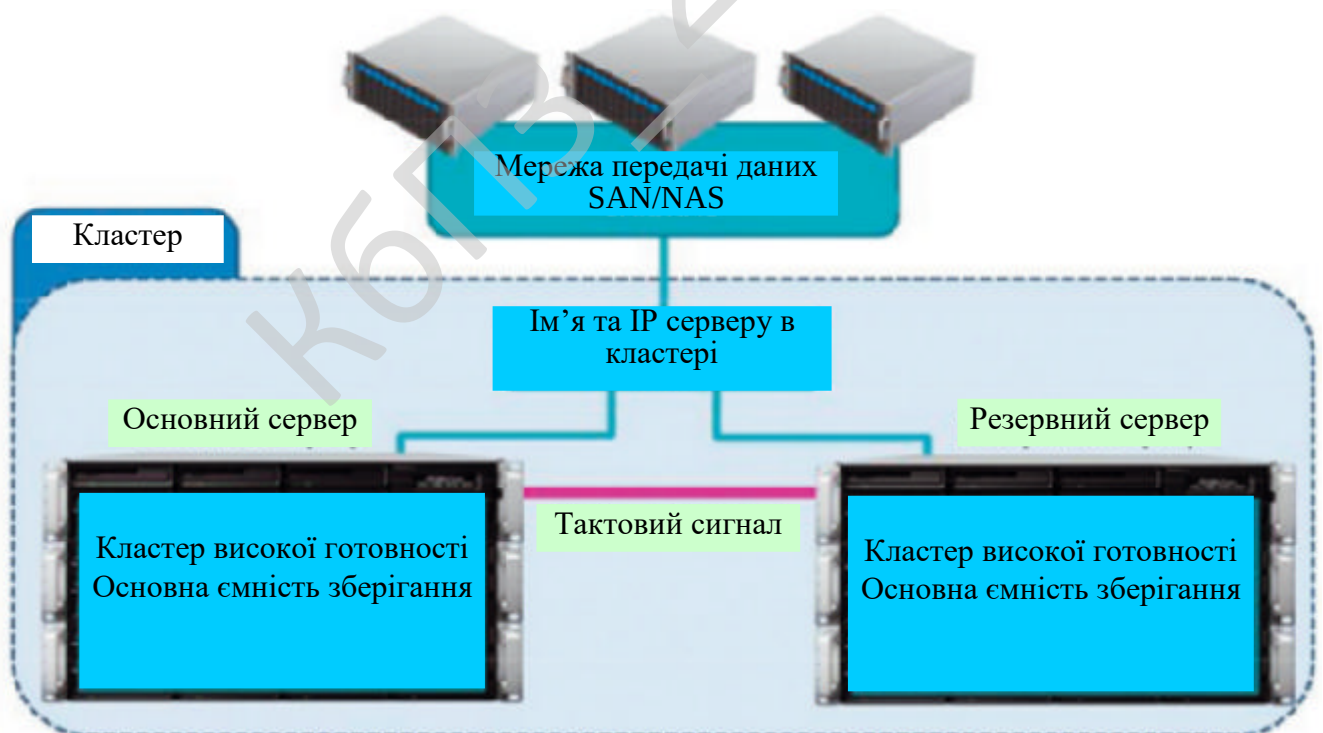


Рисунок 3.1 – Структурна схема системи

Прогнозується, що 2026 рік стане феноменальним для корпоративних систем зберігання даних. Нові узгодження, конвергенція та нестримне зростання змінюють уявлення про інфраструктуру зберігання даних.

Штучний інтелект та корпоративне сховище даних. Тепер вони нерозривно пов'язані. Кібербезпека та корпоративне сховище даних. Ця конвергенція вивела кіберстійке сховище даних на передній план корпоративного ринку. Енергоефективність та корпоративне сховище даних. Енергозбереження стало новою валютою в центрах обробки даних.

Капітальні витрати/операційні витрати та корпоративне сховище даних. Можливості зниження витрат змінюють розмову про інфраструктуру даних. Зростання обсягів даних та корпоративне сховище даних. Одне не процвітає без іншого. У кожній із цих захопливих комбінацій є тенденція, яку варто дослідити.

П'ять головних тенденцій у сфері зберігання даних 2026 року заклали основу для того, що готується до потужного, підсиленого штучним інтелектом та орієнтованого на кібербезпеку року для корпоративних сховищ даних. Ці тенденції отримані на основі широкого внеску клієнтів, партнерів з каналів, аналітиків та партнерів з технічних альянсів у всьому світі. Ось короткий огляд тенденцій у сфері зберігання даних:

- Перехід до штучного інтелекту: подальше зростання кількості застосувань та робочих навантажень на основі штучного інтелекту для прискорення трансформації підприємств
- Стрибок уперед у кібербезпеці: стійкість кіберсховищ із захистом даних наступного покоління для більш проактивних та превентивних заходів безпеки
- Переосмислення капітальних та операційних витрат: постійний тиск на ІТ-бюджети для створення можливостей для корпоративного сховища даних
- Живлення для сховища: зростаючий попит на енергоефективність для зміни структури корпоративних даних
- Зберігати більше даних, більше уніфікованого сховища: нестримне зростання даних змушує поєднувати кібернетичні технології, штучний інтелект, енергетику, капітальні/операційні витрати та резервне копіювання

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

У цьому розділі ви знайдете детальну інформацію та плани щодо тенденцій у сфері корпоративних сховищ даних, щоб ви були готові до дій у новому році.

Перехід до штучного інтелекту

Перехід до штучного інтелекту – подальше зростання кількості застосунків та робочих навантажень на основі штучного інтелекту для прискорення трансформації підприємств

Швидке впровадження штучного інтелекту (ШІ) мало величезний, прямий вплив на інфраструктуру зберігання даних підприємств. Очікується, що у 2026 році розгортання ШІ продовжиться експоненціально та безперервно, оскільки підприємства прагнуть трансформуватися за допомогою ШІ. Gartner прогнозує, що до 2026 року понад 80% підприємств використовуватимуть API ШІ або розгортатимуть програми на базі ШІ, перевівши технології ШІ з пілотної фази в основні операції підприємства.

Цей мегатренд «трансформації штучного інтелекту» сигналізує про перехід до архітектури сховища, орієнтованої на штучний інтелект, високопродуктивних рішень для робочих навантажень штучного інтелекту на основі пошуку даних та накладок штучного інтелекту, які використовують існуючу інфраструктуру для максимізації інвестицій у корпоративне сховище. Планом штучного інтелекту, який має стати «архітектурою штучного інтелекту року» у 2026 році, є архітектура розгортання робочого процесу Retrieval-Augmented Generation (RAG), яка значно підвищує точність та релевантність моделей штучного інтелекту.

Прорив у кібербезпеці

Прорив у кібербезпеці – стійкість кіберсховищ для більш проактивних та превентивних заходів безпеки.

На корпоративному ринку відбувається перехід від залежності від традиційного виявлення та реагування в кібербезпеці до пріоритетності проактивного кіберзахисту та превентивних заходів, щоб зробити інфраструктуру

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		26

зберігання даних справді кібербезпечною. Крім того, загрози штучного інтелекту в кіберпросторі зростають тривожними темпами. Кіберзлочинці стають дедалі витонченішими у використанні штучного інтелекту, щоб зробити кібератак більш небезпечним, підступним та складнішим для виявлення.

Згідно з опитуванням PwC Global Digital Trust Insights за 2026 рік (жовтень 2025 року), 60% керівників бізнесу та технологій відносять інвестиції в кіберризики до трійки своїх головних стратегічних напрямків. Лише 6% відчують впевненість щодо всіх опитаних вразливостей. Крім того, нестача кіберталентів залишається однією з найбільших перешкод для прогресу в кібербезпеці.

Саме тому у 2026 році стійкість та відновлення корпоративних кіберсховищ повинні стати частиною комплексних стратегій кібербезпеки всіх підприємств. Яскравим прикладом проактивного підходу є використання можливості кібервиявлення, вбудованої в первинне сховище. Ключовим прикладом превентивного підходу є використання автоматизованої можливості кіберзахисту для автоматизації створення незмінних знімків даних для пришвидшення відновлення після атаки.

Переосмисліть капітальні та операційні витрати

Переосмисліть капітальні та операційні витрати – Постійний тиск на ІТ-бюджети для створення можливостей для корпоративного зберігання даних

Інформаційні технології є основоположними для всіх підприємств у світі. Технології більше не є просто дискреційними. Іншими словами, ІТ-бюджети залишатимуться важливими. Gartner прогнозував раніше цього року, що до кінця 2026 року очікується, що світові витрати на ІТ становитимуть 5,43 трильйона доларів, що на 7,9% більше, ніж у 2025 році. Прогнозується, що світові витрати на ІТ досягнуть 6,08 трильйона доларів у 2026 році, що на 9,8% більше, ніж у 2025 році, значною мірою завдяки значним інвестиціям в інфраструктуру, програмне забезпечення та пристрої штучного інтелекту. Однак у 2026 році тиск щодо підвищення економічної ефективності ІТ та перерозподілу бюджету з

традиційної інфраструктури на штучний інтелект та інші новітні технології продовжуватиме зростати.

Корпоративне сховище даних має вирішальне значення для зменшення капітальних та операційних витрат шляхом консолідації, підвищення енергоефективності та збільшення місткості високоякісних корпоративних сховищ у менших форм-факторах. Звільнивши кошти з бюджетів на сховища, підприємства зможуть у 2026 році створювати та краще фінансувати нові інноваційні проекти, включаючи агентний штучний інтелект. Хоча ІТ-бюджети можуть залишатися стабільними або здебільшого незначно зростати на однозначні числа, мантра «робити більше з меншими витратами» є наріжним каменем нового тиску на стримування ІТ-витрат.

Живлення для сховища даних

Живлення для сховища даних – Зростаючий попит на енергоефективність для зміни форми інфраструктури даних

Потреба в більшій потужності в центрах обробки даних продовжуватиме зростати у 2026 році, що призведе до нових вимог щодо перерозподілу та звільнення джерел живлення для енергоємних робочих навантажень та програм штучного інтелекту. Штучний інтелект не тільки є основним фактором збільшення споживання енергії, але й технології, що інтенсивно працюють з даними та не пов'язані зі штучним інтелектом, також потребують більше енергії. За даними 451 Research, до кінця 2025 року центри обробки даних у США потребуватимуть на 22% більше потужності мережі, ніж рік тому, і ця тенденція продовжиться і в новому році. У 2026 році попит на ІТ-обладнання в центрах обробки даних лише в США зросте до 75,8 ГВт. У довгостроковій перспективі, за прогнозами, до 2030 року центрам обробки даних знадобиться майже втричі більше енергії, ніж у 2024 році.

У 2026 році підприємства прагнуть заощадити енергію, зменшуючи її використання в інфраструктурі даних підприємства та в інших місцях. Це вимагатиме впровадження більш енергоефективних систем зберігання даних та

пов'язаних з ними технологій, що підживлюють інфраструктуру даних. Оскільки технології більше не є дискреційним явищем для підприємств, потреба в кращому управлінні живленням ще ніколи не була такою важливою.

Зберігати більше даних, більше уніфікованого сховища

Зберігати більше даних, більше уніфікованого сховища – Нестримне зростання даних змушує поєднувати кібербезпеку, штучний інтелект, енергетику, капітальні/операційні витрати та резервне копіювання

Очікуйте нестримного зростання обсягу даних у 2026 році. За даними Statista, світовий обсяг даних у 2024 році становив приблизно 149 зеттабайтів. До кінця 2026 року світ перевищить 181 зеттабайт даних. Очікується, що до 2026 року обсяг створених даних досягне 230–240 зеттабайтів. Величезні стрибки в обсягах цифрової інформації, безумовно, вимагатимуть значного зростання ємності сховищ. Ця тенденція пов'язана з чотирма іншими тенденціями зберігання даних 2026 року – штучним інтелектом, кібербезпекою, капітальними/операційними витратами та енергоспоживанням. Але вона також вводить п'ятий вимір, який не можна ігнорувати – з усім цим зростанням даних, яке очікується протягом наступного року, виникає потреба в більшому резервному копіюванні та захисті даних. Нестримне зростання даних призведе до нестримного резервного копіювання.

Резервне копіювання зазвичай уповільнює продуктивність програм і робочих навантажень, незалежно від їхньої складності. Якщо система резервного копіювання швидка, то вплив на основні робочі навантаження та програми менший. Коли вам потрібно відновити дані – а дані, очевидно, є життєво важливою частиною кожного підприємства – відновлення даних має бути якомога швидшим. У 2026 році все більше організацій вживатимуть заходів для задоволення потреби в майже миттєвому запуску та роботі сховищ резервних копій.

Що таке локальні рішення для зберігання даних?

Локальні рішення для зберігання даних передбачають розміщення та керування даними на фізичному обладнанні, розташованому у власних приміщеннях організації або в спільному розміщенні. Такий підхід забезпечує прямий контроль над даними, обладнанням та безпекою, пропонуючи передбачувану продуктивність та меншу затримку для критично важливих програм і конфіденційних даних. Однак він вимагає значних початкових інвестицій, постійного обслуговування та спеціалізованого ІТ-персоналу для обслуговування апаратного та програмного забезпечення, а масштабованість обмежена фізичною інфраструктурою та оновленням обладнання.

Ці рішення зазвичай обслуговують підприємства з критично важливими потребами в управлінні даними, нормативних актах або затримках. Такі сектори, як охорона здоров'я, фінанси та уряд, вважають локальне сховище привабливим, оскільки воно зменшує ризики, пов'язані із зовнішнім розміщенням даних, та відповідає суворим вимогам дотримання нормативних вимог.

Хоча початкове розгортання пов'язане з вищими витратами та складністю, багато організацій обирають локальне сховище, коли суверенітет даних, налаштування та спеціалізована продуктивність не підлягають обговоренню.

Переваги локальних рішень для зберігання даних

Локальне сховище пропонує низку переваг для організацій, які надають пріоритет контролю, продуктивності та відповідності вимогам. Хоча воно вимагає більше власних ресурсів, переваги можуть переважувати компроміси для багатьох підприємств, особливо тих, які працюють у регульованому або високобезпечному середовищі.

Ключові переваги включають:

– Повний контроль над інфраструктурою: організації мають повний контроль над апаратним забезпеченням, програмним забезпеченням та конфігурацією. Це дозволяє проводити індивідуальну оптимізацію для задоволення конкретних операційних або продуктивних потреб.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

– Підвищена безпека даних: дані залишаються в межах організації, що зменшує ризик порушень з боку третіх осіб. Політики безпеки можна застосовувати безпосередньо, без залежності від зовнішніх постачальників.

– Відповідність нормативним вимогам: Локальні рішення спрощують виконання галузевих вимог, які передбачають локалізацію даних, журнали аудиту або обмежений доступ.

– Низька затримка та висока продуктивність: доступ до даних та їх обробка відбуваються локально, що може значно зменшити затримку та підвищити продуктивність програм, що важливо для робочих навантажень, чутливих до часу.

– Налаштування та інтеграція: Системи можна налаштувати для інтеграції зі застарілою інфраструктурою, спеціалізованими програмами або унікальними бізнес-процесами.

– Передбачувані витрати: Хоча початкові інвестиції є високими, довгострокові експлуатаційні витрати можуть бути більш передбачуваними та часто нижчими порівняно зі змінною платою за підписку на хмарні послуги.

– Доступ офлайн та безперервність: Локальні середовища можуть продовжувати працювати незалежно від наявності Інтернету, забезпечуючи доступ під час перебоїв або проблем із підключенням.

Варіанти використання локальних рішень для зберігання даних

Суверенітет даних

У країнах зі суворими законами про місцезнаходження даних зберігання інформації в іноземних юрисдикціях може призвести до юридичних санкцій або порушень вимог. Локальне зберігання дозволяє організаціям зберігати повний юрисдикційний контроль над даними, гарантуючи, що вони ніколи не залишають фізичного місцезнаходження організації. Це важливо в таких секторах, як оборона, охорона здоров'я та правоохоронні органи, де дані засекречені або підлягають внутрішньому нагляду. Наприклад, GDPR ЄС або PIPEDA Канади

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		31

можуть обмежувати транскордонну передачу даних, що робить локальні рішення єдиним варіантом, що відповідає вимогам, для багатьох підприємств.

Крім того, локальні системи спрощують аудит та локалізацію даних, пропонуючи прозорі та перевірені методи зберігання. ІТ-команди можуть точно продемонструвати, де знаходяться дані та як вони захищені, що часто є вимогою під час юридичних або регуляторних перевірок.

Захист конфіденційних даних

Локальне сховище дозволяє підприємствам впроваджувати детальні політики безпеки, адаптовані до їхнього профілю ризику. Це включає власні схеми шифрування, сегментацію мережі, обмежений фізичний доступ і багатофакторну автентифікацію – все це керується внутрішньо без доступу третіх сторін. Галузі, які обробляють власні або секретні дані, такі як фармацевтичні дослідження, аерокосмічна промисловість або юридичні фірми, отримують вигоду від мінімізації поверхонь атаки та уникнення ризиків спільного користування, що зустрічаються в моделях публічної хмари.

Крім того, локальні середовища пропонують повну видимість подій безпеки. Організації можуть розгортати передові системи виявлення вторгнень (IDS), інструменти реєстрації та платформи SIEM, які тісно інтегруються з внутрішніми робочими процесами моніторингу. Така глибина контролю допомагає запобігти витоку даних і розширює можливості реагування на інциденти.

Високошвидкісний локальний доступ

Програми, що генерують або споживають великі обсяги даних, такі як системи CAD/CAM, ферми відеорендерингу або транзакційні бази даних, потребують сховища з низькою затримкою та високою пропускнуою здатністю. Локальні налаштування усувають залежність від пропускнуої здатності Інтернету, забезпечуючи гігабітну або навіть 100GbE внутрішню мережу, з якою хмарні платформи не можуть зрівнятися без преміум-сервісів.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

Наприклад, анімаційна студія, яка працює з відеоматеріалами 8K, або фінансова фірма, що використовує алгоритми високочастотної торгівлі, зазнає значного зниження продуктивності, якщо буде змушена покладатися на віддалений доступ до даних. Завдяки локальним масивам сховищ, оптимізованим за пропускну здатністю та кількістю IOPS, організації можуть досягти стабільного, майже миттєвого доступу до критично важливих даних.

Периферійні обчислення та локальна обробка

У розподілених або віддалених середовищах, таких як нафтові платформи, гірничодобувні підприємства, виробничі заводи або військові форпости, локальне сховище підтримує локалізовану обчислювальну інфраструктуру. Це дозволяє отримувати дані в режимі реального часу, аналізувати їх та приймати рішення без вимог до затримки або підключення, характерних для централізованих хмарних моделей.

Периферійні системи з локальним сховищем можуть буферизувати дані під час збоїв, застосовувати висновки на основі штучного інтелекту/машинного навчання у джерелі або керувати виконавчими механізмами та датчиками за мілісекунди. Наприклад, системи прогнозного обслуговування на заводі можуть аналізувати дані датчиків на місці, щоб запобігти виходу з ладу обладнання, уникаючи затримок, які можуть виникнути внаслідок передачі даних до хмари та з неї.

Інтеграція зі старими системами

Багато підприємств залежать від застарілих програм, які потребують специфічних операційних середовищ, форматів сховищ або апаратних інтерфейсів, що не підтримуються сучасними хмарними платформами. Локальне сховище пропонує гнучкість для підтримки цих систем без значних перезаписів або міграції даних. Сховище можна налаштувати для використання протоколів, таких як NFS, SMB або Fibre Channel, що підтримує інтеграцію зі старішими робочими навантаженнями або власним програмним забезпеченням.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

Це особливо цінно для підприємств з давніми системами ERP, платформами SCADA або вертикальними додатками. Розміщуючи сховище разом з існуючою інфраструктурою, підприємства можуть підтримувати безперервність роботи, поступово переходячи на сучасні системи, уникаючи ризику та витрат на повномасштабну модернізацію.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана на дисках систем програмно-визначаємих сховищ для NVMe на базі технології RDDA складається з наступних основних функціональних блоків:

- сам носій інформації – система програмно-визначаємих сховищ для NVMe на базі технології RDDA;
- кодер Ріда-Соломона, який використовується, коли відбувається запис інформації на диск системи програмно-визначаємих сховищ для NVMe на базі технології RDDA, при цьому необхідно враховувати, що об'єм інформації, яка записується на диск системи програмно-визначаємих сховищ для NVMe на базі технології RDDA, повинна бути меншою, ніж об'єм диску системи програмно-визначаємих сховищ для NVMe на базі технології RDDA, в зв'язку, з тим, що коди Ріда-Соломона відносяться до кодів з надмірністю, за рахунок якої й відбувається кодування;
- декодер Ріда-Соломона, який використовується при читанні даних з відповідного диску системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

Функціонально блок, який утримує кодер Ріда-Соломона включає в себе наступні блоки:

- дані, які потрібно захистити;
- поліном для кодування;
- відмовостійки коди.

Коди Ріда-Соломона базуються на спеціальному розділі математики – полях Галуа (GF) або кінцевих полях. Арифметичні дії (+, -, x, / і т.д.) над елементами кінцевого поля дають результат, що також є елементом цього поля. Кодер та декодер Ріда-Соломона повинні вміти виконувати ці арифметичні операції. Ці операції для своєї реалізації вимагають спеціального устаткування або спеціалізованого програмного забезпечення.

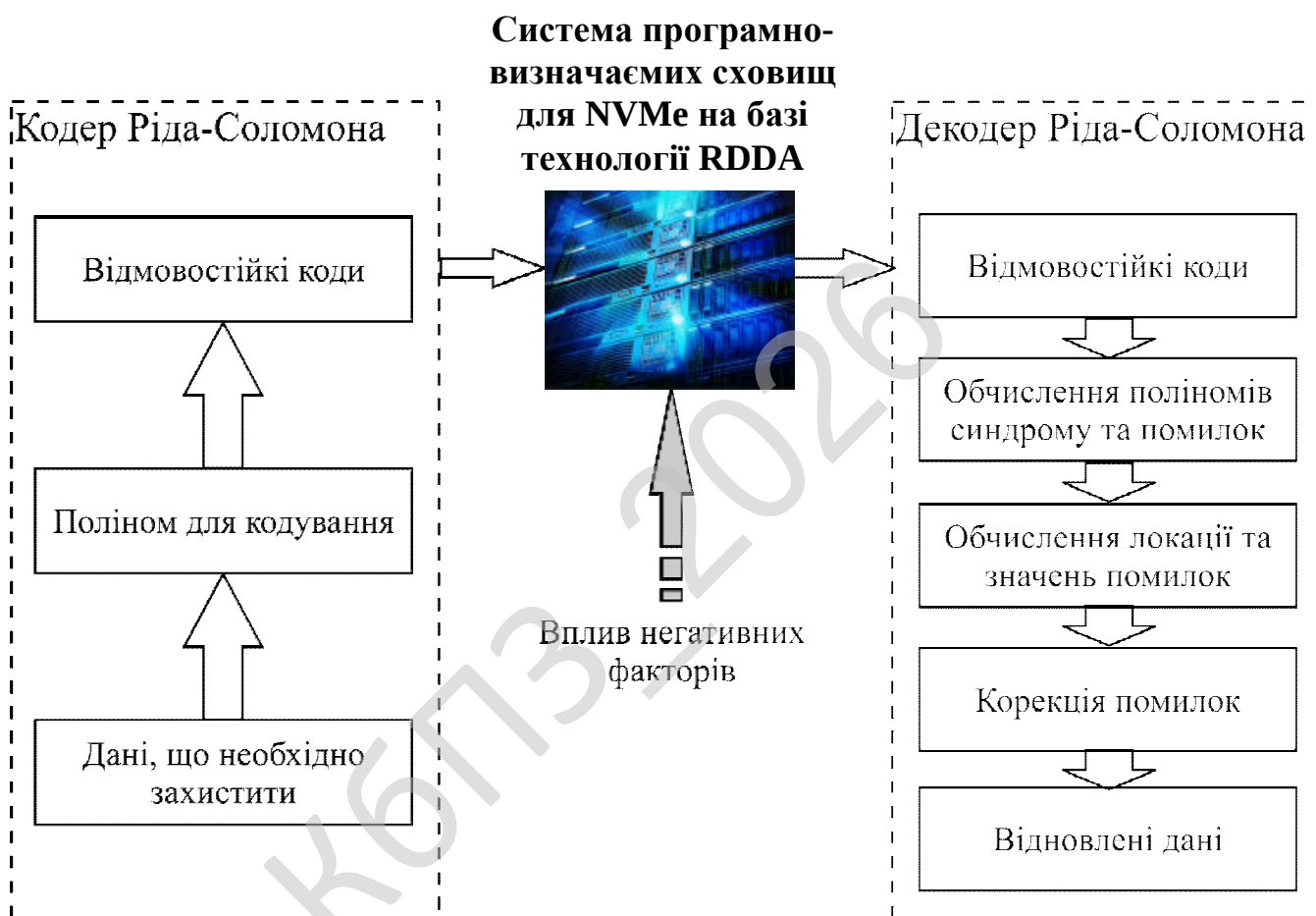


Рисунок 3.2 – Функціональна схема системи

Кодове слово Ріда-Соломона формується із залученням спеціального полінома. Всі коректні кодові слова повинні ділитися без залишку на ці утворюючі поліноми. Загальна форма утворюючого полінома має вигляд:

$$g(x) = (x - a^i)(x - a^{i+1}) \dots (x - a^{i+2t}),$$

а кодове слово формується за допомогою операції:

$$c(x) = g(x) \cdot i(x),$$

де $g(x)$ є утворюючим поліномом, $i(x)$ являє собою інформаційний блок, $c(x)$ – кодове слово, що називається простим елементом поля.

$2t$ символів парності в кодовому слові Ріда-Соломона визначаються з наступного співвідношення:

$$p(x) = i(x) \cdot x^{n-k} \bmod g(x)$$

Перейдемо до розгляду іншого функціонального блоку – декодеру Ріда-Соломона.

Цей функціональний блок містить у собі наступні блоки:

- відмовостійкий код;
- блок обчислення поліномів синдрому та помилок;
- блок обчислення локації та значень помилок;
- блок корекції помилок;
- відновлені за допомогою коді Ріда-Соломона дані.

Декодер працює наступним чином.

Введемо позначення:

- $r(x)$ – Отримане кодове слово.
- S_i – Синдроми.
- $L(x)$ – поліном локації помилок.
- X_i – положення помилок.
- Y_i – Значення помилок.
- $c(x)$ – Відновлене кодове слово.
- v – Число помилок.

Отримане кодове слово $r(x)$ являє собою вихідне (передане) кодове слово $c(x)$ плюс помилки: $r(x) = c(x) + e(x)$.

Декодер Ріда-Соломона намагається визначити позицію й значення помилки для числа t помилок (або $2t$ втрат) і виправити помилки й втрати.

Обчислення синдрому

Обчислення синдрому схоже на обчислення парності. Кодове слово Ріда-Соломона має $2t$ **синдромів**, це залежить тільки від помилок (а не переданих

кодів слів). Синдроми можуть бути обчислені шляхом підстановки $2t$ коріння утворюючого полінома $g(x)$ в $r(x)$.

Знаходження позицій символічних помилок

Це робиться шляхом рішення системи рівнянь із t невідомими. Існує кілька швидких алгоритмів для рішення цього завдання. Ці алгоритми використовують особливості структури матриці кодів Ріда-Соломона й сильно скорочують необхідну обчислювальну потужність. Робиться це у два етапи:

1. Визначення полінома локації помилок. Це може бути зроблене за допомогою алгоритму Berlekamp-Massey або алгоритму Евкліда. Алгоритм Евкліда використовується частіше на практиці, тому що його легше реалізувати, однак, алгоритм Berlekamp-Massey дозволяє одержати більш ефективну реалізацію встаткування й програм.

2. Знаходження кореня цього полінома. Це робиться із залученням алгоритму пошуку Chien.

Знаходження значень символічних помилок

Тут також потрібно вирішити систему рівнянь із t невідомими. Для рішення використовується швидкий алгоритм Forney.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Опис коду Ріда-Соломона

Коди Ріда-Соломона – недвійкові циклічні коди, що дозволяють виправляти помилки в блоках даних. Елементами кодового вектора є не біти, а групи біт (блоки). Дуже поширені коди Ріда-Соломона, що працюють із байтами (октетами).

У цей час широко використовується в системах відновлення даних з компакт-дисків, при створенні архівів з інформацією для відновлення у випадку ушкоджень, у завадостійкому кодуванні. Код Ріда-Соломона був винайдений в 1960 році співробітниками лабораторії Лінкольна Массачусетського технологічного інституту Ірвином Рідом і Густавом Соломоном. Ідея

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

використання цього коду була представлена в статті «Polynomial Codes over Certain Finite Fields». Перше застосування код Ріда-Соломона одержав в 1982 році в серійному випуску компакт-дисків. Ефективний алгоритм декодування був запропонований в 1969 році Елвином Берлекемпом і Джеймсом Мессі.

Коди Ріда-Соломона є важливим частковим випадком БЧХ-коду, корінь полінома, що породжує, якого лежать у тім же полі, над яким і будується код ($m = 1$).

Коди Боуза-Чоудхури-Хоквінгхема (БЧХ коди) – у теорії кодування це широкий клас циклічних кодів, застосовуваних для захисту інформації від помилок. Відрізняється можливістю побудови коду із заздалегідь певними коригувальними властивостями, а саме, мінімальною кодовою відстанню.

Поліном, що породжує

Визначення поліномом, що породжує, циклічного (n,k) коду C називається такий ненульовий $g(x) = \sum_{i=0}^r g_i x^i$ поліном з C , ступінь якого найменша й коефіцієнт при старшому ступені $g_r = 1$.

Теорема 3.1

Якщо C – циклічний (n,k) код і $g(x)$ – його поліном, що породжує, тоді ступінь $g(x)$ дорівнює $r = n - k$ і кожне кодове слово може бути єдиним чином представлено у вигляді $c(x) = t(x)g(x)$, де ступінь $t(x)$ менше або дорівнює $k - 1$.

Теорема 3.2

$g(x)$ – поліном, що породжує, циклічного (n,k) коду є дільником двочлена $x^n - 1$

Наслідок: у такий спосіб як поліном, що породжує, можна вибирати будь-який поліном, дільник $x^n - 1$. Ступінь обраного полінома буде визначати кількість перевірочних символів r , число інформаційних символів $k = n - r$.

Матриця, що породжує

Поліноми $g(x), xg(x), x^2g(x), \dots, x^{k-1}g(x)$ лінійно незалежні, інакше $t(x)g(x) = 0$ при ненульовому $t(x)$, що неможливо.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

Значить кодові слова можна записувати, як і для лінійних кодів, наступним чином:

$$\bar{m}G = (m_0, m_1, \dots, m_{k-1}) \begin{bmatrix} g(x) \\ xg(x) \\ \dots \\ x^{k-1}g(x) \end{bmatrix} = m(x)g(x), \quad (3.1)$$

де G є матрицею, що породжує, $m(x)$ – інформаційним поліномом.

Матрицю G можна записати в символьній формі:

$$G = \begin{bmatrix} g_0 & g_1 & \dots & g_{r-1} & g_r & 0 & \dots & 0 \\ 0 & g_0 & \dots & g_{r-2} & g_{r-1} & g_r & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & g_0 & g_1 & \dots & g_r \end{bmatrix}$$

Перевірочна матриця

Для кожного кодового слова циклічного коду справедливо $c(x) = 0 \pmod{g(x)}$. Тому перевірочну матрицю можна записати як $H = [1 \ x \ x^2 \ \dots \ x^{n-2} \ x^{n-1}] \pmod{g(x)}$.

Тоді:

$$\bar{c}H^T = \sum_{i=0}^{n-1} c_i x^i \pmod{g(x)}. \quad (3.2)$$

Нехай α – елемент поля $GF(q)$ порядку n . Якщо α – примітивний елемент, то його порядок дорівнює $q-1$, т.е. $\alpha^{q-1}=1$, $\alpha^i \neq 1$, $0 < i < q-1$.

Тоді нормований поліном $g(x)$ мінімального ступеня над полем $GF(q)$, коріннями якого є $d-1$ підряд, що йдуть ступенів, $\alpha^{l_0}, \alpha^{l_0+1}, \dots, \alpha^{l_0+d-1}$ елемента α , є поліномом, що породжує, коду над полем $GF(q)$ $g(x) = (x - \alpha^{l_0})(x - \alpha^{l_0+1}) \dots (x - \alpha^{l_0+d-1})$., де l_0 – деяке ціле число (у тому числі 0 і 1), за допомогою якого іноді вдається спростити кодер. Звичайно покладається $l_0 = 1$. Ступінь багаточлена $g(x)$ дорівнює $d-1$.

Довжина отриманого коду n , мінімальна відстань d (мінімальна відстань d лінійного коду є мінімальним із всіх відстаней Хеммінга всіх пар кодових слів).

Код містить $r = d - 1 = \deg(g(x))$ перевірочний символ, де $\deg()$ позначає ступінь полінома; число інформаційних символів $k = n - r = n - d + 1$. У такий спосіб $d = n - k - 1$ і код Ріда-Соломона є роздільним кодом з максимальною відстанню (є оптимальним у змісті границі Синглтона).

Кодовий поліном $c(x)$ може бути отриманий з інформаційного полінома $m(x)$, $\deg m(x) \leq k - 1$, шляхом перемножування $m(x)$ і $g(x)$: $c(x) = m(x)g(x)$

Властивості

Код Ріда-Соломона над $GF(q^m)$, що виправляє t помилок, вимагає $2t$ перевірочних символів і з його допомогою виправляються довільні пакети довжиною t і менше. Відповідно до теореми про границю Рейгера, коди Ріда-Соломона є оптимальними з погляду співвідношення довжини пакета й можливості виправлення помилок – використовуючи $2t$ додаткових перевірочних символів виправляються t помилок (і менш).

Теорема (границя Рейгера). Кожний лінійний блоковий код, що виправляє всі пакети довжиною t і менш, повинен містити щонайменше $2t$ перевірочних символів.

Виправлення багаторазових помилок

Код Ріда-Соломона є одним з найбільш потужних кодів, що виправляють багаторазові пакети помилок. Застосовується в каналах, де пакети помилок можуть утворюватися настільки часто, що їх уже не можна виправляти за допомогою кодів, що виправляють одиночні помилки. $(q^m - 1, q^m - 1 - 2t)$ -код Ріда-Соломона над полем $GF(q^m)$ з кодовою відстанню $d = 2t + 1$ можна розглядати як $((q^m - 1)m, (q^m - 1 - 2t)m)$ -код над полем $GF(q)$, що може виправляти будь-яку комбінацію помилок, зосереджену в t або меншому числі блоків з m символів.

Найбільше число блоків довжини m , які може торкнути пакет довжини l_i , де $l_i \leq mt_i - (m - 1)$, не перевершує t_i , тому код, що може виправити t блоків помилок, завжди може виправити й будь-яку комбінацію з p пакетів загальної довжини l , якщо $l + (m - 1) \leq mt$.

Практична реалізація

Кодування за допомогою коду Ріда-Соломона може бути реалізовано двома способами: систематичним і несистематичним.

При несистематичному кодуванні інформаційне слово множиться на якийсь поліном, що неприводиться, у полі Галуа. Отримане закодоване слово повністю відрізняється від вихідного й для добування інформаційного слова потрібно виконати операцію декодування й уже потім можна перевірити дані на зміст помилок. Таке кодування вимагає більші витрати ресурсів тільки на добування інформаційних даних, при цьому вони можуть бути без помилок.

При систематичному кодуванні до інформаційного блоку з k символів приписуються $2t$ перевірочних символів, при обчисленні кожного перевірочного символу використовуються всі k символів вихідного блоку.

У цьому випадку немає витрат ресурсів при добуванні вихідного блоку, якщо інформаційне слово не містить помилок, але кодер/декодер повинен виконати $k(n - k)$ операцій додавання й множення для генерації перевірочних символів. Крім того, тому що всі операції проводяться в поле Галуа, те самі операції кодування/декодування вимагають багато ресурсів і часу. Швидкий алгоритм декодування, заснований на швидкому перетворенні Фур'є, виконується за час порядку $\ln n^2$.

Кодування

При операції кодування інформаційний поліном множиться на багаточлен, що породжує. Множення вихідного слова S довжини k на не приводиться поліном, що, при систематичному кодуванні можна виконати в такий спосіб:

- До вихідного слова приписуються $2t$ нулів, виходить поліном $T = Sx^{2t}$.
- Цей поліном ділиться на поліном, що породжує, G , перебуває залишок R , $Sx^{2t} = QG + R$, де Q – частка.
- Цей залишок й буде коригувальним кодом Ріда-Соломона, він приписується до вихідного блоку символів. Отримане кодове слово $C = Sx^{2t} + R$.

Кодер будується зі регістрів зсуву, суматорів і перемножувачів. Регістр

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

зсуву складається з комірок пам'яті, у кожній з яких перебуває один елемент поля Галуа.

Наведений як приклад кодер Ріда-Соломона генерує 16 коригувальних байт, що дозволяє виправляти до 8 і виявляти до 16 помилок у кадрі даних.

Перемножувачі на константи $GF(0)...GF(15)$ у полі Галуа реалізуються в такий спосіб: спочатку вихідне число й константа перетворюються в індексну форму, потім складаються в межах байта без обліку переносу.

Результатом операції є результат додавання, перетворена обернено в поліноміальну форму.

При переході від однієї форми подання даних до іншої доцільно використовувати таблицю істинності розміром 256 байт, що становить ємність одного ЕАВ (Embedded Array Block – блок зосередженої пам'яті). Для реалізації кодеру потрібно 16 таких перемножувачів, при цьому те саме число множиться на різні константи, що дозволяє використовувати для його перекладу в індексну форму один ЕАВ.

Для перекладу результатів у поліноміальну форму потрібно вже 16 таких таблиць, що вимагає застосування ІМС FPGA дуже великої ємності. У запропонованій схемі використовується тактування кодера із частотою, в 8 разів перевищуючу частоту надходження байт даних.

Це дає можливість використовувати дві пари «суматор – ЕАВ», мультиплексуючи константи на входах суматорів і дозволяючи роботу регістрів-накопичувачів у моменти появи відповідних даних на виходах засувки ЕАВ.

На структурній схемі кодера (рисунок 3.2) символу «С» відповідають дві константи $GF(n)$.

Символ «L» у логіці регістрів-накопичувачів відповідає наступний: вихід компаратора нуля (символи CMP0 і SYNC) дозволяє роботу схеми «АБО що виключає », на входи якої подаються вихід попереднього регістра й ЕАВ. Якщо ж вектор зворотного зв'язку дорівнює "0", схема пропускає дані з виходу попереднього регістра-накопичувача на вхід наступного.

У результаті кодер з урахуванням схеми синхронізації (на рисунку не показана) займає 255 LE (Logic Element – логічний елемент) і 3 ЕАВ, що дозволяє розмістити його в ІМС EPF10K10. Після оптимізації розміщення схеми на кристалі FPGA швидкодія схеми досяглася 11,57 МГц (частота надходження байт даних, далі – байтова частота).

При використанні ІМС EPF10K20, у складі якої 6 ЕАВ, використовуючи 4 пари "суматор – ЕАВ", можна тактувати кодер із частотами, що перевищують байтову частоту не в 8, а в 4 рази, що дозволить підняти її до 25...30 МГц.

Декодування

Декодер, що працює по авторегресивному спектральному методі декодування, послідовно виконує наступні дії:

- Обчислює синдром помилки.
- Будує поліном помилки.
- Знаходить корінь даного полінома.
- Визначає характер помилки.
- Виправляє помилки.

Обчислення синдрому помилки

Обчислення синдрому помилки виконується синдромним декодером, що ділить кодове слово на багаточлен, що породжує. Якщо при діленні виникає остача, то в слові є помилка. Остача від ділення є синдромом помилки.

Побудова полінома помилки

Обчислений синдром помилки не вказує на положення помилок. Ступінь полінома синдрому дорівнює $2t$, що багато менше ступеня кодового слова n . Для одержання відповідності між помилкою і її положенням у повідомленні будується поліном помилок.

Поліном помилок реалізується за допомогою алгоритму Берлекемпа-Месси, або за допомогою алгоритму Евкліда. Алгоритм Евкліда має просту реалізацію, але вимагає більших витрат ресурсів. Тому частіше застосовується більше складний, але менш затратоємний алгоритм Берлекемпа-Месси.

Коефіцієнти знайденого полінома безпосередньо відповідають коефіцієнтам помилкових символів у кодовому слові.

Алгоритм Евкліда виконується наступним чином.

Нехай a і b суть цілі числа, не рівні одночасно нулю, і послідовність чисел $a, b, r_1 > r_2 > r_3 > r_4 > \dots > r_n$ визначена тим, що кожне r_k це остача від ділення перед-попереднього числа на попереднє, а передостаннє ділиться на останнє націло, тобто

$$\begin{aligned} a &= bq_0 + r_1 \\ b &= r_1q_1 + r_2 \\ r_1 &= r_2q_2 + r_3 \\ &\dots \\ r_{n-1} &= r_nq_n \end{aligned} \tag{3.3}$$

Тоді (a,b) , найбільший загальний дільник a і b , дорівнює r_n , останньому ненульовому члену цієї послідовності.

Існування таких $r_1, r_2, \dots$, тобто можливість ділення з остачею m на n для будь-якого цілого m і цілого $n \neq 0$, доводиться індукцією по m .

Коректність цього алгоритму впливає з наступних двох тверджень:

- Нехай $a = bq + r$, тоді $(a,b) = (b,r)$.
- $(0,r) = r$. для будь-якого ненульового r .

Знаходження корню

На цьому етапі шукаються коріння полінома помилки, що визначають положення перекручених символів у кодовому слові. Реалізується за допомогою процедури Ченя, рівносильній повному перебору. У поліном помилок послідовно підставляються всі можливі значення, коли поліном звертається в нуль – коріння знайдені.

Визначення характеру помилки і її виправлення

По синдрому помилки й знайдених корінь полінома за допомогою алгоритму Форні визначається характер помилки й будується маска перекручених символів. Ця маска накладається на кодове слово за допомогою операції XOR і

перекручені символи відновлюються. Після цього відкидаються перевірочні символи й виходить відновлене інформаційне слово.

Застосування

У даний момент коди Ріда-Соломона мають дуже широку область застосування завдяки їхній здатності знаходити й виправляти багаторазові пакети помилок.

Запис і зберігання інформації

Код Ріда-Соломона використовується при записі й читанні в контролерах оперативної пам'яті, при архівуванні даних, запису інформації на жорсткі диски (ЕСС), запису на система програмно-визначаємих сховищ для NVMe на базі технології RDDA. Навіть якщо ушкоджено значний обсяг інформації, зіпсовано кілька секторів дискового носія, то коди Ріда-Соломона дозволяють відновити більшу частину загубленої інформації. Також використовується при записі на такі носії, як магнітні стрічки й штрихкоди.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходить взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

- процеси які являють собою трансформацію даних в рамках описуваної системи.
- Сховища даних (репозиторії).
- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.

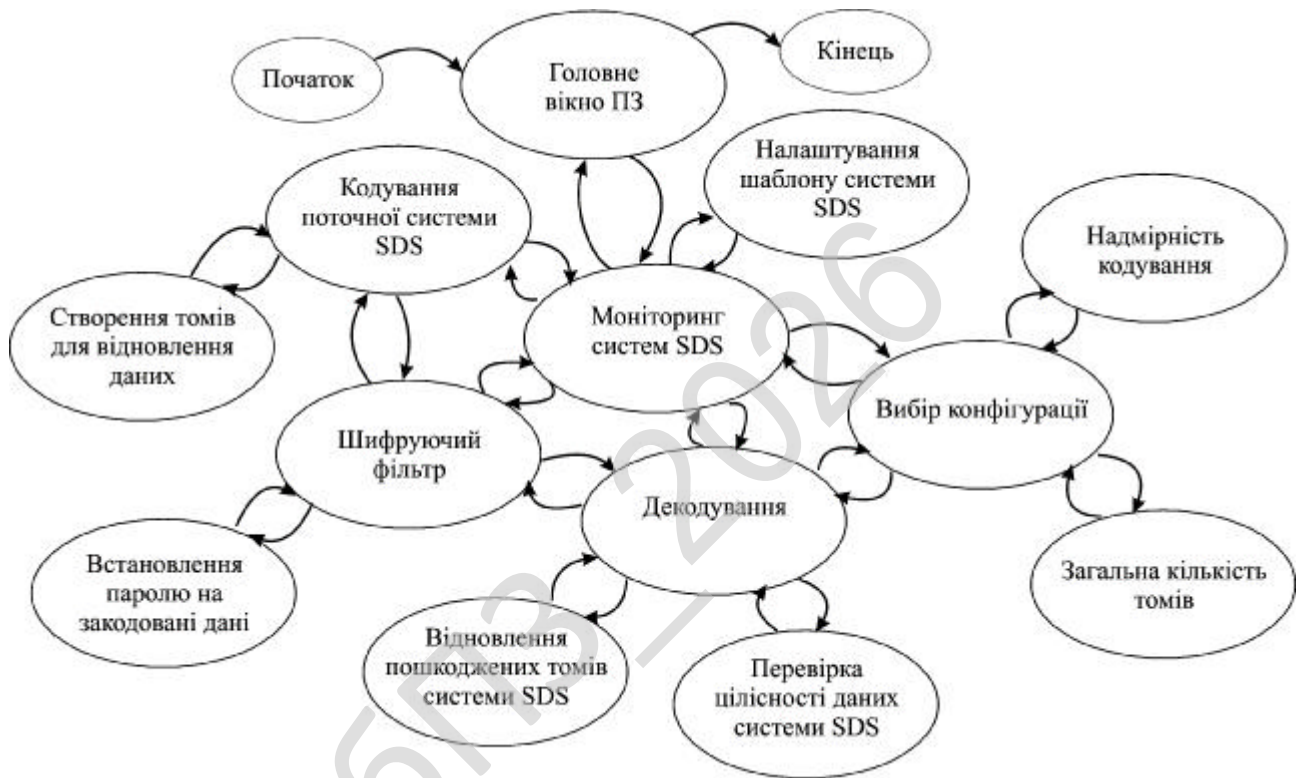


Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

В першу чергу розглянемо архітектурний шаблон який використовувався у роботі – подійно-орієнтована архітектура (Event-driven architecture, надалі EDA) це шаблон архітектури програмного забезпечення, який призначений для створення подій, їх виявлення, споживання і реагування на них.

Подія може бути визначена як значна зміна стану. Наприклад, коли споживач купує автомобіль, стан автомобіля змінюється з "на продаж" до "продано". Архітектура системи дилера автомобілів може трактувати цю зміну стану як подію, поява якої може стати відомою іншим програмам даної архітектури.

З формальної точки зору, те, що виробляється, публікується, поширюється, виявляється і споживається (як правило, асинхронно) є повідомленням, яке називають сповіщенням про подію (або нотифікацією), а не самою подією, яка є зміною стану, що викликає появу повідомлення.

Події не подорожують, вони просто відбуваються. Проте термін подія часто використовується метонімічно для позначення самого нотифікаційного повідомлення, що може призвести до певної плутанини.

Цей архітектурний шаблон може застосовуватися при проектуванні і реалізації ПЗ і систем, які передають події між слабкозв'язаними компонентами програмного забезпечення і сервісами (службами).

Подійно-орієнтована система як правило складається з емітерів подій (або агентів) і споживачів подій (або стоків). Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

забезпечена самим стоком. Наприклад, стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію. Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень (message oriented middleware, MOM), в той час, як друга категорія стоків (самостійна реакція в режимі он-лайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій.

Розробка ПЗ і систем в подійно-орієнтованій архітектурі дозволяє їм бути сконструйованими способом, який більш відповідає вимогам до їх створення, оскільки такі системи в більшій мірі пристосовуються до непередбачуваних і асинхронних середовищ.

Подійно-орієнтована архітектура (EDA) може доповнювати сервісно-орієнтовану архітектуру (SOA), оскільки сервіси (служби) можуть бути активовані тригерами, які ініціюються при настанні подій.

Ця парадигма особливо корисна, коли стік не забезпечує власного виконання будь-яких дій.

Подійно-орієнтована SOA (SOA-2) розвиває архітектури SOA і EDA для забезпечення більш глибокого і надійного рівня сервісів за рахунок використання раніше невідомих причинно-наслідкових зв'язків, щоб сформувати новий шаблон подій. Цей новий шаблон бізнес-аналітики дає поштовх до небаченого раніше зростання рівня автоматизації підприємства за рахунок привнесення додаткової цінної інформації в описану раніше модель діяльності.

Обчислювальна техніка та сенсорні пристрої (сенсори, датчики, контролери) можуть виявляти зміни стану об'єктів або умов і створювати події, які потім можуть бути оброблені сервісом (службою) або системою.

Подія може складатися з двох частин: заголовка події та тіла події. Заголовок події може включати в себе інформацію таку як, наприклад, назва події, часова мітка події і тип події. Тіло події – це частина, яка описує факт, що стався в

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		48

дійсності. Тіло події не слід плутати з шаблоном або логікою, яка може бути застосована як реакція на саму подію.

Архітектура, керована подіями, складається з чотирьох логічних рівнів (шарів). Вона починається з виявлення факту, його технічного подання у формі події і закінчується непустою множиною реакцій на цю подію.

Генератор подій

Першим логічним шаром є генератор подій, який виявляє факт і представляє цей факт подією. Оскільки фактом може бути практично все, що може бути сприйнято, то ним може бути і генератор подій. Наприклад, генератором може бути клієнт електронної пошти, система електронної комерції або певний тип датчика.

Перетворення різних даних, отриманих від датчиків, в єдину стандартизовану форму даних, які можуть бути оцінені, є основною проблемою при розробці та реалізації цього шару. Однак, враховуючи, що подія є строго декларативною, можна легко застосовувати будь-які операції трансформації, тим самим усуваючи необхідність забезпечення високого рівня стандартизації.

Канал подій

Канал подій – це механізм, через який інформація від генератора подій передається до обробника подій (event engine) або стоку.

Це може бути з'єднання TCP/IP або вхідний файл будь-якого типу (простий текст, формат XML, e-mail тощо). В один і той же час може бути відкрито кілька каналів подій. Як правило, оскільки обробник подій повинен працювати в режимі, наближеному до реального часу, канали подій зчитуються асинхронно. Події зберігаються в черзі, очікуючи наступної обробки механізмом обробки подій.

Механізм обробки подій

Механізм обробки подій (event processing engine) є місцем, де подія ідентифікується і вибирається відповідна реакція на нього, яка потім виконується. Це також може призвести до породження ряду тверджень. Якщо подія, яка надійшла до механізму обробки подій, є наприклад такою «Запаси продукту ID

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

досягли нижнього допустимого рівня», це може ініціювати, наприклад, такі реакції як «Замовити продукт ID» і «Сповістити персонал».

Наступна подійно-орієнтована дія (післядія)

Щодо того, як можуть проявлятися наслідки події, слід відмітити, що вони можуть проявитись багатьма різними способами і у різноманітних формах (наприклад, повідомлення електронної пошти, надіслане комусь, або ПЗ, що виводить деяке попередження на екран). Залежно від рівня автоматизації, який забезпечується стоком (механізмом обробки подій), ці дії можуть виявитись зайвими.

Є три основні стилі обробки подій: простий, потоковий і складний. Часто ці три стилі використовуються спільно у розвинутій подійно-орієнтованій архітектурі.

Проста обробка подій

Проста обробка подій стосується подій, які безпосередньо належать до специфічних вимірних змін умов. У випадку простої обробки подій, мають справу з появою відомих подій, що ініціюють післядію (післядії). Проста обробка подій зазвичай використовується для управління потоком робіт в реальному часі, скорочуючи тим самим час затримки і вартість робіт.

Наприклад, прості події можуть створюватись (породжуватись) датчиком, що виявляє зміну тиску в шині або температуру навколишнього середовища.

Обробка потоку подій

При обробці потоку подій (event stream processing, далі ESP) відбуваються як звичайні, так і відомі події. Звичайні події (заявки, передачі RFID) перевіряються на те, чи є вони відомими, і передаються інформаційним передплатникам. Обробка потоку подій зазвичай використовується для управління потоком інформації в реальному часі і на рівні підприємства, що дозволяє своєчасно приймати рішення.

Обробка складних подій

Обробка складних подій (Complex event processing (CEP)) дозволяє за шаблонами простих і звичайних подій проводити аналіз того, чи напустила складна подія. Обробка складних подій полягає в оцінюванні взаємного впливу подій і в наступному виконанні дій. При цьому, типи подій (відомих або звичайних) можуть перетинатись, а події можуть виникати протягом тривалого періоду часу.

Кореляція подій може бути причинною, тимчасовою або просторовою. CEP вимагає використання складних інтерпретаторів подій, визначення і підбору шаблонів подій, а також відповідних кореляційних методів. Обробка складних подій зазвичай використовується для виявлення і реагування на аномальну поведінку, загрози і можливості у бізнесі.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елемента. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елемента. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи програмно-визначаємих сховищ для NVMe на базі технології RDMA.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

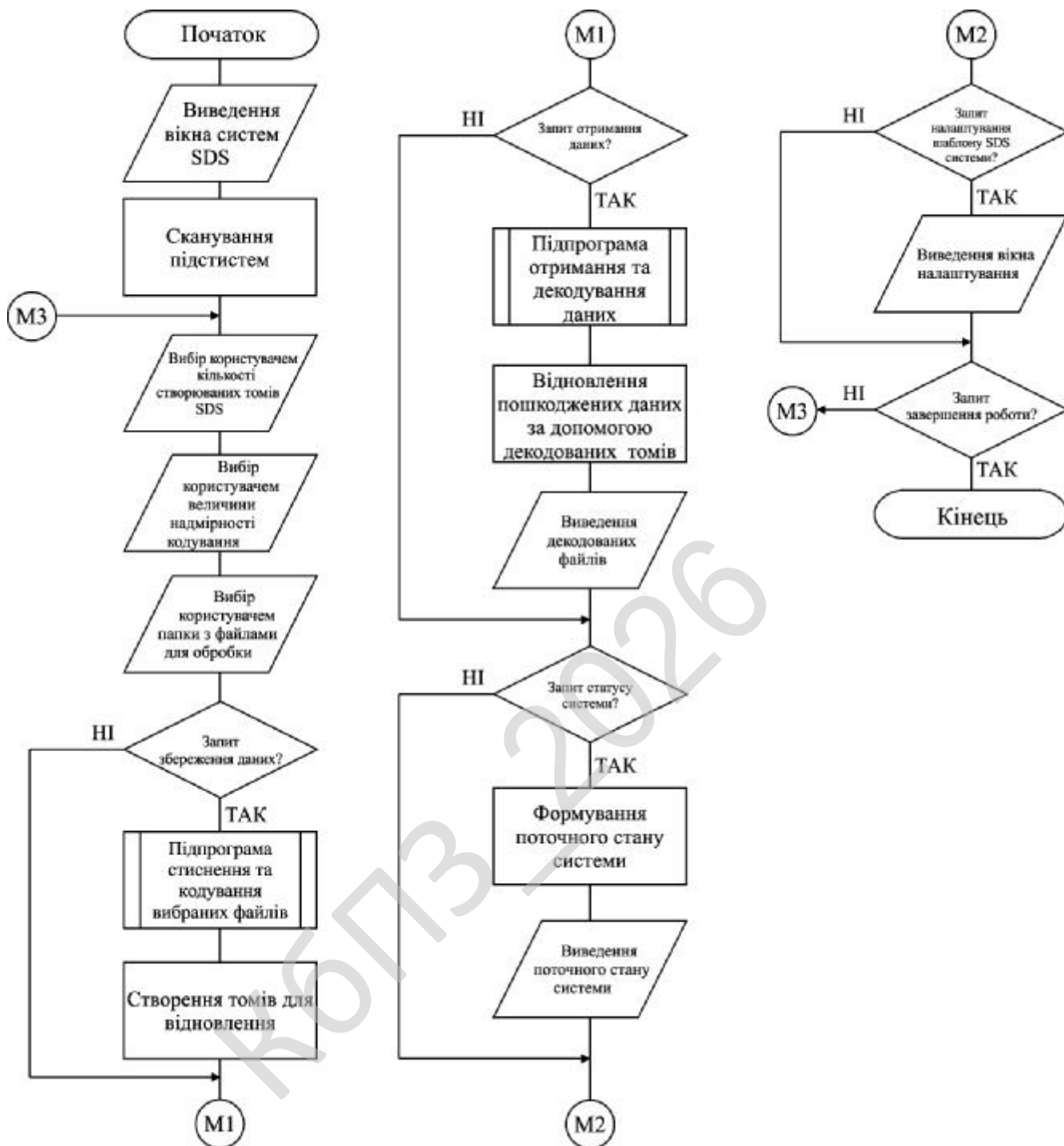


Рисунок 4.1 – Блок-схема основної програми

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

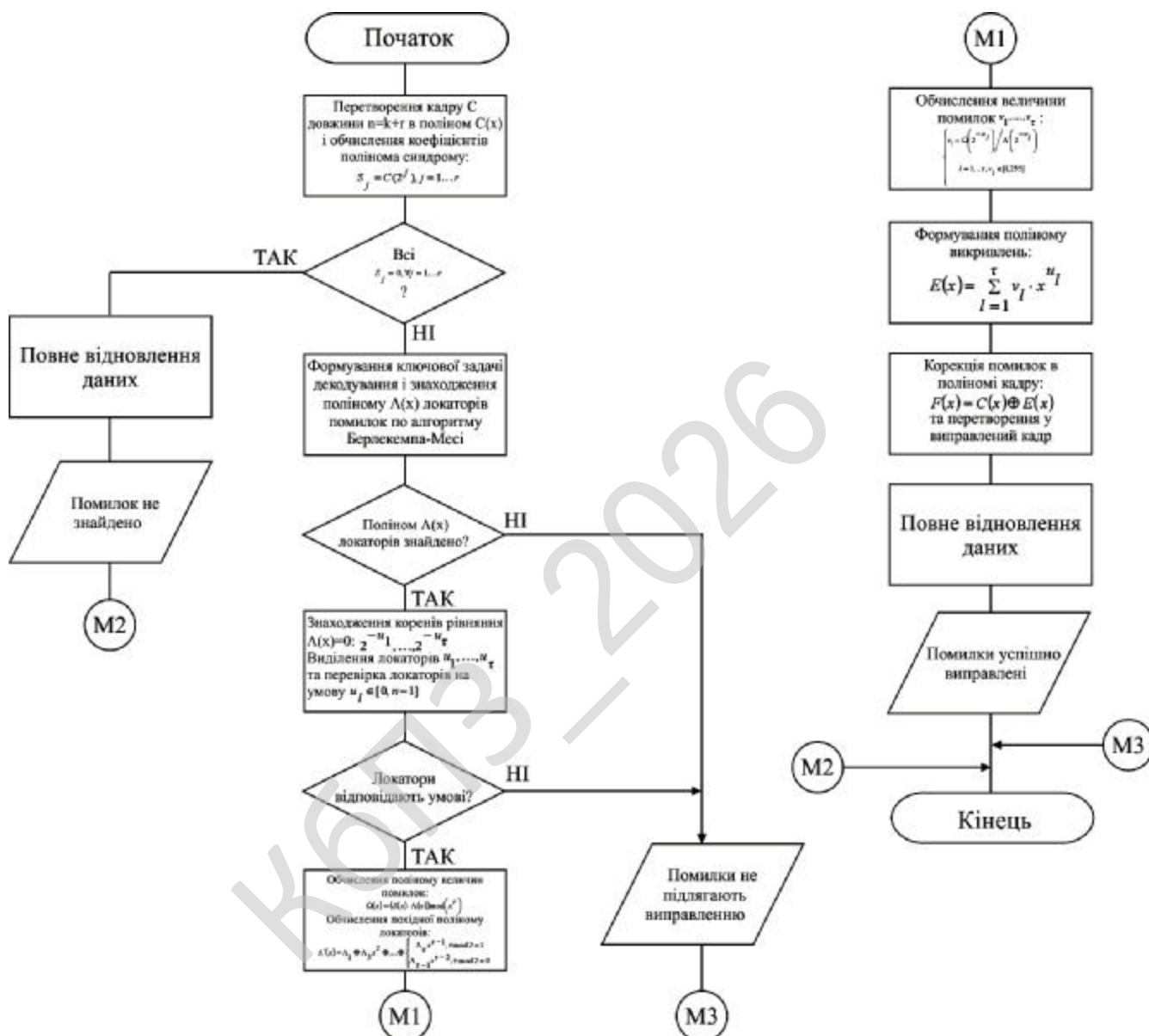


Рисунок 4.2 – Блок-схема роботи підпрограми

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (association relationship);
- включення (include relationship);
- розширення (extend relationship);
- узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посылатися на них і відрізнати один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації

з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою MARS, який є блочно-симетричним шифром з відкритим ключем. Розмір блоку при шифруванні 128 біта, розмір ключа може варіюватися від 128 до 448 біт включно (кратні 32бітам). Творці прагнули поєднати в своєму алгоритмі швидкість кодування і стійкість шифру. В результаті вийшов один з самих криптостійкий алгоритм з алгоритмів, які брали участь в конкурсі AES.

Алгоритм унікальний тим, що використовував практично всі існуючі технології, застосовувані в криптоалгоритмах, а саме:

- Найпростіші операції (додавання, віднімання, виключаюче або).
- Підстановки з використанням таблиці замін.
- Фіксований циклічний зсув.
- Залежний від даних циклічний зсув.
- Множення за модулем 2^{32} .
- Ключове забілювання.

Використання подвійного перемішування представляє складність для криптоаналіз а, що деякі відносять до недоліків алгоритму. У той же час на даний момент не існує будь-яких ефективних атак на алгоритм, хоча деякі ключі можуть генерувати слабкі підключі.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

Структура алгоритму

Автори шифру виходили з наступних припущень:

1. Вибір операцій. MARS був спроектований для використання на найсучасніших комп'ютерах того часу. Для досягнення найкращих захисних характеристик в нього були включені самі «сильні операції» підтримувані в них. Це дозволило добитися більшого відносини securityper-instruction для різних реалізації шифру.

2. Структура шифру. Двадцятирічний досвід роботи в області криптографії підштовхнув творців алгоритму до думки, що кожен раунд шифрування грає свою роль в забезпеченні безпеки шифру. Зокрема, ми можемо бачити, що перший і останній раунди зазвичай сильно відрізняються від проміжних («центральных») раундів алгоритму в плані захисту від криптоаналітичних атак. Таким чином, при створенні MARSa використовувалася змішана структура, де перший і останній раунди шифрування істотно відрізняються від проміжних.

3. Аналіз. Швидше за все, алгоритм з гетерогенною структурою буде краще протистояти криптоаналітичним методам майбутнього, ніж алгоритм, всі раунди якого ідентичні. Розробники алгоритму MARS надали йому сильно гетерогенну структуру – раунди алгоритму дуже різняться між собою.

У шифрі MARS використовувалися такі методи шифрування:

1. Робота з 32-х бітними словами. Всі операції застосовуються до 32-бітовим словами, тобто вся початкова інформація розбивається на блоки по 32біта. (Якщо ж блок опинявся меншої довжини, то він доповнювався до 32біт)

2. Мережа Фейстеля. Творці шифру вважали, що це найкращий варіант поєднання швидкості шифрування і криптостійкості. В MARS використана мережа Фейстеля 3-го типу.

3. Симетричність алгоритму. Для стійкості шифру до різних атакам всі його раунди були зроблені повністю симетричними, тобто друга частина раунду є дзеркальне повторення першої його частини.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

Структуру алгоритму MARS можна описати таким чином:

1. Попереднє накладення ключа: на 32-бітові субблоки A, B, C, D накладаються 4 фрагмента розширеного ключа $k_0 \dots k_3$ операцією складання за модулем 2^{32} .
2. Виконуються 8 раундів прямого перемішування (без участі ключа шифрування).
3. Виконуються 8 раундів прямого криптоперетворення.
4. Виконуються 8 раундів зворотного криптоперетворення. [2]
5. Виконуються 8 раундів зворотного перемішування, також без участі ключа шифрування.
6. Фінальне накладення фрагментів розширеного ключа $k_{36} \dots k_{39}$ операцією віднімання за модулем 2^{32} .

Пряме перемішування

У першій фазі на кожне слово даних накладається слово ключа, а потім відбувається вісім раундів змішування згідно з мережею Фейстеля третього типу спільно з деякими додатковими змішування. У кожному раунді ми використовуємо одне слово даних (зване, вихідним словом) для модифікації трьох інших слів (звані, цільовими словами). Ми розглядаємо чотири байта вихідного слова як індексів на двох S-блоків, S_0 і S_1 , кожен, що складається з 256 32-розрядних слів, а далі проводимо операції XOR або додавання даних відповідного S-блоку в три інших слова.

Якщо чотири байти вихідного слова b_0, b_1, b_2, b_3 (де b_0 є першим байтом, а b_3 є старшим байтом), то ми використовуємо b_0, b_2 , як індекси в блоку S_0 і байти b_1, b_3 , як індекси в S-блоці S_1 . Спочатку зробимо XOR S_0 до першого цільовим речі, а потім додамо S_1 до того ж слова. Ми також додаємо S_0 до другого цільовим слову і XOR блоку- S_1 до третього цільовим слову. У висновку, ми обертаємо вихідне слово на 24 біта вправо.

У наступному раунді ми обертаємо наявні у нас чотири слова: таким чином, нинішні перші цільове слово стає наступним вихідним словом, поточним

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

другий цільове слово стає новим першим цільовим словом, третє цільове слово стає наступний другий цільовим словом, і поточне вихідне слово стає третім цільовим словом.

Більш того, після кожного з чотирьох раундів ми додаємо одне з цільових слів назад у вихідне слово. Зокрема, після першого і п'ятого раундів ми додав третю цільове слово назад у вихідне слово, а після другого і шостого раунду ми додаємо першої цільової слово назад у вихідне слово. Причиною цих додаткових операцій змішування, є ліквідація декількох простих диференціальних криптоатаки в фазі перемішування, щоб порушити симетрію у фазі змішування та отримати швидкий потік.

Криптографічне ядро

Криптографічне ядро MARS – мережа Фейстеля 3-го типу, що містить в собі 16 раундів. У кожному раунді ми використовуємо ключову Е-функцію, яка є комбінацією множень, обертань, а також звернень до S-блоків. Функція приймає на вхід слово даних, а повертає три слова, з якими згодом буде здійснена операція додавання або XOR до інших трьох слів даними. У доповненні вихідне слово обертається на 13 біт вліво.

Для забезпечення, серйозного опору до криптоатаки, три вихідних значення Е-функції (O_1 , O_2 , O_3) використовуються в перших восьми раундах і в останніх восьми раундах в різних порядках. У перші вісім раундів ми додаємо O_1 і O_2 до першого і другого цільовим речі, відповідно, і XOR O_3 у третьому цільовим слову. За останні вісім раундів, ми додаємо O_1 і O_2 до третього і другого цільовим речі, відповідно, і XOR O_3 до першого цільовим слову.

Е-функція

Е-функція приймає як вхідні дані одне слово даних і використовує ще два ключових слова, виробляючи на виході три слова. У цій функції ми використовуємо три тимчасові змінні, що позначаються L, M і R (для лівої, середньої та правої).

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

Спочатку ми встановлюємо в R значення вихідного слова зміщеного на 13 біт вліво, а в M – сума вихідних слів і першого ключового слова. Потім ми використовуємо перші дев'ять бітів M як індекс до однієї з 512S-блоків (яке виходить суміщенням S_0 і S_1 змішуванням фази), і зберігаємо в L значення відповідного S-блоку.

Потім помножимо друге ключове слово на R , зберігши значення в R . Потім обертаємо R на 5 позицій вліво (так, 5 старших бітів стають 5 нижніми бітами R після обертання). Тоді ми робимо XOR R в L , а також переглядаємо п'ять нижніх біт R для визначення величини зсуву (від 0 до 31), і обертаємо M вліво на цю величину. Далі ми обертаємо R ще на 5 позицій вліво і робимо XOR в L . У висновку, ми знову дивимося на 5 молодших бітів R , як на величину обертання і обертаємо L на цю величину вліво. Таким чином результат роботи E-функції – 3 слова (по порядку): L , M , R .

Зворотне перемішування

Зворотне перемішування практично збігається з прямим перемішуванням, за винятком того факту, що дані обробляються в зворотному порядку. Тобто, якби ми поєднали пряме і зворотне перемішування так, щоб їх виходи і входи були б з'єднані в зворотному порядку ($D[0]$ прямого і $D[3]$ зворотного, $D[1]$ прямого і $D[2]$ зворотного), то не побачили б результату перемішування. Як і в прямому змішуванні, тут ми теж використовуємо одне вихідне слово і три цільових. Розглянемо чотири перших байта вихідного слова: b_0, b_1, b_2, b_3 . Будемо використовувати b_0, b_2 як індекс до S-блоку – S_1 , а b_1, b_3 для S_0 . Зробимо XOR $S_1[b_0]$ в перше цільове слово, віднімемо $S_0[b_3]$ з другого слова, віднімемо $S_1[b_2]$ з третього цільового слів і потім проробимо XOR $S_0[b_1]$ також до третього цільового слова. Нарешті, ми повертаємо початкове слово на 24 позицій вліво. Для наступного раунду ми обертаємо наявні слова так, щоб нинішнє перше цільове слово стало наступним вихідним словом, поточне друге цільове слово стало першим цільовим словом, поточне третє цільове слово стало другим цільовим словом, і поточне вихідне слово стало третім цільовим словом. Крім

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

того, перед одним з чотирьох «особливих» раундів ми віднімаємо одне з цільових слів з вихідного слова: перед четвертим і восьмим раундами ми віднімаємо першої цільової слово, перед третьому і сьомим раундами ми віднімаємо третя цільова слово з вихідного.

Дешифрування

Процес декодування обернений процесу кодування. Код дешифрування схожий (але не ідентичний) на код шифрування.

К6ПЗ_2026

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		64

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи програмно-визначаємих сховищ для NVMe на базі технології RDDA яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

- Навігаційне меню: Файл; Інструменти; параметри; Довідка.
- Розділу обрання групи файлів дисків системи.
- Навігаційного меню яке викликається натисканням правої клавіші манипулятора миші.
- Функціональних кнопок ПЗ.

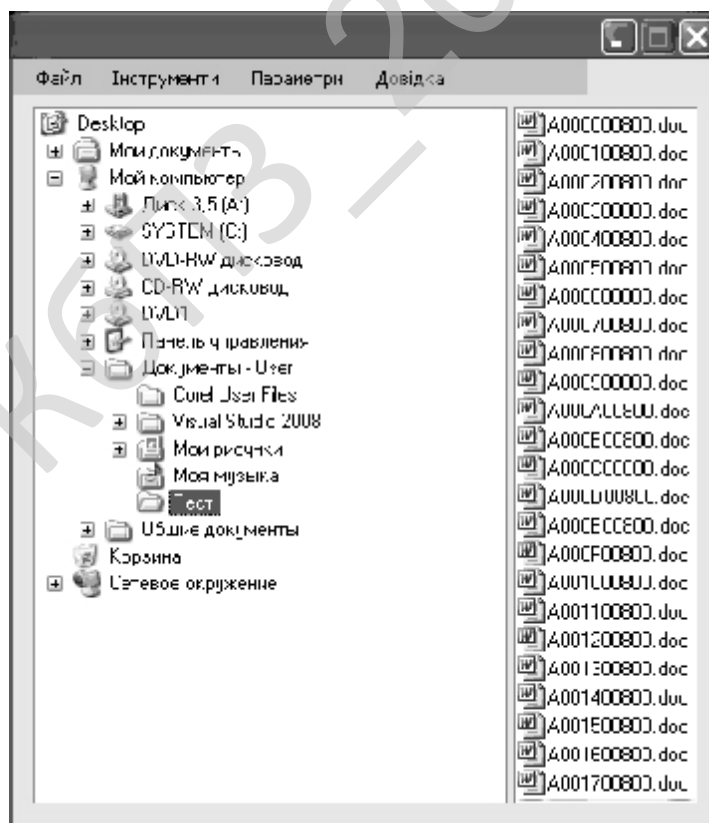


Рисунок 5.1 – Головне вікно ПЗ

Тенденція абстрагування ПЗ від нижчележачого устаткування дає шанс українським розроблювачам запропонувати передові рішення й вийти на перспективний ринок, не маючи ресурси західних компаній. І вони намагаються цей шанс використовувати: цілий ряд вітчизняних компаній займаються розробками в області програмно визначаємих мереж і систем зберігання.

П'ять-десять років тому все остерігалися використовувати програмні СЗД, але зараз вони не уступають апаратним платформам ні по функціональності, ні по надійності. Як і інше програмні СЗД, рішення може бути встановлене на будь-яке устаткування x86, а компоненти можна використовувати стандартні, доступні на ринку: сервери x86 і диски оригінальних виробників.

Оскільки замовники усе ще консервативні й поки не готові відмовлятися від переваг, які дає покупка готового рішення в порівнянні із самостійною установкою програмного забезпечення на сервери, продукт поставляється й у вигляді апаратного комплексу. При виникненні яких-небудь збоїв замовники не хочуть з'ясовувати, хто винуватий – постачальник ПЗ або устаткування, а купуючи апаратно-програмний комплекс, вони одержують протестоване рішення, за надійну роботу якого повністю відповідає постачальник.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему

готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

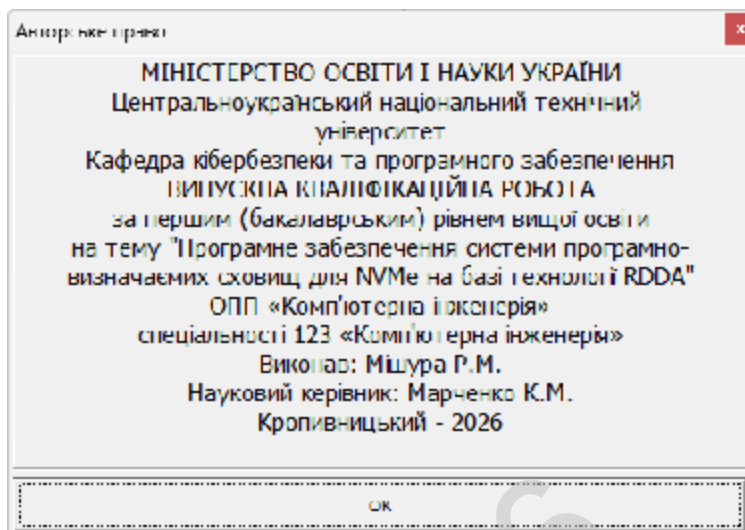


Рисунок 5.2 – Авторське право

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.
- Адаптація.
- Обновлення.
- Вмонтування.

- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

- Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження.

- Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

- Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

- відповідності поставленим вимогам;
- правильна відповідь для усіх можливих вхідних даних;

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		68

- виконання функцій за прийнятний час;
- практичність;
- сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

- Як виконуються функції програми.
- Як приймаються вихідні дані.
- Як виробляються результати.
- Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10} . Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		69

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

- Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;
- Сформулювати такі очікувані результати, які з високою ймовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

- Некоректних чи відсутніх функцій.
- помилок інтерфейсу.
- помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.
- помилок характеристик (необхідна ємність пам'яті і т.д.).
- помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільноті як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		70

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

К6ПЗ_2026

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		71

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем програмно-визначаємих сховищ для NVMe на базі технології RDDA.
- Досліджена система програмно-визначаємих сховищ для NVMe на базі технології RDDA.
- На основі отриманих результатів досліджень створена програмна реалізація системи програмно-визначаємих сховищ для NVMe на базі технології RDDA.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання програмно-визначаємих сховищ для NVMe на базі технології RDDA.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C#.. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи програмно-визначаємих сховищ для NVMe на базі технології RDDA. Це

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		72

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MARS.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		73

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.
2. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.
3. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447
4. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.
5. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.
6. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.
7. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.
8. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises».

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		74

9. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи*, 2023, том 7, № 2, С. 49-56.

10. Smirnov, O., Neskrodieva, T., Fedorov, E., Rudakov, K., Neskrodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

11. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sherroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022.

12. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science*, Vol 2, 337, 2021. <https://doi.org/10.1007/s42979-021-00739-w>

13. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

14. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

15. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

16. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

17. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

18. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

19. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

20. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

21. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

22. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

23. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

24. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

25. Zhurakovskiy, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

26. Smirnov O. Kuznetsov A., Zaichenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

28. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

30. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and

Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

31. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

32. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

33. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

34. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

35. Т.В. Смірнова, О.М. Дреєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макروفотografій». IV міжнародна науково-практична конференція «Інформаційна безпека та комп'ютерні технології», м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

36. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

37. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		78

38. Смірнов О.А., Дреєва Г.М., Дреєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральнотраїнський науковий вїсник. Технїчні науки. № 2(33). с. 161-172, 2019.

39. О. Смірнов, Є. Деменко, О. Онїкійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послїдовностей для приховування даних в зображеннях» Комп'ютерні науки та кїбербезпека. № 4. С. 30-37. 2019.

40. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Полїщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посїбник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

41. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографїя: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

42. Смірнов О.А., Дреєва Г.М. Метод генерування фрактального трафіку за допомогою моделї генератора на графї. Монографїя: Інформаційна безпека та інформаційні технологїї : монографїя / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

43. Дреєва Г.М., Смірнов О.А., Дреєв О.М. Метод генерування фрактальноподїбної числової послїдовностї на основї скїнченного автомату для моделювання трафіку у мережї. Центральнотраїнський науковий вїсник. Технїчні науки. № 1(32). с. 173-183, 2019.

44. Смірнов О.А., Кавун С.В., Коваленко О.В., Дреєв О.М. Мережні інформаційні технологїї. Навчальний посїбник – Кїровоград: РВЛ КНТУ, 2016. – 159 с.

45. Смірнов О.А., Смірнов С.А. Дїдик А.К., Дреєв О.М. Моделї системи нейромережевих експертів безпечної маршрутизацїї у хмарних

антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

46. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дреєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

47. Смірнов О.А., Дреєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". - Випуск 2 (118). т.2. - Х.: ХУПС - 2014. - С. 64-67

48. Смірнов О.А., Дреєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції “Проблеми та перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. - 2014. - С. 240.

49. Смірнов О.А., Коваленко О.В., Кожанова А.С., Лєвошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

50. Смірнов О.А., Дреєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

51. Смірнов О.А., Доренський О.П., Дреєв О.М. Аналіз процесів стиснення та відновлення зображень на основі цифрових методів. Наука і техніка Повітряних сил Збройних Сил України. – Випуск 3(12). – Х.: ХУПС. – 2013. – С.122-127.

52. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

53. Смірнов О.А., Євсєєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. – Кіровоград: КНТУ 2012. – 454 с

					ВКРБ-123.26.0052.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		80